µEZ™ Software Quickstart Guide

FDI Future Designs, Inc.

Your Development Partner

2702 Triana Boulevard SW, Huntsville, AL 35805

Table of Contents

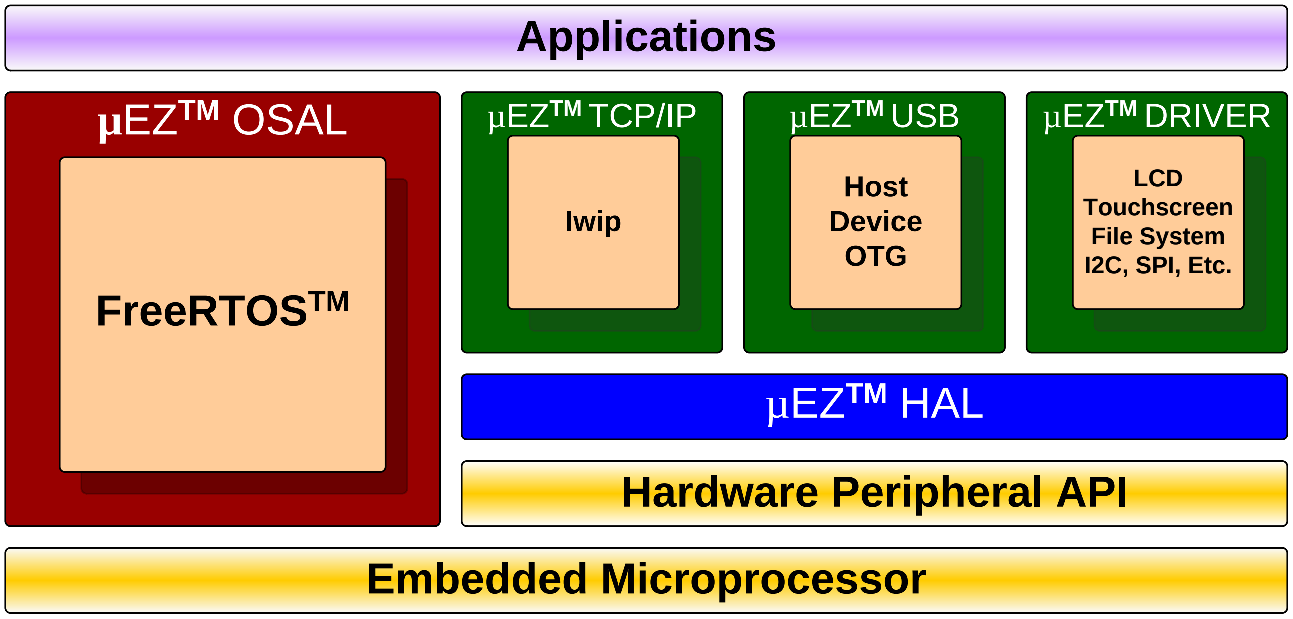
FDI PN: MA00015
Revision: Rev 1.4, 3/4/2010 5:00:00 PM
Printed in the United States of America

1. Introduction

µEZ™ takes its name from the Muses of Greek mythology. A Muse was a goddess who inspired the creation process for the arts and sciences. Like its ancient Greek namesake, the **µEZ™** platform inspires rapid development by supplying customers with an extensive library of open source software, drivers, and processor support - all under a common framework. **µEZ™** development works on the premise of ”design once, reuse many times”. This provides an open source standard for embedded developers to build upon and support. **µEZ™** allows companies to focus on innovation and on their own value-added applications while minimizing development time and maximizing software reuse.

The diagram below shows a typical embedded application stack. **µEZ™** has three primary categories of components that help simplify embedded application development:

1. **Operating System Abstraction Layer (µEZ™ OSAL)**
2. **Sub-system drivers (µEZ™ TCP/IP, µEZ™ USB, µEZ™ Driver)**
3. **Hardware Abstraction Layer (µEZ™ HAL)**



The selection of an RTOS can be one of the most daunting aspects of an embedded system development. With **µEZ™** the primary features of common multi-tasking operating systems are abstracted, thus easing the transition to an open source or low-cost RTOS. The **µEZ™** OSAL provides applications access to the following features in an OS-independent fashion:

- Pre-emptive multitasking
- Stack overflow detection
- Unlimited number of tasks
- Queues
- Semaphores (binary, counting, mutex)

The **µEZ™** sub-system drivers utilize the OSAL functions to provide protected access to the processor peripherals. The sub-system driver API functions are typically protocol layer interfaces (TCP/IP, USB, etc) designed as high-level access routines such as open, close, read, write, etc. where possible.

