



APPLICATION NOTE 2777

Using Keil's MON390 Program with the TINIm400

Abstract: C programming with the DS80C400 microcontroller requires the need for debugging. One means of debugging is to use Keil's MON390, a debugging program that runs code on the target platform. In order to use the monitor, several configuration options have to be considered. This application note outlines those considerations and debugs a simple network program.

Introduction

When the DS80C400 Silicon Software was designed, it was decided that a suite of functionality should be exposed that could be accessed by C programmers. With the aid of a few libraries, C programmers can access the network stack, process manager, memory manager, and other useful functions burned into the DS80C400's ROM. Naturally, the introduction of C programming also means the introduction of C programs in need of debugging. One means of debugging is to use Keil's MON390, a debugging program that runs code on the target platform. To use the monitor, several configuration options have to be considered. This application note outlines those considerations and debugs a simple network program.

What You Need for this Demonstration

This application note was written for use with a TINIm400 board and TINIs400 socket, available on our website at [Maxim Sales and Distribution](#). Two serial cables are also needed. One serial cable is attached to serial 0 on the TINIm400 board, and is used to read standard output from your programs. The other serial cable should be attached to the TINIm400's serial 2 connector (located at J13 on the TINIs400 socket), and needs to be attached through a gender changer and a null modem adapter. This port will be used for communication with the monitor PC-side program. The image below shows a socket and board connected to two serial cables (one with a null modem adapter), an Ethernet cable, and power.

