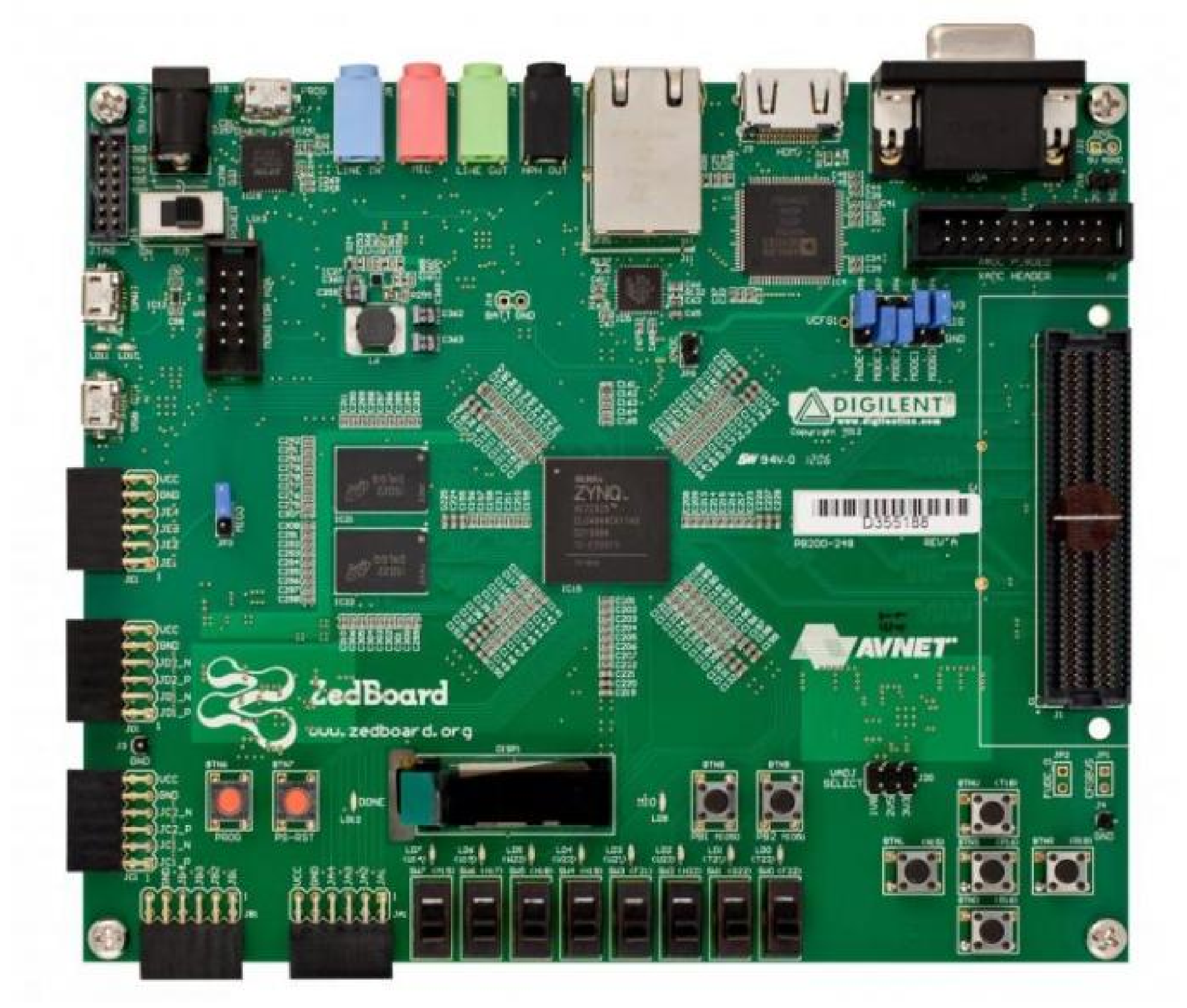
# Getting Started with TargetOS on the ZedBoard



## 1 Introduction

This document covers how to get started with Blunk Microsystems' TargetOS embedded operating system on the ZedBoard. It covers the following steps:

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

## 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 'ZedBoard' 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 ZedBoard has a small number of jumpers. Only the following need to be checked or changed from the default positions:

JP7 – Located at the right end in a row of jumpers in the upper right quadrant of board. Connect JP7 (MIO2) to 'GND' (down) for cascaded JTAG mode.

