Analog Devices EVAL-ADuCM3029 EZ-KIT
ADuCM3029: Cortex-M3 Tutorial
ARM Keil MDK 5 Toolkit Spring 2018 V 1.80 bob.boys@arm.com

The latest version of this document is here: www.keil.com/appnotes/docs/apnt_303.asp

Introduction:

The purpose of this lab is to introduce you to the ADuCM3029 Cortex®-M3 processor using the ARM® Keil® MDK toolkit featuring the IDE µVision®. We will demonstrate all debugging features available on this processer. At the end of this tutorial, you will be able to confidently work with these processors and Keil MDK.
We recommend you obtain the ***new*** **Getting Started MDK 5:** from here: www.keil.com/gsg.

Keil MDK supports and has examples for Analog Devices ARM and 8051 processors. See www.keil.com/analog-devices. Check the Keil Device Database® on www.keil.com/dd2. This list is also provided by the µVision Pack Installer utility for ARM processors only.

Keil MDK-Lite™ is a free evaluation version that limits code size to 32 Kbytes. Nearly all Keil examples will compile within this 32K limit. The addition of a valid license number will turn it into an unrestricted commercial version.

RTX RTOS: All variants of MDK contain the full version of RTX with Source Code. See www.keil.com/mdk5/cmsis/rtx/.

Why Use Keil MDK ?

MDK provides these features particularly suited for Analog Devices users:

1. µVision IDE with Integrated Debugger, Flash programmer and the ARM® Compiler toolchains. MDK is turn-key "out-of-the-box".
2. ARM Compiler 5 and ARM Compiler 6 (LLVM) are included. GCC is supported. https://launchpad.net/gcc-arm-embedded
3. Also see: https://developer.arm.com/tools-and-software
4. Dynamic Syntax checking on C/C++ source lines.
5. ***NEW!*** **Event Recorder for RTOS and User programs.** www.keil.com/support/man/docs/uv4/uv4_db_dbg_evr.htm
6. MISRA C/C++ support using PC-Lint. www.gimpel.com
7. **Compiler Safety Certification Kit:** www.keil.com/safety/
8. **TÜV** certified. SIL3 (IEC 61508) and ASILD (ISO 26262).
9. CMSIS-RTOS RTX is included. RTX has a BSD or Apache 2.0 license with source code. See www.keil.com/RTX and https://github.com/ARM-software/CMSIS_5 Also FreeRTOS support.
10. **Debug Adapters:** Keil ULINK™2, ULINK*pro*, ULINK*plus*, Segger J-Link and CMSIS-DAP.
11. ***NEW!*** Event Recorder for Keil Middleware, RTX and User programs. Search for: *Keil event recorder*
12. ***NEW!*** Power Measurement with ULINK*plus*: www.keil.com/ulinkplus
13. MDK includes board support for many Analog Devices processors and boards. www.keil.com/dd2/pack
14. Affordable perpetual and term licensing with support. Contact Keil sales for pricing options. Inside-Sales@arm.com
15. Keil Technical Support is included for one year and is renewable. This helps you get your project completed faster.
16. Micrium µC/Probe compatible. www.micrium.com/ucprobe
17. Serial Wire Viewer data trace: This processor does not support SWV. Other Analog Devices do. This provides the display of interrupts, data reads and writes and RTOS kernel awareness windows for Keil RTX and FreeRTOS.

ADuCM3029 EZ-KIT and Keil ULINK2

This document includes details on these features plus more:

1. Real-time Read and Write to memory locations for the Watch, Memory and Peripheral windows. These are non-intrusive to your program. No CPU cycles are stolen. No instrumentation code is added to your source files.
2. Two Hardware Breakpoints (can be set/unset on-the-fly).
3. RTX and RTX Threads window: a kernel awareness program for RTX that updates while your program is running.
4. A DSP example program using ARM CMSIS-DSP libraries. <coming>
5. How to create your own µVision projects and an extensive list of available document resources.

General Information:

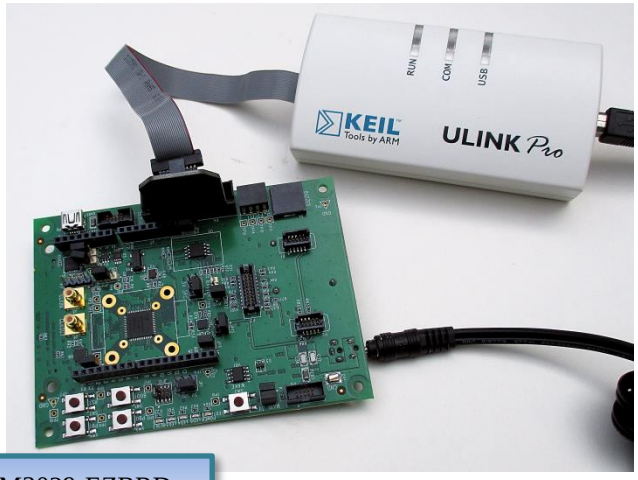
Keil Software and Software Packs:

Using Debug Adapters:

Beep_Example and Debugging Features:

Creating Your Own µVision Projects:

Other Useful Information:



Keil ULINK*pro* with ADuCM3029-EZBRD

1) Analog Devices Evaluation Boards & Keil Evaluation Software:

Keil MDK provides board support for many Analog Devices Cortex-M processors. Here is the current list available at www.keil.com/dd2/pack

On the second last page of this document is an extensive list of resources that will help you successfully create your projects. This list includes application notes, books and labs and tutorials for other boards.

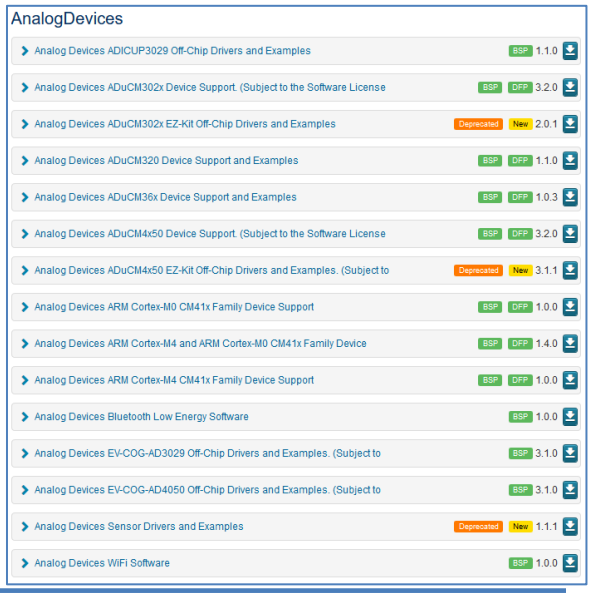
We recommend you obtain the latest Getting Started Guide for MDK5:

It is available here: www.keil.com/gsg/.

ARM forums: https://developer.arm.com

Keil Forums: www.keil.com/forum/

Analog Info: www.keil.com/analog-devices



2) MDK 5 Keil Software Information: *This document uses MDK 5.23 or later.*

MDK 5 Core is the heart of the MDK toolchain. This will be in the form of MDK Lite which is the evaluation version. The addition of a Keil license will turn it into one of the commercial versions available. Contact Keil Sales for more information at Inside-Sales@arm.com or your local distributor. For the latest MDK options see: www.keil.com/mdk5/selector/

Device and board support are distributed via Software Packs. These Packs are downloaded from the web with the "Pack Installer". The version(s) can be selected with "Select Software Packs" and your project configured with the "Manage Run Time Environment" (MRTE) utilities. These are components of µVision, the IDE of Keil MDK.

A Software Pack is an ordinary .zip file with the extension changed to .pack. It contains various header, Flash programming, example files and more. Contents of a Pack is described by a .pdsc file in XML format.

See www.keil.com/dd2/pack for the current list of available Software Packs. More packs are being added.

Example Project Files: This lab uses the examples provided in the Analog Devices::ADuCM302_DFP Software Pack.

3) Debug Adapters Supported:

These are listed below with a brief description.

1. **Segger J-Link:** Any J-Link including J-Link Lite is supported by µVision.

