



4 Steps Demonstration

RealView Microcontroller Development Kit

MCBSTM32 Evaluation Board

Overview

Aim

To demonstrate the unique features of RealView Microcontroller Development Kit (RealView MDK).

- Ease of Use
- Benefits of Complete Device Simulation
 - Core, peripherals, registers and memory
- Logic Analyzer
- Code Analysis
 - Coverage, profiling and trace analysis


Requirements

- RealView MDK v3.15 or higher
- MCBSTM32 Evaluation Board.
- ULINK2

Note: MCBSTM32 and ULINK2 are only required for the final part of the demo when the program is downloaded to the target device Flash. The demo can be run entirely in simulation if required and omit step 4.


Note: This demo works with the evaluation version of MDK. When running the evaluation version it displays a warning message that the maximum code size of the project is limited to 16KByte.

Conventions

	Box shows actions for demonstrator or user of demo. Icon represents the button controls to be used.
---	--

Set-up

Before running the demo for the first time.

	Make a copy of the folder ../Keil/ARM/Boards/Keil/MCBSTM32/STLIB_Blinky Into a new demo folder ../Keil/ARM/Boards/Keil/MCBSTM32/Demo
---	---

4 Steps...it's easy!

Step 1: Select Device & Specify Target Hardware



The Device Database™ has more than 150 ARM based microcontrollers. The database is available in μ Vision and at www.keil.com/dd.

Step 2: Create & Build Application



Preconfigured device start-up code

A built-in configuration wizard simplifies device setup

Extensive project templates, program examples and applications notes are included in RealView MDK.

Step 3: Analyze Program with μ Vision Device Simulator



Use the μ Vision Simulator to accelerate your project development cycle.

Develop software before hardware is available.

Verify and optimize your application using advanced features like instruction trace, code coverage, execution profiling and the μ Vision Logic Analyzer.

Step 4: Flash Download and Final Testing in Target

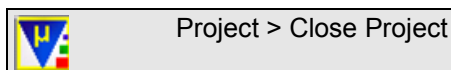


Download code to the Flash ROM of your target system easily using the Keil ULINK2 USB-JTAG Adapter.

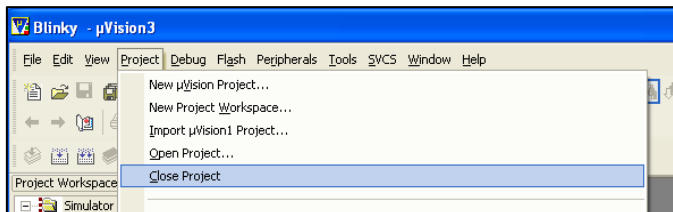
ULINK2 supports Flash programming, single-stepping, real-time program execution with breakpoints, access to memory and CPU registers, and real-time trace using Serial Wire Viewer (SWV).

Step 1: Select Device & Specify Target Hardware

Open and Clear RealView MDK



Project > Close Project

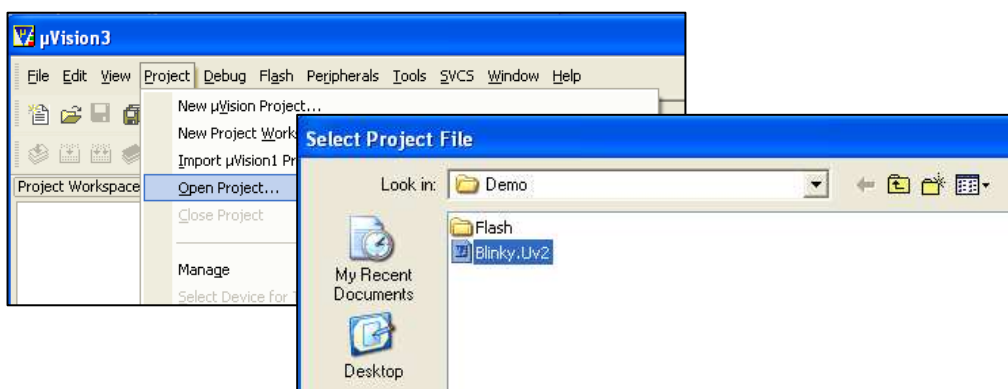


Open an Example Project



Project > Open Project...

Navigate to ../Keil/ARM/Boards/Keil/MCBSTM32/Demo/Blinky.Uv2

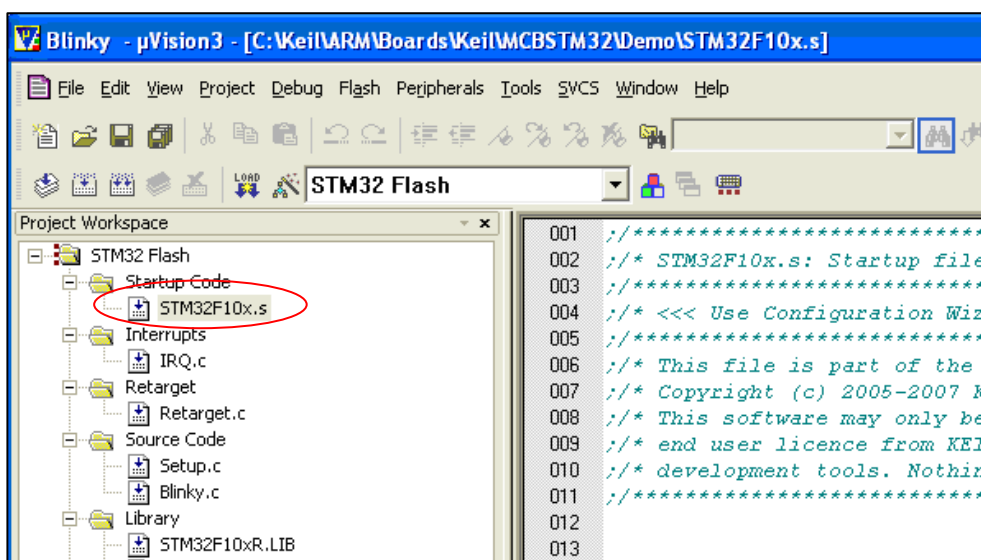


Startup Code

The example project 'Blinky' contains pre-configured target settings and start-up code for the STM32 microcontroller used on the MCBSTM32. This is more than 350 lines of code which the developer does not have to write!



