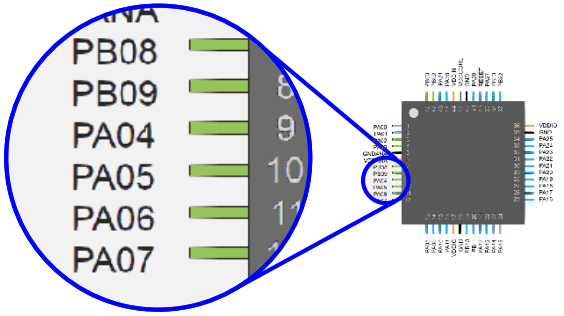
MICROCHIP

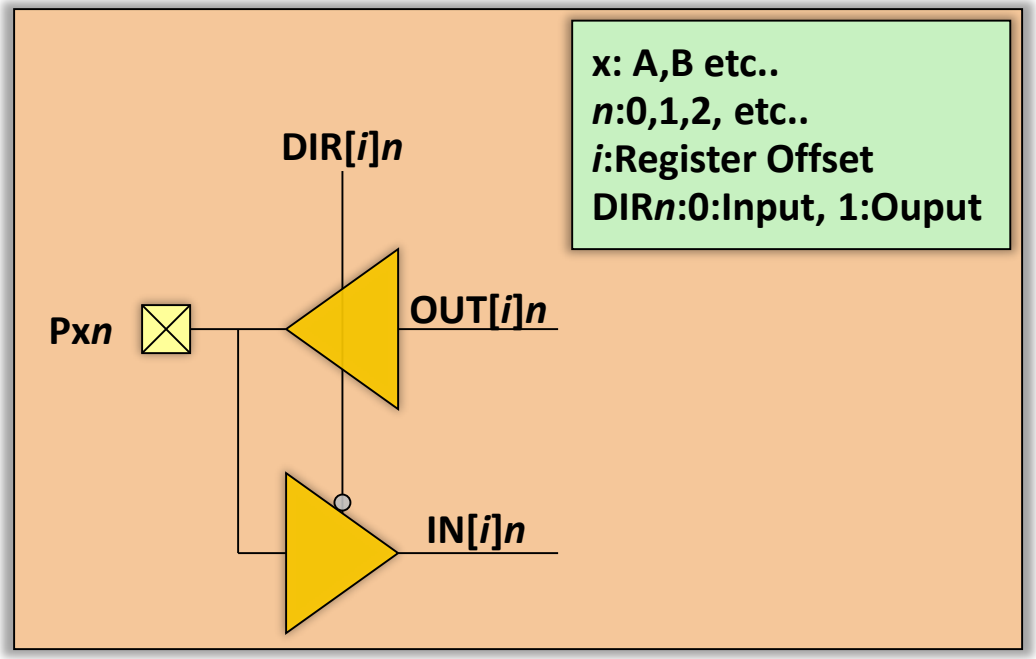
Regional Training Centers

Section 5
PORT I/O Architecture

PORT Block Diagram

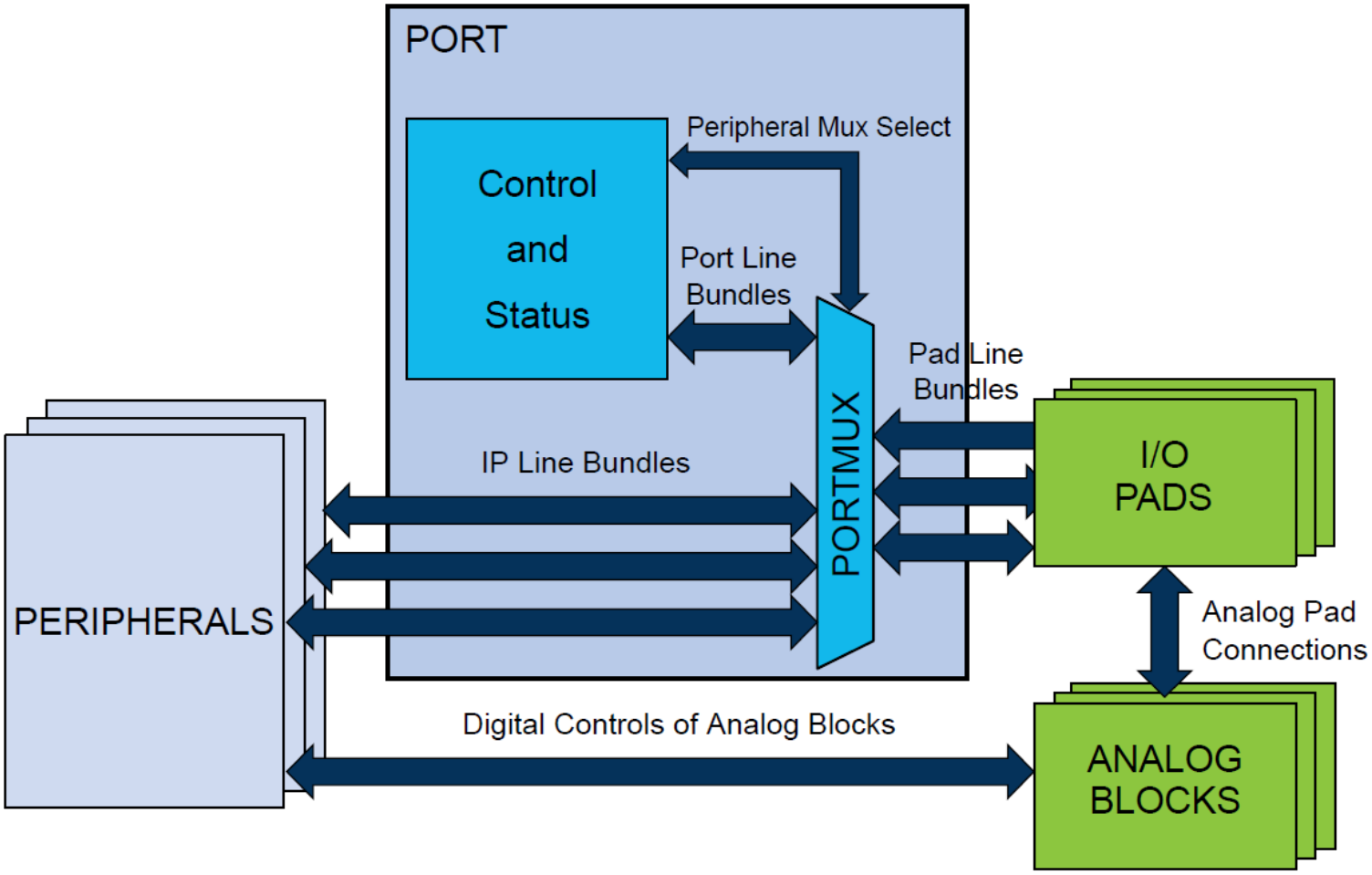
- Please refer to the block diagram, this is the concept for programming model. The real and detail block diagram please refer to the following slide.
- DIR: use to determine in or out. 0 for input, 1 for output.
- OUT: set output drive value for Output pins.
- IN: reaction logical level from Input pins.