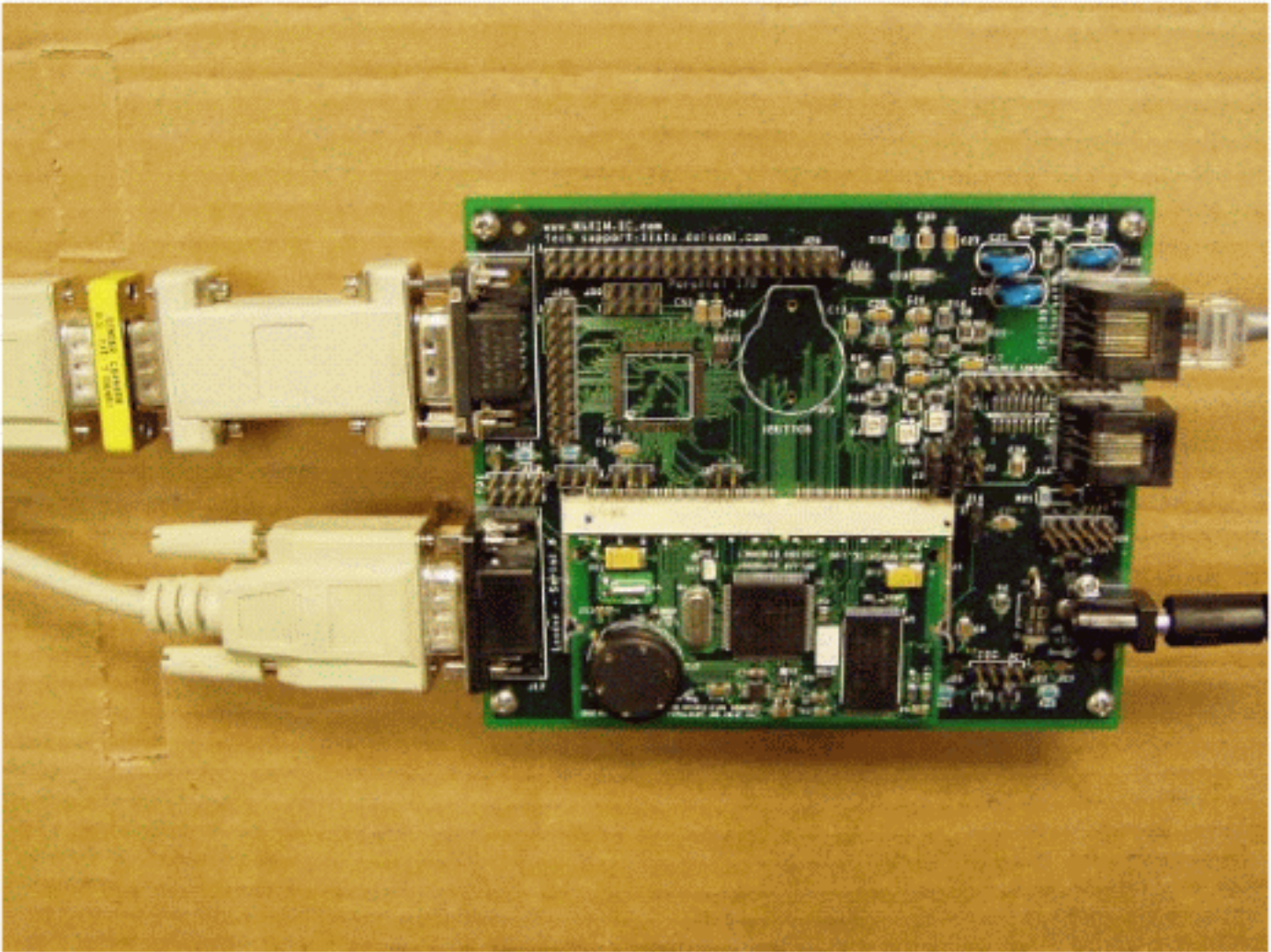


Figure 1. Connectors needed for working with the monitor.

Please also make sure you have the latest version of the Keil μ Vision2™ IDE downloaded (available at www.keil.com). At the time this application note was written, the latest μ Vision2 IDE was version 2.37, and the latest C Compiler was 7.07.

For an introduction to using the Keil tools with the DS80C400, you may want to read application note 613, [Using the Keil C Compiler for the DS80C400](#). It gives a more thorough explanation on getting up and running, and should be read before this application note.

Building and Loading the Monitor Onto the TINIm400 Board

The file **mon400.hex** must be built from the source available from Keil. With the compiler update version 7.07 (and later), a project is included that contains the monitor source. It is located in [Keil Install]\C51\MON390\TINIm400. Open the project **mon400.Uv2** and press F7 to build it. The file **mon400.hex** should appear in the same directory. The monitor is configured to load into bank 47h (into the flash), and uses the TINIm400's serial2 port to communicate with the monitor host at 115200 baud.

You can load the monitor executable by using the *JavaKit* program. *JavaKit* is part of the TINI® SDK. The latest version of the the TINI SDK is available at [TINI Firmware Download](#). At the time of this writing, version 1.13 was the latest released SDK. Please follow the instructions in the file *docs\Running_JavaKit.txt* (also part of the TINI SDK download) for installing and running *JavaKit*. If you have any problems getting up and running with *JavaKit*, you may want to search the [TINI interest list archives](#). It is likely that someone else has had similar problems, and has posted a solution online.

From *JavaKit*, open the serial port on your PC that will be communicating with the serial loader of the DS80C400.

This will be the PC serial port that is connected to the serial port located at J12 on the TINI400 socket. Once the port is open, click the RESET button and the DS80C400's loader prompt should appear. The text of the loader prompt is:

```
DS80C400 Silicon Software - Copyright (C) 2002 Maxim Integrated Products
Detailed product information available at http://www.maxim-ic.com
```

```
Welcome to the TINI DS80C400 Auto Boot Loader 1.0.1
>
```

From here, select Load HEX file as TBIN from the File menu, and find the file **mon400.hex**. The load should only take about 5 seconds and you should see the following message print:

```
Loading file: C:\work\monitor\mon400.hex.
Please wait... (ESC to abort.)
Load complete.
```

Now back at the loader prompt, type 'E' and hit RETURN. No message should print, but the monitor should be running at this point. We are now ready to start working with the Keil μ Vision2 IDE.

***HelloWorld* with the Keil Monitor**

Keil has provided a simple, HelloWorld-style application with the source for the monitor. It is in the *Examples/Hello* directory, and can be built and used with the monitor to quickly show its functionality. However, this section walks through the configuration of a simple, new project for use with the Keil monitor so that developers can use the monitor with their own projects.