2. **Keil ULINK2 and ULINK-ME:** ULINK-ME is only offered as part of certain evaluation board packages. ULINK2 can be purchased separately. These are electrically the same and both support Serial Wire Viewer (SWV), Run-time memory reads and writes for the Watch, Memory, Peripheral, RTOS awareness windows and hardware breakpoint set/unset on-the-fly.

3. **Keil ULINK*pro*:** ULINK*pro* supports all SWV features and adds ETM Instruction Trace. ETM records all executed instructions. ETM provides Code Coverage, Execution Profiling and Performance Analysis features. ULINK*pro* also provides the fastest Flash programming times. Not all Cortex-M devices have ETM. Consult your specific datasheet.

4. ***NEW !*** **ULINK*plus*:** High SWV performance plus Power Measurement. See www.keil.com/ulink/ulinkplus/ for complete details. Coming Soon.

5. **CMSIS-DAP:** An extra processor on your board can be programmed to be an on-board debug adapter using the CMSIS-DAP ARM standard. CMSIS-DAP has an Apache 2.0 license.

   See https://github.com/ARM-software/CMSIS_5

ULINK*plus*

4) CoreSight Definitions: It is useful to have a basic understanding of these terms:

The Analog Devices ADuCM3024 does not have SWV, ITM or Watchpoints. It does have DAP reads and writes. It does not have a JTAG port. Use SWD (SW) instead. It works just as well..

Cortex-M0 and Cortex-M0+ may have only features 2) and 4) plus 11), 12) and 13) implemented. Cortex-M3, Cortex-M4 and Cortex-M7 can have all features listed implemented. MTB is normally found on Cortex-M0+. It is possible some processors have all features except ETM Instruction trace and the trace port. Consult your specific datasheet.

1. **JTAG:** Provides access to the CoreSight debugging module located on the Cortex processor. It uses 4 to 5 pins.
2. **SWD:** Serial Wire Debug is a two pin alternative to JTAG and has about the same capabilities except Boundary Scan is not possible. SWD is referenced as SW in the µVision Cortex-M Target Driver Setup.
   The SWJ box must be selected in ULINK2/ME or ULINK*pro*. Serial Wire Viewer (SWV) must use SWD because the JTAG signal TDO shares the same pin as SWO. The SWV data normally comes out the SWO pin or Trace Port.
3. JTAG and SWD are functionally equivalent. The signals and protocols are not directly compatible.
4. **DAP:** Debug Access Port. This is a component of the ARM CoreSight debugging module that is accessed via the JTAG or SWD port. One of the features of the DAP are the memory read and write accesses which provide on-the-fly memory accesses without the need for processor core intervention. µVision uses the DAP to update Memory, Watch, Peripheral and RTOS kernel awareness windows while the processor is running. You can also modify variable values on the fly. No CPU cycles are used, the program can be running and no code stubs are needed. You do not need to configure or activate DAP. µVision configures DAP when you select a function that uses it. Do not confuse this with CMSIS_DAP which is an ARM on-board debug adapter standard.
5. **SWV:** Serial Wire Viewer: A trace capability providing display of reads, writes, exceptions, PC Samples and printf.
6. **SWO:** Serial Wire Output: SWV frames usually come out this one pin output. It shares the JTAG signal TDO.
7. **Trace Port:** A 4 bit port that ULINK*pro* uses to collect ETM frames and optionally SWV (rather than SWO pin).
8. **ITM:** Instrumentation Trace Macrocell: As used by µVision, ITM is thirty-two 32 bit memory addresses (Port 0 through 31) that when written to, will be output on either the SWO or Trace Port. This is useful for printf type operations. µVision uses Port 0 for printf and Port 31 for the RTOS Event Viewer. The data can be saved to a file.
9. **ETM:** Embedded Trace Macrocell: Displays all the executed instructions. The ULINK*pro* provides ETM. ETM requires a special 20 pin CoreSight connector. ETM also provides Code Coverage and Performance Analysis. ETM is output on the Trace Port or accessible in the ETB (ETB has no Code Coverage or Performance Analysis).
10. **ETB:** Embedded Trace Buffer: A small amount of internal RAM used as an ETM trace buffer. This trace does not need a specialized debug adapter such as a ULINK*pro*. ETB runs as fast as the processor and is especially useful for very fast Cortex-A processors. Not all processors have ETB. See your specific datasheet.
11. **MTB:** Micro Trace Buffer. A portion of the device internal user RAM is used for an instruction trace buffer. Only on Cortex-M0+ processors. Cortex-M3/M4 and Cortex-M7 processors provide ETM trace instead.
12. **Hardware Breakpoints:** The Cortex-M0+ has 2 breakpoints. The Cortex-M3, M4 and M7 usually have 6. These can be set/unset on-the-fly without stopping the processor. They are no skid: they do not execute the instruction they are set on when a match occurs. The CPU is halted before the instruction is executed.
13. **Watchpoints:** Both the Cortex-M0, M0+, Cortex-M3, Cortex-M4 and Cortex-M7 can have 2 Watchpoints. These are conditional breakpoints. They stop the program when a specified value is read and/or written to a specified address or variable. There also referred to as Access Breaks in Keil documentation.

5) Keil MDK Software Download and Installation:

1. Download MDK 5.23 Core or later from the Keil website. www.keil.com/mdk5/install 

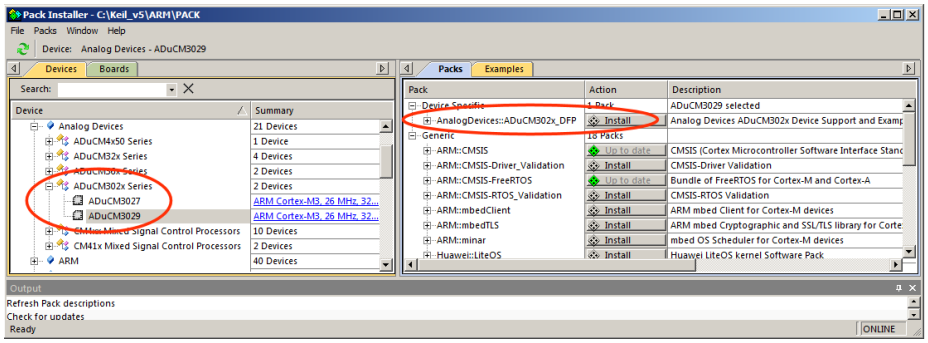
2. Install MDK into the default folder. You can install into any folder, but this lab uses the default C:\Keil_v5
3. We recommend you use the default folders for this tutorial. We will use C:\00MDK\ for the examples.
4. If you install MDK into a different folder, you will have to adjust for the folder location differences.
5. You can use the J-Link Lite adapter that comes with this board or any Keil ULINK or an external J-Link.
6. You do not need a Keil MDK license for this tutorial. All examples compile within the 32 K limit.

6) µVision Software Pack Download Install Process:

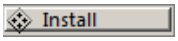
A Software Pack contain components such as header, Flash programming, documents and other files used in a project.

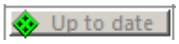
1) Start µVision and open Pack Installer:

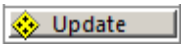
1. Connect your computer to the internet. This is needed to download the Software Packs. Start µVision: 
2. Open the Pack Installer by clicking on its icon:  A Pack Installer Welcome screen will open. Read and close it.
3. This window opens up: Select the Devices tab:
4. Note "ONLINE" is displayed at the bottom right. If "OFFLINE" is displayed, connect to the Internet now.
5. If there are no entries shown because you were not connected to the Internet when Pack Installer opened, select Packs/Check for Updates or  to refresh once you have connected to the Internet.



2) Install The Analog Software Pack:

1. In the Devices tab, select Analog and then ADuCM3024x Series as shown above: The actual devices supported are displayed. You can select either the header or an actual device since in this case as the Pack supports them all.
2. The appropriate Pack will be displayed under the Packs tab.
3. Select Install beside AnalogDevices::ADuCM3024x _DFP. 