The HAL functions provide single-threaded unprotected access to the processor peripherals. Customers can use the **µEZ™** HAL routines provided by FDI or they can write their own. The HAL routines provide for RTOS/**µEZ™** independence and allow portability within a family of processors.

µEZ™ is ideally suited for Embedded Systems with standard features such as:

- Processor and Platform BSPs
  (Board Support Packages)
- Real Time Operating System (RTOS)
- Memory Management
- NAND/NOR Flash
- SDRAM and DDR Memory
- TCP/IP stack
- USB Device/Host Libraries
- Mass Storage Devices
- LCD Displays with Touch Screen
- Input / Output Devices

2. Downloading uEZ™

Start by downloading the latest version of uEZ™ from https://sourceforge.net/projects/uez/. Unzip to a working folder. In this document we will use a simple directory structure of /uEZ but the user is free to modify this as desired.

The uEZ™ file directory structure should be as follows:

Directory	**Description**
/Build	Projects/makefiles
/Include	uEZ™ system files and Config.h
/Include/Device	Device Driver class definitions.
/Include/HAL	Hardware Abstraction Layer (HAL) driver class definitions.
/Include/Types	Common data types used by both HAL and Device Drivers.
/Source	Source code
/Source/App	User application source code and demos
/Source/BSP	BSP generic startup code
/Source/Devices/<category>/<manufacturer>/<device>	Device specific code organized by category (I2C, SSP, etc.), manufacturer, and specific device.
/Source/Library/<category>/<package>	Various support libraries organized by category (graphics, file system, etc.) and package name.
/Source/Platform/<manufacturer>/<platform>	Platforms/boards code organized by manufacturer and specific platform build.
/Source/Processor/<manufacturer>/<processor>	Processor specific code in separate directories organized by manufacturer and specific processor.
/Source/RTOS/<RTOS>/	RTOS source code in separate directories
/Source/uEZSystem	uEZ™ System Core routines

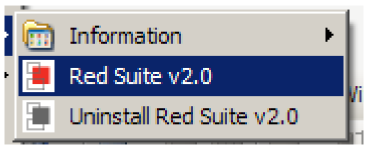
3. Project Configuration

uEZ™ uses a simple one project configuration. Depending on the compiler tools, use one of the following subsections.

Code Red 2.0 Project Configuration

Check Code Red Version

uEZ™ can be built using Red Suite v2.0, or later. Confirm the version by looking at the version in the Start menu.

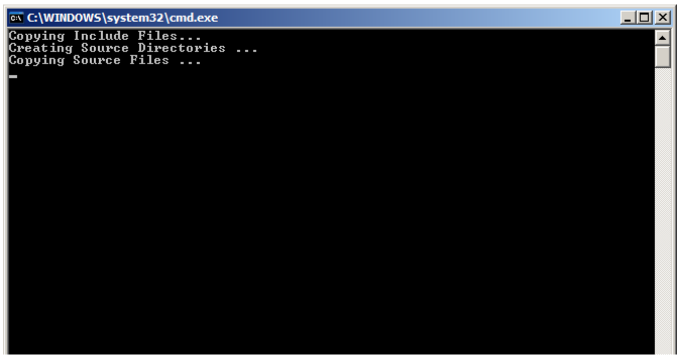


Preparing the uEZ™ Source Code

Download the uEZ™ v1.04 source code from http://www.sourceforge.net/projects/uez. Unzip the file to where you will be working. It should create a folder called /UEZ_SRC.