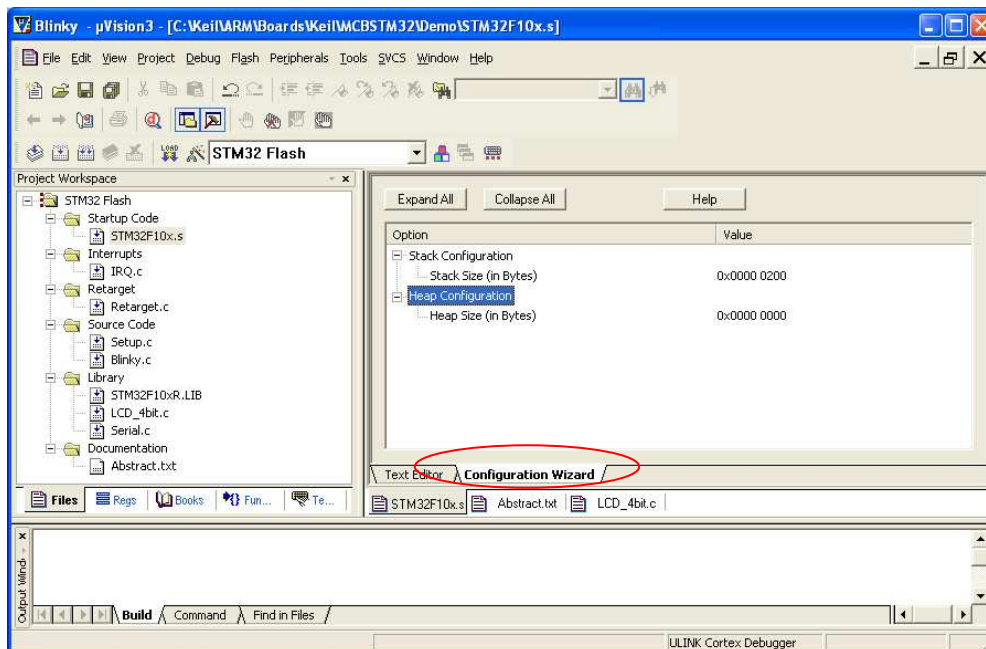
Double Click STM32F10x.s to open the start-up code



The Configuration Wizard allows the customer to easily select parameters related with a source file.



Click on the Configuration Wizard tab
Show the default settings for the size of the software stack and heap

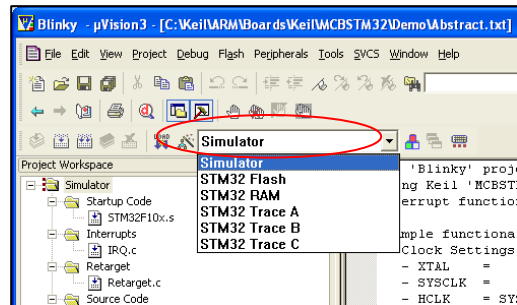


Step 2: Create & Build Application

Project Targets



Select the Simulator project target



Project targets allow customers to have different build options for a common set of source files. The Simulator target's build and debug options are pre-configured for software simulation.

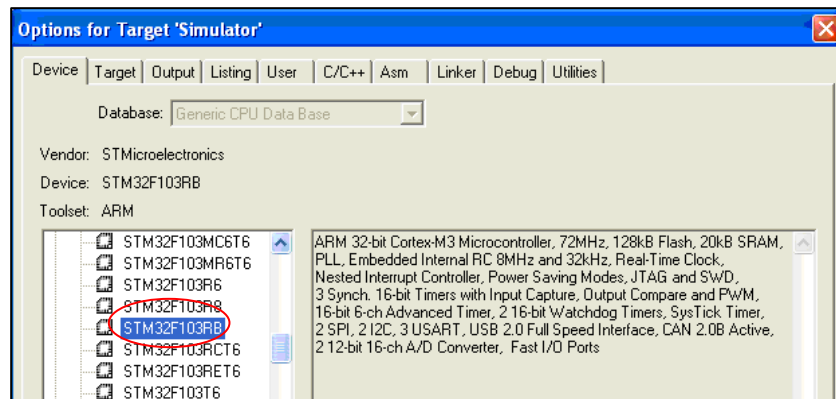
Target Options

All options for the device configuration can be examined and changed to the user's requirements.



Click 'Options for Target' button
Click on the 'Device' Tab

This dialog shows the microcontroller used on the MCBSTM32 Evaluation Board.



The CPU speed and memory mapping options are shown in the Target tab.

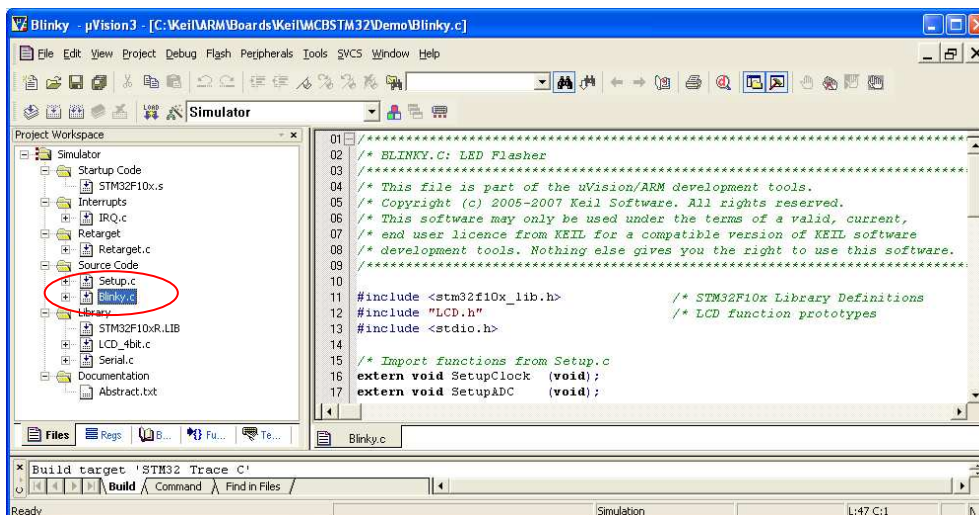
The Debug tab allows the customer to select simulation or connection to real targets, as well as the settings for each type of connection.

