APM32F003x4x6

Tutorial by Imeldushiii

The easiest way, using Keil Uvision instead of OpenOCD :)

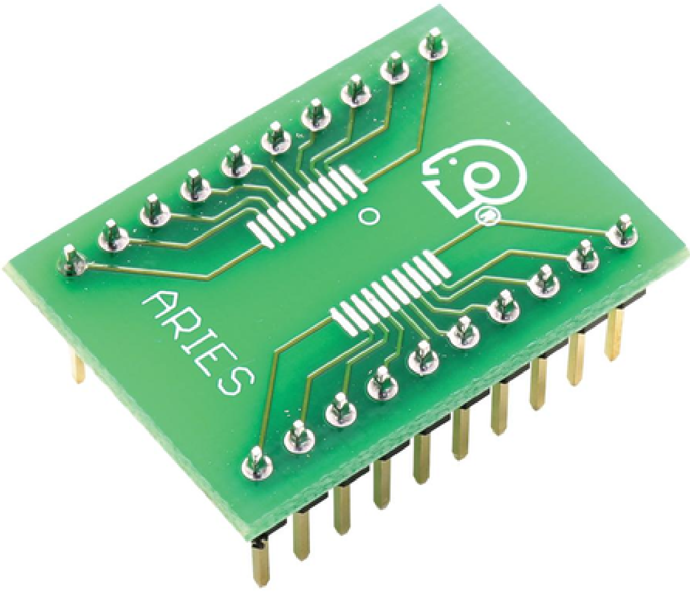
1. Gather the things you need

OR

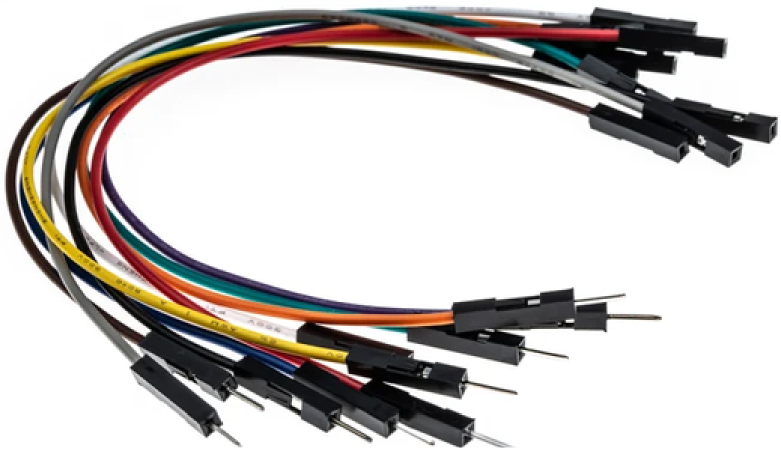
I use Geehy DAP Debbuger, because its cheaper and is from the same manufacturer that created this MCU

arm
KEIL

ARM Keil IDE
https://www.keil.com/download/product/

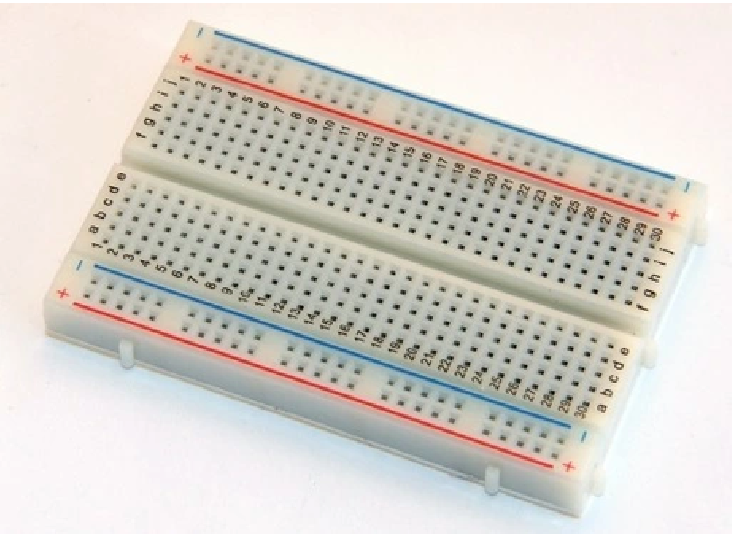


TSSOP 20 Adapter
or if you know how to create PCB, then you can create your own adapter :)



Jump Wires

The most important thing, 1uF Capacitor! Later i will explain why



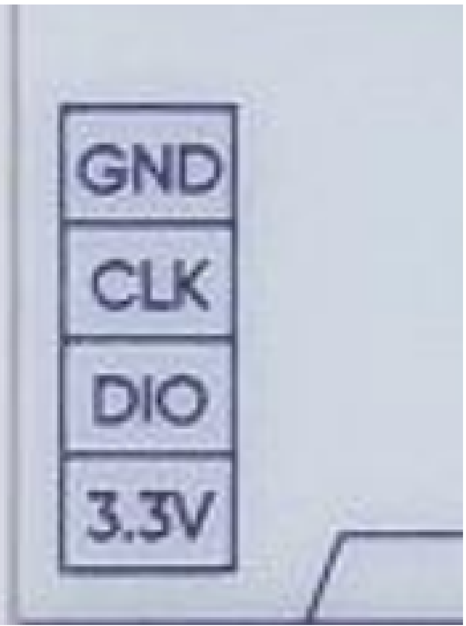
Prototype Board

And of course our lovely MCU :)
APM32F003

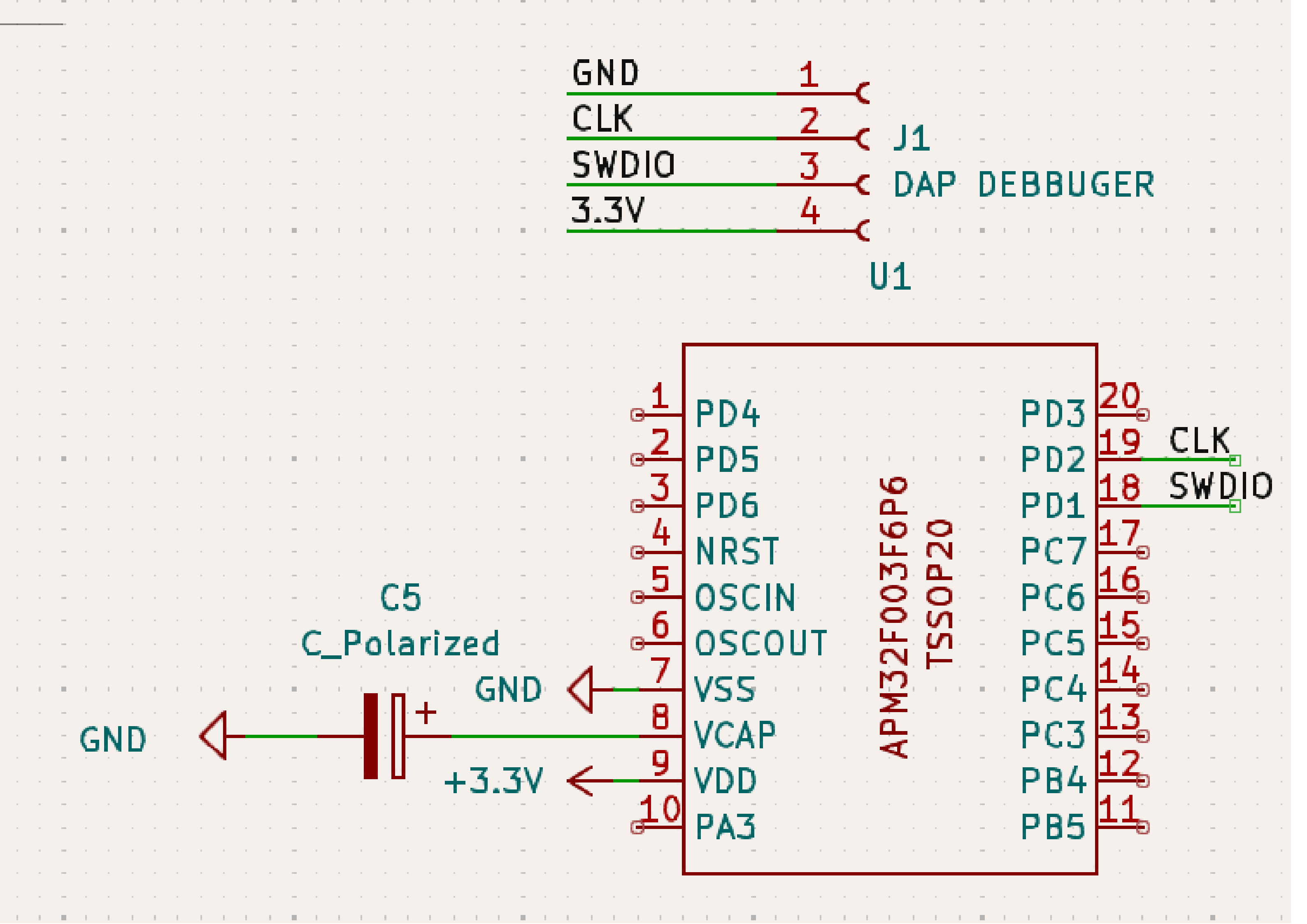
BTW, why im using this MCU?
Why Tssop20?
Why APM32 instead of STM32?
Answer is simple..
i bought them because I found them at an auction for a veeeeery cheap money xD

2. Connect MCU to debbuger

Its Simple, im using DAP debbuger by Geehy so that's how I connected them:



Using this 4 pins, just like in st-link



+3.3V to Pin(9)

GND to Pin(7)

CLK to Pin(19)

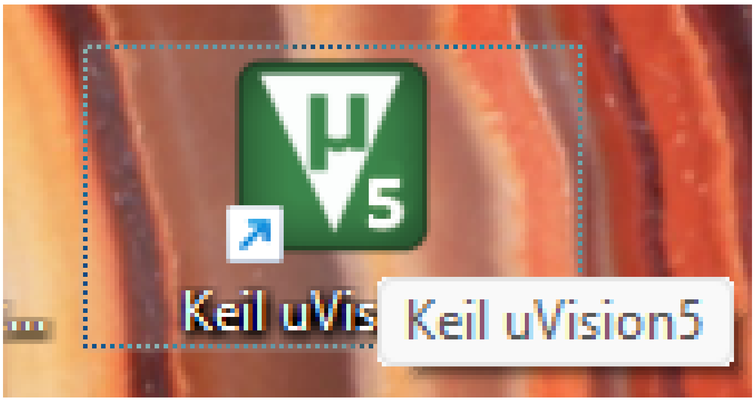
SWDIO to Pin(18)

Polarized Capacitor (1 uF) to Pin(8)

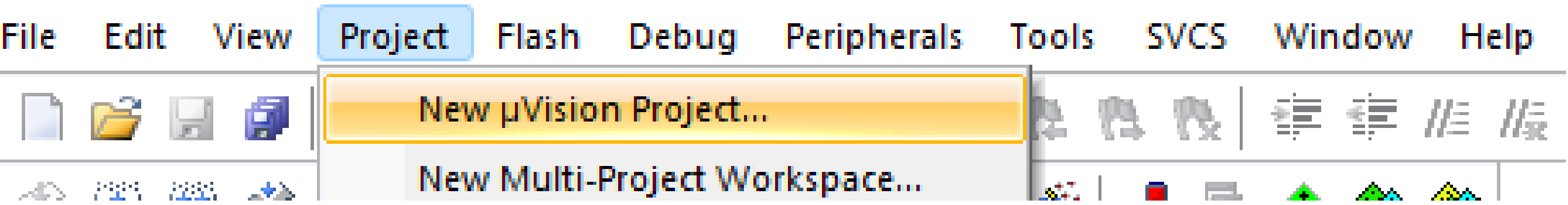
Warning! Capacitor is veery Important, I've spend hour debbuging with different Errors just to realise i need connect Capacitor. Read Documentation veery carefully!