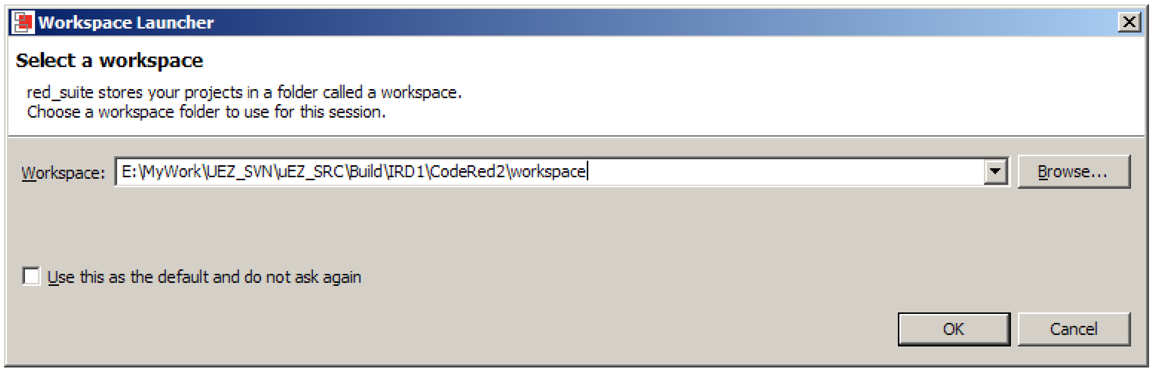
Find the project you want to build in the Build project. For example, when building the demo source code for the IRD 1.0, you will need to go to /UEZ_SRC/Build/IRD1/CodeRed2/workspace/IRDDemo

Code Red projects are prepared by running a batch file to prepare the directories needed for compiling. Go to /UEZ_SRC/Build/IRD1/CodeRed2/workspace/IRDDemo and double click on Setup.bat. You should see the following output on the screen.



```
C:\WINDOWS\system32\cmd.exe
Copying Include Files...
Creating Source Directories ...
Copying Source Files ...
_
```

The files are now ready to access. Run Red Suite 2.0 and tell it to go to the workspace at /UEZ_SRC/Build/IRD1/CodeRed2/workspace. For example,



