EE3220 System-on-Chip Design

Tutorial 3: Debug with Keil Studio

Objective:

- **To get familiar with ARM Keil Studio and create project.**
- **To learn how to use debug tools in ARM Keil Studio.**
- **To find the change of LEDs during debug process.**

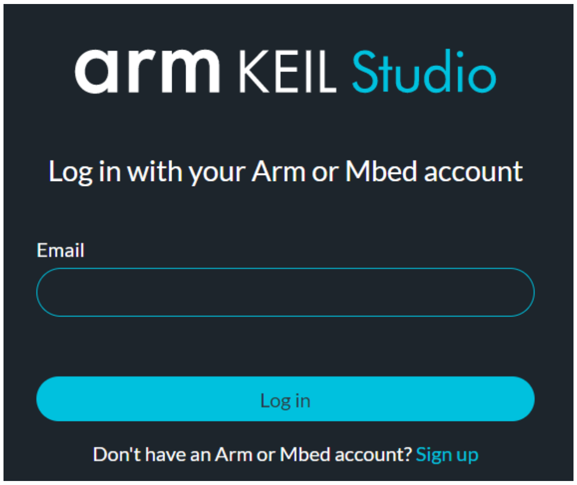
In this tutorial, we aim to change LED blink frequency by changing C code and run debug on Keil Studio. Keil Studio is the successor to the Mbed Online Compiler, and allows you to debug from supported browsers without the need to install any software.

Introducing the ARM Keil Studio

Arm announced the open beta of the Keil Studio Cloud in the year 2021, a new browser-based IDE. It's free to use, comes with a powerful, modern editor. Keil Studio Cloud is the evolution of Mbed Studio, using the same underlying IDE framework and codebase, adapted with additional support for the CMSIS ecosystem. It is a big step up from the Online Compiler, bringing much better code hinting/IntelliSense, comprehensive git source control integration and direct development board interaction from the browser. Like the Online Compiler, Keil Studio Cloud requires no installation and provides you with an integrated workspace.

Step 1: Log in to the ARM Keil Studio website.

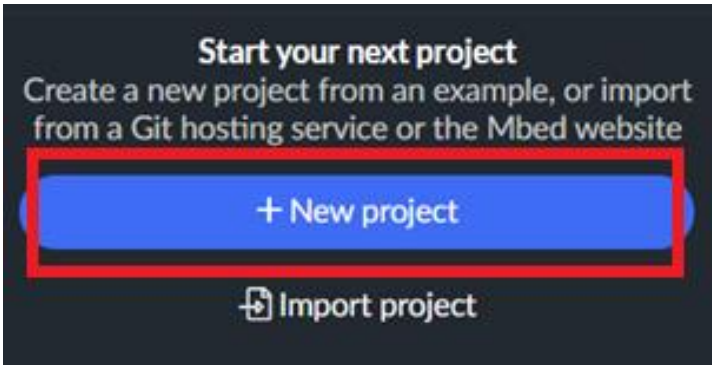
Go to the website of ARM Keil Studio and log in with your Mbed account.
https://studio.keil.arm.com/auth/login/



arm KEIL Studio
Log in with your Arm or Mbed account
Email
Log in
Don't have an Arm or Mbed account? Sign up

Step 2: Create a blinky project and change the parameter.

