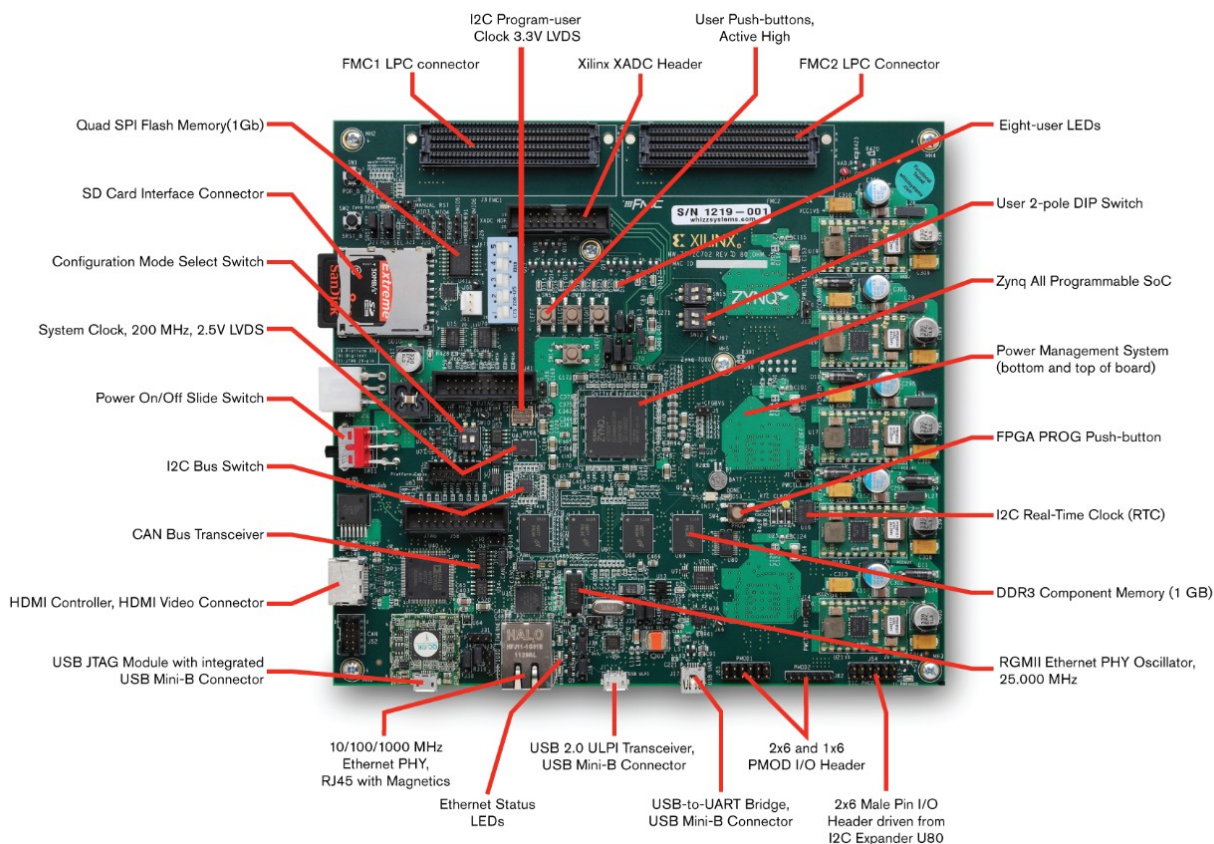


Getting Started with TargetOS on Xilinx's ZC702 Zynq™ Evaluation Kit



1 Introduction

This document covers how to get started with Blunk Microsystems' TargetOS embedded operating system on Xilinx's ZC702 Zynq evaluation kit. It covers the following steps:

- Installing the CrossStep IDE, TargetOS RTOS (lite-kernel), and ZC702 demo package.
- Setting up a new ZC702: cables, jumpers and power.
- Installing the First Stage Boot Loader, FPGA bitstream image, and target monitor (with debug-over-Ethernet support) into the ZC702's SPI NOR flash
- Building Blunk's kbench kernel benchmark sample application and running it on the ZC702.

