# LAPIS MCU Introduction Training

# LAPIS MCU Workshop Training



Pichet Sriphomma pichet@es.co.th

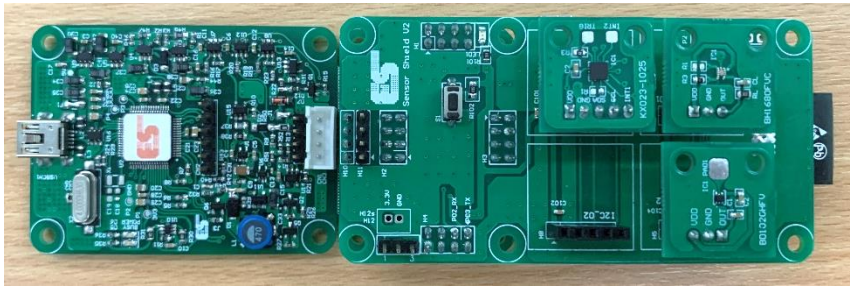
# LAPIS MCU Introduction Training

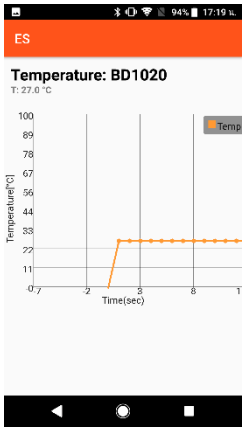


Pichet Sriphomma pichet@es.co.th

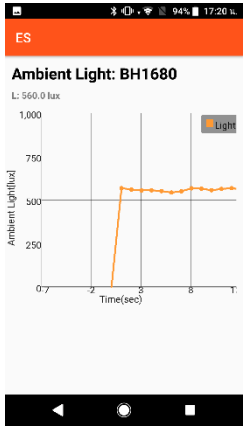
# Overview Tooling & S/W for training

LEXIDE-U16

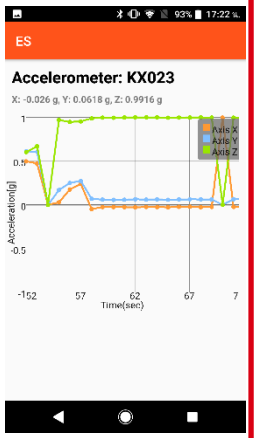




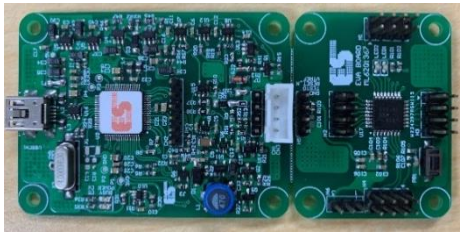

BD1020



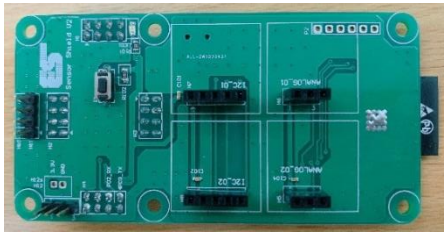

BH1680




KX023

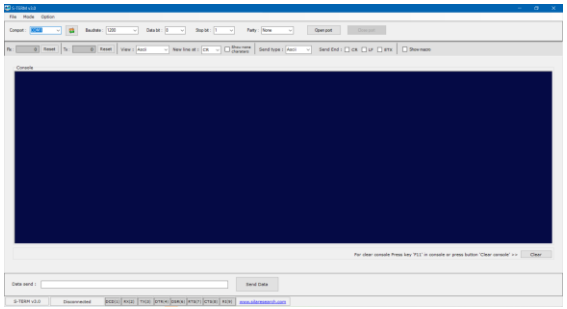


ES-ICD-V1



Sensor Shield Board

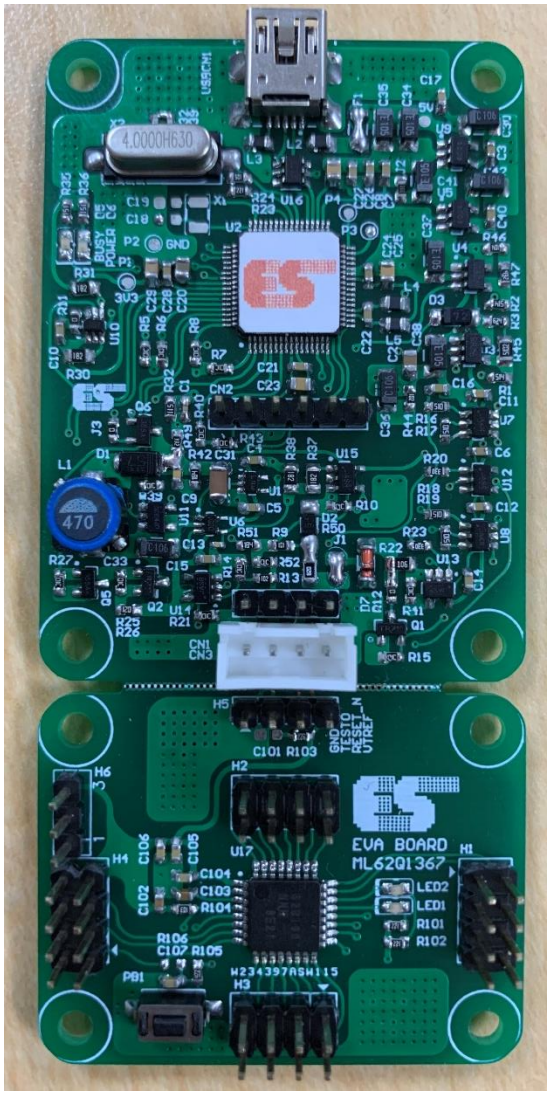
BD1020
BH1680
KX023



S-TERM

Pichet Sriphomma pichet@es.co.th

# ES-ICD-V1 Reference board




**Debugger LAPIS**

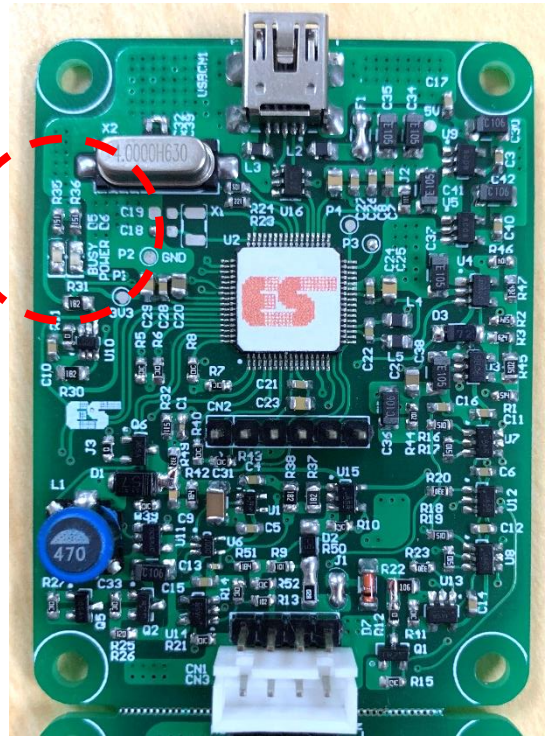
**EVA ML62Q1367 Board**

Pichet Sriphomma pichet@es.co.th

# ES-ICD-V1 Reference board

## Debugger LAPIS

| LED Status |
| --- |
| POWER |
| BUSY |



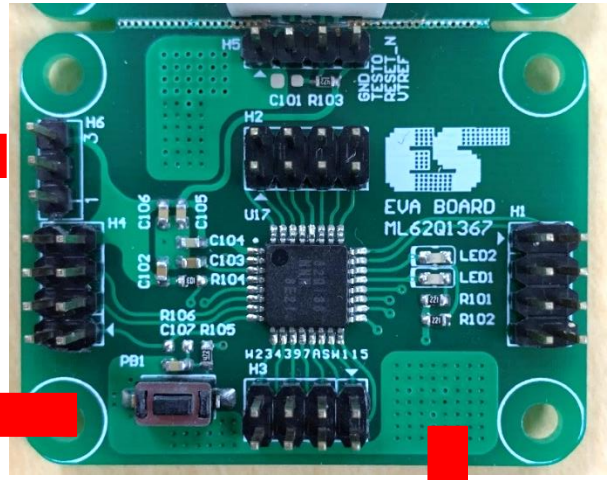
| VTREF |
| --- |
| RESET_N |
| TEST0 |
| GND |

Pichet Sriphomma pichet@es.co.th

# ES-ICD-V1 Reference board

## EVA ML62Q1367 Board

| VTREF |
| --- |
| RESET_N |
| TEST0 |
| GND |




| VTREF |
| --- |
| |
| GND |

**Switch (P03)**

| LED2 (P21) |
| --- |
| LED1 (P20) |



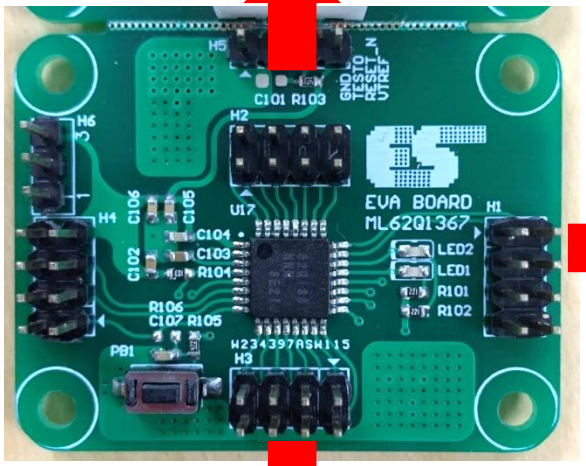
Pichet Sriphomma pichet@es.co.th

# ES-ICD-V1 Reference board

## EVA ML62Q1367 Board

| P33 | P31 | P27 | P25 |
|---|---|---|---|
| P32 | P30 | P26 | P24 |

| GND | VTREF |
|---|---|
| | P01 |
| | P02 |
| | P03 |



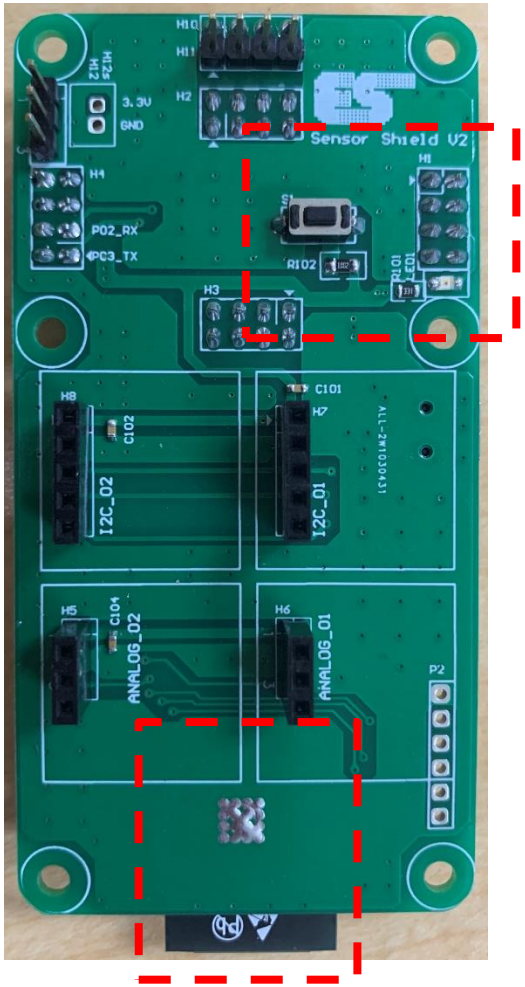

| P22 | P23 |
|---|---|
| P20 | P21 |
| P16 | P17 |
| P14 | P15 |

| P04 | P06 | P10 | P12 |
|---|---|---|---|
| P05 | P07 | P11 | P13 |



Pichet Sriphomma pichet@es.co.th

# Sensor Shield Board




**Switch (P16)**
**LED (P17)**

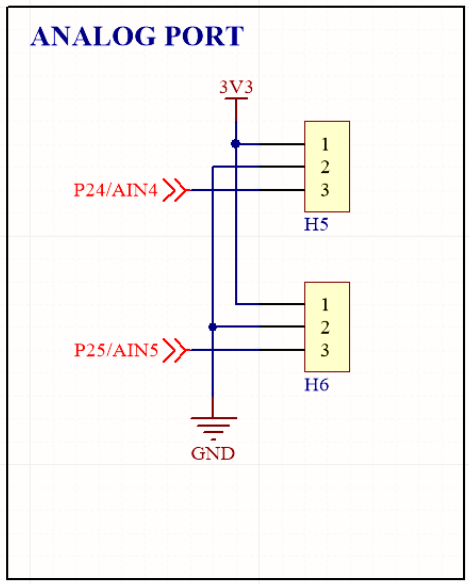
**I2C Port**

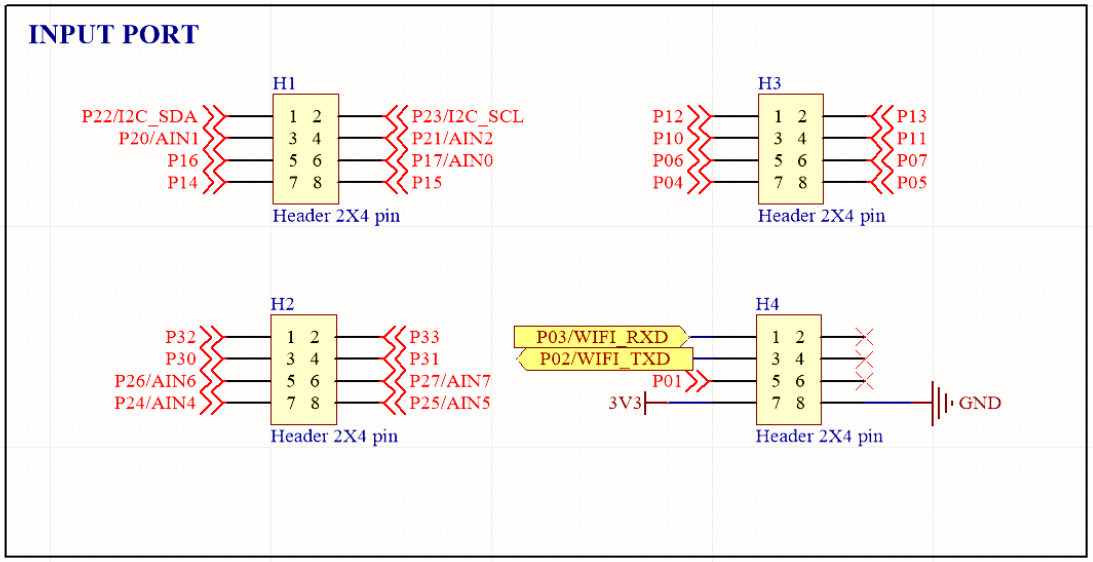
**ADC Port**

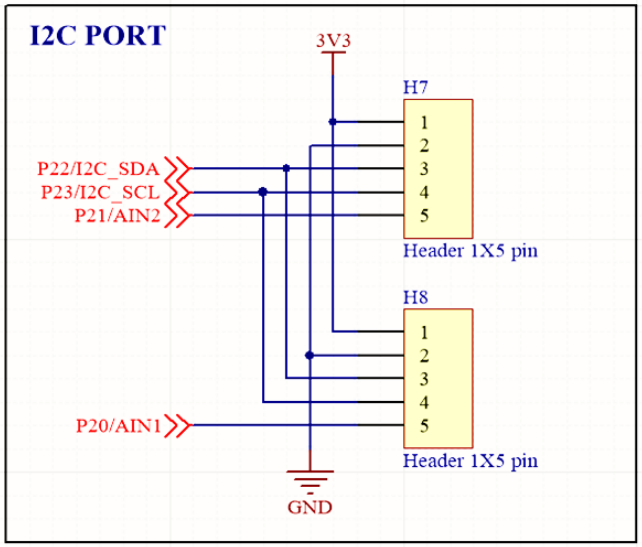
**Bluetooth Module**

Pichet Sriphomma pichet@es.co.th

# Sensor Shield Board

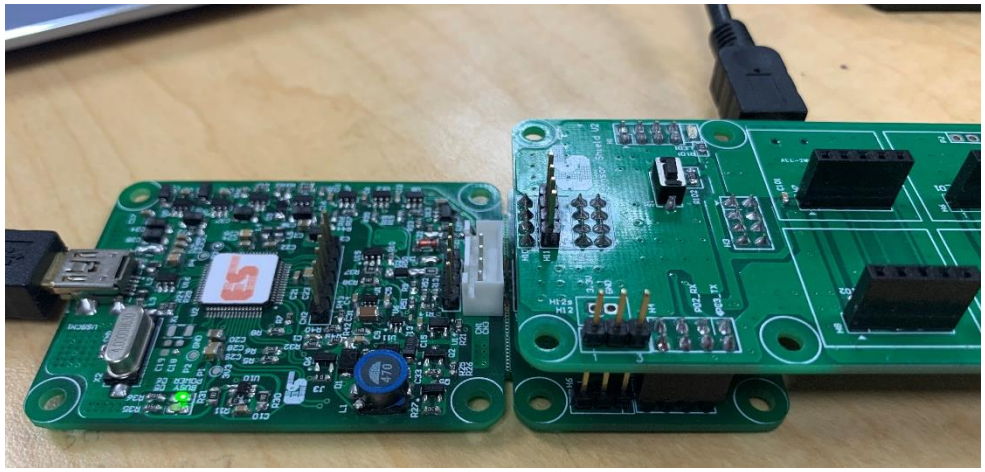


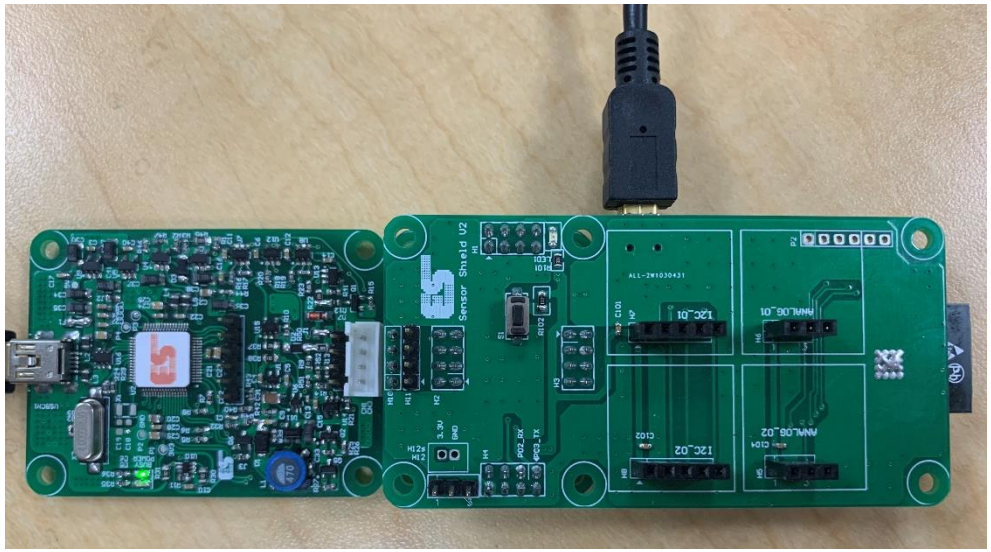




Pichet Sriphomma pichet@es.co.th

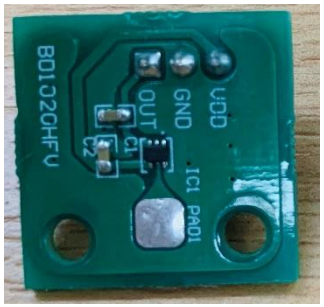
# Connection between ES-ICD-V1 and Sensor Shield.





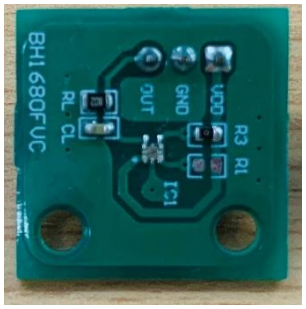

Pichet Sriphomma pichet@es.co.th

# Sensor Board




**Temperature Sensor (BD1020)**

**[ADC]**




**Ambient Light Sensor (BH1680)**

**[ADC]**


**Accelerometer Sensor (KX023)**

**[I2C]**

Back to Overview



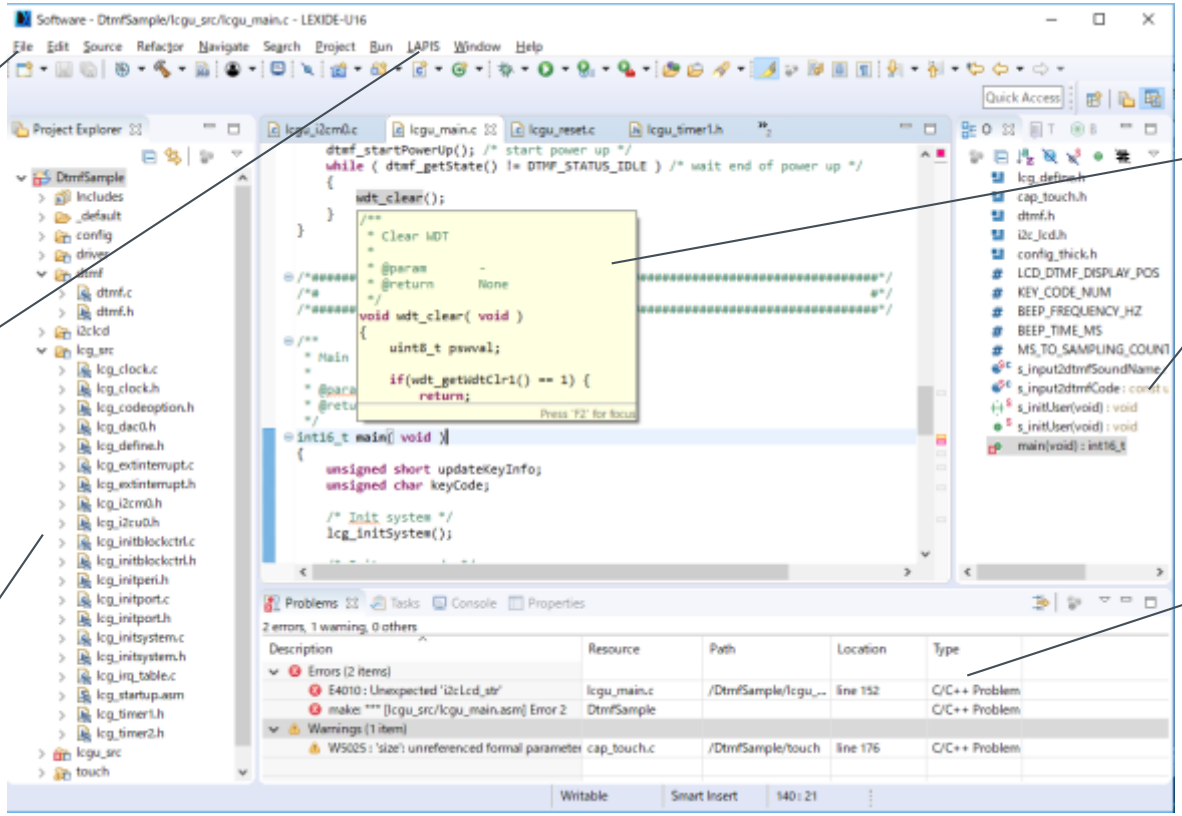
Pichet Sriphomma pichet@es.co.th

# LEXIDE-u16 (Eclipse base IDE)

The IEDU8 project is importable

LAPIS menu runs the original tools

Project tree




Display underlying functions

Automatic display of functions in the source files

Display the error list. Double-clicking the error message jumps to the error line in the source file.

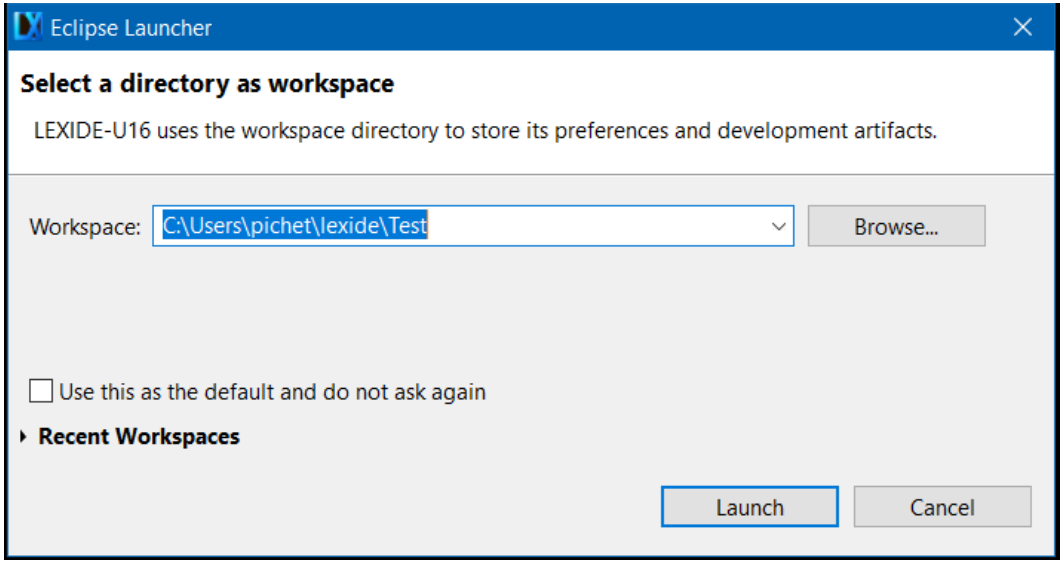


Pichet Sriphomma pichet@es.co.th

# LEXIDE-u16 (Eclipse base IDE)


Open Program LEXIDE-U16

When clicked, the following workspace setting dialog box will be output. Set a path to workspace at [Workspace]. After that Click [Launch].

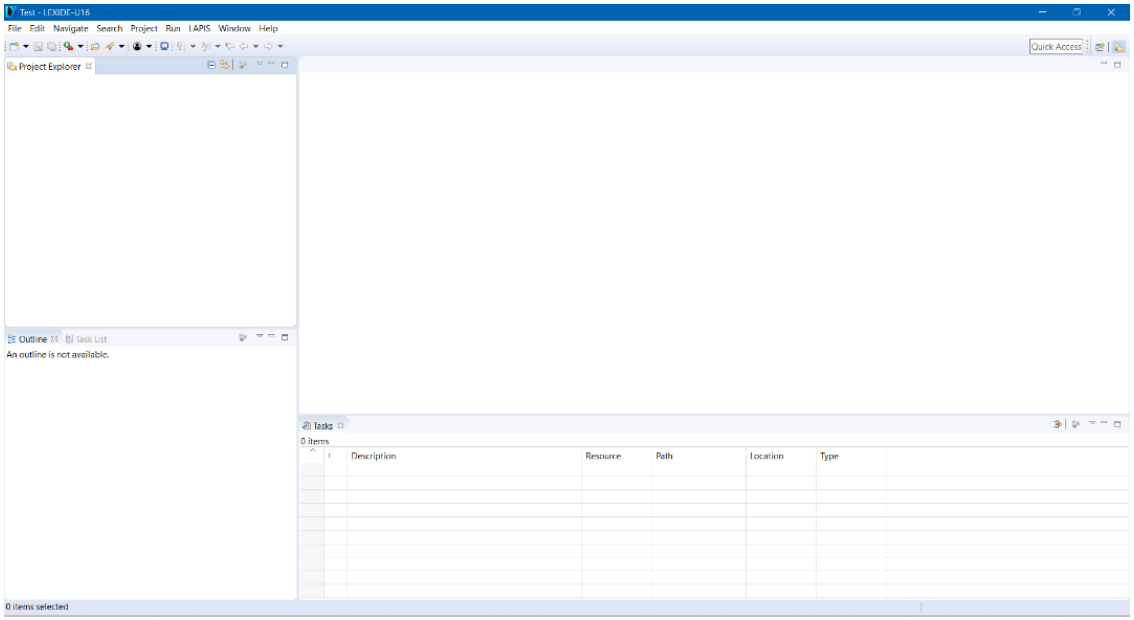





Pichet Sriphomma pichet@es.co.th

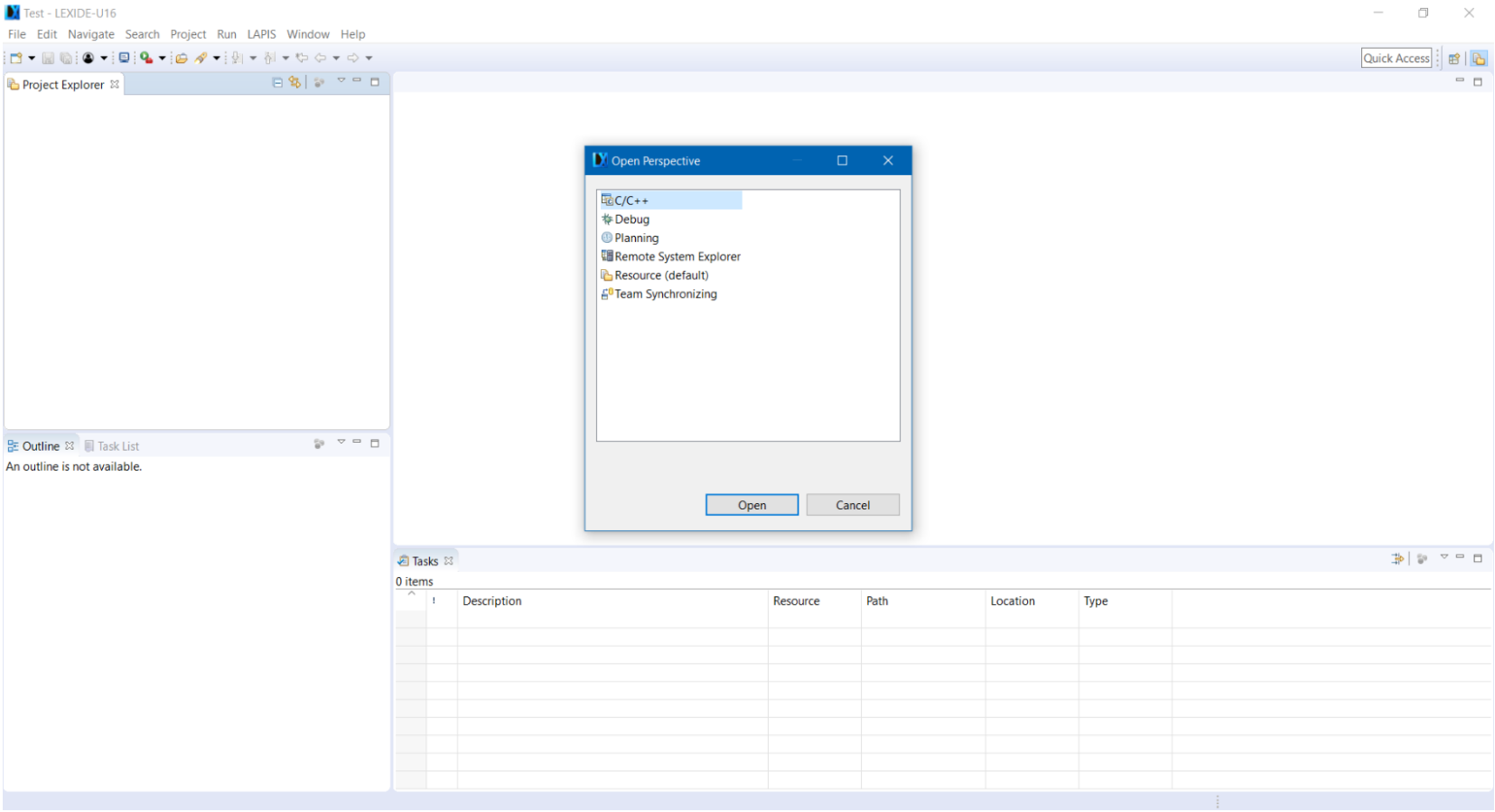
# LEXIDE-u16 (Eclipse base IDE)


The first window after opening the program




Pichet Sriphomma pichet@es.co.th

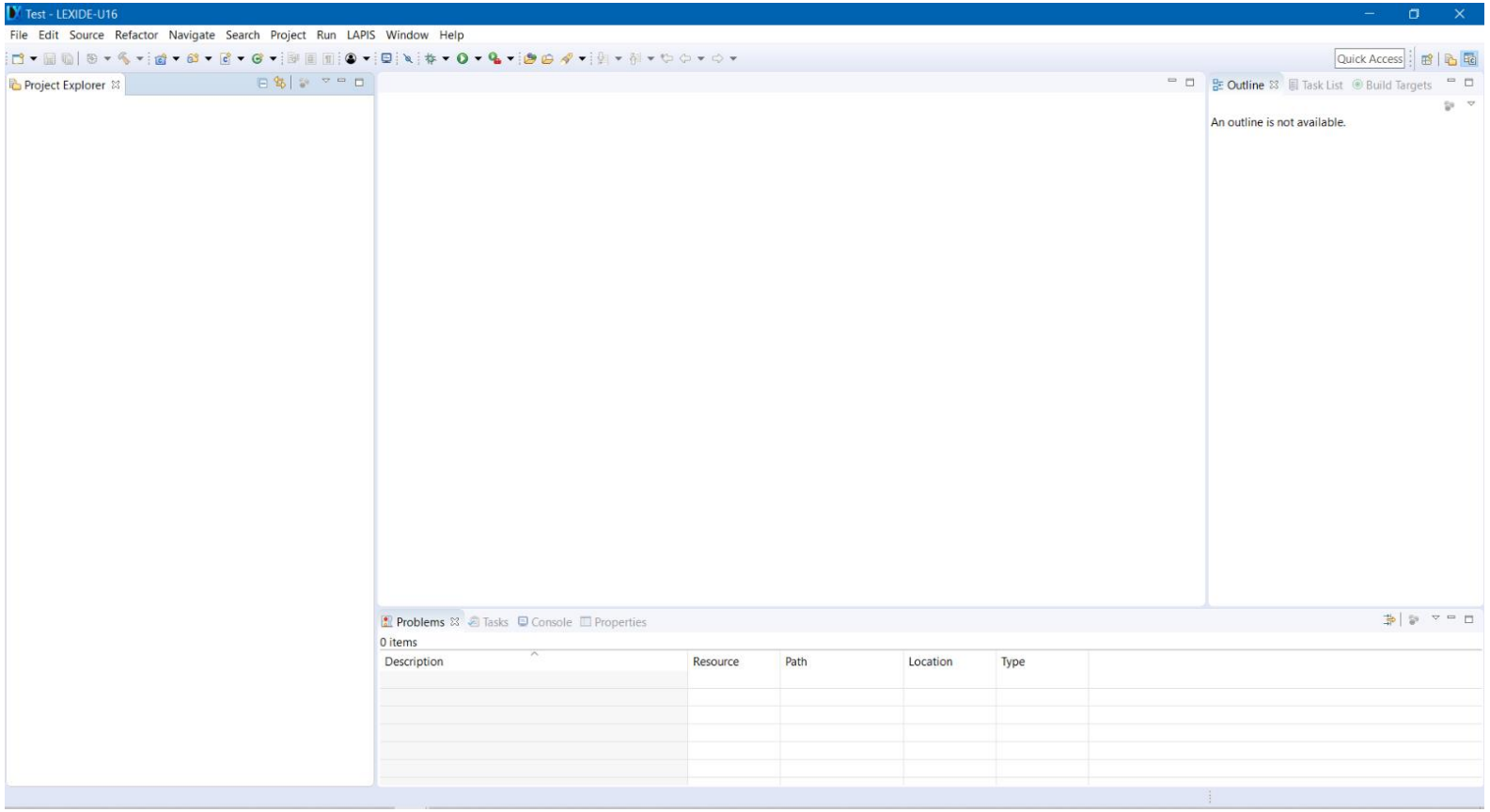
Click the button in the upper right  to display the [Open Perspective] dialog. Select [C/C++] and Click [OK].




Pichet Sriphomma pichet@es.co.th

# LEXIDE-u16 (Eclipse base IDE)

The Perspective set to [C / C ++].



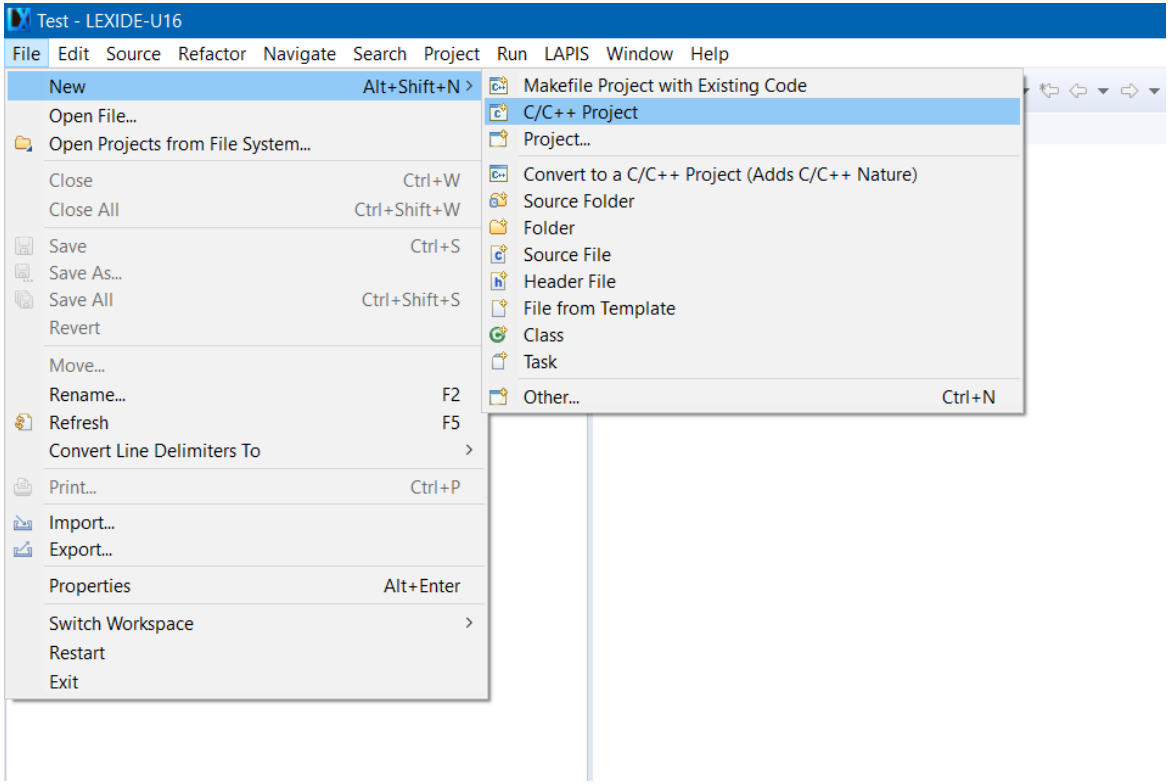



Pichet Sriphomma pichet@es.co.th

# LEXIDE-u16 (Eclipse base IDE)

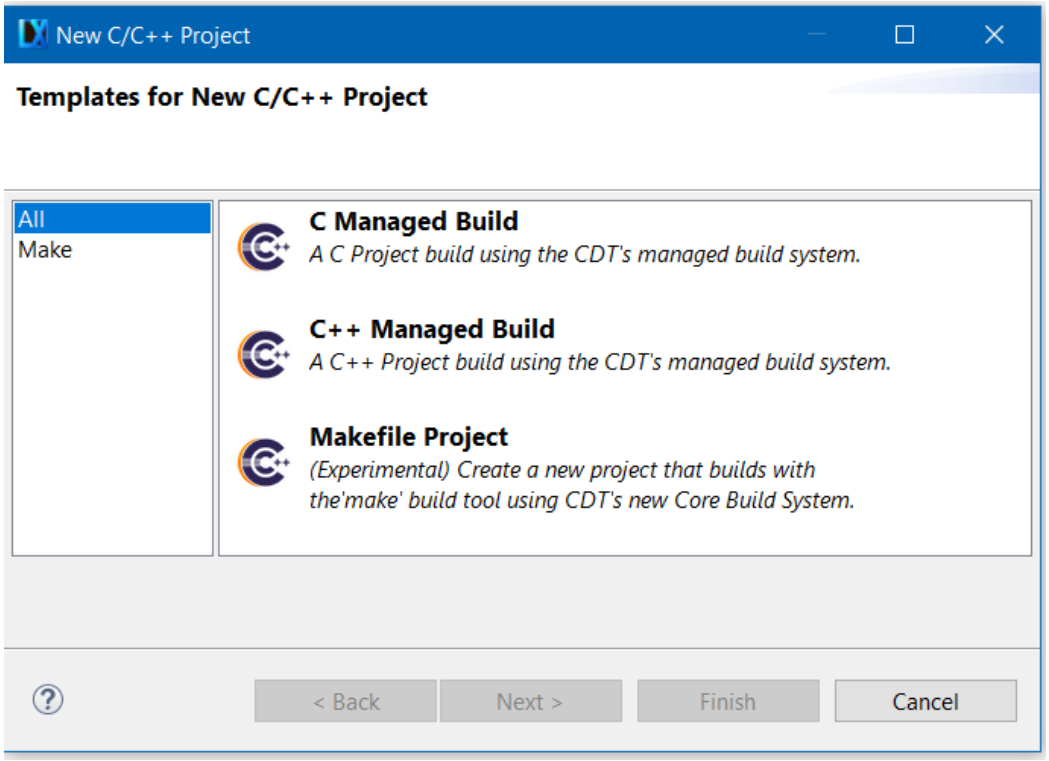
To create a project

From the menu, select [File] > [New] > [C/C++ Project] to open the [New C/C++ Project] dialog box.




Pichet Sriphomma pichet@es.co.th

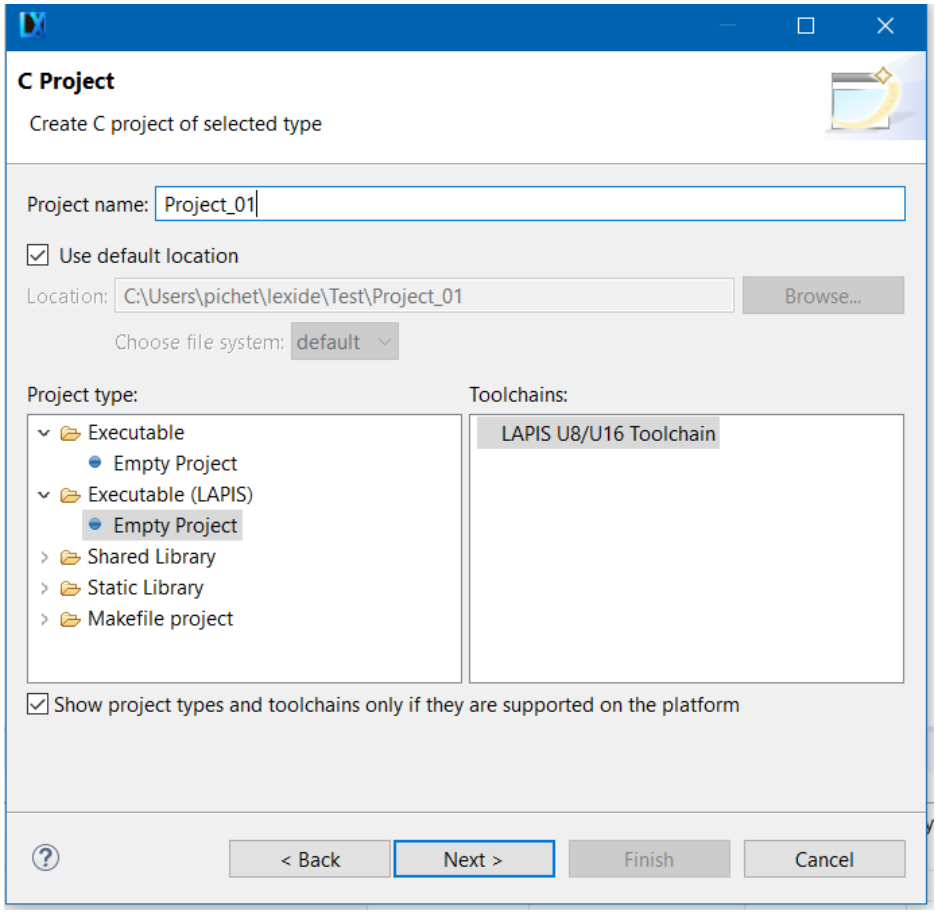
Select [C Managed Build] and then click [Next]. The [Create C project of selected type] page will be opened.






Pichet Sriphomma pichet@es.co.th

# LEXIDE-u16 (Eclipse base IDE)

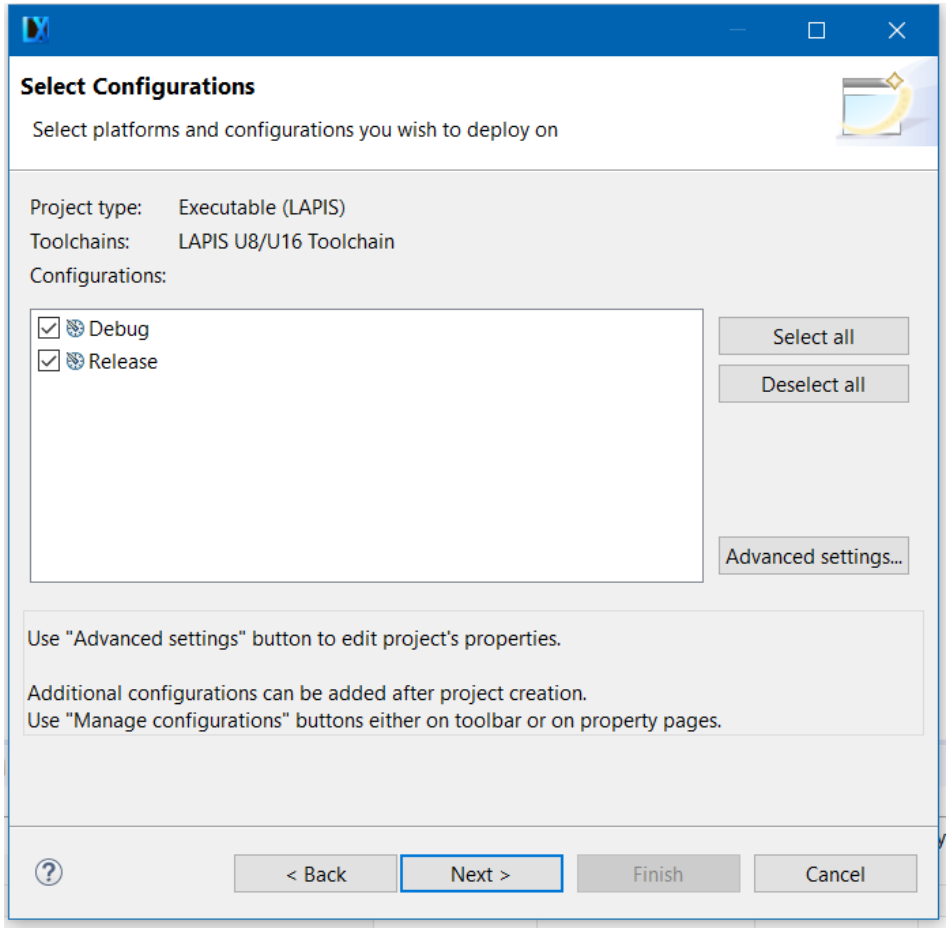



Set a project name.
Select [Executable (LAPIS)] > [Empty Project] in the [Project type] pane and select [LAPIS U8/U16 Toolchain] in the [Toolchains] pane.
Click [Next].



Pichet Sriphomma pichet@es.co.th

# LEXIDE-u16 (Eclipse base IDE)

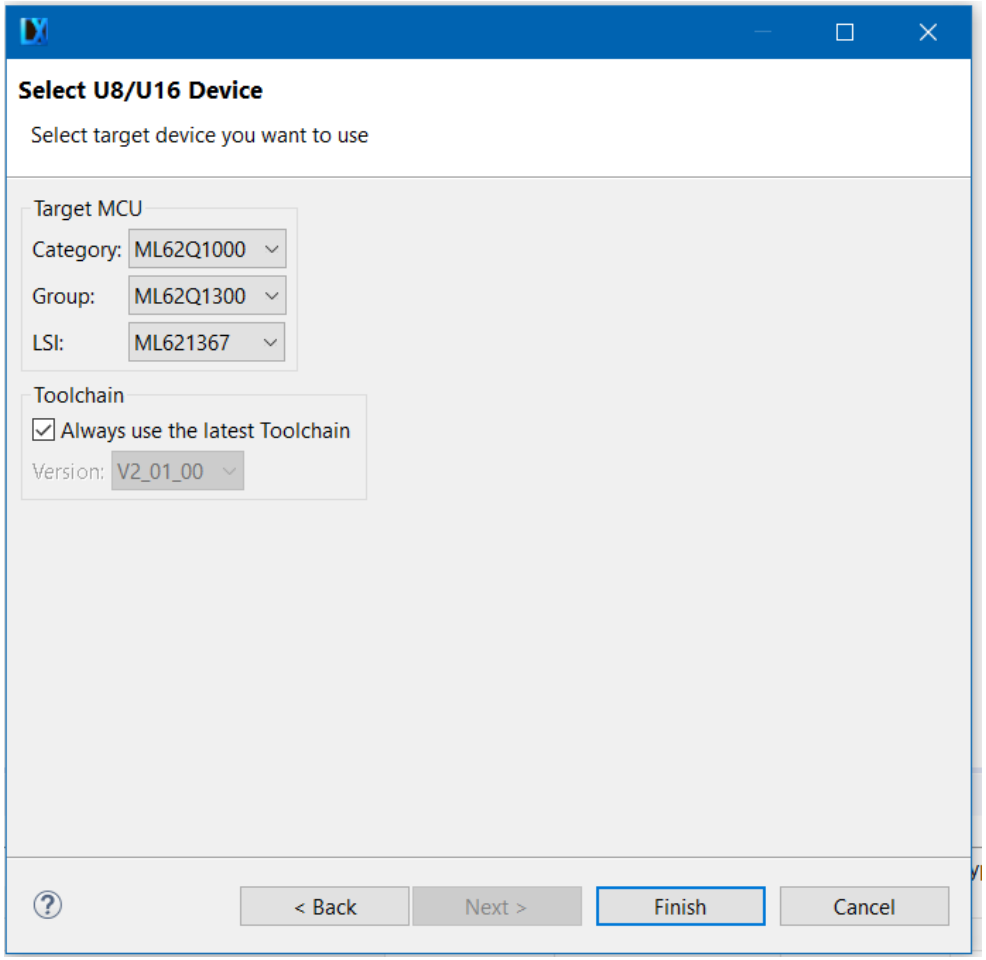



Click [Next].



Pichet Sriphomma pichet@es.co.th

# LEXIDE-u16 (Eclipse base IDE)

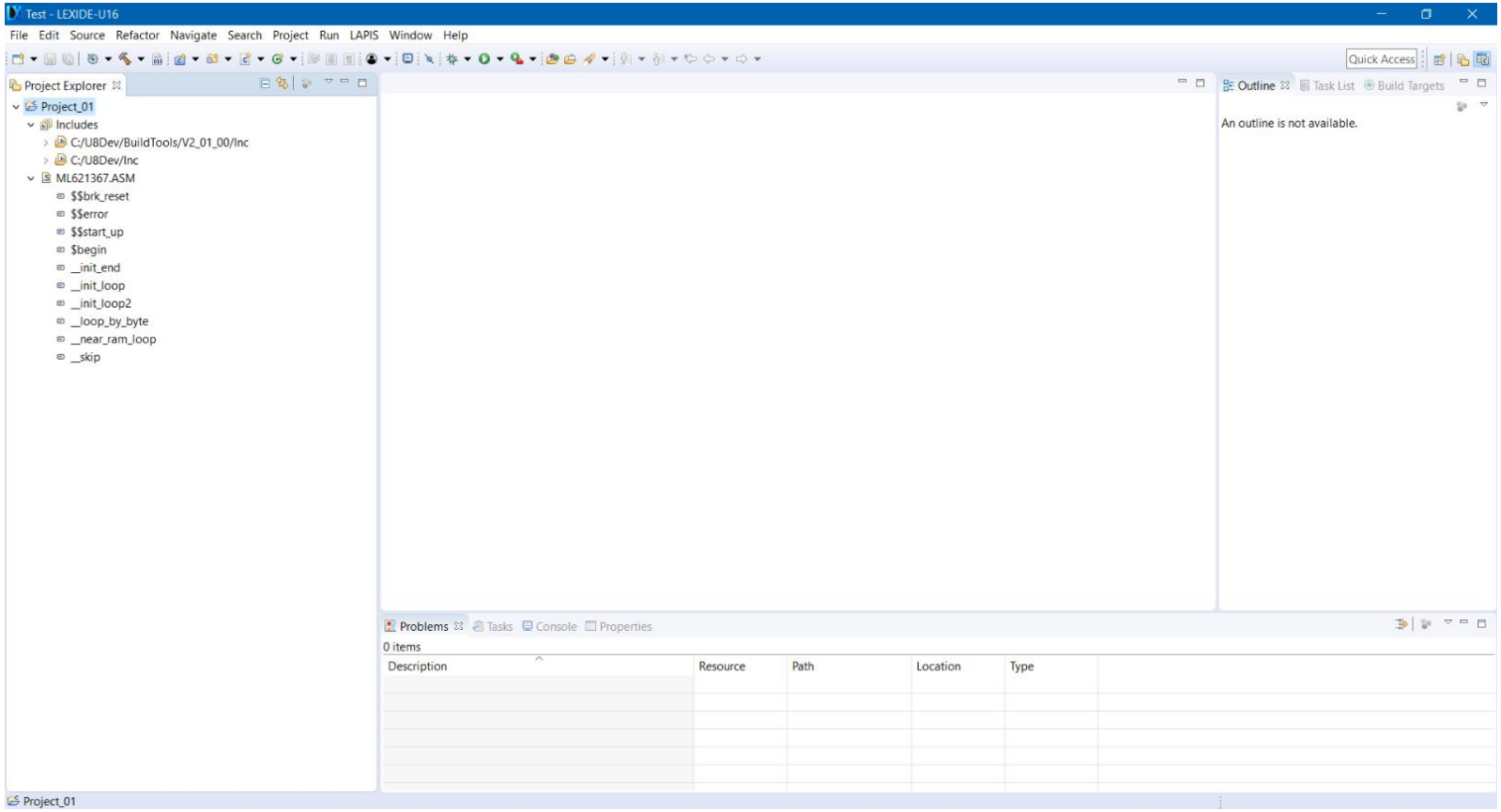



Click [Next] to open the [Select U8/U16 Device] page.



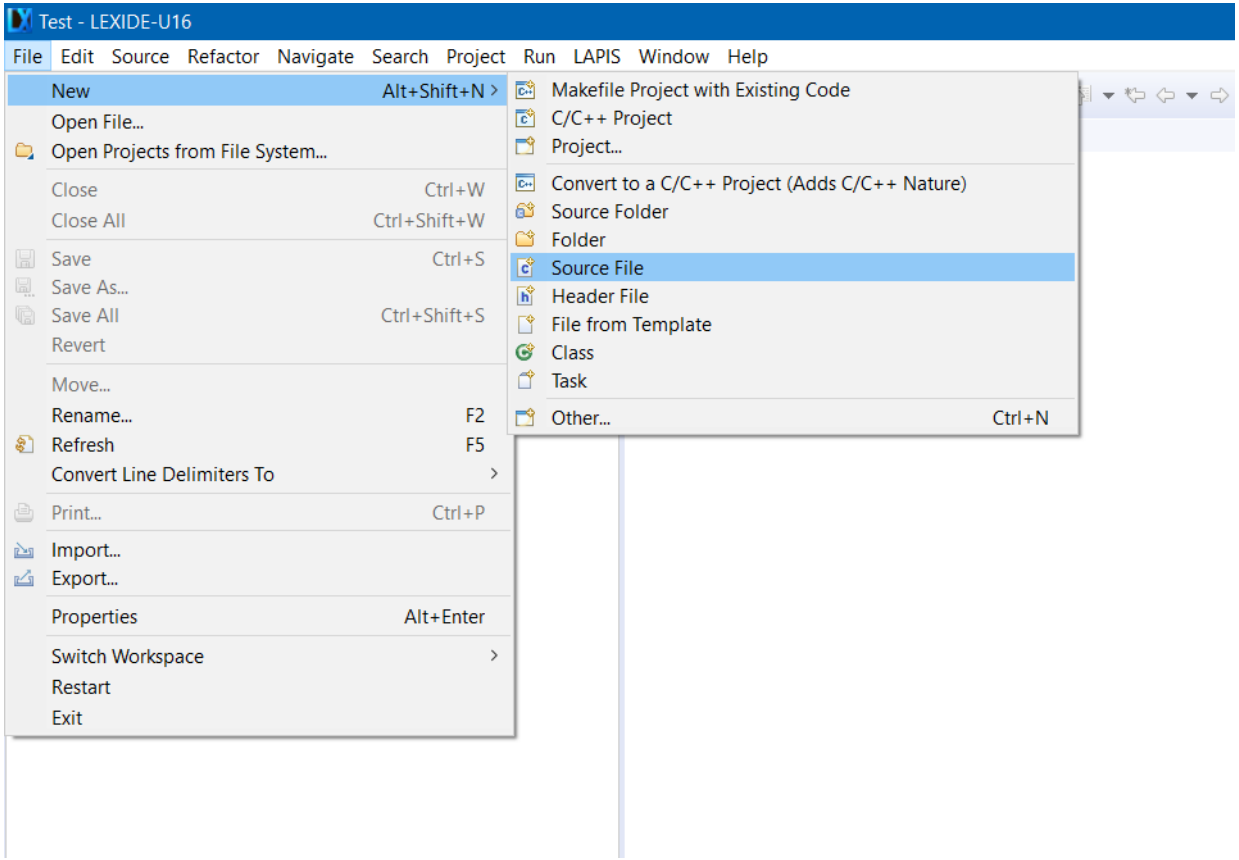
Pichet Sriphomma pichet@es.co.th

Click [Finish] to exit the project wizard and check that the project has been created.



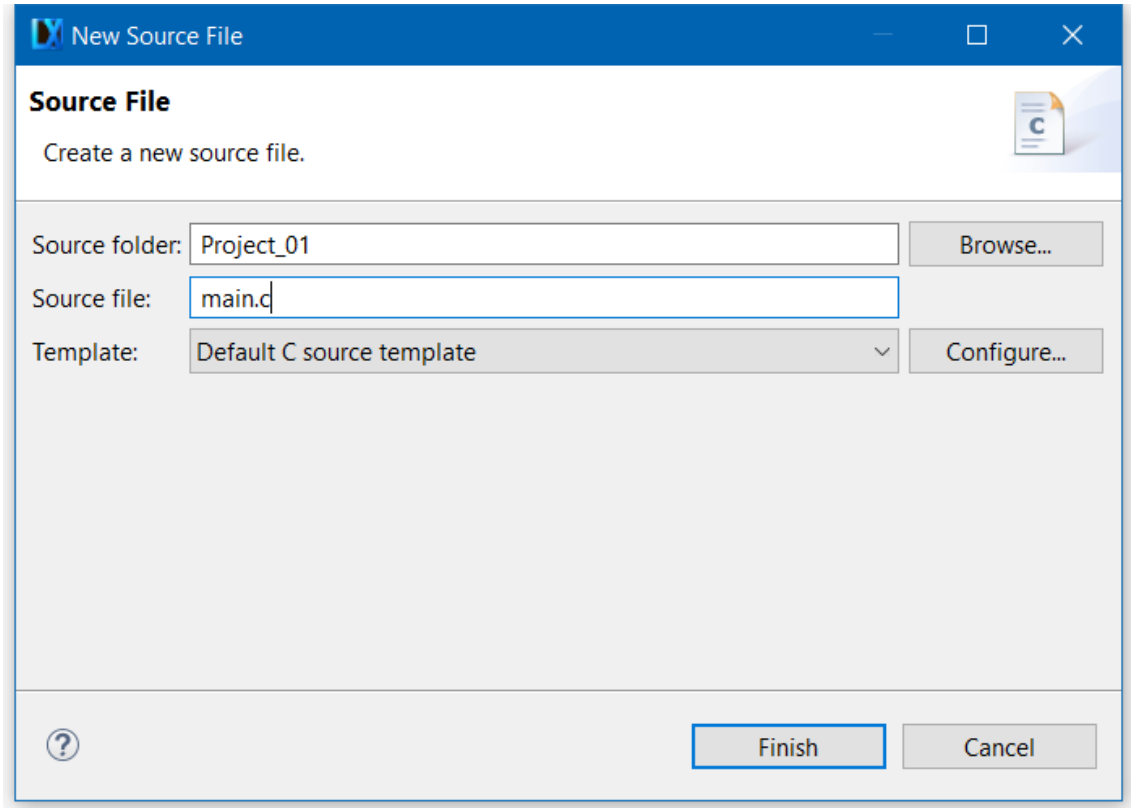

Pichet Sriphomma pichet@es.co.th

add a new source file. Right-click on a project folder and select [New] > [Source File].




Pichet Sriphomma pichet@es.co.th

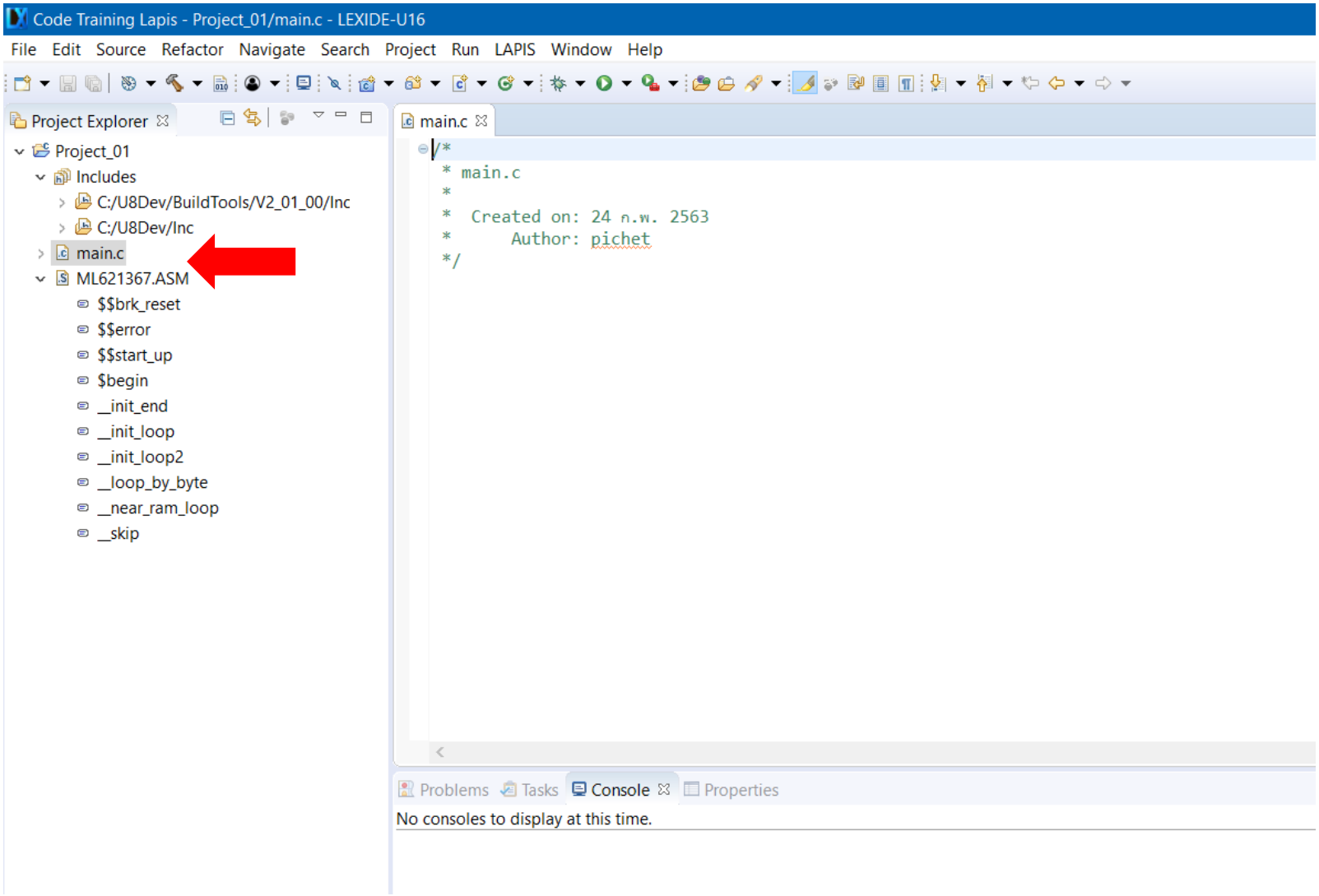
The [New Source File] dialog box will be opened.
“main.c” is entered to the [Source file]
Select [Default C source template] for [Template].
After that, click [Finish] to add the entered source file to the project.






Pichet Sriphomma pichet@es.co.th

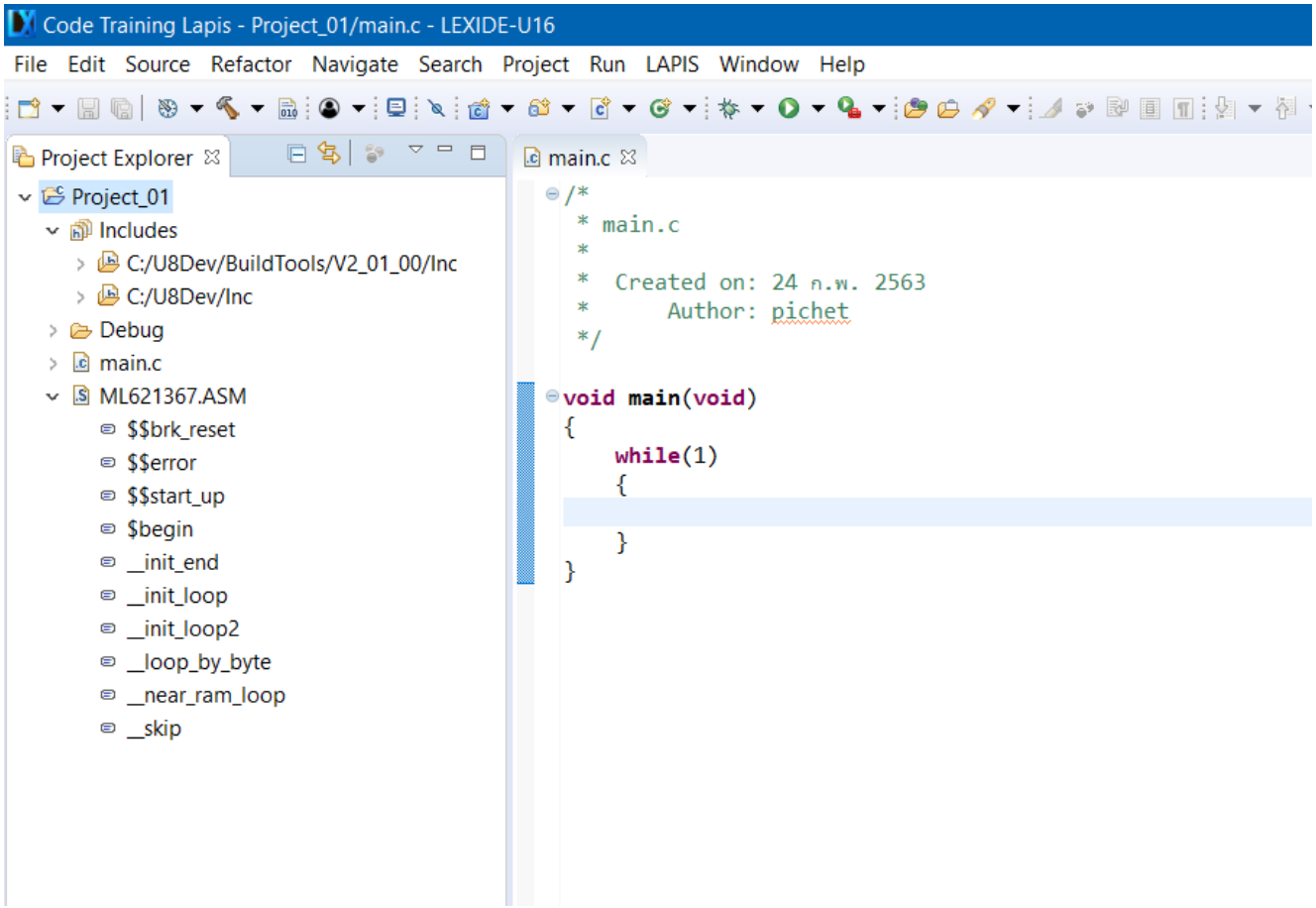
# LEXIDE-u16 (Eclipse base IDE)




Pichet Sriphomma pichet@es.co.th

# LEXIDE-u16 (Eclipse base IDE)

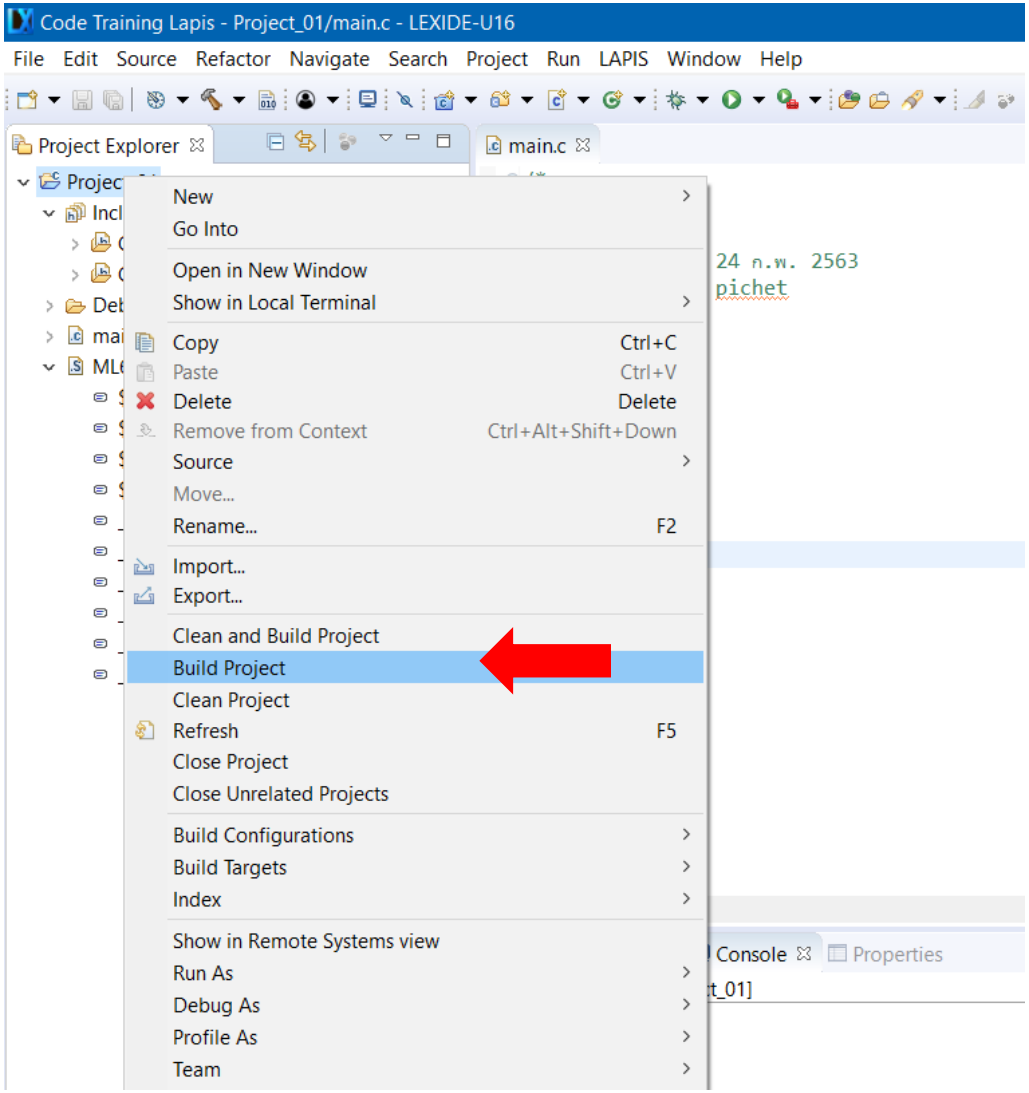
Next, create a main function entry in the created “main.c”.
After that **click Save**.






