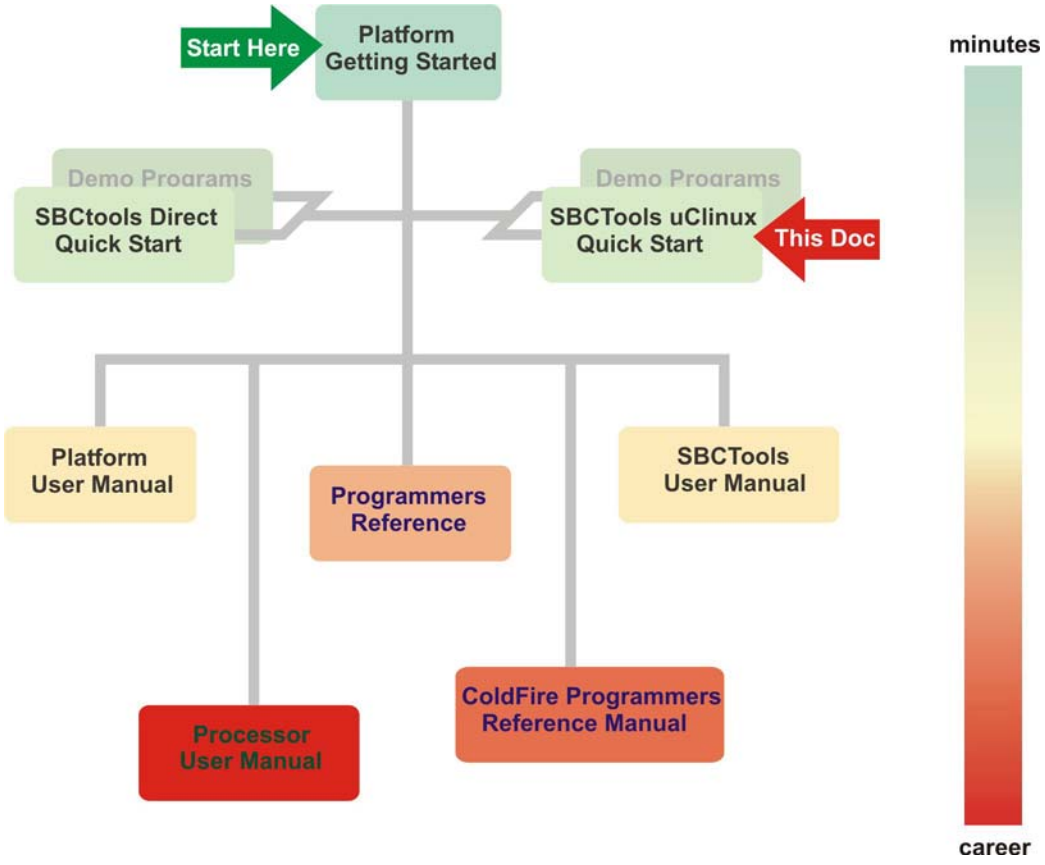
SBCTools uClinux Quick Start



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

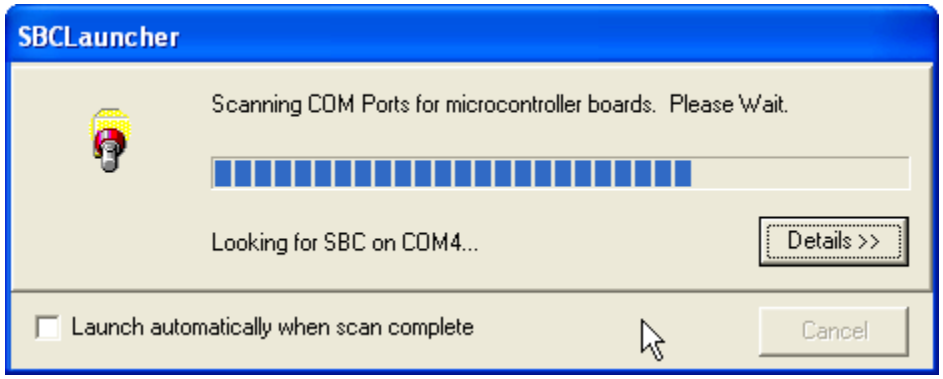
SBCTools uClinux Quick Start

Prerequisites

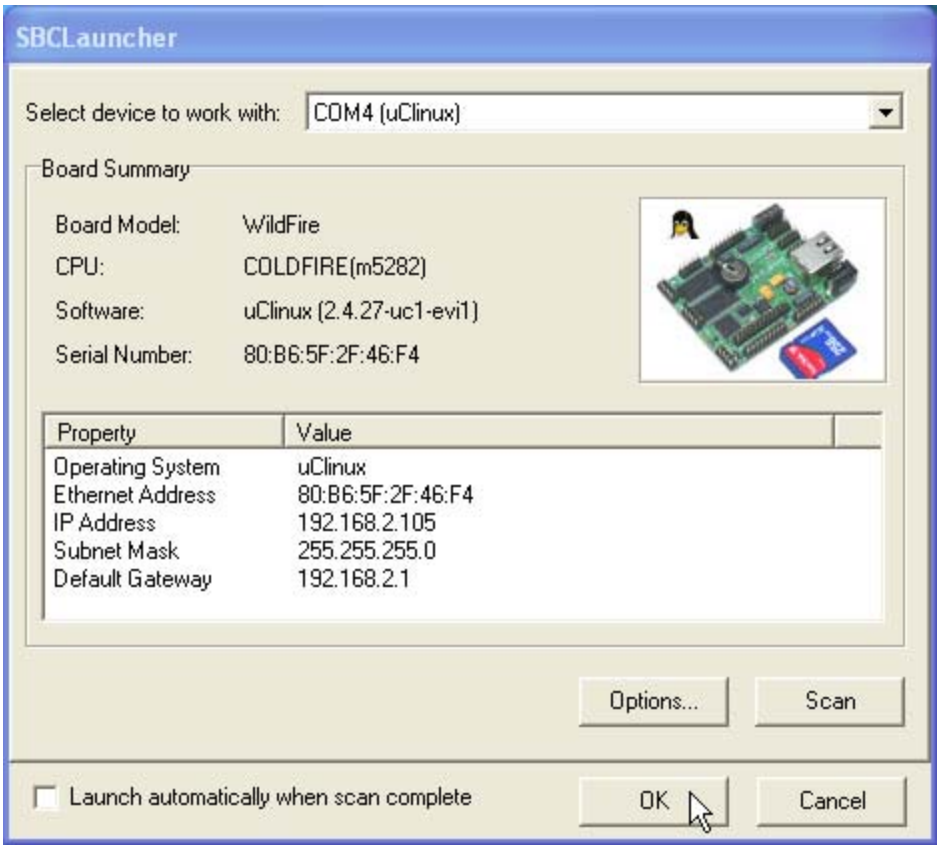
Before starting this quick start for SBCTools the target microcontroller should be connected and powered up as per the Getting Started document. (e.g. WildFireGS.pdf).

Preparing the environment

1. Start **SBCTools** – by double clicking on the  icon on the desktop or launching it from the Start Menu.
2. Immediately the SBCLauncher utility will scan your PC for a connected SBC.

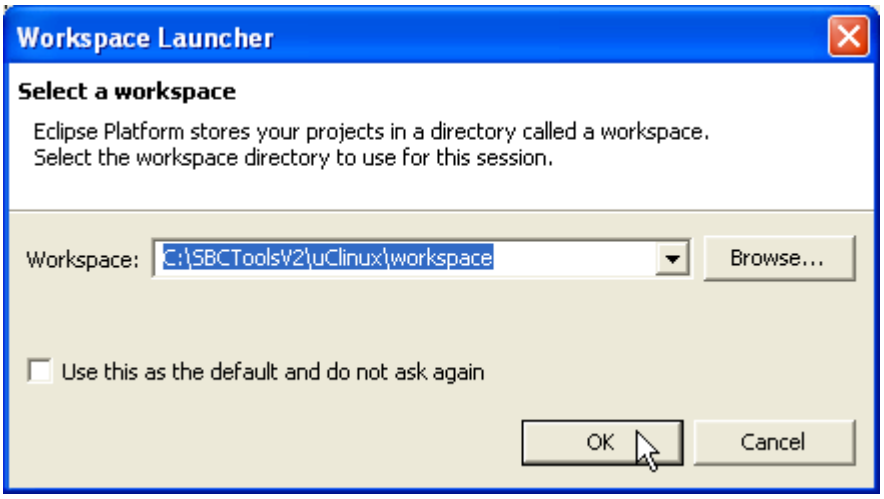


3. When the scan is completed you will see the following screen with information about the board that was detected.



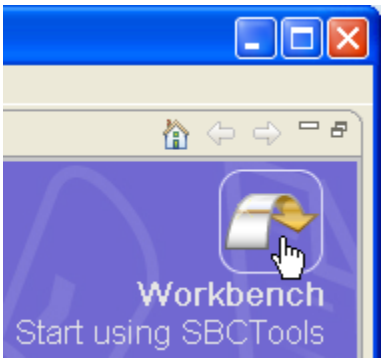
Attention! If you do not see this screen then either the board is not correctly connected, or uClinux has not booted on your board. Please see Preparing the WildFire for uClinux if you are using a WildFire.

4. When you press OK SBCTools for uClinux will be launched. You will now be prompted with a workspace selection dialog. The workspace directory is where SBCTools will keep all project information. It is highly recommended to keep the default **C:\SBCTools\uClinux\Workspace**. Select OK to continue.

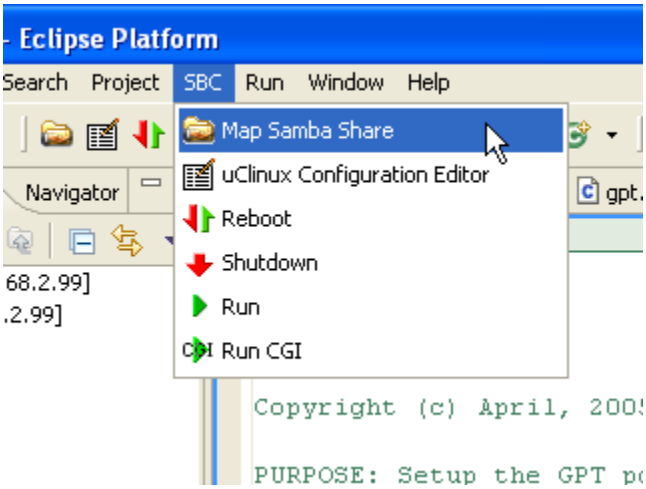


Warning! Loss of Functionality. Spaces in any path name will prevent **SBCTools** from working properly. The workspace directory must have **no spaces** in the path name.

5. SBCTools opens 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 switch to the C/C++ Perspective. This view can be opened again at any time by clicking Help → Welcome.