Workspace Launcher

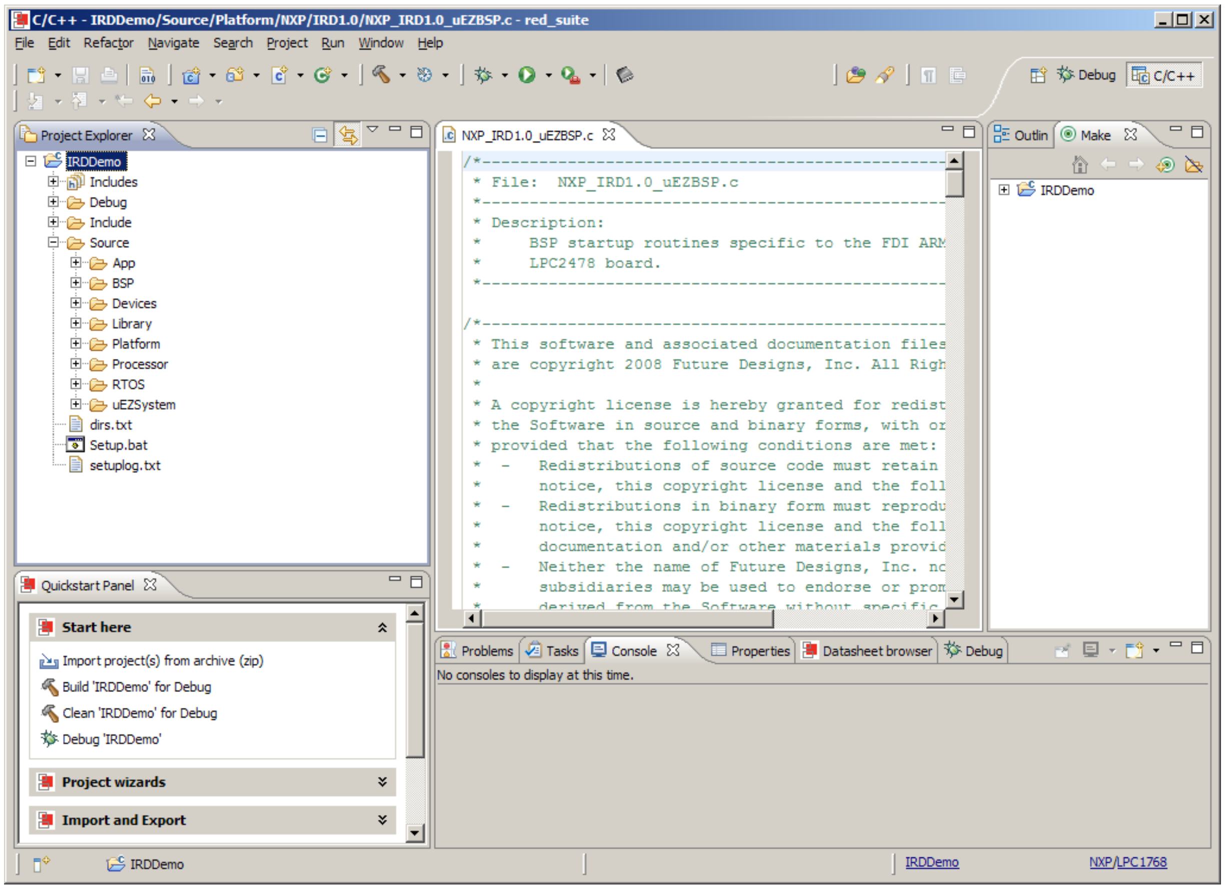
Select a workspace

red_suite stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: E:\MyWork\UEZ_SVN\uEZ_SRC\Build\IRD1\CodeRed2\workspace

☐ Use this as the default and do not ask again

The following screen should appear:


C/C++ - IRDDemo/Source/Platform/NXP/IRD1.0/NXP_IRD1.0_uEZBSP.c - red_suite
File Edit Refactor Navigate Search Project Run Window Help
Debug C/C++
Project Explorer
IRDDemo
Includes
Debug
Include
Source
App
BSP
Devices
Library
Platform
Processor
RTOS
uEZSystem
dirs.txt
Setup.bat
setuplog.txt
Quickstart Panel
Start here
Import project(s) from archive (zip)
Build 'IRDDemo' for Debug
Clean 'IRDDemo' for Debug
Debug 'IRDDemo'
Project wizards
Import and Export
NXP_IRD1.0_uEZBSP.c
Outlin Make
IRDDemo
Problems Tasks Console Properties Datasheet browser Debug
IRDDemo
IRDDemo
NXP/LPC1768

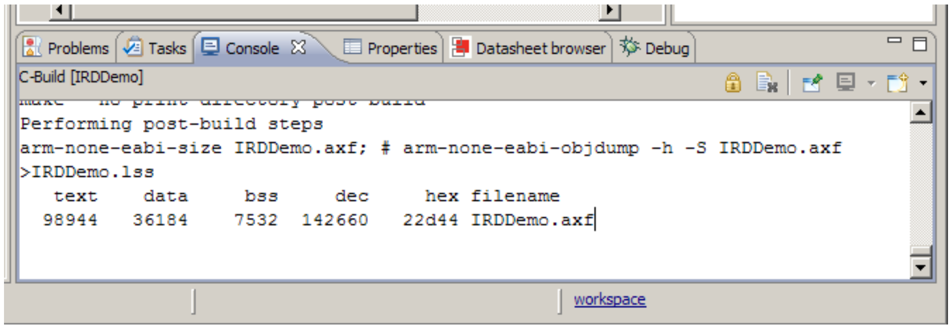
```
/*-------------------------------------------------
 * File:  NXP_IRD1.0_uEZBSP.c
 *-------------------------------------------------
 * Description:
 *      BSP startup routines specific to the FDI ARM
 *      LPC2478 board.
 *-------------------------------------------------

/*-------------------------------------------------
 * This software and associated documentation files
 * are copyright 2008 Future Designs, Inc. All Righ
 *
 * A copyright license is hereby granted for redist
 * the Software in source and binary forms, with or
 * provided that the following conditions are met:
 *  -   Redistributions of source code must retain
 *      notice, this copyright license and the foll
 *  -   Redistributions in binary form must reprodu
 *      notice, this copyright license and the foll
 *      documentation and/or other materials provid
 *  -   Neither the name of Future Designs, Inc. nc
 *      subsidiaries may be used to endorse or prom
 *      derived from the Software without specific
```

```
No consoles to display at this time.
```

Click on IRDDemo in the top left (under Project Explorer) and press F5 to refresh the project. Wait until the indexing notification at the bottom completes.

