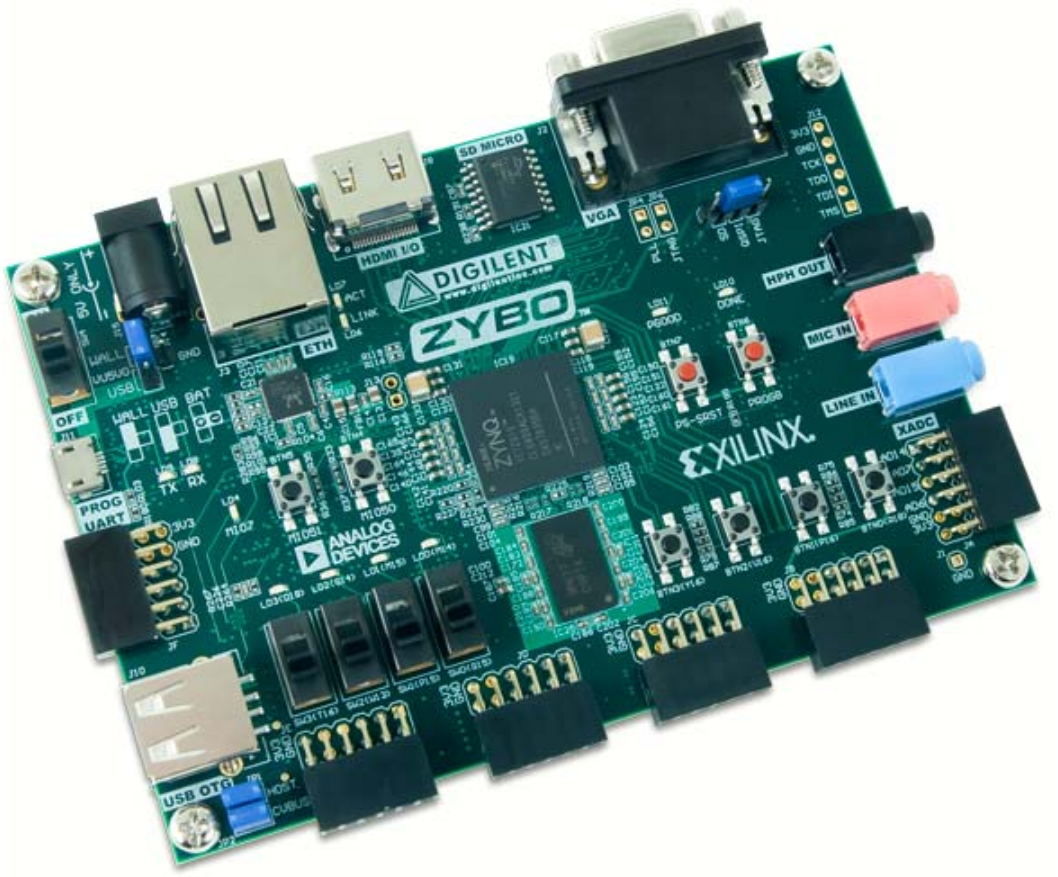
Getting Started with TargetOS

on Digilent's

ZYBO Zynq™ Development Board



1 Introduction

This document covers how to get started with Blunk Microsystems' TargetOS embedded operating system on Digilent's ZYBO Zynq development board. It covers the following steps:

- Installing the CrossStep IDE, TargetOS RTOS (lite-kernel), and ZYBO demo package.
- Setting up a new ZYBO: cables, jumpers and power.
- Installing the First Stage Boot Loader (FSBL), bootrom app (debug-over-Ethernet support) and an FPGA bitstream image into the ZYBO's SPI NOR flash
- Building Blunk's kbench kernel benchmark sample application and running it on ZYBO.