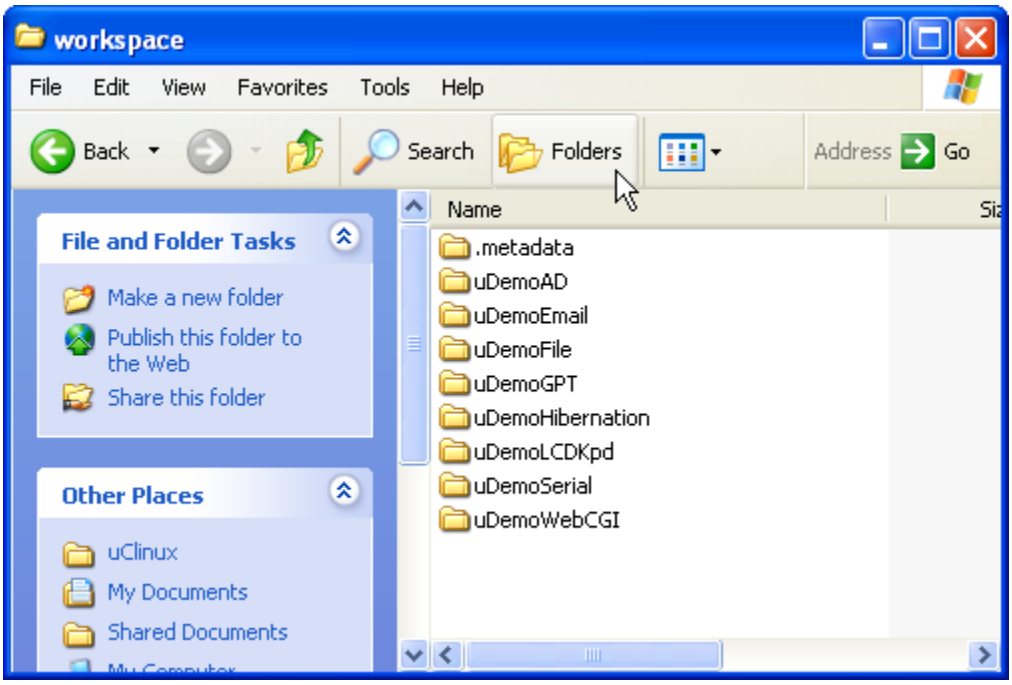


6. Before working with any projects you will first want to map your workspace as a shared folder so that you can run and transparently transfer files between the SBC and uClinux. You can share your workspace by going to SBC->Map Samba Share.

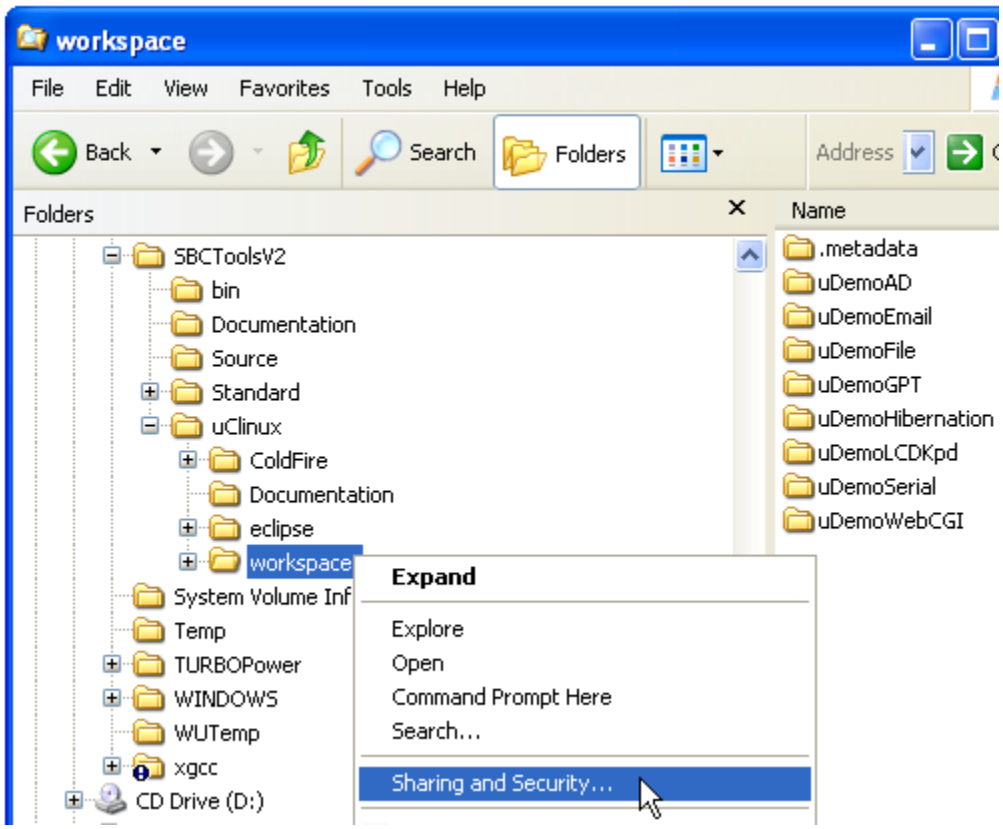


7. You will now be prompted to share your workspace as “workspace”. This will only happen this one time. After this, SBCTools will remember.

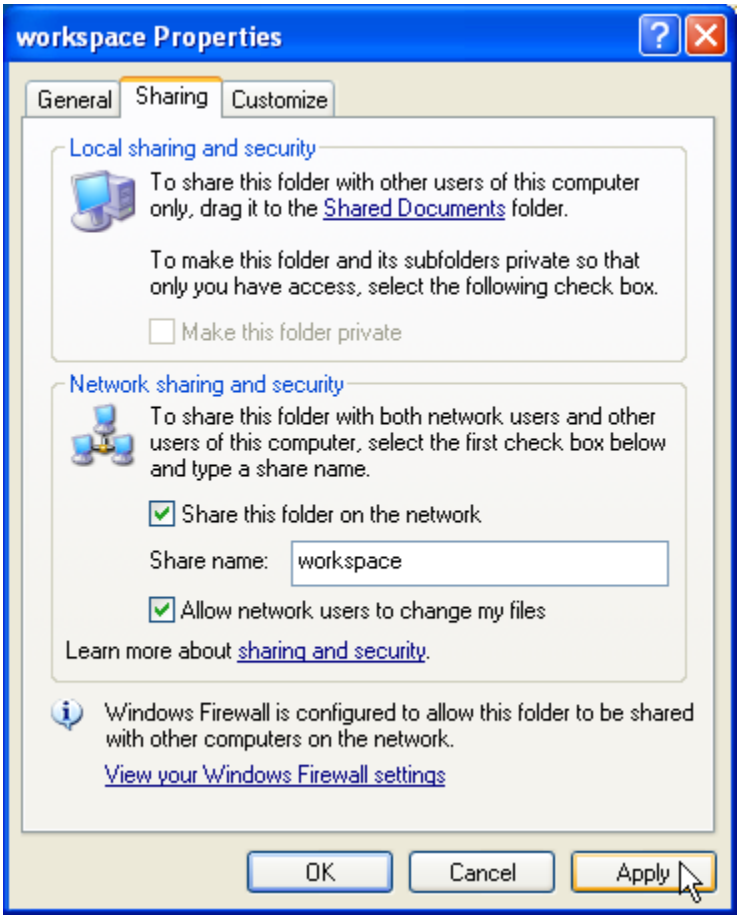
8. When you press OK windows explorer will open with the folder selected which must be shared. If you are not already in the "Folders View" then switch to it by selecting the "Folders" button. Otherwise skip to step 9.



9. You can share this folder by right-clicking on it and selecting "Sharing and Security…" or by selecting "Properties" and then clicking on the "sharing" Tab.



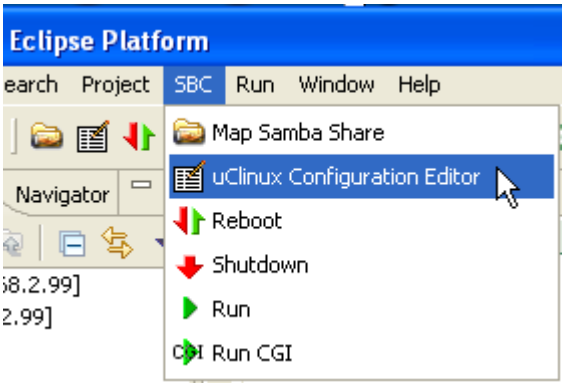
10. From the sharing properties click “Share this folder on the network” and “Allow network users to change my files” and set the Share name to **workspace.**



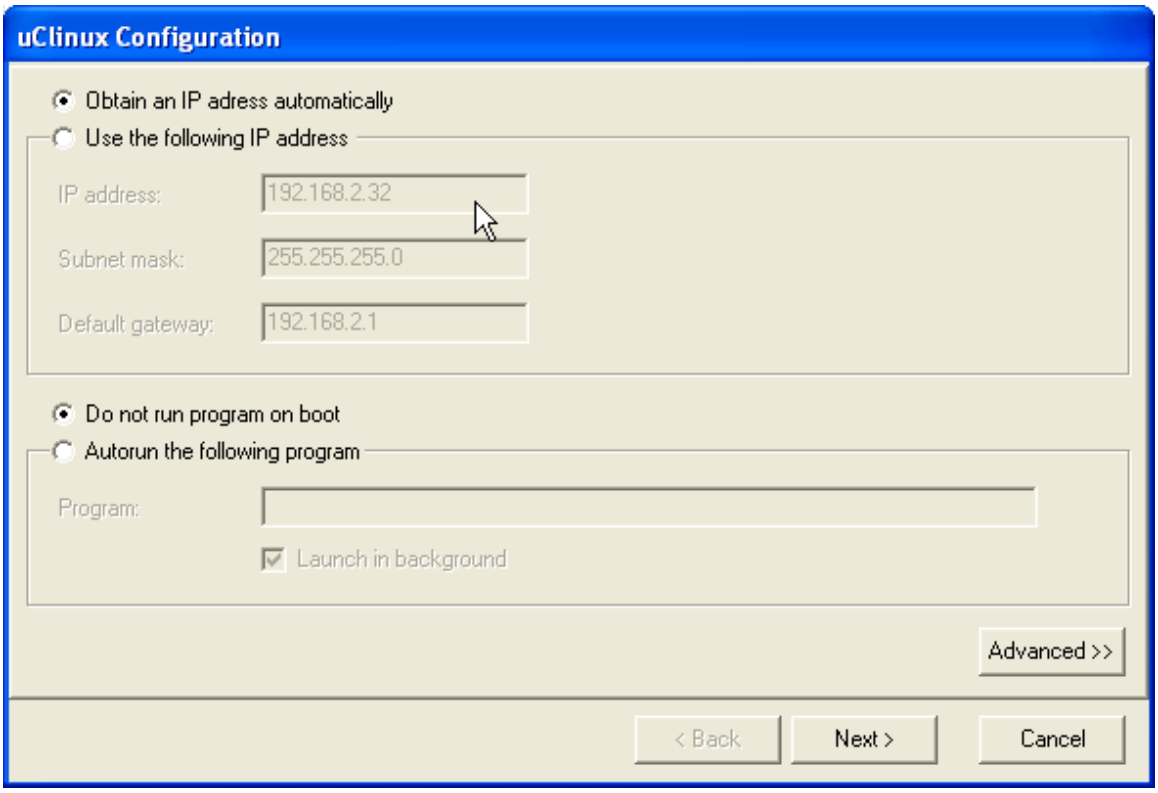
Attention! If you do not see a screen like this then windows file sharing is most likely not enabled on your computer. See Windows File Sharing

11. Now press OK, close the windows explorer window and return to the SBCTools environment.

12. Now the network parameters need to be verified and setup before proceeding. To view/edit the network properties for uClinux go to SBC->uClinux Configuration Editor.



13. This action will probe your board for settings and display them to you in a new window. This window displays the current IP address and other network settings, including whether your board is currently using DHCP to obtain an IP address.

