Microcode Monitor for Basic 2.4

1.0 Introduction

The purpose of the Microcode Monitor, hereinafter referred to as DEBUG, is to allow the Assembler programmer a means of inspecting, tracing and 'debugging' machine code programs. The second benefit derived from the program is the capability to trace the flow of a program, thereby allowing us to understand the processes required for the execution of atoms.

This program was the first machine code program written by the author for the Wang 2200 MVP system, and the style of programming will reflect this statement. However, the DEBUG function has proved itself as being invaluable during code breaking sessions.

DEBUG resides from locations 5000 to 5700 in Control Memory, and uses some storage from 5AA0 to 5B50 for buffer space. On entry, all registers are preserved to ensure integrity upon return from the DEBUG function.

DEBUG was not designed to run in a true Multi-user enviroment. Because interception of commands may at times cause 'abnormal' termination of internal Wang processes, we strongly recommend that DEBUG not be executed when any other jobs are running. The complexity of DEBUG does not allow the luxury of multiple users entering the monitor area. The queue for users is non-existant, and may be said to be a 'whoever is calling gets me' program.

Again, I wish to stress that the DEBUG function is NOT to be used on an active, working enviroment!

To prevent accidental entry to the DEBUG function, a built in lock is performed. This lock must be unlocked prior to the entry into DEBUG.

The normal means of entry to DEBUG is by executing the DEBUG command without arguments. If someone on the system accidently typed this command, and the system was in the locked state, they simply get a S error back. If unlocked, the command DEBUG vectors to the DEBUG monitor.

The unlocking of the DEBUG monitor is performed by execution of the following command:

```
DEBUG(4,VAL(HEX(5C03),2))=HEX(DC0050)
```

This removes the S error trap, and causes the enabling for the DEBUG monitor.

2.0 Initial Entry to DEBUG

As stated before, the initial entry to the DEBUG monitor is performed by 'unlocking' the function, then executing a DEBUG statement without parenthesis. On entry, DEBUG clears the screen and dumps the current set of registers for view by the user. The below screen is an example with the Breakpoints enabled:

```
048D = Call Stack
048C = Trapped Breakpoint Address

          Registers

R0 R1 R2 R3 R4 R5 R6 R7 SL SH K  -- CL CH
85 39 35 22 00 00 85 04 40 12 35 00 04 B1

Auxiliary Registers

0000 54CB 0000 2BAC 5503 048C 0001 4400
0000 0000 2B07 2AFE FFD2 220C 2BAC 0000
0485 2BAD 2BAA FFDC 7C4E 2350 FFDC 0022
0000 FFCF 0007 6000 0000 5000 2BA9 0007

B104 = PHPL
```

The above screen is the state of the machine at the trapped breakpoint address. All registers displayed are saved, so a return will restore all registers to their initial state, allowing resumption of execution. Note that the auxiliary registers start at AR 00, and end with AR 1F.

If no Breakpoint was in effect, which will occur during the initial call to DEBUG, the Trapped Breakpoint message will not appear on the screen.

After this display, the prompt for entry of a command is displayed on the screen as follows:

```
>
```

The user may now enter a single digit command. Invalid commands return a '?' character, and the '>' symbol is redisplayed. Note that partition slicing will still occur, and programs not requiring IO will still function correctly in other partitions.

3.0 Help Command

When in the hardware DEBUG mode, the execution of the 'H' command will produce a Help list of available commands.

```
> H

Command    Function

A          Dump Block of Data Memory  E.G.  A 0200
B          Breakpoint Functions
C          Change Control Memory   C 3000
D          Change Data Memory      D 0040
F          Find Occurance of Data  F 0000 3400 87800F
J          Jump to location without Restoration  J 1200
K          Kill all Checksums in Memory
M          Set Search Mask
P          Print Stored register data
R          Restore registers and Return    R 018B
S          Set SL register (Bank Selection)  S 80
V          Print version of Monitor
X          Return to Breakpointed Program
```

The monitor will then return once again to the ')' prompt, awaiting your input.

4.0 A command - Dump Block of Data Memory

This command allows the user to display 256 bytes of Data Memory at the location specified by the user, and at the current Bank selected.

```
> A 0900

40
0900 00 02 20 00 00 00 00 00 00 00 00 03 02 00 30 00 .. ...........0.
0910 02 01 30 00 00 00 00 00 00 00 00 00 00 00 00 00 ..0.............
0920 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0930 05 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 ................
0940 00 04 02 40 02 02 00 00 80 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 02 01 03 40 41 38 38 28 28 28 28 28 28 28 28 ....@A88((((((((
0980 08 08 08 08 08 08 08 08 00 00 00 00 00 00 00 00 ................
0990 0C 86 20 90 20 90 20 90 00 00 00 00 00 00 00 00 .. . . .........
09A0 C0 FF 0C 00 00 00 03 01 0D 12 2B 10 00 42 FF FF ..........+..B..
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 8E 00 00 00 00 00 00 00 00 ................
09E0 00 00 40 40 80 80 C0 C0 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 first byte printed is the current Bank selection, followed by the addresses and 256 bytes of data.

5.0 B command - Breakpoint functions

Breakpoints are an important function of the DEBUG monitor. When entered, the following menu is displayed:

```
> B

Breakpoint Command Menu

 .     Display Current Breakpoints
(CR)   Return to Monitor
 D     Delete Breakpoint    (D 0200)
 X     Return from Current Breakpoint
 C     Clear all current Breakpoints
 S     Set new breakpoint   (S 1254)

BR >
```

The prompt, 'BR>', will alert the user that he is in the sub-menu for Breakpoints. Illegal commands in this mode will revert the user back to the normal Monitor. The clean exit out of this sub-mode is to simply press a RETURN key. The normal '>' prompt will then be displayed.

Breakpoints are defined as being an interception of the normal flow of a program, thus 'Breaking' the logical conclusion of a series of statements. Our breakpoints do just that.

5.1 . Command - Display current breakpoints

When the . key is depressed, all active breakpoints are displayed. If a current Breakpoint has caused entry to the DEBUG monitor, that breakpoint is highlighted to the user. Assume that we had set a breakpoint at location 12C0 in memory. When that point was executed, a call to DEBUG was executed. Assume that we also had breakpoints at other locations. If we depress the . key, the following display would result:

```
BR > .
12C0 = Trapped Breakpoint Address
1040 584E0F
12C0 560F10
0466 01810F

BR >
```

Note that our examples show that three breakpoints are currently set, at addresses 1040,12C0 and 0466. However, the current entry to the DEBUG monitor was through the breakpoint at 12C0.

5.2 D command - Delete Breakpoint

Once a breakpoint has been set, we may remove the breakpoint by simply executing the sub-command D.

BR> D 0466   Breakpoint Removed

After the D command has been typed, enter the 4 digit value of the breakpoint that is to be deleted. The above example shows the successfull deletion of the breakpoint for address 0466.

If we attempt to delete a non-existing breakpoint, we get the following message:

```
BR> D 0467
     ( No such Address is in Breakpoint Stack )

BR>
```

The internal function of the D command causes the requested breakpoint to be removed from the Breakpoint stack, and the contents of that address be restored to the original value.

5.3 X command - Return from Breakpoint

In either the sub-mode for Breakpoint commands, or the main DEBUG monitor, if this command is executed, the current Breakpoint register is examined, and if containing a valid breakpoint address, execution is continued from that location. The instruction located at the breakpoint address is EXECUTED, and program flow commences.

If no current stored breakpoint exists, the following message is displayed:

---- No Trapped Breakpoint Address to Return To! -----

Note the power of this command! We can actually interrupt the flow of a program, view the interim results, and continue the execution of the program!

5.4 C Command - Clear all current Breakpoints

Execution of the C command in the BR submode will result in ALL breakpoints being removed, and all Breakpoint addresses restored to their original values. Any Trapped Breakpoint address will be removed as well.

5.5 S Command - Enter breakpoints

Finally, to set the breakpoints into the program, the S command is used during the Breakpoint sub-modes. Though not very practical, up to 256 seperate breakpoints may be entered. The S command removes the contents of the selected address, and replaces it with the special breakpoint control word. The removed contents are then stored away in the Breakpoint stack.

WARNING!

The S command does not check for duplicates in the Breakpoint stack. Therefore, if not sure if you set an address, use the '.' command to display current breakpoints. Failure to do so will result in the possible blowup of your program.

```
BR > S 0400
```

The above command set the breakpoint at location 0400. During course of execution, whenever that address is executed, a breakpoint will occur, and control is transferred to the DEBUG monitor.

Because the S command alters Control Memory, any execution of an S command causes the Control Memory checksum function to be disabled.

6.0 C command - Change Control Memory

During the normal DEBUG monitor prompts, >, the C command allows us to individually inspect and or change any location in Control memory.

```
> C 12C0 560F10
```

If we wish to change the location, type 6 hex digits. To view the next location, press RETURN. To exit back to DEBUG monitor, press the space bar.

Please note that if you change Control Memory, you must ensure that the correct parity bit is set, else you will abort with a PECM when that instruction is executed. If you change any checksummed location, < (5000), make sure that the checksum check function is disabled. (See K command)

7.0 D command - Modify/Examine Data Memory

Similar to the C command, the D command allows us to modify or just view Data memory. All references above 2000 take place in the selected bank, while references below 2000 are always from Bank 00.