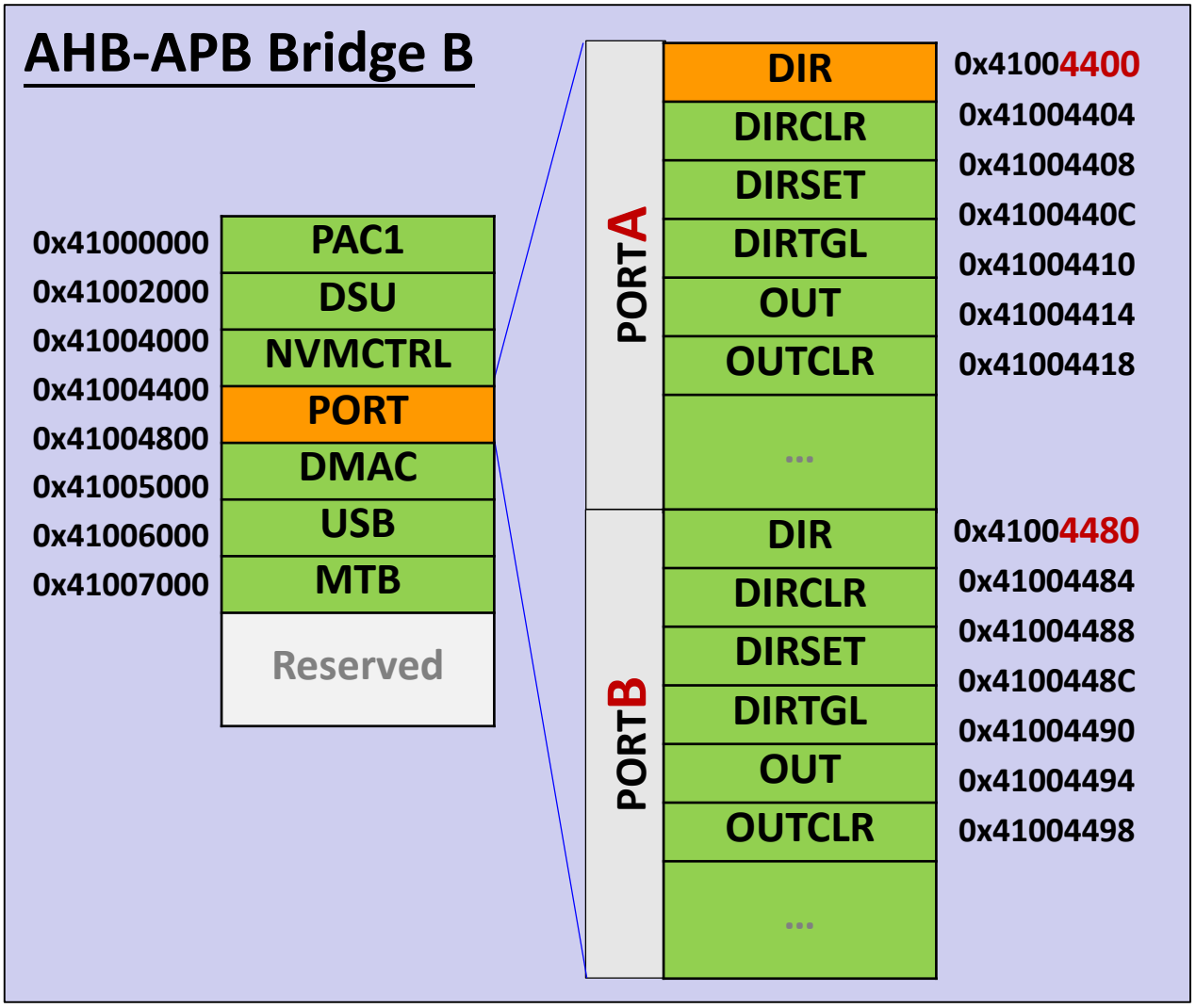


PORT Block Diagram

- **SAMD21G18A PORT Block Diagram**



Register Access



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

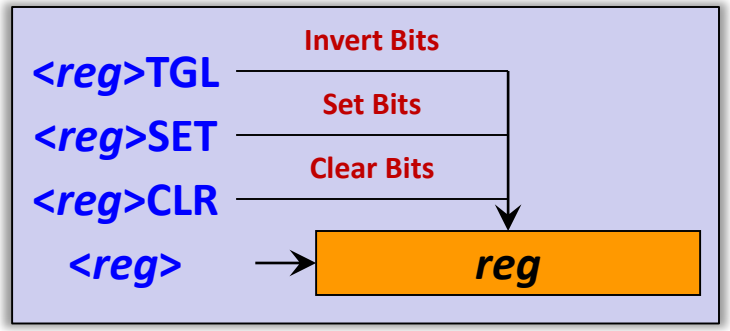
```
typedef struct
{  /* Port Module */
      port_group_registers_t    GROUP[GROUP_NUMBER];
} port_registers_t;
```

port.h

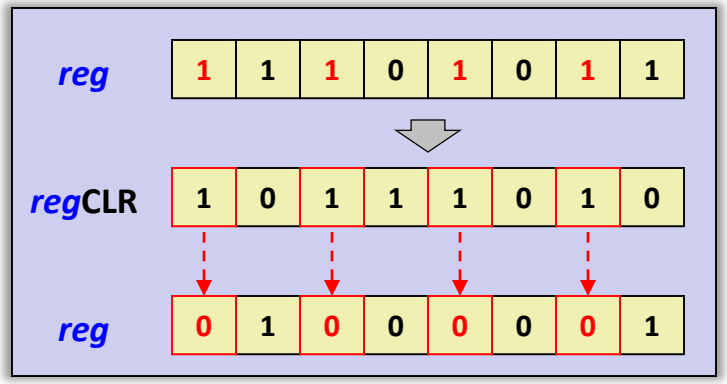
```
typedef struct
{
  __IO  uint32_t    PORT_DIR;
  __IO  uint32_t    PORT_DIRCLR;
  __IO  uint32_t    PORT_DIRSET;
  __IO  uint32_t    PORT_DIRTGL;
  ...
} port_group_registers_t;
```

Atomic Bits Manipulation

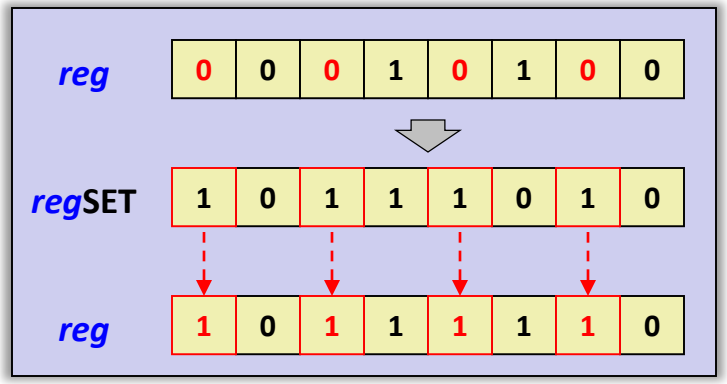
- **Some register provide <*reg*>CLR, <*reg*>SET , <*reg*>TGL register for atomic bits manipulation.**
- **This register manipulated without doing a read-modify-write operation by using below**
  - <*reg*> Toggle (<*reg*>TGL)
  - <*reg*> Clear (<*reg*>CLR)
  - <*reg*> Set (<*reg*>SET)
- **Setting the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



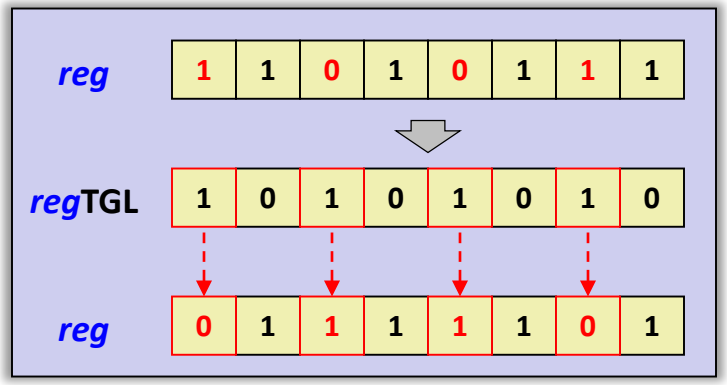
Atomic Bits Manipulation



*reg*CLR : corresponding bits clear to 0



*reg*SET : corresponding bits set to 1



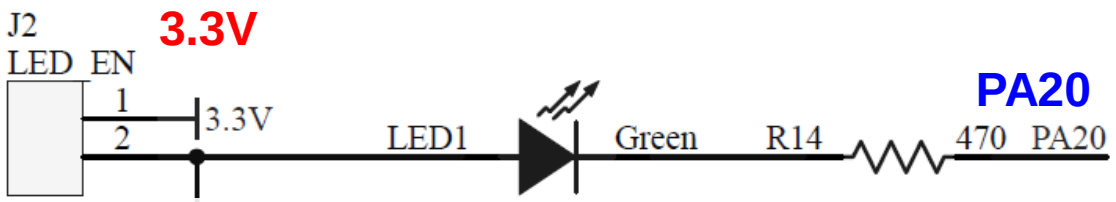
*reg*TGL : corresponding bits toggled
(0 -> 1, 1 -> 0)