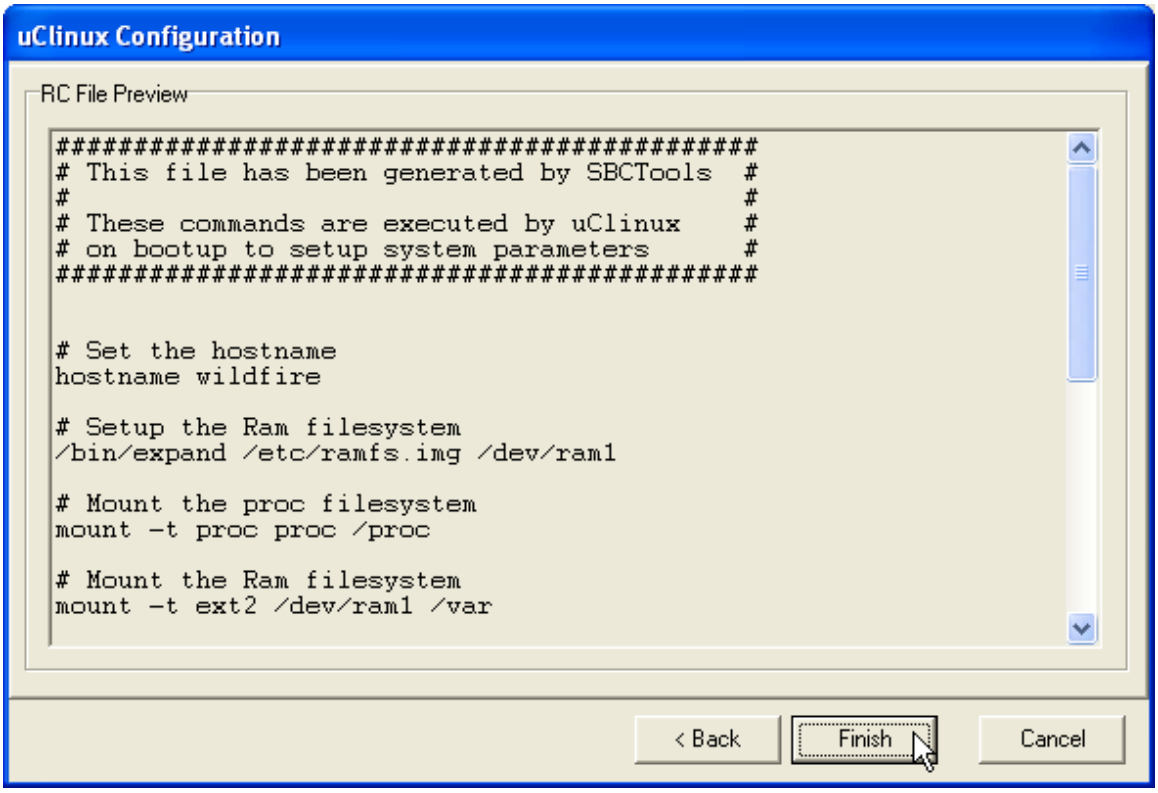


Attention! If your **IP address** is blank then your network has not provided you with an IP address and you will need to set the IP address manually. uClinux Configuration Editor

14. Now Select "Next >" to proceed (accepting the default settings[1]) and view a preview of the uClinux **rc**[2] file. Now press "Finish" to close the configuration editor.

[1] These settings can be changed at a later time. The current concern is getting your WildFire to participate on the network.

[2] The rc file in uClinux is similar to that of autoexec.bat in windows or DOS. The commands in this file will get executed automatically when uClinux boots up.



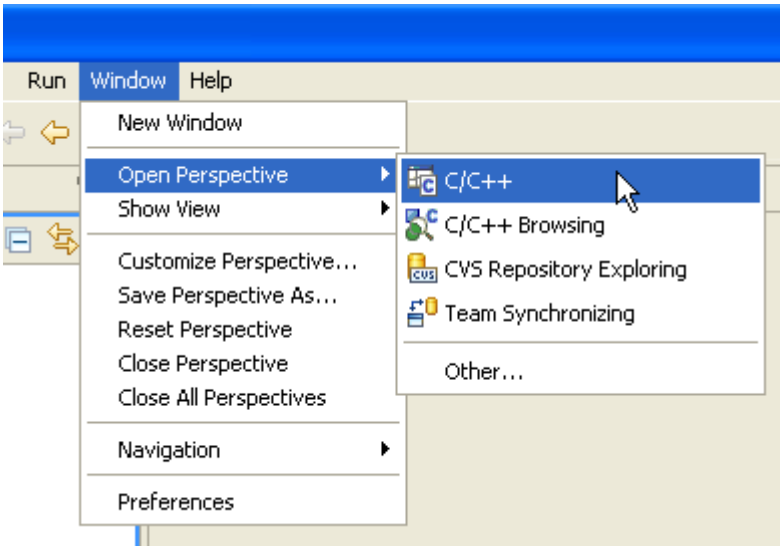
15. SBCTools now prompts for information about applying the settings permanently. Only apply the settings permanently if you modified the settings. Selecting “Yes” will copy a new rc file to the /etc directory and reboot the board such that the new settings will take affect. By Selecting “No” the IP settings will only be valid until the next reboot.

Working With A Project

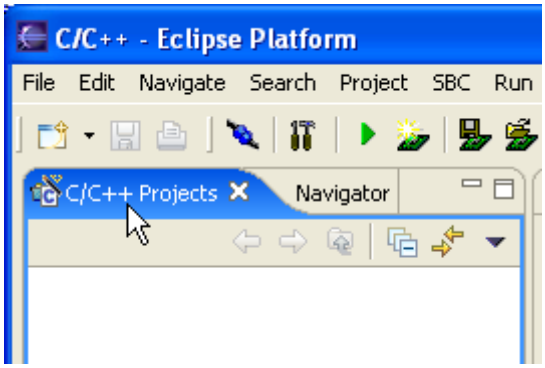
1. Import a project:

a. If the perspective, as shown on the top right hand corner of the screen is the **C/C++ Perspective** then skip to step 2.

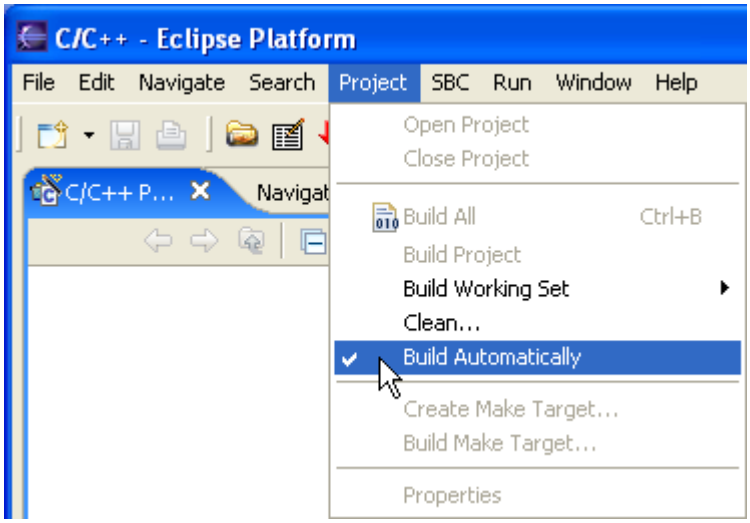
i. Switch to the **C/C++ Perspective**: Click on **Window → Open Perspective → C/C++**.



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

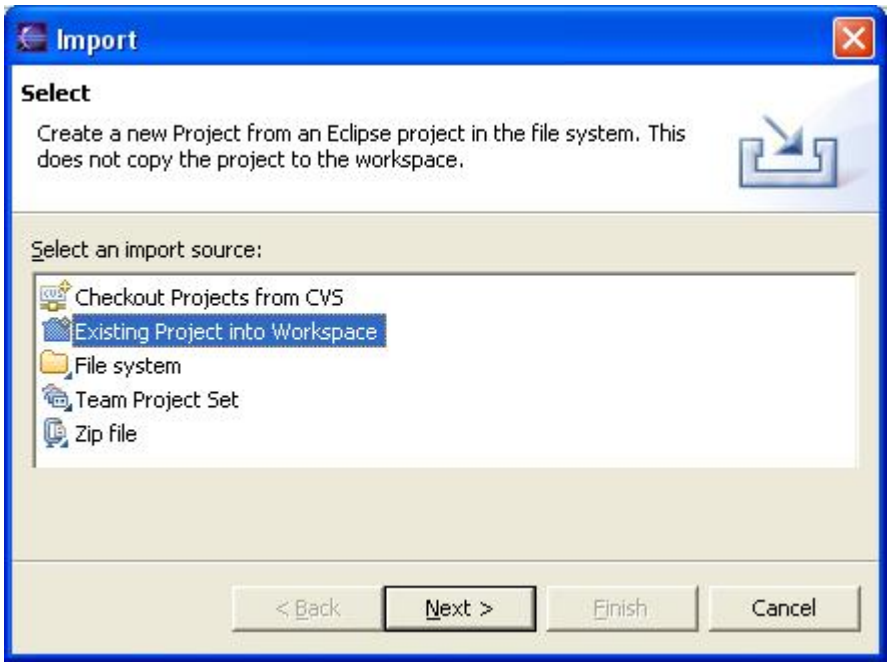


c. Turn off the Build Automatically option by unchecking this option under **Project → Build Automatically**

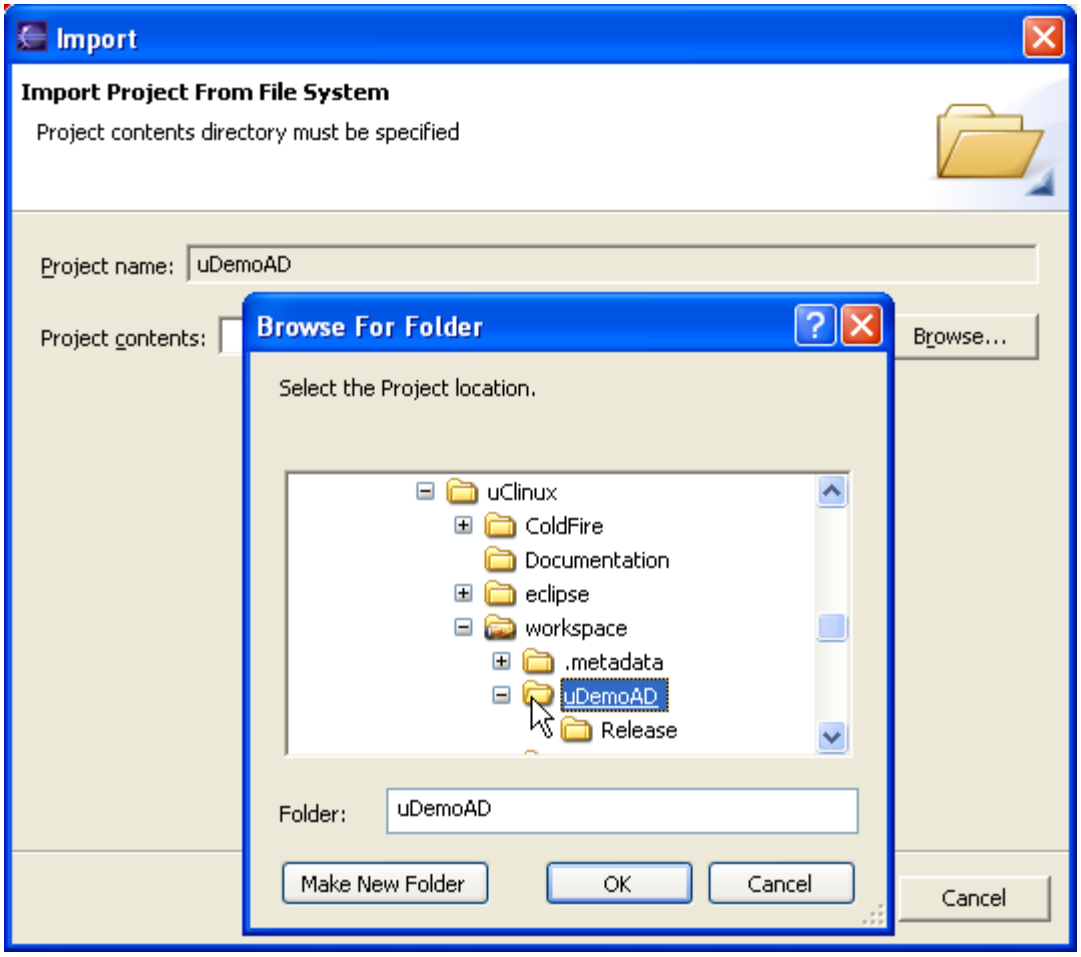


d. Select File → Import… to open the Import dialog.

e. 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.