This Pack will download and install into the MDK folders. This download can take several minutes.
4. Its status is indicated by the "Up to date" icon: 

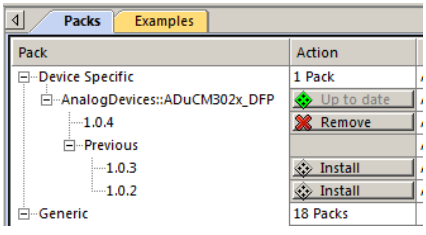
5. Update means there is an updated Software Pack available for download. 

6. Leave Pack Installer open to install the examples as described on the next page.

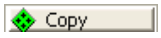
TIP: The left hand pane filters the selections displayed on the right pane. You can start with either Devices or Boards.

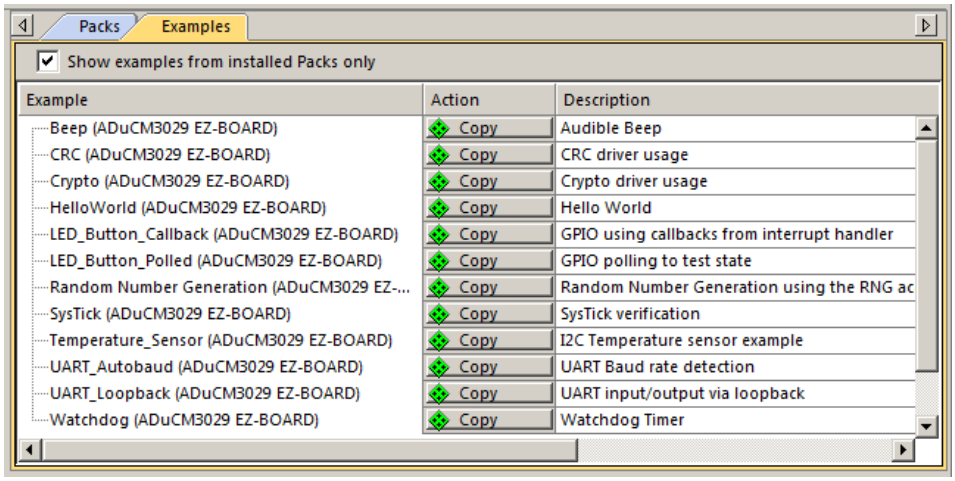
TIP: If you expand AnalogDevices::ADuCM3024x _DFP you can see any previous versions and could download any of them: You can then select which one to be used in the Select Software Packs utility. 
This utility is described in two pages.

This is useful for freezing or selecting versions of software during product development.

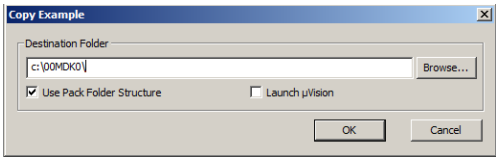


3) Install the Examples:

1. Select the Examples tab.
2. Note the examples for this board are filtered by the selection in the Devices tab.
3. You can also select examples by selecting the board you are using in the Boards tab if it is supported.
4. Opposite Beep (ADCuCM3029 EZ-BOARD) select Copy: 



5. The Copy Example window opens up as shown below: Select Use Pack Folder Structure. Unselect Launch µVision.
6. Type in C:\00MDK\ as shown: Click OK to copy the Beep project.
7. The Beep example will now copy to C:\00MDK\Boards\ADuCM3029-EZ-Board\Keil
8. Repeat these steps to copy the rest of the examples. (optional)
9. Close the Pack Installer. You can open it any time by clicking on its icon.
10. If you are asked to Reload the Packs click YES.



TIP: The default folder for copied examples the first time you install MDK is C:\Users\<user>\Documents. For simplicity, we will use the default folder of C:\00MDK\ in this tutorial. You can use any folder you prefer.

Read-Only Source Files:

Some source files in the Project window will have a yellow key on them:  This means they are read-only. This is to help unintentional changes to these files. This can cause difficult to solve problems. These files normally need no modification. µVision icon meanings are found here: www.keil.com/support/man/docs/uv4/uv4_ca_filegrp_att.htm

If you need to modify one, you can use Windows Explorer to modify its permission.

1. In the Projects window, double click on the file to open it in the Sources window.
2. Right click on its source tab and select Open Containing folder.
3. Explorer will open with the file selected.
4. Right click on the file and select Properties.
5. Unselect Read-only and click OK. You are now able to change the file in the µVision editor.
6. It is a good idea to make the file read-only when you are finished modifications.

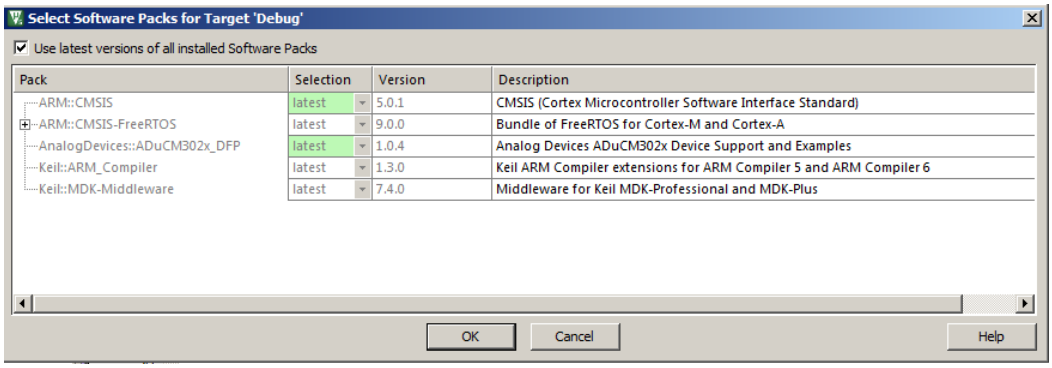
TIP: µVision icon meanings are found here: www.keil.com/support/man/docs/uv4/uv4_ca_filegrp_att.htm

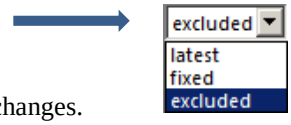
7) Other features of Software Packs:

A) Select Software Pack Version:

This µVision utility provides the ability to choose among the various software pack versions installed in your computer.

1. Open the Select Software Pack by clicking on its icon: 
2. This window opens up. Note **Use latest versions …** is selected. The latest version of the Pack will be used.
3. Unselect this setting and the window changes to allow various versions of Packs to be selected.

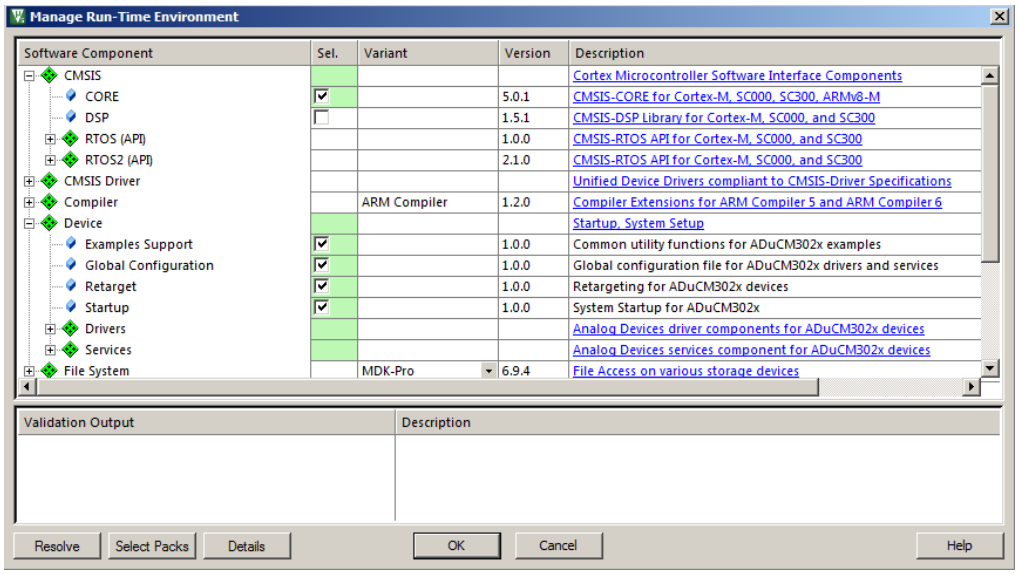


4. Note various options are visible as shown here: 

5. Select excluded and see the options as shown:
6. Select Use latest versions… Do not make any changes.
7. Click Cancel to close this window to make sure no changes are made.

