

Using Motorola's HCS12 Serial Monitor on Wytec's Dragon-12 boards

Wytec's Dragon-12 development boards are pre-installed with *DBug-12*, a small monitor program which allows a user to interact with the board through a serial connection and with the aid of a standard terminal program (e.g. *HyperTerminal*). Unfortunately, when it comes to using Metrowerks' CodeWarrior IDE and the built-in source level debugger (Hi-Wave) *DBug-12* seems to be of rather limited use: The line speed of the host-target communication interface appears to be limited to a maximum data transmission rate of 57600 bps. Furthermore, all applications need to be run from within RAM as no hardware breakpoints seem to be supported.

This contribution proposes an alternative to *DBug-12*, namely the use of the compact *HCS12 Serial Monitor* (Motorola, cf. application note AN2548/D). This small program can be placed in the protected area of Flash EEPROM block 3 (0xF800 – 0xFFFF, 2 kByte). The monitor communicates with an external host application through a serial connection on SCI0 with a line speed of 115200 bps. In contrast to *DBug-12* with its clear text commands, the *HCS12 Serial Monitor* uses a concise set of 1-byte commands with optional parameters. On the host, a customised terminal program is required to translate clear text user interactions such as 'read registers' or 'set breakpoint' into the corresponding sequence of HCS12 Serial Monitor commands.

One such application is *uBug12*, a small terminal program which has been developed by *Technological Arts* for their MC9S12 based products. Alternatively, the CodeWarrior source level debugger (Hi-Wave) can be configured for the HCS12 Serial Monitor. Metrowerks' *Generic Debug Instrument* interface (GDI) provides a specification for general purpose interaction between Hi-Wave and external applications. Making use of a small driver application (*hcs12serialmon.dll*) Hi-Wave can produce the command sequences which are in full compliance with the HCS12 Serial Monitor user interface.

Both *uBug12* as well as the Hi-Wave debugger allow the downloading of user applications into RAM and/or the Flash EEPROM of the microcontroller. With *uBug12*, a suitable S-Record has to be generated. Programs destined for Flash EEPROM are downloaded using the host command *fload*. Both S2 records as well as the older S1 records (command line option '*;b*') are supported. Some applications might rely on the provision of an interrupt vector table. HCS12 Serial Monitor expects this vector table in the address space from 0xFF80 – 0xFFFF. This is within the protected area of the Flash EEPROM. The monitor therefore places the supplied interrupt vectors in the address space just below the monitor program (0xF780 – 0xF800) and maps all original interrupts (0xFF80 – 0xFFFF) to this *secondary vector table*. From a programmer's point of view, the vector table always goes into the space from 0xFF80 – 0xFFFF.

Implementation details

The source code of the HCS12 Serial Monitor can be obtained from Motorola's web site (www.motorola.com, see application note *AN2548/D* and the supporting CodeWarrior project *AN2548SW2.zip*).

A few minor modifications are necessary to adapt this project to the hardware of the Dragon-12 development board:

- (1) It is proposed to use the on-board switch SW7\1 (EVB, EEPROM) as LOAD/RUN switch. The state of this switch is tested by the monitor during reset. Placing SW7\1 in position 'EVB' (see corresponding LED) forces the monitor to become active (LOAD mode). Switching SW7\1 to position 'EEPROM' diverts program execution to the user code (jump via the secondary RESET interrupt vector at 0xF77E – 0xF77F).
- (2) The monitor is configured for a crystal clock frequency of 4 MHz, leading to a PLL controlled bus speed of 24 MHz.
- (3) The CodeWarrior project (*AN2546SW2.zip*) has been modified to produce an S1-format S-Record. The frequently problematic S0 header has been suppressed and the length of all S-Records has been adjusted to 32 bytes. This makes it possible to download the monitor program using DDebug-12 and two Dragon-12 boards connected in BDM mode.

The protected Flash EEPROM area of the MC9S12DP256C (Dragon-12) can only be programmed using a BDM interface. An inexpensive approach is to use two Dragon-12 boards connected to each other via a 6-wire BDM cable. The *master board* should run DDebug-12 (installed by default) and be connected to the host via a serial communication interface (9600 bps, 8 bits, no parity, 1 stop bit, no flow control). A standard terminal program (e.g. HyperTerminal) can be used to control this board. The *slave board* is connected to the master board through a 6-wire BDM cable. Power only needs to be supplied to either the master or the slave board (i. e. one power supply is sufficient, see Figure 1).

The BDM interface of DDebug-12 allows the target system (slave) to be programmed using the commands *fbulk* (erases the entire Flash EEPROM of the target system) and *fload* (downloads an S-Record into the Flash EEPROM of the target system). Unfortunately, DDebug-12 requires all S-Records to be of the same length, which the S-Records produced by CodeWarrior are not. CodeWarrior has therefore been configured to run a small script file (*//bin/make_s19.bat*) which calls upon Gordon Doughman's S-Record conversion utility *SRecCvt.exe*. The final output file (*S12SerMon2r0.s19*) is the required S-Record in a downloadable format.

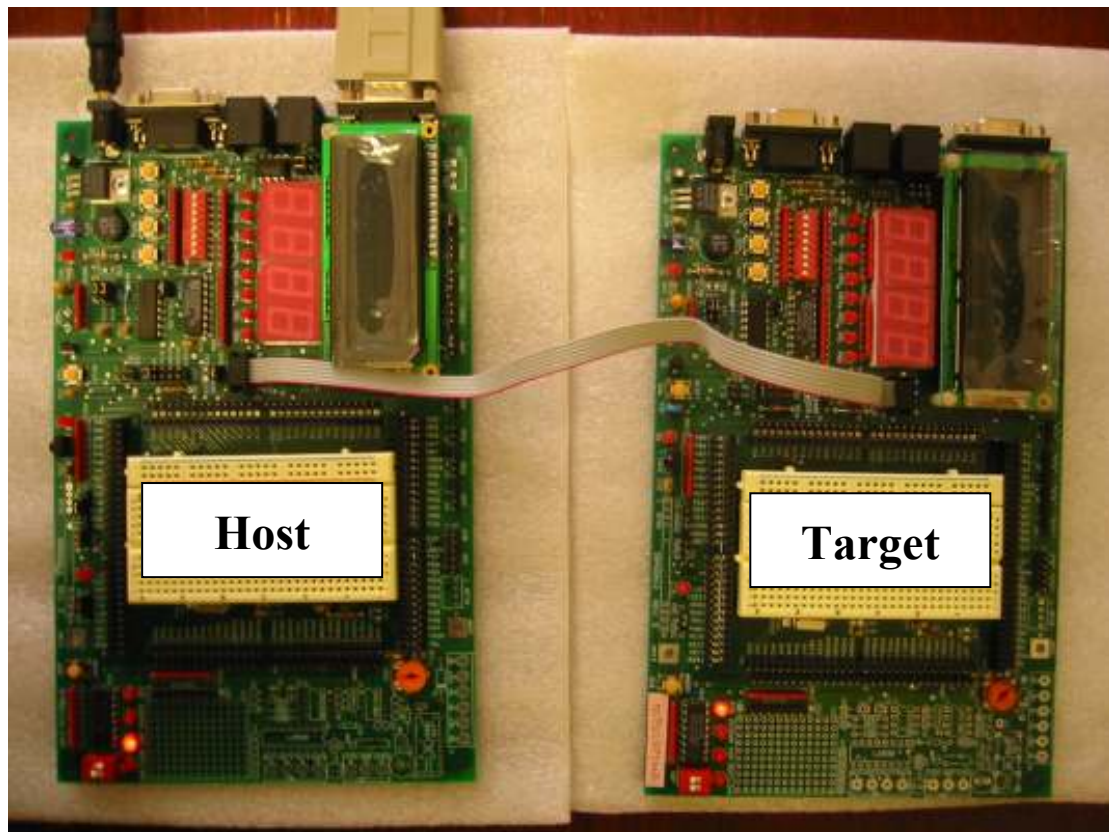


Figure 1 Connecting two Dragon-12 boards via a BDM cable

Click on the green debug button (cf. Figure 5) to build the monitor program and convert the output file to a downloadable S-Record file.

Start HyperTerminal and reset the host board. Make sure the host board is set to *POD mode*. You should be presented with a small menu (Figure 2). Select menu item (2) to reset the target system. You should be presented with a prompt '*S>*' indicating that the target system is stopped.

Erase the Flash EEPROM of the target system by issuing the command *fbulk*. After a short delay, the prompt should reappear.