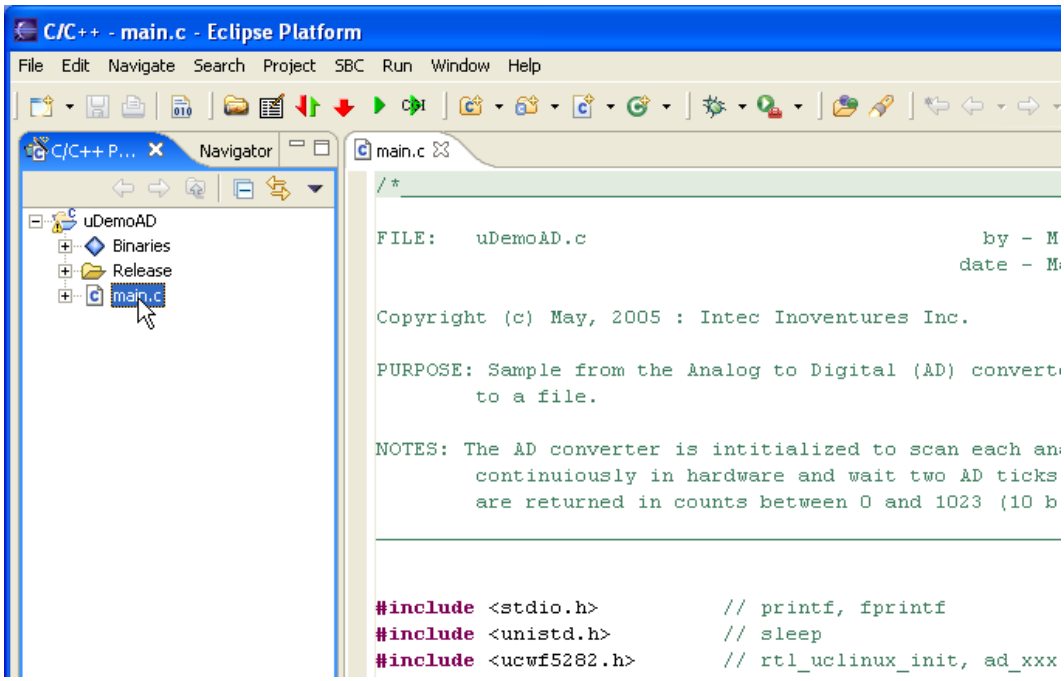

Import
Select
Create a new Project from an Eclipse project in the file system. This does not copy the project to the workspace.
Select an import source:
Checkout Projects from CVS
Existing Project into Workspace
File system
Team Project Set
Zip file
< Back
Next >
Finish
Cancel

f. Browse to "**C:\SBCToolsV2\uClinux\Workspace\uDemoAD**" and click OK.



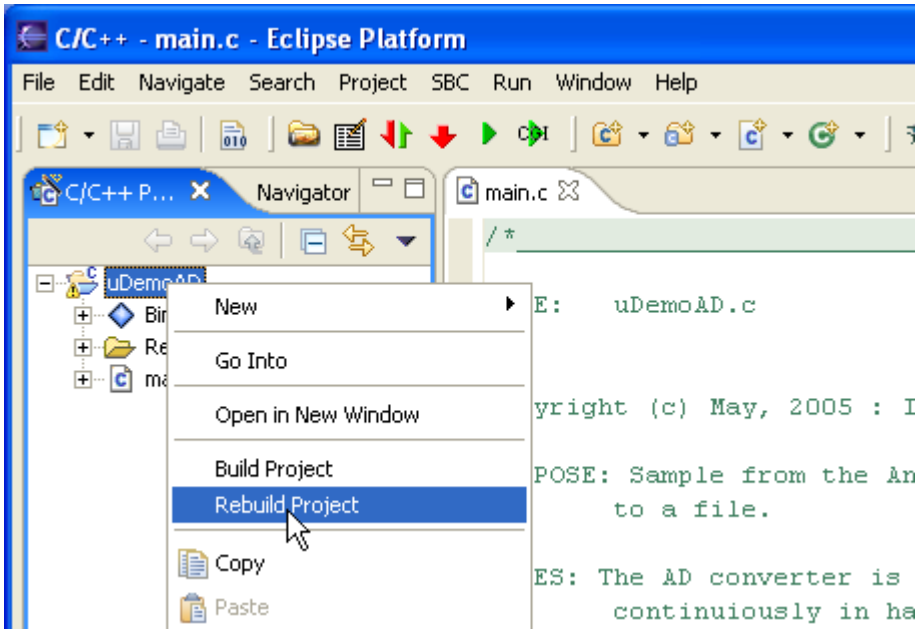
g. Click Finish on the Import dialog. This will import the project into the current workspace and, had "Build Automatically" been checked, it would automatically build it. "Importing" sets up a folder in the Workspace, along with files that enable Eclipse to keep track of the project.

h. View a File: Expand the uDemoAD 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.

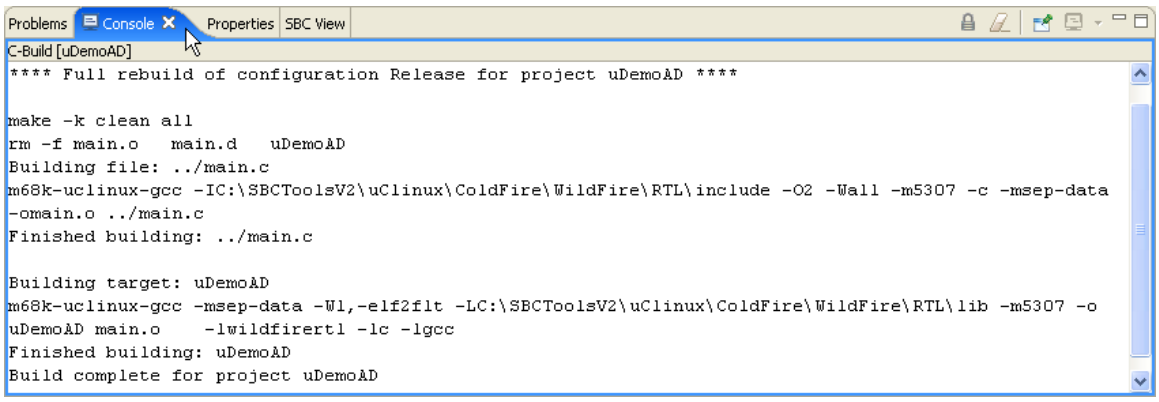


2. Compile a Project:

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



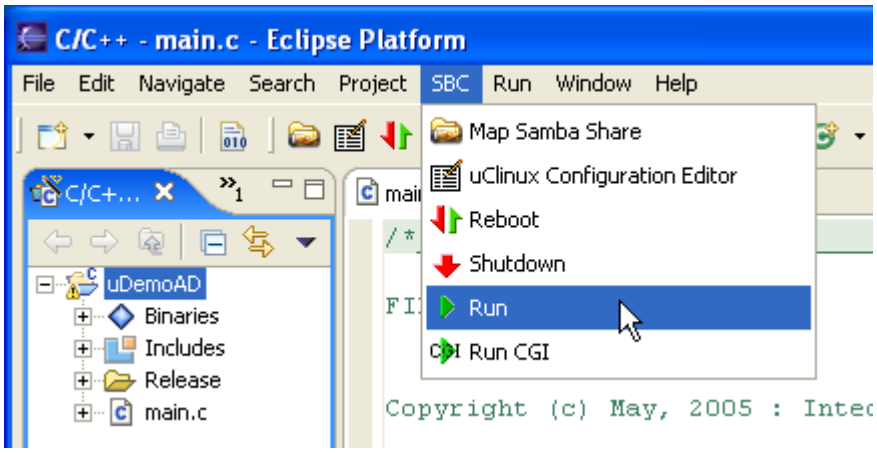
b. As the project is built, all compiler and linker output is written to the **Console** view at the bottom of the screen.



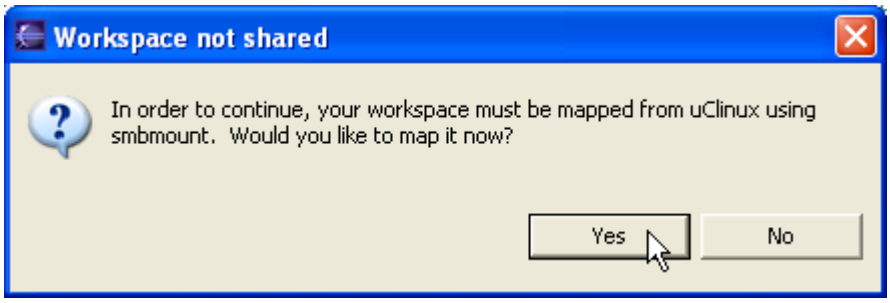
c. If the **SBC view** is in a separate window, pull the **SBC View** tab over to join the **Problems, Console, Properties** tabs in the neighboring window.

