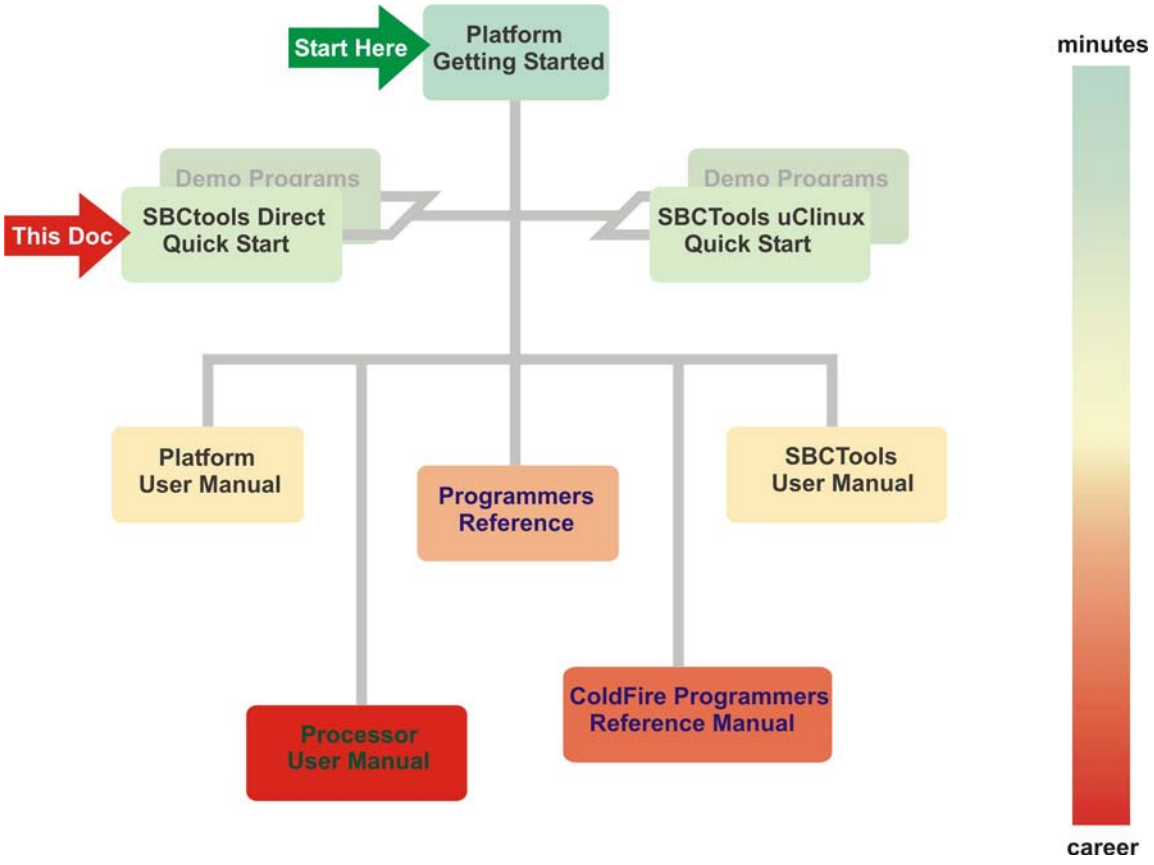
SBCTools Direct
Quick Start



Intec Automation Inc.
2751 Arbutus Road
Victoria, BC V8N5X7
Canada

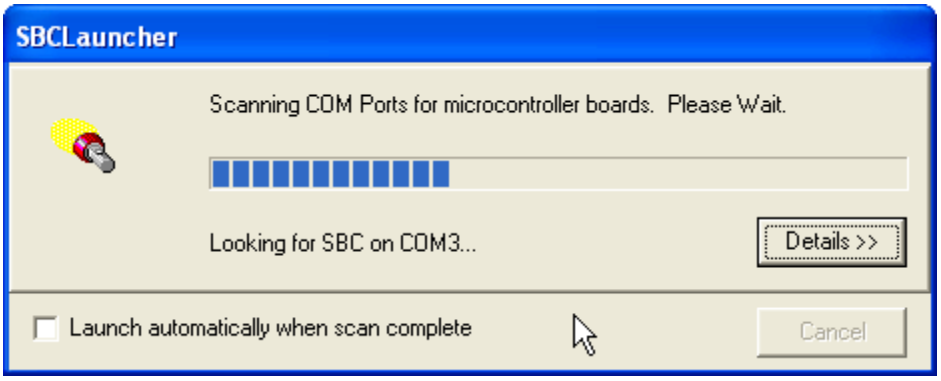
SBCTools Direct Quick Start

Prerqusites

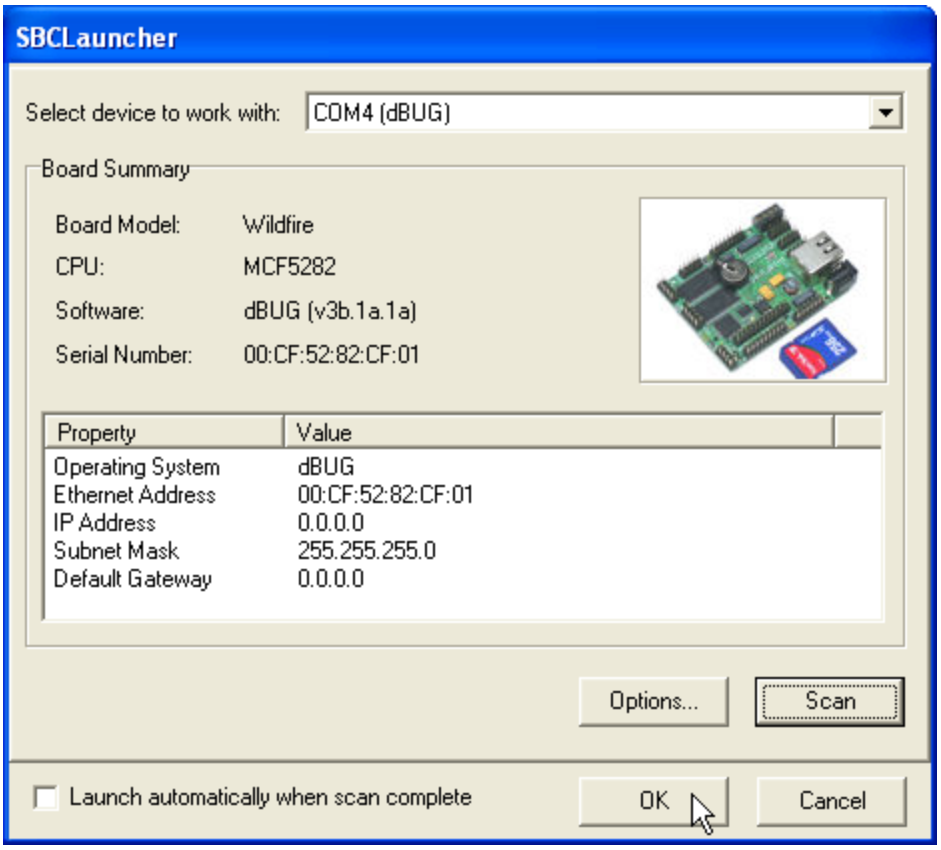
Before starting this quick start for SBCTools the target microcontroller should be connected and powered up as per the Getting Started document. (e.g. WildFireGS.pdf, M5208EVBGS.pdf).