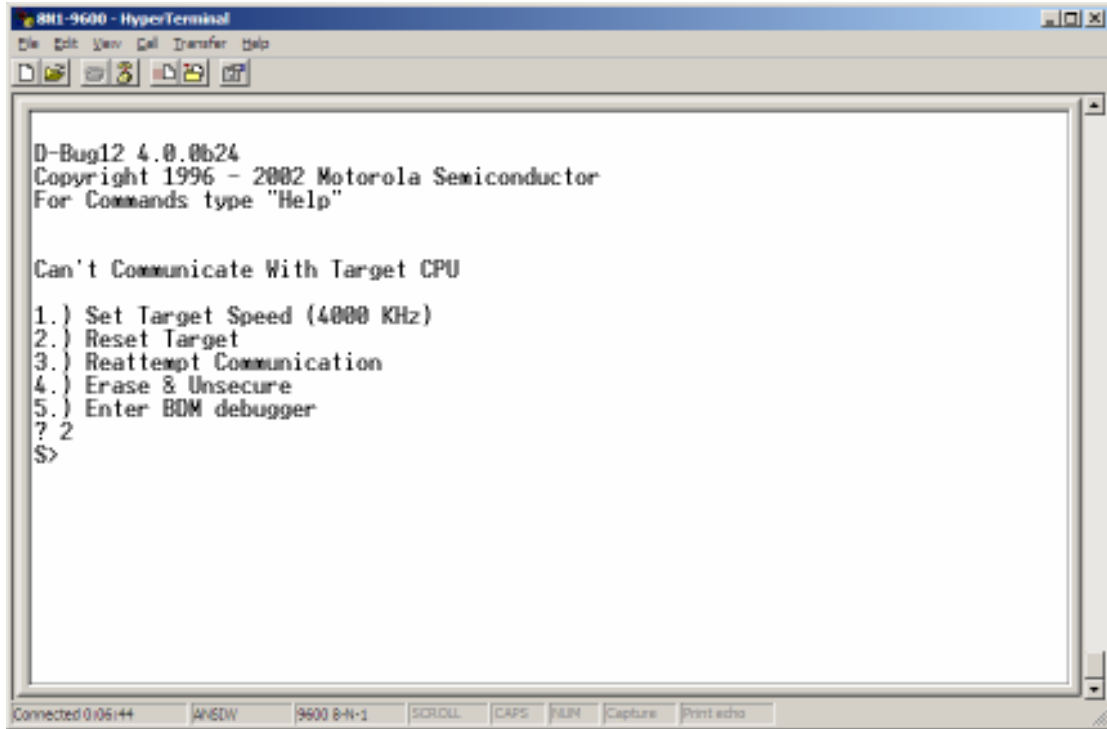


Figure 2 POD mode, host system

Issue the command *fload ;b* to download the S-Record file of the *HCS12 Serial Monitor*. Option *‘b’* indicates that this file is in S1-format (as opposed to S2). From the *Transfer* menu select *Send Text File...*. Find and select the file *S12SerMon2r0.s19*. You will have to switch the displayed file type to *‘All Files (*.*)’* (see Figure 3). Click on *Open* to start the download.

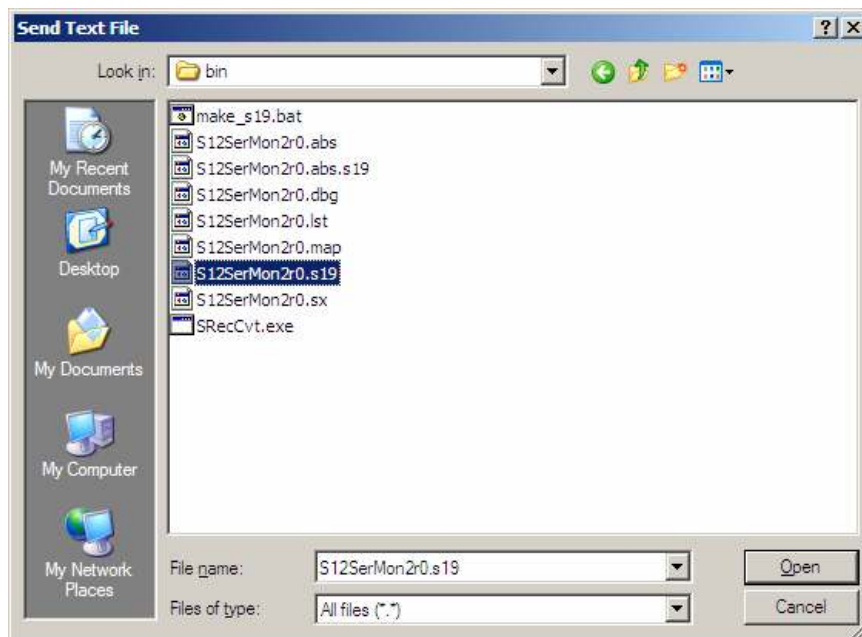
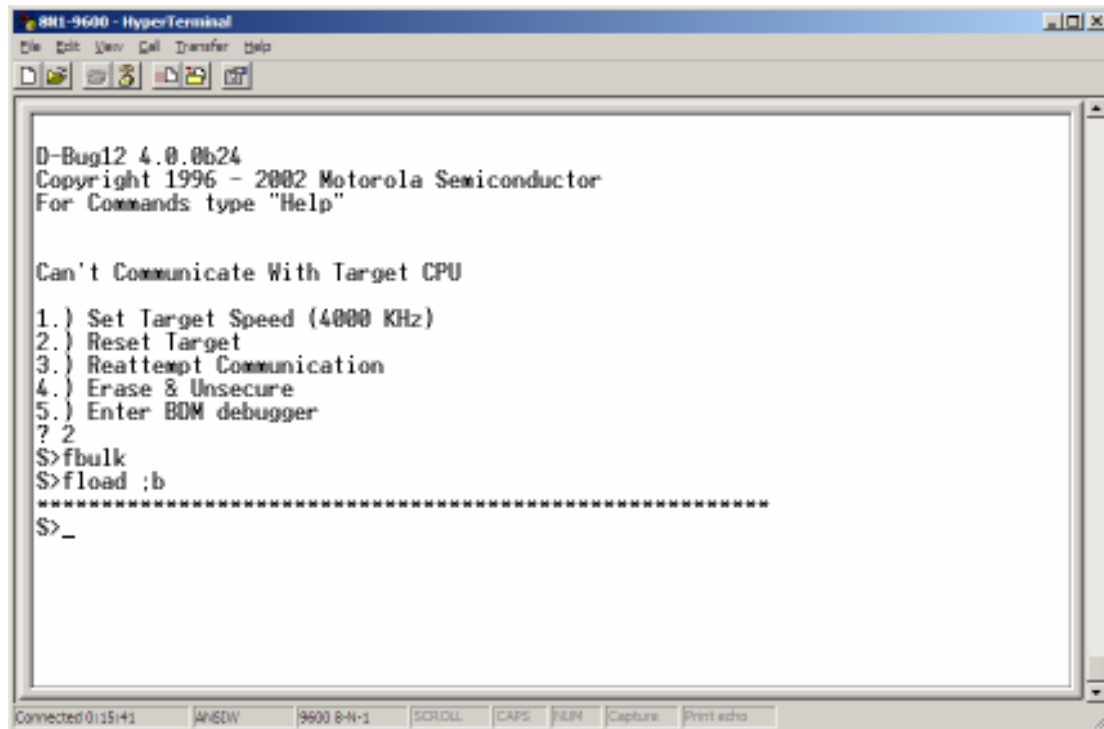


Figure 3 Downloading the *HCS12 Serial Monitor*

Once the download is complete, you should be presented with the prompt (Figure 4). The *HCS12 Serial Monitor* has successfully been written to the target board. Disconnect the BDM cable from the target system and close the terminal window.



```
D-Bug12 4.0.0b24
Copyright 1996 - 2002 Motorola Semiconductor
For Commands type "Help"

Can't Communicate With Target CPU

1.) Set Target Speed (4000 KHz)
2.) Reset Target
3.) Reattempt Communication
4.) Erase & Unsecure
5.) Enter BDM debugger
? 2
$>fbulk
$>fload ;b
*****
$>_

Connected 0:15:41  ANSI  9600 8-N-1  SCROLL  CAPS  RUN  Capture  Print echo
```

Figure 4 Download complete

Connect the serial interface SCI0 of the target system to the serial port of the host computer and start uBug12.