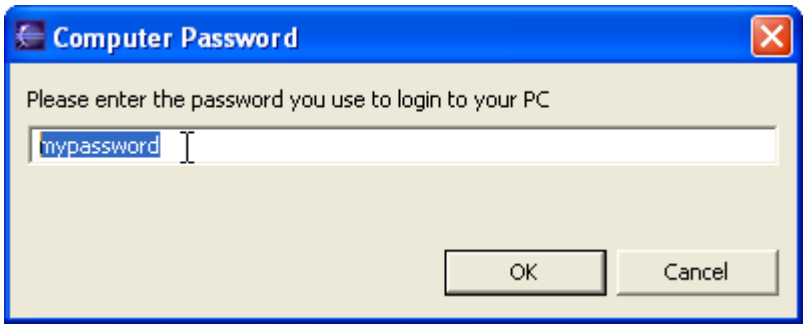
3. Run application

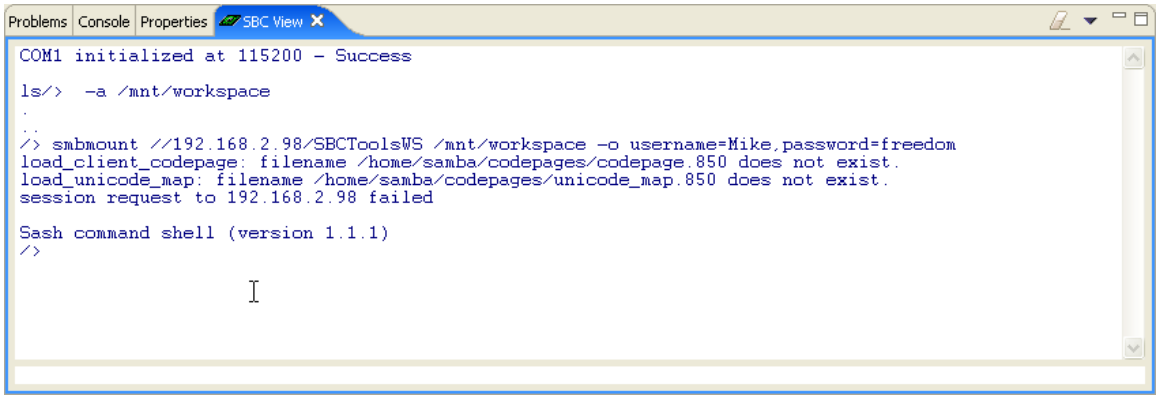
a. Select **SBC → Run**



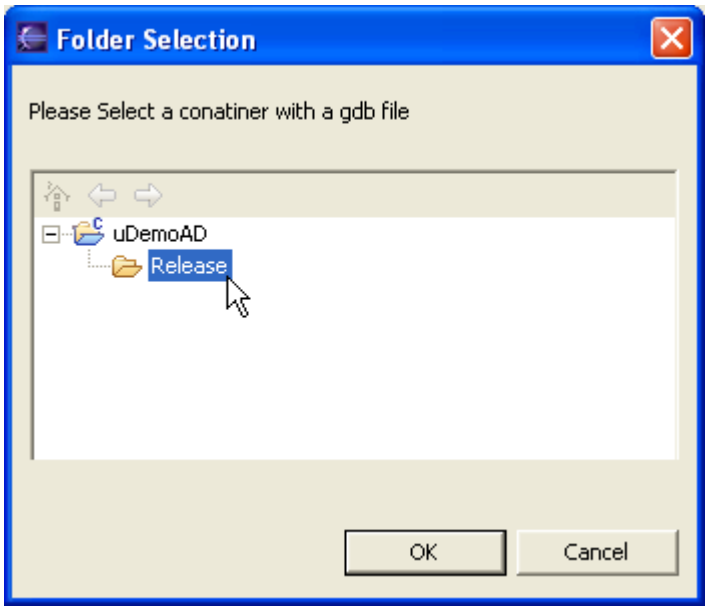
b. You may be prompted to map your workspace, select "Yes" and then enter your PC password and wait for the command to complete. There may be some warning messages while the command executes.



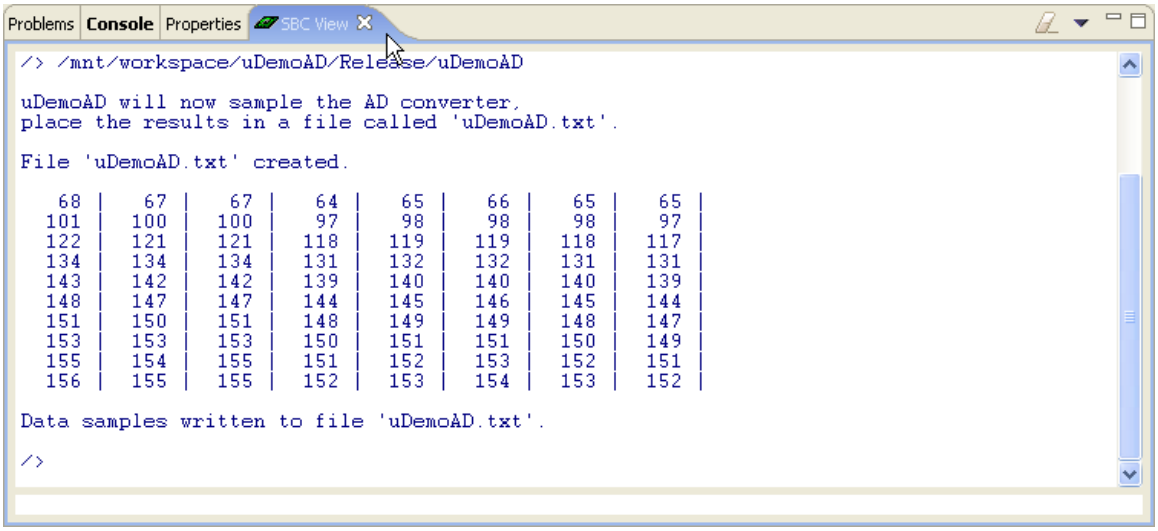




c. When the Folder Selection dialog appears, open the “uDemoAD “ folder, select “Release” and press OK.



d. To terminate execution of uDemoAD, press Ctrl+C in the SBCView.



4. Debugging a program (Compiling for debugging):

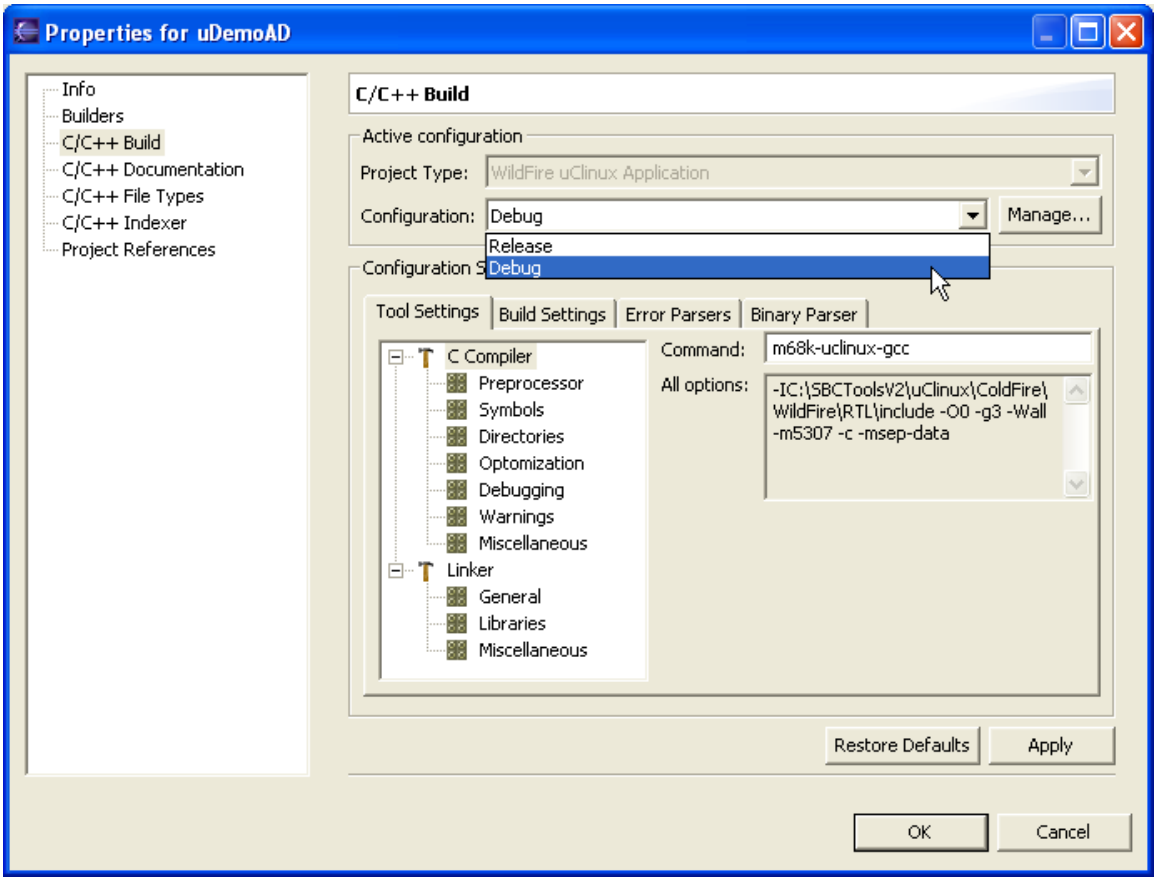
a. Before the program can be debugged, it must be compiled for debugging:

i. Right-click on uDemoAD 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.

vi. Right-click on the "uDemoAD" 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.

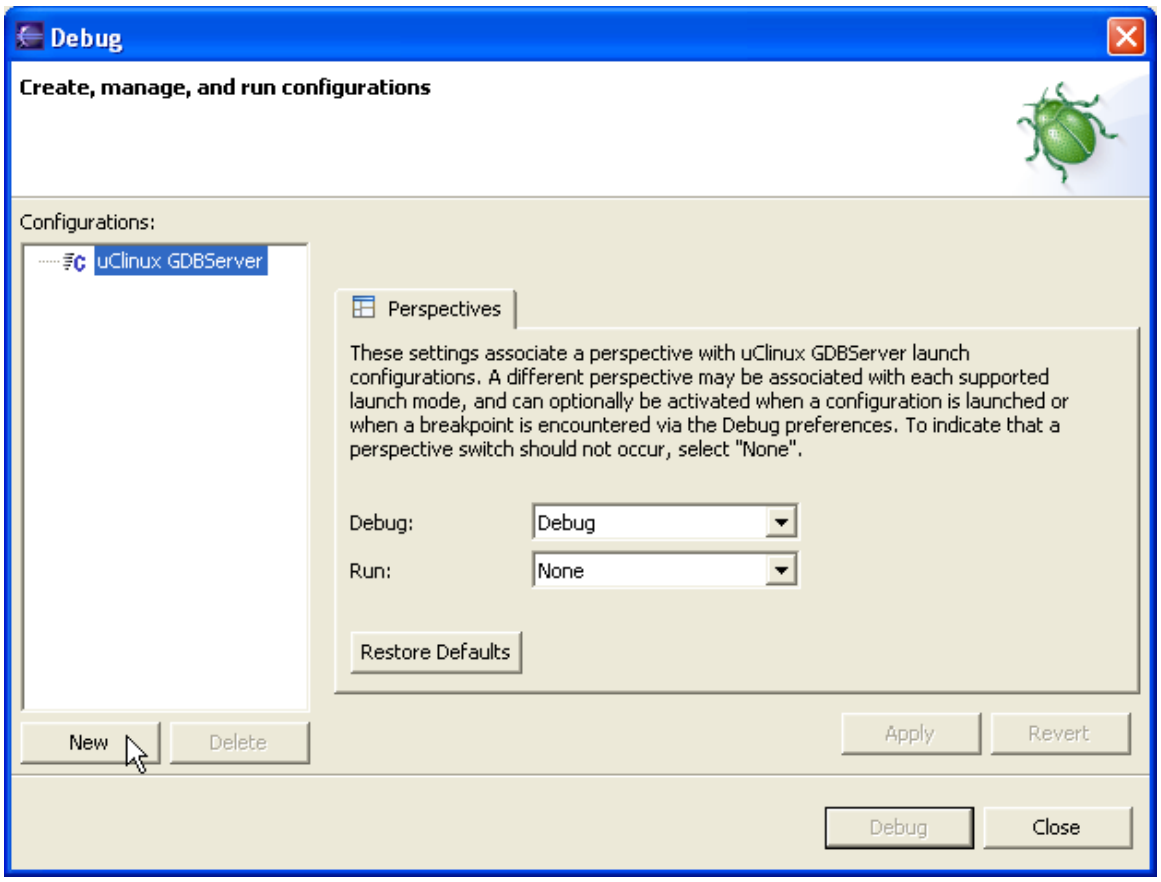
b. Creating a Debug Launch Configuration:

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

ii. In **C/C++ Projects** view, select the uDemoAD project.

iii. Click **Run → Debug**.

iv. In the Debug dialog box, select **uClinux GDBServer** debug configuration type from the **Configurations** list.

