Tutorial 4 – ARC EM9D AIoT DK Introduction and SDK Hands-on (Lab 1-3)

Willie Tsai

2023/02/06

ARC EM9D AIoT DK Introduction

ARC EM9D AIoT DK Introduction

- CPU: HX6537 (ARC EM9D DSP with FPU)
- Frequency: 400MHz
- SPI program flash: 1MB / 2MB
- Ram size: ~2MB

  320kB program ICCM

  320kB data DCCM/XCCM/YCCM

  1472kB system memory
- 64kB boot ROM

- Camera: HM0360 640 x 480 8bit crayscale CMOS VGA camera

ARC EM9D AIoT DK Introduction

- UART: 2

  UART0: CP2102

  UART1: FT232

- SPI: 1 Master and Salve

  Use for programmer interface or

  Arduino extension GPIO (SC16IS752)

- I2C Master: 2

  I2C0: Use for IMU, Arduino extension GPIO (PCA9672) and Wi-Fi/BT

  I2C1: Use for user define

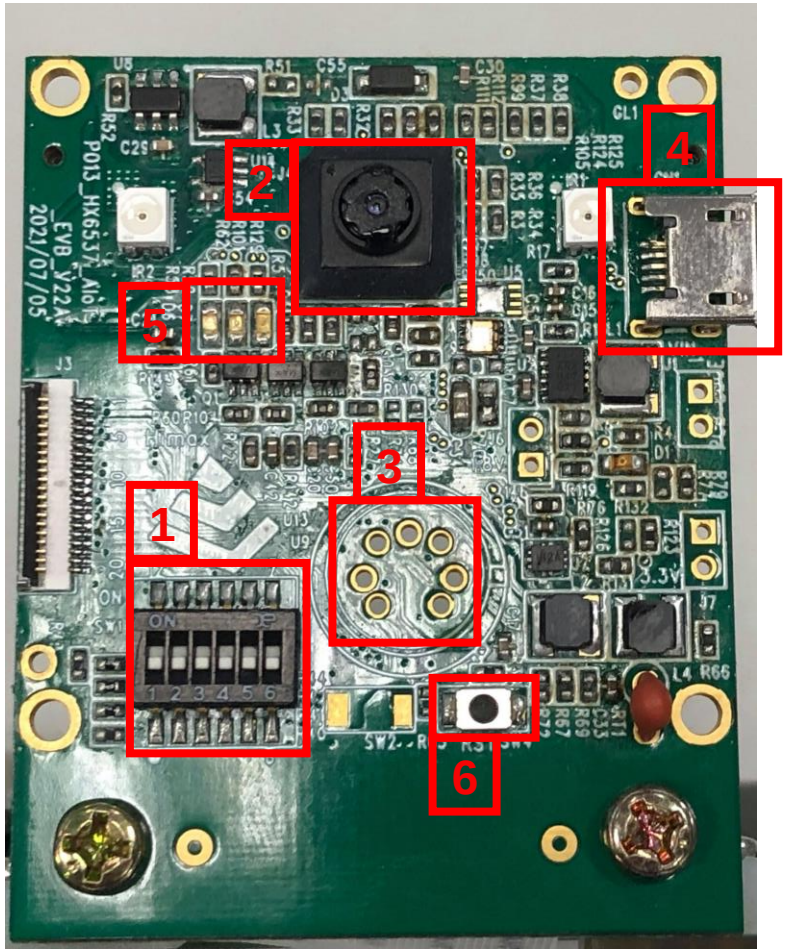
ARC EM9D AIoT DK: CPU Board

1. Boot mode switch
2. HM0360 AoSTM VGA camera

   Back side illuminated (BSI) CMOS image sensor

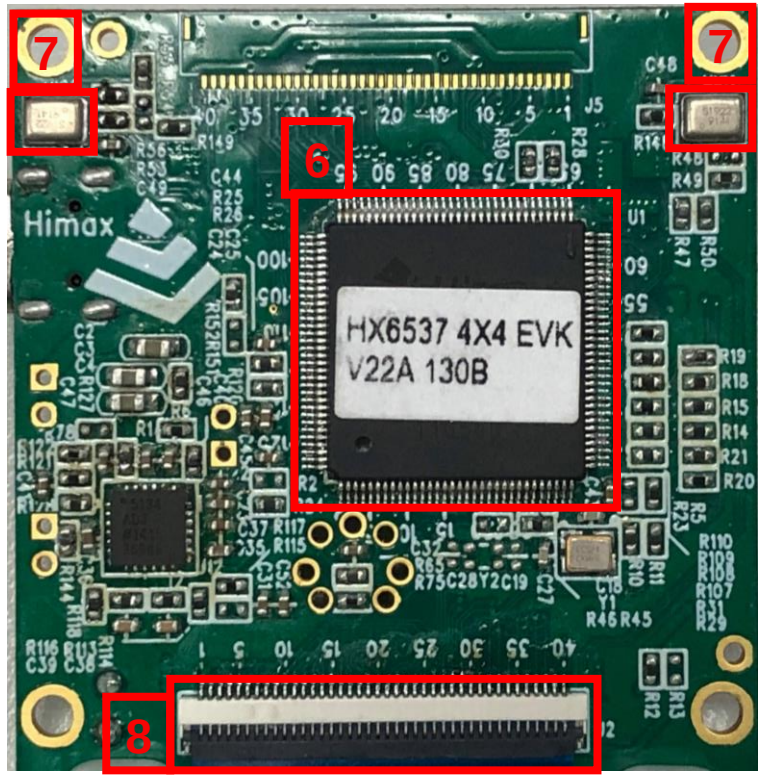
   Active pixel array: 656 x 496

   Frame rate: QQVGA 1FPS to VGA 60FPS
3. IR sensor
4. CPU board power input
5. LED
6. Reset button