Ensure that you can connect to the target by entering 'con 1' (for serial port COM1). You should receive the acknowledgement message 'CONNECTED'. Issue the command 'help' to see what options are available to control the monitor. Disconnect from the target using command 'discon' (Figure 5).

Note that *HCS12 Serial Monitor* provides set of commands very similar to that of *DBug-12*: A user application can be downloaded into the Flash EEPROM of the microcontroller using *fload (;b)*, the unprotected sectors of Flash EEPROM can be erased using *fbulk*, etc. Following the download of a user program into the Flash EEPROM of the microcontroller, the code can be started by switching SW7\1 to RUN (EEPROM) and resetting the board (reset button, SW6).

```

uBug12
File Help
>con 1
CONNECTED
>help

--- REGISTER ---
RD - Register Display
RM <RegisterName> <Data8/16> - Register Modify
CCR <Data8> - Set CCR register
D <Data16> - Set D register
PC <Data16> - Set PC register
PP <Data8> - Set PP register
SP <Data16> - Set SP register
X <Data16> - Set X register
Y <Data16> - Set Y register
--- MEMORY MODIFY ---
BF <StartAdd> <EndAdd> <Data8> - Block fill byte
BFW <StartAdd> <EndAdd> <Data16> - Block fill word
MD <StartAdd> [<EndAdd>] - Memory display
MDW <StartAdd> [<EndAdd>] - Memory display word
MM <Address> <Data8> - Memory modify byte
MMW <Address> <Data16> - Memory modify word
--- FLASH ---
FBULK - Flash bulk erase
FLOAD [;B][;M] - Flash load
--- DEVICE INFO ---
DEVICE - Get device name
--- GO/HALT ---
GO [<StartAddress>] - Start execution
HALT - Halt execution
RESET - Reset target
TRACE - Execute one instruction
--- GUI ---
CON <Comport> - Connect to target
DISCON - Disconnect from target
EXIT - Terminate GUI
HELP - Display help
OP <Opacity%> - Set main gui opacity

>discon
DISCONNECTED

Cold Reset Executed Unknown Error COM 1

```

Figure 5 Testing the monitor with uBug12 (Technological Arts)

This completes the installation and testing of *HCS12 Serial Monitor* on the Dragon-12 development boards. The following section outlines how this monitor can be used with CodeWarrior's source level debugger *Hi-Wave*.

Source level debugging with Metrowerks' CodeWarrior and the HCS12 Serial Monitor via GDI Target Interface

The CodeWarrior IDE includes a powerful source level debugger (Hi-Wave) which can be connected to HCS12 hardware through a background debug module (BDM) or using the serial interface of the microcontroller. An interface standard, the so-called *Generic Debug Instrument (GDI)* interface, has been defined by Metrowerks to provide 3rd party manufacturers with a specification for the development of BDM hardware and/or monitor programs. CodeWarrior currently ships with GDI drivers for a number of debugging environments, including Motorola's HCS12 Serial Monitor (*/prog/hcs12serialmon.dll*).

This section describes the necessary steps which allow the CodeWarrior debugger to be used with Dragon-12 boards (Wytec). It is assumed that the default monitor program *DBug-12* has now been replaced by Motorola's *HCS12 Serial Monitor* (see previous section).

The debugger is commonly started from within the CodeWarrior IDE by clicking on the green 'play/debug' button (Figure 6) or through menu *Project* → *Debug*.

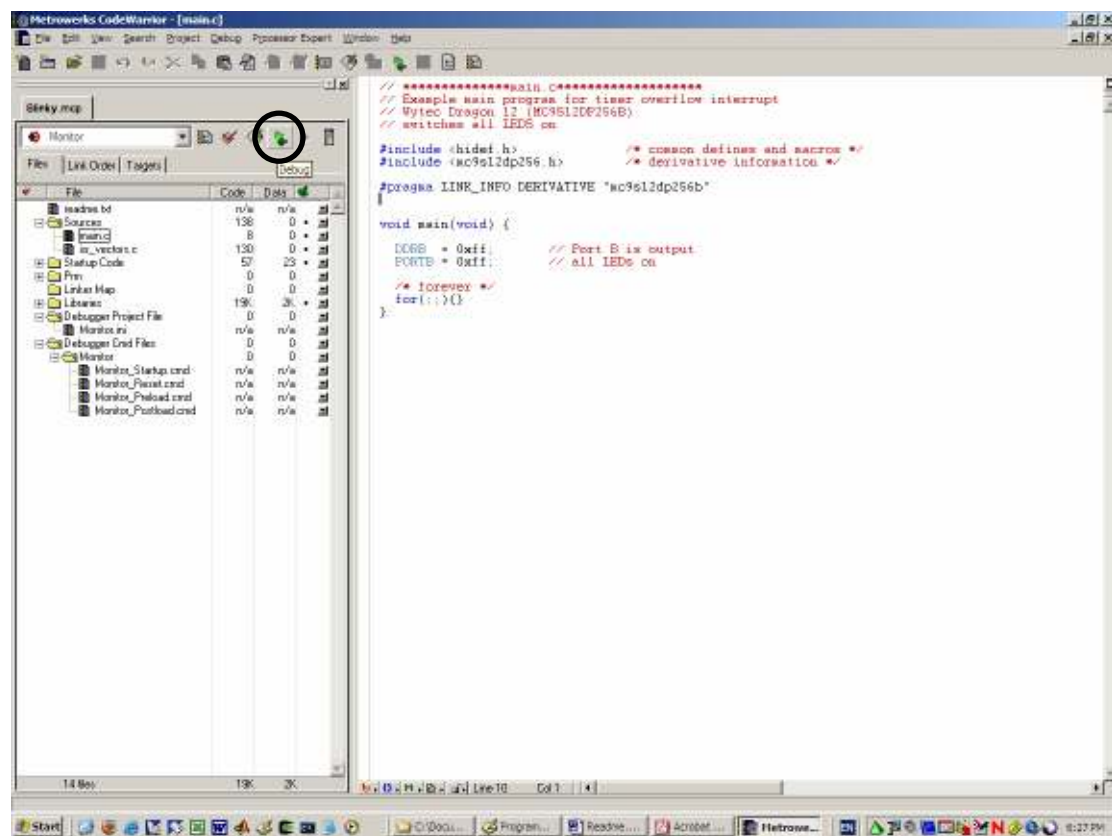


Figure 6 Launching the debugger from within the CodeWarrior IDE

However, depending on the mask set of the MC9S12DP256B/C microcontroller (0K79X, 1K79X, 2K79X) some of the CodeWarrior configuration files might have to be modified before the GDI driver recognizes the Wytec board.

During the start-up of the CodeWarrior debugger (Hi-Wave), a short sequence of commands is transmitted between host and target. This allows Hi-Wave to gather some information about the connected target. One of the target specific parameters is the *part ID* of the microcontroller, a 2-byte reference number stored in memory location \$001A/\$001B. The debugger uses this part-ID to locate and load a *register set* configuration file which determine the overall appearance of the user interface (register window, etc.). The map between part-ID and register set is defined in the configuration file */prog/hc12.ini*. This file may need to be modified to make the debugger work with some Dragon-12 boards.

Procedure:

Connect the board to be programmed to the serial port of the host PC and launch uBug12. Ensure that the LOAD/RUN switch (SW7) is set to LOAD mode (EVB). Connect to the board using the uBug12 command 'con 1'. You should be presented with the message 'CONNECTED'. Display the contents of memory location \$001A using the command 'mdw 001a' (memory display word). Find the part-ID stored in \$001A/\$001B (in our case: *0012*, see Figure 7). Disconnect from the target using the command 'discon'.

