Assembly Programming with nRF52840 (1)

한동대학교

마이크로프로세서응용

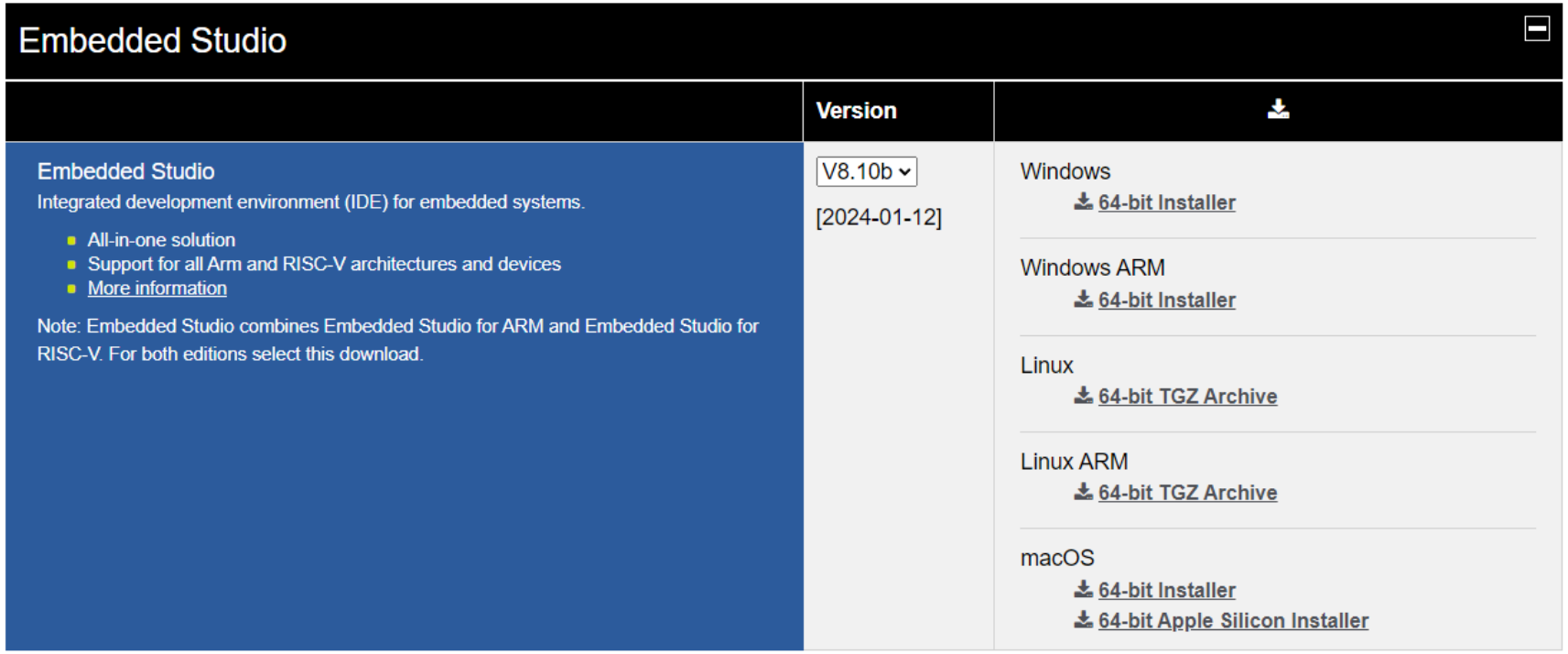
Agenda

1. **Install SEGGER Embedded Studio**
2. **Basic setting**
3. **GPIO**
4. **Lab-1**
5. **Lab-2**
6. **Lab-3**
7. **Lab-4**
8. **Assignment**

SEGGER Embedded Studio installation

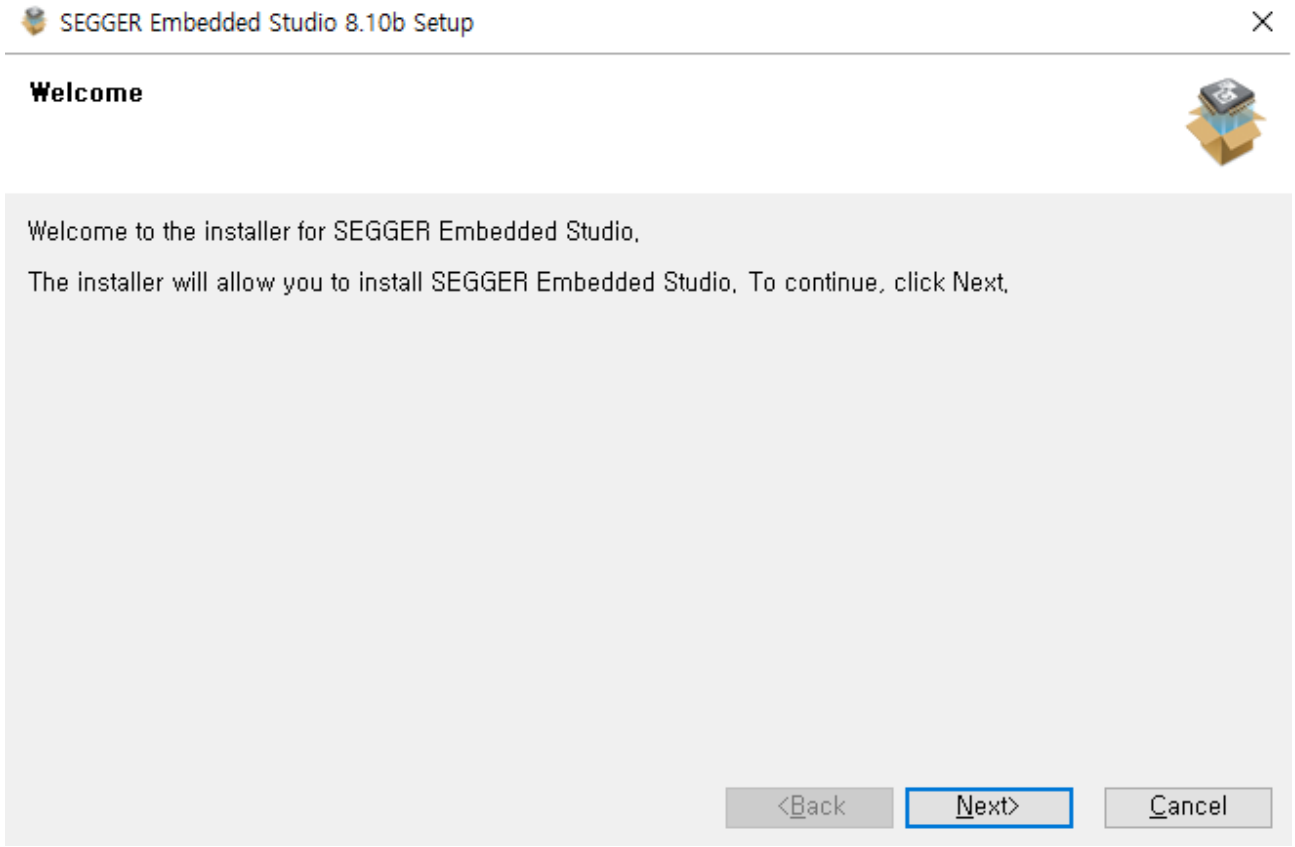
- Here you can download installer which fits for your OS

https://www.segger.com/downloads/embedded-studio/



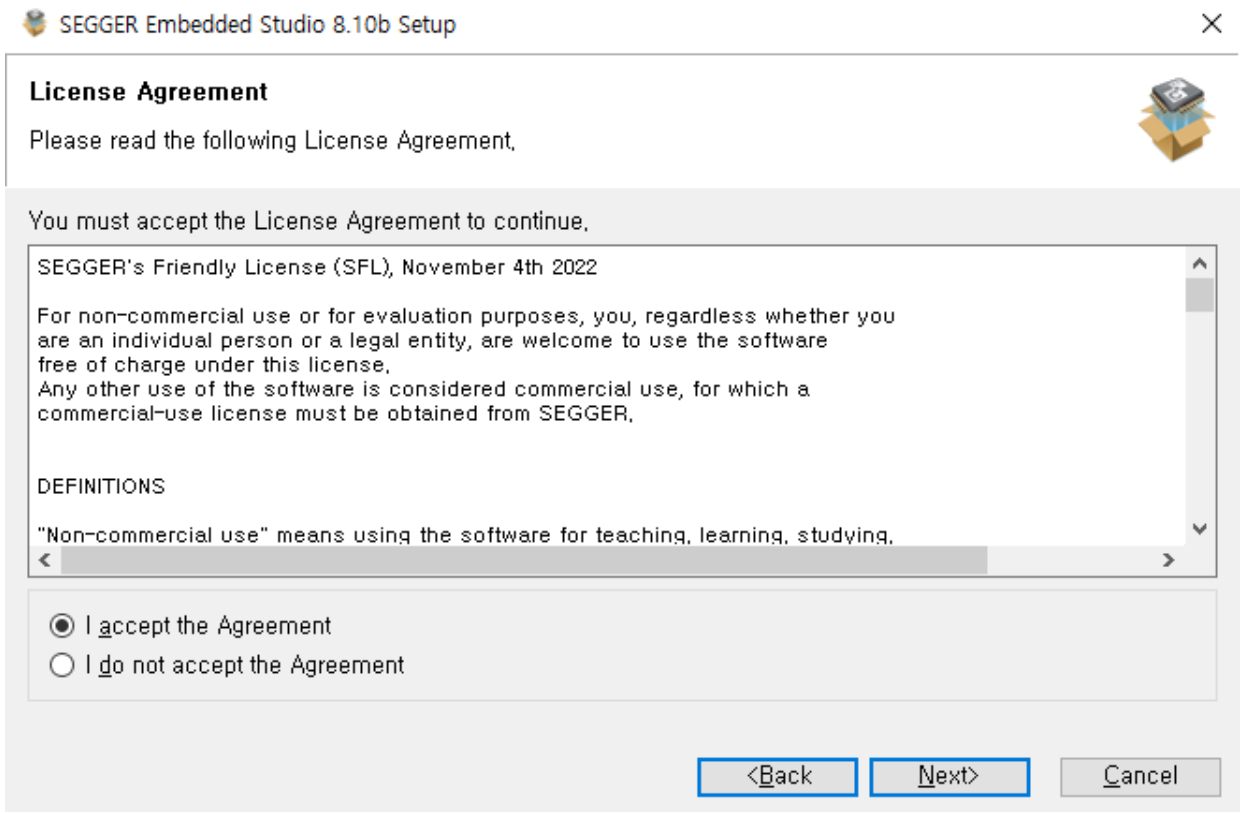
Segger Embedded Studio installation

- ‘Next’



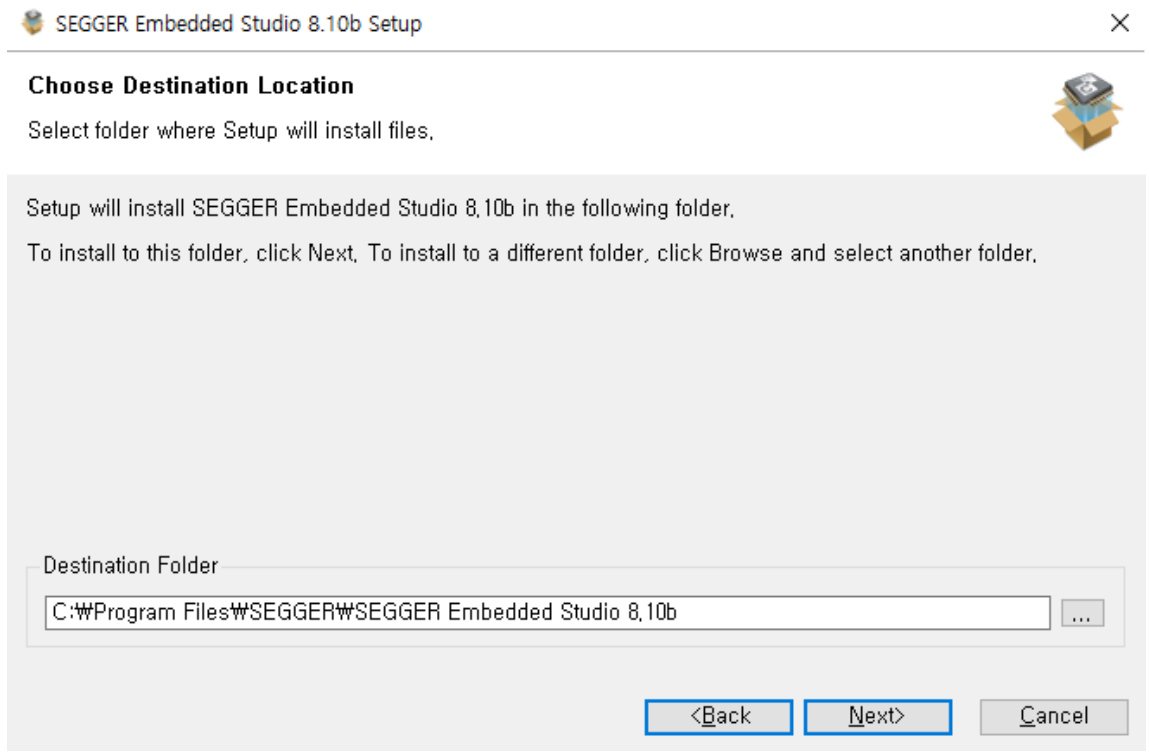
Segger Embedded Studio installation

- ‘Accept’ -> ‘Next’



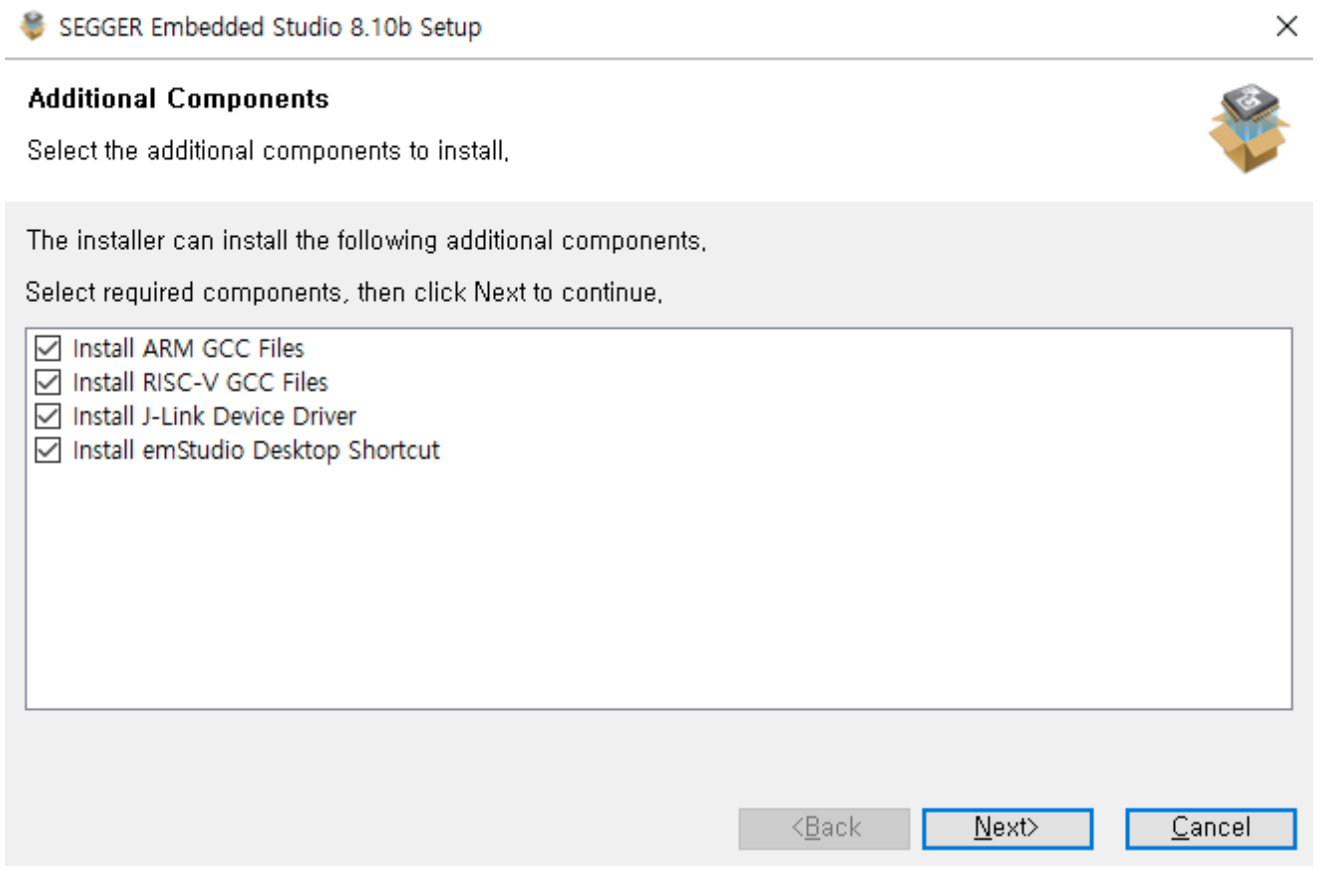
Segger Embedded Studio installation

- ‘Next’



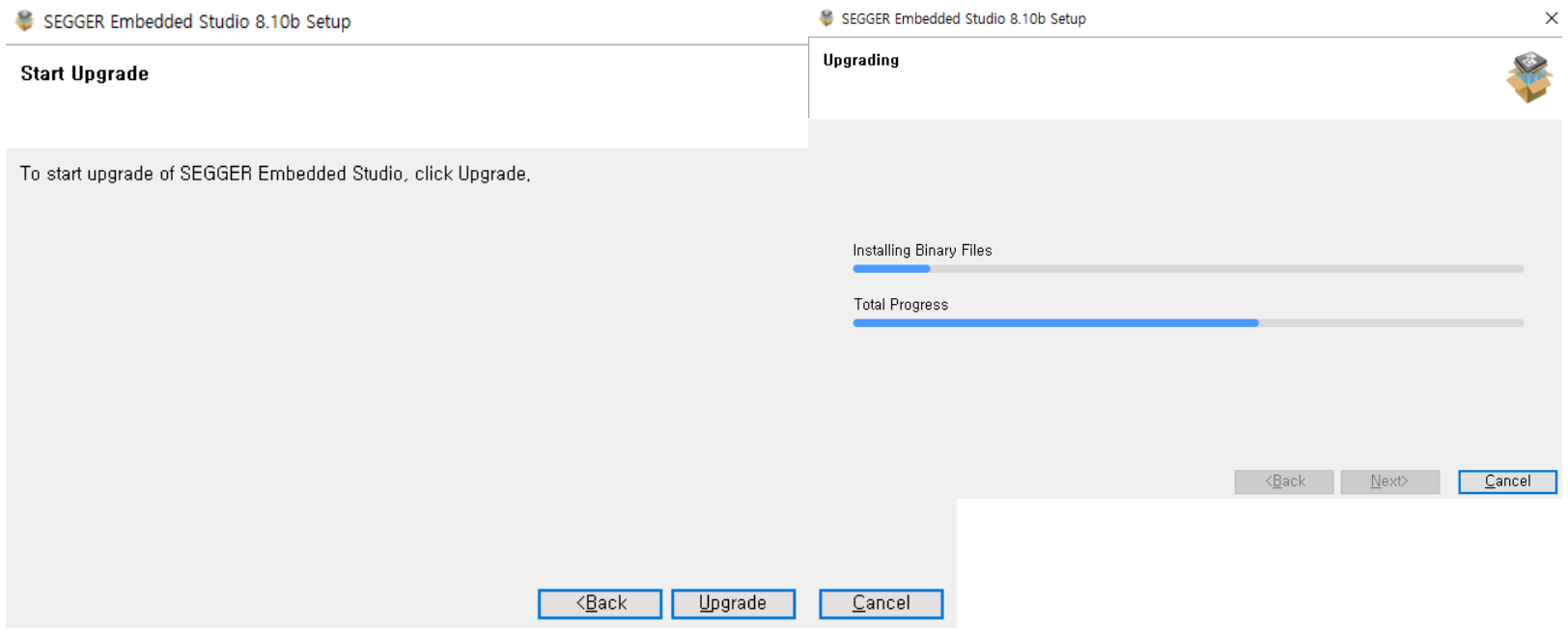
Segger Embedded Studio installation

- ‘Next’