3. Create Project

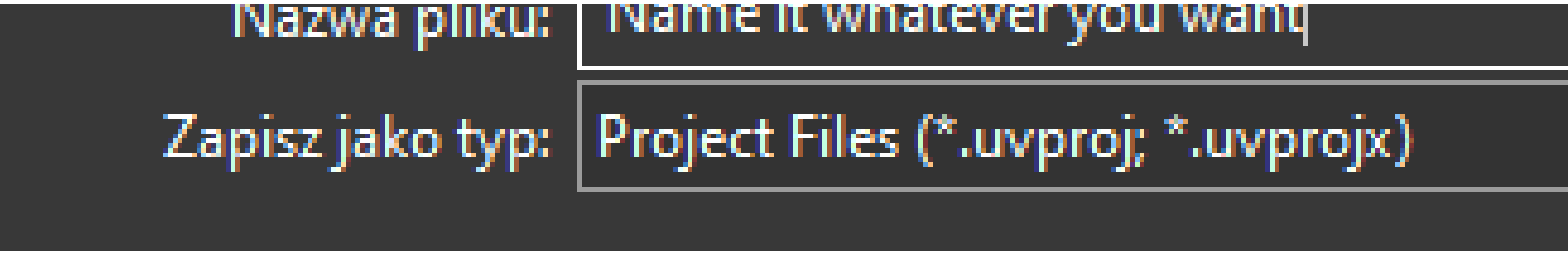
3.1 Open Keil



3.2 Create Uvision project



3.3 Save project Anywhere on your drive you want

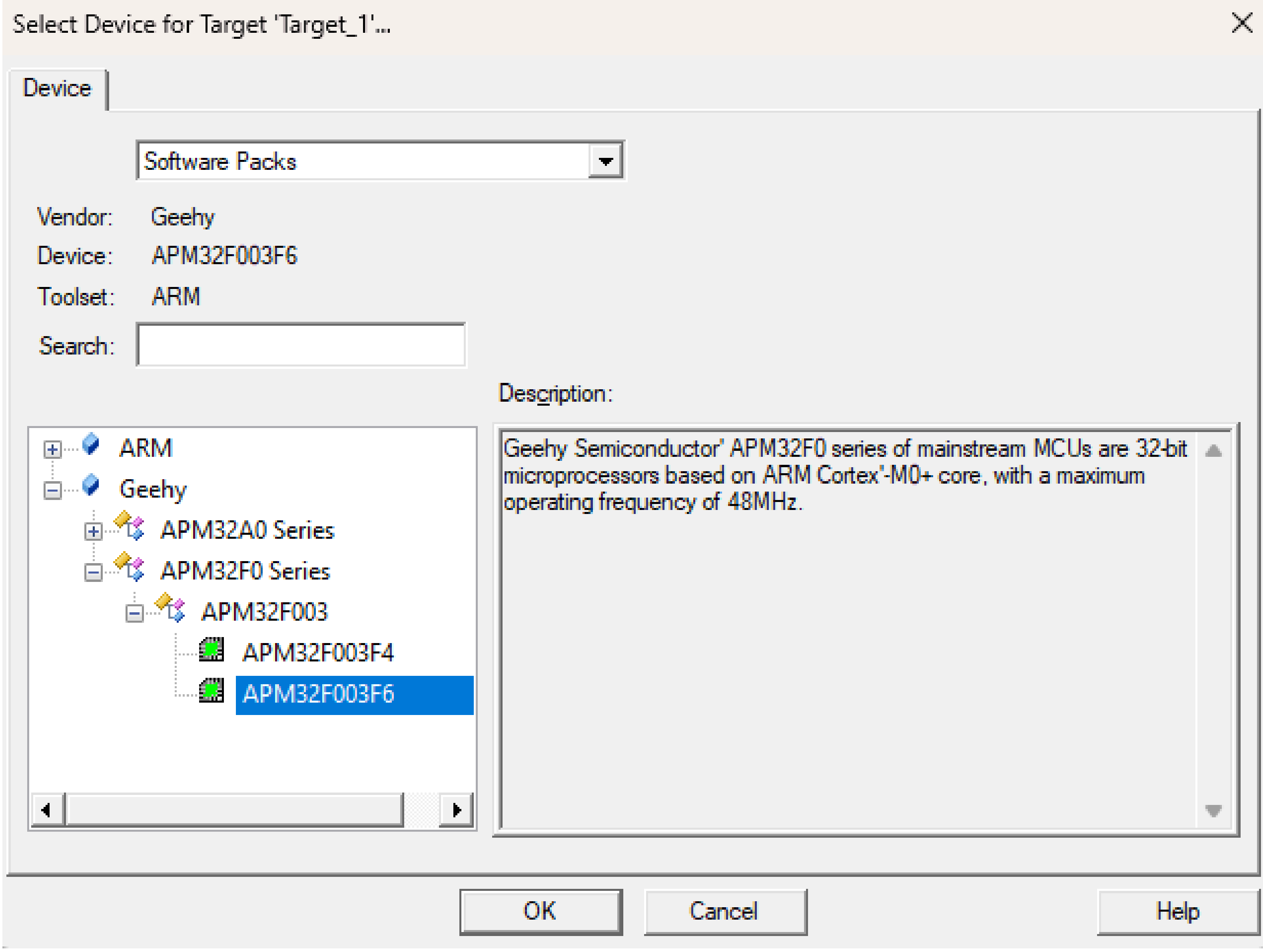


Nazwa pliku: Name it whatever you want

Zapisz jako typ: Project Files (*.uvproj; *.uvprojx)

And click “Save”

3.4 Select device



Select Device for Target 'Target_1'...

Software Packs

Vendor: Geehy

Device: APM32F003F6

Toolset: ARM

Search:

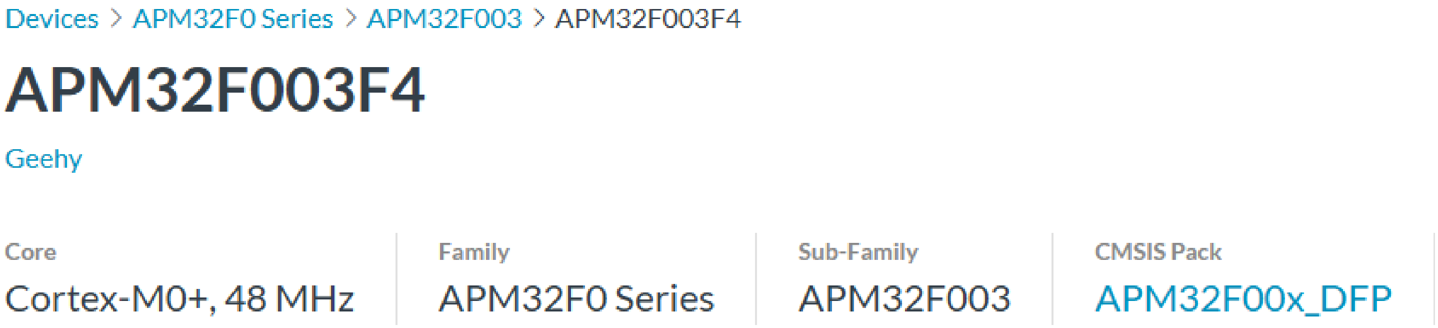
Description:

Geehy Semiconductor' APM32F0 series of mainstream MCUs are 32-bit microprocessors based on ARM Cortex'-M0+ core, with a maximum operating frequency of 48MHz.

WARNING! if you dont have APM32 packet, download here: https://www.keil.arm.com/packs/apm32f00x_dfp-geehy/boards/

Here’s how to download:

Click APM32F00x_DFP



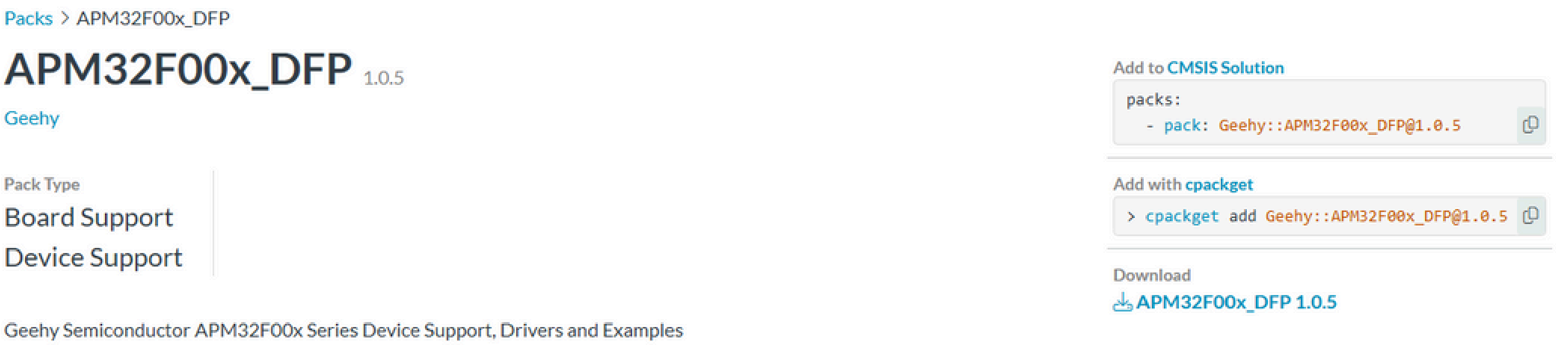
Devices > APM32F0 Series > APM32F003 > APM32F003F4

APM32F003F4

Geehy

Core	Family	Sub-Family	CMSIS Pack
Cortex-M0+, 48 MHz	APM32F0 Series	APM32F003	APM32F00x_DFP

And then click APM32F00x_DFP 1.0.5



Packs > APM32F00x_DFP

APM32F00x_DFP 1.0.5

Geehy

Pack Type
Board Support
Device Support

Geehy Semiconductor APM32F00x Series Device Support, Drivers and Examples

Add to CMSIS Solution

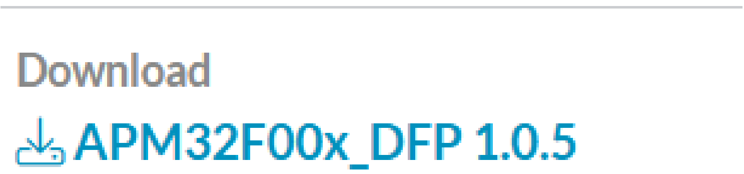
```
packs:
  - pack: Geehy::APM32F00x_DFP@1.0.5
```

Add with cpackget

```
> cpackget add Geehy::APM32F00x_DFP@1.0.5
```

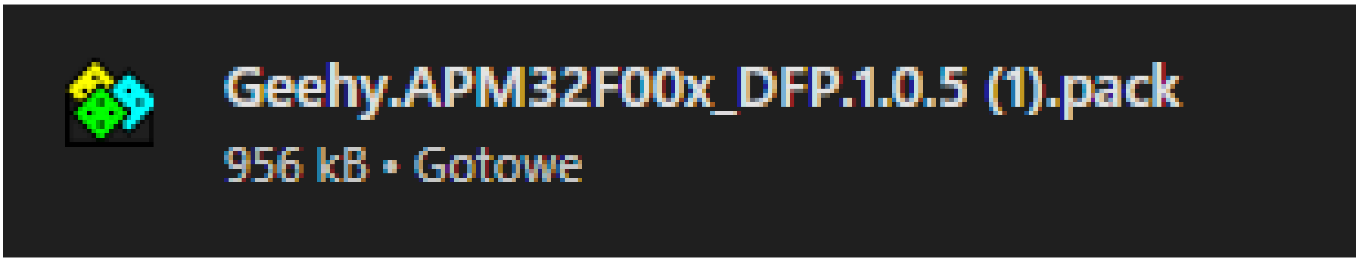
Download
APM32F00x_DFP 1.0.5

i mean CLICK this:



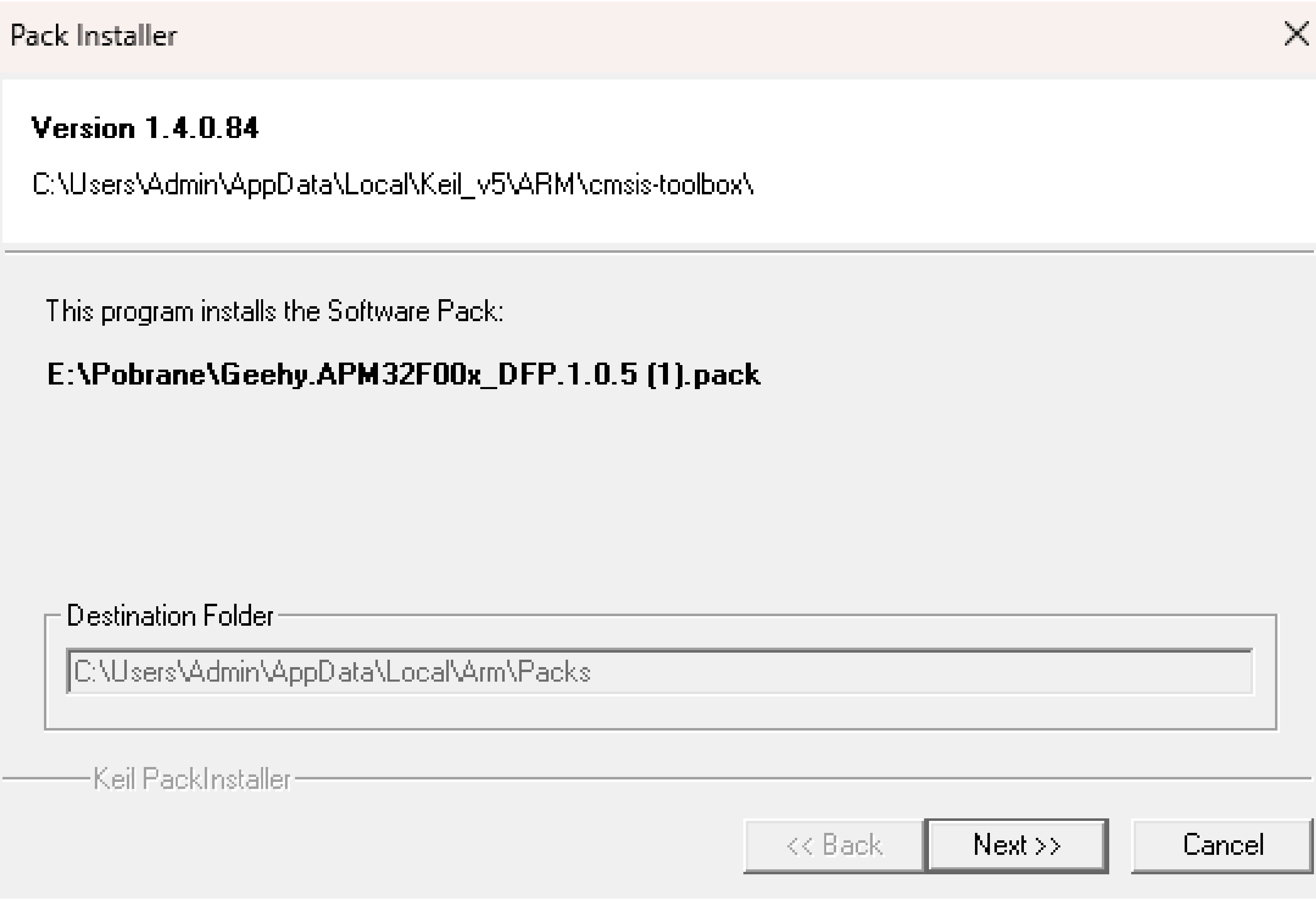
Download
APM32F00x_DFP 1.0.5

Okay, if you have already downloaded Click this:



Geehy.APM32F00x_DFP.1.0.5 (1).pack
956 kB • Gotowe

And you will see this:



Pack Installer

Version 1.4.0.84

C:\Users\Admin\AppData\Local\Keil_v5\ARM\cmsis-toolbox\

This program installs the Software Pack:

E:\Pobrane\Geehy.APM32F00x_DFP.1.0.5 (1).pack

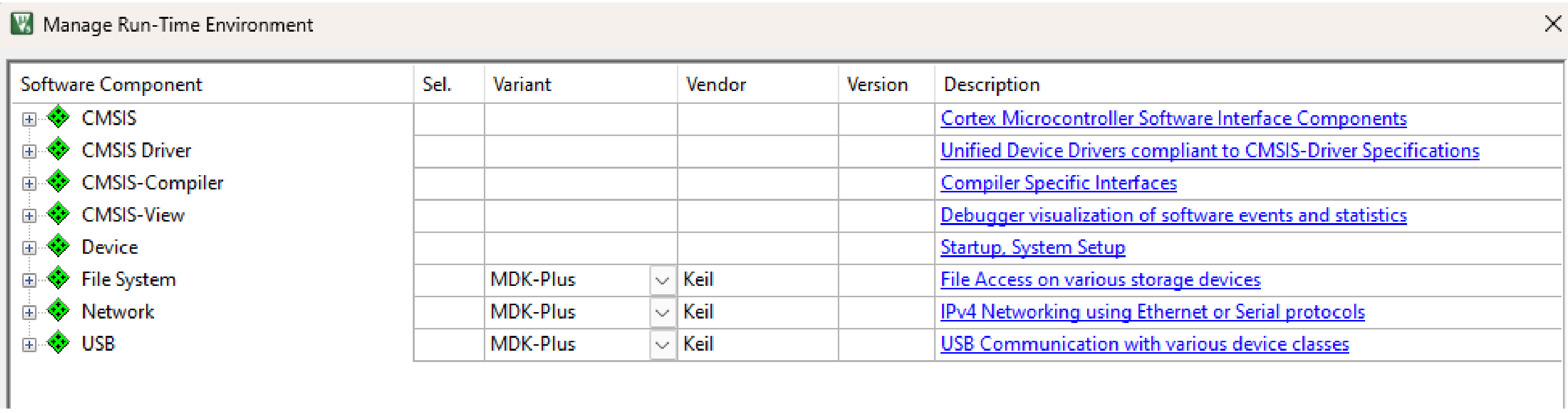
Destination Folder

C:\Users\Admin\AppData\Local\Arm\Packs

Click Next and You got this!

You downloaded full APM32F003 pack!

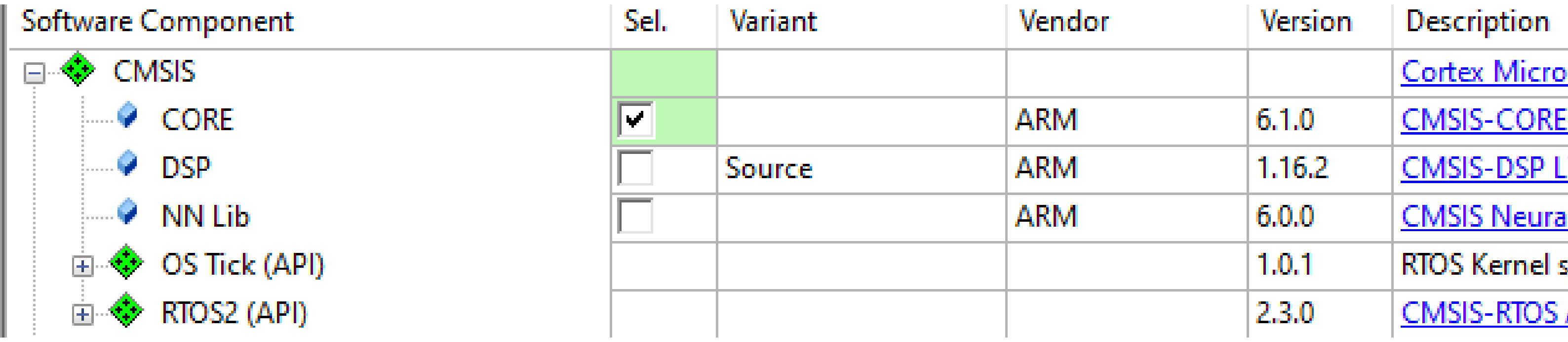
3.5 Select Packets to your project



Manage Run-Time Environment

Software Component	Sel.	Variant	Vendor	Version	Description
CMSIS					Cortex Microcontroller Software Interface Components
CMSIS Driver					Unified Device Drivers compliant to CMSIS-Driver Specifications
CMSIS-Compiler					Compiler Specific Interfaces
CMSIS-View					Debugger visualization of software events and statistics
Device					Startup, System Setup
File System		MDK-Plus	Keil		File Access on various storage devices
Network		MDK-Plus	Keil		IPv4 Networking using Ethernet or Serial protocols
USB		MDK-Plus	Keil		USB Communication with various device classes

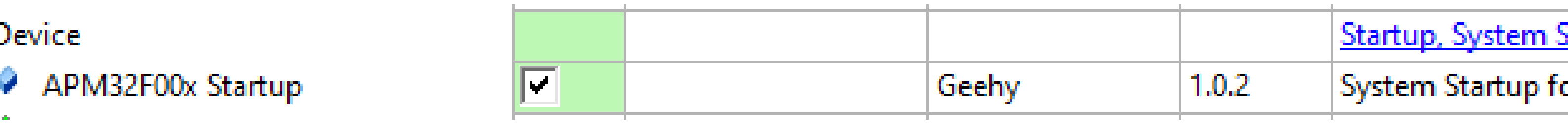
3.6 First, we need Cortex M0+ Core



Software Component	Sel.	Variant	Vendor	Version	Description
CMSIS					Cortex Micro
CORE	☑		ARM	6.1.0	CMSIS-CORE
DSP	□	Source	ARM	1.16.2	CMSIS-DSP L
NN Lib	□		ARM	6.0.0	CMSIS Neura
OS Tick (API)				1.0.1	RTOS Kernel s
RTOS2 (API)				2.3.0	CMSIS-RTOS

CMSIS -> CORE

3.7 Then, we need System Start up

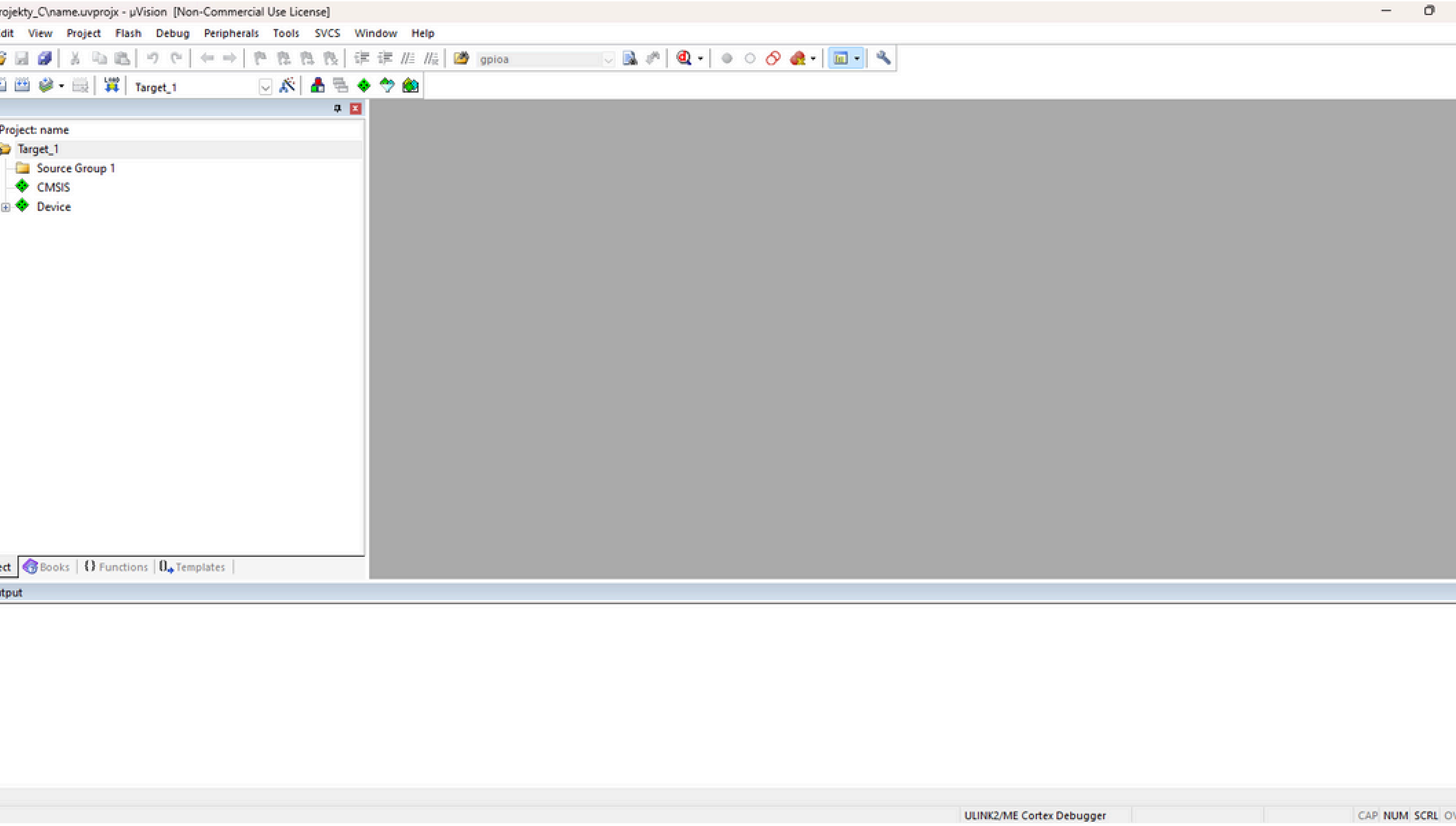


Device					Startup, System S
APM32F00x Startup	☑		Geehy	1.0.2	System Startup fo

Device -> APM32F00x Start up

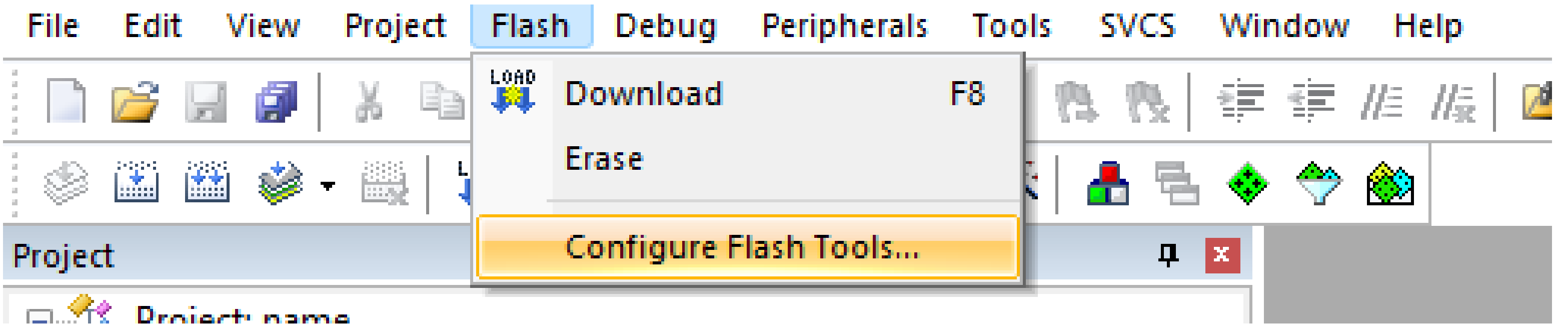
And Click “OK”

And Our Project is finally Created!

