



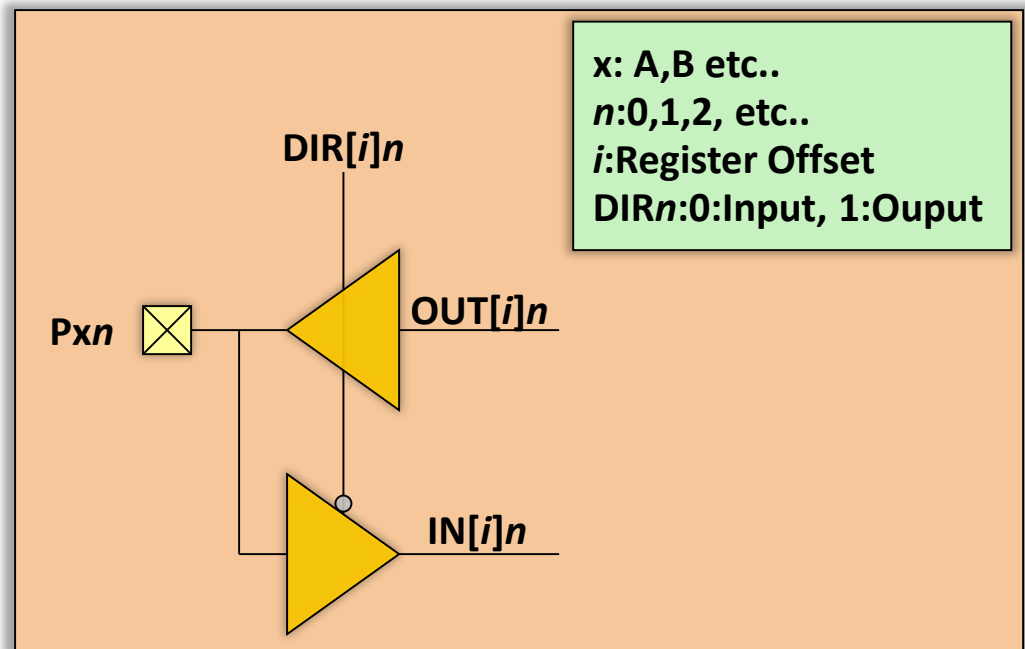
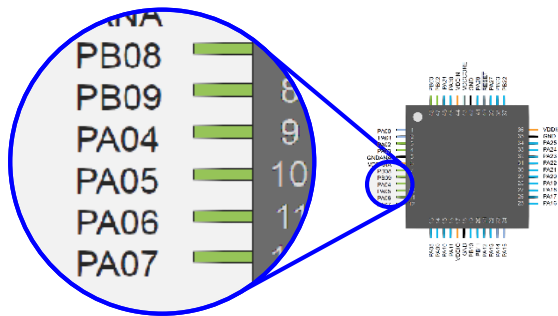
MICROCHIP

Regional Training Centers

Section 5
PORT I/O Architecture

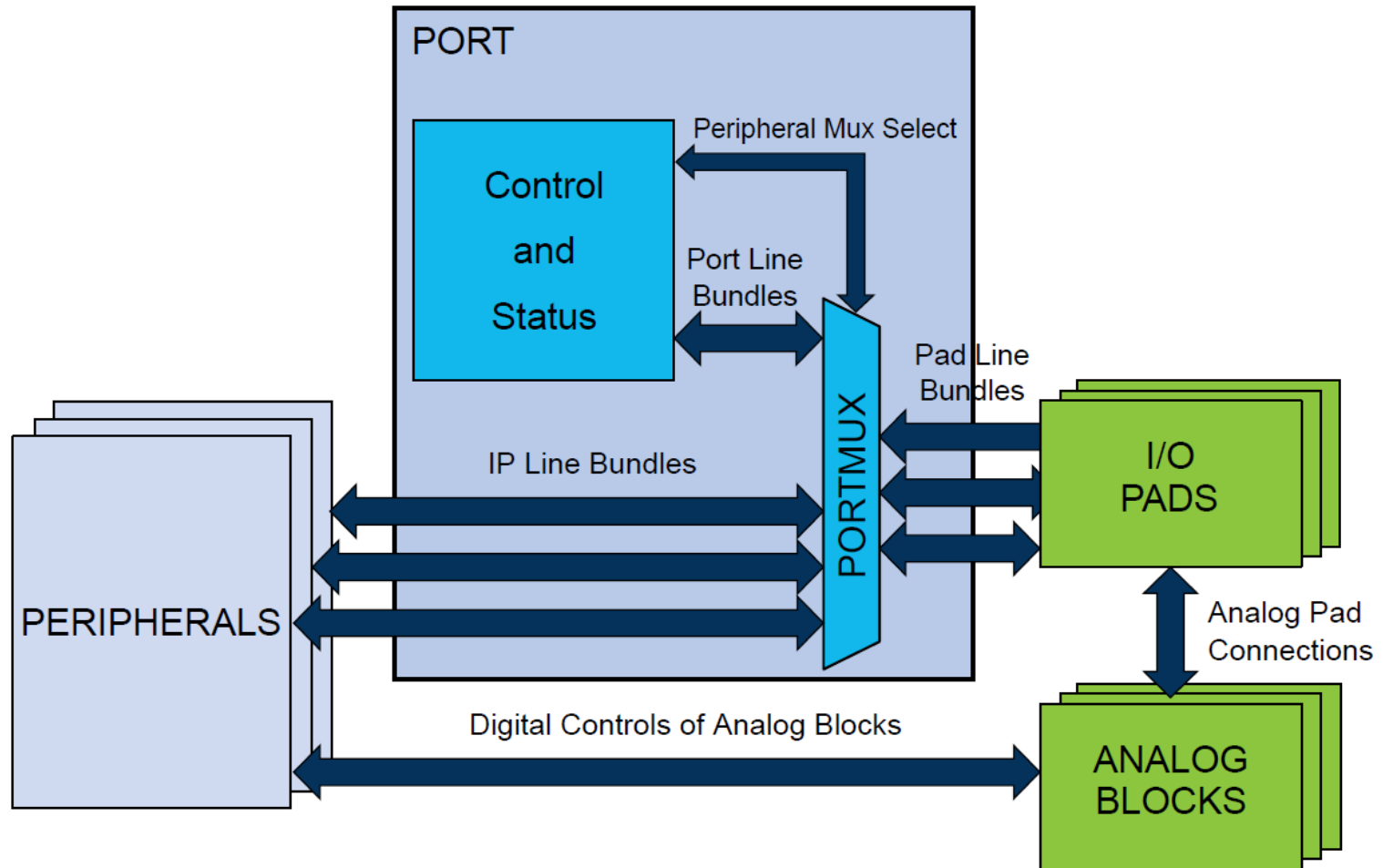
PORT Block Diagram

- ❖ Please refer to the block diagram, this is the concept for programming model. The real and detail block diagram please refer to the following slide.
- ❖ **DIR:** use to determine in or out. 0 for input, 1 for output.
- ❖ **OUT:** set output drive value for Output pins.
- ❖ **IN:** reaction logical level from Input pins.