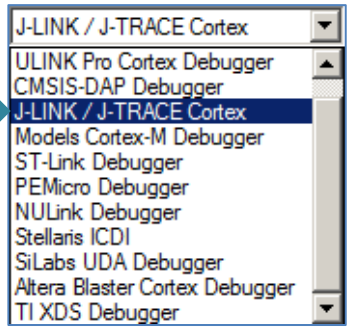
B) Manage Run-Time Environment (MRTE):

1. Select Project/Open Project.
2. Open the project: C:\00MDK\Boards\ADuCM3029-EZ-Board\Keil\beep\beep_example.uvprojx.
3. Click on the Manage Run-Time Environment (MRTE) icon:  The window below opens:
4. Expand various headers and note the selections you can make. A selection made here will automatically insert the appropriate source files into your project.
   Files are selected when there is a green block. A red block means some files are missing. An orange block means other installed components need to be selected. Click the Resolve icon to ask the MRTE to attempt to automatically select them.
5. Do not make any changes. Click Cancel to close this window.



8) Testing the Debug Adapter Connection and Configuration:

1. Connect a J-Link (Lite) or and Keil ULINK to the DEBUG connector P4.
2. Power the board to P8 with the supplied AC adapter.
3. µVision must be running with any example loaded.
4. Select Project/Open Project.
5. Open C:\00MDK\Boards\ADuCM3029-EZ-Board\Keil\beep\beep_example.uvprojx. Or any other project.
6. Select Debug in the Beep project. Debug Other projects might not have this exact selection.
7. Select Options for Target or ALT-F7. Click on the Debug tab.
8. Select your debug adapter. Valid options are ULINK2/ME, ULINK Pro Cortex Debugger, CMSIS-DAP or J-Link / JTRACE as shown: J-Link is chosen by default in the Beep example.
9. Click on Settings: and the window below opens up: An IDCODE and Device name will be displayed indicating a proper connection to the processor.
10. If nothing or an error is displayed in this SW Device box, this ***must*** be corrected before you can continue.

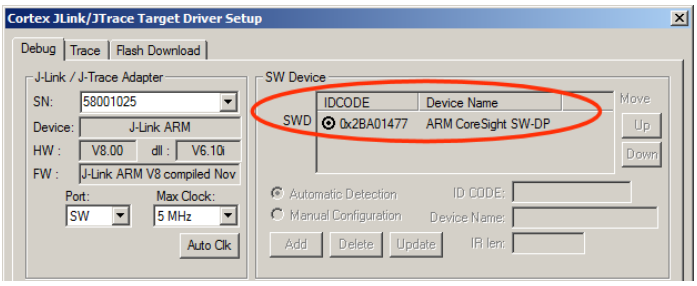


.

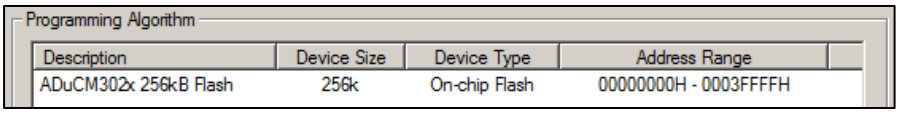
Verify the Flash Program Algorithm:

This is preset by the Software Pack when you select the processor. This step is optional.

1. Select the Flash Download tab.
2. The window that opens will display the correct algorithm. This is selected automatically according to the processor selected in the Device tab.



3. Below is the correct algorithm for the ADuCM302x processor:
4. Click OK twice to return to the main µVision window.
5. Select File/Save All or .

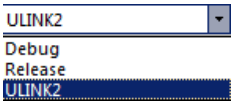


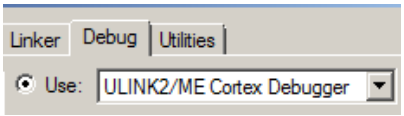
Your Debug Adapter is now confirmed to be ready to use.

Creating a New Target Options Pull-down Menu: (for reference only)

The Beep project contains two target options: Debug and Release. You can quickly create your own target options. A set of all the configurations set under the Options For Target windows will be created , saved and available for selection.

1. Select an existing selection to be used as a template. In this case, select Debug or whatever is available in your case.
2. Select Project/Manage/Project Items… or select:
3. In the Project Targets area, select NEW or press your keyboard INSERT key.
4. Enter **ULINK2** and press Enter. Click OK to close this window.
5. In the Target Selector menu, select the ULINK2 selection you just made:
6. Select Options for Target or ALT-F7. Click on the Debug tab.
7. Select ULINK2/ME Cortex Debugger… as shown here:
8. Click OK twice to return to the main µVision menu.
9. Select File/Save All or .





9) *Beep_example* program:

Now we will connect a Keil MDK development system using the ADuCM3029-EZBRD board. This page will use the J-Link Lite debug adapter but you can select any that you have programmed and connected.

1. Connect a J-Link or Keil ULINK to the DEBUG connector P4.
2. Connect another USB cable to USB to UART P6. This step is optional as it only provides a status message.
3. If you want the messages: Configure a standard terminal program (such as PuTTY) and set the speed to 57600 baud.
4. Power the board to P8 with the AC adapter. The green POWER led will illuminate.
5. Start µVision by clicking on its desktop icon.
6. Select Project/Open Project.
7. Open C:\00MDK\Boards\ADuCM3029-EZ-Board\Keil\beep\beep_example.uvprojx.
8. Compile the source files by clicking on the Rebuild icon.
9. Enter Debug mode by clicking on the Debug icon. The Flash memory will be programmed. Progress will be indicated in the Output Window. Select OK if the Evaluation Mode box appears.
10. Click on the RUN icon.

The Buzzer D1 will output four musical notes. Repeat by clicking RESET:

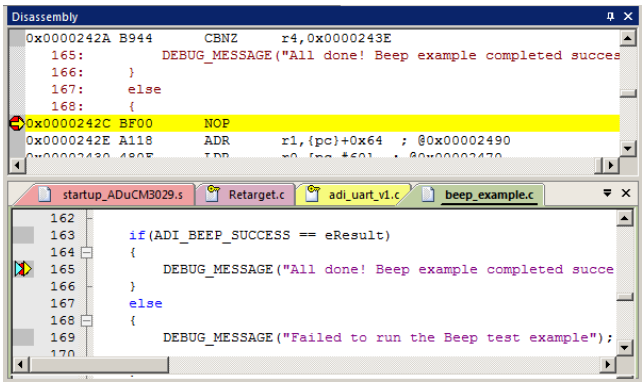
Now you know how to compile a program, program it into the processor Flash, run it and stop it !

Note: The board will start Blinky stand-alone. Blinky is now permanently programmed in the Flash until reprogrammed.

10) Hardware Breakpoints:

The ADuCM3029 has two hardware breakpoints that can be set or unset on the fly while the program is running.

1. In the Beep_example.c file, locate the "DEBUG_MESSAGE All Done!..." line near or on line 165 as shown below:
2. Note there is a grey block to the left of this line. These gray blocks means there are assembly instructions present at these points. You can set a hardware breakpoint only on any of these grey blocks.
3. Click on RESET to restart the program. While the tones are still sounding, click on the grey block to the left of line 165. You can also click in the Disassembly window to set a breakpoint. Be quick before the program ends !
4. A red circle will appear and the program will halt if you set the breakpoint before the tones ended.
5. Note the breakpoint is displayed in both the Disassembly and source windows as shown here:
6. To repeat, click on RESET and then RUN.
7. The yellow arrow is the current program counter value. The instruction at this point has not run yet.
8. Clicking in the source window will indicate the appropriate code line in the Disassembly window and vice versa. This is relationship indicated by the cyan arrow and the yellow highlight:
9. Leave this breakpoint set for the Call Stack exercise on the next page.



TIP: If you set too many breakpoints, µVision will warn you. This processor has two hardware breakpoints available.

TIP: ARM hardware breakpoints do **not** execute the instruction they are set to and land on. ARM CoreSight hardware breakpoints are no-skid. This is a rather important feature for effective debugging.