JP8-JP10 – Located next to JP7, these jumpers control where Zynq's bootrom fetches the First Stage Boot Loader (FSBL) from. In order to load Blunk's FSBL from an SD card, connect JP8 (MIO3) to 'GND' (down), JP9 (MIO4) to '3V3' (up), and JP10 (MIO5) to '3V3' (up). After the FSBL is copied to the ZedBoard's QSPI NOR flash, JP9 can be changed to 'GND' (down) to load the FSBL from QSPI NOR flash for a much faster boot time.

JP11 – Located next to JP8-JP10, Connect JP11 (MIO6) to 'GND' (down) for internal PS PLLs.

JP3 - Located on the board's left edge, between two micro-USB connectors. Controls whether USB micro B connector J13 (labeled "USB OTG") supports USB device, host, or OTG mode. To use Blunk's TargetUSB host mode stack, connect pins 1 and 2.

JP2 - Located on the board's left edge, below JP3. To use Blunk's TargetUSB host mode stack, connect pins 1 and 2.

JP6 - Located on the board's left edge, below JP2, and labeled "MIO0". To use the SD card interface, connect pins 1 and 2.

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

- Connect an Ethernet cable between the network your PC is on and the ZedBoard's Ethernet connector (J11), located at the top of the board near centerline.
- Connect a USB cable between your PC and USB micro B connector J14 (labeled "UART"), the top USB connector on the board's left side. This serves as a USB-UART connection.
- To use TargetUSB, connect a USB flash drive to USB micro B connector J13 (labeled "USB OTG"), the bottom USB connector on the board's left side.
- With power switch SW8 in the OFF position, connect the ZedBoard's 12v power supply to barrel jack J20.
- Turn on power switch SW8. This should add a UART channel to your PC. The ZedBoard has a USB-UART bridge based on Cypress's CY7C64225. If the USB-UART is not automatically recognized, its driver can be downloaded from www.cypress.com/?rID=63794. 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 ZedBoard, you need to install several files into its QSPI NOR flash. That is the topic of the next section.

## 4 Preparing ZedBoard'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 ZedBoard's 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 to run the FSBL on your board before it can do this. So, although later you'll have the ZedBoard boot from QSPI NOR flash, initially you need it to boot from an SD card and that is why JP9 was configured for this above.

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

```
   bsps\arm\zedboard_demo\fsbl\bootgen\
```

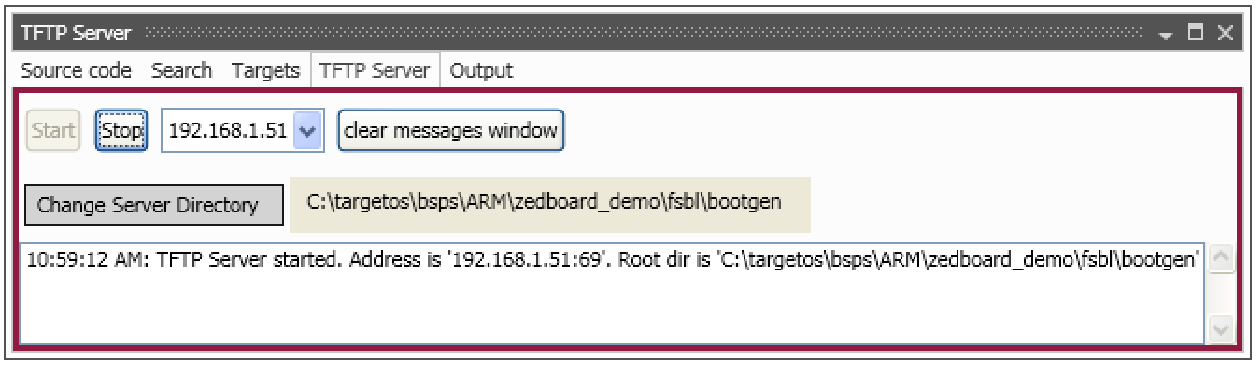
To run the FSBL, copy this file to an SD card, insert the card into your ZedBoard (connector J12, on the bottom of the board, on the right edge), and press the soft reset switch (the black button labeled "SRST_B/SW2"). You should see the following on your terminal program:

```
 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 the SD card. Using CrossStep's TFTP server (accessed via the Tools menu), this looks like the following:



```
10:59:12 AM: TFTP Server started. Address is '192.168.1.51:69'. Root dir is 'C:\targetos\bsps\ARM\zedboard_demo\fsbl\bootgen'
```

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 ZedBoard’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 design by ZedBoard.org. You can customize the name used for the bitstream file by editing the ZedBoard’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 ZedBoard’s QSPI NOR flash, test booting from flash memory by switching power off, changing jumper JP9 (MIO4) to ‘GND’ (down) and then 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 ZedBoard’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 blue “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 after each reset. But currently there is no application it can load and start. Blunk’s bootrom application contains the TargetMon debug-over-Ethernet target monitor and is located in the ‘app.elf’ file in the TargetOS subdirectory below:

```
bsps\arm\zedboard_demo\bootrom\
```

To install bootrom in the ZedBoard’s SPI NOR flash, change your TFTP server to the directory shown above, enter ‘3’ and then ‘1’ in the FSBL menu. 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 2015.1. Built at 12/6/2015 6:30:51 AM.
Copyright 2015, Blunk Microsystems. All rights reserved.
---------------------------------------------------------
```

```
Starting TargetMon
SDRAM size = 255MB
CPU0 revision r3p0, PS version 1.0
Hostname: zedboard
IPv4 Default gateway: 192.168.1.1
GEM0 IP addr/mask: DHCP
GEM0 address: 00:00:00:02:2E:E2
Starting FTP server for IPv4
Starting Telnet server for IPv4
Starting TFTP server for IPv4
Using: 0x00100020 - 0x00200000 (cold boot)
---------------------------------------------------------
Press 'M' in 3 seconds to modify these settings: .Starting..
GEM0: link=up speed=100M duplex=full
GEM0: DHCP IPv4 address 192.168.1.131
```