PORT Block Diagram

■ SAMD21G18A PORT Block Diagram



Register Access

AHB-APB Bridge B

0x41000000
0x41002000
0x41004000
0x41004400
0x41004800
0x41005000
0x41006000
0x41007000

PAC1
DSU
NVMCTRL
PORT
DMAC
USB
MTB
Reserved

PORTA	DIR	0x41004400
	DIRCLR	0x41004404
	DIRSET	0x41004408
	DIRTGL	0x4100440C
	OUT	0x41004410
	OUTCLR	0x41004418
PORTB	...	
	DIR	0x41004480
	DIRCLR	0x41004484
	DIRSET	0x41004488
	DIRTGL	0x4100448C
	OUT	0x41004490
	OUTCLR	0x41004494
	...	0x41004498

Register Summary from datasheet

Offset	Name	Bit Pos.
0x00	DIR	7:0
0x01		15:8
0x02		23:16
0x03		31:24
0x04	DIRCLR	7:0
0x05		15:8
0x06		23:16
0x07		31:24
0x08	DIRSET	7:0
0x09		15:8
0x0A		23:16
0x0B		31:24
0x0C	DIRTGL	7:0
0x0D		15:8
0x0E		23:16
0x0F		31:24
0x10	OUT	7:0
0x11		15:8
0x12		23:16
0x13		31:24
0x14	OUTCLR	7:0
0x15		15:8
0x16		23:16
0x17		31:24

Register Access

❖ There are two way to access Register:

📖 Access by Pointer

```
*(__IO uint32_t *) (0x41004400U) = 0x55; /* 0x41004400U : PORTA DIR */
```

```
*(__IO uint32_t *) (0x41004480U) = 0x55; /* 0x41004480U : PORTB DIR */
```

📖 Access by Pre-defined struct

```
PORT_REGS->GROUP[0].PORT_DIR = 0x55; /* Group[0] : PORTA Register Set */
```

```
PORT_REGS->GROUP[1].PORT_DIR = 0x55; /* Group[1] : PORTB Register Set */
```

atsamd21g18a.h

```
#define PORT_REGS ((port_registers_t*)0x41004400)
```

port.h

```
typedef struct
```

```
{ /* Port Module */
```

```
    port_group_registers_t GROUP[GROUP_NUMBER];
```

```
} port_registers_t;
```

port.h

```
typedef struct
```

```
{
```

```
    __IO uint32_t PORT_DIR;
```

```
    __IO uint32_t PORT_DIRCLR;
```

```
    __IO uint32_t PORT_DIRSET;
```

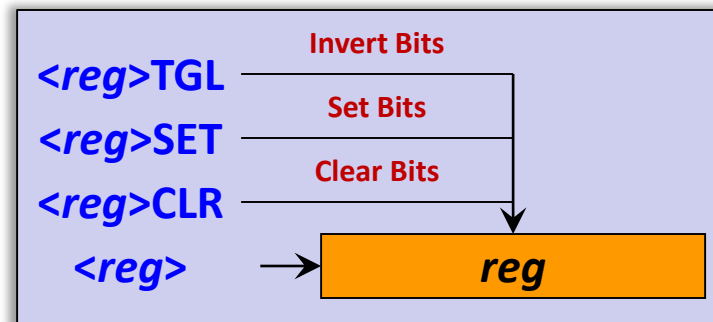
```
    __IO uint32_t PORT_DIRTGL;
```

```
    ...
```

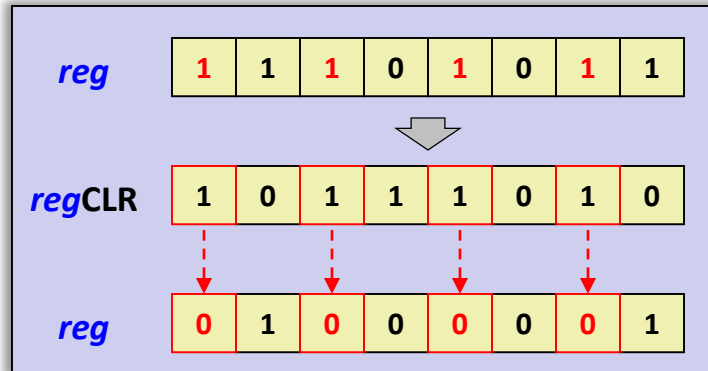
```
} port_group_registers_t;
```

Atomic Bits Manipulation

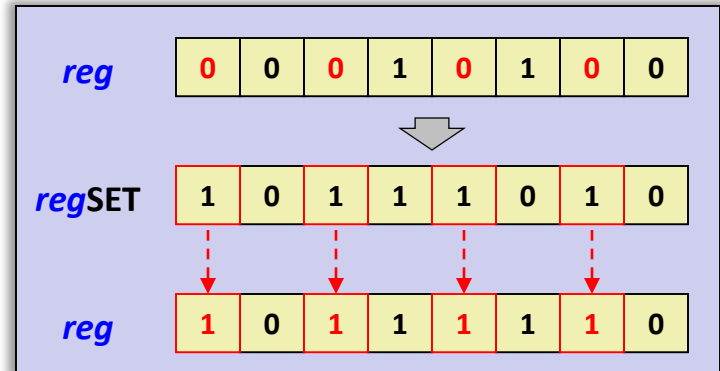
- Some register provide **<reg>CLR**, **<reg>SET** , **<reg>TGL** register for atomic bits manipulation.
- This register manipulated without doing a read-modify-write operation by using below
 - <reg> Toggle (<reg>TGL)**
 - <reg> Clear (<reg>CLR)**
 - <reg> Set (<reg>SET)**
- Setting the manipulation register bits corresponding to the base register will set the base bits to 1, clear to 0 or toggle.



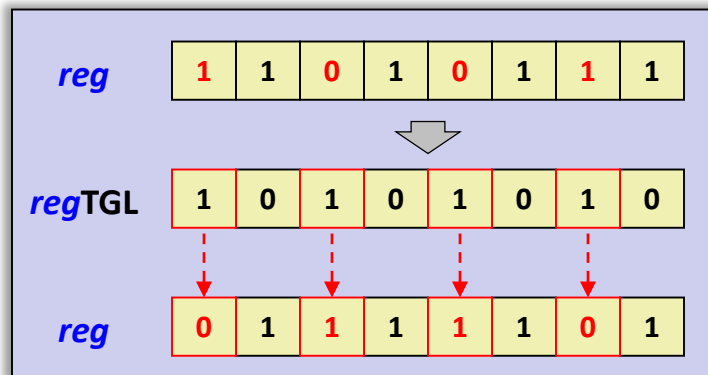
Atomic Bits Manipulation



regCLR : corresponding bits clear to 0



regSET : corresponding bits set to 1



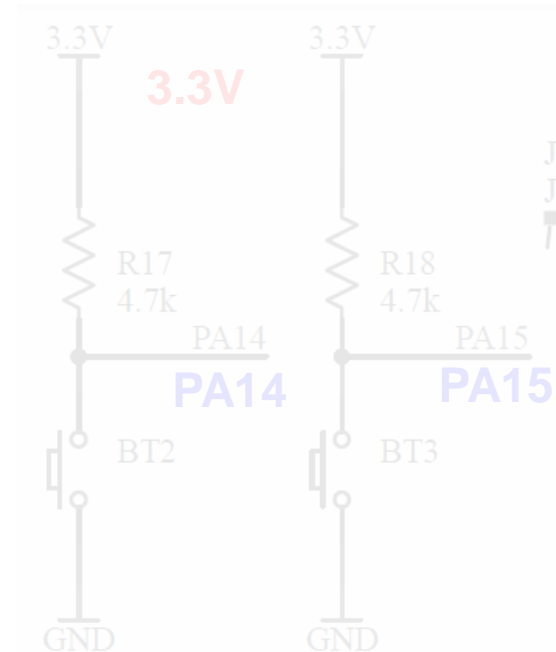
regTGL : corresponding bits toggled
(0 -> 1, 1 -> 0)