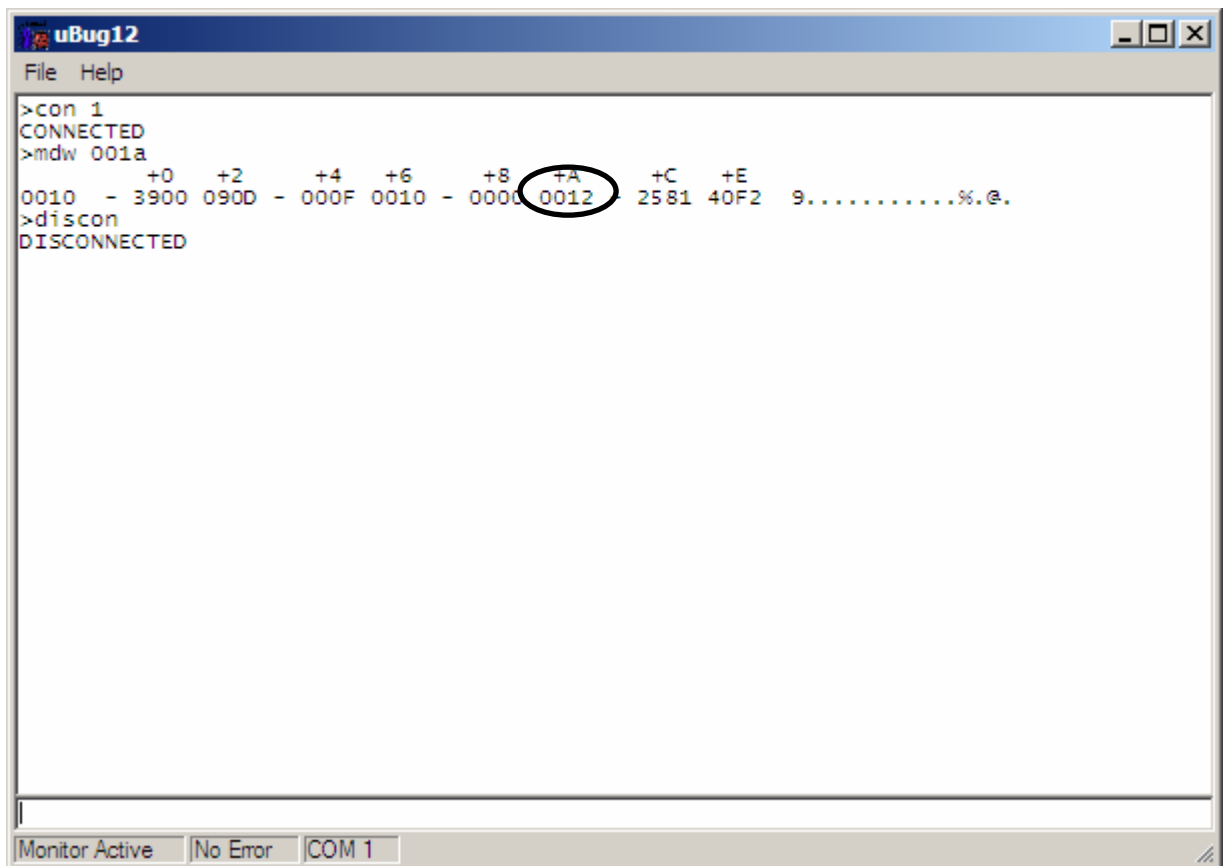


Figure 7 Using uBug12 to find the part-ID of the MC9S12DP256C

Open the configuration file ...*Metrowerks\CodeWarrior CW12_V3.0\prog\hc12.ini* and look for the entry [MULTIPLEMCUIDS PARTIDS]. You should be presented with a list of part-IDs known to CodeWarrior:

```
[MULTIPLEMCUIDS PARTIDS]
PARTID0=0x0010
PARTID1=0x0011
PARTID2=0x0100
PARTID3=0x0200
PARTID4=0x0300
PARTID5=0x0400
PARTID6=0x1000
PARTID7=0x5100
PARTID8=0x6300
PARTID9=0x3100
PARTID10=0x0101
PARTID11=0x2100
PARTID12=0x3300
PARTID13=0x4220
PARTID14=0x5202
PARTID15=0x5102
```

The required part-ID (i. e. 0x0012 or whatever came up in your case) might not be listed. In this case, a new entry will have to be created and added to the existing list:

```
PARTID16=0x0012
```

The above line defines the new label *PARTID16* as reference to a microcontroller with a part ID of 0x0012 (MC9S12Dx256C, mask set: 2K79X). This new reference will now have to be linked to the register definition file of the MC9S12DP256B/C (used on the Dragon-12). Create a new entry [PARTID0012] with the following definitions:

```
[PARTID0012]
;Barracuda-2 2K79X
;MC9S12DP256B
MCUID0=0x3C6
;MC9S12DT256B
MCUID1=0x3D7
;MC9S12DJ256B
MCUID2=0x3D8
;MC9S12DG256B
MCUID3=0x3D9
;MC9S12A256B
MCUID4=0x3CA
```

This block lists the individual *MCUID reference numbers* for all members of the MC9S12Dx256B/C family. The MC9S12DP256B/C is mapped to the *MCUID 0x3C6*. Save the modified configuration file. This was the tricky part...

We can now begin to configure the GDI interface of the Hi-Ware debugger for Motorola's HCS12 Serial Monitor. Start CodeWarrior with a project of your choice. A small sample project (*led_test.mcp*) has been included in this contribution. This project simply activates the on-board LEDs of the Dragon-12 (port B):

```
/* Example program for the Wyttec Dragon 12 (MC9S12DP256B) */
#include <mc9s12dp256.h>          /* derivative information */

void main(void) {

    DDRB = 0xff;                // Port B is output
    PORTB = 0xff;               // all LEDs on

    /* forever */
    for(;;){}
}
```

The remainder of this document assume that *led_test.mcp* has been loaded into the CodeWarrior IDE. Note that this project has already been configured for the Hi-Wave debugger: The *Build Extras* options page (Edit → Monitor Settings...) specifies a debugger initialisation file *Monitor.ini* (see Figure 8). This file will be created when launching the debugger for the first time.

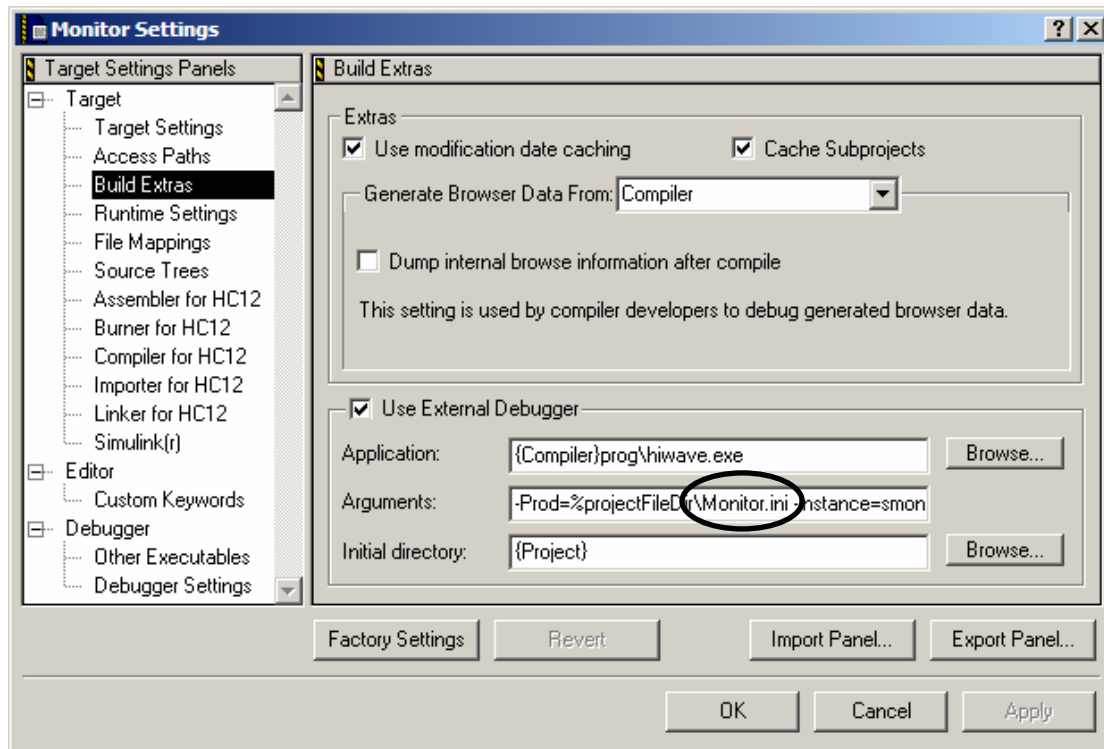


Figure 8 Configuring project *led_test.mcp* for Hi-Wave

Launch the debugger by clicking on the small green 'play' button (see Figure 6). A warning appears, pointing out that the configuration file *Monitor.ini* can not be found and Hi-Wave asks if a new Target should be installed. Ignore the warning and click on *Yes* (Figure 9).