Write Application Code

For the demonstration we will use an existing application program.



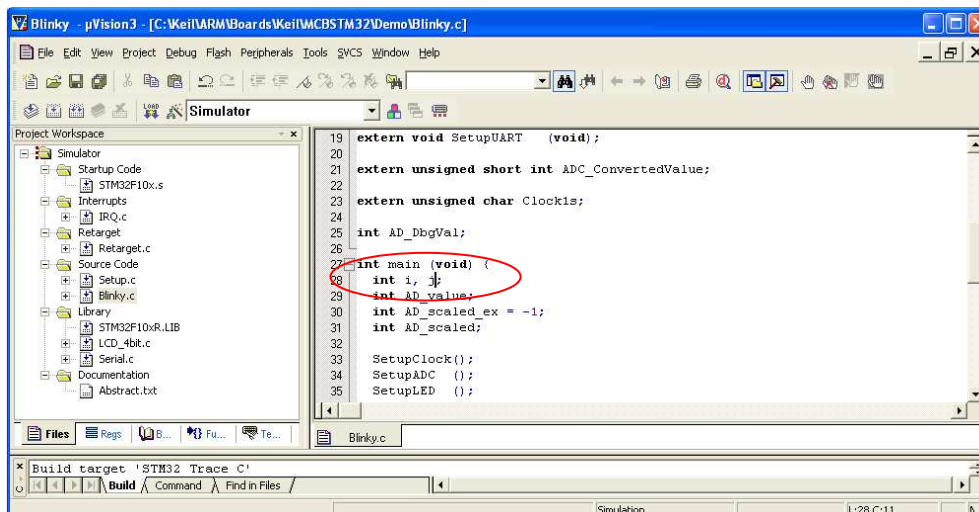
Double click on Blinky.c in the Files Panel



Blinky.c includes a main() function, which is executed after the system configuration following a reset.



Declare a new variable 'j' inside main()
Type CTRL+S to save the document

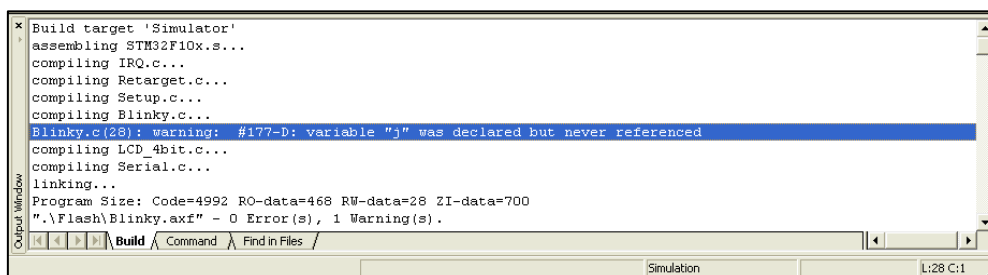


Build Program



Build Project using 'Build Target' button.


The example project builds but reports a warning.



The error message shows that the variable j declared on line 28 is not used in the code.

Correct Program Error and check the Code Size

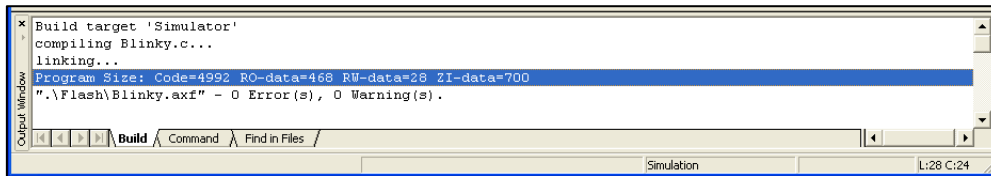
You can go directly to the code which has caused the error by double clicking the error message.

	<p>Double Click 'Blinky.c (28)' Delete variable 'j' from the source code Type CTRL+S to save the document</p>
---	---

Once the error is corrected the project can be rebuilt with no errors.


	<p>Rebuild Project Build Project using 'Build Target' button</p>
---	--

After compiling building without any error messages, the code size is reported to be 4,992 Bytes in the build window.



Use the MicroLib Library

MicroLib is a highly-optimized library for ARM-based embedded applications written in C. When compared to the standard C library included with the RealView Compilation Tools, MicroLib provides significant code size advantages required for many embedded systems.


	<p>Click 'Options for Target' button Target Tab > use MicroLib Click OK to close Options</p>
---	---

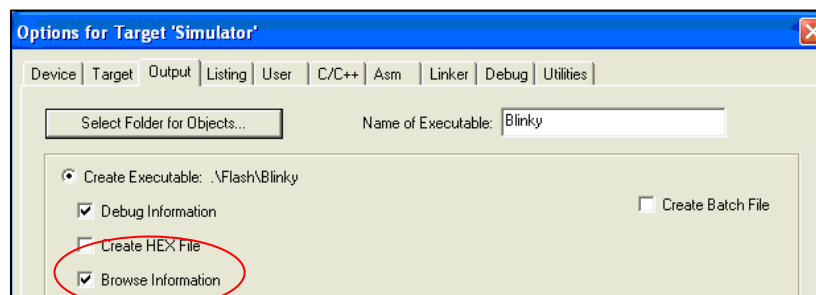
	<p>Rebuild Project Build Project using 'Build Target' button</p>
---	--

After building, the code size is reduced to 4,512 Bytes. Using the MicroLib library to compile Blinky.c results in a 10% reduction in the size of the compiled code.