2 IDE/RTOS installation

This topic is covered lightly here because a CrossStep wizard guides you through this during its installation. Mainly you just go to www.crosstep.com and download the CrossStep installer. Pick 'ZYBO' when asked to select a board. When asked to pick an initial sample application, please pick 'kbench' to be in sync with the explanations below.

3 Board Setup

ZYBO has a small number of jumpers. Only three locations are fitted with jumper posts and their settings are well-labeled. Those jumpers should be set as described below.

JP7 – Located near on/off switch in upper left cover of board. Controls how the board is powered. For low-complexity designs, the board may be powered by the USB cable used for the UART connection by connecting the pins labeled "USB" and "VU5V0". To use the external power supply in the ZYBO Accessory Kit, connect the pins labeled "WALL" and "VU5V0".

JP1 - Located near bottom left corner of board. Controls whether the board supports USB host or device mode. To support Blunk's TargetUSB host mode USB stack, these pins should be jumpered.

JP5 - Located near top right corner of board. Controls where Zynq's bootrom fetches Blunk's First Stage Boot Loader (FSBL) from: SD card or QSPI NOR flash. Initially the left two pins (labeled "SD") should be shorted, in order to run Blunk's FSBL from the SD card. After the FSBL has been copied to the NOR flash, the jumper should be moved to the middle two pins (labeled "QSPI") for a much faster boot time.

With the configuration above, connect a USB cable between your PC and ZYBO micro USB connector J11, located below the power switch and then turn on the board via the on/off switch (SW4). If you have the proper driver installed, this will add a USB serial channel to your PC. You should open a terminal program on this channel using 115200 baud rate, 1 stop bit, no parity, and 8-bit character length.

Before you can do anything useful with ZYBO, you need to install several files into the board's QSPI NOR flash, which is the topic of the next section.

4 Preparing ZYBO's SPI NOR Flash

ZYBO's QSPI NOR flash holds: 1) the FSBL, which is the first user program to run after-reset, whose main jobs are to configure the FPGA, initialize SDRAM, and load applications into the SDRAM, 2) the bitstream file containing the IP used to configure the FPGA, and 3) CPU0/CPU1 applications. (It also holds NVRAM parameters and a flash file system volume).

Blunk's FSBL uses TFTP to install the above three files into ZYBO's QSPI NOR flash, but you need the FSBL running on your board before you can use it. So, although later you will always have ZYBO boot from QSPI NOR flash, initially you need it to boot from an SD card and that is how JP5 was configured above.

The eval package includes 'BOOT.BIN', which is Blunk's ZYBO FSBL in the special format recognized by Zynq's bootrom, in the following directory:

```
bsps\arm\zybo_demo\fsbl\bootgen\
```

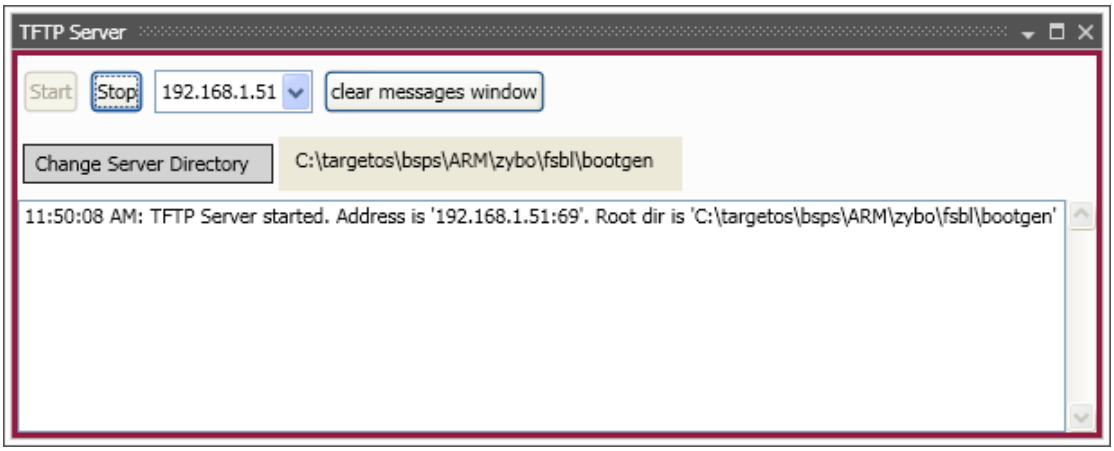
To run the FSBL on your board, copy this file to a microSD card, insert the card into your ZYBO (connector J4, underside of the board), and press the soft reset switch (PS-SRST/BTN7). You should see the following on your terminal program:

```
FPGA bit image is erased
CPU0 boot app is erased
SD boot, init finished, press any key for menu
```

