

The latest version of this document is here: [www.keil.com/appnotes/docs/apnt\\_318.asp](http://www.keil.com/appnotes/docs/apnt_318.asp)

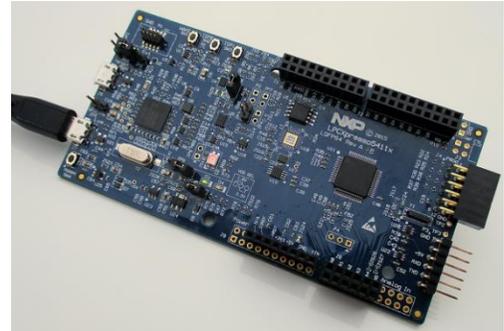
The Video describing Multi-Core Debugging: [www.youtube.com/watch?v=IMX-2lrV7Zs](http://www.youtube.com/watch?v=IMX-2lrV7Zs)

### Introduction:

The purpose of this tutorial is to explain the video created to demonstrate multi-core debugging using Keil µVision. Debug adapters such as CMSIS-DAP and any Keil ULINK can be used. This tutorial uses the NXP LPCXpresso54114 evaluation board. The LPC54114 contains an Arm Cortex®-M4 and a Cortex®-M0+.

Keil MDK has an evaluation version that limits code and data size to 32 Kbytes. Most Keil examples are under this 32K. The addition of a license number will turn it into a full, unrestricted version. Contact Keil sales for a temporary full version license to evaluate MDK with larger programs or Keil Middleware. MDK includes a full version of Keil RTX™ RTOS with all source code included. See [www.keil.com/NXP](http://www.keil.com/NXP) for information concerning Keil support of NXP products.

**ULINKplus:** Examples are provided showing ULINKplus Power measurement, interrupt tracing and many other features.



For the lab using the Keil MCB54110 board: [www.keil.com/appnotes/docs/apnt\\_295.asp](http://www.keil.com/appnotes/docs/apnt_295.asp)

For the lab using the LPCXpresso5411x: [www.keil.com/appnotes/docs/apnt\\_295.asp](http://www.keil.com/appnotes/docs/apnt_295.asp)

### About the Example programs:

1. There are two µVision projects: mailbox\_cm0plus.uvprojx and mailbox\_cm4.uvprojx.
2. They are for the Cortex-M0+ and Cortex-M4 respectively. Cortex-M4 is deemed to be the master processor.
3. The Cortex-M0+ runs in SRAM1 and data is stored in SRAM2.
4. The Cortex-M4 runs in Flash memory and data is stored in SRAM0.
5. The Cortex-M0+ executable is converted to core1\_image.bin and then provided to the Cortex-M4 project.
6. The Cortex-M4 project requires this bin file. This means that the Cortex-M0+ must always be built first if changed.
7. You can debug the Cortex-M4 only, or only the Cortex-M0+ or both at the same time.
8. If one project is being debugged, the other program runs standalone.
9. Debugging both requires running two instances of µVision. The MDK license permits this.
10. The Cortex-M4 is running Keil RTX5 RTOS.

### Operation of the Example:

1. The Cortex-M4 uses an NXP mailbox to request the Cortex-M0+ to toggle the LED. Interrupt 47 is used.
2. The Cortex-M0+ toggles the LED and sends an acknowledgement back to the Cortex-M4 via the Mailbox.
3. The MAILBOX feature is provided by the NXP SDK.

### Features of the two Multi-Core Projects:

1. The Cortex-M0+ does not have Serial Wire Viewer (SWV) which is a data trace. The Cortex-M4 does have SWV.
2. The Cortex-M0+ hardware breakpoints cannot be used because the program is in RAM. Software breaks are used.
3. The Cortex-M4 hardware breakpoints work. You can set/unset them on-the-fly while the program runs.
4. Both cores support Keil Event Recorder since this does not use SWV. You can easily annotate your code.
5. A printf utility is available that does not use a UART. Data is displayed in µVision's Debug printf Viewer.

<b>General Information, Hardware and Software:</b>	<b>3</b>
1. NXP Evaluation Boards, 6 Steps, Keil MDK Software Installation:	3
2. Software Pack Installation Process:	4
3. Copy the Examples to your PC:	5
4. Configuring, Testing and Using the on-board NXP LPC-Link2 Debug Adapter:	5
<b>Debugging Cortex-M4 Example Program: Cortex-M0+ runs free</b>	<b>6</b>
1. Open, Compile and RUN the Project:	6
2. Viewing printf Messages:	7
3. Watch Window:	7
<b>Debugging both Cortex-M4 and Cortex-M0+</b>	<b>8</b>
1. Debugging Both Cores:	8
2. Open Cortex-M0+:	8
3. Run the Cortex-M0+ Program:	8
4. Breakpoints (Hardware and Software):	8
<b>Debugging only the Cortex-M0+</b>	<b>9</b>
1. Debug Only the Cortex-M0+:	9
2. Exit mailbox_CM4:	9
3. Stop mailbox_CM0plus:	9
4. Cycle LPCXpresso Board Power:	9
5. Control the Cortex-M0+:	9
6. Rebuilding the Projects:	9
<b>ULINKplus: Debug with Serial Wire Viewer and Exceptions:</b>	<b>10</b>
1. Debug SWV Data Trace:	10
2. Connections to the Target:	10
3. Open and Build the Projects:	10
4. Run the Cortex-M4 Project:	10
5. Debug printf Viewer:	10
6. View Interrupts:	11
7. Trace Data Window:	11
8. ULINKplus Exceptions in the System Analyzer:	12
9. Measure the Duration of the MAILBOX Interrupt Handler:	12
<b>ULINKplus: Power Measurement:</b>	<b>13</b>
1. Measuring Power Consumption with Keil ULINKpro:	13
2. Connecting ULINKplus Power Connections to LPCXpresso Target:	13
3. Configure $\mu$ Vision to Collect Power Data:	13
4. Start $\mu$ Vision and View Power Signals:	13
5. Event Counters:	13
6. ULINKplus Window:	14
7. Power Consumption Examples:	14
<b>General Information:</b>	<b>15</b>
1. Document Resources:	15
2. Keil Products and Contact Information:	16

## 1) NXP ARM Cortex Processor Evaluation Boards:

This tutorial is a companion document for the Multi-Core Debug video using the NXP LPCXpresso54114 evaluation board. Keil supports and makes boards with other NXP processors. See [www.keil.com/NXP](http://www.keil.com/NXP) for more information.

All NXP processors implement SWV with the exception of Cortex-M0 and M0+, which have neither SWV nor ETM. The Cortex-M0+ (LPC800, LPC81x, LPC82x and LPC83x) has a MTB Micro Trace Buffer. All have non-intrusive read/write to memory locations (for Watch, Memory and Peripheral windows), hardware breakpoints and Watchpoints. Many NXP Cortex-M3, M4 and M7 have ETM instruction trace which also provides Code Coverage and Performance Analysis.

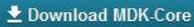
Keil produces the MCB54110 full feature board. See [www.keil.com/NXP](http://www.keil.com/NXP) for more information.

### Summary: Five Easy Steps to Get Connected and Configured:

1. Obtain and install Keil MDK Core (evaluation version) on your PC. Use the default directory C:\Keil\_v5\.
2. Download the Software Pack for LPC54000\_DFP and the examples.
3. Program the LPC-Link firmware into the LPCXpresso board with LPScript.
4. Power the LPCXpresso board and start  $\mu$ Vision and start debugging !
5. If you have a Keil ULINK*plus*, you can learn how to determine power consumption, interrupt display and more.

These steps are detailed in the next few pages.