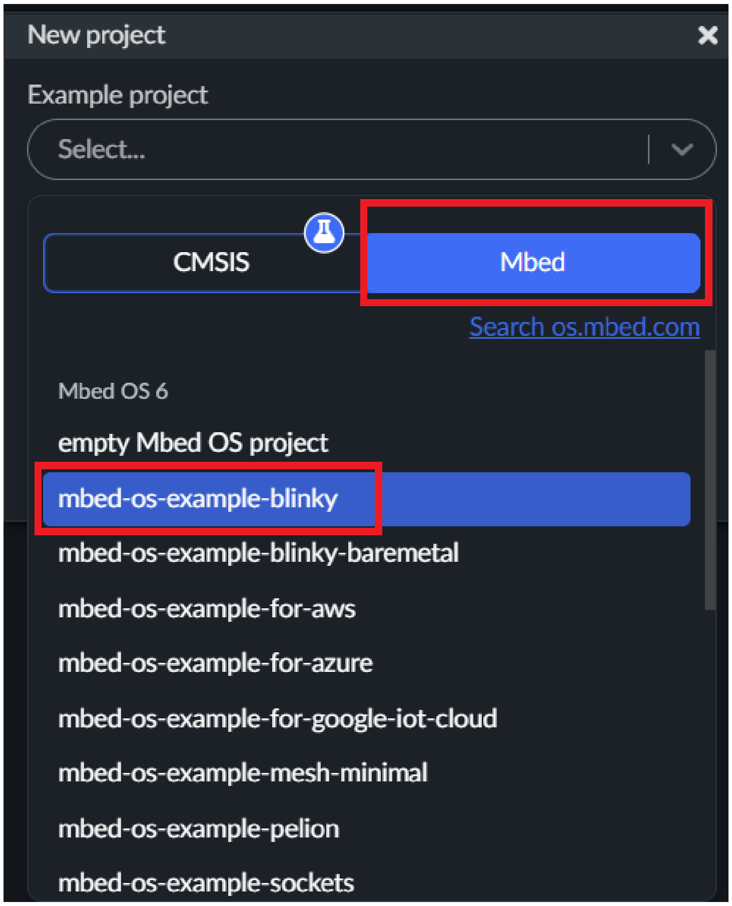
- Click **New project**.



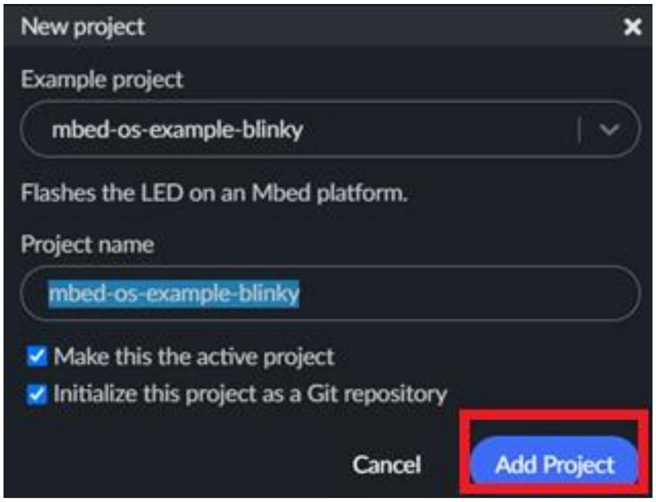
Start your next project

Create a new project from an example, or import from a Git hosting service or the Mbed website

- Select **mbed-os-example-blinky** template.



- Add Project.



New project

Example project

mbed-os-example-blinky

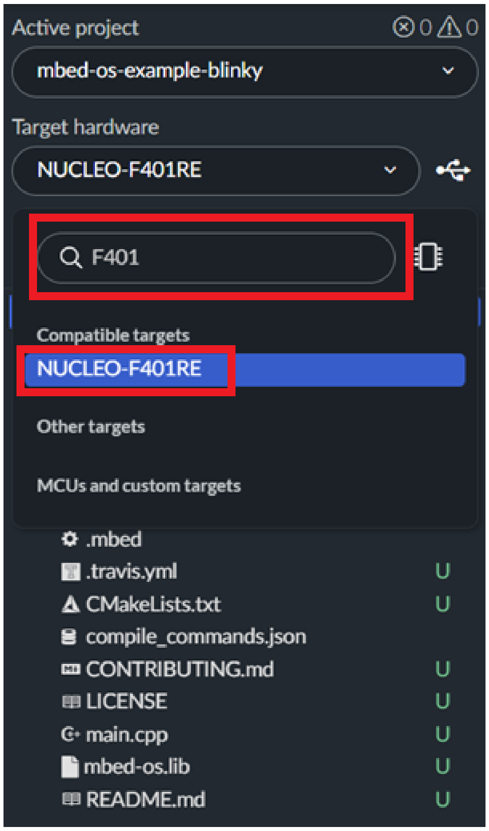
Flashes the LED on an Mbed platform.