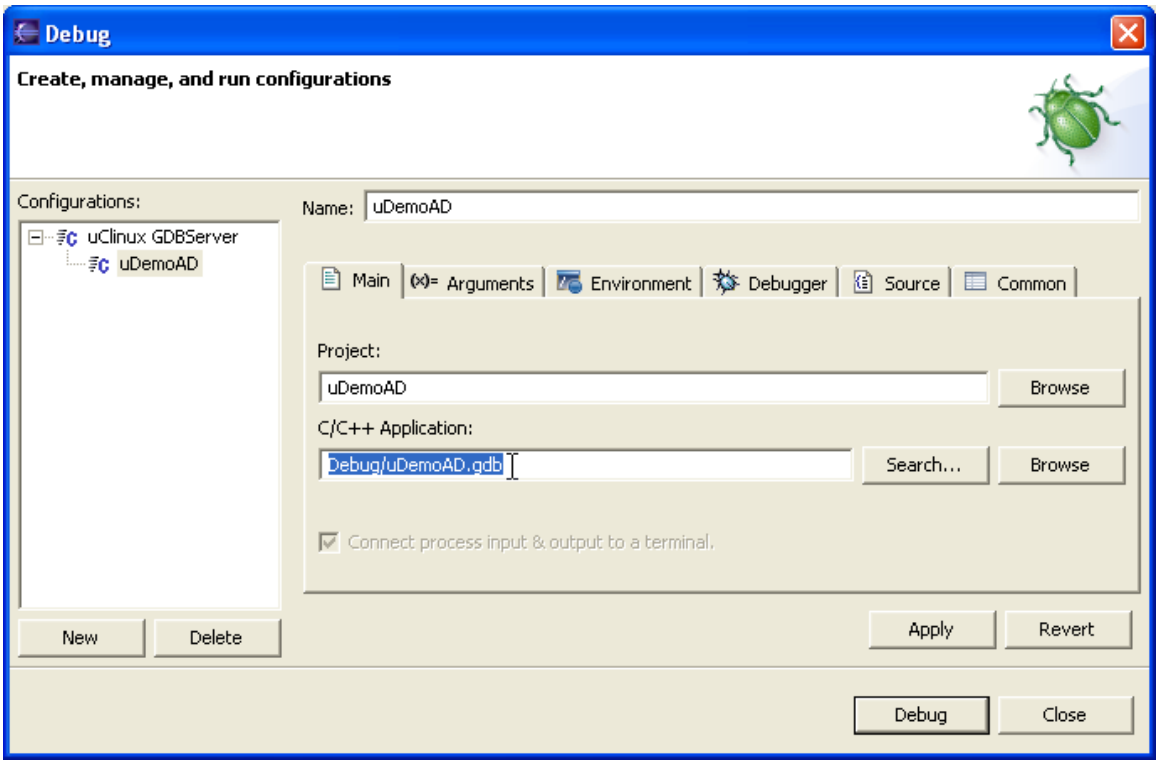


v. Click **New**.

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

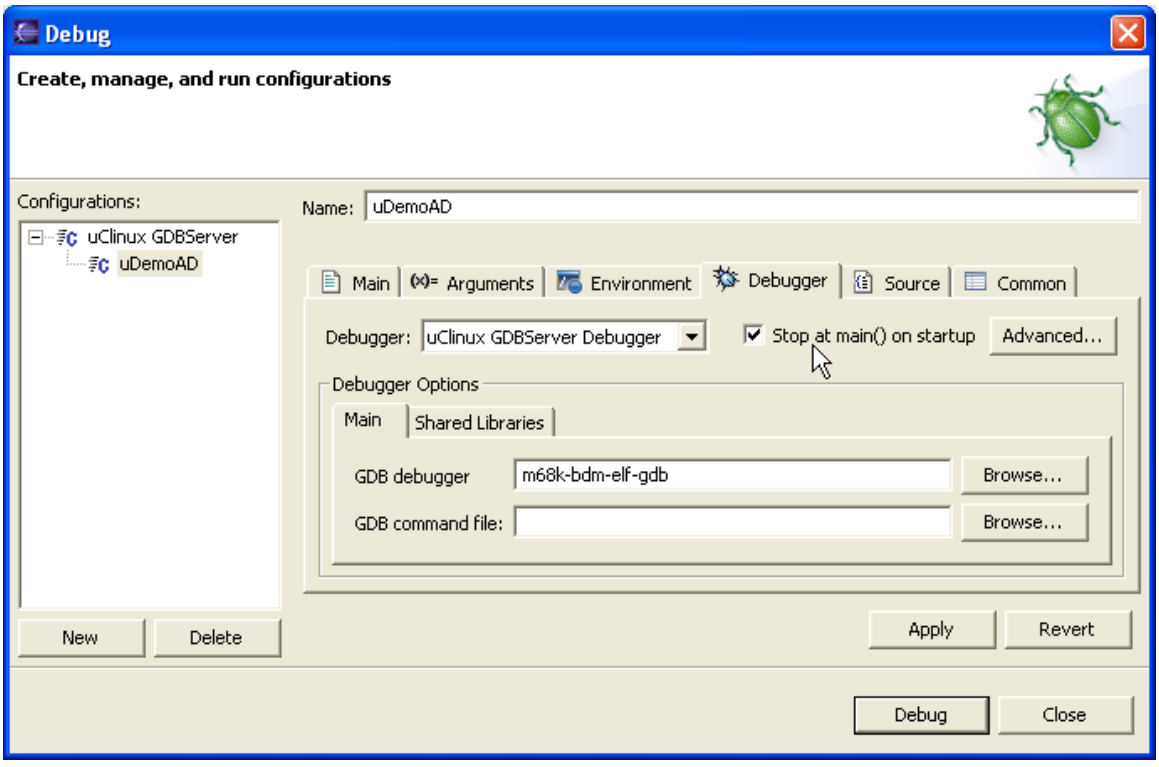
vii. In the **Project** box, type or browse for “uDemoAD”.

viii. In the **C/C++ Application** box, type or browse for “Debug\uDemoAD.gdb”.



ix. Click the Debugger tab.

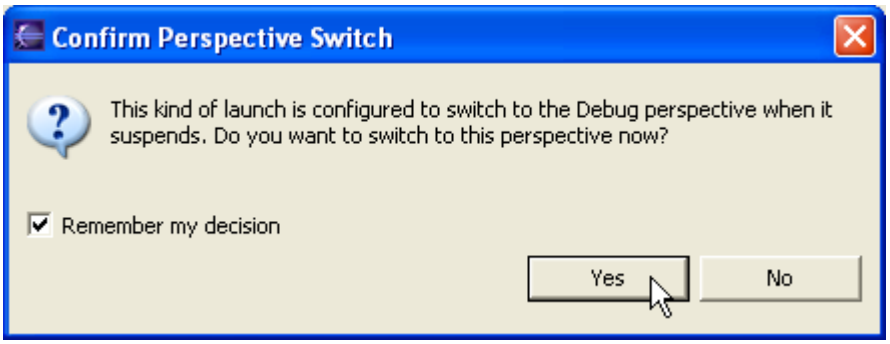
x. Select the **Stop at main() on startup** checkbox.



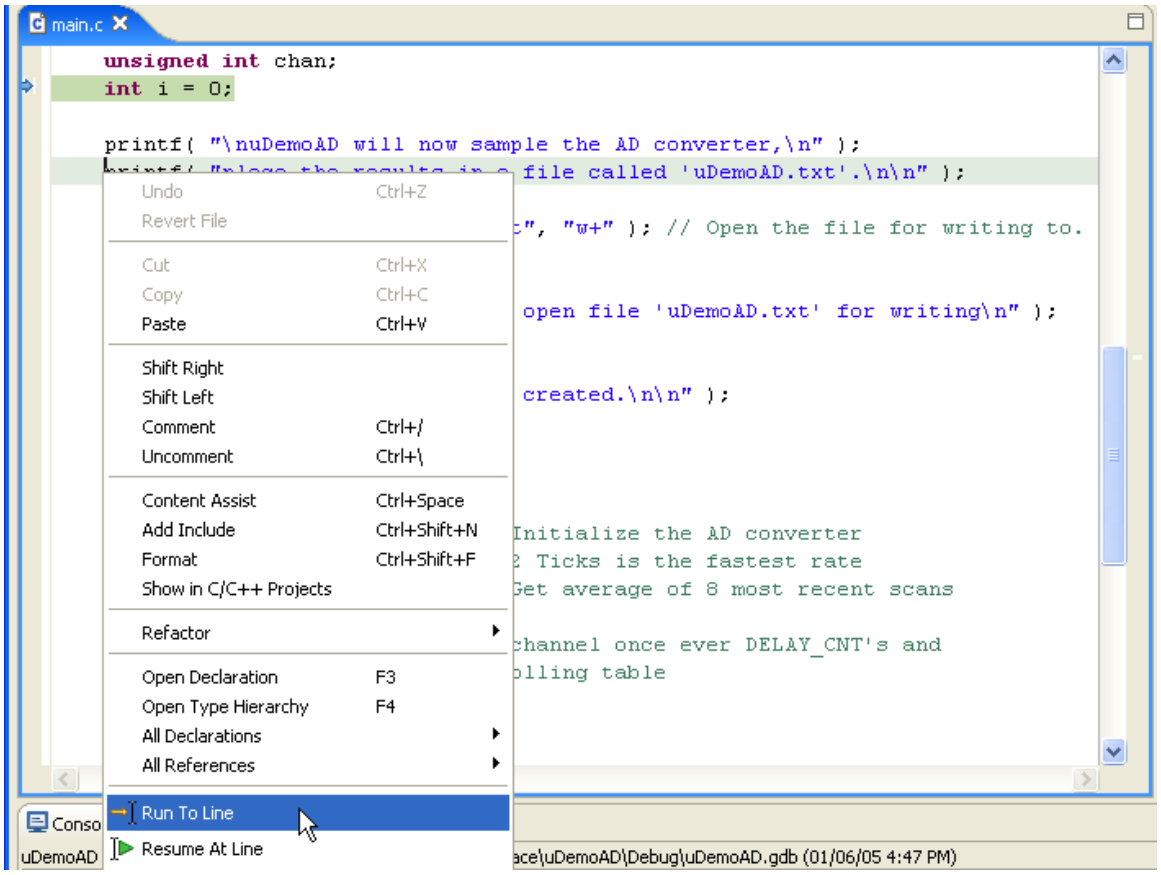
xi. Click Debug. This saves the debug configuration under the name “uDemoAD” and launches the debugger. It will appear under the **Debug Toolbar** for quick access. “DemoAD” is now ready to be debugged.

c. Debugging a program:

i. 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.

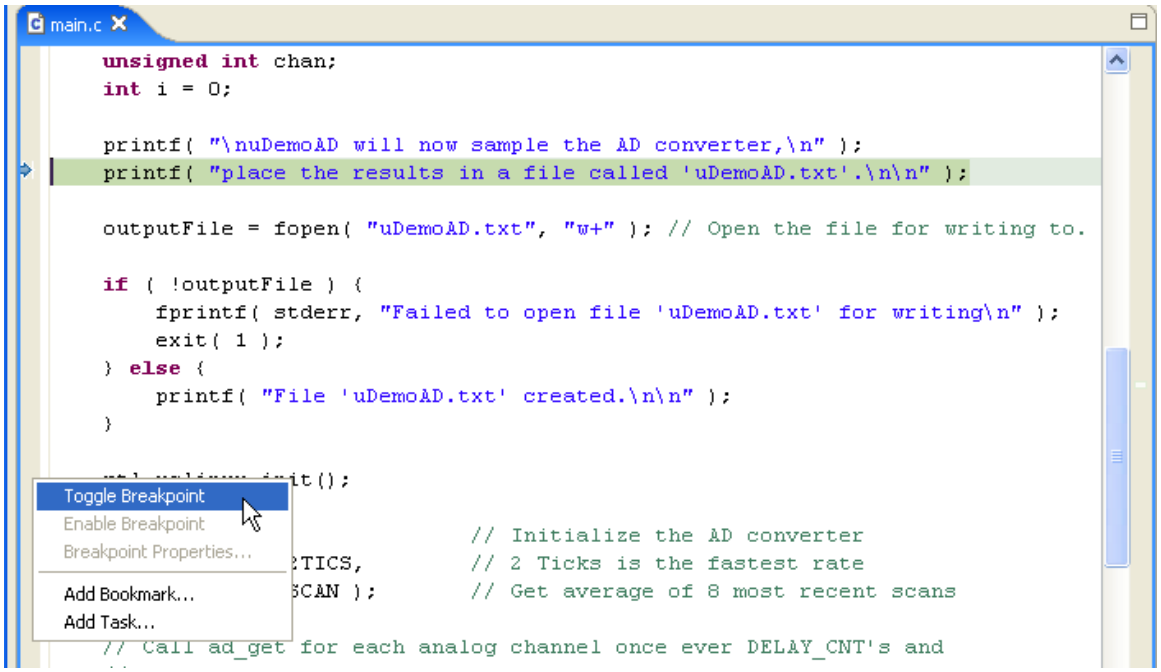


ii. Put the cursor on a line in the program, right-click and select **Run To Line**.