Pichet Sriphomma pichet@es.co.th

# LEXIDE-u16 (Eclipse base IDE)

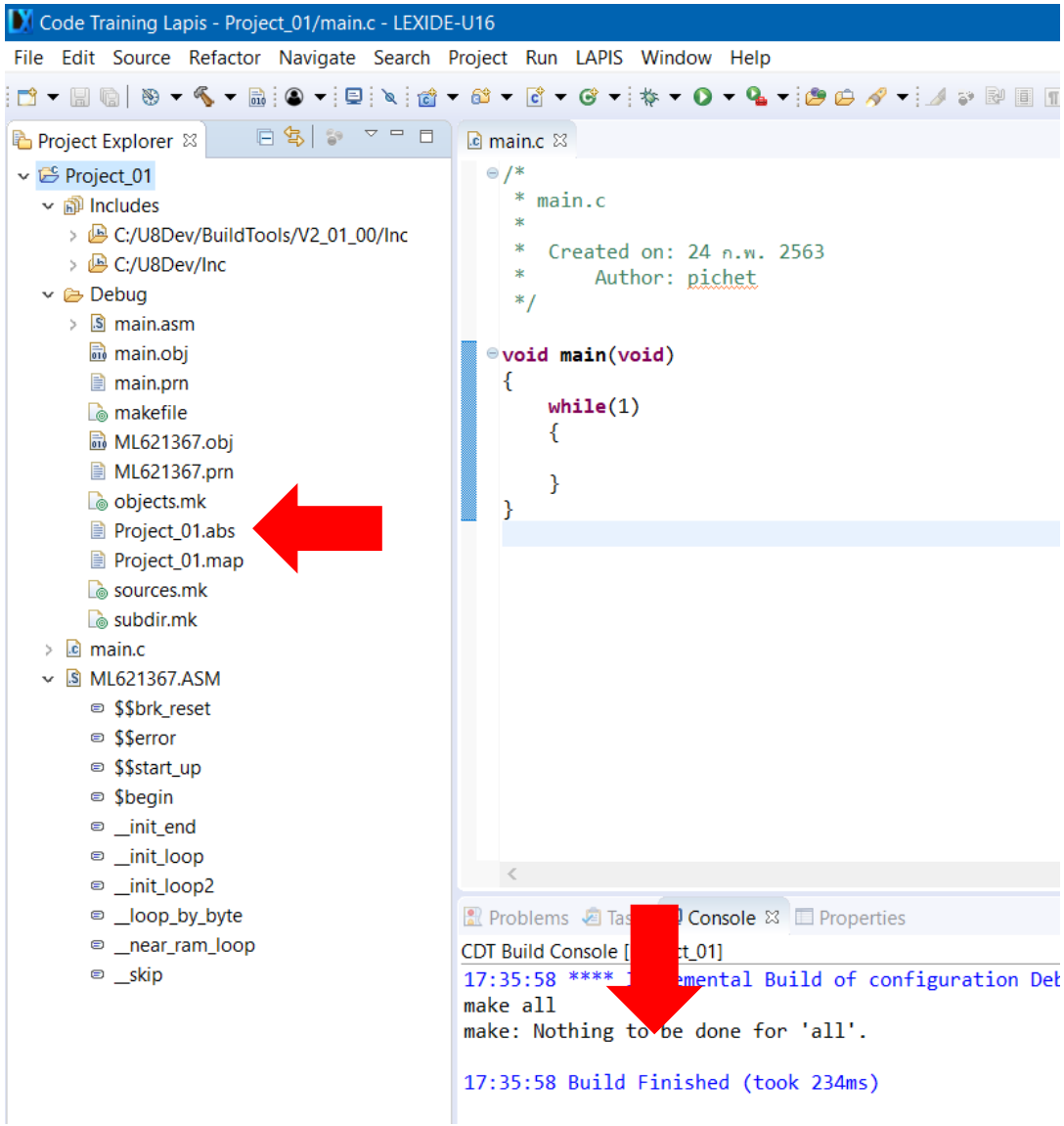



Right-click on a project folder and select [Build Project] to start the build process.



Pichet Sriphomma pichet@es.co.th

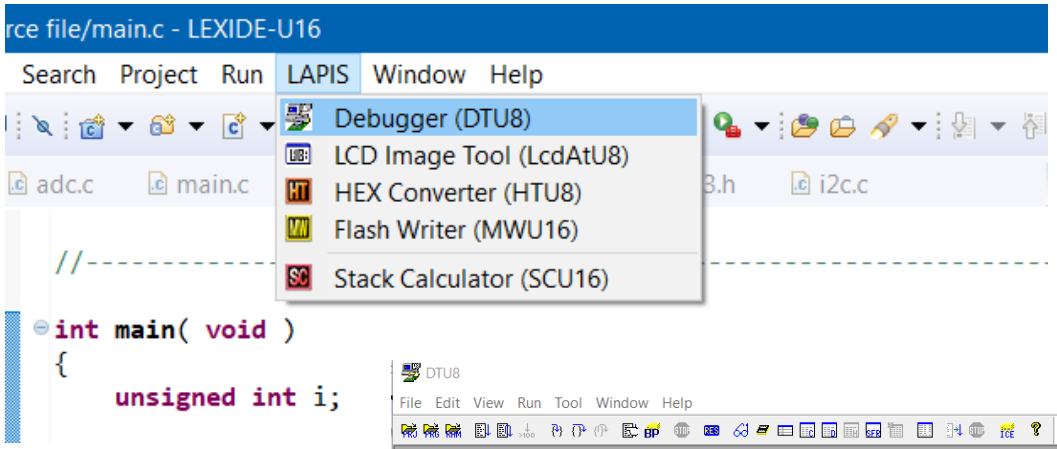
# LEXIDE-u16 (Eclipse base IDE)

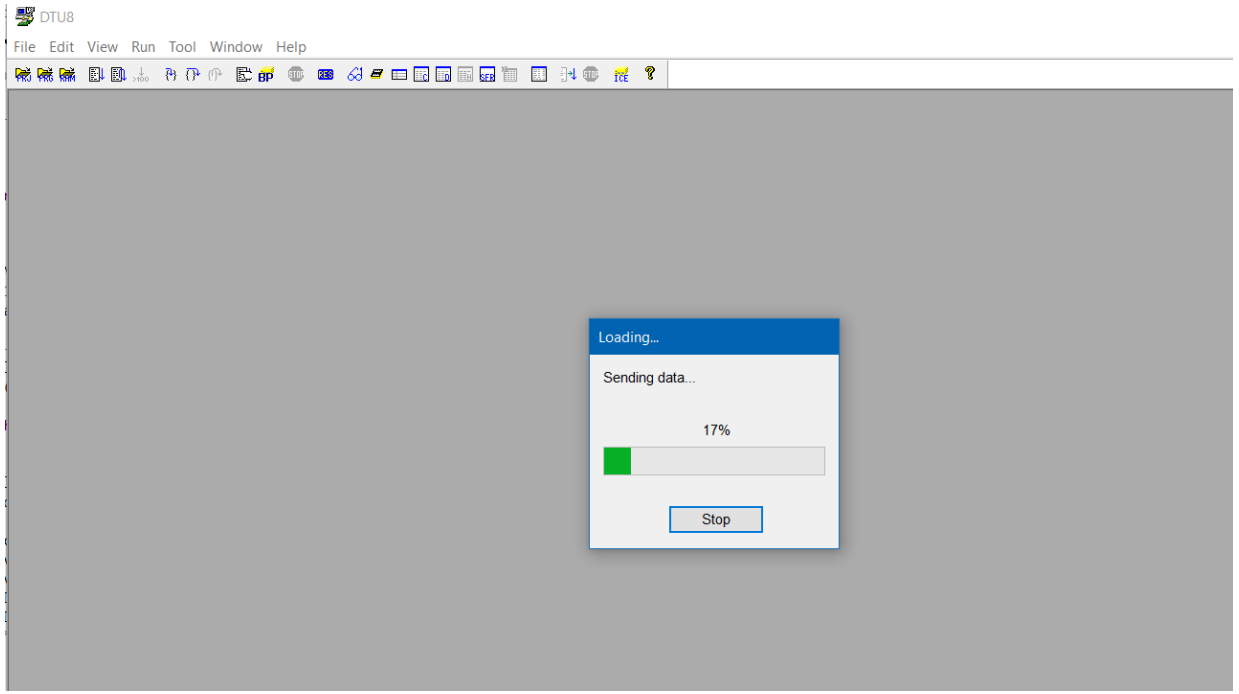



When the build succeeds , an ABS file is generated.

Pichet Sriphomma pichet@es.co.th

# LEXIDE-u16 (Eclipse base IDE)

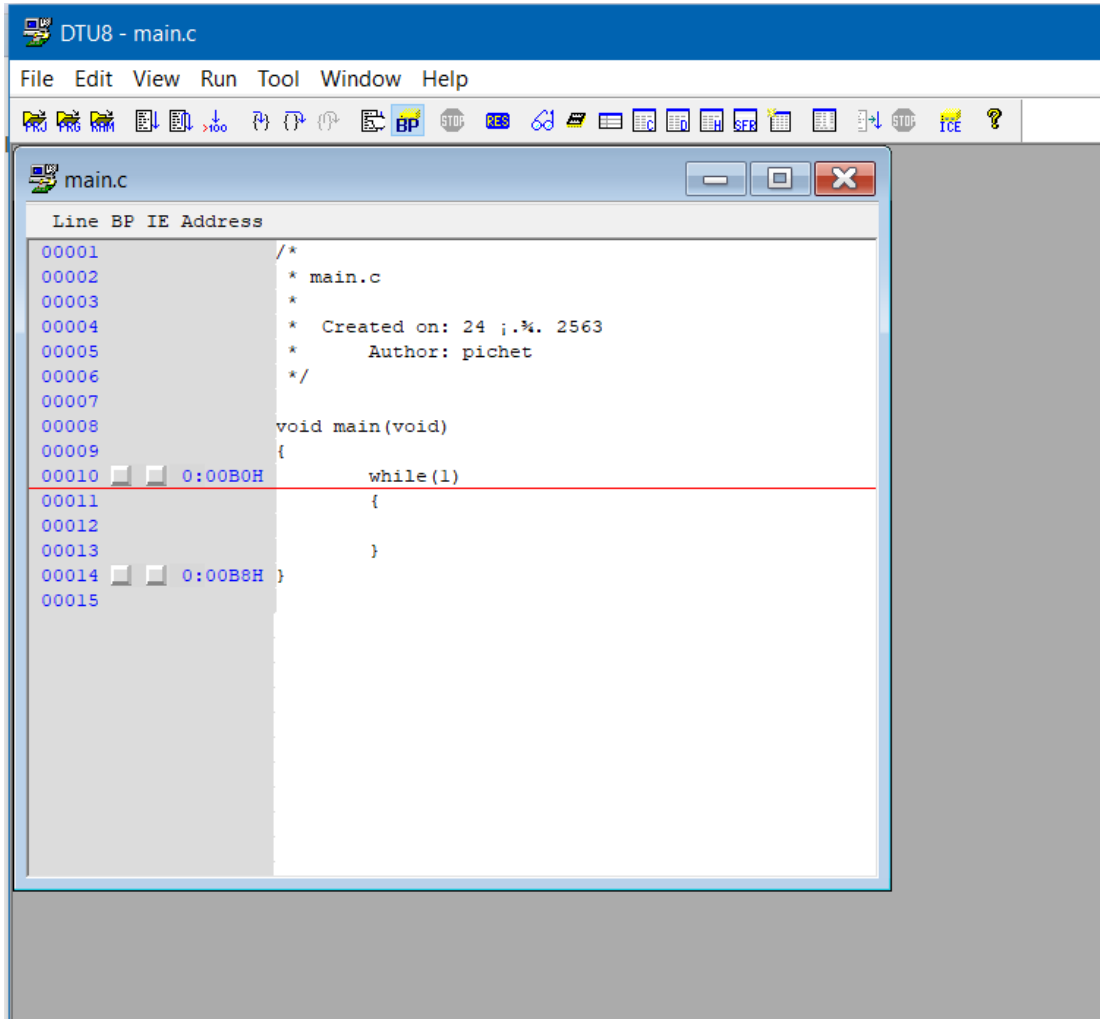



After that Click LAPIS and Choose Debugger (DTU8)

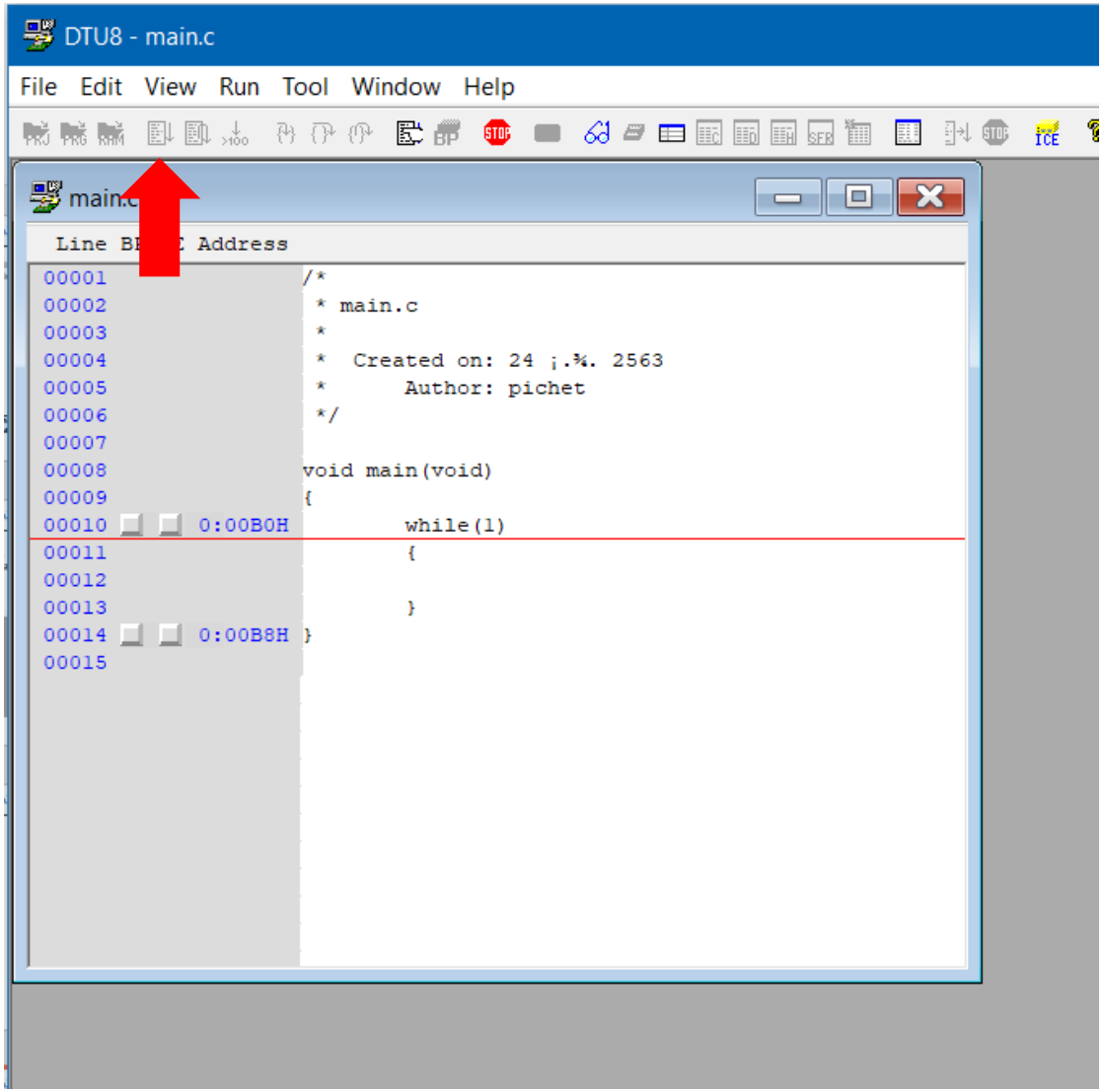
Pichet Sriphomma pichet@es.co.th

# LEXIDE-u16 (Eclipse base IDE)




Pichet Sriphomma pichet@es.co.th

# LEXIDE-u16 (Eclipse base IDE)




Click Run program

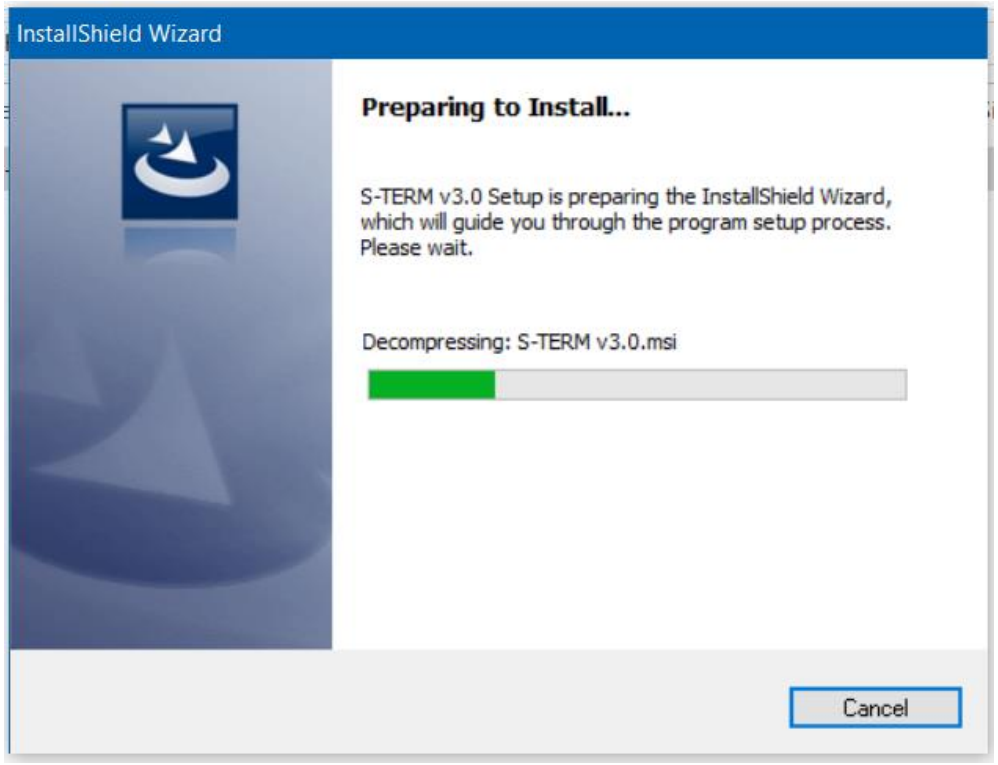
Back to Overview

Pichet Sriphomma pichet@es.co.th

# S-TERM Install

S-TERM_v30_setup

Double Click File “S-TERM_v30_setup”

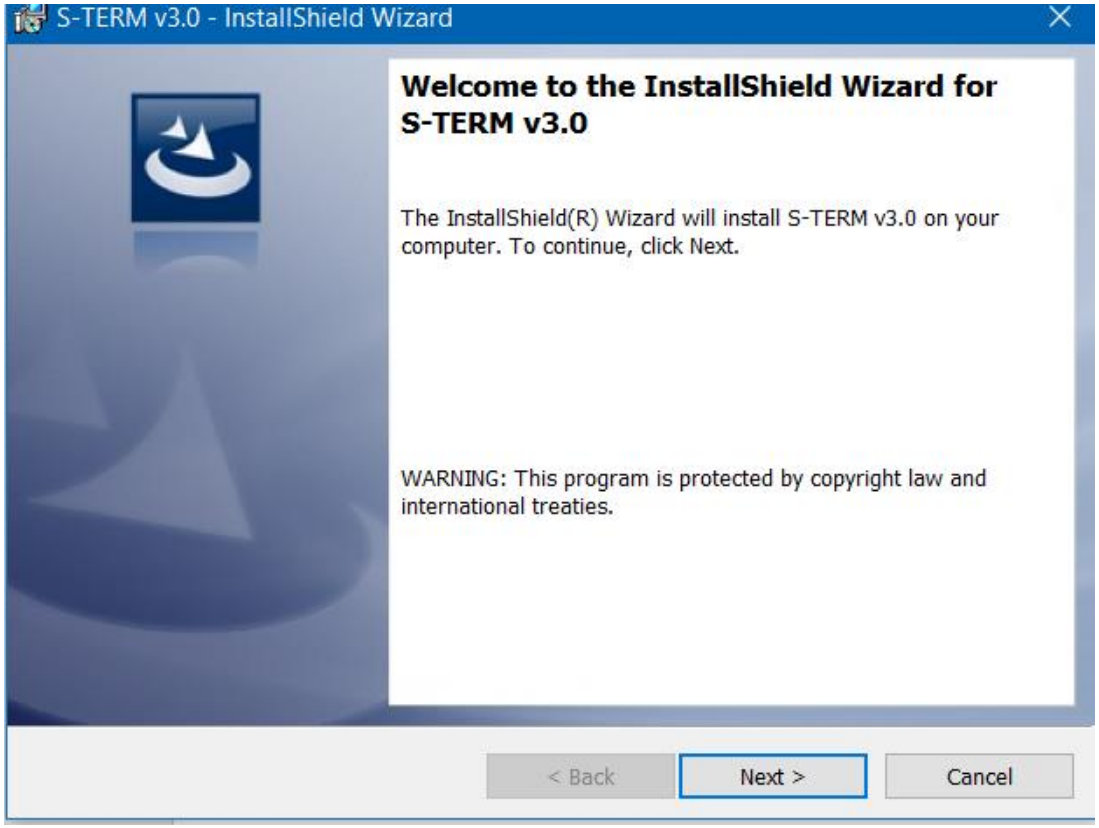



After that Appear new window



Pichet Sriphomma pichet@es.co.th

# S-TERM Install

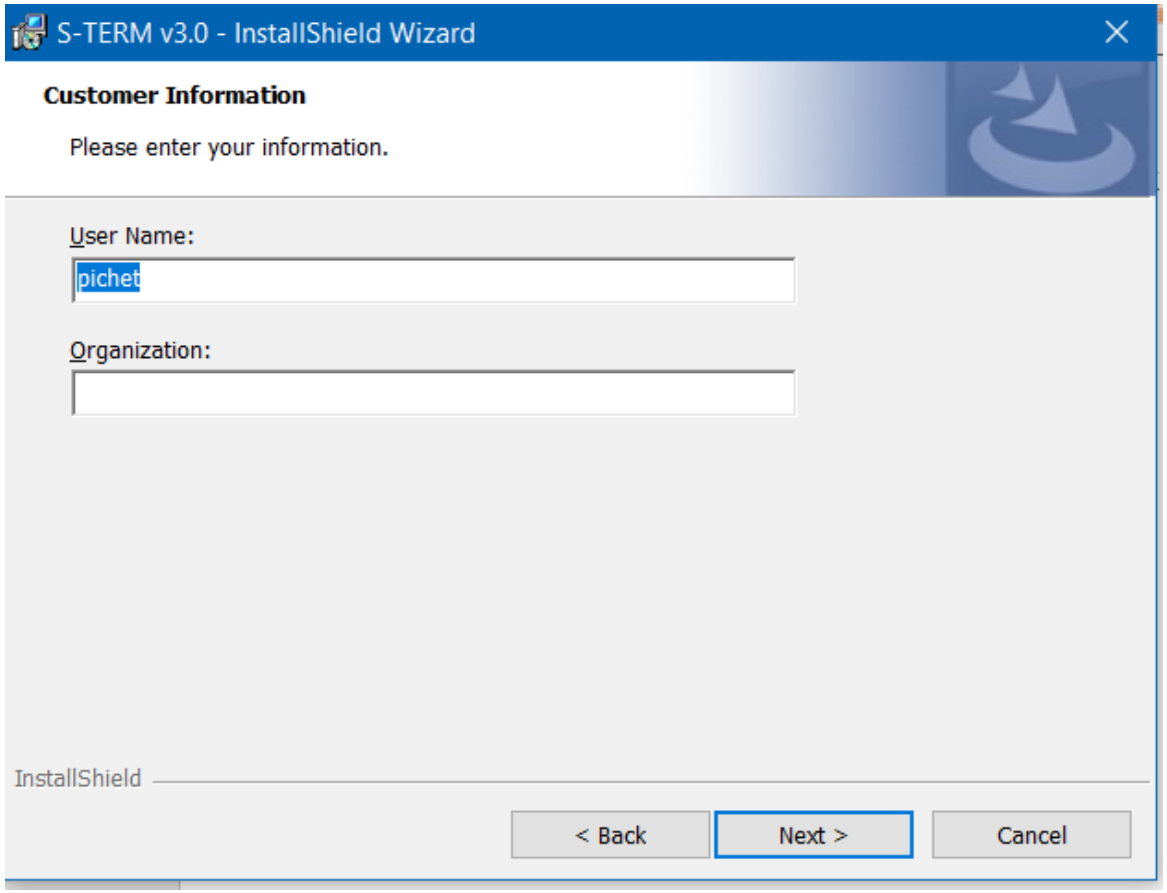



Click Next >



Pichet Sriphomma pichet@es.co.th

# S-TERM Install

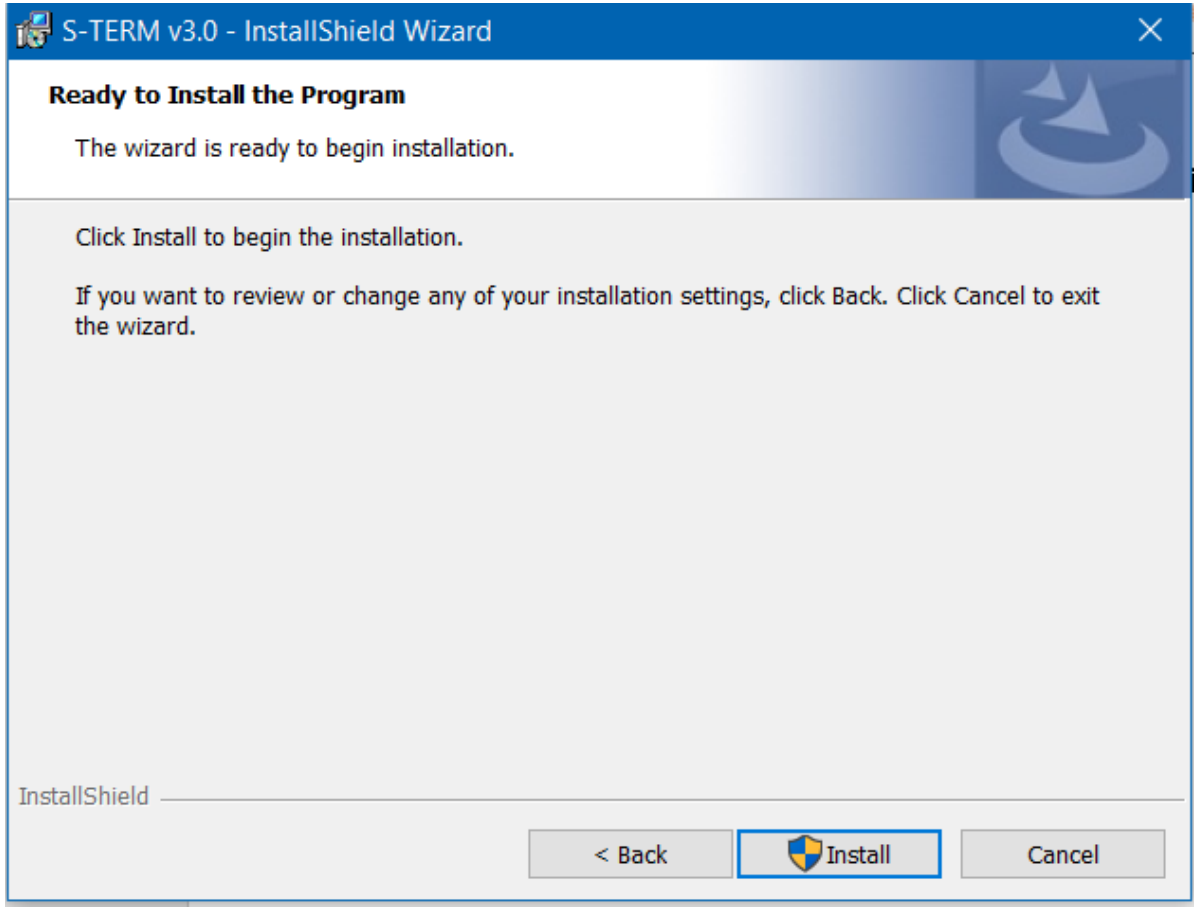



Click Next >



Pichet Sriphomma pichet@es.co.th

# S-TERM Install

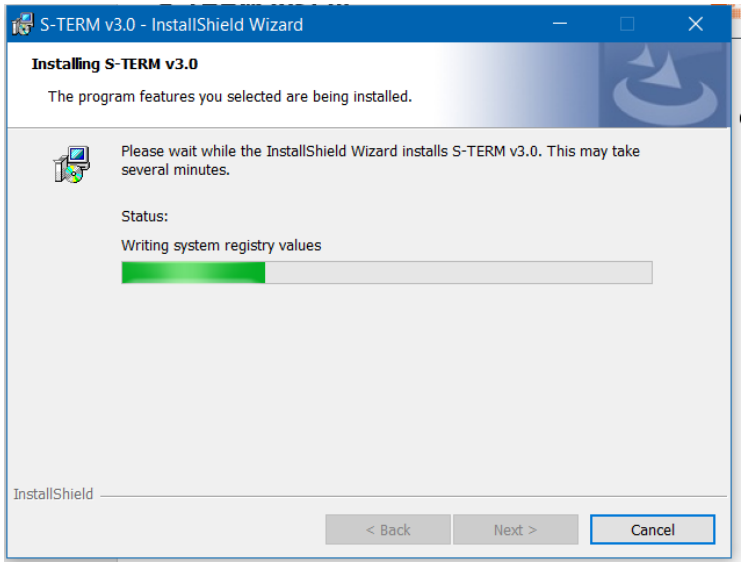



Click Install

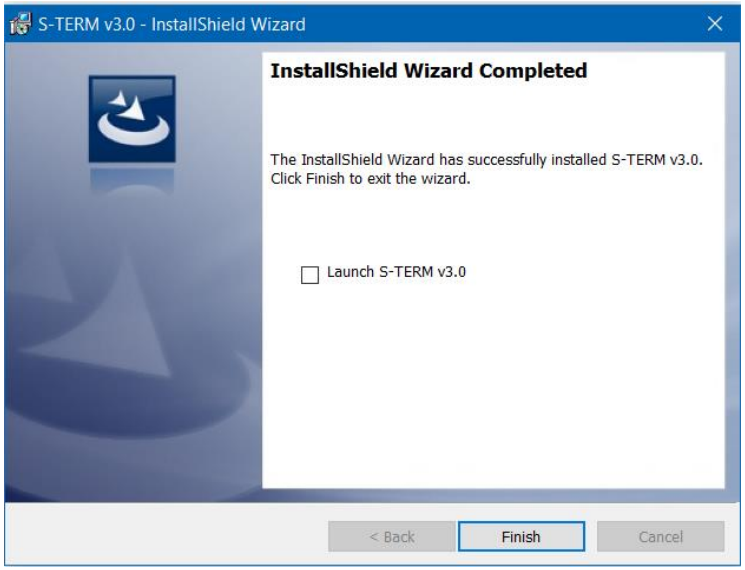


Pichet Sriphomma pichet@es.co.th

# S-TERM Install

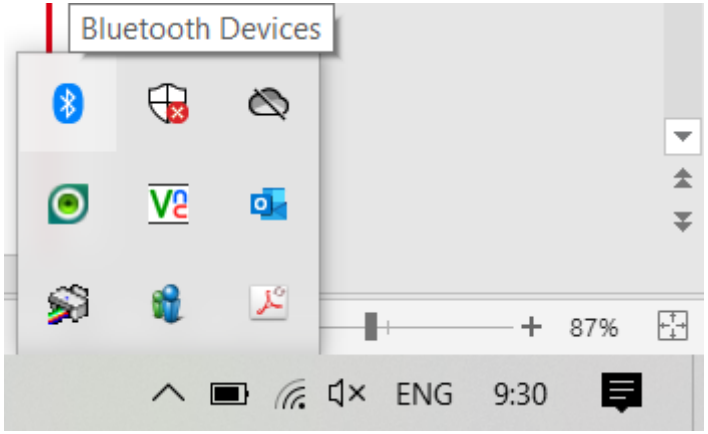



Wait for Install




Click Finish

Pichet Sriphomma pichet@es.co.th

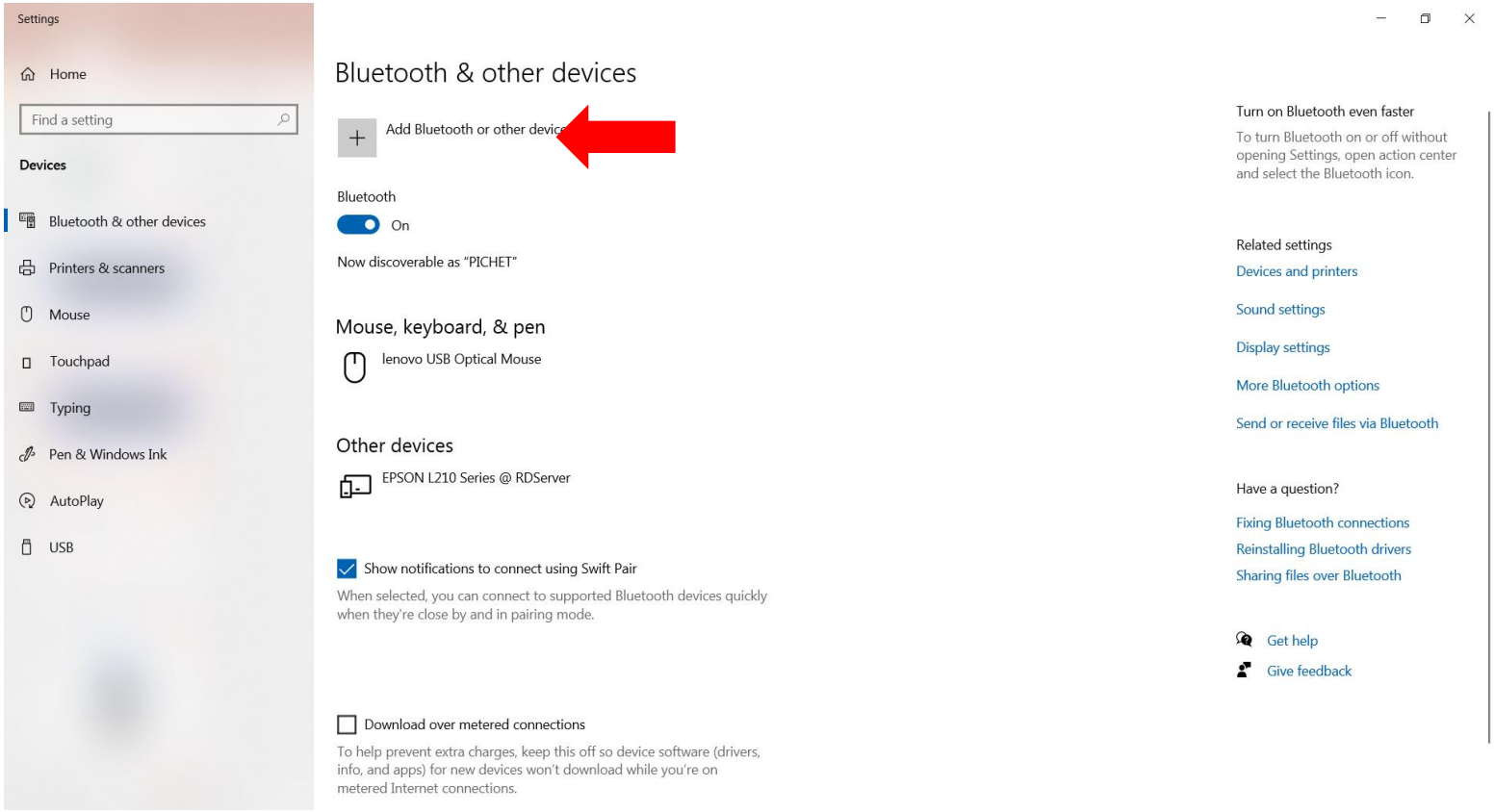



Before launching the program S-TERM.
Must connect device.
Open Bluetooth & other devices



Pichet Sriphomma pichet@es.co.th

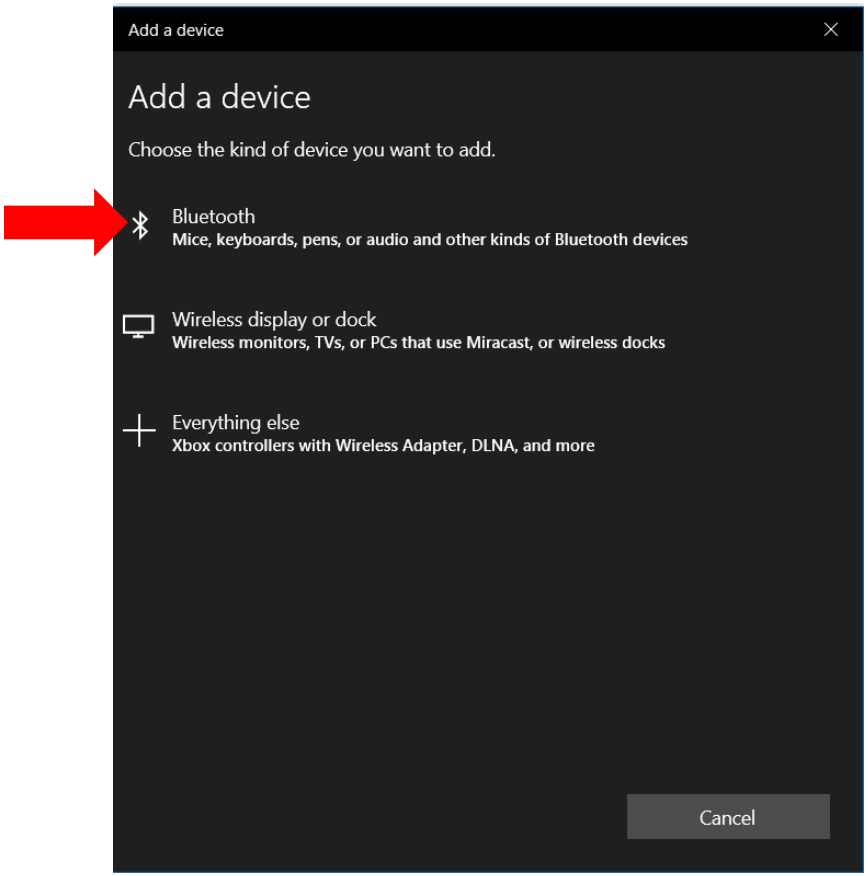
# Click Add Bluetooth or other device

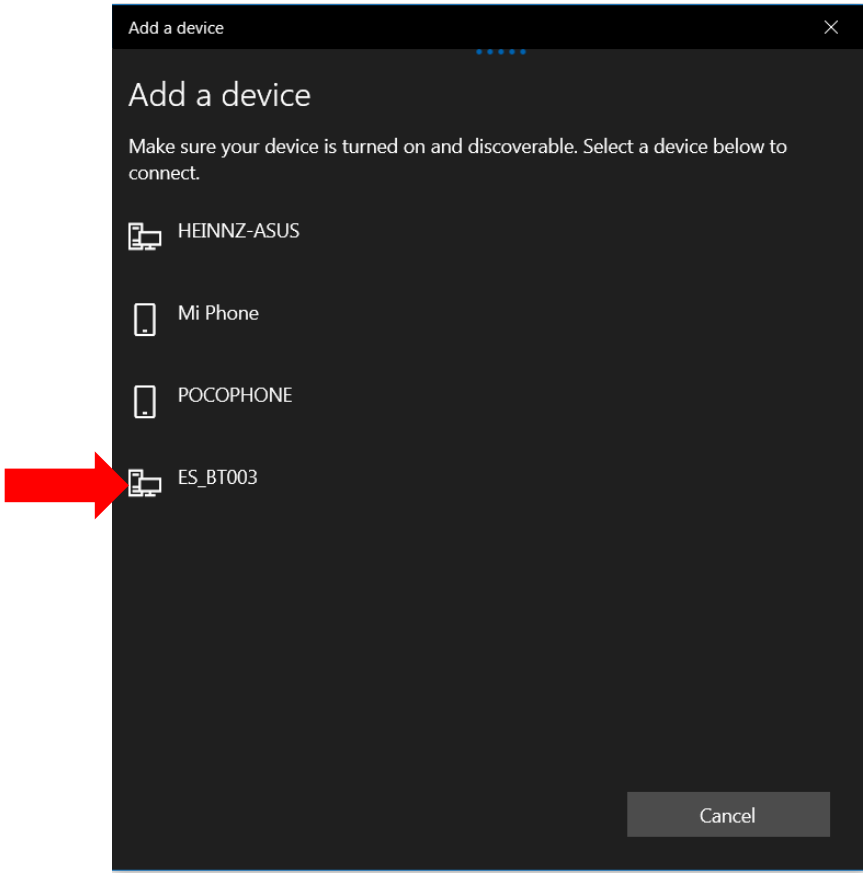





Pichet Sriphomma pichet@es.co.th

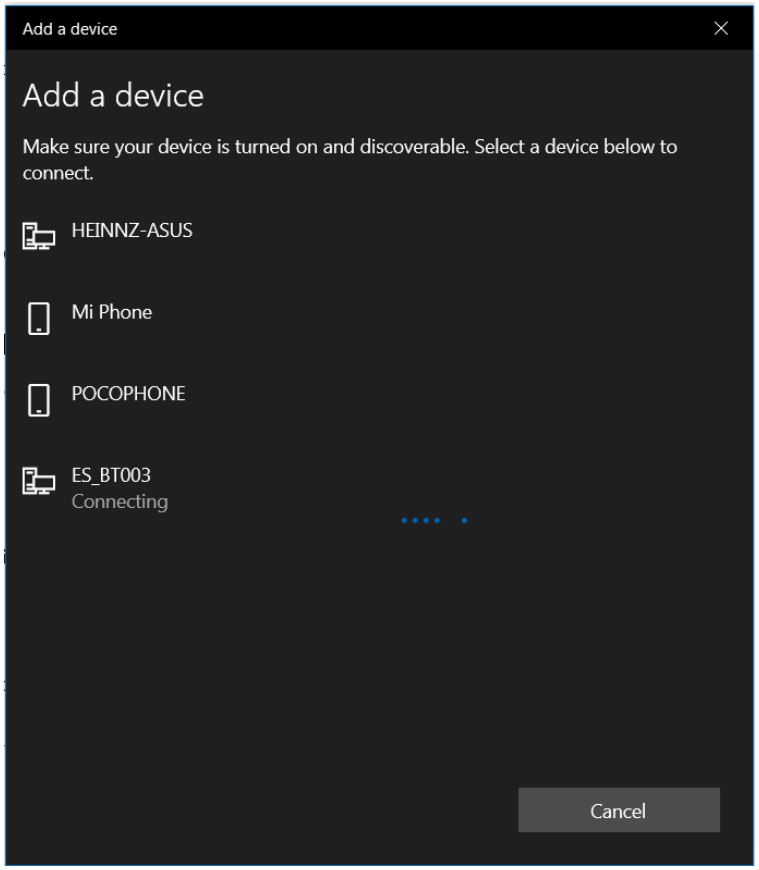
# Click Bluetooth and select device

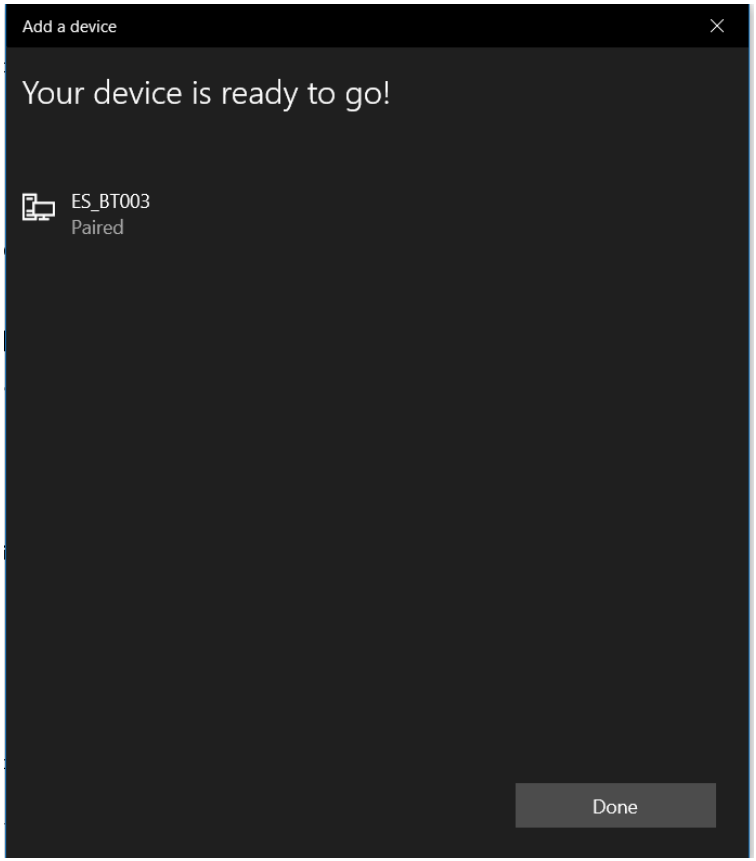



Pichet Sriphomma pichet@es.co.th

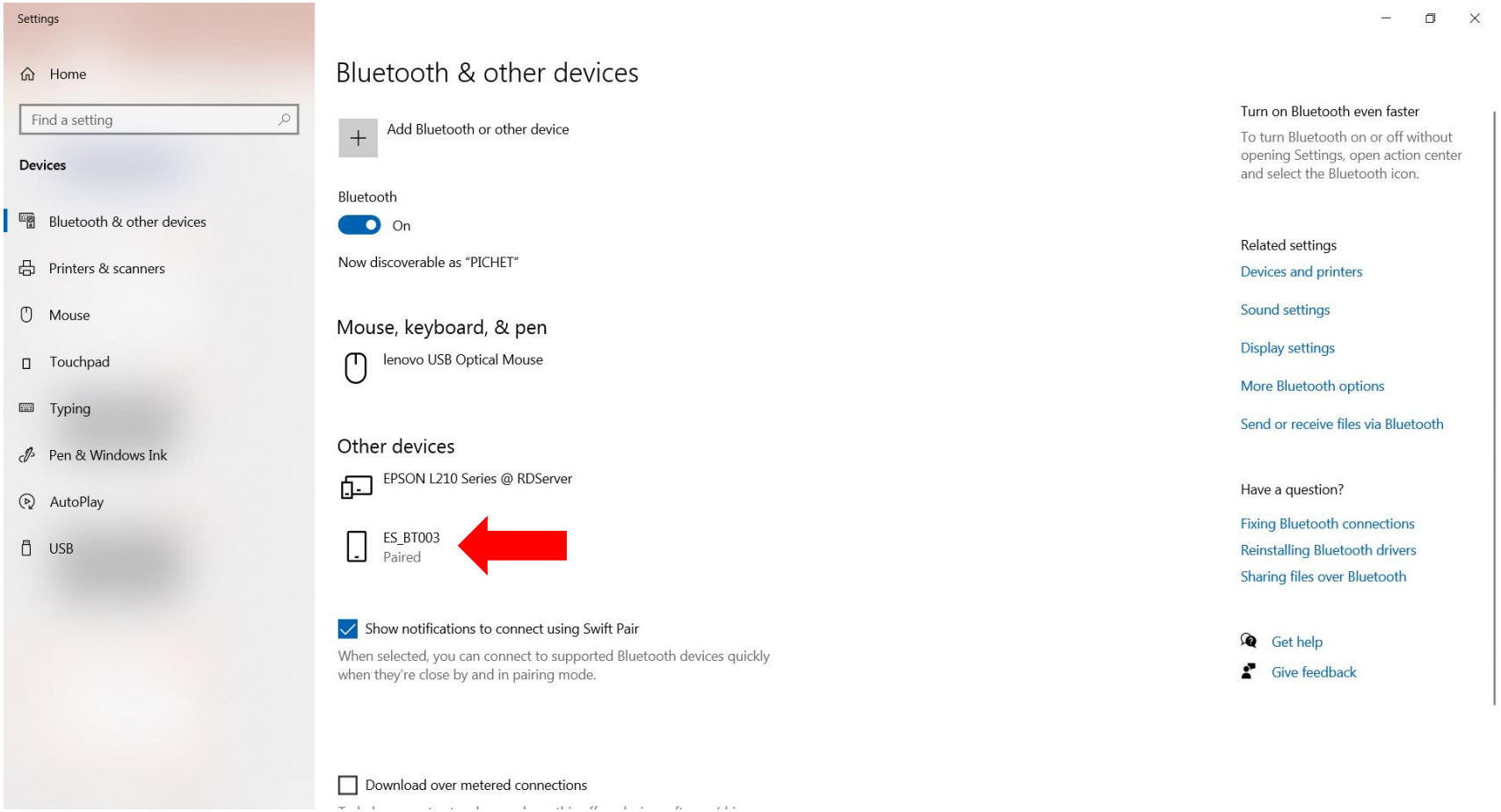
Wait for connecting..




Pichet Sriphomma pichet@es.co.th

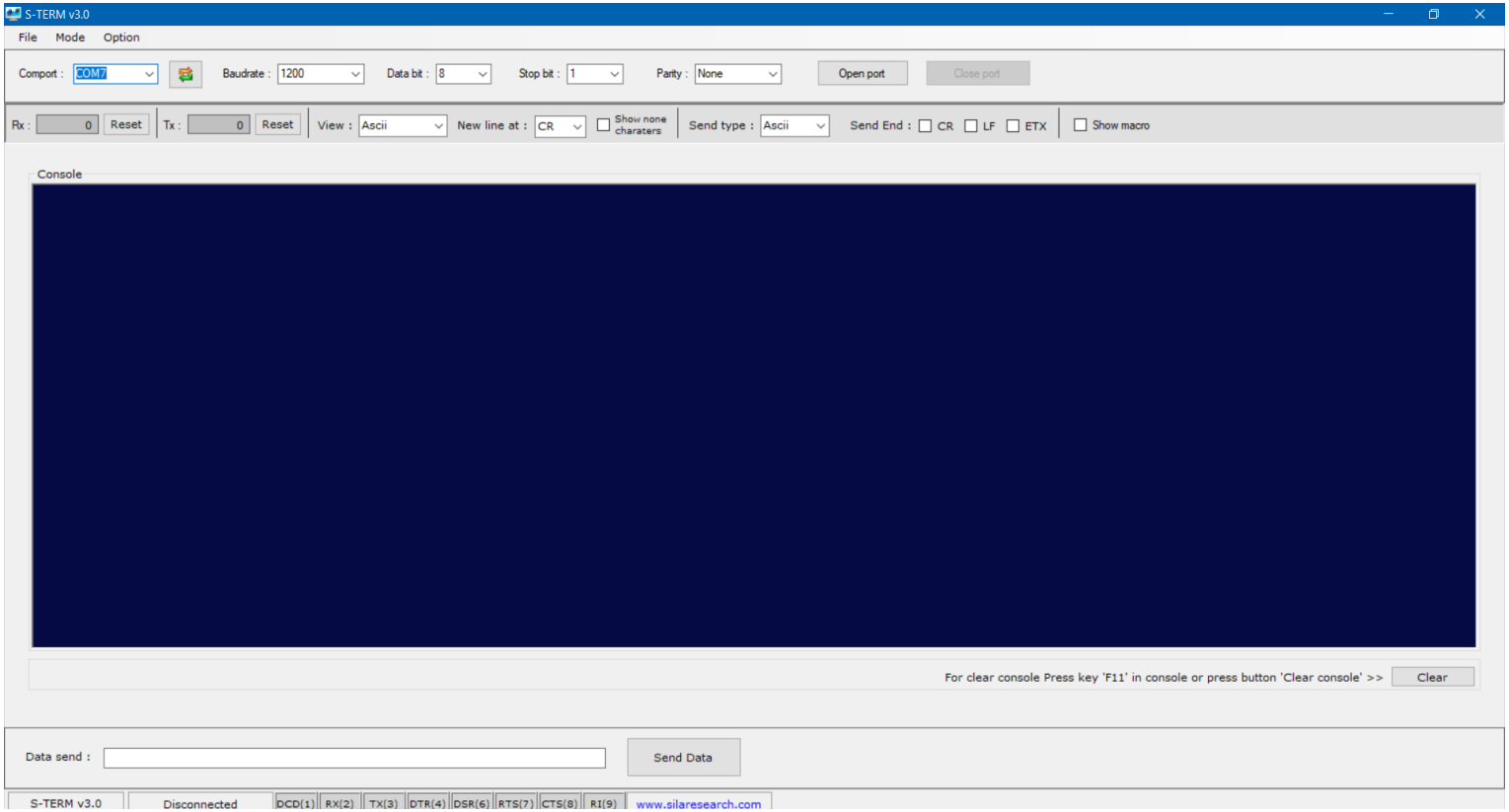
# Show device connected






Pichet Sriphomma pichet@es.co.th

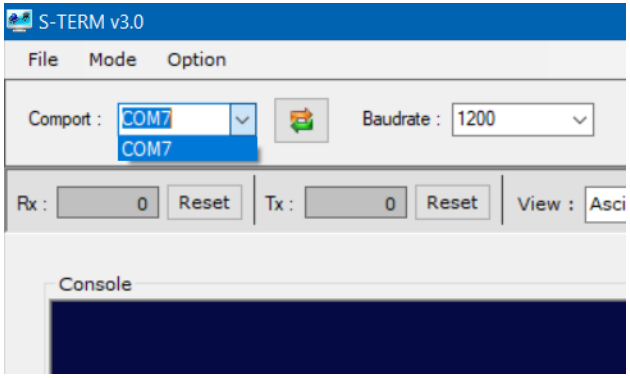
## Open Program S-TERM

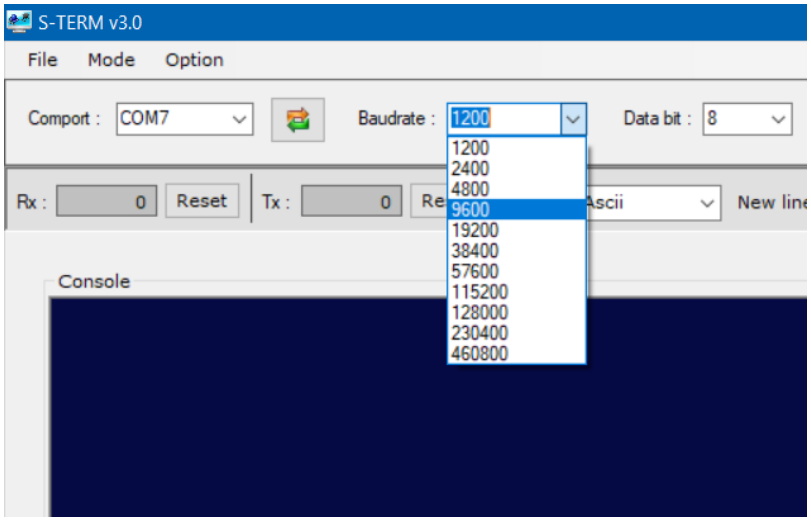



Pichet Sriphomma pichet@es.co.th

# S-TERM




Select Comport



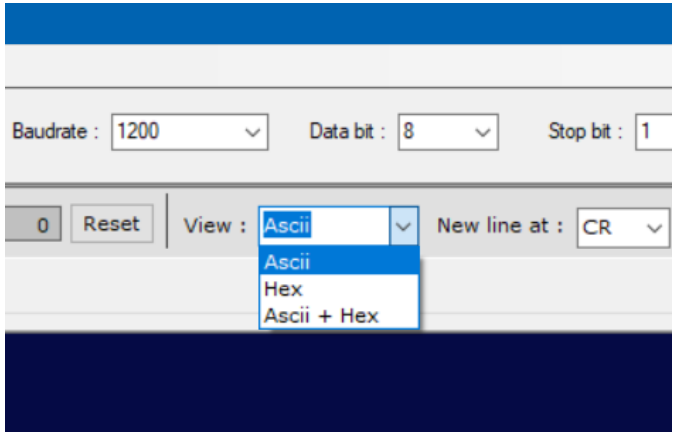

Select Baudrate.

This workshop select 9600.



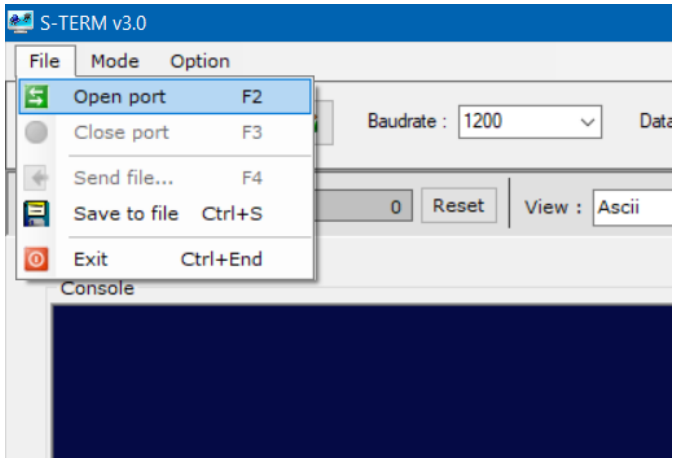
Pichet Sriphomma pichet@es.co.th

# S-TERM




Select view type.

This workshop select Ascii.



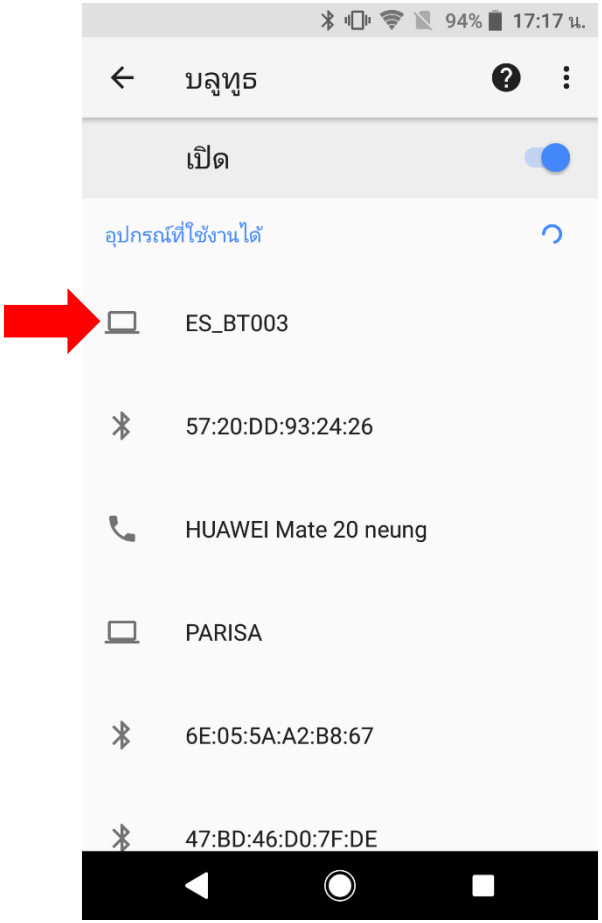

Click File and select Open port.

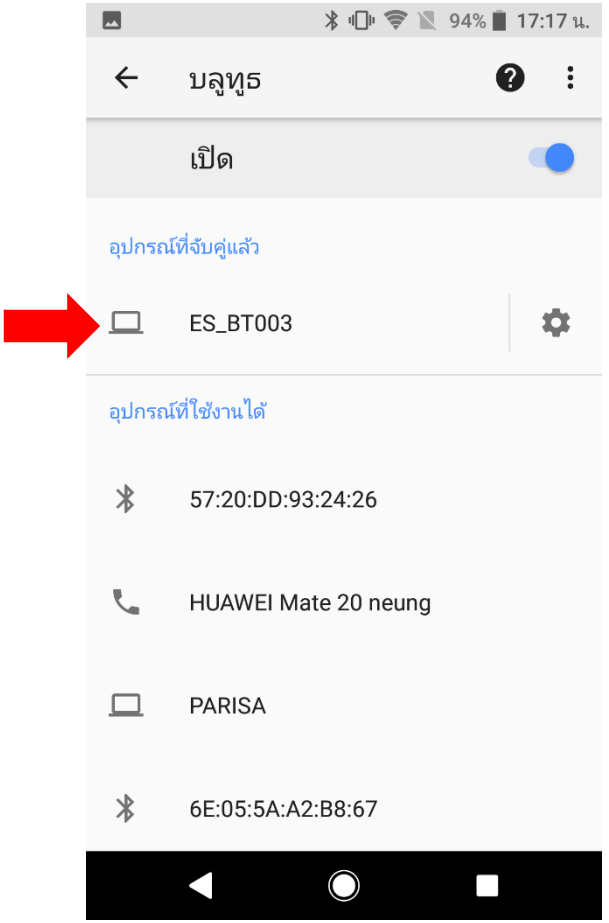
Back to Overview

Pichet Sriphomma pichet@es.co.th

## Using the application

### Step 1 Connect MCU Board with Bluetooth






Pichet Sriphomma pichet@es.co.th

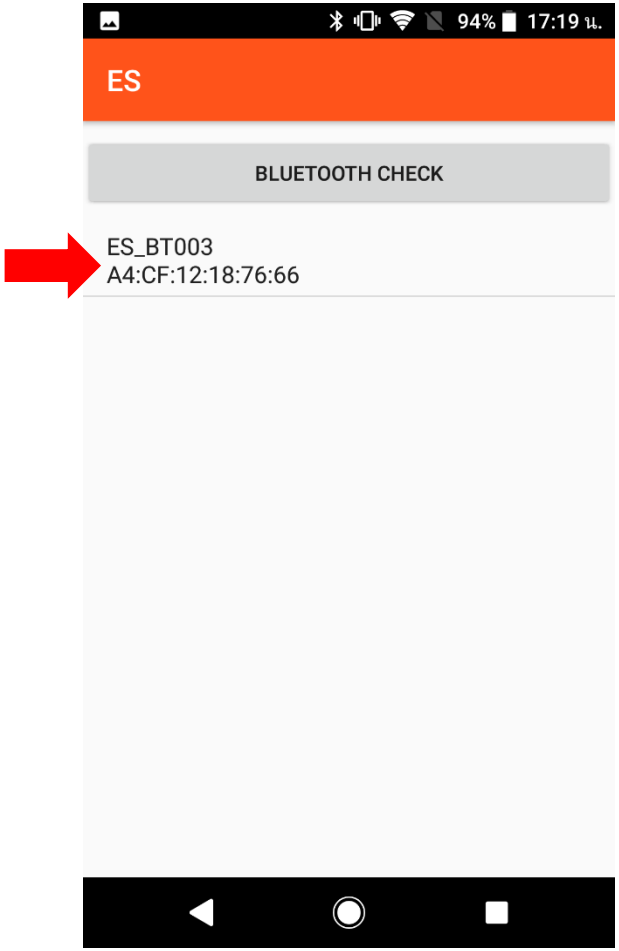
## Using the application

### Step 2 Open application and click “Bluetooth Check”


Pichet Sriphomma pichet@es.co.th

## Using the application

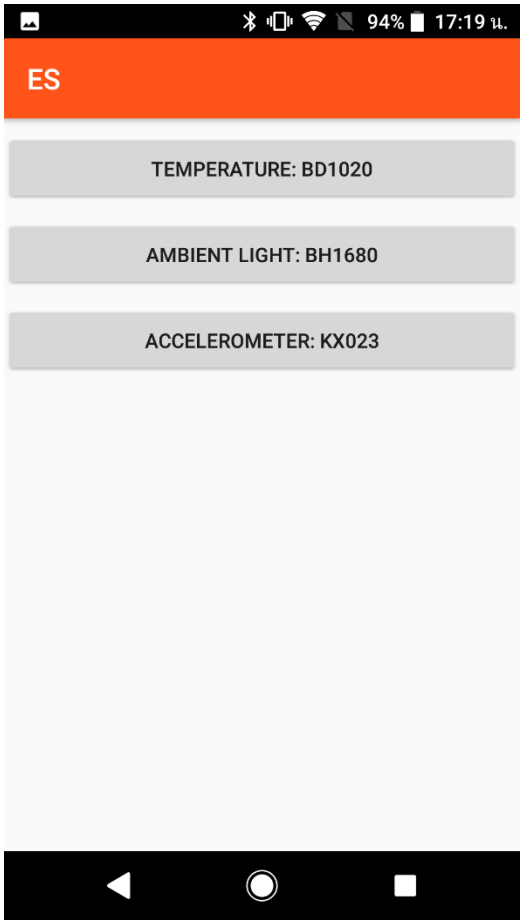
### Step 3 Choose device and connect




Pichet Sriphomma pichet@es.co.th

## Using the application

### Step 4 After connected this app Show button to choose




Back to Overview

Pichet Sriphomma pichet@es.co.th

# LAPIS MCU Workshop Training



Pichet Sriphomma pichet@es.co.th

# AGENDA

1. GPIO
2. UART
3. ADC
4. I2C



Pichet Sriphomma pichet@es.co.th

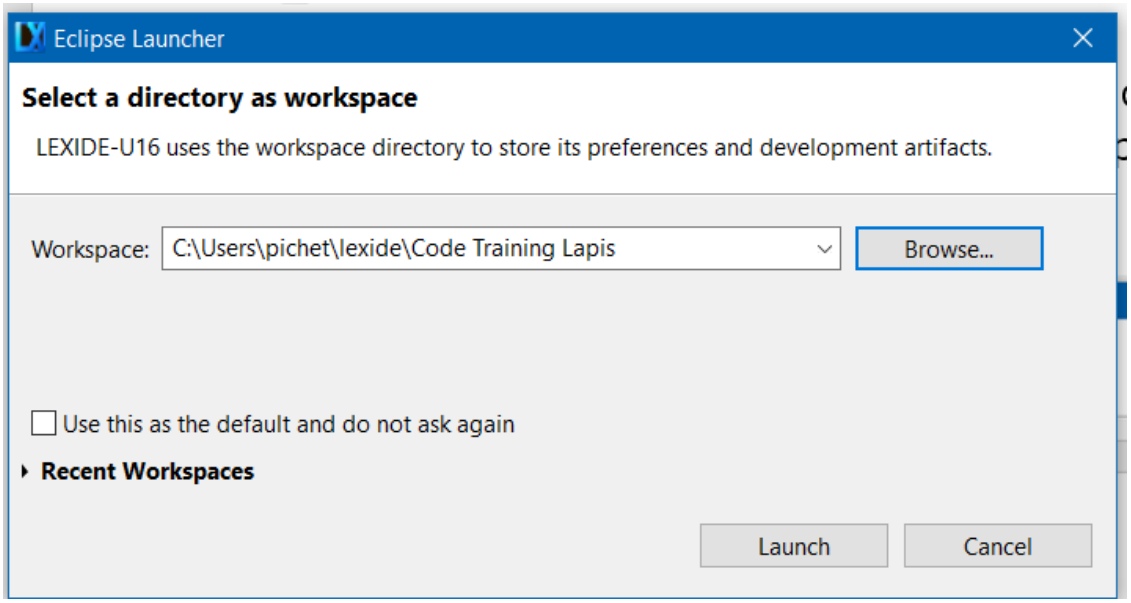
# 1. GPIO



Pichet Sriphomma pichet@es.co.th


Open Program LEXIDE-U16

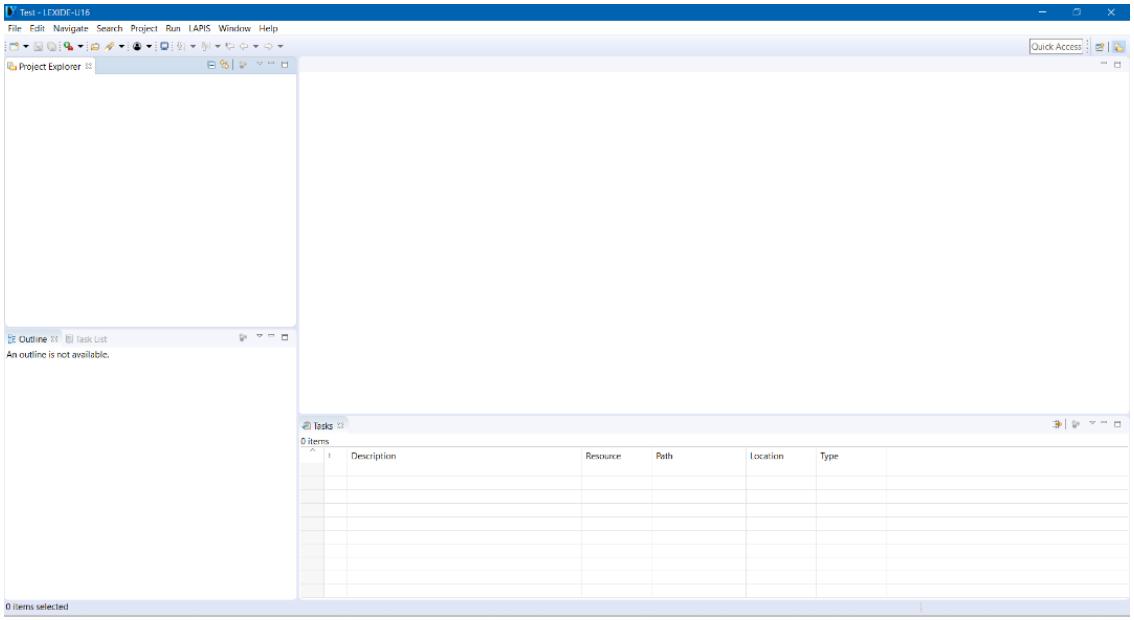
When clicked, the following workspace setting dialog box will be output. Set a path to workspace at [Workspace]. After that Click [Launch].