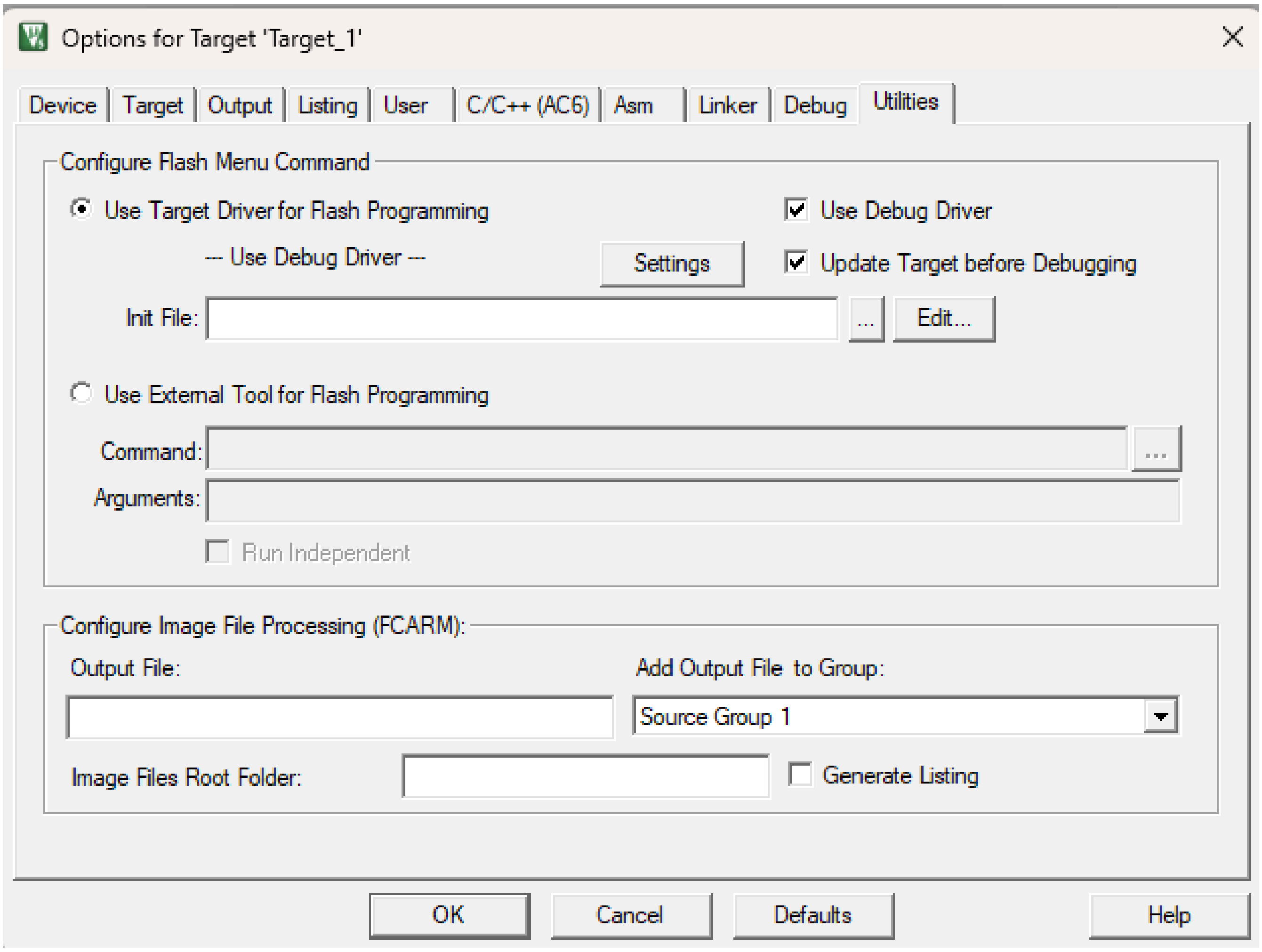


4. Configure Debbuger

Click “Flash” and then “Configure flash tools”



You should see a window like this



Configure Flash Menu Command

● Use Target Driver for Flash Programming

-- Use Debug Driver --

☑ Use Debug Driver

☑ Update Target before Debugging

Init File:

○ Use External Tool for Flash Programming

Command:

Arguments:

□ Run Independent

Configure Image File Processing (FCARM):

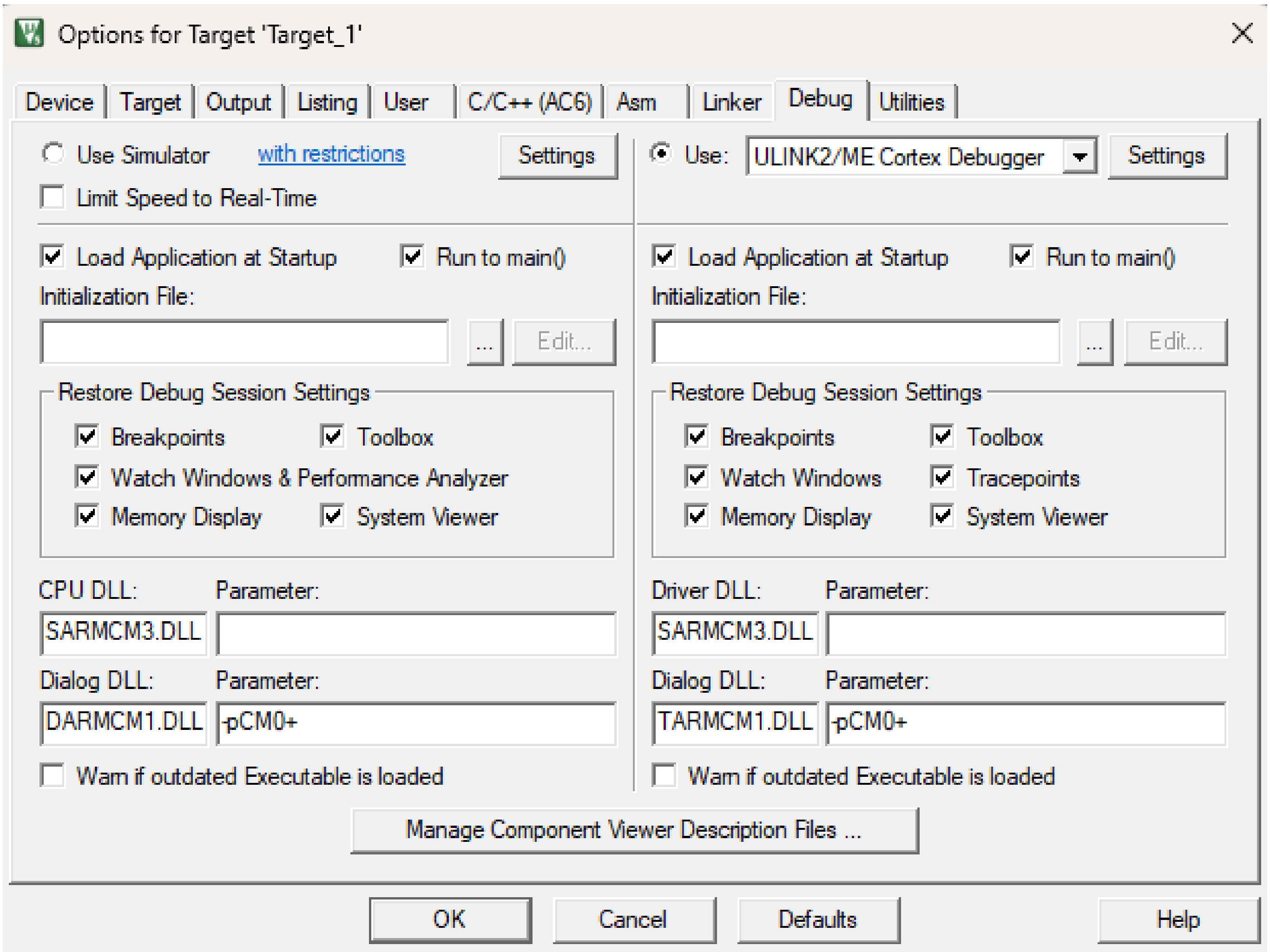
Output File:

Add Output File to Group: Source Group 1

Image Files Root Folder:

□ Generate Listing

Click “Debug” and you should see this:



Options for Target 'Target_1'

Device | Target | Output | Listing | User | C/C++ (AC6) | Asm | Linker | Debug | Utilities

○ Use Simulator with restrictions Settings

□ Limit Speed to Real-Time

☑ Load Application at Startup ☑ Run to main()

Initialization File:

... Edit...

Restore Debug Session Settings

☑ Breakpoints ☑ Toolbox

☑ Watch Windows & Performance Analyzer

☑ Memory Display ☑ System Viewer

CPU DLL: SARMCM3.DLL Parameter:

Dialog DLL: DARMCM1.DLL Parameter: -pCM0+

□ Warn if outdated Executable is loaded

● Use: ULINK2/ME Cortex Debugger Settings

☑ Load Application at Startup ☑ Run to main()

Initialization File:

... Edit...

Restore Debug Session Settings

☑ Breakpoints ☑ Toolbox

☑ Watch Windows ☑ Tracepoints

☑ Memory Display ☑ System Viewer

Driver DLL: SARMCM3.DLL Parameter:

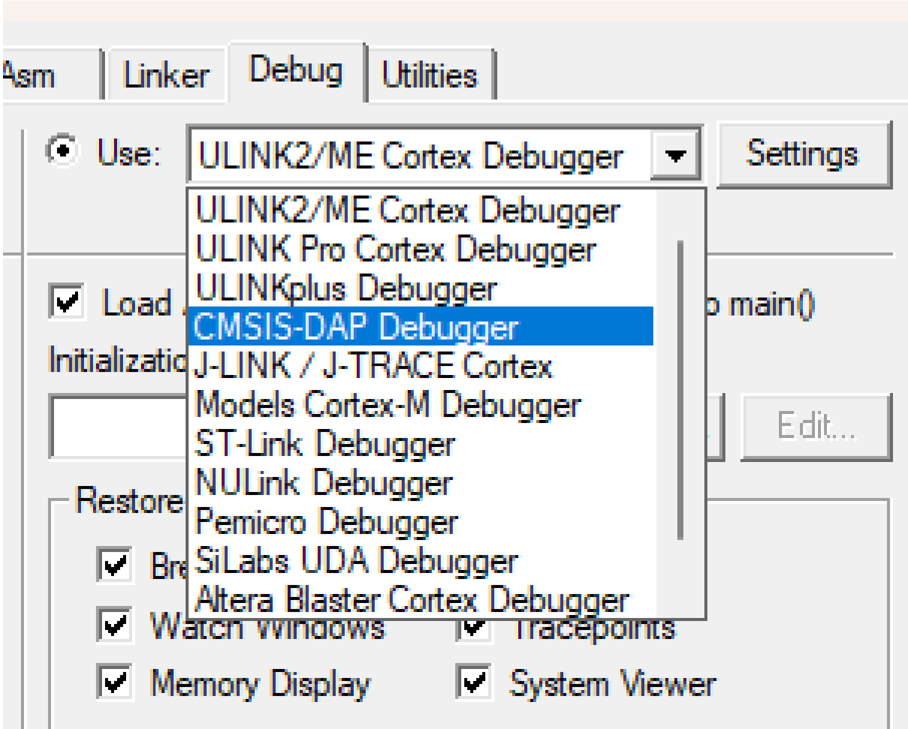
Dialog DLL: TARMCM1.DLL Parameter: -pCM0+

□ Warn if outdated Executable is loaded

Manage Component Viewer Description Files ...

OK Cancel Defaults Help

Select your Debbuger, in my case “CMSIS dap debbuger”



Asm | Linker | Debug | Utilities

● Use: ULINK2/ME Cortex Debugger Settings

☑ Load ... main()

Initializatio

Edit...