ARC EM9D AIoT DK: CPU Board

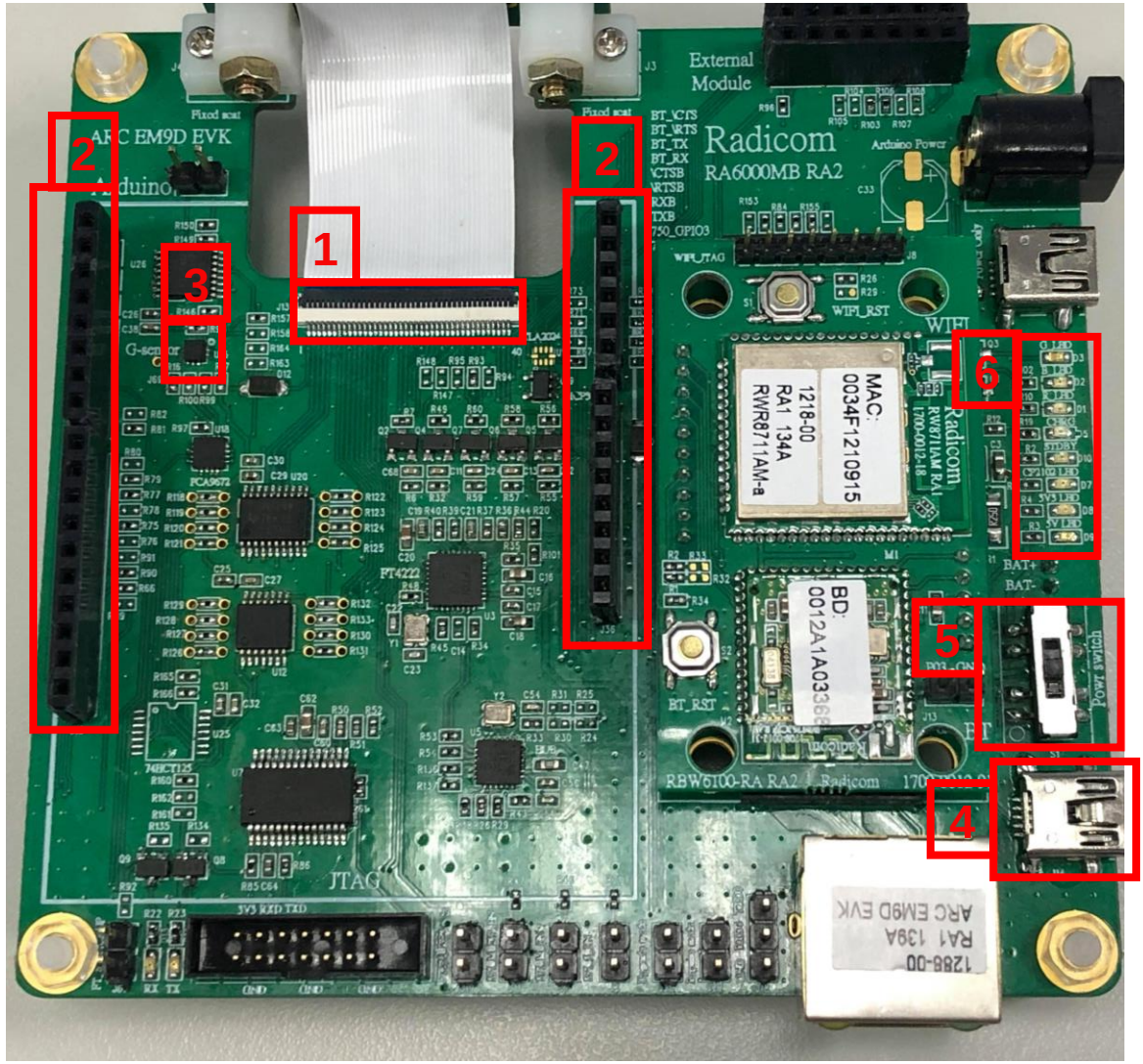
6. CPU HX6537
7. Microphones (L/R) at back side
8. FPC 40Pin connect to extension board



ARC EM9D AIoT DK: Extension Board

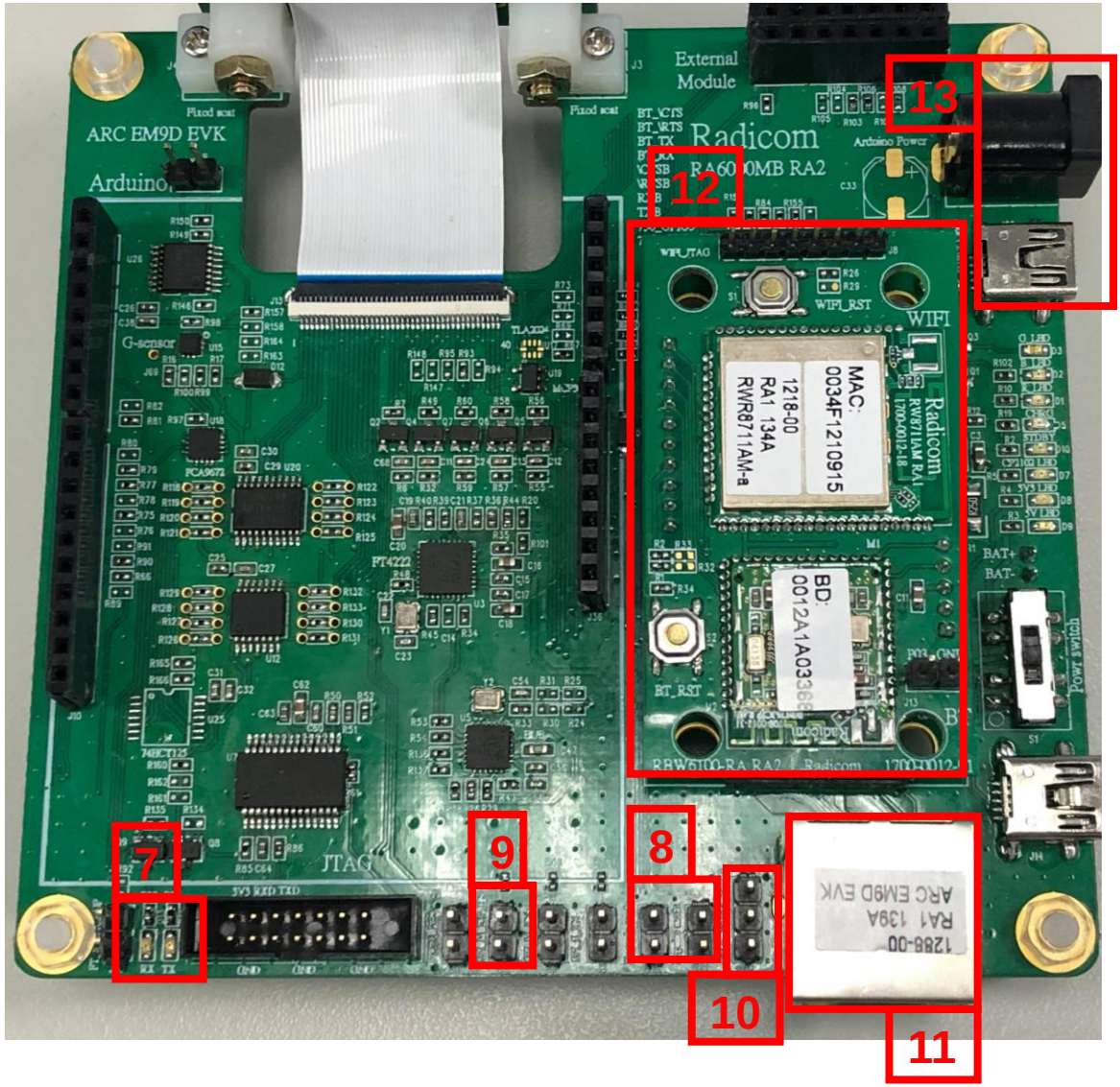
1. FPC 40Pin connect to CPU board
2. Arduino extension connector