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

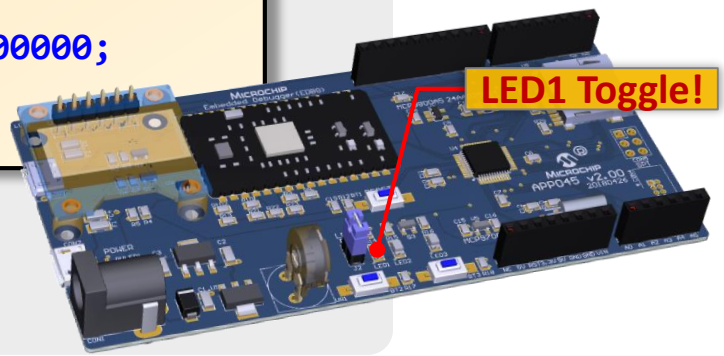
Lab1 PORT Output

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



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	OUTTGL
0x1D	
0x1E	
0x1F	

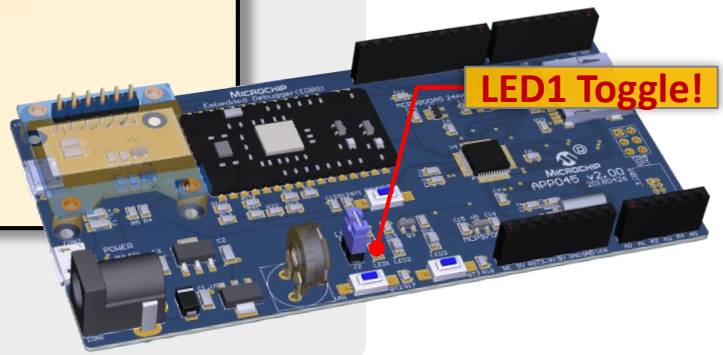
Lab1 PORT Output

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;
   }
}
```

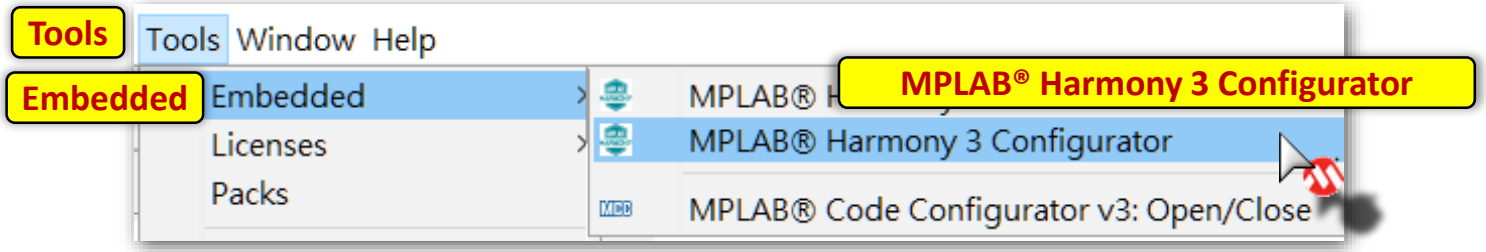
Period Control



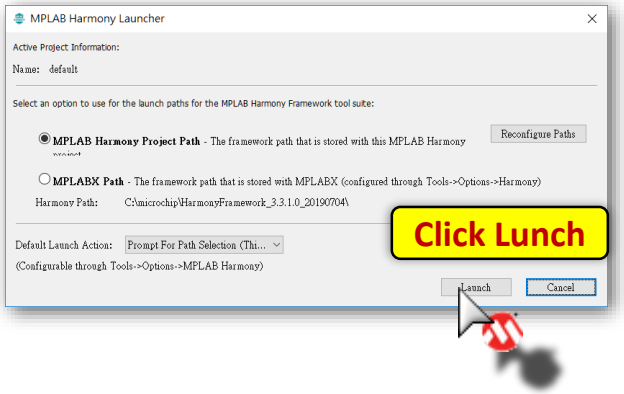