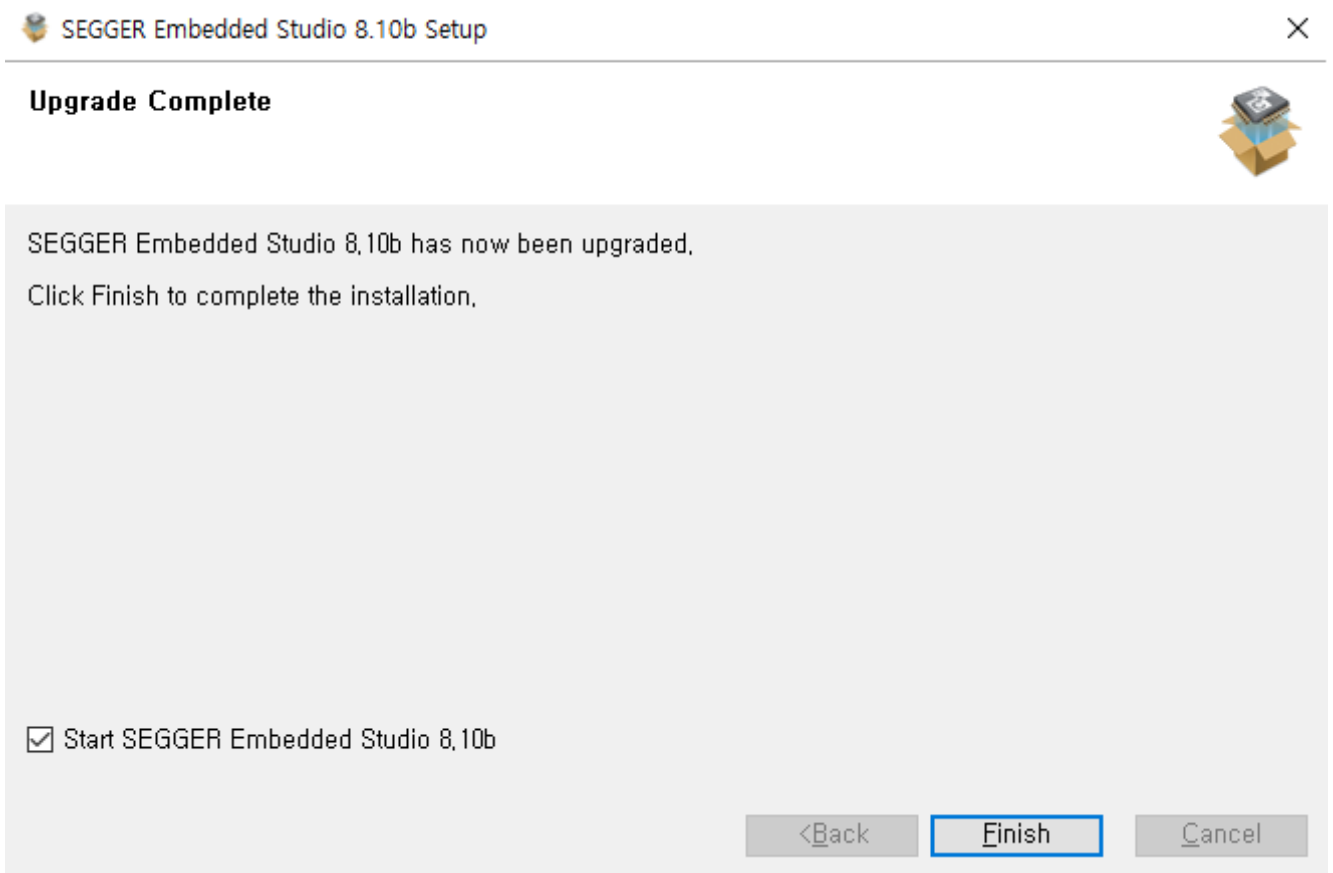
Segger Embedded Studio installation

- ‘Upgrade’



Segger Embedded Studio installation

- ‘Finish’

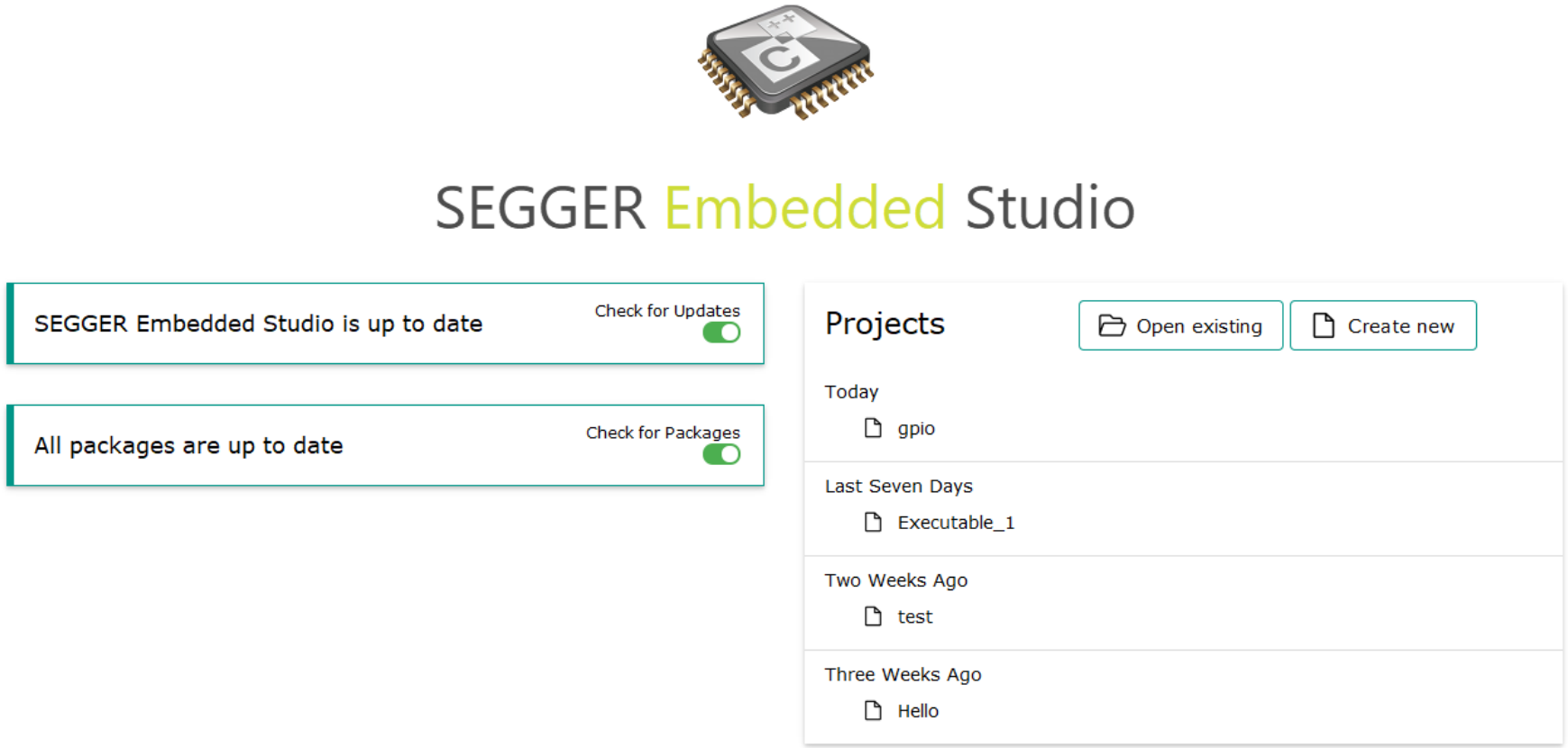


Agenda

1. Install SEGGER Embedded Studio
2. **Basic Setting**
3. GPIO
4. Lab-1
5. Lab-2
6. Lab-3
7. Assignment

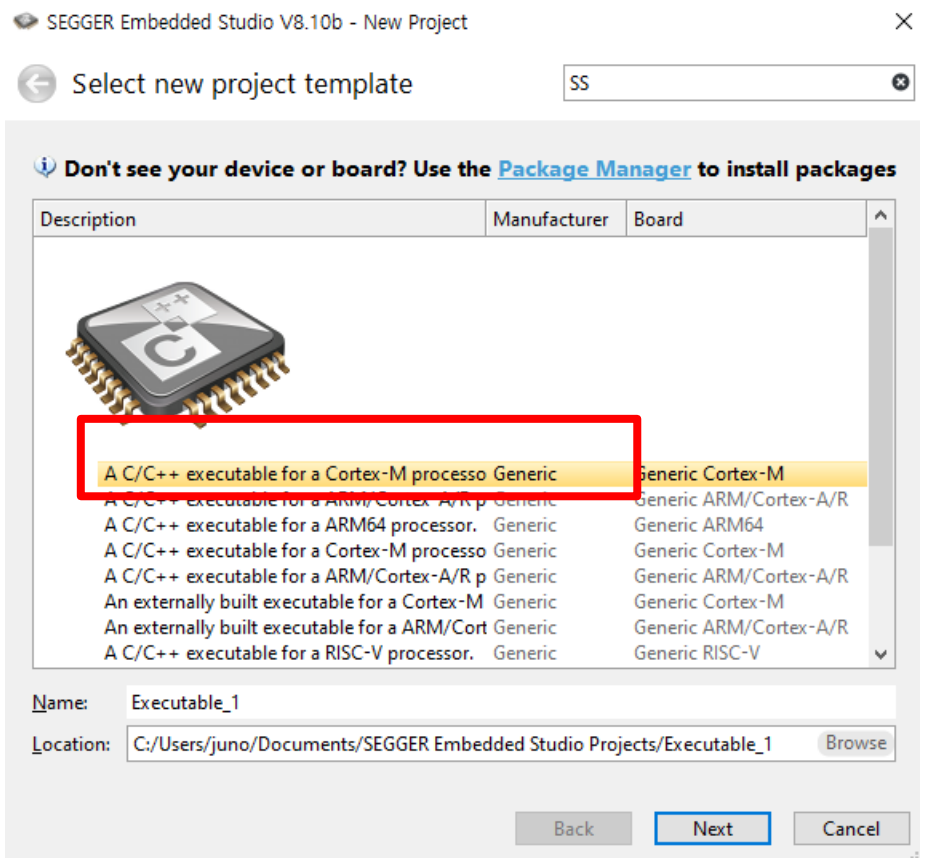
Create project

- ‘Create new’



Create project

- ‘A C/C++ executable for a Cortex-M processor.’ -> ‘Next’

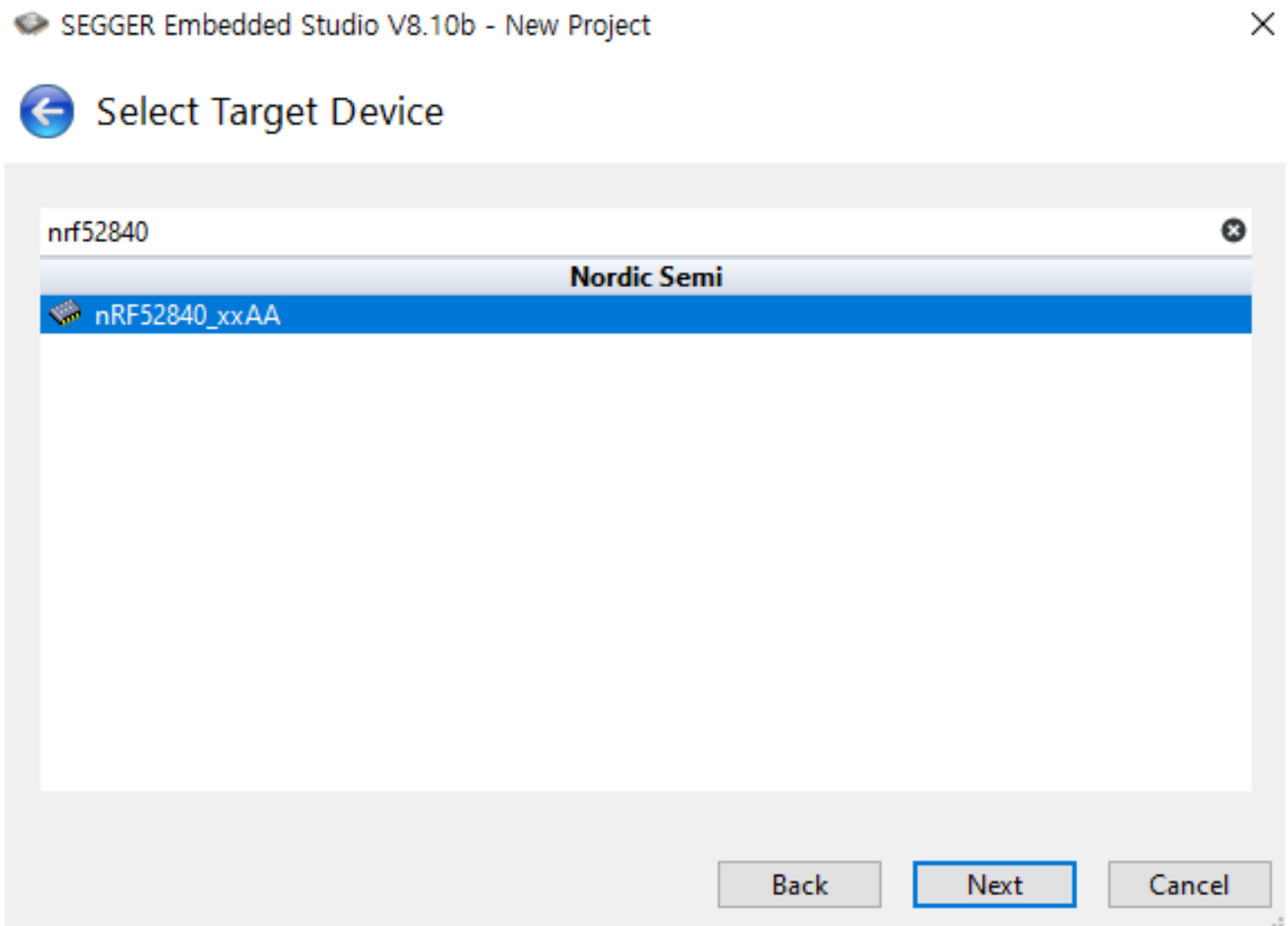


nRF52840 product specification

- ARM® Cortex®-M4 32-bit processor with FPU, 64 MHz
  - 212 EEMBC CoreMark score running from flash memory
  - 52 µA/MHz running from flash memory
  - Watchpoint and trace debug modules (DWT, ETM and ITM)
  - Serial wire debug (SWD)