   (Extend by SC16IS752 & PCA9672)
3. GMA303KU accelerometer
4. USB and dc power input connector
5. Power switch
6. LED



ARC EM9D AIoT DK: Extension Board

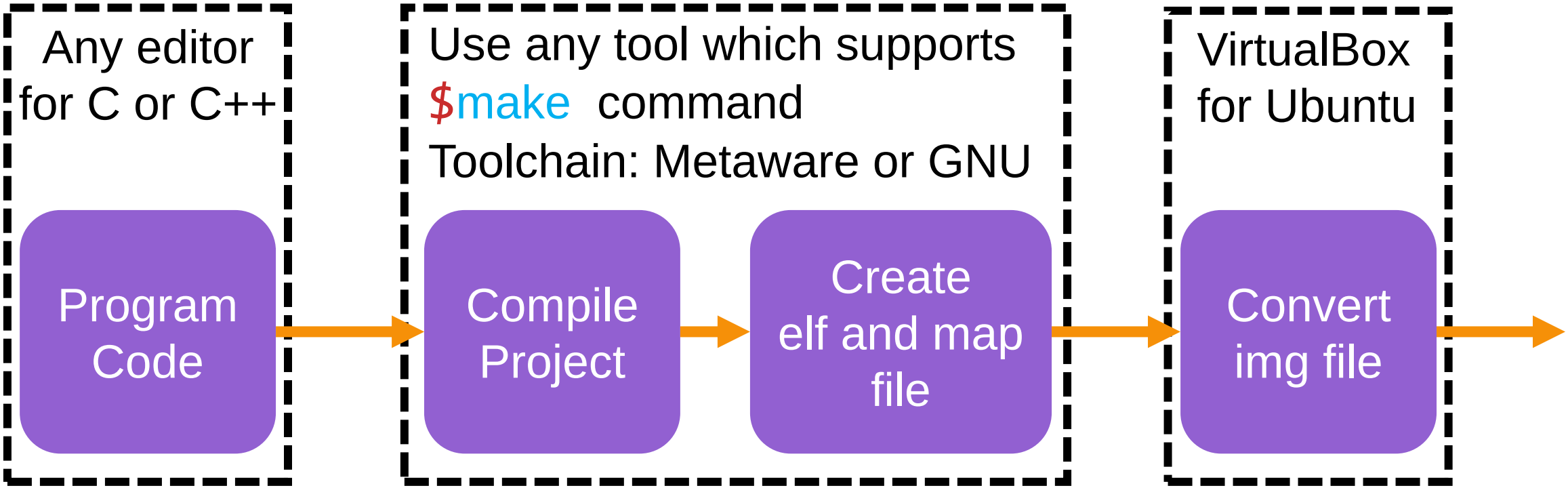
7. FT232 RX/TX led
8. Boot mode select header
9. SPI mode select header
10. UART0 RX/TX header
11. RJ45 connector
12. Bluetooth / Wi-Fi module
13. External power input



