µEZ® Software
Quickstart Guide

Copyright ©2011, Future Designs, Inc., All Rights Reserved

Future Designs, Inc.
Your Development Partner
2702 Triana Boulevard SW. Huntsville, AL 35805

Table of Contents

Information in this document is provided solely to enable the use of Future Designs products. FDI assumes no liability whatsoever, including infringement of any patent or copyright. FDI reserves the right to make changes to these specifications at any time, without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Future Designs, Inc. 2702 Triana Blvd, Huntsville, AL 35805

NOTE: The inclusion of vendor software products in this kit does not imply an endorsement of the product by Future Designs, Inc.
© 2011 Future Designs, Inc. All rights reserved.

For more information on FDI or our products please visit www.teamfdi.com.

µEZ® is a registered trademark of Future Designs, Inc.
Microsoft, MS-DOS, Windows, Windows XP, Microsoft Word are registered trademarks of Microsoft Corporation.
Other brand names are trademarks or registered trademarks of their respective owners.

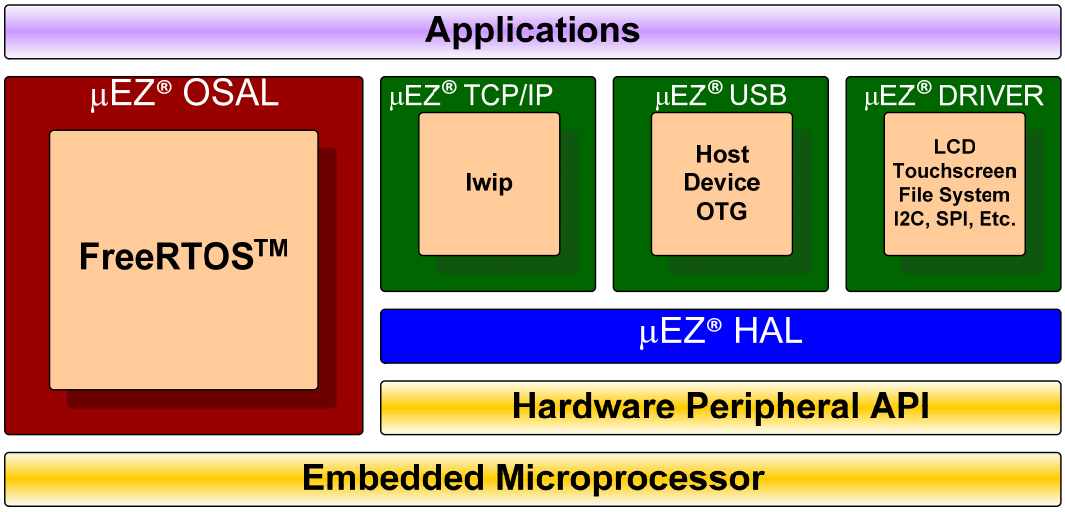
FDI PN: MA00015
Revision: Rev 1.11, 9/1/2011 12:09:00 PM
Printed in the United States of America

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 for different applications/demos
/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 shared among multiple builds.
/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.

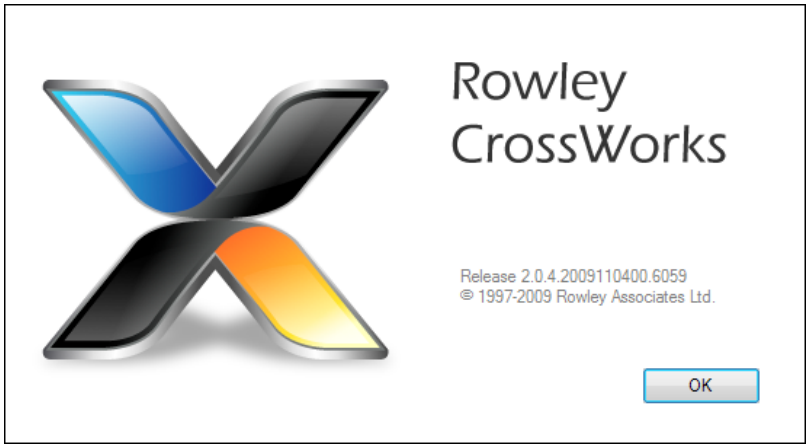
Preparing the uEZ® Source Code

Download the uEZ® v1.11 (or later) 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.

Rowley CrossWorks CrossStudio v2.0 Project Configuration

Check CrossStudio Version

uEZ® is built using v2.0, 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 middle of the dialog.