Lab1 PORT Output

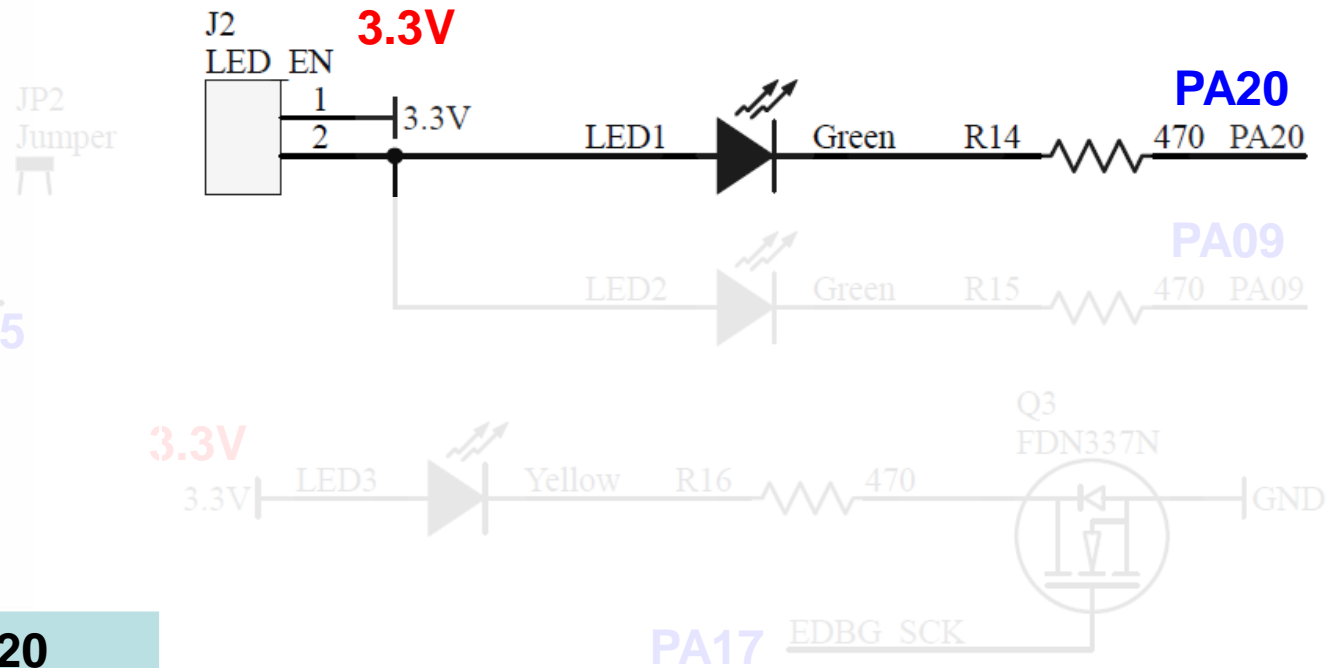
- Try to control LED1(PA20) continues toggle (period around 500ms ~ 1S).
- You can use **register pointer** or **pre-defined struct** to access PORT.

 **How to start ?**

Check Schematic



LED1	PA20
LED2	PA09
LED3	PA17
BT2	PA14
BT3	PA15



PA19	28	EDBG_MISO
PA18	27	EDBG_CS
PA17	26	EDBG_SCK
PA16	25	EDBG_MOSI

Lab1 PORT Output

Step 1

a Add code segment to your main loop

```
// TODO 1.01
uint32_t i = 0;

int main (void)
{
    SYS_Initialize ( NULL );

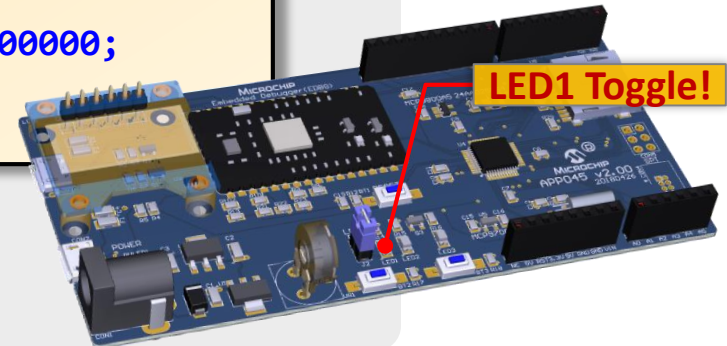
    // TODO 1.02
    *(uint32_t *) (0x41004400U + 0x08U) = 0x00100000;

    while(1)
    {
        for ( i = 0 ; i < 2000000 ; i++ )
        {
            // TODO 1.03
            *(uint32_t *) (0x41004400U + 0x1CU) = 0x00100000;
        }
    }
}
```

Period Control

LED1 Toggle!

b Program firmware to target board then observe result.



Lab1 PORT Output

Register Describe

0x00100000 (Hex)
0001 0000 0000 0000 0000 0000 (Binary)
2222 1111 1111 11
3210 9876 5432 1098 7654 3210

```
// TODO 1.01
uint32_t i = 0;

int main (void)
{
    SYS_Initialize ( NULL );

    // TODO 1.02
    *(uint32_t *) (0x41004400U + 0x08U) = 0x00100000;

    while(1)
    {
        for ( i = 0 ; i < 2000000 ; i++ )
        {}
        // TODO 1.03
        *(uint32_t *) (0x41004400U + 0x1CU) = 0x00100000;
    }
}
```

LED1 Pin PA20

Offset	Name
0x00	DIR
0x01	
0x02	
0x03	
0x04	DIRCLR
0x05	
0x06	
0x07	
0x08	DIRSET
0x09	
0x0A	
0x0B	
0x1C	OUTGL
0x1D	
0x1E	
0x1F	

Lab1 PORT Output

Step 2

a Modify code segment to your main loop

```
// TODO 1.01
uint32_t i = 0;

int main (void)
{
    SYS_Initialize ( NULL );

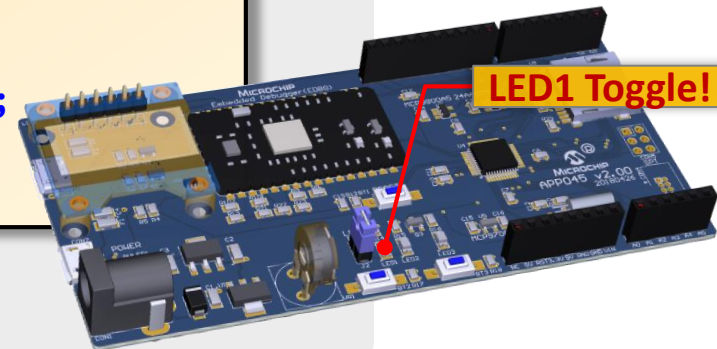
    // TODO 1.02
    PORT_REGS->GROUP[0].PORT_DIRSET = 1<<20;

    while(1)
    {
        for ( i = 0 ; i < 2000000 ; i++ )
        {}

        // TODO 1.03
        PORT_REGS->GROUP[0].PORT_OUTTGL = PORT_PA20;
    }
}
```

Period Control

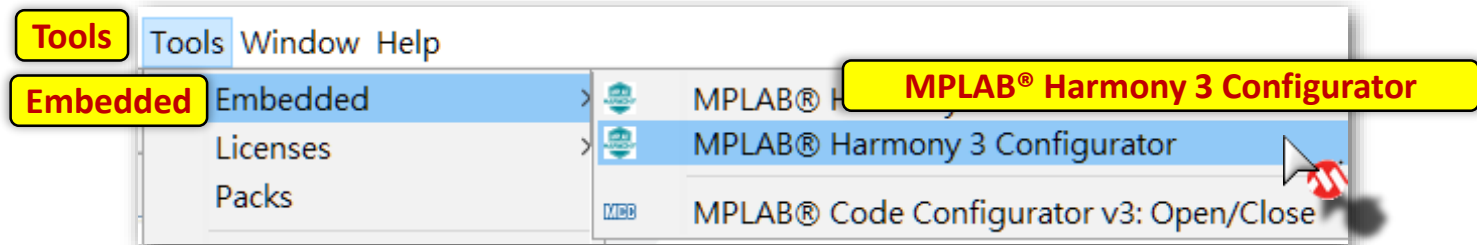
LED1 Toggle!