Examine Variables with the Source Browser

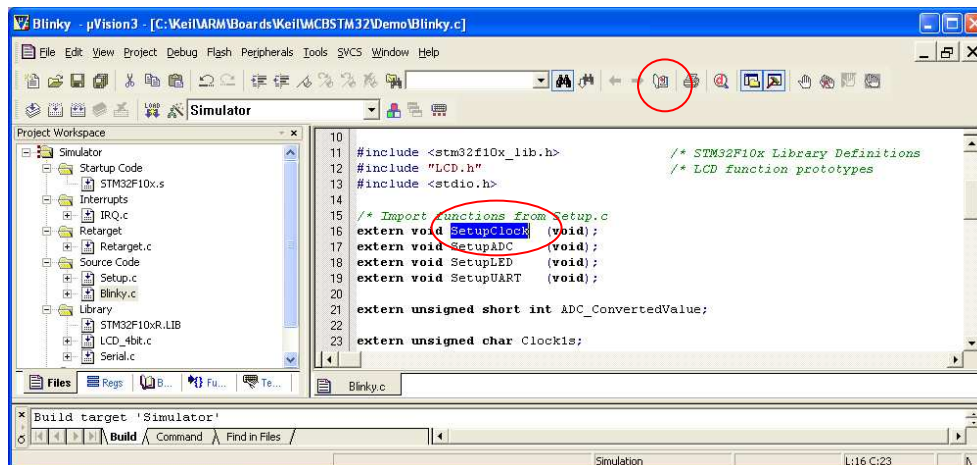
The source browser collects and displays information about program variables. μ Vision tracks where variables are defined as well as where they are used and how. In addition to displaying and arranging program symbols, the source browser allows you to jump between an object's name (in your source code) and its definition.

	<p>Click 'Options for Target' button Output Tab > Browse Information Click OK to close Options</p>
---	---





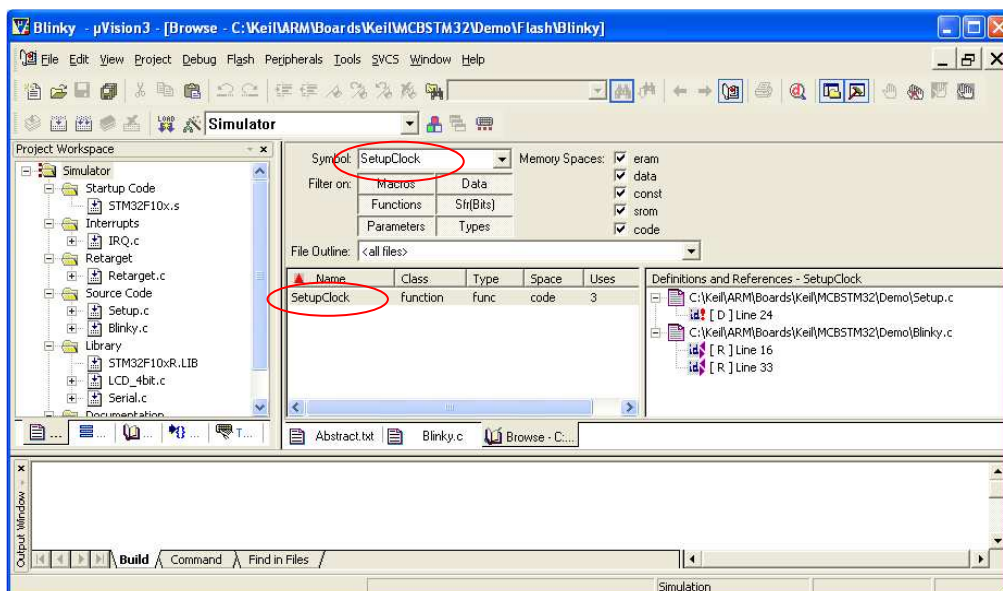
Highlight the function SetupClock at line 16 of Blinky.c
Click on the source browser window icon



This opens up the source browser window.



Click on the Symbol box and hit 'Enter' to use SetupClock as a filter
Click on SetupClock in the Browse window to update the references



The right part of the Browse window shows where the function is defined [D] and where it is being used [R]. Double click on 'Line 24' to go to the function definition.

Going to the definition of SetupClock can also be done directly from the 'Blinky.c' editor window by right clicking on the function name and selecting "Go to definition of 'SetupClock'".

Step 3: Analyze Program with μ Vision Device Simulator

The application can be built, run, analyzed and debugged all within μ Vision and the Device Simulator.

Start Debug Session



Start Debug Session by clicking 'Start/Stop Debug Session' icon

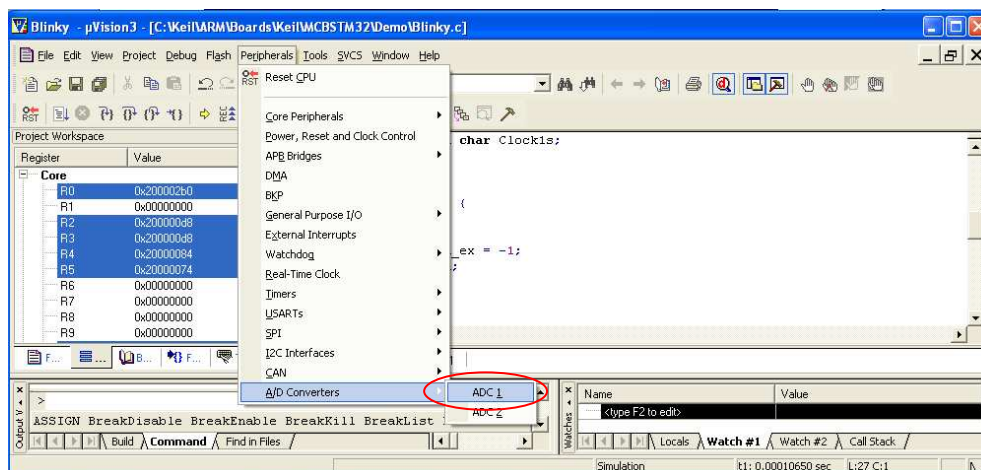
The processor executes the start-up code and stops at main() under control from uVision.