### Keil MDK Core Software Download and Installation: This tutorial used MDK 5.27.

1. Download MDK Core from the Keil website. [www.keil.com/mdk5/install](http://www.keil.com/mdk5/install) 
2. Install MDK into the default folder. You can install into any folder, but this lab uses the default C:\Keil\_v5\
3. We recommend you use the default folders for this tutorial. We will use C:\00MDK\ for the examples.
4. If you install MDK or the examples into different folders, you will have to adjust for the folder location differences.

### Notes on Licensing:

1. You can use the evaluation version (MDK-Lite) for this lab. No license is needed.
2. To try the Middleware examples or to compile code greater than 32K, you can obtain a one-time free 7 day MDK Pro license in File/License Management. If you are eligible, this button is visible: 
3. This gives you unlimited code size compilation and access to Keil Middleware. Contact Keil Sales to extend this license for evaluation purposes.

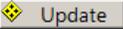
**TIP:** You can easily control both processors using two instances of  $\mu$ Vision. A standard MDK license allows this operation as does the unlicensed Evaluation version. You do not need to purchase two licenses to operate the two Cortex cores as demonstrated in this document.

## 2) Software Pack Installation Process:

A Software Pack is a zip file with a .pack file extension. It can contain header files, Flash programming algorithms, examples, documentation and more. The contents of a Pack are described by a .pdsc file contained inside the Pack. A Pack is installed using the  $\mu$ Vision Pack Installer utility or installed manually. Packs are downloaded free from the web.

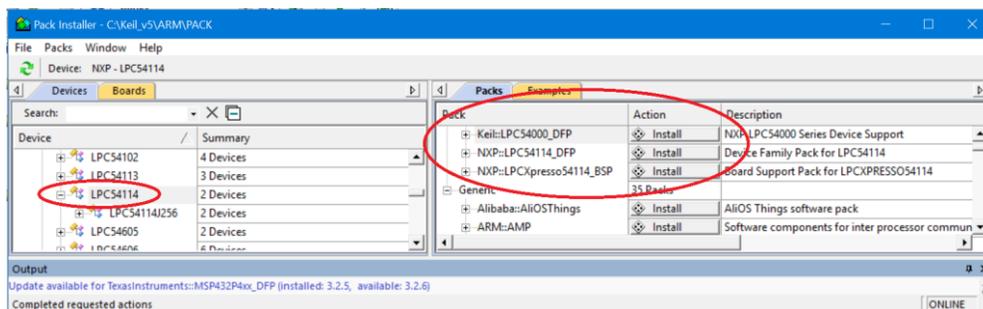
### 1) Start $\mu$ Vision and open Pack Installer (PI):

When the first MDK install is complete and if you are connected to the Internet,  $\mu$ Vision and Software Packs will automatically start. Otherwise, follow Steps 1 and 2 below.

1. Connect your computer to the Internet. This is normally needed to download the Software Packs unless you have a standalone Pack. In this case, import it with Pack Installer or double-click on it to manually install it.
2. Start  $\mu$ Vision by clicking on its desktop icon:  The Pack descriptions will download on the initial  $\mu$ Vision run.
3. Open the Pack Installer by clicking on its icon:  A Pack Installer Welcome screen will open. Read and close it.
4. If there are any Updates available, you can download them now if they are applicable. 
5. The window below opens up: Select the Devices tab. Scroll down and expand NXP. Select LPC54114 as shown below. You could also select a specific NXP processor but in this case one Pack supports all of them at this time.
6. Alternatively, you can click on the Boards tab and select LPCXpresso54114 to filter the Packs or Examples tabs.

**TIP:** The Devices and Boards tabs are used to filter the items displayed on the right side in the Packs and Examples tabs.

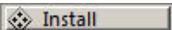
7. Note: "ONLINE" is displayed at the bottom right. If "OFFLINE" is displayed, connect to the Internet.



**TIP:** If there are no entries shown because you are not connected to the Internet when Pack Installer opened, select Packs/Check for Updates or  to refresh once you have connected to the Internet. This is not done automatically.

### 2) Install the NXP LPC54000 Packs:

**TIP:** **\_DFP** = Device Family Pack and **\_BSP** = Board Support Package.

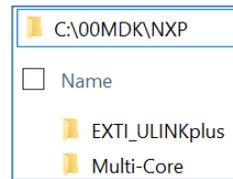
1. Click on the Packs tab.
2. Beside **Keil::LPC54000\_DFP** as shown above. Select the Install icon. 
3. Repeat for **NXP::LPC54114\_DFP** and **NXP::LPCXpresso54114\_BSP**.
4. The latest Packs will download and install to C:\Keil\_v5\ARM\PACK\NXP\ by default. This download can take two to four minutes depending on your Internet connection speed.
5. The Pack's status will then be indicated by the "Up to date" icon: 
6. Close Pack Installer. Open it any time by clicking: 
7. If a dialog box opens stating the Software Packs folder has been modified, select Yes to "Reload Packs ?"

**TIP:** You can also install a Pack manually. A Pack has a file extension .pack. It is an ordinary zip file with the extension changed so it is recognized by  $\mu$ Vision. You can download the Pack from the web or transfer the file in any other way. Double-click on this file and it will automatically be recognized (.pack) and installed by Pack Installer.

**TIP:** You can create your own Pack to distribute a DFP, BSP or an SDK. This is a way to distribute confidential material.

### 3) Copy the Examples to your PC:

1. There are two examples contained where you got this tutorial: [www.keil.com/appnotes/docs/apnt\\_318.asp](http://www.keil.com/appnotes/docs/apnt_318.asp)
2. Start Microsoft File Explorer.
3. Create a folder C:\00MDK\NXP\
4. Unzip the example files into this folder.
5. Your folder structure will look like this:



**TIP:** You can put examples into any folder you prefer. But this tutorial will use C:\00MDK\NXP for simplicity.

### 4) Using the NXP LPC-Link2 On-board Debug Adapter:

A small processor located on the target board acts as the Debug Adapter. On the LPCXpresso54114 board, this extra processor is U12. Connect a USB cable to connector J7 Link to use LPC-LINK2. JP5 DFULink must be removed.

LPC-Link2 in CMSIS-DAP mode provides run control debugging, Flash and RAM programming and Watchpoints. Hardware breakpoints can be set/unset while the program runs. Variables and registers in the Watch, Memory and Peripheral windows are updated in real-time as well as the RTX System kernel awareness viewer. At this time, the LPC-Link2 version of CMSIS-DAP does not support Serial Wire Viewer (SWV). Use any Keil ULINK to obtain SWV.

LPC-Link2 has both CMSIS-DAP and J-Link Lite modes by using different firmware using LPCScript. Multi-Core debugging is not supported by J-Link. Please note J-Link Lite cannot be used in a commercial setting. A license must be purchased. CMSIS-DAP as used by LPC-LINK2 has an Apache 2.0 license and is therefore free to use anywhere.

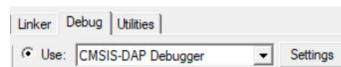
#### To Install/Update the LPC-Link2 firmware on the LPCXpresso54114:

1. Download and Install NXP LPCScript on your PC: [www.nxp.com/lpcscript](http://www.nxp.com/lpcscript)
2. Install the DFULink boot jumper (JP5) and plug a USB cable from your PC to connector J7.
3. Open the LPCscript folder under Windows Start/All Programs/LPCScript.
4. Run the “Program LPC-Link2 with CMSIS-DAP” utility.
5. Follow the instructions provided in the terminal window.
6. When finished, remove JP5 DFULink and cycle the power to the board.

