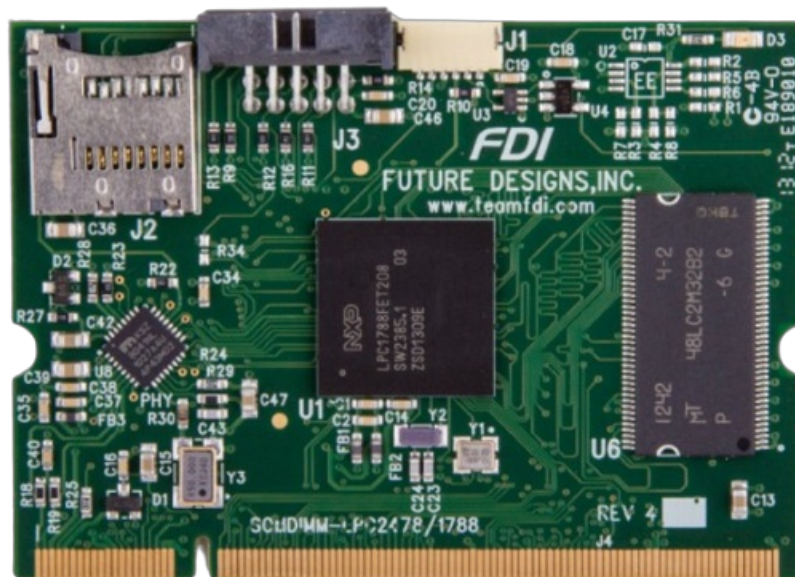


FDI LPC1788 Microcontroller-based SOMDIMM module User Guide

[Home](#) » [FDI](#) » FDI LPC1788 Microcontroller-based SOMDIMM module User Guide 

FDI LPC1788 Microcontroller-based SOMDIMM module User Guide



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. 996 A Cleaner Way, 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.

© 2023 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 2.13, 07/11/2023

Published in the United States of America

Contents

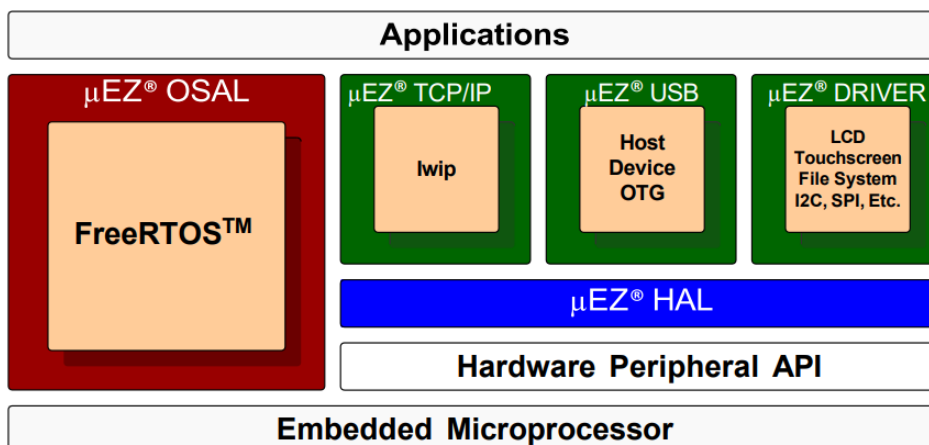
- [1 Introduction](#)
- [2 Downloading uEZ®](#)
- [3 Project Configuration](#)
- [4 Rowley CrossWorks CrossStudio v4.10.x Project Configuration](#)
- [5 Downloading and Debugging uEZ® on the Target](#)
- [6 Downloading and Debugging uEZ® on the Target](#)
- [7 Questions and Support](#)
- [8 Can we use another RTOS?](#)
- [9 Which compiler suites do you support?](#)
- [10 What debug tools are available?](#)
- [11 Which processors are supported?](#)
 - [11.1 Documents / Resources](#)
 - [11.1.1 References](#)
 - [11.2 Related Posts](#)

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 minimising development time and maximising 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 μ EZ® the primary features of common multi-tasking operating systems are abstracted, thus easing the transition to an open source or low-cost RTOS. The μ EZ® 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 μ EZ® sub-system drivers utilise 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 μ EZ® HAL routines provided by FDI or they can write their own. The HAL routines provide for RTOS/ μ EZ® independence and allow portability within a family of processors.