ARM CoreSight hardware breakpints do not substitute or change any of your code in Flash memory for maximum reliability.

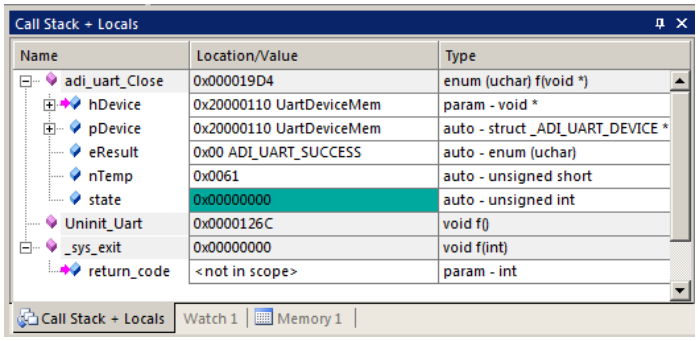
14) Call Stack + Locals Window:

Local Variables:

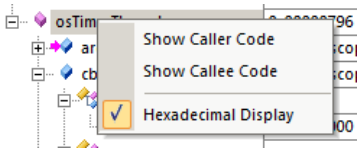
The Call Stack and Locals windows are incorporated into one integrated window. Whenever the program is stopped, the Call Stack + Locals window will display call stack contents as well as any local variables located in the active function or thread.

If possible, the values of the local variables will be displayed and if not the message <not in scope> will be displayed. The Call + Stack window presence or visibility can be toggled by selecting View/Call Stack Window in the main µVision window when in Debug mode.

1. Use the same exercise and setup from the previous page. Leave the breakpoint activated.
2. Click on RESET and then RUN. The program Beep will stop on line 64 as before.
3. Click on the Call Stack + Locals tab if necessary to open it. Expand some of the entries.
4. With Beep_example.c in focus (the filename is underlined in its tab), click on the Step In icon .
5. As you click on Step In, you can see the program entering and perhaps leaving various functions. Note the local variables are displayed if possible.
6. Shown is an example Call Stack + Locals window:
7. The functions as they were called are displayed. If these functions had local variables, they would be displayed.
8. Click Step Out to immediately exit a function.



9. Right click on a function and select either Callee or Caller code and this will be highlighted in the source and disassembly windows.



10. When you ready to continue, remove the hardware breakpoint by clicking on its red circle ! You can also type Ctrl-B, select Kill All and then Close.

TIP: You can modify a variable value in the Call Stack & Locals window when the program is stopped.

TIP: This window is only valid when the processor is halted. It does not update while the program is running because locals are normally kept in a CPU register. These cannot be read by the debugger while the program is running. Any local variable values are visible only when they are in scope.

Do not forget to remove any hardware breakpoints before continuing.

Single-Stepping:

1. **By C Source Line:** With Beep_Example.c in focus (its tab is underlined), click on the Step In icon or F11 a few times: You will see the program counter changes one C line at a time. The yellow arrow indicates the next C line to be executed.
2. **By Assembly Instruction:** Click on the top margin of the Disassembly window to bring it into focus. Clicking Step Into now advances the program counter one assembly instruction at a time.

15) Watch and Memory Windows and how to use them:

The Watch and Memory windows display variable values updated in real-time. It does this using the ARM CoreSight debugging technology that is part of Cortex-M processors. It is also possible to "put" or insert variable values into the Memory window in real-time. This is possible with a Watch if the variable is changing slowly or the program is stopped. It is possible to enter variable names into windows manually.

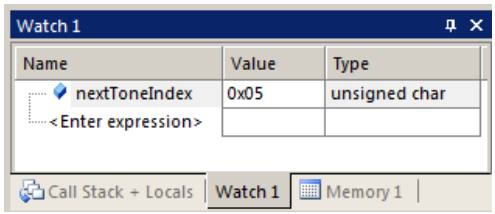
You can also right click on a variable and select Add *varname* to.. and select the appropriate window. The System Viewer windows work using the same CoreSight technology known as "DAP reads & writes". Call Stack, Watch and Memory windows can't see local variables unless stopped in their function.

Watch window:

A global variable: The static variable nextToneIndex is declared in beep_example.c near line 70.

```
69  /* tone index */
70  static  uint8_t nextToneIndex = 0;   /
```

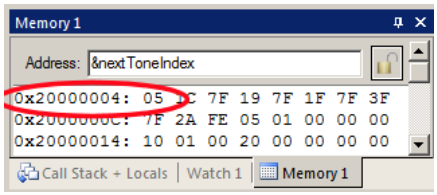
1. Leave the beep_example running from the previous page.
2. You can configure a Watch or Memory window while the program is running.
3. In Blinky.c, right click on nextToneIndex and select Add nextToneIndex to … and select Watch 1. Watch 1 will automatically open. delay_val will be displayed as shown here: It might say Can't Evaluate.
4. Click on RESET and then RUN.
5. As the program runs, this variable will update:
6. No CPU cycles are stolen and there is no code stubs in your sources. This non-intrusive.



TIP: A Watch or Memory window can display and update global and static variables, structures and peripheral addresses while the program is running. These are unable to display local variables because these are typically stored in a CPU register. These cannot be read by µVision in real-time. To view a local variable in these windows, convert it to a static or global variable.

Memory window:

1. Right click on nextToneIndex and select Add nextToneIndex to … and select Memory 1.
2. Note the value of nextToneIndex is displaying its address in Memory 1 as if it is a pointer. This is useful to see what address a pointer is pointing to but this not what we want to see at this time.
3. Add an ampersand "&" in front of the variable name and press Enter. The physical address here is 0x2000_0004.
4. The data contents of delay_val is displayed as shown here:
5. Right click in the Memory window and select Unsigned/Int.
6. Both the Watch and Memory windows are updated in real-time.
7. Right-click with the mouse cursor over the desired data field and select Modify Memory. You can change a memory or variable on-the-fly while the program is still running. You will not see any change when this program is running as this variable is constantly updated periodically.



TIP: No CPU cycles are used to perform these operations.

TIP: To view variables and their location use the Symbol window. Select View/Symbol Window while in Debug mode.

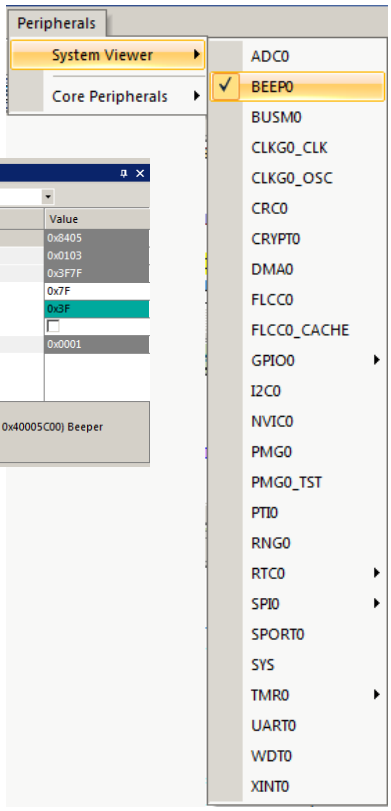
TIP: To Drag 'n Drop into a tab that is not active, pick up the variable and hold it over the tab you want to open; when it opens, move your mouse into the window and release the variable.

13) Peripherals System Viewer (SV):

The System Viewer provides the ability to view certain registers in the CPU core and in peripherals. In most cases, these Views are updated periodically in real-time while your program is running. These Views are available only while in Debug mode. There are two ways to access these views: **a)** View/System Viewer and **b)** View/Peripherals/System Viewer.

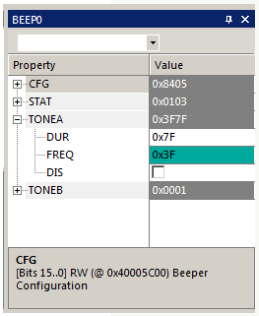
The CPU core register are accessible by selecting Peripherals/Core Peripherals as shown belwo:

1. Click on RUN. You can open SV windows when your program is running.



Select BEEP0:

2. Select Peripherals/System Viewer and then BEEP0 as shown here.
3. This window opens up. Expand TONEA:



4. Click on RESET  and then RUN. 
5. The values in BEEP0 will update in real-time.
6. You can change the values in the System Viewer on-the-fly. In this case, the values are updated quickly so it is hard to see the change.