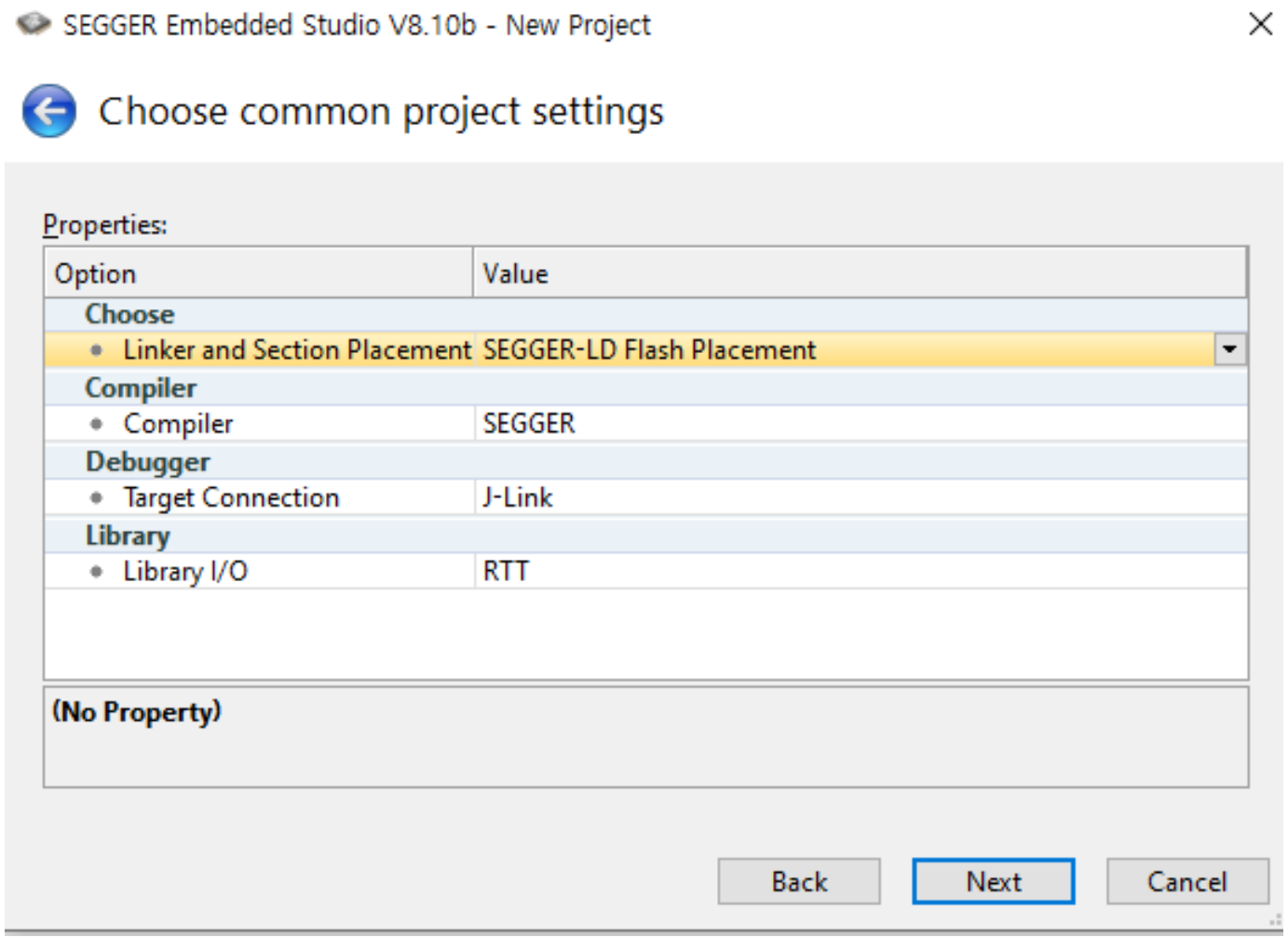
Create project

- ‘Search’ -> type ‘nrf52840’ -> select -> Next



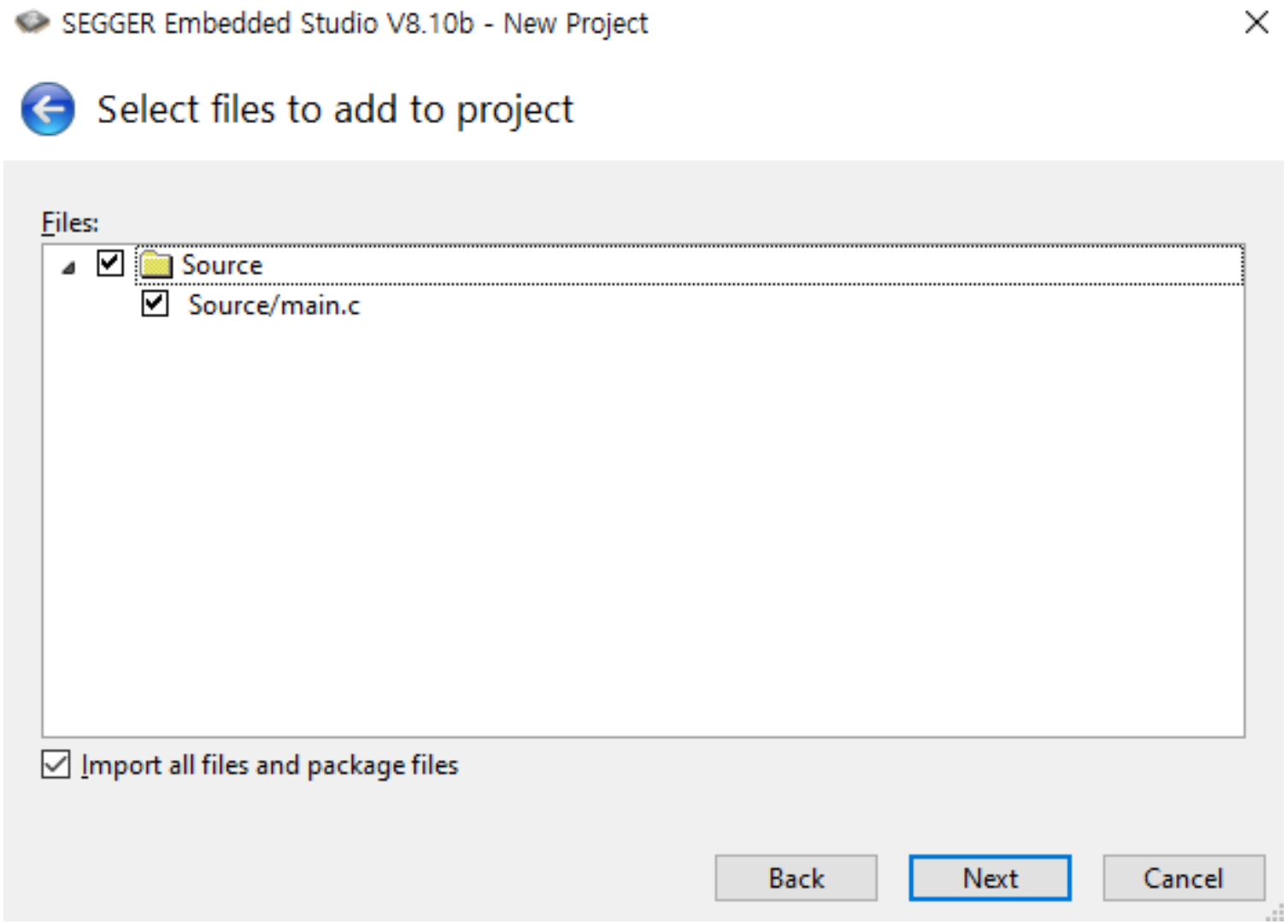
Create project

- ‘Next’ (you don’t need to change options)



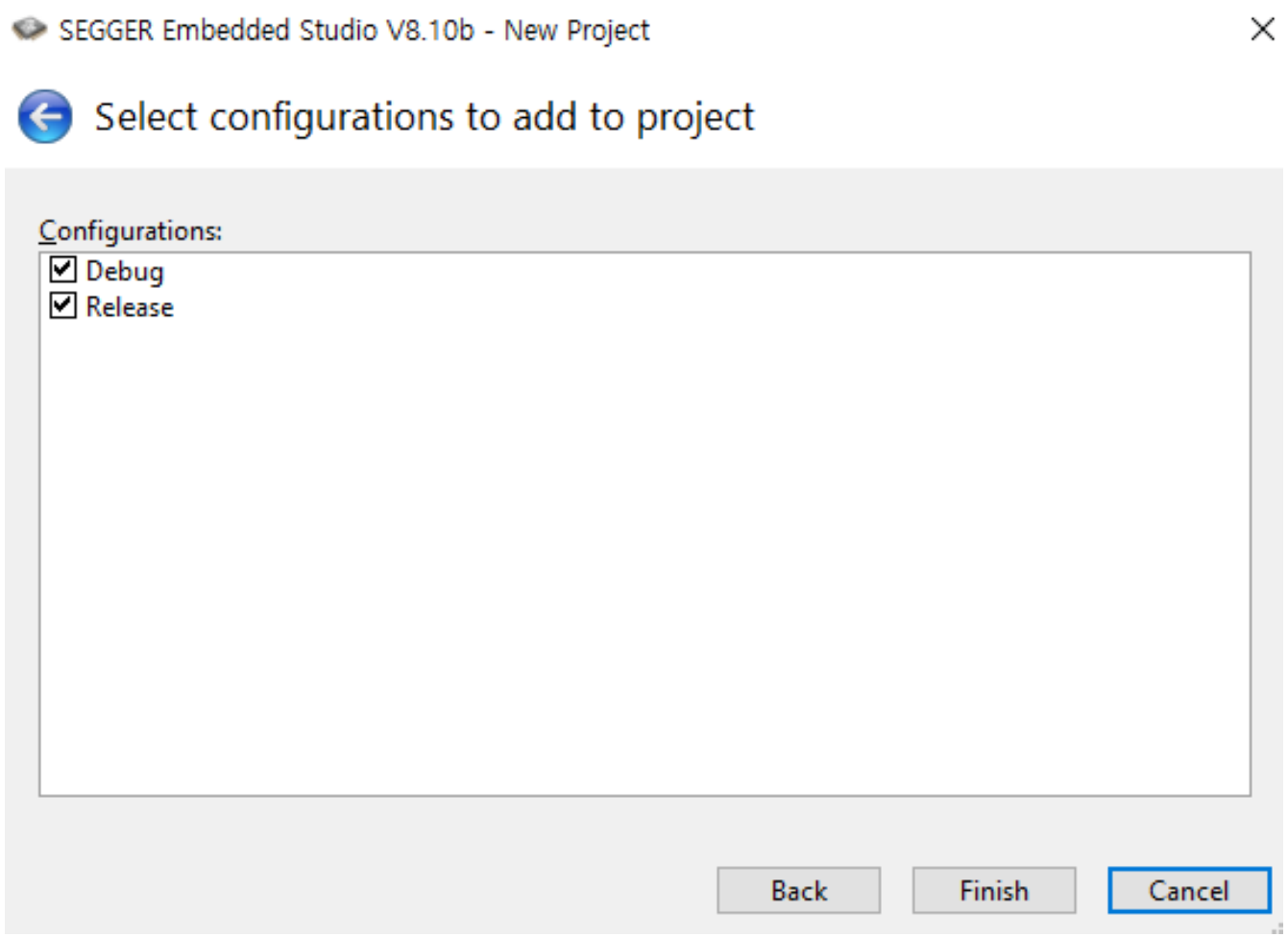
Create project

- ‘Next’



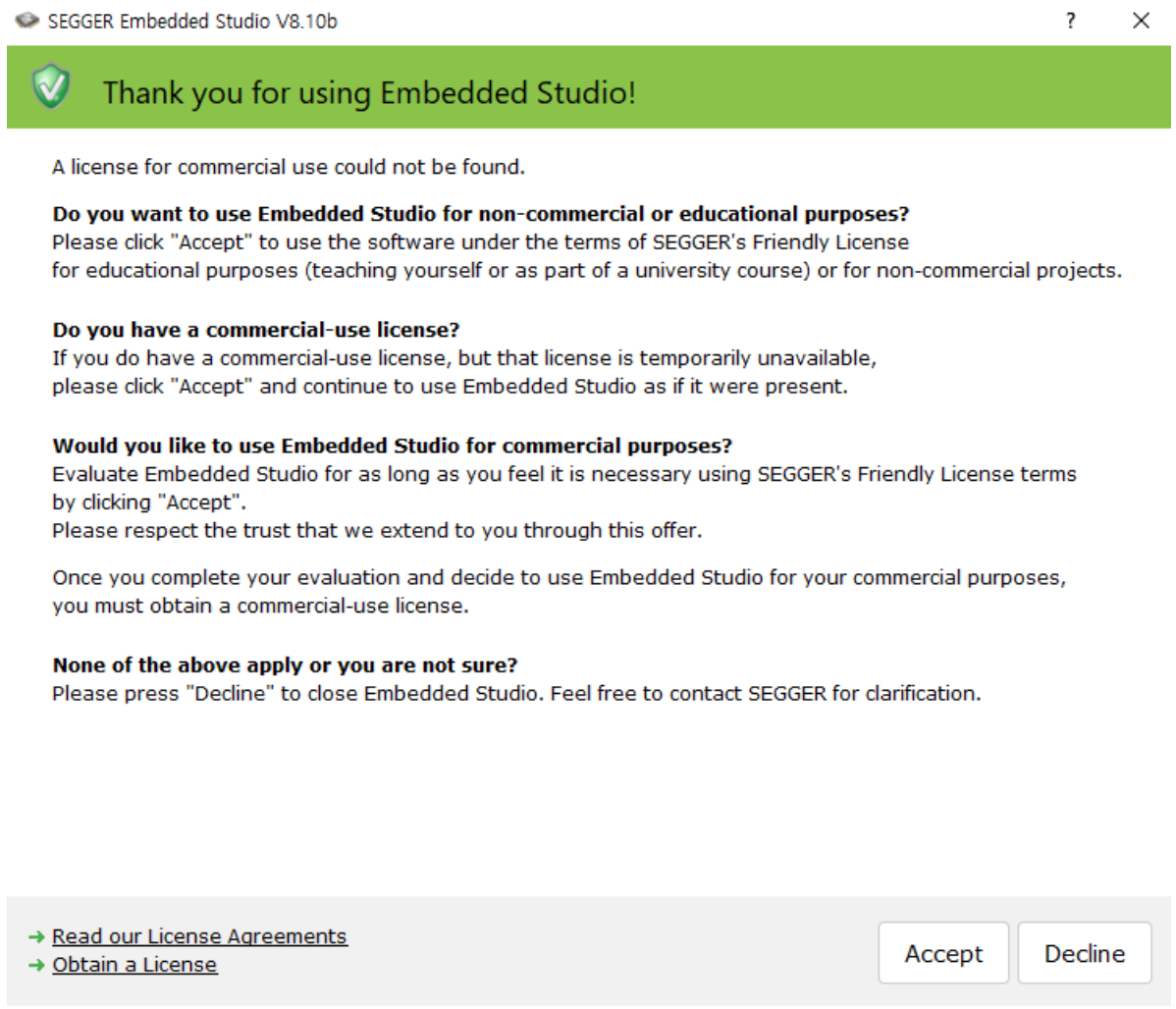
Create project

- ‘Finish’



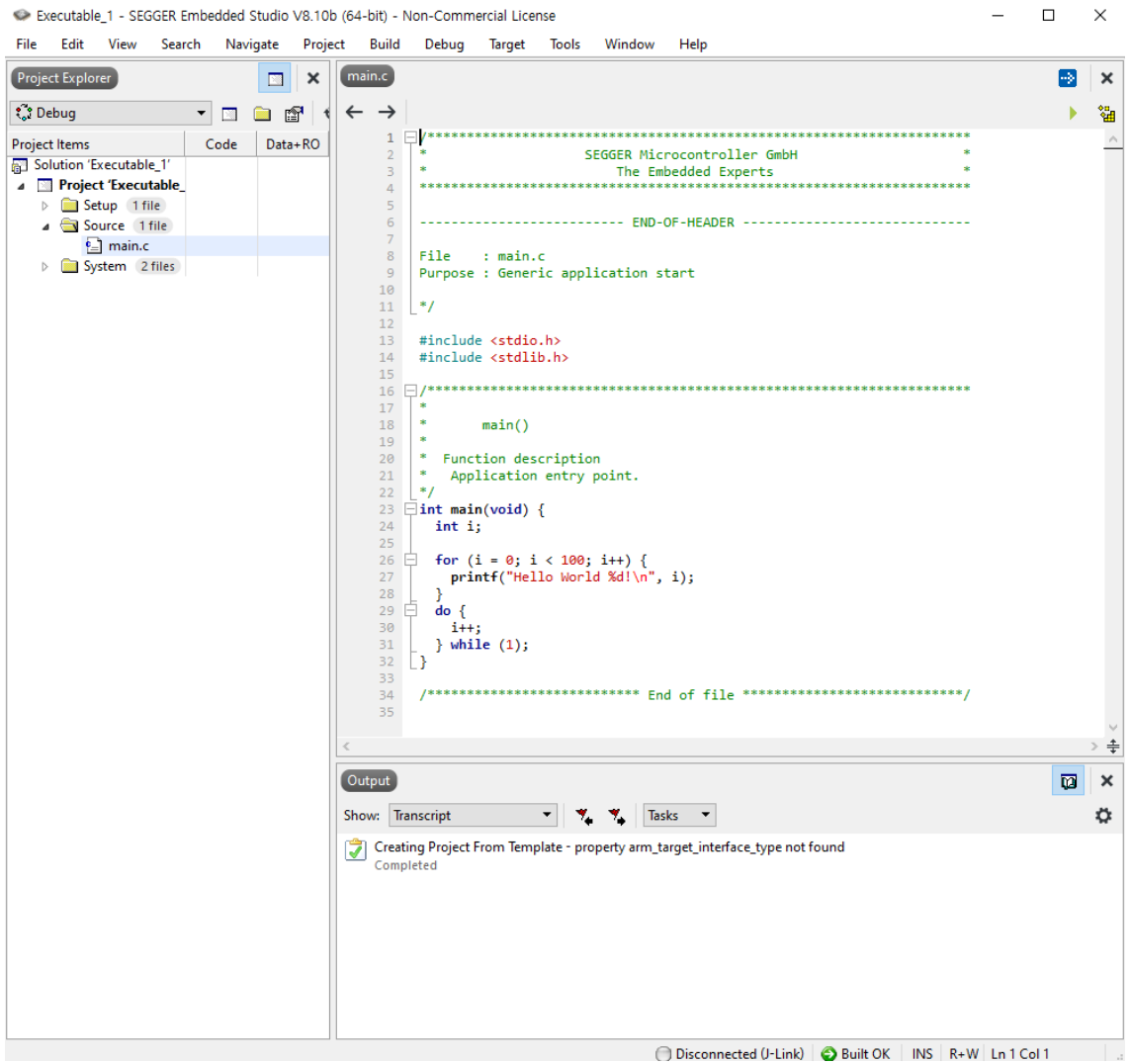
Create project

- ‘Accept’



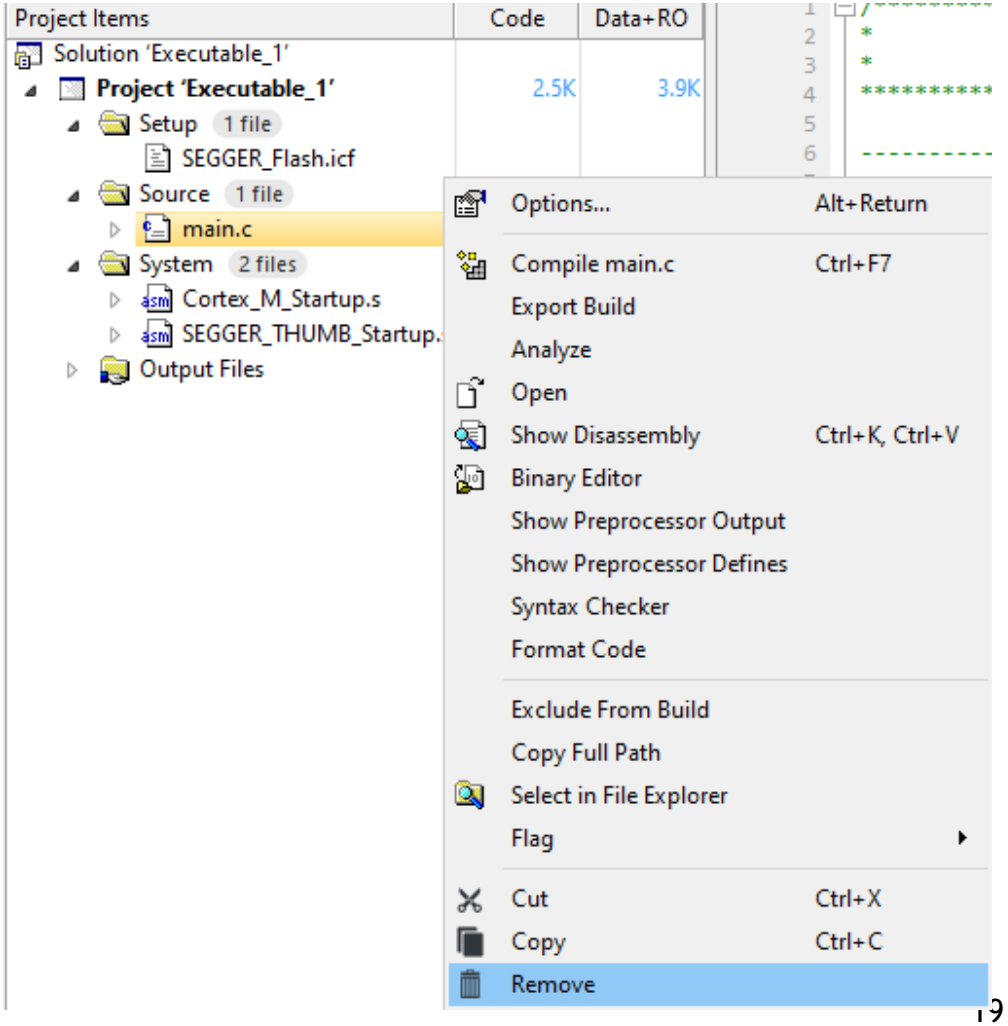
Create project

- You can see workspace



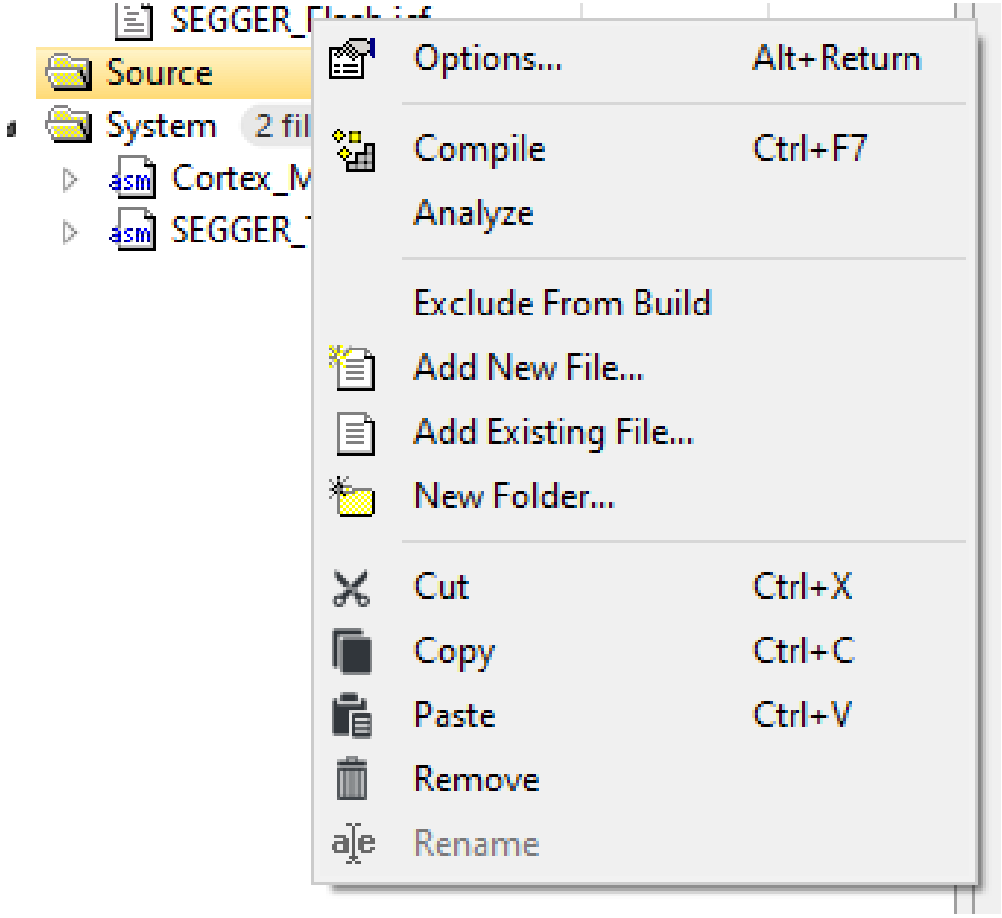
Basic settings

- Before run Assembly code, you have to remove main.c(Source directory)
- Source -> main.c(right click) -> Remove