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.crossstep.com and download the CrossStep installer.

Pick 'ZC702' 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

The ZC702 has a large number of jumpers, but - to start - only the following need to be checked or changed from the default positions:

SW16 – Located in the upper left corner of board. Set SW16.1 to '0' (down) for cascaded JTAG mode. Switches SW16.2-5 control where Zynq's bootrom fetches Blunk's First Stage Boot Loader (FSBL) from. Initially set SW16.2-5 to '0110', to run Blunk's FSBL from the SD card. After the FSBL has been copied to the ZC702's QSPI NOR flash, the setting can be changed to '0010' for a much faster boot time.

J7 - Located near bottom centerline of board. Controls whether the board's J1 USB mini connector supports USB device, host, or OTG mode. To use Blunk's TargetUSB host mode USB stack, these two pins must be jumpered.

J33 - Located near J7 and with similar function. To use J1 in USB host mode, jumper pins 2-3.

J35 - Located near J33 and with similar function. To use J1 in USB host mode, jumper pins 2-3.

With the board configured as above, perform the following steps:

- Connect an Ethernet cable between the network your PC is on and the ZC702's Ethernet connector (P2), located near the bottom left corner of the board.
- Connect a USB cable between your PC and ZC702 mini-USB connector J17, located at the bottom of the board, just right of centerline.
- Optionally, to use TargetUSB, connect a USB flash drive to mini-USB connector J1, located at the bottom of the board, just left of centerline.
- With power switch J11 in the off (down) position, connect the ZC702 power supply to connector J60.
- Turn the board on via switch J11. This should add a UART to your PC. If the USB-UART is not automatically recognized, its driver can be downloaded from www.silabs.com. Find the appropriate driver by searching for "VCP drivers CP210x". After the driver is installed, you should verify that your PC has a new USB UART channel.
- 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 the ZC702, you need to install several files into the board's QSPI NOR flash. That is the topic of the next section.

4 Preparing ZC702's SPI NOR Flash

The First Stage Boot Loader (FSBL) is the first code to run post-reset after the Zynq's bootrom. Its purpose is to configure the FPGA, initialize SDRAM, and load application code into SDRAM.

To support this, three files must be stored in the ZC702 QSPI NOR flash: 1) the FSBL image, which the Zynq bootrom copies to on-chip memory and starts, 2) the FPGA bitstream file, and 3) application code.

Blunk's FSBL uses TFTP to install these files into the QSPI NOR flash, but you need the FSBL running on your board before it can do this. So, although later you'll have the ZC702 boot from QSPI NOR flash, initially you need it to boot from an SD card and that is how SW16 was configured above.

The eval package includes 'BOOT.BIN', which is Blunk's ZC702 FSBL in the special format recognized by Zynq's bootrom, in the following subdirectory within the TargetOS directory:

```
bsps\arm\zc702_demo\fsbl\bootgen\
```