ARC EM9D AIoT DK Project Development Flow

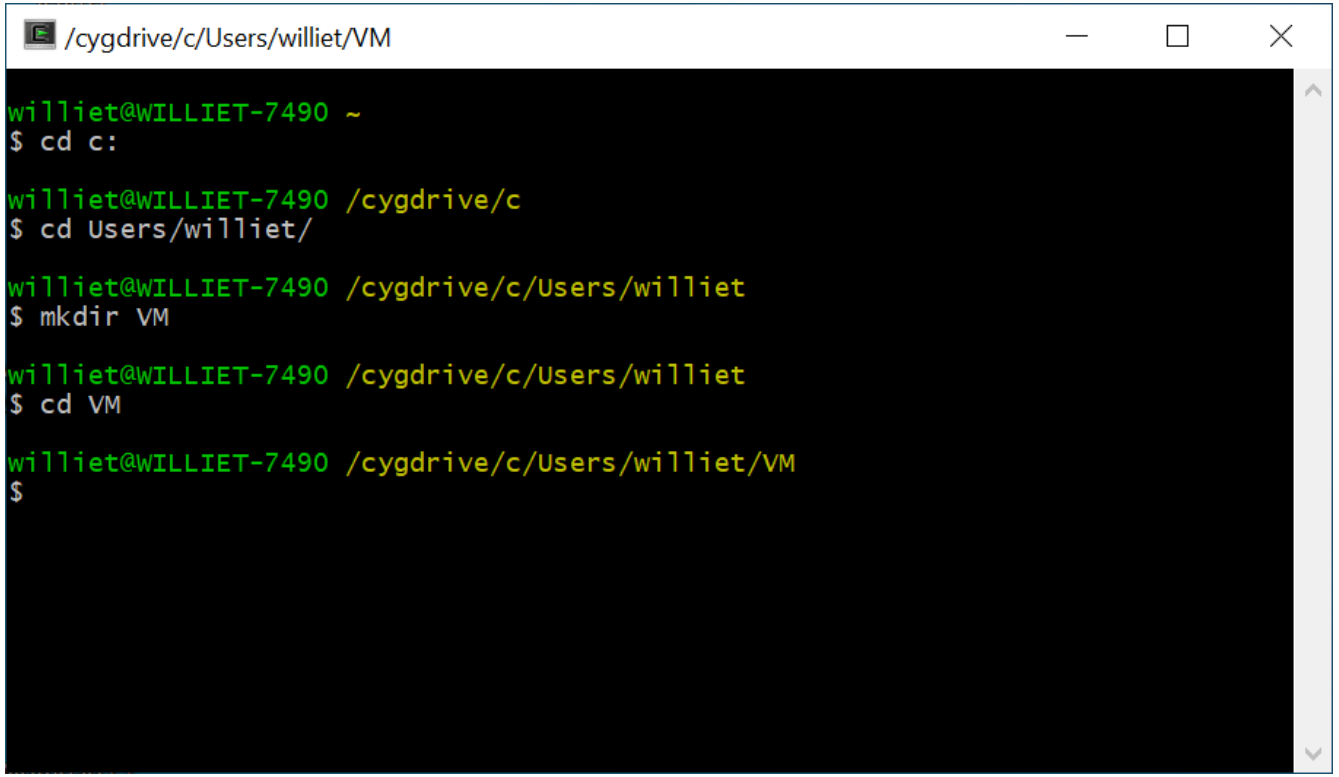
Stage	TensorFlow Model Development	Firmware Development	Run / Update Application On ARC EM9D AIoT DK
Tool	Anaconda Cygwin	Cygwin Metaware or ARC GNU VirtualBox (Ubuntu 20.04)	JTAG Himax-FT4222-GUI USB Cable
Language	Python 3	C language C++ language	

Firmware Development



Example Project Download (Already done in Tutorial-2)

1. Open Cygwin64 Terminal
   - $ cd c:
   - $ cd Users/{username}/ (to your working file path)
   - $ mkdir VM (Suggest create a new folder named “VM”)
   - $ cd VM



Example Project Download (Already done in Tutorial-2)

2. Download SDK and unzip to folder *“C:\Users\{username}\VM”*
   Please contact Synopsys Taiwan for the SDK

Example Project Download (Already done in Tutorial-2)

After these steps, your file structure will be like:

Synopsys_SDK_Vxx
|
---- *Example_Project*
    |
    ---- *Labx (Firmware project)*
    ---- *Labx (Python project for TensorFlow Ppoject)*
    ---- *LabPY (Python project for data convert)*
---- *others (Library, toolchain setting …)*
---- *tools*
    |
    ---- *image_gen_cstm (Convert elf and map file to image file)*
    ---- *HMX_FT4222H_GUI (Download image file to MCU)*

API Library

- “……/Synopsys_SDK_Vxx/platform/inc”
  “……/Synopsys_SDK_Vxx/library/…”
  You can find many API header file here.

- If you need to use API, please include it in your source code.
  Example: “……/Synopsys_SDK_Vxx/Example_Project/Lab1_UART”
  You will see folder “src” and “inc”
  “src” folder: always keep your .c and .cpp file in here.
  “inc” folder: always keep your .h file in here.
  (c file: c language)
  (cpp file: c++ language)

Synopsys SDK

- “……/Synopsys_SDK_Vxx/Example_Project”
  Example Project from Lab1~5, you can copy or reference it.

- For example: “……/Synopsys_SDK_Vxx/Example_Project/Lab1_UART”
  You will see folder “src” and “inc”
  “src” folder: always keep your .c and .cpp file in here.
  “inc” folder: always keep your .h file in here.
  (c file: c language)
  (cpp file: c++ language)

Make Project and Flash File

There are some commands can be used,
$ make : compile and link your project with default toolchain.
$ make TOOLCHAIN=gnu: compile and link your project with GNU.
$ make TOOLCHAIN=mw: compile and link your project with Metaware.
$ make clean : remove all .o file of default toolchain of this project
$ make boardclean : remove all .o file of all toolchain of this project

You can add a command for changing toolchain
(default toolchain is gnu, define in makefile)
“TOOLCHAIN=mw”: compile with MetaWare toolchain
“TOOLCHAIN=gnu”: compile with ARC GNU toolchain
Please use $ make boardclean after you change toolchain.