Build versions of 2.0.x 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	Status
Generic ARM CPU Support Package	1.3	Installed
NXP LPC1000 CPU Support Package	1.7	Installed
NXP LPC2000 CPU Support Package	1.30	Installed

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 doing development for the DK-TS-KIT with the SOMDIMM-LPC1788, the following packages should be installed:

Generic ARM® CPU Support Package

NXP LPC1000 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.

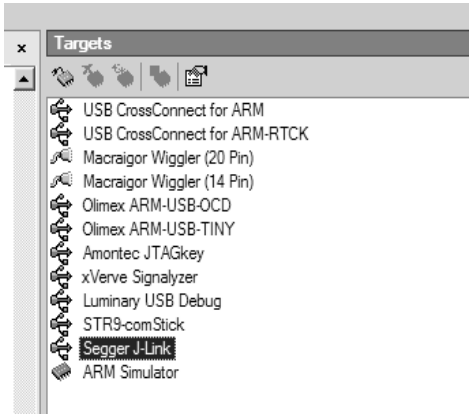
Opening and Compiling uEZ®

Open the one project build files in the /uEZ/Build directories. For example, when working on the DK-57TS-LPC2478, open /uEZ/Build/DK-TS-KIT/CrossWorks 2.0/DK-57TS-LPC2478/DK-57TS-LPC2478 uEZDemo.hzp. The Project Explorer should appear at the right showing all the files in the project. The uEZ distribution comes with the uEZ Demonstration Application in the /uEZ/App/UEZDemo directory and is configured for the DK-57TS-LPC2478 kit.

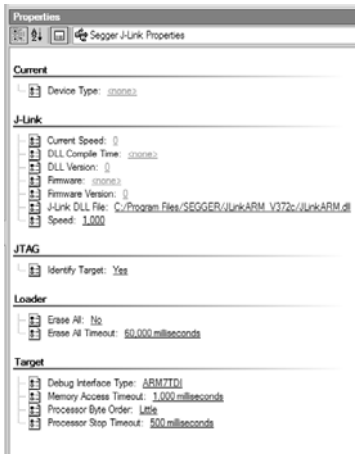
To compile the code for the first time, select **Build->Rebuild** uEZ from the main menu. When complete, the output should report "Build up to date" when done.

Downloading and Debugging uEZ® on the Target

1) Plug the J-Link device into the PC and install any drivers as directed. The Segger J-Link drivers can be found at http://www.segger.com/cms/jlink-software.html with additional information at http://www.segger.com/cms/development-tools.html.
2) Plug the J-Link's JTAG cable into the target (e.g., SOMDIMM-LPC2478's J3 connector).
3) Power on the target board.
4) Select **Target** menu and choose **Targets**. The following list will appear to the right.



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



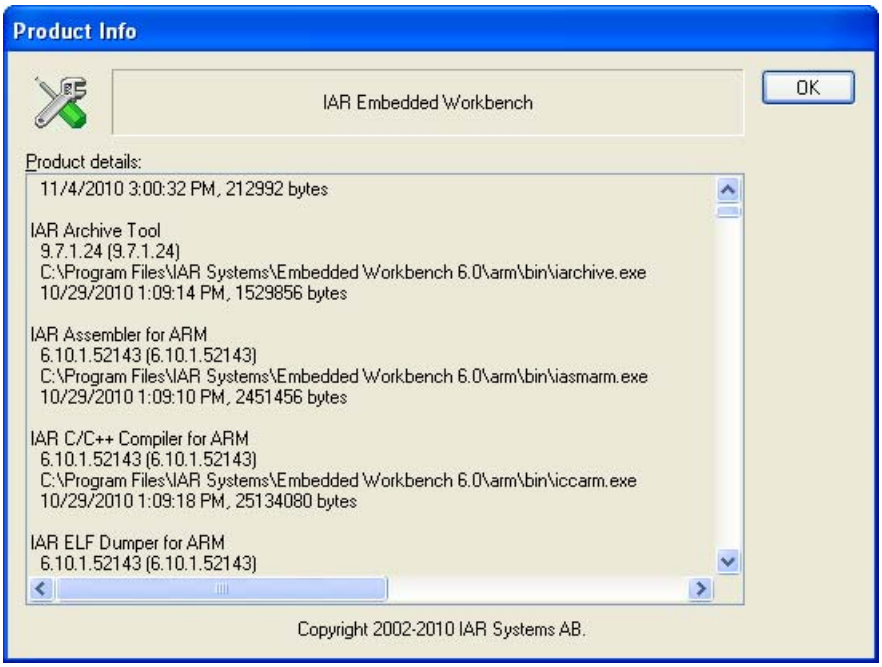
6) If this is the first time you are programming with the J-Link on the Rowley Platform, select J-Link DLL File, press the "…" button and find the file JLinkARM.dll (usually installed in C:/Program Files/SEGGER/")

7) 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-xTS-LPC2478 are pre-programmed.
8) Go back to menu **Target** and select "Connect Segger J-Link"
9) 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.
10) 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.
11) When done debugging, select **Debug->Stop**. The debugger will return to standard editor mode.
12) From this point on, the process is simply a matter of editing code, compiling the code (**Build->Build uEZ** or pressing F7), and then running the debugger.

