

## Abstract

This application note provides instructions for connecting and setting up the LogicPD LH7A404 Card Engine evaluation board for use with the ULINK2 USB/JTAG adapter and the MDK-ARM development tools. It also provides step-by-step instructions for building, downloading and debugging a simple  $\mu$ Vision example project on that board.

---

## Contents

You will need:.....	2
Hardware Setup .....	2
Software Setup.....	3
BLINKY Example Program .....	4
Editing BLINKY .....	4
Building BLINKY .....	5
Debugging BLINKY.....	5

## Revision History

- August 2009: Initial Version

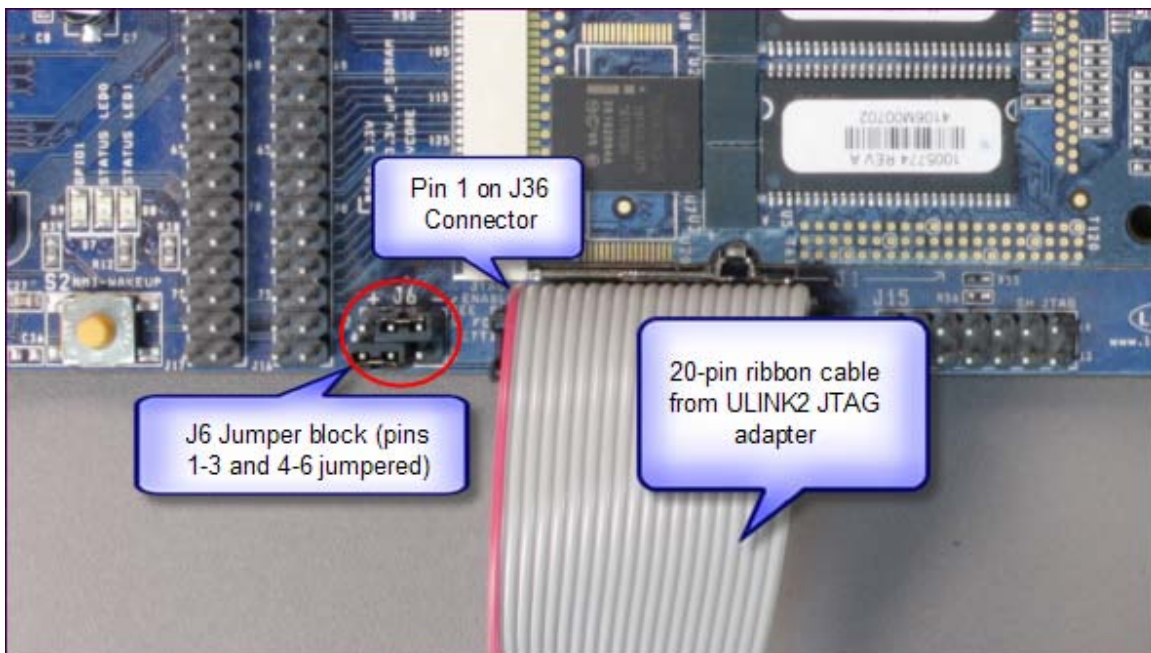
Information in this file, the accompany manuals, and software is  
Copyright (c) ARM. All rights reserved.

## You will need:

- Zoom SDK baseboard
- LH7A404 Card Engine
- 5 VDC power supply
- ULINK2 USB-JTAG adapter
- USB A/B cable
- Keil MDK-ARM Evaluation Tools
- The LH7A404.ZIP file containing the BLINKY example project