Press any key to get the following menu:

```
'0' - display FSBL and partition info
'1' - edit IP addresses
'2' - erase flash partition
'3' - update flash partition
'4' - verify flash partition
'5' - configure FPGA
'6' - start applications, if valid
'7' - load/run CPU0 /dev/tftp/app.elf
'8' - perform SDRAM reliability test
'9' - edit NVRAM registry
```

Start a TFTP server on your PC, serving the directory that contains ‘BOOT.BIN’, the file you copied onto a microSD card. Using CrossStep’s TFTP server (accessed via the Tools menu), this looks like the following:



In the terminal window for the FSBL menu, enter ‘1’ to edit IP addresses and then ‘1’ again to set the IP address of the TFTP server. This would be the IP address of your PC.

To install the FSBL, enter ‘3’ to update a flash partition and then ‘0’ to copy BOOT.BIN to the first partition. Next install the FPGA bitstream by entering ‘3’ and then ‘3’ again. ‘BOOT.BIN’ and ‘system.bit’ are in the same directory, so no TFTP server change is needed.

The bitstream file in the demo package is an example Digilent design that implements a video controller. Not every bitstream file is named ‘system.bit’, but it simplifies FSBL use to not have to enter a different name. It is possible to customize Blunk’s FSBL, but you might find it easier to just rename your bitstream files to ‘system.bit’ before installing a new copy via the FSBL menu.

With the FSBL now installed in QSPI NOR flash, you can test booting from flash by switching power off, changing jumper JP5 to the middle position (“QSPI”) and switching power back on. Your terminal window should show this:

```
CPU0 boot app is erased
QSPI boot, init finished, press any key for menu
```

Some terminals need you to disconnect and re-connect after power cycling boards. Try that and press ZYBO’s soft reset button if this text doesn’t appear at first. Note the “QSPI boot” message versus the prior “SD boot”. You should also see ZYBO’s green “DONE” LED lit, showing that the FSBL configured the FPGA using the ‘system.bit’ file you installed.

The FSBL tries to stay out of view as much as possible, but has a problem currently in that there is no application it can load and start. If an application file is installed, the FSBL silently starts the application at each boot. The only terminal output will be generated by the application.

To return to the FSBL menu, hold down the button labeled BTN4/MIO50 while depressing and releasing the soft reset button. This skips the application start, taking you right to the FSBL menu. It also skips FPGA configuration, but you can manually invoke that by entering ‘5’ and then ‘0’ in the boot menu. You’ll have a chance to do this later, but first you need to install a CPU0 application (use of CPU1 is outside the scope of the demo package).