- 1) Create a new project in the Keil μ Vision2 IDE. We've called ours **hellomon**. When prompted to select a device, find the DS80C400 listed under Dallas Semiconductor. Click on the boxes that read Use Extended Linker and Use Extended Assembler. Click OK to continue.
- 2) When the μ Vision2 IDE asks *Copy Dallas DS80C390 Startup Code to Project Folder and Add File to Project?*, select No. We will supply our own startup code.
- 3) Copy the files *startup400.a51* and *main.c* (available with the source files to this application note in the subdirectory *hellomon*) to the folder where you opened this Keil project. From the μ Vision2 IDE, open *Target 1* and right click on *Source Group 1*. Select *Add Files to Group 'Source Group 1'*. Select the file *main.c* and click *Add*. Go to the *Add Files to Group 'Source Group 1'* dialog again and change the file filter to *Asm Source File*. Click the file *startup400.a51*, and click *Add*. You should now be able to open the item *'Source Group 1'* and see the source files listed.
- 4) Open the file *startup400.a51* by double clicking on it. Find the definition of USE_MONITOR, and set it to 1. The line of code should now look like:

```
$set (USE_MONITOR = 1)
```

Also make sure that XTALMULT is set to 1.

- 5) Right click on *Target 1* and select *Options for Target 'Target 1'*. For memory model, choose *Large: Variables in XDATA*. For Code Rom size, select *Contiguous Mode: 16MB Program*. Select the check boxes for *Use On-Chip Arithmetic Accelerator*, *Use Multiple DPTR registers*, and *'far' memory type support*. For Off-chip Code memory, use a value of **0x200000** for Start and **0x70000** for Size. Under Off-chip XDATA Memory, fill in the first entry with a Start value of **0x10000**, and a Size value of **0x3FFF**.