Preparing the Environment

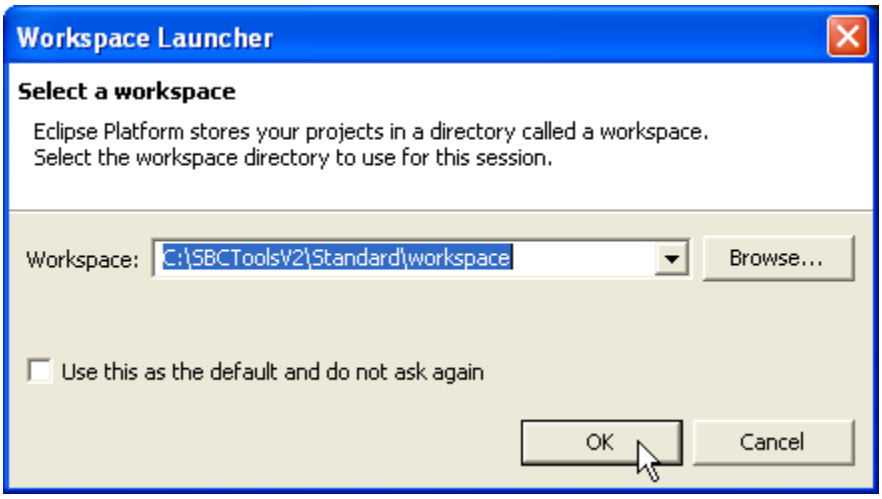
1. Close any applications that may be using PC COM ports connected to the SBC. Connect the PC to the SBC COM1 and COM2 through 2 serial ports on the PC. Power the board with an 8V – 24V **center-positive** power supply.
2. Start **SBCToolsV2** – Eclipse Edition by double clicking on the  icon on the desktop or launching it from the Start Menu.
3. The SBCLauncher will now scan your PC COM Ports searching for a target SBC.



4. When and SBC is found it will be analyzed and information about it parameters and setup will be displayed on a new screen. When you press OK on this window SBCTools will startup.

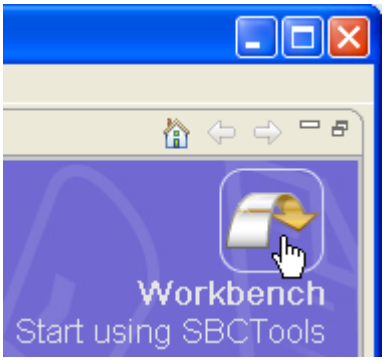

SBCLauncher
Select device to work with:
COM4 (dBUG)
Board Summary
Board Model:
Wildfire
CPU:
MCF5282
Software:
dBUG (v3b.1a.1a)
Serial Number:
00:CF:52:82:CF:01
Property
Value
Operating System
dBUG
Ethernet Address
00:CF:52:82:CF:01
IP Address
0.0.0.0
Subnet Mask
255.255.255.0
Default Gateway
0.0.0.0
Options...
Scan
Launch automatically when scan complete
OK
Cancel

5. The first time **SBCTools** runs, the Workspace launcher appears. In the **Workspace** box, enter "**C:\SBCToolsV2\Standard\workspace**" and check the "...do not ask again" check box to avoid subsequent prompts. The workspace can be changed at any time by selecting **File → Switch Workspace…**



Warning! Loss of Functionality. Spaces in any path name will prevent **SBCTools** from working properly. The workspace directory must have **no spaces** in the path name.

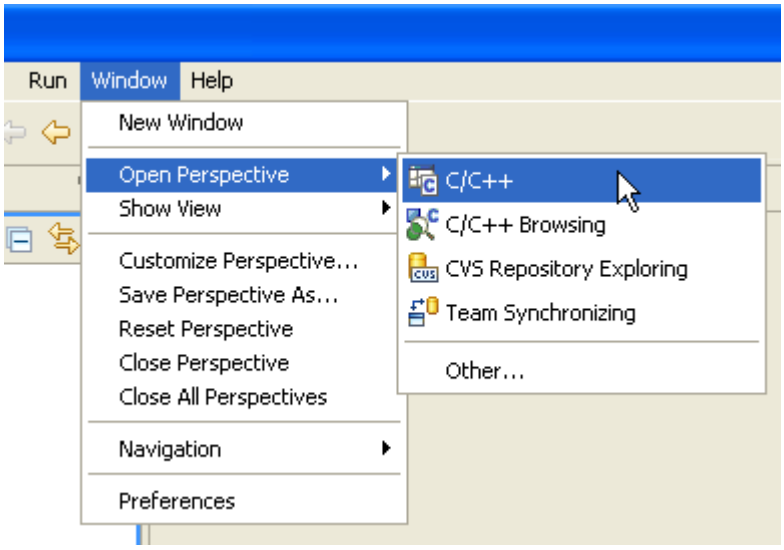
6. **SBCTools**, Eclipse Edition opens in the resource perspective, and displays a welcome screen. This screen has links to useful tutorials and information on using the Eclipse development environment. Click on the arrow in the top right corner to minimize this window. This view can be opened again at any time by clicking **Help → Welcome**.



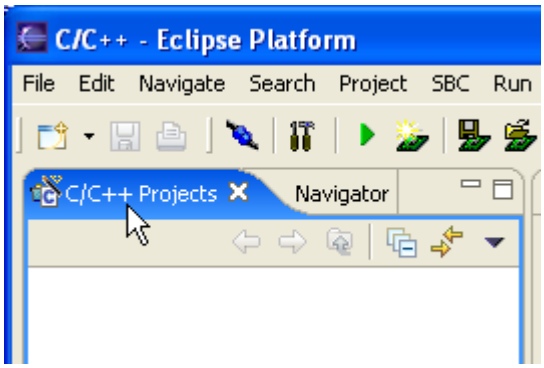
Working with a Project

1. Import a project:

a. Switch to the **C/C++ Perspective**: Click on **Window → Open Perspective → C/C++**.

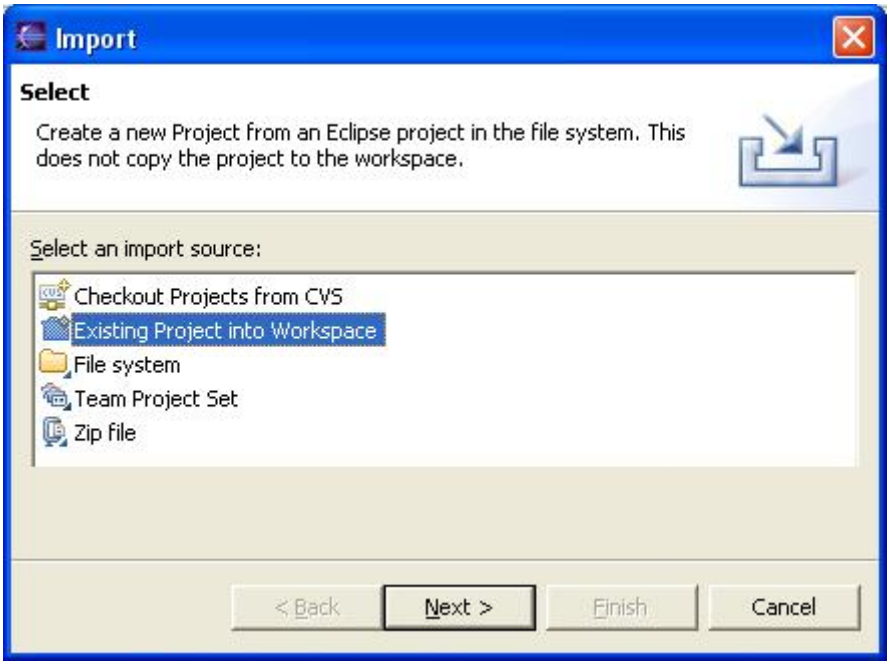


b. Click on the **C/C++ Projects** tab to switch to the **C/C++ Projects** view.