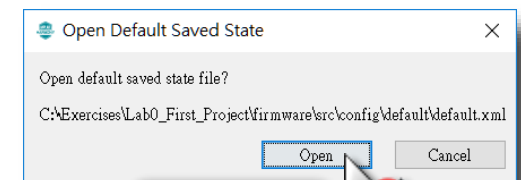
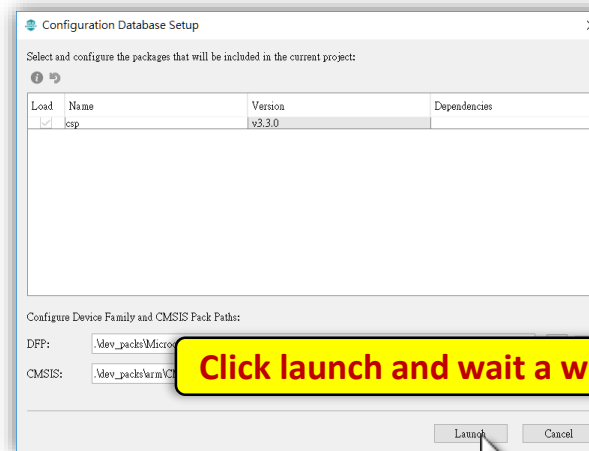
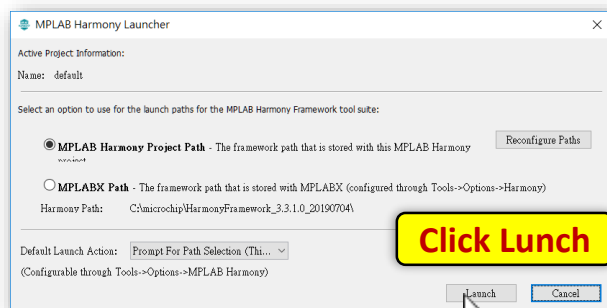
b Program firmware to target board then observe result.

Pin Configuration using MHC

- ❖ If you already close MHC, please open it through below steps.
- ❖ Select **Tools ► Embedded ► MPLAB® Harmony 3 Configurator**

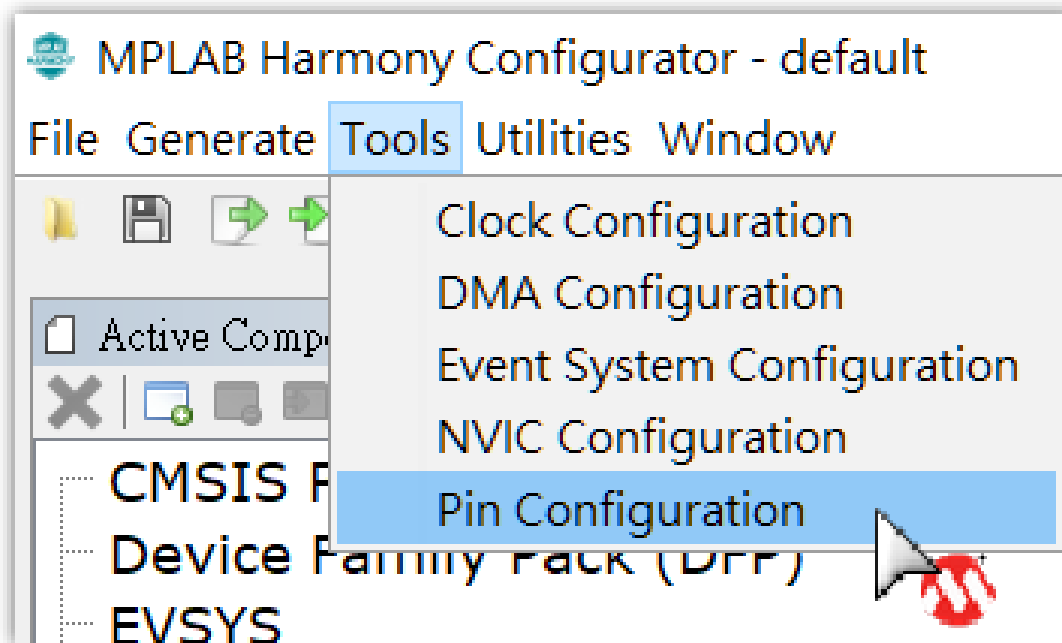


- ❖ Following MHC reopen steps reviews.



Pin Configuration using MHC

- Return to Harmony Configurator.
- Select **Tools ► Pin Configuration**



Pin Configuration using MHC

❖ Docking three **Pin Configuration** windows as below.

The screenshot displays the MPLAB Harmony Configurator interface with three windows docked for pin configuration:

- Pin Settings:** A table listing pins and their configurations. A yellow label "Pin Setting" points to this window.
- Pin Diagram:** A visual representation of the pin connections. A yellow label "Pin Diagram" points to this window.
- Pin Table:** A table mapping modules to pins. A yellow label "Pin Table" points to this window.

Pin Settings Table:

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down
1	PA00		Available	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
2	PA01		Available	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
3	PA02		Available	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
4	PA03		Available	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
5	GNDANA			Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
6	VDDANA			Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
7	PB08		Available	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
8	PB09			Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
9	PA04			Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
10	PA05		Available	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
11	PA06		Available	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
12	PA07		Available	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
13	PA08		Available	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
14	PA09	LED2	GPIO	Digital	Out	High	<input type="checkbox"/>	<input type="checkbox"/>
15	PA10		Available	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
16	PA11		Available	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
17	VDDIO			Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
18	GNDIO			Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>

Pin Table:

Module	Function	PA00	PA01	PA02	PA03	GNDAN	VDDAN	PB08	PB09	PA04	PA05	PA06	PA07	PA08	LED2	PA10	VDDIO	GNDIO	PB10	PB11	PA12	PA13	BT1	BT2	PA16	PA17	PA18	PA19	LED1	PA21	PA22	PA23	PA24	PA25	GNDIO	VDDIO	PB22	PB23	PA27	RESET	PA28	GNDIO	VDDIO	VDDIN	PA30	PA31	PB02	PB03			
AC	AC_AIN0																																																		
	AC_AIN1																																																		
	AC_AIN2																																																		
	AC_AIN3																																																		
	AC_CMP0																																																		
	AC_CMP1																																																		
	ADC_AIN0																																																		
	ADC_AIN1																																																		

Lab1 PORT Output

Step 3

Pin Table configure. (Enable or Disable Pin)

Pin Table

Package: TQFP48

PA20

Module	Function	PA11	VDDIO	GNDIO	PB10	PB11	PA12	PA13	PA14	PA15	PA16	PA17	PA18	PA19	PA20	PA21	PA22	PA23	PA24
	GCLK_IO7																		
GPIO	GPIO																		
GPIO	I2S_FS0																		
	I2S_MCKO																		

Click Available

Pin Table

Package: TQFP48

Module	Function	PA11	VDDIO	GNDIO	PB10	PB11	PA12	PA13	PA14	PA15	PA16	PA17	PA18	PA19	GPIO_P	PA21	PA22	PA23	PA24
	GCLK_IO7																		
GPIO	GPIO																		
	I2S_FS0																		

Green is Enable

Lab1 PORT Output

Step 4

Pin Setting. (Pin Function Setting)

Pin Settings

Order: Pins Table View ☒ Easy View

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down
20	PA17		Available	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
27	PA18		Available	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
28	PA19		Available	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
29	PA20	GPIO_PA20	GPIO	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
30	PA21		Available	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>