Examine Peripherals

Before running the application program, it is a good idea to look at the peripheral views, this will help you to confirm correct program operation when running, as you will be able to examine and change peripheral registers.



Open A/D Converter Peripheral Window
Peripherals > A/D Converters > ADC1



This opens the A/D Converter Peripheral Window.



Open GPIO Peripheral Window
Peripherals > General Purpose I/O > GPIOB



Click on the Serial Window #1' icon
This opens a window to display messages sent to UART #1

Run Program



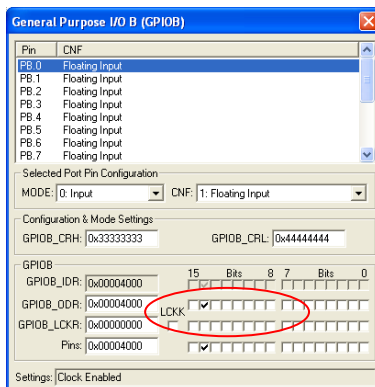
Run Program
Using Run Button

Blinky reads the value of a potentiometer approximately every 10 seconds. It sends this value to UART #1 and toggles LEDs with a frequency related to it.

With the program running it is possible to examine the peripherals to confirm correct program operation.

GPIO


On a hardware target, GPIOB would be connected to LEDs, allowing the Blinky program to active the individual LEDs. In simulation you can see this is functioning correctly with GPIOB_ODR[15:8] changing in a set sequence.




A/D Input

The input voltage of the A/D can be changed to any value up to its maximum. In this case the microcontroller can convert analog inputs between 0 and 3.3V to a digital value between 0 and 0xFFFF.

The inputs are 5V tolerant, but analog values over 3.3V are read as 0xFFFF. Analog values over 5V will result in an error message from μ Vision. Try it!

 **A/D Converter Peripheral Window**
Enter a value of 6.0V in the ADC1_IN1 box and press TAB
 μ Vision displays an error message

 Enter a value of 3.0V in the ADC1_IN1 box and press TAB

The screenshot shows the 'Analog / Digital Converter 1 (ADC1)' configuration window. The 'ADC1_IN1' input is highlighted with a red circle, showing a value of 3.0000. The 'Analog Inputs' section shows ADC1_IN1: 3.0000 and ADC1_IN0: 0.0000. The 'VOLTAGES' section shows VREFP: 3.3000, VREFN: 0.0000, VTEMP1: 3.3000, and VREFINT: 3.3000. The 'Settings' section shows 'Clock Enabled, ADCCLK: 18.00 MHz, Converting channel: 1'.

The next AD value displayed in the UART #1 window changes to 0x0E8B. In the GPIOB window you can see how the speed at which the outputs toggle has been reduced.

Logic Analyzer

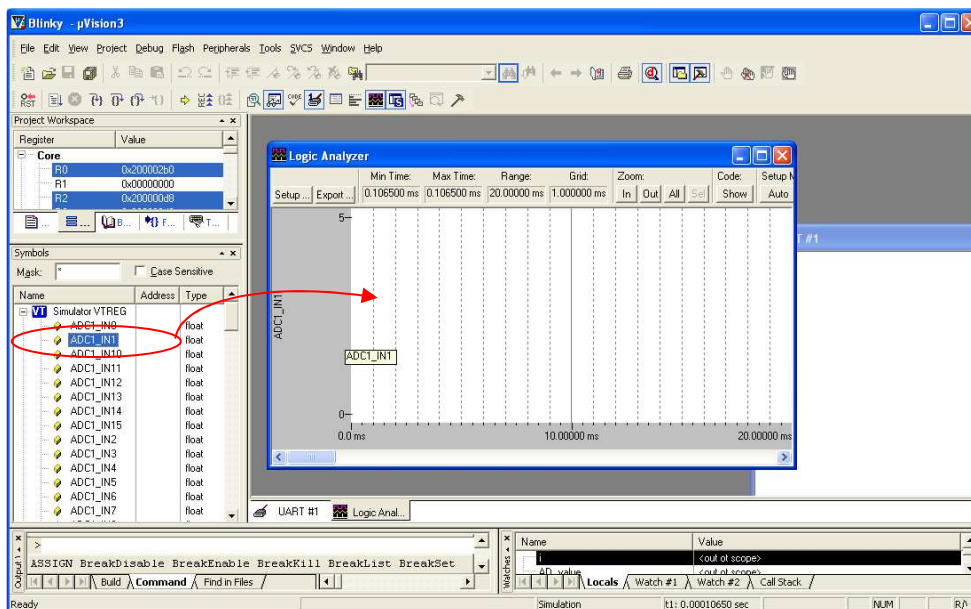
μ Vision includes a logic analyzer window which enables you to examine and manipulate peripherals in a visual, interactive manner.



Open Symbol Window using
Expand 'Simulator VTREG'



Open Logic Analyzer Window
Drag and drop ADC1_IN1 and PORTB from Symbol Window



The Logic Analyzer shows the voltage on ADC1_IN1 and the digital value of GPIOB. Blinky toggles the LEDs connected to GPIOB at a frequency determined by the level of ADC1_IN1.