**Trouble ?** Try a different USB port on your PC and/or cycle the power to the board and try programming again. Sometimes LPCScript indicates an error which may or may not be important.

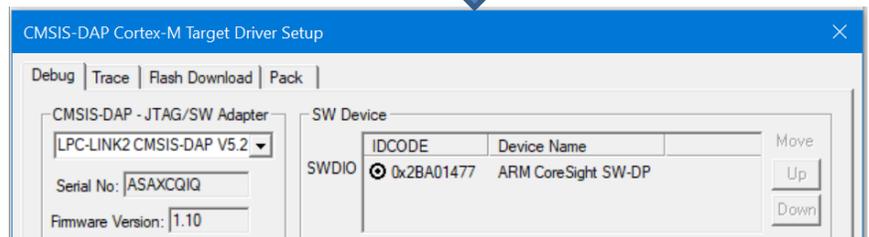
#### Testing LPC-Link2 with Keil µVision:

1. Start µVision.  Connect the LPC board USB connector J7 Link to a USB port on your PC.
2. Select Project/Open Project and navigate to C:\00MDK\NXP\Multi-Core\cm4.
3. Select the µVision project file: mailbox\_cm4.uvprojx and select Open.
4. Select Options for Target  or ALT-F7. Click on the Debug tab.
5. CMSIS-DAP will be selected: 
6. Click on Settings: and the Target Driver Setup will display as shown here: 



#### Correct Operation is Indicated by:

1. LPC-LINK2 CMSIS-DAP will be displayed under CMSIS-DAP. This means µVision is connected to the LPC-Link2
2. In the SW-Device box an IDCODE with SWDIO must be displayed. This means µVision is connected to the Arm CoreSight® debugger via the onboard LPC-Link2.
3. This must work correctly as shown before you can continue. You might have to reprogram the LPC board again. Make sure JP5 DFULink is removed.
4. If everything looks good, Click OK twice to return to the main menu. **The LPC-Link2 Adapter is ready to use.**



## 1) Debugging the Cortex-M4 example program: (the Cortex-M0+ runs free)

Now we will run the Multi-Core example. There are two projects: one for the Cortex-M0+ and another for the Cortex-M4. It is essential the Cortex-M0+ be compiled first as its output is then used by the Cortex-M4 build. If the builds are not completed in this order, then this error will be seen during the Cortex-M4 build: incbin.S(16): error: A1270E: File "core1\_image.bin" not found. This means you need to build the Cortex-M0+ project and try the Cortex-M4 build again.

### Hardware Connections:

1. Connect a USB cable to J7 Link from your PC. JP5 DFULink must be open.

### Open the Cortex-M0+ example program: (this was probably already done on the previous page)

1. Start  $\mu$ Vision by clicking on its desktop icon. 
2. Select Project/Open Project from the  $\mu$ Vision main menu.
3. Navigate to C:\00MDK\NXP\Multi-Core\cm0plus.
4. Select the  $\mu$ Vision project file: mailbox\_cm0plus.uvprojx and select Open.

### Build the Cortex-M0+ Project:

1. Compile the source files by clicking on the Rebuild icon.  Progress is indicated in the Build Output window.
2. It will compile with no warnings or errors.

### Open the Cortex-M4 example program:

1. Select Project/Open Project from the  $\mu$ Vision main menu.
2. Navigate to C:\00MDK\NXP\Multi-Core\cm4.
3. Select the  $\mu$ Vision project file: mailbox\_cm4.uvprojx and select Open.

### Build the Cortex-4 Project:

1. Compile the source files by clicking on the Rebuild icon.  Progress is indicated in the Build Output window.
2. It will compile with no warnings or errors.

### Program Flash Memory and Run the programs:

1. Enter Debug mode by clicking on the Debug icon.  Select OK if the Evaluation Mode notice appears. Flash programming progress is indicated in bottom left corner.
2. Click on the RUN icon  to start the Cortex-M4 program.
3. If the green LED D2 is blinking, the program is now running ! Both Cores are running.

***The green LED D2 will now flash ! with a 1 second interval and 50% duty cycle.***  
**Now you know how to compile a program, load it into the LPC54114 Flash, run it and stop it.**

4. Stop the program with the STOP icon.  The program will probably stop in the osRtxIdleThread function.

**TIP: Single Step:**  (F11): If you click inside a source window to bring it into focus, the program will step one source line at a time. If you click inside the Disassembly window to bring it into focus, it will then step one instruction at a time.

**What is Happening:**  
**The Cortex-M4 processor is under debug control. You can start and stop it and set breakpoints and much more.**  
**The Cortex-M0+ is free running. The Cortex-M0+ program was loaded and was started by the Cortex-M4.**  
**To debug the Cortex-M0+, we will need to start a second instance of  $\mu$ Vision.**

The board will run the example program stand-alone. It is now permanently programmed in the Flash until reprogrammed.

## 2) Viewing printf Messages:

LPC-Link2 contains a serial COM port you can use.  $\mu$ Vision also has a printf window. See page 10 for information.

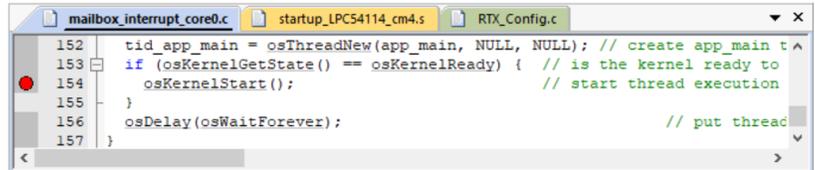
### Configuring A Terminal Program:

1. Locate the COM port LPC-Link2 is using in Windows using Device Manager.
2. Configure your favourite terminal program such as PuTTY or Tera-Term in the usual way.
3. The speed is 115,200 baud.



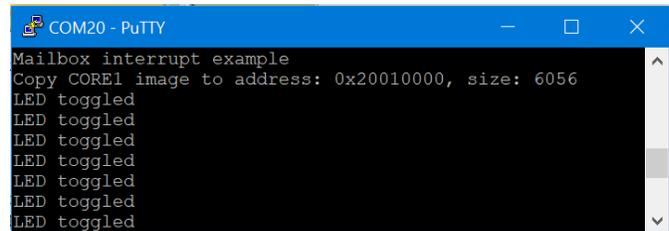
### Restart and Configure the Program:

1. Stop the program  and click the RESET icon . You also exit and re-enter Debug mode.  
2. Set a breakpoint in mailbox\_interrupt\_core0.c near line 154 as shown here at osKernelStart();.
3. The program will stop just before the start of RTX.



### RUN the program:

1. Click RUN: 
2. The program will run to the breakpoint.
3. Examine the terminal program.
4. You can see the Cortex-M4 program has loaded the Cortex-M0+ code and started it.
5. Remove the breakpoint.
6. Click RUN: 
7. The program will start and the LED will flash.
8. "LED toggled" will display in the terminal window as shown here:



## 3) Watch Window:

$\mu$ Vision has Watch windows that can be used to view and modify variables. Variables can be global, static, structures or any variable that is always in focus. It is unable to display local or automatic variables. Just make these global or static.

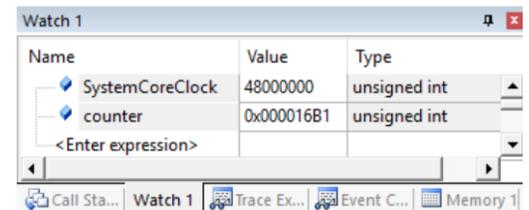
1. Stop the program  and exit Debug mode. 
2. Create a global variable counter near line 19 in the file mailbox\_interrupt\_core0.c: **uint32\_t counter = 0;**
3. Add the line **counter++;** in the function app\_main just after Delay(500); near line 112.
4. Compile the source files by clicking on the Rebuild icon.  Progress is indicated in the Build Output window.
5. It will compile with no warnings or errors.