To compile, press CTRL-B. If everything goes well, the Console should have the following output:

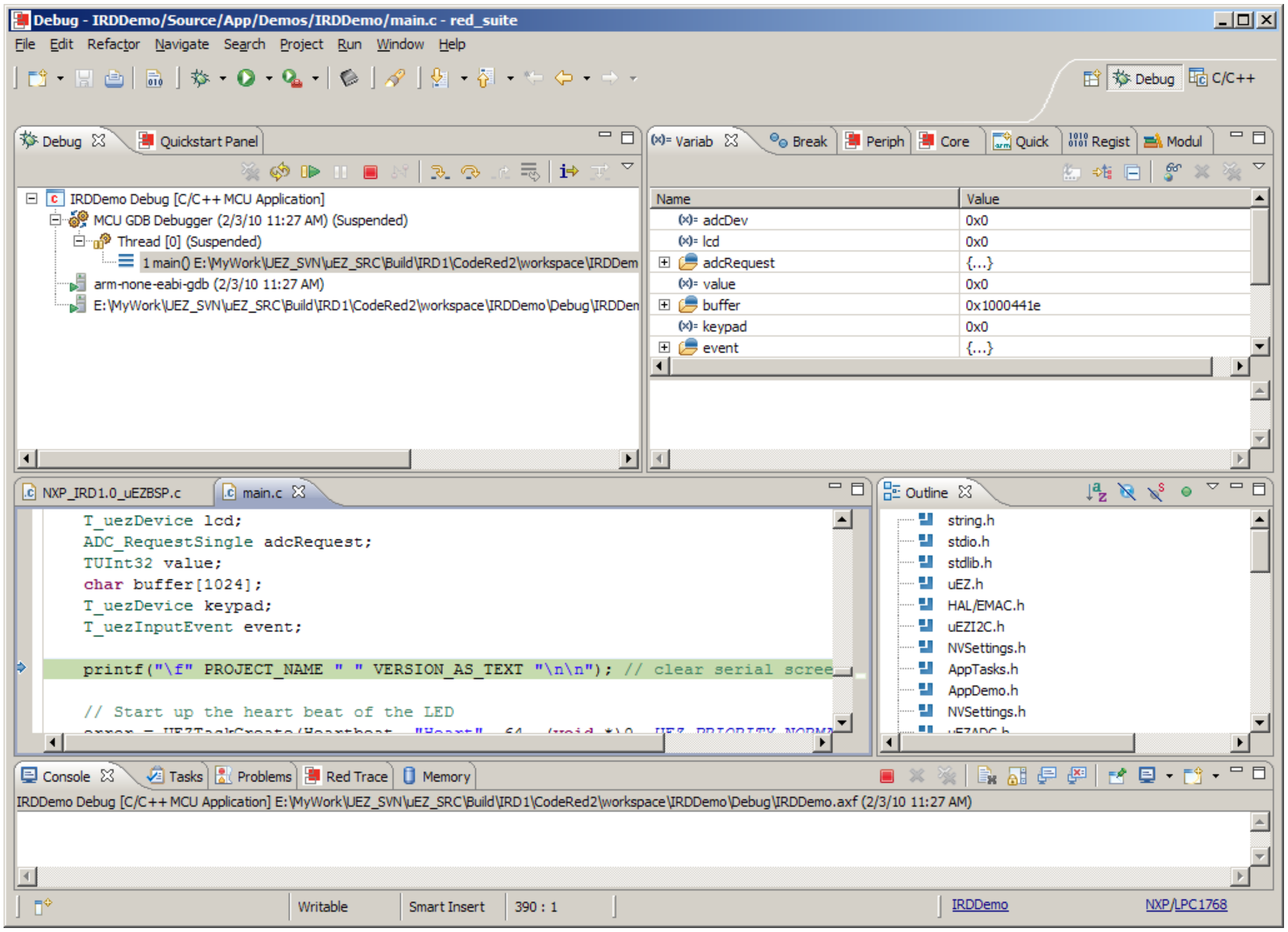


```
C-Build [IRDDemo]
make  no print directory post build
Performing post-build steps
arm-none-eabi-size IRDDemo.axf; # arm-none-eabi-objdump -h -S IRDDemo.axf
>IRDDemo.lss
   text    data     bss     dec     hex filename
  98944   36184    7532  142660   22d44 IRDDemo.axf
```

Debugging with Code Red 2.0

In order to debug, you will need a Red Probe. Connect the Red Probe to the target board (an IRD 1.0 with 1768 in this case) and power on the target. Click the debug icon at the top. The firmware is now downloaded into the target board.