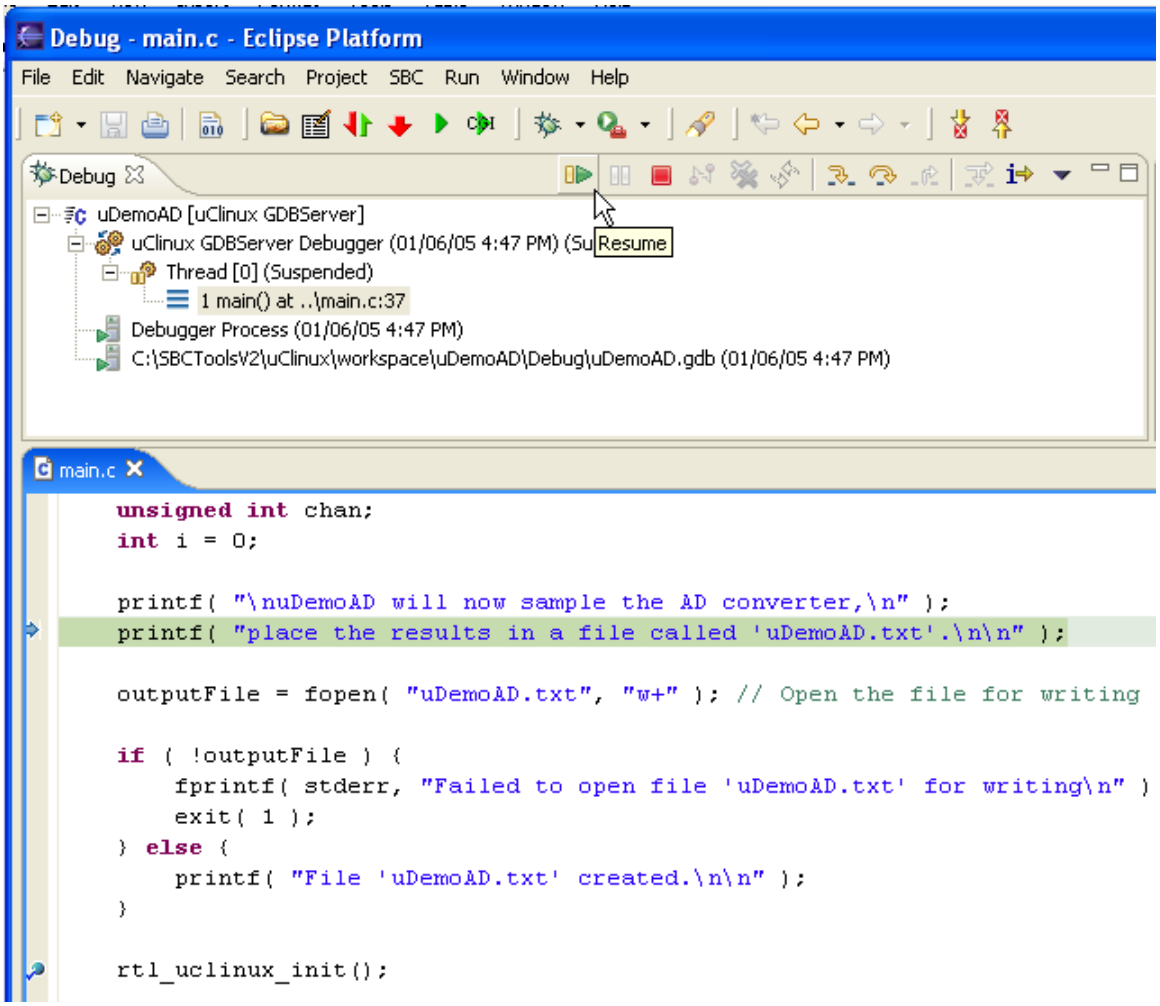


iii. Control returns to the debugger when the program stops at the highlighted line.

iv. 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.



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



vi. You have now successfully launched and run the demo program in the debugger over a TCP/IP connection to GDBServer.

5. Conclusion

This exercise has demonstrated that the **SBCTools** installation has been successful and the SBC is correctly connected to the PC for uClinux TCP/IP 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.

Preparing the WildFire for uClinux

Requirements

In order for uClinux to run on the WildFire the following requirements must be met:

- The WildFire is running dBUG (v3b.1a.1a build 589) or higher.
- uClinux formatted SD card is securely inserted into the Socket
- The autorun parameter is set to “on”

Determining dBUG version

Open a terminal emulator on the PC (i.e. HyperTerminal). Configure it to use the COM port that is connected to the WildFire board. Open the properties for the connection and adjust them to the following:

Baud Rate: 115200bps

Data bits: 8

Parity: None

Stop Bits: 1

Flow Control: None.

Reset the WildFire by shorting the reset pads with a pen tip or short piece of wire. The reset pads are located between CN4 and CN3 as shown in The version is printed as part of the boot up text. The Version and build number in the picture below are Version 3b.1a.1a Build 589.

Updating dBUG

In order for dBUG monitor to boot uClinux you will need dBUG version v3b.1a.1a Build 589 or higher for the WildFire. The new version of the dBUG monitor is backward compatible with older versions, but also contains support for booting uClinux from an SD card. dBUG version 589 is shipped with the SBCTools V2 and is stored in "C:\SBCToolsV2\Standard\ColdFire\WildFire\dBUGImage\". In order to update your board to the new version you will first need to download the updater application and then follow the instructions.

1. Open HyperTerminal with the settings as described above.
2. At the dBUG prompt type "dl". You will then need to select **Transfer -> Send Text File.**
3. Broswe to the directory with the dBUG updater (C:\SBCToolsV2\Standard\ColdFire\WildFire\dBUGImage\) – make sure that file type in the open dialog box is set to "All Files (*.*)"
4. Select dBUGUpdater.s19 and then OK. The image will download and you will see a spinning text to indicate it is working.
5. Once it is done you want to start the application by typing "go 0x10020000" at the dBug prompt.
6. After accepting the warning you will need to download the new dBUG monitor. Again select **Transfer -> Send Text File.** and this time select wildfire_iflash.s19 It will take a couple of minutes to download.
7. Now after another warning the new dBUG monitor will be flashed over the old one.
8. Reset power on the board only after the operation is complete.

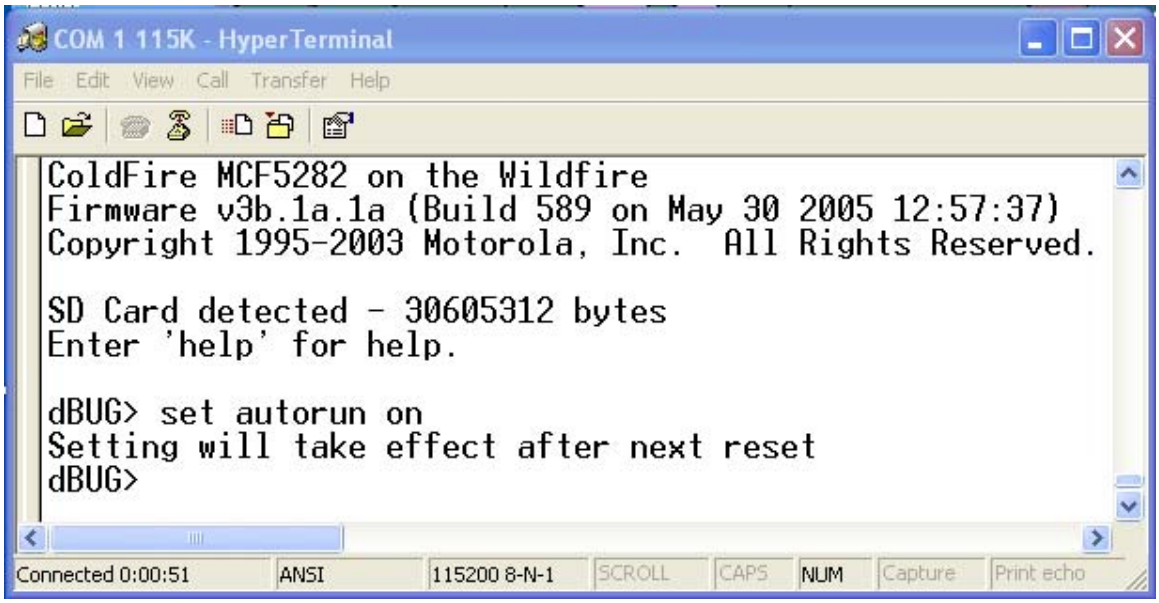
uClinux SD Card

All uClinux development kits come shipped with a preformatted SD card containing the uClinux kernel and root file system. To reformat you SD card back to factory defaults follow the instructions below.