Project name

mbed-os-example-blinky

☑ Make this the active project
☑ Initialize this project as a Git repository

- Search and select the target hardware: **F401.**



Active project ⊗0 ⚠0

mbed-os-example-blinky

Target hardware

NUCLEO-F401RE

F401

Compatible targets

NUCLEO-F401RE

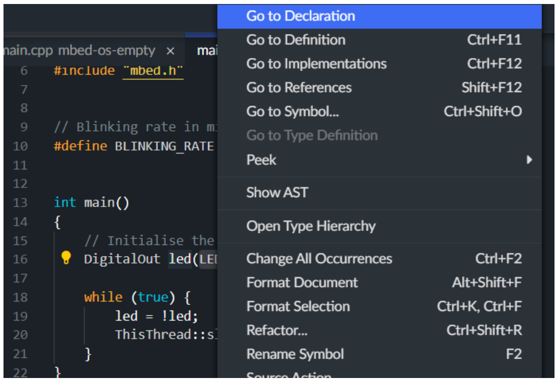
Other targets

MCUs and custom targets

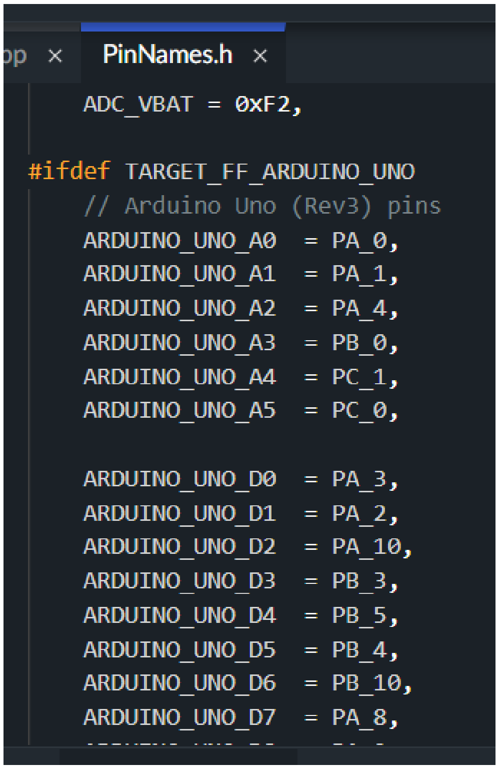
```
.mbed
.travis.yml               U
CMakeLists.txt            U
compile_commands.json
CONTRIBUTING.md           U
LICENSE                   U
main.cpp                  U
mbed-os.lib               U
README.md                 U
```

- Install the ST-link USB driver from the web page, prior to connecting the board. https://www.st.com/zh/development-tools/stsw-link009.html#get-software (After downloading, go to device manager -> Ports, select driver).
- Open PinNames.h, read more about pin definition on NUCLEO_F401RE board

(Open **main.cpp** file, select and right click **LED1**, Click **Go to declaration**).

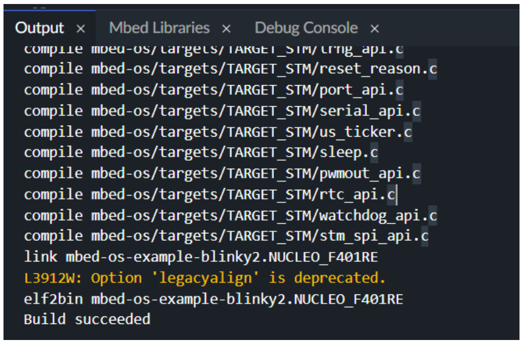

nain.cpp mbed-os-empty ×
mai
Go to Declaration
Go to Definition Ctrl+F11
Go to Implementations Ctrl+F12
Go to References Shift+F12
Go to Symbol... Ctrl+Shift+O
Go to Type Definition
Peek
Show AST
Open Type Hierarchy
Change All Occurrences Ctrl+F2
Format Document Alt+Shift+F
Format Selection Ctrl+K, Ctrl+F
Refactor... Ctrl+Shift+R
Rename Symbol F2
Source Action

```
#include "mbed.h"


// Blinking rate in m
#define BLINKING_RATE


int main()
{
    // Initialise the
    DigitalOut led(LE

    while (true) {
        led = !led;
        ThisThread::s
    }
}
```



```
    ADC_VBAT = 0xF2,

#ifdef TARGET_FF_ARDUINO_UNO
    // Arduino Uno (Rev3) pins
    ARDUINO_UNO_A0  = PA_0,
    ARDUINO_UNO_A1  = PA_1,
    ARDUINO_UNO_A2  = PA_4,
    ARDUINO_UNO_A3  = PB_0,
    ARDUINO_UNO_A4  = PC_1,
    ARDUINO_UNO_A5  = PC_0,

    ARDUINO_UNO_D0  = PA_3,
    ARDUINO_UNO_D1  = PA_2,
    ARDUINO_UNO_D2  = PA_10,
    ARDUINO_UNO_D3  = PB_3,
    ARDUINO_UNO_D4  = PB_5,
    ARDUINO_UNO_D5  = PB_4,
    ARDUINO_UNO_D6  = PB_10,
    ARDUINO_UNO_D7  = PA_8,
```