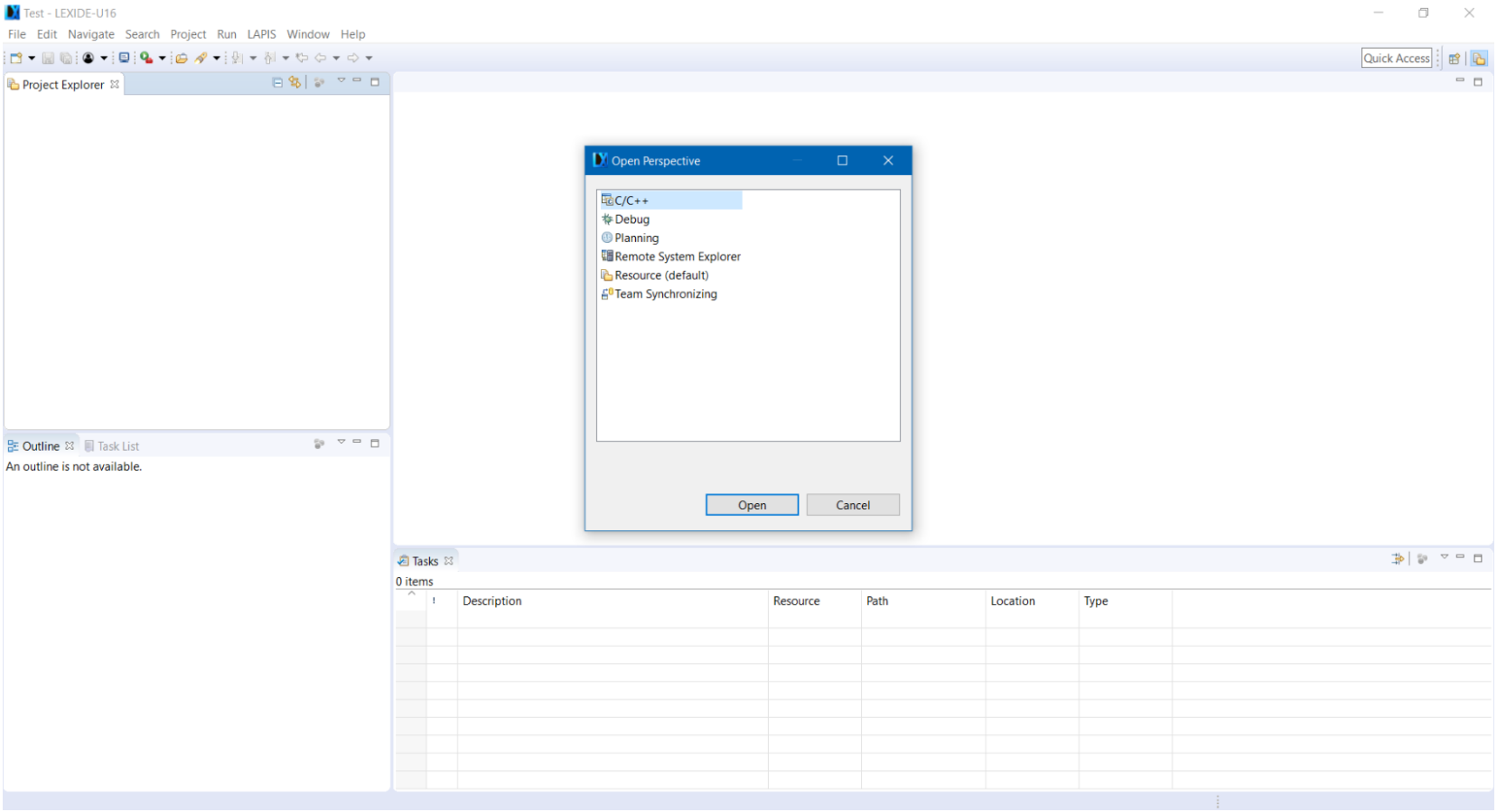

Pichet Sriphomma pichet@es.co.th


The first window after opening the program



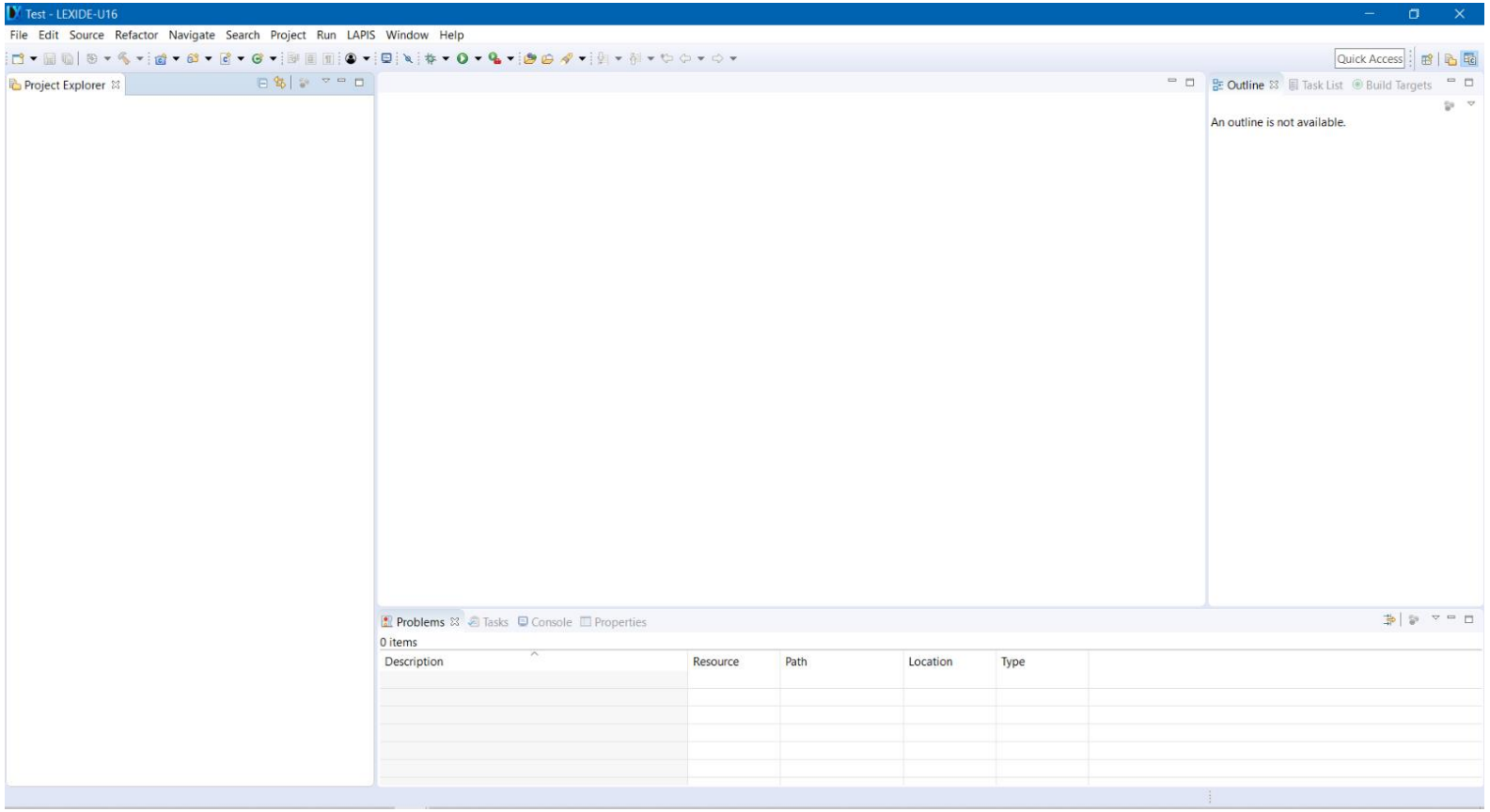

Pichet Sriphomma pichet@es.co.th

Click the button in the upper right  to display the [Open Perspective] dialog. Select [C/C++] and Click [OK].



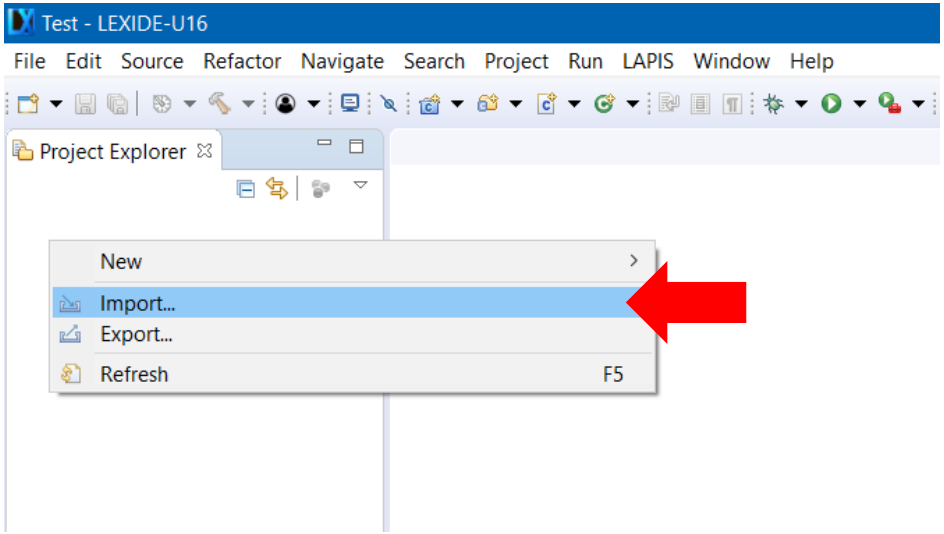

Pichet Sriphomma pichet@es.co.th

The Perspective set to [C / C ++].




Pichet Sriphomma pichet@es.co.th

## Import Project

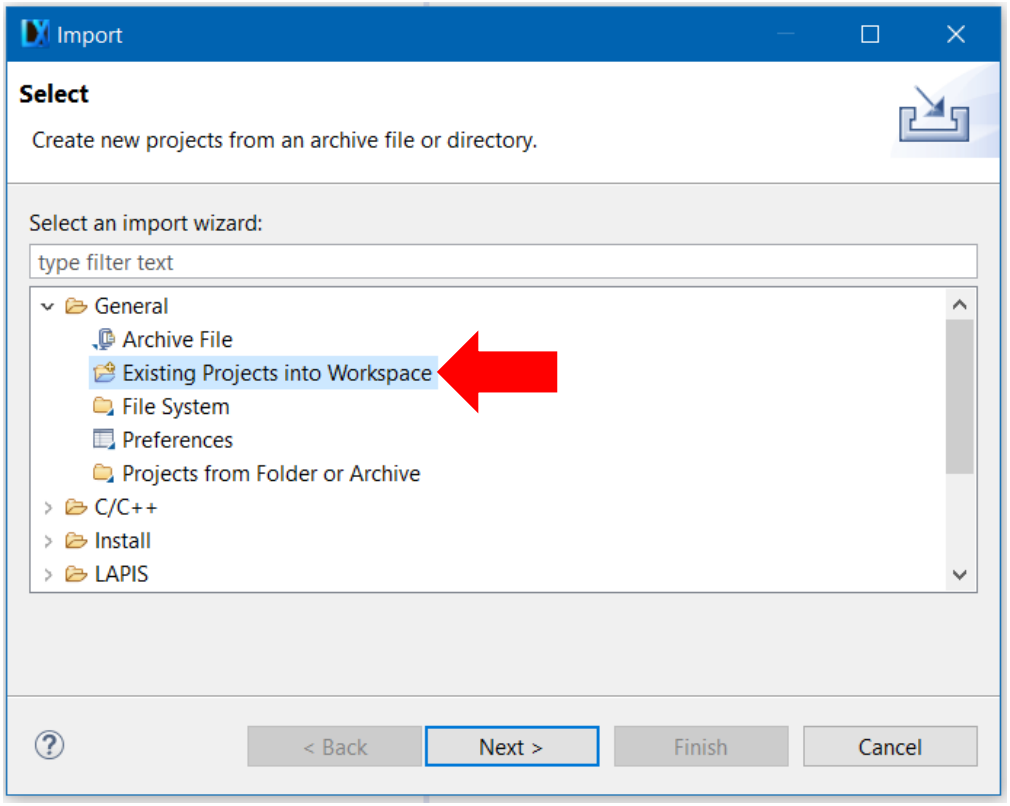



Right-click on project Explorer and select Import.



Pichet Sriphomma pichet@es.co.th

## Import Project

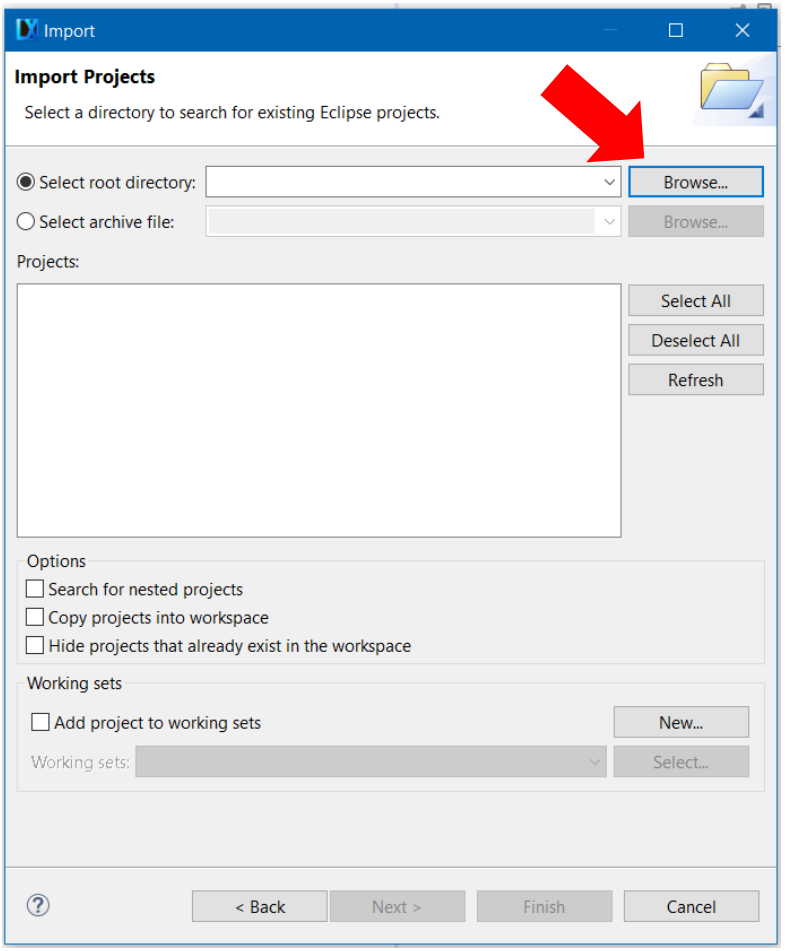



Select General and choose Existing Projects into Workspace. Click Next.

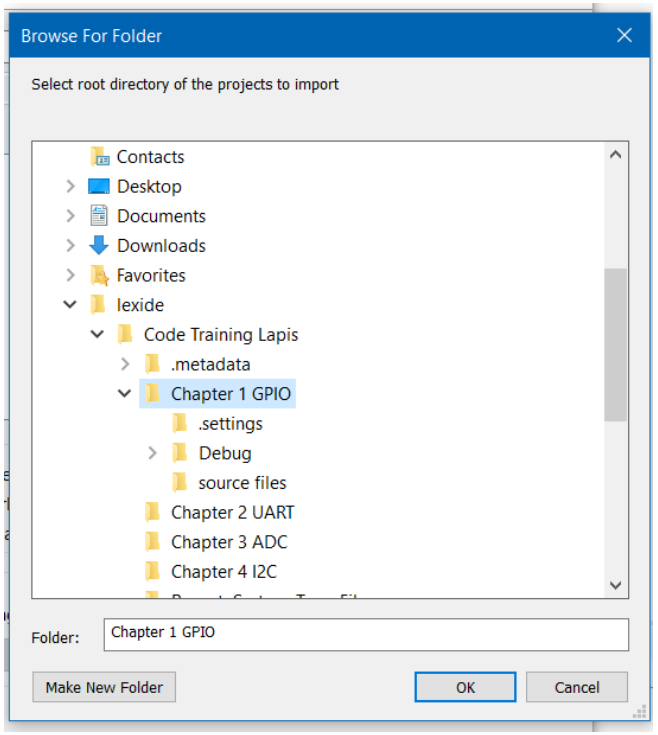


Pichet Sriphomma pichet@es.co.th

## Import Project



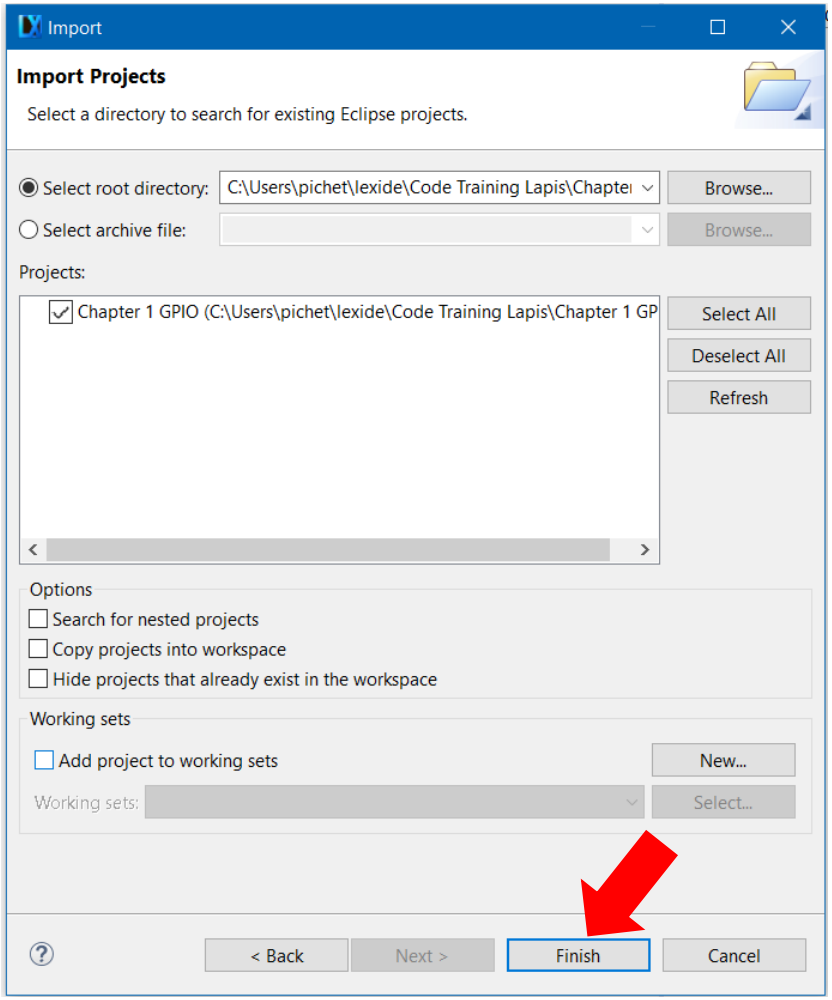

LEXIDE up the new window.
Click Browse.. at Select root directory.
Choose “Chapter 1 GPIO” in Folder window.






Pichet Sriphomma pichet@es.co.th

## Import Project

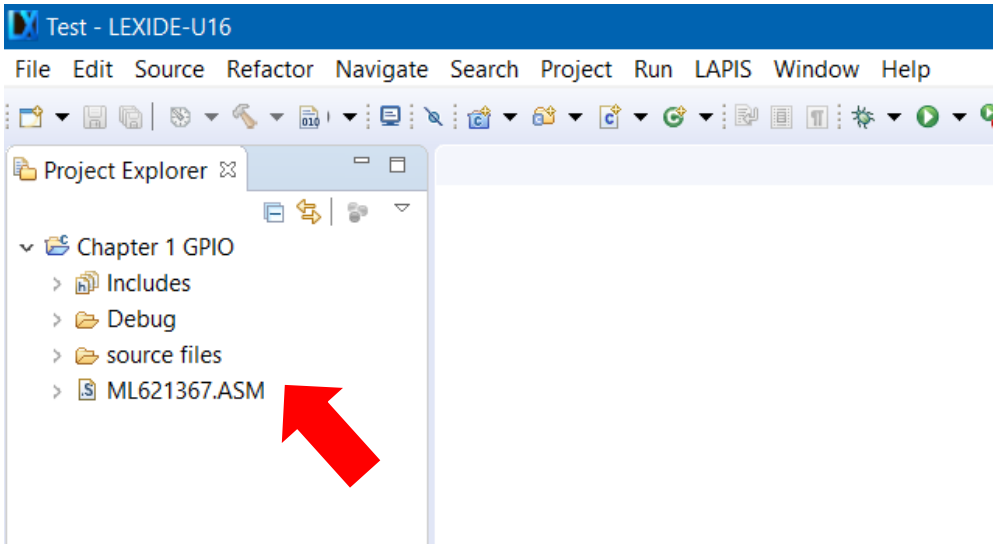



After choosing Project Click Finish.



Pichet Sriphomma pichet@es.co.th

## Import Project

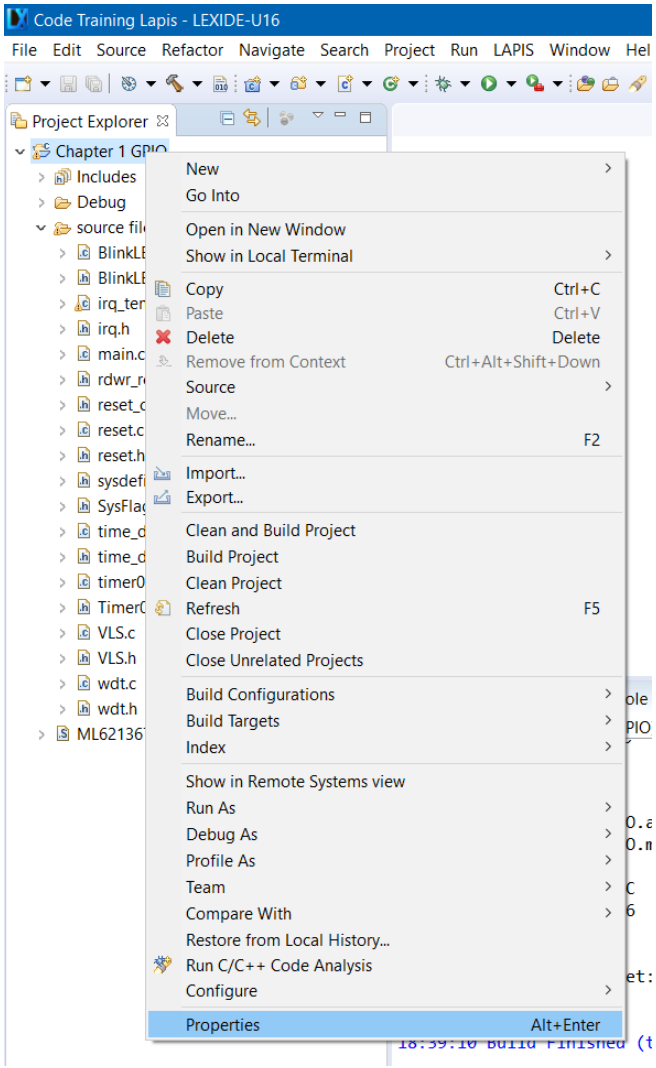



Then appear the project on Project Explorer.



Pichet Sriphomma pichet@es.co.th

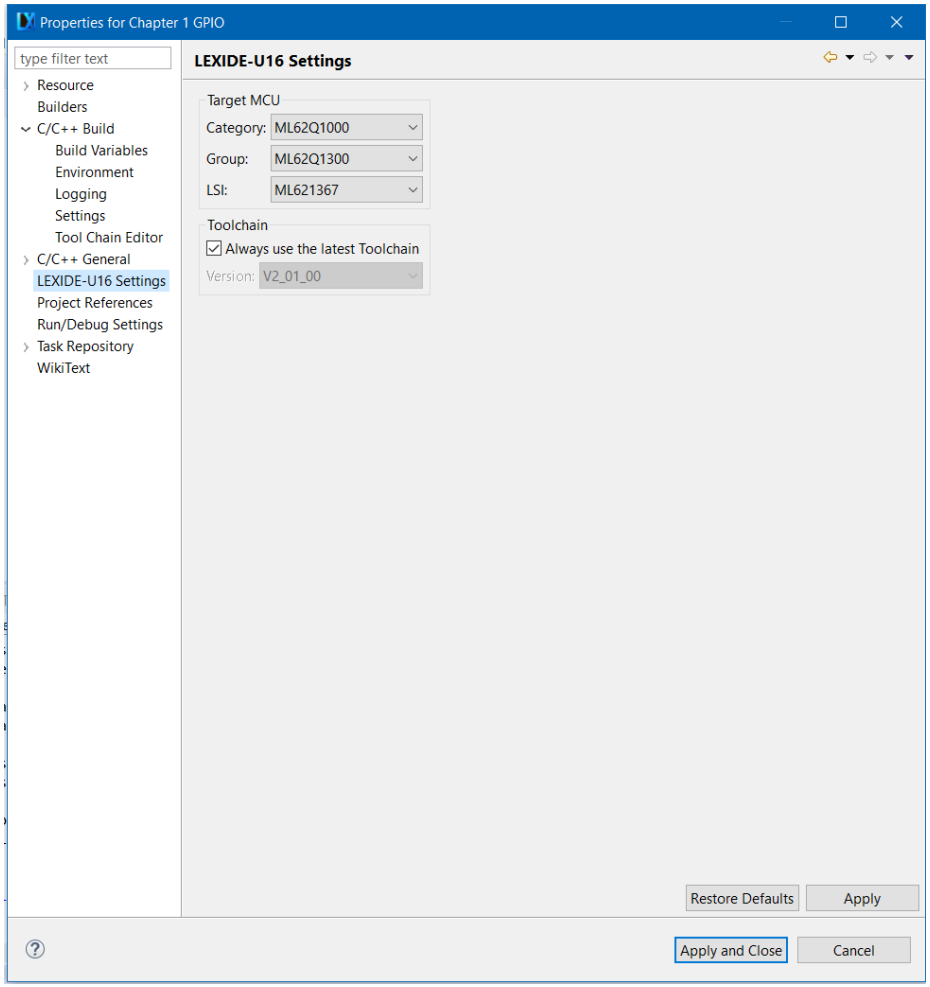
## Check Device




Right-click on a project folder and select [Properties] .

Pichet Sriphomma pichet@es.co.th

## Select Device

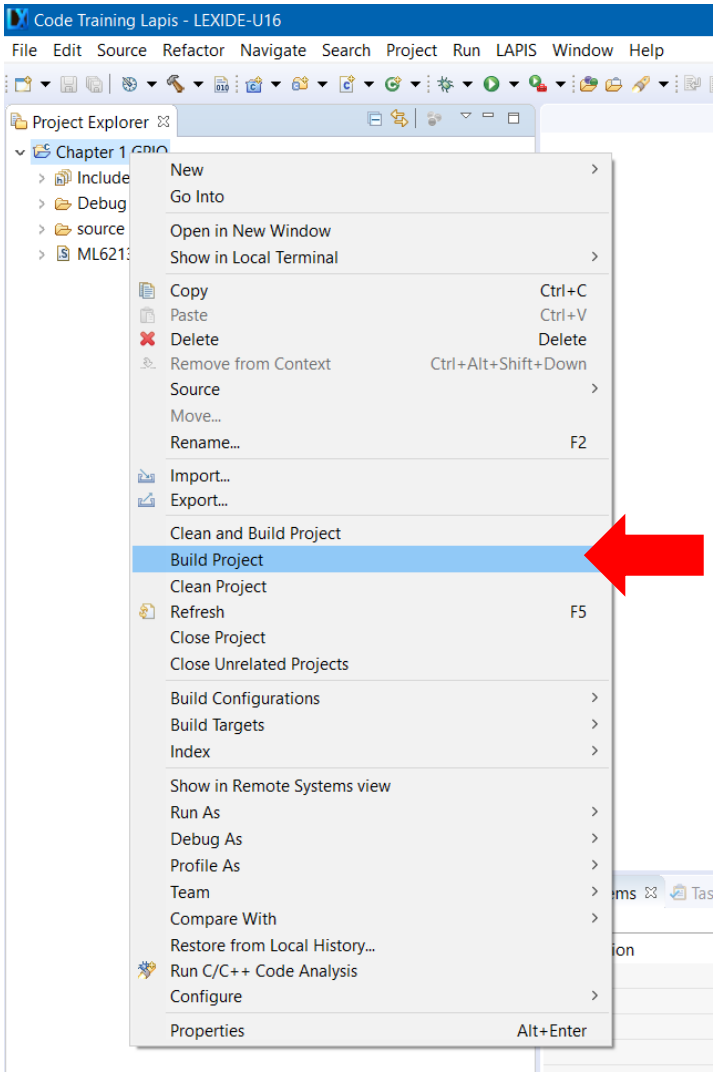



Choose LEXIDE-U16 Settings



Pichet Sriphomma pichet@es.co.th

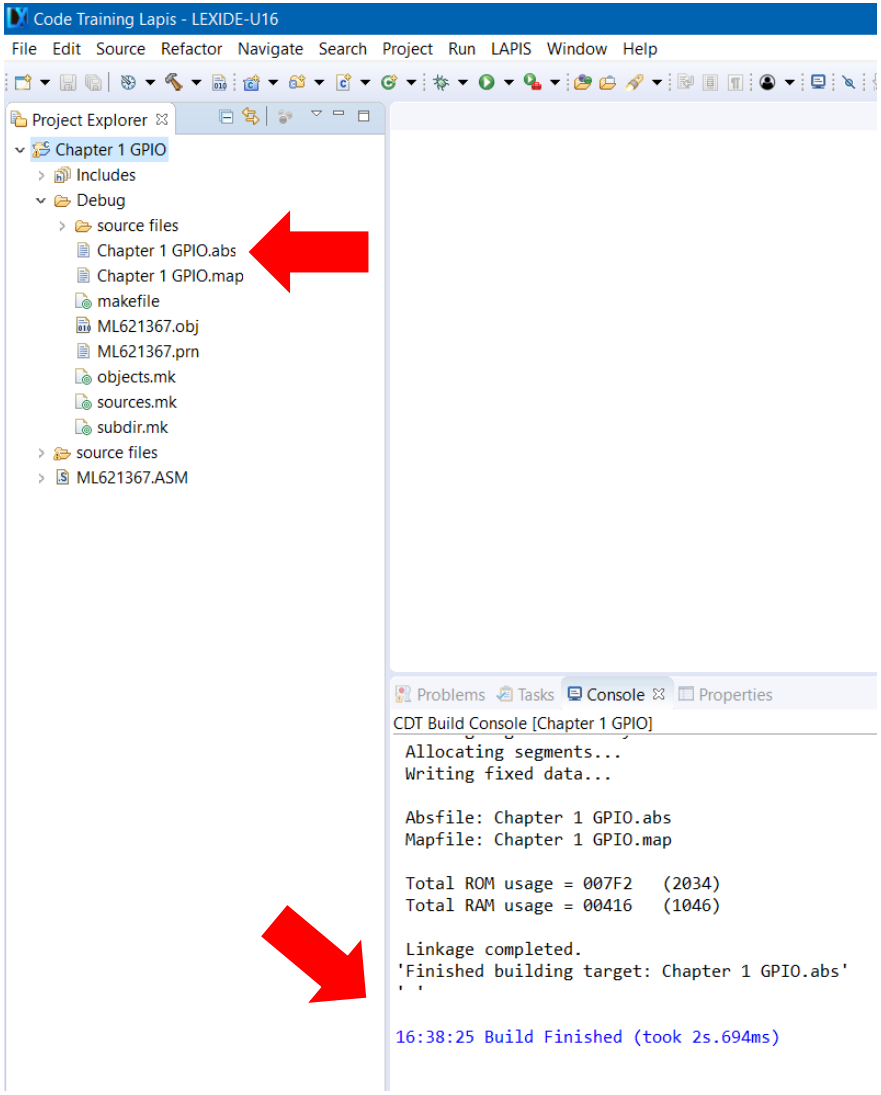
## Build Project




Right-click on a project folder and select [Build Project] to start the build process.

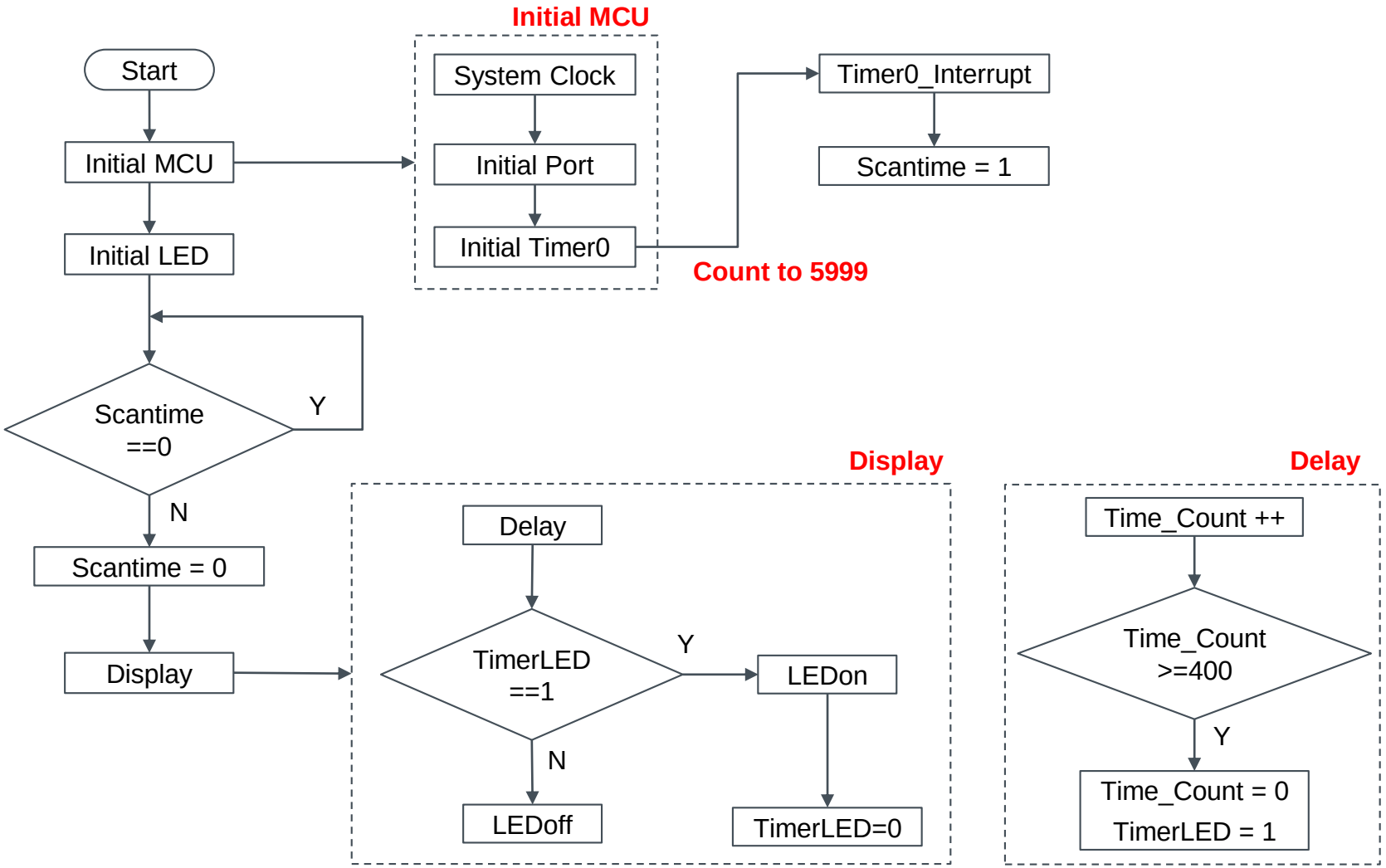
Pichet Sriphomma pichet@es.co.th

## Build Project




When the build succeeds , an ABS file is generated.

Pichet Sriphomma pichet@es.co.th




Pichet Sriphomma pichet@es.co.th

# INTRODUCTION : LAPIS ML62Q1367

**CPU**

**- 16-bit RISC CPU (CPU name: nX-U16/100)**

- **Instruction system: 16-bit length instruction**
- **On-chip debug function built-in (supported by LAPIS on-chip debug emulator EASE1000)**
- **ISP (In-System Programming) function built-in**

**Operating voltage and temperature**

- **Operating voltage: VDD = 1.6 to 5.5 V**
- **Operating temperature: -40 to +105 °C**

**Clock Generation Circuit**

- **Low-speed clock**
- **Internal low-speed RC oscillation: Approx.32.768 kHz**
- **High-speed clock**
- **PLL oscillation: 24MHz/16MHz is selectable by code option**



Pichet Sriphomma pichet@es.co.th

# INTRODUCTION : LAPIS ML62Q1367

Table 1-1 ML62Q1300 Group Product List

| Program memory | Data memory | Data Flash | 16pin SSOP16 WQFN16 | 20pin TSSOP20 | 24pin WQFN24 | 32pin TQFP32 WQFN32 |
|---|---|---|---|---|---|---|
| 64Kbyte | 4Kbyte | 2Kbyte | — | — | ML62Q1347 | ML62Q1367 |
| 48Kbyte | | | — | — | ML62Q1346 | ML62Q1366 |
| 32Kbyte | | | — | — | ML62Q1345 | ML62Q1365 |
| 32Kbyte | 2Kbyte | | ML62Q1325 | ML62Q1335 | — | — |
| 24Kbyte | | | ML62Q1324 | ML62Q1334 | — | — |
| 16Kbyte | | | ML62Q1323 | ML62Q1333 | — | — |



Pichet Sriphomma pichet@es.co.th

# INTRODUCTION : LAPIS ML62Q1367

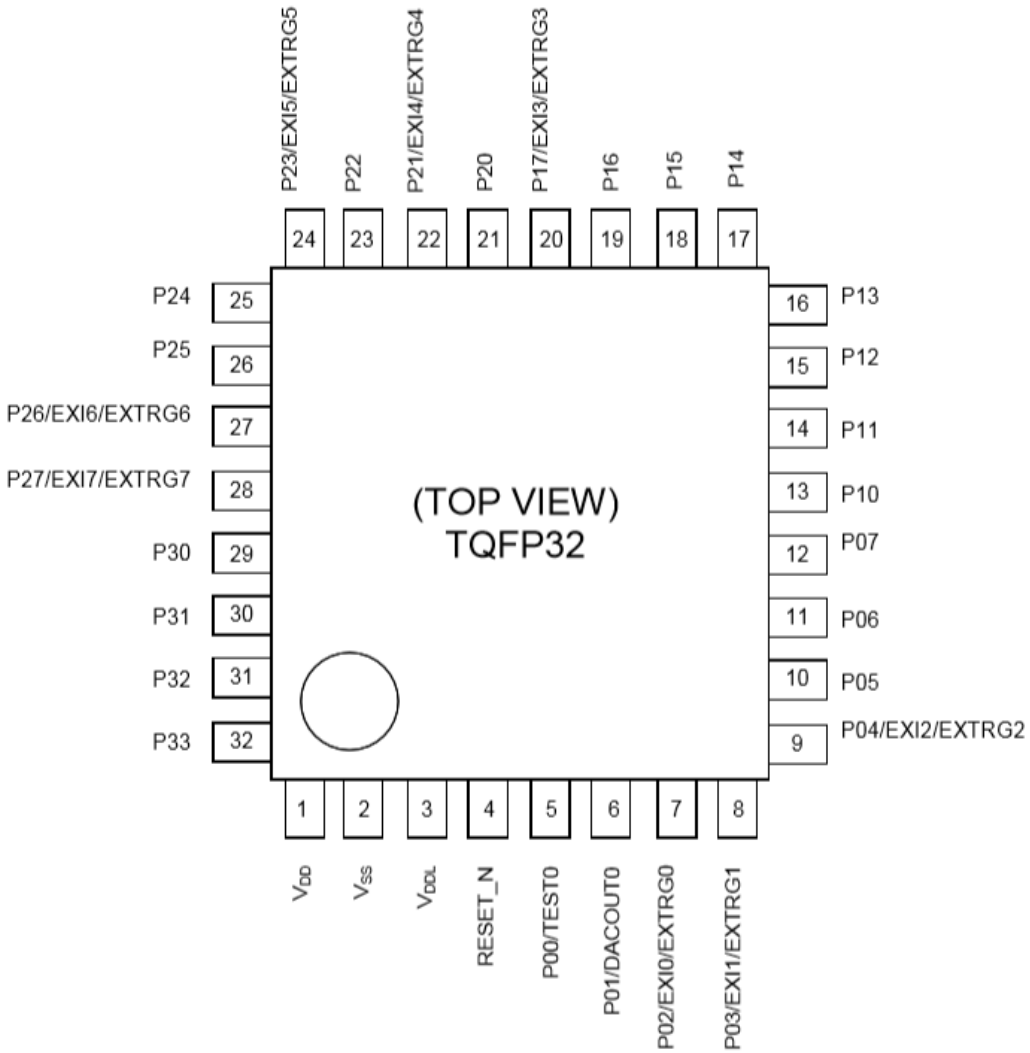
Table 1-4 Main Function List

| Part number | Pin | | | | | Interrupt | | Timer | | Serial | | | Analog | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Total pin-counts | Power pin counts | Reset Input pinInput pin | I/O port | LED drive port (shared with the I/O port) | Internal interrupt [source] | External interrupt [port] | Functional Timer [ch] | 16bit General I Timer*1 [ch] | Full-duplex UART or Synchronous serial*2 [ch] | I²C bus unit (Master/Slave) [ch] | I²C bus interface (Master only) [ch] | 10bit Successive type A/D converter [ch] | Analog comparator [ch] | Analog comparator [input pin] | 8bit D/A converter [ch] |
| ML62Q1323 | 16 | 3 | 1 | 12 | 11 | 23 | 8 | 4 | 4 | 2 | 1 | 1 | 6 | 1 | 2 | 0 |
| ML62Q1324 | | | | | | | | | | | | | | | | |
| ML62Q1325 | | | | | | | | | | | | | | | | |
| ML62Q1333 | 20 | | | 16 | 15 | | | | | | | | 8 | | | |
| ML62Q1334 | | | | | | | | | | | | | | | | |
| ML62Q1335 | | | | | | | | | | | | | | | | |
| ML62Q1345 | 24 | | | 20 | 19 | 25 | | | 6 | | | | | | | 1 |
| ML62Q1346 | | | | | | | | | | | | | | | | |
| ML62Q1347 | | | | | | | | | | | | | | | | |
| ML62Q1365 | 32 | | | 28 | 27 | | | | | | | | | | | |
| ML62Q1366 | | | | | | | | | | | | | | | | |
| ML62Q1367 | | | | | | | | | | | | | | | | |

*1 [illegible]

Pichet Sriphomma pichet@es.co.th

# INTRODUCTION : LAPIS ML62Q1367






Pichet Sriphomma pichet@es.co.th

# 1.1 SYSTEM CLOCK



Pichet Sriphomma pichet@es.co.th

## Clock Generation Circuit

Table 6-1 Clocks generated by the clock generation circuit

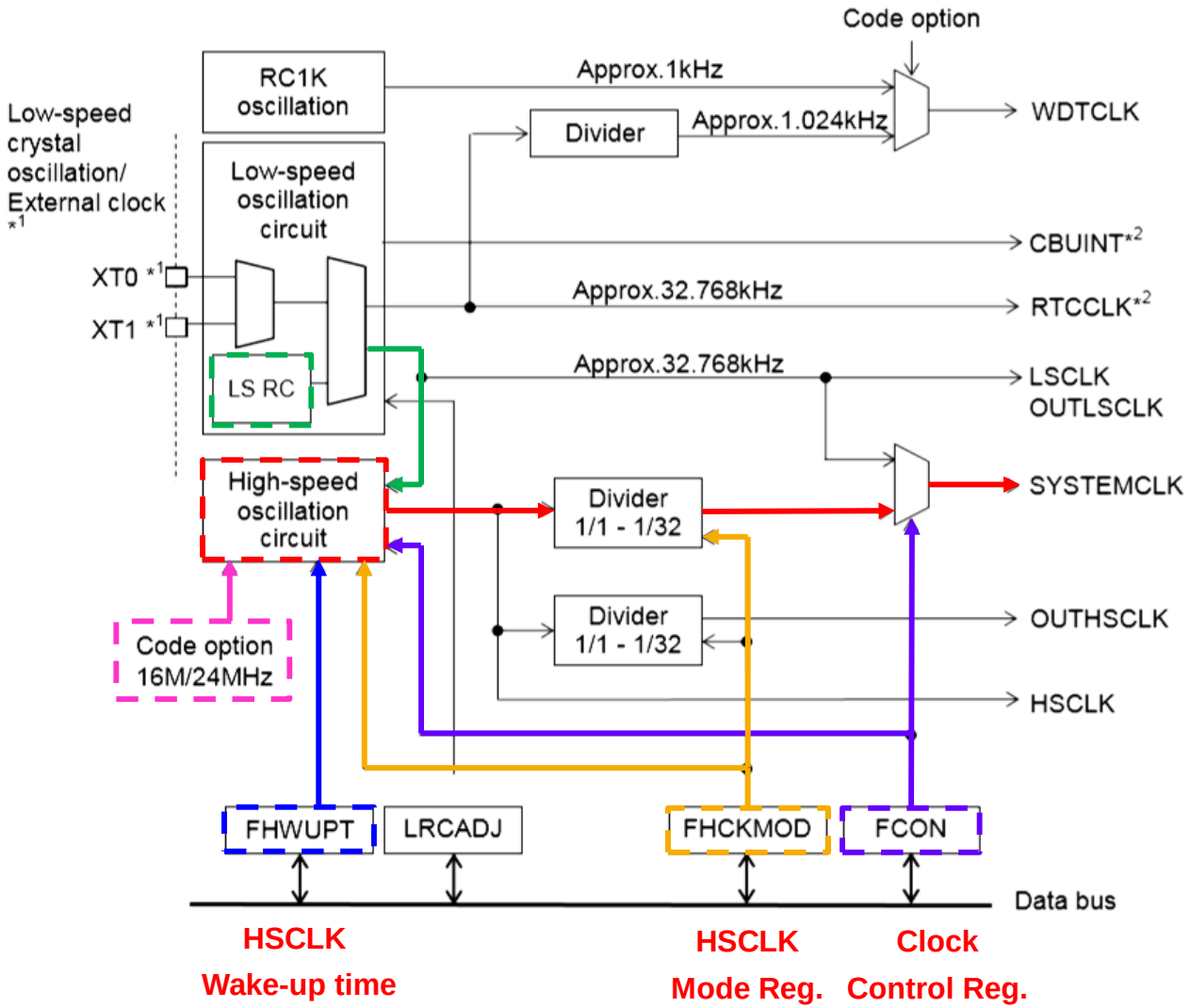
| Clock Name | Symbol | Description |
|---|---|---|
| Low-speed clock | LSCLK | Low speed clock for peripherals (32.768kHz) |
| Simplified RTC clock*1 | RTCCLK | Low speed clock for the simplified RTC (32.768kHz) |
| High-speed clock | HSCLK | High speed clock for peripherals (Max. 24MHz) |
| System clock | SYSTEMCLK | CPU operating clock (32.768kHz or Max. 24MHz)<br>The maximum frequency depends on the CPU operation mode(See Table 6-2) |
| Low-speed output clock | OUTLSCLK | Low speed output from a general port (32.768kHz) |
| High-speed output clock | OUTHSCLK | High speed output from an general port (Max. 12MHz) |
| WDT clock | WDTCLK | Clock for the watch dog timer (1.024kHz) |

*1 Available on the ML62Q1500 and ML62Q1700 group



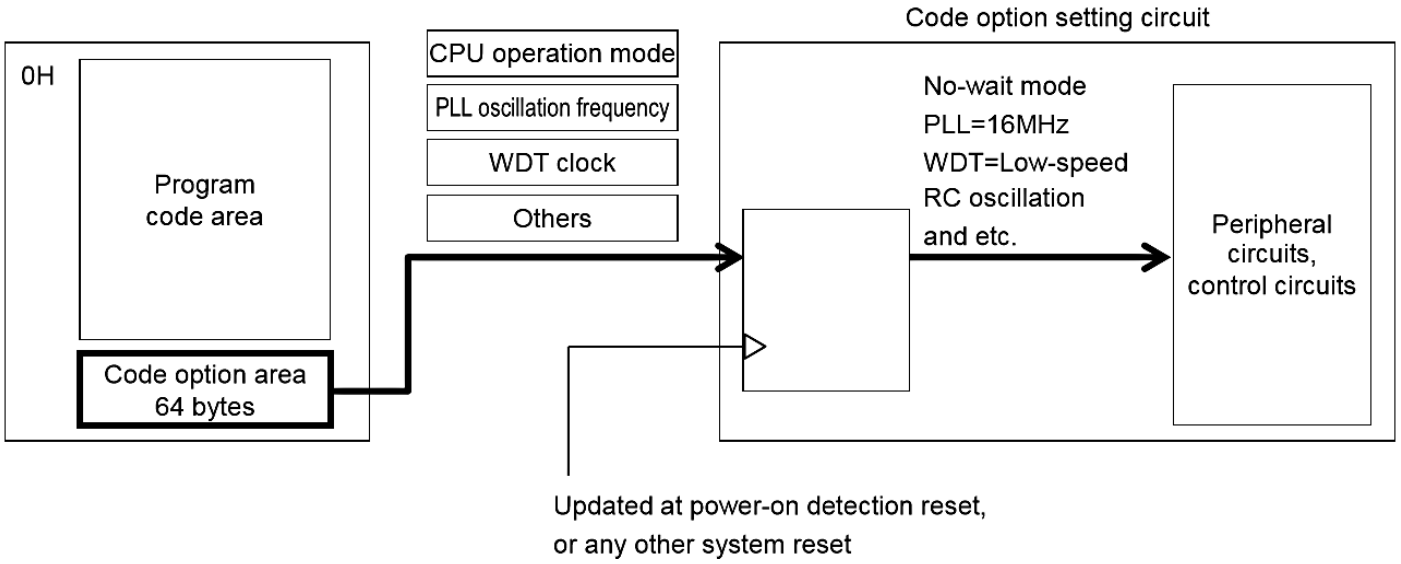
Pichet Sriphomma pichet@es.co.th

## Clock Generation Circuit




Pichet Sriphomma pichet@es.co.th

## Code Options (ML621367.ASM)




ML62Q1300 group

| Product name | Program memory space size | Code Option area | Address | | |
|---|---|---|---|---|---|
| | | | CODEOP2 | CODEOP1 | CODEOP0 |
| ML62Q1323/1333 | 16K byte | 0x0:3FC0 to 0x0:3FFF | 0x0:3FD4 | 0x0:3FD2 | 0x0:3FD0 |
| ML62Q1324/1334 | 24K byte | 0x0:5FC0 to 0x0:5FFF | 0x0:5FD4 | 0x0:5FD2 | 0x0:5FD0 |
| ML62Q1325/1335/1345 /1365 | 32K byte | 0x0:7FC0 to 0x0:7FFF | 0x0:7FD4 | 0x0:7FD2 | 0x0:7FD0 |
| ML62Q1346/1366 | 48K byte | 0x0:BFC0 to 0x0:BFFF | 0x0:BFD4 | 0x0:BFD2 | 0x0:BFD0 |
| ML62Q1347/1367 | 64K byte | 0x0:FFC0 to 0x0:FFFF | 0x0:FFD4 | 0x0:FFD2 | 0x0:FFD0 |



Pichet Sriphomma pichet@es.co.th

# Code Options (ML621367.ASM)

## 26.2.1 Code Options 0 (CODEOP0)

This is the symbol assigned to address in the code option area of the program memory space (different from the special function registers (SFR)).

Address: (See Table 26-1)
Access: R/W
Access size: 16 bits
Initial value: 0xFFFF (factory default setting for products with blank flash memory)

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | CODEOP0 | | | | | | | | | | | | | | | |
| Byte | - | | | | | | | | - | | | | | | | |
| Bit | - | - | - | PCERMD | - | - | - | REMAPMD | - | - | - | - | - | WDTNMCK | WDTSPMD | WDTMD |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

WDT

## Code Options (ML621367.ASM)

| Bit No. | Bit symbol name | Description |
|---|---|---|
| 15 to 13 | - | Reserved bits |
| 12 | PCERMD | This bit is used to choose to enable/disable the unused ROM area access reset.<br>See Chapter 29.3.2 "Unused ROM Area Access Reset Function" for the unused ROM area access reset.<br>0: Unused ROM area access reset disabled<br>1: Unused ROM area access reset enabled (initial value) |
| 11 to 9 | - | Reserved bits |
| 8 | REMAPMD | This bit is used to choose to enable/disable the remapping function (software remap or hardware remap) operation.<br>See Chapter 2.7 "Remapping Function" for details of the remapping function.<br>0: Remapping function operation enabled<br>1: Remapping function operation disabled (initial value) |
| 8 | REMAPMD | This bit is used to choose to enable/disable the remapping function (software remap or hardware remap) operation.<br>See Chapter 2.7 "Remapping Function" for details of the remapping function.<br>0: Remapping function operation enabled<br>1: Remapping function operation disabled (initial value) |
| 7 to 3 | - | Reserved bits |
| 2 | WDTNMCK | This bit is used to choose the watchdog timer (WDT) operation clock.<br>0: Clock with divided frequency (1.024 kHz) of low-speed oscillation clock (32.768 kHz)<br>1: Watchdog timer RC1K oscillation clock (initial value)<br>See Chapter 10 "Watchdog Timer" for details of the watchdog timer. |
| 1 | WDTSPMD | Set this bit to "0". |
| 0 | WDTMD | This bit is used to choose to enable/disable the watchdog timer (WDT) operation.<br>0: WDT operation disabled<br>1: WDT operation enabled (initial value) |



Pichet Sriphomma pichet@es.co.th

# Code Options (ML621367.ASM)

## 26.2.2 Code Options 1 (CODEOP1)

This is the symbol assigned to address in the code option area of the program memory space (different from the special function registers (SFR)).

Address: (See Table 26-1)
Access: R/W
Access size: 16 bits
Initial value: 0xFFFF (factory default setting for products with blank flash memory)

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | CODEOP1 | | | | | | | | | | | | | | | |
| byte | - | | | | | | | | - | | | | | | | |
| Bit | - | - | - | - | - | - | - | - | - | - | - | - | PLLMD1 | PLLMD0 | CPUMD1 | CPUMD0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## Code Options (ML621367.ASM)

| | | |
|---|---|---|
| 3, 2 | PLLMD1, PLLMD0 | These bits are used to choose the PLL reference frequency.<br>00: Do not use<br>01: Do not use<br>10: PLL reference frequency=24 MHz<br>11: PLL reference frequency=16 MHz (initial value)<br>The following table shows the relation between the PLL reference frequency and the maximum operating frequency of CPU and peripheral circuits. |

| PLL reference frequency | Maximum operating frequency | | |
|---|---|---|---|
| | Peripheral circuit | CPU (wait mode) | CPU (no-wait mode) |
| 24MHz | 24MHz | 24MHz | 6MHz |
| 16MHz | 16MHz | 16MHz | 8MHz |

| | | |
|---|---|---|
| | | See Chapter 2 "CPU and Memory Space" and Appendix C "Instruction Execution Cycle" for the CPU operation modes (wait mode and no-wait mode). |
| 1, 0 | CPUMD1, CPUMD0 | These bits are used to choose the CPU operation mode.<br>00: Prohibited to use (wait mode)<br>01: Wait mode<br>10: Prohibited to use (no-wait mode)<br>11: No-wait mode (initial value) |



Pichet Sriphomma pichet@es.co.th

## Code Options (ML621367.ASM)

```
;-------------------------------------------------------------
;       Setting the code-option data (for ML621367)
;-------------------------------------------------------------
        cseg #0 at 0ffc0h       ; address
        dw      0ffffh          ; 0ffc0h
        dw      0ffffh          ; 0ffc2h
        dw      0ffffh          ; 0ffc4h
        dw      0ffffh          ; 0ffc6h
        dw      0ffffh          ; 0ffc8h
        dw      0ffffh          ; 0ffcah
        dw      0ffffh          ; 0ffcch
        dw      0ffffh          ; 0ffceh

        cseg #0 at 0ffd0h       ; address
        dw      0fff9h          ; 0ffd0h    ;Disable WDT in STOP mode

        cseg #0 at 0ffd2h       ; address
        dw      0fff9h          ; 0ffd2h    ;PLL=24 MHz and CPU wait mode
        dw      0ffffh          ; 0ffd4h
        dw      0ffffh          ; 0ffd6h
        dw      0ffffh          ; 0ffd8h
        dw      0ffffh          ; 0ffdah
        dw      0ffffh          ; 0ffdch
        dw      0ffffh          ; 0ffdeh
```

| | CODEOP1 | CODEOP0 |
|---|---|---|
| Address | FFD2 | FFD0 |
| Value | FFF9 | FFF9 |

Back to Clock Generation Circuit

Pichet Sriphomma pichet@es.co.th

# Clock Generation Circuit (main.c)

```
void InitSystemClock(void)
{
    unsigned int i;

    ENOSC = 0;                  ble the high spe

    //--------------------------------------
    Set_VLS_Startup();


    HOSCM0 = 0;
    SYSC2=0;SYSC1=0;SYSC0=0;
    OUTC2=1;OUTC1=0;OUTC0=0;
    FHUT0=0;

    ENOSC = 1;  // Enable the high speed os
    i = 200;    // Wait for OSC. is stable.
    while(--i)
        wdt_clear();

    SELSCLK = 1;    // System Clock = HSCLK
    __asm("nop");
    __asm("nop");
    __asm("nop");
    __asm("nop");
}
```

### 6.2.4 Clock Control Register (FCON)

FCON is a specific function register (SFR) to control the clock generation circuit and choose the system clock.

Address: 0xF006
Access: R/W
Access size: 8 bits
Initial value: 0x00

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | - | | | | | | | | | | | | | | | |
| Byte | - | | | | | | | | FCON | | | | | | | |
| Bit | - | - | - | - | - | - | - | - | LPLL | - | - | - | - | - | ENOSC | SELSCLK |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit No. | Bit symbol name | Description |
|---|---|---|
| 15 to 8 | - | Reserved bit |
| 7 | LPLL | This bit indicates that the frequency of the PLL oscillation is within the target error. The LPLL has the read-only attribute.<br>0: The frequency of PLL oscillation is out of the target error or the PLL oscillation is stopped (Initial value)<br>1: The frequency of PLL oscillation is within the target error |
| 6 to 2 | - | Reserved bit |
| 1 | ENOSC | This bit is used to enable/start or disable/stop the oscillation of the high-speed clock oscillation circuit.<br>0: Disable/Stop the high-speed clock oscillation (Initial value)<br>1: Enable/Star the high-speed clock oscillation (Initial value) |
| 0 | SELSCLK | This bit is used to choose the system clock.<br>When the high-speed generation circuit is stopped (ENOSC bit = "0"), the SELSCLK bit is fixed to "0" and the low-speed clock (LSCLK) is chosen for the system clock.<br>0: LSCLK (Initial value)<br>1: High-speed clock chosen by the SYSC2 to SYSC0 bit |



Pichet Sriphomma pichet@es.co.th

# Clock Generation Circuit (main.c)

```
void InitSystemClock(void)
{
    unsigned int i;

    ENOSC = 0;      // Disable the high

    //---------------------------------
    Set_VLS_Startup();


    HOSCM0 = 0;
    SYSC2=0;SYSC1=0;SYSC0=0;
    OUTC2=1;OUTC1=0;OUTC0=0;
    FHUT0=0;

    ENOSC = 1;  // Enable the high speed os
    i = 200;    // Wait for OSC. is stable.
    while(--i)
        wdt_clear();

    SELSCLK = 1;    // System Clock = HSCLK by the SYSC2 to SYSC0 bit
    __asm("nop");
    __asm("nop");
    __asm("nop");
    __asm("nop");
}
```

### 6.2.2 High-Speed Clock Mode Register (FHCKMOD)

FHCKMOD is a specific function register (SFR) to choose the oscillation mode of the high-speed clock oscillation circuit (PLL oscillation circuit) and the frequency of high-speed clock.

Address: 0xF002(FHCKMODL/FHCKMOD), 0xF003(FHCKMODH)
Access: R/W
Access size: 8/16bit
Initial value: 0x4400

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | FHCKMOD | | | | | | | | | | | | | | | |
| Byte | FHCKMODH | | | | | | | | FHCKMODL | | | | | | | |
| Bit | - | OUTC2 | OUTC1 | OUTC0 | - | SYSC2 | SYSC1 | SYSC0 | - | - | - | - | - | - | - | HOSCM0 |
| R/W | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R | R | R | R | R | R | R | R/W |
| Initial value | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | |
|---|---|---|
| 7 to 1 | - | Reserved bit |
| 0 | HOSCM0 | This bit is used to choose the oscillation mode of the high-speed oscillation circuit (PLL oscillation mode)<br>0: PLL oscillation mode (Initial value)<br>1: Do not use (PLL oscillation mode) |

# Clock Generation Circuit (main.c)

```
void InitSystemClock(void)
{
    unsigned int i;

    ENOSC = 0;      // Disable the high

    //---------------------------------
    Set_VLS_Startup();


    HOSCM0 = 0;
    SYSC2=0;SYSC1=0;SYSC0=0;
    OUTC2=1;OUTC1=0;OUTC0=0;
    FHUT0=0;

    ENOSC = 1;  // Enable the high speed
    i = 200;    // Wait for OSC. is stab
    while(--i)
        wdt_clear();

    SELSCLK = 1;    // System Clock = HS
    __asm("nop");
    __asm("nop");
    __asm("nop");
    __asm("nop");
}
```

| | | |
|---|---|---|
| 10 to 8 | SYSC2 to SYSC0 | These bits are used to choose a division ratio of the frequency of the high-speed of the high-speed clock used for the system clock (SYSTEMCLK). At the system reset, 1/16 HSCLK is chosen.<br>Choose a proper division ratio of the frequency, so that the frequency does not exceed the maximum frequency of the CPU operating frequency shown in the Table 6-2 "CPU operation mode and PLL oscillation reference frequency".<br>000: HSCLK (in Wait mode)<br>1/2 HSCLK *1 (in No wait mode)<br>001: 1/2 HSCLK (in Wait mode)<br>1/2 HSCLK *1 (in No wait mode)<br>010: 1/4 HSCLK<br>011: 1/8 HSCLK<br>100: 1/16 HSCLK (Initial value)<br>101: 1/32 HSCLK<br>110: Do not use (1/32 HSCLK)<br>111: Do not use (1/32 HSCLK)<br>*1 : When the PLL reference frequency is 24MHz, do not use the 1/2 HSCLK. |
| 14 to 12 | OUTC2 to OUTC0 | These bits are used to choose a division ratio of the frequency of the high-speed output clock (OUTHSCLK) output from the general port. At the system reset, 1/16 HSCLK is chosen.<br>000: Do not use(HSCLK)<br>001: 1/2 HSCLK<br>010: 1/4 HSCLK<br>011: 1/8 HSCLK<br>100: 1/16 HSCLK (Initial value)<br>101: 1/32 HSCLK<br>110: Do not use (1/32 HSCLK)<br>111: Do not use (1/32 HSCLK) |

# Clock Generation Circuit (main.c)

```
void InitSystemClock(void)
{
    unsigned int i;

    ENOSC = 0;      // Disable the high s

    //-----------------------------------
    Set_VLS_Startup();


    HOSCM0 = 0;
    SYSC2=0;SYSC1=0;SYSC0=0;
    OUTC2=1;OUTC1=0;OUTC0=0;
    FHUT0=0;

    ENOSC = 1;  // Enable the high speed
    i = 200;    // Wait for OSC. is stabl..
    while(--i)
        wdt_clear();

    SELSCLK = 1;    // System Clock = HSCLK b
    __asm("nop");
    __asm("nop");
    __asm("nop");
    __asm("nop");
}
```

### 6.2.5 High-Speed Clock Wake-up Time Setting Register (FHWUPT)

FHWUPT is a specific function register (SFR) to choose the wake-up time of the high-speed clock.
FHWUPT is writable only when the high-speed oscillation is stopped.
The bit symbol "rsvd" indicates the reserved bit, always write "0" to them.
See Table 4-5 "Wake-up Time from Standby Mode" in the Chapter 4 for details about the wake-up time from the standby modes.

Address: 0xF008 (FHWUPT)
Access: R/W
Access size: 8 bits
Initial value: 0x00

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | - | | | | | | | | | | | | | | | |
| Byte | - | | | | | | | | FHWUPT | | | | | | | |
| Bit | - | - | - | - | - | - | - | - | - | - | - | - | - | rsvd | rsvd | FHUT 0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

0 FHUT0

This bit is used to choose the wake-up time of the high-speed clock.
Two options are chosen. One is to supply the frequency-stable clock after enabling the high-speed oscillation. The other one is to supply clock before the frequency gets stable.
In the case this bit is set to "1", the clock starts to be supplied approx. 30us after enabling the high-speed oscillation. Then, the frequency of the clock is gradually getting higher and reaches to the target frequency in approx. 2ms.
The frequency in the approx.2ms is not guaranteed as the specification, however it is useable for the system clock.

0: The clock starts to be supplied after it's stabilized: approx. 2.5 ms (initial value)
1: The clock starts to be supplied before it's stabilized: approx. 30 µs



Pichet Sriphomma pichet@es.co.th

# Clock Generation Circuit (main.c)

```
void InitSystemClock(void)
{
    unsigned int i;

    ENOSC = 0;      // Disable the high spe

    //-------------------------------------
    Set_VLS_Startup();


    HOSCM0 = 0;
    SYSC2=0;SYSC1=0;SYSC0=0;
    OUTC2=1;OUTC1=0;OUTC0=0;
    FHUT0=0;

    ENOSC = 1;  //      he high speed os
    i = 200;    // Wait for OSC. is stable.
    while(--i)
        wdt_clear();

    SELSCLK = 1;    // System Clock = HSCLK
    __asm("nop");
    __asm("nop");
    __asm("nop");
    __asm("nop");
}
```

### 6.2.4 Clock Control Register (FCON)

FCON is a specific function register (SFR) to control the clock generation circuit and choose the system clock.

Address: 0xF006
Access: R/W
Access size: 8 bits
Initial value: 0x00

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | - | | | | | | | | | | | | | | | |
| Byte | - | | | | | | | | FCON | | | | | | | |
| Bit | - | - | - | - | - | - | - | - | LPLL | - | - | - | - | - | ENOSC | SELSCLK |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit No. | Bit symbol name | Description |
|---|---|---|
| 15 to 8 | - | Reserved bit |
| 7 | LPLL | This bit indicates that the frequency of the PLL oscillation is within the target error. The LPLL has the read-only attribute.<br>0: The frequency of PLL oscillation is out of the target error or the PLL oscillation is stopped (Initial value)<br>1: The frequency of PLL oscillation is within the target error |
| 6 to 2 | - | Reserved bit |
| 1 | ENOSC | This bit is used to enable/start or disable/stop the oscillation of the high-speed clock oscillation circuit.<br>0: Disable/Stop the high-speed clock oscillation (Initial value)<br>1: Enable/Star the high-speed clock oscillation (Initial value) |
| 0 | SELSCLK | This bit is used to choose the system clock.<br>When the high-speed generation circuit is stopped (ENOSC bit = "0"), the SELSCLK bit is fixed to "0" and the low-speed clock (LSCLK) is chosen for the system clock.<br>0: LSCLK (Initial value)<br>1: High-speed clock chosen by the SYSC2 to SYSC0 bit |



Pichet Sriphomma pichet@es.co.th

# Clock Generation Circuit (main.c)

```
void InitSystemClock(void)
{
    unsigned int i;

    ENOSC = 0;       // Disable the high

    //-----------------------------------
    Set_VLS_Startup();


    HOSCM0 = 0;
    SYSC2=0;SYSC1=0;SYSC0=0;
    OUTC2=1;OUTC1=0;OUTC0=0;
    FHUT0=0;

    ENOSC = 1;  // Enable the high speed
    i = 200;    // Wait for OSC. is stabl
    while(--i)
        wdt_clear();

    SELSCLK = 1;          stem Clock = HSCLK by the SYSC2 to SYSC0 bit
    __asm("nop");
    __asm("nop");
    __asm("nop");
    __asm("nop");
}
```

### 6.2.4 Clock Control Register (FCON)

FCON is a specific function register (SFR) to control the clock generation circuit and choose the system clock.