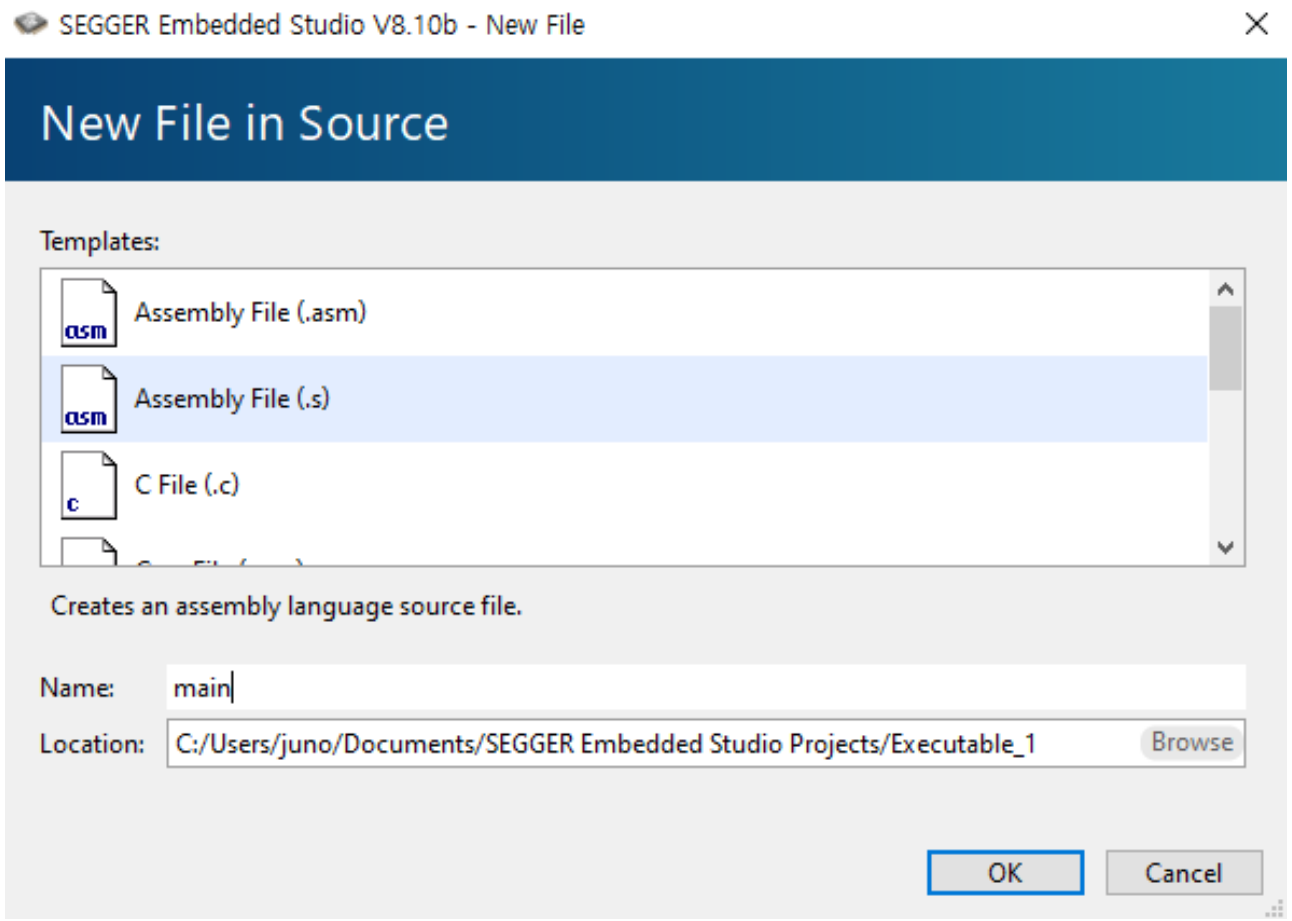
Basic settings

- Then, add a new file for assembly code
- Source(right click) -> ‘Add New File…’



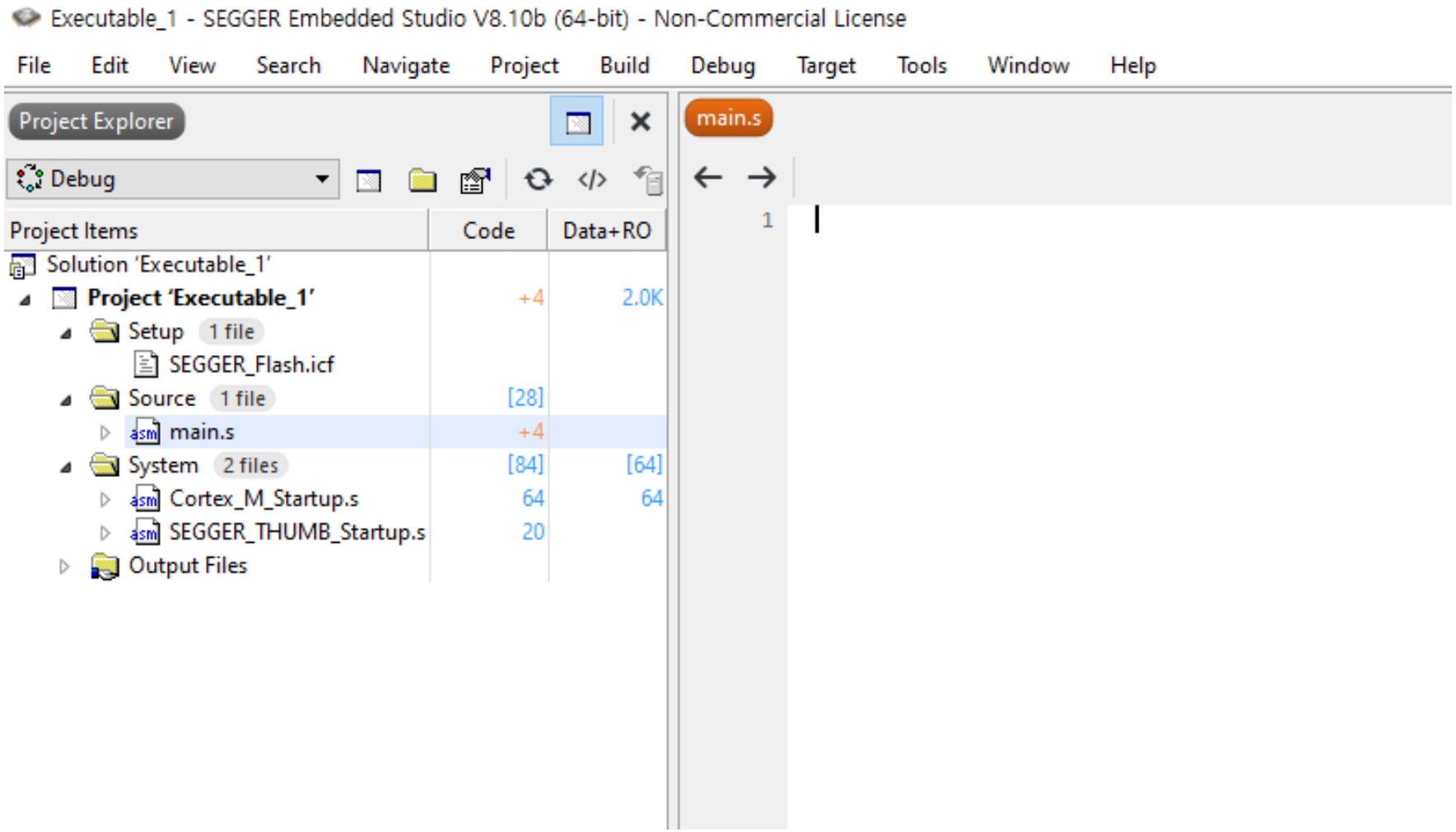
Basic settings

- Select ‘Assembly File(.s)’ -> ‘OK’



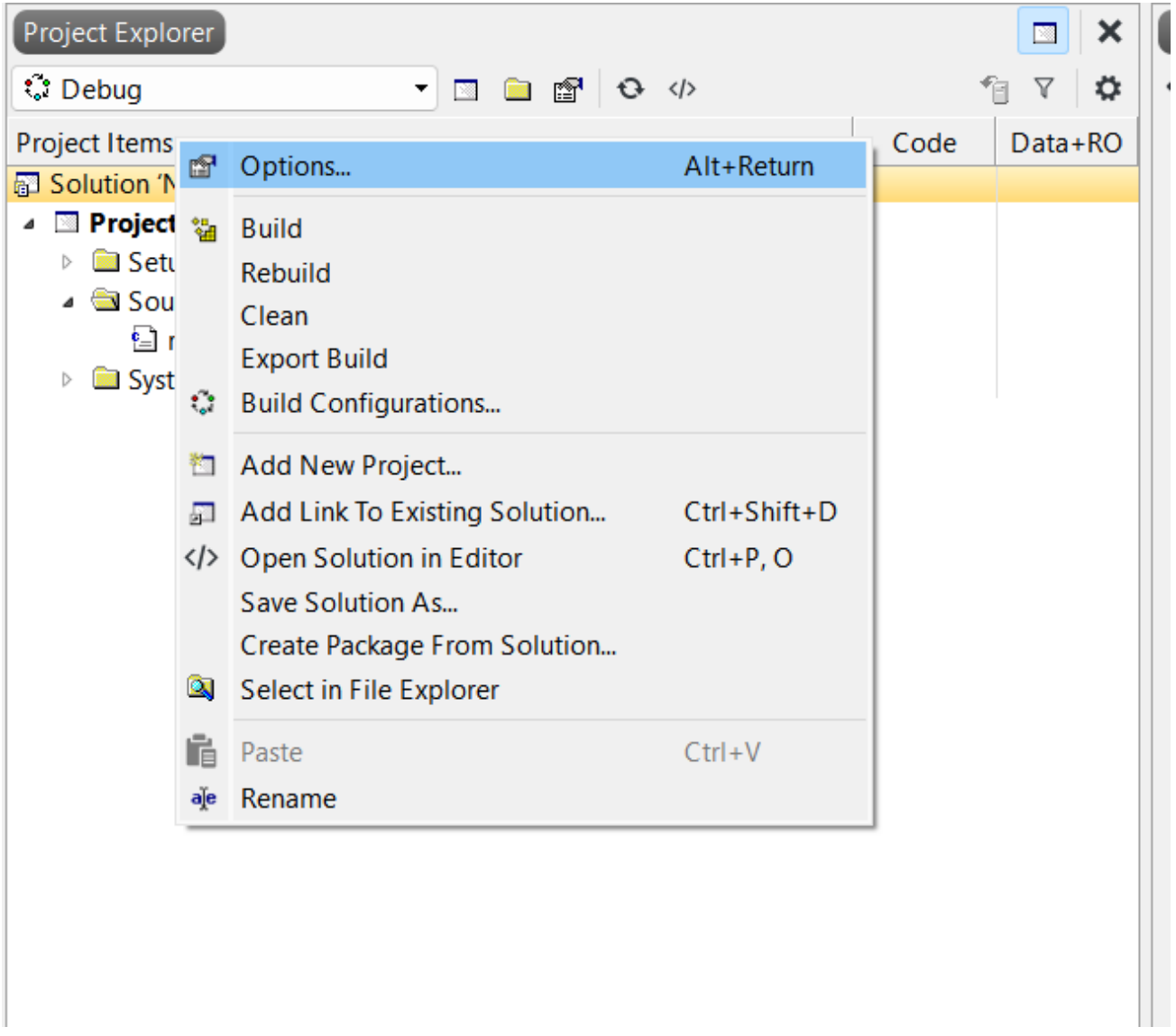
Basic settings

- main.c is replaced by main.s



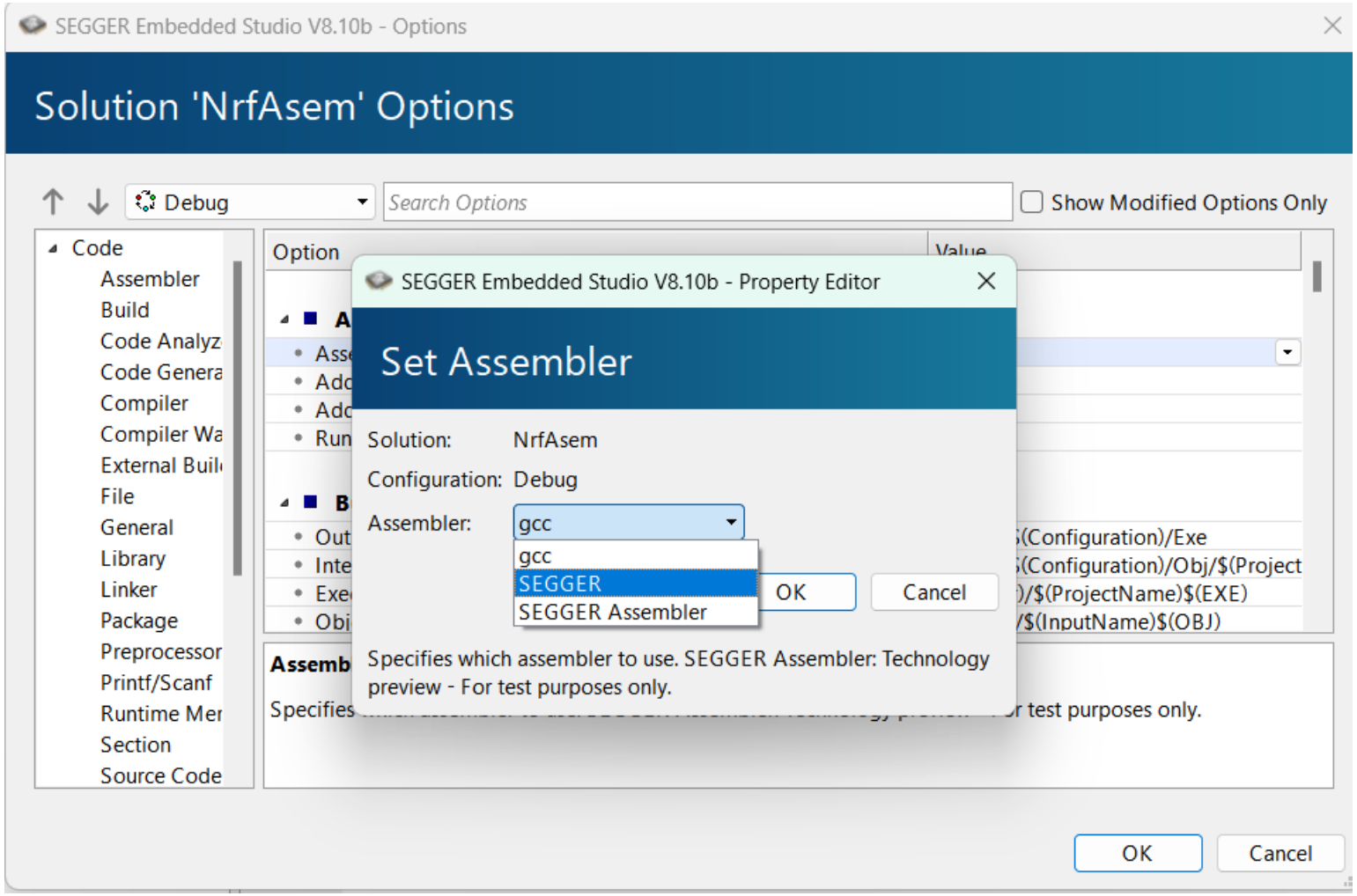
Basic settings

- Set Assembler



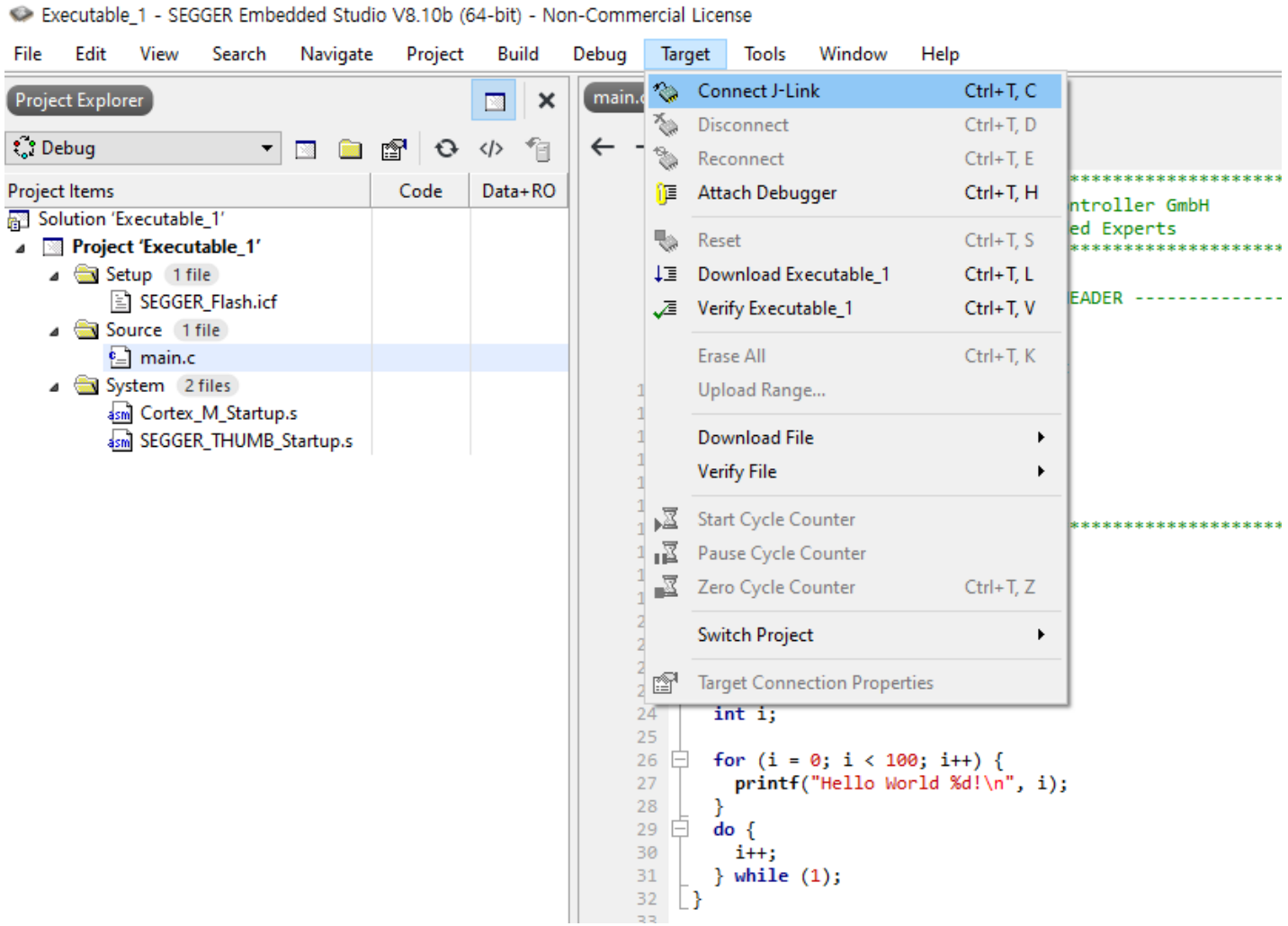
Basic settings

- Set Assembler (gcc -> segger)