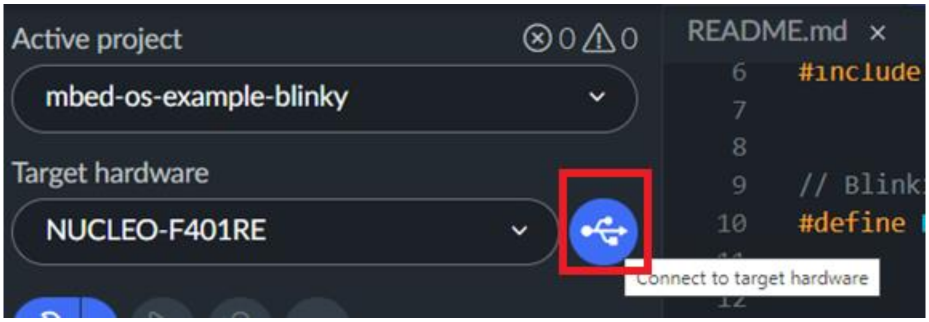
- Click Build Project, it starts to compile and generate **mbed-os-example-blinky.NUCLEO_F401RE.bin** file.



```
compile mbed-os/targets/TARGET_STM/trng_api.c
compile mbed-os/targets/TARGET_STM/reset_reason.c
compile mbed-os/targets/TARGET_STM/port_api.c
compile mbed-os/targets/TARGET_STM/serial_api.c
compile mbed-os/targets/TARGET_STM/us_ticker.c
compile mbed-os/targets/TARGET_STM/sleep.c
compile mbed-os/targets/TARGET_STM/pwmout_api.c
compile mbed-os/targets/TARGET_STM/rtc_api.c
compile mbed-os/targets/TARGET_STM/watchdog_api.c
compile mbed-os/targets/TARGET_STM/stm_spi_api.c
link mbed-os-example-blinky2.NUCLEO_F401RE
L3912W: Option 'legacyalign' is deprecated.
elf2bin mbed-os-example-blinky2.NUCLEO_F401RE
Build succeeded
```

(Note: Deprecating means that it may be usable but is regarded as obsolete. This warning does not affect the execution here.)

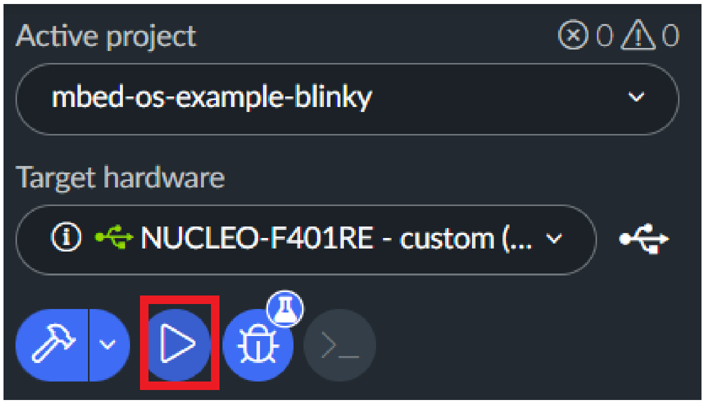
- Connect your board with USB port and then click **connect to target hardware** on website. Choose **STM32Link** and connect.



Active project: mbed-os-example-blinky

Target hardware: NUCLEO-F401RE

- Click run project and the generated executable file is downloaded to your board.