Always remember press [Enter] after any value input in MHC

Pin Settings

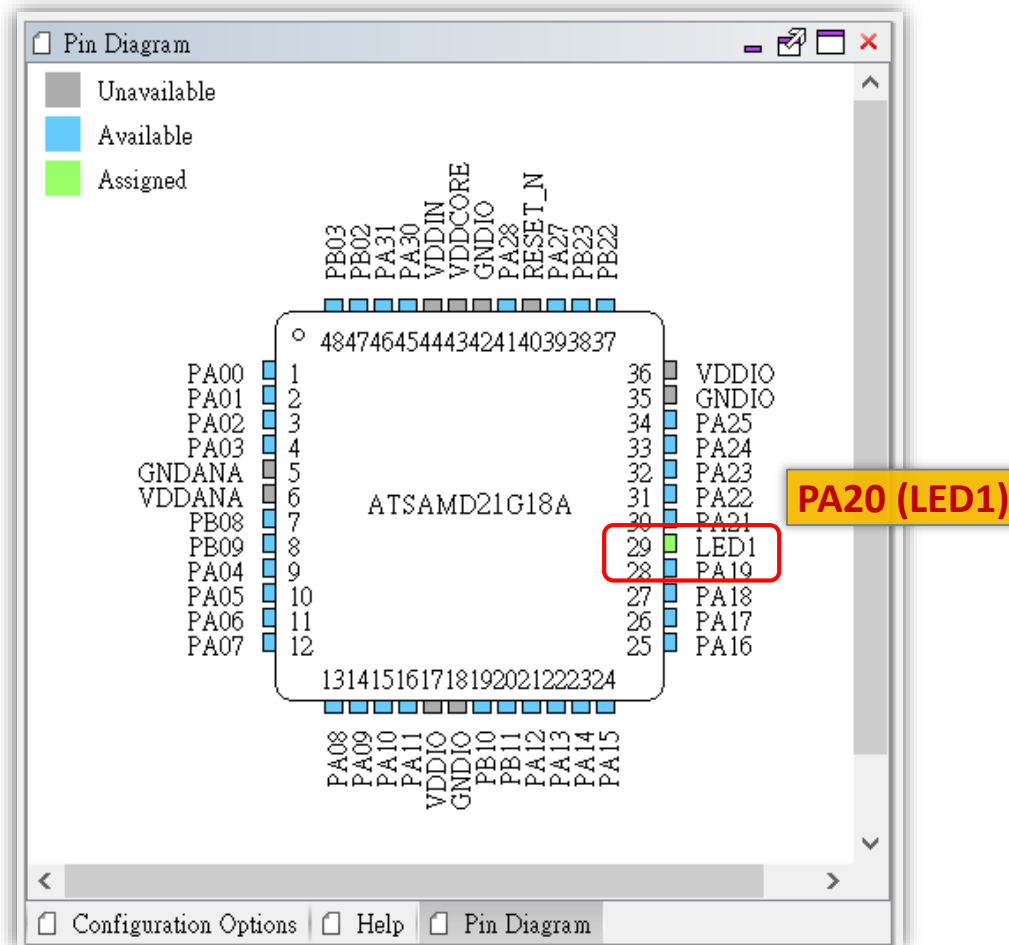
Easy View

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down
20	PA17		Available	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
27	PA18	LED1	Available	Digital	Out	High	<input type="checkbox"/>	<input type="checkbox"/>
28	PA19		Available	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>
29	PA20	LED1	GPIO	Digital	Out	High	<input type="checkbox"/>	<input type="checkbox"/>
30	PA21		Available	Digital	High I...	Low	<input type="checkbox"/>	<input type="checkbox"/>

Lab1 PORT Output

Step 5

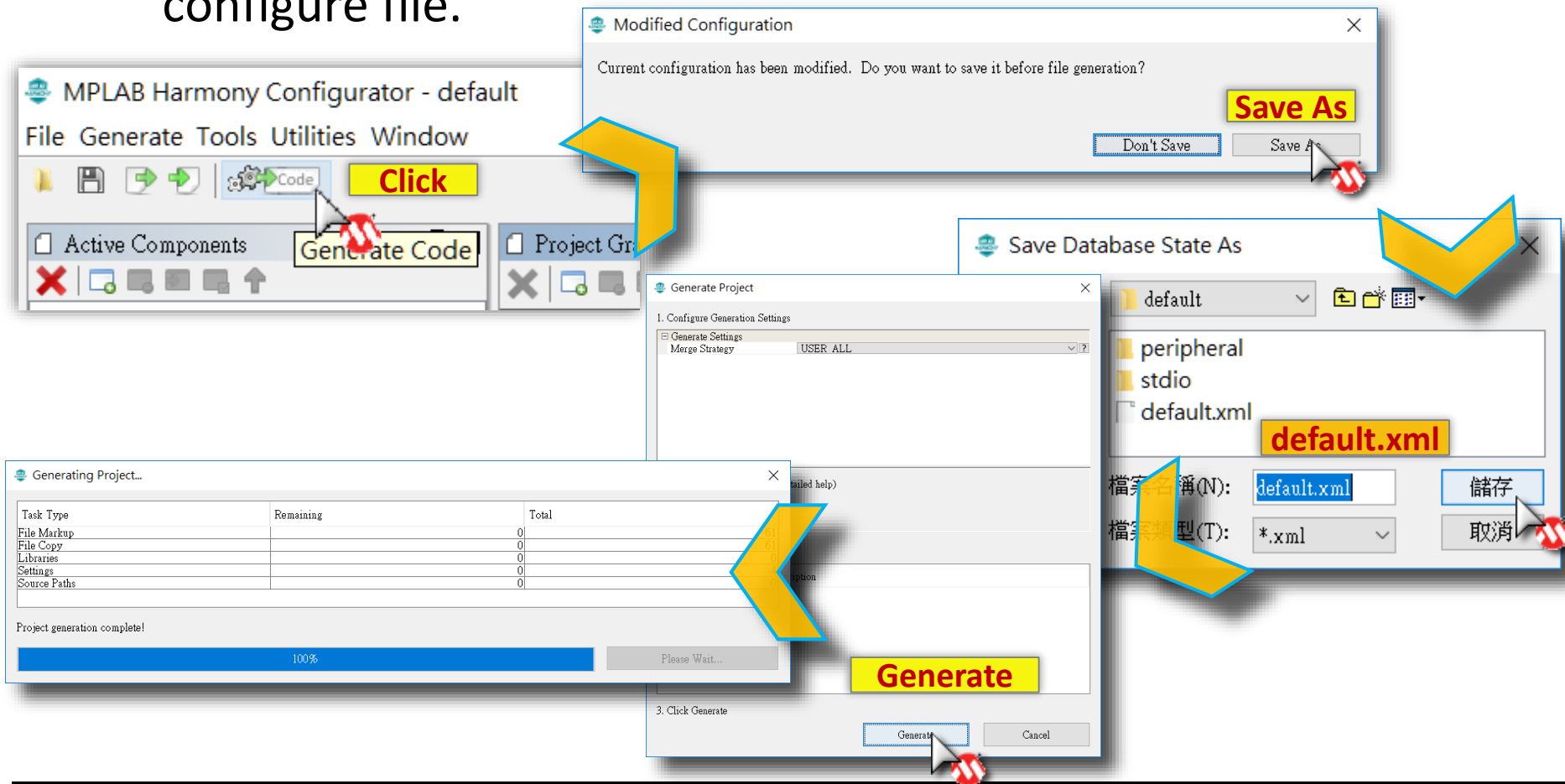
Pin Diagram. (Hardware design reference)



Lab1 PORT Output

Step 6

- Click  to Generate Code and save changes to MHC configure file.