### Program Flash Memory and Run the programs:

1. Enter Debug mode by clicking on the Debug icon. 
2. Click on the RUN icon  to start the Blinky program.
3. Right-click on the variable **counter** and select Add 'counter' to... and choose Watch 1.
4. counter will be displayed and will update while the program runs.
5. counter increments whenever the Cortex-M4 is running.

**TIP:** You can change a variable in a Watch window by clicking on it.

**TIP:** You can also add a Memory window in the same manner.

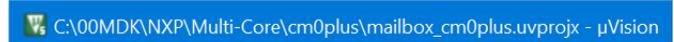


## 1) Debugging both the Cortex-M4 and Cortex-M0+:

We have shown how to debug the Cortex-M4 in the LPC54114 processor. The Cortex-M0+ processor is running by itself. We will now connect to the Cortex-M0+ by using a second instance of  $\mu$ Vision.

**TIP:** You can position the two instances of  $\mu$ Vision side-by-side on one screen, on top of each other or a split screen with two monitors. To distinguish them, refer to the header which indicates which CPU it is referenced to as shown below:

**TIP:** You can see a source file from one instance of  $\mu$ Vision in the other instance. *Be careful to modify the correct one.*



1. Have  $\mu$ Vision still running with the Cortex-M4 project.
2. Start a second instance of  $\mu$ Vision by clicking on its icon. 

## 2) Open the Cortex-M0+ example program in the Second Instance of $\mu$ Vision:

1. Select Project/Open Project from the  $\mu$ Vision main menu.
2. Navigate to C:\00MDK\NXP\Multi-Core\cm0plus.
3. Select the  $\mu$ Vision project file: mailbox\_cm0plus.uvprojx and select Open.

## 3) Run the Cortex-M0+ Program:

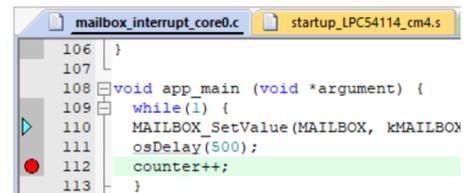
1. **Note:** The Cortex-M0+ source was built and incorporated into the Cortex-M4 executable previously.
2. Enter Debug mode:  The green LED will continue flashing. RUN is automatically selected.
3. **Cortex-M4:** Note that variable counter is incrementing in Watch 1. This means the Cortex-M4 is running.
4. **Cortex-M0+:** Note that variable toggle\_count is incrementing in Watch 1. This means the Cortex-M0+ is running.
5. Stop  and RUN  on each processor in turn to see the effects.
6. You can easily tell if a processor is running by the Watch 1 variables.

## 4) Breakpoints:

Cortex-M processors have hardware breakpoints that can be set and unset while the program is running. They can be used only in the first 512 Mbyte (0x2000 0000 or decimal 536,870,912 bytes). At higher addresses, software breakpoints must be used. The RAM on this device starts at 0x2000 0000 so software breakpoints must be used.

### Cortex-M4 Breakpoints:

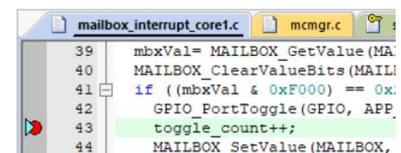
1. **With both CPUs running**, in the gray area left of the line numbers in the file mailbox\_interrupt\_core0.c, (grey indicates instructions are there) click on the counter++; line to set a hardware breakpoint. A red circle will appear and presently the program will stop at this point. This area is in the Flash memory.
2. Unselect the breakpoint and click RUN.



```
106 }
107
108 void app_main (void *argument) {
109     while(1) {
110         MAILBOX_SetValue(MAILBOX, kMAILBOX
111             osDelay(500);
112         counter++;
113     }
```

### Cortex-M0+ Breakpoints:

1. **With both CPUs running**, in the gray area left of the line numbers in the file mailbox\_interrupt\_core1.c, click opposite toggle\_count++; near line 43:
2. An error window will open stating you can't set a breakpoint here. This is because the Cortex-M0+ program is running in RAM > 0x2000 0000.
3. Stop the Cortex-M0+. 
4. Click opposite toggle\_count++; near line 43.
5. Click RUN  and the program will run and stop on the software breakpoint.
6. Remove the breakpoint.



```
39     mbxVal= MAILBOX_GetValue(MA
40     MAILBOX_ClearValueBits(MAIL
41     if ((mbxVal & 0xF000) == 0x
42     GPIO_PortToggle(GPIO, APP
43     toggle_count++;
44     MAILBOX_SetValue(MAILBOX,
```

### Breakpoint Notes:

1. The Cortex-M4 code runs in Flash so hardware breakpoints work. They can be set/unset while the program runs.
2. The Cortex-M0+ code runs in RAM. Software breakpoints are used. The program must be stopped to set/unset.

## 1) Debugging Only the Cortex-M0+: (the Cortex-M4 is free running)

At this point, the two Blinky programs can run on both cores from Flash memory without any intervention by  $\mu$ Vision. If you exit Debug mode  for both projects and cycle the board power, the LED will blink indicating both cores are running.

## 2) Exit mailbox\_CM4:

1. Bring the CM4 instance of  $\mu$ Vision into focus by clicking on or selecting it.
2. Stop the program  and exit Debug mode for M4 Blinky .
3. Exit this instance of  $\mu$ Vision.

**TIP:** If you leave this instance running but not in Debug mode, it will not affect the target. It can be useful to keep it running because if you change the program in mailbox\_CM0plus, you need to compile it first and then compile the mailbox\_CM4 project which uses the executable from the mailbox\_CM0plus. This is then programmed into the Flash. You can also have one instance running and switch the project back and forth as needed.

## 3) Stop mailbox\_CM0plus:

1. Bring the Cortex-M0+ instance of  $\mu$ Vision into focus by clicking on it or selecting it.
2. Stop the program  and exit Debug mode for M0+ Blinky .
3. Leave this instance of  $\mu$ Vision running.

## 4) Cycle the LPCXpresso Board Power:

1. Unplug the USB cable and plug it back in.
2. The green LED will start blinking. If it does not, this means that something is wrong with either core program.

## 5) Control the Cortex-M0+ Core with $\mu$ Vision:

1. On  $\mu$ Vision running the Cortex-M0+ project (assuming the M4 was closed above) enter Debug mode .
2. When  $\mu$ Vision is in Debug mode, the LED is still blinking because both cores are running.
3. Note `Connect: without Stop` is set so  $\mu$ Vision via LPC-Link2 merely connects to the Cortex-M0+ core and does not stop it.

## 6) Rebuilding the Projects:

If you make a modification to the Cortex-M0+ program, you must first build it and then build the Cortex-M4 project and program this .axf file into the Flash memory. The reason is the M4 project uses the executable in the form of a bin file. We will modify the `toggle_count` to decrement instead of incrementing.