Click Yes when to the Confirm Perspective Switch dialog. The following screen should now appear:


Debug - IRDDemo/Source/App/Demos/IRDDemo/main.c - red_suite
File Edit Refactor Navigate Search Project Run Window Help
Debug C/C++
Debug Quickstart Panel
Variab Break Periph Core Quick Regist Modul
NXP_IRD1.0_uEZBSP.c main.c
Outline
Console Tasks Problems Red Trace Memory
Writable Smart Insert 390 : 1 IRDDemo NXP/LPC1768

```
IRDDemo Debug [C/C++ MCU Application]
  MCU GDB Debugger (2/3/10 11:27 AM) (Suspended)
    Thread [0] (Suspended)
      1 main() E:\MyWork\UEZ_SVN\uEZ_SRC\Build\IRD1\CodeRed2\workspace\IRDDem
  arm-none-eabi-gdb (2/3/10 11:27 AM)
  E:\MyWork\UEZ_SVN\uEZ_SRC\Build\IRD1\CodeRed2\workspace\IRDDemo\Debug\IRDDem
```

Name	Value
adcDev	0x0
lcd	0x0
adcRequest	{...}
value	0x0
buffer	0x1000441e
keypad	0x0
event	{...}

```
    T_uezDevice lcd;
    ADC_RequestSingle adcRequest;
    TUInt32 value;
    char buffer[1024];
    T_uezDevice keypad;
    T_uezInputEvent event;

    printf("\f" PROJECT_NAME " " VERSION_AS_TEXT "\n\n"); // clear serial scree

    // Start up the heart beat of the LED
    error = UEZTaskCreate(Heartbeat, "Heart", 64, (void *)0, UEZ_PRIORITY_NORMA
```

```
string.h
stdio.h
stdlib.h
uEZ.h
HAL/EMAC.h
uEZI2C.h
NVSettings.h
AppTasks.h
AppDemo.h
NVSettings.h
uEZADC.h
```

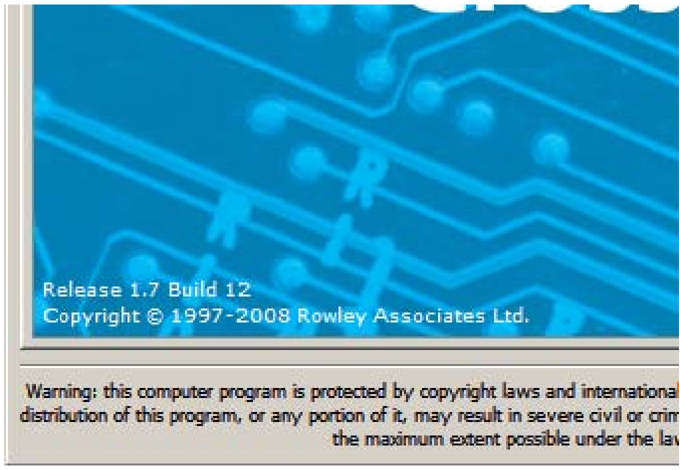
```
IRDDemo Debug [C/C++ MCU Application] E:\MyWork\UEZ_SVN\uEZ_SRC\Build\IRD1\CodeRed2\workspace\IRDDemo\Debug\IRDDemo.axf (2/3/10 11:27 AM)
```

The debugging session is now waiting for commands. Press F5 to step into a line of code and F6 to step over a line of code. Press F8 to run the code. Go to Run->Suspend to break at any point. Double click on the left of any line to set a breakpoint. Press CTRL-F2 to terminate the debug session (click on the C/C++ at the top right to change the perspective back to before).

Rowley CrossWorks CrossStudio v1.7 Project Configuration

Check CrossStudio Version

uEZ™ is built using v1.7, or later, of the Rowley CrossWorks CrossStudio for ARM® toolset. To confirm the version number of the tools, go to Help->About in the main menu and the version number should appear in the lower left of the dialog.