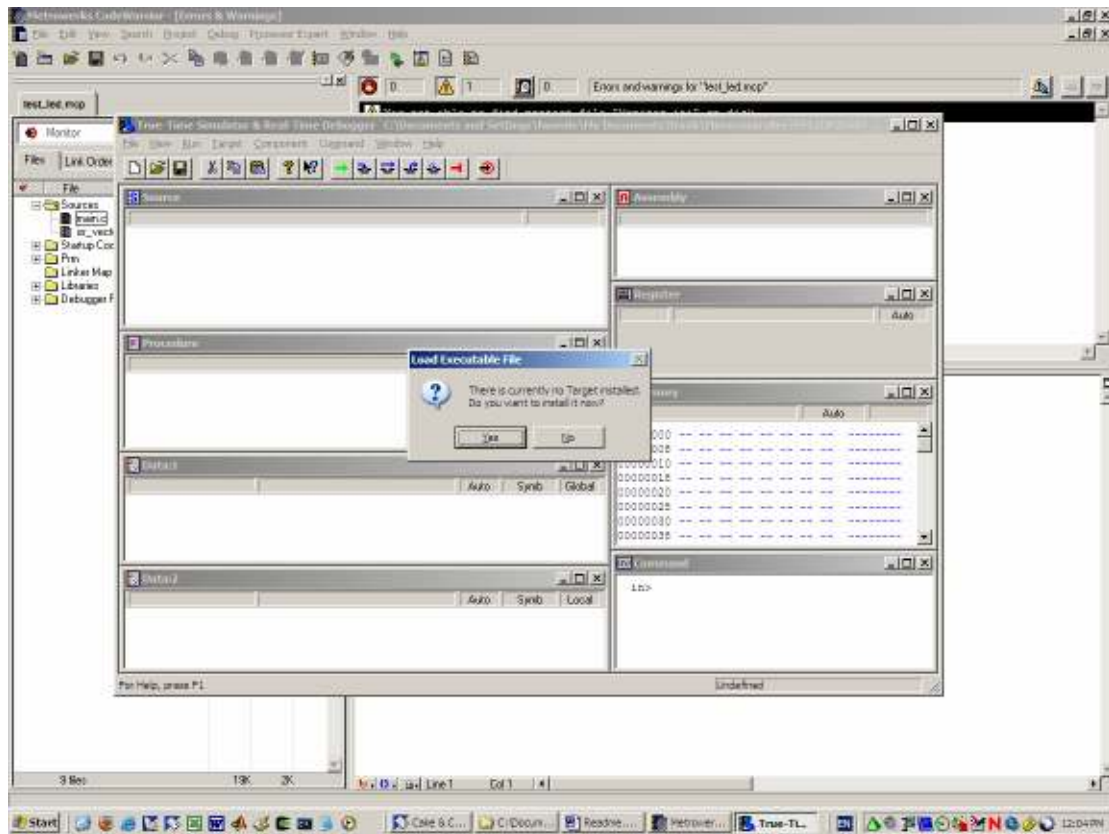


Figure 9 Launching the debugger for the first time

In the appearing *Set Target* window choose *HC12* and *GDI Target Interface* and click on *OK* (Figure 10).

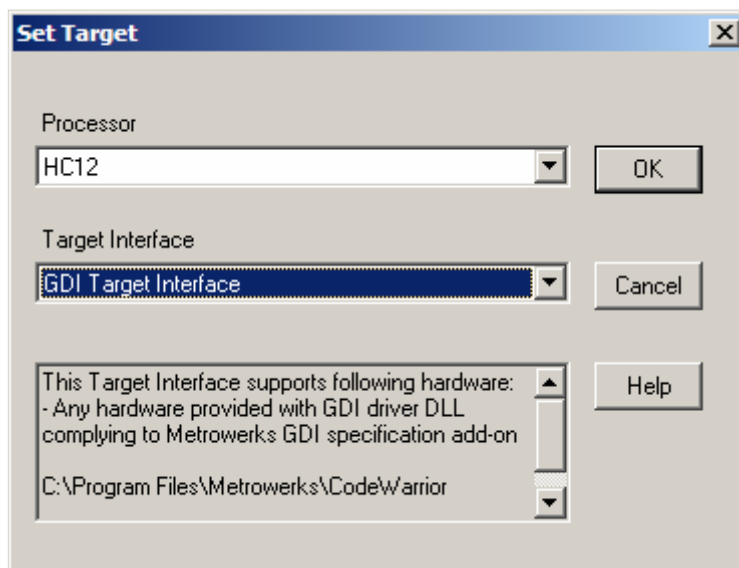


Figure 10 Setting the target interface of the debugger

Set the GDI Driver DLL to

...*Metrowerks\CodeWarrior CW12_V3.0\prog\hcs12serialmon.dll*

and click on *OK* (Figure 11).

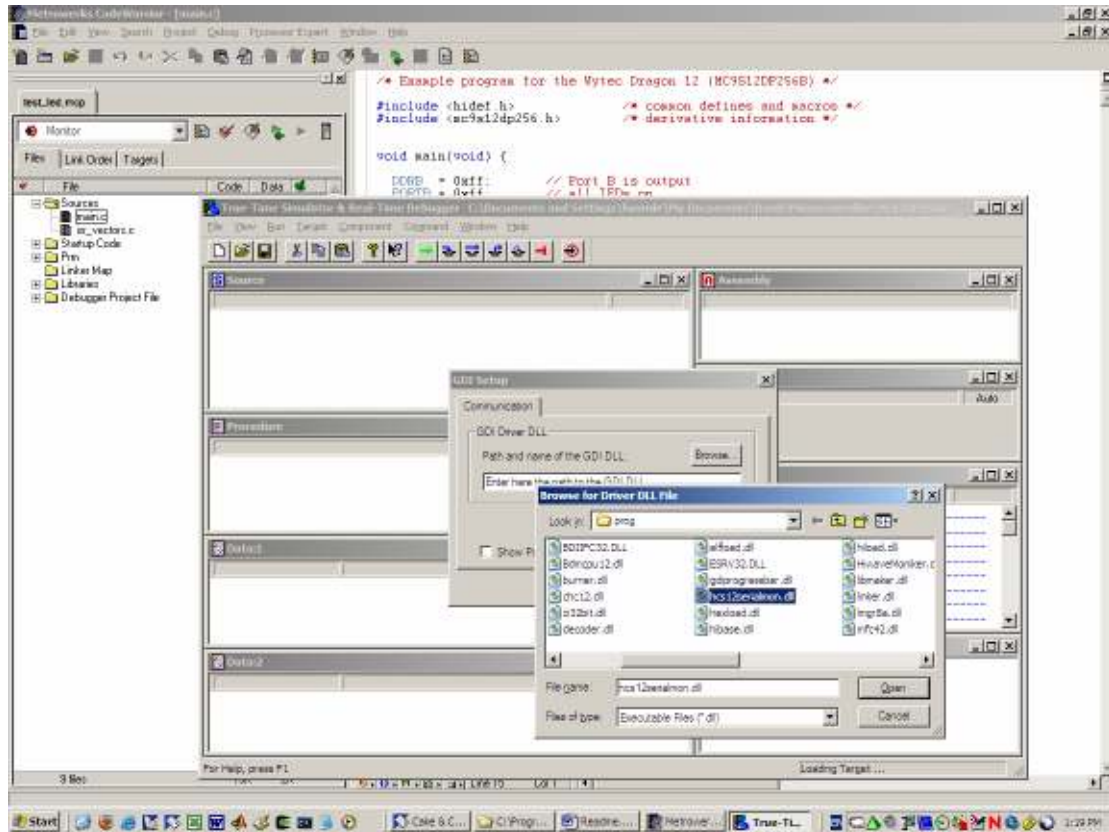


Figure 11 Choosing the GDI Driver DLL

In the emerging requester, select the serial port of the host (COM1, COM2, ...) and click *OK* (Figure 12).

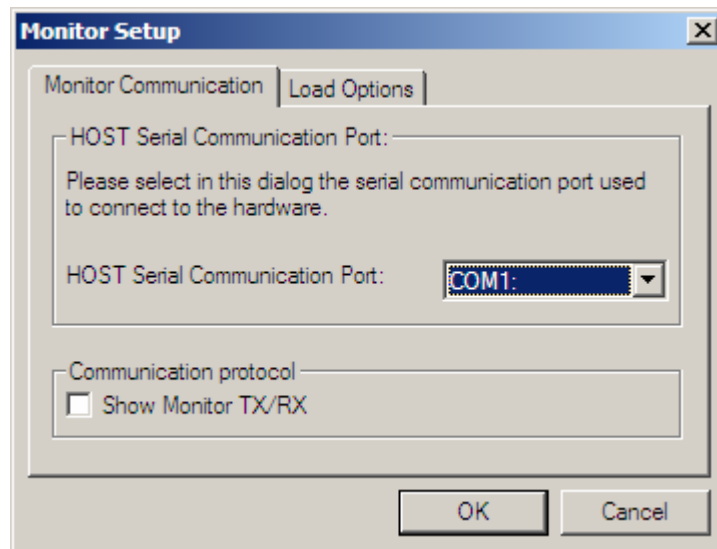


Figure 12 Setting the host communication parameters

The debugger now tries to read the *part-ID* of the connected hardware and displays the returned MCUID (should be 0x3C6 for the MC9S12DP256C on the Dragon-12, see Figure 13).