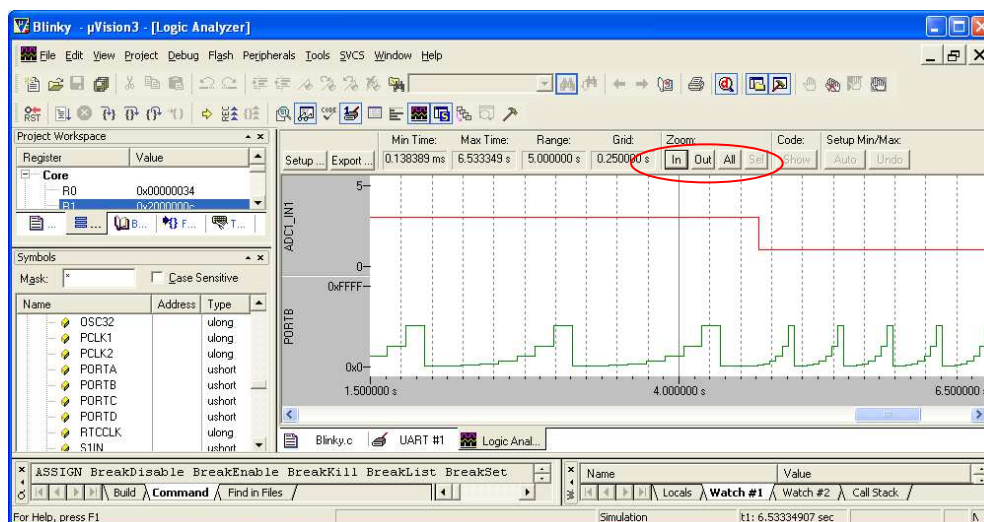


Select the A/D Converter 1 Window
Change ADC1_IN1 value to 1.0 and press TAB

In the logic analyzer window you will see ADC1_IN1 decrease to 1.0V and following a short delay, the frequency of PORTB reduces.



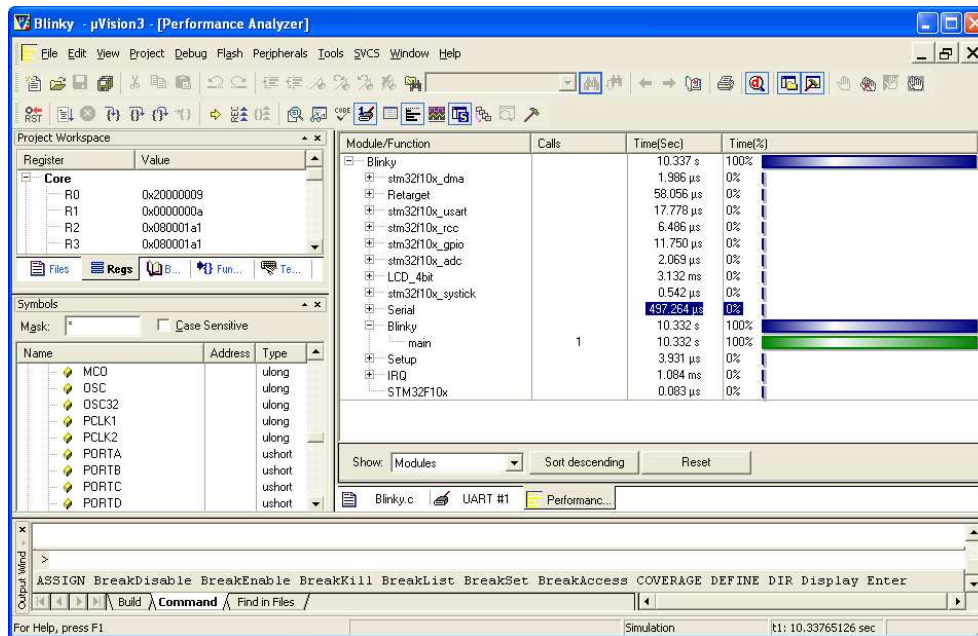
Select the Logic Analyzer Window
Use the Zoom buttons to focus on the right side of the window



Analyze Performance

Whilst running the program in simulation you are able to optimize its performance by using the μ Vision's Trace and Profiling tools.

Open Performance Analyzer Window
Double-click Blinky to show running processes



By double clicking 'main' you go to the code view for this function, allowing easy editing and optimization.


When running Blinky the microcontroller spends most of the time in an infinite loop in main(), waiting for the interrupt service routine to update the value read from ADC1_IN1.

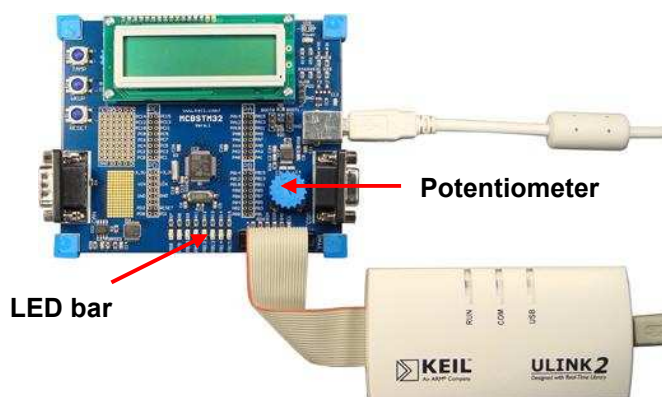
Step 4: Flash Download and Final Testing in Target