An ‘app.elf’ file for Blunk’s bootrom application, which runs the TargetMon debug-over-Ethernet target monitor, is in the directory below:

```
bsps\arm\zybo_demo\bootrom\
```

To install the bootrom application on your ZYBO, change your TFTP server directory to the path above, enter ‘3’, enter ‘1’, and then let installation finish. Now if you press the soft reset button, your terminal shows the bootrom app start, which should look similar to the lines below.

```
TargetOS, Release 2015.0. Built at 10/12/2015 5:38:27 PM.
Copyright 2015, Blunk Microsystems. All rights reserved.
--------------------------------------------------------
Starting TargetMon
Terminal baud rate: 38400
SDRAM size = 255MB
CPU0 revision r3p0, PS version 3.1
Hostname: zybo
IPv4 Default gateway: 192.168.1.1
GEM0 IP addr/mask: DHCP
GEM0 address: 00:1E:C0:DD:5D:8A
Starting FTP server for IPv4
Starting Telnet server for IPv4
Starting TFTP server for IPv4
Using: 0x00100020 - 0x00200000 (cold boot)
--------------------------------------------------------
.Starting..
GEM0: link=up speed=1000M duplex=full
GEM0: DHCP IPv4 address 192.168.1.138
```

You can now try getting back to the FSBL menu by holding the BTN4/MIO50 button down while depressing and releasing the soft reset button. You should see the familiar FSBL menu. To see another FSBL feature, enter ‘0’ to view a summary of its status. It should appear like this:

```
Cause of last reset: SRST_B soft reset
CPU revision r3p0, PS version 3.1
PL configured
Design Name: system.ncd;HW_TIMEOUT=FALSE;UserID=0xFFFFFFFF
Zynq Name: 7z010clg400
Design Date: 2014/04/07
Design Time: 22:37:48
Image Length: 2083740
FSBL partition (3 blks = 192KB) is valid, 43.8% full
CPU0 partition (16 blks = 1MB) is valid, 27.8% full
CPU1 partition (16 blks = 1MB) is erased
FPGA partition (32 blks = 2MB) is valid, 99.4% full
Free OCM heap = 111792B
```

Finally, to prepare for the next section, press the soft reset button again and leave the bootrom application running.