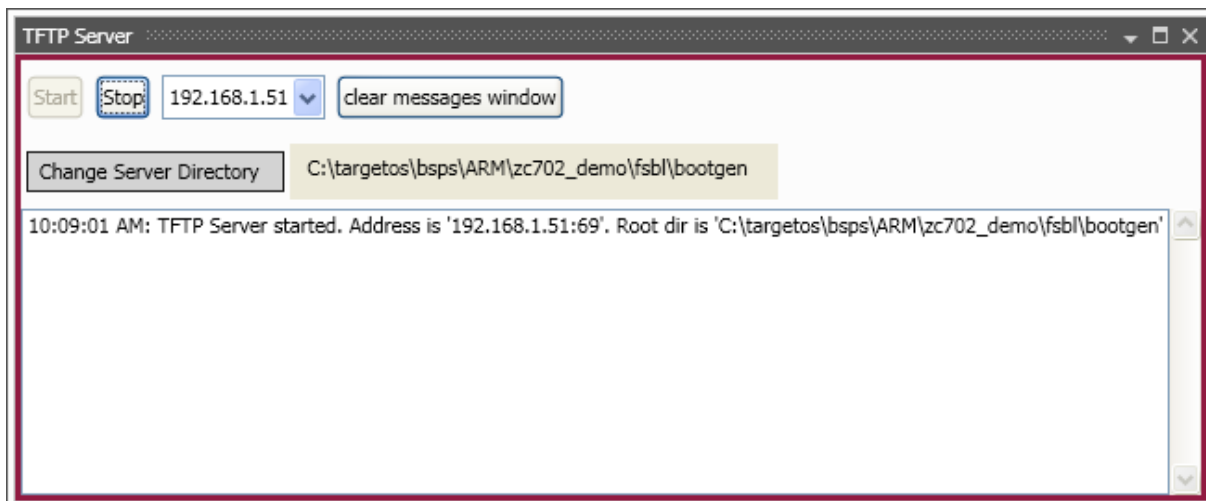
To run the FSBL on your board, copy this file to an SD card, insert the card into your ZC702 (connector J64, top left corner of the board), and press the soft reset switch (the black button labeled "SRST_B/SW2"). You should see the following on your terminal program:

```
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 containing 'BOOT.BIN', the file you copied onto an SD 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. Usually this would be the IP address of your PC.

Install two files, the FSBL 'BOOT.BIN' and the FPGA bitstream file, on the ZC702's SPI NOR flash using the following steps:

- Enter '3' to update a flash partition and then '0' to copy 'BOOT.BIN' to the first partition.
- Enter '3' to update a flash partition and then "3" to copy 'system.bit' to its partition.

The demo package 'system.bit' file is an example Xilinx design. You can customize the name used for the bitstream file by editing the ZC702's non-volatile parameters ('9' in the FSBL menu) and changing the value of the 'BITSTREAM' entry from its "system.bit" default.

Now that the FSBL is installed in the ZC702's QSPI NOR flash, test booting from flash memory by switching power off, changing SW16.2-5 to '0010' 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 require you to disconnect and re-connect after power cycling USB UART connections. Try that and then press the ZC702's soft reset button if the above text doesn't appear at first. Note the "QSPI boot" text versus the prior "SD boot". You should also see the ZC702's green "DONE" LED lit, showing the FSBL configured the FPGA using the installed bitstream file.

The FSBL stays out of view as much as possible. If an application file is installed in flash, the FSBL silently starts it at each reset. But currently there is no application it can load and start. Blunk's bootrom application, which runs the TargetMon debug-over-Ethernet target monitor, is at the TargetOS subdirectory path below:

```
bsps/arm/zc702_demo/bootrom/app.elf
```

To install bootrom on your ZC702's SPI NOR flash, change your TFTP server to the directory holding 'app.elf', enter '3', enter '1', and then let the installation finish. Now if you press the soft reset button, your terminal shows the bootrom application start, which should look similar to this.

```
TargetOS, Release 2015.0. Built at 10/29/2015 12:16:24 PM.
Copyright 2015, Blunk Microsystems. All rights reserved.
-----
Starting TargetMon
SDRAM size = 255MB
CPU0 revision r3p0, PS version 1.0
Thu Oct 29 23:16:07 2015
Hostname: zc702
IPv4 Default gateway: 192.168.1.1
GEM0 IP addr/mask: DHCP
GEM0 address: 00:00:00:00:2E:E2
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=100M duplex=full
GEM0: DHCP IPv4 address 192.168.1.141
```

Return to the FSBL menu by holding down SW14 (a black button near SW16, toward center) while depressing and releasing the soft reset button. This takes you to the FSBL menu, skipping the application start. It also skips FPGA configuration, which you can manually invoke by entering '5' and then '0' in the FSBL menu. Enter '0' to view a summary of FSBL status, as below:

```
CPU revision r3p0, PS version 1.0
GEM0 IP addr/mask: DHCP
GEM0 address: 00:00:00:00:2E:E2
Configured PL
Design Name: system.ncd;HW_TIMEOUT=FALSE;UserID=0xFFFFFFFF
Zynq Name: 7z020clg484
Design Date: 2014/02/18
Design Time: 17:27:37
Image Length: 4045564
FSBL partition (3 blks = 192KB) is valid, 44.8% full
CPU0 partition (16 blks = 1MB) is valid, 27.9% full
CPU1 partition (16 blks = 1MB) is erased
FPGA partition (62 blks = 3.9MB) is valid, 99.6% full
Free OCM heap = 110160B
```

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

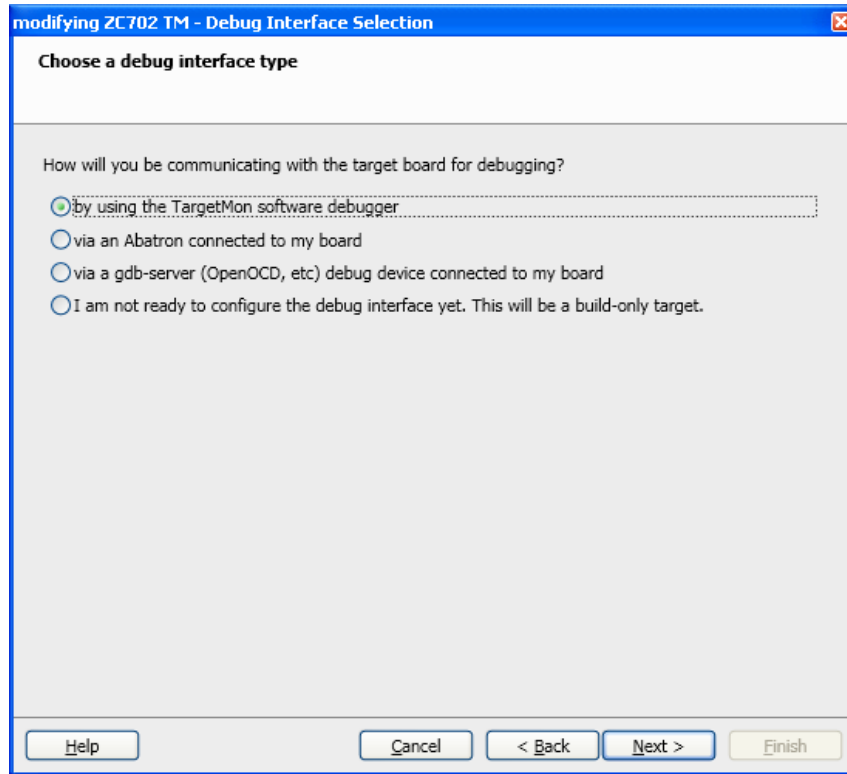
5 Running kbench