TIP: If you click on a register in the properties column, a description about this register will appear at the bottom of the window. In this case, CFG is selected and displayed.

TIP: It is true: you can modify values in the SV while the program is running or stopped. This is very useful for making slight timing value changes instead of the usual modify, compile, program, run cycle.

You must make sure a given peripheral register allows and will properly react to such a change. Changing such values indiscriminately is a good way to cause serious and difficult to find problems.

This is the end of the Beep exercises.

You can open some of the other projects and see how they work. Each contains an instruction file under the Documentation header in the Project window.

The section describes how to create your own µVision projects from the beginning.

14) Creating your own MDK 5 project from scratch: no RTOS:

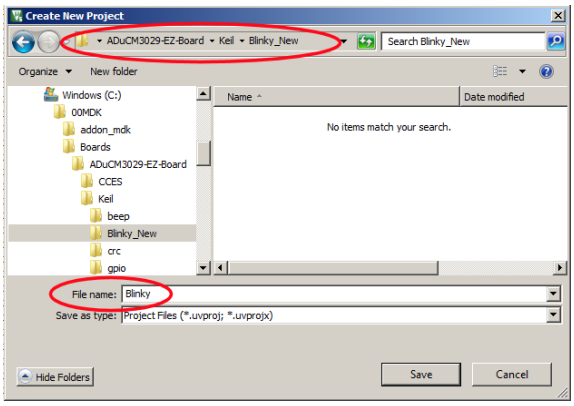
All examples provided by Keil are pre-configured. All you have to do is compile them. You can use them as a starting point for your own projects. However, we will start this example project from the beginning to illustrate how easy this process is. Once you have the new project configured; you can build, load and run a bare metal (no OS) Blinky example. It will have a simple incrementing counter to monitor. However, the processor startup sequences are present and you can easily add your own source code and/or files. You can use this process to create any new project, including one using an RTOS.

Install the Software Pack for your processor:

1. Start µVision and leave it in Edit mode. Do not enter Debug mode. A project must be loaded. Any project at all.
2. **Pack Installer:** The AnalogDevices::ADuCM3024x _DFP Pack must be installed. This was done on page 4.

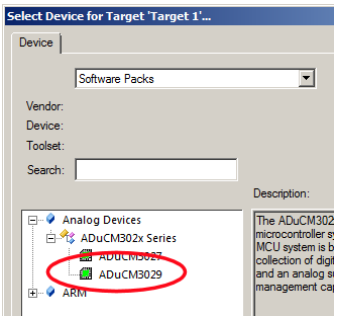
Create a new Directory and a New Project:

3. In the main µVision menu, select Project/New µVision Project… Create New Project window opens:
4. In this window, shown here, navigate to the folder C:\00MDK\Boards\ADuCM3029-EZ-Board\Keil \
5. Right click in this window and select New (or click New Folder) and create a new folder. I called it BlinkyNEW.
6. Double click on BlinkyNew to open it or highlight it and select Open.
7. In the File name: box, enter Blinky. Click on Save.
8. This creates the project Blinky.uvproj. (is MDK 4 format)
9. The Select Device for Target…opens:



Select the Device you are using:

1. Select Analog Devices and expand it. Select ADuCM3029 as shown here:



TIP: Make sure you select the deepest layer device or this will not work correctly.

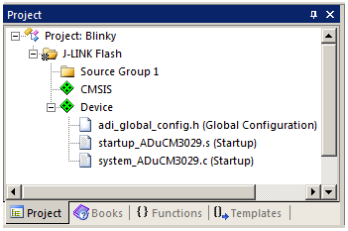
2. Click OK and the Manage Run Time Environment window shown below right opens.

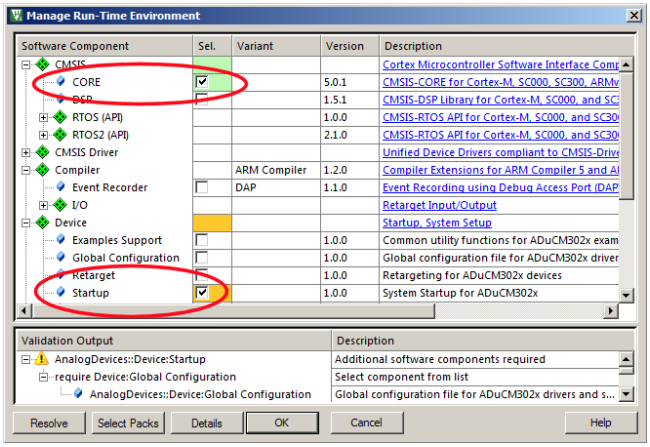
Select the CMSIS components you want:

1. Expand CMSIS and Device. Select CORE and Startup as shown below. There will be some yellow blocks indicating more files need to be added.
2. Click the Resolve button and these will be added by uVision.
3. All blocks will now be highlighted in Green indicating there are no other files needed. Click OK to close.
4. Click on File/Save All or select the Save All icon:
5. The project Blinky.uvproj will now be changed to Blinky.uvprojx. (MDK 4 ➡ MDK 5 format)
6. You now have a new project list as shown on the bottom left below: The CMSIS files you selected have been automatically entered and configured into your project for your selected processor.
7. Note the Target Selector says Target 1. Highlight Target 1 in the Project window shown below left.
8. Click once on it and change its name to **J-Link Flash** and press Enter. The Select Target name box will also change. You can choose any name.

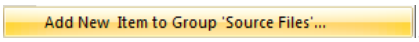
What has happened to this point:

You have created a blank µVision project using MDK 5 Software Packs. All you need to do now is add your own source files and select your debug adapter. The Software Pack has pre-configured many settings for your convenience.

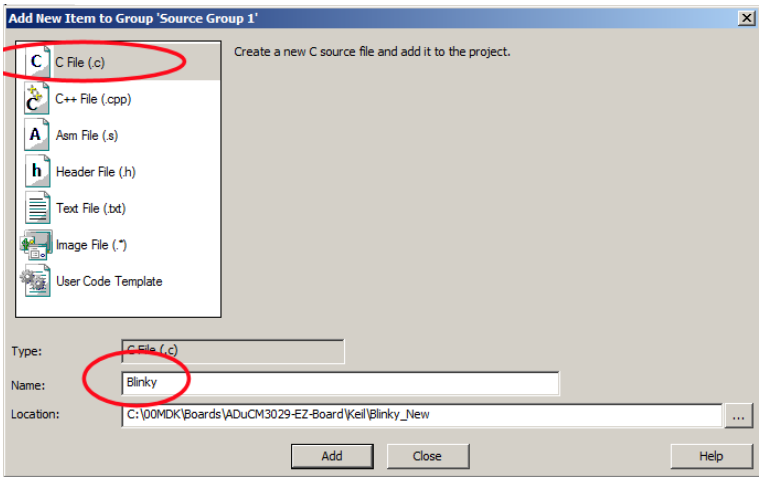




Create a blank C Source File:

1. Right click on Source Group 1 in the Project window and select 

.
2. This window opens up:
3. Highlight the upper left icon: C file (.c):
4. In the Name: field, enter Blinky.
5. Click Add to close this window.
6. Click on File/Save All or 
7. Expand Source Group 1 in the Project window and Blinky.c will now display. It is a blank file.



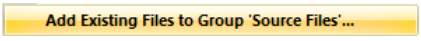
Add Some C Code to Blinky.c:

1. Right click in Blinky.c and select Insert '#include file'.
2. Select ADuCM3029.h. This will be added to Blinky.c at the top of the file.
3. In the nearly blank Blinky.c, add the C code below:
4. Click on File/Save All or 

```
#include "ADuCM3029.h"        //already added
unsigned int counter = 0;

/*-------------------------------------------
  MAIN function
 *-------------------------------------------*/
int main (void) {
    SystemInit();
    SystemCoreClockUpdate();
  while(1) {
    counter++;
if (counter > 0x0F) counter = 0;
}
}        //make sure you add a CR Carriage Return or Enter after the last parentheses.
```

5. Build the files.  There will be no errors or warnings if all was entered correctly.

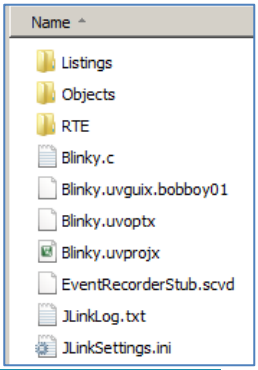
TIP: You could also add existing source files: 

But not at this time.