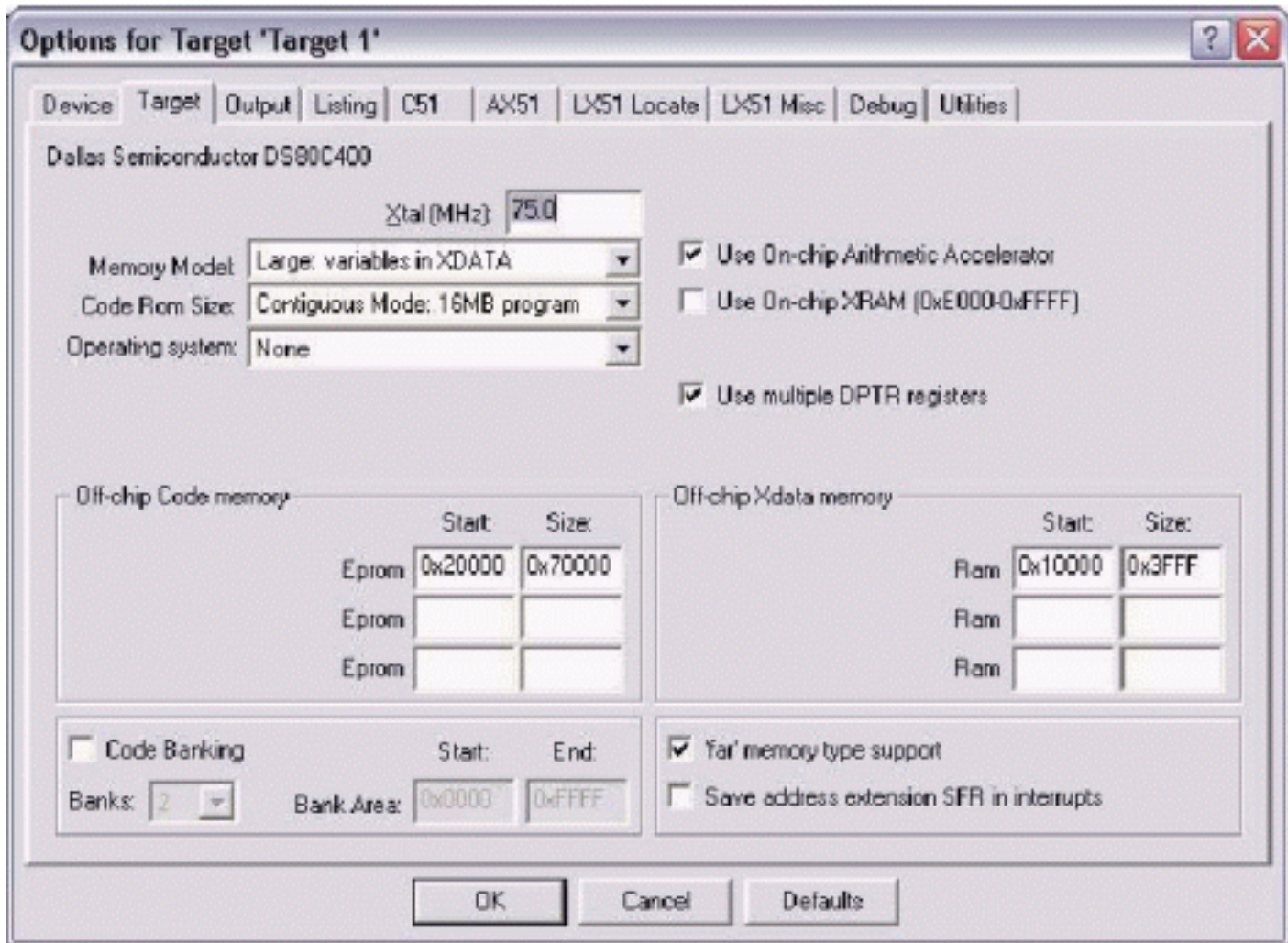


Figure 2. Configuration options for step 5.

- 6) While still looking at the Options for *Target 'Target 1'* dialog, select the *Debug* tab. There are 2 radio buttons at the top of this panel. One says *Use Simulator*, and the other says *Use* and is followed by a drop-down list. Select the radio button on the right with the drop-down list. From the drop-down list, select *MON390: Dallas Contiguous Mode*. Press the *Settings* button. Choose the serial port on your PC that you have connected with serial2 on the TINIs400 socket. Set the baud rate to 115200, and make sure the box marked *Enable Serial Break* is checked. You can now close the options dialog.