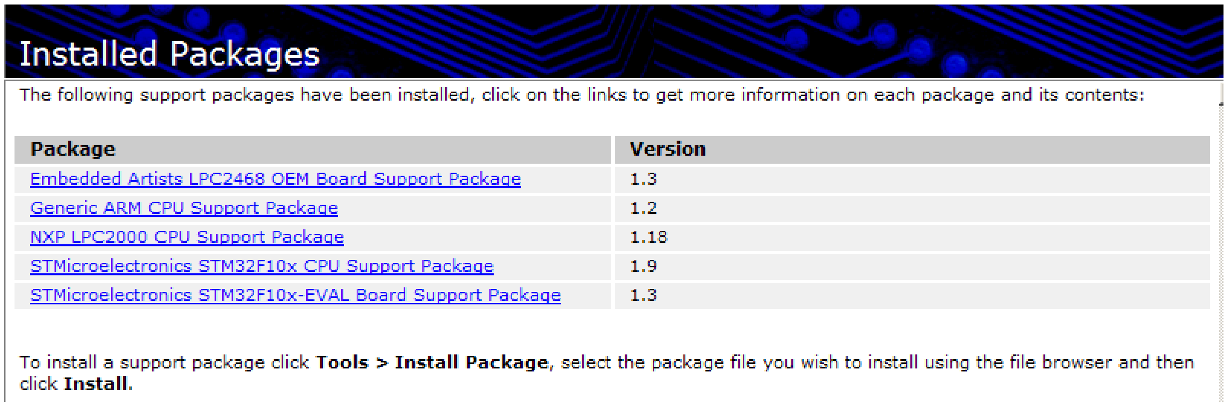
Release 1.7 Build 12
Copyright © 1997-2008 Rowley Associates Ltd.

Warning: this computer program is protected by copyright laws and international
distribution of this program, or any portion of it, may result in severe civil or crim
the maximum extent possible under the la

Build versions of 12 or above are acceptable.

Check Installed Packages

In addition, packages for your target processor(s) should be installed. Go to **Tools->Show Installed Packages** and see which packages have been installed. For example,



Installed Packages

The following support packages have been installed, click on the links to get more information on each package and its contents:

Package	Version
Embedded Artists LPC2468 OEM Board Support Package	1.3
Generic ARM CPU Support Package	1.2
NXP LPC2000 CPU Support Package	1.18
STMicroelectronics STM32F10x CPU Support Package	1.9
STMicroelectronics STM32F10x-EVAL Board Support Package	1.3

To install a support package click **Tools > Install Package**, select the package file you wish to install using the file browser and then click **Install**.

If doing development for the DK-TS-KIT with the SOMDIMM-LPC2478, the following packages should be installed:

Generic ARM® CPU Support Package
NXP LPC2000 CPU Support Package

If the packages are not installed, go to **Tools->Download Packages from Web**, download the missing packages, and then use **Tools->Install Package…** to install them.

Opening and Compiling uEZ™

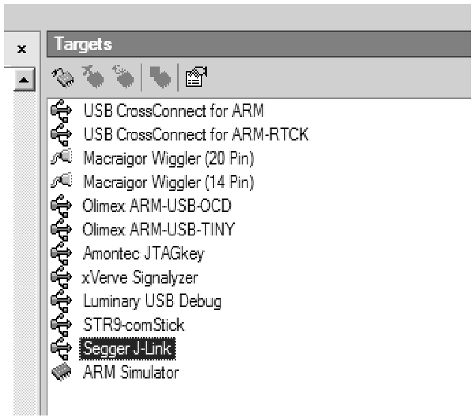
Open the project file "uEZ.hzp" in the /uEZ directory. The Project Explorer should appear at the right. The uEZ distribution comes with the uEZ Demonstration Application in the /uEZ/App directory and is configured for the DK-TS-KIT build for the LPC2478 processor.

If either a different platform or processor is needed, go to **File->Open** and open file /uEZ/App/App_Config.h. Change the #define PLATFORM to match the hardware platform name (open file /uEZ/Include/Config.h for a complete list of platforms).

To compile the code for the first time, select **Build->Rebuild** uEZ from the main menu. When complete, the output should report "Build up to date" when done.