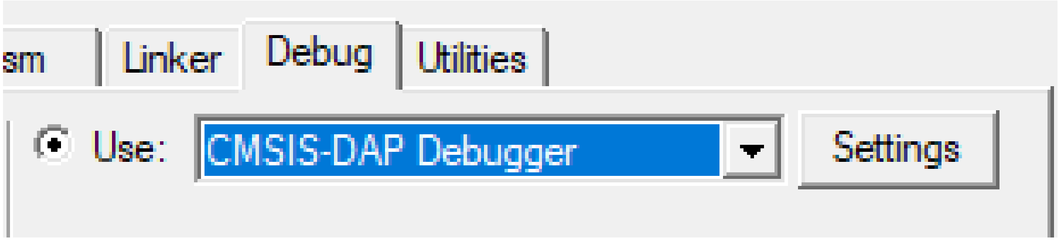
Restore

☑ Bre

☑ Watch Windows ☑ Tracepoints

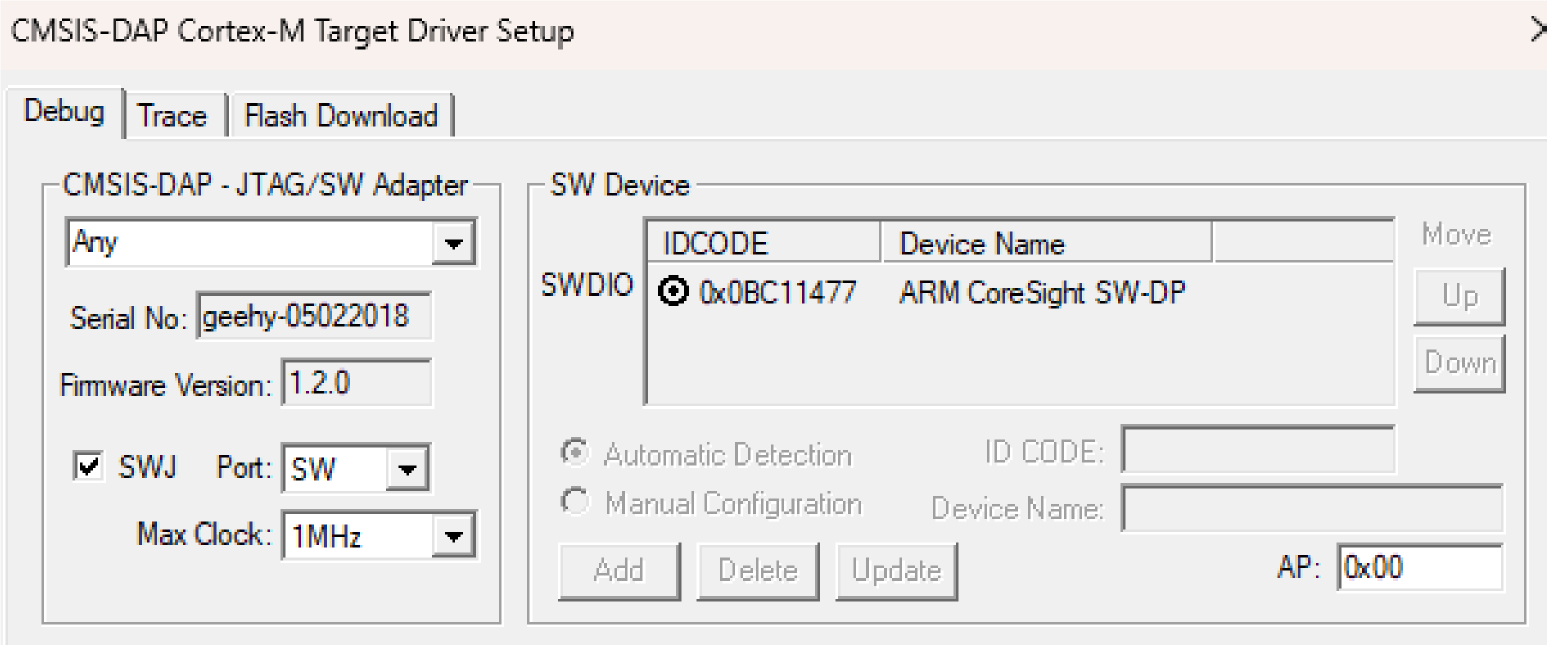
☑ Memory Display ☑ System Viewer

Click “Settings”



○ Use: CMSIS-DAP Debugger

And if you setup your debbuger corectly you should see this:



CMSIS-DAP Cortex-M Target Driver Setup

CMSIS-DAP - JTAG/SW Adapter

Any

Serial No: geehy-05022018

Firmware Version: 1.2.0

☑ SWJ Port: SW

Max Clock: 1MHz

SW Device

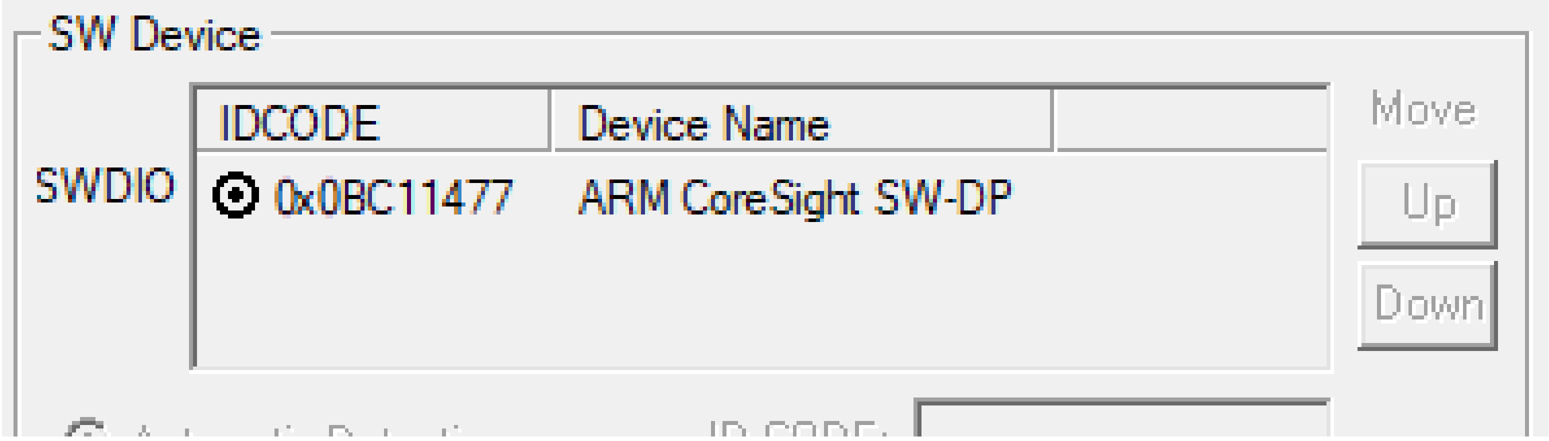
	IDCODE	Device Name
SWDIO	⊙ 0x0BC11477	ARM CoreSight SW-DP

● Automatic Detection ID CODE:

○ Manual Configuration Device Name:

AP: 0x00

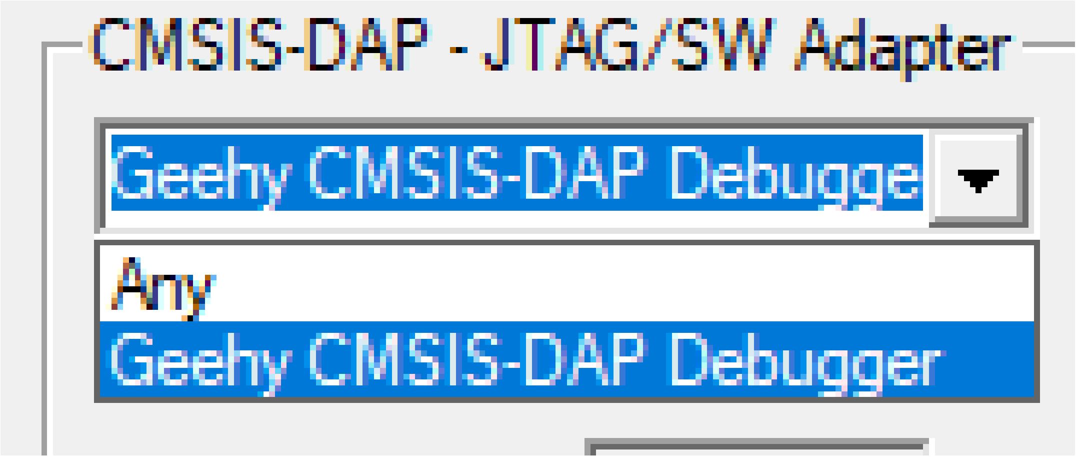
I mean THIS:



SW Device

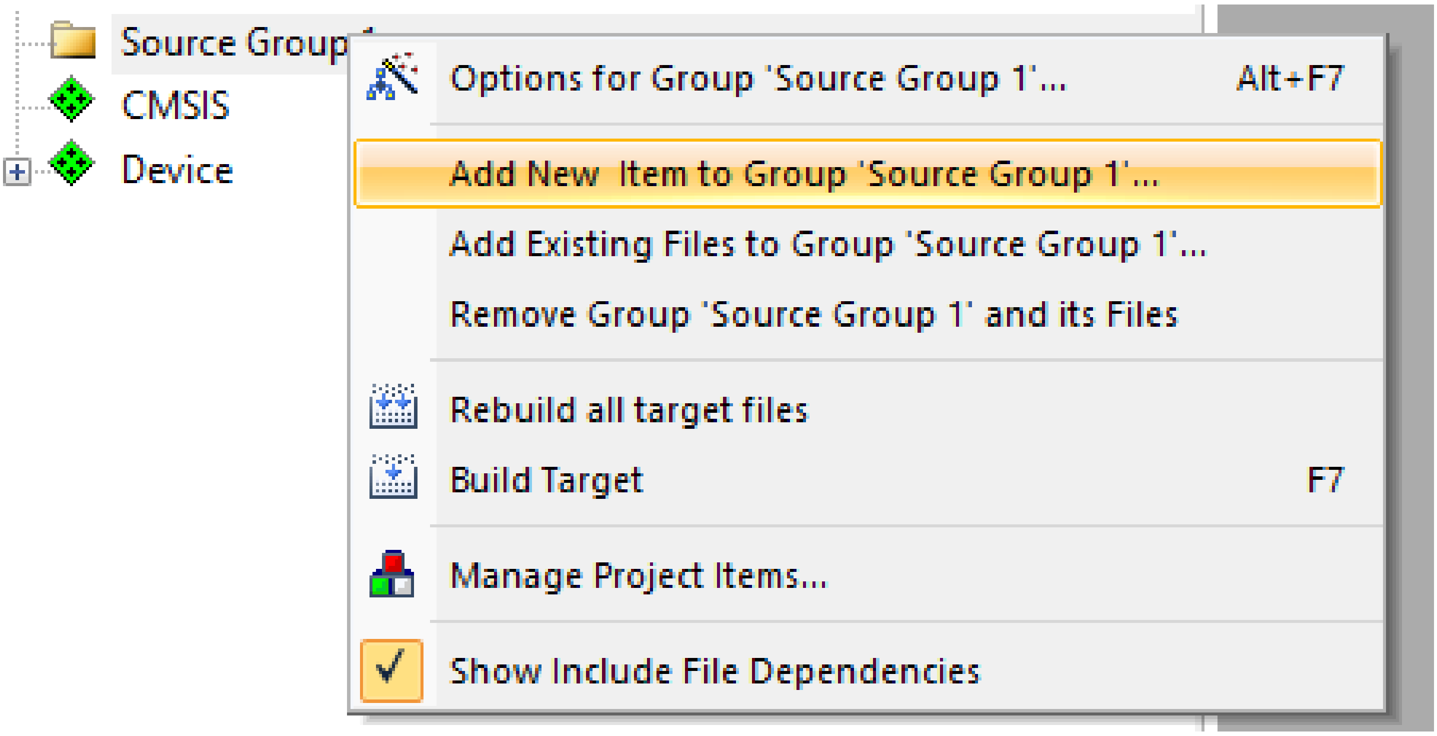
	IDCODE	Device Name
SWDIO	⊙ 0x0BC11477	ARM CoreSight SW-DP

AND THIS:



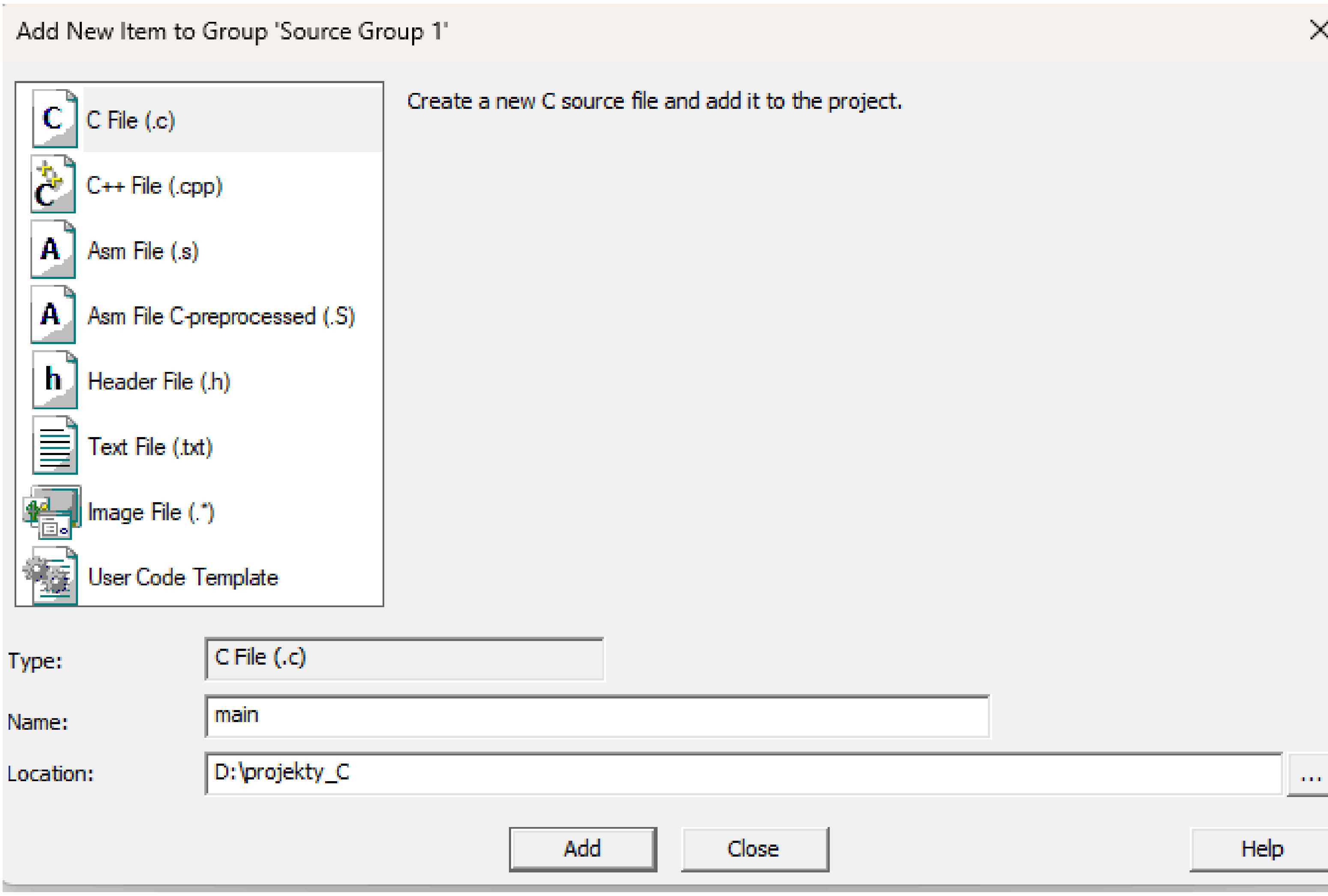
5. Lets add some files

Right click on Source Group 1


Source Group 1
CMSIS
Device
Options for Group 'Source Group 1'... Alt+F7
Add New Item to Group 'Source Group 1'...
Add Existing Files to Group 'Source Group 1'...
Remove Group 'Source Group 1' and its Files
Rebuild all target files
Build Target F7
Manage Project Items...
Show Include File Dependencies

Click “Add New item to group”

Name it “Main” and choose “C file”



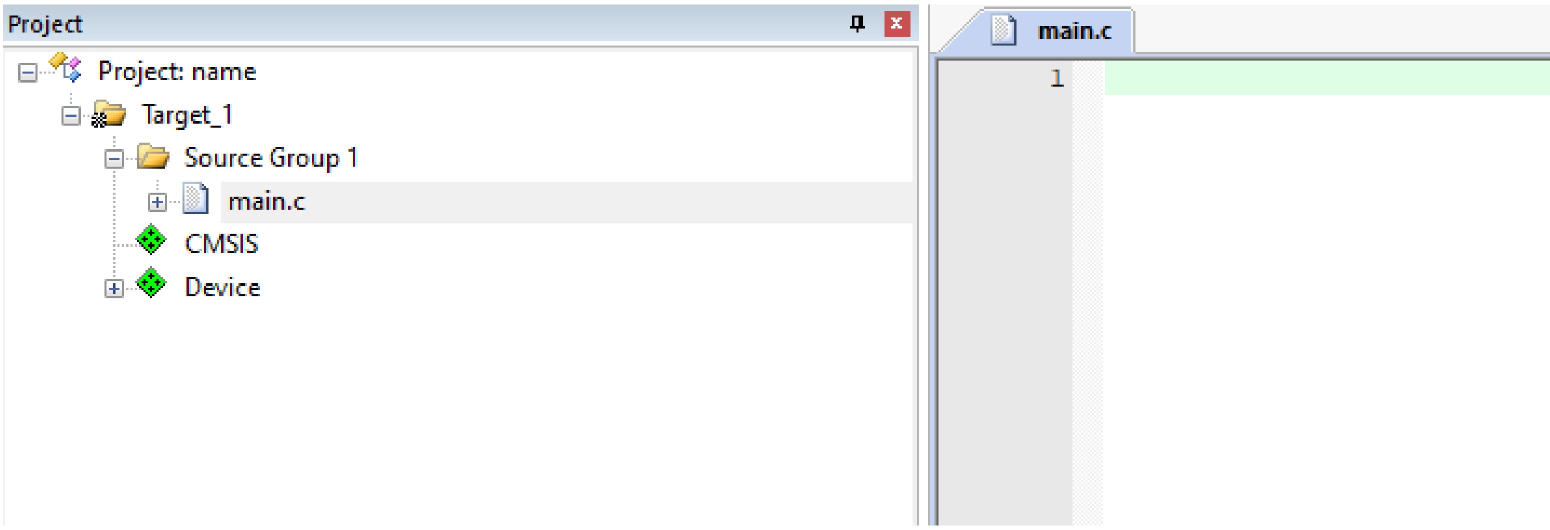
Create a new C source file and add it to the project.

Type: C File (.c)

Name: main

Location: D:\projekty_C

You should see empy file

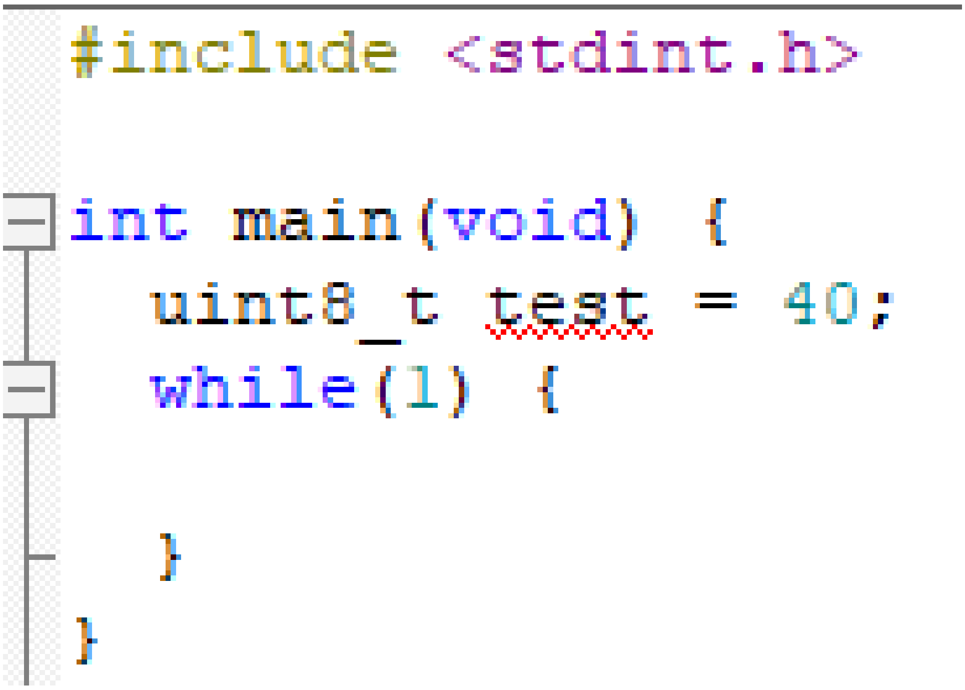


Okay, lets write some code that do nothing, just to verify if MCU is working!

5. Write main and verify MCU

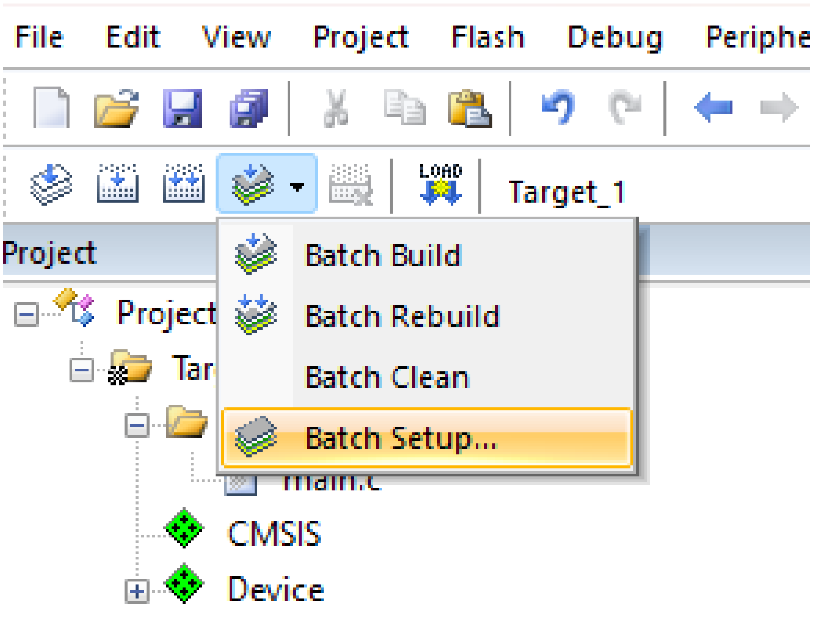
Write this code

WARNING! this code do nothing, really..

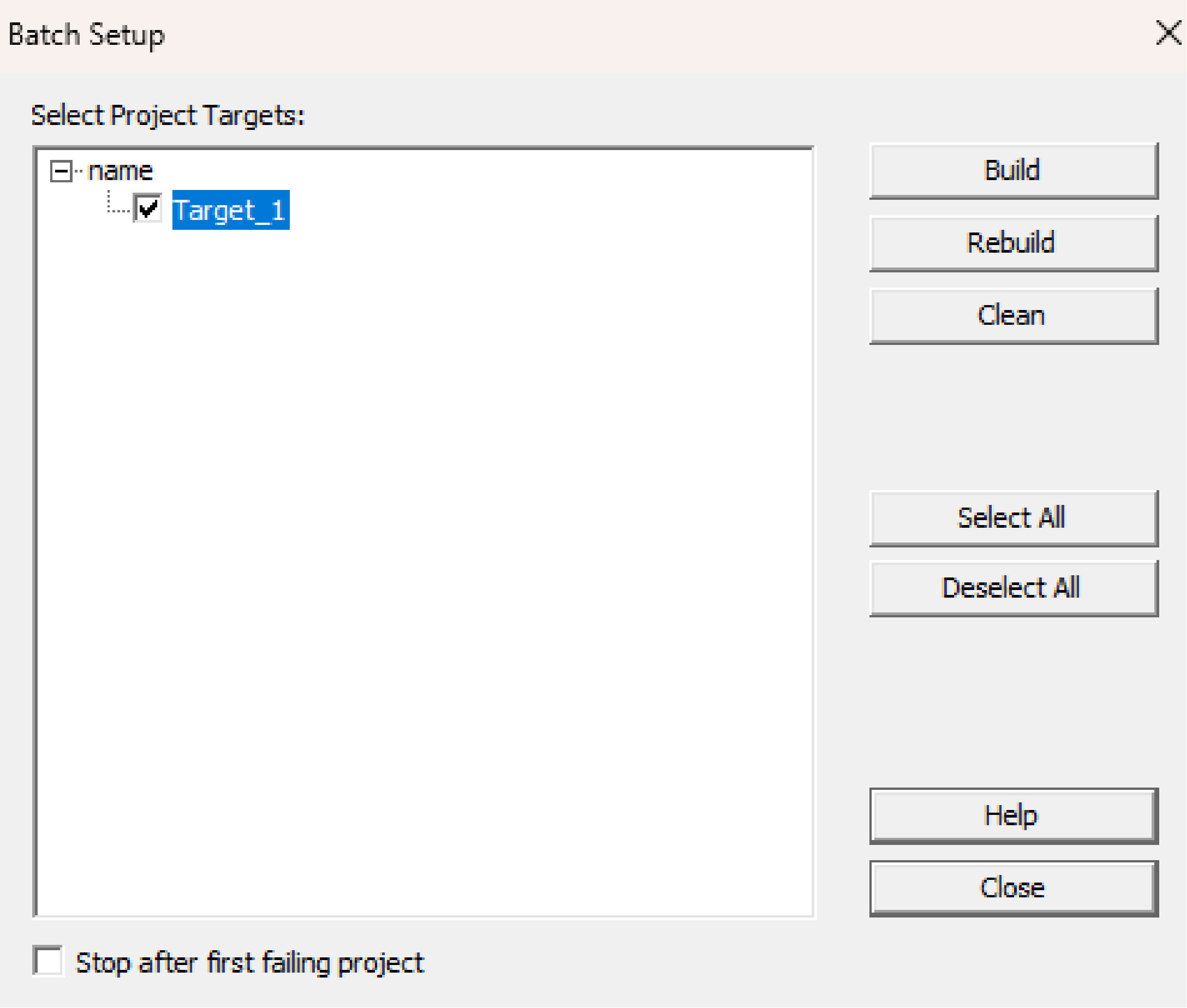


```
#include <stdint.h>

int main(void) {
   uint8_t test = 40;
   while(1) {


   }
}
```