1. Stop the mailbox\_CM0plus project  and exit Debug mode .
2. In mailbox\_cm0plus.c, change the increment on `toggle_count` to decrement:
3. Build the cm0plus project. .
4. Select Project in  $\mu$ Vision and select mailbox\_cm4.uvprojx:
5. Build the cm4 project. .
6. Program the Flash memory using the Load icon: .
7. Cycle the power to the LPCXpresso board. The green LED will start to blink.
8. Select Project in  $\mu$ Vision and select mailbox\_cm0plus.uvprojx:
9. Enter Debug mode .  $\mu$ Vision will connect to the Cortex-M0+ core but will not stop it leaving it running.
10. `toggle_count` will now be decrementing.
11. These are the steps to take when modifying the mailbox\_CM0plus project.
12. Stop the program  and exit Debug mode .

```
41 if ((mbxVal & 0xF000) ==
42     GPIO_PortToggle(GPIO,
43         toggle_count--;
```

```
✓ 1 C:\00MDK\NXP\Multi-Core\cm0plus\mailbox_cm0plus.uvprojx
2 C:\00MDK\NXP\Multi-Core\cm4\mailbox_cm4.uvprojx
```

```
✓ 1 C:\00MDK\NXP\Multi-Core\cm4\mailbox_cm4.uvprojx
2 C:\00MDK\NXP\Multi-Core\cm0plus\mailbox_cm0plus.uvprojx
```

Name	Value	Type
toggle_count	-149	int
<Enter expression>		

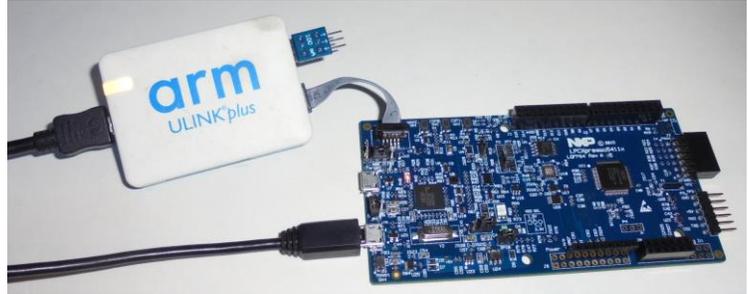
This is the end of the LPC-Link2 examples. The next examples use the Keil ULINKplus Debug Adapter.

## 1) ULINKplus: Debug with Serial Wire Viewer (SWV) and Exceptions (interrupts):

The Keil ULINKplus offers Power Measurement and Serial Wire Viewer. See [www.keil.com/mdk5/ulink/ulinkplus](http://www.keil.com/mdk5/ulink/ulinkplus)  
The examples using SWV are pre-configured. To configure SWV, see [www.keil.com/apnotes/docs/apnt\\_295.asp](http://www.keil.com/apnotes/docs/apnt_295.asp)

## 2) Connection:

1. Connect a Keil ULINKplus to P1 as shown.
2. Power the board USB to J5 Target PWR.  
Note this is different when using LPC-Link2



## 3) Open and Build the Projects:

1. In  $\mu$  Vision, select Projects/Open Projects.
2. Navigate to  
C:\00MDK\NXP\EXTI\_ULINKplus\cm0plus
3. Open mailbox\_cm0plus.uvprojx
4. Build the cm0plus project.
5. Select Project/Open Projects and navigate to C:\00MDK\NXP\EXTI\_ULINKplus\cm4
6. Select mailbox\_cm4.uvprojx. The Cortex-M4 project will be loaded in  $\mu$  Vision.
7. Build the cm4 project.

## 4) Run the Cortex-M4 Program:

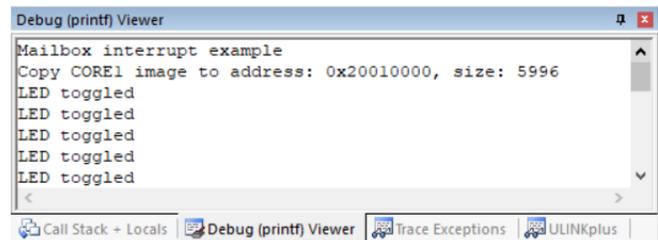
1. Enter Debug mode. and click RUN . The Cortex-M0+ will run standalone.
2. The green LED D2 will blink at 1 second intervals with a 50% duty cycle as in the previous example.

## 5) Debug printf Window:

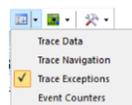
$\mu$  Vision offers printf display using the ITM register Port 0 which is a component of SWV. Your printf statements are displayed in the Debug (printf) Viewer as shown here:

No UART hardware is used. It is nearly non-intrusive to your program. It uses only one write to ITM Port 0 per character.

1. The projects must be running.
2. Click on the Debug (printf) Viewer tab.
3. If this is not visible, select View/Serial Windows and then select Debug (printf) Viewer.
4. Note LED toggled is displayed as the program is running.
5. Stop the mailbox\_CM0plus project
6. Scroll to the top in the printf window as shown here. These are from the printf statements in the program.
7. Click RUN to start the program.



**TIP:** Finding and Opening Windows: To bring a window in focus, normally you just click on its tab. If the window is not currently active, you can open it by going to View and select it or click an icon. If a window is already active and opened somewhere, it will have an orange background as shown here:

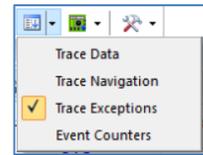


If you have trouble finding an activated window, you can deactivate and then reactivate it to make it more visible. In this example, only Trace Exceptions is already displayed.

**TIP:** Cortex-M0+ does not have SWV. Only Cortex-M3 and higher has SWV.  $\mu$  Vision has Event Recorder which can output information as it does not use SWV. It uses DAP: Debug Access Port via a debug adapter such as LPC-Link2.

## 6) View Interrupts:

1. Open the Trace Exceptions window by clicking on its tab or selecting the Trace icon and Trace Exceptions as shown here: 
2. In the Trace Exceptions window as shown below, click in the Count area and any active exceptions will rise to the top of this window arranged by the number of occurrences.
3. You can see the SysTick, SVCcall and the Mailbox exceptions with associated statistics.
4. This window updates in real-time and is non-intrusive. It uses SWV.



#	Name	Count	Total Time	Min Time In	Max Time In	Min Time Out
15	SysTick	8527	44.964 ms	4.854 us	17.854 us	982.146 us
11	SVCcall	22	260.792 us	5.312 us	36.708 us	2.375 us
47	MAILBOX	19	1.394 ms	1.208 us	78.542 us	92.458 us
2	NonMaskable	0	0 s			

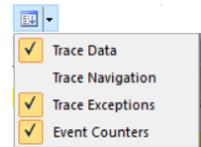
## 7) Trace Data Window:

The Trace Data window can display Data Trace frames from Serial Wire Viewer (SWV) including interrupts, data reads and writes, counters and more. This works with any Keil ULINK. CMSIS-DAP V2 supports SWV. Currently LPC-LINK2 supports only CMSIS-DAP V1 and therefore it has no SWV support.

If you have a Keil ULINK<sub>pro</sub>, ETM Instruction trace can be captured. ETM provides a list all the instructions executed. Code Coverage and Performance Analysis is offered in  $\mu$ Vision. The processor needs to have ETM implemented and a 20 pin CoreSight connector.

1. Stop the program  The Trace Data window is updated and displayed only while the program is stopped.

1. Open the Trace Data window by clicking on the Trace icon and selecting Trace Data as shown:
2. The Trace Data window will open and probably display mostly SysTick frames.
3. In the search box enter mailbox as shown below and press Enter.
4. Here you can see the exceptions that are displayed in the Trace Exceptions in a different format.
5. The P:0 frames are the printf frames.



6. Right click on any frame and select Show Record Descriptions: This information will be displayed at the bottom. Click on various frames.