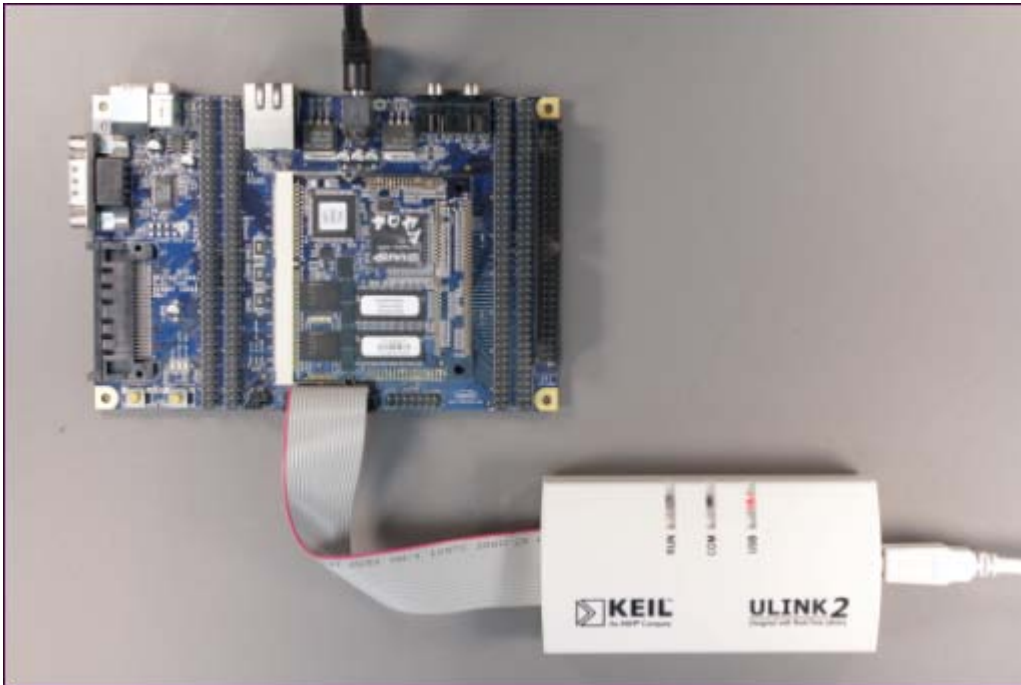
## Hardware Setup

1. Install the LH7A404 Card Engine on the Zoom SDK baseboard according to the LogicPD User's Guide.
2. Connect the 20-pin ribbon cable from the ULINK2 JTAG adapter to the **J36** header on the Zoom SDK baseboard. This connector is not keyed. Make sure pin 1 on the cable (red tracer) is oriented toward pin 1 on the header as shown below.



3. On the Zoom SDK baseboard, at jumper block **JP6** make sure pins 1-3 and pins 4-6 are jumpered (normal JTAG mode).
4. Connect the 5VDC power supply to the Zoom SDK baseboard.