1. Under Construction…

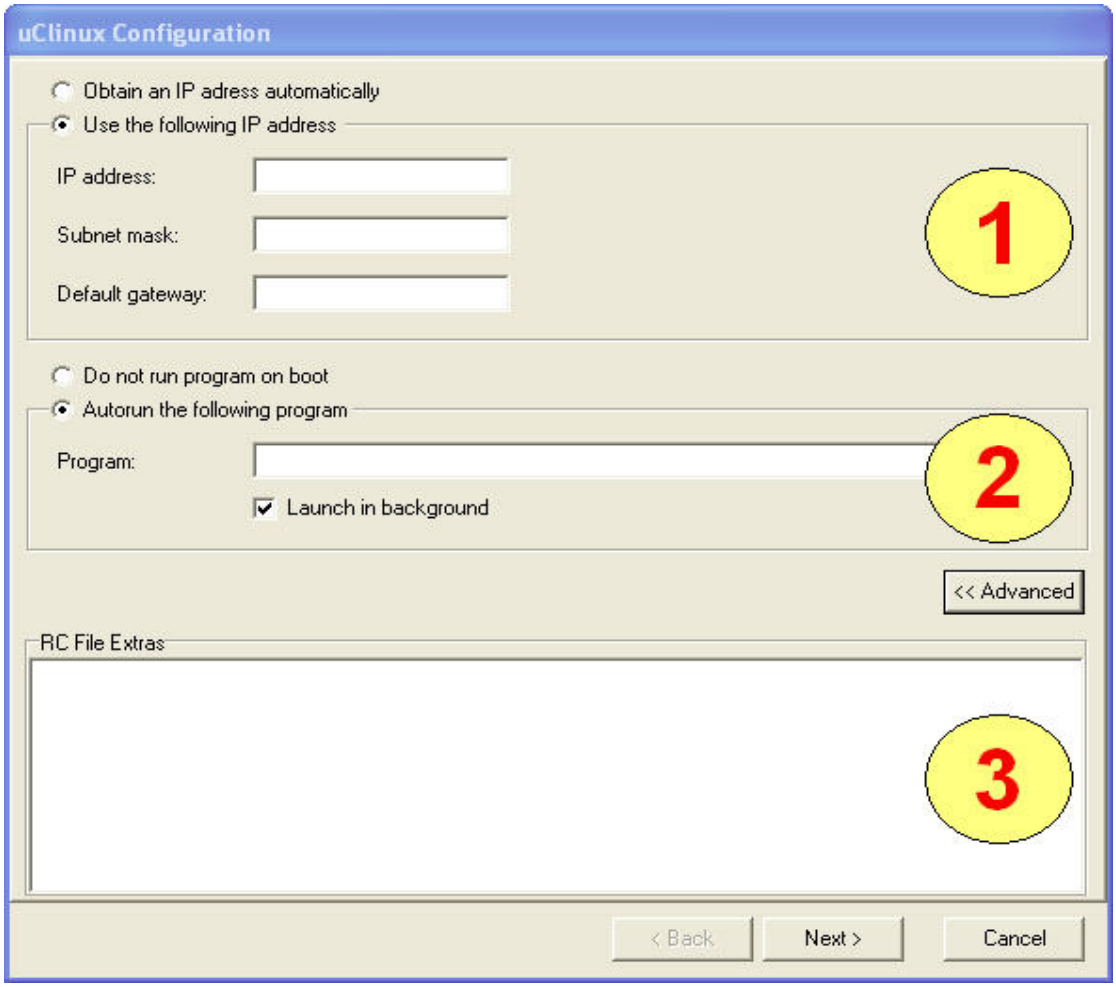
Setting Autorun

Autorun can be turned on from the dBUG command prompt. If you follow the instructions above in section 1.4 you will have a dBUG command prompt. At the prompt type "set autorun on", you should then see the text "This setting will take effect on the next reboot"



If you reset you board the dBUG monitor will now automatically boot uClinux from the SD card if present.

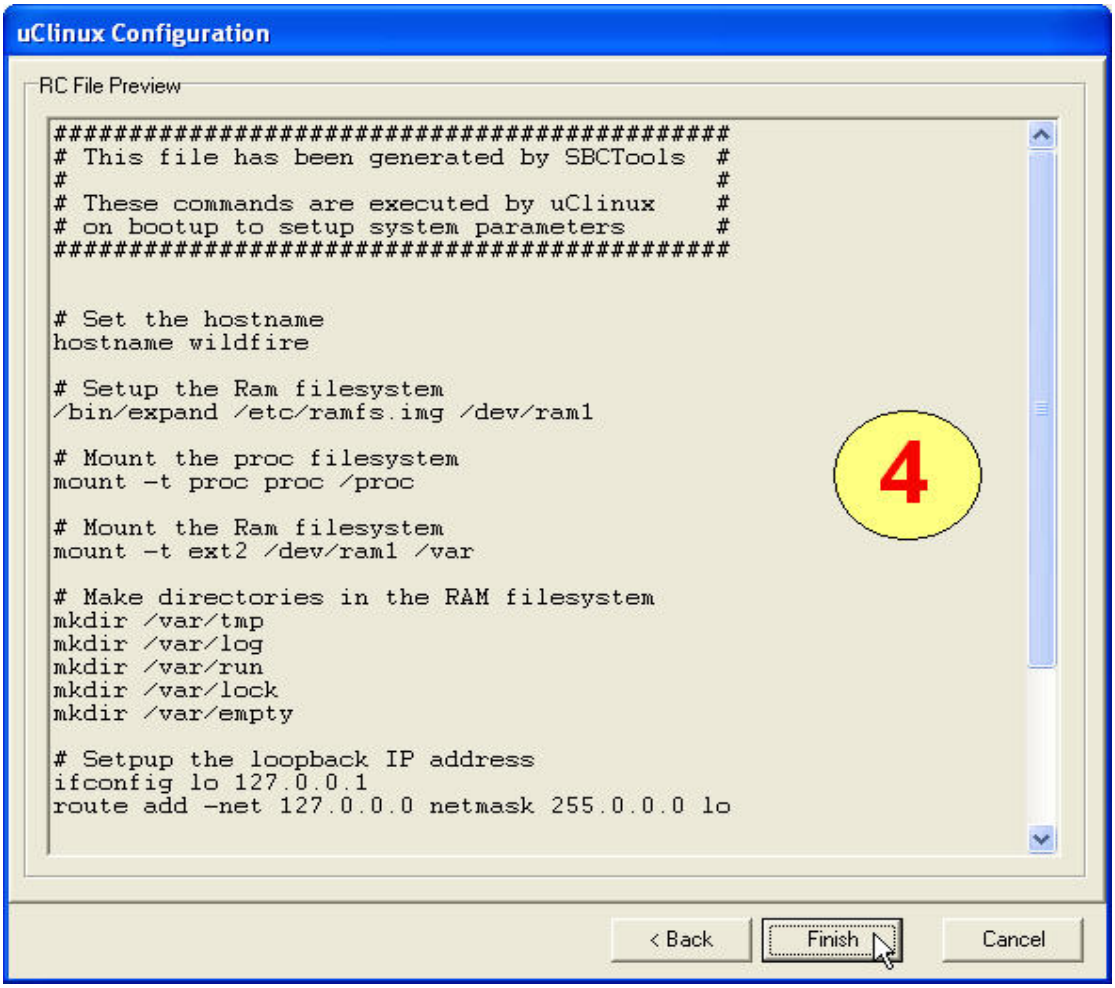
uClinux Configuration Editor



1. *Network Settings* – If you would like to use a DHCP server for automatically getting you IP address then select this option. Otherwise you may specify a unique static IP address, Subnet mask, and Default Gateway. If you are unsure of how to specify these settings see your network administrator.

2. *Autorun Program* – This setting will be useful when deploying a finished product. You can specify the uClinux root relative path to the application you would like to run when uClinux boots. If you wish to launch your program in the background such that you always have access to the uClinux shell then select "Launch in background" option.

3. *RC File Extras* – Here you can enter any more uClinux commands which you would like to add to your boot up procedure. again this would be useful at deployment time. Beware of commands which take a lot of time and run in the foreground as they may cause you uClinux on the WildFire to boot slowly or incorrectly.

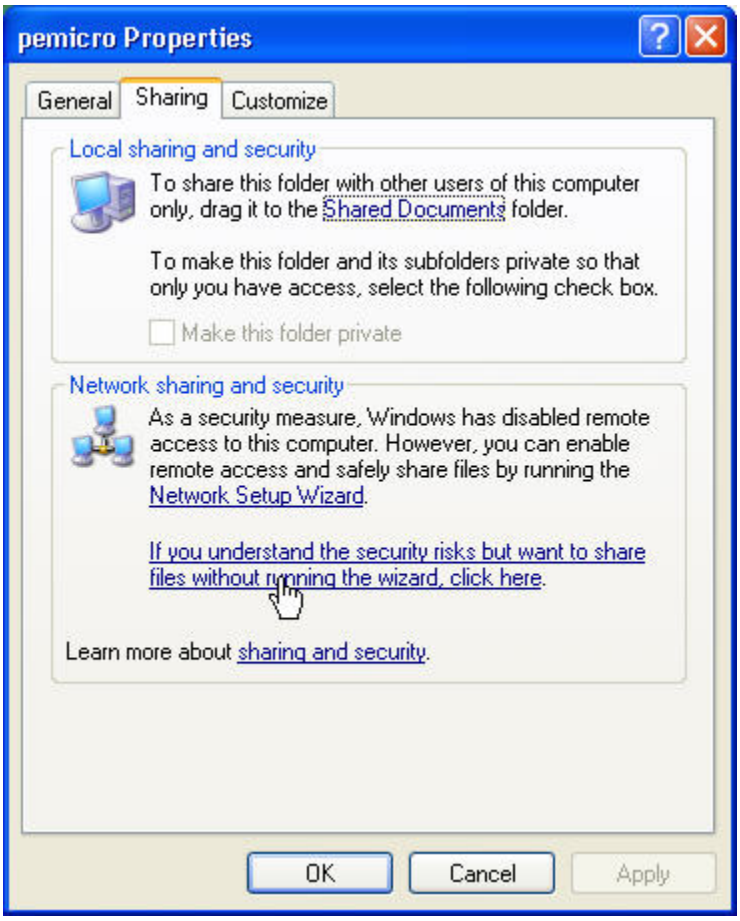


4. *Preview* - After pressing the "Next >" button you will be able to preview the exact format of the uClinux "rc" file. This rc file will be saved to the root of your workspace if Finish is selected. You will then be prompted by SBCTools to Apply the settings permanently. If you choose "yes" then the rc file will be copied to /etc/rc on your microcontroller with a backup of what is currently there being saved to /etc/rc.bak and uClinux will be rebooted so the new settings can take effect. If you choose "No" then the IP settings will be applied immediately and will only be in affect until the board is rebooted.

Windows File Sharing

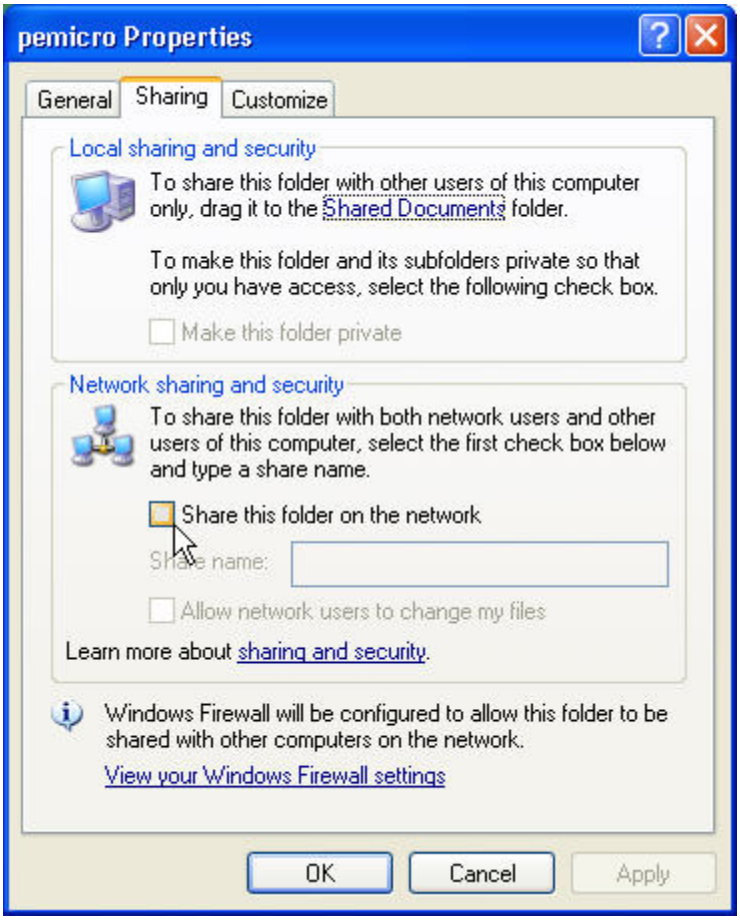
Some computers may have the Windows file sharing protocol disabled (this is the default for Windows XP Service Pack 2). In this case Windows file sharing will need to be enabled before using samba file sharing with SBCTools.

1. Turn on Windows file sharing by right-clicking the folder to be shared → Sharing & Security → click on the hyperlink as shown to bypass the Network Setup Wizard. (If you do not see this dialog, and you do see a dialog similar to the one in step 3, then Windows File sharing is already enabled).



2. When prompted to use the wizard decline and "Just enable file sharing"

3. You should now see a screen similar to this.



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.