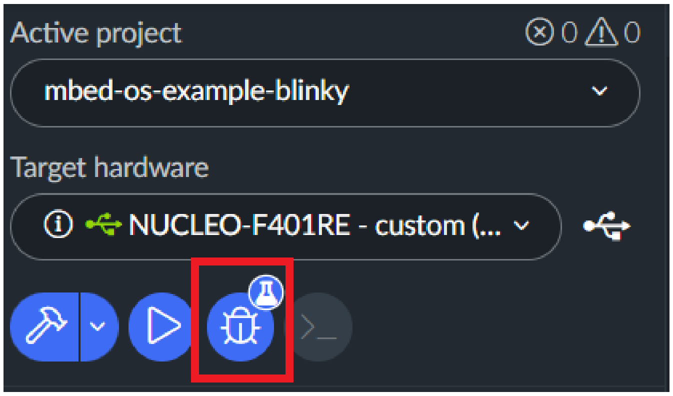


Active project: mbed-os-example-blinky

Target hardware: NUCLEO-F401RE - custom (...

Step 3: Debug your project.

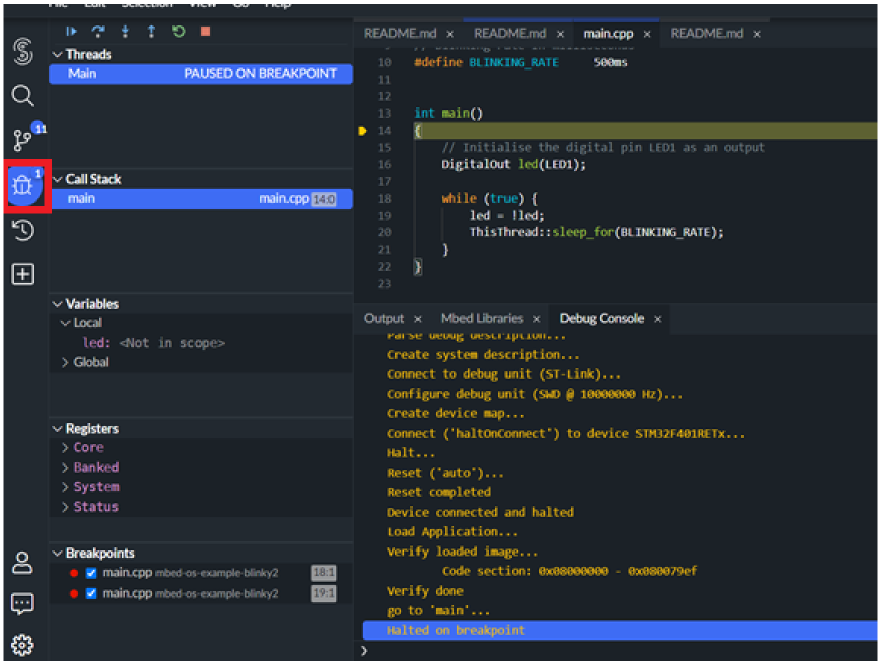
- Click Debug project.



Active project: mbed-os-example-blinky

Target hardware: NUCLEO-F401RE - custom (...

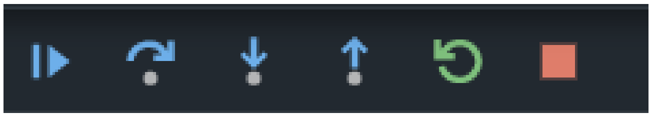
- Enter the Debug window when the buildup project is over.
  Confirm the generation of the debug interface.



```
#define BLINKING_RATE     500ms


int main()
{
    // Initialise the digital pin LED1 as an output
    DigitalOut led(LED1);

    while (true) {
        led = !led;
        ThisThread::sleep_for(BLINKING_RATE);
    }
}
```

```
Parse debug description...
Create system description...
Connect to debug unit (ST-Link)...
Configure debug unit (SWD @ 10000000 Hz)...
Create device map...
Connect ('haltOnConnect') to device STM32F401RETx...
Halt...
Reset ('auto')...
Reset completed
Device connected and halted
Load Application...
Verify loaded image...
        Code section: 0x08000000 - 0x080079ef
Verify done
go to 'main'...
Halted on breakpoint
```

- Find out the Debug menu.



Continue button: To start debugging, click the Continue button. The debugger runs to the first breakpoint it encounters and stops.