5. Connect a standard USB A/B cable between the ULINK2 adapter and your PC.



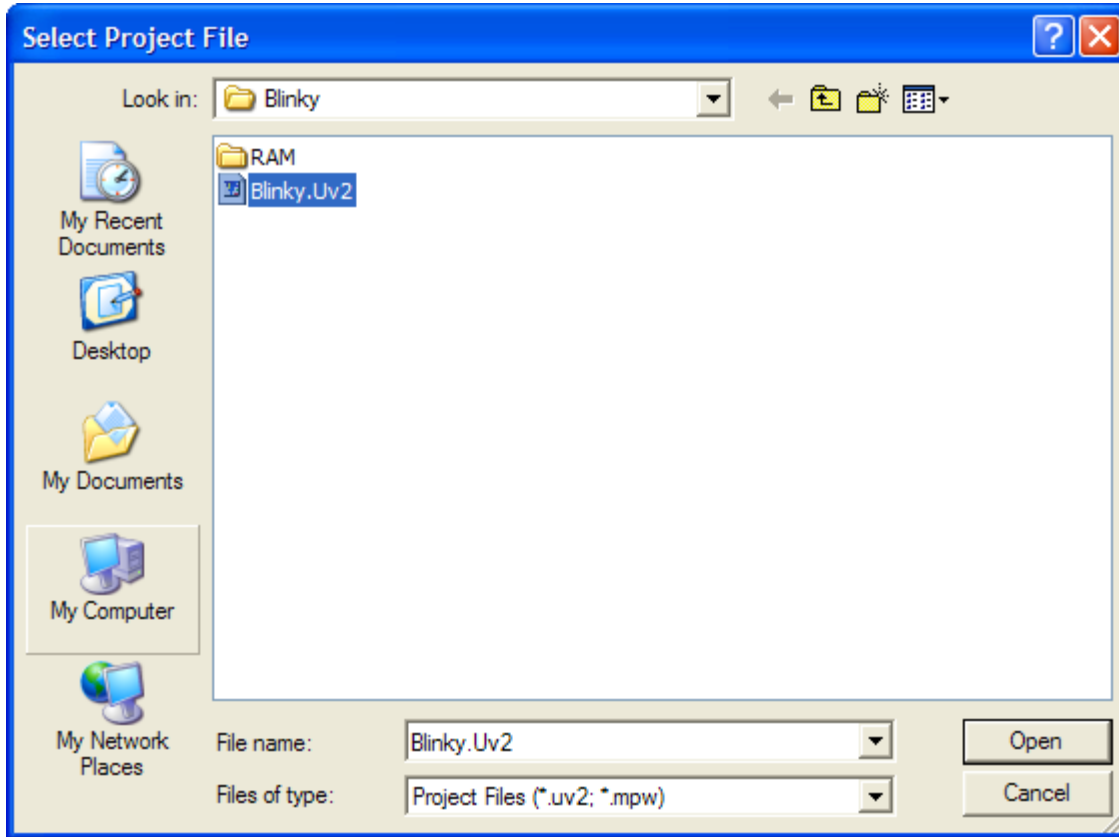
## Software Setup

1. Install the Keil MDK-ARM Development tools (either the Evaluation or Licensed version) on your PC.
2. Extract the BLINKY project from the LN7A404.ZIP file to a folder on your local hard drive.

## BLINKY Example Program

The BLINKY example program is a simple application that blinks the GPIO0 and GPIO1 LEDs continuously.

- To load the BLINKY example project select **Open Project** from the **Project** menu and open **BLINKY.UV2** from the folder chose in Software Setup, Step 2.



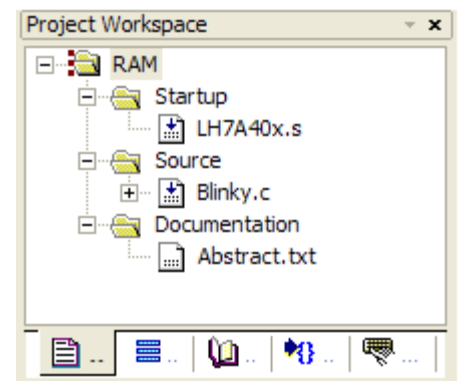
- When the BLINKY project opens, the  $\mu$ Vision Project Workspace window displays the source files that make up the project.

- LH7A40x.s** contains the CPU startup code for the Sharp LN7A404 device.
- Blinky.c** contain the application modules that toggles the LEDs.

The **Abstract.txt** file contains documentation about the project.

## Editing BLINKY

You may edit Blinky.c or review the source code. Double click on Blinky.c in the Files page of the **Project Workspace** window.  $\mu$ Vision loads and displays the contents of Blinky.c in an editor window.



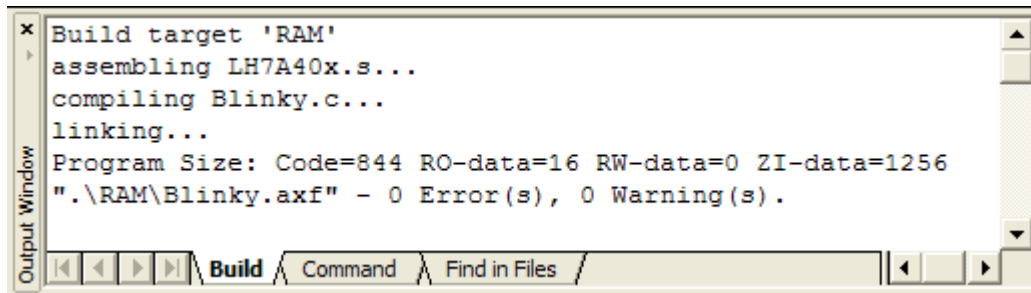
## Building BLINKY

Build the BLINKY example program following this procedure. The executable files are placed in an output folder, ready for downloading.



When you are ready to compile and link your project, use the **Build Target** command from the **Project** menu or the Build toolbar. On the Build toolbar, select the appropriate target for your program.

$\mu$  Vision translates and links the source files and creates an absolute object module that you may load into the  $\mu$  Vision debugger for testing. The status of the build process displays in the **Build** page of the **Output Window**.



```
Build target 'RAM'
assembling LH7A40x.s...
compiling Blinky.c...
linking...
Program Size: Code=844 RO-data=16 RW-data=0 ZI-data=1256
"..\RAM\Blinky.axf" - 0 Error(s), 0 Warning(s).
```

Once the BLINKY application is built, you are ready to start debugging.

## Debugging BLINKY



Use the **Start/Stop Debug Session** toolbar button or the  $\mu$  Vision command **Debug** — **Start/Stop Debug Session** to connect the debugger to the Keil ULINK2 Adapter. The CPU stops and connects to the Embedded ICE of the LN7A404 device.

When the debugger starts, the application program is downloaded to the LN7A404 target board's RAM and the program executes then halts at the start of the `main()` function. From here you can use the debugging buttons or commands to verify the application executes properly.

For more details on the debugging features, refer to the  [\$\mu\$  Vision User's Guide](#).