Time	Address / Port	Instruction / Data	Src Code / Trigger Addr	Function
0.001 501 521 s	P: 0	0x65		
0.001 502 542 s	P: 0	0x64		
0.001 503 562 s	P: 0	0x0D		
0.001 504 583 s	P: 0	0x0A		
0.001 510 688 s		Exception Exit - MAILBOX		
0.001 510 875 s		Exception Return		
0.001 512 208 s		Exception Entry - SVCcall		
0.001 522 333 s		Exception Exit - SVCcall		
0.001 522 500 s		Exception Return		
0.002 398 583 s		Exception Entry - SysTick		
0.002 403 438 s		Exception Exit - SysTick		
0.002 403 604 s		Exception Return		
0.003 398 583 s		Exception Entry - SysTick		
0.003 403 688 s		Exception Exit - SysTick		

**SW Stimulus**

Port : 0 (Debug printf)

Access Size : 1 Byte

Data Value : 0x6F

**Exception Entry:** when the exception or interrupt is entered.

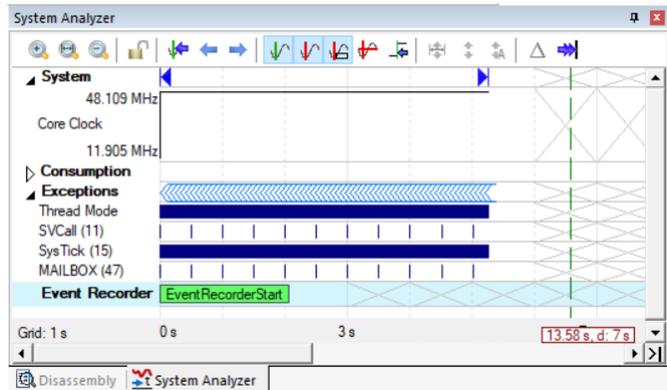
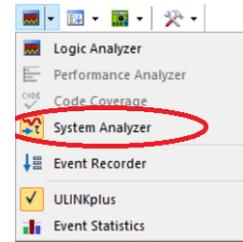
**Exception Exit:** when the exception or interrupt exits.

**Exception Return:** when all exceptions or interrupts exit. This indicates that no Cortex tail chaining is occurring.

## 8) ULINKplus Exceptions in the System Analyzer Window:

System Analyzer is a useful window to display the processor frequency, voltage and current, interrupts and the Event Recorder events.

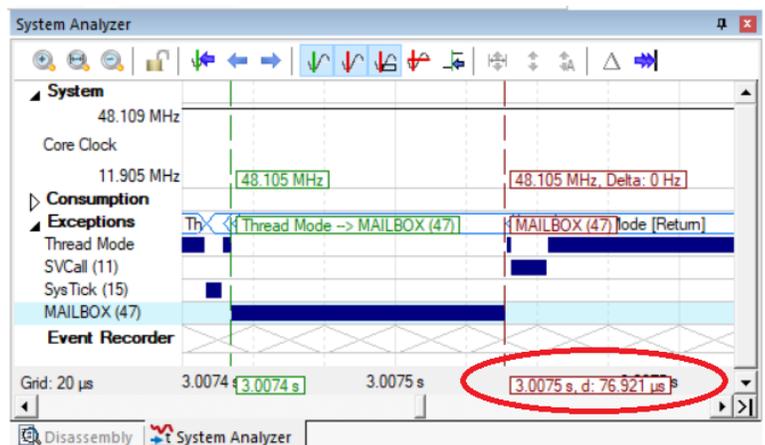
1. Open System Analyzer: The window similar to the one below opens.
2. The Consumption view has been collapsed to save space in this window.
3. Stop the program  or click Freeze Data .
4. Note the CPU frequency started out at 11.905 MHz and ended up at 48.109 MHz after the PLL initialization is complete. ULINKpro measures this frequency.
5. RTX Threads are displayed but in this case there is only one thread.
6. Exceptions (Interrupts) SVCALL, SysTick and MAILBOX are shown graphically. You can easily measure timings in this window. This is an excellent feature to show you where your interrupts are occurring in relation to other events and their duration.
7. Event Recorder is a  $\mu$ Vision feature where you can easily instrument your code.
8.  Zoom in and out.
9.  Freeze Data: Stops the collection in System Analyzer while your program continues to run.
10.  Jump to End: This is useful for scrolling to the right to locate waveforms.



11. For meaning of more icons see: [www.keil.com/support/man/docs/uv4/uv4\\_db\\_dbg\\_systemanalyzer.htm](http://www.keil.com/support/man/docs/uv4/uv4_db_dbg_systemanalyzer.htm)

## 9) Measure the Duration of the MAILBOX Interrupt Handler:

1. If you have a wheel on your mouse: Hover your mouse pointer over a MAILBOX event. Rotate the wheel to zoom into MAILBOX as shown below. The event will stay centered.
2. If you do not have a mouse wheel, use the Zoom in/out icons and the scrolling bar to keep it centered.
3. In this window you can now see the durations of the interrupts are visible.
4. Click on the left edge of MAILBOX to set the cursor and move the mouse to the right edge.
5. The time is displayed as 76.921  $\mu$ sec.
6. You can clearly see where the program switched between Thread Mode (your program) to the interrupts and back again.
7. This type of information is invaluable for finding overly long (or short) exception routines and other timing issues or to better understand various program flows.
8. System Analyzer uses SWV. Event Recorder does not. It uses DAP.



**TIP:** RTX RTOS uses the term “threads” instead of “tasks” for consistency.

## 1) ULINKplus Power Measurement:

ULINKplus provides accurate powers measurements. See [www.keil.com/mdk5/ulink/ulinkplus](http://www.keil.com/mdk5/ulink/ulinkplus)

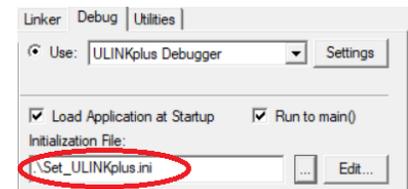
## 2) Connecting ULINKplus Power Connections to the LPCXpresso Target:

1. Stop the program  and exit Debug mode .
2. Remove the power from both the ULINKplus and the LPCXpresso board.
3. There are several small shunt resistor boards provided with ULINKplus
4. Obtain the one marked 100 mA and plug it into ULINKplus in the 3 pin Power connector in any orientation as shown on the right: 
5. Connect three jumper cables as shown. The middle pin is ground.
6. Remove the jumper from pins 2 and 3 on JP9. Connect the IN and OUT cables to JP9 Pins 2 and 3 as shown here. Orientation not important at this point.
7. Connect the ground to GND on P3 on the edge of the LPCXpresso. The ground is mandatory because the ULINKplus is completely isolated.