Address: 0xF006
Access: R/W
Access size: 8 bits
Initial value: 0x00

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | - | | | | | | | | | | | | | | | |
| Byte | - | | | | | | | | FCON | | | | | | | |
| Bit | - | - | - | - | - | - | - | - | LPLL | - | - | - | - | - | ENOSC | SELSCLK |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

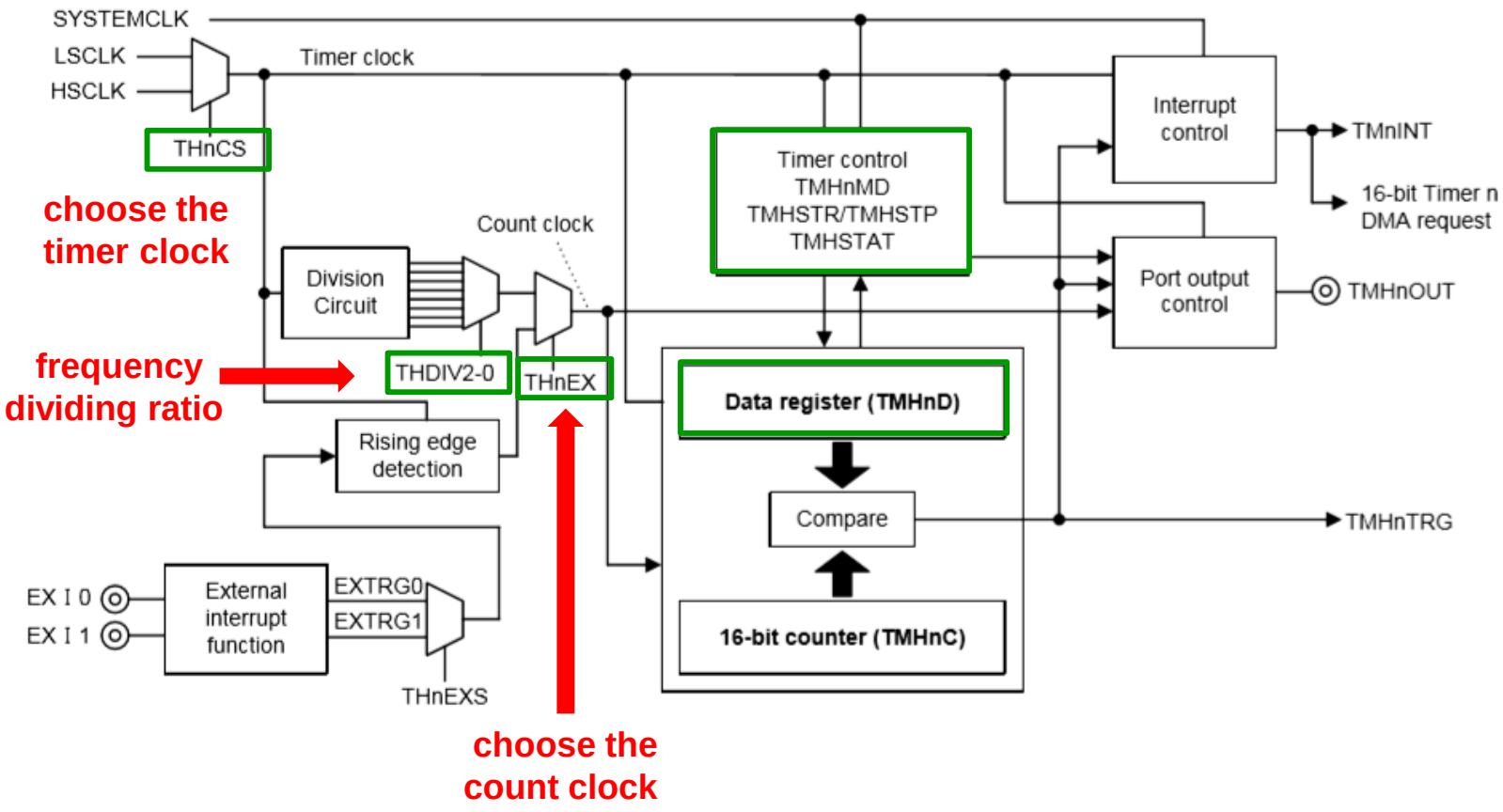
| | | |
|---|---|---|
| 0 | SELSCLK | This bit is used to choose the system clock.<br>When the high-speed generation circuit is stopped (ENOSC bit = "0"), the SELSCLK bit is fixed to "0" and the low-speed clock (LSCLK) is chosen for the system clock.<br>0: LSCLK (Initial value)<br>1: High-speed clock chosen by the SYSC2 to SYSC0 bit |

# 1.2 Timer 16-Bit



Pichet Sriphomma pichet@es.co.th

## Timer 16-Bit



## Timer 16-Bit

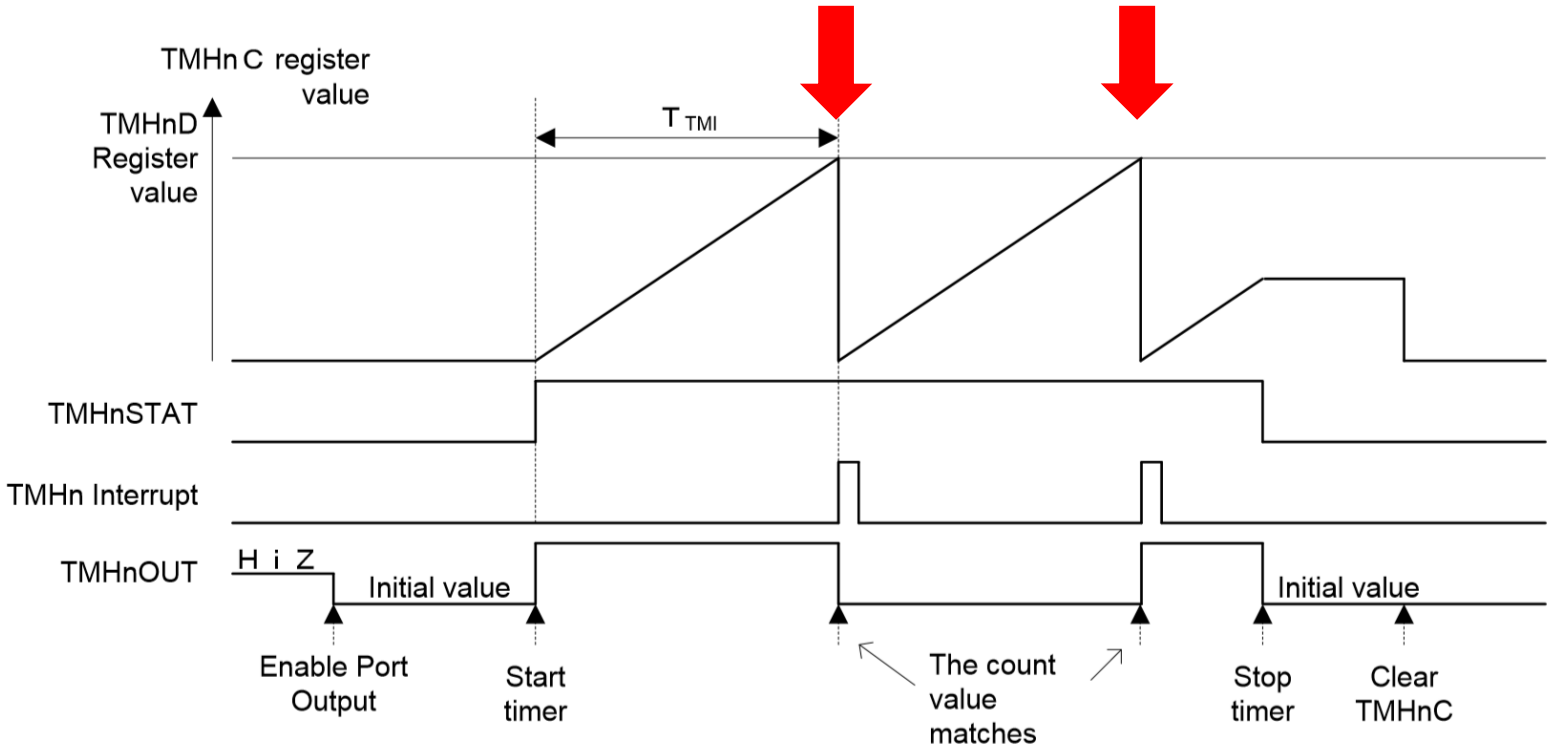



Figure 8-3 Repeat Mode Operation Timing in 16-Bit Timer Mode



Pichet Sriphomma pichet@es.co.th

## Timer 16-Bit

TMHn Interrupt Frequency = 4 kHz = 0.25 mS

$$\text{TMHn Interrupt Frequency} = \frac{\text{HSCLK}}{\text{TMH0D+1}}$$

4 kHz = 24 MHz/TMH0D + 1

TMH0D = 5999



Pichet Sriphomma pichet@es.co.th

## Initial Timer 16-Bit (timer0.c)

```
TH0CS=1;
TH0EX=0;

TH0EXS=0;
TH0DIV2=0;TH0DIV1=0;TH0DIV0=0;

TH08BM = 0;
TH0OST = 0;
TH0NEG = 0;

TMH0D = 5999;

ETM0 = 1;        // Timer 0 interrupt en
TH0RUN = 1;      // Start Counting
```

### 8.2.4 16-Bit Timer n Mode Register (TMHnMOD: n = 0 to 7)

TMHnMOD (n = 0 to 7) is a specific function register (SFR) to control the operation mode of 16-bit timer.

Address: 0xF320(TMH0MODL/TMH0MOD), 0xF321(TMH0MODH),
0xF322(TMH1MODL/TMH1MOD), 0xF323(TMH1MODH),
0xF324(TMH2MODL/TMH2MOD), 0xF325(TMH2MODH),
0xF326(TMH3MODL/TMH3MOD), 0xF327(TMH3MODH),
0xF328(TMH4MODL/TMH4MOD), 0xF329(TMH4MODH),
0xF32A(TMH5MODL/TMH5MOD), 0xF32B(TMH5MODH),
0xF32C(TMH6MODL/TMH6MOD), 0xF32D(TMH6MODH),
0xF32E(TMH7MODL/TMH7MOD), 0xF32F(TMH7MODH)

Access: R/W
Access size: 8/16 bit
Initial value: 0x0000

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | TMHnMOD | | | | | | | | | | | | | | | |
| Byte | TMHnMODH | | | | | | | | TMHnMODL | | | | | | | |
| Bit | - | - | - | - | - | THn NEG | THn OST | THn 8BM | - | THn DIV2 | THn DIV1 | THn DIV0 | THn EXS | THn EX | - | THn CS |
| R/W | R | R | R | R | R | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

clock chosen by the THnCS bit.

| | | |
|---|---|---|
| 1 | - | Reserved bit |
| 0 | THnCS | This bit is used to choose the timer clock of the 16-bit timer n.<br>0: LSCLK (initial value)<br>1: HSCLK |
| 2 | THnEX | This bit is used to choose the count clock (THnCK) of the 16-bit timer n.<br>0: The timer is counted by the clock chosen by the THnCS bit and divided by the ratio chosen by the THnDIV2 to 0 bit. (initial value)<br>1: The timer is counted by the rising edge of the external trigger signal detected by the |



Pichet Sriphomma pichet@es.co.th

## Timer 16-Bit (timer0.c)

```
TH0CS=1;
TH0EX=0;

TH0EXS=0;
TH0DIV2=0;TH0DIV1=0;TH0DIV0=0;

TH08BM = 0;
TH0OST = 0;
TH0NEG = 0;

TMH0D = 5999;

ETM0 = 1;        // Timer 0 inte
TH0RUN = 1;      // Start Counti
```

| | | |
|---|---|---|
| 6 to 4 | THnDIV2 to THnDIV0 | These bits are used to choose frequency dividing ratio for the count clock in the 16-bit timer n .<br>000: No dividing (initial value)<br>001: 1/2 of the timer clock<br>010: 1/4 of the timer clock<br>011: 1/8 of the timer clock<br>100: 1/16 of the timer clock<br>101: 1/32 of the timer clock<br>110: 1/64 of the timer clock<br>111: 1/128 of the timer clock |
| 3 | THnEXS | This bit is used to choose the external trigger supplied as the count clock of the 16-bit timer n.<br>0: EXTRG0 (initial value)<br>1: EXTRG1 |
| 10 | THnNEG | This bit is used to choose the output polarity of timer out (TMHnOUT).<br>0: Positive logic (initial level is "L") (initial value)<br>1: Negative logic (initial level is "H") |
| 9 | THnOST | This bit is used to choose the operation mode of the 16-bit timer n.<br>0: Repeat timer mode (initial value)<br>1: One-shot timer mode |
| 8 | THn8BM | This bit is used to choose whether the timer works as one 16-bit timer or two channels of 8-bit timer.<br>0: 16-bit timer mode (initial value)<br>1: 8-bit timer mode |



Pichet Sriphomma pichet@es.co.th

# Timer 16-Bit (timer0.c)

```
TH0CS=1;
TH0EX=0;

TH0EXS=0;
TH0DIV2=0;TH0DIV1=0;TH0DIV0=0;

TH08BM = 0;
TH0OST = 0;
TH0NEG = 0;

TMH0D = 5999;

ETM0 = 1;        // Timer 0 int
TH0RUN = 1;      // Start Count
```

## 8.2.2 16-Bit Timer n Data Register (TMHnD: n = 0 to 7)

TMHnD (n = 0 to 7) is a specific function register (SFR) to set the comparison value with the 16-bit timer n counter register (TMHnC).
In the 8-bit timer mode, TMHnDL (n = 0 to 5) is compared to TMHnCL (n = 0 to 5) and TMHnDH (n = 0 to 5) is compared to TMHnCH (n = 0 to 5).

Address: 0xF300(TMH0DL/TMH0D), 0xF301(TMH0DH), 0xF302(TMH1DL/TMH1D), 0xF303(TMH1DH)
0xF304(TMH2DL/TMH2D), 0xF305(TMH2DH), 0xF306(TMH3DL/TMH3D), 0xF307(TMH3DH)
0xF308(TMH4DL/TMH4D), 0xF309(TMH4DH), 0xF30A(TMH5DL/TMH5D), 0xF30B(TMH5DH)
0xF30C(TMH6DL/TMH6D), 0xF30D(TMH6DH), 0xF30E(TMH7DL/TMH7D), 0xF30F(TMH7DH)
Access: R/W
Access size: 8/16 bit
Initial value: 0xFFFF

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | TMHnD | | | | | | | | | | | | | | | |
| Byte | TMHnDH | | | | | | | | TMHnDL | | | | | | | |
| Bit | THnD 15 | THnD 14 | THnD 13 | THnD 12 | THnD 11 | THnD 10 | THnD 9 | THnD 8 | THnD 7 | THnD 6 | THnD 5 | THnD 4 | THnD 3 | THnD 2 | THnD 1 | THnD 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

# Timer 16-Bit (timer0.c)

```
TH0CS=1;
TH0EX=0;

TH0EXS=0;
TH0DIV2=0;TH0DIV1=0;TH0DIV0=0;

TH08BM = 0;
TH0OST = 0;
TH0NEG = 0;

TMH0D = 5999;

ETM0 = 1;          Timer 0 inte
TH0RUN = 1;        // Start Counti
```

### 5.2.3 Interrupt Enable Register 23 (IE23)

IE23 is a specific function register (SFR) to enable or disable the interrupt for each interrupt request.
The bits are unwriteable when the products do not have the peripheral circuits and they return "0" for reading.
After the interrupt is accepted, the master interrupt enable flag (MIE) of the CPU is reset to "0", however, the applicable each flag of IE01 is not reset and remains "1".

Address: 0xF022 (IE2/IE23), 0xF023(IE3)
Access: R/W
Access size: 8/16bit
Initial value: 0x0000

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | IE23 | | | | | | | | | | | | | | | |
| Byte | IE3 | | | | | | | | IE2 | | | | | | | |
| Bit | ETM1 | ETM0 | EFTM 1 | EFTM 0 | EI2C M1 | EI2C M0 | - | EEXT X | - | ESAD | - | ESIU 01 | ESIU 00 | EMC S | EDM A | ECBU |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R | R/W | R | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | |
|---|---|---|
| 14 | ETM0 | This bit controls to enable or disable the 16bit Timer 0 interrupt (TM0INT).<br>0: Disable the interrupt (initial value)<br>1: Enable the interrupt |

# Timer 16-Bit (timer0.c)

```
TH0CS=1;
TH0EX=0;

TH0EXS=0;
TH0DIV2=0;TH0DIV1=0;TH0DIV0=0;

TH08BM = 0;
TH0OST = 0;
TH0NEG = 0;

TMH0D = 5999;

ETM0 = 1;          // Timer 0 int
TH0RUN = 1;        // tart Countin
```

## 8.2.7 16-Bit Timer Start Register (TMHSTR)

TMHSTR is a specific function register (SFR) to control to start counting the 16-bit timer n.
TMHSTRL is used in the 16-bit timer mode.
TMHSTRH is used to start counting the upper side 8bit counter in the 8-bit timer mode.
TMHSTRL is used to start counting the lower side 8bit counter in the 8-bit timer mode.
TMHSTR is a write-only register and returns always "0x0000"for reading.

Address: 0xF350
Access: W
Access size: 8/16 bit
Initial value: 0x0000

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | TMHSTR | | | | | | | | | | | | | | | |
| Byte | TMHSTRH | | | | | | | | TMHSTRL | | | | | | | |
| Bit | TH7H RUN | TH6H RUN | TH5H RUN | TH4H RUN | TH3H RUN | TH2H RUN | TH1H RUN | TH0H RUN | TH7 RUN | TH6 RUN | TH5 RUN | TH4 RUN | TH3 RUN | TH2 RUN | TH1 RUN | TH0 RUN |
| R/W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | |
|---|---|---|
| 0 | TH0RUN | In the 16bit timer mode, controls the counter of 16bit timer 0<br>In the 8bit timer mode, controls the lower side 8bit counter of 16bit timer 2<br>Writing "0": Invalid<br>Writing "1": Start counting |



Pichet Sriphomma pichet@es.co.th

## Timer Interrupt (timer0.c)

```
void Timer0_Interrupt(void)
{

    static unsigned char tick_cnt=0;
    static unsigned int t_base_cnt=0;
    //-------------------------------------
    //4000 kHz -> 0.25 mS
    //count to 16 -> (4000 Hz/16) = 250 Hz
    //250 Hz -> 4 mS
    if(++tick_cnt >= 16)    // Count for 250Hz (4000
    {
        tick_cnt = 0;
        Flag._ScanTime=1;
    }

    //------ only test ----------------
    //
    t_base_cnt++;
    if(t_base_cnt >= (4000*10))
    //if(t_base_cnt >= 4000)
        t_base_cnt = 0;

    BlinkLED();
}
```

Timer 16-Bit -> 4 kHz

-> 0.25 mS

Count to 16 -> $\frac{4000Hz}{16} = 250Hz$

Timer Interrupt -> 250 Hz

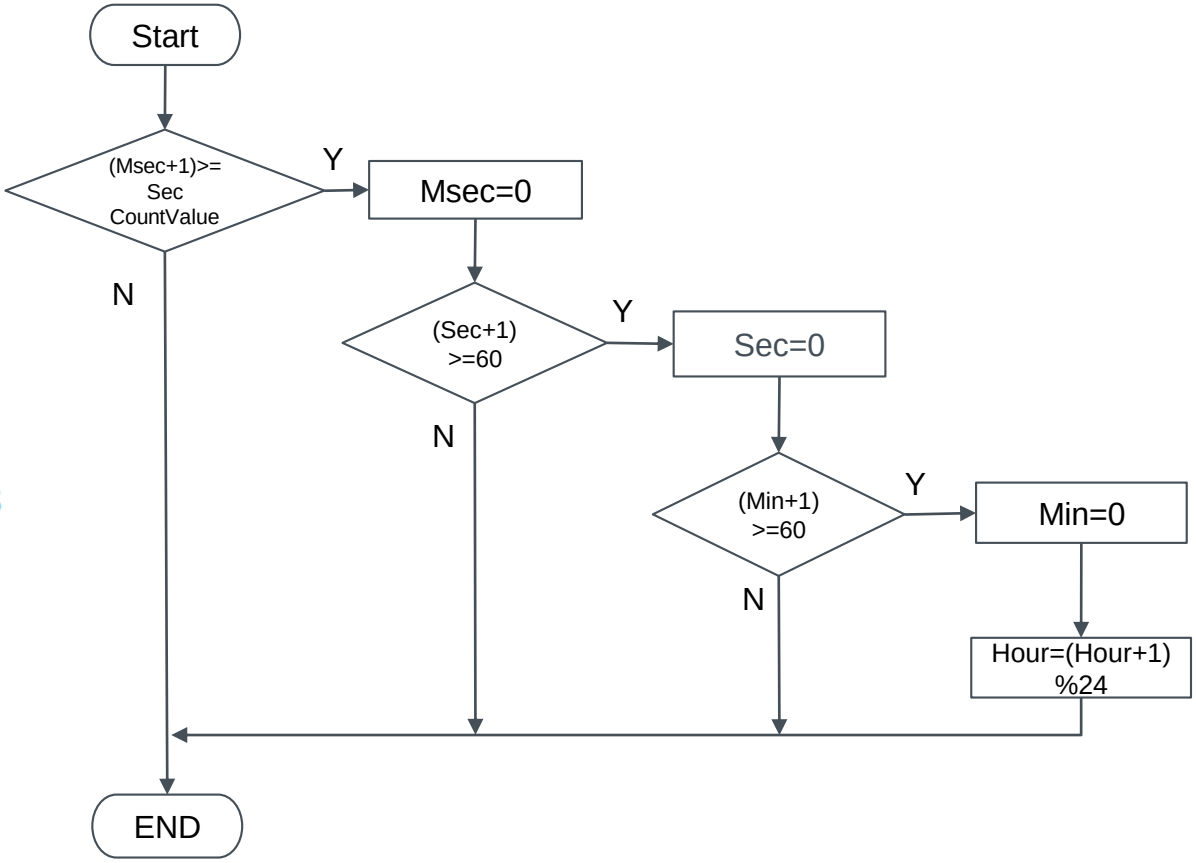
-> 4 mS



Pichet Sriphomma pichet@es.co.th

## Real Time Clock (timer0.c)

```
void RealTimeClock(void)
{
    if(++Msec >= SecCountValue)
    {
        Msec=0;
        if(++Sec >= 60)
        {
            Sec=0;
            if(++Min >= 60)
            {
                Min=0;
                Hour = ++Hour%24;
            }
        }
    }
}
```





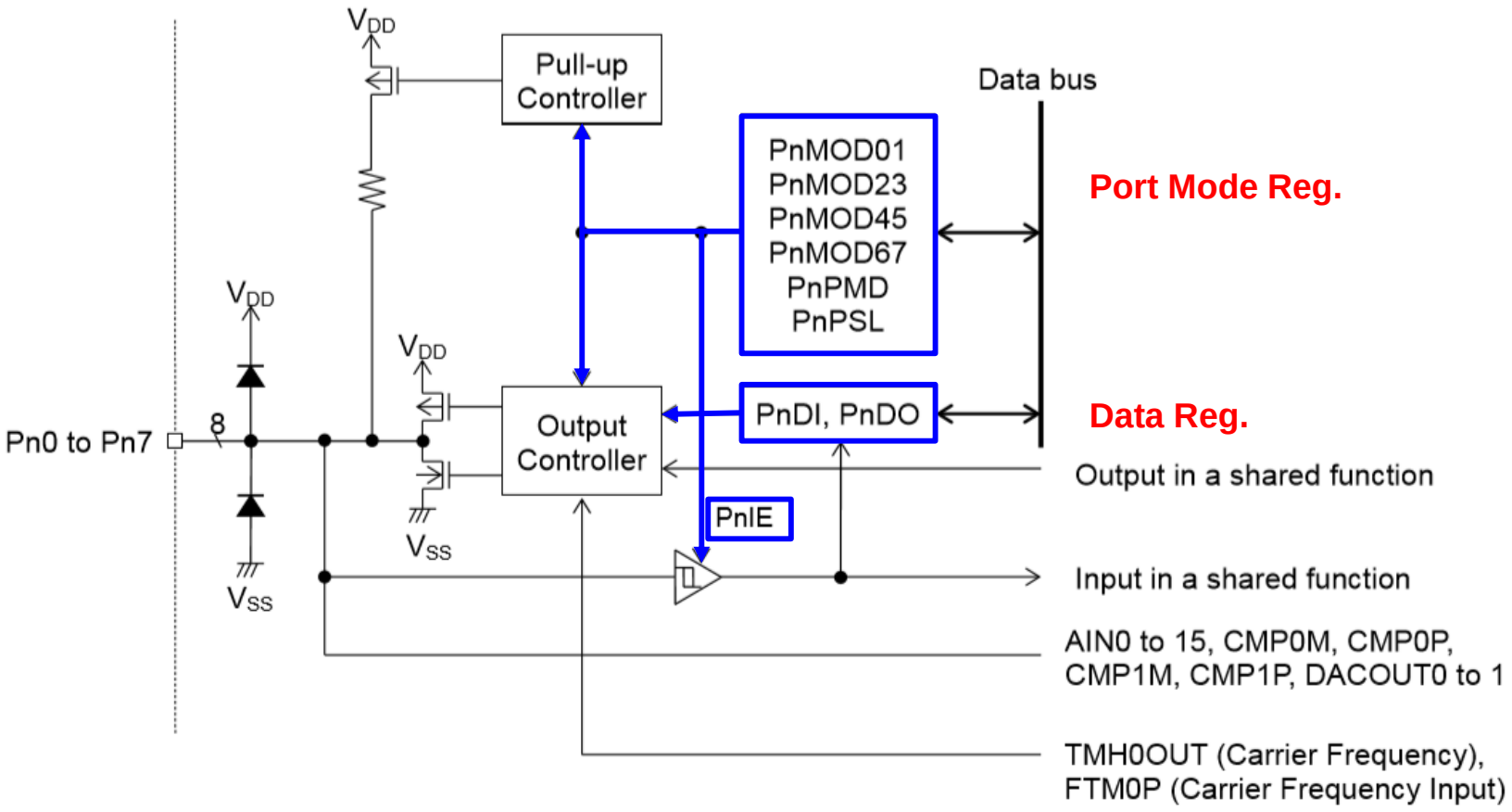

Pichet Sriphomma pichet@es.co.th

# 1.2 GPIO Port



Pichet Sriphomma pichet@es.co.th

## Initial Port (main.c)



## Initial Port (main.c)

```
void Init_Port(void)
{
    //---------------------------------
    //-------- Port 0 Init ------------
    //---------------------------------
    //P00DO=0;  // Output = 0
    P01DO=0;    // Output = 0
    P02DO=0;    // Output = 0
    P03DO=0;    // Output = 0
    P04DO=0;    // Output = 0
    P05DO=0;    // Output = 0
    P06DO=0;    // Output = 0
    P07DO=0;    // Output = 0

    //P00IE=0;P00OE=0;P00OD=0;P00PU=0;
    P01IE=0;P01OE=0;P01OD=0;P01PU=0;
    P02IE=0;P02OE=0;P02OD=0;P02PU=0;    //
    P03IE=0;P03OE=0;P03OD=0;P03PU=0;    //
    P04IE=0;P04OE=0;P04OD=0;P04PU=0;    //
    P05IE=0;P05OE=0;P05OD=0;P05PU=0;    //
    P06IE=0;P06OE=0;P06OD=0;P06PU=0;    //
    P07IE=0;P07OE=0;P07OD=0;P07PU=0;    //

    //P00MD3=0;P00MD2=0;P00MD1=0;P00MD0=0;
    P01MD3=0;P01MD2=0;P01MD1=0;P01MD0=0;
    P02MD3=0;P02MD2=0;P02MD1=0;P02MD0=0;
    P03MD3=0;P03MD2=0;P03MD1=0;P03MD0=0;
    P04MD3=0;P04MD2=0;P04MD1=0;P04MD0=0;
    P05MD3=0;P05MD2=0;P05MD1=0;P05MD0=0;
    P06MD3=0;P06MD2=0;P06MD1=0;P06MD0=0;
    P07MD3=0;P07MD2=0;P07MD1=0;P07MD0=0;
    //-------------------------------------
```

### 17.2.2 Port n Data Register (PnD:n=0 to 9, A, B)

PnD is a special function register (SFR) used to read the level of the port n pin and write output data.
The input level of the port n pin can be read by reading PnDI in the input mode.
Data written to PnDO in the output mode are output to the port n pin.
The PnDO is readable.
Enable or disable the input or output by using the port n mode register.
See Table 17-2 "List of Registers / Bits" to check avaible pins and bits.
Wirte "0" to the bits of PnDO register that have no corresponding pin.
The bits of PnDI register that has no corresponding pin always return "0" for reading.

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | PnD | | | | | | | | | | | | | | | |
| Byte | PnDO | | | | | | | | PnDI | | | | | | | |
| Bit | Pn7DO | Pn6DO | Pn5DO | Pn4DO | Pn3DO | Pn2DO | Pn1DO | Pn0DO | Pn7DI | Pn6DI | Pn5DI | Pn4DI | Pn3DI | Pn2DI | Pn1DI | Pn0DI |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit No. | Bit symbol name | Description |
|---|---|---|
| 15 to 8 | Pn7DO to Pn0DO | This bit is used to set the output level of port n pin.<br>0: Output "L" (initial value)<br>1: Output "H" |
| 7 to 0 | Pn7DI to Pn0DI | This bit is used to set the input level of port n pin.<br>0: The input level is "L"<br>1: The input level is "H" (Initial value) |

# Initial Port (main.c)

```
void Init_Port(void)
{
    //------------------------------
    //-------- Port 0 Init ----------
    //------------------------------
    //P00DO=0;  // Output = 0
    P01DO=0;    // Output = 0
    P02DO=0;    // Output = 0
    P03DO=0;    // Output = 0
    P04DO=0;    // Output = 0
    P05DO=      // Output = 0
    P06DO=      // Output = 0
    P07DO=      // Output = 0

    //P00IE=0;P00OE=0;P00OD=0;P00PU=0;  //
    P01IE=0;P01OE=0;P01OD=0;P01PU=0;    //
    P02IE=0;P02OE=0;P02OD=0;P02PU=0;    //
    P03IE=0;P03OE=0;P03OD=0;P03PU=0;    //
    P04IE=0;P04OE=0;P04OD=0;P04PU=0;    //
    P05IE=0;P05OE=0;P05OD=0;P05PU=0;    //
    P06IE=0;P06OE=0;P06OD=0;P06PU=0;    //
    P07IE=0;P07OE=0;P07OD=0;P07PU=0;    //

    //P00MD3=0;P00MD2=0;P00MD1=0;P00MD0=0;
    P01MD3=0;P01MD2=0;P01MD1=0;P01MD0=0;
    P02MD3=0;P02MD2=0;P02MD1=0;P02MD0=0;
    P03MD3=0;P03MD2=0;P03MD1=0;P03MD0=0;
    P04MD3=0;P04MD2=0;P04MD1=0;P04MD0=0;
    P05MD3=0;P05MD2=0;P05MD1=0;P05MD0=0;
    P06MD3=0;P06MD2=0;P06MD1=0;P06MD0=0;
    P07MD3=0;P07MD2=0;P07MD1=0;P07MD0=0;
    //------------------------------------
```

## 17.2.3 Port n Mode Register 01 (PnMOD01:n=0 to 9, A, B)

PnMOD01 is a special function register (SFR) to choose the input/output mode, input/output status, and shared function of Pn0 pin and Pn1 pin.
See Table 17-2 "List of Registers / Bits" to check avaible pins and bits.
Wirte "0" to the bits of PnMOD01 register that have no corresponding pin.

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | PnMOD01 | | | | | | | | | | | | | | | |
| Byte | PnMOD1 | | | | | | | | PnMOD0 | | | | | | | |
| Bit | Pn1MD3 | Pn1MD2 | Pn1MD1 | Pn1MD0 | Pn1OD | Pn1PU | Pn1OE | Pn1IE | Pn0MD3 | Pn0MD2 | Pn0MD1 | Pn0MD0 | Pn0OD | Pn0PU | Pn0OE | Pn0IE |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | * | 0 | * |

* :The initial value of P00IE and P00PU for the Port0 is "1" and other bits are "0".

| | | |
|---|---|---|
| 9 | Pn1OE | This bit is used to enable the output of Pn1 pin<br>0: Disable the output (initial value)<br>1: Enable the output |
| 8 | Pn1IE | This bit is used to enable the input of Pn1 pin<br>0: Disable the input (initial value)<br>1: Enable the input |

# Initial Port (main.c)

```
void Init_Port(void)
{
    //------------------------------
    //-------- Port 0 Init ----------
    //------------------------------
    //P00DO=0;  // Output = 0
    P01DO=0;    // Output = 0
    P02DO=0;    // Output = 0
    P03DO=0;    // Output = 0
    P04DO=0;    // Output = 0
    P05DO=0;    // Output
    P06DO=0;    // Output
    P07DO=0;    // Output

    //P00IE=0;P00OE=0;P00OD=0;P00PU=0;  /
    P01IE=0;P01OE=0;P01OD=0;P01PU=0;    /
    P02IE=0;P02OE=0;P02OD=0;P02PU=0;    /
    P03IE=0;P03OE=0;P03OD=0;P03PU=0;    /
    P04IE=0;P04OE=0;P04OD=0;P04PU=0;    /
    P05IE=0;P05OE=0;P05OD=0;P05PU=0;    /
    P06IE=0;P06OE=0;P06OD=0;P06PU=0;    //
    P07IE=0;P07OE=0;P07OD=0;P07PU=0;    //

    //P00MD3=0;P00MD2=0;P00MD1=0;P00MD0=0;
    P01MD3=0;P01MD2=0;P01MD1=0;P01MD0=0;
    P02MD3=0;P02MD2=0;P02MD1=0;P02MD0=0;
    P03MD3=0;P03MD2=0;P03MD1=0;P03MD0=0;
    P04MD3=0;P04MD2=0;P04MD1=0;P04MD0=0;
    P05MD3=0;P05MD2=0;P05MD1=0;P05MD0=0;
    P06MD3=0;P06MD2=0;P06MD1=0;P06MD0=0;
    P07MD3=0;P07MD2=0;P07MD1=0;P07MD0=0;
    //-----------------------------------
```

## 17.2.3 Port n Mode Register 01 (PnMOD01:n=0 to 9, A, B)

PnMOD01 is a special function register (SFR) to choose the input/output mode, input/output status, and shared function of Pn0 pin and Pn1 pin.
See Table 17-2 "List of Registers / Bits" to check avaible pins and bits.
Wirte "0" to the bits of PnMOD01 register that have no corresponding pin.

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | PnMOD01 | | | | | | | | | | | | | | | |
| Byte | PnMOD1 | | | | | | | | PnMOD0 | | | | | | | |
| Bit | Pn1MD3 | Pn1MD2 | Pn1MD1 | Pn1MD0 | Pn1OD | Pn1PU | Pn1OE | Pn1IE | Pn0MD3 | Pn0MD2 | Pn0MD1 | Pn0MD0 | Pn0OD | Pn0PU | Pn0OE | Pn0IE |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | * | 0 | * |

* :The initial value of P00IE and P00PU for the Port0 is "1" and other bits are "0".

| | |
|---|---|
| Pn1OD | This bit is used choose the output type of Pn1 pin.<br>An LED is directly drive-able by enlarging the current when the N-channel open drain output mode is chosen.<br>See the data sheet for details about the current drive ability.<br>0: CMOS output (initial value)<br>1: N-channel open drain output |
| Pn1PU | This bit is used to enable the internal pull-up resistor of Pn1 pin.<br>The internal pull-up resistor can be enabled on following conditions of the port.<br>The input is enabled and the output is disabled on the port<br>The input is enabled and the N-channel open drain output is chosen on the port<br>0: Without a pull-up resistor (initial value)<br>1: With a pull-up resistor<br><br>The conditions of the port are specified by Pn1IE, Pn1OE and Pn1OD bit.<br>10X: Setting of Pn1PU bit is enable<br>111: Setting of Pn1PU bit is enable<br>Others: Setting of Pn1PU bit is disable<br>X: 0 or 1 (don't care) |

# Initial Port (main.c)

```
void Init_Port(void)
{
    //------------------------------
    //-------- Port 0 Init ----------
    //------------------------------
    //P00DO=0;  // Output = 0
    P01DO=0;    // Output = 0
    P02DO=0;    // Output = 0
    P03DO=0;    // Output = 0
    P04DO=0;    // Output = 0
    P05DO=0;    // Output = 0
    P06DO=0;    // Output = 0
    P07DO=0;    // Output = 0

    //P00IE=0;P00OE=0;P00OD=0;P00PU=0;  //
    P01IE=0;P01OE=0;P01OD=0;P01PU=0;    //
    P02IE=0;P02OE=0;P02OD=0;P02PU=0;    //
    P03IE=0;P03OE=0;P03OD=0;P03PU=0;    //
    P04IE=0;P04OE=0;P04OD=0;P04PU=0;    //
    P05IE=0;P05OE=0;P05OD=0;P05PU=0;    //
    P06IE=0;P06OE=0;P06OD=0;P06PU=0;    //
    P07IE=0;P07OE=0;P07OD=0;P07PU=0;    //

    //P00MD3=0;P00MD2=0;P00MD1=0;P00MD0=0;
    P01MD3=0;P01MD2=0;P01MD1=0;P01MD0=0;
    P02MD3=0;P02MD2=0;P02MD1=0;P02MD0=0;
    P03MD3=0;P03MD2=0;P03MD1=0;P03MD0=0;
    P04MD3=0;P04MD2=0;P04MD1=0;P04MD0=0;
    P05MD3=0;P05MD2=0;P05MD1=0;P05MD0=0;
    P06MD3=0;P06MD2=0;P06MD1=0;P06MD0=0;
    P07MD3=0;P07MD2=0;P07MD1=0;P07MD0=0;
    //-------------------------------------
```

## 17.2.3 Port n Mode Register 01 (PnMOD01:n=0 to 9, A, B)

PnMOD01 is a special function register (SFR) to choose the input/output mode, input/output status, and shared function of Pn0 pin and Pn1 pin.
See Table 17-2 "List of Registers / Bits" to check avaible pins and bits.
Wirte "0" to the bits of PnMOD01 register that have no corresponding pin.

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | PnMOD01 | | | | | | | | | | | | | | | |
| Byte | PnMOD1 | | | | | | | | PnMOD0 | | | | | | | |
| Bit | Pn1MD3 | Pn1MD2 | Pn1MD1 | Pn1MD0 | Pn1OD | Pn1PU | Pn1OE | Pn1IE | Pn0MD3 | Pn0MD2 | Pn0MD1 | Pn0MD0 | Pn0OD | Pn0PU | Pn0OE | Pn0IE |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | * | 0 | * |

* :The initial value of P00IE and P00PU for the Port0 is "1" and other bits are "0".

**Pn1MD3 to Pn1MD0** — This bit is used to choose the shared function of Pn1 pin.
For the details of the shared function, see Table 1-7 "ML62Q1300 Group Pin List", Table 1-8 "ML62Q1500 Group Pin List" and Table 1-9 "ML62Q1700 Group Pin List".

- 0000: Primary function (initial value)
- 0001: 2nd function
- 0010: 3rd function
- 0011: 4th function
- 0100: 5th function
- 0101: 6th function
- 0110: 7th function
- 0111: 8th function
- 1XXX: Do not use (Primary function)

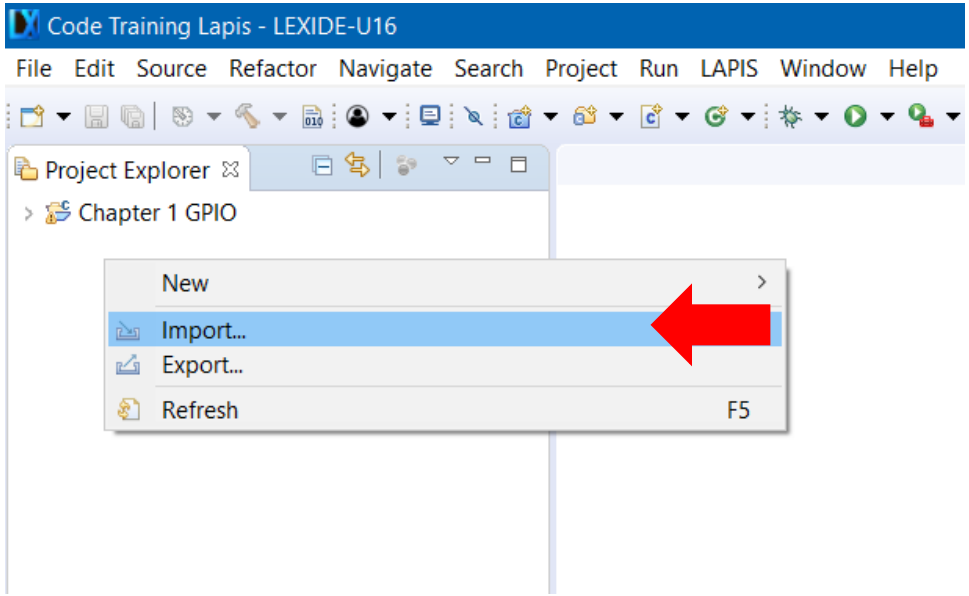
X: 0 or 1 (don't care)

# 2. UART



Pichet Sriphomma pichet@es.co.th

## Import Project

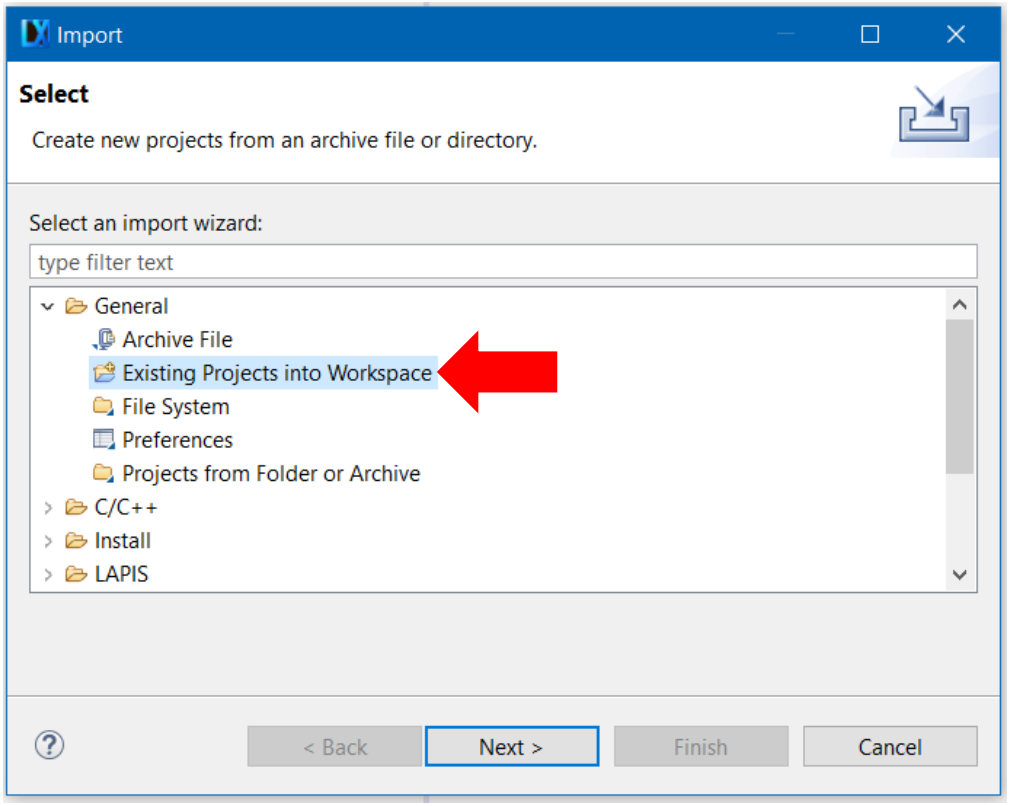



Right-click on project Explorer and select Import.



Pichet Sriphomma pichet@es.co.th

## Import Project

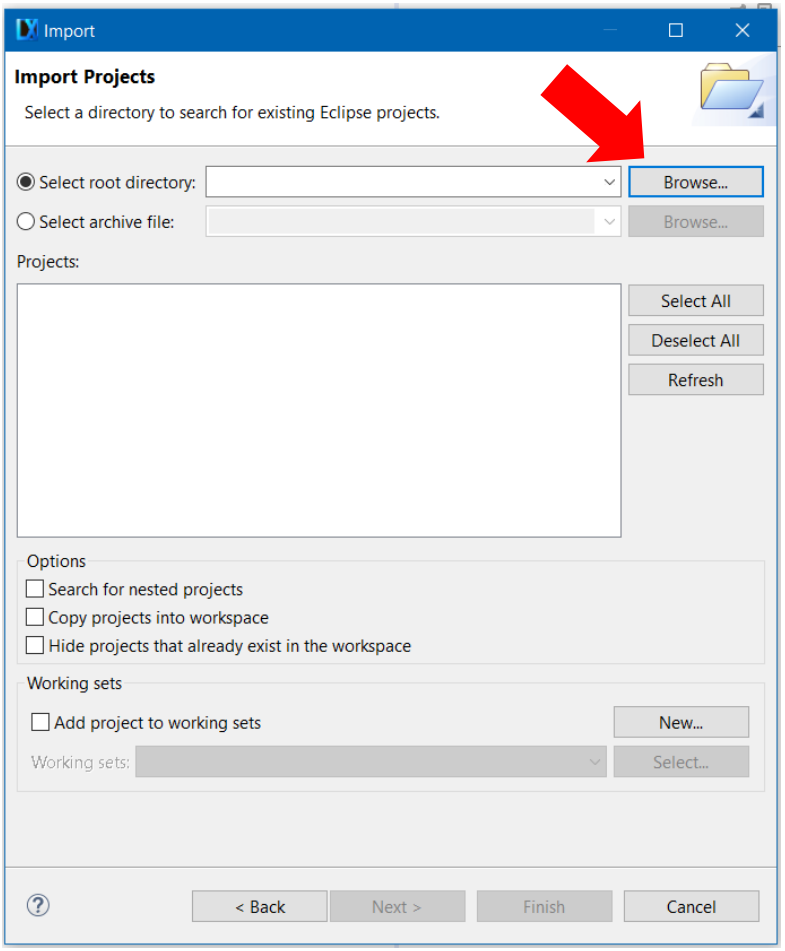



Select General and choose Existing Projects into Workspace. Click Next.

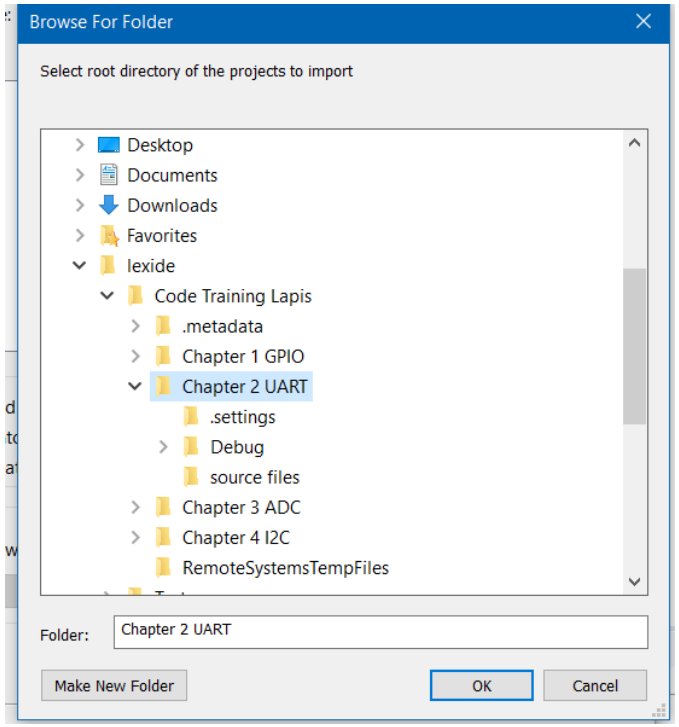


Pichet Sriphomma pichet@es.co.th

## Import Project



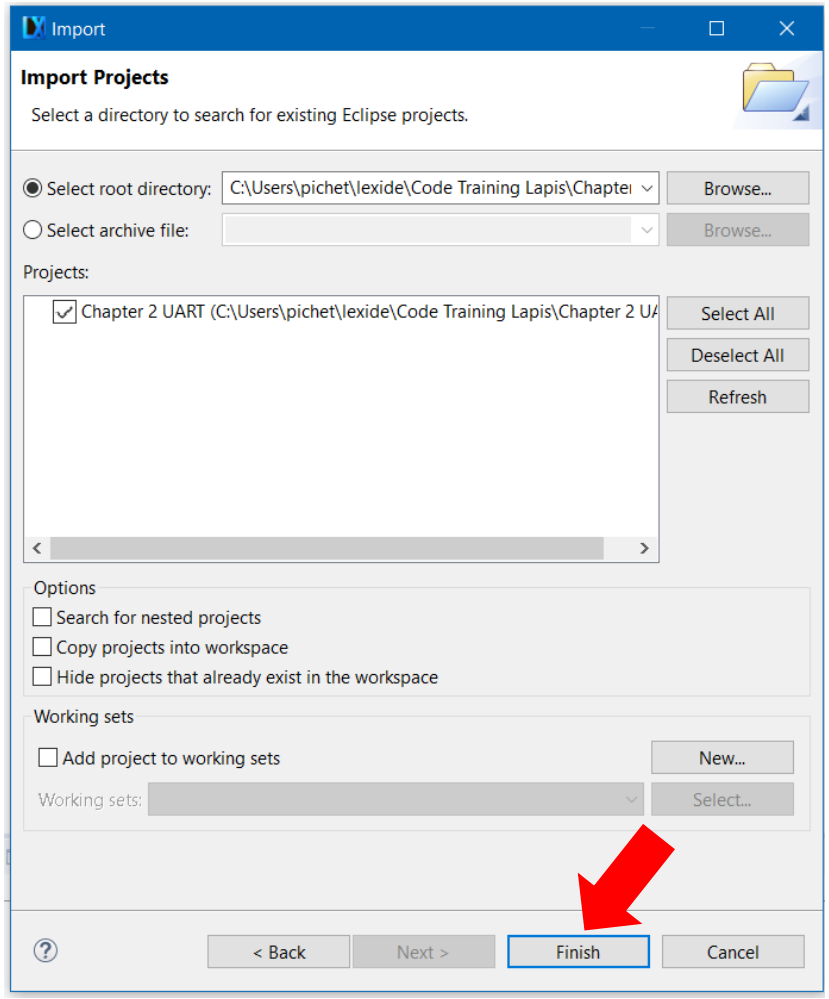

LEXIDE up the new window.
Click Browse.. at Select root directory.
Choose “Chapter 2 UART” in Folder window.






Pichet Sriphomma pichet@es.co.th

# Import Project

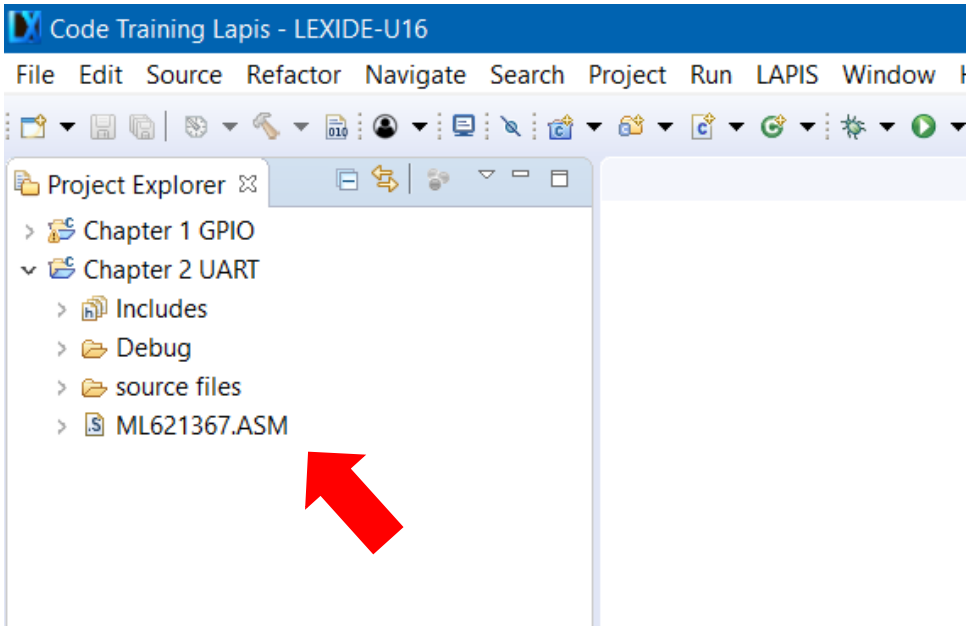



After choosing Project Click Finish.



Pichet Sriphomma pichet@es.co.th

## Import Project

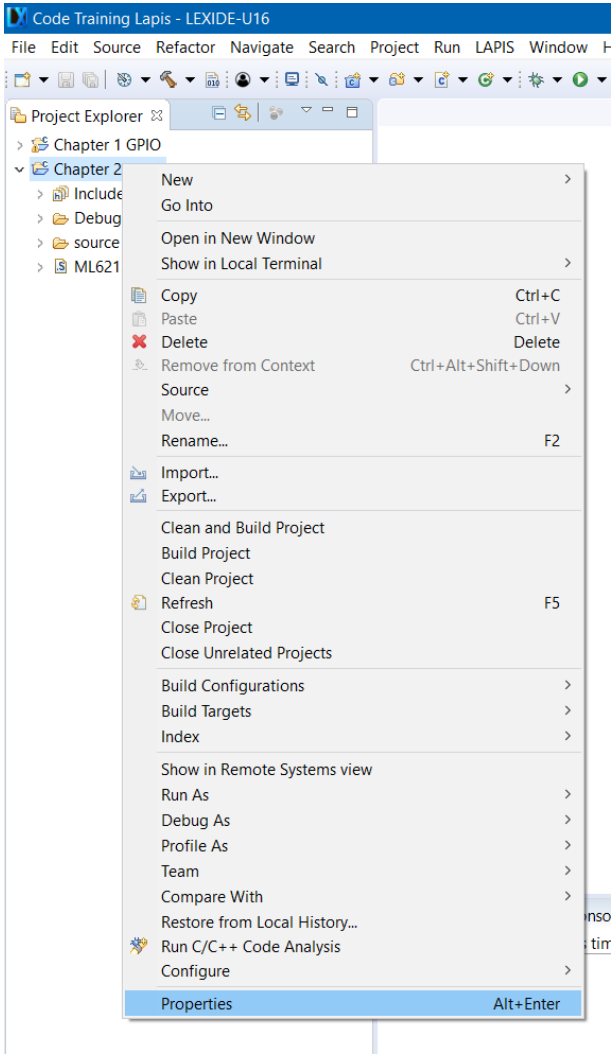



Then appear the project on Project Explorer.



Pichet Sriphomma pichet@es.co.th

## Check Device

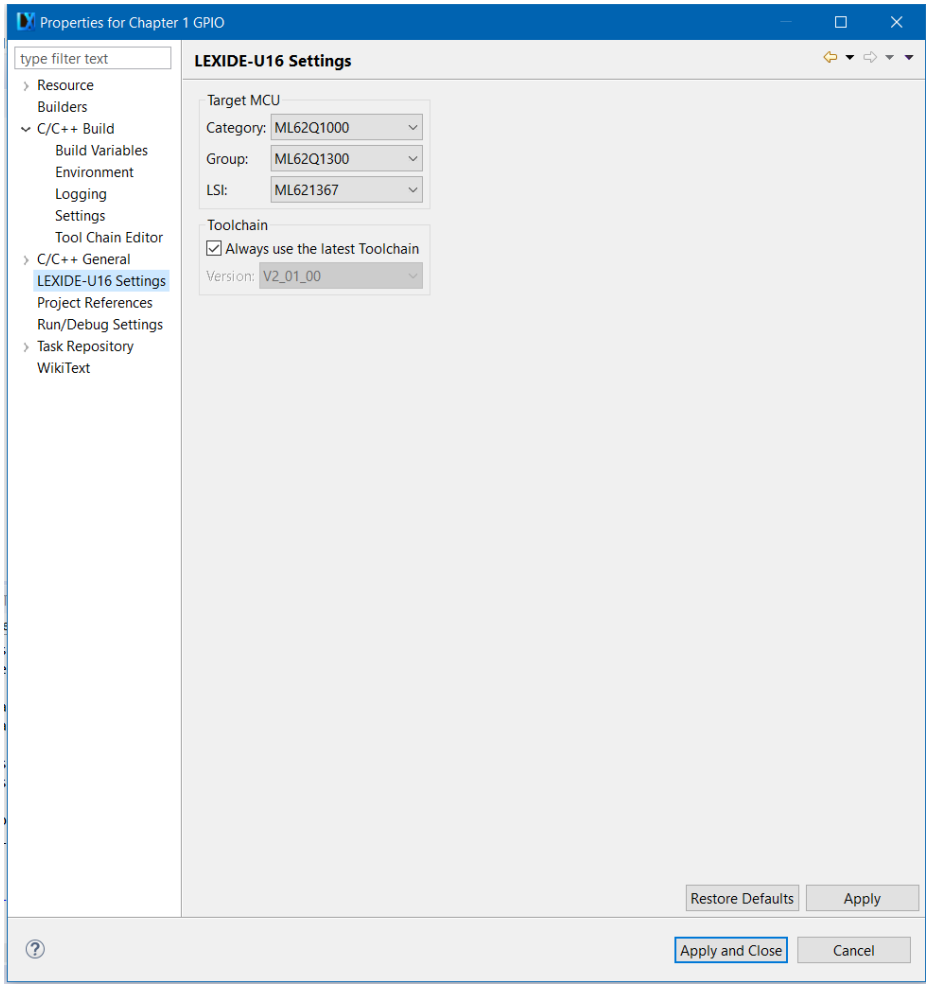



Right-click on a project folder and select [Properties] .



Pichet Sriphomma pichet@es.co.th

## Select Device

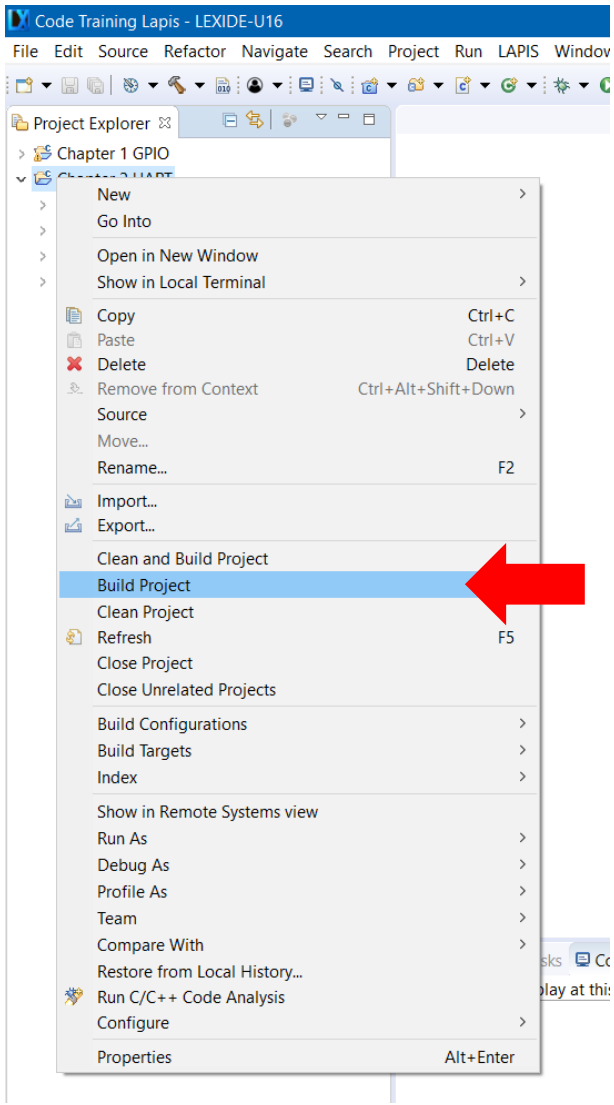



Choose LEXIDE-U16 Settings



Pichet Sriphomma pichet@es.co.th

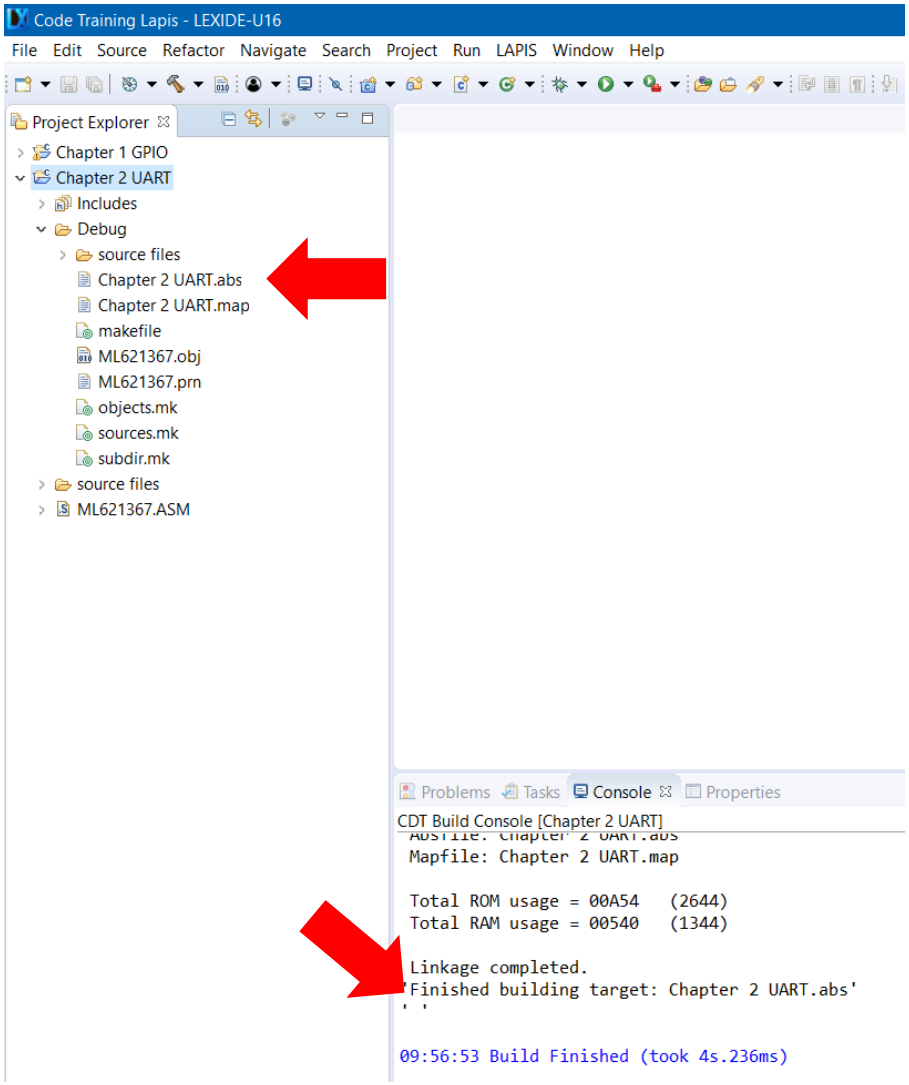
## Build Project




Right-click on a project folder and select [Build Project] to start the build process.

Pichet Sriphomma pichet@es.co.th

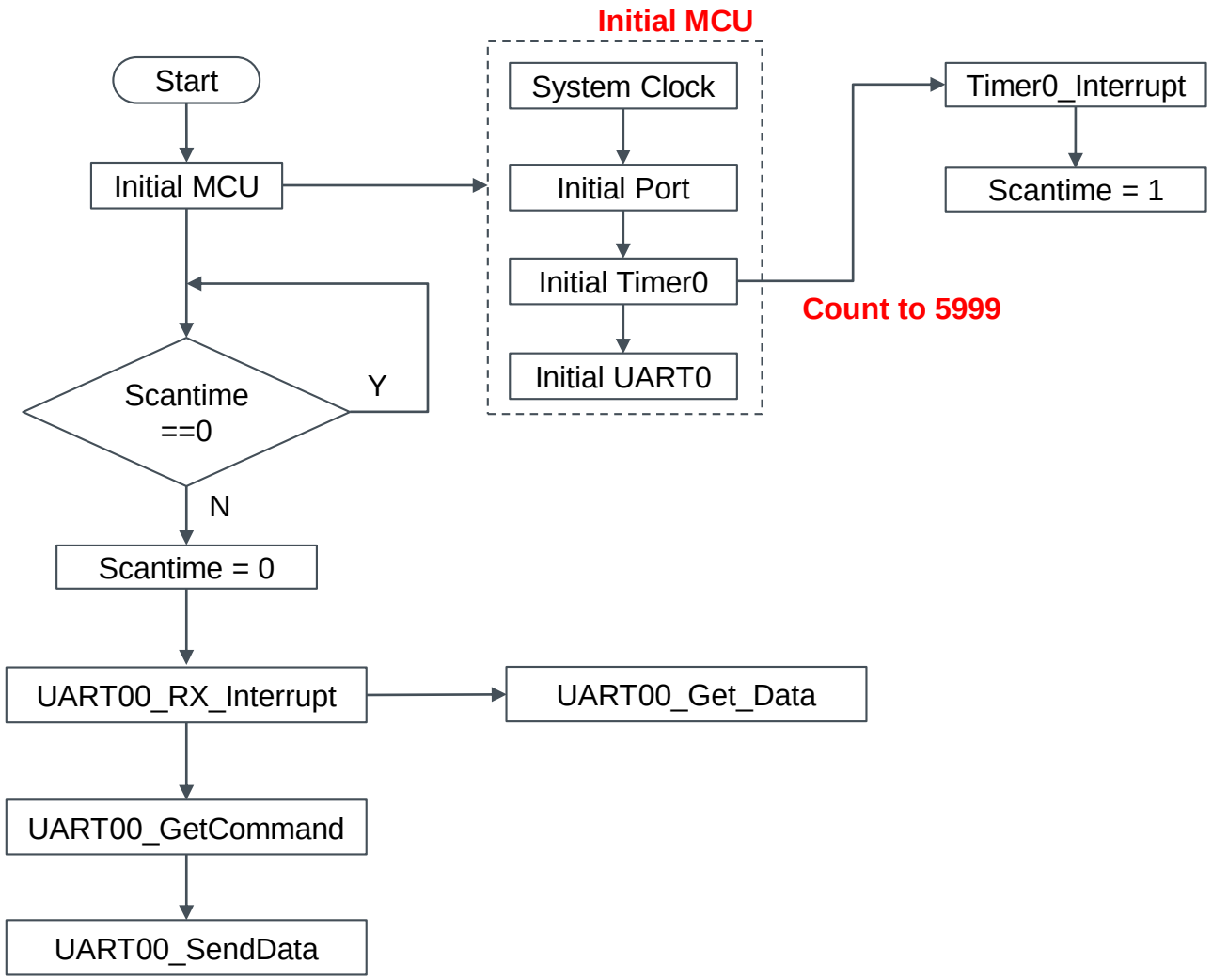
## Build Project




When the build succeeds , an ABS file is generated.