MHC GPIO Macros

Plib_port.h

/** Macros for LED1 pin **/

```
#define LED1_Set()    (PORT_REGS->GROUP[0].PORT_OUTSET = 1 << 20)
#define LED1_Clear() (PORT_REGS->GROUP[0].PORT_OUTCLR = 1 << 20)
#define LED1_Toggle()      (PORT_REGS->GROUP[0].PORT_OUTTGL = 1 << 20)
#define LED1_Get()    (((PORT_REGS->GROUP[0].PORT_IN >> 20) & 0x01)
#define LED1_OutputEnable() (PORT_REGS->GROUP[0].PORT_DIRSET = 1 << 20)
#define LED1_InputEnable()  (PORT_REGS->GROUP[0].PORT_DIRCLR = 1 << 20)
#define LED1_PIN            PORT_PIN_PA20
```

Plib_port.c

void **PORT_Initialize**(void)

```
{
    /******* GROUP 0 Initialization *****/
    PORT_REGS->GROUP[0].PORT_DIR = 0x100000;
    PORT_REGS->GROUP[0].PORT_OUT = 0x100000;

    /******* GROUP 1 Initialization *****/
}
```

Lab1 PORT Output

Step 7

a Modify code segment to your main loop

```
// TODO 1.01
uint32_t i = 0;

int main (void)
{
    SYS_Initialize ( NULL );

    // TODO 1.02
    //PORT_REGS->GROUP[0].PORT_DIRSET = 1<<20;

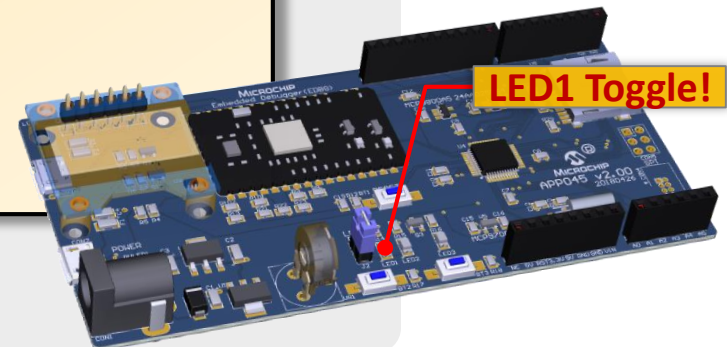
    while(1)
    {
        for ( i = 0 ; i < 2000000 ; i++ )
        {}

        // TODO 1.03
        LED1_Toggle();
    }
}
```

PORT initialization already down here

Comment out this line

b Program firmware to target board then observe result.



MPLAB X IDE Shortcut key

- ❖ **MPLAB X IDE Provide a few shortcut key to enhance your performance.**

📁 **Ctrl + Alt + ** : Show All Code Completion
(Ctrl + Alt + Space, Alt + \ : Alternative shortcut)

📁 **Ctrl + Mouse Left Click** : Open Macro

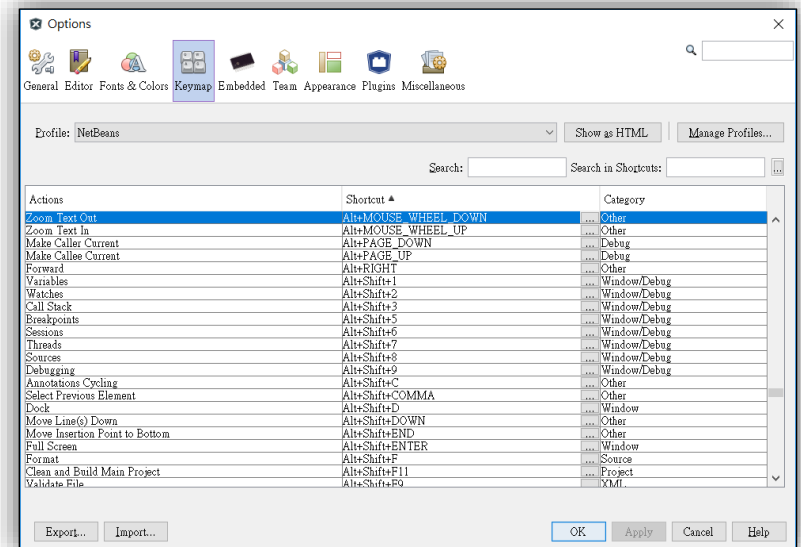
📁 **Alt + Left** : Backward

📁 **Alt + Shift + F** : Code Format

📁 **Ctrl + Shift + C** : Toggle Comment

- ❖ **More ?**

📁 Tools > Options > Keymap

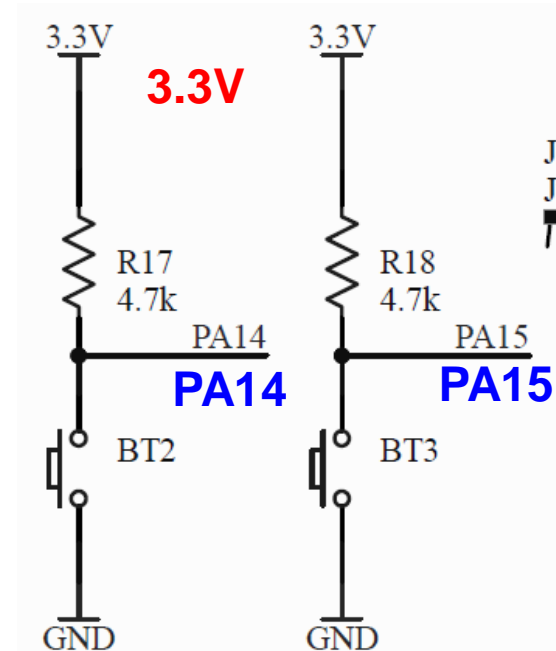


Lab2 PORT Input and Output

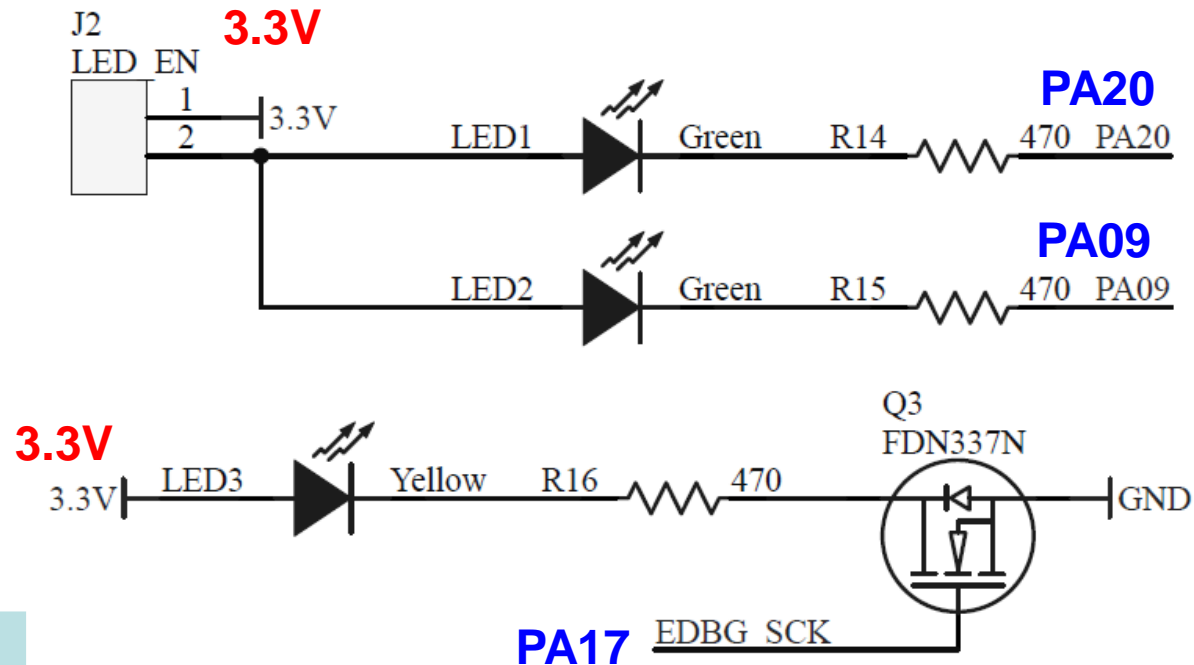
- Base On Lab1, Try to add LED2(PA09) continues toggle (Period around 500ms ~ 1S).
- Get BT2(PA14) status then use to control LED3(PA17).
Pressed -> LED3 Light
Released -> LED3 Dark

■ **Let's go!**

Check Schematic



LED1	PA20
LED2	PA09
LED3	PA17
BT2	PA14
BT3	PA15



PA17

PA19	28	EDBG MISO
PA18	27	EDBG CS
PA17	26	EDBG SCK
PA16	25	EDBG MOSI

Lab2 PORT Input and Output

Step 1

- Pin Table configure. (Enable or Disable Pin)