μ EZ® 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

Downloading μ EZ®

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

The μ EZ® file directory structure should be as follows:

Directory	Description
uEZ/Build	Projects/make files for different applications/demos
uEZ/Include	uEZ® system files and Config.h
uEZ/Include/Device	Device Driver class definitions.
uEZ/Include/HAL	Hardware Abstraction Layer (HAL) driver class definitions.
uEZ/Include/Types	Common data types used by both HAL and Device Drivers.
uEZ/Source	Source code
uEZ/Source/Devices/<category>/<manufacturer>/<device>	Device specific code organised by category (I2C, SSP, etc.), manufacturer, and specific device.
uEZ/Source/Library/<category>/<package>	Various support libraries organised by category (graphics, file system, etc.) and package name.
uEZ/Source/Platform/<manufacturer>/<platform>	Platforms/boards code organised by manufacturer and specific platform build.
uEZ/Source/Processor/<manufacturer>/<processor>	Processor specific code in separate directories organised by manufacturer and specific processor.
uEZ/Source/RTOS/<RTOS>/	RTOS source code in separate directories
uEZ/Source/uEZSystem	uEZ® System Core routines
uEZDemos/Build	Demo Project files are stored by type and specific board.
uEZDemos/Source/App	User Application/Shared Demo source code

Project Configuration

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

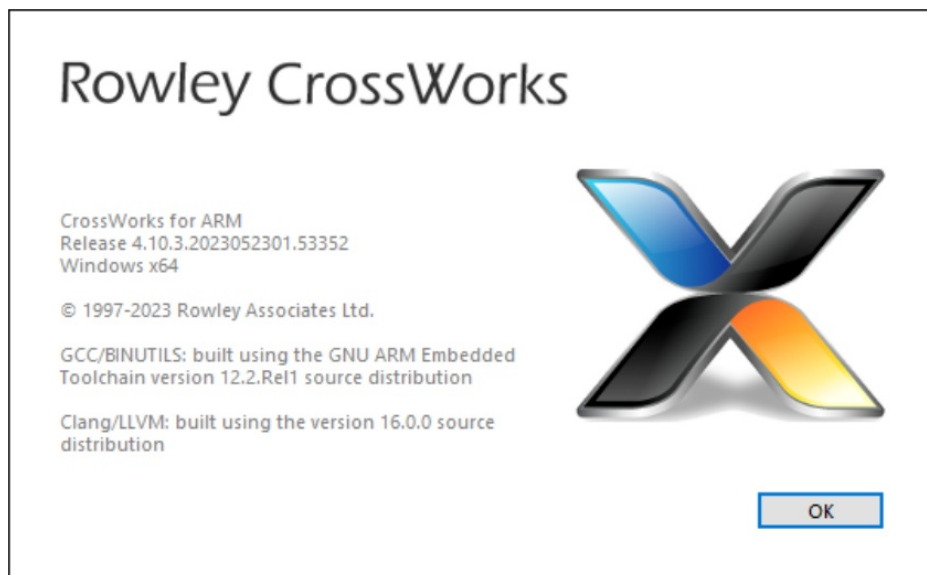
Preparing the uEZ® Source Code

Download the uEZ® v2.13 (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 v4.10.x Project Configuration

Check CrossStudio Version

uEZ® is built using v4.10, 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.



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,

Package	Version	Status
NXP LPC1000 CPU Support Package	1.32	Installed
NXP LPC4300 CPU Support Package	3.7	Installed

If doing development with the **NXP LPC1788/LPC4088**, the following packages should be installed:

ARM CPU Support Package

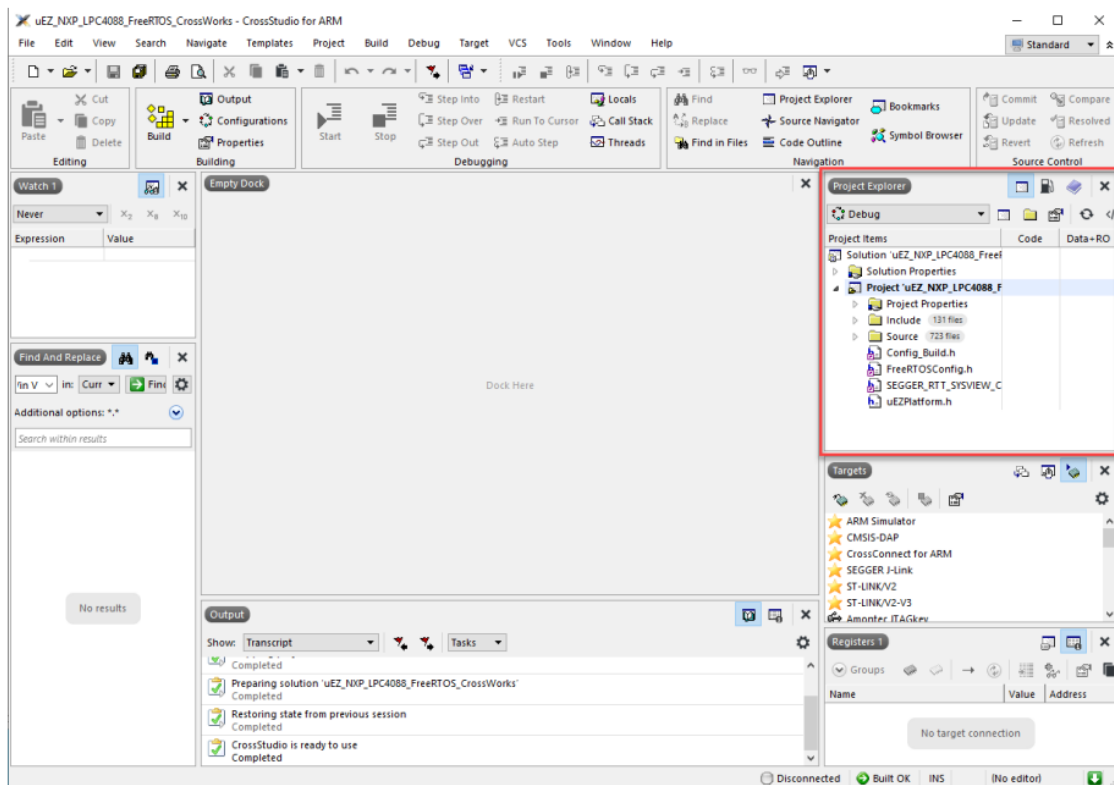
NXP LPC1000 CPU Support Package

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.

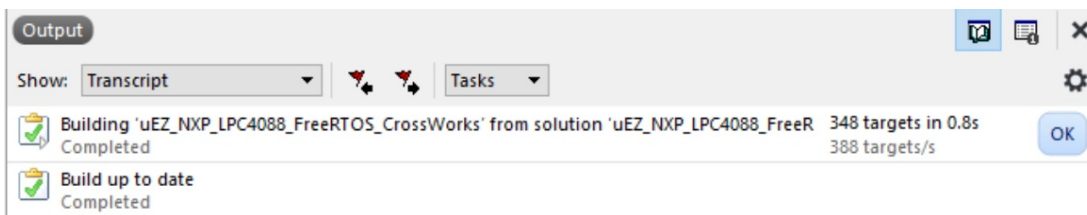
Opening and Compiling uEZ®

Before the uEZ project can be downloaded and debugged, the uEZ Library needs to be built.

Open the library project file in the \uEZ\Build directory. For example, when working on the uEZGUI-4088-43WQN, open “\uEZ\Build\Generic\NXP\LPC4088\Free RTOS\ Cross Works\ uEZ_NXP_LPC4088_Free RTOS_ Cross Works.hz “. The Project Explorer should appear at the right showing all the files in the project.



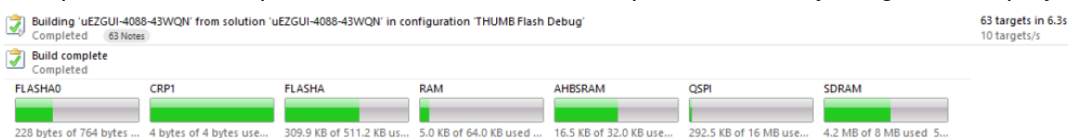
To compile the code for the first time, select “Build” → “Build uEZ_NXP_LPC4088_Free RTOS_Cross Works” from the toolbar menu, or press . When complete, the output should report “Build Complete” or “Build up to date” when done.



Close the uEZ Library project.