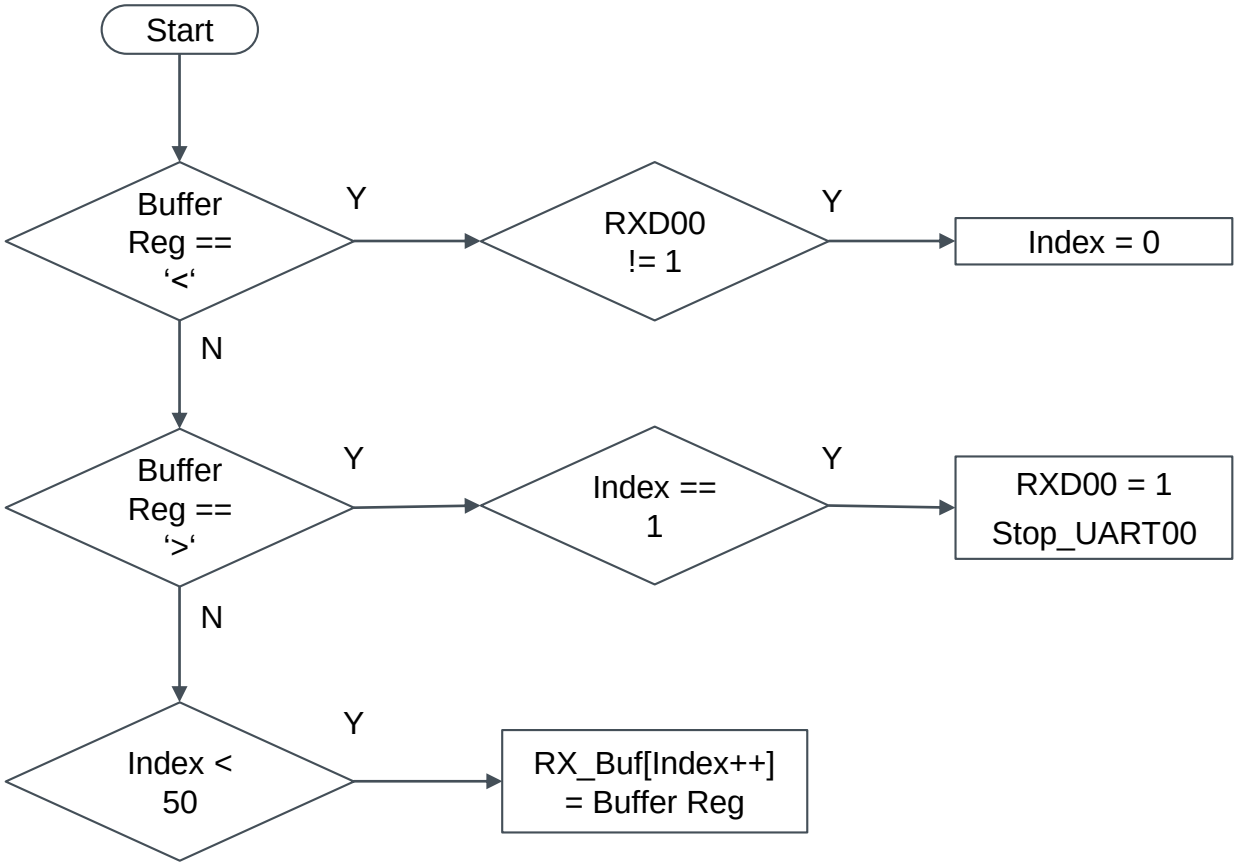


Pichet Sriphomma pichet@es.co.th



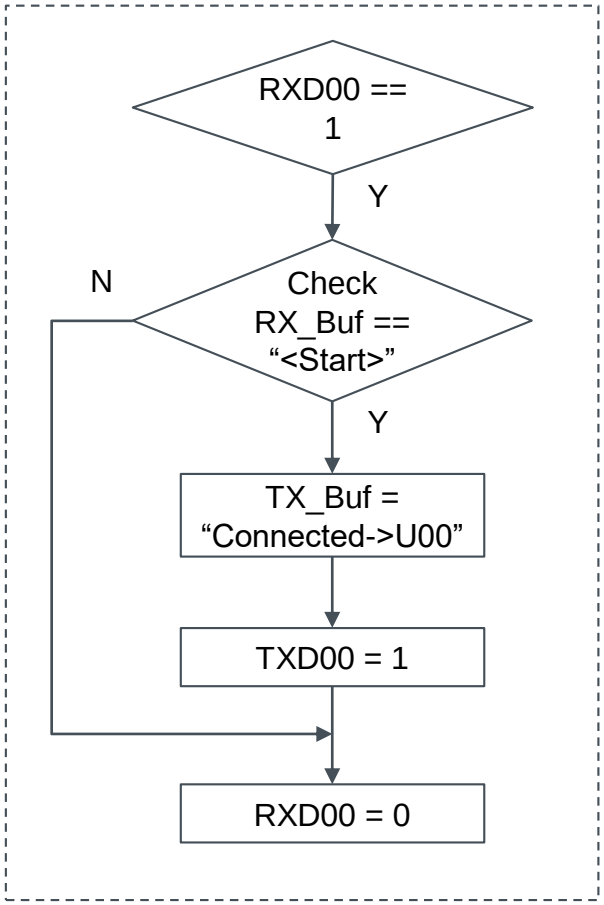

Pichet Sriphomma pichet@es.co.th

## UART00_Get_Data

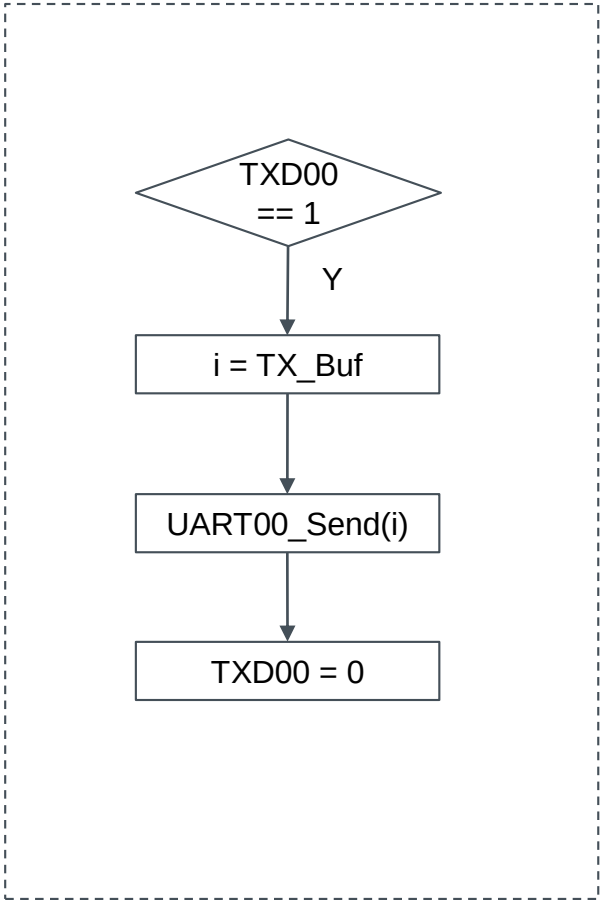



Pichet Sriphomma pichet@es.co.th

**UART00_GetCommand**




**UART00_SendData**




Pichet Sriphomma pichet@es.co.th

## General Description

| Channel no. | ML62Q1300 group | | | | ML62Q1500/ML62Q1700 group | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 16pin product | 20pin product | 24pin product | 32pin product | 48pin product | 52pin product | 64pin product | 80 pin product | 100pin product |
| 0 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| 1 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| 2 | – | – | – | – | – | – | – | ● | ● |
| 3 | – | – | – | – | – | – | – | ● | ● |
| 4 | – | – | – | – | – | – | – | ● | ● |
| 5 | – | – | – | – | – | – | – | ● | ● |

●: Available - : Unavailable



Pichet Sriphomma pichet@es.co.th

## Features

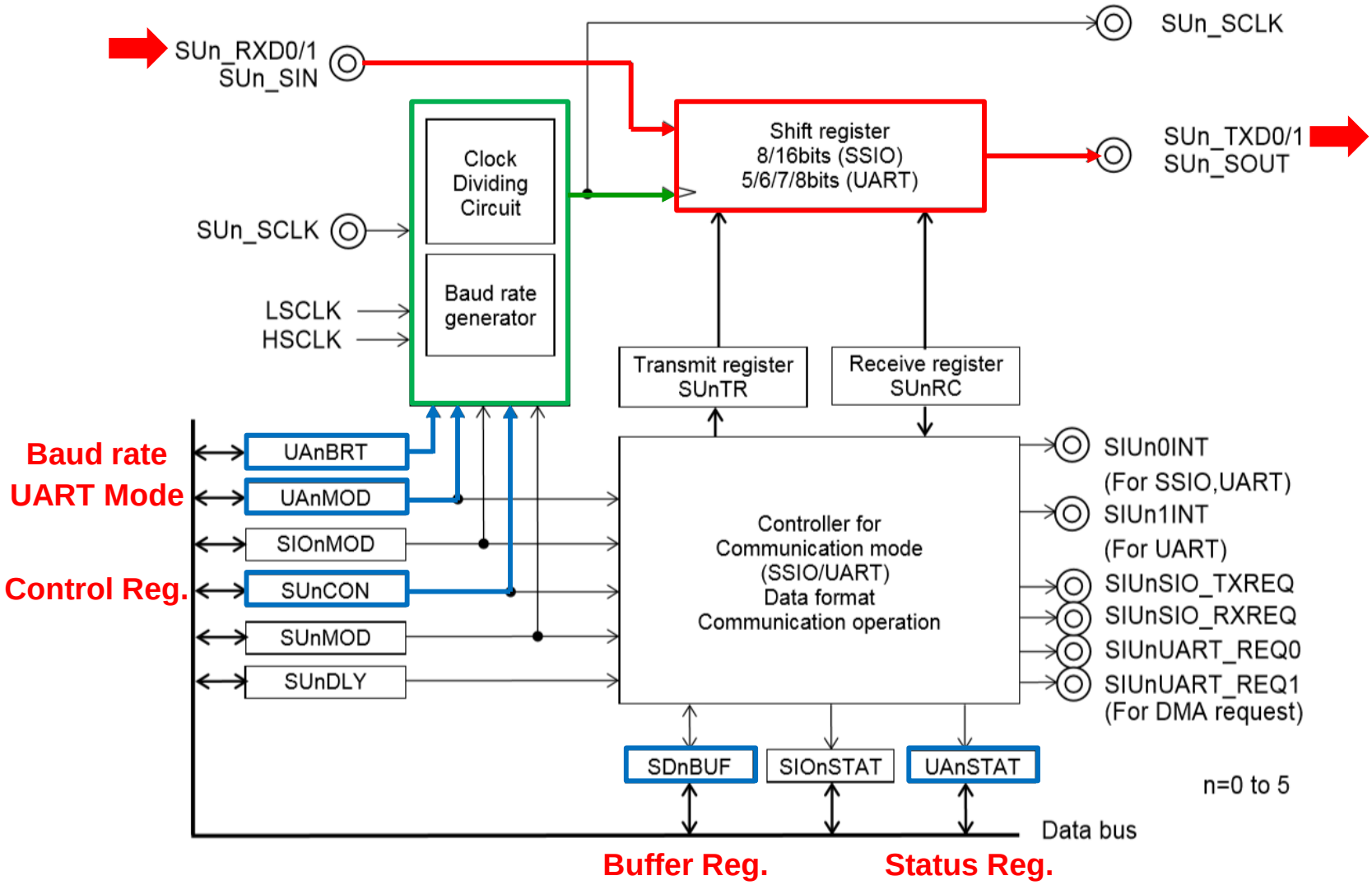
Table 11-2 Features of the Serial Communication

| Serial Communication mode | Operation mode | Features |
|---|---|---|
| Synchronous Serial I/O Port (SSIO) | 8-bit mode | ・ Max. 6ch (Both SSIO and UART are unavailable to use in the same channel)<br>・ Master mode / Slave mode<br>・ MSB first / LSB first<br>・ 8bit / 16bit data length<br>・ Self-test function using the master and slave modes.<br>For the self-test functions, see Chapter 29 "Safety Function." |
| | 16-bit mode | |
| UART mode | Half-duplex communication | ・ 5-bit/6-bit/7-bit/8-bit data length<br>・ Odd parity/even parity/0 parity/1 parity/and no parity<br>・ One stop bit/Two stop bits<br>・ Positive logic/negative logic for communication logic<br>・ MSB first / LSB first<br>・ Wide range of communication speed<br>- 1bps to 4,800bps (Clock frequency is 32.768kHz)<br>- 600bps to 3Mbps (Clock frequency is 24MHz)<br>- 300bps to 2Mbps(Clock frequency is 16MHz)<br>・ Built-in baud rate generator for each channel<br>・ One channel is usable as two channels of half-duplex communication UART<br>・ Parity error flag, overrun error flag, framing error flag, transmission buffer status flag, reception buffer status flag<br>・ Self-test function using transmission and reception<br>For the self-test functions, see Chapter 29 "Safety Function." |
| | Full-duplex communication | |



Pichet Sriphomma pichet@es.co.th

## Configuration




Pichet Sriphomma pichet@es.co.th

## List of Pins

| Channel no. | Pin name | Shared port | | Setting register | Setting value | ML62Q1300 group | | | | ML62Q1500 ML62Q1700 group | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | 16pin product | 20 pin product | 24 pin product | 32 pin product | 48 pin product | 52 pin product | 64 pin product | 80 pin product | 100 pin product |
| 0 | SU0_TXD0 | P03 | 2nd Func. | P0MOD3 | 0001_XXXX*1 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | | P13 | 2nd Func. | P1MOD3 | 0001_XXXX*1 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | SU0_RXD0 | P02 | 2nd Func. | P0MOD2 | 0001_XXXX*2 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | | P07 | 3rd Func. | P0MOD7 | 0010_XXXX*2 | - | - | - | ● | ● | ● | ● | ● | ● |
| | | P12 | 2nd Func. | P1MOD2 | 0001_XXXX*2 | - | - | ● | ● | ● | ● | ● | ● | ● |
| | | P17 | 3rd Func. | P1MOD7 | 0010_XXXX*2 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | SU0_TXD1 | P03 | 3rd Func. | P0MOD3 | 0010_XXXX*1 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | | P10 | 2nd Func. | P1MOD0 | 0001_XXXX*1 | - | - | - | ● | ● | ● | ● | ● | ● |
| | | P13 | 3rd Func. | P1MOD3 | 0010_XXXX*1 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | | P20 | 2nd Func. | P2MOD0 | 0001_XXXX*1 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | SU0_RXD1 | P07 | 2nd Func. | P0MOD7 | 0001_XXXX*2 | - | - | - | ● | ● | ● | ● | ● | ● |
| | | P17 | 2nd Func. | P1MOD7 | 0001_XXXX*2 | ● | ● | ● | ● | ● | ● | ● | ● | ● |



Pichet Sriphomma pichet@es.co.th

## Combination of UART port

- Full-duplex communication

| Input/output pin | Combination 1 | Buffer register | |
|---|---|---|---|
| | | SDnBUFL | SDnBUFH |
| SUn_RXD | SUn_RXD0 | Received data (SUn_RXD0) | Transmit data (SUn_TXD1) |
| SUn_TXD | SUn_TXD1 | | |

n=0 to 5

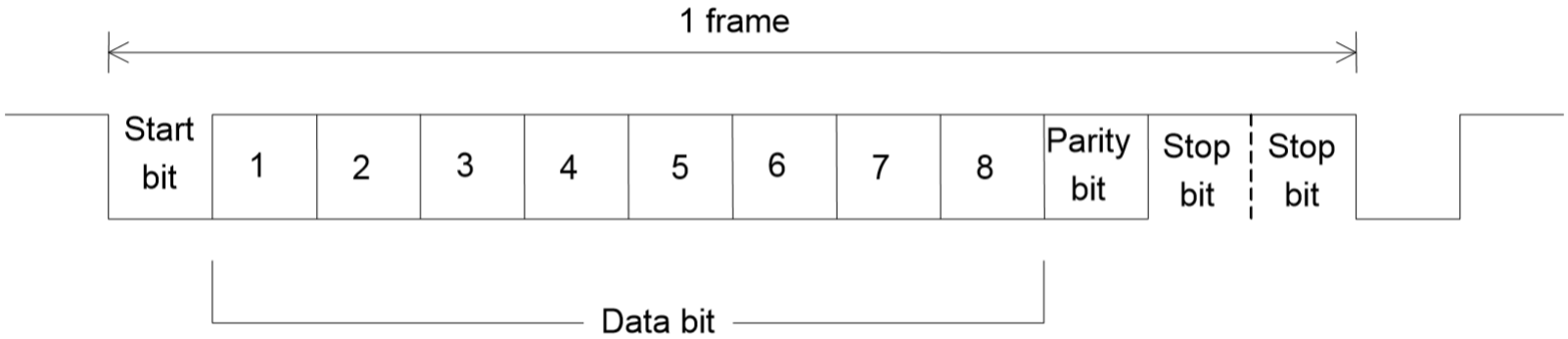
- Half-duplex communication

| Input/output pin | Combination 1 | Combination 2 | Buffer register | |
|---|---|---|---|---|
| | | | SDnBUFL | SDnBUFH |
| SUn_RXD | SUn_RXD0 | SUn_RXD1 | UARTn0 Receive data (SUn_RXD0) | UARTn1 Receive data (SUn_RXD1) |
| SUn_TXD | SUn_TXD1 | SUn_TXD0 | UARTn0 Transmit data (SUn_TXD0) | UARTn1 Transmit data (SUn_TXD1) |

n=0 to 5

**Channel 0** **Channel 1**

## Transfer Data Format




- 1 frame
  MAX ..... 12 bits
  MIN..... 7 bits

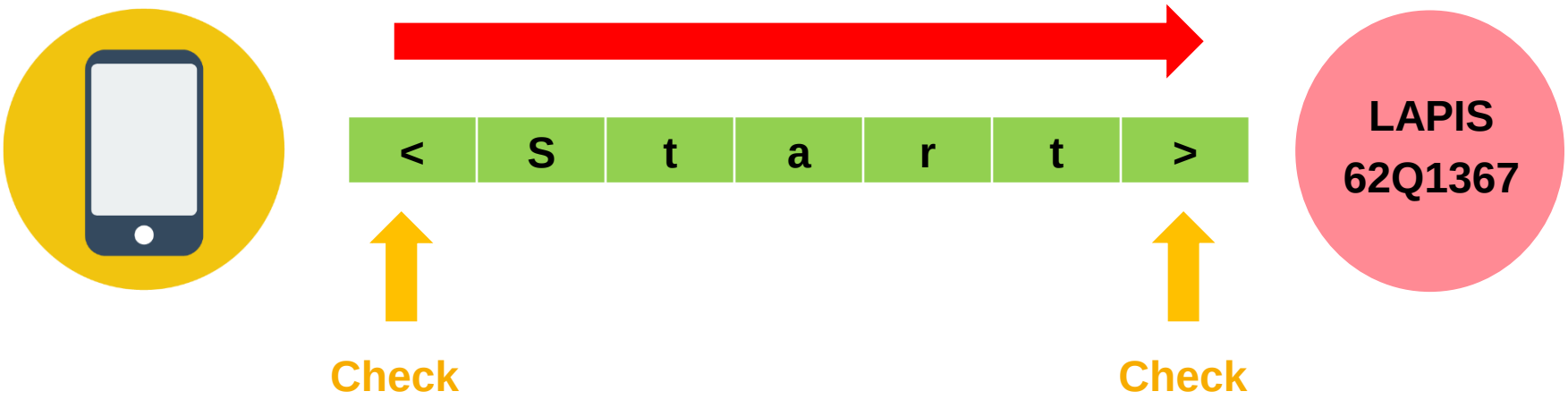
- Data bit length .... 8 to 5 bits
- Parity bit..... Use or not use
  Odd parity or even parity
  Parity fixed to "1" or "0"
- Stop bit ..... 1 stop bit or 2 stop bits

Figure 11-12 Format of Positive Logic Input/Output (LSB First)



Pichet Sriphomma pichet@es.co.th

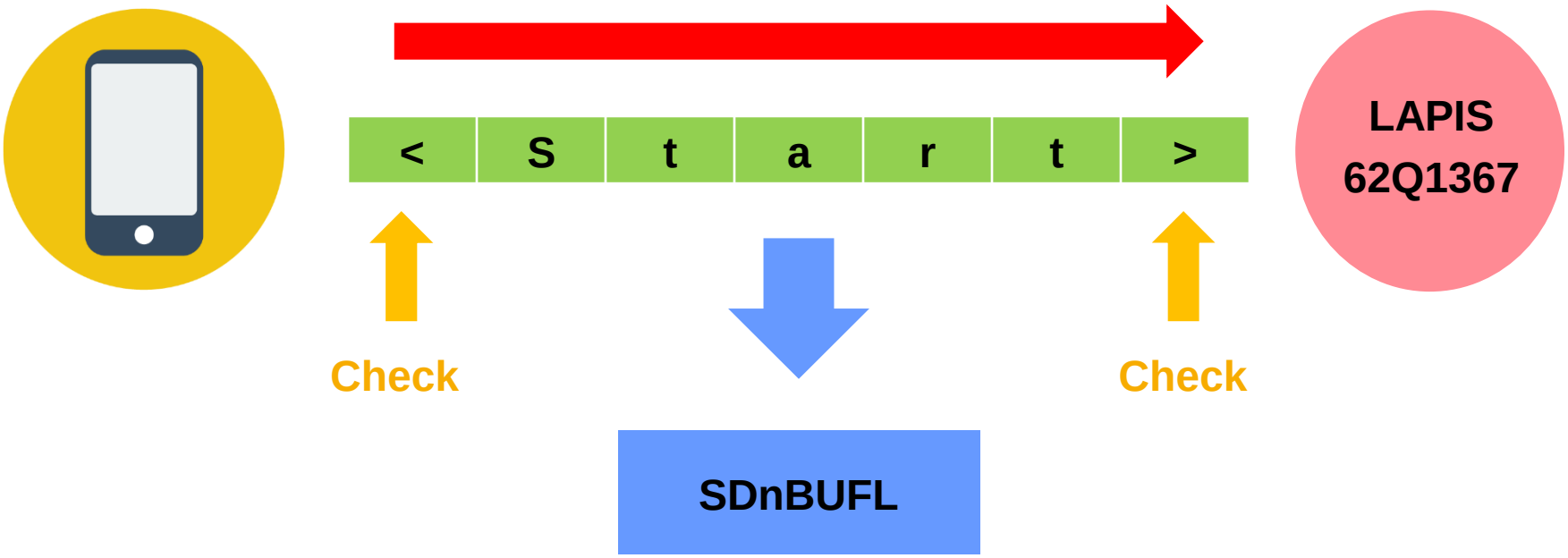
## Concept






Pichet Sriphomma pichet@es.co.th

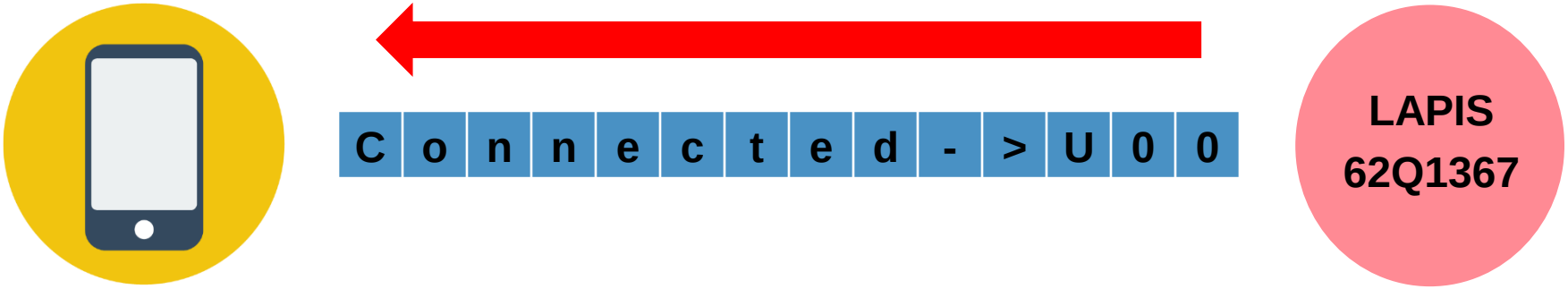
## Concept






Pichet Sriphomma pichet@es.co.th

## Concept






Pichet Sriphomma pichet@es.co.th

## Set Pin (uart0.c)

```
void Set_UART_00_Pin(void)
{
    // UART00 RXD Pin
    P02IE=1;P02OE=0;P02OD=0;P02PU=0;
    P02MD3=0;P02MD2=0;P02MD1=0;P02MD0=1;

    P03IE=0;P03OE=1;P03OD=0;P03PU=0;
    P03MD3=0;P03MD2=0;P03MD1=0;P03MD0=1;
}
```

### 17.2.3 Port n Mode Register 01 (PnMOD01:n＝0 to 9, A, B)

PnMOD01 is a special function register (SFR) to choose the input/output mode, input/output status, and shared function of Pn0 pin and Pn1 pin.
See Table 17-2 “List of Registers / Bits” to check avaible pins and bits.
Wirte “0” to the bits of PnMOD01 register that have no corresponding pin.

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | PnMOD01 | | | | | | | | | | | | | | | |
| Byte | PnMOD1 | | | | | | | | PnMOD0 | | | | | | | |
| Bit | Pn1MD3 | Pn1MD2 | Pn1MD1 | Pn1MD0 | Pn1OD | Pn1PU | Pn1OE | Pn1IE | Pn0MD3 | Pn0MD2 | Pn0MD1 | Pn0MD0 | Pn0OD | Pn0PU | Pn0OE | Pn0IE |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | * | 0 | * |

* :The initial value of P00IE and P00PU for the Port0 is "1" and other bits are "0".

| | | |
|---|---|---|
| 9 | Pn1OE | This bit is used to enable the output of Pn1 pin<br>0: Disable the output (initial value)<br>1: Enable the output |
| 8 | Pn1IE | This bit is used to enable the input of Pn1 pin<br>0: Disable the input (initial value)<br>1: Enable the input |



Pichet Sriphomma pichet@es.co.th

## Set Pin (uart0.c)

```
void Set_UART_00_Pin(void)
{
    // UART00 RXD Pin
    P02IE=1;P02OE=0;P02OD=0;P02PU=0;
    P02MD3=0;P02MD2=0;P02MD1=0;P02MD0=1;

    P03IE=0;P03OE=1;P03OD=0;P03PU=0;
    P03MD3=0;P03MD2=0;P03MD1=0;P03MD0=1;
}
```

### 17.2.3 Port n Mode Register 01 (PnMOD01:n=0 to 9, A, B)

PnMOD01 is a special function register (SFR) to choose the input/output mode, input/output status, and shared function of Pn0 pin and Pn1 pin.
See Table 17-2 "List of Registers / Bits" to check avaible pins and bits.
Wirte "0" to the bits of PnMOD01 register that have no corresponding pin.

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | PnMOD01 | | | | | | | | | | | | | | | |
| Byte | PnMOD1 | | | | | | | | PnMOD0 | | | | | | | |
| Bit | Pn1MD3 | Pn1MD2 | Pn1MD1 | Pn1MD0 | Pn1OD | Pn1PU | Pn1OE | Pn1IE | Pn0MD3 | Pn0MD2 | Pn0MD1 | Pn0MD0 | Pn0OD | Pn0PU | Pn0OE | Pn0IE |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | * | 0 | * |

* :The initial value of P00IE and P00PU for the Port0 is "1" and other bits are "0".

| | |
|---|---|
| Pn1OD | This bit is used choose the output type of Pn1 pin.<br>An LED is directly drive-able by enlarging the current when the N-channel open drain output mode is chosen.<br>See the data sheet for details about the current drive ability.<br>0: CMOS output (initial value)<br>1: N-channel open drain output |
| Pn1PU | This bit is used to enable the internal pull-up resistor of Pn1 pin.<br>The internal pull-up resistor can be enabled on following conditions of the port.<br>The input is enabled and the output is disabled on the port<br>The input is enabled and the N-channel open drain output is chosen on the port<br>0: Without a pull-up resistor (initial value)<br>1: With a pull-up resistor<br><br>The conditions of the port are specified by Pn1IE, Pn1OE and Pn1OD bit.<br>10X: Setting of Pn1PU bit is enable<br>111: Setting of Pn1PU bit is enable<br>Others: Setting of Pn1PU bit is disable<br>X: 0 or 1 (don't care) |

## Set Pin (uart0.c)

```
void Set_UART_00_Pin(void)
{
    // UART00 RXD Pin
    P02IE=1;P02OE=0;P02OD=0;P02PU=0;
    P02MD3=0;P02MD2=0;P02MD1=0;P02MD0=1;

    P03IE=0;P03OE=1;P03OD=0;P03PU=0;
    P03MD3=0;P03MD2=0;P03MD1=0;P03MD0=1;
}
```

### 17.2.3 Port n Mode Register 01 (PnMOD01:n=0 to 9, A, B)

PnMOD01 is a special function register (SFR) to choose the input/output mode, input/output status, and shared function of Pn0 pin and Pn1 pin.
See Table 17-2 "List of Registers / Bits" to check avaible pins and bits.
Wirte "0" to the bits of PnMOD01 register that have no corresponding pin.

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | PnMOD01 | | | | | | | | | | | | | | | |
| Byte | PnMOD1 | | | | | | | | PnMOD0 | | | | | | | |
| Bit | Pn1MD3 | Pn1MD2 | Pn1MD1 | Pn1MD0 | Pn1OD | Pn1PU | Pn1OE | Pn1IE | Pn0MD3 | Pn0MD2 | Pn0MD1 | Pn0MD0 | Pn0OD | Pn0PU | Pn0OE | Pn0IE |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | * | 0 | * |

* :The initial value of P00IE and P00PU for the Port0 is "1" and other bits are "0".

Pn1MD3 to Pn1MD0 — This bit is used to choose the shared function of Pn1 pin.
For the details of the shared function, see Table 1-7 "ML62Q1300 Group Pin List", Table 1-8 "ML62Q1500 Group Pin List" and Table 1-9 "ML62Q1700 Group Pin List".

- 0000: Primary function (initial value)
- 0001: 2nd function
- 0010: 3rd function
- 0011: 4th function
- 0100: 5th function
- 0101: 6th function
- 0110: 7th function
- 0111: 8th function
- 1XXX: Do not use (Primary function)

X: 0 or 1 (don't care)

## Start ,Stop UART (uart0.c)

```
void Stop_UART00(void)
{
    U00EN = 0;        // S
}
//----------------------
void Start_UART00(void)
{
    unsigned char i;

    i = SD0BUFL;      // R
    UA00STAT = 0xff;
    U00EN = 1;        // S
}
```

### 11.2.5 Serial Communication Unit n Control Register (SUnCON)

SUnCON is a specific function register (SFR) to control the serial communication unit.

Address: 0xF606(SU0CONL/SU0CON), 0xF607(SU0CONH), 0xF626(SU1CONL/SU1CON), 0xF627(SU1CONH), 0xF646(SU2CONL/SU2CON), 0xF647(SU2CONH), 0xF666(SU3CONL/SU3CON), 0xF667(SU3CONH), 0xF686(SU4CONL/SU4CON), 0xF687(SU4CONH), 0xF6A6(SU5CONL/SU5CON), 0xF6A7(SU5CONH)
Access: R/W
Access size: 8/16bit
Initial value: 0x0000

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | SUnCON | | | | | | | | | | | | | | | |
| Byte | SUnCONH | | | | | | | | SUnCONL | | | | | | | |
| Bit | – | – | – | – | – | – | Un1EN | – | – | – | – | – | – | – | Un0EN | SnEN |
| R/W | R | R | R | R | R | R | R/W | R | R | R | R | R | R | R | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | |
|---|---|
| Un0EN | This bit is used to enable the UARTn communication in the UART mode.<br>• UART Full-duplex mode<br>0: Stop the UARTn communication (Initial value)<br>1: Start the UARTn communication<br>• UART Half-duplex mode<br>0: Stop the UARTn0 communication (Initial value)<br>1: Start the UARTn0 communication<br>• SSIO mode<br>The Un0EN bit is unwritable.<br>0: Unused<br>1: Unused |

## Config UART (uart0.c)

```
void Config_UART_00(void)
{
    unsigned char i;

    Set_UART_00_Pin();
    Stop_UART00();

    //UA00MOD -> UART00 Mode
    U00IO = 0;              // Recei
    U00CK1=1;U00CK0=0;      // Selec
    U00RSS = 0;             // Value
    U00LG1=0;U00LG0=0;      // 8-bit
    U00PT2=0;U00PT1=0;U00PT0=0;
    U00STP = 0;      // 1 stop bi
    U00NEG = 0;      // Positive
    U00DIR = 0;      // LSB first

    // 9600 bps Baud Rate Settir
        UA00BRT = 0x09c1;    // E
        UA00BRC = 0x04; // Baud
    // 115,200 bps Baud Rate Set
    //  UA00BRT = 0x00cf;    // E
    //  UA00BRC = 0x02; // Baud

    Set_UART00_RX();
    //Set_UART00_TX();
}
```

### 11.2.8 UARTn0 Mode Register (UAn0MOD)

UAn0MOD is a specific function register (SFR) to set the mode in UARTn0 full-duplex communiction mode and half-duplex communication mode.

Address: 0xF60C(UA00MODL/UA00CON), 0xF60D(UA00MODH),
0xF62C(UA10MODL/UA10MOD), 0xF62D(UA10MODH),
0xF64C(UA20MODL/UA20CON), 0xF64D(UA20MODH),
0xF66C(UA30MODL/UA30MOD), 0xF66D(UA30MODH),
0xF68C(UA40MODL/UA40CON), 0xF68D(UA40MODH),
0xF6AC(UA50MODL/UA50MOD), 0xF6AD(UA50MODH)
Access: R/W
Access size: 8/16bit
Initial value: 0x0000

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | UAn0MOD | | | | | | | | | | | | | | | |
| Byte | UAn0MODH | | | | | | | | UAn0MODL | | | | | | | |
| Bit | Un0DIR | Un0NEG | Un0STP | Un0PT2 | Un0PT1 | Un0PT0 | Un0LG1 | Un0LG0 | Un0RSS | – | – | – | – | Un0CK1 | Un0CK0 | Un0IO |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | |
|---|---|
| Un0CK1 to Un0CK0 | This bit is used to choose base clock of baud rate generator in UARTn0 full-duplex and half-duplex mode.<br>00: LSCLK (initial value)<br>01: Do not use (LSCLK)<br>10: HSCLK<br>11: Do not use (HSCLK) |
| Un0IO | This bit is used to choose the transmission mode or reception mode in UARTn0 full-duplex and half-duplex mode. When the full-duplex communication mode is chosen, this bit is fixed to “1” and the UART performs as the reception mode.<br>0: Transmission mode (Initial value)<br>1: Reception mode |



Pichet Sriphomma pichet@es.co.th

# Config UART (uart0.c)

```
void Config_UART_00(void)
{
    unsigned char i;

    Set_UART_00_Pin();
    Stop_UART00();

    //UA00MOD -> UART00 Mode
    U00IO = 0;           // Recei
    U00CK1=1;U00CK0=0;   // Selec
    U00RSS = 0;                lue
    U00LG1=0;U00LG0=0;   // 8-bit
    U00PT2=0;U00PT1=0;U00PT0=0;
    U00STP = 0;      // 1 stop bi
    U00NEG = 0;      // Positive
    U00DIR = 0;      // LSB first

    // 9600 bps Baud Rate Settir
        UA00BRT = 0x09c1;    // B
        UA00BRC = 0x04; // Baud
    // 115,200 bps Baud Rate Set
    //  UA00BRT = 0x00cf;    // B
    //  UA00BRC = 0x02; // Baud

    Set_UART00_RX();
    //Set_UART00_TX();
}
```

## 11.2.8 UARTn0 Mode Register (UAn0MOD)

UAn0MOD is a specific function register (SFR) to set the mode in UARTn0 full-duplex communiction mode and half-duplex communication mode.

Address: 0xF60C(UA00MODL/UA00CON), 0xF60D(UA00MODH),
0xF62C(UA10MODL/UA10MOD), 0xF62D(UA10MODH),
0xF64C(UA20MODL/UA20CON), 0xF64D(UA20MODH),
0xF66C(UA30MODL/UA30MOD), 0xF66D(UA30MODH),
0xF68C(UA40MODL/UA40CON), 0xF68D(UA40MODH),
0xF6AC(UA50MODL/UA50MOD), 0xF6AD(UA50MODH)

Access: R/W

Access size: 8/16bit

Initial value: 0x0000

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | UAn0MOD | | | | | | | | | | | | | | | |
| Byte | UAn0MODH | | | | | | | | UAn0MODL | | | | | | | |
| Bit | Un0DIR | Un0NEG | Un0STP | Un0PT2 | Un0PT1 | Un0PT0 | Un0LG1 | Un0LG0 | Un0RSS | – | – | – | – | Un0CK1 | Un0CK0 | Un0IO |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Un0RSS — This bit is used to choose sampling timing of the reception data in UARTn0 full-duplex and half-duplex mode.

- 0: (Values set to UAn0BRTH and UAn0BRTL registers)/2 (initial value)
- 1: {(Values set to UAn0BRTH and UAn0BRTL registers)/2} -1

Un0LG1 to Un0LG0 — This bit is used to choose the communication data length in UARTn0 full-duplex and half-duplex mode.

- 00: 8-bit length (Initial value)
- 01: 7-bit length
- 10: 6-bit length
- 11: 5-bit length

# Config UART (uart0.c)

```
void Config_UART_00(void)
{
    unsigned char i;

    Set_UART_00_Pin();
    Stop_UART00();

    //UA00MOD -> UART00 Mode
    U00IO = 0;           // Recei
    U00CK1=1;U00CK0=0;   // Selec
    U00RSS = 0;          // Value
    U00LG1=0;U00LG0=0;   // 8-bit
    U00PT2=0;U00PT1=0;U00PT0=0;
    U00STP = 0;          // 1 stop bi
    U00NEG = 0;          // Positive
    U00DIR = 0;          // LSB first

    // 9600 bps Baud Rate Settir
        UA00BRT = 0x09c1;    // E
        UA00BRC = 0x04; // Baud
    // 115,200 bps Baud Rate Set
    //  UA00BRT = 0x00cf;    // E
    //  UA00BRC = 0x02; // Baud

    Set_UART00_RX();
    //Set_UART00_TX();
}
```

## 11.2.8 UARTn0 Mode Register (UAn0MOD)

UAn0MOD is a specific function register (SFR) to set the mode in UARTn0 full-duplex communiction mode and half-duplex communication mode.

Address: 0xF60C(UA00MODL/UA00CON), 0xF60D(UA00MODH),
0xF62C(UA10MODL/UA10MOD), 0xF62D(UA10MODH),
0xF64C(UA20MODL/UA20CON), 0xF64D(UA20MODH),
0xF66C(UA30MODL/UA30MOD), 0xF66D(UA30MODH),
0xF68C(UA40MODL/UA40CON), 0xF68D(UA40MODH),
0xF6AC(UA50MODL/UA50MOD), 0xF6AD(UA50MODH)

Access: R/W

Access size: 8/16bit

Initial value: 0x0000

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | UAn0MOD | | | | | | | | | | | | | | | |
| Byte | UAn0MODH | | | | | | | | UAn0MODL | | | | | | | |
| Bit | Un0DIR | Un0NEG | Un0STP | Un0PT2 | Un0PT1 | Un0PT0 | Un0LG1 | Un0LG0 | Un0RSS | – | – | – | – | Un0CK1 | Un0CK0 | Un0IO |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Un0STP — This bit is used to choose the stop bit length in UARTn0 full-duplex and half-duplex mode.
- 0: 1 stop bit (Initial value)
- 1: 2 stop bit

Un0PT2 to Un0PT0 — This bit is used to choose the parity bit in UARTn0 full-duplex and half-duplex mode.
- 000: No parity bit (initial value)
- 001: Odd parity
- 010: No parity bit
- 011: Even parity
- 100: No parity bit
- 101: Parity bit is fixed to "1"
- 110: No parity bit
- 111: Parity bit is fixed to "0"

# Config UART (uart0.c)

```
void Config_UART_00(void)
{
    unsigned char i;

    Set_UART_00_Pin();
    Stop_UART00();

    //UA00MOD -> UART00 Mode
    U00IO = 0;           // Recei
    U00CK1=1;U00CK0=0;   // Selec
    U00RSS = 0;          // Value
    U00LG1=0;U00LG0=0;   // 8-bit
    U00PT2=0;U00PT1=0;U00PT0=0;
    U00STP = 0;      // 1 stop bi
    U00NEG = 0;      // Positive
    U00DIR = 0;      // LSB first

    // 9600 bps Baud Rate Settir
        UA00BRT = 0x09c1;    // E
        UA00BRC = 0x04; // Baud
    // 115,200 bps Baud Rate Set
    //  UA00BRT = 0x00cf;    // E
    //  UA00BRC = 0x02; // Baud

    Set_UART00_RX();
    //Set_UART00_TX();
}
```

## 11.2.8 UARTn0 Mode Register (UAn0MOD)

UAn0MOD is a specific function register (SFR) to set the mode in UARTn0 full-duplex communiction mode and half-duplex communication mode.

Address: 0xF60C(UA00MODL/UA00CON), 0xF60D(UA00MODH),
0xF62C(UA10MODL/UA10MOD), 0xF62D(UA10MODH),
0xF64C(UA20MODL/UA20CON), 0xF64D(UA20MODH),
0xF66C(UA30MODL/UA30MOD), 0xF66D(UA30MODH),
0xF68C(UA40MODL/UA40CON), 0xF68D(UA40MODH),
0xF6AC(UA50MODL/UA50MOD), 0xF6AD(UA50MODH)
Access: R/W
Access size: 8/16bit
Initial value: 0x0000

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | UAn0MOD | | | | | | | | | | | | | | | |
| Byte | UAn0MODH | | | | | | | | UAn0MODL | | | | | | | |
| Bit | Un0DIR | Un0NEG | Un0STP | Un0PT2 | Un0PT1 | Un0PT0 | Un0LG1 | Un0LG0 | Un0RSS | – | – | – | – | Un0CK1 | Un0CK0 | Un0IO |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | |
|---|---|
| Un0DIR | This bit is used to choose the communication direction in UARTn0 full-duplex and half-duplex mode.<br>0: LSB first (Initial value)<br>1: MSB first |
| Un0NEG | This bit is used to choose logic of the data output in UARTn0 full-duplex and half-duplex mode.<br>0: Positive logic (Initial value)<br>1: Negative logic |

## Config UART (uart0.c)

```
void Config_UART_00(void)
{
    unsigned char i;

    Set_UART_00_Pin();
    Stop_UART00();

    //UA00MOD -> UART00 Mode
    U00IO = 0;           // Recei
    U00CK1=1;U00CK0=0;   // Selec
    U00RSS = 0;          // Value
    U00LG1=0;U00LG0=0;   // 8-bit
    U00PT2=0;U00PT1=0;U00PT0=0;
    U00STP = 0;      // 1 stop bi
    U00NEG = 0;      // Positive
    U00DIR = 0;      // LSB first

    // 9600 bps Baud Rate Settir
        UA00BRT = 0x09c1;
        UA00BRC = 0x04; //
    // 115,200 bps Baud Rate Set
    //  UA00BRT = 0x00cf;    // E
    //  UA00BRC = 0x02; // Baud

    Set_UART00_RX();
    //Set_UART00_TX();
}
```

UAn0BRT and UAn0BRC can be calculated by the following formulae.

UARTn0 Baud Rate Register (UAn0BRT)

$$UAn0BRT = \frac{\text{Base clock frequency (Hz)}}{\text{Baud rate (bps)} - 1}$$

UARTn0 Baud Rate Adjustment Register (UAn0BRC)

$$UAn0BRC = \frac{\text{Base clock frequency (Hz)} \ \%\ \text{Baud rate (bps)} \times 8}{\text{Baud rate (bps)}}$$

In Addition you can use table lists the count values for typical baud rates.

## lists the count values for typical baud rates

Table 11-7 Count Values for Typical Baud Rates (1/2)

| Base clock | Baud rate | UAn0BRT<br>UAn1BRT | UAn0BRC<br>UAn1BRC | Actual<br>baud rate |
|---|---|---|---|---|
| Approx. 24 MHz<br>(approx. 23.986176 MHz) | 1,200bps | 0x4E13 | 0x04 | 1199.99bps |
| | 2,400bps | 0x2709 | 0x02 | 2399.99bps |
| | 4,800bps | 0x1384 | 0x01 | 4799.99bps |
| | 9,600bps | 0x09C1 | 0x04 | 9600.23bps |
| | 19,200bps | 0x04E0 | 0x02 | 19200.46bps |
| | 38,400bps | 0x026F | 0x05 | 38400.92bps |
| | 57,600bps | 0x019F | 0x03 | 57607.14bps |
| | 115,200bps | 0x00CF | 0x02 | 115179.71bps |

# Set UART00 TX ,RX (uart0.c)

```
void Set_UART00_TX(void)
{
    Stop_UART00();
    U00IO = 0;          Th
    QSIU00 = 0;       // Cl
    ESIU00 = 0;       // Di
    //SU0CON -> Serial Co
    Start_UART00();     /
}
//-----------------------
void Set_UART00_RX(void)
{
    Stop_UART00();
    U00IO = 1;          Th
    QSIU00 = 0;       // Cl
    ESIU00 = 1;       // En

    //SU0CON -> Serial Co
    Start_UART00();     /
}
```

## 11.2.8 UARTn0 Mode Register (UAn0MOD)

UAn0MOD is a specific function register (SFR) to set the mode in UARTn0 full-duplex communiction mode and half-duplex communication mode.

| | |
|---|---|
| Address: | 0xF60C(UA00MODL/UA00CON), 0xF60D(UA00MODH),<br>0xF62C(UA10MODL/UA10MOD), 0xF62D(UA10MODH),<br>0xF64C(UA20MODL/UA20CON), 0xF64D(UA20MODH),<br>0xF66C(UA30MODL/UA30MOD), 0xF66D(UA30MODH),<br>0xF68C(UA40MODL/UA40CON), 0xF68D(UA40MODH),<br>0xF6AC(UA50MODL/UA50MOD), 0xF6AD(UA50MODH) |
| Access: | R/W |
| Access size: | 8/16bit |
| Initial value: | 0x0000 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | UAn0MOD | | | | | | | | | | | | | | | |
| Byte | UAn0MODH | | | | | | | | UAn0MODL | | | | | | | |
| Bit | Un0DIR | Un0NEG | Un0STP | Un0PT2 | Un0PT1 | Un0PT0 | Un0LG1 | Un0LG0 | Un0RSS | – | – | – | – | Un0CK1 | Un0CK0 | Un0IO |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | |
|---|---|
| Un0IO | This bit is used to choose the transmission mode or reception mode in UARTn0 full-duplex and half-duplex mode. When the full-duplex communication mode is chosen, this bit is fixed to “1” and the UART performs as the reception mode.<br>0: Transmission mode (Initial value)<br>1: Reception mode |

## Set UART00 TX ,RX (uart0.c)

```
void Set_UART00_TX(void)
{
    Stop_UART00();
    U00IO = 0;          // Th
    QSIU00 = 0;         Cl
    ESIU00 = 0;         // Di
    //SU0CON -> Serial Co
    Start_UART00();     /
}
//-----------------------
void Set_UART00_RX(void)
{
    Stop_UART00();
    U00IO = 1;          // Th
    QSIU00 = 0;         Cl
    ESIU00 = 1;         // En

    //SU0CON -> Serial Co
    Start_UART00();     /
}
```

### 5.2.7 Interrupt Request Register 23 (IRQ23)

Address: 0xF02A(IRQ2/IRQ23), 0xF02B(IRQ3)
Access: R/W
Access size: 8/16bit
Initial value: 0x0000

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | IRQ23 | | | | | | | | | | | | | | | |
| Byte | IRQ3 | | | | | | | | IRQ2 | | | | | | | |
| Bit | QTM1 | QTM0 | QFTM1 | QFTM0 | QI2CM1 | QI2CM0 | - | QEXTX | - | QSAD | - | QSIU01 | QSIU00 | QMCS | QDMA | QCBU |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R | R/W | R | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

QSIU00 This bit controls to request the Serial Communication unit 00 interrupt (SIU00INT).
0: Not request the interrupt (initial value)
1: Request the interrupt

# Set UART00 TX ,RX (uart0.c)

```
void Set_UART00_TX(void)
{
    Stop_UART00();
    U00IO = 0;          // Th
    QSIU00 = 0;         // Cl
    ESIU00 = 0;         Di
    //SU0CON -> Serial Co
    Start_UART00();      /
}
//-----------------------
void Set_UART00_RX(void)
{
    Stop_UART00();
    U00IO = 1;          // Th
    QSIU00 = 0;         // Cl
    ESIU00 = 1;         En

    //SU0CON -> Serial Co
    Start_UART00();      /
}
```

### 5.2.3 Interrupt Enable Register 23 (IE23)

IE23 is a specific function register (SFR) to enable or disable the interrupt for each interrupt request.
The bits are unwriteable when the products do not have the peripheral circuits and they return "0" for reading.
After the interrupt is accepted, the master interrupt enable flag (MIE) of the CPU is reset to "0", however, the applicable each flag of IE01 is not reset and remains "1".

Address: 0xF022 (IE2/IE23), 0xF023(IE3)
Access: R/W
Access size: 8/16bit
Initial value: 0x0000

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | IE23 | | | | | | | | | | | | | | | |
| Byte | IE3 | | | | | | | | IE2 | | | | | | | |
| Bit | ETM1 | ETM0 | EFTM 1 | EFTM 0 | EI2C M1 | EI2C M0 | - | EEXT X | - | ESAD | - | ESIU 01 | ESIU 00 | EMC S | EDM A | ECBU |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W | R | R/W | R | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

ESIU00 This bit controls to enable or disable the Serial Communication unit 00 interrupt (SIU00INT).
0: Disable the interrupt (initial value)
1: Enable the interrupt

## Get Command (uart0.c)

```
void UART00_GetData(void)
{
    unsigned char i;

    i = SD0BUFL;    //Serial communicatior

    if(i=='<')
    {
        if(!Flag._RXD00)
            RXD00_Index = 0;
    }
    else if(i=='>')
    {
        if(RXD00_Index)
        {
            Flag._RXD00 = 1;
            Stop_UART00();
        }
    }
    else if(RXD00_Index < RXD00_BUF_NUM)
        UART00_RX_Buf[RXD00_Index++] = i;
}
```

Read Buffer

Check symbol '<'

Set Index Array of UART00_RX_Buf = 0

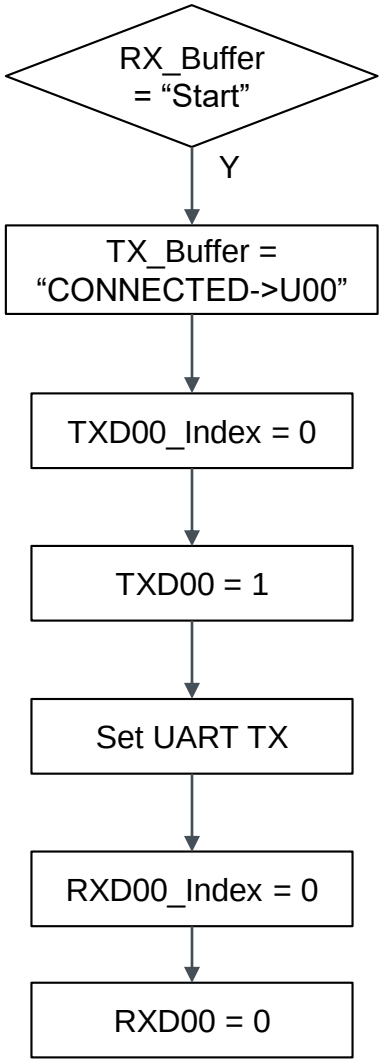
Check symbol '>'

Set Flag RXD00 = 1

Send Data from Buffer to Array UART00_RX_Buf

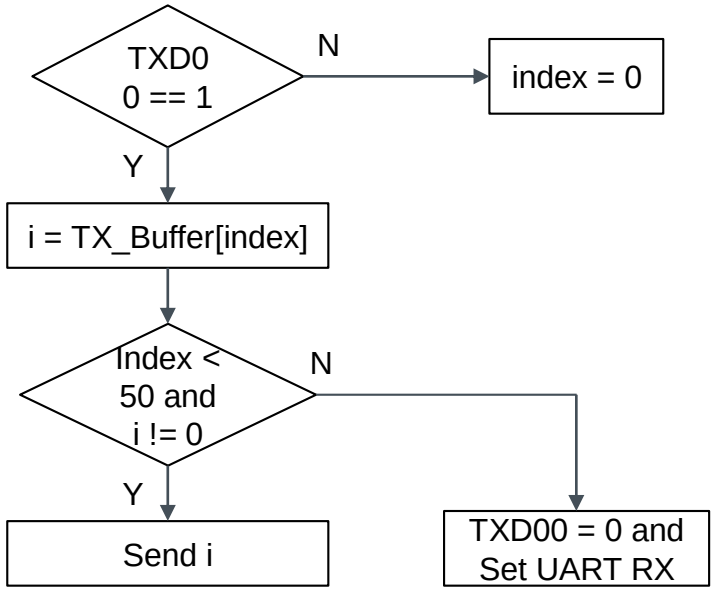
## Get Command (uart0.c)

```
void UART00_GetCommand(void)
{
    if(Flag._RXD00)
    {
        if((UART00_RX_Buf[0]=='S')&&(UART00_RX_Buf[1]=='t')&&(UART00_RX_Buf[2]=='a')
            &&(UART00_RX_Buf[3]=='r')&&(UART00_RX_Buf[4]=='t'))
        {
            UART00_TX_Buf[0] = 'C';
            UART00_TX_Buf[1] = 'o';
            UART00_TX_Buf[2] = 'n';
            UART00_TX_Buf[3] = 'n';
            UART00_TX_Buf[4] = 'e';
            UART00_TX_Buf[5] = 'c';
            UART00_TX_Buf[6] = 't';
            UART00_TX_Buf[7] = 'e';
            UART00_TX_Buf[8] = 'd';
            UART00_TX_Buf[9] = '-';
            UART00_TX_Buf[10] = '>';
            UART00_TX_Buf[11] = 'U';
            UART00_TX_Buf[12] = '0';
            UART00_TX_Buf[13] = '0';

            UART00_TX_Buf[14] = 0x0a;
            UART00_TX_Buf[15] = 0x0d;
            UART00_TX_Buf[16] = 0;

            TXD00_Index = 0;
            Flag._TXD00 = 1;
            Set_UART00_TX();
        }

        RXD00_Index = 0;
        Flag._RXD00 = 0;
    }
}
```



## UART00 Send Data (uart0.c)

```
void UART00_SendData(void)
{
    unsigned int i=0;

    if(Flag._TXD00)
    {
        i = UART00_TX_Buf[TXD00_Index++];

        if((TXD00_Index < TXD00_BUF_NUM)&&(i!=0))
            UART00_Send(i);
        else
        {
            Flag._TXD00 = 0;
            Set_UART00_RX();
        }
    }
    else
        TXD00_Index = 0;
}
```

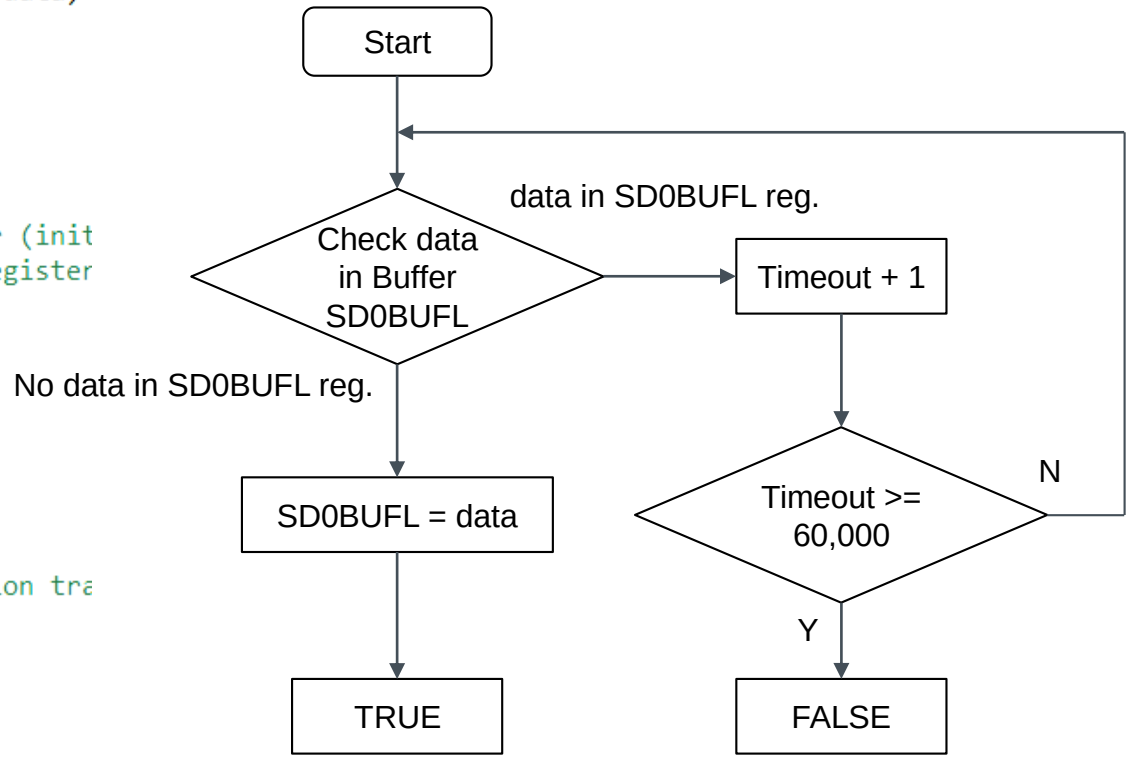




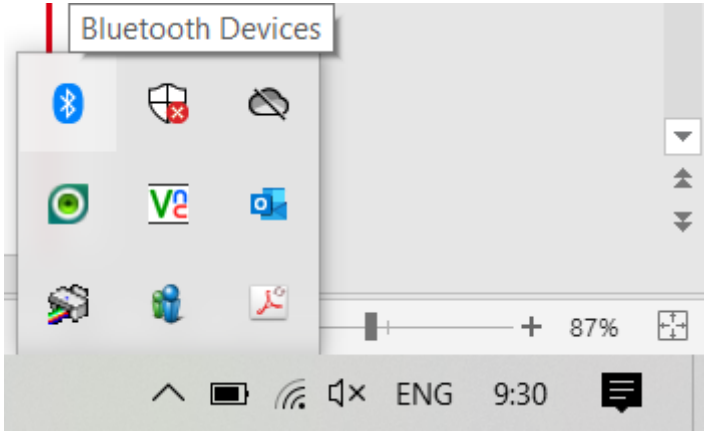

Pichet Sriphomma pichet@es.co.th

## UART00 Send (uart0.c)

```
unsigned char UART00_Send(unsigned char data)
{
    unsigned int TimeOut;

    TimeOut=0;
    /*
    * U00FUL
    * 0: No data in the SD0BUFL register (init
    * 1: There is data in the SD0BUFL register
    */
    while(U00FUL)
    {
        TimeOut++;
        if(TimeOut >= 60,0000)
            return FALSE;
    }

    SD0BUFL = data; //Serial communication tra
    return TRUE;
}
```




Note : Un0FUL This bit is used to indicate the state of the transmit/receive buffer.

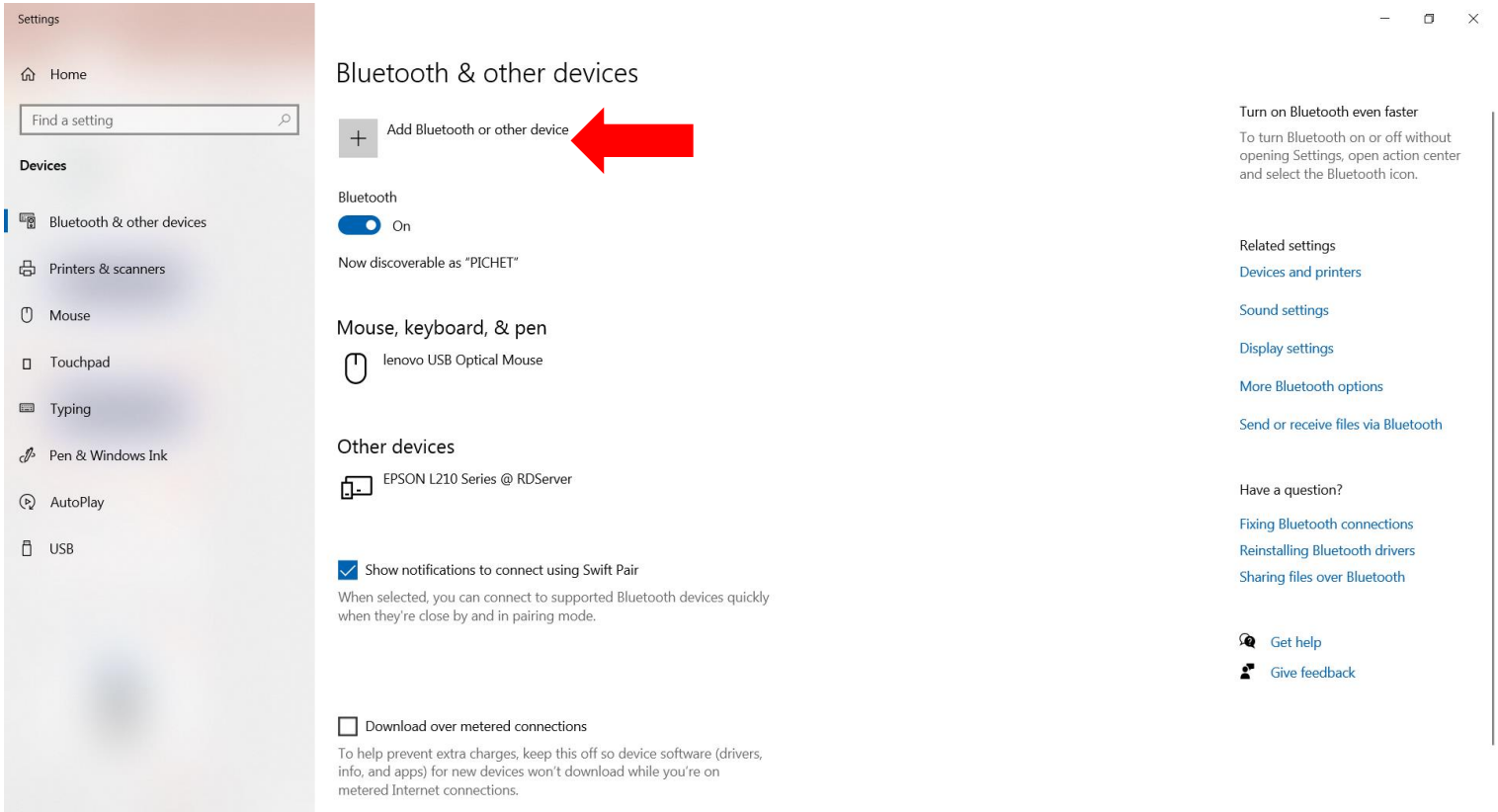



Before launching the program S-TERM.
Must connect device.
Open Bluetooth & other devices



Pichet Sriphomma pichet@es.co.th

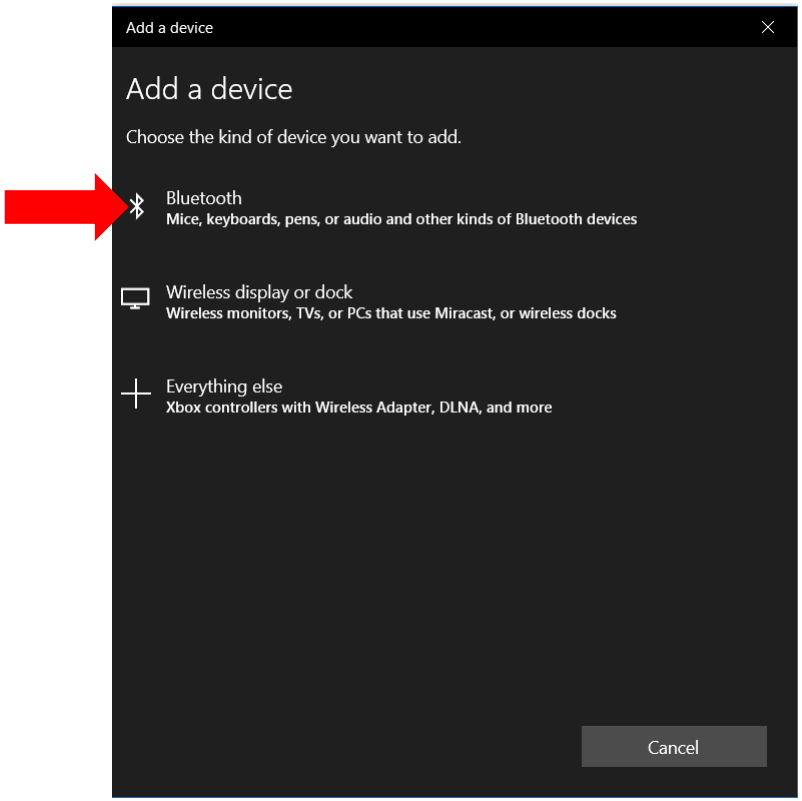
# Click Add Bluetooth or other device

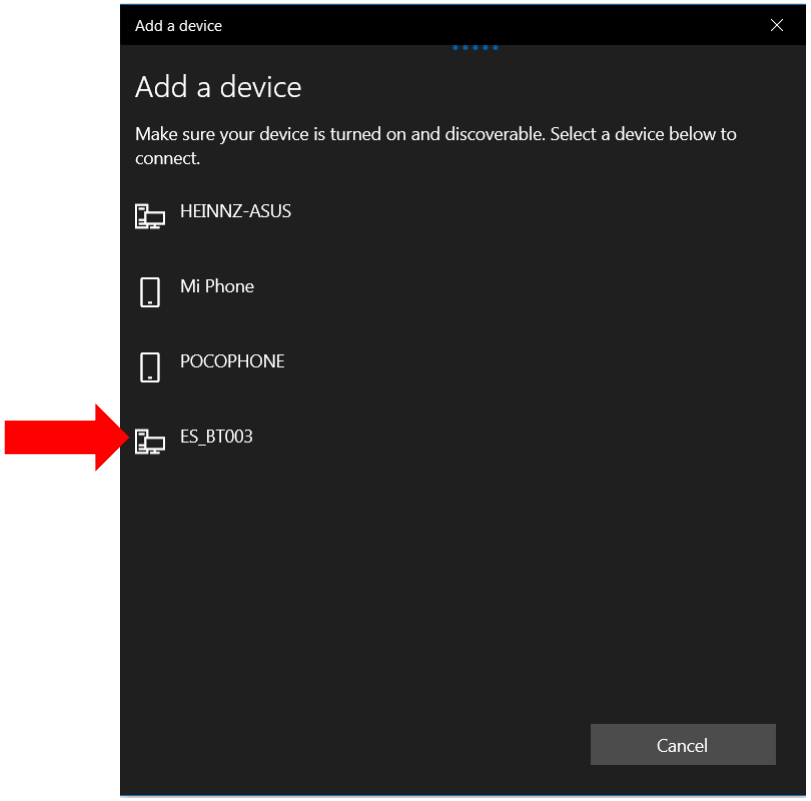





Pichet Sriphomma pichet@es.co.th

# Click Bluetooth and select device

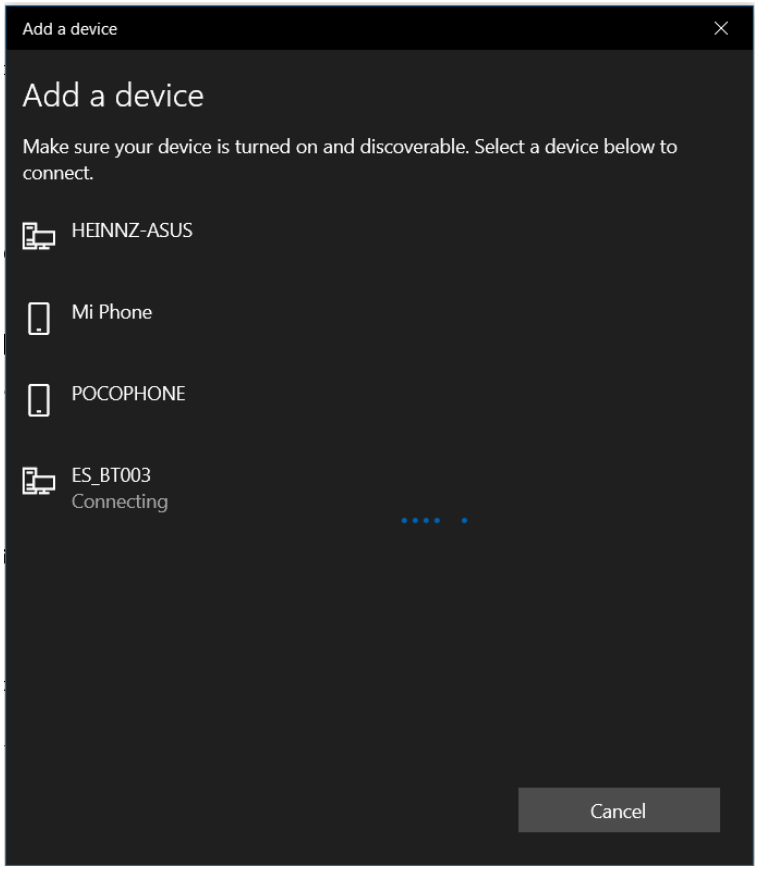


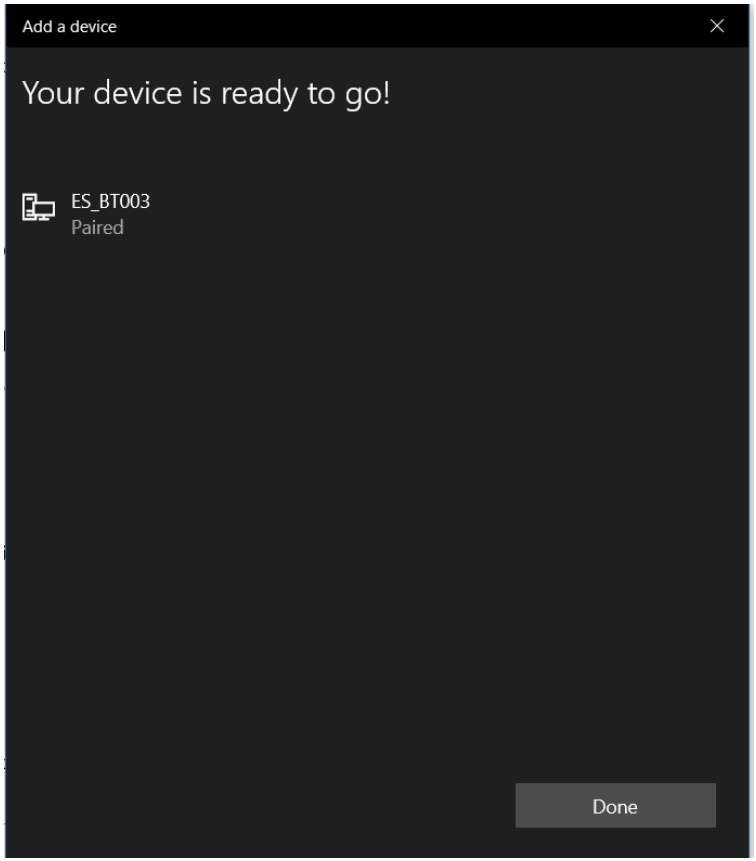






Pichet Sriphomma pichet@es.co.th

Wait for connecting..