Okay, now click Batch Setup



Okay, now select “Target_1”


Batch Setup

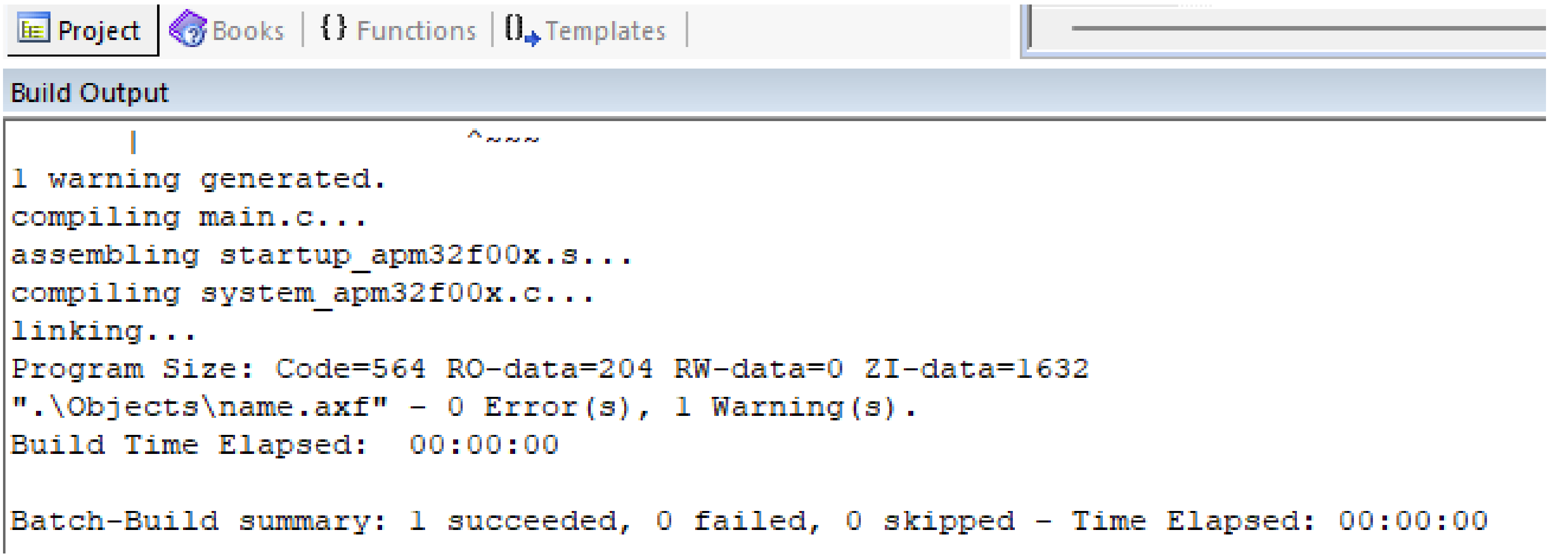
Select Project Targets:

⊟ name
☑ Target_1

☐ Stop after first failing project

Then Click “Rebuild”

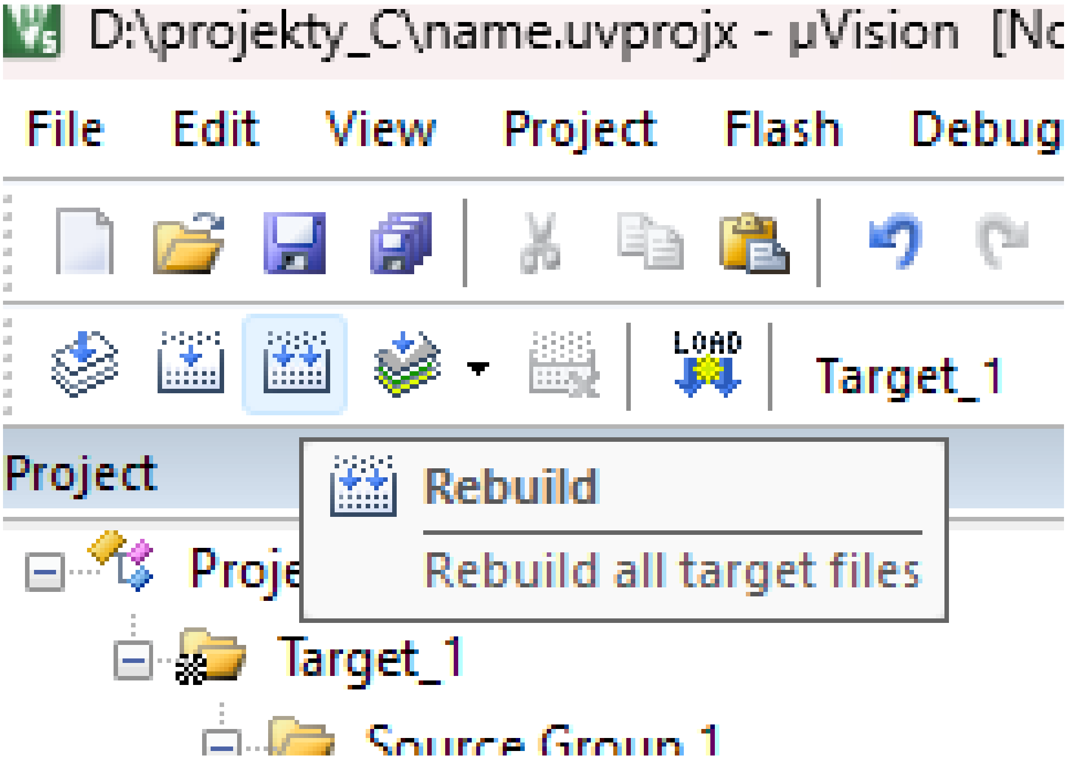
In Console, you should see Something like this:



```
      |                      ^~~~
1 warning generated.
compiling main.c...
assembling startup_apm32f00x.s...
compiling system_apm32f00x.c...
linking...
Program Size: Code=564 RO-data=204 RW-data=0 ZI-data=1632
".\Objects\name.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed:  00:00:00

Batch-Build summary: 1 succeeded, 0 failed, 0 skipped - Time Elapsed: 00:00:00
```

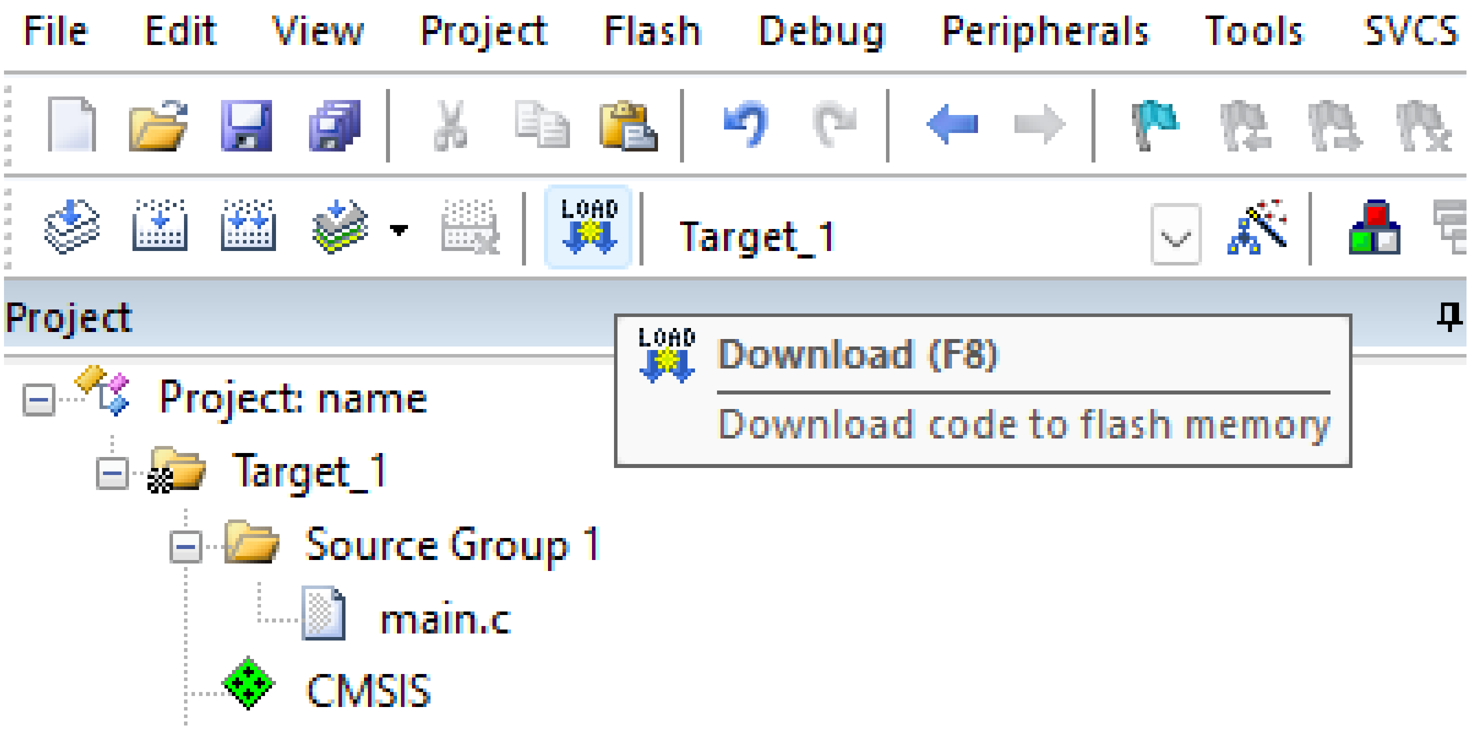
Which means, its okay, and now, lets build project!

Click “Rebuild”

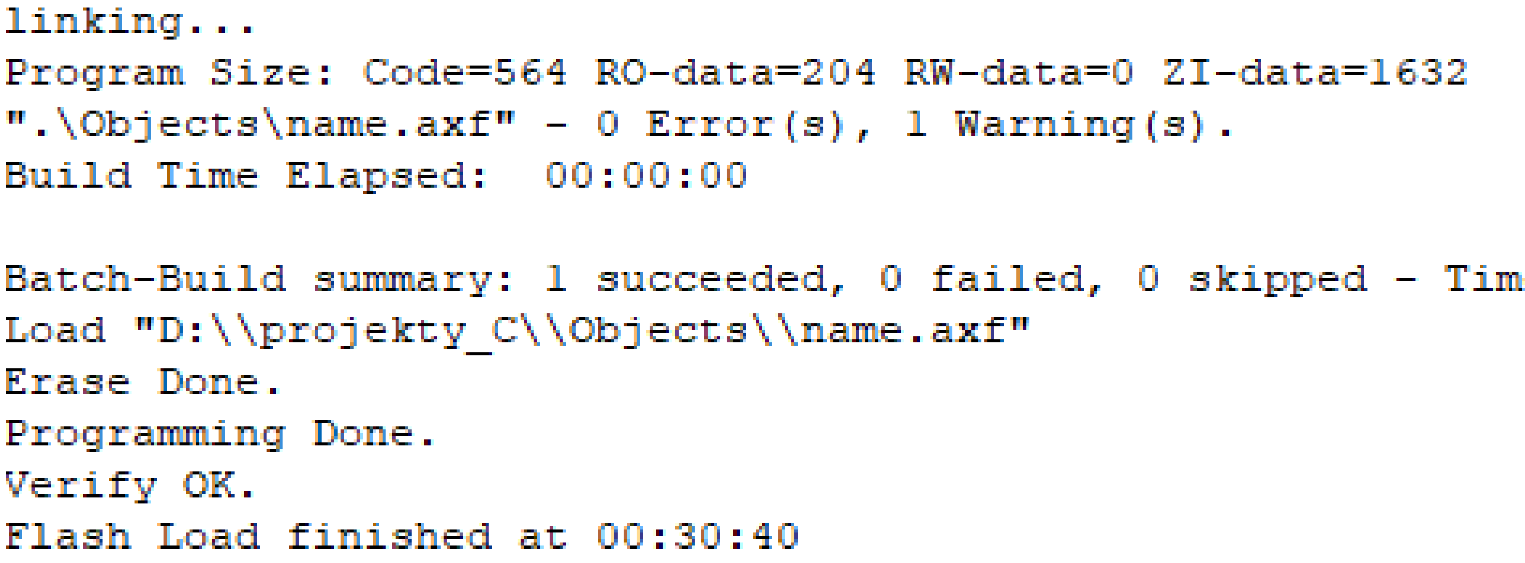


AND NOW, load our program to MCU, finally!