Pin Table

Package: TQFP48

Module	Function	PA07	PA08	PA09	PA10	PA11	VDDIO	GNDIO	PB10	PB11	PA12	PA13	PA14	PA15	PA16	PA17	PA18	PA19	PA20	PA21
	GCLK_IO7																			
GPIO	GPIO																			
	IOCON																			

Click

Lab2 PORT Input and Output

Step 2

Pin Setting. (Pin Function Setting)

Pin Settings								
Order: Pins		Table View		<input checked="" type="checkbox"/> Easy View				
Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down
PA09	PA09	LED2	LED2	Digital	Out	Low	Out	Low
15	PA10		Available	Digital	High I...	Low		
16	PA11		Available	Digital	High I...	Low		
17	VDDIO			Digital	High I...	Low		
18	GNDIO			Digital	High I...	Low		
19	PB10		Available	Digital	High I...	Low		
20	PB11		Available	Digital	High I...	Low		
21	PA12		Available	Digital	High I...	Low		
22	PA13		Available	Digital	High I...	Low		
PA14	PA14	BT2	BT2	Digital	In	Low	In	
PA15	PA15	BT3	BT3	Digital	In	Low	In	
25	PA16		Available	Digital	High I...	Low		
PA17	PA17	LED3	LED3	Digital	Out	Low	Out	Low
27	PA18		Available	Digital	High I...	Low		
28	PA19		Available	Digital	High I...	Low		
PA20	PA20	LED1	LED1	Digital	Out	High	Out	High

Lab2 PORT Input and Output

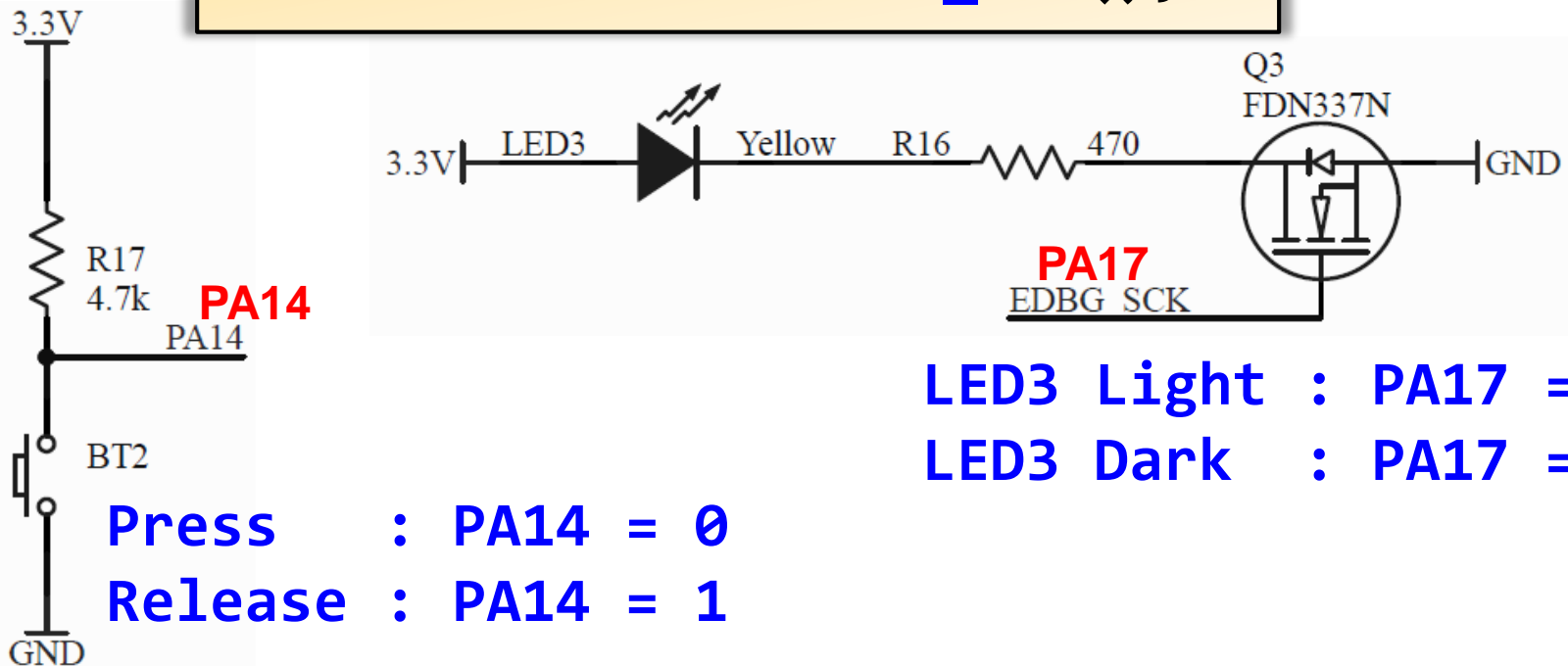
Schematic Describe

- Get **BT2(PA14)** status then use to control **LED3(PA17)**.

Pressed -> LED3 Light

Released -> LED3 Dark

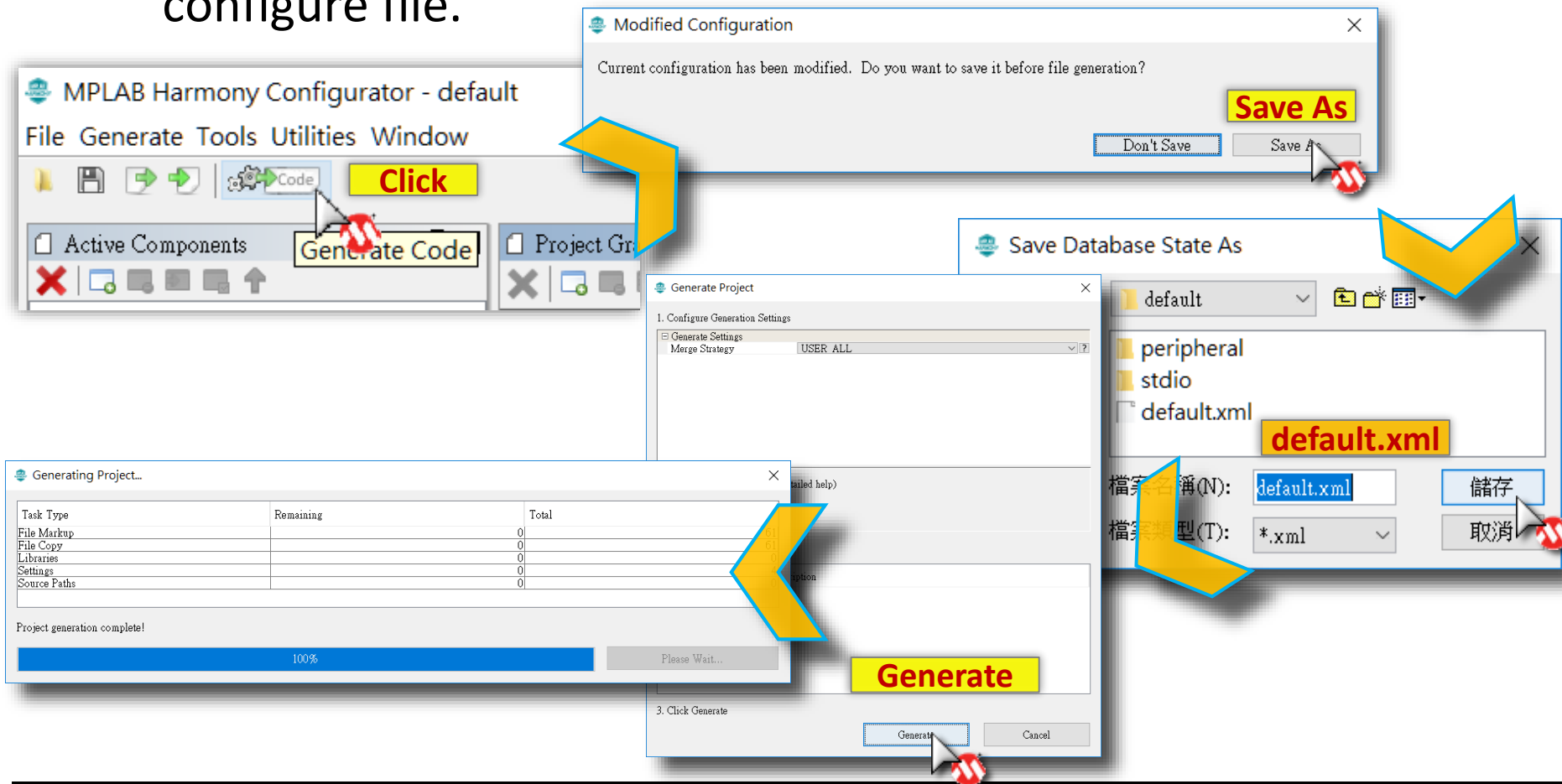
```
if ( BT2_Get() ) LED3_Clear();  
else  
    LED3_Set();
```



Lab2 PORT Input and Output

Step 3

- Click  to Generate Code and save changes to MHC configure file.