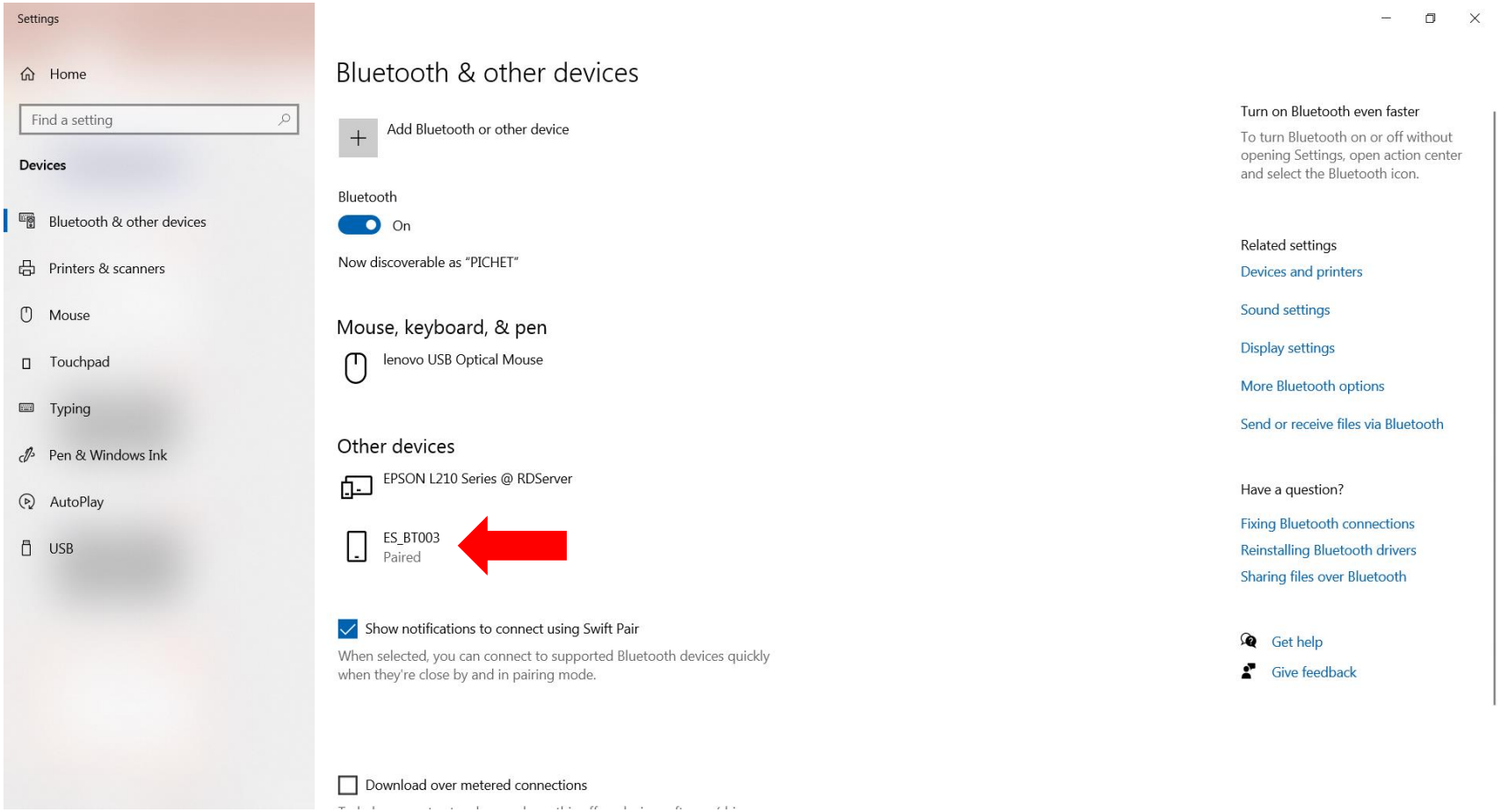





Pichet Sriphomma pichet@es.co.th

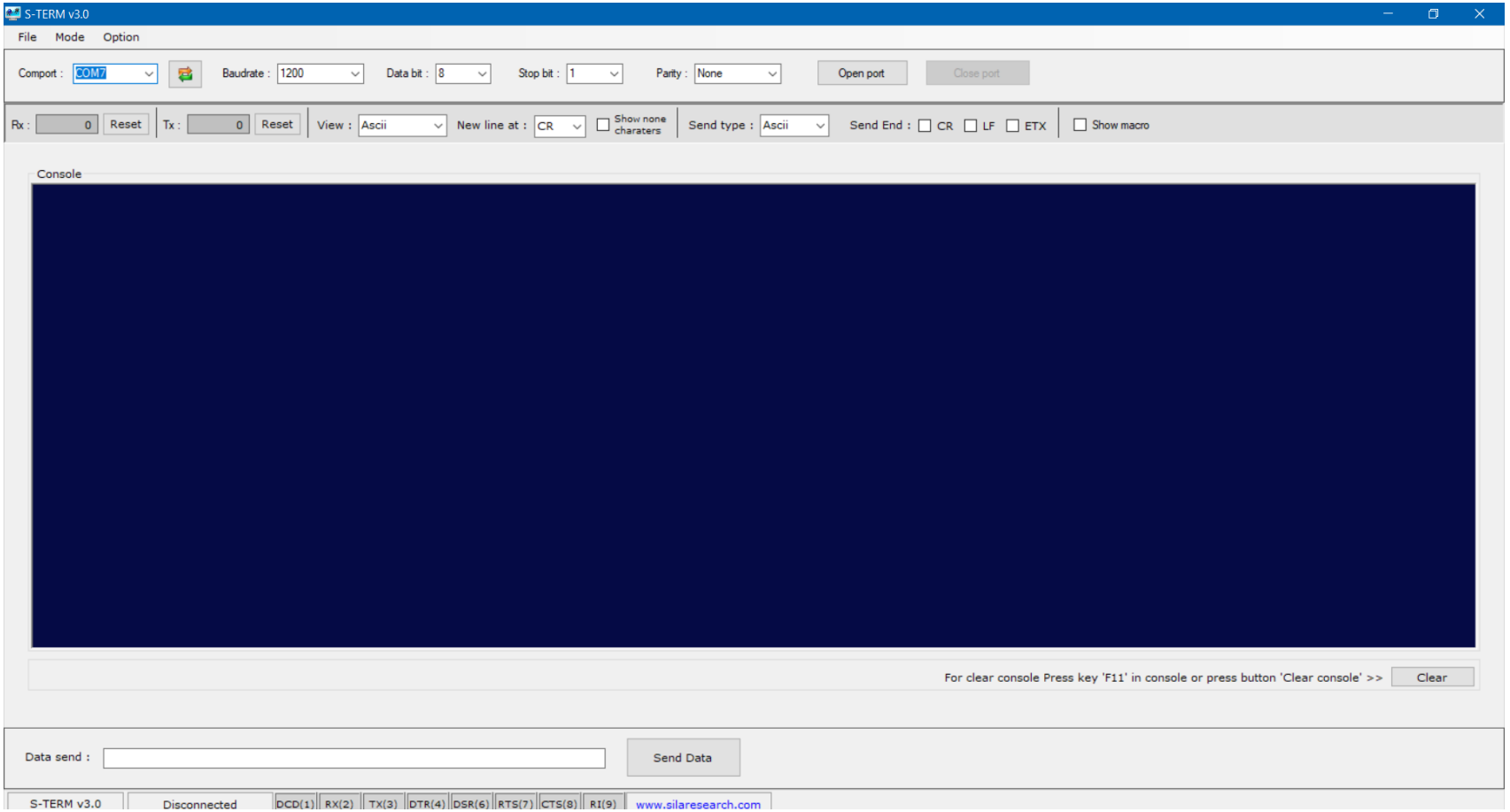
# Show device connected






Pichet Sriphomma pichet@es.co.th

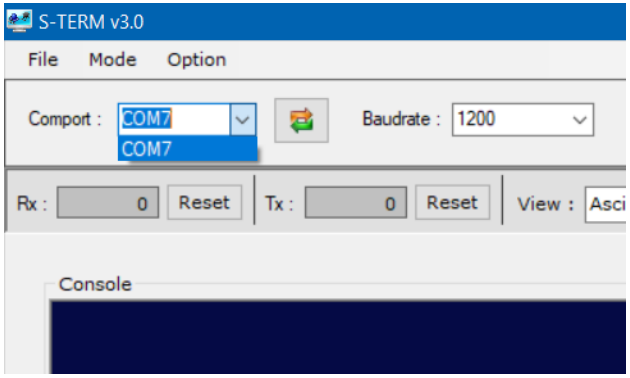
## Open Program S-TERM



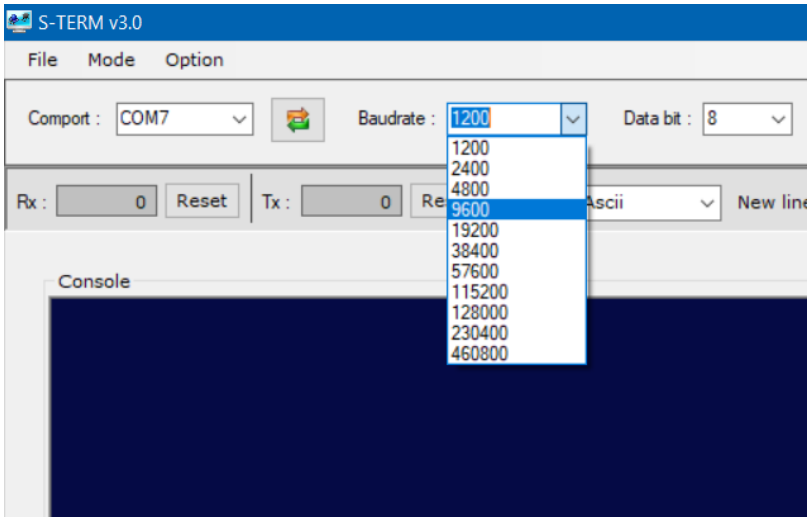



Pichet Sriphomma pichet@es.co.th

# S-TERM




Select Comport



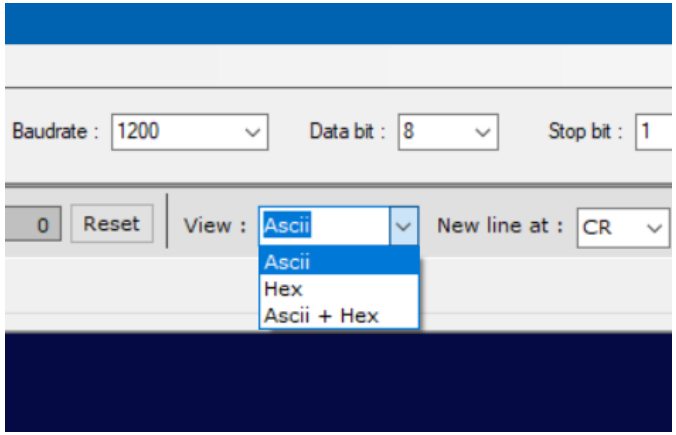

Select Baudrate.

This workshop select 9600.



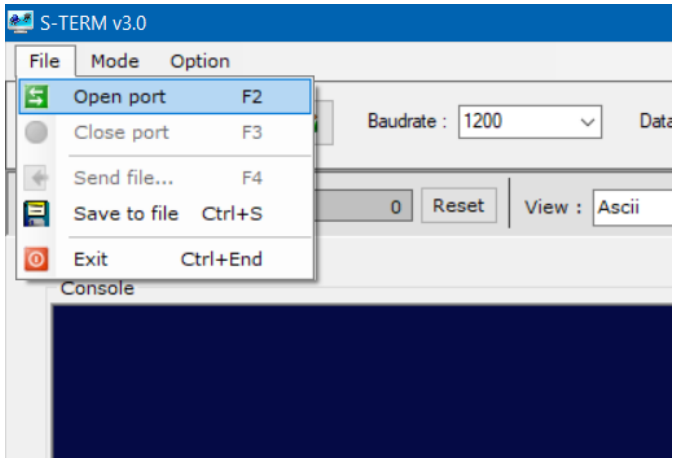
Pichet Sriphomma pichet@es.co.th

# S-TERM




Select view type.

This workshop select Ascii.

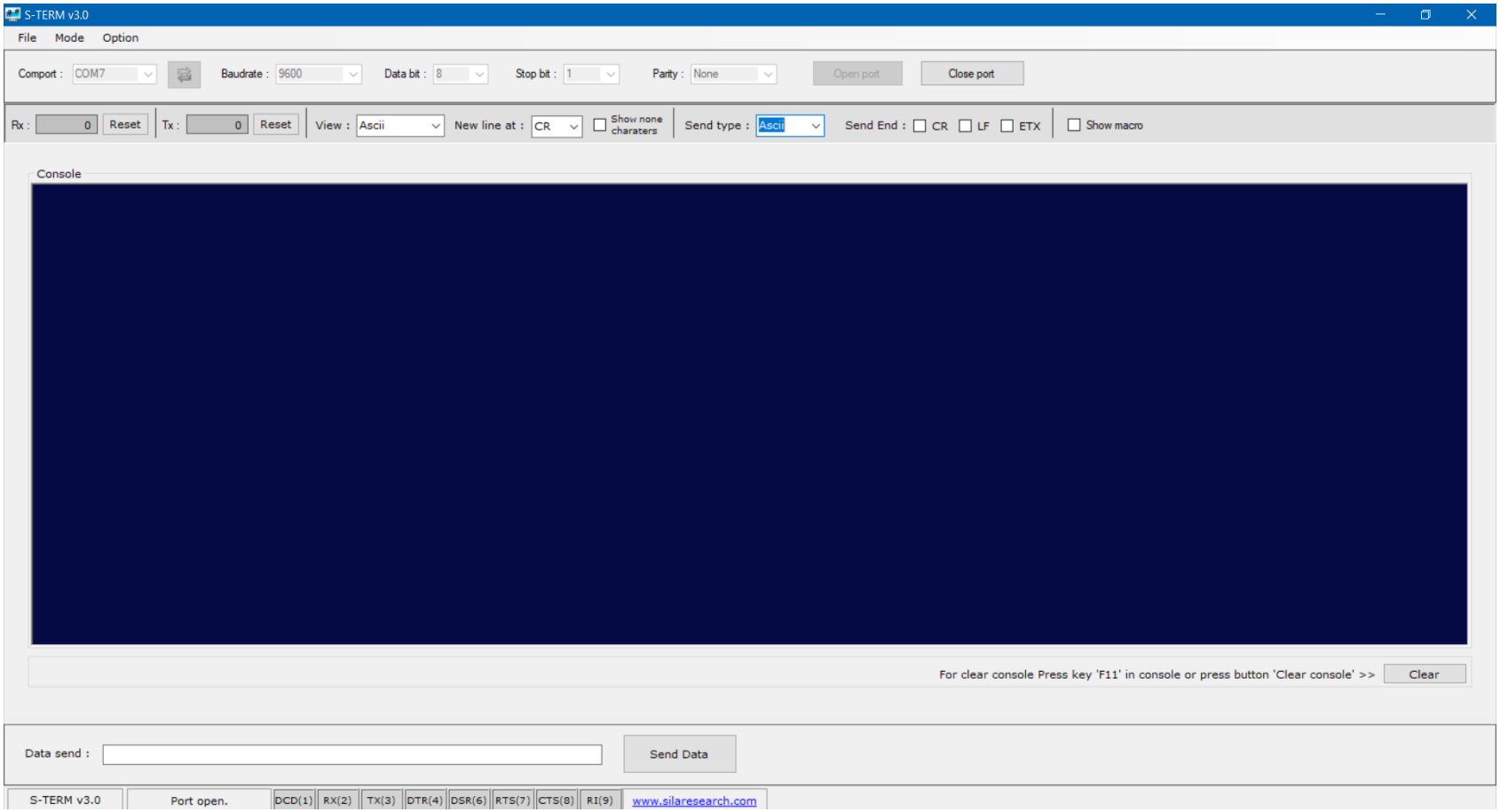



Click File and select Open port.



Pichet Sriphomma pichet@es.co.th

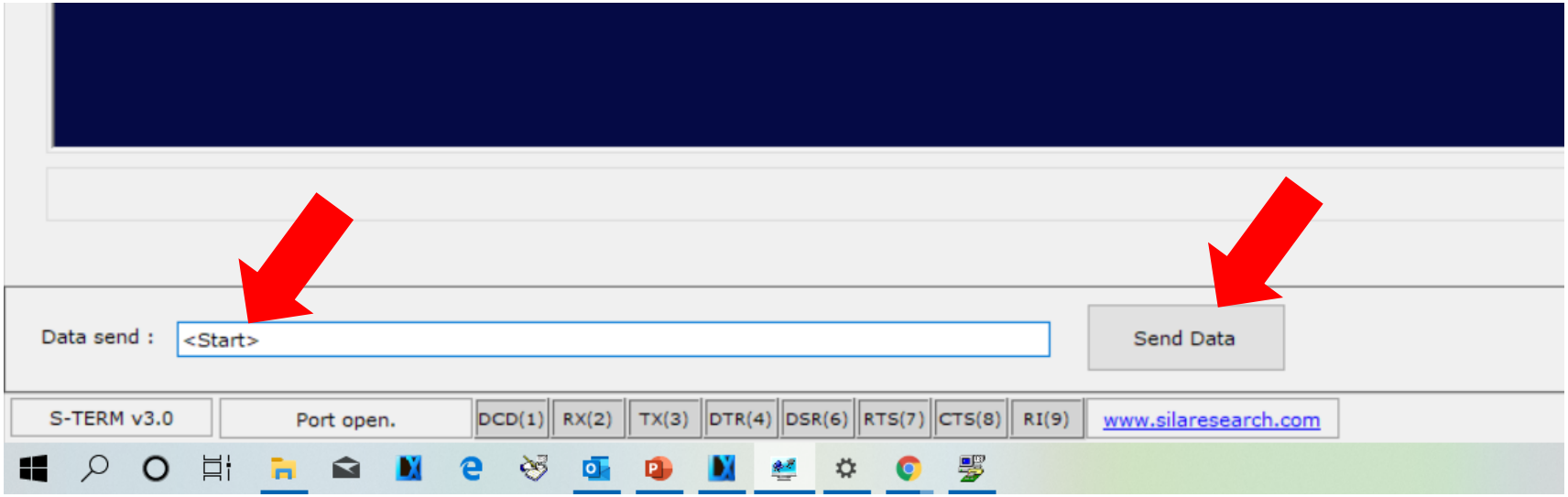
# S-TERM






Pichet Sriphomma pichet@es.co.th

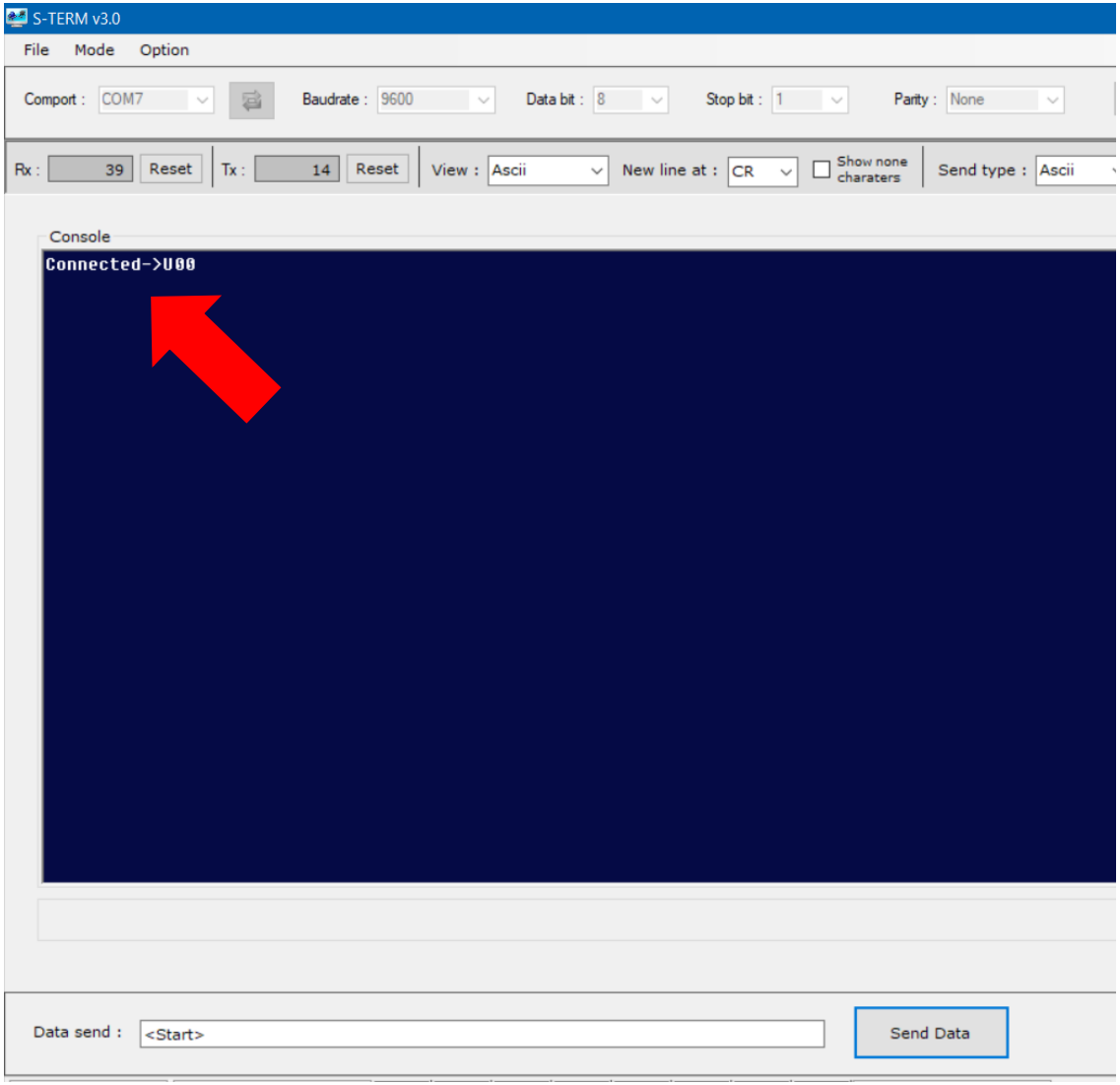
# S-TERM




Print message “<Start>”. After that click Send Data.

Pichet Sriphomma pichet@es.co.th

# S-TERM




After Send message “<Start>” to MCU Board.

After that MCU Board return Message “Connected->U00”



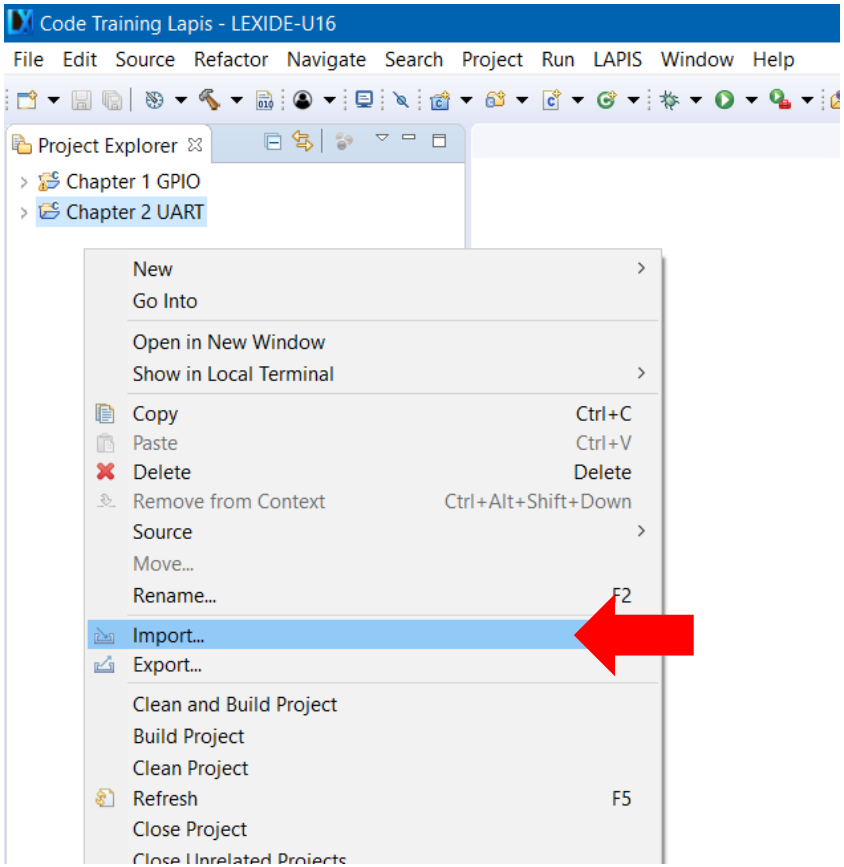
Pichet Sriphomma pichet@es.co.th

# 3. ADC



Pichet Sriphomma pichet@es.co.th

## Import Project

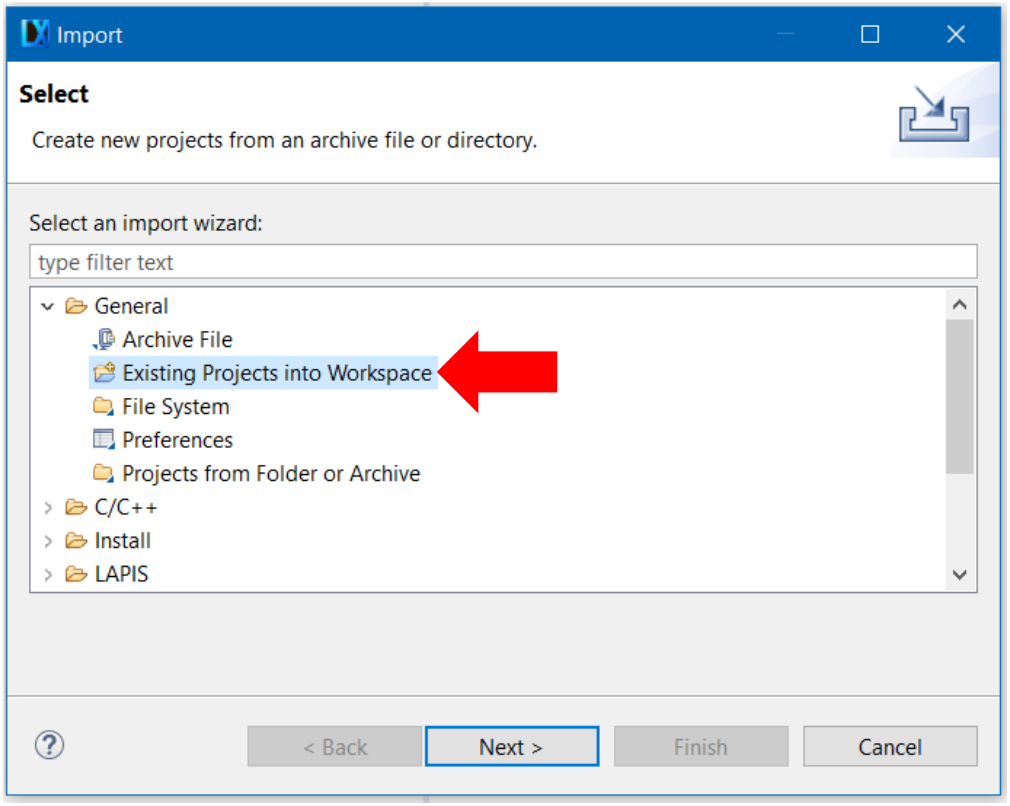



Right-click on project Explorer and select Import.



Pichet Sriphomma pichet@es.co.th

## Import Project

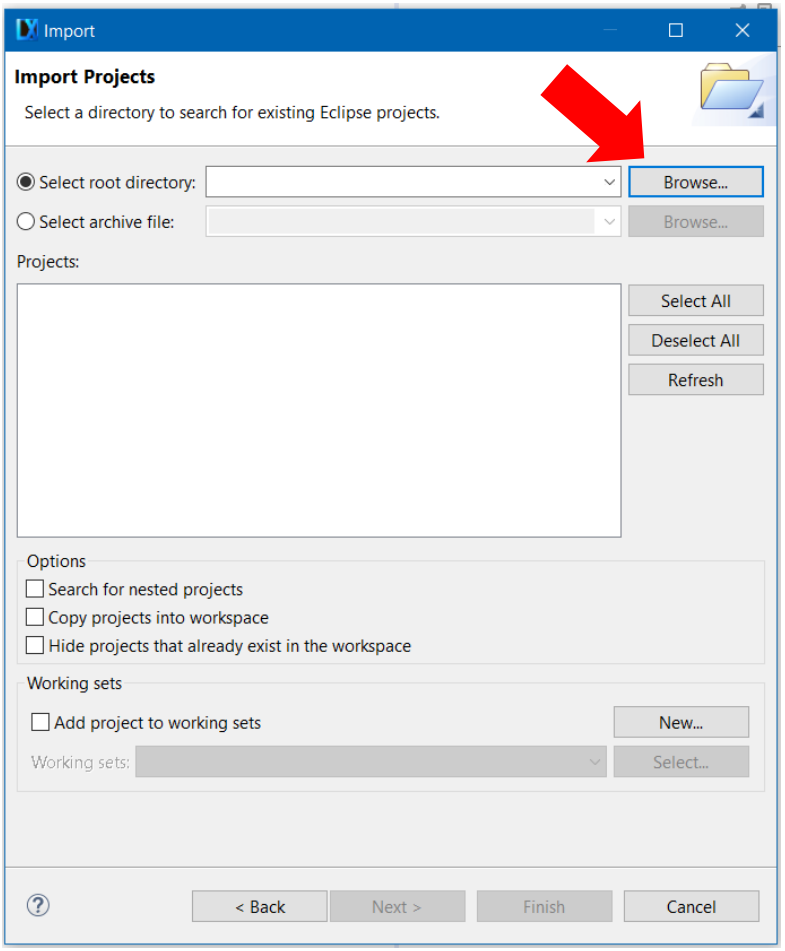



Select General and choose Existing Projects into Workspace. Click Next.

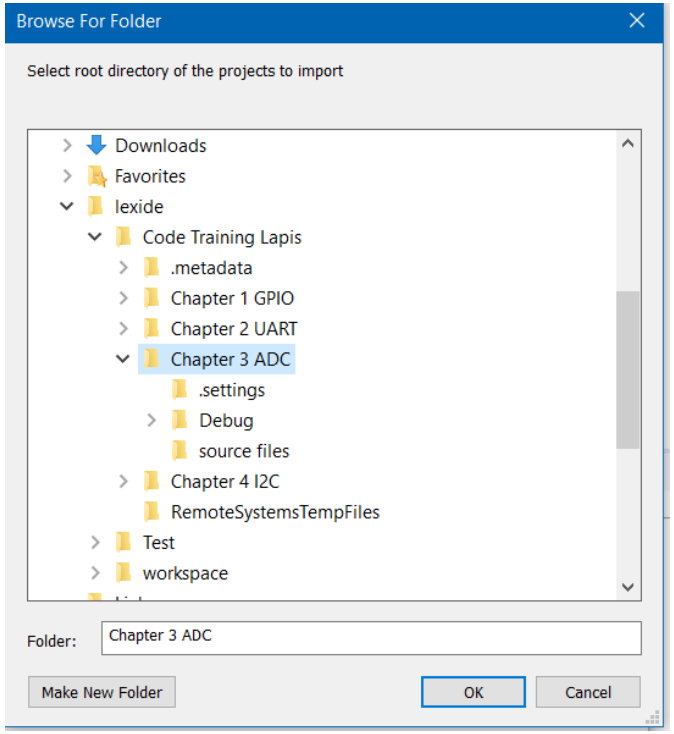


Pichet Sriphomma pichet@es.co.th

## Import Project



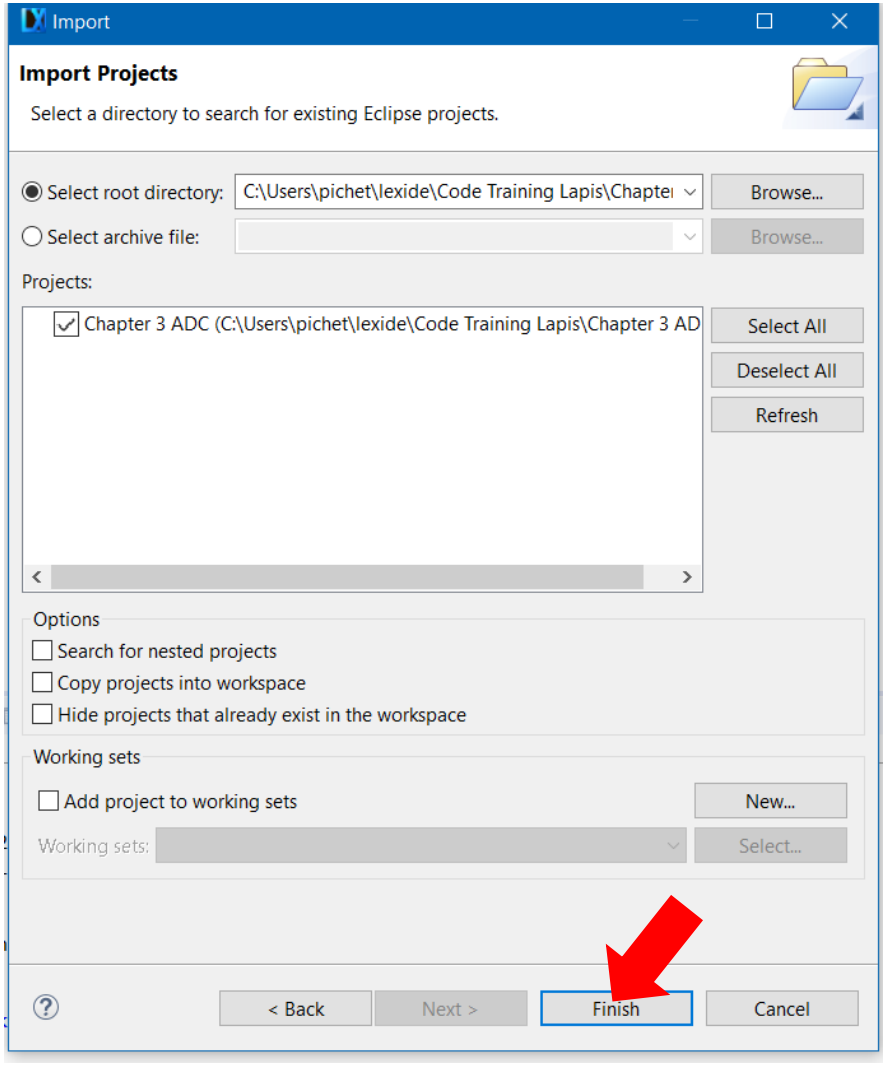

LEXIDE up the new window.
Click Browse.. at Select root directory.
Choose “Chapter 3 ADC” in Folder window.






Pichet Sriphomma pichet@es.co.th

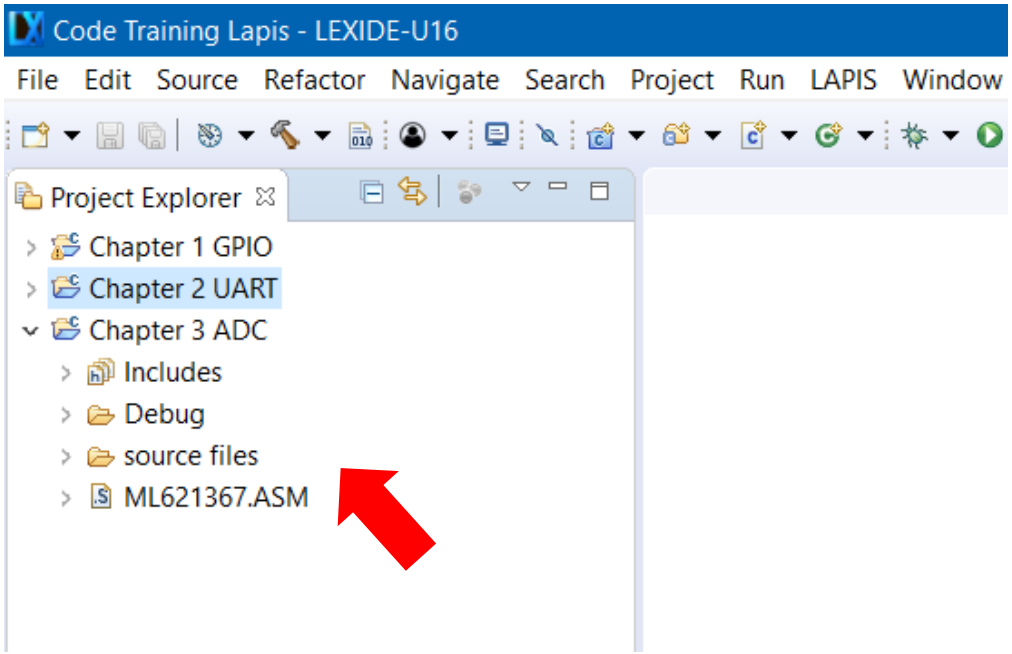
## Import Project




After choosing Project Click Finish.

Pichet Sriphomma pichet@es.co.th

## Import Project

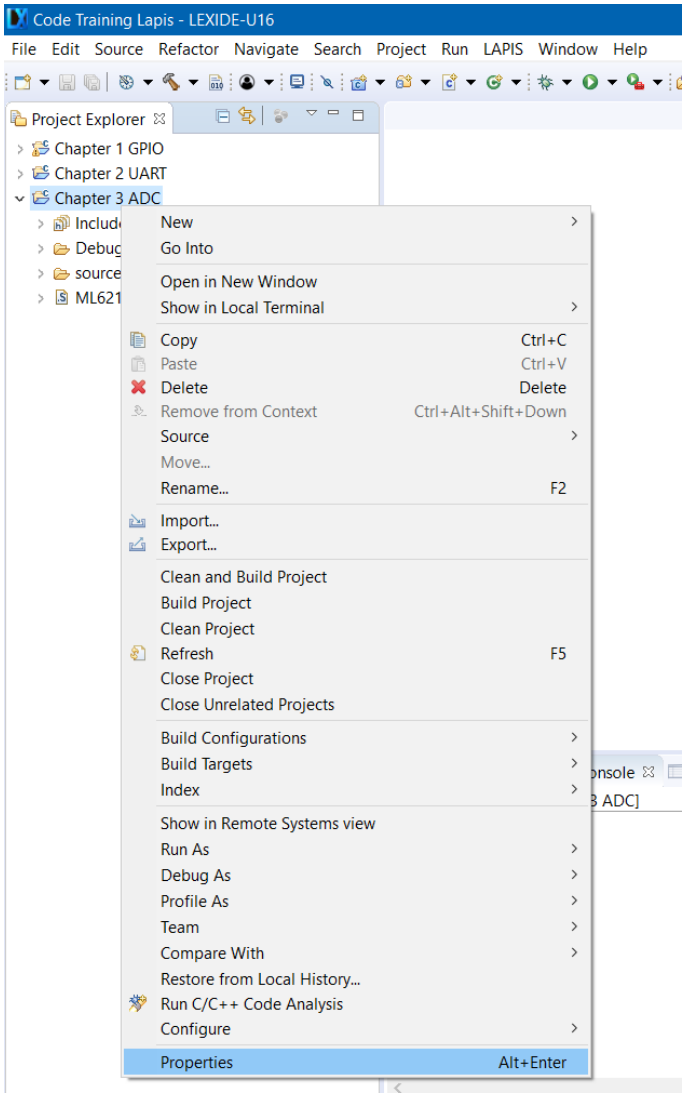



Then appear the project on Project Explorer.



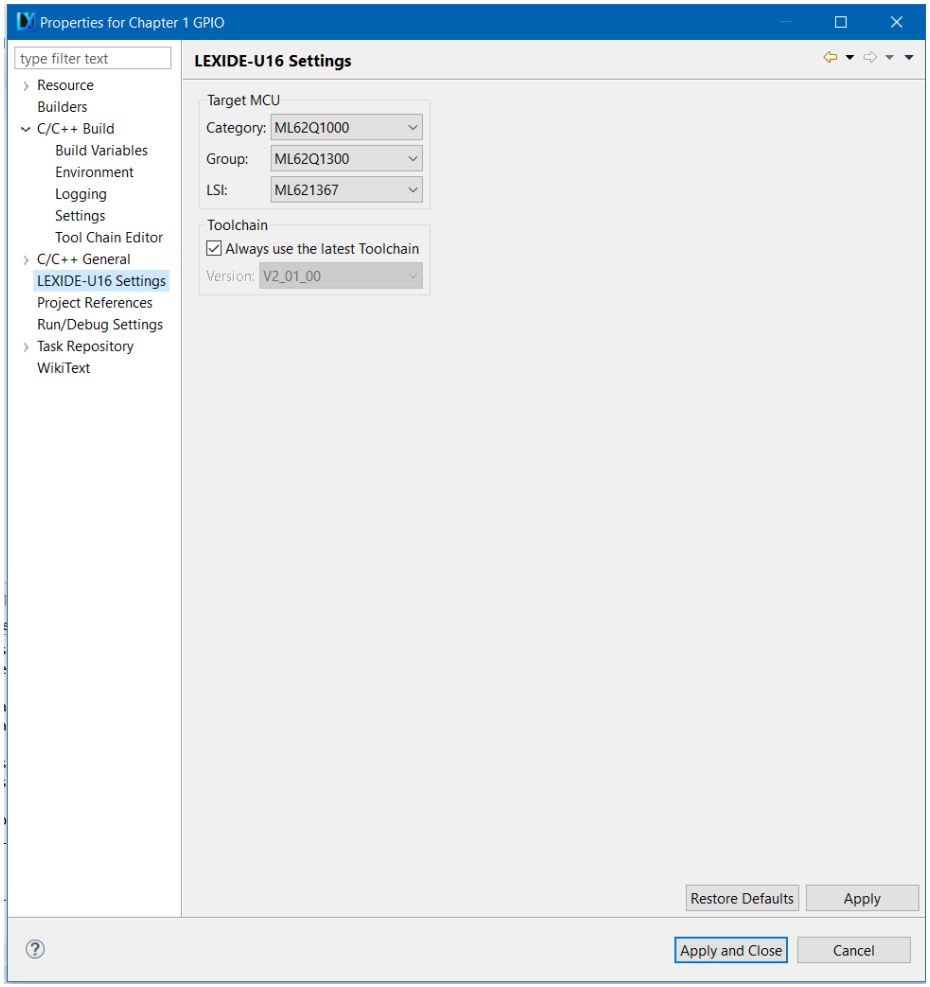
Pichet Sriphomma pichet@es.co.th

## Check Device




Right-click on a project folder and select [Properties] .

## Select Device

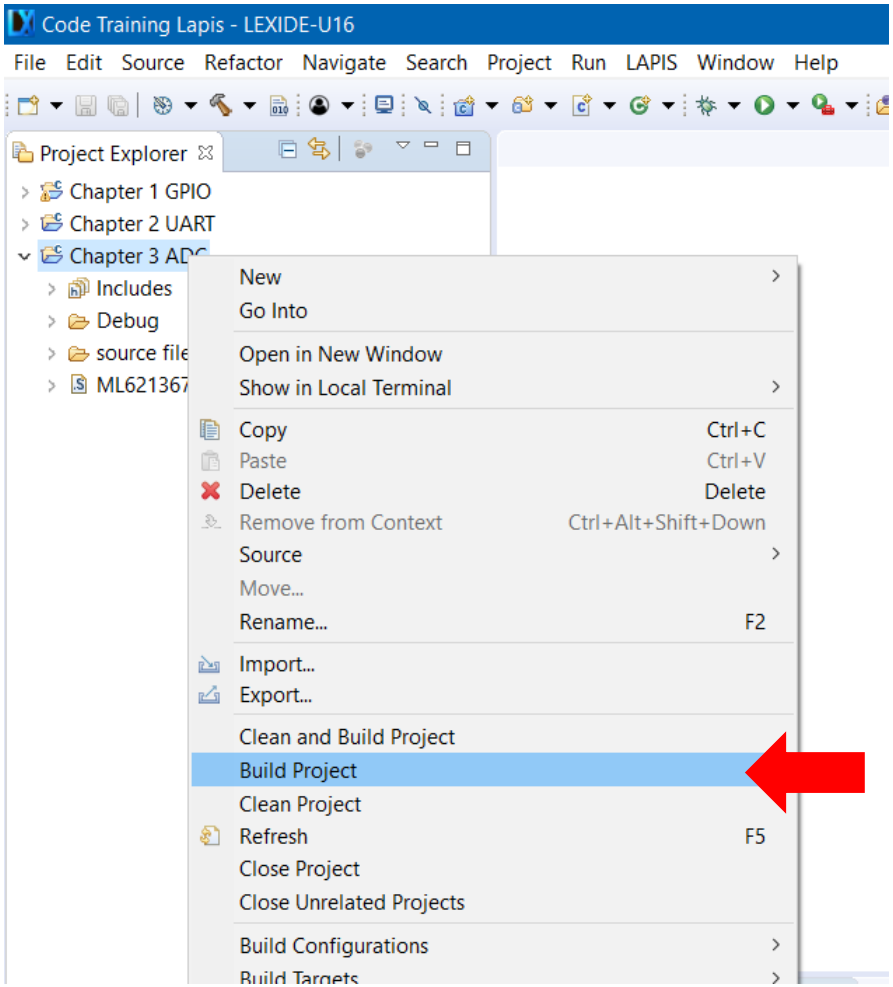



Choose LEXIDE-U16 Settings



Pichet Sriphomma pichet@es.co.th

## Build Project

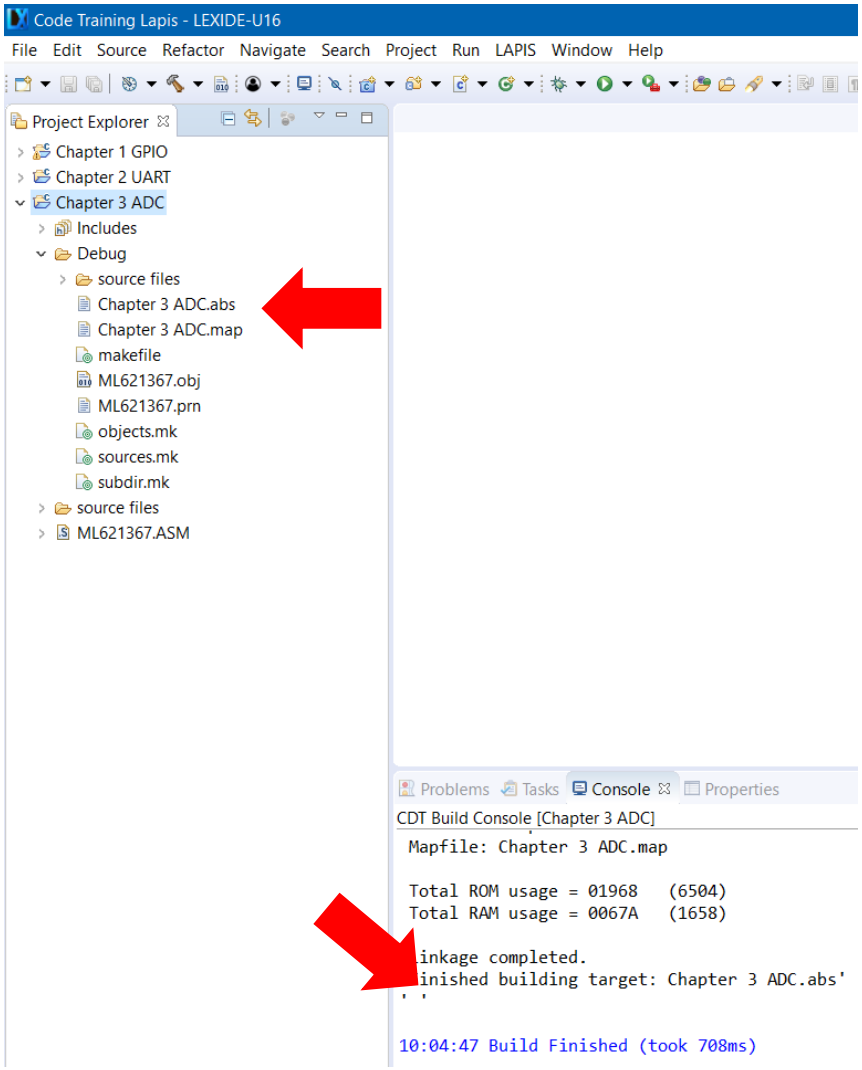



Right-click on a project folder and select [Build Project] to start the build process.



Pichet Sriphomma pichet@es.co.th

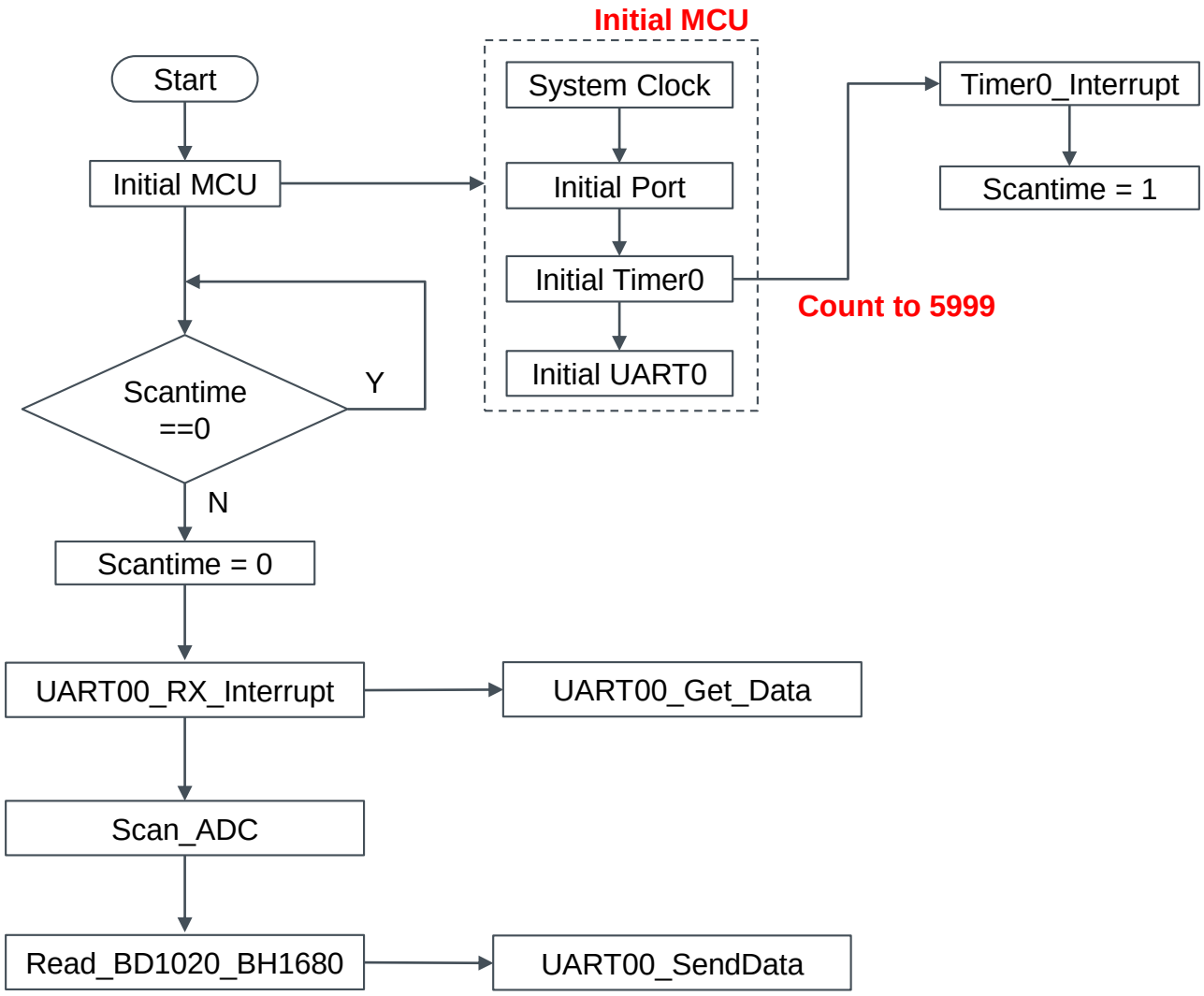
## Build Project




When the build succeeds , an ABS file is generated.

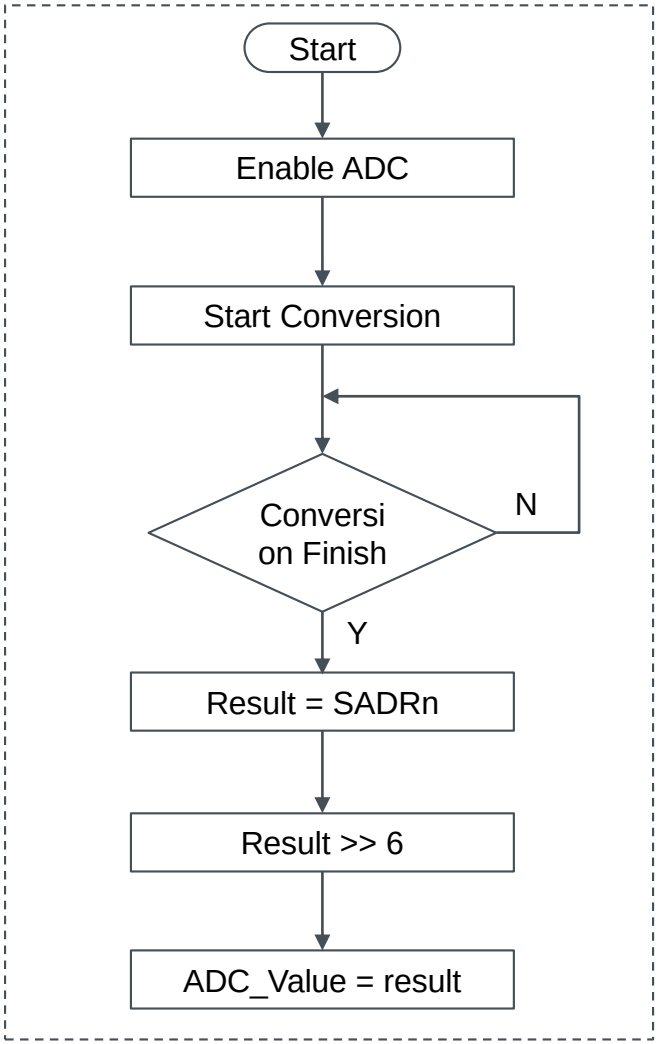


Pichet Sriphomma pichet@es.co.th

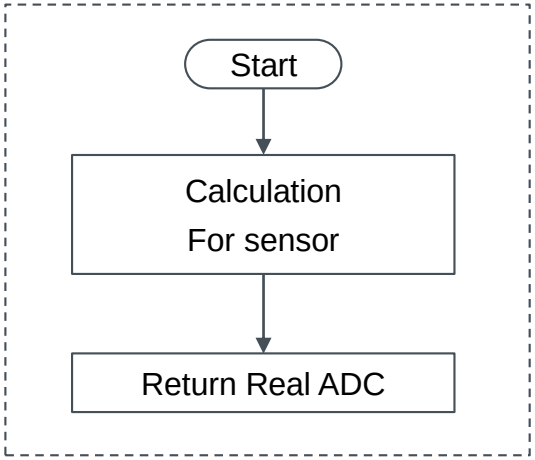



Pichet Sriphomma pichet@es.co.th

**Scan_ADC**



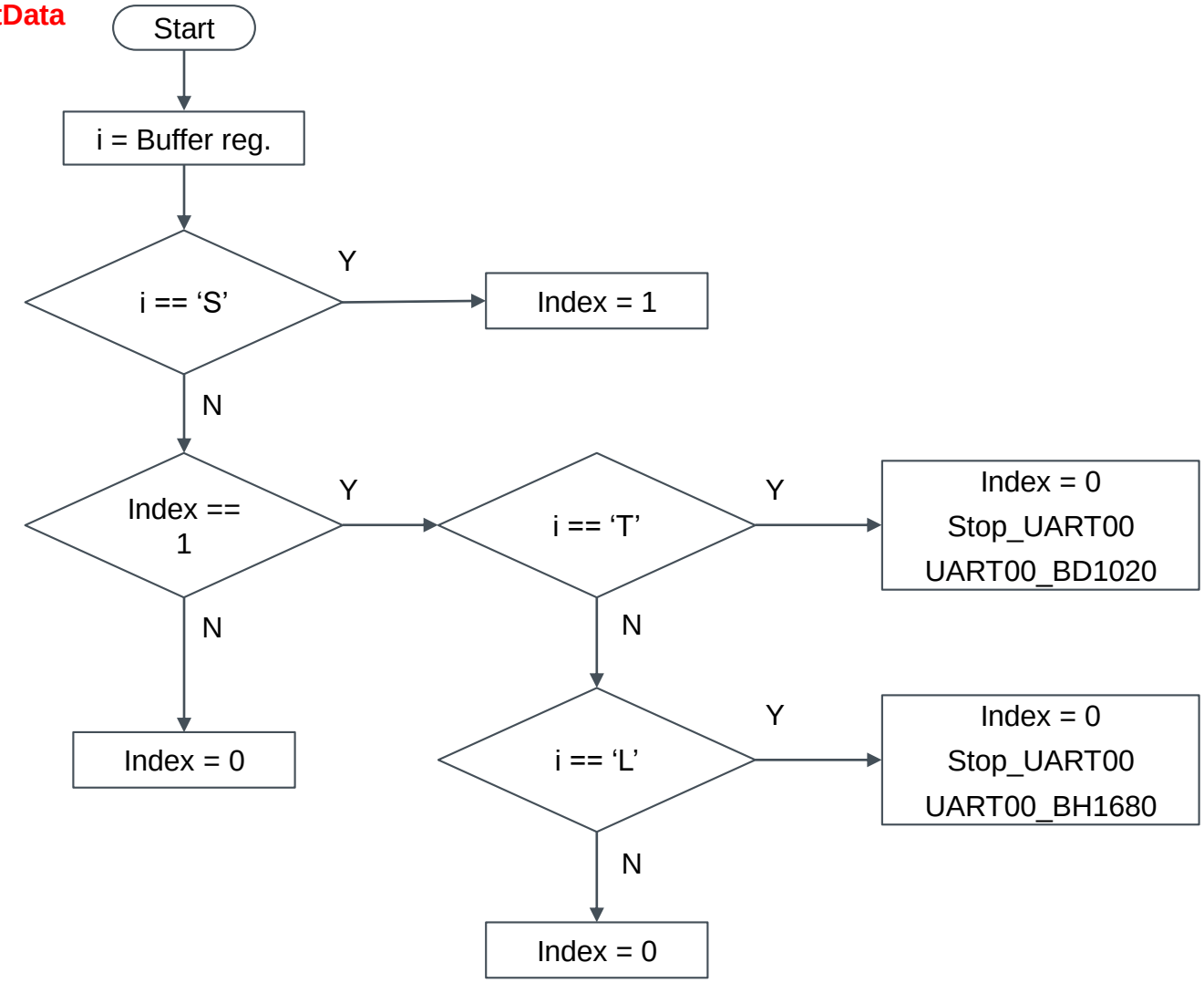

**Read BD1020 ,BH1680**






Pichet Sriphomma pichet@es.co.th

**UART00_GetData**




Pichet Sriphomma pichet@es.co.th

## General Description

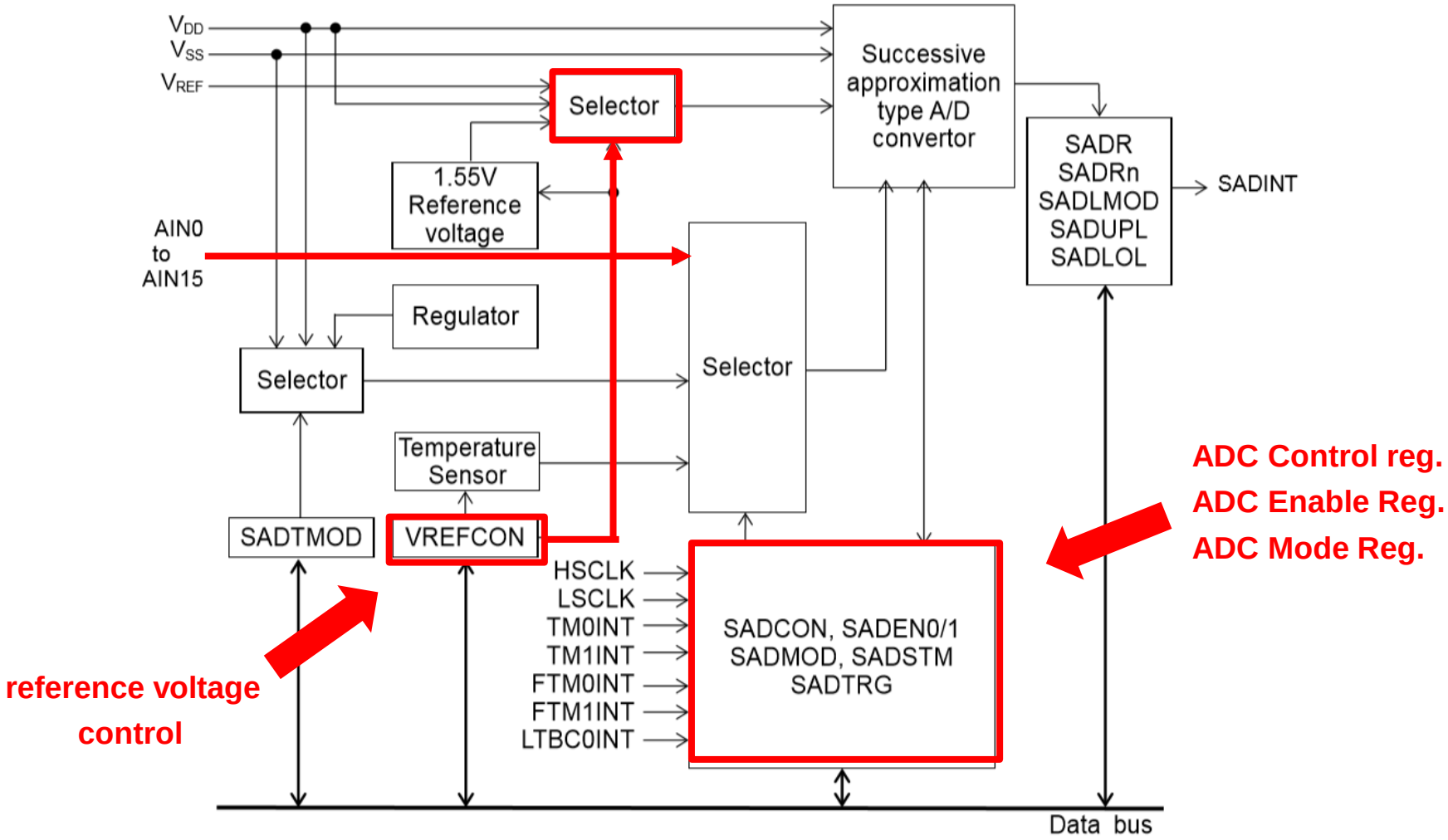
| Channel no. | ML62Q1300 group | | | | ML62Q1500 / ML62Q1700 group | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 16pin product | 20pin product | 24pin product | 32pin product | 48pin product | 52pin product | 64pin product | 80pin product | 100pin product |
| 0 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| 1 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| 2 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| 3 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| 4 | – | ● | ● | ● | ● | ● | ● | ● | ● |
| 5 | – | ● | ● | ● | ● | ● | ● | ● | ● |
| 6 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| 7 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| 8 | – | – | – | – | ● | ● | ● | ● | ● |
| 9 | – | – | – | – | ● | ● | ● | ● | ● |
| 10 | – | – | – | – | ● | ● | ● | ● | ● |
| 11 | – | – | – | – | ● | ● | ● | ● | ● |
| 12 | – | – | – | – | – | – | – | ● | ● |
| 13 | – | – | – | – | – | – | – | ● | ● |
| 14 | – | – | – | – | – | – | – | ● | ● |
| 15 | – | – | – | – | – | – | – | ● | ● |

●: Available - : Unavailable



Pichet Sriphomma pichet@es.co.th

## General Description



## General Description

| Channel no. | Pin name | Shared port | | Setting Register | Setting value | ML62Q1300 group | | | | ML62Q1500 ML62Q1700 group | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | 16pin product | 20pin product | 24pin product | 32pin product | 48pin product | 52pin product | 64pin product | 80pin product | 100pin product |
| 0 | AIN0 | P17 | 8th Func. | P1MOD7 | 0111_0000 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| 1 | AIN1 | P20 | 8th Func. | P2MOD0 | 0111_0000 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| 2 | AIN2 | P21 | 8th Func. | P2MOD1 | 0111_0000 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| 3 | AIN3 | P22 | 8th Func. | P2MOD2 | 0111_0000 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| 4 | AIN4 | P24 | 8th Func. | P2MOD4 | 0111_0000 | – | ● | ● | ● | ● | ● | ● | ● | ● |
| 5 | AIN5 | P25 | 8th Func. | P2MOD5 | 0111_0000 | – | ● | ● | ● | ● | ● | ● | ● | ● |
| 6 | AIN6 | P26 | 8th Func. | P2MOD6 | 0111_0000 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| 7 | AIN7 | P27 | 8th Func. | P2MOD7 | 0111_0000 | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| 8 | AIN8 | P65 | 8th Func. | P6MOD5 | 0111_0000 | – | – | – | – | ● | ● | ● | ● | ● |
| 9 | AIN9 | P66 | 8th Func. | P6MOD6 | 0111_0000 | – | – | – | – | ● | ● | ● | ● | ● |
| 10 | AIN10 | P43 | 8th Func. | P4MOD3 | 0111_0000 | – | – | – | – | ● | ● | ● | ● | ● |

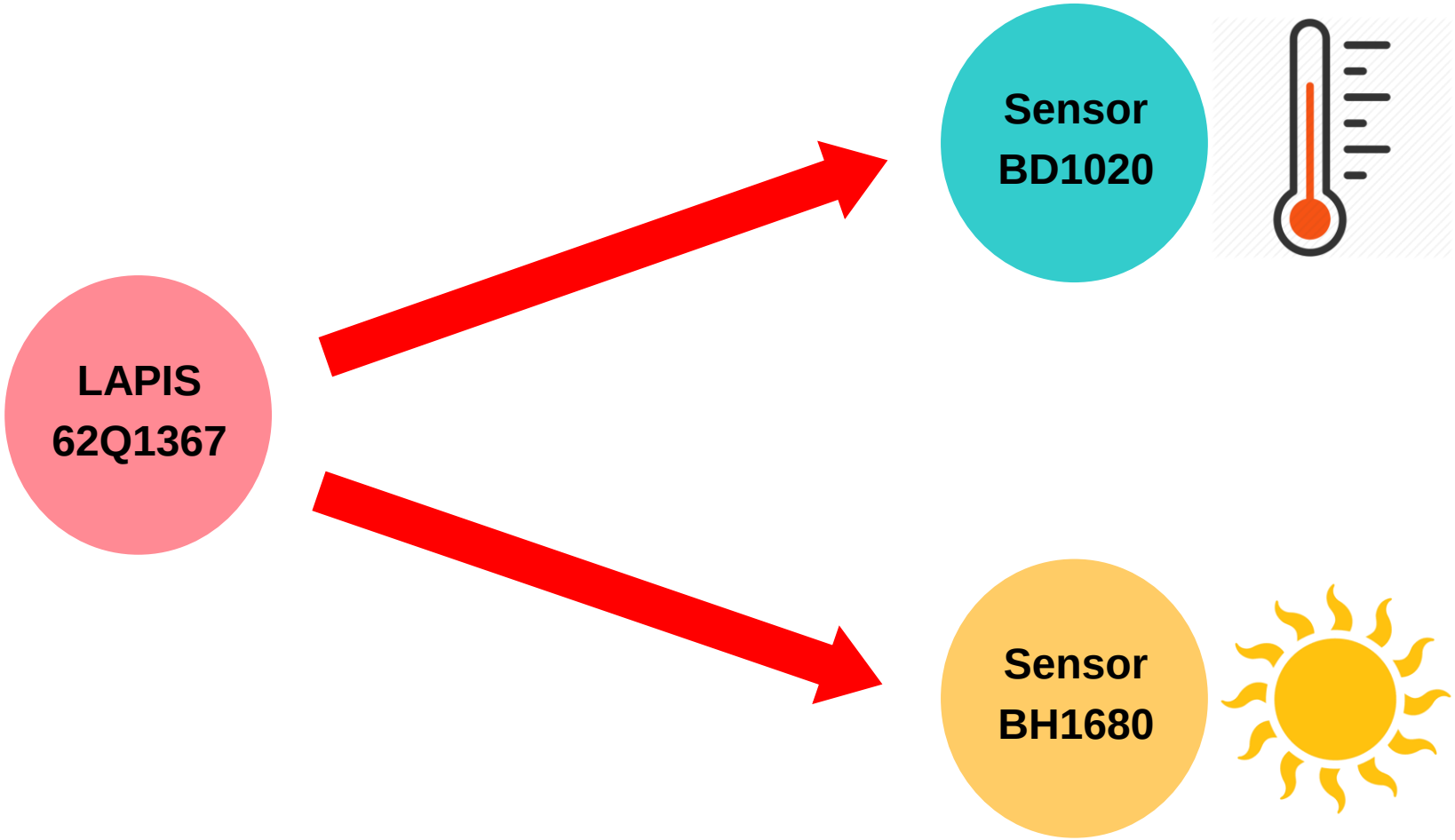


Pichet Sriphomma pichet@es.co.th

## SA-ADC Result Register n

| Symbol name | Channel |
|---|---|
| SADR0 | The conversion result of channel 0 (AIN0) |
| SADR1 | The conversion result of channel 1 (AIN1) |
| SADR2 | The conversion result of channel 2 (AIN2) |
| SADR3 | The conversion result of channel 3 (AIN3) |
| SADR4 | The conversion result of channel 4 (AIN4) |
| SADR5 | The conversion result of channel 5 (AIN5) |
| SADR6 | The conversion result of channel 6 (AIN6) |
| SADR7 | The conversion result of channel 7 (AIN7) |

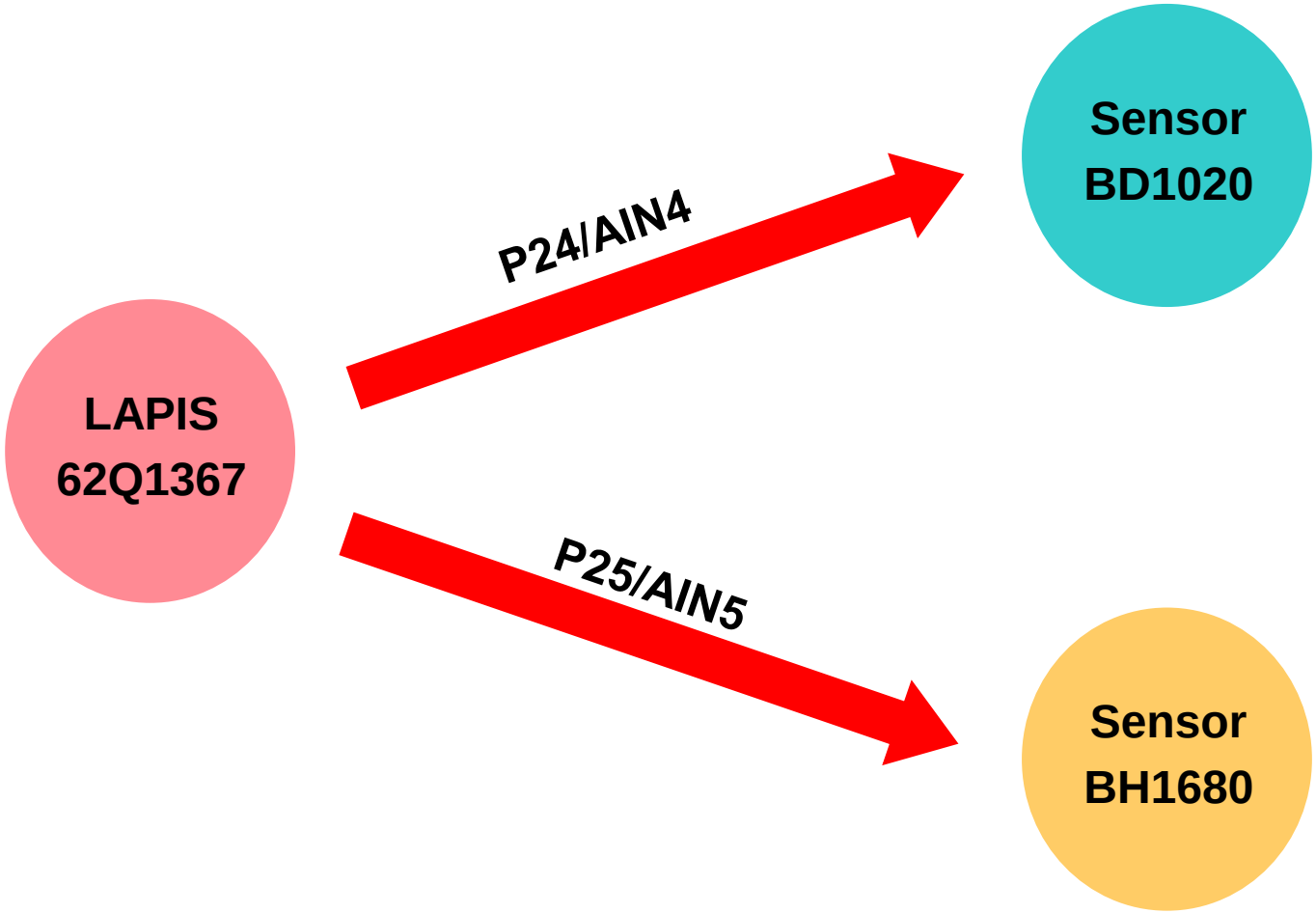
## Concept






Pichet Sriphomma pichet@es.co.th

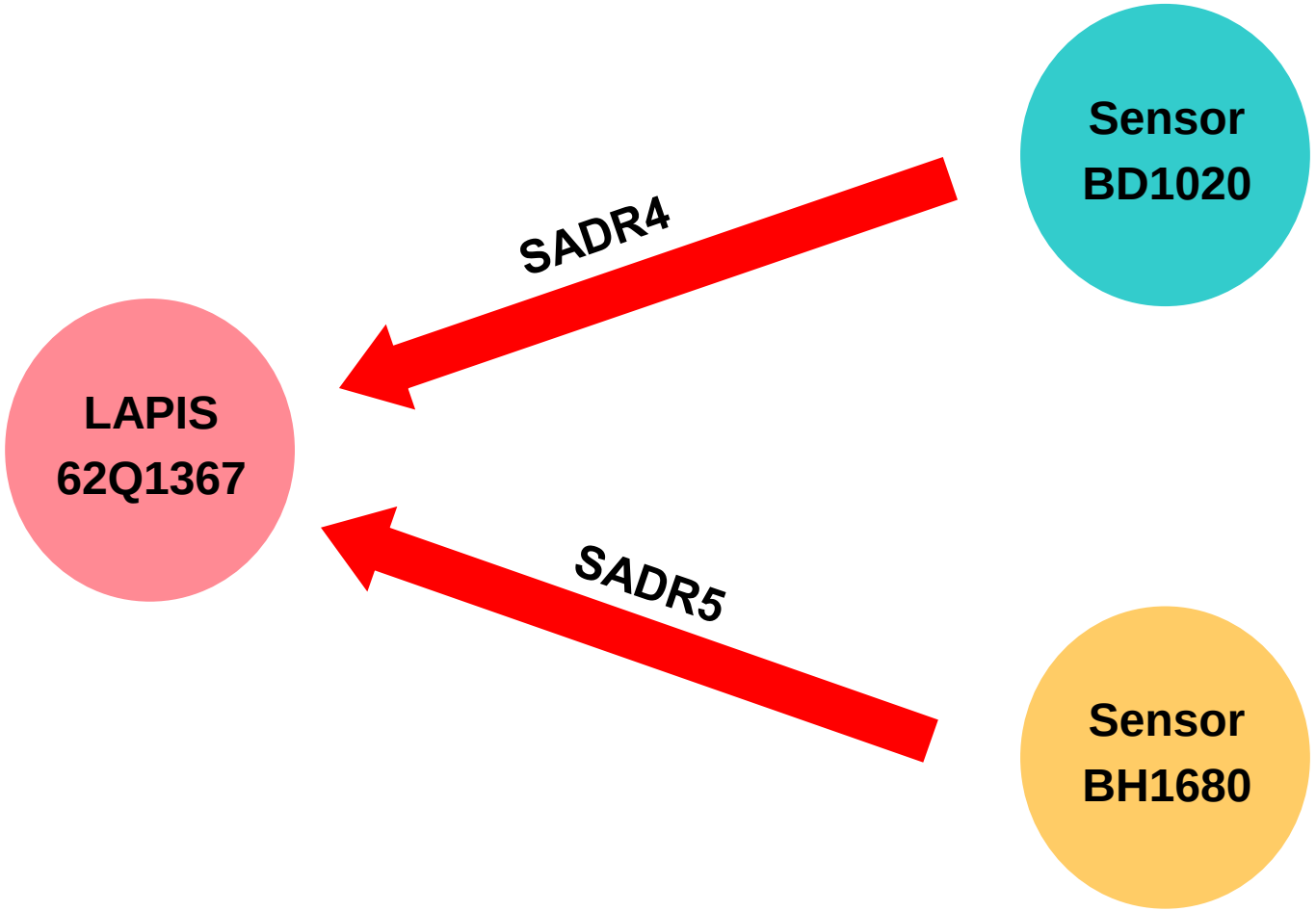
## Concept






Pichet Sriphomma pichet@es.co.th

## Concept






Pichet Sriphomma pichet@es.co.th

## BH1680FVC

This IC can detect the illuminance from 0lx to 50000lx.
Supply voltage operates from 2.4V to 5.5V.
Supply Current1 ( Operate )  75 µA.

| Illuminance detection range [lx] | Gain Mode |
|---|---|
| ~100 | H-Gain Mode |
| ~1,000 | M-Gain Mode |
| ~50,000 | L-Gain Mode |

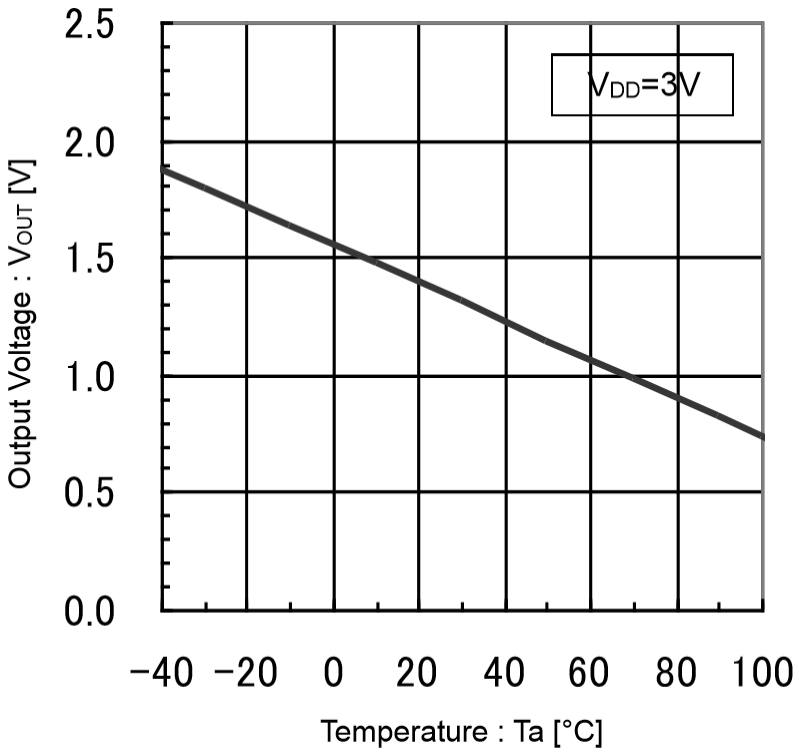
M-Gain mode

$$V_{iout} = 0.61 \times 10^{-6} \times E_V \times R1$$



Pichet Sriphomma pichet@es.co.th

## BD1020HFV




**Temperature Sensitivity** $(V_{SE})$

$$\text{Slope} = \frac{1-1.9}{70-(-40)}$$

$$\text{Slope} = -8.1818 \times 10^{-3}$$

$$\text{Slope} \approx -8.2\ [mV/C]$$



Pichet Sriphomma pichet@es.co.th

## BD1020HFV

### Electrical Characteristics

(Unless otherwise specified, $V_{DD}$=3.0V, Ta=25°C)

| Parameter | Symbol | Limit | | | Unit | Conditions |
|---|---|---|---|---|---|---|
| | | Min | Typ | Max | | |
| Accuracy | Tacc | - | - | ±1.5 | °C | Ta = 30°C |
| | | - | - | ±2.5 | | Ta = 100°C |
| | | - | - | ±2.5 | | Ta = -30°C |
| Temperature Sensitivity | $V_{SE}$ | -8.4 | -8.2 | -8.0 | mV/°C | |
| Supply Current | $I_S$ | - | 4.0 | 7.0 | µA | |
| Output Voltage | $V_{OUT}$ | 1.288 | 1.300 | 1.312 | V | Ta = 30°C |
| Output Voltage Line Regulation | $\Delta V_{OUTVDD}$ | - | - | 4 | mV | $V_{DD}$ = 2.4V to 5.5V |
| Output Voltage Load Regulation | $\Delta V_{OUTRL}$ | - | - | 1 | mV | Difference of $I_{OUT}$: 0µA/0.7µA |



Pichet Sriphomma pichet@es.co.th

## BD1020HFV

### Output Voltage

Find linear equations :

At $T_a = 30\ C$ $V_{out} = 1.3\ V$

$$y - y_0 = m(x - x_0)$$

$$V_{out} - V_{out30C} = V_{SE}\left(T_a - T_{a30C}\right)$$

$$V_{out} = V_{SE}\left(T_a - T_{a30C}\right) + V_{out30C}$$

$$V_{out}[mV] = -8.2T_a[CmV/C] + 246[mV] + 1300[mV]$$

$$V_{out}[mV] = -8.2T_a[CmV/C] + 1546[mV]$$

$$T_a[C] = \frac{V_{out}[mV] - 1546[mV]}{-8.2[mV/C]}$$

$$T_a[C] = \frac{\left(-V_{out}[V] \times 1000\right) + 1546[mV]}{8.2[mV/C]}$$

$$V_{out}[mV] = -8.2[mV/C]\left(T_a[C] - 30[C]\right) + 1300[mV]$$



Pichet Sriphomma pichet@es.co.th

## Set ADC Pin (adc.c)

```
void Set_ADC_Pin4(void)
{
    P24IE=0;P24OE=0;P24OD=0;P24PU=0;     //
    P24MD3=0;P24MD2=1;P24MD1=1;P24MD0=1;
}
//----------------------------------------
void Set_ADC_Pin5(void)
{
    P25IE=0;P25OE=0;P25OD=0;P25PU=0;     //
    P25MD3=0;P25MD2=1;P25MD1=1;P25MD0=1;
}
```

### 17.2.3 Port n Mode Register 01 (PnMOD01:n=0 to 9, A, B)

PnMOD01 is a special function register (SFR) to choose the input/output mode, input/output status, and shared function of Pn0 pin and Pn1 pin.
See Table 17-2 "List of Registers / Bits" to check avaible pins and bits.
Wirte "0" to the bits of PnMOD01 register that have no corresponding pin.