Step Over: Advance the debugger to the next source line that follows in the program execution to go straight to the parts of code you are more interested in.

Step Into: Advance the debugger into each function. The debugger then breaks on the first line that gets executed in the function.

Step Out: Advance the debugger until the current function returns (in other words, advance all the way through the current function).

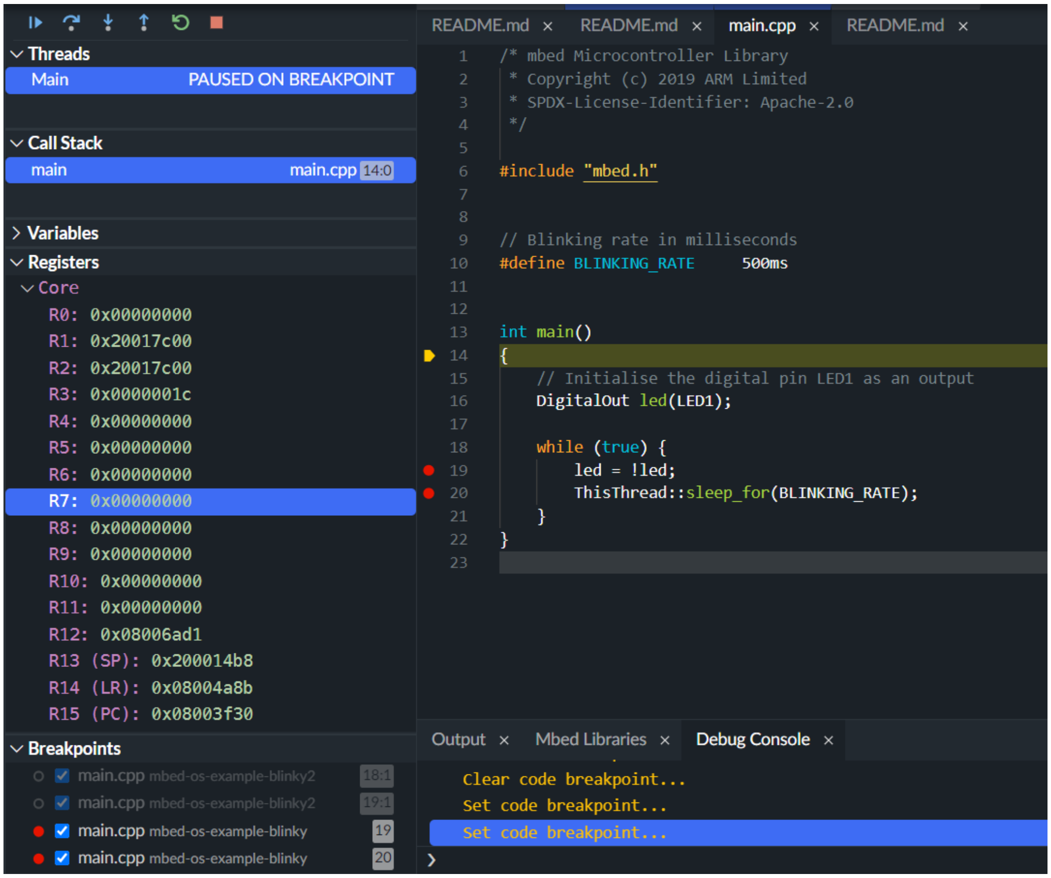
Restart button.

Stop button: To stop the debugger and return to the editor.

- Select the breakpoint you want to debug, then click continue button.

- Observe the changes of the small LED lights during debugging.

- Look at the value changes in Registers.



Threads
Main PAUSED ON BREAKPOINT
Call Stack
main main.cpp 14:0
Variables
Registers
Core
Breakpoints
○ ☑ main.cpp mbed-os-example-blinky2 18:1
○ ☑ main.cpp mbed-os-example-blinky2 19:1
● ☑ main.cpp mbed-os-example-blinky 19
● ☑ main.cpp mbed-os-example-blinky 20
README.md ×
README.md ×
main.cpp ×
README.md ×
Output ×
Mbed Libraries ×
Debug Console ×

```
R0: 0x00000000
R1: 0x20017c00
R2: 0x20017c00
R3: 0x0000001c
R4: 0x00000000
R5: 0x00000000
R6: 0x00000000
R7: 0x00000000
R8: 0x00000000
R9: 0x00000000
R10: 0x00000000
R11: 0x00000000
R12: 0x08006ad1
R13 (SP): 0x200014b8
R14 (LR): 0x08004a8b
R15 (PC): 0x08003f30
```