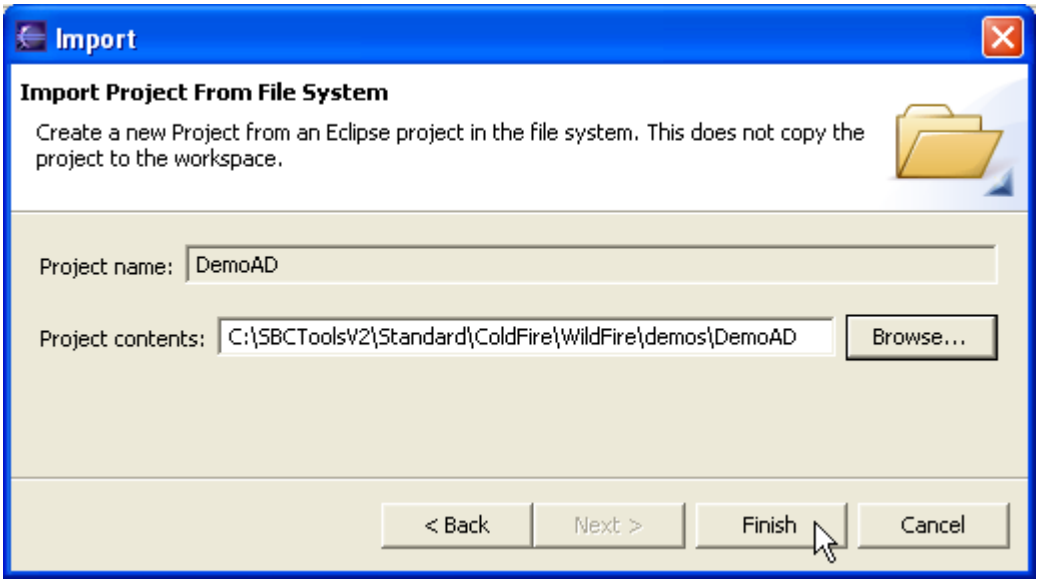


c. Select **File → Import…** to open the Import dialog.

i. Select "Existing Project into Workspace" as the source and click "Next". This will create a new project in the current workspace that references the original files (which can be located anywhere). It does not copy any files into the workspace.

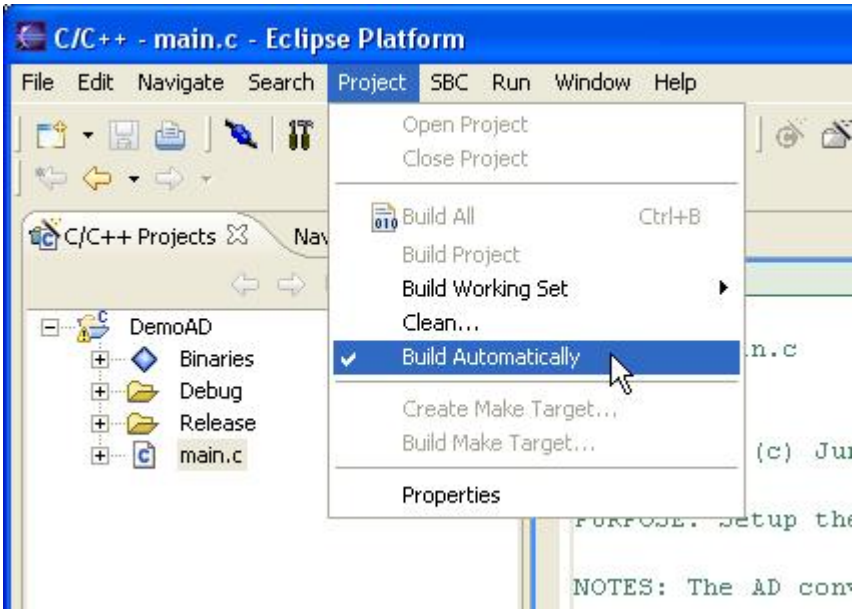


ii. Browse to "**C:\SBCTools\ColdFire\[Board]\demos\DemoAD**" and click OK. If DemoAD doesn't exist for your board then substitute any demo program in its place.

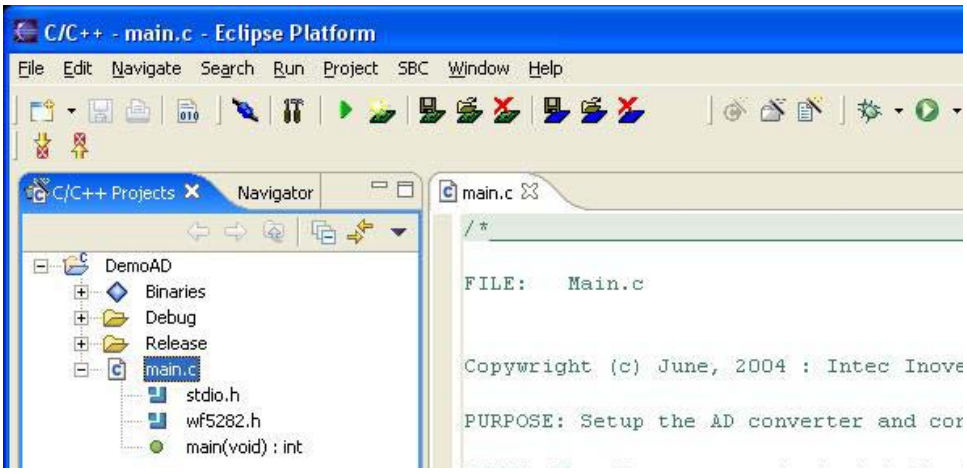


iii. Click Finish on the Import dialog. This will import the project into the current workspace and automatically build it. "Importing" sets up a folder in the Workspace, along with files that enable Eclipse to keep track of the project.

iv. Turn off the Build Automatically option by unchecking this option under **Project → Build Automatically**.