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | PnMOD01 | | | | | | | | | | | | | | | |
| Byte | PnMOD1 | | | | | | | | PnMOD0 | | | | | | | |
| Bit | Pn1MD3 | Pn1MD2 | Pn1MD1 | Pn1MD0 | Pn1OD | Pn1PU | Pn1OE | Pn1IE | Pn0MD3 | Pn0MD2 | Pn0MD1 | Pn0MD0 | Pn0OD | Pn0PU | Pn0OE | Pn0IE |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | * | 0 | * |

* :The initial value of P00IE and P00PU for the Port0 is "1" and other bits are "0".

| | |
|---|---|
| Pn1MD3 to Pn1MD0 | This bit is used to choose the shared function of Pn1 pin.<br>For the details of the shared function, see Table 1-7 "ML62Q1300 Group Pin List", Table 1-8 "ML62Q1500 Group Pin List" and Table 1-9 "ML62Q1700 Group Pin List".<br>0000: Primary function (initial value)<br>0001: 2nd function<br>0010: 3rd function<br>0011: 4th function<br>0100: 5th function<br>0101: 6th function<br>0110: 7th function<br>0111: 8th function<br>1XXX: Do not use (Primary function)<br>X: 0 or 1 (don't care) |



Pichet Sriphomma pichet@es.co.th

# Initial ADC (adc.c)

```
void Init_ADC(void)
{
    //Set_ADC_Pin0();
    //Set_ADC_Pin1();
    //Set_ADC_Pin2();
    //Set_ADC_Pin3();
    Set_ADC_Pin4();
    Set_ADC_Pin5();
    //Set_ADC_Pin6();
    //Set_ADC_Pin7();
    //Set_ADC_Pin8();
    //Set_ADC_Pin9();
    //Set_ADC_Pin10();
    //Set_ADC_Pin11();

    // VREFCON
    VREFEN = 0;      // Disable internal
    VREFP1=0;VREFP0=0;   // Select VDD as

    // SADMOD
    SALP = 0;        // Single conversion
    SACK2=0;SACK1=0;SACK0=0;    // At PL
                     // Clock Peroid
    SASHT3=1;SASHT2=0;SASHT1=1;SASHT0=0;
                            // this sett
    SAINIT = 1;      // Discharge sample h

    // SADEN0

}
```

## 23.2.14 SA-ADC Reference Voltage Control Register (VREFCON)

VREFCON is a special function register (SFR) used to choose the internal reference voltage operation and control the operation of the temperature sensor,.

Address: 0xF83A(VREFCON)
Access: R/W
Access size: 8bit
Initial value: 0x00

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | - | | | | | | | | | | | | | | | |
| Byte | - | | | | | | | | VREFCON | | | | | | | |
| Bit | - | - | - | - | - | - | - | - | - | - | VREFP1 | VREFP0 | - | - | - | VREFEN |
| R/W | R | R | R | R | R | R | R | R | R | R | R/W | R/W | R | R | R | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | |
|---|---|
| VREFP1, VREFP0 | These bits are used to choose the reference voltage for the A/D conversion.<br>00: Voltage input from the VDD pin (Initial value)<br>01: Voltage input from the VREF pin<br>10: Voltage generated by the internal reference voltage circuit (approx.1.55V)<br>11: Do not use (Voltage input from the VDD pin) |
| - | Reserved bit |
| VREFEN | This bit is used to enable the operation of internal reference voltage and the temperature sensor. When using the internal reference voltage (approx. 1.55V) or temperature sensor, set the VREFEN bit to "1".<br>0: Disable the operation of internal reference voltage and temperature sensor (Initial value)<br>1: Enable the operation of internal reference voltage and temperature sensor |

# Initial ADC (adc.c)

```
void Init_ADC(void)
{
    //Set_ADC_Pin0();
    //Set_ADC_Pin1();
    //Set_ADC_Pin2();
    //Set_ADC_Pin3();
    Set_ADC_Pin4();
    Set_ADC_Pin5();
    //Set_ADC_Pin6();
    //Set_ADC_Pin7();
    //Set_ADC_Pin8();
    //Set_ADC_Pin9();
    //Set_ADC_Pin10();
    //Set_ADC_Pin11();

    // VREFCON
    VREFEN = 0;      // Disable internal
    VREFP1=0;VREFP0=0;  // Select VDD as

    // SADMOD
    SALP = 0;        // Single conversion
    SACK2=0;SACK1=0;SACK0=0;     // At PL
                     // Clock Peroid
    SASHT3=1;SASHT2=0;SASHT1=1;SASHT0=0;
                                 // this set
    SAINIT = 1;      // Discharge sample

    // SADEN0

}
```

## 23.2.6 SA-ADC Mode Register (SADMOD)

SADMOD is a special function register (SFR) used to set the operation mode and operating clock frequency of the A/D converter. The bit symbol "rsvd" means a reserved bit, write "0" to those bits.

Address: 0xF828(SADMODL/SADMOD), 0xF829(SADMODH)
Access: R/W
Access size: 8/16bit
Initial value: 0x0000

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | SADMOD | | | | | | | | | | | | | | | |
| Byte | SADMODH | | | | | | | | SADMODL | | | | | | | |
| Bit | rsvd | rsvd | rsvd | rsvd | rsvd | rsvd | rsvd | SAINIT | SASHT3 | SASHT2 | SASHT1 | SASHT0 | SACK2 | SACK1 | SACK0 | SALP |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | |
|---|---|
| SACK2 to SACK0 | These bits are used to choose the frequency of the A/D conversion operating clock (SAD_CLK). See Chapter 24.3.2 "A/D Conversion Time Setting" for the operating clock, A/D conversion time and sample time.<br>000: 8MHz (Initial value)<br>001: 4MHz<br>010: 2MHz<br>011: 1MHz<br>100: 0.5MHz<br>101: Do not use<br>110: Do not use<br>111: 32kHz |
| SALP | This bit is used to choose whether the A/D conversion is performed once only for each channel or consecutively. The conversion interval time in the consecutive scan A/D conversion mode is specified in the SADSTM register.<br>0: Single A/D conversion (Initial value)<br>1: Consecutive scan A/D conversion |



Pichet Sriphomma pichet@es.co.th

# Initial ADC (adc.c)

```
void Init_ADC(void)
{
    //Set_ADC_Pin0();
    //Set_ADC_Pin1();
    //Set_ADC_Pin2();
    //Set_ADC_Pin3();
    Set_ADC_Pin4();
    Set_ADC_Pin5();
    //Set_ADC_Pin6();
    //Set_ADC_Pin7();
    //Set_ADC_Pin8();
    //Set_ADC_Pin9();
    //Set_ADC_Pin10();
    //Set_ADC_Pin11();

    // VREFCON
    VREFEN = 0;      // Disable inter
    VREFP1=0;VREFP0=0;  // Select VD

    // SADMOD
    SALP = 0;        // Single conver
    SACK2=0;SACK1=0;SACK0=0;     // A
                     // Clock Peroid =
    SASHT3=1;SASHT2=0;SASHT1=1;SASHT0=0;
                          // this sett
    SAINIT = 1;      // Discharge sample h

    // SADEN0

}
```

## 23.2.6 SA-ADC Mode Register (SADMOD)

SADMOD is a special function register (SFR) used to set the operation mode and operating clock frequency of the A/D converter. The bit symbol "rsvd" means a reserved bit, write "0" to those bits.

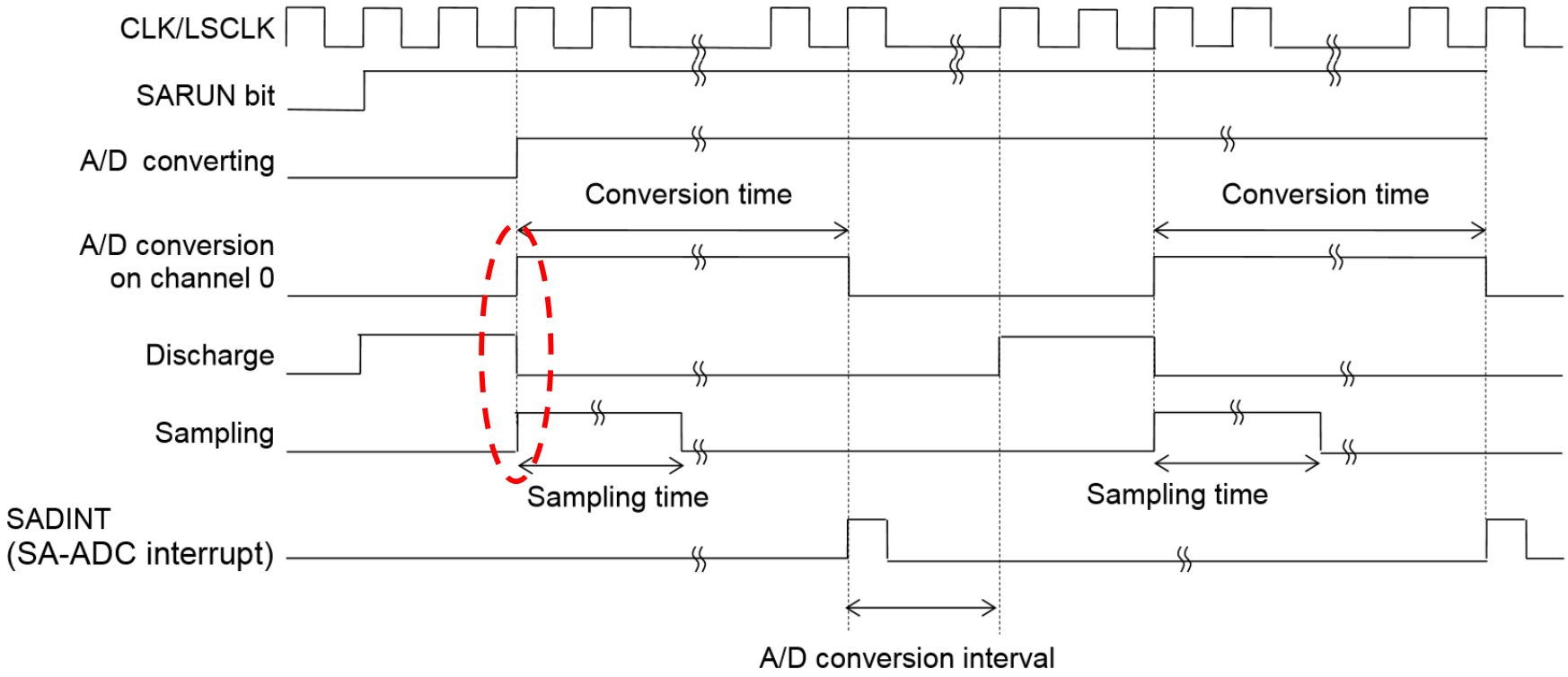
Address: 0xF828(SADMODL/SADMOD), 0xF829(SADMODH)
Access: R/W
Access size: 8/16bit
Initial value: 0x0000

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | SADMOD | | | | | | | | | | | | | | | |
| Byte | SADMODH | | | | | | | | SADMODL | | | | | | | |
| Bit | rsvd | rsvd | rsvd | rsvd | rsvd | rsvd | rsvd | SAINIT | SASHT3 | SASHT2 | SASHT1 | SASHT0 | SACK2 | SACK1 | SACK0 | SALP |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | |
|---|---|
| SAINIT | This bit is used to control whether or not to discharge the electrical charge remained in the sample hold capacitor on the previous A/D conversion, before starting the next SA-ADC conversion.<br>0: Start the A/D conversion without discharging the electrical charge accumulated in the sample hold capacitor (Initial value)<br>1: Start the A/D conversion after discharging the electrical charge accumulated in the sample hold capacitor |
| SASHT3 to SASHT0 | These bits are used to set the sampling time.<br>See Chapter 24.3.2 "A/D Conversion Time Setting" for details. |

## Conversion time

Figure 23-6 shows the operation waveforms when the continuous A/D conversion is performed using channel 0.






Pichet Sriphomma pichet@es.co.th

## Conversion time

Table 23-3 A/D Conversion time when using $V_{DD}$ or $V_{REF}$ pin as reference voltage

| SADMOD | | | | Conversion clock count | Conversion time*1 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | SAD_CLK | | | | | |
| SASHT[3:0] | | | | | 32kHz | 0.5MHz | 1MHz | 2MHz | 4MHz | 8MHz |
| 0 | 0 | 0 | 0 | 14 | 427 µs | 28 µs | Prohibited | Prohibited | Prohibited | Prohibited |
| 0 | 0 | 0 | 1 | 15 | Prohibited | 30 µs | 15 µs | | | |
| 0 | 0 | 1 | 0 | 16 | | 32 µs | 16 µs | 8 µs | | |
| 0 | 0 | 1 | 1 | 17 | | 34 µs | 17 µs | 8.5 µs | 4.25 µs | |
| 0 | 1 | 0 | 0 | 18 | | 36 µs | 18 µs | 9 µs | 4.5 µs | 2.25 µs |
| 0 | 1 | 0 | 1 | 19 | | 38 µs | 19 µs | 9.5 µs | 4.75 µs | 2.375 µs |
| 0 | 1 | 1 | 0 | 20 | | 40 µs | 20 µs | 10 µs | 5 µs | 2.5 µs |
| 0 | 1 | 1 | 1 | 21 | | 42 µs | 21 µs | 10.5 µs | 5.25 µs | 2.625 µs |
| 1 | 0 | 0 | 0 | 29 | | 58 µs | 29 µs | 14.5 µs | 7.25 µs | 3.625 µs |
| 1 | 0 | 0 | 1 | 45 | | 90 µs | 45 µs | 22.5 µs | 11.25 µs | 5.625 µs |
| 1 | 0 | 1 | 0 | 61 | | 122 µs | 61 µs | 30.5 µs | 15.25 µs | 7.625 µs |
| 1 | 0 | 1 | 1 | 77 | | 154 µs | 77 µs | 38.5 µs | 19.25 µs | 9.625 µs |
| 1 | 1 | 0 | 0 | 93 | | 186 µs | 93 µs | 46.5 µs | 23.25 µs | 11.625 µs |
| 1 | 1 | 0 | 1 | 109 | | 218 µs | 109 µs | 54.5 µs | 27.25 µs | 13.625 µs |
| 1 | 1 | 1 | 0 | 125 | | 250 µs | 125 µs | 62.5 µs | 31.25 µs | 15.625 µs |
| 1 | 1 | 1 | 1 | 141 | | 282 µs | 141 µs | 70.5 µs | 35.25 µs | 17.625 µs |

*1: The A/D conversion time does not include discharging time (two clocks of the SAD_CLK) and the clock frequency error.



Pichet Sriphomma pichet@es.co.th

## Sampling time

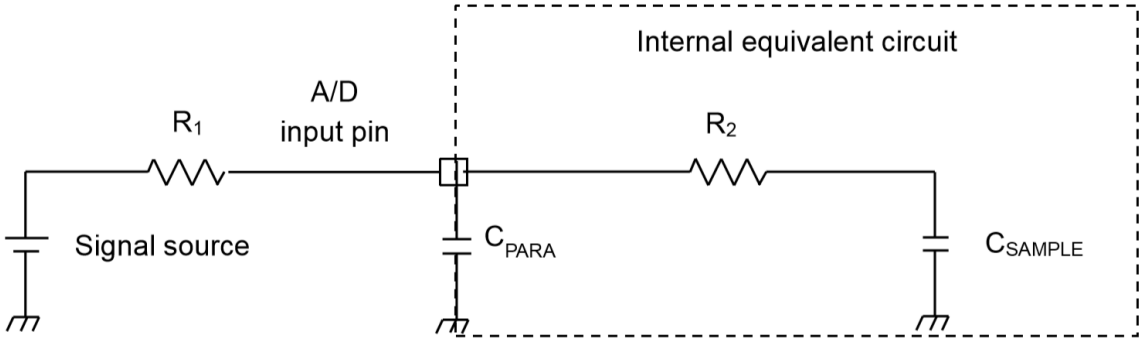
Table 23-5 Sampling time when using $V_{DD}$ or $V_{REF}$ pin as reference voltage

| SADMOD | | | | Sample/ hold clock count | Sampling time | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | SAD_CLK | | | | | |
| SASHT[3:0] | | | | | 32kHz | 0.5MHz | 1MHz | 2MHz | 4MHz | 8MHz |
| 0 | 0 | 0 | 0 | 1 | 30 µs | 2 µs | Prohibited | Prohibited | Prohibited | Prohibited |
| 0 | 0 | 0 | 1 | 2 | Prohibited | 4 µs | 2 µs | | | |
| 0 | 0 | 1 | 0 | 3 | | 6 µs | 3 µs | 1.5 µs | | |
| 0 | 0 | 1 | 1 | 4 | | 8 µs | 4 µs | 2 µs | 1 µs | |
| 0 | 1 | 0 | 0 | 5 | | 10 µs | 5 µs | 2.5 µs | 1.25 µs | 0.625 µs |
| 0 | 1 | 0 | 1 | 6 | | 12 µs | 6 µs | 3 µs | 1.5 µs | 0.75 µs |
| 0 | 1 | 1 | 0 | 7 | | 14 µs | 7 µs | 3.5 µs | 1.75 µs | 0.875 µs |
| 0 | 1 | 1 | 1 | 8 | | 16 µs | 8 µs | 4 µs | 2 µs | 1 µs |
| 1 | 0 | 0 | 0 | 16 | | 32 µs | 16 µs | 8 µs | 4 µs | 2 µs |
| 1 | 0 | 0 | 1 | 32 | | 64 µs | 32 µs | 16 µs | 8 µs | 4 µs |
| 1 | 0 | 1 | 0 | 48 | | 96 µs | 48 µs | 24 µs | 12 µs | 6 µs |
| 1 | 0 | 1 | 1 | 64 | | 128 µs | 64 µs | 32 µs | 16 µs | 8 µs |
| 1 | 1 | 0 | 0 | 80 | | 160 µs | 80 µs | 40 µs | 20 µs | 10 µs |
| 1 | 1 | 0 | 1 | 96 | | 192 µs | 96 µs | 48 µs | 24 µs | 12 µs |
| 1 | 1 | 1 | 0 | 112 | | 224 µs | 112 µs | 56 µs | 28 µs | 14 µs |
| 1 | 1 | 1 | 1 | 128 | | 256 µs | 128 µs | 64 µs | 32 µs | 16 µs |



Pichet Sriphomma pichet@es.co.th

## Sampling Time Setting




| $V_{DD}$ | $R_2$[kΩ] | $C_{SAMPLE}$[pF] |
|---|---|---|
| 1.8 V≤$V_{DD}$≤2.2V | 500k | 5pF |
| 2.2 V≤$V_{DD}$≤2.7V | 100k | 5pF |
| 2.7 V≤$V_{DD}$≤4.5V | 8k | 5pF |
| 4.5 V≤$V_{DD}$≤5.5V | 7k | 5pF |

$$\text{Sampling time} > 8C_{SAMPLE}R_2$$



Pichet Sriphomma pichet@es.co.th

## Scan ADC (adc.c)

```
void ScanADC(void)
{
    static unsigned char index=0;

    //GetADC(0,index);
    //GetADC(1,index);
    //GetADC(2,index);
    //GetADC(3,index);
    GetADC(4,index);
    GetADC(5,index);
    //GetADC(6,index);
    //GetADC(7,index);
    //GetADC(8,index);
    //GetADC(9,index);
    //GetADC(10,index);
    //GetADC(11,index);

    index = ++index%8;
}
```

Receive ADC value from Channel 4,5



Pichet Sriphomma pichet@es.co.th

## Get ADC (adc.c)

```
void GetADC(unsigned char ch,unsigned char id)
{
    unsigned int result;

    SADEN0 = ADC_CH[ch];
    SARUN  = 1;
    __asm("nop");
    __asm("nop");
    __asm("nop");
    __asm("nop");

    while(SARUN == 1)
        __asm("nop");

    result = Get_ADC_Result(ch);
    ADC_Buf[ch][id] = result>>6;
    ADC_Value[ch] = Average8(ch);
}
```

### 23.2.8 SA-ADC Enable Register 0 (SADEN0)

SADEN0 is a special function register (SFR) used to choose channels of the A/D converter and enable/disable the conversion.

Address: 0xF82C(SADEN0L/SADEN0), 0xF82D(SADEN0H)
Access: R/W
Access size: 8/16bit
Initial value: 0x0000

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | SADEN0 | | | | | | | | | | | | | | | |
| Byte | SADEN0H | | | | | | | | SADEN0L | | | | | | | |
| Bit | SACH15 | SACH14 | SACH13 | SACH12 | SACH11 | SACH10 | SACH09 | SACH08 | SACH07 | SACH06 | SACH05 | SACH04 | SACH03 | SACH02 | SACH01 | SACH00 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SACH15 to SACH00 — These bits are used to choose channel n (n=0 to 15) of the A/D converter and enable/disable the conversion.

- SACH00: Enable or Disable the A/D conversion on channel 0
- SACH01: Enable or Disable the A/D conversion on channel 1
- SACH02: Enable or Disable the A/D conversion on channel 2
- SACH03: Enable or Disable the A/D conversion on channel 3
- SACH04: Enable or Disable the A/D conversion on channel 4
- SACH05: Enable or Disable the A/D conversion on channel 5
- SACH06: Enable or Disable the A/D conversion on channel 6
- SACH07: Enable or Disable the A/D conversion on channel 7
- SACH08: Enable or Disable the A/D conversion on channel 8
- SACH09: Enable or Disable the A/D conversion on channel 9
- SACH10: Enable or Disable the A/D conversion on channel 10
- SACH11: Enable or Disable the A/D conversion on channel 11
- SACH12: Enable or Disable the A/D conversion on channel 12
- SACH13: Enable or Disable the A/D conversion on channel 13
- SACH14: Enable or Disable the A/D conversion on channel 14
- SACH15: Enable or Disable the A/D conversion on channel 15

0: Disable the conversion on channel n (initial value)
1: Enable the conversion on channel n

## Get ADC (adc.c)

```
void GetADC(unsigned char ch,unsigned char id)
{
    unsigned int result;

    SADEN0 = ADC_CH[ch];
    SARUN  = 1;
    __asm("nop");
    __asm("nop");
    __asm("nop");
    __asm("nop");

    while(SARUN == 1)
        __asm("nop");

    result = Get_ADC_Result(ch);
    ADC_Buf[ch][id] = result>>6;
    ADC_Value[ch] = Average8(ch);
}
```

### 23.2.7 SA-ADC Control Register (SADCON)

SADCON is a special function register (SFR) used to control the operation of the A/D converter.

Address: 0xF82A(SADCONL/SADCON), 0xF82B(SADCONH)
Access: R/W
Access size: 8/16bit
Initial value: 0x0000

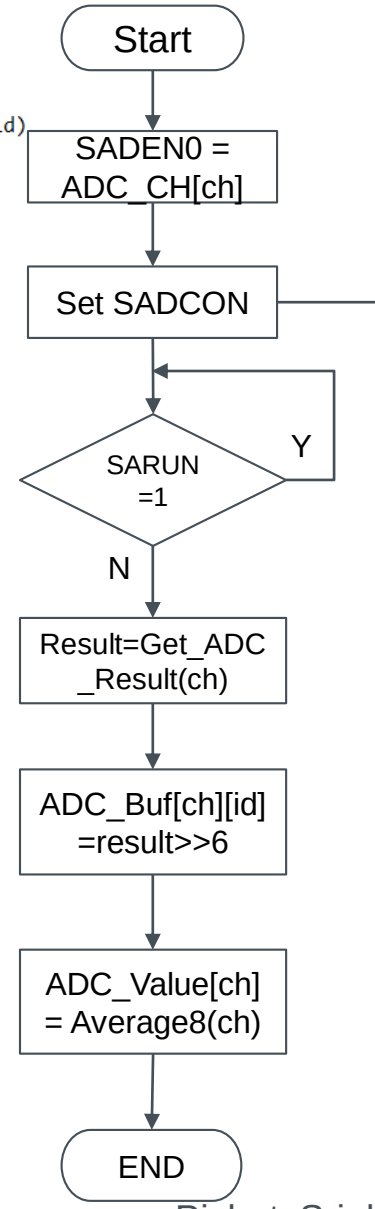
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | SADCON | | | | | | | | | | | | | | | |
| Byte | SADCONH | | | | | | | | SADCONL | | | | | | | |
| Bit | - | - | - | - | - | - | - | - | - | - | - | - | - | - | SATGEN | SARUN |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SARUN — This bit is used to start or stop the A/D conversion.
Write "1" to this bit to start the A/D conversion, and "0" to stop it.
When "0" is written to SALP bit and the A/D conversion on the largest number of channel is ended, this SARUN bit is automatically reset to "0".
When "1" is written to SALP, the A/D conversion repeats until the SARUN bit is reset to "0" by the software.

0: Stop the A/D conversion (Initial value)
1: Start the A/D conversion

## Get ADC (adc.c)

```
void GetADC(unsigned char ch,unsigned char id)
{
    unsigned int result;

    SADEN0 = ADC_CH[ch];
    SARUN   = 1;
    __asm("nop");
    __asm("nop");
    __asm("nop");
    __asm("nop");

    while(SARUN == 1)
        __asm("nop");

    result = Get_ADC_Result(ch);
    ADC_Buf[ch][id] = result>>6;
    ADC_Value[ch] = Average8(ch);
}
```




Enable Channel

Start Conversion

เช็ค Flag SARUN ว่า ADC แปลงเสร็จหรือยัง ?

ADC_Buf เป็น Buffer ที่ใช้เก็บข้อมูลดิบ

ADC_Value เป็น Buffer ที่ใช้เก็บข้อมูลที่เฉลี่ยแล้ว (หาร 8)

ข้อมูลที่เก็บไว้ใน Buffer ADC_Buf ที่เป็น 16 Bit แต่ ADC เป็น 10 Bit ดังนั้นต้องเลื่อน Bit ข้อมูลที่อยู่ทางซ้าย 16 Bit เลื่อนไปทางขวาเป็น 10 Bit

## Get ADC Result (adc.c)

```
unsigned int Get_ADC_Result(unsigned char ch)
{
    switch (ch)
    {
        case 0: return(SADR0);
        case 1: return(SADR1);
        case 2: return(SADR2);
        case 3: return(SADR3);
        case 4: return(SADR4);
        case 5: return(SADR5);
        case 6: return(SADR6);
        case 7: return(SADR7);

        default:    return 0;

    }
}
```

### 23.2.2 SA-ADC Result Register n (SADRn : n=0 to 15, 16)

SADRn is a special function register (SFR) used to store the SA-ADC conversion results on channels 0 to 15 and channel 16 (temperature sensor).
The A/D conversion result of each channel can be read from SADRn.

| Symbol name | Channel |
|---|---|
| SADR0 | The conversion result of channel 0 (AIN0) |
| SADR1 | The conversion result of channel 1 (AIN1) |
| SADR2 | The conversion result of channel 2 (AIN2) |
| SADR3 | The conversion result of channel 3 (AIN3) |
| SADR4 | The conversion result of channel 4 (AIN4) |
| SADR5 | The conversion result of channel 5 (AIN5) |
| SADR6 | The conversion result of channel 6 (AIN6) |
| SADR7 | The conversion result of channel 7 (AIN7) |

Access: R
Access size: 8/16bit
Initial value: 0x0000

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | SADRn | | | | | | | | | | | | | | | |
| Byte | SADRnH | | | | | | | | SADRnL | | | | | | | |
| Bit | d15 | d14 | d13 | d12 | d11 | d10 | d9 | d8 | d7 | d6 | - | - | - | - | - | - |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



Pichet Sriphomma pichet@es.co.th

## Average8 (adc.c)

```
unsigned int Average8(unsigned char ch)
{
    unsigned char i;
    unsigned int sum=0;

    for(i=0;i<8;i++)
        sum += ADC_Buf[ch][i];

    return(sum /8);
}
```

Find the average of ADC value all 8 values



Pichet Sriphomma pichet@es.co.th

## BD1020 Read (adc.c)

```
void BD1020_Read_Temp(void){

    unsigned int vin;
    float TIN;

/*************************************************************
    //vin = SADR4;
    vin = Avg_ADC4;

/*************************************************************

    // Calculations for Temp Sensor - BD1020
    // Math:  ADC_Voltage = sensorValue * (VCC/1024)
    // Tempurature = - (1000 * ADC_Voltage - 1546) / 8.2
    //TIN = (1546-(1000*(vin*((float)Vcc/4096))))/8.2;
    TIN = (1546-(1000*(vin*((float)Vcc/1024))))/8.2;

    Real_ADC4 = TIN;

}
```

$$T_a[C] = \frac{\left(-V_{out}[V] \times 1000\right) + 1546[mV]}{8.2[mV/C]}$$

$$V_{out} = V_{in} \frac{V_{CC}}{1024}$$



Pichet Sriphomma pichet@es.co.th

## BH1680 Read (adc.c)

```
void BD1680_Read_ALS(void){

    unsigned int vin;
    float EV;
    float R1 = 11000;
    float H_const = 6.1;
    float M_const = 0.61;

/*************************************************************
    //vin = SADR5;
    vin = Avg_ADC5;

/*************************************************************

    // Calculations for ALS - BD1680
    // Math:  ADC_Voltage = sensorValue * `(VCC/4096)
    // H-Gain Mode: Viout = 6.1* (10^(-6)) * EV *R1
    // M-Gain Mode: Viout = 0.61* (10^(-6)) * EV *R1
    // L-Gain Mode: Viout = 0.061* (10^(-6)) * EV *R1

    EV = (vin*((float)Vcc/1024)/(R1*M_const*0.000001));

    //EV = (((vin*((float)Vcc/4096))/R1)/(H_const*0.000001));

    Real_ADC5 = EV;
}
```

$$V_{iout} = 0.61 \times 10^{-6} \times E_V \times R1$$

$$E_V = \frac{V_{iout}}{0.61 \times 10^{-6} \times R1}$$

$$V_{iout} = V_{in} \frac{V_{CC}}{1024}$$



Pichet Sriphomma pichet@es.co.th

## Concept


ST,SL <Hundreds – Thousands><Tens><decimal>



Pichet Sriphomma pichet@es.co.th

## UART00 BD1020 (uart0.c)

```
void UART00_BD1020(void)
{
    unsigned int i=0;
    Sensor_Index = 0;
    //*************** Temperature Sensor ************
1   ADC4_float = Real_ADC4;

2   ADC4_int = ADC4_float;
3   ADC4_point = (ADC4_float - ADC4_int)*100;

4   ADC4_Char_High = (ADC4_int & 0xFF00)>>8;

5   ADC4_Char_Low = ADC4_int & 0x00FF;

    Clear_Buffer();


        UART00_TX_Buf[0] = ADC4_Char_High;
        UART00_TX_Buf[1] = ADC4_Char_Low;
        UART00_TX_Buf[2] = ADC4_point;


        TXD00_Index = 0;
        Flag._TXD00 = 1;
        Sensor_Index = 1;
        Set_UART00_TX();


    RXD00_Index = 0;
    //Flag._RXD00 = 0;
}
```

**Example**

**ADC Value of BD1020 (Real_ADC4) = 30.24**

**1. ADC4_float = 30.24**

**2. ADC4_int = 30**

**3. ADC4_point = (30.24 – 30)*100 = 24**

**4. ADC4_Char_High = 0x0030 & 0xFF00 = 0x0000**

**= 0x0000 >> 8 = 0x0000**

**5. ADC4_Char_Low = 0x0030 & 0x00FF = 0x0030**

**UART_TX_Buf[0] = 0**

**UART_TX_Buf[1] = 30**

**UART_TX_Buf[2] = 24**

## UART00 BH1680 (uart0.c)

```
void UART00_BH1680(void)
{
    unsigned int i=0;
    Sensor_Index = 0;
    //*************** Ambient Light Sensor **********
1   ADC5_float = Real_ADC5;

2   ADC5_int = ADC5_float;
3   ADC5_point = (ADC5_float - ADC5_int)*100;

4   ADC5_Char_High = (ADC5_int & 0xFF00)>>8;

5   ADC5_Char_Low = ADC5_int & 0x00FF;

    Clear_Buffer();

        UART00_TX_Buf[0] = ADC5_Char_High;
        UART00_TX_Buf[1] = ADC5_Char_Low;
        UART00_TX_Buf[2] = ADC5_point;

        TXD00_Index = 0;
        Flag._TXD00 = 1;
        Sensor_Index = 1;
        Set_UART00_TX();


    RXD00_Index = 0;
    //Flag._RXD00 = 0;
}
```

**Example**

**ADC Value of BH1680 (Real_ADC5) = 240.68**

**1. ADC5_float = 240.68**

**2. ADC5_int = 240**

**3. ADC5_point = (240.68 – 240)*100 = 68**

**4. ADC5_Char_High = 0x0240 & 0xFF00 = 0x0200**

**= 0x0200 >> 8 = 0x0002**

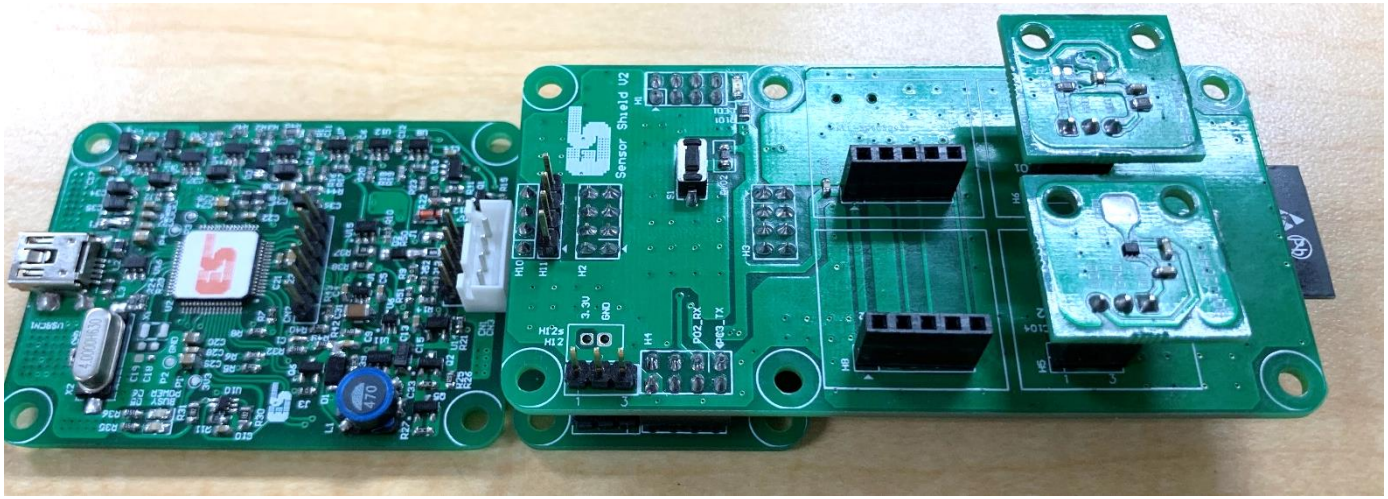
**5. ADC5_Char_Low = 0x0240 & 0x00FF = 0x0040**

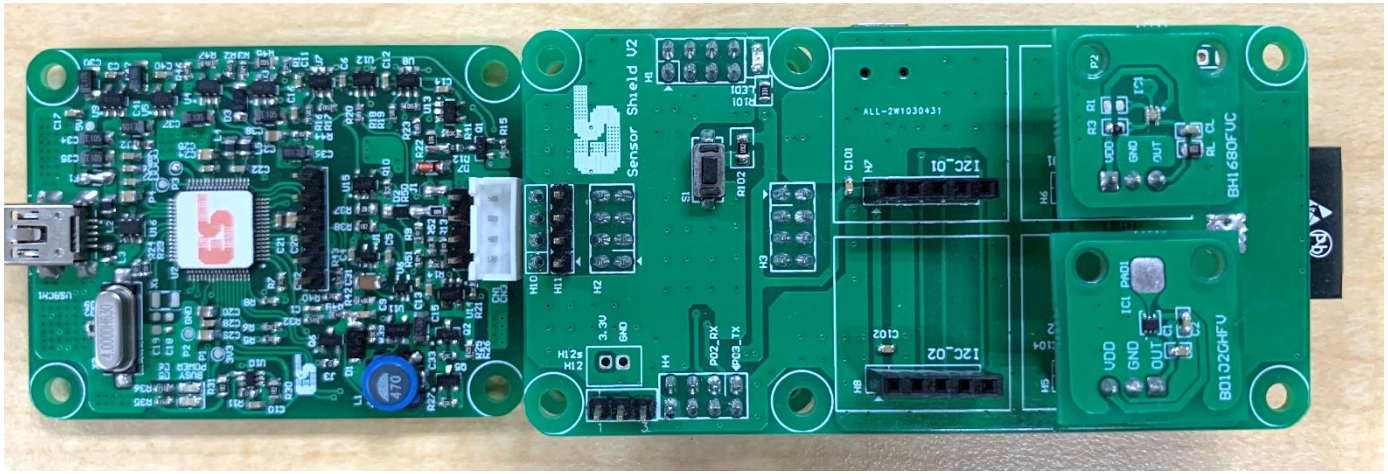
**UART_TX_Buf[0] = 2**

**UART_TX_Buf[1] = 40**

**UART_TX_Buf[2] = 68**

# Connection ES-ICD-V1 ,Sensor Shield and Module

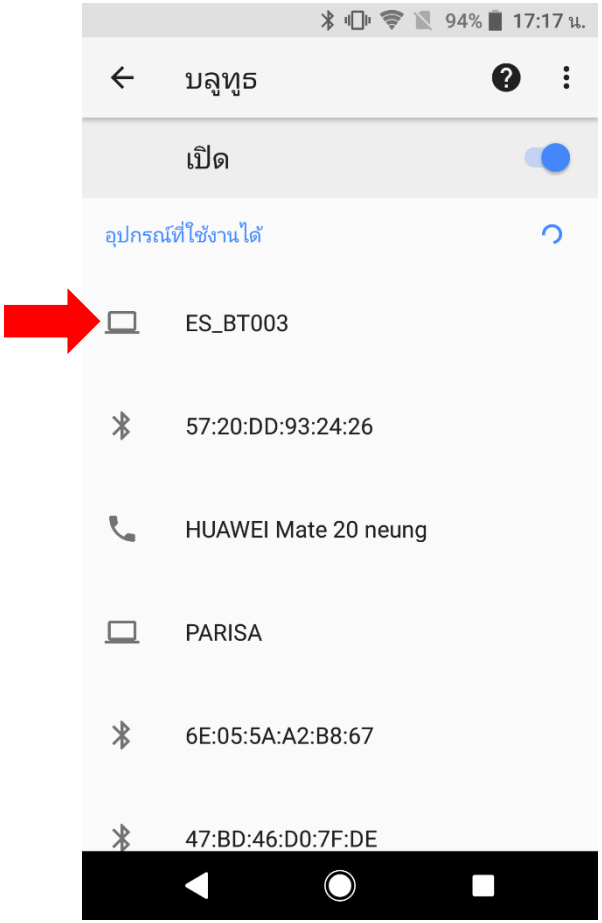


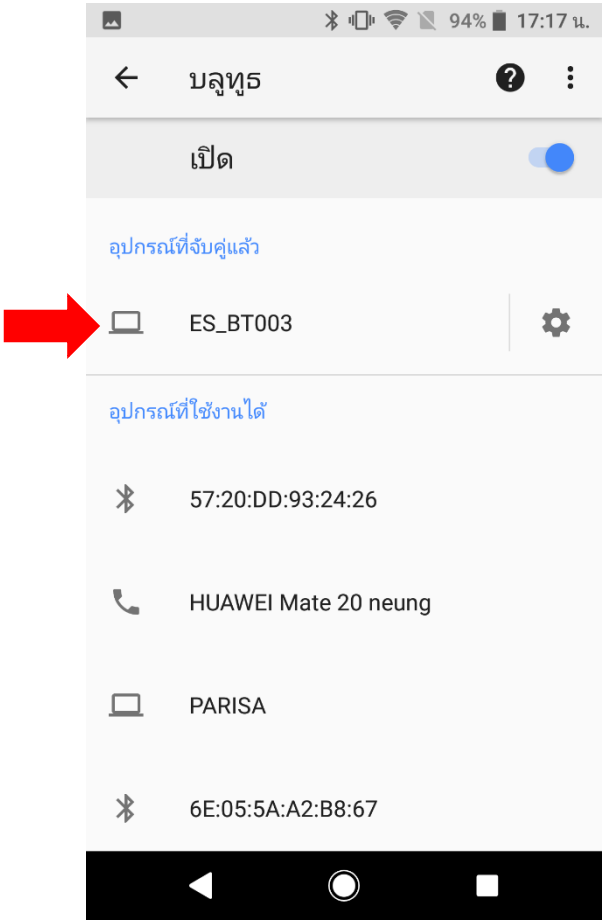




Pichet Sriphomma pichet@es.co.th

## Using the application

### Step 1 Connect MCU Board with Bluetooth






Pichet Sriphomma pichet@es.co.th

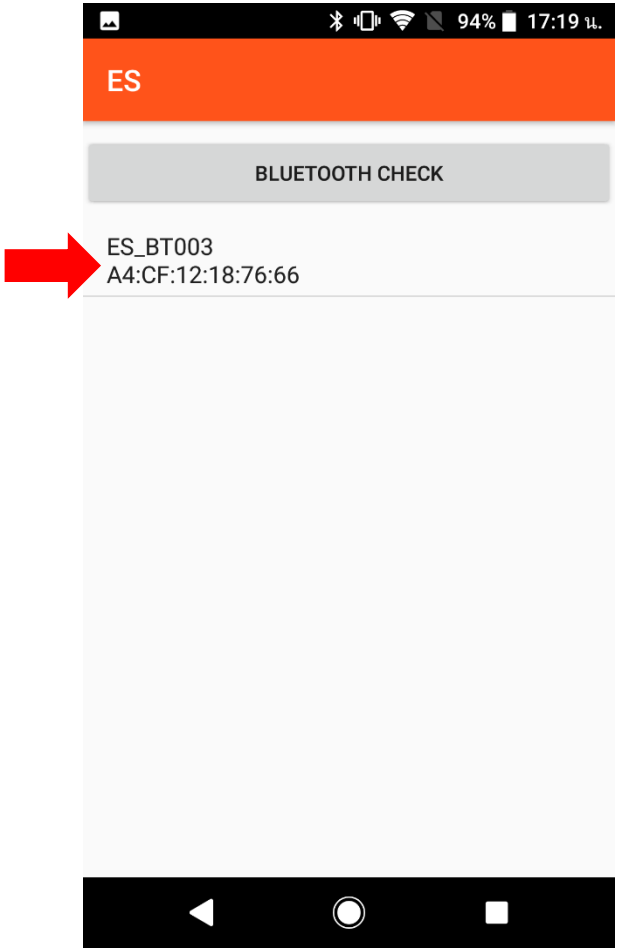
## Using the application

### Step 2 Open application and click “Bluetooth Check”


Pichet Sriphomma pichet@es.co.th

## Using the application

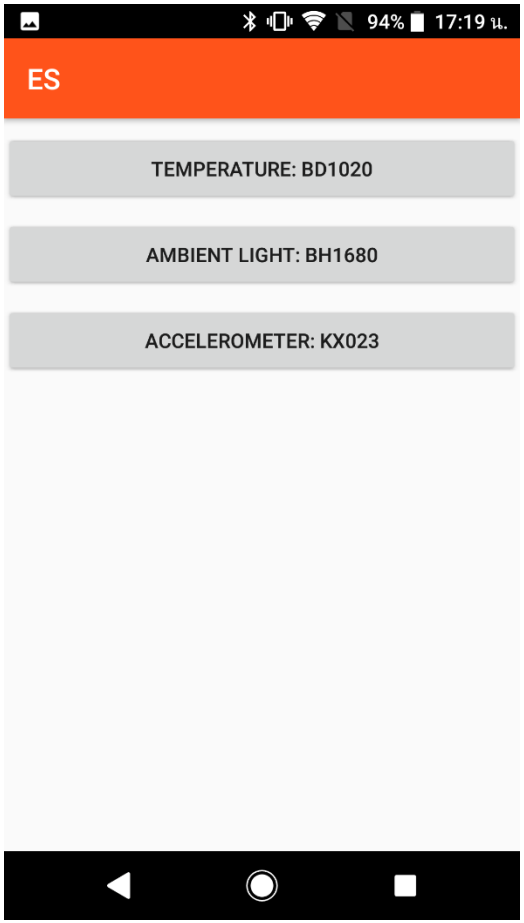
### Step 3 Choose device and connect




Pichet Sriphomma pichet@es.co.th

## Using the application

### Step 4 After connected this app Show button to choose



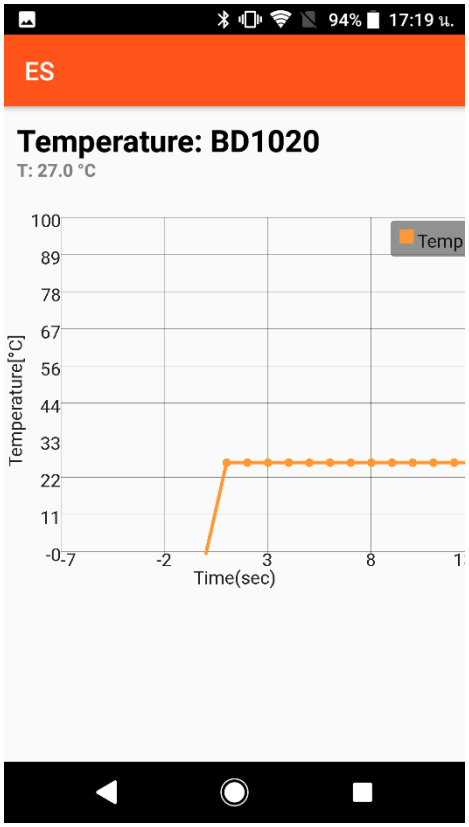



Pichet Sriphomma pichet@es.co.th

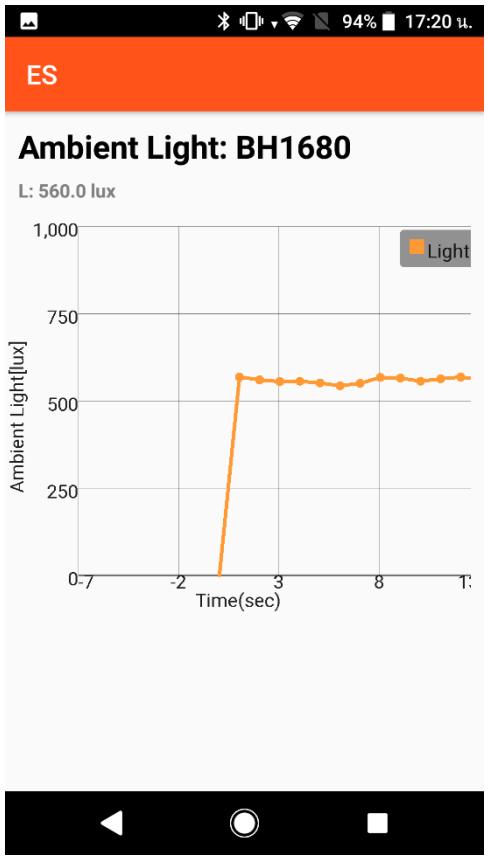
## Using the application

**Step 5 if you choose “Temperature : BD1020” this app show Signal graph Of BD1020 (picture 1) but you choose “ Ambient Light : BH1680” this app show signal graph of BH1680 (picture 2)**

1




2





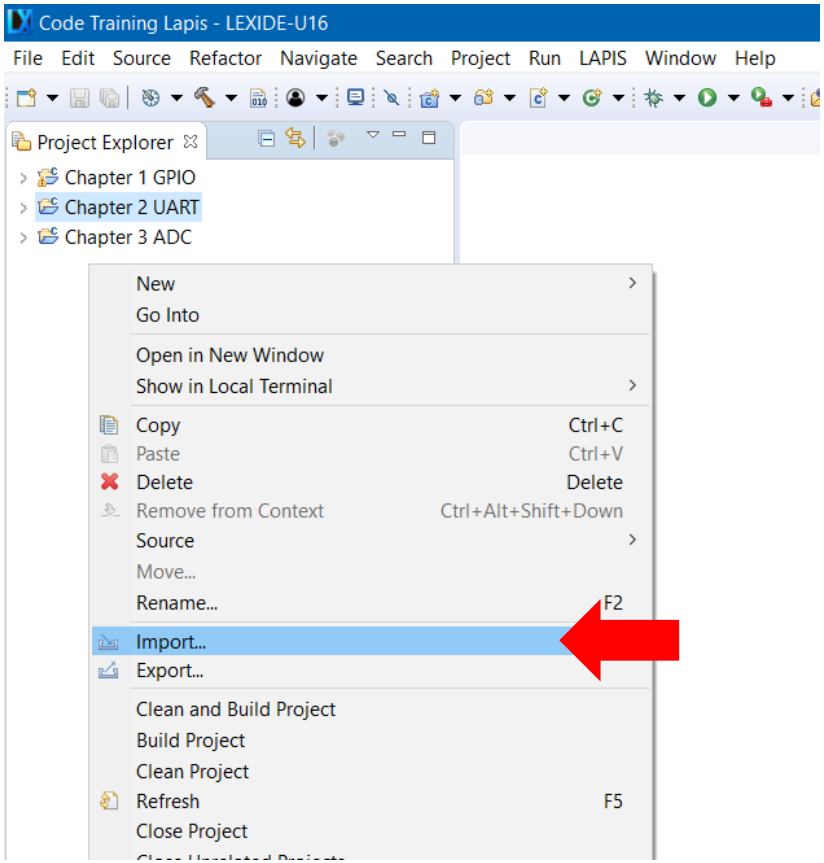

Pichet Sriphomma pichet@es.co.th

# 4. I²C



Pichet Sriphomma pichet@es.co.th

## Import Project

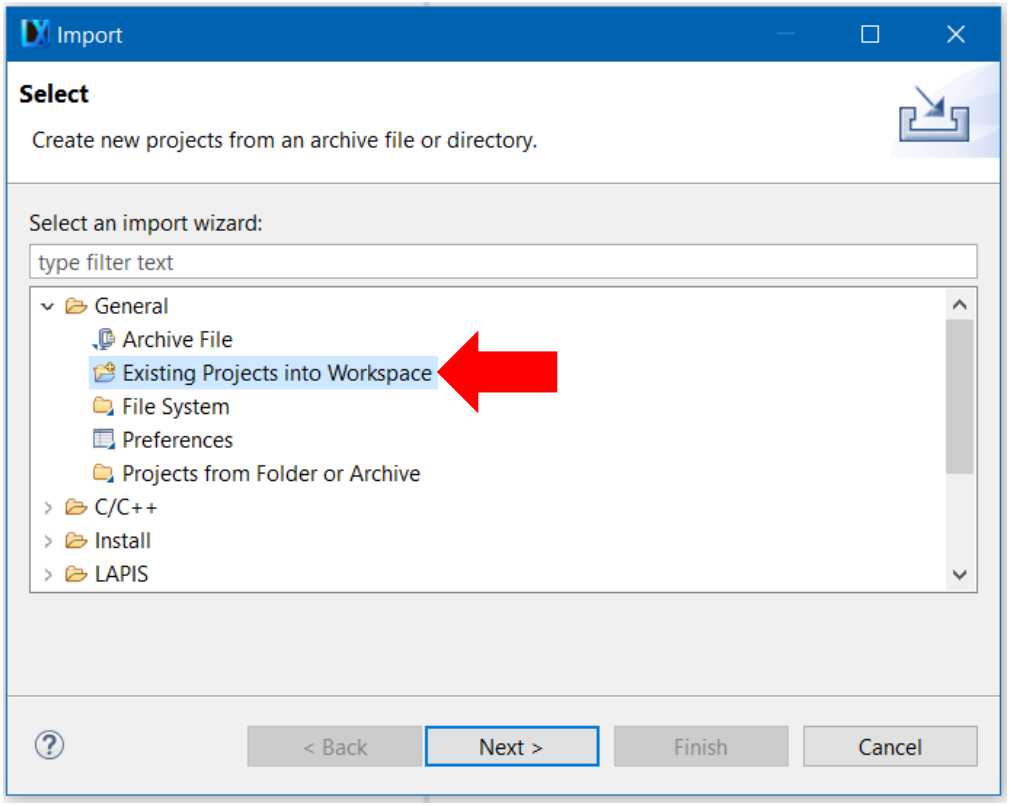



Right-click on project Explorer and select Import.



Pichet Sriphomma pichet@es.co.th

## Import Project

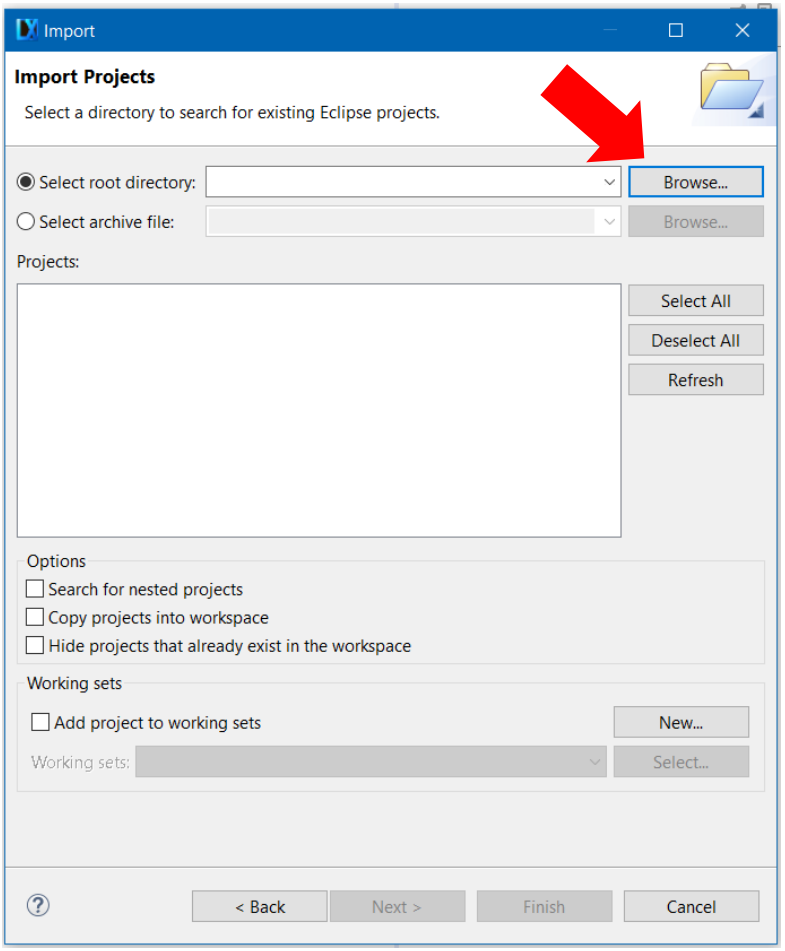



Select General and choose Existing Projects into Workspace. Click Next.

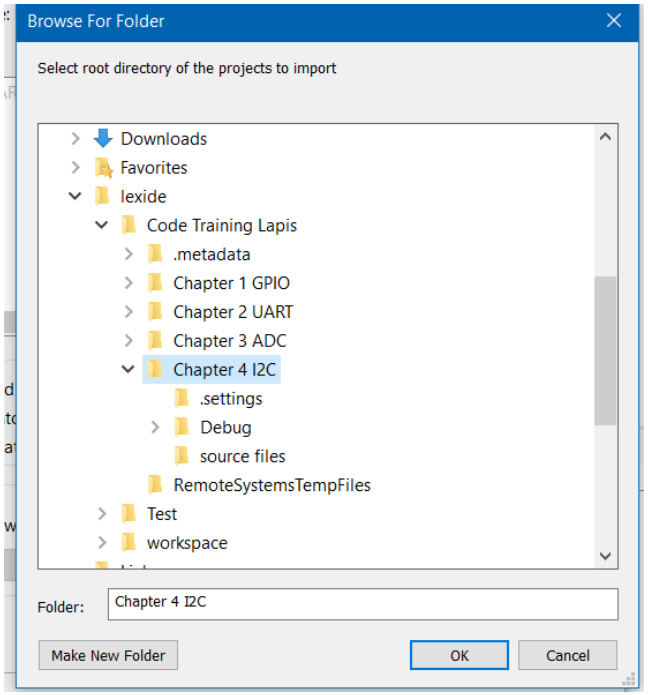


Pichet Sriphomma pichet@es.co.th

# Import Project



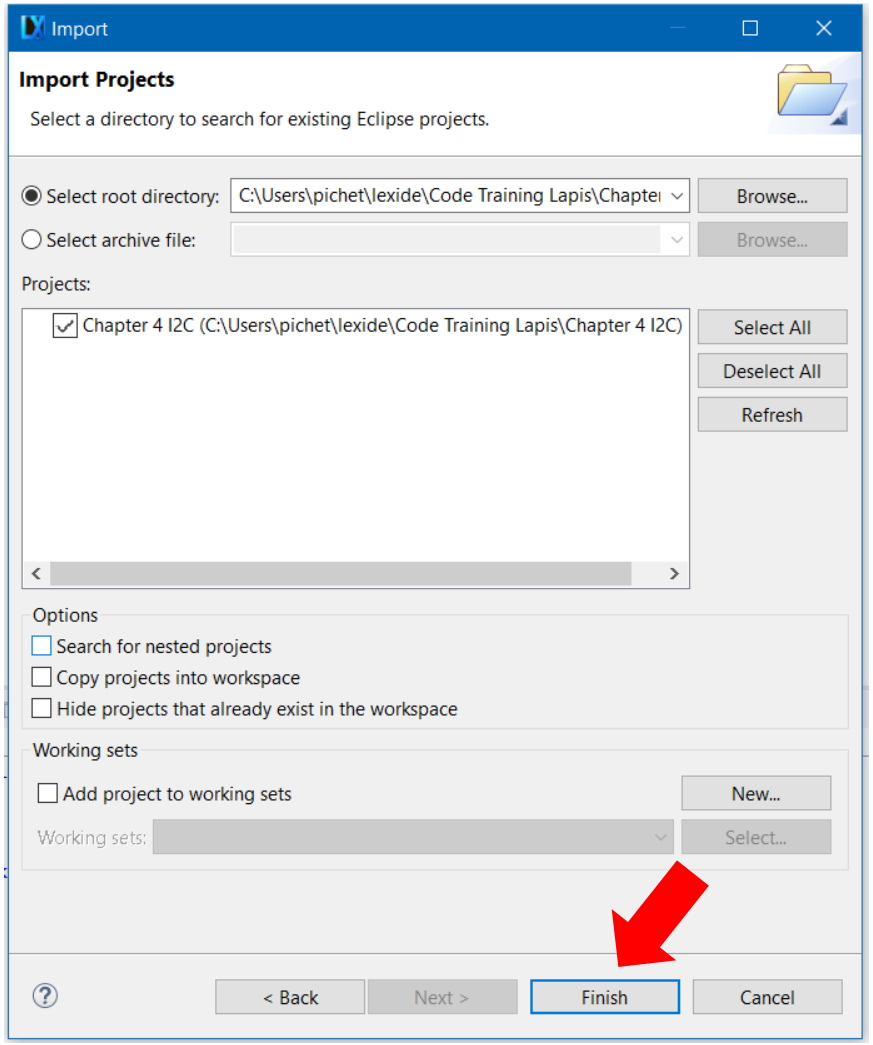

LEXIDE up the new window.
Click Browse.. at Select root directory.
Choose “Chapter 4 I2C” in
Folder window.






Pichet Sriphomma pichet@es.co.th

# Import Project

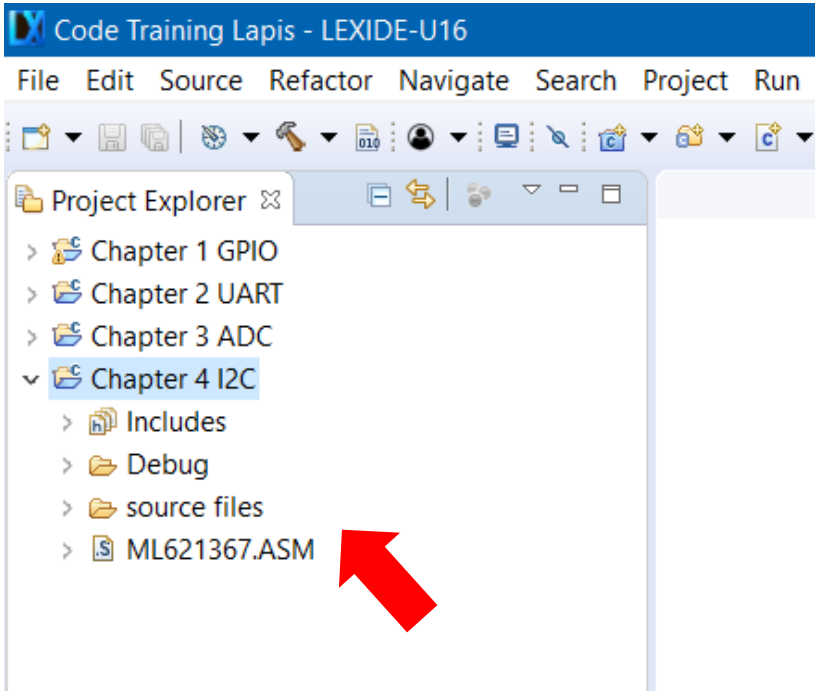



After choosing Project Click Finish.



Pichet Sriphomma pichet@es.co.th

## Import Project

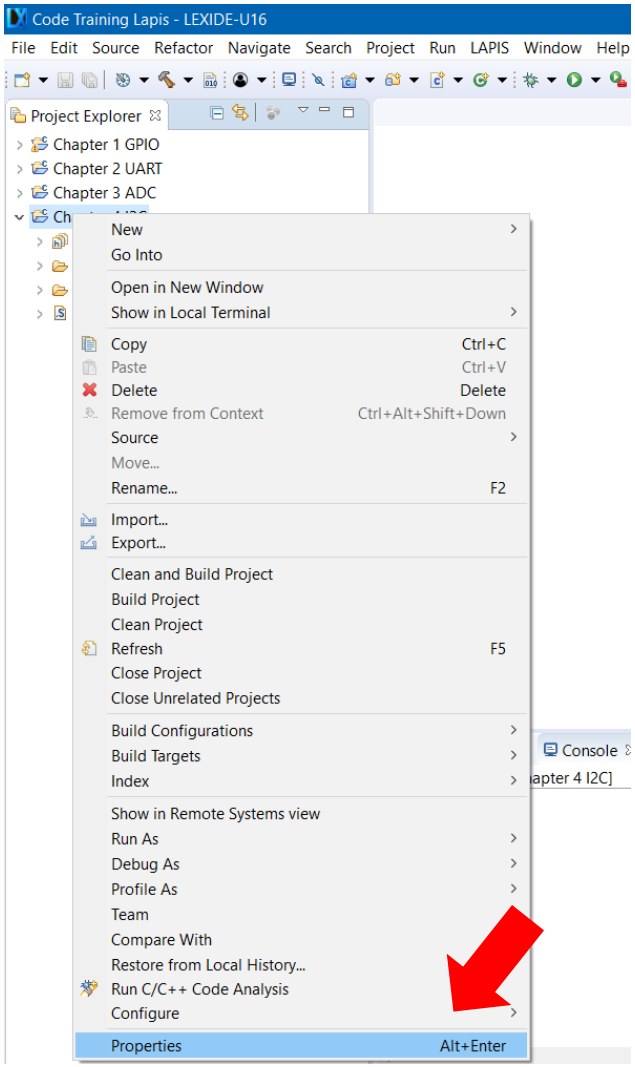



Then appear the project on Project Explorer.