Lab2 PORT Input and Output

Step 4

a Add code segment to your main loop

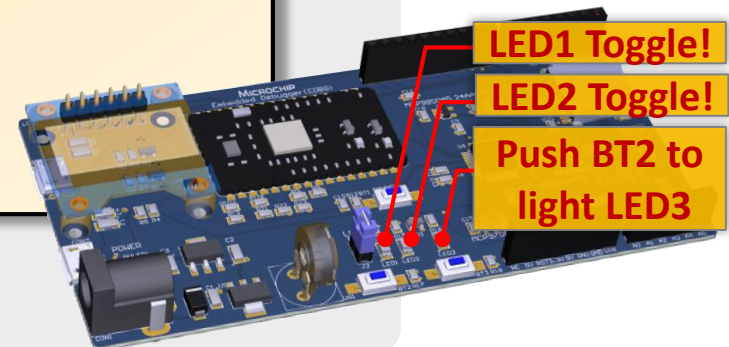
```
uint32_t i = 0;
int main (void)
{
    SYS_Initialize ( NULL );

    while(1)
    {
        for ( i = 0 ; i < 2000000 ; i++ )
        {}

        // TODO 2.01
        LED1_Toggle();
        LED2_Toggle();

        // TODO 2.02
        if ( BT2_Get() ) LED3_Clear();
        else             LED3_Set();
    }
}
```

b Program firmware to target board then observe result.



LED1 Toggle!

LED2 Toggle!

Push BT2 to
light LED3

Lab2 PORT Input and Output

Button Response Discuss

❖ Button Response Time Issue ?

The software delay will influences the polling interval.
Try to change coding style to state machine mode.

```
while(1)
{
    for ( i = 0; i < 2000000 ; i++ )
    {}

    LEDs_Toggle();

    if ( Button_Detected() )
        LED_Light();
    else
        LED_Dark();
}
```

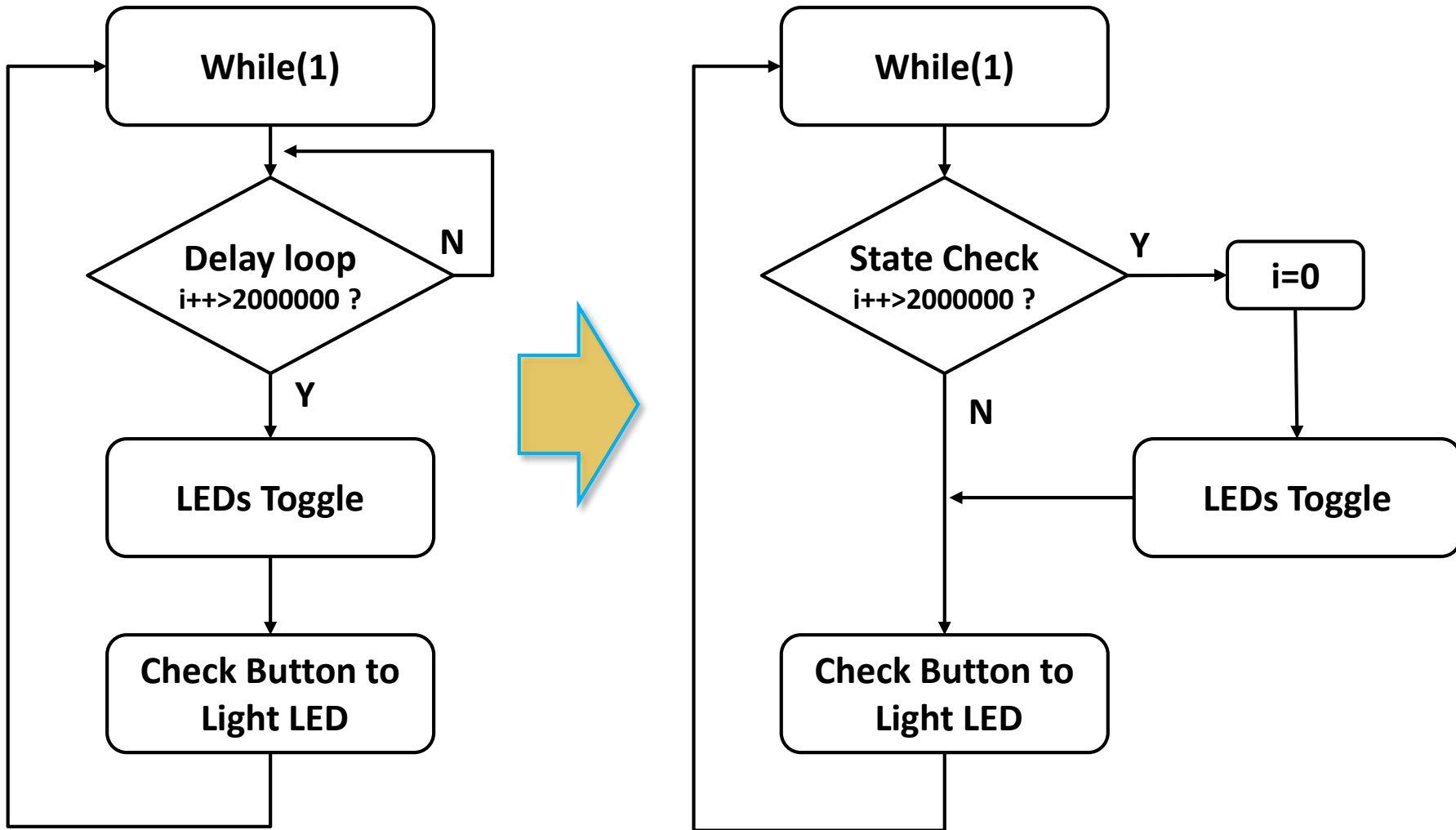


```
while(1)
{
    if ( i++ > 2000000 )
    {
        i = 0;

        LEDs_Toggle();
    }

    if ( Button_Detected() )
        LED_Light();
    else
        LED_Dark();
}
```

State Machine



Lab2 PORT Input and Output

Step 5

a change coding style to state machine mode

```
uint32_t i = 0;
int main (void)
{
    SYS_Initialize ( NULL );

    while(1)
    {
        if ( i++ > 2000000 )
        {
            i = 0;

            // TODO 2.01
            LED1_Toggle();
            LED2_Toggle();
        }

        // TODO 2.02
        if ( BT2_Get() ) LED3_Clear();
        else             LED3_Set();
    }
}
```

**How about current LED toggle speed after
change to state machine flow control ?
Why ?**

Bonus Lab

- Press BT3(PA15) to toggle below scenario:
- State 0 :
 - ▢ LED1 / LED2 toggle mutually.
 - ▢ BT2(PA14) Pressed -> LED3 Light, Released -> LED3 Dark
- State 1 :
 - ▢ LED2 / LED3 toggle mutually.
 - ▢ BT2(PA14) Pressed -> LED1 Light, Released -> LED1 Dark

■ **Let's go!**