Now you are ready to build and run the kbench sample application. kbench measures TargetOS kernel performance. If you specified the kbench application during the CrossStep installation, it will already be loaded. Otherwise, use the Project/Open dialog to browse to and select this file:

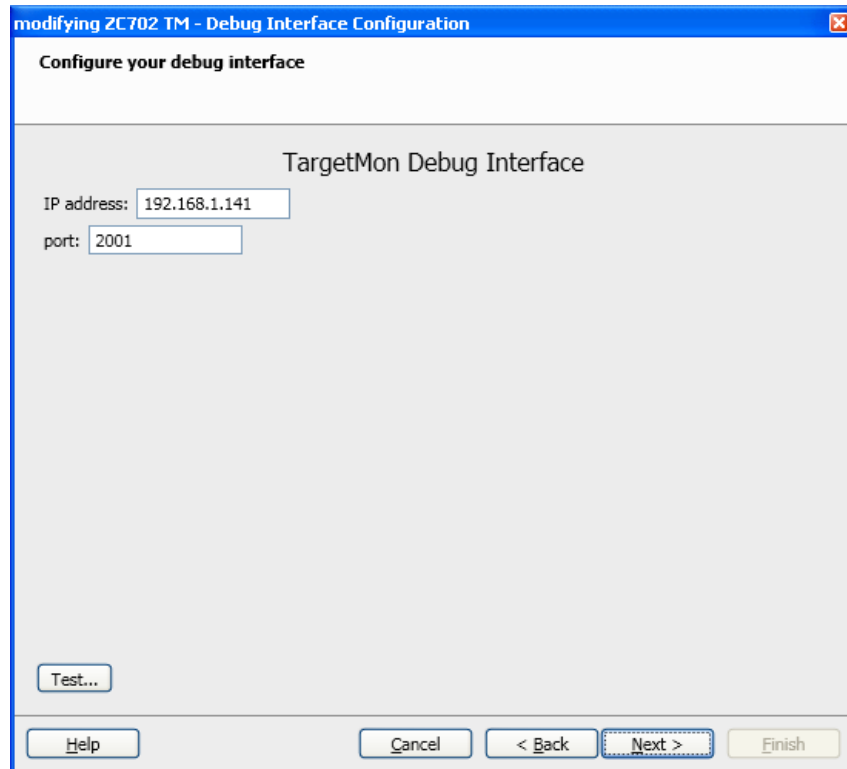
```
apps\kbench\kbench.gbp
```

To use a TargetMon debug connection, you need to modify your CrossStep target to use the IP address of your ZC702. In CrossStep's Options menu, select 'Manage Targets' 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 "zc702 - Xilinx Zynq Evaluation Kit (demo BSP)" as your target board, accept