Pichet Sriphomma pichet@es.co.th

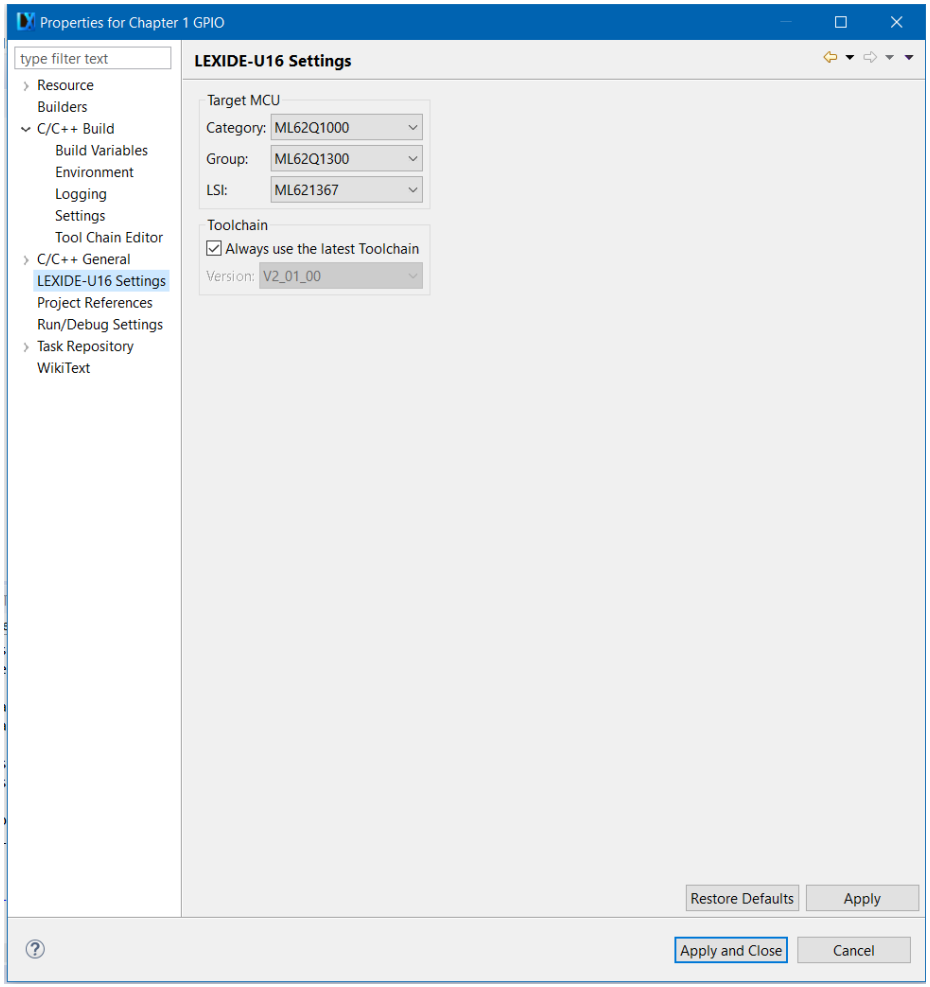
## Check Device




Right-click on a project folder and select [Properties] .

Pichet Sriphomma pichet@es.co.th

## Select Device

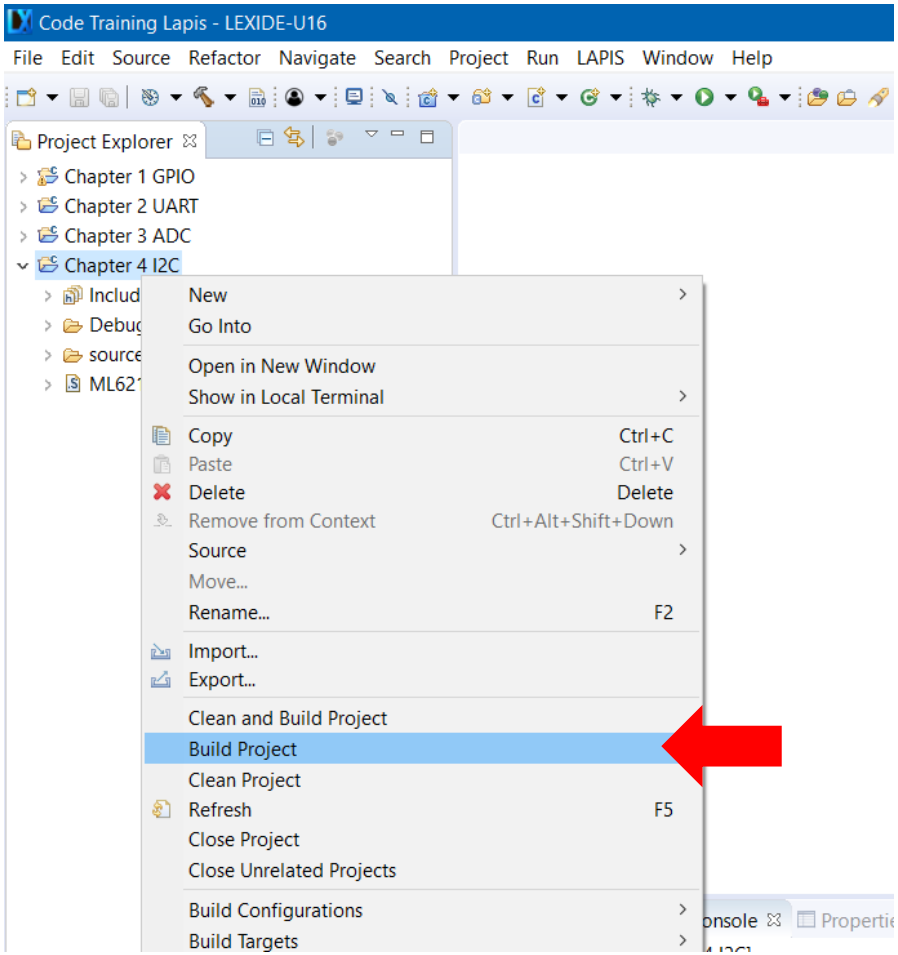



Choose LEXIDE-U16 Settings



Pichet Sriphomma pichet@es.co.th

## Build Project

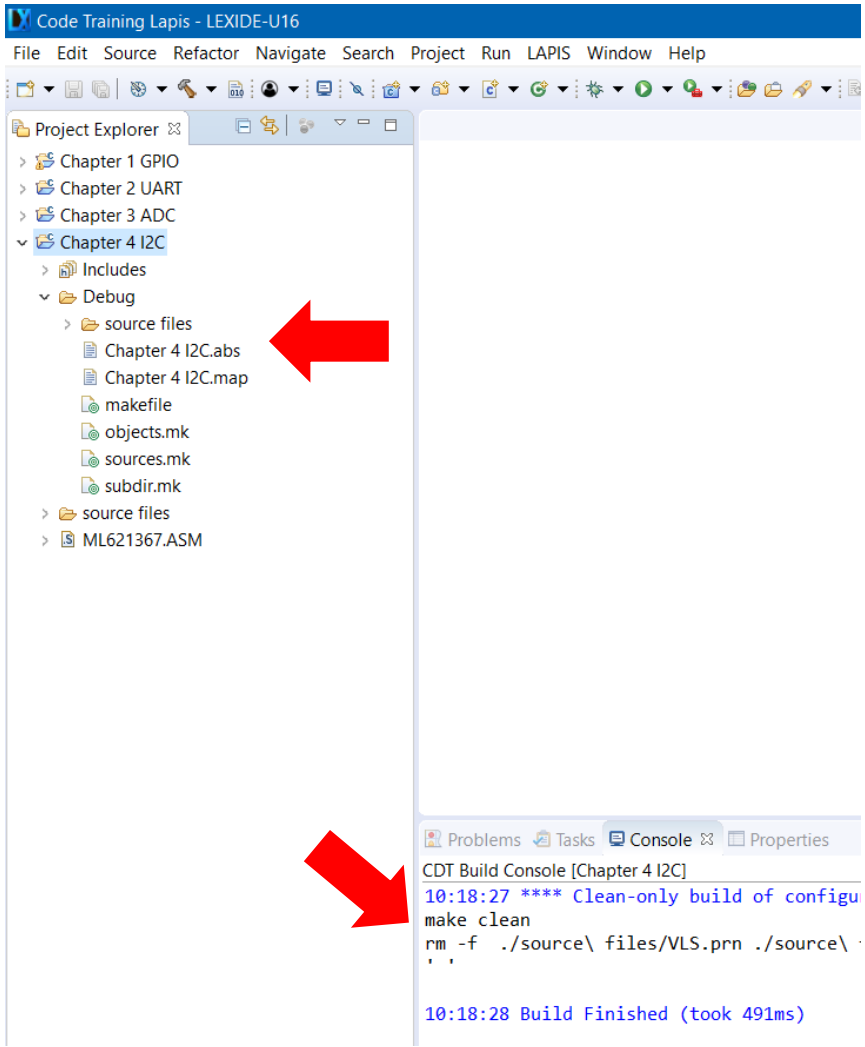



Right-click on a project folder and select [Build Project] to start the build process.



Pichet Sriphomma pichet@es.co.th

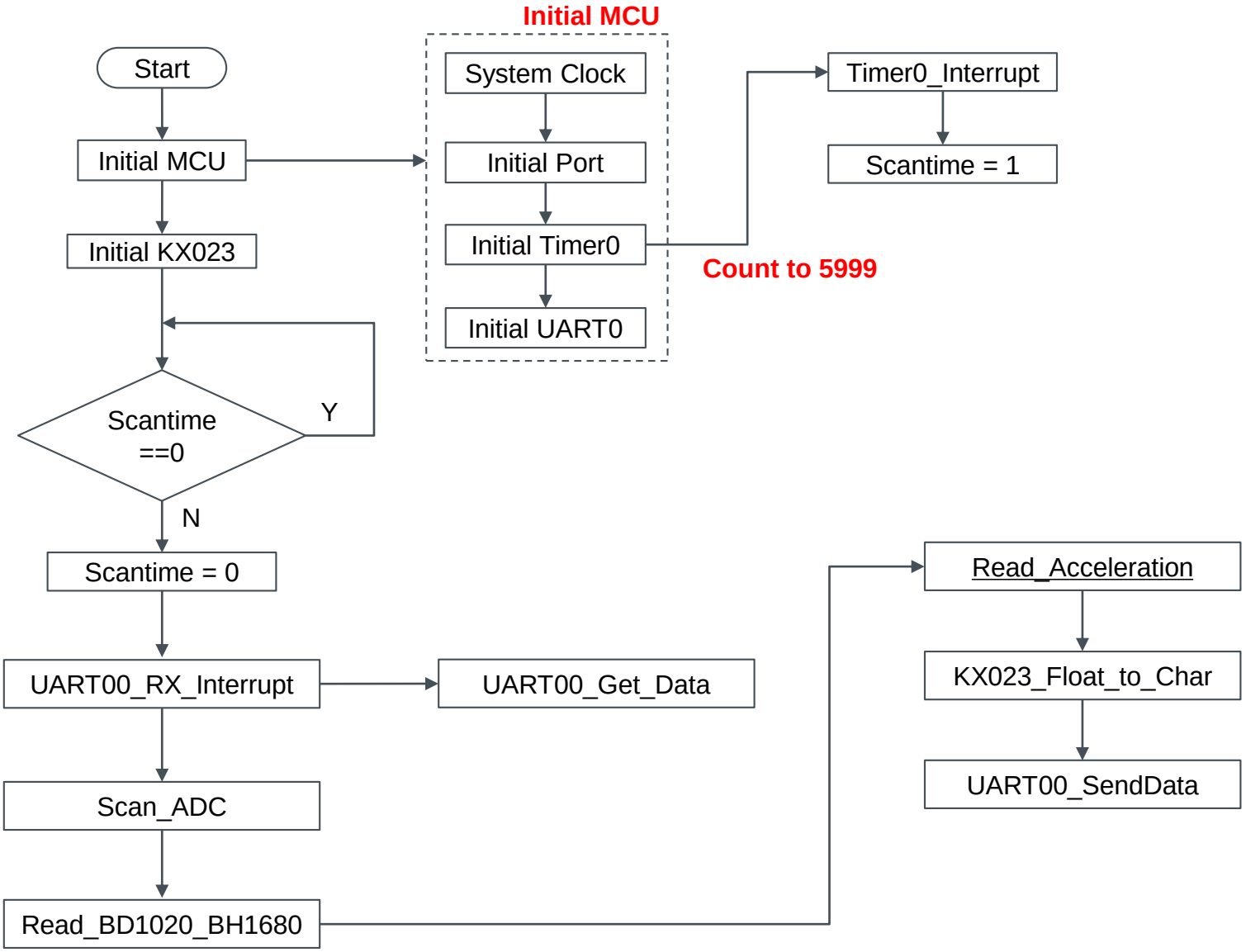
## Build Project




When the build succeeds , an ABS file is generated.



Pichet Sriphomma pichet@es.co.th




Pichet Sriphomma pichet@es.co.th

# Initial i2c on (i2c.c)

```
#define SDA_OUT                 P22DO
#define SDA_IN                  P22DI

#define SDA_DIR_IN              P22IE
#define SDA_DIR_OUT             P22OE

#define SCL_OUT                 P23DO
#define SCL_IN                  P23DI


#define SCL_DIR_IN              P23IE
#define SCL_DIR_OUT             P23OE
```

```
void initial_i2c_on(void){

    P22IE=0;P22OE=0;P22OD=0;P22PU=1;
    P23IE=0;P23OE=0;P23OD=0;P23PU=1;


    SDA_DIR_IN=0;
    SDA_DIR_OUT=1;
    SCL_DIR_OUT=1;
}
```

## 17.2.2 Port n Data Register (PnD:n＝0 to 9, A, B)

PnD is a special function register (SFR) used to read the level of the port n pin and write output data.
The input level of the port n pin can be read by reading PnDI in the input mode.
Data written to PnDO in the output mode are output to the port n pin.
The PnDO is readable.
Enable or disable the input or output by using the port n mode register.
See Table 17-2 “List of Registers / Bits” to check avaible pins and bits.
Wirte “0” to the bits of PnDO register that have no corresponding pin.
The bits of PnDI register that has no corresponding pin always return “0” for reading.

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | PnD | | | | | | | | | | | | | | | |
| Byte | PnDO | | | | | | | | PnDI | | | | | | | |
| Bit | Pn7DO | Pn6DO | Pn5DO | Pn4DO | Pn3DO | Pn2DO | Pn1DO | Pn0DO | Pn7DI | Pn6DI | Pn5DI | Pn4DI | Pn3DI | Pn2DI | Pn1DI | Pn0DI |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | R | R | R | R | R | R | R |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit No. | Bit symbol name | Description |
|---|---|---|
| 15 to 8 | Pn7DO to Pn0DO | This bit is used to set the output level of port n pin.<br>0: Output "L" (initial value)<br>1: Output "H" |
| 7 to 0 | Pn7DI to Pn0DI | This bit is used to set the input level of port n pin.<br>0: The input level is "L"<br>1: The input level is "H" (Initial value) |



Pichet Sriphomma pichet@es.co.th

## Initial i2c on (i2c.c)

```
#define SDA_OUT                 P22DO
#define SDA_IN                  P22DI

#define SDA_DIR_IN              P22IE
#define SDA_DIR_OUT             P22OE

#define SCL_OUT                 P23DO
#define SCL_IN                  P23DI


#define SCL_DIR_IN              P23IE
#define SCL_DIR_OUT             P23OE
```

```
void initial_i2c_on(void){

    P22IE=0;P22OE=0;P22OD=0;P22PU=1;
    P23IE=0;P23OE=0;P23OD=0;P23PU=1;


    SDA_DIR_IN=0;
    SDA_DIR_OUT=1;
    SCL_DIR_OUT=1;
}
```

### 17.2.3 Port n Mode Register 01 (PnMOD01:n=0 to 9, A, B)

PnMOD01 is a special function register (SFR) to choose the input/output mode, input/output status, and shared function of Pn0 pin and Pn1 pin.
See Table 17-2 "List of Registers / Bits" to check avaible pins and bits.
Wirte "0" to the bits of PnMOD01 register that have no corresponding pin.

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word | PnMOD01 | | | | | | | | | | | | | | | |
| Byte | PnMOD1 | | | | | | | | PnMOD0 | | | | | | | |
| Bit | Pn1MD3 | Pn1MD2 | Pn1MD1 | Pn1MD0 | Pn1OD | Pn1PU | Pn1OE | Pn1IE | Pn0MD3 | Pn0MD2 | Pn0MD1 | Pn0MD0 | Pn0OD | Pn0PU | Pn0OE | Pn0IE |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | * | 0 | * |

* :The initial value of P00IE and P00PU for the Port0 is "1" and other bits are "0".

**Pn1OD** This bit is used choose the output type of Pn1 pin.
An LED is directly drive-able by enlarging the current when the N-channel open drain output mode is chosen.
See the data sheet for details about the current drive ability.
0: CMOS output (initial value)
1: N-channel open drain output

**Pn1PU** This bit is used to enable the internal pull-up resistor of Pn1 pin.
The internal pull-up resistor can be enabled on following conditions of the port.
The input is enabled and the output is disabled on the port
The input is enabled and the N-channel open drain output is chosen on the port
0: Without a pull-up resistor (initial value)
1: With a pull-up resistor

The conditions of the port are specified by Pn1IE, Pn1OE and Pn1OD bit.
10X: Setting of Pn1PU bit is enable
111: Setting of Pn1PU bit is enable
Others: Setting of Pn1PU bit is disable
X: 0 or 1 (don't care)



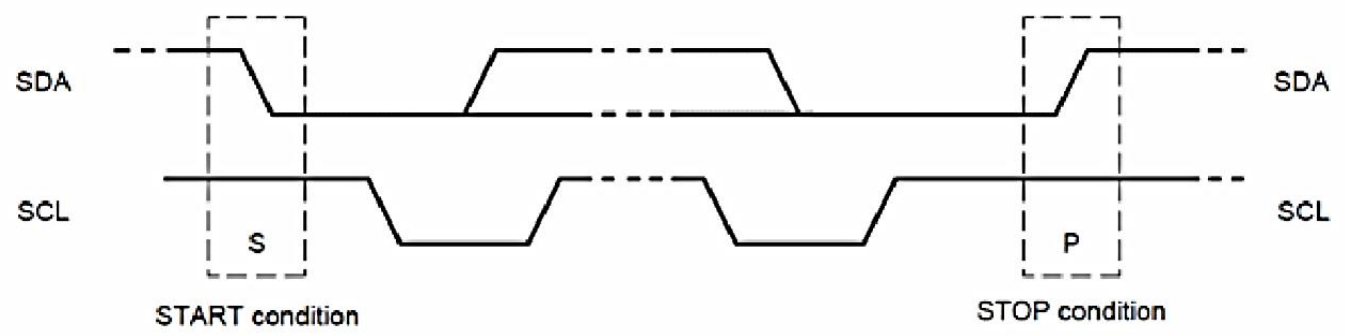
Pichet Sriphomma pichet@es.co.th

## i2c start (i2c.c)

```
void i2c_start(void){
    SDA_OUT=1;
    SCL_OUT=1;
    i2c_delay();
    SDA_OUT=0;
    i2c_delay();
    SCL_OUT=0;
}
```

## i2c stop (i2c.c)

```
void i2c_stop(void){
    i2c_delay();
    SDA_OUT=0;
    i2c_delay();
    SCL_OUT=1;
    i2c_delay();
    SDA_OUT=1;
}
```






Pichet Sriphomma pichet@es.co.th

## i2c write (i2c.c)

```
void i2c_write(unsigned char dat){
unsigned char i;
    Ack=1;
    for(i=1;i<=8;i++){
        if((dat&0x80)!=0)
        {
            SDA_OUT=1;
        }else{SDA_OUT=0;}
        i2c_clk();
        dat=dat<<1;
    }
    //check Ack
    SDA_DIR_OUT=0;
    SDA_DIR_IN=1;

    i2c_delay();
    SCL_OUT=1;
    i2c_delay();
    if(SDA_IN){Ack=0;};
    SCL_OUT=0;
    SDA_DIR_IN=0;
    SDA_DIR_OUT=1;

}
```

ACK = 1

If bit 7 of dat != 0 SDA = 1

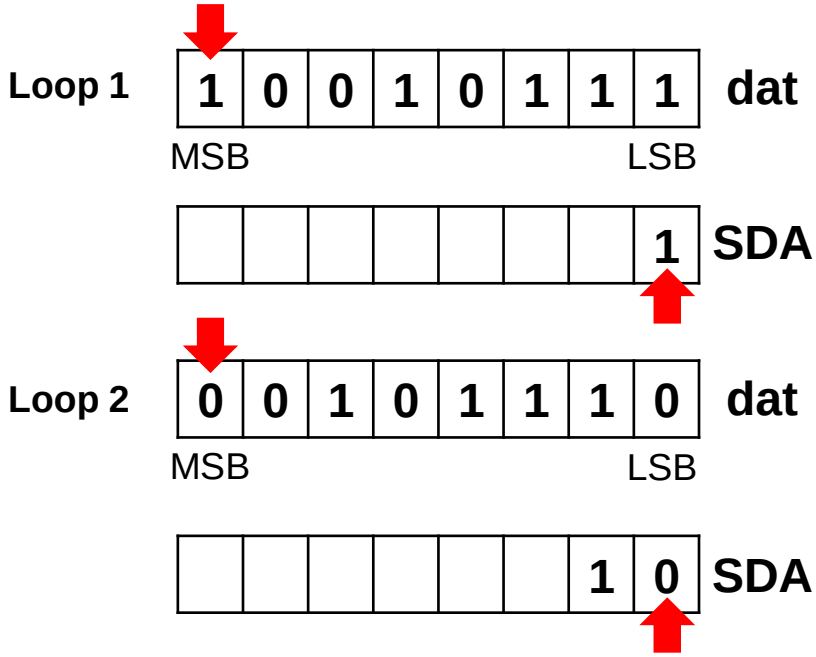
,If bit 7 of dat = 0 SDA = 0

After that shift bit.

Enable Input SDA

Check ACK

Enable Output SDA




| Master | S | SAD + W | | RA | | DATA | | P |
|---|---|---|---|---|---|---|---|---|
| Slave | | | ACK | | ACK | | ACK | |

## i2c read (i2c.c)

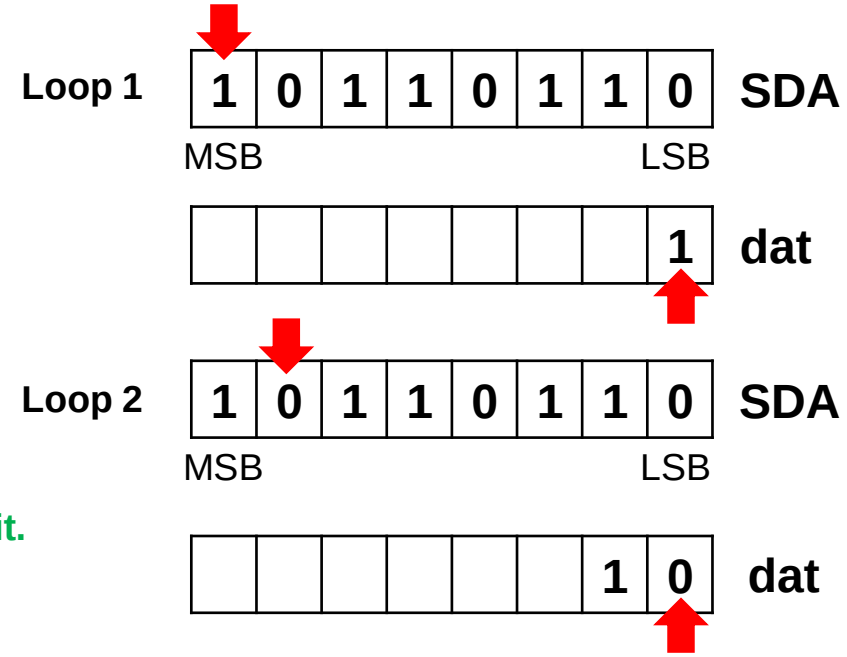
```
unsigned char i2c_read(void){
    unsigned char i;
    unsigned char dat=0;
    SDA_DIR_OUT=0;
    SDA_DIR_IN=1;

    for(i=0;i<8;i++){
        SCL_OUT=1;
        dat=dat<<1;
        if(SDA_IN){dat|=1;}
        i2c_delay();
        SCL_OUT=0;
        i2c_delay();
    }
    SDA_DIR_IN=0;
    SDA_DIR_OUT=1;
    i2c_delay();

    return(dat);
}
```

Enable Input SDA

Read SDA_IN ,
send values to each
bit of dat and shift bit.

Enable Output SDA

Return dat

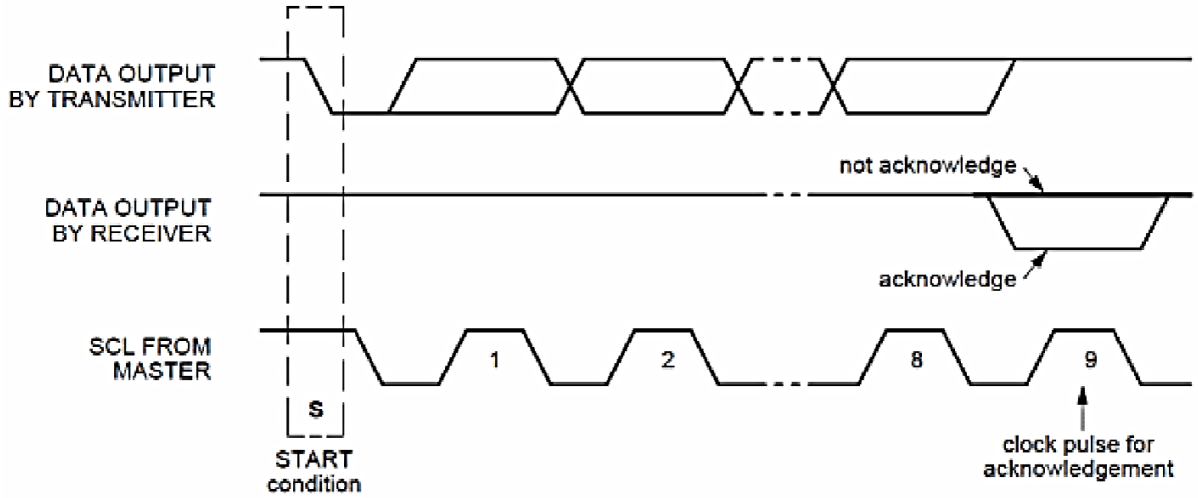



| Master | S | SAD + W | | RA | | Sr | SAD + R | | | NACK | P |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Slave | | | ACK | | ACK | | | ACK | DATA | | |

## i2c ACK NACK (i2c.c)

```
void i2c_ack_bit(void){
    SDA_OUT=0;
    i2c_delay();
    i2c_clk();
    SDA_OUT=1;
}
void i2c_nack_bit(void){
    SDA_OUT=1;
    i2c_delay();
    i2c_clk();
    SCL_OUT=1;
}
```

I2C Bus Specification






Pichet Sriphomma pichet@es.co.th

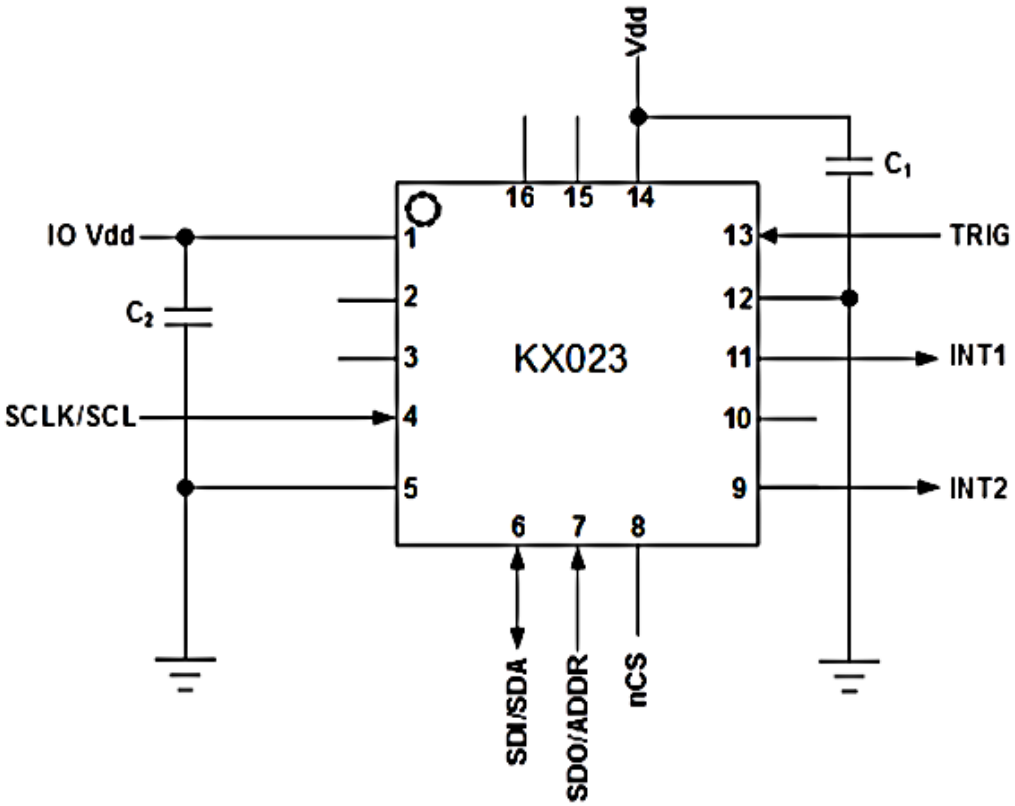
## KX023

1. tri-axis +/-2g, +/-4g or +/-8g accelerometer
2. Enhanced integrated Directional Tap/Double-TapTM , and Device-orientation Algorithms
3. Digital I2C up to 3.4 MHz
4. Digital 3-wire and 4-wire SPI up to 10 MHz
5. Self-test Function



Pichet Sriphomma pichet@es.co.th

## KX023




| Pin | Name |
|---|---|
| 1 | IO Vdd |
| 2 | NC |
| 3 | NC |
| 4 | SCLK/SCL |
| 5 | GND |
| 6 | SDI/SDA |
| 7 | SDO/ADDR |
| 8 | nCS |
| 9 | INT2 |
| 10 | NC |
| 11 | INT1 |
| 12 | GND |
| 13 | TRIG |
| 14 | Vdd |
| 15 | NC |
| 16 | NC |

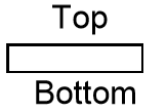
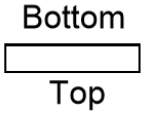


Pichet Sriphomma pichet@es.co.th

## KX023

**Static X/Y/Z Output Response versus Orientation to Earth's surface (1g):**

GSEL1=0, GSEL0=0 (± 2g)

| Position | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Diagram |  | |  | |  | |  | |  | |  | |
| Resolution (bits) | 16 | 8 | 16 | 8 | 16 | 8 | 16 | 8 | 16 | 8 | 16 | 8 |
| X (counts) | 0 | 0 | -16384 | -64 | 0 | 0 | 16384 | 64 | 0 | 0 | 0 | 0 |
| Y (counts) | -16384 | -64 | 0 | 0 | 16384 | 64 | 0 | 0 | 0 | 0 | 0 | 0 |
| Z (counts) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 16384 | 64 | -16384 | -64 |
| | | | | | | | | | | | | |
| X-Polarity | **0** | | **-** | | **0** | | **+** | | **0** | | **0** | |
| Y-Polarity | **-** | | **0** | | **+** | | **0** | | **0** | | **0** | |
| Z-Polarity | **0** | | **0** | | **0** | | **0** | | **+** | | **-** | |

$$\text{Static output response} = \text{output value} \times \text{g\_ratio} = -16384 \times \frac{2}{32768} = -1$$



Pichet Sriphomma pichet@es.co.th

## KX023 I2C Operation

The Slave Address associated with the KX023 is 001111X.

X is determined by the assignment of ADDR (pin 7) to GND or IO_Vdd.

| Term | Definition |
|---|---|
| S | Start Condition |
| Sr | Repeated Start Condition |
| SAD | Slave Address |
| W | Write Bit |
| R | Read Bit |
| ACK | Acknowledge |
| NACK | Not Acknowledge |
| RA | Register Address |
| Data | Transmitted/Received Data |
| P | Stop Condition |



Pichet Sriphomma pichet@es.co.th

## Writing to a KX023 8-bit Register

1. Master send Start condition (S) and SAD+W ,and the KX023 acknowledges. The KX023 return ACK.
2. After ACK Transmission. An 8-bit Register Address (RA) command is transmitted by the Master. The KX023 return ACK.
3. After Master receive ACK from KX023. Master Send DATA to KX023.
4. The KX023 Return ACK. ,and Waiting to receive Stop condition (P) from Master.

**Sequence 1.** The Master is writing one byte to the Slave.

| Master | S | SAD + W | | RA | | DATA | | P |
|---|---|---|---|---|---|---|---|---|
| Slave | | | ACK | | ACK | | ACK | |

**Sequence 2.** The Master is writing multiple bytes to the Slave.

| Master | S | SAD + W | | RA | | DATA | | DATA | | P |
|---|---|---|---|---|---|---|---|---|---|---|
| Slave | | | ACK | | ACK | | ACK | | ACK | |



Pichet Sriphomma pichet@es.co.th

## Reading from a KX023 8-bit Register

1. The Master first transmits a start condition (S) and the appropriate Slave Address (SAD) with the LSB set at ‘0’ to write. The KX023 return ACK.
2 The KX023 acknowledges and the Master transmits the 8-bit RA of the register it wants to read.
3. The KX023 again acknowledges, and the Master transmits a repeated start condition (Sr).
4. After the repeated start condition, the Master addresses the KX023 with a ‘1’ in the LSB (SAD+R) to read.
5. The KX023 acknowledges and transmits the data from the requested register.
6. The Master does not acknowledge (NACK) it received the transmitted data
, and transmits a stop condition to end the data transfer.

**Sequence 3.** The Master is receiving one byte of data from the Slave.

| Master | S | SAD + W | | RA | | Sr | SAD + R | | | NACK | P |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Slave | | | ACK | | ACK | | | ACK | DATA | | |

**Sequence 4.** The Master is receiving multiple bytes of data from the Slave.

| Master | S | SAD + W | | RA | | Sr | SAD + R | | | ACK | | NACK | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Slave | | | ACK | | ACK | | | ACK | DATA | | DATA | | |

Pichet Sriphomma pichet@es.co.th

## KX023 Embedded Registers

The KX023 has 57 embedded 8-bit registers that are accessible by the user.

XHPL,XHPH,YHPL,YHPH,ZHPL,ZHPH high pass filter accelerometer output.
Data is updated at the ODR frequency determined by OWUF in CNTL3.

| Address | Register Name | R/W |
|---|---|---|
| 00h | XHPL | R |
| 01h | XHPH | R |
| 02h | YHPL | R |
| 03h | YHPH | R |
| 04h | ZHPL | R |
| 05h | ZHPH | R |



Pichet Sriphomma pichet@es.co.th

## KX023 Embedded Registers

XOUTL,XOUTH,YOUTL,YOUTH,ZOUTL,ZOUTH accelerometer output .
Data is updated at the ODR frequency determined by OSA in ODCNTL.

| Address | Register Name | R/W |
|---|---|---|
| 06h | XOUTL | R |
| 07h | XOUTH | R |
| 08h | YOUTL | R |
| 09h | YOUTH | R |
| 0Ah | ZOUTL | R |
| 0Bh | ZOUTH | R |



Pichet Sriphomma pichet@es.co.th

## KX023 Embedded Registers

COTR  This register can be used to verify proper integrated circuit functionality. It always has a byte value of 0x55h

WHO_AM_I  This register can be used for supplier recognition, as it can be factory written to a known byte value.  The default value is 0x15h.

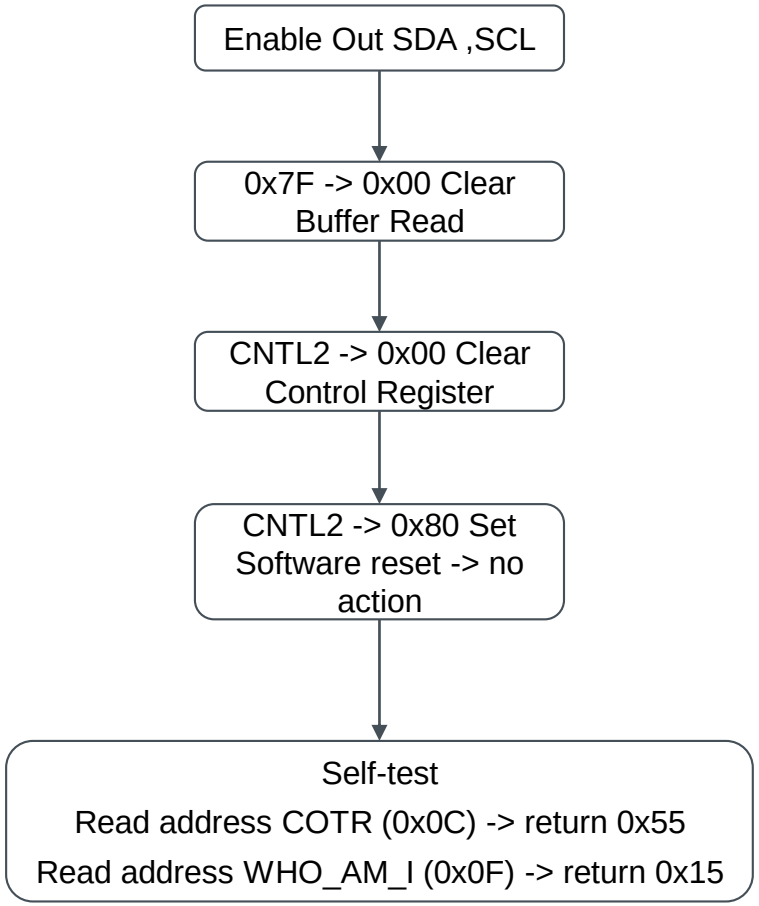
| Address | Register Name | R/W |
|---|---|---|
| 0Ch | COTR | R |
| 0Dh | Kionix Reserved | |
| 0Eh | Kionix Reserved | |
| 0Fh | Who_AM_I | R/W |



Pichet Sriphomma pichet@es.co.th

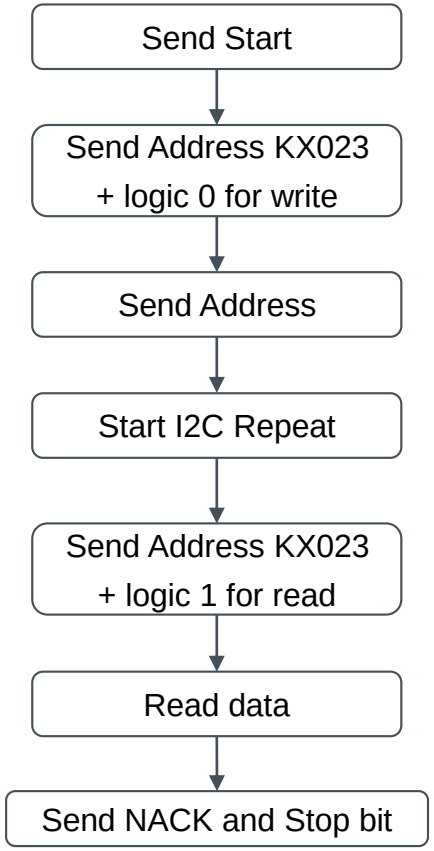
## Initial KX023 (kx023.c)

```
void Initial_kx023(void)
{
    unsigned char i[1]={0};
    unsigned int cnt;
    initial_i2c_on();


    //**wait for more than 0.1ms
    for (cnt=0;cnt<1600;cnt++)
    {
        __asm("nop");
    }

    write_kx(0x7F,0x00);
    write_kx(CNTL2,0x00);
    write_kx(CNTL2,0x80);

    //** wait for more than 2ms
    for (cnt=0;cnt<32000;cnt++)
    {
        __asm("nop");
    }
    i[0]=read_kx(WHO_AM_I);

    i[0]=read_kx(add_COTR);         //i =0x55


    G_ratio = (((float)G_range)/((float)N_dectimal));

    KX112_Start();
}
```



## Read KX023 (kx023.c)

```
unsigned char read_kx(unsigned char addr){
  unsigned char data;
  i2c_start();
  i2c_write(kx_Addr);
  i2c_write(addr);
  i2c_start();
  i2c_write(kx_Addr+1);
  data=i2c_read();
  i2c_nack_bit();
  i2c_stop();
  return(data);
}
```



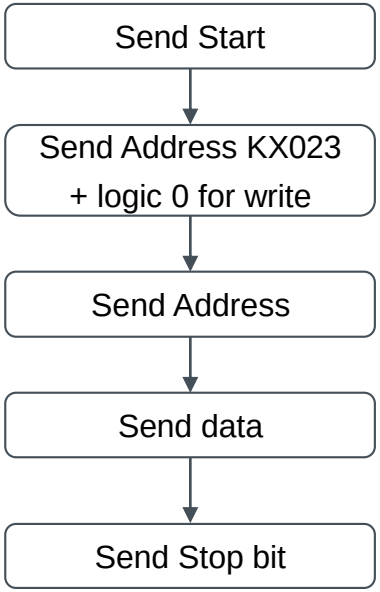

| Master | S | SAD + W | | RA | | Sr | SAD + R | | | NACK | P |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Slave | | | ACK | | ACK | | | ACK | DATA | | |



Pichet Sriphomma pichet@es.co.th

## Write KX023 (kx023.c)

```
void write_kx(unsigned char addr,unsigned char data){

  unsigned char read;
  i2c_start();
  i2c_write(kx_Addr);
  i2c_write(addr);
  i2c_write(data);
  i2c_stop();

}
```



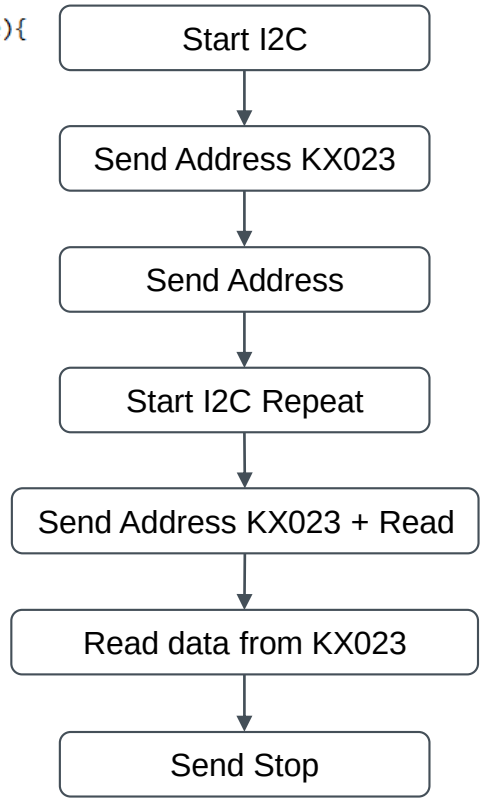

| Master | S | SAD + W | | RA | | DATA | | P |
|---|---|---|---|---|---|---|---|---|
| Slave | | | ACK | | ACK | | ACK | |



Pichet Sriphomma pichet@es.co.th

## MulRead KX023 (kx023.c)

```
unsigned char Mulread_kx(unsigned char addr,unsigned char *data,unsigned int size){

  unsigned int i,DataLen;

  //initial_i2c_on();

  DataLen = size-1;

  i2c_start();
  i2c_write(kx_Addr);
  i2c_write(addr);
  i2c_start();
  i2c_write(kx_Addr+1);
  for(i=0;i<DataLen;i++){
    rdata[i] =i2c_read();
    i2c_ack_bit();
  }

  rdata[DataLen] =i2c_read();
  i2c_nack_bit();
  i2c_stop();

  for(i=0;i<=size;i++){
        *(data+i) = rdata[i];
  }

  return 0;
 // initial_i2c_off();
}
```

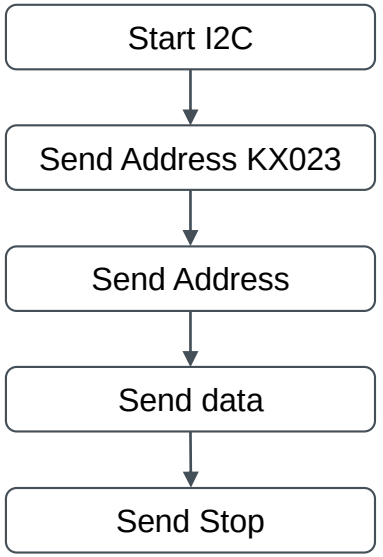





Pichet Sriphomma pichet@es.co.th

## Mulwrite KX023 (kx023.c)

```
void Mulwrite_kx(unsigned char addr,unsigned char *data, unsigned int size){
  unsigned int i,DataLen;

  DataLen = size;

  //initial_i2c_on();

  for(i=0;i<size;i++){
    sdata[i] = *(data+i);
  }

  i2c_start();
  i2c_write(kx_Addr);
  i2c_write(addr);

  for(i=0;i<DataLen;i++){
    i2c_write(sdata[i]);
  }
  i2c_stop();

  //initial_i2c_off();

}
```



## KX023 On Off (kx023.c)

```
void Initial_kx_on(void)
{
     initial_i2c_on();
}

void Initial_kx_off(void)
{
     initial_i2c_off();
}
```

KX023 On

Pull – up

SDA Enable Output

SCL Enable Output

KX023 Off

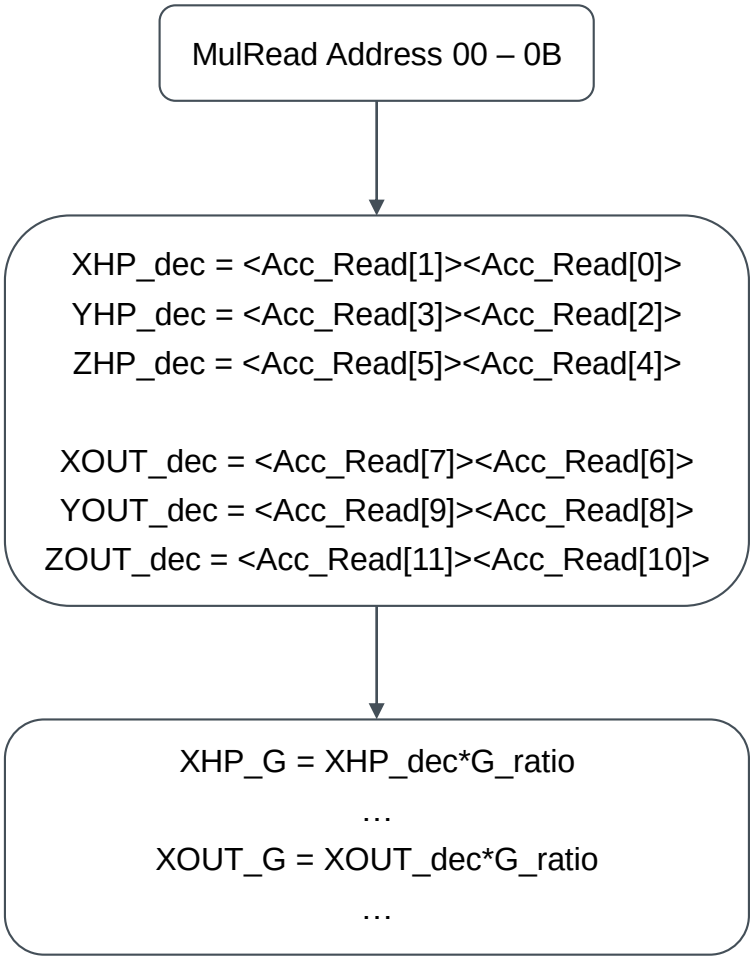
Not Pull – up



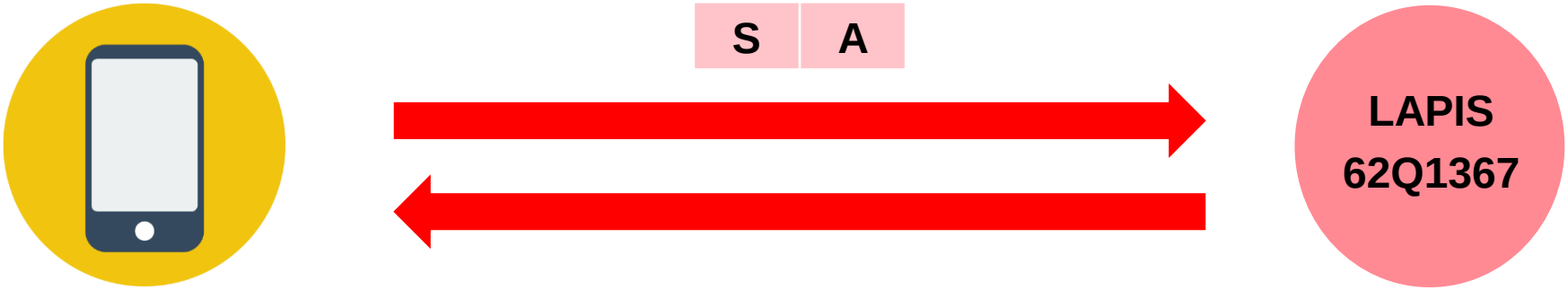
Pichet Sriphomma pichet@es.co.th

## Read Acceleration (kx023.c)

```
void Read_Acceleration(void){

  Mulread_kx(0,Acc_Read,13);
  XHP_dec = (Acc_Read[1]<<8)|Acc_Read[0];
  YHP_dec = (Acc_Read[3]<<8)|Acc_Read[2];
  ZHP_dec = (Acc_Read[5]<<8)|Acc_Read[4];
  XOUT_dec = (Acc_Read[7]<<8)|Acc_Read[6];
  YOUT_dec = (Acc_Read[9]<<8)|Acc_Read[8];
  ZOUT_dec = (Acc_Read[11]<<8)|Acc_Read[10];

  XHP_G = XHP_dec*G_ratio;
  YHP_G = YHP_dec*G_ratio;
  ZHP_G = ZHP_dec*G_ratio;

  XOUT_G = XOUT_dec*G_ratio;
  YOUT_G = YOUT_dec*G_ratio;
  ZOUT_G = ZOUT_dec*G_ratio;
  //convert signed2unsigned
  // XOUT_uint = (~((int)XOUT_dec))&0x7fff;
  // YOUT_uint = (~((int)YOUT_dec))&0x7fff;
}
```



## Concept




XHP,YHP,ZHP < 0,1 >< decimal ><decimal>



Pichet Sriphomma pichet@es.co.th

## KX023_Float_to_Char (uart0.c)

```
void KX023_Float_to_Char(void)
{
    //*************** Accelerometer Sensor *************
1   kx023_XHP_G_float = XOUT_G;

    if(kx023_XHP_G_float < 0)
    {
2       kx023_XHP_G_float *= -1;
3       kx023_XHP_G_int = kx023_XHP_G_float;
4       kx023_XHP_G_char = kx023_XHP_G_int | 0x80;

    }
    else
    {
        kx023_XHP_G_int = kx023_XHP_G_float;
        kx023_XHP_G_char = kx023_XHP_G_int & 0x7F;
    }

        //kx023_XHP_G_float = kx023_XHP_G_float - kx023_XHP_G_int;
        //kx023_XHP_G_point = kx023_XHP_G_float*10000;
5       kx023_XHP_G_point = (kx023_XHP_G_float - kx023_XHP_G_int)*10000;
6       kx023_XHP_G_Point_Char_High = (kx023_XHP_G_point & 0xFF00)>>8;
7       kx023_XHP_G_Point_Char_Low = kx023_XHP_G_point & 0x00FF;
```

**Example**

**Value of KX023 (XOUT_G) = -0.1227**

**1. XHP_G_float = -0.1227**

**2. XHP_G_float = 0.1227**

**3. XHP_G_int = 0x00**

**4. XHP_G_Char = 0x80**

**5. XHP_G_point = (0.1227 – 0)*10000 = 1227**

**6. XHP_G_Point_Char_High = 0x1227 & 0xFF00**

**= 0x1200**

**= 0x1200 >> 8 = 0x0012**

**7. XHP_G_Point_Char_Low = 0x1227 & 0x00FF**

**= 0x0027**

## KX023_Float_to_Char (uart0.c)

```
//*****************************************

1 kx023_YHP_G_float = YOUT_G;

  if(kx023_YHP_G_float < 0)
  {
      kx023_YHP_G_float *= -1;
      kx023_YHP_G_int = kx023_YHP_G_float;
      kx023_YHP_G_char = kx023_YHP_G_int | 0x80;
  }
  else
  {
2     kx023_YHP_G_int = kx023_YHP_G_float;
3     kx023_YHP_G_char = kx023_YHP_G_int & 0x7F;
  }

4     kx023_YHP_G_point = (kx023_YHP_G_float - kx023_YHP_G_int)*10000;
5     kx023_YHP_G_Point_Char_High = (kx023_YHP_G_point & 0xFF00)>>8;
6     kx023_YHP_G_Point_Char_Low = kx023_YHP_G_point & 0x00FF;

//*****************************************
```

**Example**

**Value of KX023 (YOUT_G) = 0.5325**

**1. YHP_G_float = 0.5325**

**2. YHP_G_int = 0**

**3. YHP_G_Char = 0x00**

**4. YHP_G_point = (0.5325 – 0)*10000 = 5325**

**5. YHP_G_Point_Char_High = 0x5325 & 0xFF00**

**= 0x5300**

**= 0x5300 >> 8 = 0x0053**

**6. YHP_G_Point_Char_Low = 0x5325 & 0x00FF**

**= 0x0025**

## KX023_Float_to_Char (uart0.c)

```
//*****************************************

1  kx023_ZHP_G_float = ZOUT_G;

   if(kx023_ZHP_G_float < 0)
   {
2      kx023_ZHP_G_float *= -1;
3      kx023_ZHP_G_int = kx023_ZHP_G_float;
4      kx023_ZHP_G_char = kx023_ZHP_G_int | 0x80;
   }
   else
   {
       kx023_ZHP_G_int = kx023_ZHP_G_float;
       kx023_ZHP_G_char = kx023_ZHP_G_int & 0x7F;
   }

5      kx023_ZHP_G_point = (kx023_ZHP_G_float - kx023_ZHP_G_int)*10000
6      kx023_ZHP_G_Point_Char_High = (kx023_ZHP_G_point & 0xFF00)>>8;
7      kx023_ZHP_G_Point_Char_Low = kx023_ZHP_G_point & 0x00FF;
```

**Example**

**Value of KX023 (ZOUT_G) = -0.0827**

**1. ZHP_G_float = -0.0827**

**2. ZHP_G_float = 0.0827**

**3. ZHP_G_int = 0x00**

**4. ZHP_G_Char = 0x80**

**5. ZHP_G_point = (0.0827 – 0)*10000 = 827**

**6. ZHP_G_Point_Char_High = 0x0827 & 0xFF00**

**= 0x0800**

**= 0x0800 >> 8 = 0x0008**

**7. ZHP_G_Point_Char_Low = 0x0827 & 0x00FF**

**= 0x0027**

## KX023_Float_to_Char (uart0.c)

```
void UART00_KX023(void)
{
    Sensor_Index = 0;

        Clear_Buffer();

        UART00_TX_Buf[0] = kx023_XHP_G_char;
        UART00_TX_Buf[1] = kx023_XHP_G_Point_Char_High;
        UART00_TX_Buf[2] = kx023_XHP_G_Point_Char_Low;

        UART00_TX_Buf[3] = kx023_YHP_G_char;
        UART00_TX_Buf[4] = kx023_YHP_G_Point_Char_High;
        UART00_TX_Buf[5] = kx023_YHP_G_Point_Char_Low;

        UART00_TX_Buf[6] = kx023_ZHP_G_char;
        UART00_TX_Buf[7] = kx023_ZHP_G_Point_Char_High;
        UART00_TX_Buf[8] = kx023_ZHP_G_Point_Char_Low;

        TXD00_Index = 0;
        Flag._TXD00 = 1;
        Sensor_Index = 2;
        Set_UART00_TX();
    //}

    RXD00_Index = 0;
    //Flag._RXD00 = 0;
}
```

**Example**

**UART00_TX_Buf[0] = 0x80**

**UART00_TX_Buf[1] = 0x12**

**UART00_TX_Buf[2] = 0x27**

**UART00_TX_Buf[3] = 0x00**

**UART00_TX_Buf[4] = 0x53**

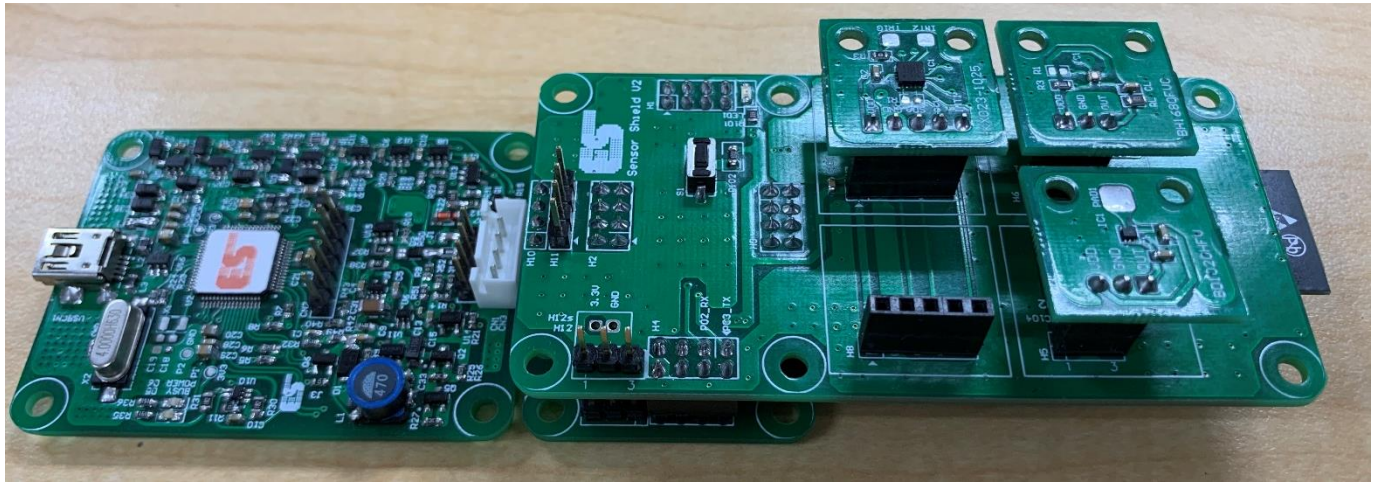
**UART00_TX_Buf[5] = 0x25**

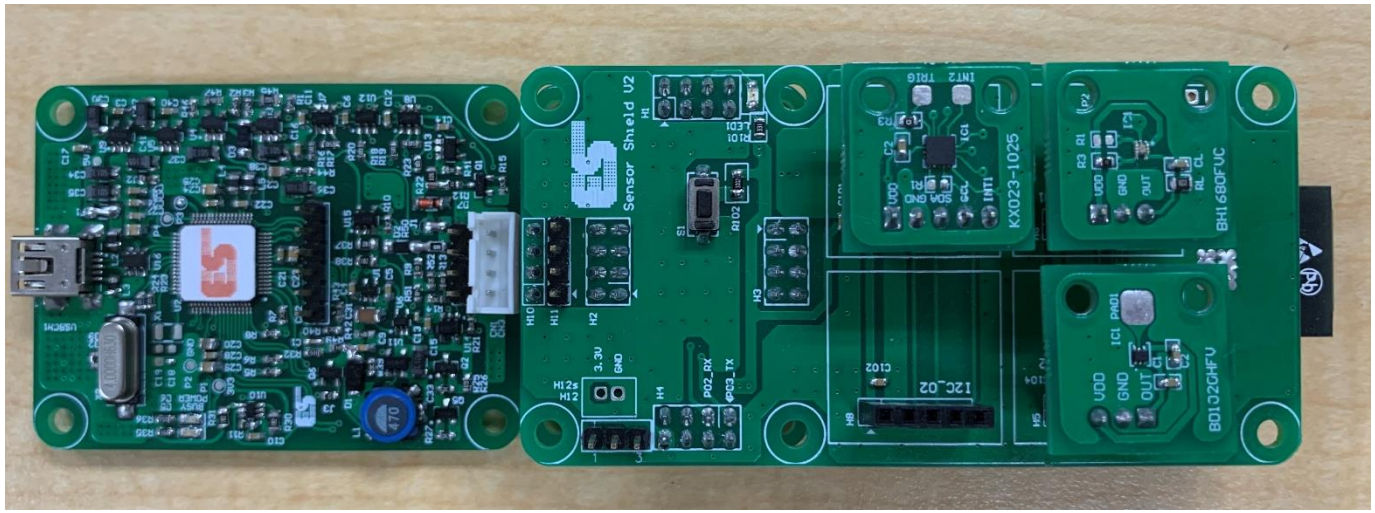
**UART00_TX_Buf[6] = 0x80**

**UART00_TX_Buf[7] = 0x08**

**UART00_TX_Buf[8] = 0x27**

# Connection ES-ICD-V1 ,Sensor Shield and Module

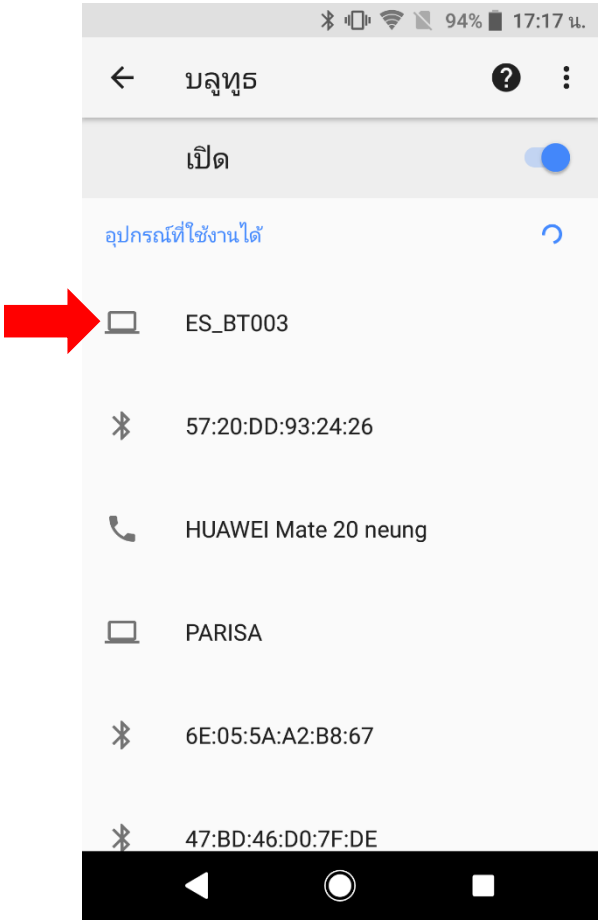


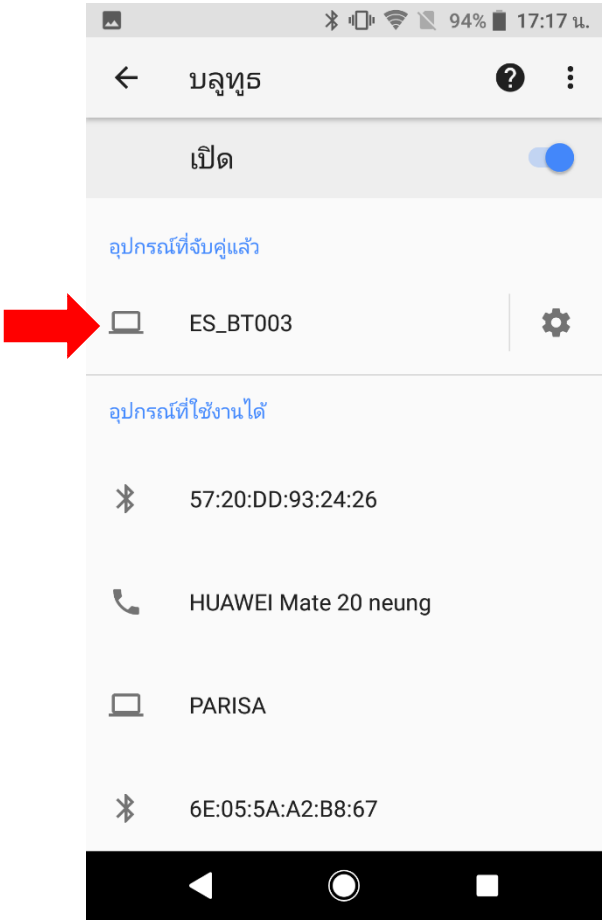




Pichet Sriphomma pichet@es.co.th

## Using the application

### Step 1 Connect MCU Board with Bluetooth




Pichet Sriphomma pichet@es.co.th

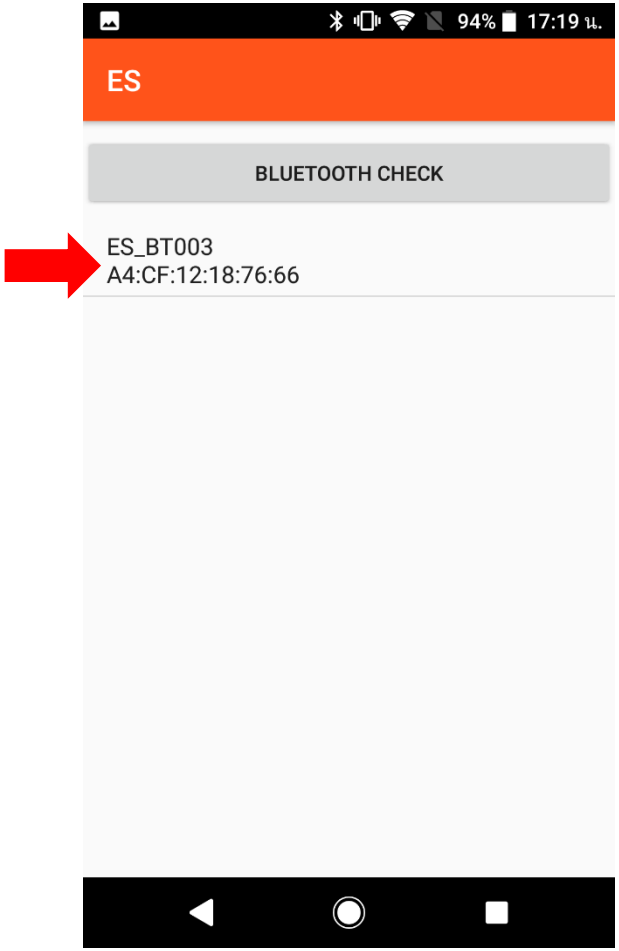
## Using the application

### Step 2 Open application and click “Bluetooth Check”


Pichet Sriphomma pichet@es.co.th

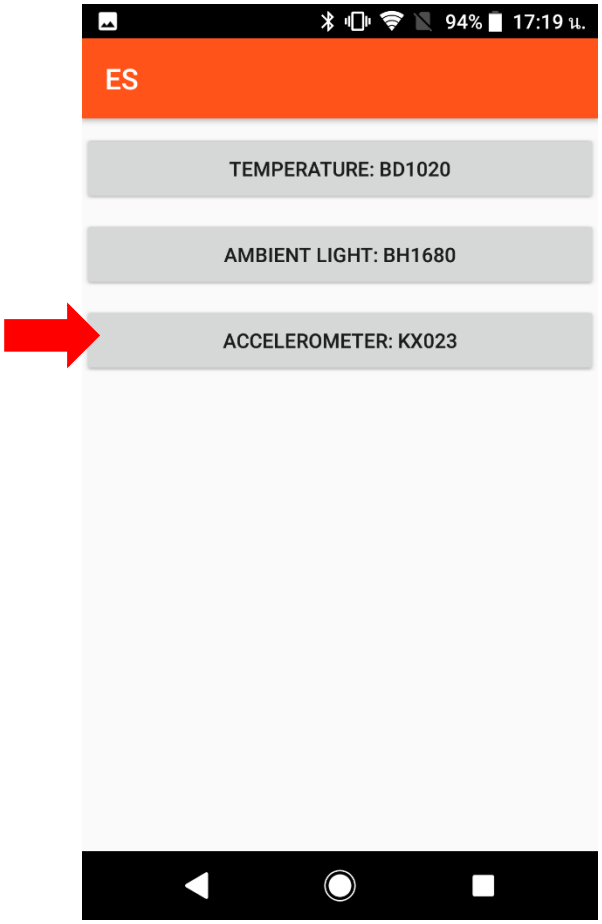
## Using the application

### Step 3 Choose device and connect




Pichet Sriphomma pichet@es.co.th

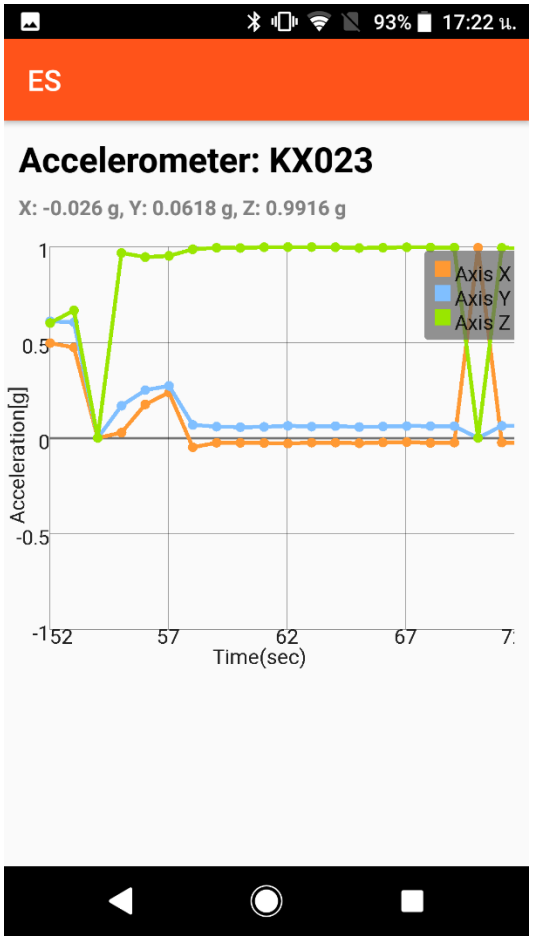
**Step 4 After connected this app Show button to choose.**
**Choose “Accelerometer : KX023”**




Pichet Sriphomma pichet@es.co.th

## Using the application

**Step 5 After Choose this app show Signal graph of KX023.**




Pichet Sriphomma pichet@es.co.th

# ML62Q1300 Group Pin List

| Pin name (Primary function) | Primary func. Others | 2nd func. SIU | 3nd func. SIU | 4nd func. I2C | 5nd func. Timers | 6nd func. Others | 7nd func. Others | 8nd func. ADC |
|---|---|---|---|---|---|---|---|---|



Pichet Sriphomma pichet@es.co.th

https://www.lapis-semi.com/cgi-bin/MyLAPIS/regi/login.cgi

pichet@es.co.th


ROHM Semiconductor (Thailand) Co.,Ltd.