b **Program firmware to target board then observe result.**

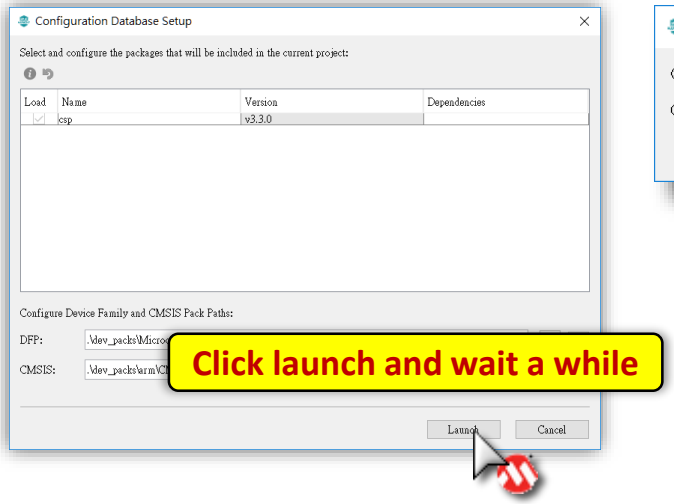
Pin Configuration using MHC

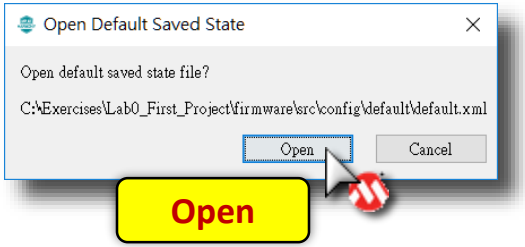
- If you already close MHC, please open it through below steps.
- Select **Tools ▸ Embedded ▸ MPLAB® Harmony 3 Configurator**



- Following MHC reopen steps reviews.

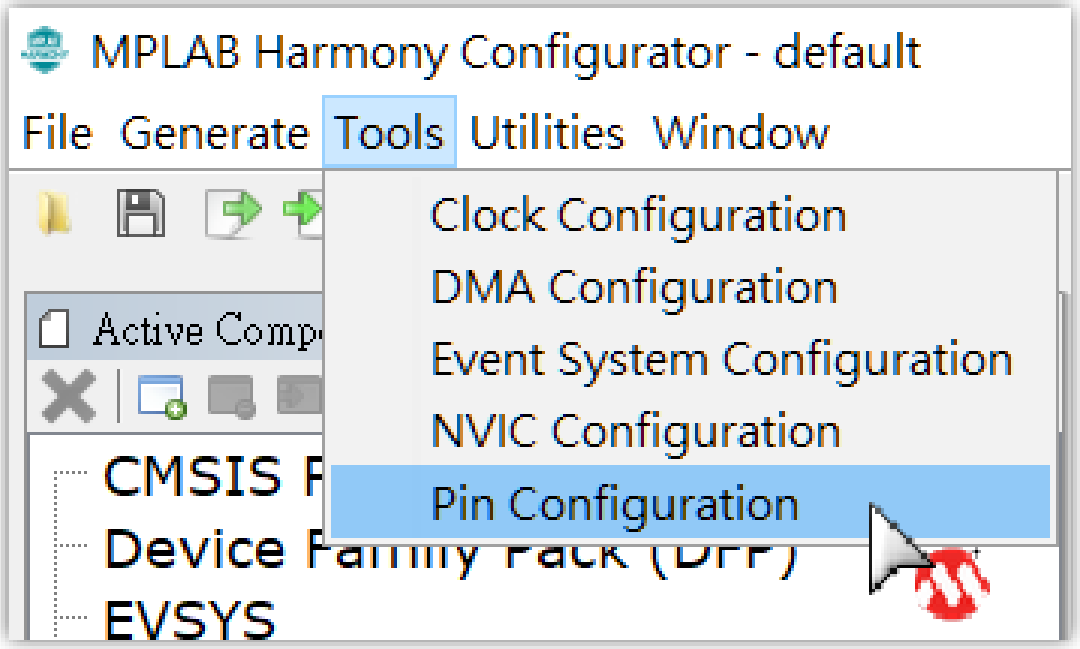






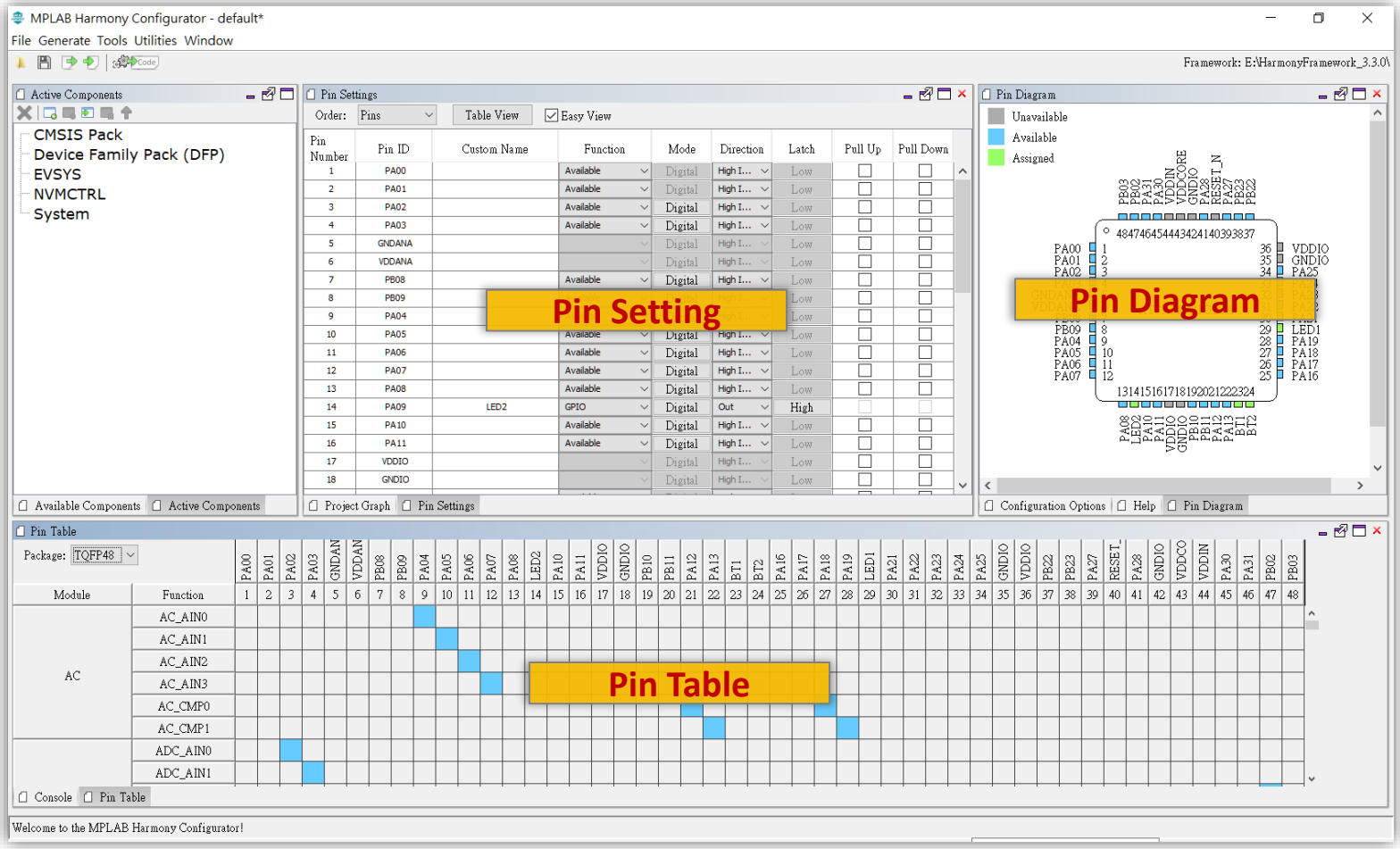
Pin Configuration using MHC

- Return to Harmony Configurator.
- Select Tools ▸ Pin Configuration



