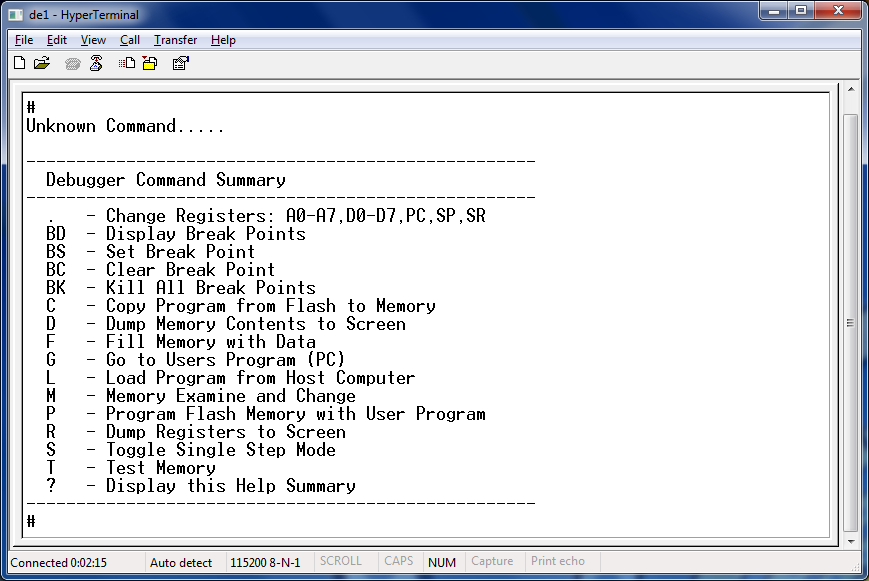
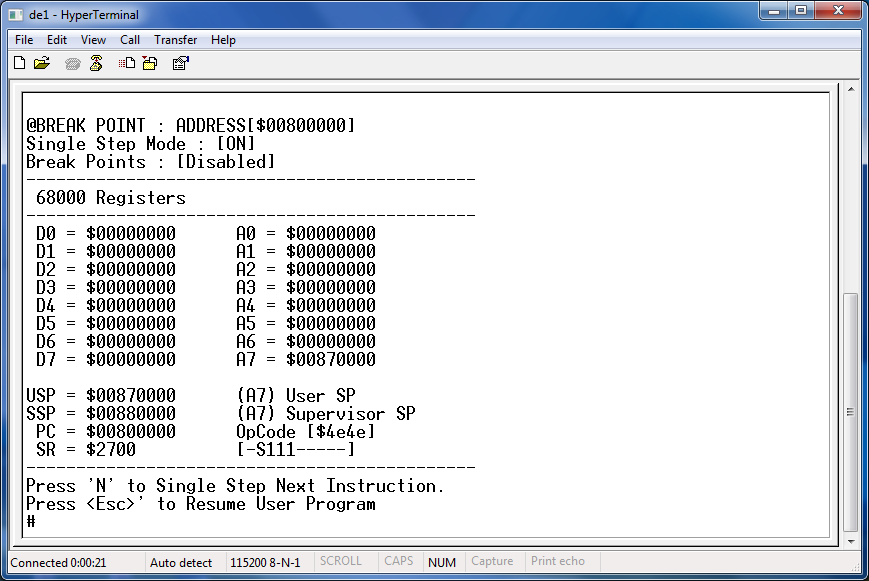
**Using the 68000 Soft Core Processor on DE1 Board**

The debug monitor for the soft core 68000 processor that runs on the DE1 board is quite easy to use. The Debugger has the following commands – as shown below



**Debugging Programs**

Break points can be set in memory at the address of the op-code for an instruction. When the program is run, and it encounters the break point, the program will stop and produce a dump of the 68000 registers as shown below for a breakpoint set at address hex 00800000 (32 bit address)



From here, you can press ‘**ESC’** to resume the program, or press ‘**N’** to single step it one instruction at a time which produces a new register dump showing the state of the internal 68000 registers after each instruction. Note the value in the Program Counter (PC) is the address of the NEXT instruction to be executed.

Running Programs can be interrupted at any time by pressing **Key 3** (the left most blue button the DE1 board). This freezes the program at that point and you get a register dump, again you can resume or single step from here.

You can also turn the debugger into single step mode by issuing the command ‘**S**’ followed by ‘**G**’ to run the program one instruction at a time.

**Setting up the 68000 registers prior to running a program**

When a program is run, it is assumed to run starting at the address held by the program counter register (PC) maintained by the debugger. The debugger command ‘**R**’ will display the registers and they can all be changed by issuing a command like ‘**.A0**’, ‘**.D0**’, ‘**.PC**’ etc.