## 3) Configure µVision to Collect Power Data:

1. Apply power to both the ULINKplus and the LPCXpresso board.
2. Select Options for Target  or ALT-F7. Click on the Debug tab.
3. A file Set\_ULINKplus.ini is provided to initialize the ULINKplus.
4. Using the Browse icon , select Set\_Ulinkplus.ini as shown.
5. Click Edit to open this file and then OK to close the Options window.
6. Examine Set\_ULINKplus.ini and note the correct 100 mA is selected:

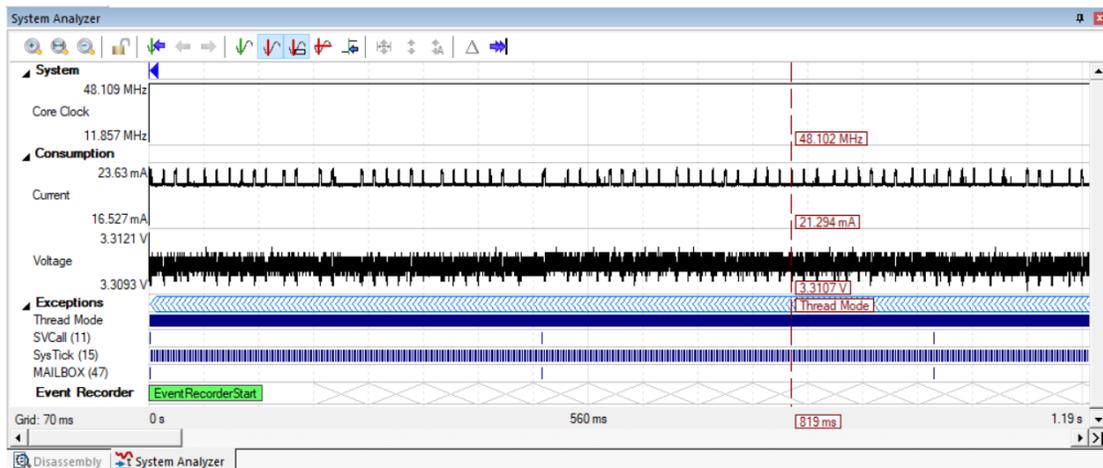


## 4) Start µVision Debugger and View Power Signals:

1. Enter Debug mode.  and click RUN . After a few seconds click STOP  or Freeze Data: 
2. Examine the System Analyzer and you can see the Current and Voltage waveforms.

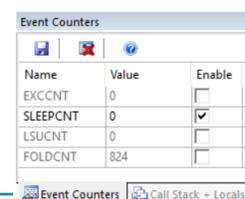
**TIP:** If the current and/or voltage have negative numbers, switch the two power IN and OUT wires.

3. Using the cursors, you can make various measurements. It is easier to do this when this window is stopped.



## 5) Event Counters:

1. Select View/Trace and select Event Counters. You can also use the Trace icon.
2. If the program is stopped, click RUN .
3. Select SLEEPCNT. Note that it remains at zero. Others will increment if enabled.

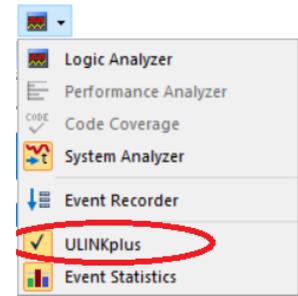
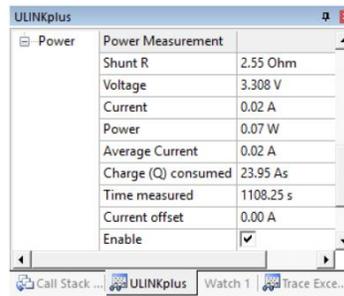


Name	Value	Enable
EXCCNT	0	<input type="checkbox"/>
SLEEPCNT	0	<input checked="" type="checkbox"/>
LSUCNT	0	<input type="checkbox"/>
FOLDCNT	824	<input type="checkbox"/>

## 6) ULINKplus Window:

Open the ULINKplus window either by clicking on its tab or with the Analysis Windows icon:

1. The ULINKplus window shows various values derived from the current and voltage waveforms provided by ULINKplus. This is shown below right:
2. Note these values are updated while your program is running.



**TIP:** You can annotate your code to select portions to collect and display this type of information using the Event Recorder. The data is displayed in the in the Event Statistics window. [www.keil.com/support/man/docs/uv4/uv4\\_db\\_dbg\\_evr\\_stat.htm](http://www.keil.com/support/man/docs/uv4/uv4_db_dbg_evr_stat.htm)

## 7) Power Consumption Example:

We will add the instruction wfi (wait for interrupt) after the line osDelay(500). This instruction puts the CPU into sleep mode until the next interrupt happens. Then, the CPU exits sleep mode and executes normally until the next wfi.

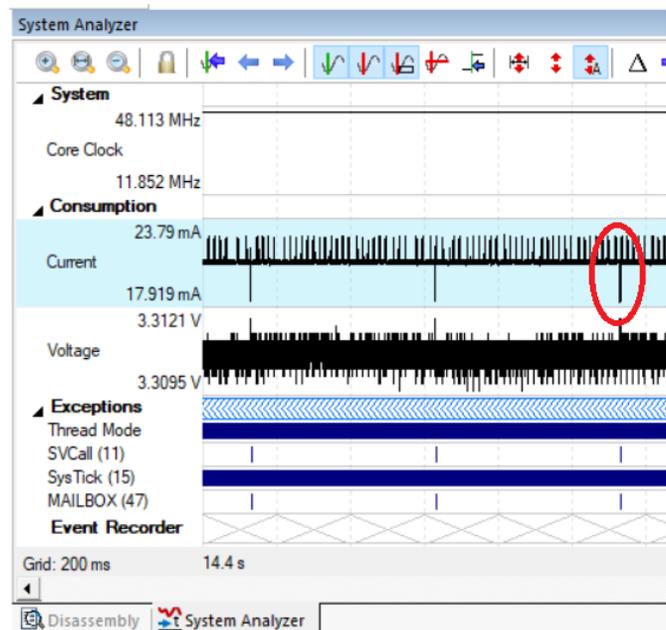
1. Stop the program and exit Debug mode .
2. In the file mailbox\_interrupt\_core0.c, near line 112, enter this compiler intrinsic: \_\_wfi(); as shown below:

```

108 void app_main (void *argument) {
109     while(1) {
110         MAILBOX_SetValue(MAILBOX, kMAILBOX_CM0Plus, 0x2000U); // perform a LED action
111         osDelay(500);
112         __wfi();
113     }
114 }
115

```

3. Build the project.
4. Enter Debug mode. and click RUN . After a few seconds click STOP or Freeze . Data:
5. Examine the System Analyzer and you can see where the Current dips with the wfi instruction.
6. These spikes are 500 msec from each other.
7. The power savings are not much but measurable as we can measure the time the processor sleeps is only 1,000 msec.
8. It is clear to see that measuring power consumption is easy with a ULINKplus.
9. If you look at SLEEPCTR you will see it incrementing as it counts cycles spent when CPU is asleep.



**This is the end of the exercises. Thank you !**

Please watch the video and other appnotes listed in the first page.

## 2) Document Resources:

See [www.keil.com/NXP](http://www.keil.com/NXP)

### Books:

1. **NEW!** **Getting Started with MDK 5:** Obtain this free book here: [www.keil.com/gsg/](http://www.keil.com/gsg/)
2. There is a good selection of books available on ARM: [www.arm.com/support/resources/arm-books/index.php](http://www.arm.com/support/resources/arm-books/index.php)
3.  $\mu$ Vision contains a window titled Books. Many documents including data sheets are located there.
4. A list of resources is located at: [www.arm.com/products/processors/cortex-m/index.php](http://www.arm.com/products/processors/cortex-m/index.php)
5. Or search for the Cortex-M processor you want on [www.arm.com](http://www.arm.com).
6. The Definitive Guide to the ARM Cortex-M0/M0+ by Joseph Yiu. Search the web for retailers.
7. The Definitive Guide to the ARM Cortex-M3/M4 by Joseph Yiu. Search the web for retailers.
8. **Embedded Systems: Introduction to Arm Cortex-M Microcontrollers** (3 volumes) by Jonathan Valvano
9. MOOC: Massive Open Online Class: University of Texas: <http://users.ece.utexas.edu/~valvano/>