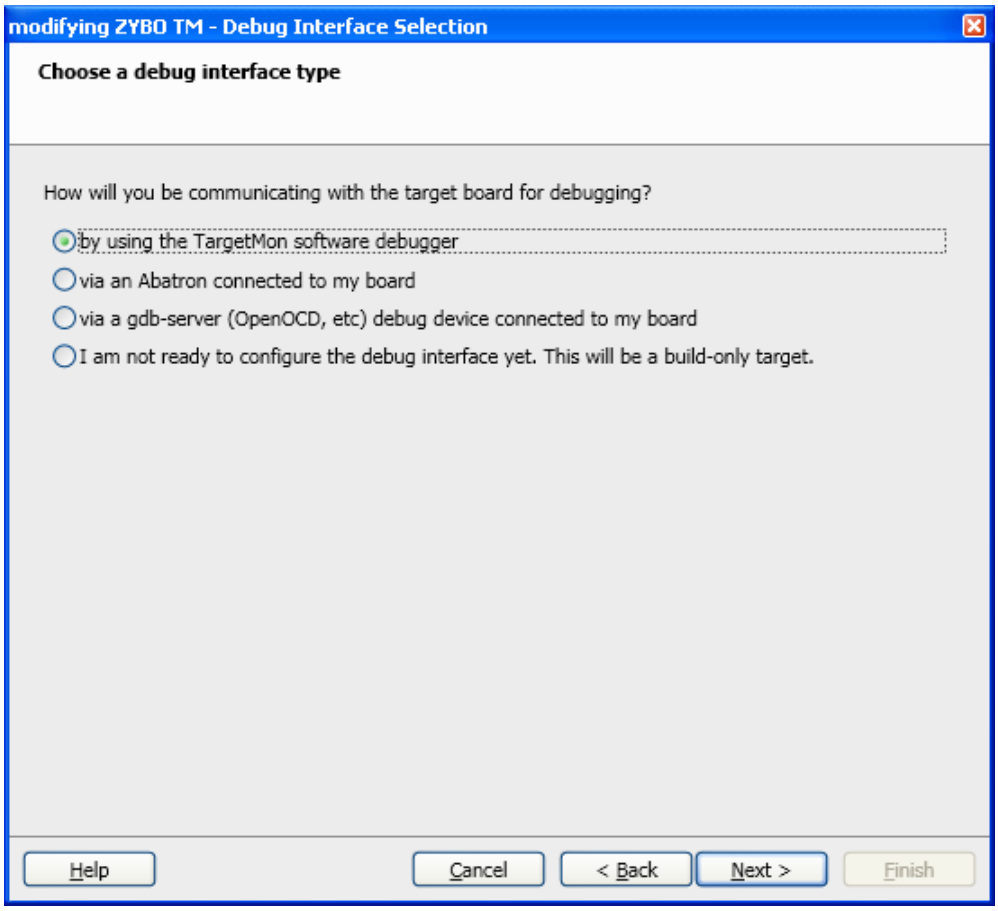
5 Running kbench

Now you are ready to build the kbench sample application and run it on ZYBO. kbench is a kernel benchmark application. If you specified the kbench application during CrossStep's initialization, it will already be loaded. Otherwise, use the Project/Open dialog to browse to and select this file:

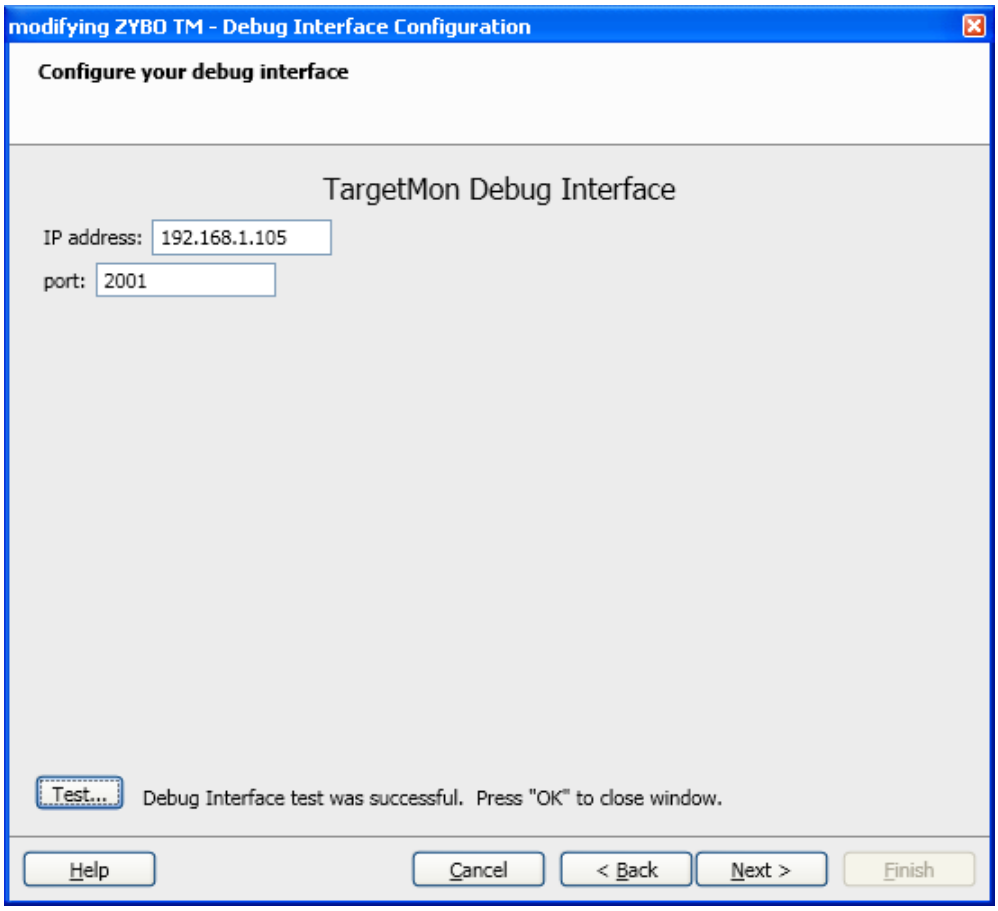
```
apps\kbench\kbench.gbp
```

To use a TargetMon debug connection, you need to modify your selected target to specify the IP address of your ZYBO. Use the 'Manage Targets' command in the Options menu to bring up the Targets dialog. Select the entry for the target name you created during installation, then right-click on it and select "modify.." in the menu that comes up.