Downloading and Debugging uEZ™ on the Target

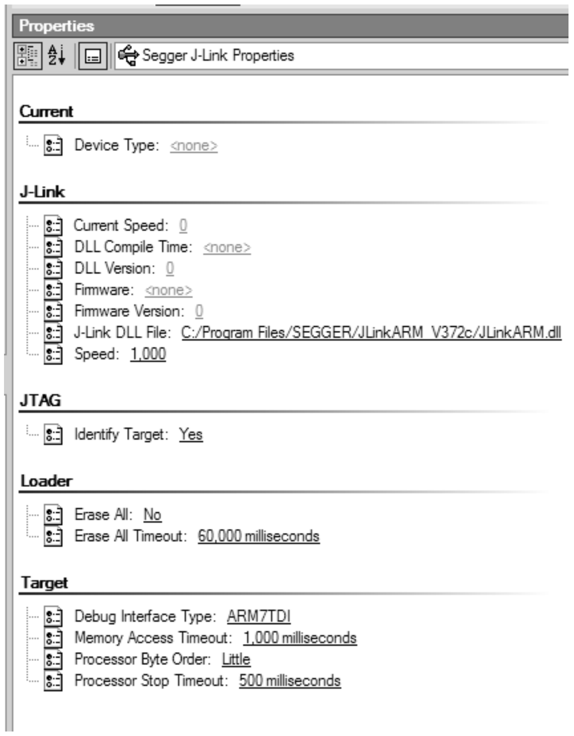
1) Plug the J-Link device into the PC and install any drivers as directed.
2) Plug the J-Link's JTAG cable into the SOMDIMM-LPC2478 board at J3.
3) Select **Target** menu and choose **Targets**. The following list will appear to the right.



Targets

- USB CrossConnect for ARM
- USB CrossConnect for ARM-RTCK
- Macraigor Wiggler (20 Pin)
- Macraigor Wiggler (14 Pin)
- Olimex ARM-USB-OCD
- Olimex ARM-USB-TINY
- Amontec JTAGkey
- xVerve Signalyzer
- Luminary USB Debug
- STR9-comStick
- Segger J-Link
- ARM Simulator

4) Right click on "Segger J-Link" and select Properties,



Properties

Segger J-Link Properties

Current

- Device Type: <none>

J-Link

- Current Speed: 0
- DLL Compile Time: <none>
- DLL Version: 0
- Firmware: <none>
- Firmware Version: 0
- J-Link DLL File: C:/Program Files/SEGGER/JLinkARM_V372c/JLinkARM.dll
- Speed: 1,000

JTAG

- Identify Target: Yes

Loader

- Erase All: No
- Erase All Timeout: 60,000 milliseconds

Target

- Debug Interface Type: ARM7TDI
- Memory Access Timeout: 1,000 milliseconds
- Processor Byte Order: Little
- Processor Stop Timeout: 500 milliseconds

5) If this is the first time you are programming with the J-Link on the Rowley Platform, select J-Link DLL File, press the "…" button and find the file JLinkARM.dll (usually installed in C:/Program Files/SEGGER/")
6) If programming a blank LPC2478 part, select a Speed of 100. If the LPC2478 has already been programmed, select a Speed of 1000. All LPC2478's that come with the DK-TS-KIT are pre-programmed.
7) Go back to menu **Target** and select "Connect Segger J-Link"
8) Press F5 to download the application to the target and start debugging. When the application starts, it will pause. Press F5 again to start executing the code.
9) To stop at any line of code, right click the line and select Toggle Breakpoint. Execution will stop automatically at the breakpoint. Press F5 again to continue debugging.
10) When done debugging, select **Debug->Stop**. The debugger will return to standard editor mode.
11) From this point on, the process is simply a matter of editing code, compiling the code (**Build->Build uEZ** or pressing F7), and then returning to the debugger.

4. Questions and Support

For all questions, bug reports and general technical support, go to https://sourceforge.net/projects/uez/ and use the Sourceforge.net tools or email FDI directly at support@teamfdi.com .

Marketing updates and details on technical support are available at **www.teamfdi.com/uez .**

Can we use another RTOS?
All **µEZ™** components are made to connect through the **µEZ™** OSAL (Operating System Abstraction Layer) to the RTOS ensuring compatibility with many different RTOS's. Currently all **µEZ™** development by FDI is being focused on the FreeRTOS™ platform since it satisfies the low cost tool requirement because it is "free". RTOS products from other vendors can also be used with **µEZ™**.

Which compiler suites do you support?
Currently, most **µEZ™** development by FDI has been focused on the low cost Rowley CrossWorks compiler, but we also support the IAR EWARM tool suite. In addition, Keil, ARM® RealView, GNU and other compilers can be used with **µEZ™**.

What debug tools are available?
Since **µEZ™** uses the debug tools that are provided in the customers compiler suite, it can be used with any of the tools listed above.

Which processors are supported?
Even though **µEZ™** is processor independent, all of our initial development has been focused on various members of the ARM Family. We currently support the NXP LPC24xx family, and processors like Cortex™-M3, ARM9®, and other variations of ARM7® are being added.