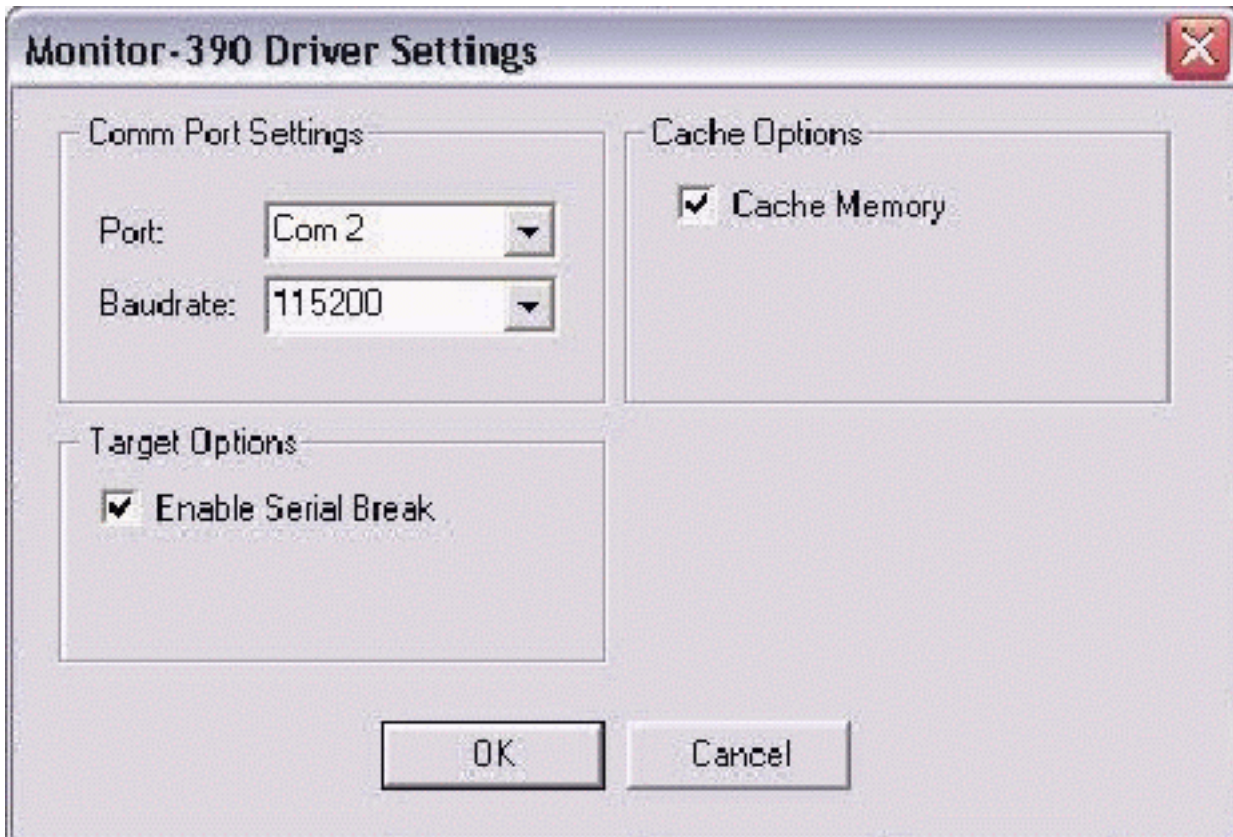


Figure 3. Configuration options for the debugger.

- 7) Press F7 to build the project. It should build without errors.
- 8) With the monitor program running on the TINIm400 (see *Building and Loading the Monitor onto the TINIm400 board*), go under the *Debug* menu and select *Start/Stop Debug Session*. On the left side of the IDE, registers and SFR's should be displayed. The monitor is now running, and we are now debugging.
- 9) Turn on the variable watch window by going under the *View* menu and selecting *Watch & Call Stack Window*. Also enable the *Memory Window* and *Symbol Window* in the same way. In the symbol window, select *Mode* to be *Locals*, and select *Current* to be *MAIN*. This shows you the addresses if the local variables that are used in the `main()` method. In the memory window, type in **i: 0x00** in the edit box labeled *Address*. This window now shows you the values in the internal RAM. You can match up the first few memory values with the registers r0 through r7 shows on the left.