Select "Digilent Zynq Development Board (demo BSP)" as your target board, accept the GNU toolchain settings, and when you get to the panel below choose 'TargetMon', as shown.



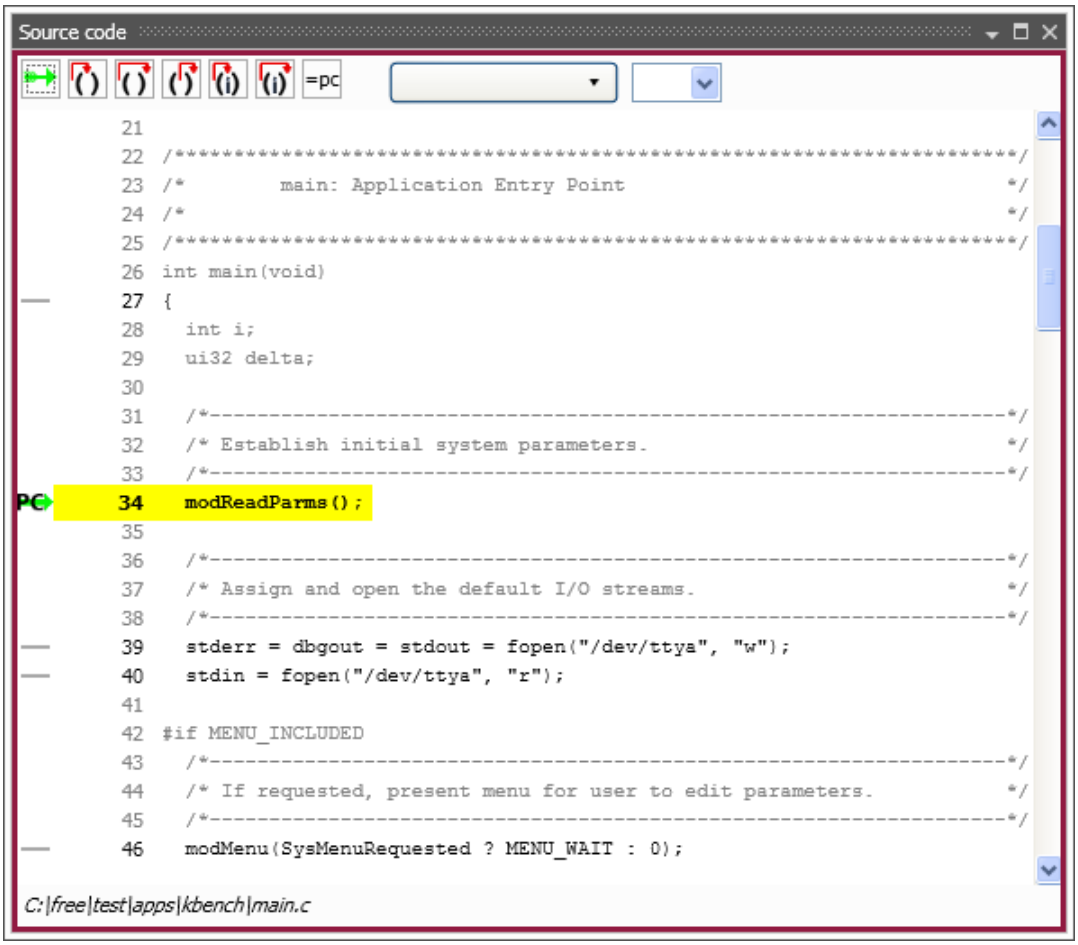
Enter your ZYBO's IP address in the next panel, getting the address from the terminal window that shows the bootrom application's start-up. The IP address is on the last line. Leave the port number as 2001.