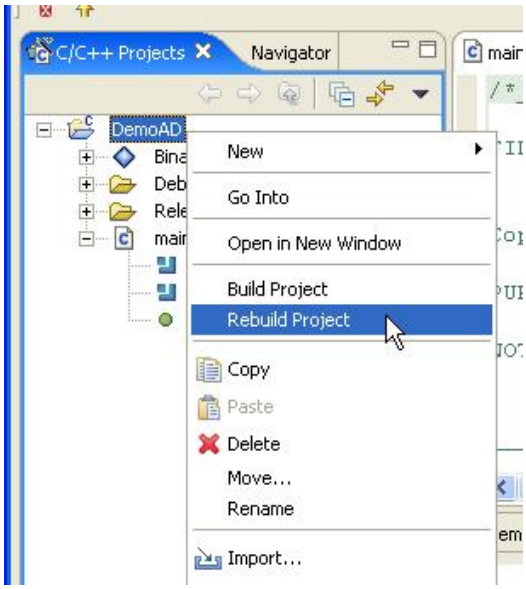


1. View a File: Expand the DemoAD project in the **C/C++ Projects** view and double-click on main.c. The file will appear in the editor area. The file can be edited and saved from here. To save a file, click **File → Save** in the Eclipse menu bar.

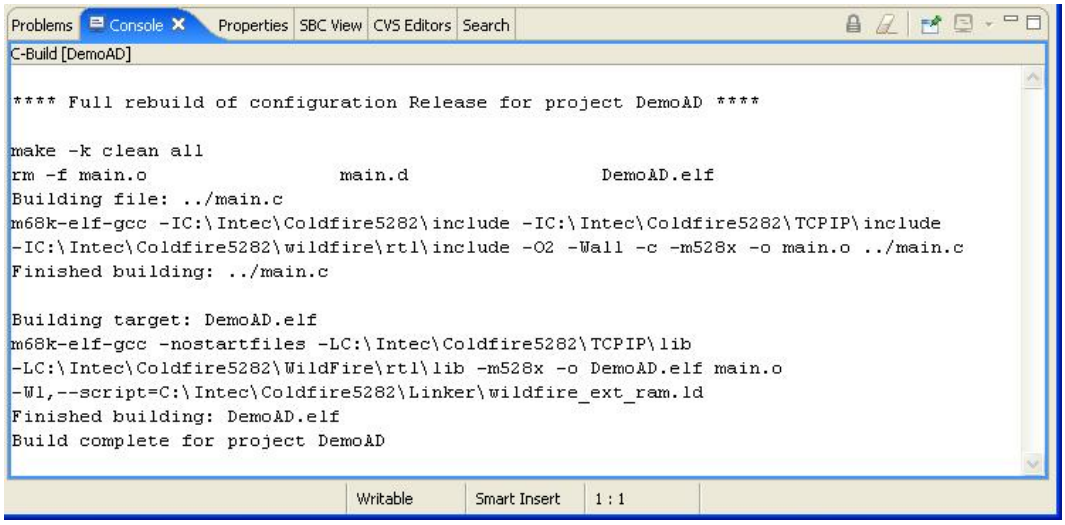


2. Compile a Project:

a. Right-click on the DemoAD project in the **C/C++ Projects** View and select **Rebuild Project** from the popup menu.



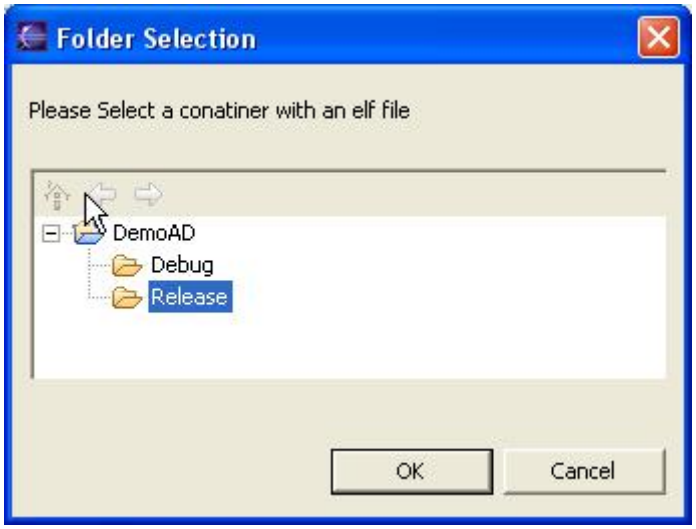
As the project is built, all compiler and linker output is written to the **Console** view at the bottom of the screen.



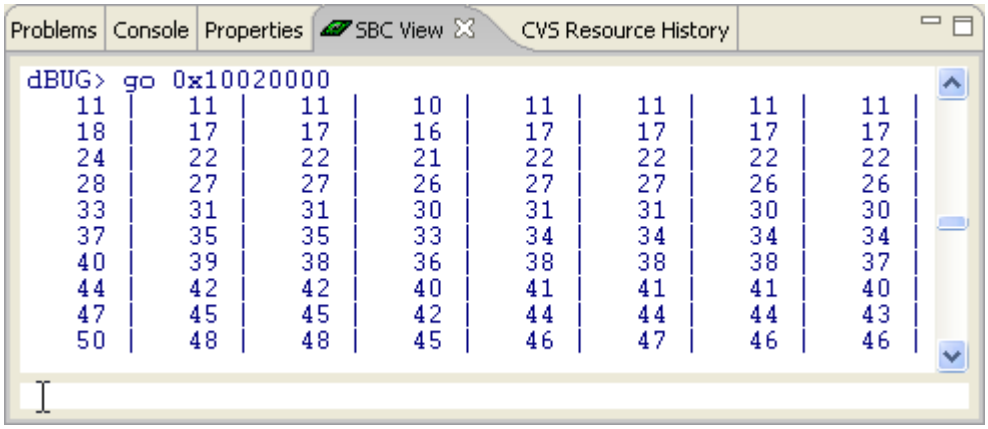
3. Download to target

a. Select **SBC → Download/Run**

b. When the Folder Selection dialog appears, select "DemoAD/Release" and press OK.



c. The program downloads on the SBC, which may take a few moments. Once it has completed, the demo will run automatically, displaying its output in the SBC View.



d. To terminate execution of demoAD, short the reset pads beside CN4, on the CN3 side.

4. Debugging a program

a. Compiling for debugging:

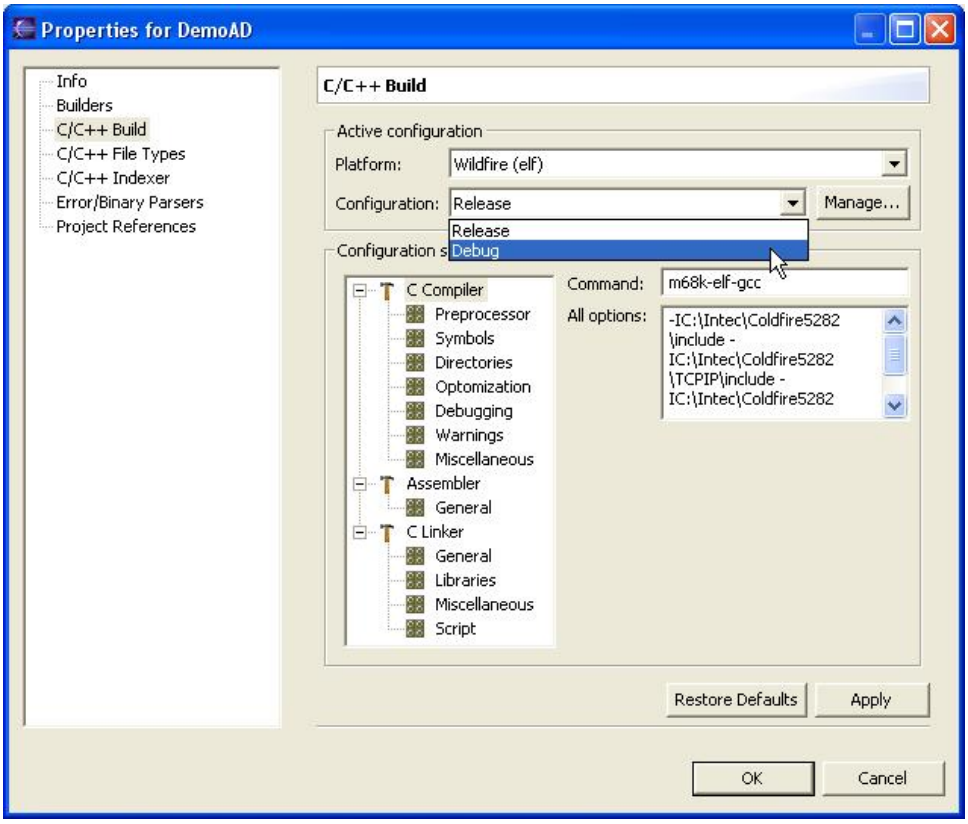
Before the program can be debugged, it must be compiled for debugging. To do this:

i. Right-click on DemoAD project in the **C/C++ Project** view and select **Properties**

ii. Click on **C/C++ Build**

iii. Change the Configuration to Debug using the drop down menu.

iv. Click OK



v. Ensure all files that have been modified are saved. The compiler only recognizes saved files. To save a file, click **File → Save** in the Eclipse menu bar.

vi. Right-click on the "DemoAD" project in the **C/C++ Project** view again, and select **Rebuild Project**. This forces a rebuild whether the compiler has recognized changes to the code or not.

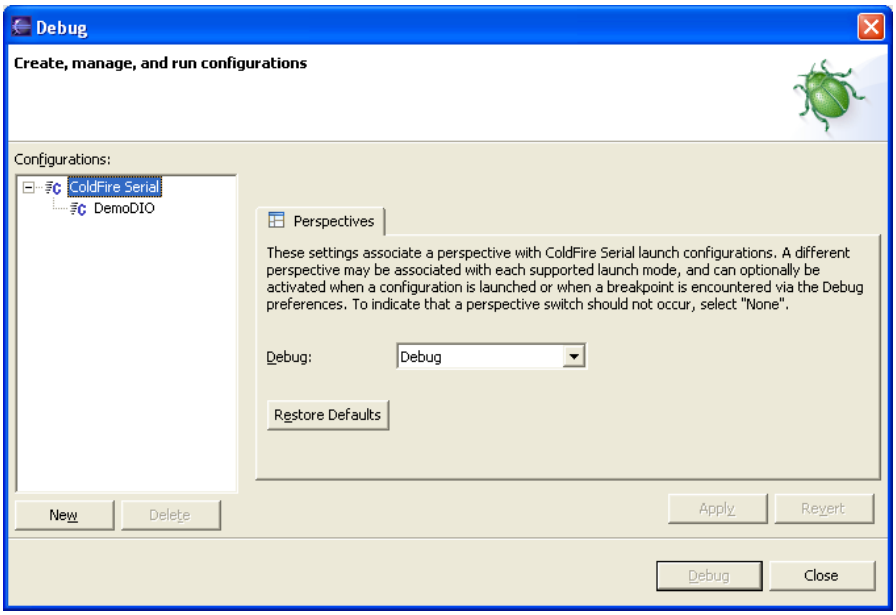
b. Creating a Debug Launch Configuration:

A debug launch configuration must be created the first time a program is debugged. To create a debug configuration:

i. In **C/C++ Projects** view, select the DemoAD project.

ii. Click **Run → Debug…**.

iii. In the Debug dialog box, select **ColdFire Serial** debug configuration type from the **Configurations** list.

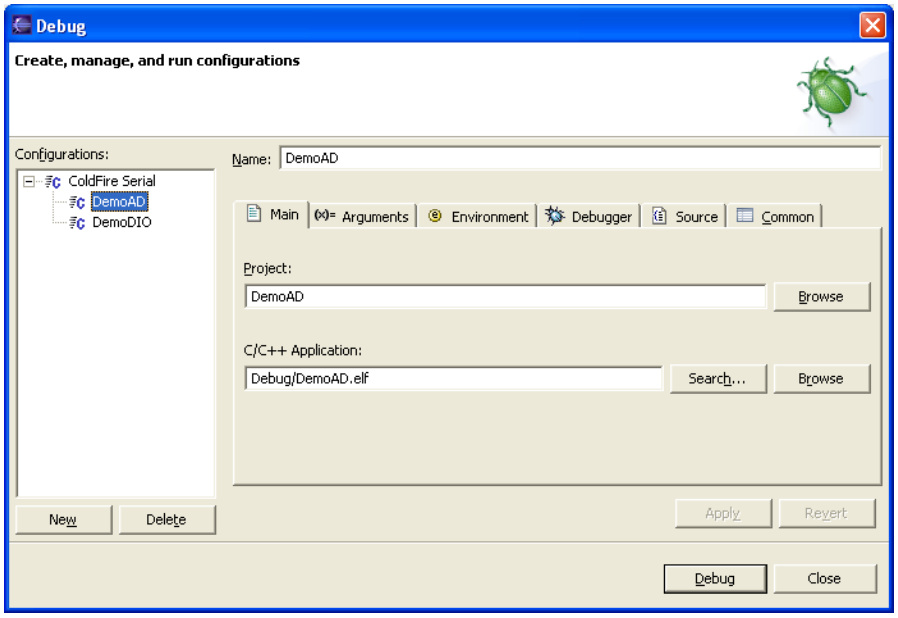


iv. Click **New**.

v. In the **Name** box, type “DemoAD”, a descriptive name for this debug configuration.

vi. In the **Project** box, type or browse for “DemoAD”.

vii. In the **C/C++ Application** box, type or browse for “Debug\DemoAD.elf”. The ‘.elf’ file only appears after the project has been rebuilt.