Hands-on (Lab 1): UART

Lab1: UART

- Header File: hx_drv_uart.h
- Create UART structural pointer

```
DEV_UART * uart_x_ptr;
```

- Set UART structural pointer to UARTx

```
DEV_UART_PTR hx_drv_uart_get_dev(USE_SS_UART_E);
```

```
// (USE_SS_UART_E) options are bellow
   USE_SS_UART_0 = DFSS_UART_0_ID      /* Select UART 0  */
   USE_SS_UART_1 = DFSS_UART_1_ID      /* Select UART 1  */
   USE_SS_UART_ALL = DFSS_UART_all_ID  /* Select UART All  */

e.g.   DEV_UART * uart0_ptr; //Create a structural pointer
       uart0_ptr = hx_drv_uart_get_dev(USE_SS_UART_0); //Pointer will be UART 0
```

Lab1: UART

- Open UART and set baud rate

```
int32_t (*uart_open) (uint32_t baud);

// (uint32_t baud) options are bellow
   UART_BAUDRATE_600          /* uart baudrate 600bps */
   UART_BAUDRATE_1200         /* uart baudrate 1200bps */
   UART_BAUDRATE_2400         /* uart baudrate 2400bps */
   UART_BAUDRATE_4800         /* uart baudrate 4800bps */
   UART_BAUDRATE_9600         /* uart baudrate 9600bps */
   UART_BAUDRATE_14400        /* uart baudrate 14400bps */
   UART_BAUDRATE_19200        /* uart baudrate 19200bps */
   UART_BAUDRATE_38400        /* uart baudrate 38400bps */
   UART_BAUDRATE_57600        /* uart baudrate 57600bps */
   UART_BAUDRATE_115200       /* uart baudrate 115200bps */

e.g.  uart0_ptr->uart_open(UART_BAUDRATE_115200); //Open UART with 115200bps
```

Lab1: UART

- UART send data

```
    int32_t (*uart_write) (const void *data, uint32_t len);
// (const void *data)  Send data pointer
// (uint32_t len)      How many byte need to send

e.g.   char str_buf[100];
       sprintf(str_buf, "Start While Loop\r\n");
       uart0_ptr->uart_write(str_buf, strlen(str_buf));
```

Lab1: UART

- UART read data

```
int32_t (*uart_read) (void *data, uint32_t len);
int32_t (*uart_read_nonblock) (void *data, uint32_t len);
// (const void *data)  Read data pointer
// (uint32_t len)      How many byte need to read

e.g.   char str_buf[100];
       uart0_ptr->uart_read(str_buf, 5); //It blocks in here and waits for 5 byte

e.g.   char str_buf[100];
       uart0_ptr->uart_read_nonblock(str_buf, 5);
       //It will return “How many bytes it gets but no greater than 5”
```

Lab1: UART

Conclusion

- Header File: hx_drv_uart.h
- UART need to create a structural pointer and set pointer to UARTx

```
DEV_UART * uart_x_ptr;
DEV_UART_PTR hx_drv_uart_get_dev(USE_SS_UART_E);
```

- Should initialize and set baud rate before send and get UART data

```
int32_t (*uart_open) (uint32_t baud);
```

- After you initialize UART, you can send and get UART data

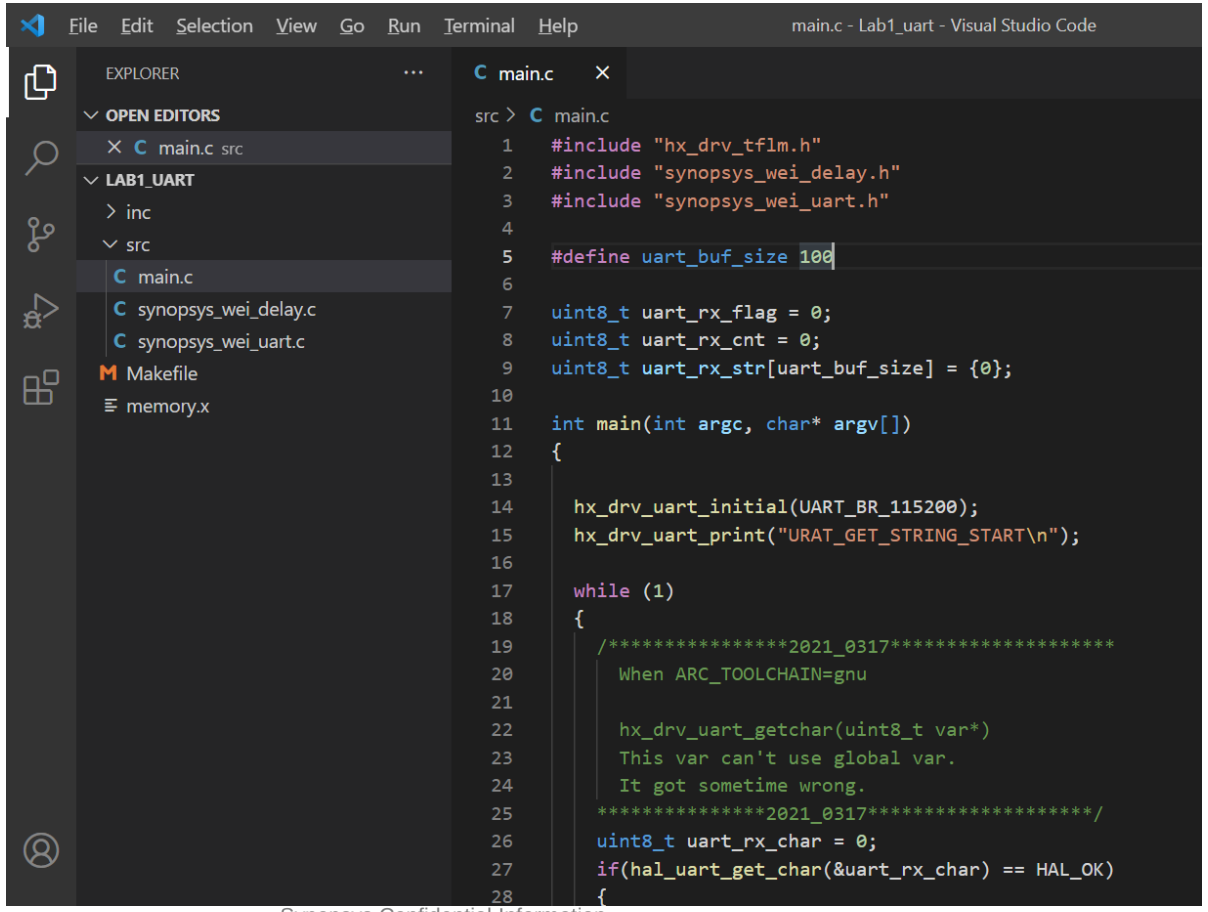
```
int32_t (*uart_write) (const void *data, uint32_t len);
int32_t (*uart_read) (void *data, uint32_t len);
int32_t (*uart_read_nonblock) (void *data, uint32_t len);
```