You can optionally test the TargetMon debug connection by clicking the “Test” button. Click past the following two dialogs and then click “Finish” to close the target modification dialog.

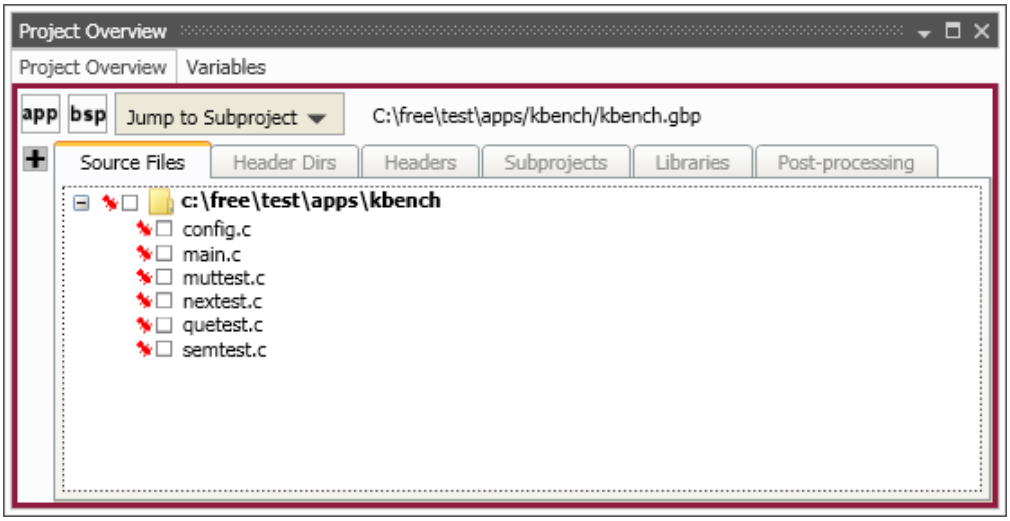
Build the sample application by clicking the button in CrossStep’s upper-left corner that has the hammer icon. Or you can use the keyboard shortcut control-B. When the build finishes, a build complete message appears in the Output window. The Stop button next to the Build button becomes active whenever a build is in progress. You can click that icon if you need to stop a build for any reason.

After the build completes, or instead of clicking the build button if you want to give a combined “build and launch debugger” command, either click on the “debug” button or use keyboard shortcut control-D. This ensures the build is up-to-date, launches the source code debugger, downloads your application and (optionally) runs to its main() function.