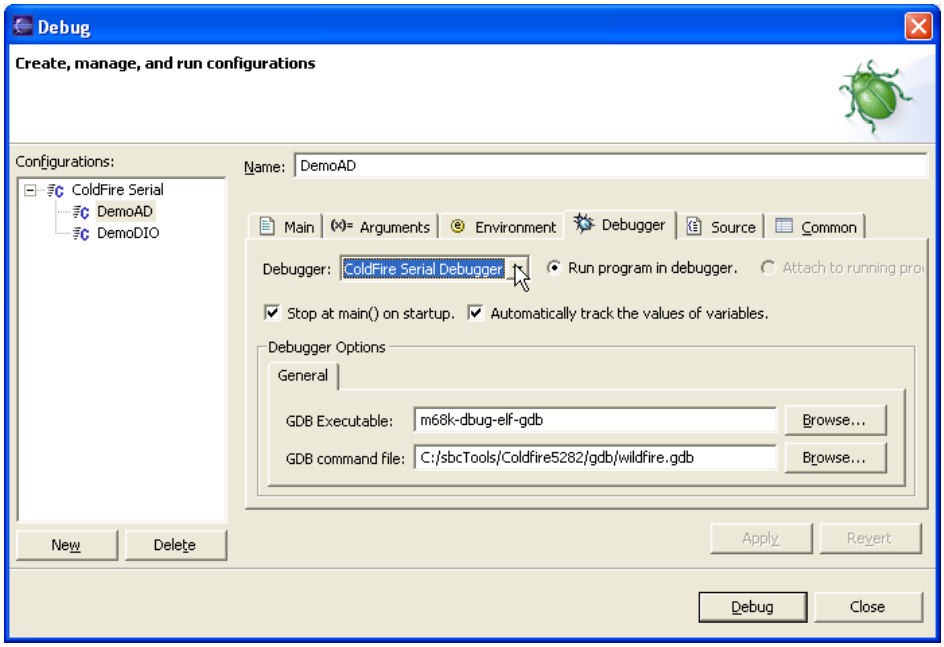


viii. Click the Debugger tab.

ix. Select “ColdFire Serial Debugger” in the **Debugger Box.** This will automatically fill in the Debugger Options, below.

x. Select **Run program in debugger**.

xi. Select the **Stop at main() on startup** checkbox.



xii. Click **Apply**. This saves the debug configuration under the name “DemoAD”. It will appear under the **Debug Toolbar** (see below) for quick access. “DemoAD” is now ready to be debugged.

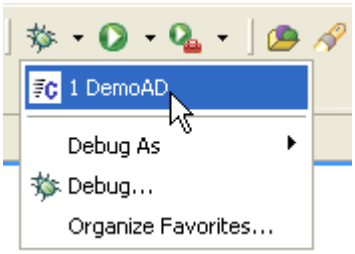
xiii. Click **Apply**. This saves the debug configuration under the name “DemoAD”. It will appear under the **Debug Toolbar** (see below) for quick access. “DemoAD” is now ready to be debugged.

xiv. Click **Close**.

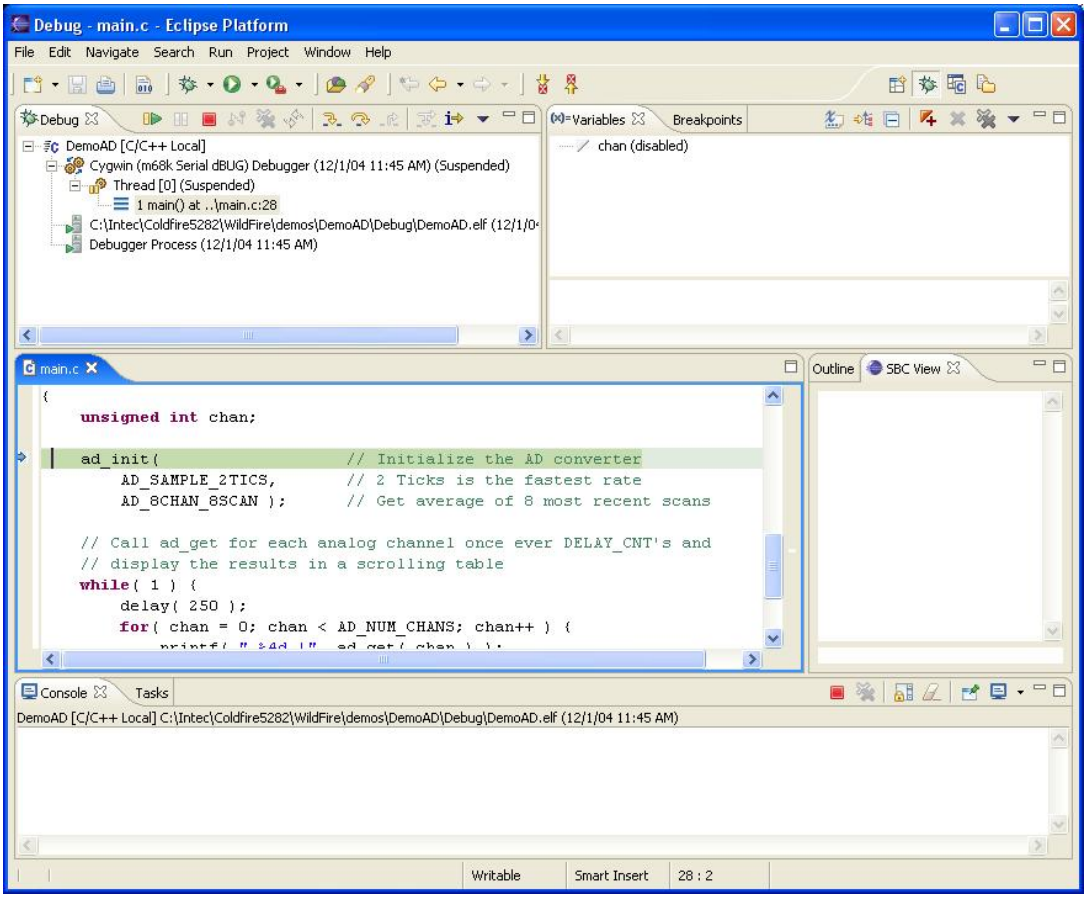
c. Debugging a program:

The program is ready to be debugged; to do this:

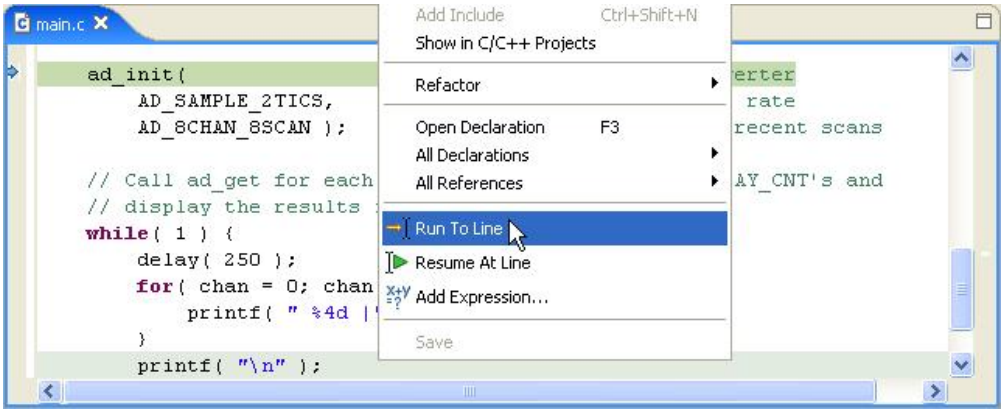
i. Click the little black triangle beside the icon on the **Debug Toolbar** and select **DemoAD**, or click on **Run → Debug…** to get the Debug Configuration window shown above. Then, select **DemoAD** and click on the **Debug** button.



ii. The debug perspective is opened after prompting to switch perspectives. The **C/C++ Editor** window is repositioned in the debug perspective. The debugger starts execution of the program and stops it when main is reached. A blue arrow indicates the next instruction pointer.