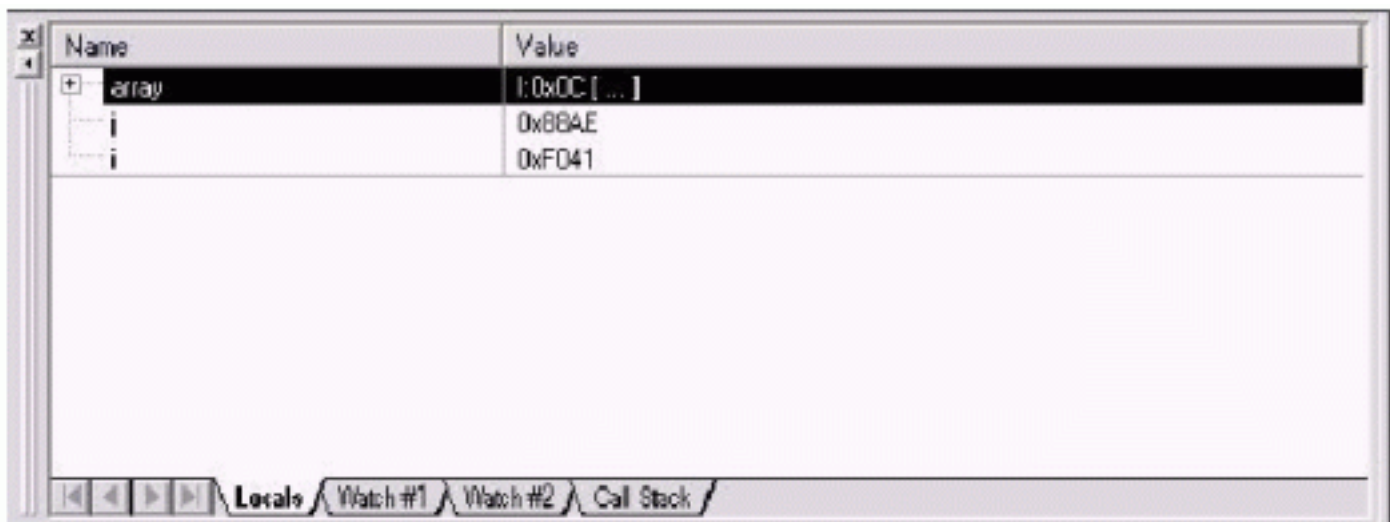


Figure 4. Watch window as seen in the `main()` method.

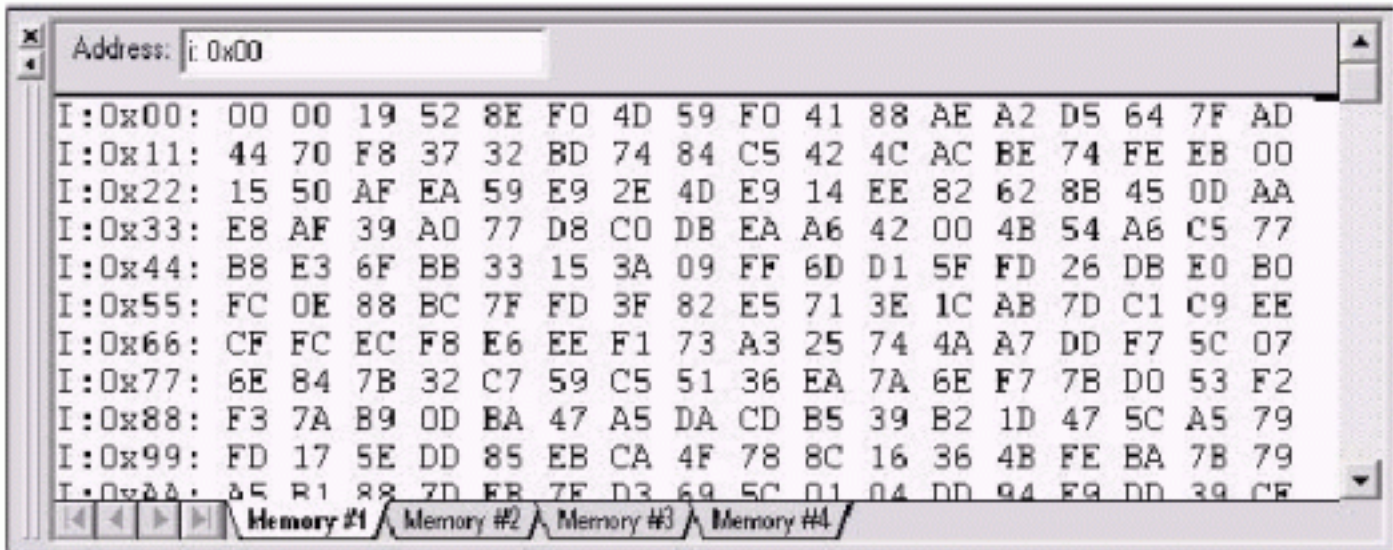


Figure 5. Memory window set to watch internal RAM.

- 10) Go to the source file `main.c` and find the line `printf("DS80C400 Monitor Demonstration\r ")`

Right click somewhere on that line and select *Insert/Remove Breakpoint*. A red box should appear to the left of the code on that line. Now press F5 to run. It should stop at the breakpoint, and a yellow arrow appears to show where in the code we currently are. Also at this point, our local variables **array**, **i**, and **j** should appear in our watch window.