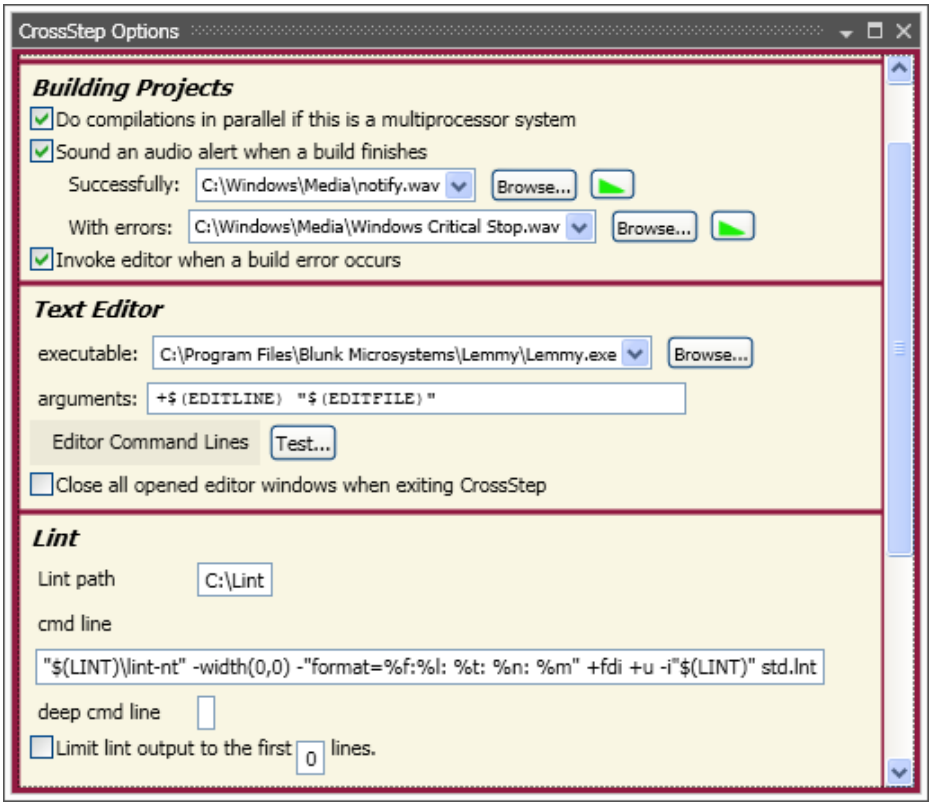


You can use the debugger's step-in, step-over, step-out, etc. commands and breakpoint facility to peruse the sample application. The user interface is fairly intuitive.

Clicking the **Source Files** tab in the Project Overview window displays the application's source files. Double-click on any source file to launch an editor on the selected file. Include files listed in the **Headers** tab can be opened similarly.



To specify the editor CrossStep opens source files with, use the General Options command in the Options menu to open the following dialog:



Provide the path to your preferred editor on the “executable” line. $(EDITLINE) and $(EDITFILE) on the “arguments” line are replaced with the desired line number and file name, respectively, when CrossStep™ launches your editor to, for example, open a source file at the line of a compile error. You must use these macros in the expected format for your editor. Click on “Editor Command Lines” to select among prefilled argument strings for several popular editors.

When finished with the debugger, click on the “End debug session” button to close it. If you make changes to the code and save them, you can use the “debug” button to rebuild the project, download it, and launch the debugger again.

6 Sample Applications

You can experiment with other sample applications in the demo package. The following sample applications are provided, which demonstrate various TargetOS middleware components. Each application contains a “readme.txt” with more details on the application.

bsearch - Loops over all installed file system volumes, measuring the number of binary searches performed in a set time interval.

clients – Cycles through this list of network clients: TCP echo, TCP discard, TCP chargen, UDP echo, and UDP chargen. The server IP address is specified via the TargetOS boot menu.

file_test - Loops over all installed TargetOS file systems, performing file system regression tests that primarily focus on the POSIX API.

graphics – Demonstrates TargetGraphics, Blunk's embedded graphics drawing library. Only supported on targets with a frame buffer based display.

kbench - Measures kernel operations, including context switches in various conditions. The worst case value after multiple measurements is printed to stdout.

server - Runs the following servers: TCP echo, TCP discard, TCP chargen, UDP echo, and UDP discard. The server IP address is specified via the TargetOS boot menu.

shell - Launches the TargetOS command line interface. This is a shell that can be extended with user commands and accesses via UART connections and Telnet.

ui_demo – Demonstrates TargetOS's support for touch panels, button, and joy sticks. Only supported on targets with a frame buffer based display and some UI elements.

webserv - Demonstrates TargetWeb, Blunk's embedded web server, both sending data to the target using browser forms and getting data from the target using custom pages that contain target data. Includes sample HTML pages.

Blunk Microsystems, LLC
Tel: 408/323-1758
Web: www.blunkmicro.com
Email: support@blunkmicro.com