Lab1: UART

- Open Visual Studio Code & open folder

  “……/ Synopsys_SDK_Vxx/Example_Project/Lab1_UART”

- Open “src/main.c”



Lab1: UART

- Open Terminal and key-in

  make

```
PS C:\Users\williet\VM\ARC_EVB_SDK\GNU_SDK_S1_0\Synopsys_SDK_Vxx\Example_Project\Lab1_UART> make
../../def_linker_script/def_linker_script.mk:27: ./../../def_linker_script/linker_template_gnu_M.ld
copy /Y obj_socket_24\gnu_arcem9d_wei_r16\WEI_FW_gnu_arcem9d_wei_r16.elf .\..\..\tools\image_gen_cstm
6.elf
        1 file(s) copied.
copy /Y obj_socket_24\gnu_arcem9d_wei_r16\WEI_FW_gnu_arcem9d_wei_r16.map .\..\..\tools\image_gen_cstm
6.map
        1 file(s) copied.
PS C:\Users\williet\VM\ARC_EVB_SDK\GNU_SDK_S1_0\Synopsys_SDK_Vxx\Example_Project\Lab1_UART>
```

- Make sure there are no error message
- You will get output.elf and memory.map.
- Files will also be copied to image convert tool folder automatically

  "……/Synopsys_SDK_Vxx/tools/image_gen_cstm/input/"

Lab1: UART

- Open Virtual Machine Ubuntu and go to same project path:

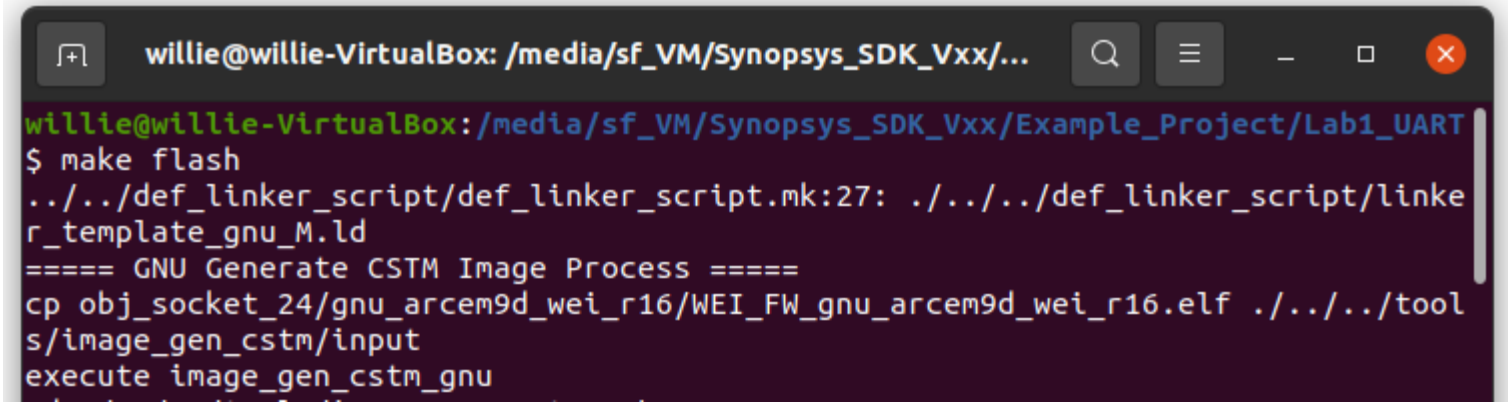
  {Share Folder…}/Synopsys_SDK_Vxx/Example_Project/Lab1_UART”

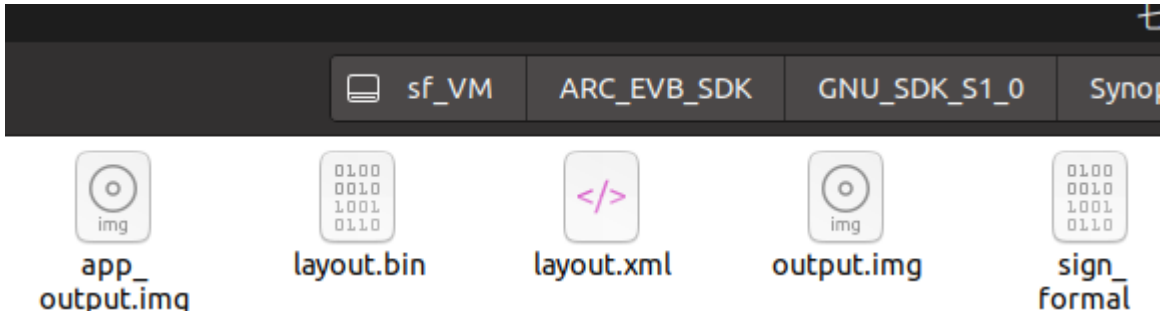
- Open Terminal and key-in

  $ make flash

- You will get image file in:

  “……/Synopsys_SDK_Vxx/tools/image_gen_cstm/output/output.img”