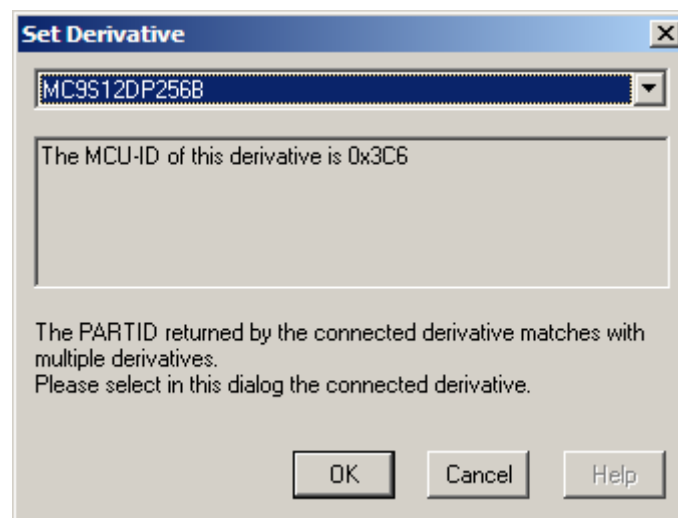


Figure 13 Selecting / confirming the microcontroller derivative

Confirm the detected MCUID. This triggers the erasing of the Flash EEPROM followed by the downloading of the user code (led_test.abs). Upon successful completion of this download you should be presented with the following screen (Figure 14).

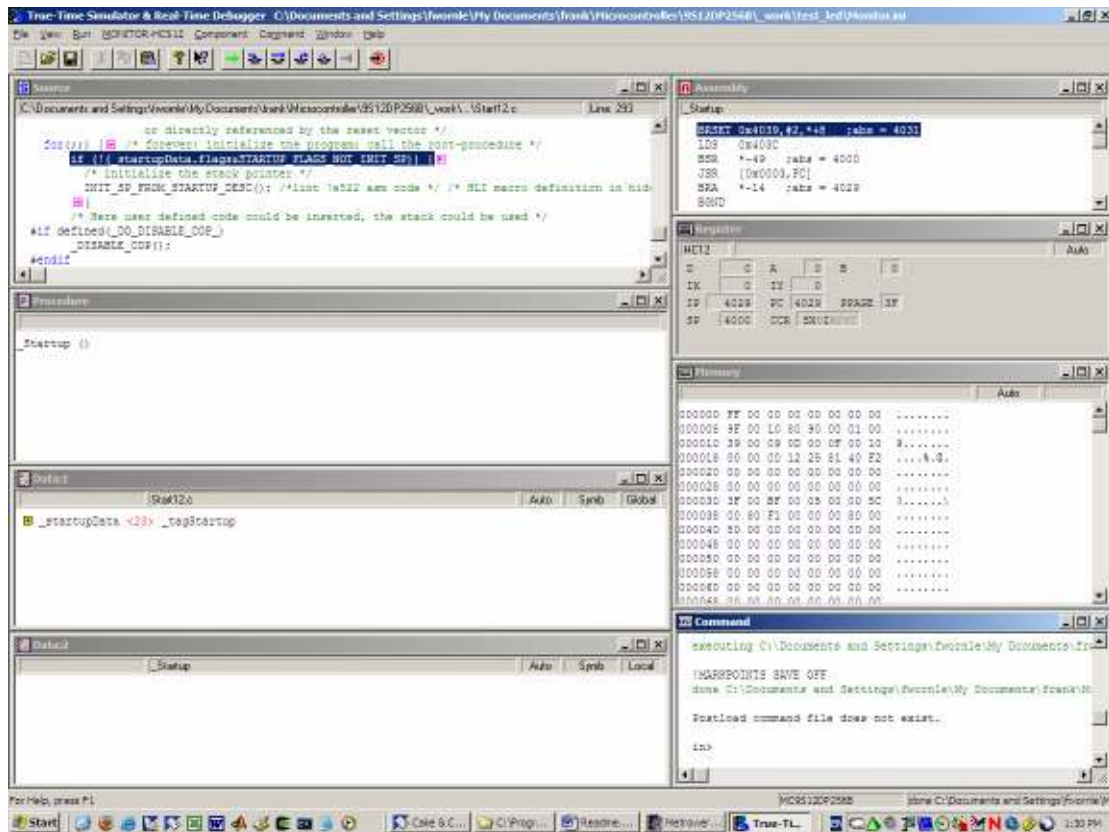


Figure 14 Hi-Ware debugger, ready to go...

Note that the instruction pointer (IP, PC) has been loaded with the start address of the program (initialisation routine `_Startup`). You can single-step through the code until you reach the beginning of the `main` function of the `led_test` program (Figure 15). Continue to step through the program. As you step over the line

```
PORTB = 0xff; // all LEDs on
```

the LEDs should come on. All subsequent steps are ineffective as the microcontroller is blocked by the following line:

```
/* forever */
for(;;){}
```

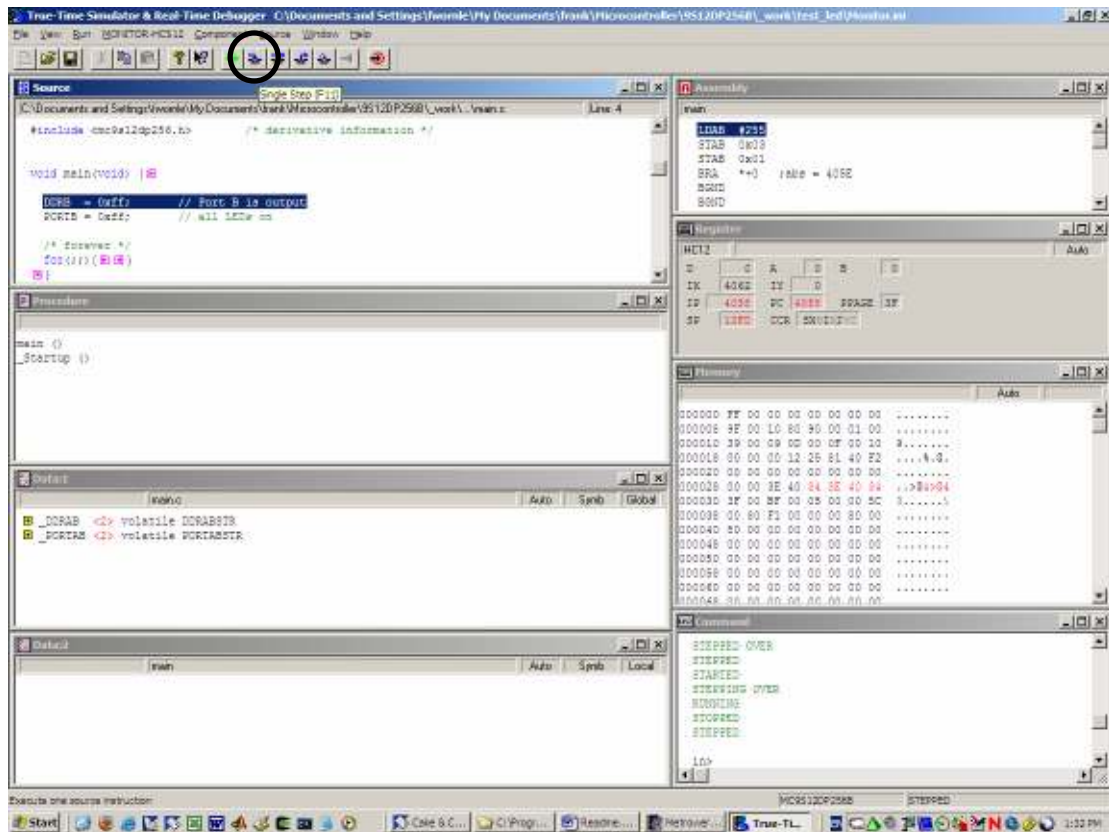


Figure 15 Single-stepping through the code

Notice that, while we are single stepping through lines of C-code, the corresponding assembly instructions are shown in the assembly window. You can customize this display by right-hand mouse clicking inside the assembly window. A small pop-up menu should appear. Select the displaying of *symbolics*, *addresses* and *absolute addresses* (Figure 16).