the GNU toolchain settings, and when you get to the panel below choose 'TargetMon', as shown below.




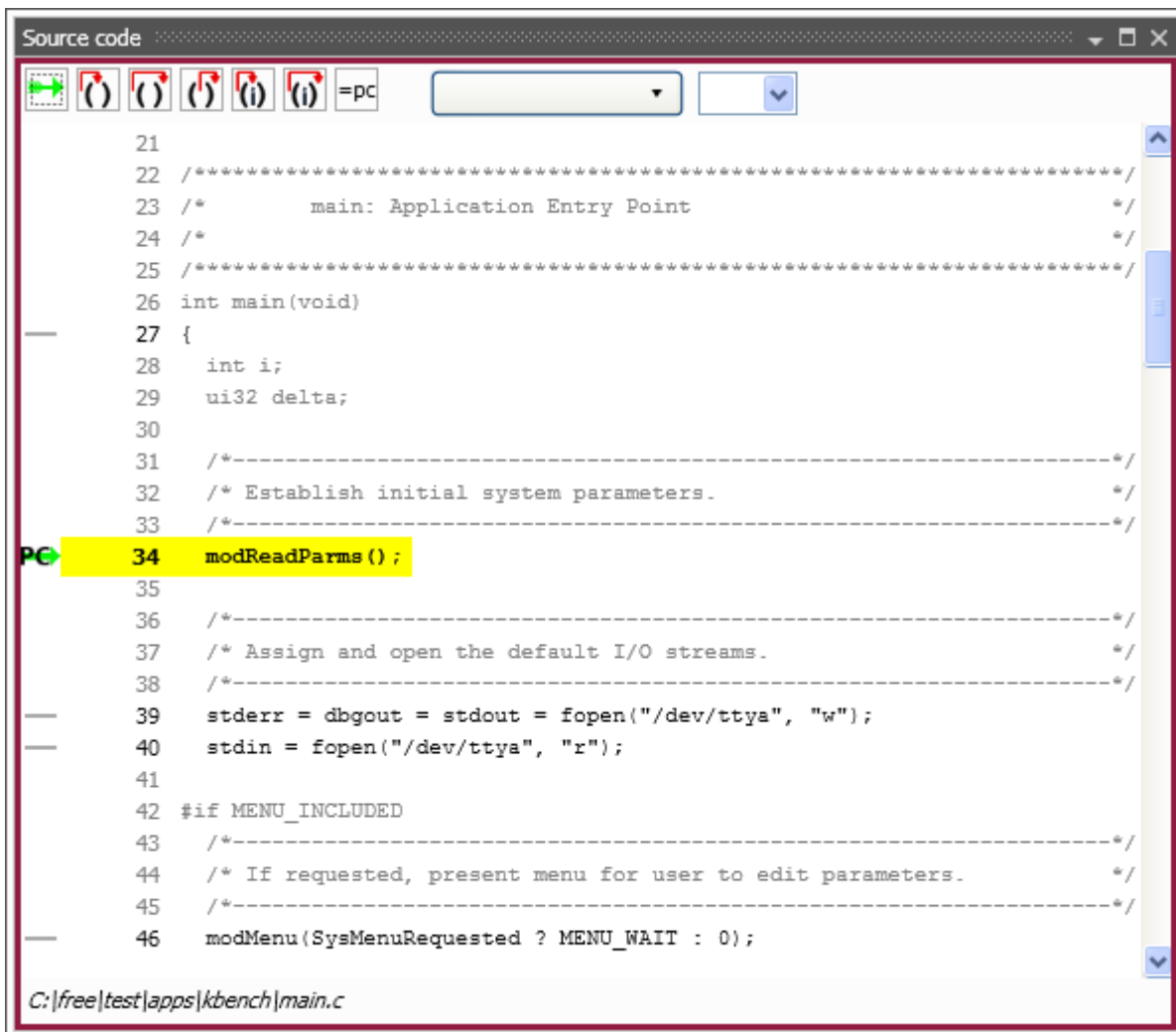
Enter your ZC702's IP address in the next panel, getting the address from the terminal window that shows the bootrom application's start-up. The IP address is on the last line. Leave the port number as 2001.



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 with the hammer icon  in CrossStep’s upper-left corner. 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, 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.



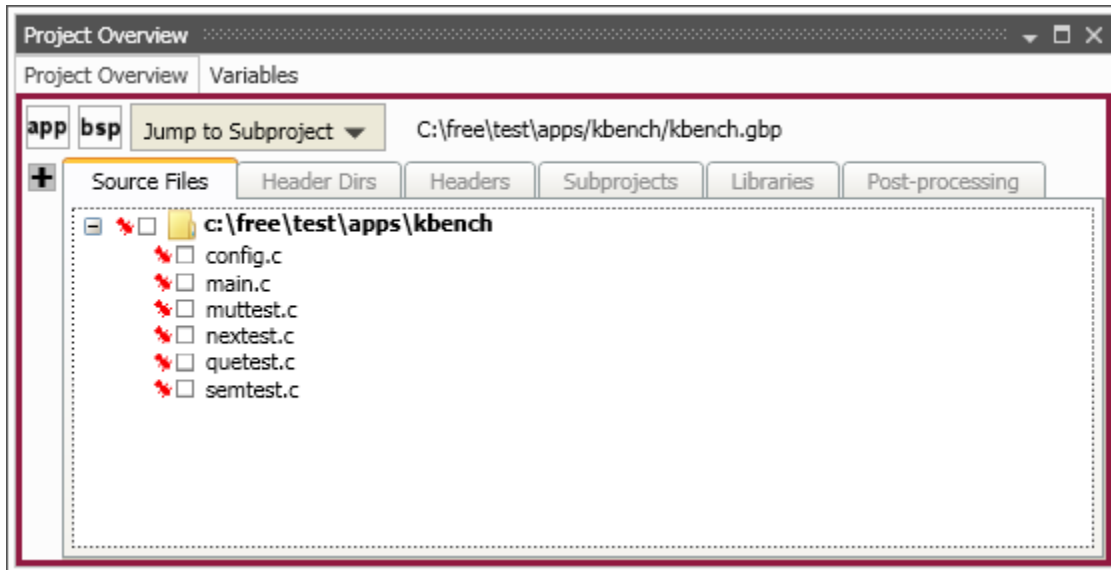
```