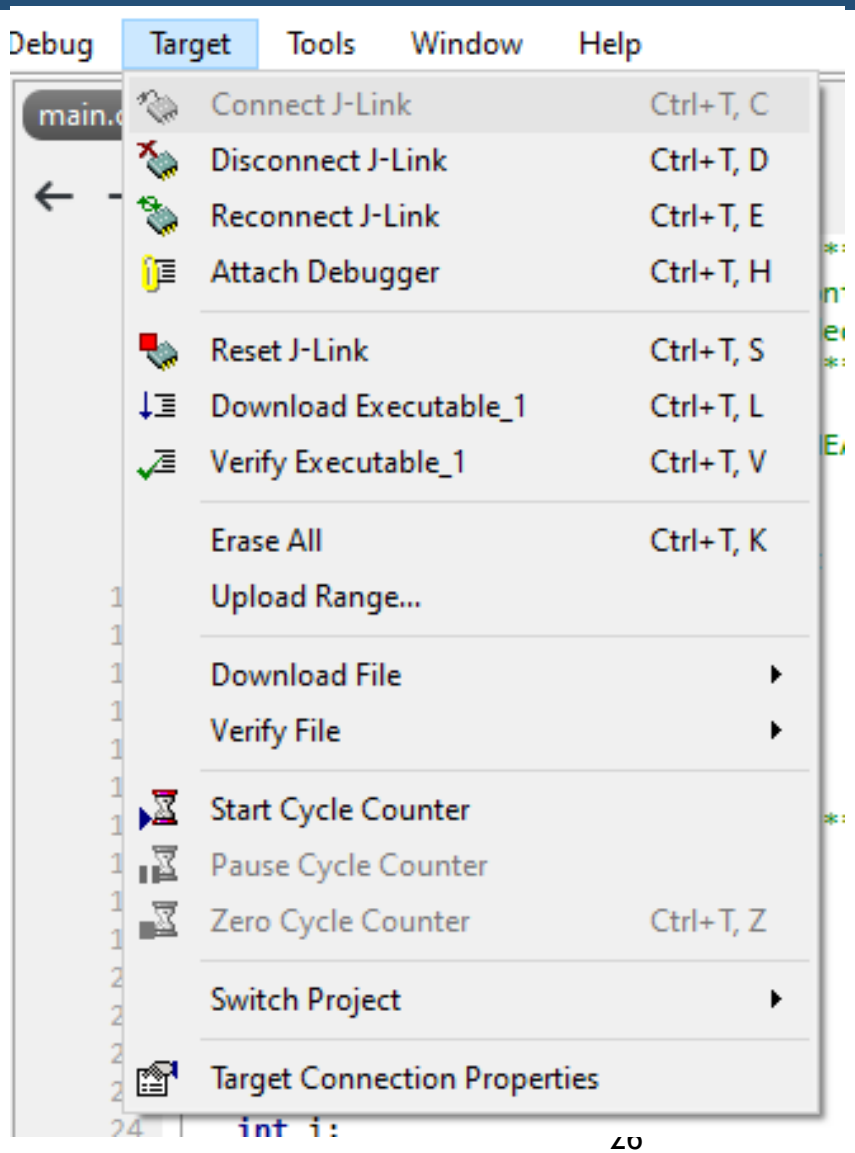
Basic settings

1. Connect nRF52840 with your PC
2. ‘Target’ -> ‘Connect J-Link’



Basic settings

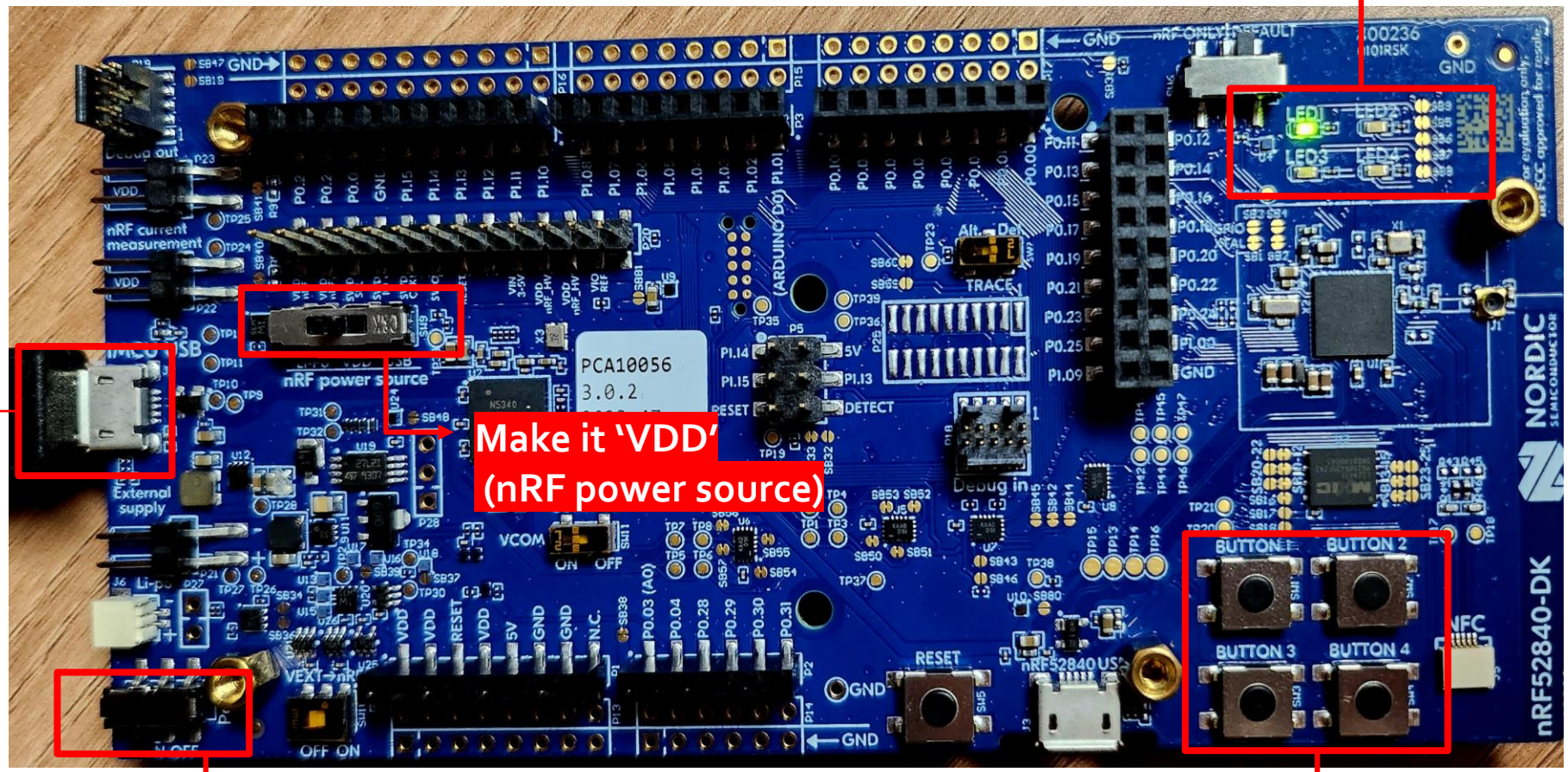
- If connection is completed, it will be changed like this



Agenda

1. Install SEGGER Embedded Studio
2. Basic setting
3. **GPIO**
4. Lab-1
5. Lab-2
6. Lab-3
7. Lab-4
8. Assignment

GPIO pins (LEDs, Buttons)

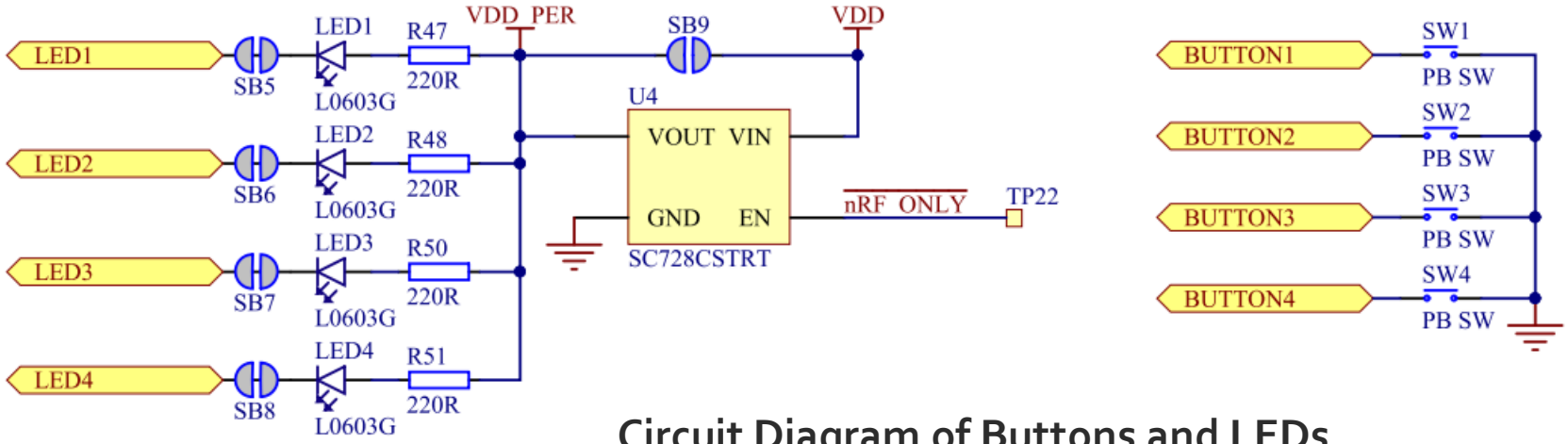


GPIO pins

- LED1~LED3: **Active Low**
- Button1~Button3:

  Active Low

  => Need Internal Pull-Up



Circuit Diagram of Buttons and LEDs

Part	GPIO	GPIO alternative	Solder bridge
Button 1	P0.11	P1.07	-
Button 2	P0.12	P1.08	-
Button 3	P0.24		-
Button 4	P0.25		-
LED 1	P0.13		SB5
LED 2	P0.14		SB6
LED 3	P0.15		SB7
LED 4	P0.16		SB8

GPIO of nRF52840

- The general purpose input/output pins (GPIOs) are grouped as one or more ports with each port having up to 32 GPIOs.
- The GPIO port peripheral implements up to 32 pins, PIN0 through PIN31. Each of these pins can be individually configured in the PIN_CNF[n] registers (n=0..31).
- GPIOTE handles GPIO ports Events and Tasks
- NVIC is responsible for the external and internal interrupts and exception requests. (Vector Table, Interrupt Priority, Interrupt Masking, State Saving and Restoration with ISR invocation)

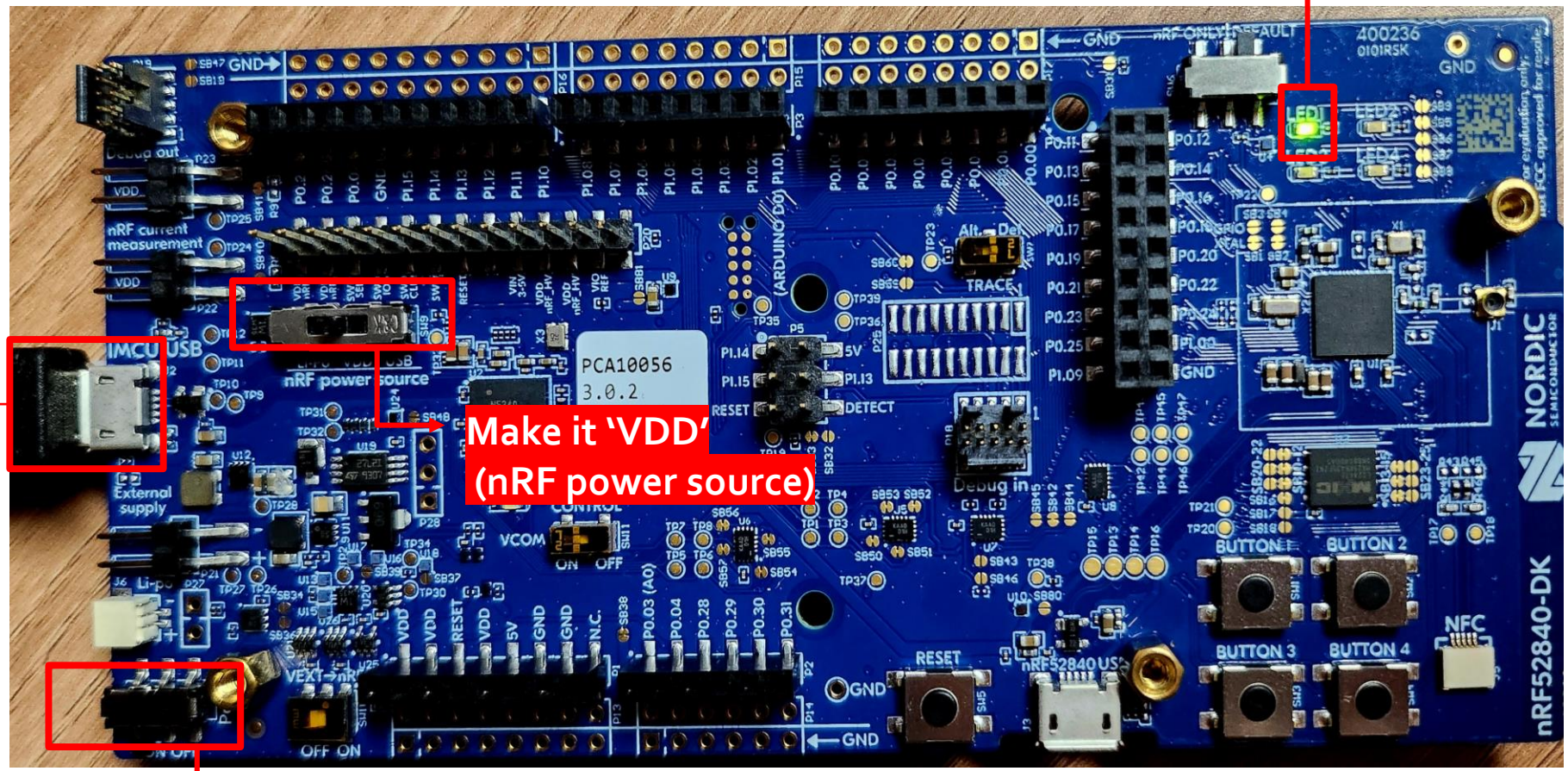
Agenda

1. Install SEGGER Embedded Studio
2. Basic setting
3. GPIO
4. **Lab-1**
5. Lab-2
6. Lab-3
7. Lab-4
8. Assignment

Lab-1

- Mission for Lab-1
  : Turn ON the LED1 on the nRF52840DK
  (by writing OURCLR register)

GPIO pins (LEDs, Buttons)



Basic Steps for nRF52840 Bare-Metal Programming

- Turn ON the LED-1 of nRF52840DK board by Storing the Memory

1. Obtain Memory Location (address) where the registers to configure that pin of that particular port are located
2. The fields of this register are set to configure the pin as **output**
3. Set the pin **output value as ‘0’** to make LED ON (Active_Low)

Lab-1

Part	GPIO	GPIO alternative	Solder bridge
Button 1	P0.11	P1.07	-
Button 2	P0.12	P1.08	-
Button 3	P0.24		-
Button 4	P0.25		-
LED 1	P0.13		SB5
LED 2	P0.14		SB6
LED 3	P0.15		SB7
LED 4	P0.16		SB8

Lab-1

- GPIO Registers
- GPIO P0 Base Address : 0x500000000

Base address	Peripheral	Instance	Description	Configuration	
0x50000000	GPIO	GPIO	General purpose input and output		Deprecated
0x50000000	GPIO	P0	General purpose input and output, port 0	P0.00 to P0.31 implemented	
0x50000300	GPIO	P1	General purpose input and output, port 1	P1.00 to P1.15 implemented	

Table 44: Instances

Register	Offset	Description
OUT	0x504	Write GPIO port
OUTSET	0x508	Set individual bits in GPIO port
OUTCLR	0x50C	Clear individual bits in GPIO port
IN	0x510	Read GPIO port
DIR	0x514	Direction of GPIO pins
DIRSET	0x518	DIR set register
DIRCLR	0x51C	DIR clear register
LATCH	0x520	Latch register indicating what GPIO pins that have met the criteria set in the PIN_CNF[n].SENSE registers
DETECTMODE	0x524	Select between default DETECT signal behaviour and LDETECT mode
PIN_CNF[0]	0x700	Configuration of GPIO pins
PIN_CNF[1]	0x704	Configuration of GPIO pins
PIN_CNF[2]	0x708	Configuration of GPIO pins

Lab-1

- Configure PORT Pin's Direction as OUTPUT by Setting DIRSET register

6.9.2.6 DIRSET

Address offset: 0x518

DIR set register

Read: reads value of DIR register.

Bit number				31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID				f	e	d	c	b	a	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
Reset 0x00000000				**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	RW	Field	Value ID	Value	Description
A-f	RW	PIN[i] (i=0..31)			Set as output pin i
			Input	0	Read: pin set as input
			Output	1	Read: pin set as output
			Set	1	Write: writing a '1' sets pin to output; writing a '0' has no effect

Lab-1

- Write ‘0’ value into Output Pins of P0 port (using ‘OUTCLR’ register)

6.9.2.3 OUTCLR

Address offset: 0x50C

Clear individual bits in GPIO port

Read: reads value of OUT register.