The Next Step ? First we will do a summary of what we have done so far and then …. you will run the program !

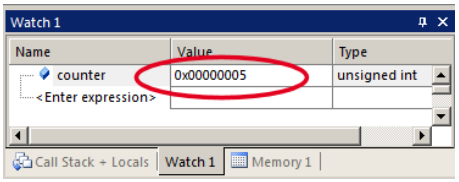
What we have so far ?

1. A project has been created in C:\00MDK\Boards\ADuCM3029-EZ-Board\Keil\Blinky_New
2. The folders have been created as shown here: 
3. RTE contains the CMSIS-Core startup and system files.
4. The Software Pack has pre-configured many items in this new project for your convenience.



Running Your Program:

1. Enter Debug mode by clicking on the Debug icon . The Flash will be programmed.
2. Click on the RUN icon. Note: you stop the program with the STOP icon.
3. Right click on counter in Blinky.c and select Add 'counter' to … and select Watch 1.
4. counter should be updating as shown here:



5. You can also set a breakpoint in Blinky.c and the program should stop at this point if it is running properly. If you do this, remove the breakpoint.
6. You are now able to add your own source code to create a meaningful project. You can select software components in the Manage Run-time Environment window. You can experiment with this later.

TIP: Watch 1 is updated periodically, not when a variable value changes. Since Blinky is running very fast without any time delays inserted, the values in Watch 1 will appear to jump and skip some sequential values that you know must exist.

Measuring the CPU Clock:

The file system_ADuCM3029.h" contains the CPU clock setup code. This project is running at the default of 26 MHz.

1. STOP the program. Exit Debug mode .
2. In Blinky.c, near line 22, just after int main(void) {, add this line:

```
22   SystemCoreClockUpdate();
```

3. Click on File/Save All or
4. Build the files. There will be no errors or warnings.
5. Enter Debug mode. The Flash will be programmed.
6. Click on the RUN icon.
7. In Watch 1, double click on <Enter expression> and enter SystemCoreClock and press Enter. You can also right-click on SystemCoreClock as found in system_ADuCM3029.h"and enter it in the usual way.
8. Right click on SystemCoreClock in Watch1 and unselect Hexadecimal Display.
9. The CPU speed is displayed as 26 MHz as shown in the global variable SystemCoreClock in Watch 1.
10. Stop the CPU. and exit Debug mode.

What else can we do ?

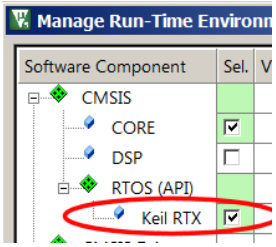
5. You can create new source files using the Add New Item window. See the top of the previous page.
6. You can add existing source files by right clicking on a Group name and selecting Add Existing Files.
7. You can easily add Analog Devices example files to your project. Select them with the MRTE utility.
8. If you use RTX or Keil Middleware, source and template files are provided in the Add New window.
9. Now, we will add RTX to your new project !

15) Adding RTX to your MDK 5 project:

Software Packs contain all the code needed to add RTX to your project. RTX is CMSIS-RTOS compliant.

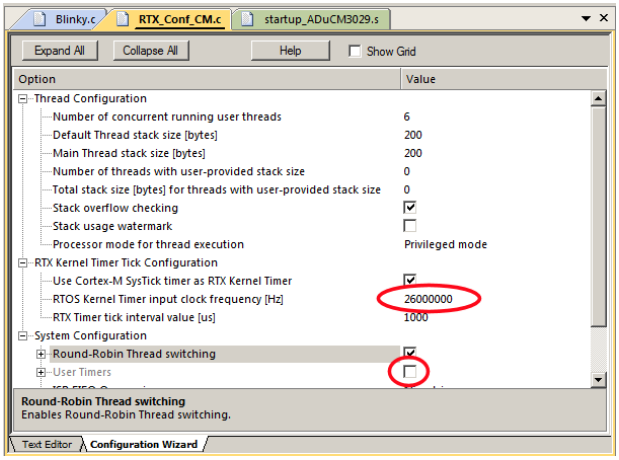
Configuring RTX is easy in MDK 5. These steps use the preceding Blinky example you constructed.

1. Using the same example from the preceding pages, Stop the program and Exit Debug mode.
2. Open the Manage Run-Time Environment window:
3. Expand all the elements as shown here:
4. Select Keil RTX as shown and click OK.
5. Appropriate RTX files will be added to your project. See the Project window.
6. In Blinky.c, near the first line, right click and select Insert '# include file'. Select "cmsis_os.h". This RTX header will be added to Blinky.c.



Configure RTX:

1. In the Project window, expand the CMSIS group.
2. Double click on RTX_Conf_CM.c to open it.
3. Select the Configuration Wizard tab at the bottom of this window: Select Expand All.
4. The window is displayed here:
5. Select Use Cortex-M SysTick Timer as RTX Kernel Timer.
6. Set Timer clock value: to 26000000 as shown: (26 MHz)
7. Unselect User Timers: (or increase stack sizes to 300 bytes)
8. Use the defaults for the other settings.

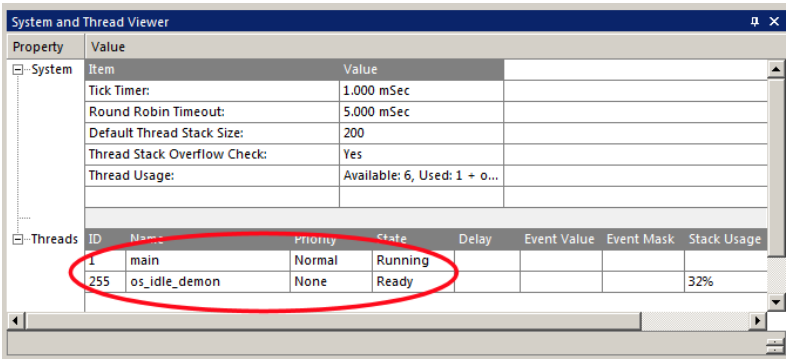


Build and Run Your RTX Program:

1. Click on File/Save All or
2. Build the files. There will be no errors or warnings.
3. Enter Debug mode: Click on the RUN icon.
4. Select Debug/OS Support/System and Thread Viewer. The window below opens up.
5. You can see three threads: the main thread is the only one running. As you add more threads to create a real RTX program, these will automatically be added to this window.
6. Stop the program and Exit Debug mode.

What you have to do now:

1. You must add the RTX framework into your code and create your threads to make this into a real RTX project.
2. **Getting Started MDK 5:** Obtain this useful book here: www.keil.com/gsg/. It has information on implementing RTX as well as other subjects.



TIP: The Configuration Wizard is a scripting language as shown in the Text Editor as comments starting such as a </h> or <i>. See www.keil.com/support/docs/2735.htm for instructions on how to add this feature to your own source code.

16) Adding an RTX Thread:

We will create and activate a thread. We will add an additional variable counter2 that will be incremented by this new thread.

1. In Blinky.c, add this line near line 5: unsigned int counter2 = 0;

```
5   unsigned int counter2 = 0;
```

Create the Thread job1:

2. Add this code before main():
   This will be the new thread named job1. osDelay(200) delays the program by 500 clock ticks to slow it down so we can easier see the values of counter and counter2 increment by 1.

```
7    void job1 (void  const *argument) {
8    for (;;) {
9            counter2++;
10             if (counter2 > 0x0F) counter2 = 0;
11            osDelay(200);
12    }
13  }
```

Add another osDelay to main(): (main() is a thread !)