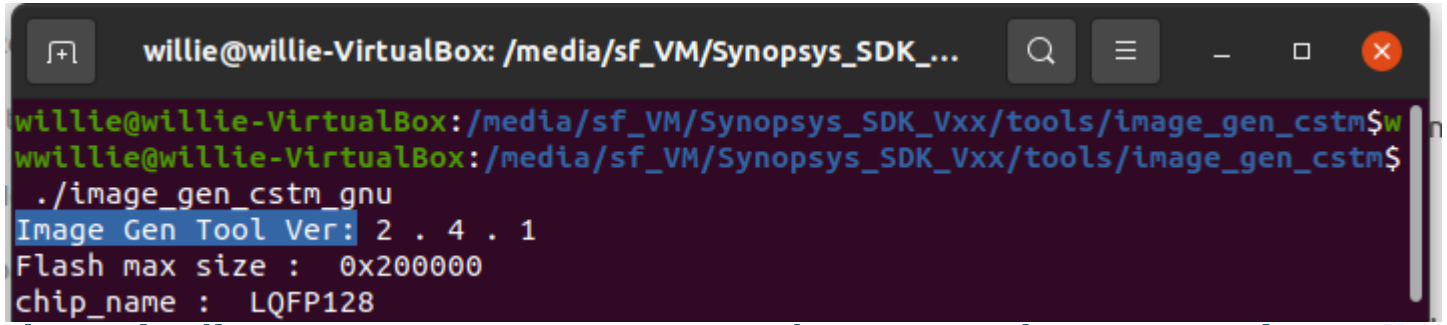
Lab1: UART

- You can also use “image_gen_cstm_gnu” manually.

  Open terminal in VirtualBox Ubuntu, and go to the same path:

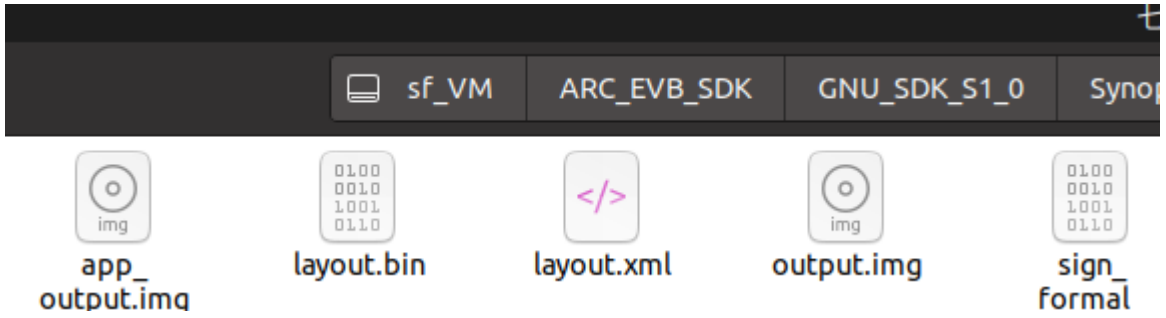
  “{Share Folder…}/ Synopsys_SDK_Vxx/tools/image_gen_cstm”

  $ ./image_gen_cstm_gnu

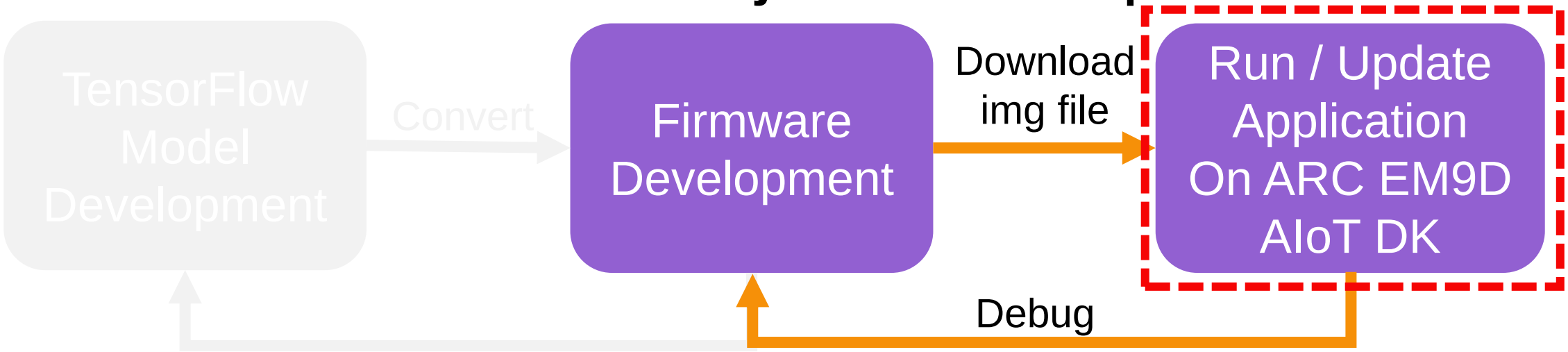


- You will get image file in:

  “……/Synopsys_SDK_Vxx/tools/image_gen_cstm/output/output.img”