```
> D 0311 44
```

To examine the next location, press the RETURN key. To change a location, enter the two hex digits to store. To return to the DEBUG monitor, press the space bar.

Special note: Due to the context switching in DEBUG, locations 0900 and 0901 will not reveal their true selves to the user. However, using the A command will allow us to view, but never change, the actual values at these locations.

8.0 F command - Find occurance of Data

A very powerful command. The F function permits us to search Control Memory between set limits for the occurance of data patterns we specify. Each location is examined, masked by the previously set mask (See M command), and compared to our data. If a match occurs, the address of the location is displayed.

```
> F 0000 5000 87800F
```

The above example tells the monitor to search all locations from 0000 to location 5000 for the occurance of 87800F (RTS instruction). If we set the mask to 7FFFFF, only those exactly matching will be displayed.

But their are many RTS instructions between these limits that do not match 87800F because of RD or W1,W2 functions. By setting the Mask to 7FC9F0, all RTS instructions will be displayed!

9.0 J command - Jump to location without restoration

The J command allows us to continue execution at any specified address. However, the original values of the machine, saved upon entry to DEBUG, are NOT restored.

This command may become useful when writing small hand coded machine programs, not associated with Basic.

10.0 K Command - Kill checksums

As previously stated in prior documents, Wang maintains two checksums in Basic. One is in Control Memory, while the other is in Data Memory. When we 'play' with code, we obviously are going to alter locations.

This result of this would be VECM errors, because we can not readily change checksums every time we alter one piece of data. Therefore, the K command allows us to permanently disable the checking of checksums.

In Data Memory, Locations 0000 and 0001 are zeroed. In Control Memory, location 000F is set to 800000. This effectively inhibits the checking of checksums, but does not disable parity checking in Control Memory.

Restoration of checksums can only be done by reloading the program. (Rebooting)

The S command, Breakpoint Sub-mode, kills the checking of Control Memory checksums only.

11.0 M Command - Set Search mask

The M command allows us to alter the Search mask for the F instruction. When first executed, the current mask is displayed. This may be retained by executing the RETURN key. If we wish to alter the mask, simply type 6 hex digits, corresponding to the mask desired.

```
> M 7FFFFF 7C0000
```

The above example sets the mask to 7C0000. In effect, we will search only for the significant bits of the instruction type.

Example: F 0000 5000 DC677C

```
1: DC677C  is what we are searching for, but
   7C0000 is the mask, so after the AND operation
   ------
   5C0000  is what we are actually going to compare!
```

Each location read is masked by our mask, and then compared. Therefore, our particular example will find all JMP instructions!

12.0 P command - Dump stored registers

A total dump of all 'saved' registers will occur in the same format as described in section 2.0. This allows us to refresh our memory, or examine in more detail the effects of our experimenting.

13.0 R Command - Restore all Registers and return

When executed, all registers previously saved are restored, and a Jump command to the location specified is performed.

```
> R 018B
```

Causes all registers to be restored, and execution commences at location 018B

14.0 S Command - Set Bank

The currently selected Bank of Data memory is set by this command. Initially, the bank selected is the one that we were in when the breakpoint occurred. Therefore, all Data commands, (D and A), will reference that bank. If we wish to alter the bank selection, we execute the S command. Valid S commands are:

```
>S 00      Set Bank 0
>S 40               1
>S 80               2
>S C0               3
>S 20               4     (512k machines only)
>S 60               5
>S A0               6
>S E0               7
```

15.0 V Command - Print version

The current version of the DEBUG monitor is displayed for information purposes:

```
>V Version 2.4
```

16.0 X command - Restore all registers and Return to Breakpointed Program

See section 5.3

17.0 Miscellaneous Notes

Abnormal exits from the DEBUG monitor may be performed by execution of the RESET key. This will abort DEBUG, and the normal flow of the Basic program will resume.

The stored registers may be changed by the user. These changed registers will then be restored on a R or X command. Changing the registers allows us to modify results to see what happened.

The locations for the stored registers are as follows:

```
5AFD  00 --  K
5AFE  00 CH CL
5AFF  00 PH PL

5B00  00 R1 R0
5B01  00 R3 R2
5B02  00 R5 R4
5B03  00 R7 R6

5B04  00 SH SL

5B05  00 AR00         5B15  00  AR10
5B06  00 AR01         5B16  00  AR11
5B07  00 AR02         5B17  00  AR12
5B08  00 AR03         5B18  00  AR13
5B09  00 AR04         5B19  00  AR14
5B0A  00 AR05         5B1A  00  AR15
5B0B  00 AR06         5B1B  00  AR16
5B0C  00 AR07         5B1C  00  AR17
5B0D  00 AR08         5B1D  00  AR18
5B0E  00 AR09         5B1E  00  AR19
5B0F  00 AR0A         5B1F  00  AR1A
5B10  00 AR0B         5B20  00  AR1B
5B11  00 AR0C         5B21  00  AR1C
5B12  00 AR0D         5B22  00  AR1D
5B13  00 AR0E         5B23  00  AR1E
5B14  00 AR0F         5B24  00  AR1F
```