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID				f e d c b a Z Y X W V U T S R Q P O N M L K J I H G F E D C B A
Reset 0x00000000				**0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0**
ID	RW	Field	Value ID	Value	Description
A-f	RW	PIN[i] (i=0..31)			Pin i
			Low	0	Read: pin driver is low
			High	1	Read: pin driver is high
			Clear	1	Write: writing a '1' sets the pin low; writing a '0' has no effect

Lab-1

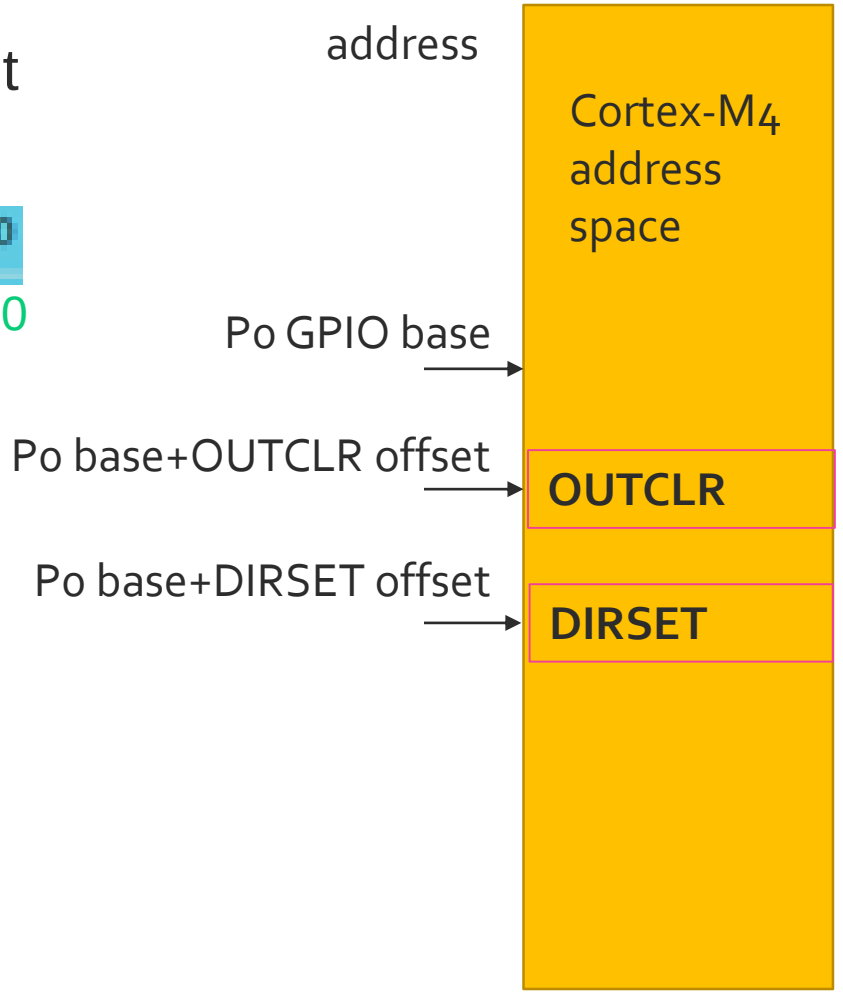
Mission: Turn On LED1 of nRF52840DK board

- LED1 is connected P0.13 GPIO pin with active-low circuit
- P0 has 32 bits and P0.13 pin mask is 0x**00002000**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Remove the main.c file from initial project
2. Create assembly program file xxx.s file
3. Code assembly program
   - main: label should be present at the beginning of code
   - Use DIRSET register for setting direction as ‘output’

   and OUTCLR register for applying value as ‘0’



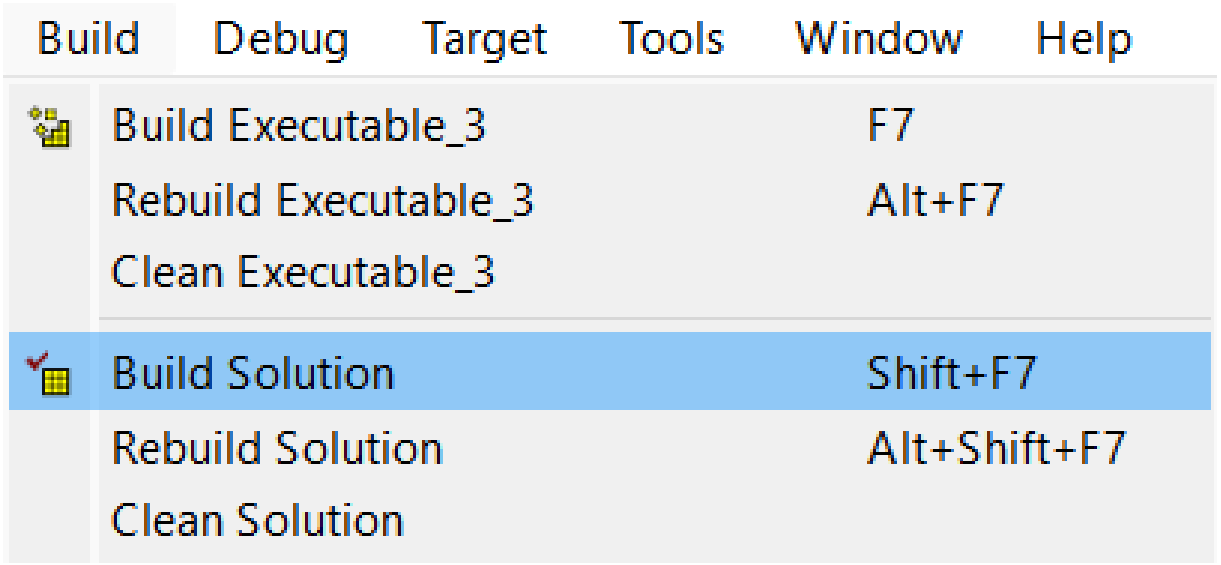
Lab-1

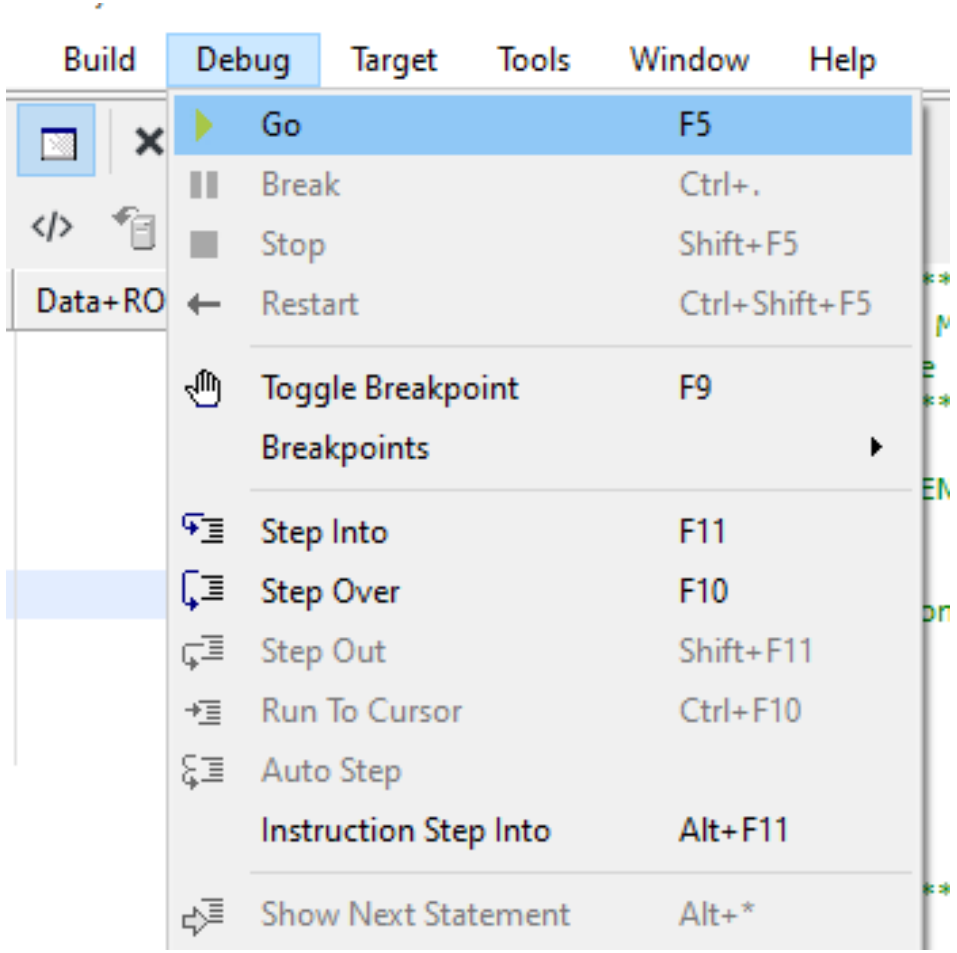
- Should Set as Thumb state
  - ARMv7-M supports only Thumb-2
- main Labeled Code is required
- Port Pin Related Registers Address
  - P0 GPIO port base address = 0x50000000
  - DIRSET address offset is 0x514
  - OUTCLR address offset is 0x50C

```
.thumb
.thumb_func
.global main


main:
//DIRSET register
  LDR R0, =0x50000000
  LDR R1, =0x02000
  STR R1, [R0, #0x514]

// OUTCLR register
  LDR R0, =0x50000000
  LDR R1, =0x02000
  STR R1, [R0, #0x50C]
b loop b // infinite loop
```

Lab-1: Assemble and Run the Code

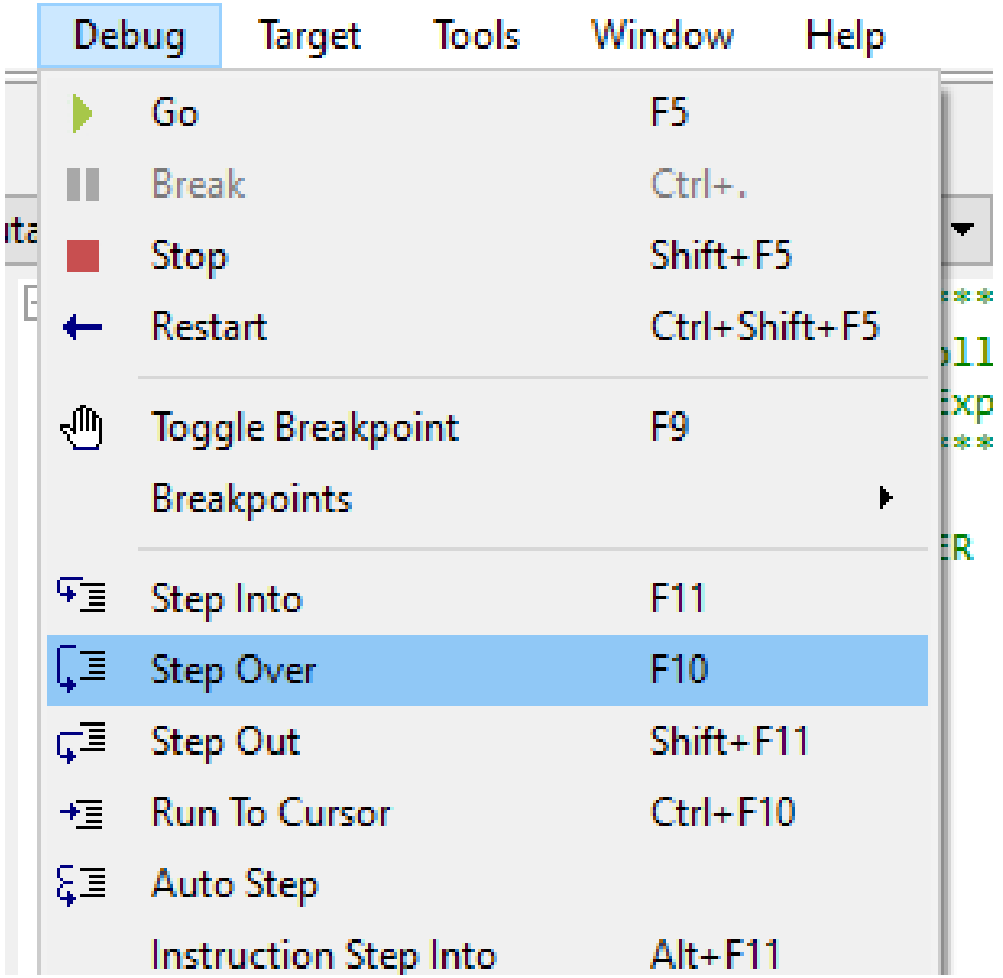
1. **‘Build’ menu → Build Solution**
2. **‘Debug’ menu -> ‘Go’**



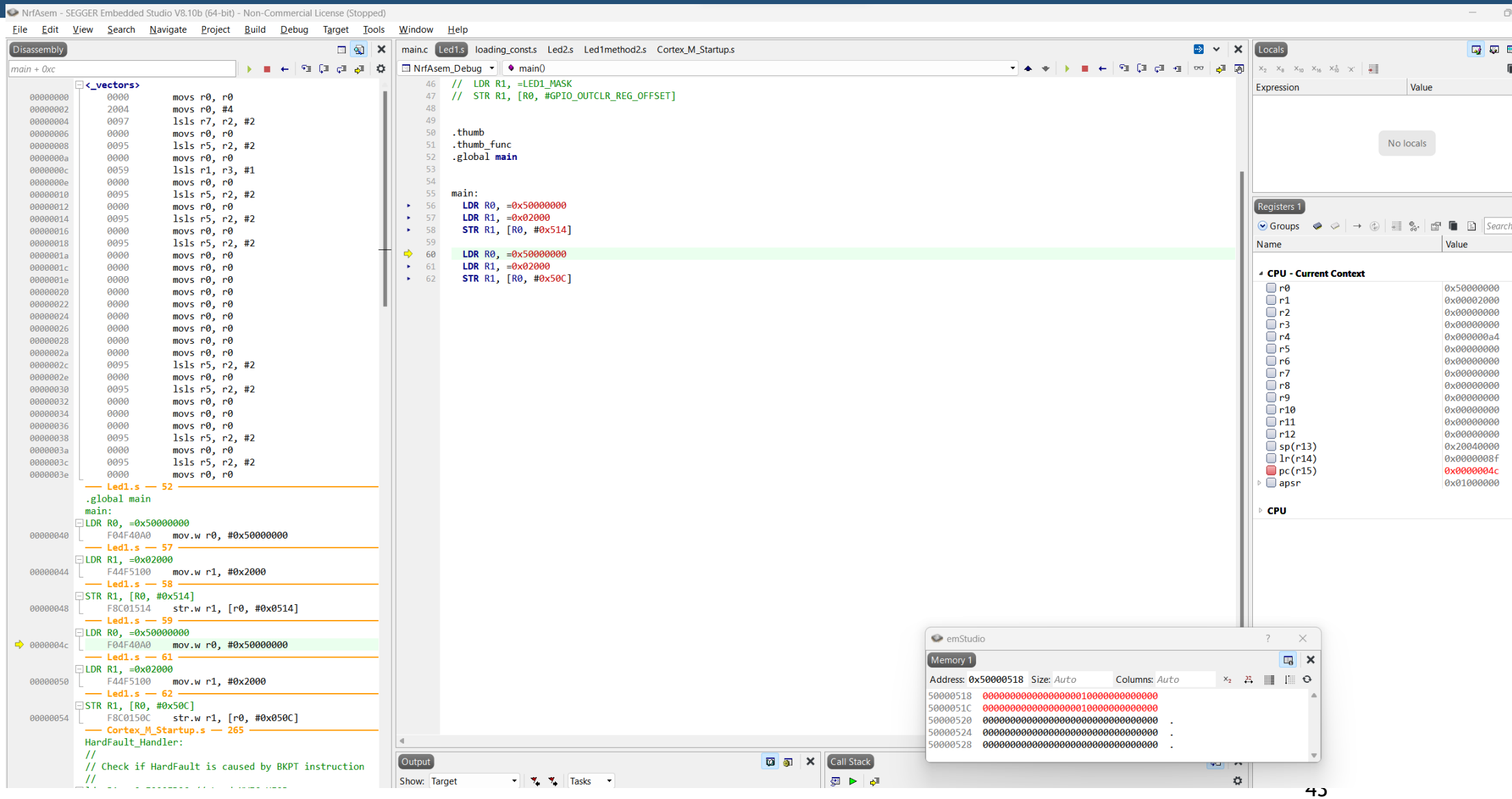


Lab-1

- Execute the Program line by line
- ‘Debug’ → ‘Click Step Over’ or ‘Step into’