- 11) Press F10 to Step-Over to the next line of code. If you still have *JavaKit* open and connected to the TINI400's serial0 port, then the message "DS80C400 Monitor Demonstration" should have printed to the *JavaKit* terminal. Continue to use F10 to step through the next few lines of code. Note the variables in the watch list updating as new lines of code are run, and the changing of registers and SFR's as well. Also, you can cross-reference the local variables to the memory window using the addresses supplied in the symbol table.

To stop debugging, select *Start/Stop Debug Session* again from the *Debug* menu. You will need to restart the monitor program by resetting the board through *JavaKit* (hit *RESET* and type 'E') or power cycling the board. To interrupt running code (hit the *STOP* button), you need to enable the second serial interrupt. You can do this as the first line of your `hellomon` program with the line:

```
ES2 = 1;
```

However, you'll probably want this functionality for every program that you use the debugger for. Open the monitor project again (**mon400.Uv2**) and find the line *InitSerial*: which follows the line *\$IF (SERIAL = 2)*. Insert the line of code `setb es2` near the end of the function, like this:

```
orl    T3CM, #040H           ; Enable timer 3 to run
orl    SCON2, #002H         ; Serial 2 transmit indicator set
setb   es2                  ; THIS LINE ADDED!!!
JMP    MONSTART
```

Rebuild and reload the monitor. Now, in your `hellomon` application, insert a line into the while loop at the end of the program:

```
while(1)
{
    i++;
```

```
}
```

Now run the program again in the debugger. Press F5 to get past every breakpoint, and the program should enter the infinite loop. Now press the stop button and look at the value of **I** in the watch window. You can also step and watch the value of **I** increment.

Using DS80C400 Libraries with the Keil Monitor

When using the DS80C400 C Libraries in a program, there are some additional issues with using the Keil monitor. Any application that uses one of the C Libraries should call the `init_rom` function, which writes over the interrupt vector table. However, there is some data in the interrupt vector table that the monitor requires. To correct for the alterations done by the `init_rom` function, applications should call the function `init_usekeilmonitor` immediately after calling the `init_rom` function, but only when using the monitor. To help make this more intuitive, the initialization library now contains a macro that will only call the `init_usekeilmonitor` if the line

```
#define MONITOR
```

is present in the application. Follow along with a simple networked application to see how this is done.

- 1) Start a new application for a math server (it will receive 2 16-bit numbers and output their sum). Follow steps 1 through 6 of the previous section to configure the projects for the proper memory usage and for use of the monitor. Instead of the **hellomon** source code, though, use the **mathserver** source code. It can be found in the archive of source for this application note¹. Make sure to also add the library files to the project for the DHCP library, the socket library, the task scheduler, and the initialization library in the same way that you add the files `startup400.a51` and `main.c`. Alternatively, the project file for this project is available - you can just open this file and the project is ready to go.
- 2) First, let's run the application without the monitor to see what it is doing. In the Project Target Options (right click on Target 1), under the Output tab, select the *Create HEX File* checkbox and select *HEX-386* from the drop-down list. Make sure the line near the top of the application `#define MONITOR` is commented out. The value of `USE_MONITOR` in the startup code can stay at 1. Press F7 to build the application. It should build with a number of warnings about functions being uncalled. This refers to unused functions in the libraries, and can be ignored.
- 3) In *JavaKit*, load the generated hex file (in my case, it is `mathserver.hex`). After it has loaded, type in `B 20 <RETURN>` and then `X <RETURN>`. The application should start running, and a few seconds later, it should report the address it leases from a DHCP server.
- 4) From a command prompt, compile the file `connect.java`. This Java program is the client side of the application. You can compile by simply typing:

```
javac connect.java
```

- 5) Now run the connect program on the PC. Use the IP address reported through JavaKit as your IP address argument. The second argument is the port number, which should be 15555. Your command line should look something like this:

```
java connect 180.0.6.131 15555
```