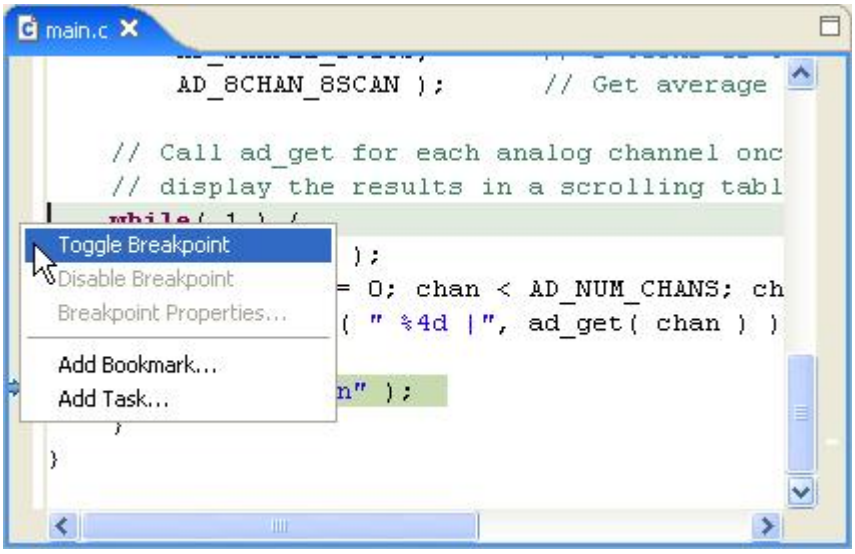


vii. Put the cursor on a line in the program, right-click and select **Run To Line**.

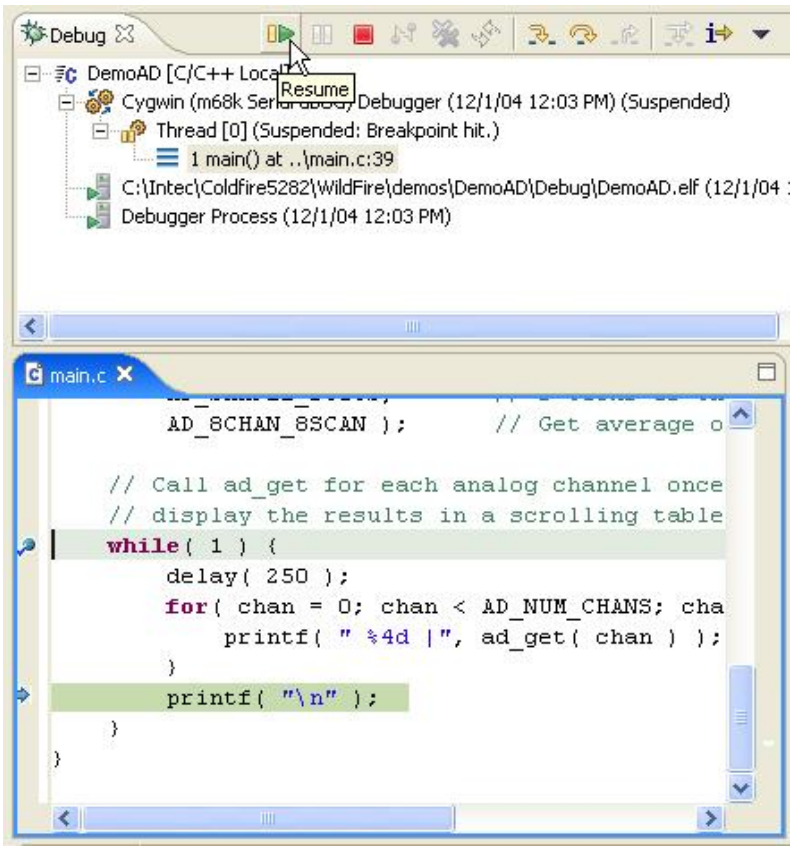


viii. Control returns to the debugger when the program stops at the highlighted line.

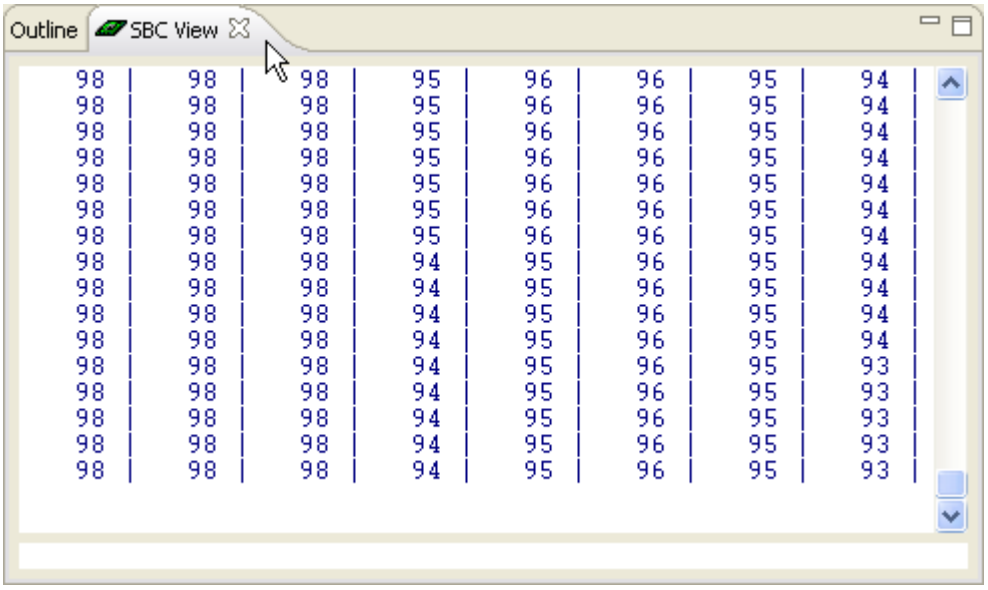
ix. Add a breakpoint at this line by selecting the line and right-clicking and selecting **Toggle Breakpoint** in the gray area left of the margin.



x. Run to the breakpoint by clicking the **Resume** arrow on the **Toolbar**.



As the program steps through each loop, the printf display in the SBC View.



Touching the AtoD pins with one finger, while touching a Gnd or Vcc pin with another will alter the AtoD readings as your body acts as a very weak pull-up or pull-down resistor.

5. Conclusion

This excercie has demonstrated that the **SBCTools** installation has been successful and the SBC is correctly connected to the PC for serial debugging. Refer to the **SBCTools** section of the user manual and the Eclipse documentation and help guides for more advanced information on using **SBCTools** and Eclipse.

Appendix A

Annotated Demo Programs

Demo Programs

Demo programs provide good examples of using the various modules available in the Run Time Library (RTL). Demo programs are available in the install directory of SBCTools (e.g. C:\SBCToolsV2\[Standard][uClinux]\ColdFire\[Board]\Demos). They can be imported into SBCTools by using the *Import* option on the *File* menu (Import Existing Project).

dBUG

DemoAD *WildFire: ☑*

Setup the AD converter to the fastest conversion rate possible (2 AD ticks) with the mode set to scan all 8 channels continuously. Display the value of each channel to the console with a small delay. The value of the channel is the average of the 8 most recent scans in terms of AD counts which range from 0 to 1023 and correspond a an input voltage between 0 and 5 volts respectively.