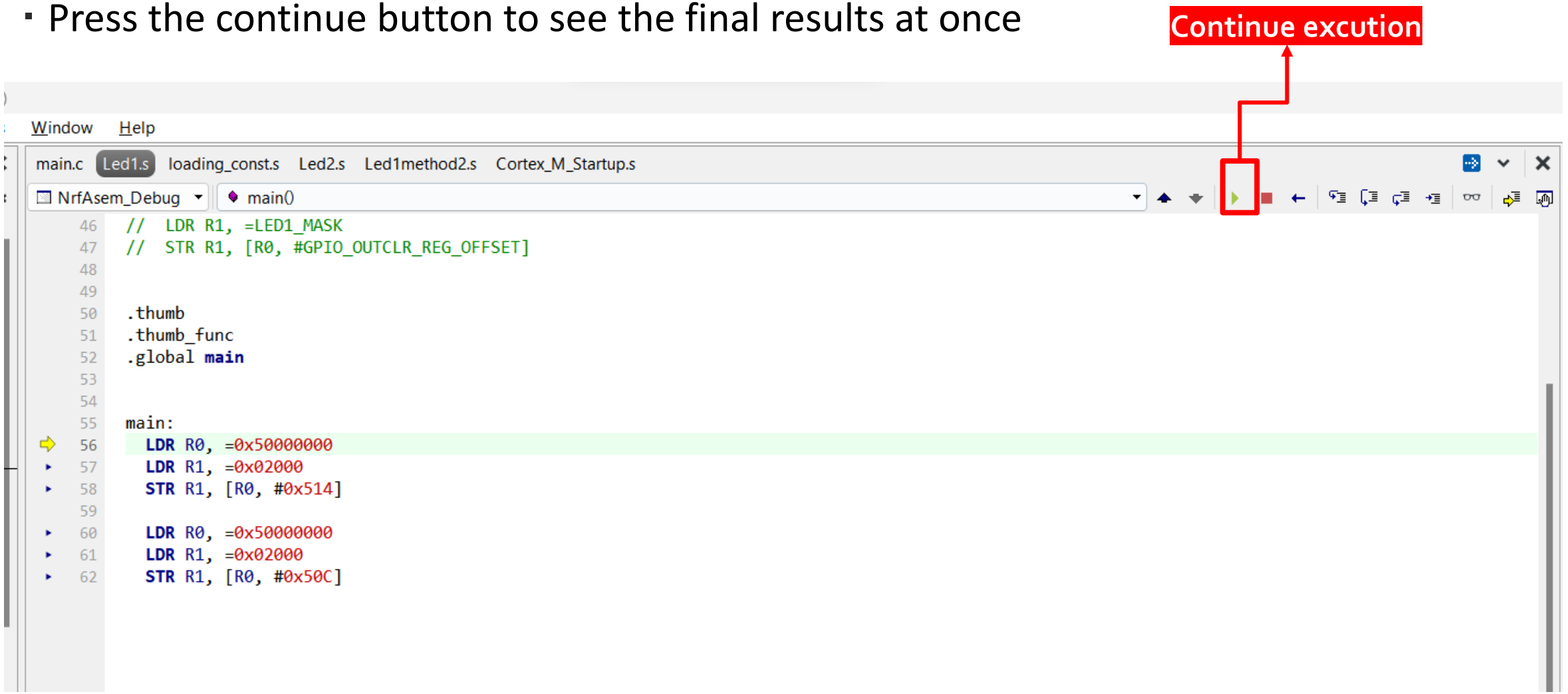


Lab-1


NrfAsem - SEGGER Embedded Studio V8.10b (64-bit) - Non-Commercial License (Stopped)
File Edit View Search Navigate Project Build Debug Target Tools Window Help
Disassembly
main + 0xc
<_vectors>
00000000 0000 movs r0, r0
00000002 2004 movs r0, #4
00000004 0097 lsls r7, r2, #2
00000006 0000 movs r0, r0
00000008 0095 lsls r5, r2, #2
0000000a 0000 movs r0, r0
0000000c 0059 lsls r1, r3, #1
0000000e 0000 movs r0, r0
00000010 0095 lsls r5, r2, #2
00000012 0000 movs r0, r0
00000014 0095 lsls r5, r2, #2
00000016 0000 movs r0, r0
00000018 0095 lsls r5, r2, #2
0000001a 0000 movs r0, r0
0000001c 0000 movs r0, r0
0000001e 0000 movs r0, r0
00000020 0000 movs r0, r0
00000022 0000 movs r0, r0
00000024 0000 movs r0, r0
00000026 0000 movs r0, r0
00000028 0000 movs r0, r0
0000002a 0000 movs r0, r0
0000002c 0095 lsls r5, r2, #2
0000002e 0000 movs r0, r0
00000030 0095 lsls r5, r2, #2
00000032 0000 movs r0, r0
00000034 0000 movs r0, r0
00000036 0000 movs r0, r0
00000038 0095 lsls r5, r2, #2
0000003a 0000 movs r0, r0
0000003c 0095 lsls r5, r2, #2
0000003e 0000 movs r0, r0
Led1.s — 52
.global main
main:
LDR R0, =0x50000000
00000040 F04F40A0 mov.w r0, #0x50000000
Led1.s — 57
LDR R1, =0x02000
00000044 F44F5100 mov.w r1, #0x2000
Led1.s — 58
STR R1, [R0, #0x514]
00000048 F8C01514 str.w r1, [r0, #0x0514]
Led1.s — 59
LDR R0, =0x50000000
0000004c F04F40A0 mov.w r0, #0x50000000
Led1.s — 61
LDR R1, =0x02000
00000050 F44F5100 mov.w r1, #0x2000
Led1.s — 62
STR R1, [R0, #0x50C]
00000054 F8C0150C str.w r1, [r0, #0x050C]
Cortex_M_Startup.s — 265
HardFault_Handler:
//
// Check if HardFault is caused by BKPT instruction
//
main.c Led1.s loading_const.s Led2.s Led1method2.s Cortex_M_Startup.s
NrfAsem_Debug
main()
46 // LDR R1, =LED1_MASK
47 // STR R1, [R0, #GPIO_OUTCLR_REG_OFFSET]
48
49
50 .thumb
51 .thumb_func
52 .global main
53
54
55 main:
56 LDR R0, =0x50000000
57 LDR R1, =0x02000
58 STR R1, [R0, #0x514]
59
60 LDR R0, =0x50000000
61 LDR R1, =0x02000
62 STR R1, [R0, #0x50C]
Output
Show: Target
Tasks
Call Stack
Locals
Expression
Value
No locals
Registers 1
Groups
Search R
Name
Value
CPU - Current Context
r0 0x50000000
r1 0x00002000
r2 0x00000000
r3 0x00000000
r4 0x000000a4
r5 0x00000000
r6 0x00000000
r7 0x00000000
r8 0x00000000
r9 0x00000000
r10 0x00000000
r11 0x00000000
r12 0x00000000
sp(r13) 0x20040000
lr(r14) 0x0000008f
pc(r15) 0x0000004c
apsr 0x01000000
CPU
emStudio
Memory 1
Address: 0x50000518 Size: Auto Columns: Auto
50000518 00000000000000000001000000000000
5000051C 00000000000000000001000000000000
50000520 00000000000000000000000000000000 .
50000524 00000000000000000000000000000000 .
50000528 00000000000000000000000000000000 .

Lab-1

- Press the continue button to see the final results at once



Agenda

1. Install SEGGER Embedded Studio
2. Basic setting
3. GPIO
4. Lab-1
5. **Lab-2**
6. Lab-3
7. Lab-4
8. Assignment

Lab-2

- **Mission: Turn on/off LED1 using OUT (instead of OUTCLR) register of P0 port**

6.9.2.1 OUT

Address offset: 0x504

Write GPIO port

Bit number				31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID				f	e	d	c	b	a	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
Reset 0x00000000				**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
ID	RW	Field	Value ID	Value	Description																														
A-f	RW	PIN[i] (i=0..31)			Pin i																														
			Low	0	Pin driver is low																														
			High	1	Pin driver is high																														

Lab-2

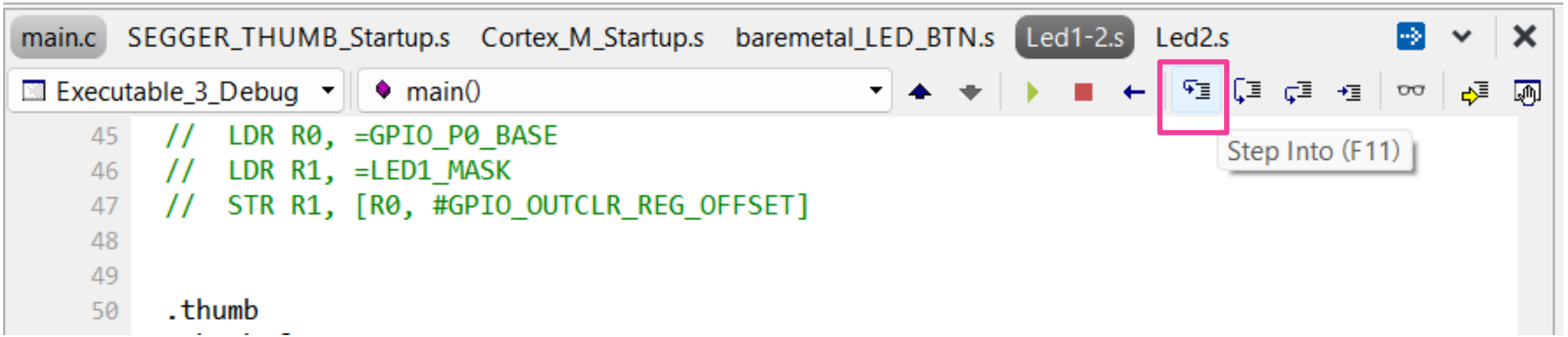
- Assembly Code to Turn LED1 ON and OFF (using ‘OUT’ register)

```
.thumb
.thumb_func

.global main

.equ LED3_MASK, 0x08000
.equ GPIO_PO_BASE, 0x50000000
.equ GPIO_OUT_REG_OFFSET, 0x504
.equ GPIO_DIR_REG_OFFSET, 0x514
.equ GPIO_OUTCLR_REG_OFFSET, 0x50C
.equ GPIO_DIRSET_REG_OFFSET, 0x518
```

```
main:
  LDR R0, =GPIO_PO_BASE+GPIO_DIR_REG_OFFSET
  LDR R1, =LED3_MASK
  STR R1, [R0]

  LDR R0, =GPIO_PO_BASE+GPIO_OUT_REG_OFFSET
  LDR R1, =LED3_MASK
  STR R1, [R0]     // OUT_REG ‘1’ ➔ LED3 OFF

  MVN R2, R1
  STR R2, [R0]   // OUT_REG ‘0’ ➔ LED3 ON

loop: b loop
```

Lab-2

1. Build → Build Solution
2. Debug → Go
3. Debug → Step : Execute Line by Line using Step to confirm LED ON/OFF



Agenda

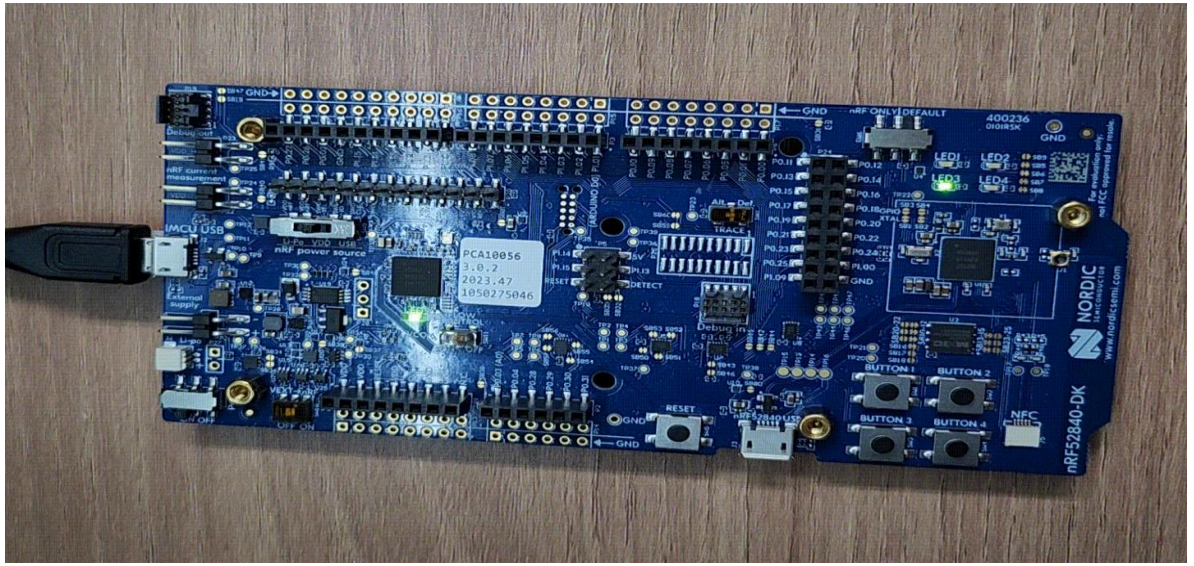
1. Install SEGGER Embedded Studio
2. Basic setting
3. GPIO
4. Lab-1
5. Lab-2
6. **Lab-3**
7. Lab-4
8. Assignment

Lab-3

- **Mission:**

  Turn LED3 ON and OFF Repeatedly for every 1 sec

  (Turn on 1 second and Turn off 1 second)



Lab-3

- How to delay 1 second?
  - ARM® Cortex®-M4 32-bit processor with FPU, 64 MHz
    - System Clock Rate = 64MHz = 64,000,000 clock cycles per second
    - One instruction takes 1 clock cycle due to Pipeline Structure of ARM Core
    - Countdown Loop consists 4 instructions = 4 clock cycles per counting 1
    - for 1 second counting, 64M / 4 = 16M

Lab-3

- GPIO P0 Register Address
  - PORT BASE address = 0x50000000
  - **DIRSET** register offset = 0x518 (writing '1' to a bit => set the pin OUTPUT, '0' => No effect
  - **OUTCLR** register offset = 0x50C (writing '1' to a bit => set the pin LOW, '0' => No effect)
  - **OUTSET** register offset = 0x508 (Writing '1' to a bit => set the pin HIGH, '0' => No effect)
  - **IN** register offset = 0x510 (Reading Port input value)

- Mask of LEDs
  - LED1: P0.13 mask ➔ 0b 0000 **0010** 0000 0000 0000 = 0x02000
  - LED2: P0.14 mask ➔ 0b 0000 **0100** 0000 0000 0000 = 0x04000
  - LED3: P0.15 mask ➔ 0b 0000 **1000** 0000 0000 0000 = 0x08000
  - LED4: P0.16 mask ➔ 0b **0001 0000** 0000 0000 0000 = 0x10000

Lab-3

```
.thumb
.thumb_func
.global main


.equ LED1_MASK, 0x02000
.equ LED2_MASK, 0x04000
.equ LED3_MASK, 0x08000
.equ LED4_MASK, 0x10000
.equ LED_ALL_MASK, (LED1_MASK+LED2_MASK+LED3_MASK+LED4_MASK)


.equ GPIO_P0_BASE, 0x50000000
.equ GPIO_OUT_REG_OFFSET, 0x504
.equ GPIO_DIR_REG_OFFSET, 0x514
.equ GPIO_DIRSET_REG_OFFSET, 0x518
```

Lab-3

```
main:
  LDR R0, =GPIO_P0_BASE
  LDR R1, = LED3_MASK
  STR R1, [R0,#GPIO_DIR_REG_OFFSET]  // GPIO DIR (1:out, 0: input)

loop:
  BL TURN_OFF_LED
  BL Delay_One_second
  BL TURN_ON_LED
  BL Delay_One_second
  B loop
```

Lab-3

```
// Delay Routtine
 Delay_One_second:
   LDR R2, = 16000000 // loop counter for 1 sec with 64MHz system clock
 count_down:
   CMP R2, #0
   ITT NE
   SUBNE R2, R2, #1
   BNE count_down
   MOV PC, LR                    // return

// LED ON Routine
TURN_ON_LED:
   LDR R0, =GPIO_P0_BASE+0x504  // OUT register (for my board LED, 0:ON, 1:OFF)
   LDR R1, = LED3_MASK
   MVN R1, R1                    // complement
   STR R1, [R0]
   MOV PC, LR                    // return

// LED OFF Routine
TURN_OFF_LED:
   LDR R0, =GPIO_P0_BASE+0x504      // OUT register
   LDR R1, =LED_ALL_MASK
   STR R1,[R0]
   MOV PC, LR                       // return
```

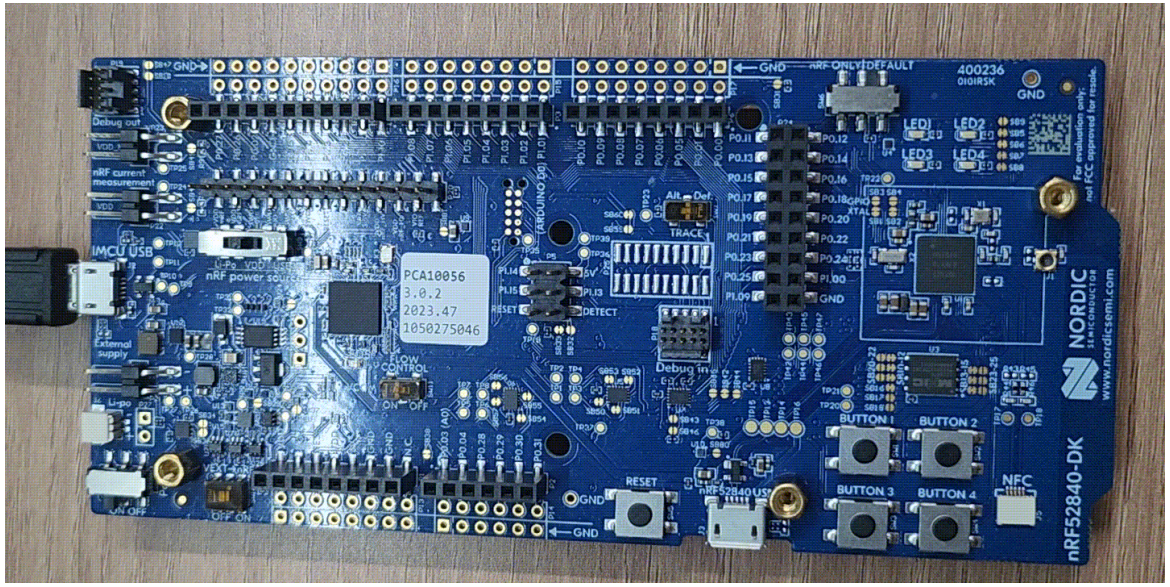
Agenda

1. Install SEGGER Embedded Studio
2. Basic setting
3. GPIO
4. Lab-1
5. Lab-2
6. Lab-3
7. **Lab-4**
8. Assignment

Lab-4

Mission:

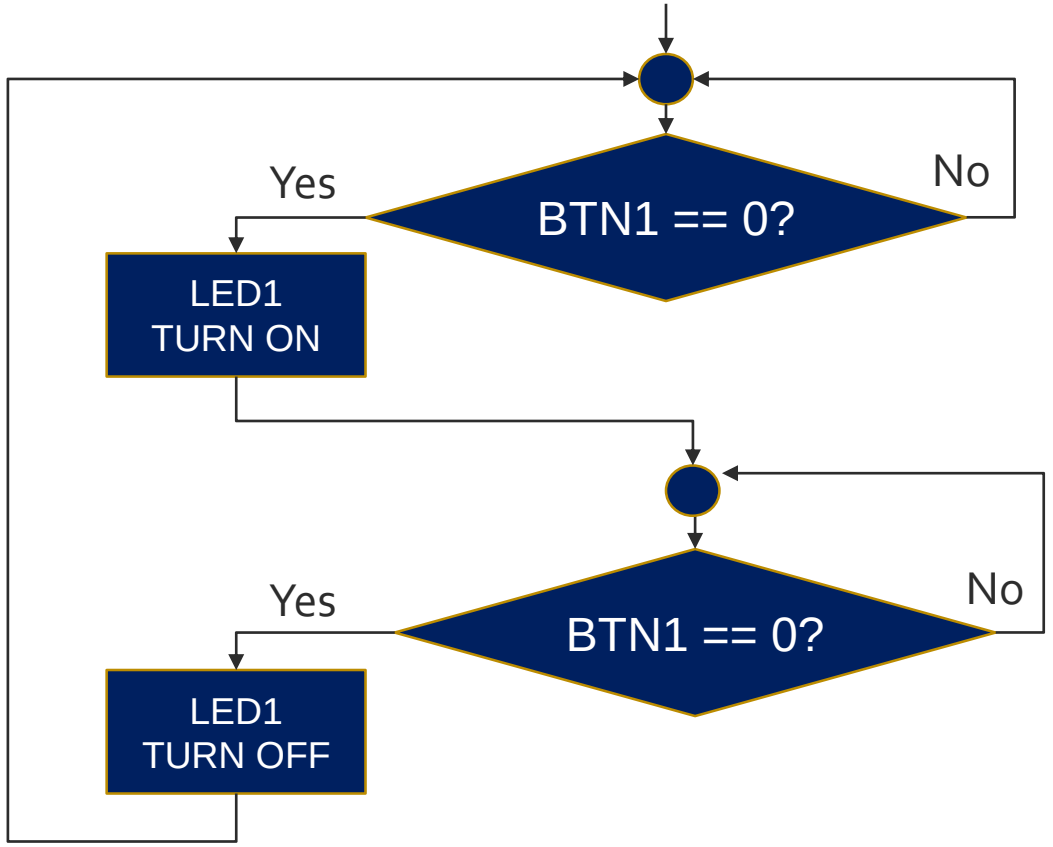
With Button 1 pressed Event, make the state of LED1 toggle (polling method version)



Lab-4 LED1 Toggling by Button1

Use Polling method to check Button Push.

Polling means continuously reading the status of the pin to check if it has changed.