Return to the FSBL menu by holding down BTN8 (a black button just to the right of the OLED display near the bottom of the board) 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 MAC address: 00:00:00:02:2E:E2
Configured PL
Design Name: system_stub_routed.ncd;UserID=0xFFFFFFFF
Zynq Name: 7z020clg484
Design Date: 2012/08/09
Design Time: 16:23:17
Image Length: 4045564
FSBL partition (3 blks = 192KB) is valid, 42.1% full
CPU0 partition (16 blks = 1MB) is valid, 27.4% full
CPU1 partition (16 blks = 1MB) is erased
FPGA partition (62 blks = 3.9MB) is valid, 99.6% full
Free OCM heap = 115760B
```

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

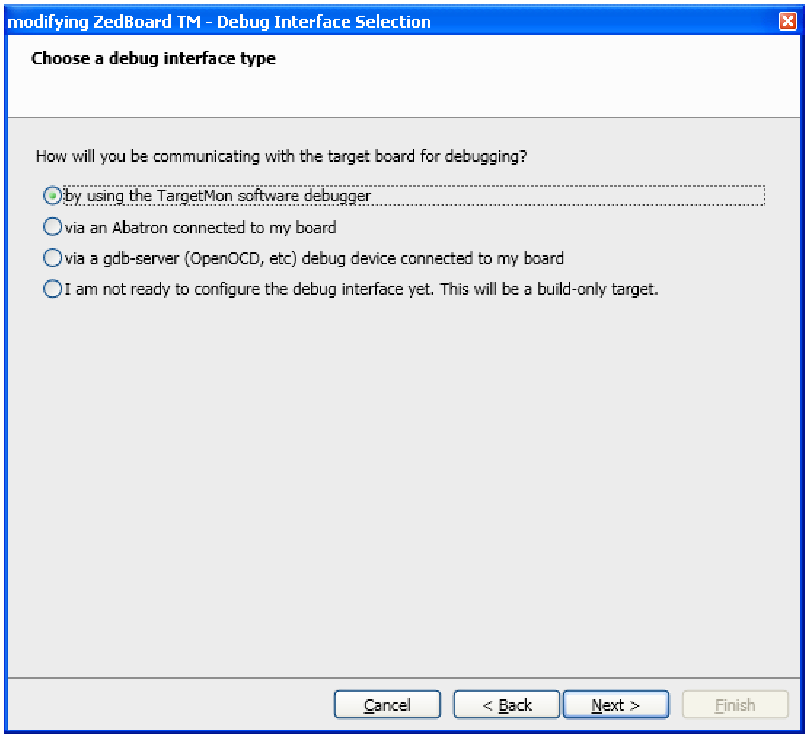
## 5 Running kbench

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 give CrossStep the IP address of your ZedBoard. 