Pin Configuration using MHC

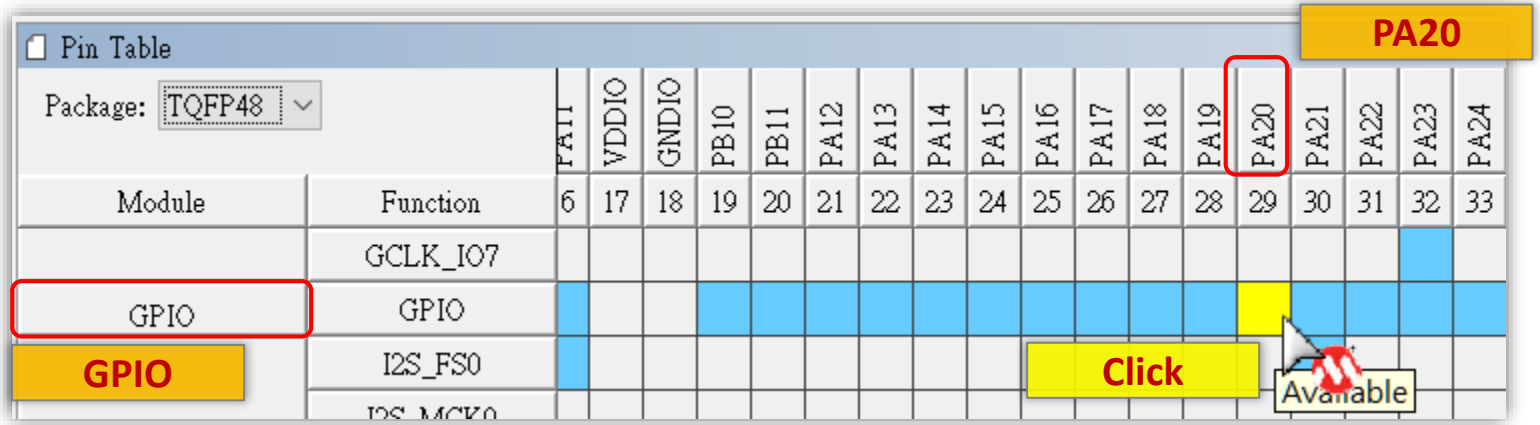
- Docking three **Pin Configuration** windows as below.



Lab1 PORT Output

Step 3

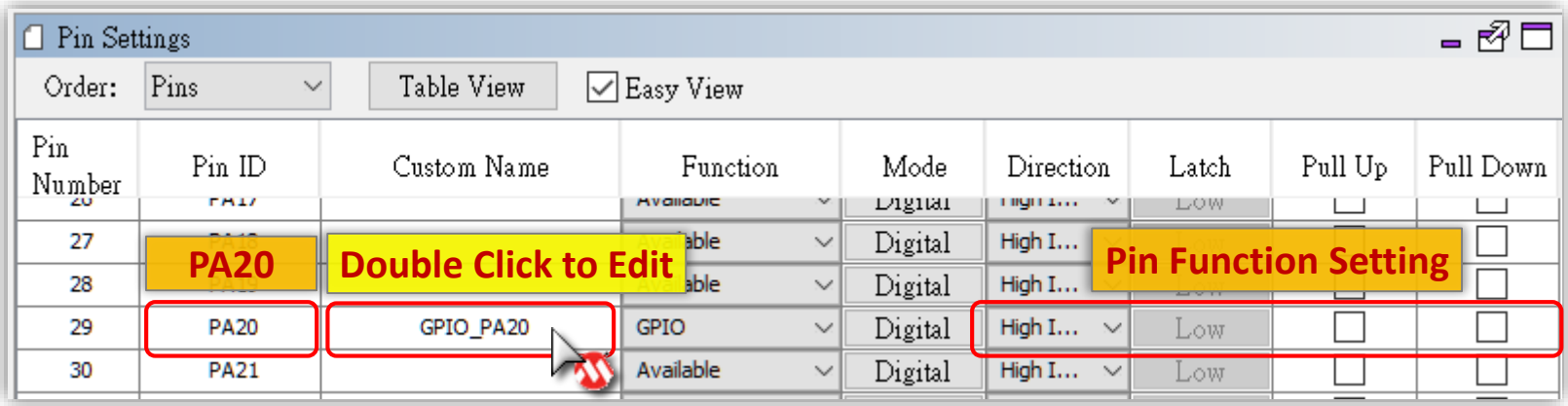
- Pin Table configure. (Enable or Disable Pin)

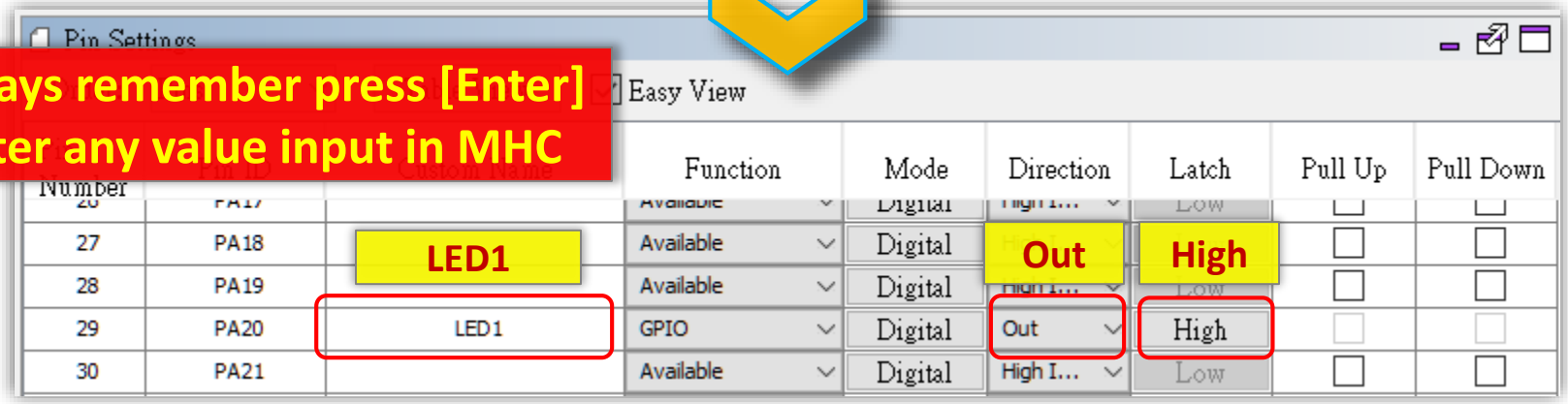


Lab1 PORT Output

Step 4

- **Pin Setting. (Pin Function Setting)**

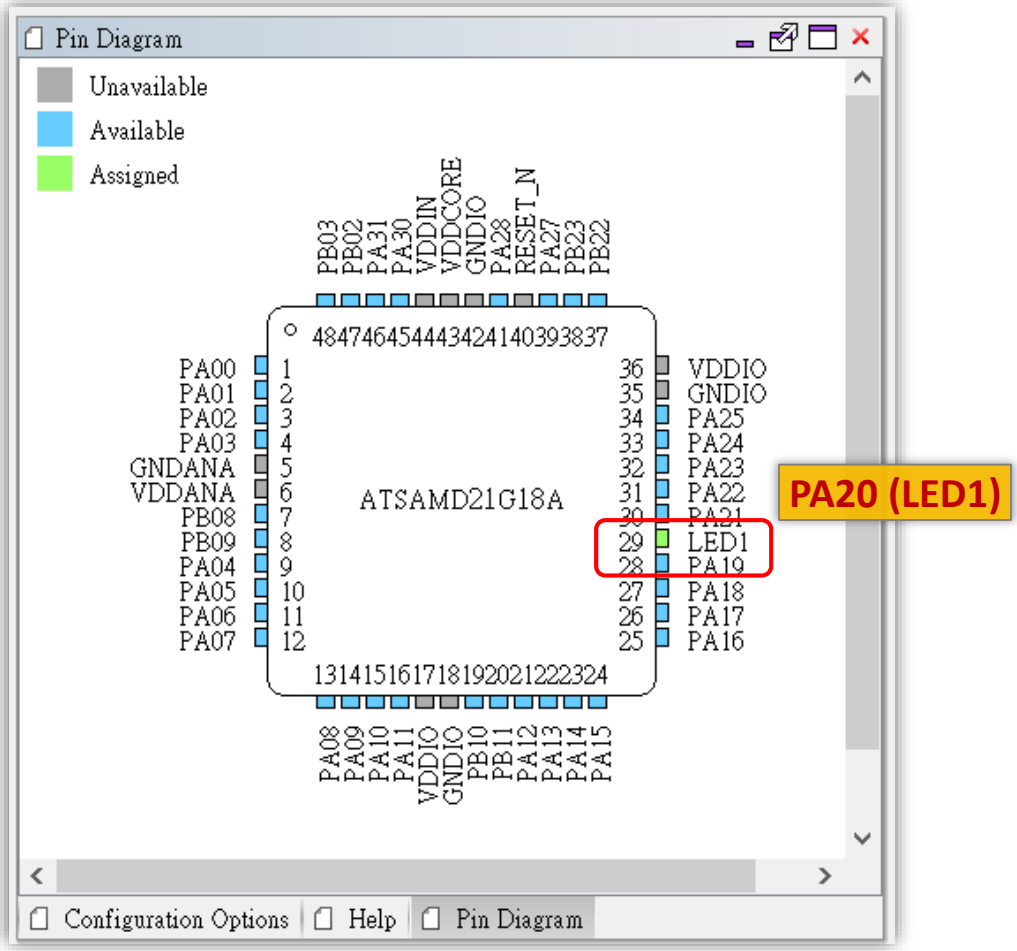




Lab1 PORT Output

Step 5

- Pin Diagram. (Hardware design reference)

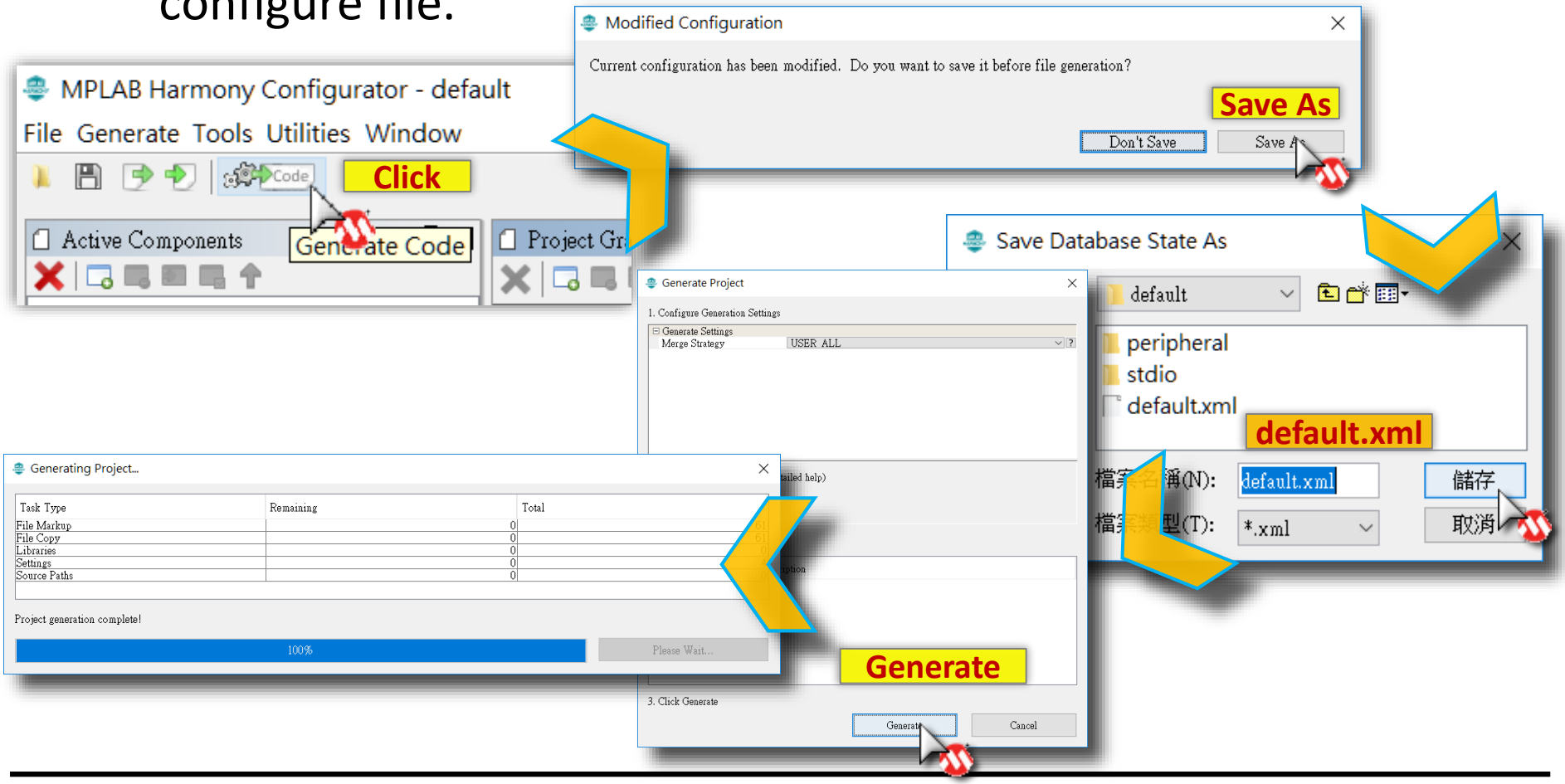