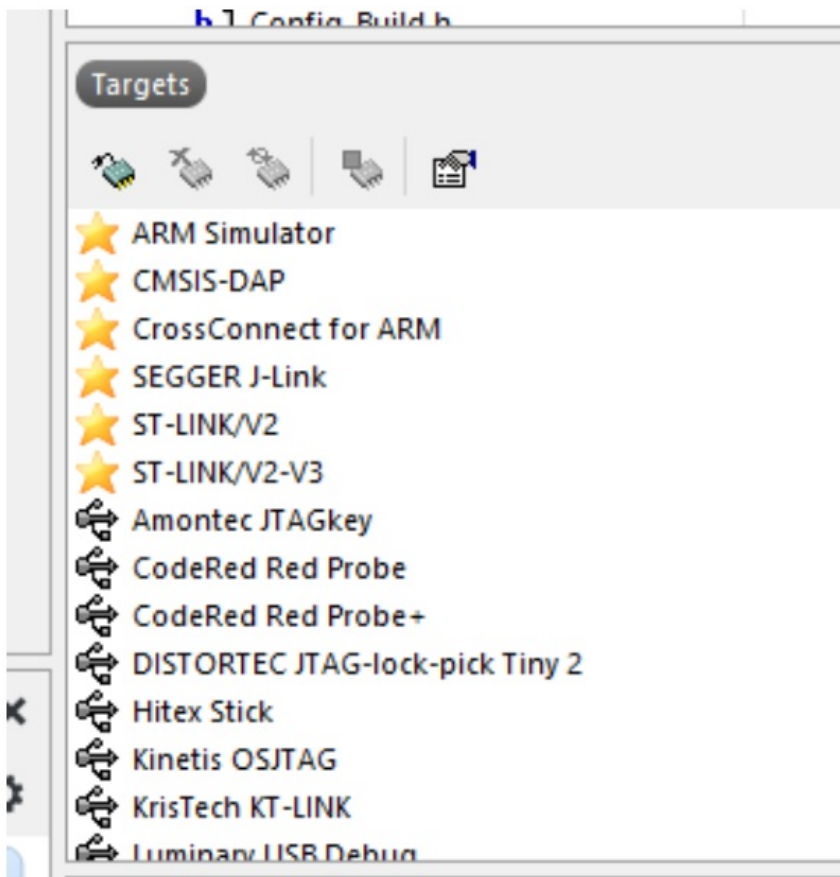
The uEZ distribution comes with a uEZ GUI Demonstration Application project file: “uEZDemos\Build\FDI\uEZGUI\uEZGUI-4088-43WQN\CrossWorks\uEZGUI-4088-43WQN.hzp”.

Open the “**uEZGUI-4088-43WQN.hzp**” project and compile just as with the uEZ Library. Select “Build” → “Build **uEZGUI-4088-43WQN**” from the toolbar menu, or press . When complete, the output should report “Build Complete” or “Build up to date” when finished and report the memory usage for this project.



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., uEZGUI-4088-43WQN’s J5 connector).
3. Power on the target board.
4. Select “View” from the toolbar and choose “Targets”. The following list will appear on the right:



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

SEGGER J-Link Properties	
Property	Setting
J-Link	
• Additional J-Link Options	
• Enable Adaptive Clocking	No
• Exclude Flash Cache Range	None
• J-Link DLL File	JLink_x64.dll
• Log File	None
• Reset Type	Default
• Script File	None
• Settings File	None
• Show Log Messages In Output Window	Yes
• Speed	4,000 kHz
• Supply Power	No
• Trace Buffer Size	65,536 Bytes
• Use Built-in Flash Loader	Auto
• Use Built-in RTT support	Auto
• Use Built-in TRACE support	Yes
• Verify Read Operations	No
Target	
• Host Connection	None

6. If this is the first time you are programming with the J-Link on the Rowley Platform...

a) Click on "J-Link DLL File".

b) Press the "..." button and find the file Treblinka (usually installed in C:/Program Files/SEGGER/)