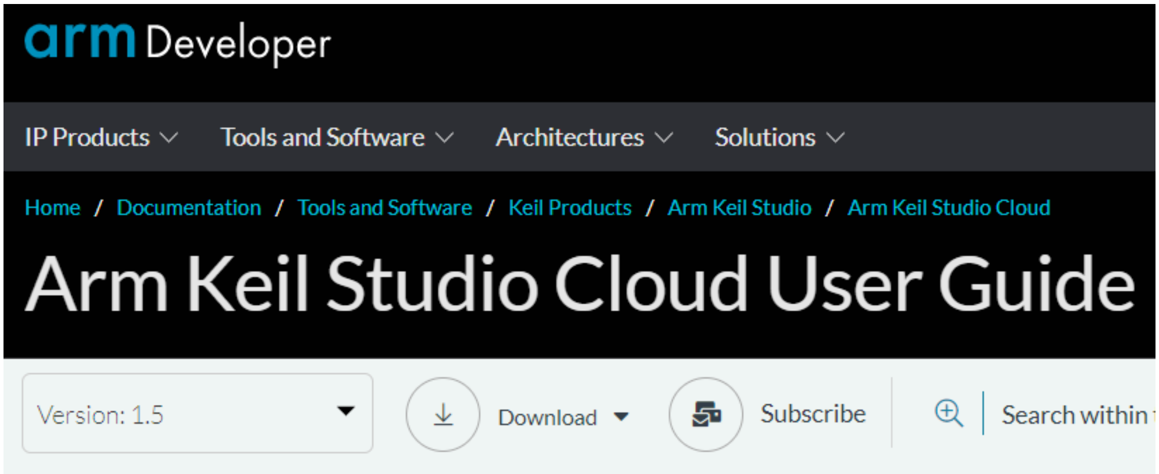
```cpp
/* mbed Microcontroller Library
 * Copyright (c) 2019 ARM Limited
 * SPDX-License-Identifier: Apache-2.0
 */

#include "mbed.h"


// Blinking rate in milliseconds
#define BLINKING_RATE     500ms


int main()
{
    // Initialise the digital pin LED1 as an output
    DigitalOut led(LED1);

    while (true) {
        led = !led;
        ThisThread::sleep_for(BLINKING_RATE);
    }
}

```

```
Clear code breakpoint...
Set code breakpoint...
Set code breakpoint...
```

- Observe carefully: How many registers change the value? How it is changed?
- Change the BLINKING_RATE parameter, is there any value changed in the register during debug?

Reference website for Keil Studio

- Arm Keil Studio Cloud User Guide. https://developer.arm.com/documentation/102497/1-5/Monitor-and-debug


arm Developer
IP Products
Tools and Software
Architectures
Solutions
Home / Documentation / Tools and Software / Keil Products / Arm Keil Studio / Arm Keil Studio Cloud
Download
Subscribe
Search within

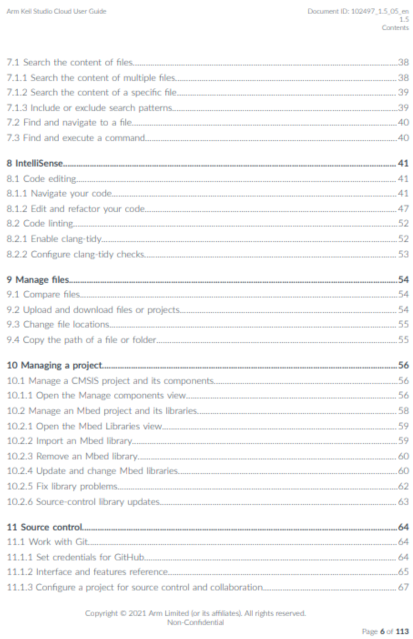
Arm Keil Studio Cloud User Guide

Version: 1.5

- Download the full user guide pdf file.

https://documentation-service.arm.com/static/616ea229ac265639eac59559?token=

Contents



END