 Halt Program
 Move to Build View by clicking 'Start/Stop Debug Session' icon

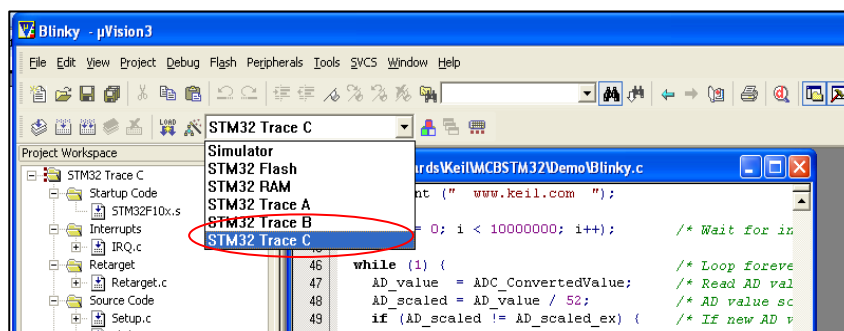
Hardware Setup


 Connect a USB cable from your computer to the MCBSTM32 board
 Connect a USB cable from your computer to ULINK2
 Connect ULINK2 to the 20-pin JTAG connector on the board




Project Target


 Select the STM32 Trace C project target




The STM32 Trace C target's build and debug options are pre-configured for program execution from internal Flash with trace enabled.

Rebuild Program and Download to Flash


 Rebuild Project using 'Rebuild all Target Files' button.

The example project builds without errors or warnings.


 Download program to Device Flash
 Click 'Load' button

The program will download to the on-chip Flash within a few seconds, μ Vision will confirm that it has loaded correctly.

Debug your Program on the Target Board



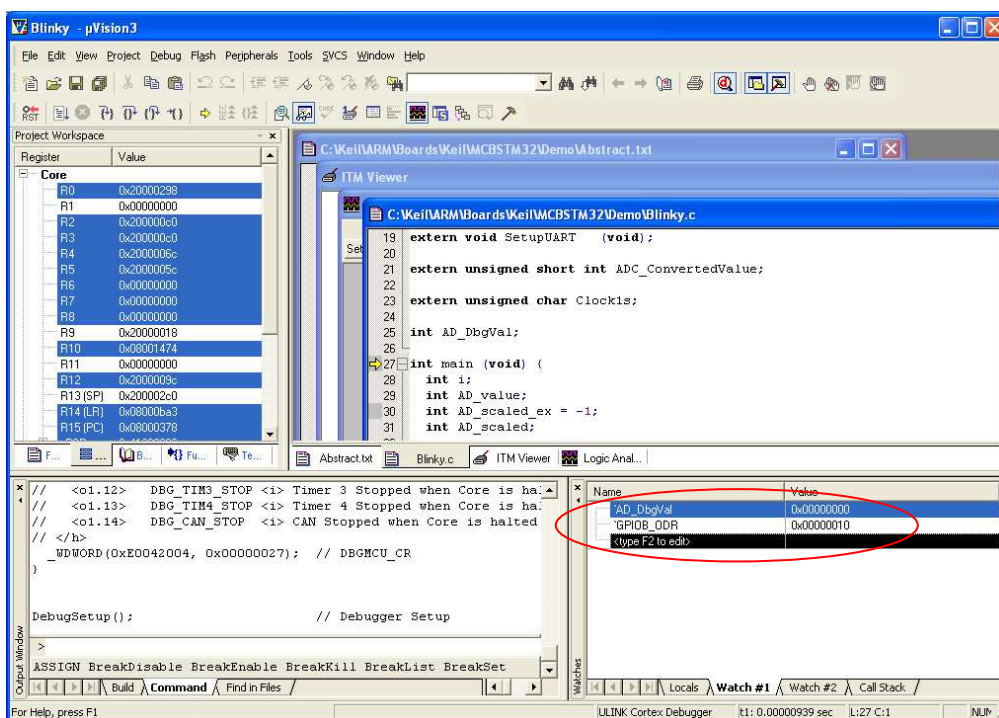
Start Debug Session by clicking 'Start/Stop Debug Session' icon

The processor executes the start-up code and stops at main() under control from uVision.

When connected to the STM32 microcontroller μ Vision can read and write variables in memory while the processor is running. This is done in hardware, so the microcontroller does not need to run a debug agent application in the background.



Click on the 'Watch and Call Stack Window' icon
 Click on the Watch #1 Tab of the watch window
 Click on the first entry, press F2 and write AD_DbgVal
 Click on the second entry, press F2 and write GPIOB_ODR



AD_DbgVal is a variable in Blinky.c that contains the potentiometer value read from the A/D Converter.

GPIOB_ODR is the output port that drives the LED bar. The LEDs are driven by bits [15:8] of GPIOB_ODR.



Run Program
 Using Run Button

The value of variable AD_DbgVal changes as you rotate the potentiometer. The value of GPIOB_ODR changes all the time as the LEDs are updated by the software.



In the Watch #1 window click on the value of GPIOB_ODR
 Press F2 and write a new value 0x0000FF10

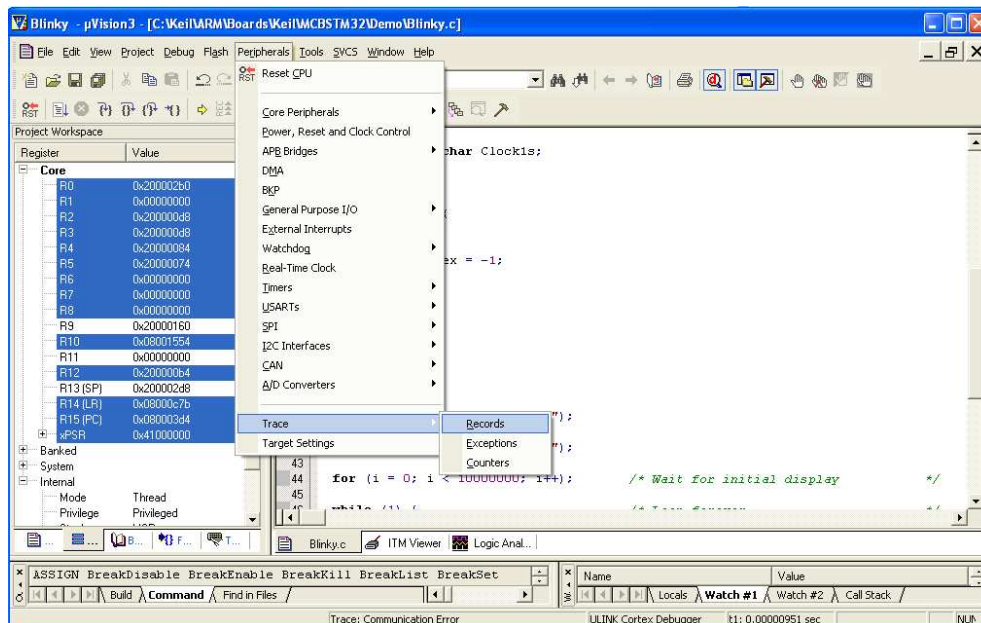
When you update GPIOB_ODR the microcontroller flashes the 8 LEDs for an instant before the running software overwrites the value of GPIOB_ODR.