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 “zedboard – ZedBoard.org Zynq Development Kit (demo BSP)” as your BSP, accept the GNU toolchain settings, and when you get to the panel below choose ‘TargetMon’, as shown.

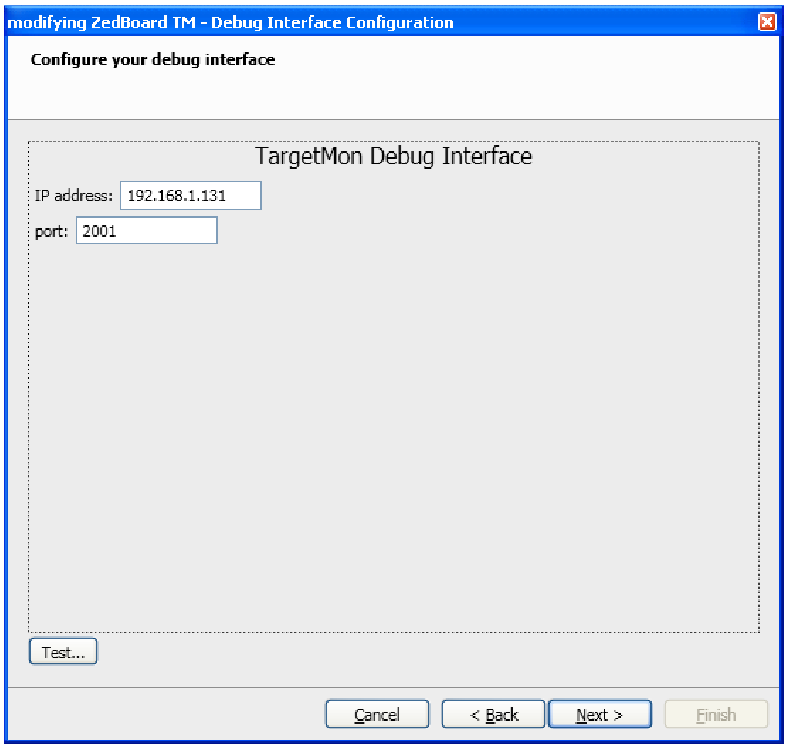


Choose a debug interface type

How will you be communicating with the target board for debugging?

[✓ by using the TargetMon software debugger]
via an Abatron connected to my board
via a gdb-server (OpenOCD, etc) debug device connected to my board
I am not ready to configure the debug interface yet. This will be a build-only target.

Enter your ZedBoard’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.



Configure your debug interface

TargetMon Debug Interface

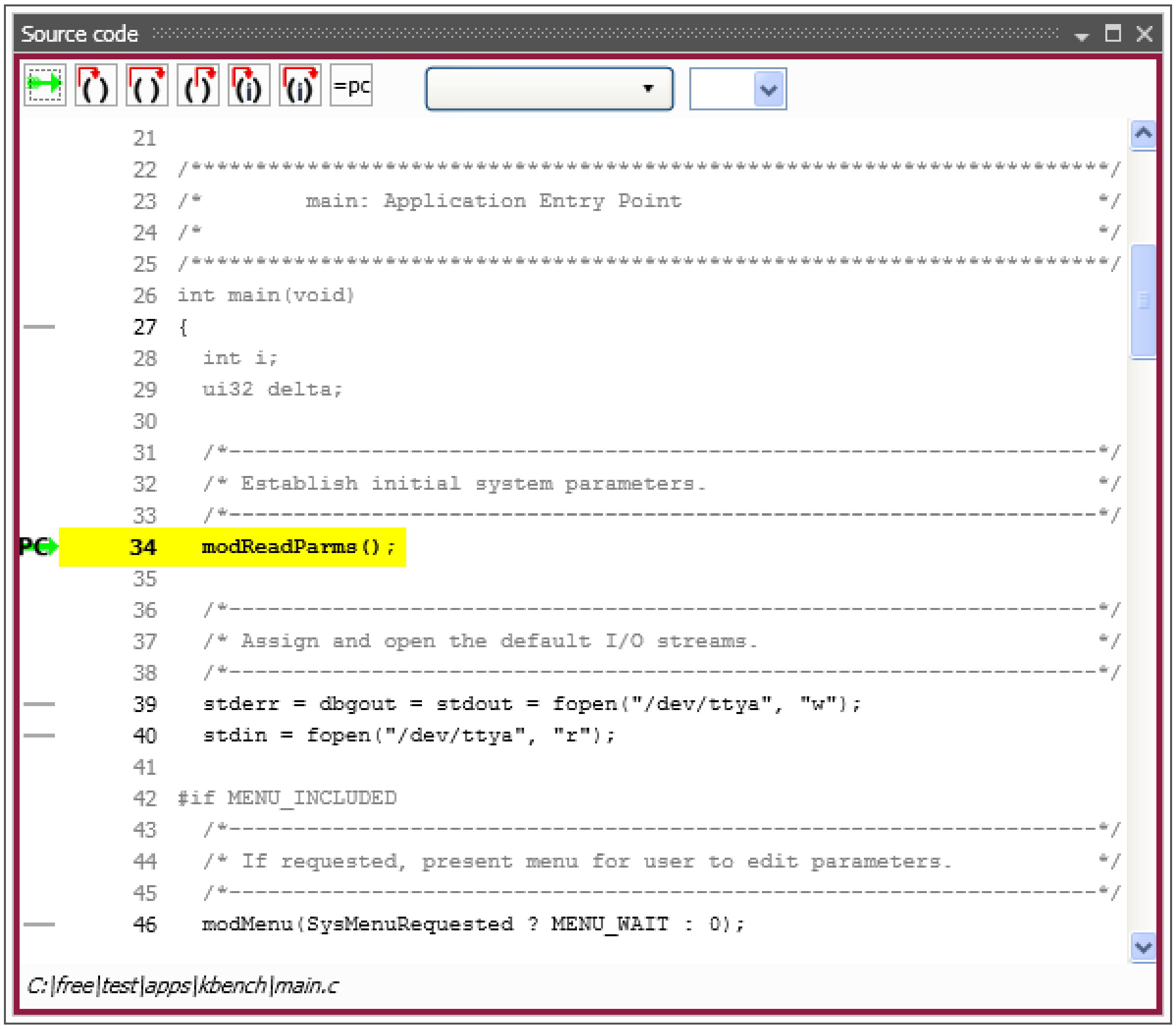
IP address: 192.168.1.131

port: 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.

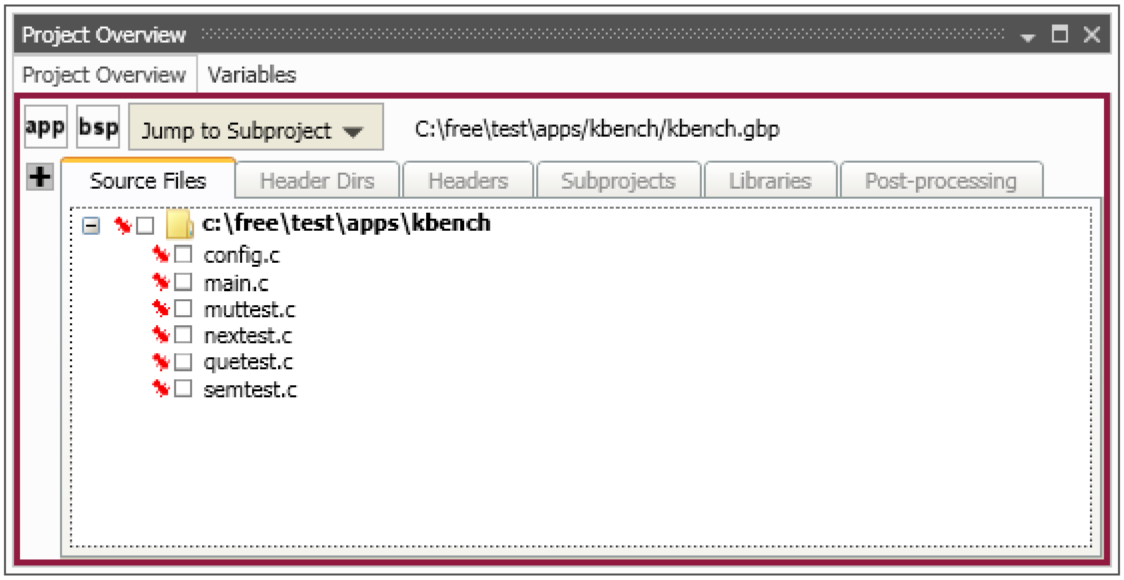