Lab-4

- Button 1 is p0.11. (slide 32)

Register	Offset	Description
OUT	0x504	Write GPIO port
OUTSET	0x508	Set individual bits in GPIO port
OUTCLR	0x50C	Clear individual bits in GPIO port
IN	0x510	Read GPIO port
DIR	0x514	Direction of GPIO pins
DIRSET	0x518	DIR set register
DIRCLR	0x51C	DIR clear register
LATCH	0x520	Latch register indicating what GPIO pins that have met the criteria set in the PIN_CNF[n].SENSE registers
DETECTMODE	0x524	Select between default DETECT signal behaviour and LDETECT mode
PIN_CNF[0]	0x700	Configuration of GPIO pins
PIN_CNF[1]	0x704	Configuration of GPIO pins
PIN_CNF[2]	0x708	Configuration of GPIO pins
PIN_CNF[3]	0x70C	Configuration of GPIO pins
PIN_CNF[4]	0x710	Configuration of GPIO pins
PIN_CNF[5]	0x714	Configuration of GPIO pins
PIN_CNF[6]	0x718	Configuration of GPIO pins
PIN_CNF[7]	0x71C	Configuration of GPIO pins
PIN_CNF[8]	0x720	Configuration of GPIO pins
PIN_CNF[9]	0x724	Configuration of GPIO pins
PIN_CNF[10]	0x728	Configuration of GPIO pins
PIN_CNF[11]	0x72C	Configuration of GPIO pins

Lab-4

- pin configuration

6.9.2.10 PIN_CNF[n] (n=0..31)

Address offset: 0x700 + (n × 0x4)

Configuration of GPIO pins

Bit number	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID	E E D D D C C B A
Reset 0x00000002	**0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0**

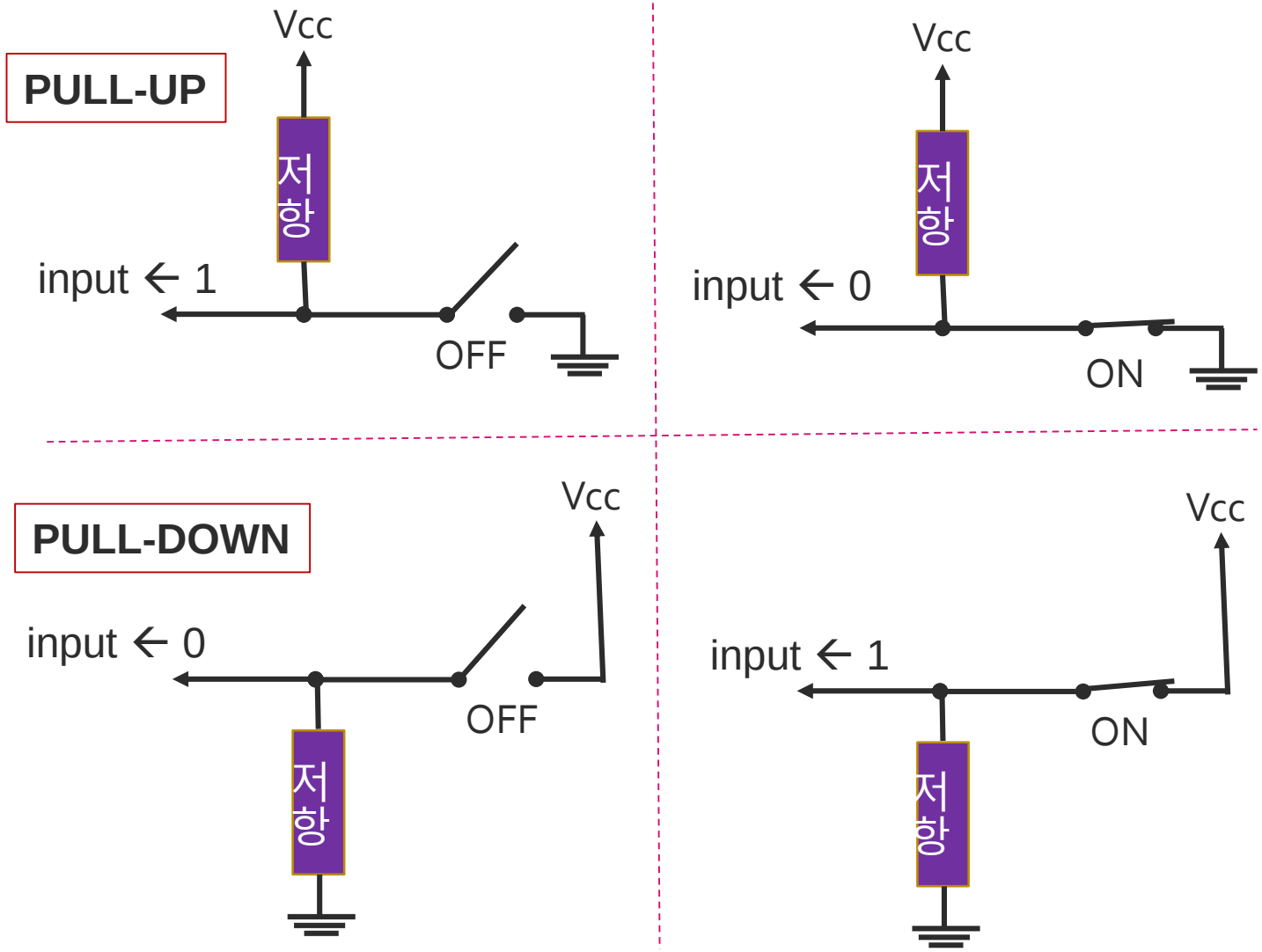
ID	RW	Field	Value ID	Value	Description
A	RW	DIR			Pin direction. Same physical register as DIR register
			Input	0	Configure pin as an input pin
			Output	1	Configure pin as an output pin
B	RW	INPUT			Connect or disconnect input buffer
			Connect	0	Connect input buffer

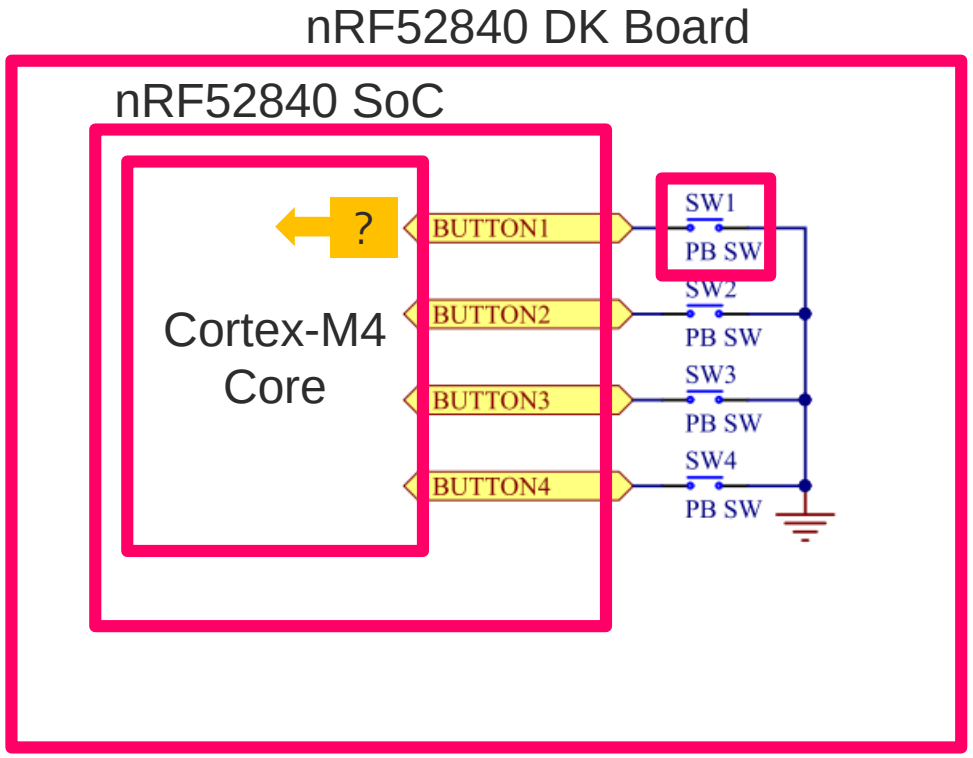
Lab-4

Bit number	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID	E E D D D C C B A
Reset 0x00000002	**0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0**

ID	RW	Field	Value ID	Value	Description
			Disconnect	1	Disconnect input buffer
C	RW	PULL			Pull configuration
			Disabled	0	No pull
			Pulldown	1	Pull down on pin
			Pullup	3	Pull up on pin
D	RW	DRIVE			Drive configuration
			S0S1	0	Standard '0', standard '1'
			H0S1	1	High drive '0', standard '1'
			S0H1	2	Standard '0', high drive '1'
			H0H1	3	High drive '0', high 'drive '1''
			D0S1	4	Disconnect '0' standard '1' (normally used for wired-or connections)
			D0H1	5	Disconnect '0', high drive '1' (normally used for wired-or connections)
			S0D1	6	Standard '0'. disconnect '1' (normally used for wired-and connections)
			H0D1	7	High drive '0', disconnect '1' (normally used for wired-and connections)
E	RW	SENSE			Pin sensing mechanism
			Disabled	0	Disabled
			High	2	Sense for high level
			Low	3	Sense for low level

Lab-4 Why PullUp Require for BTN1?





- The Button Circuit of our board has no pull-up or pull-down registers. With this configuration BUTTON1~4 input has undefined input voltage in normal state

➔ We have to Configure PINs for Button1~4 to have On-chip Pull-Up registers

Lab-4

```
.thumb
.thumb_func
.global main

.equ BTN1_MASK, 0x00800
.equ BTN2_MASK, 0x01000
.equ LED1_MASK, 0x02000
.equ LED2_MASK, 0x04000
.equ LED3_MASK, 0x08000
.equ LED4_MASK, 0x10000

.equ GPIO_P0_BASE, 0x50000000
.equ GPIO_OUT_REG_OFFSET, 0x504
.equ GPIO_IN_REG_OFFSET,      0x510
.equ GPIO_DIR_REG_OFFSET,   0x514
.equ GPIO_DIRSET_REG_OFFSET, 0x518
.equ GPIO_OUTCLR_REG_OFFSET, 0x50C
.equ GPIO_OUTSET_REG_OFFSET, 0x508

.equ GPIO_PIN_CNF_11_OFFSET,    0x72C  // P0.11 pin configuration address offset (BTN1)
```

Lab-4

```
main:
  LDR R0, =GPIO_P0_BASE
  LDR R1, =LED1_MASK
  STR R1, [R0, #GPIO_DIR_REG_OFFSET]  // GPIO DIR (1:out, 0: input)

  // Input pin Configuratrion
  LDR R0, =GPIO_P0_BASE+GPIO_PIN_CNF_11_OFFSET  // for GPIO P0.11 pin address

  @ A = 0(input), B = 0(connect input buffer), CC = 11 (pull up), DDD=000 (standard'0 and 1), EE=11 (SENS for LOW)
  @                              EE       D DD             CCBA
  @ 0000 0000 0000 0011 0000 0000 0000 1100 = 0x0003000C
  LDR R1, =0x0003000c //PIN_CNF[11] address
  STR R1, [R0, #GPIO_PIN_CNF_11_OFFSET ] // configure GPIO P0.11 pin
                                          // as Input, Connected input buffer,
                                          // Pull-up, Drive in standard strength, Sense Low input
```

Lab-4

```
loop:

 BL TURN_OFF_LED1
 BL WAIT_BTN1_PUSH

 BL TURN_ON_LED1
 BL WAIT_BTN1_PUSH

 B loop

/*delay_loop:
  LDR R2, = // loop counter for 1 sec with 64MHz system clock
count_down:
  CMP R2, #0
  ITT NE
  SUBNE R2, R2, #1
  BNE count_down
  MOV PC, LR*/
```

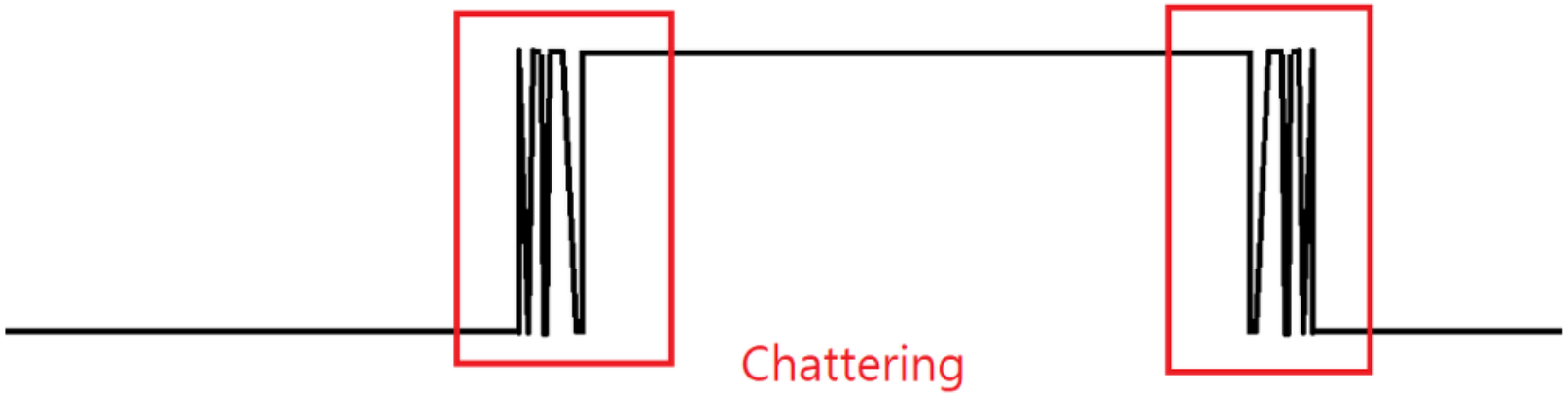
Lab-4

```
TURN_ON_LED1:
    LDR R0, =GPIO_P0_BASE+ GPIO_OUT_REG_OFFSET // OUT register (for my board LED, 0:ON, 1:OFF)
    LDR R1, =LED1_MASK
    MVN R1, R1      // complement
    STR R1, [R0]
    MOV PC, LR

TURN_OFF_LED1:
    LDR R0, =GPIO_P0_BASE+ GPIO_OUT_REG_OFFSET // OUT register
    LDR R1, =LED1_MASK
    STR R1,[R0]
    MOV PC, LR


WAIT_BTN1_PUSH:
      LDR R0, =GPIO_P0_BASE
 WAIT_LOOP:
      LDR R1, [R0, #GPIO_IN_REG_OFFSET]
      MVN R1,R1
      TST R1, #BTN1_MASK
      IT EQ                  // IF NOT PUSHED
      BEQ WAIT_LOOP
      MOV PC, LR        // return
```

Lab-4

The Previous Code works incompletely because of chattering problem.

- Chattering Problem
  - A phenomenon in which the switch contacts in the electronic circuit are repeatedly switched on and off in a very short time due to mechanical vibration at the moment of closing or opening switch



Lab-4

Solve the chattering problem by putting a small (100ms) delay

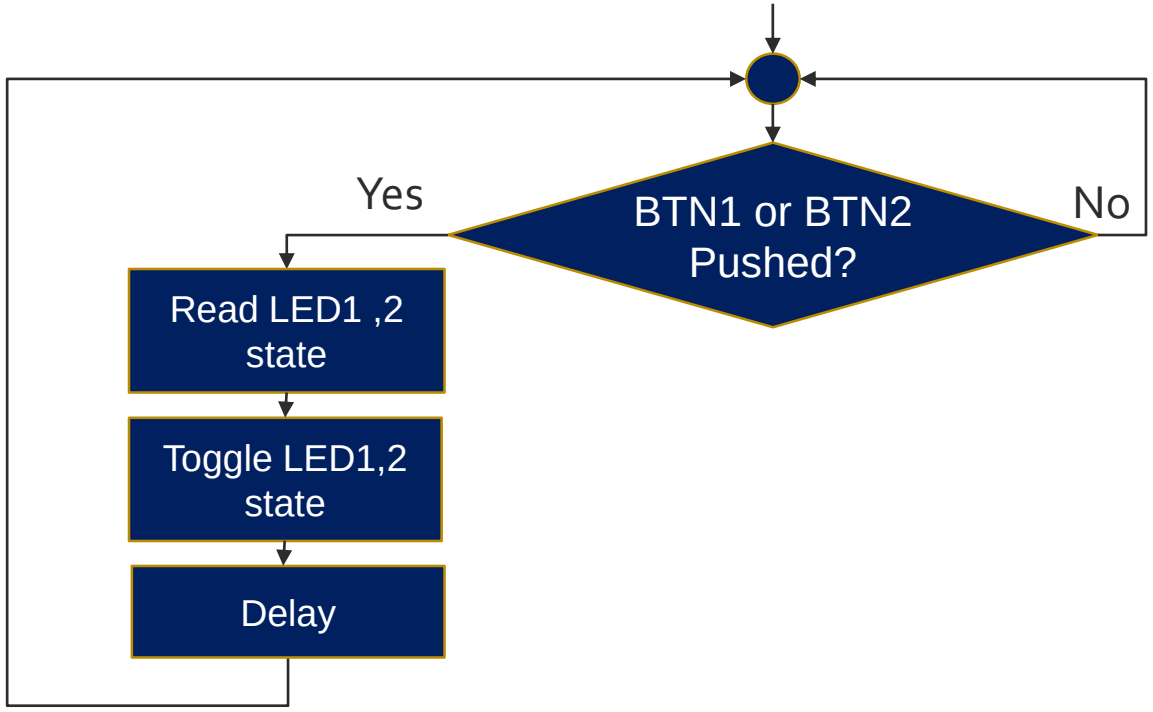
```
loop:

 BL TURN_OFF_LED
 BL Delay
 BL WAIT_INPUT_PUSH

 BL TURN_ON_LED
 BL Delay
 BL WAIT_INPUT_PUSH

 B loop

Delay:
  LDR R2, =1600000
count_down:
  CMP R2, #0
  ITT NE
  SUBNE R2, R2, #1
  BNE count_down
  MOV PC, LR
```

Agenda

1. Install SEGGER Embedded Studio
2. Basic setting
3. GPIO
4. Lab-1
5. Lab-2
6. Lab-3
7. **Assignment**

Assignments (1)

1. OUTSET, OUTCLR을 사용해서 LED1번과 LED2번이 1초 간격으로 번갈아 깜박이게 하라.
2. LED마스크에 shift 연산 사용해서 1초 간격으로 LED1→ LED2 → LED3 → LED4 순으로 반복해서 LED가 켜지도록 하라(1000 → 0100 → 0010 → 0001 → 1000 → …)
3. Lab-4의 코드를 수정하여 Button1 또는 Button2를 누를 때마다 LED1과 LED2 가 동시에 ON-OFF가 토글이 되게하라. (아래 flowchart 참조)



Assignments (2)

4. Lab4 코드를 다음 2가지 요구사항을 반영하여 수정하라

(1) Delay 함수를 100ms가 아닌 1sec를 delay하도록 변경하고

(2) 4개 버튼 중 하나를 누를 때마다 전체 4개 LED 의 현재 상태를 읽어서 눌러진 버튼에 해당하는 LED 값만 반전시켜서 LED 로 출력하도록 코드를 수정하라. (BTNn (n=1..4) button 은 LEDn (n=1..4)을 반전시킴)