Trace Data Accesses

Trace allows customers to record exceptions and accesses to memory, sample the program counter, and count processor events. All this is done while the processor is running, so it is not intrusive.



Open a Trace Records Window
Peripherals > Trace > Records



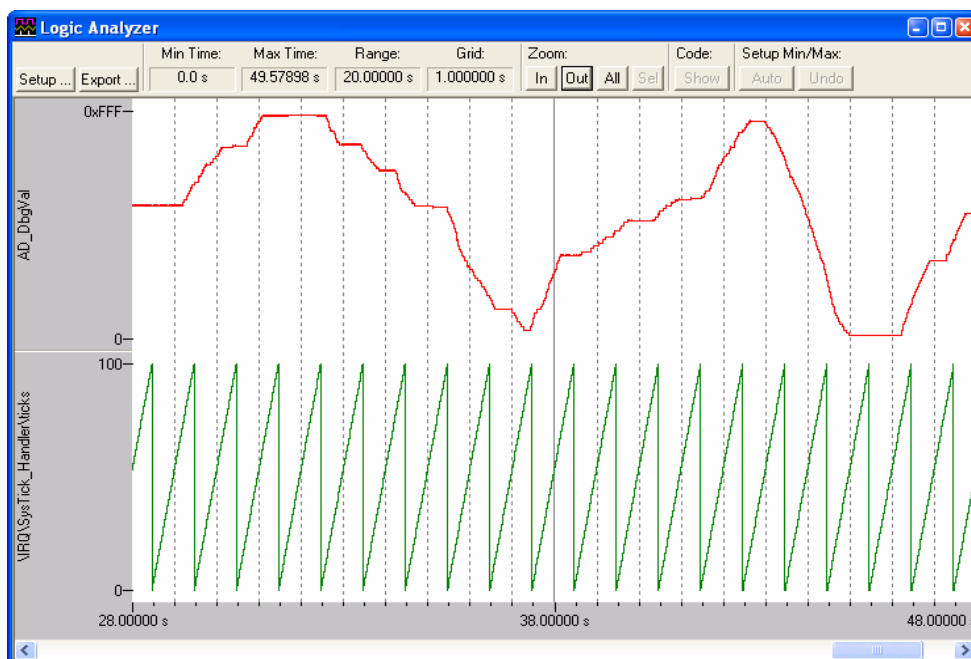
The trace records window shows the data trace: processor accesses to external memory. It includes the address of the variable in memory, the type of access (read or write), the data read or written and the time when the access happened.

Trace Records								
Type	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]
Data Read			20000004H	00000000H			709393	0.00985268
Data Write			20000004H	00000001H		X	710603	0.00986949
Data Read			20000004H	00000001H			1429401	0.01985279
Data Write			20000004H	00000002H		X	1430609	0.01986957
Data Read			20000004H	00000002H			2149409	0.02985290
Data Write			20000004H	00000003H		X	2150615	0.02986965
Data Read			20000004H	00000003H			2869417	0.03985301
Data Write			20000004H	00000004H		X	2870621	0.03986974
Data Read			20000004H	00000004H			3589425	0.04985312
Data Write			20000004H	00000005H		X	3590627	0.04986982
Data Read			20000004H	00000005H			4309434	0.05985325
Data Write			20000004H	00000006H		X	4310633	0.05986990
Data Read			20000004H	00000006H			5029441	0.06985335
Data Write			20000004H	00000007H		X	5030639	0.06986999
Data Read			20000004H	00000007H			5749449	0.07985346
Data Write			20000004H	00000008H		X	5750645	0.07987007
Data Read			20000004H	00000008H			6469457	0.08985357
Data Write			20000004H	00000009H		X	6470651	0.08987015
Data Read			20000004H	00000009H			7189465	0.09985368
Data Write			20000004H	0000000AH		X	7190657	0.09987024



Select the Logic Analyzer Window

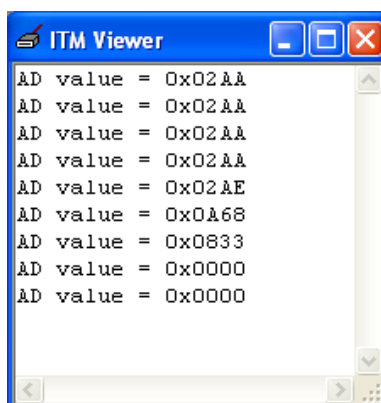
The window shows a trace of variable AD_DbgVal. As you turn the potentiometer the processor updates this variable in memory and the value of the variable is displayed in the Logic Analyzer Window.



Display Instrumentation Trace



In the “STM32 Trace C” target the printf function is redirected to the Instrumentation Trace Macrocell (ITM).



This allows the processor to print data in the ITM Viewer Window in real time, without stopping program execution and without requiring the use of communication ports such as UARTs.

Congratulations!

You have created a new embedded application, verified it in simulation and on a target device in just **4 Steps**.