Source code
21
22 /*****
23 /*      main: Application Entry Point      */
24 /*                                          */
25 /*****
26 int main(void)
27 {
28     int i;
29     ui32 delta;
30
31     /*-----*/
32     /* Establish initial system parameters. */
33     /*-----*/
PC → 34     modReadParms ();
35
36     /*-----*/
37     /* Assign and open the default I/O streams. */
38     /*-----*/
39     stderr = dbgout = stdout = fopen("/dev/ttya", "w");
40     stdin = fopen("/dev/ttya", "r");
41
42     #if MENU_INCLUDED
43     /*-----*/
44     /* If requested, present menu for user to edit parameters. */
45     /*-----*/
46     modMenu(SysMenuRequested ? MENU_WAIT : 0);
C:\free\test\apps\kbench\main.c

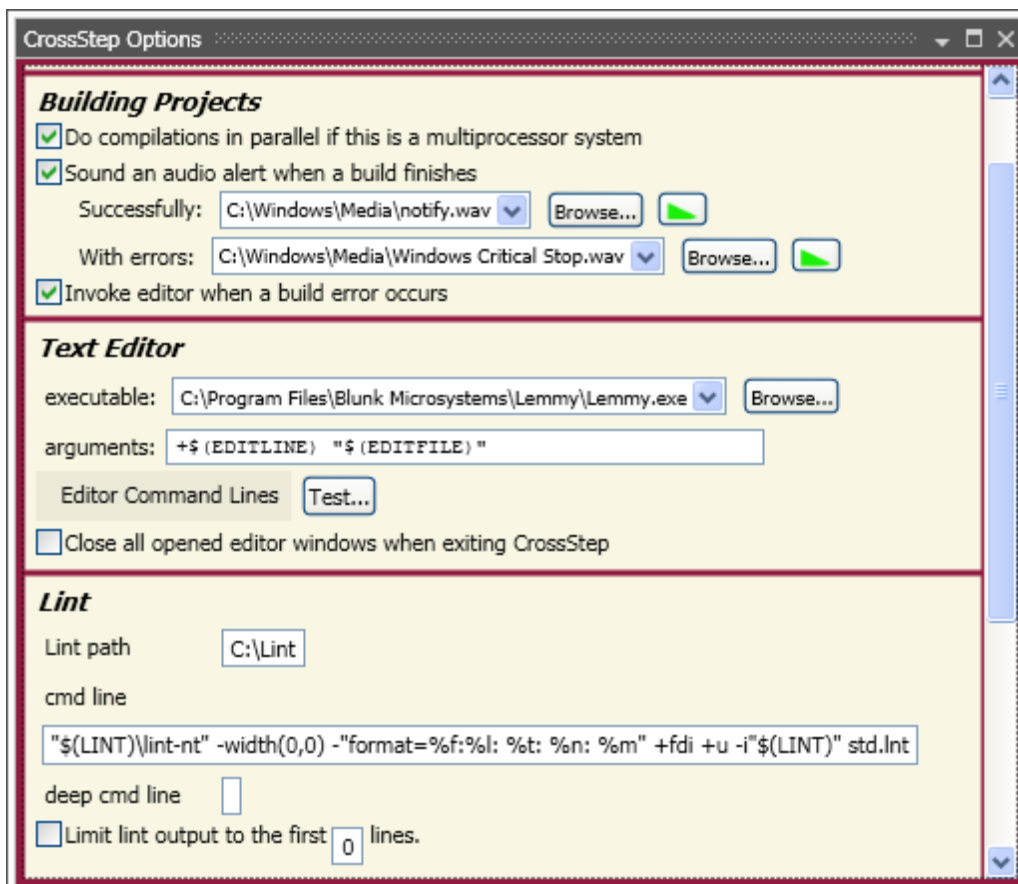
```

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.



In the Options menu, select 'General Options' to specify which editor CrossStep uses to open source files:



Enter 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 accessed 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 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