IAR Systems Embedded Workbench v6.10 Project Configuration

Check IAR Version

uEZ® is built using 6.10, or later, of the IAR Embedded Workbench Toolset. To confirm the version number of the tools, go to Help->About->Product Info in the main menu and the version number should appear in the middle of the dialog.



IAR C/C++ Compiler for ARM 6.10.1 and greater are acceptable.

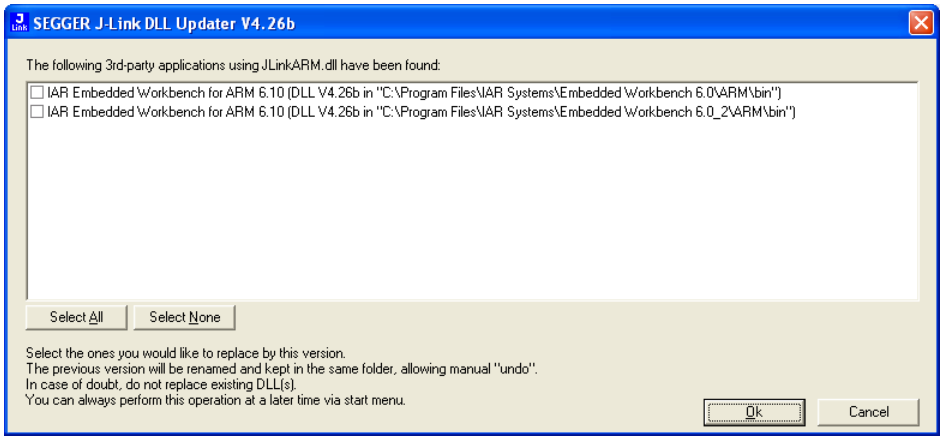
Opening and Compiling uEZ®

Open the one project build files in the /uEZ/Build directories. For example, when working on the DK-57TS-LPC2478, open uEZ\Build\DK-TS-KIT\DK-57TS-LPC2478\IAR6.10\ DK-57TS-LPC2478.eww. The Project Explorer should appear at the left showing all the files in the project. The uEZ distribution comes with the uEZ Demonstration Application in the /uEZ/App/ DK-TS-Demo directory and is configured for the DK-XXXTS-LPC2478 kit, in this case DK-57TS-LPC2478.

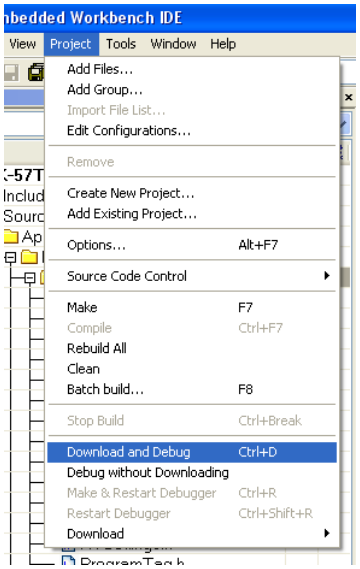
To compile the code for the first time, select **Project->Make** from the main menu or press F7. When complete, the output should report "Total number of errors: 0" when done.

Downloading and Debugging uEZ® on the Target

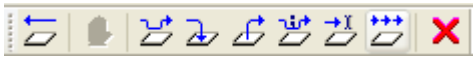
1) Plug the J-Link device into the PC and install any drivers as directed. The Segger J-Link drivers can be found at http://www.segger.com/cms/jlink-software.html with additional information at http://www.segger.com/cms/development-tools.html.
2) Plug the J-Link's JTAG cable into the target (e.g., SOMDIMM-LPC2478's J3 connector).
3) Power on the target board.
4) The project is preconfigured for the Segger J-Link. If the J-link software is installed after IAR the dll will automatically be updated. Otherwise run the SEGGER J-Link Updater from SEGGER/J-Link ARM vx.xx in the start menu.



5) Select Project->Download and Debug from the main menu or Ctrl + D to start debugging.



6) Debugging control can be operated from debug toolbar.



7) When finished debugging press the red X in the debug toolbar.

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 . A support forum is also provided at http://www.teamfdi.com/forum/ .

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, the NXP LPC17xx, and processors like Cortex™-M3, and other variations of ARM7® are being added.