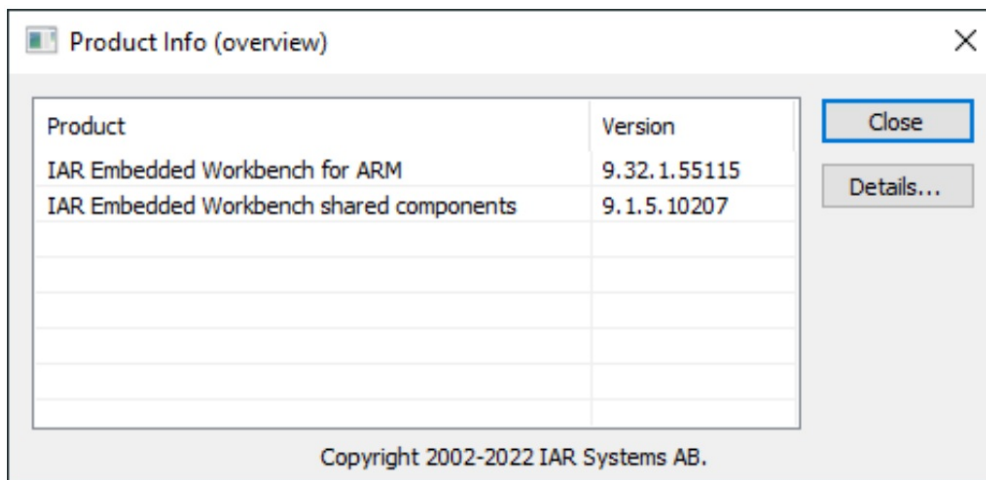
c) If programming a blank LPC4088 part, select a Speed of 100 kHz. If the LPC4088 has already been

programmed, select a Speed of 4000 kHz.

7. Right click on “Segger J-Link”, and click “Connect Segger J-Link”.
8. Press to download the application to the target and start debugging. When the application starts, it will pause at main(). Press again to continue executing the code.
9. To stop at any line of code, right click the line and select Toggle Breakpoints. Execution will stop automatically at the breakpoints. Press again to continue debugging.
10. When done debugging, select “Debug” → “Stop” from the toolbar, or the button from the Debugging Menu. 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 running the debugger.

IAR Systems Embedded Workbench v9.32.1 Project Configuration Check IAR Version

uEZ® is built using 9.32.1, or later, of the IAR Embedded Workbench Tool set and C/C++ Compiler. To confirm the version number of the tools, go to “Help” → “About” → “Product Info” in the toolbar and the version number should appear in the middle of the dialog.



Opening and Compiling uEZ®

Before the uEZ project can be downloaded and debugged, the uEZ Library needs to be built.

Open the library project file in the \uEZ\Build directory. For example, when working on the uEZGUI-4088-43WQN, open “\uEZ\Build\Generic\NXP\LPC4088\FreeRTOS\IAR\UEZ_NXP_LPC4088_FreeRTOS_IAR.eww”. The Project Explorer should appear at the left, showing all the project files.

To compile the code for the first time, select “Project” → “Make” from the toolbar, or press <F7>. When complete, the output should report “Total number of errors: 0” and “Build succeeded”.

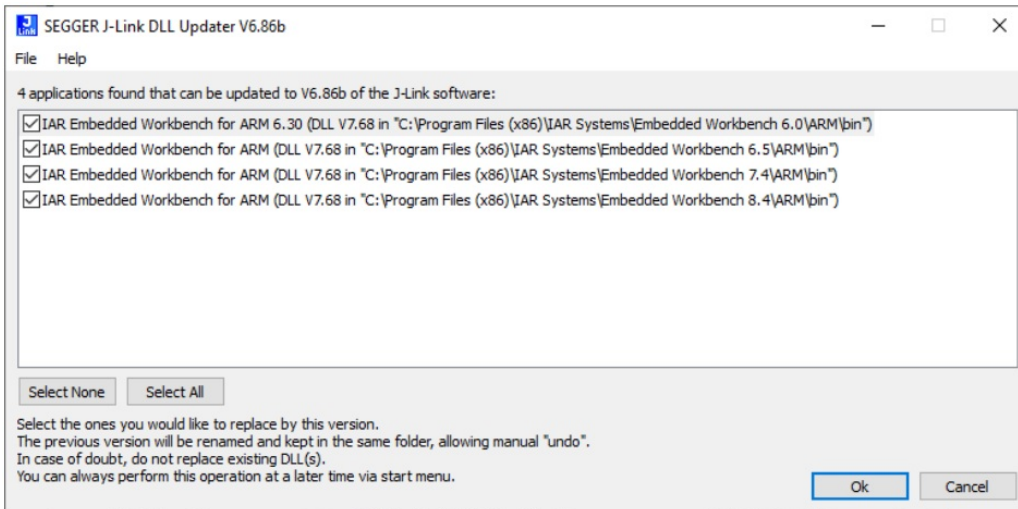
With the uEZ Library built, close “uEZ_NXP_LPC4088_Freestones_IAR.eww”. Now open the uEZ GUI Demo project: “\uEZDemos\Build\FDI\uEZGUI\uEZGUI-4088-43WQN\IAR\UEZGUI-4088-43WQN.eww”.