Lab1 PORT Output

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

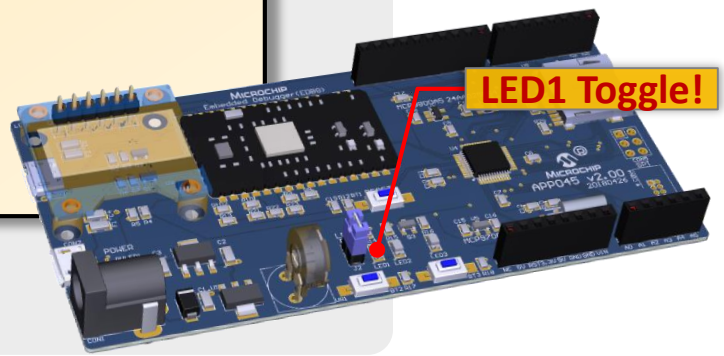
Step 7

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

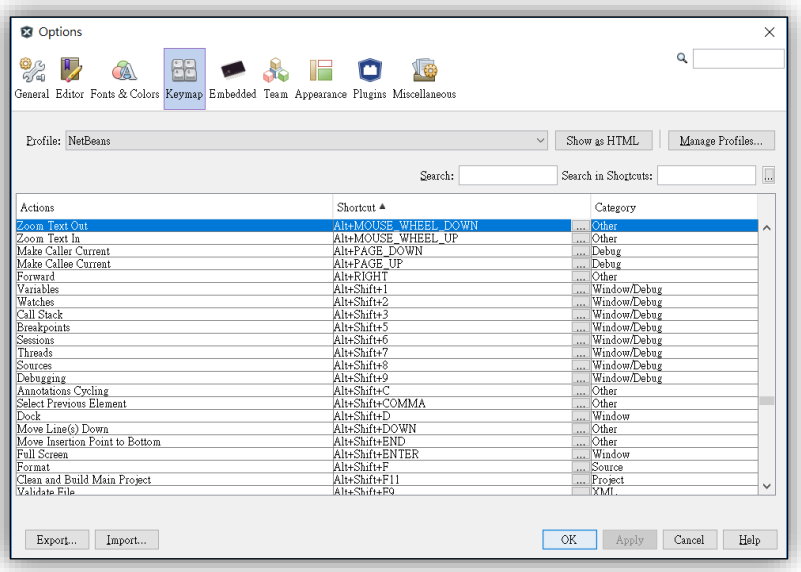


b **Program firmware to target board then observe result.**

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

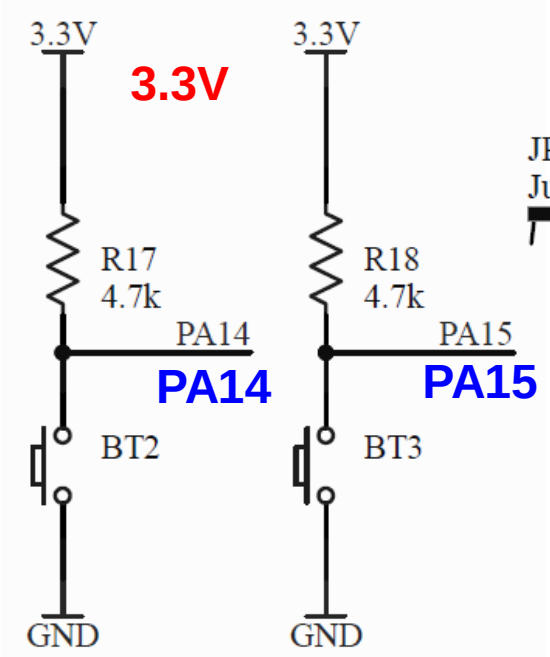


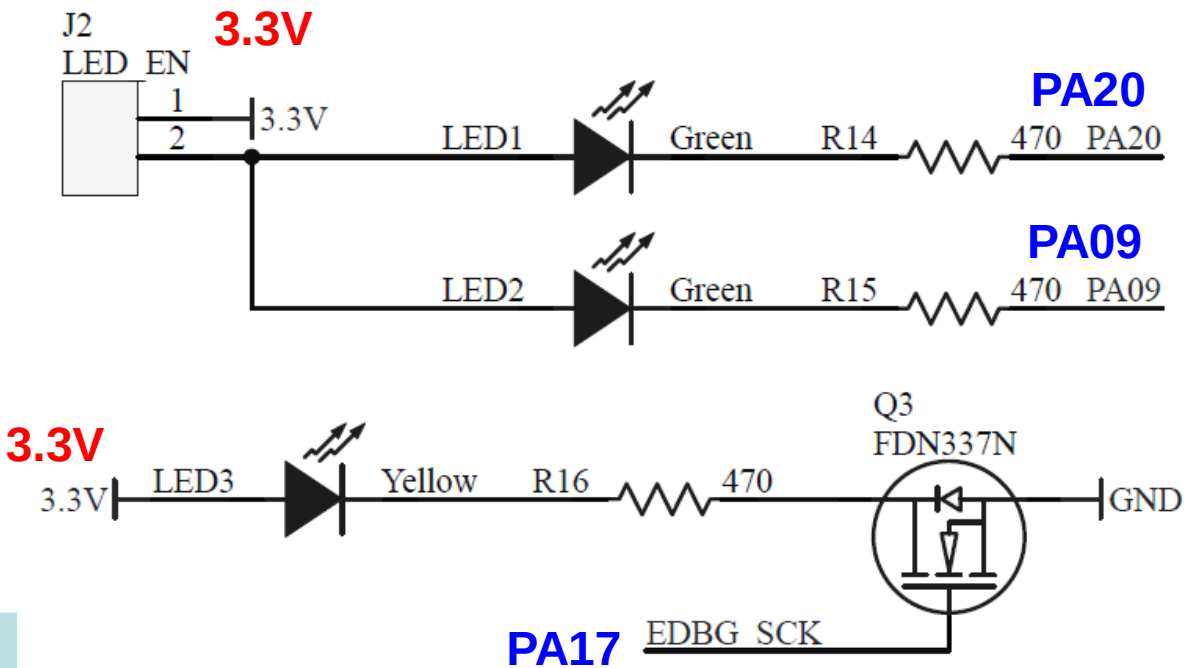
Lab2 PORT Input and Output

- 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!