The DS80C400 code will print out the numbers it receives in hex and decimal:

```
T1 is 3241 and T2 is a747 (hex)
T1 is 12865d and T2 is 42823d
Total is 55688
T1 is 78ed and T2 is ce1c (hex)
T1 is 30957d and T2 is 52764d
Total is 83721
T1 is f2e6 and T2 is e49 (hex)
T1 is 62182d and T2 is 3657d
Total is 65839
```

The PC will verify that the sum is correct and print the message *It's all good.*

- 6) Stop the client and let's go back to the Keil IDE to do some debugging.
- 7) Look in the file *main.c* for the function *init_rom*. Note that immediately after this function is called, the macro *USE_KEIL_MONITOR* is called. When the value *MONITOR* is not defined, this macro does nothing. When the value *MONITOR* is defined (before the file *rom400_init.h* is included), this macro calls a function that restores some settings wiped out by the call to *init_rom*. Go to the top of the file *main.c* and uncomment the line *#define MONITOR*. Press F7 to build the project.
- 8) Load the monitor program using *JavaKit* and run it. Go to the Debug tab on the Project Options window and make sure the correct COM port and baud rates are selected. Now start debugging using the monitor. Set a breakpoint on the first *printf* statement of the *main* function, and press F5 run to it. Now step through the functions in *main()* using the F10 key. When you step over the line *do_dhcp()*, watch the output on JavaKit as the DS80C400 tries to get a DHCP address.
- 9) Set a breakpoint on the line *if (temp != 0xFFFF)* immediately after the call to *recv* in the function *tcp_test()*. Press F5 to run and watch the output from *JavaKit*. Run the *connect* program when prompted, and the debugger should hit the new breakpoint. You can now step through the loop. Make sure you use the Go command (hit F5) to run from before the call to *send* until after the call to *recv*. Network communications may fail from the monitor pausing the DS80C400's execution.

¹http://files.dalsemi.com/tini/ds80c400/c_libraries/appnotes/monitor_appnote_source.zip

Limitations of Working with the Keil Monitor

There are currently some limitations to using the Keil monitor. Hopefully, these limitations will be reduced or removed as Keil releases future versions of the monitor.

You must use a crystal multiplier (XTALMULT) of 1 when using the Keil monitor.

The Keil monitor is not ideal for debugging network applications. The monitor stops all interrupts from firing while waiting on the user to tell the application to continue, which can cause network traffic to be missed. *Ethereal* (<http://www.ethereal.com/>) is a good tool for debugging network transmissions.

The monitor program and the Keil debugger may occasionally lose communication. Often, telling the debugger program to *Try Again* will restore communication. If it doesn't, you will probably need to restart the monitor program using *JavaKit* and start debugging again.

Conclusion

The Keil Monitor for Dallas Semiconductor contiguous mode parts is a useful tool for tracing through and debugging source code. Combined with other methods such as setting output bits and writing debug information to the serial port, the monitor is part of a suite of techniques for debugging applications written for the DS80C400 processor.

Relevant Links

[Source code](#) (ZIP) for this appnote

[TINI Software Development Kit](#)

[Keil Software Development Tools](#)

[High-Speed Microcontroller User's Guide: Network Microcontroller Supplement](#)

[C Library Project Homepage](#)

[Java Development Kit Download Page](#)

[Java Communications API](#)

Application note 613, [Using the Keil C Compiler for the DS80C400](#)

Application Note 2777: www.maxim-ic.com/an2777

More Information

For technical support: www.maxim-ic.com/support

For samples: www.maxim-ic.com/samples

Other questions and comments: www.maxim-ic.com/contact

Automatic Updates

Would you like to be automatically notified when new application notes are published in your areas of interest?

[Sign up for EE-Mail™.](#)

Related Parts

DS80C400: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [TransMissing\[Free_Samples2\]](#)

DSTINIM400: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)

DSTINIS400: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)

AN2777, AN 2777, APP2777, Appnote2777, Appnote 2777

Copyright © by Maxim Integrated Products

Additional legal notices: www.maxim-ic.com/legal