As in the uEZ Library project, select “Project” → “Make” from the toolbar, or press <F7>. When complete, the output should report “Total number of errors: 0” and “Build succeeded”.

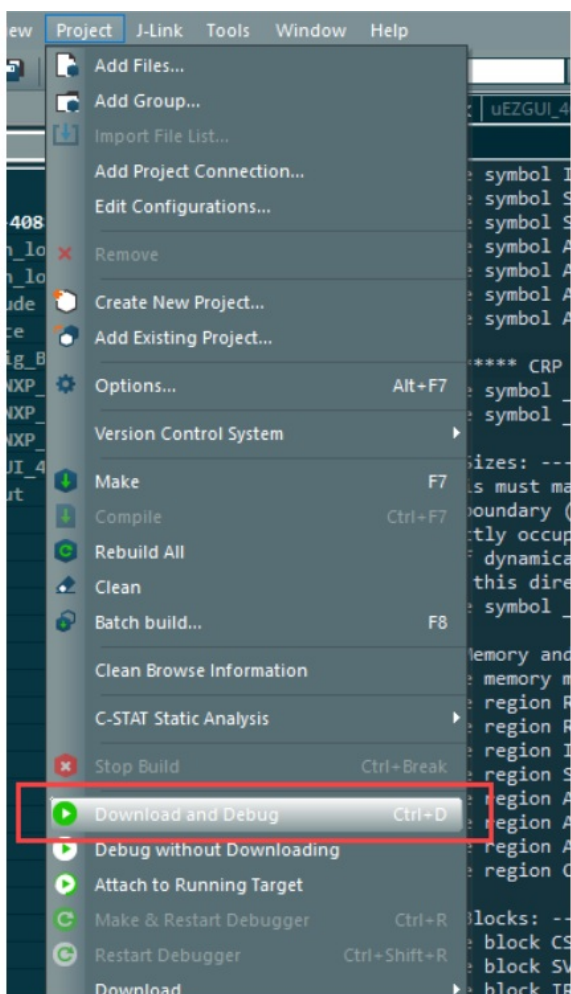
Downloading and Debugging uEZ® on the Target

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., uEZGUI-4088-43WQN's J5 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 toolbar, click the green “Download and Debug” button on the toolbar, or press + to start debugging.



6. Debugging control can be operated from debug toolbar.



7. Debugging will pause at main(). Press or the white circle with a blue arrow inside it to continue code execution.



8. When finished debugging press the red X in the debug toolbar.

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 <https://www.teamfdi.com/forum/> .

Marketing updates and details on technical support are available at <https://www.teamfdi.com/uez>.

Can we use another RTOS?

Which compiler suites do you support?

Currently, most μ EZ[®] development by FDI has been done on the following tool suites: Rowley Crossworks for ARM, and IAR EWARM. ARM[®] RealView, GNU, Keil uVision, and Renesas C/C++ with HEW, KPIT GNU and other compilers can also be used with μ EZ[®].




What debug tools are available?

Since μ EZ[®] 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 μ EZ[®] is processor independent, all of our initial development has been focused on various members of the ARM Family. We currently support the NXP LPC17xx family, the NXP LPC40xx family, the NXP LPC43xx family, the Renesas RX6XN, and processors like Cortex[™] M3/M4, and other variations of ARM v7[®].

Documents / Resources

  	FDI LPC1788 Microcontroller-based SOMDIMM module [pdf] User Guide LPC1788 Microcontroller-based SOMDIMM module, LPC1788, Microcontroller-based SOMDIMM module, based SOMDIMM module, SOMDIMM module, module
---	--

References

-  [Compare, Download & Develop Open Source & Business Software - SourceForge](#)
-  [Debug & Trace Probes](#)
-  [uEZ download | SourceForge.net](#)
-  [Home - FDI - Future Designs Inc.](#)
-  [uEZ download | SourceForge.net](#)
-  [Home - FDI - Future Designs Inc.](#)
-  [Index - FDI Forums](#)
- [User Manual](#)

[Manuals+](#), [Privacy Policy](#)

This website is an independent publication and is neither affiliated with nor endorsed by any of the trademark owners. The "Bluetooth®" word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. The "Wi-Fi®" word mark and logos are registered trademarks owned by the Wi-Fi Alliance. Any use of these marks on this website does not imply any affiliation with or endorsement.