To restart the debugging process, click on the RESET button within the toolbar (small red circle/black arrow, see Figure 16). The monitor should jump back to the beginning of the *_Startup* routine and you can start over again.

The same can be achieved by resetting the target board. Press the *reset* push-button of the Dragon-12 board (SW6). As before, the monitor should jump back to the beginning of the *_Startup* routine.

Exit the debugger by selecting File → Exit.

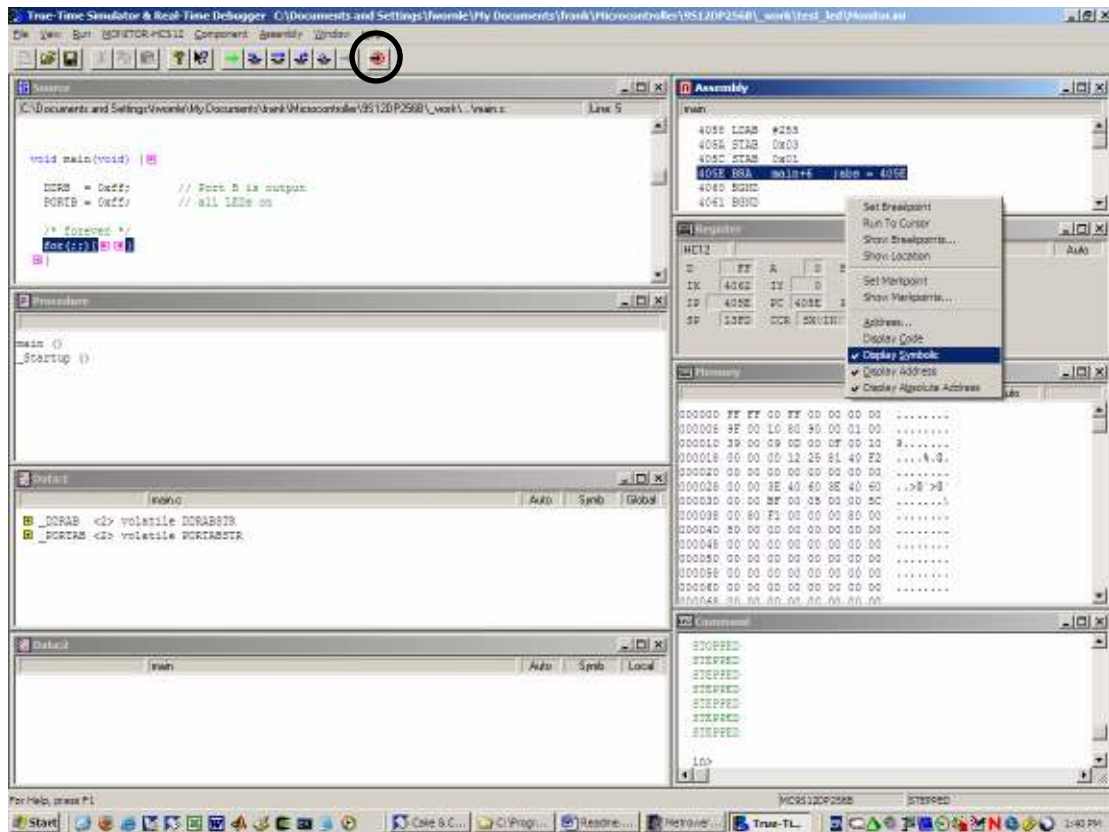


Figure 16 Customising the display, resetting the target

For many users, the internals of the `_Startup` routine are of little to no interest. Hi-Wave can be configured to hide the initialisation and return control at the beginning of `main`. This is done using a number of *initialisation commands*.

The project `test_led` includes an entry called *Debug Project File*. While not really contributing to the code building process, this entry provides convenient access to the debugger configuration file `Monitor.ini`. This file has been created during the first run of the debugger and can be used to customise its appearance (Figure 17).

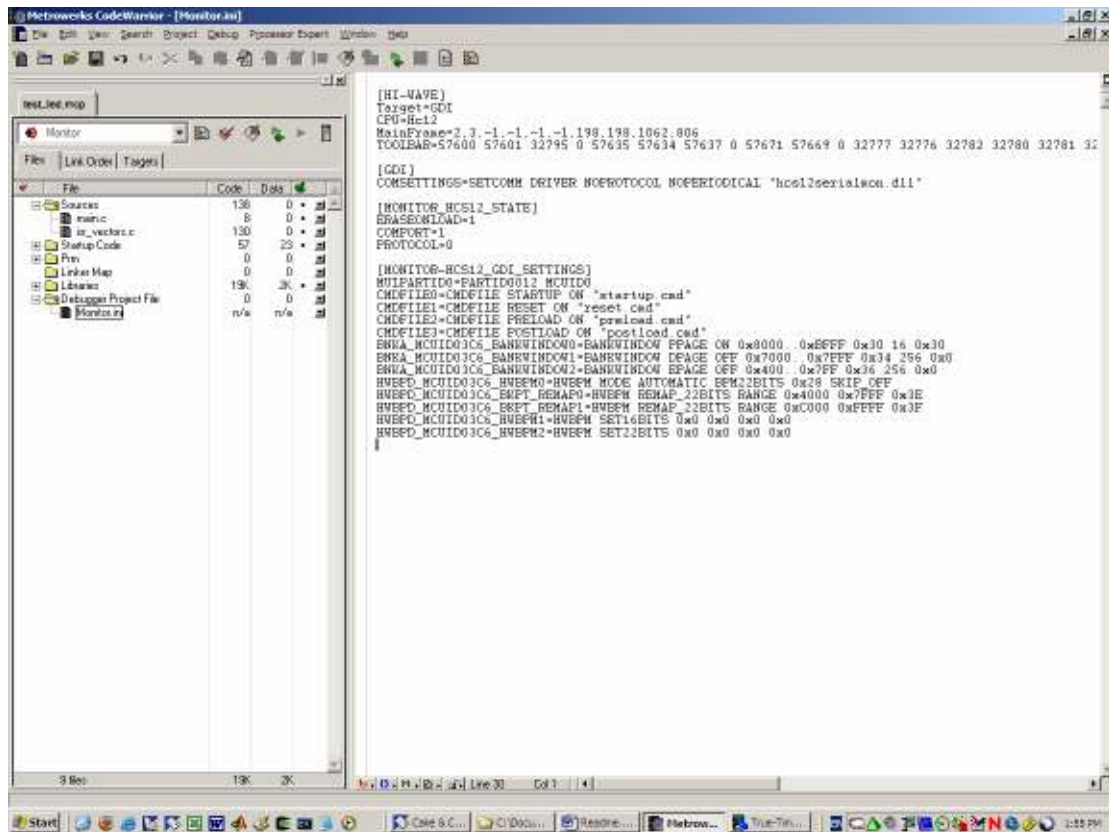


Figure 17 The debugger configuration file *Monitor.ini*

Open the configuration file *Monitor.ini*. You should see 4 categories ([HI-WAVE], [GDI], [MONITOR_HCS12_STATE] and [MONITOR-HCS12_GDI_SETTINGS]):

```
[HI-WAVE]
Target=GDI
CPU=Hc12
MainFrame=2,3,-1,-1,-1,-1,198,198,1062,806
TOOLBAR=57600 57601 32795 0 57635 57634 57637 0 57671 57669 0 32777 32776 32782 32780 32781 32778 0 32806

[GDI]
COMSETTINGS=SETCOMM DRIVER NOPROTOCOL NOPERIODICAL "hcs12serialmon.dll"

[MONITOR_HCS12_STATE]
ERASEONLOAD=1
COMPORT=1
PROTOCOL=0

[MONITOR-HCS12_GDI_SETTINGS]
MULPARTID0=PARTID0012 MCUID0
CMDFILE0=CMDFILE STARTUP ON "startup.cmd"
CMDFILE1=CMDFILE RESET ON "reset.cmd"
CMDFILE2=CMDFILE PRELOAD ON "preload.cmd"
CMDFILE3=CMDFILE POSTLOAD ON "postload.cmd"
BNKA_MCUID03C6_BANKWINDOW0=BANKWINDOW PPAGE ON 0x8000..0xBFFF 0x30 16 0x30
BNKA_MCUID03C6_BANKWINDOW1=BANKWINDOW DPAGE OFF 0x7000..0x7FFF 0x34 256 0x0
BNKA_MCUID03C6_BANKWINDOW2=BANKWINDOW EPAGE OFF 0x400..0x7FF 0x36 256 0x0
HWBPD_MCUID03C6_HWBPM0=HWBPM MODE AUTOMATIC BPM22BITS 0x28 SKIP_OFF
HWBPD_MCUID03C6_BKPT_REMAP0=HWBPM REMAP_22BITS RANGE 0x4000 0x7FFF 0x3E
HWBPD_MCUID03C6_BKPT_REMAP1=HWBPM REMAP_22BITS RANGE 0xC000 0xFFFF 0x3F
HWBPD_MCUID03C6_HWBPM1=HWBPM SET16BITS 0x0 0x0 0x0 0x0
HWBPD_MCUID03C6_HWBPM2=HWBPM SET22BITS 0x0 0x0 0x0 0x0
```