Click “Load”



YOU REALLYYY SHOULD SEE THIS:



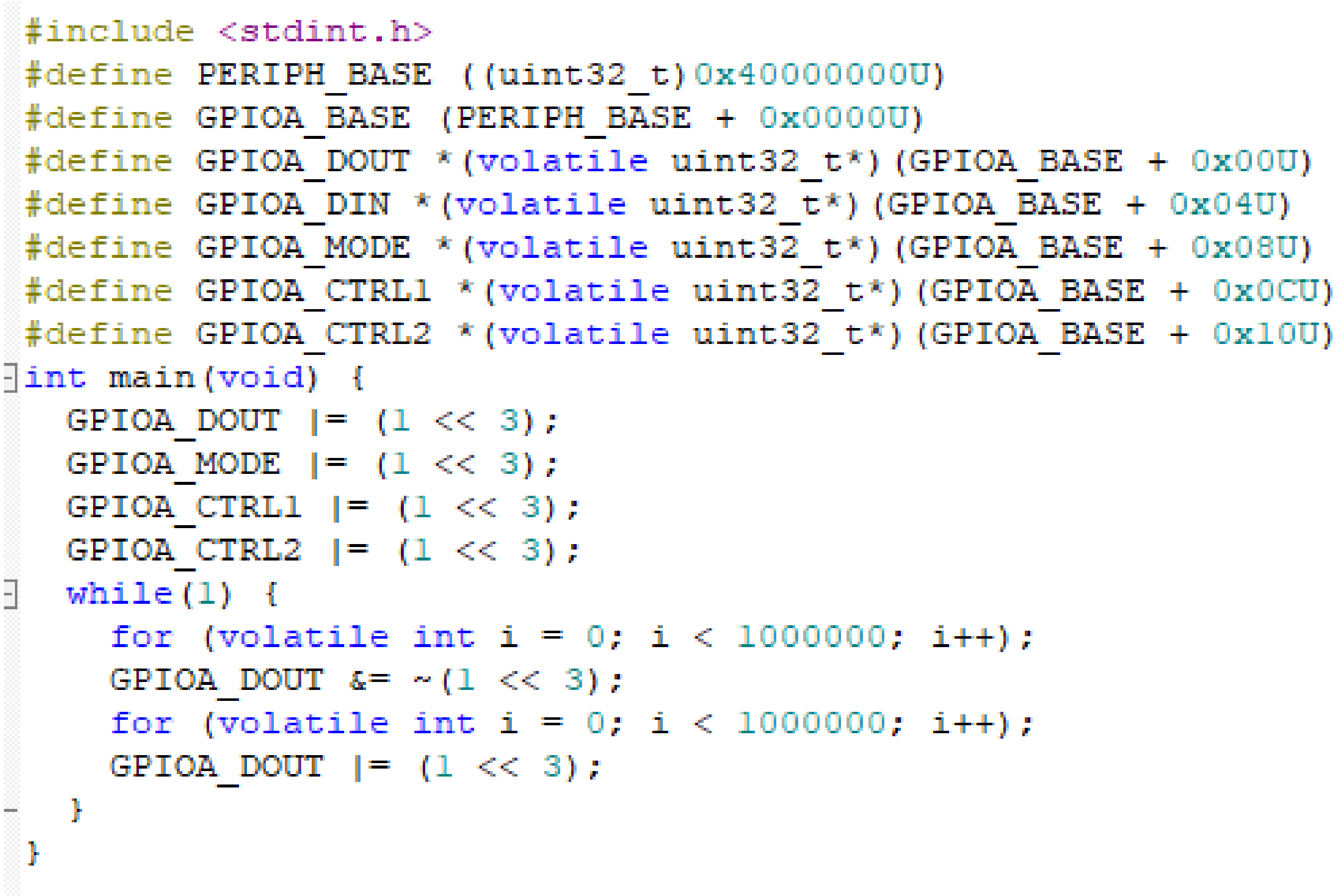
```
linking...
Program Size: Code=564 RO-data=204 RW-data=0 ZI-data=1632
".\Objects\name.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed:  00:00:00

Batch-Build summary: 1 succeeded, 0 failed, 0 skipped - Tim
Load "D:\\projekty_C\\Objects\\name.axf"
Erase Done.
Programming Done.
Verify OK.
Flash Load finished at 00:30:40
```

This Mean, Your MCU is working, and you finally load your program to APM32

6. Blink LED

I will not explain how this code work, its Bare Metal and if I were to write how everything works, it would take me another 20 pages

Just copy this Code and Paste to your main.c and compile

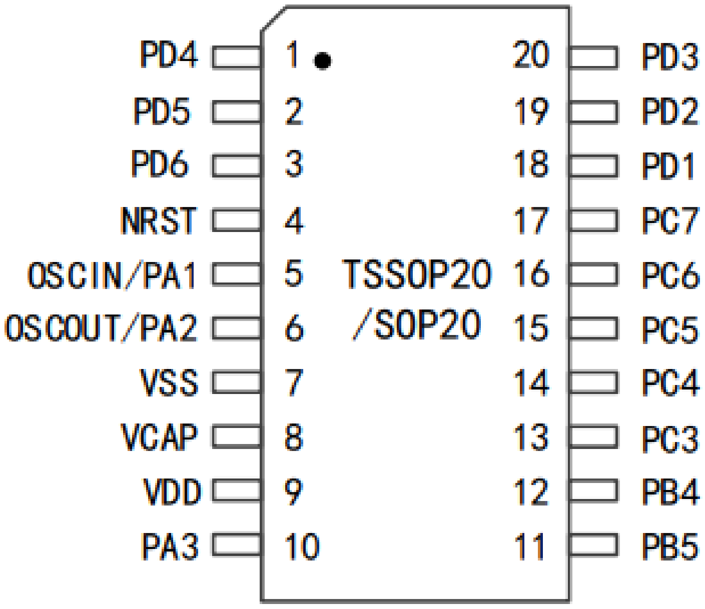


```c
#include <stdint.h>
#define PERIPH_BASE ((uint32_t)0x40000000U)
#define GPIOA_BASE (PERIPH_BASE + 0x0000U)
#define GPIOA_DOUT *(volatile uint32_t*)(GPIOA_BASE + 0x00U)
#define GPIOA_DIN *(volatile uint32_t*)(GPIOA_BASE + 0x04U)
#define GPIOA_MODE *(volatile uint32_t*)(GPIOA_BASE + 0x08U)
#define GPIOA_CTRL1 *(volatile uint32_t*)(GPIOA_BASE + 0x0CU)
#define GPIOA_CTRL2 *(volatile uint32_t*)(GPIOA_BASE + 0x10U)
int main(void) {
  GPIOA_DOUT |= (1 << 3);
  GPIOA_MODE |= (1 << 3);
  GPIOA_CTRL1 |= (1 << 3);
  GPIOA_CTRL2 |= (1 << 3);
  while(1) {
    for (volatile int i = 0; i < 1000000; i++);
    GPIOA_DOUT &= ~(1 << 3);
    for (volatile int i = 0; i < 1000000; i++);
    GPIOA_DOUT |= (1 << 3);
  }
}
```

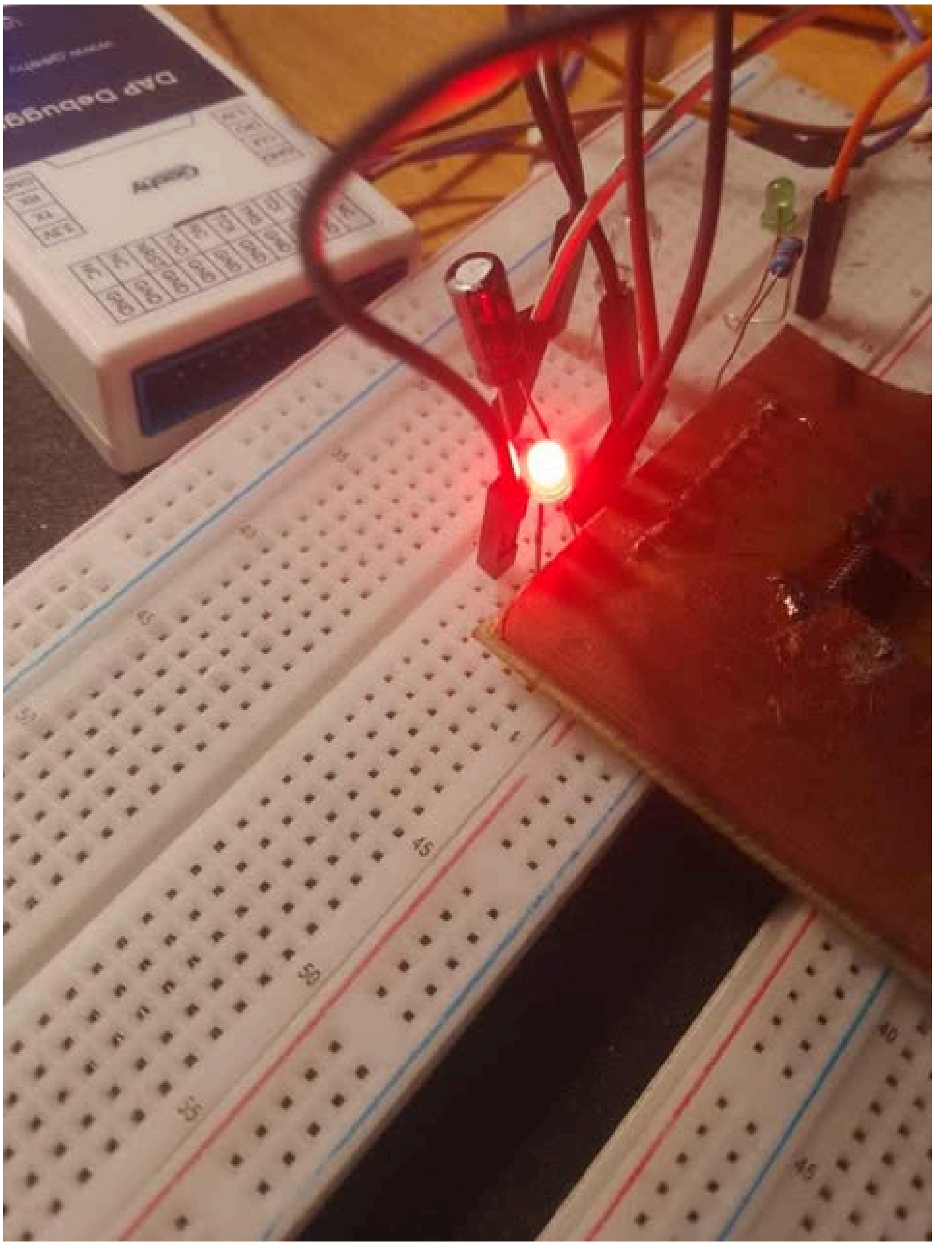
Copy paste code:

```
#include <stdint.h>
#define PERIPH_BASE ((uint32_t)0x40000000U)
#define GPIOA_BASE (PERIPH_BASE + 0x0000U)
#define GPIOA_DOUT *(volatile uint32_t*)(GPIOA_BASE + 0x00U)
#define GPIOA_DIN *(volatile uint32_t*)(GPIOA_BASE + 0x04U)
#define GPIOA_MODE *(volatile uint32_t*)(GPIOA_BASE + 0x08U)
#define GPIOA_CTRL1 *(volatile uint32_t*)(GPIOA_BASE + 0x0CU)
#define GPIOA_CTRL2 *(volatile uint32_t*)(GPIOA_BASE + 0x10U)
int main(void) {
 GPIOA_DOUT |= (1 << 3);
 GPIOA_MODE |= (1 << 3);
 GPIOA_CTRL1 |= (1 << 3);
 GPIOA_CTRL2 |= (1 << 3);
 while(1) {
  for (volatile int i = 0; i < 1000000; i++);
  GPIOA_DOUT &= ~(1 << 3);
  for (volatile int i = 0; i < 1000000; i++);
  GPIOA_DOUT |= (1 << 3);
 }
}
```

Connect LED to PA3 with resistor



The result:



So, this is the end of my tutorial.
My next target is LPC 2101, i dont know when, bcs im so busy, i create this PDF at 1AM.
date: 02.03.2025
Creator: Imeldushiii
https://github.com/Imeldushiii