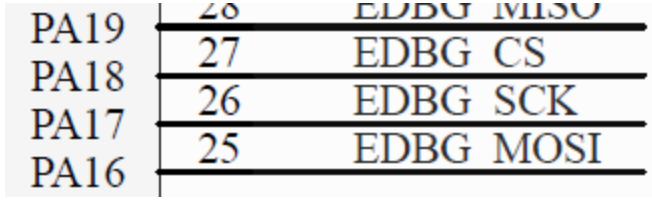
Check Schematic





LED1	PA20
LED2	PA09
LED3	PA17
BT2	PA14
BT3	PA15

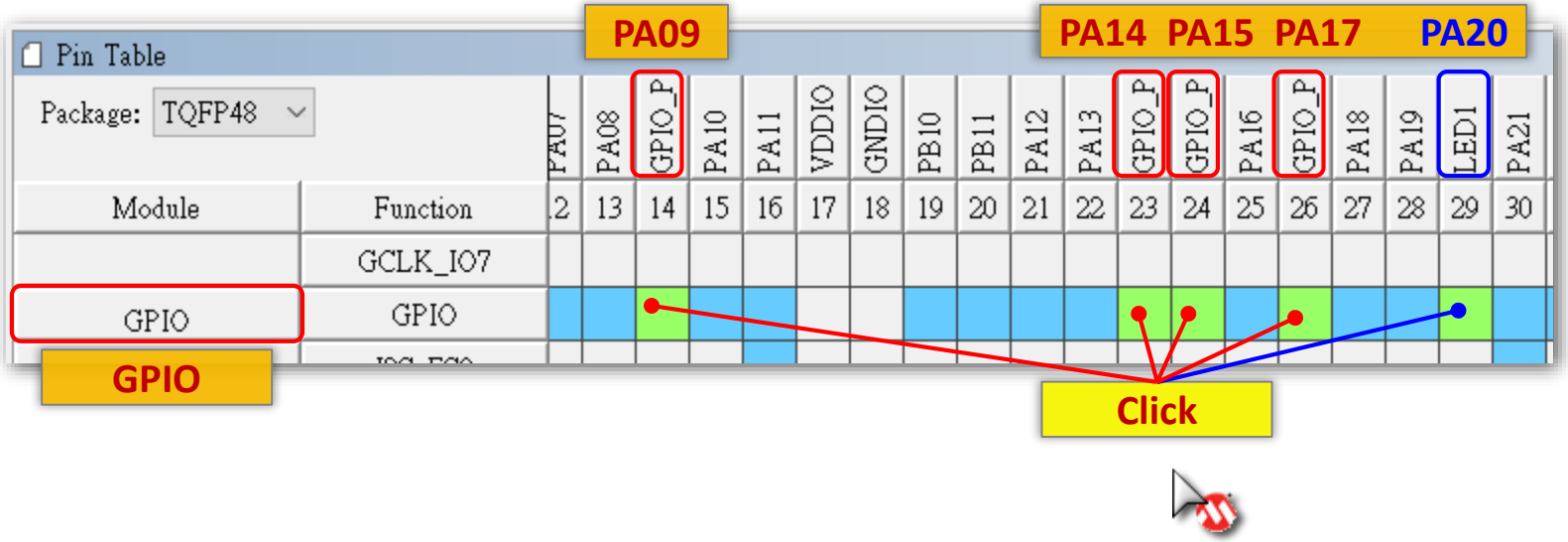
PA17



Lab2 PORT Input and Output

Step 1

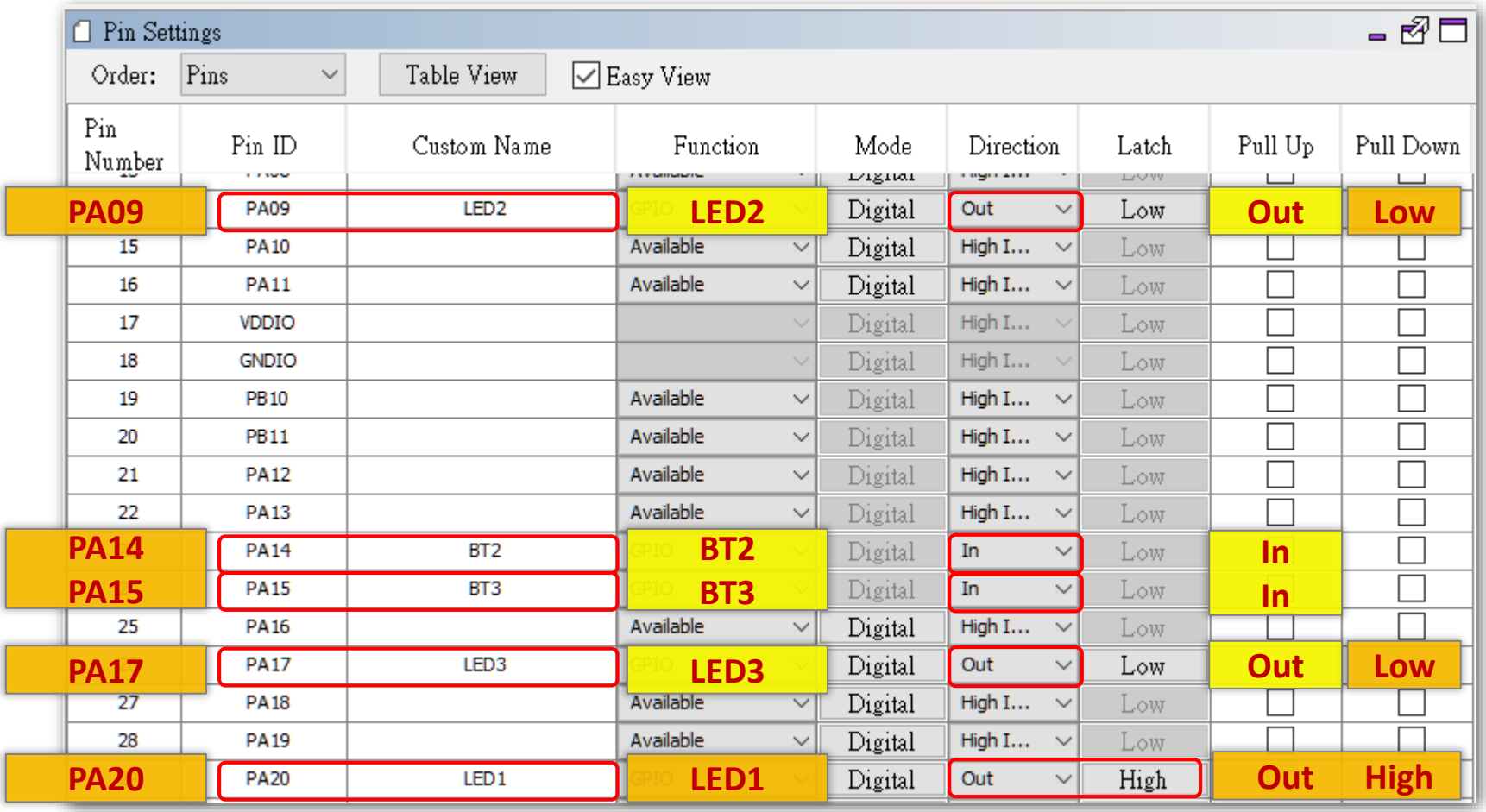
- Pin Table configure. (Enable or Disable Pin)



Lab2 PORT Input and Output

Step 2

- **Pin Setting. (Pin Function Setting)**

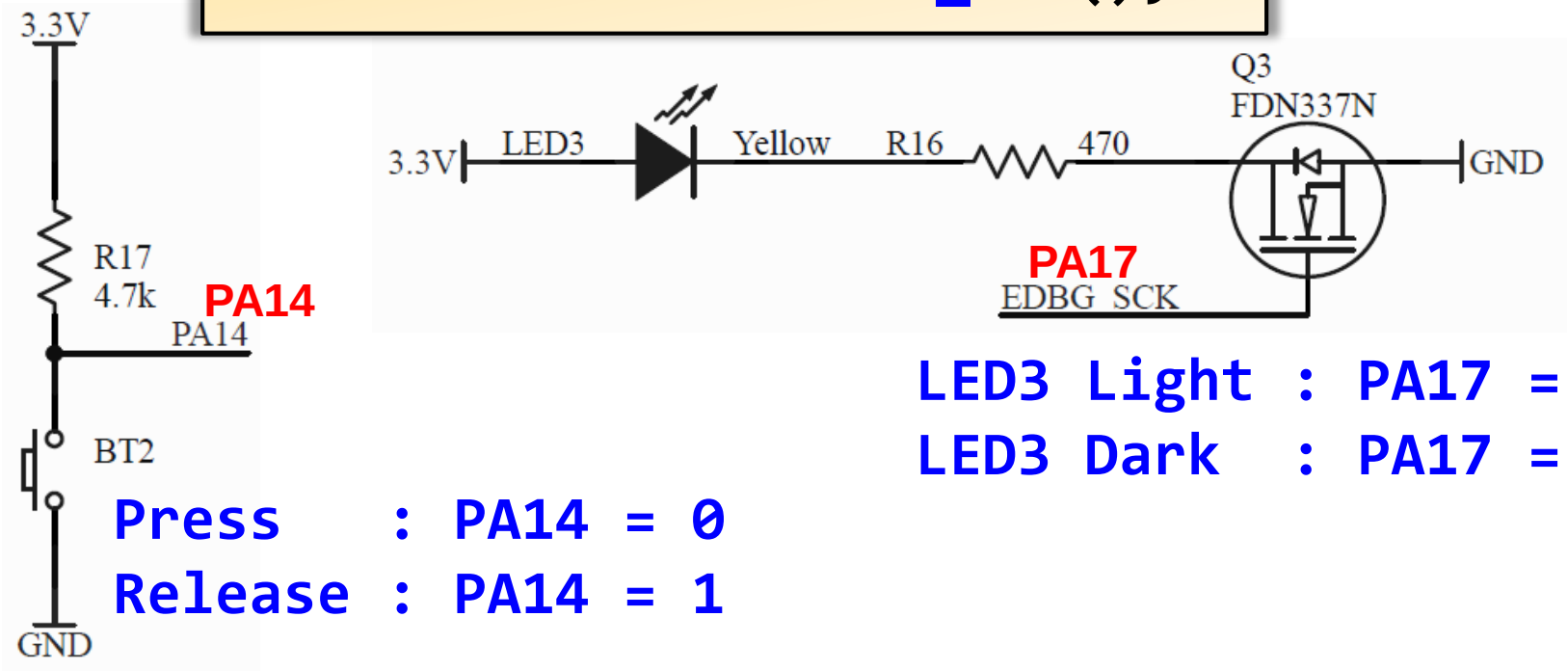


Lab2 PORT Input and Output

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();
```