Notice the *command file section* within [MONITOR-HCS12_GDI_STATE]. Modify this section to match the following example:

```
CMDFILE0=CMDFILE STARTUP OFF ".\cmd\sermon_startup.cmd"
CMDFILE1=CMDFILE RESET OFF ".\cmd\sermon_reset.cmd"
CMDFILE2=CMDFILE PRELOAD OFF ".\cmd\sermon_preload.cmd"
CMDFILE3=CMDFILE POSTLOAD ON ".\cmd\sermon_postload.cmd"
```

The first three command files (*startup*, *reset* and *preload*) are switched off, while the fourth entry (*postload*) points to the file '*.\cmd\sermon_postload.cmd*'. A file with this name has been provided in sub-folder 'cmd'. Save and close the debugger initialisation file (*Monitor.ini*).

From the project group *Debugger Command Files* open file *sermon_postload.cmd* (Figure 18). The commands in this file are carried out following the downloading of the user code to the target. The provided example sets a breakpoint at the beginning of function *main* (bs &main) before starting execution (g):

```
// After load the commands written below will be executed
// Show main function at startup

bs &main
g
```

This causes the monitor to go through the instructions of the initialisation routine (*_Startup*) before returning control to the debugger at the beginning of *main*. A similar set of commands could be used in the *reset* command file.

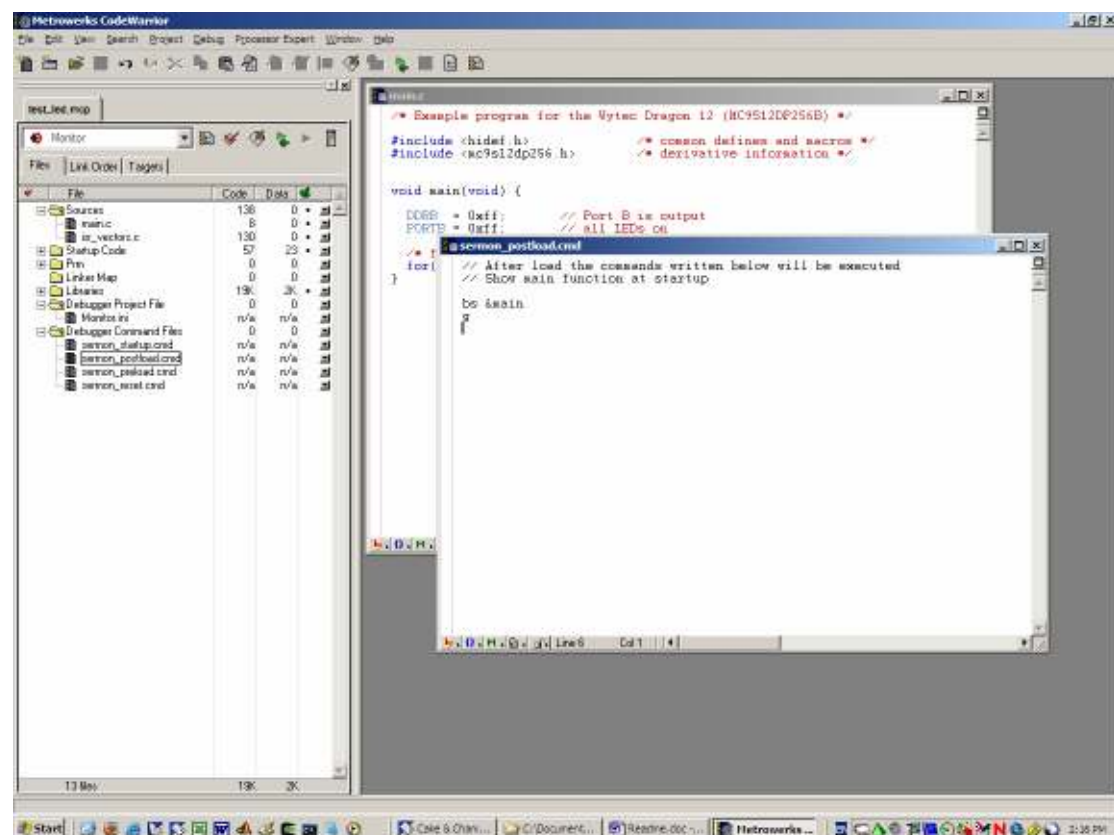


Figure 18 Postload commands

Close the postload command file and re-launch the debugger. Hi-Wave should download the user code and return at the beginning of *main*. A breakpoint has been set on the first line of code in *main* (Figure 19).

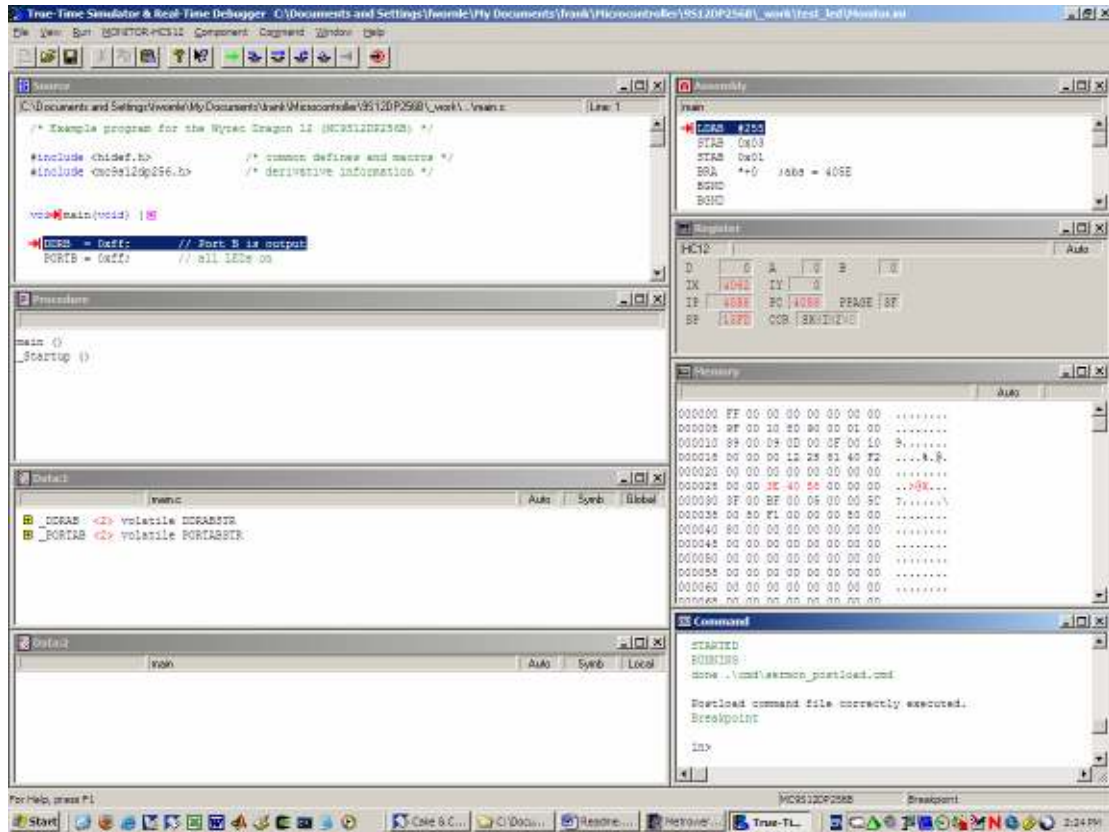


Figure 19 Starting the debugger at the first instruction of *main*

This (hardware) breakpoint can be disabled or deleted using the corresponding entries in menu Debug.