3. Add this line just after the if statement near line 21:

```
21   osDelay(200);
```

Define and Create the Thread:

1. Add this line near line 15 just before main():

```
15   osThreadDef(job1, osPriorityNormal, 1, 0);
```

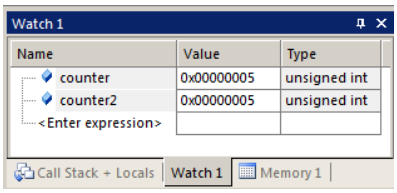
2. Create the thread job1 near line 18 just after main() and before the while(1) loop:

```
18   osThreadCreate(osThread(job1), NULL);
```

3. Click on File/Save All or 

Build and Run the Program:

1. Build the files.  There will be no errors or warnings. If there are, please fix them before continuing.
2. Enter Debug mode:  Click on the RUN icon. 
3. Right click on counter2 in Blinky.c and select Add counter2 to … and select Watch 1.
4. Both counter and counter2 will increment but slower than before:
   The two osDelay(200) function calls each slow the program down by 200 msec. This makes it easier to watch these two global variables increment.



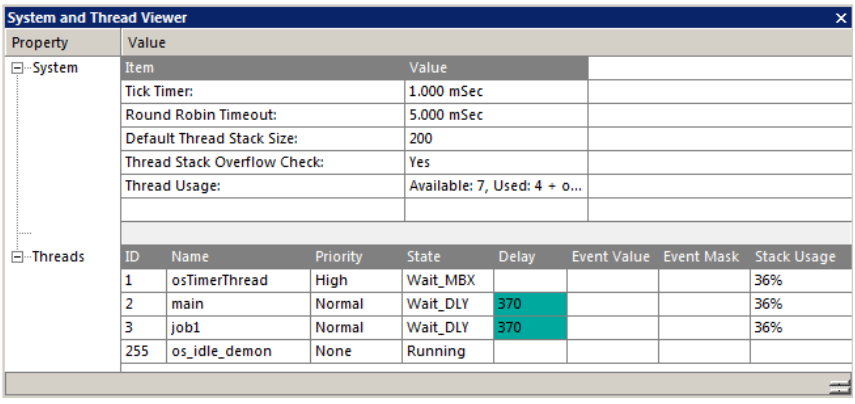
TIP: osDelay() is a function provided by RTX and is triggered by the SysTick timer.

5. Open the System and Thread Viewer by selecting Debug/OS Support.
6. Note that job1 has now been added as a thread as shown below:
7. Note os_idle_demon is always labelled as Running. This is because the program spends most of its time here.
8. Set a breakpoint in job1 and the program will stop there and job1 is displayed as "Running" in the Viewer.
9. Set another breakpoint in the while(1) loop in main() and each time you click RUN, the program will change threads.
10. There are many attributes of RTX you can add. RTX help files are located here depending on your CMSIS version: C:/Keil_v5/ARM/Pack/ARM/CMSIS/*x.x.x*/CMSIS/Documentation/RTX/html/index.html.
11. **Remove any breakpoints you have created.**

This is the end of the exercises.

RTX is part of the new CMSIS 5 See https://github.com/ARM-software/CMSIS_5.

There is a new RTX: RTX5. It is multi-core and equipped for ARMv8-M TrustZone. These include the new Cortex-M23 and Cortex-M33.



TIP: ***For good programming practice:***

1. In the beginning of main() before any statements concerning RTX, add this line: osKernelInitialize ();
2. Just after all RTX statements in main(), add this line to start the kernel: osKernelStart ();

17) Document Resources: www.keil.com/analog-devices

Books:

1. ***NEW!*** **Getting Started with MDK 5:** Obtain this free book here: www.keil.com/mdk5/
2. There is a good selection of books available on Keil: www.arm.com/support/resources/arm-books/index.php
3. and on Arm.com: www.arm.com/resources/education/textbooks
4. µVision contains a window titled Books. Many documents including data sheets are located there.
5. **The Definitive Guide to the ARM Cortex-M0/M0+** by Joseph Yiu. Search the web for retailers.
6. **The Definitive Guide to the ARM Cortex-M3/M4** by Joseph Yiu. Search the web for retailers.
7. **Embedded Systems: Introduction to Arm Cortex-M Microcontrollers** (3 volumes) by Jonathan Valvano
8. MOOC: Massive Open Online Class: University of Texas: http://users.ece.utexas.edu/~valvano/

Application Notes: www.keil.com/appnotes

1. ***NEW!*** ARM Compiler Qualification Kit: Compiler Safety Certification: www.keil.com/safety
2. Using Cortex-M3 and Cortex-M4 Fault Exceptions www.keil.com/appnotes/files/apnt209.pdf
3. CAN Primer: www.keil.com/appnotes/files/apnt_247.pdf
4. Segger emWin GUIBuilder with µVision™ www.keil.com/appnotes/files/apnt_234.pdf
5. Porting mbed Project to Keil MDK™ 4 www.keil.com/appnotes/docs/apnt_207.asp
6. MDK-ARM™ Compiler Optimizations www.keil.com/appnotes/docs/apnt_202.asp
7. GNU tools (GCC) for use with µVision https://launchpad.net/gcc-arm-embedded
8. RTX CMSIS-RTOS Download https://github.com/ARM-software/CMSIS_5
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 and search for DAI0298A
11. Cortex Debug Connectors: www.keil.com/coresight/coresight-connectors
12. Sending ITM printf to external Windows applications: www.keil.com/appnotes/docs/apnt_240.asp
13. ***NEW!*** Migrating from Cortex-M3/M4 to Cortex-M7: www.keil.com/appnotes/docs/apnt_270.asp
14. ***NEW!*** ARMv8-M Architecture Technical Overview: https://community.arm.com/docs/DOC-10896
15. ***NEW!*** Using ARM TrustZone® on ARMv8-M: www.keil.com/appnotes/docs/apnt_291.asp
16. ***NEW!*** Determining Cortex-M CPU Speed using SWV: www.keil.com/appnotes/docs/apnt_297.asp

Useful ARM Websites:

1. ***NEW!*** CMSIS 5 Standards: https://github.com/ARM-software/CMSIS_5 and www.keil.com/cmsis/
2. ARM and Keil Community Forums: www.keil.com/forum and http://community.arm.com/groups/tools/content
3. Arm Developer Website: https://developer.arm.com/
4. ARM University Program**:** www.arm.com/university. Email: university@arm.com
5. mbed™: http://mbed.org

18) Keil Products and contact information: www.keil.com/analog-devices

Keil Microcontroller Development Kit (MDK-ARM™) for NXP processors:

- **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.

For the latest details see: www.keil.com/mdk5/selector/

Keil Middleware includes Network, USB, Graphics and File System. www.keil.com/mdk5/middleware/

USB-JTAG/SWD Debug Adapter (for Flash programming too)

- ULINK2 - *(ULINK2 and ME - SWV only – no ETM)*
- ULINK-ME – sold only with a board by Keil or OEM: is equivalent to a ULINK2.
- *New* ULINK*plus*- Cortex-M*x* High performance SWV & power measurement.
- ULINK*pro* - Cortex-M*x* SWV & ETM instruction trace. Code Coverage and Performance Analysis.
- ULINK*pro* D - Cortex-M*x* SWV no ETM trace ULINK*pro* also works with ARM DS-5.

Call Keil Sales for more details on current pricing. All products are available.

For the ARM University program: go to www.arm.com/university Email: university@arm.com

All software products include Technical Support and Updates for 1 year. This can easily be renewed.

Keil RTX™ Real Time Operating System

- RTX 5 is provided free as part of Keil MDK.
- No royalties and is very easy to use. It has an Apache 2.0 license.
- RTX source code is included with all versions of MDK.
- Kernel Awareness visibility windows are integral to µVision.

For the entire Keil catalog see www.keil.com or contact Keil or your local distributor.

For Analog Devices support: www.keil.com/analog-devices.

For more information:

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

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

Global Inside Sales Contact Point: Inside-Sales@arm.com ARM **Keil World Distributors:** www.keil.com/distis

For more information: www.arm.com/cmsis, www.keil.com/forum and http://community.arm.com/groups/tools/content

CMSIS COMPLIANT
ARM® Cortex™ Microcontroller Software Interface Standard

Cortex™
Intelligent Processors by ARM®

arm KEIL