Obviously it’s bad practice to rely on these values and you should learn to initialise all registers within your own program when it starts, rather than “inherit” the values set up in the debugger, but it’s useful when you are first learning assembly language.

However, by default the PC is set to the value hex 00800000 as this points to an area of RAM memory where you can download your programs to. If you are writing in assembly language, you should put an assembler directive **ORG $00800000** into your program right at the start of your program, so that code is produced to download to this address. Once it is downloaded, you can simply issue the command ‘G’ to the debugger to start the program at the address hex 00800000  
  
If you are writing in C code, this won’t be a problem, since the ‘**CStart\_V4.0\_UserProgram.asm**’ file that you should have been given will do this for you, i.e. it contains the ORG statement above and initialises your 68000 registers for you (at least the ones that matter like the user stack pointer (A7) and the status/condition code register).

**Memory Maps**

The soft core 68000 on the DE1 board has a 32 bit address bus with a range that spans the addresses   
**[0000 0000] – [FFFF FFFF]**.

The **ROM** that is programmed with the debug monitor firmware is located in the bottom 16k area of a 64k block of memory than spans the address range **[0000 0000] – [0000 FFFF].** This is a requirement of the 68000 (rom starting at location 0). This gives extra space for expansion in the future

There is a small 8Kbyte block of **Ram** located at the base of the address space [0001 0000] – [0001 FFFF]

The **Flash Memory** is located within at the base of an address block **[0100 0000] – [01FF FFFF]**. This gives extra space for expansion in the future

The **512k byte Static Ram** is used for a VGA display and lies at the base of an address block located within **[F000 0000] – [F0FF FFFF].** This gives extra space for expansion in the future.

The **8Mb Dram** is located within the address range **[0080 0000] – [00FF FFFF]**

There are several 8 bit IO Ports located at address listed below. Some are bidirectional so “writing” to them outputs to the outside world e.g. leds, “reading” inputs from the outside world e.g. switches

**Port Name Connected to Address** A 8 Red LEDS , Switches [0-7] 0040 0000  
 B 8 Green LEDS, Switches[8-9] 0040 0002  
 C 2 Red LEDS, FREE for use 0040 0004  
 D FREE for use 0040 0006  
 E FREE for use 0040 0008

HEX A Right most pair of 7 segment displays 0040 0010  
HEX B Left most pair of 7 segment displays 0040 0012  
HEX C FREE for use 0040 0014  
HEX D FREE for use 0040 0016

LCD Command Reg LCD display Command Register 0040 0020  
LCD Data Reg LCD display Data Register 0040 0022

Timer 1 Data Reg Timer 1 0040 0030  
Timer 1 Control/Status Timer 1 0040 0032

Timer 2 Data Reg Timer 2 0040 0034  
Timer 2 Control/Status Timer 2 0040 0036

Timer 3 Data Reg Timer 3 0040 0038  
Timer 3 Control/Status Timer 3 0040 003A

Timer 4 Data Reg Timer 4 0040 003C  
Timer 4 Control/Stat Timer 4 0040 003E

RS232 Control Reg Serial Port (MC6850) Write only 0040 0040  
 RS232 Status Reg Serial Port (MC6850) Read only 0040 0040  
 RS232 Transmit Data Reg Serial Port (MC6850) Write only 0040 0042  
 RS232 Receive Data Reg Serial Port (MC 6850) Read only 0040 0042

RS232 Baud Rate Reg Write Only 0040 0044

**Writing Your Own Programs**

You should write (using the appropriate **ORG** statement if writing in assembly language) and download your program to Memory starting at address **[0080 0000]**. When you type ‘**G**’ on the debugger or boot from Flash (see next section), the system will attempt to run a program from this address, if your program is not there it won’t work.

Programs can be as large as **192k** (and arbitrary limit that could easily be changed) between **[0080 0000]** and **[0082 FFFF].** When you copy your program to flash, only this 256k area of program is copied. Likewise when you load from Flash. Finally, single stepping/tracing has specifically been designed to work for programs that run within the range **[0080 0000]** and **[0082 FFFF].**

Programs can use the 64k memory block in the range [0083 0000] to [0083 FFFF] for variables. (This is taken care of when writing in C via the CStart.asm file you add to the project

Memory in the range **[0084 0000]** **– [0084 FFFF]** is reserved for use by the debug monitor for a Ram based exception table and other variables (having it in ram allows user program to install their own exception handler e.g. interrupt handler. Do not modify this area in your program.

Memory below **[0088 0000]** is used by the debug monitor for stack space. Your programs should initialise their stack pointer to the address **[0087 0000]** (remember stacks grow downwards). This is taken care of when writing in C via the CStart.asm file you add to the project