Press : PA14 = 0
Release : PA14 = 1

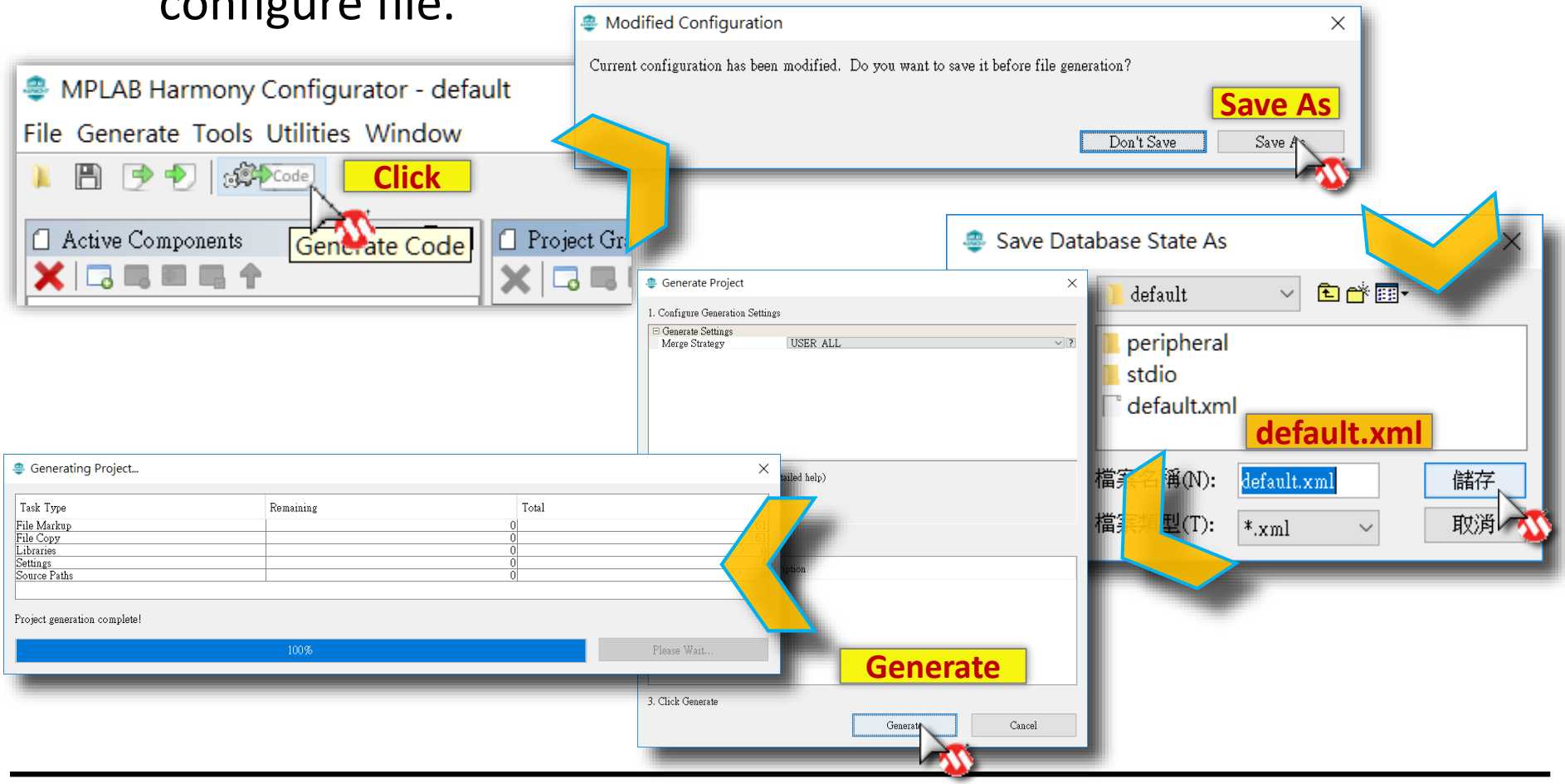
LED3 Light : PA17 = 1
LED3 Dark : PA17 = 0

Lab2 PORT Input and Output

Step 3

- Click 

to Generate Code and save changes to MHC configure file.

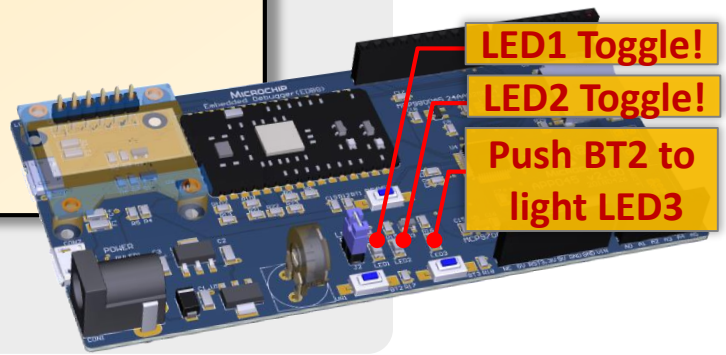


Lab2 PORT Input and Output

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();
   }
}
```



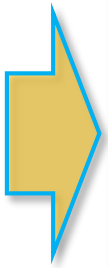
b **Program firmware to target board then observe result.**

Lab2 PORT Input and Output

Button Response Discuss

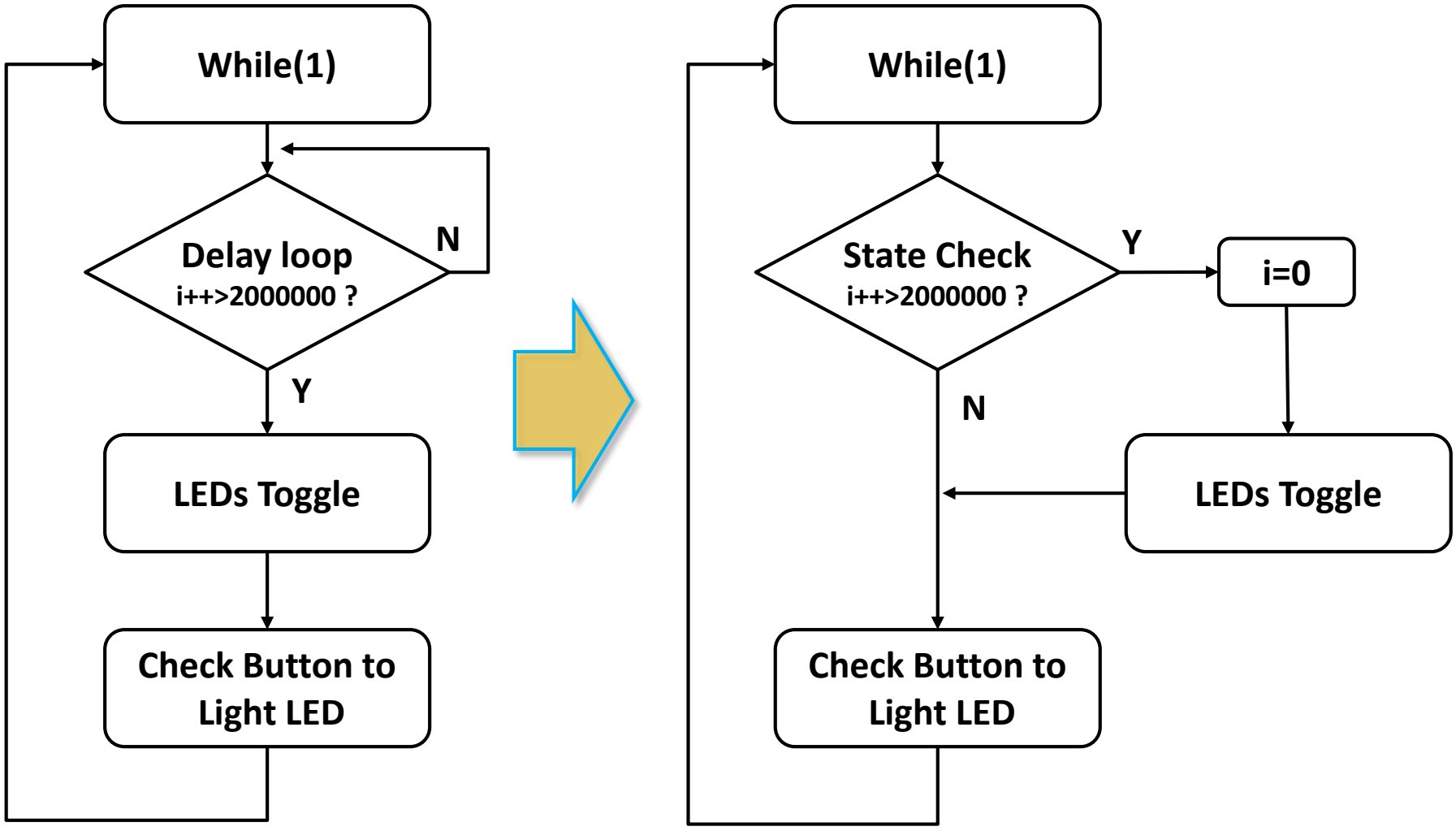
- **Button Response Time Issue ?**
  The software delay will influences the polling interval.
  Try to change coding style to state machine mode.

```
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!