DemoDIO *WildFire: ☑ m5208evb: ☑*

Toggle All GPIO pins available on the board. Output to the console the current state of the pins.

DemoDNS *WildFire: ☑ m5208evb: ☑*

Resolve a hostname in order to obtain the IP address. The TCP/IP Library sockets require an IP address to be specified, in the case the client program on the SBC does not know the IP address but only the hostname, the IP address must be obtained via the get_host_by_name API. This demo will attempt to use a dhcp server for it's configuration (if there is no dhcp server available it will timeout in 20 seconds and resort to the IP parameters on the board). After get_host_by_name is called the, mainloop continually calls tick_tcp which keeps the TCP/IP subsystem heartbeat going and eventually results in a call to the event listener function for dns with either an IP address or a timeout error.

DemoDT *WildFire: ☑ m5208evb: ☑*

Use the DMA_TIMER0 to generate a 66% on PWM signal using interrupts, also use the same timer to receive input capture events. A scope should be used to verify the output signal, and a jumper can be placed between the two pins to verify the input capture functionality.

DemoGPT *WildFire: ☑*

Use one GPT pin as an output compare and one as an input capture. run the output compare completely in hardware without interrupts and the input capture with interrupts. A scope should be used to verify the output signal, and a jumper can be placed between the two pins to verify the input capture functionality.

DemoHTTP *WildFire: ☑ m5208evb: ☑*

A very simple HTTP server with two web pages stored as static arrays. Listens for remote connections and then executes the HTTP protocol and eventually returns the index HTML page or an error page. Tick_tcp and https_run are called continuously from the mainloop to keep the TCP/IP subsystem and the http server running. http.c contains the implementation of the HTTP Server and can be reused in any application.

DemoIRQ *WildFire: ☑ m5208evb: ☑*

Setup IRQ pin 1 to fire an interrupt when a falling edge is detected, increment a count whenever the interrupt fires and then display the count to the console when it changes. This demonstrates how an interrupt pin could be used as a high frequency counter.

DemoLCDKPD *WildFire: ☑*

Initialize the LCD and Keypad, write "Hello World" to Line one and Line two, and then scan the Keypad and display the key pressed on line two of the LCD. This demo requires that the LCD Display and Keypad be attached to header CN1 On the WildFire board.

DemoPIT *WildFire: ☑ m5208evb: ☑*

Initialize Periodic Interrupt Timer (PIT) channel 0 to interrupt at an approx. two second frequency. Increment a count within the handler and display the count from the mainloop if it changes. Notice interrupt flag must be cleared from within the interrupt handler.

DemoRTC *WildFire: ☑*

Demonstrate the functionality of the battery backed Real Time Clock (RTC) and board hibernation feature. Get the current time from the clock and then set the alarm for ten minutes hence, which will turn off the board and wake it up in ten minutes.

DemoSD *WildFire: ☑*

Initialize the Secure Digital card, write to it, verify it, read from it, and then compare the read data against the initial data. The read/write/verify cycle is done 12 times in a row with increasing offsets on the SD card. The memory compare at the end is simply a sanity check and ensures that the sd_verify function is working correctly.

DemoSMTP *WildFire: ☑ m5208evb: ☑*

Send an email from the WildFire board using an Simple Mail Transfer Protocol (SMTP) server. The smtp_client module implements the SMTP client protocol and can be reused in any application. The smtp_callbacks maodule is where a programmer can hook into the protocol to fill the email message (e.g. to address, from address, subject and of course – message content). Simply changing the global variables at the top of the smtp_callbacks.c file will customize the email settings, the programmer should also ensure that the correct SMTP Server IP address is specified in the main.c file.

DemoUART *WildFire: ☑ m5208evb: ☑*

This program will continually output "Hello World" to COM2 and COM3 and if a character is received on the port it will echo the character back. We do not use COM1 in this demo since COM1 is used as the console port. COM3 is DTE and will require a null-modem cable to see text in a terminal program on the PC. Use Hypertrminal to see the output from these com ports.

DemoWDT *WildFire: ☑ m5208evb: ☑*

Demonstrate the use of the watchdog timer on the mcf5282. The watchdog timer will reset the board if not serviced continually from the executing process. The dBUG monitor has a setting which will enable the watchdog timer and feed it until it transfers control to another program via the "go" command. Use "show" and "set watchdog <on/off>" to turn the watchdog on or off (SBC->IP/Boot Parameters) in the SBCTools IDE. The program will simply continually feed the watchdog until a key is pressed on the console and then the watchdog will timeout.

DemoXFLASH *WildFire: ☑*

Demonstrate the use of the xflash functions for erasing, writing, and reading from external serial flash. The external flash must be erased before it is written to and erasing must be done by a minimum of 1 sector. Therefore this demo program does not ensure that data above the structure to the sector boundary is preserved.

Demo SMAC Host *m5208evb: ☑*

Demonstrate the use of the Zigbee capable transceiver. Waits to receive a message from a SMAC remote. When a message is received, it acknowledges the remote and prints out the received data. Depends on having a SMAC Remote, either a SARD board programmed with the data remote program, or another M5208EVB programmed with the SMAC_Remote program.

Demo SMAC Remote *m5208evb: ☑*

Demonstrate the use of the Zigbee capable transceiver. Sends packets to SMAC Host each time the Abort/IRQ7 button is pressed. Depends on having a SMAC host, either a SARD board programmed with the data host program, or another M5208EVB programmed with the SMAC_HOST program.

uClinux

uDemoAD *WildFire: ☑*

Setup the AD converter to the fastest conversion rate possible (2 AD ticks) with the mode set to scan all 8 channels continuously. Display the value of each channel to the console with a small delay. The value of the channel is the average of the 8 most recent scans in terms of AD counts which range from 0 to 1023 and correspond a an input voltage between 0 and 5 volts respectively. Also log the same data to a file in the current directory.

uDemoEmail *WildFire: ☑ m5208evb: ☑*

Similar to uDemoAD example, except send the results as an email. This demo requires a valid outgoing email server which can be replaced at the top of the main source file.

uDemoDIOFile *WildFire: ☑ m5208evb: ☑*

Initialize all digital IO pins on the WildFire as digital inputs and then query the entire set. Save results to a file in the current directory.

uDemoGPT *WildFire: ☑*

Setup a single GPT pin as an output compare to run entirely in hardware with an oscillating 75% and 25% duty cycle. Connect a scope to see the signal.

uDemoHibernation *WildFire: ☑*

A command line application which can be used to either simply display the current state of the Real Time Clock (RTC) or to set the alarm and shutdown power.

uDemoSerial *WildFire: ☑ m5208evb: ☑*

This demo program shows the use of the serial ports under uClinux. Open and configure the serial ports using the "ttyS" device driver. This program echoes all keys typed into COM2 or COM3 as their ASCII hex representation.

uDemoWebCGI *WildFire: ☑ m5208evb: ☑*

A fully functional dynamic website which uses the "Boa" web server and the Common Gateway Interface (CGI). Each webpage is generated from a template and uses CSS to define the look and feel. This demo also includes an interactive page which displays the state of digital IO header and allows each pin to be individually controlled via a push button.