```

/**************************************************************************/
/*         main: Application Entry Point                                 */
/*                                                                        */
/**************************************************************************/
int main(void)
{
  int i;
  ui32 delta;

  /*----------------------------------------------------------------------*/
  /* Establish initial system parameters.                                 */
  /*----------------------------------------------------------------------*/
  modReadParms();

  /*----------------------------------------------------------------------*/
  /* Assign and open the default I/O streams.                             */
  /*----------------------------------------------------------------------*/
  stderr = dbgout = stdout = fopen("/dev/ttya", "w");
  stdin = fopen("/dev/ttya", "r");

#if MENU_INCLUDED
  /*----------------------------------------------------------------------*/
  /* If requested, present menu for user to edit parameters.              */
  /*----------------------------------------------------------------------*/
  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.



Project Overview

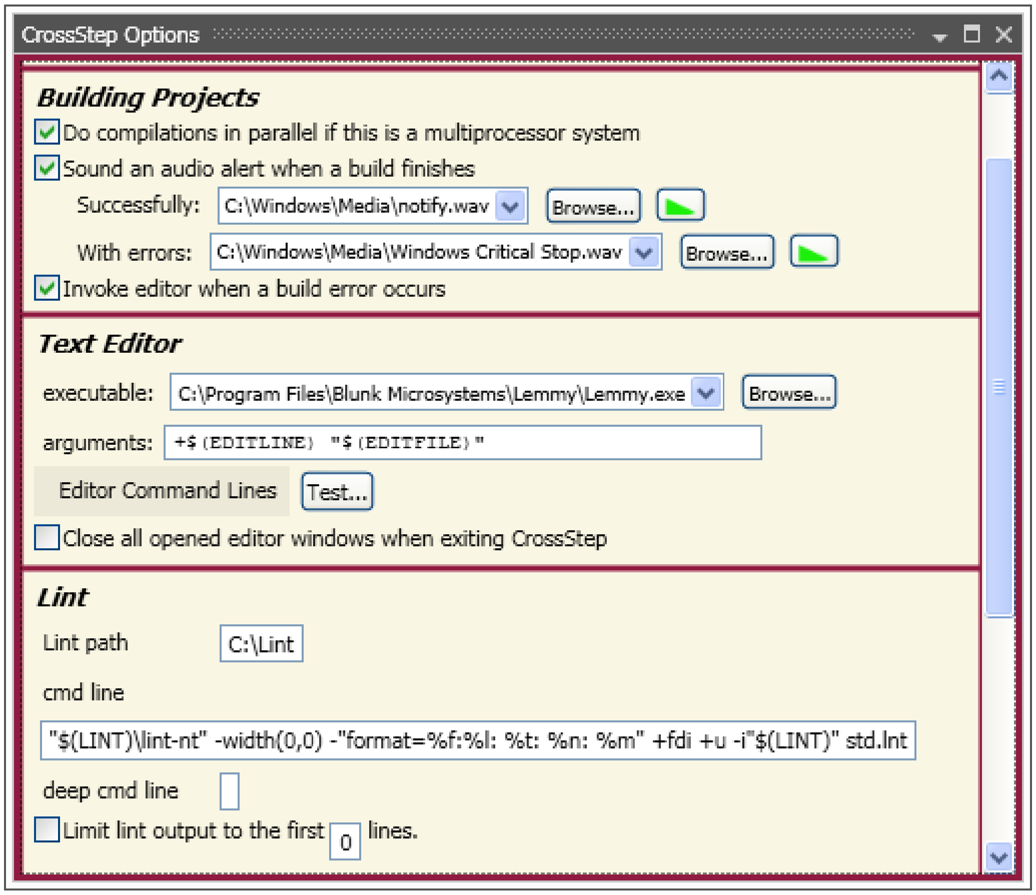
Project Overview | Variables

app | bsp | Jump to Subproject | C:\free\test\apps/kbench/kbench.gbp

Source Files | Header Dirs | Headers | Subprojects | Libraries | Post-processing

```
c:\free\test\apps\kbench
    config.c
    main.c
    muttest.c
    nextest.c
    quetest.c
    semtest.c
```

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



CrossStep Options

***Building Projects***

[✓] Do compilations in parallel if this is a multiprocessor system

[✓] Sound an audio alert when a build finishes

Successfully: C:\Windows\Media\notify.wav | Browse...

With errors: C:\Windows\Media\Windows Critical Stop.wav | Browse...

[✓] Invoke editor when a build error occurs

***Text Editor***

executable: C:\Program Files\Blunk Microsystems\Lemmy\Lemmy.exe | Browse...

arguments: `+$(EDITLINE) "$(EDITFILE)"`

Editor Command Lines | Test...

[ ] Close all opened editor windows when exiting CrossStep

***Lint***

Lint path: C:\Lint

cmd line

```
"$(LINT)\lint-nt" -width(0,0) -"format=%f:%l: %t: %n: %m" +fdi +u -i"$(LINT)" std.lnt
```

deep cmd line

[ ] Limit lint output to the first 0 lines.

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