### Application Notes:

1. **NEW!** ARM Compiler Qualification Kit: Compiler Safety Certification: [www.keil.com/safety](http://www.keil.com/safety)
2. Using Cortex-M3 and Cortex-M4 Fault Exceptions [www.keil.com/appnotes/files/apnt209.pdf](http://www.keil.com/appnotes/files/apnt209.pdf)
3. CAN Primer: [www.keil.com/appnotes/files/apnt\\_247.pdf](http://www.keil.com/appnotes/files/apnt_247.pdf)
4. Segger emWin GUIBuilder with  $\mu$ Vision™ [www.keil.com/appnotes/files/apnt\\_234.pdf](http://www.keil.com/appnotes/files/apnt_234.pdf)
5. Porting mbed Project to Keil MDK™ [www.keil.com/appnotes/docs/apnt\\_207.asp](http://www.keil.com/appnotes/docs/apnt_207.asp)
6. MDK-ARM™ Compiler Optimizations [www.keil.com/appnotes/docs/apnt\\_202.asp](http://www.keil.com/appnotes/docs/apnt_202.asp)
7. GNU tools (GCC) for use with  $\mu$ Vision <https://launchpad.net/gcc-arm-embedded>
8. RTX CMSIS-RTOS Download [www.keil.com/demo/eval/rtx.htm](http://www.keil.com/demo/eval/rtx.htm)
9. Barrier Instructions <http://infocenter.arm.com/help/topic/com.arm.doc.dai0321a/index.html>
10. Lazy Stacking on the Cortex-M4: [www.arm.com](http://www.arm.com) and search for DAI0298A
11. Cortex Debug Connectors: [www.keil.com/coresight/coresight-connectors](http://www.keil.com/coresight/coresight-connectors)
12. Sending ITM printf to external Windows applications: [http://www.keil.com/appnotes/docs/apnt\\_240.asp](http://www.keil.com/appnotes/docs/apnt_240.asp)
13. FlexMemory configuration using MDK [www.keil.com/appnotes/files/apnt220.pdf](http://www.keil.com/appnotes/files/apnt220.pdf)
14. Sending ITM printf to external Windows applications: [www.keil.com/appnotes/docs/apnt\\_240.asp](http://www.keil.com/appnotes/docs/apnt_240.asp)
15. **NEW!** Migrating Cortex-M3/M4 to Cortex-M7 processors: [www.keil.com/appnotes/docs/apnt\\_270.asp](http://www.keil.com/appnotes/docs/apnt_270.asp)
16. **NEW!** ARMv8-M Architecture Technical Overview <https://community.arm.com/docs/DOC-10896>
17. **NEW!** Using TrustZone on ARMv8-M with Keil MDK: [www.keil.com/appnotes/docs/apnt\\_291.asp](http://www.keil.com/appnotes/docs/apnt_291.asp)

### Useful ARM Websites:

1. CMSIS: [https://github.com/ARM-software/CMSIS\\_5](https://github.com/ARM-software/CMSIS_5) [www.arm.com/cmsis/](http://www.arm.com/cmsis/) [www.keil.com/cmsis](http://www.keil.com/cmsis)
2. ARM and Keil Community Forums: [www.keil.com/forum](http://www.keil.com/forum) and <http://community.arm.com/groups/tools/content>
3. ARM Developer: <https://developer.arm.com/>
4. ARM University Program: [www.arm.com/university](http://www.arm.com/university). Email: [university@arm.com](mailto:university@arm.com)
5. mbed™: <http://mbed.org>

**Sales In Americas:** [sales.us@keil.com](mailto:sales.us@keil.com) or 800-348-8051. **Europe/Asia:** [sales.intl@keil.com](mailto:sales.intl@keil.com) +49 89/456040-20

**Keil Distributors:** See [www.keil.com/distis/](http://www.keil.com/distis/) **DS-5 Direct Sales Worldwide:** [orders@arm.com](mailto:orders@arm.com)

**Keil Technical Support** in USA: [support.us@keil.com](mailto:support.us@keil.com) or 800-348-8051. Outside the US: [support.intl@keil.com](mailto:support.intl@keil.com)

**Global Inside Sales Contact Point:** [Inside-Sales@arm.com](mailto:Inside-Sales@arm.com)

### 3) Keil Products and Contact Information:

See [www.keil.com/NXP](http://www.keil.com/NXP)

#### Keil Microcontroller Development Kit (MDK-ARM™)

- **MDK-Lite™** (Evaluation version) 32K Code and Data Limit - \$0
- **New MDK-ARM-Essential™** For all Cortex-M series processors – unlimited code limit
- **New MDK-Plus™** MiddleWare Level 1. ARM7™, ARM9™, Cortex-M, SecureCore®.
- **New MDK-Professional™** MiddleWare Level 2. For details: [www.keil.com/mdk5/version520](http://www.keil.com/mdk5/version520).

For the latest MDK details see: [www.keil.com/mdk5/selector/](http://www.keil.com/mdk5/selector/)

Keil Middleware includes Network, USB, Graphics and File System. [www.keil.com/mdk5/middleware/](http://www.keil.com/mdk5/middleware/)

#### USB-JTAG adapter (for Flash programming too)

- **ULINK2** -(ULINK2 and ME - SWV only – no ETM) **ULINK-ME** – Electrically equal to ULINK2.
- **New ULINKplus-** Cortex-Mx High performance SWV & power measurement.
- **ULINKpro** - Cortex-Mx SWV & ETM trace.
- **ULINKpro D** – The same as a ULINKpro without ETM support. It can be used with µVision or DS-5.

For Serial Wire Viewer (SWV), a ULINK2 or a J-Link is needed. For ETM support, a ULINKpro is needed.

All ULINK products support MTB (Micro Trace Buffer) with NXP Cortex-M0+.

Keil RTX RTOS is provided with a BSD or Apache 2.0 license. This makes it free. All source code is provided.

See [https://github.com/ARM-software/CMSIS\\_5](https://github.com/ARM-software/CMSIS_5) and [www.keil.com/gsg](http://www.keil.com/gsg).  
RTX documentation is free. [www.keil.com/pack/doc/CMSIS/RTOS/html/](http://www.keil.com/pack/doc/CMSIS/RTOS/html/)  
For the feature list see: [www.keil.com/RTX](http://www.keil.com/RTX)

Keil provides free DSP libraries for the Cortex-M0+, Cortex-M3, Cortex-M4 and Cortex-M7 processors.

Call Keil Sales for details on current pricing, specials and quantity discounts. Sales can also provide advice about the various tools options available to you. They will help you find various labs and appnotes that are useful.

All products include Technical Support for 1 year. This is easily renewed.

Call Keil Sales for special university pricing. Go to [www.arm.com/university](http://www.arm.com/university)

Keil supports many other NXP processors including ARM7™ and ARM9™ series processors. See the Keil Device Database® on [www.keil.com/dd](http://www.keil.com/dd) for the complete list of NXP support. This information is also included in MDK.

For NXP Cortex-A processors, see DS-MDK: [www.keil.com/ds-mdk](http://www.keil.com/ds-mdk). Linux, Android and other OSs are supported.



#### For more information:

Sales In Americas: [sales.us@keil.com](mailto:sales.us@keil.com) or 800-348-8051. Europe/Asia: [sales.intl@keil.com](mailto:sales.intl@keil.com) +49 89/456040-20

Keil Technical Support in USA: [support.us@keil.com](mailto:support.us@keil.com) or 800-348-8051. Outside the US: [support.intl@keil.com](mailto:support.intl@keil.com).

Global Inside Sales Contact Point: [Inside-Sales@arm.com](mailto:Inside-Sales@arm.com)

Keil Distributors: See [www.keil.com/distis/](http://www.keil.com/distis/)

For more information: [www.arm.com/cmsis](http://www.arm.com/cmsis), [www.keil.com/forum](http://www.keil.com/forum) and <http://community.arm.com/groups/tools/content>