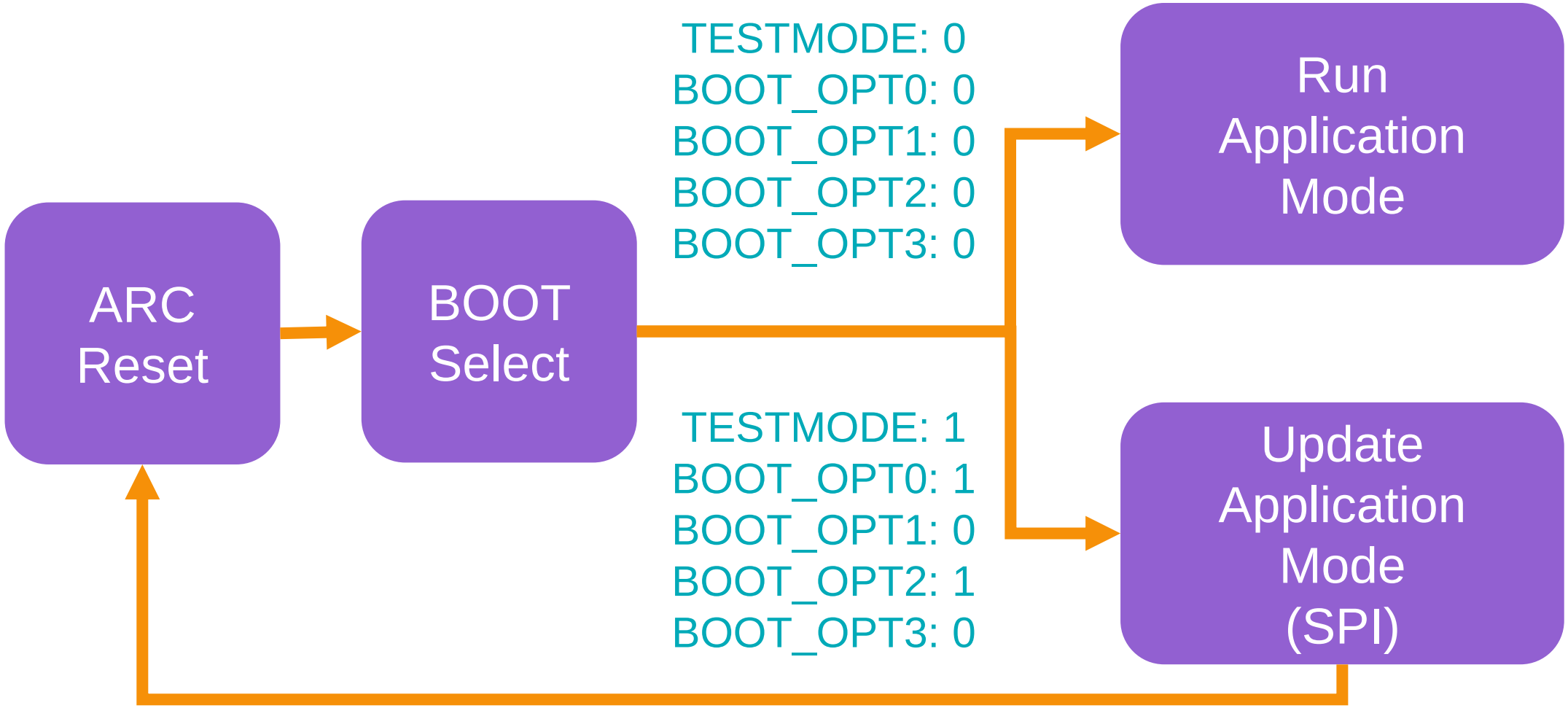


ARC EM9D AIoT DK Project Development Flow



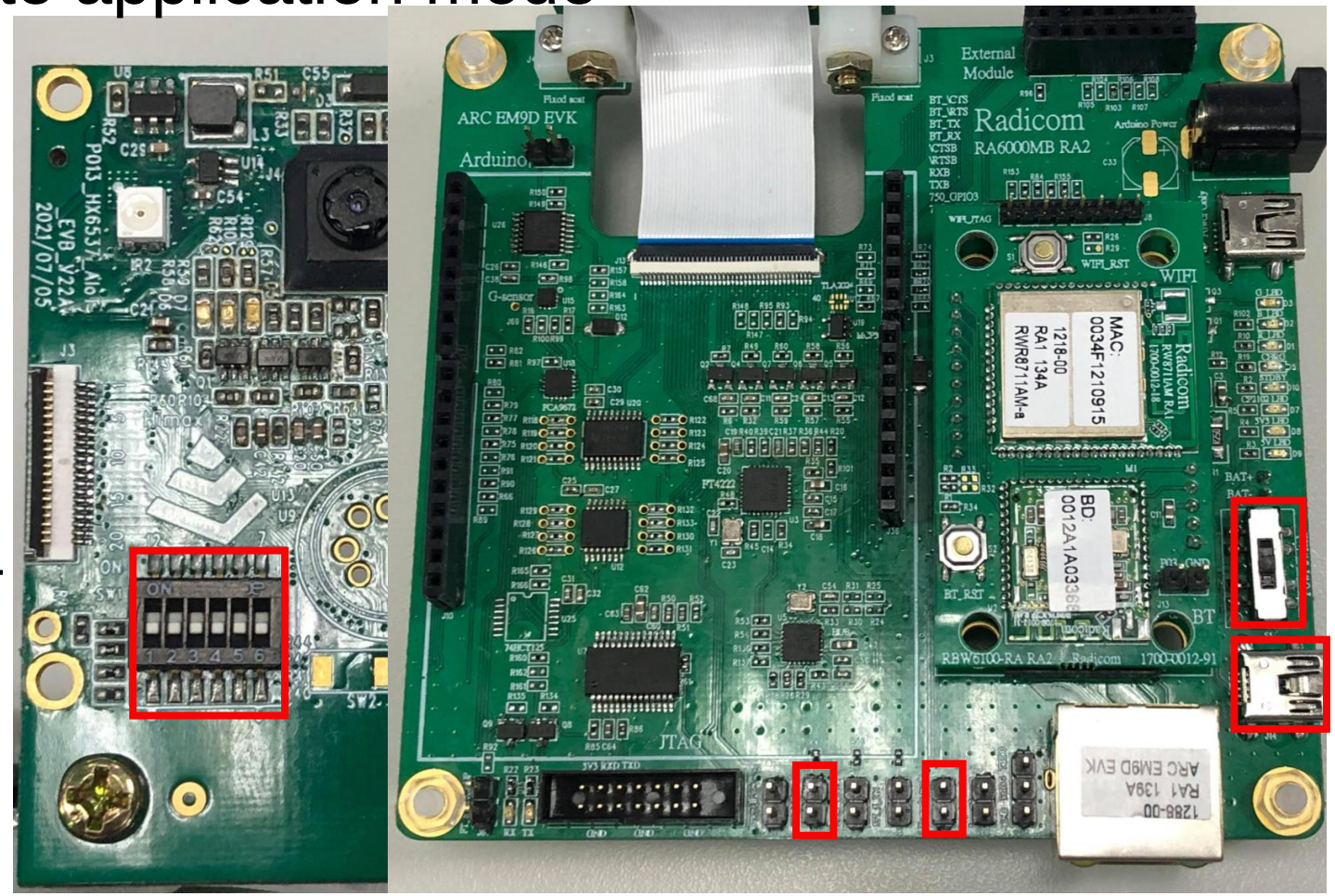
Stage	TensorFlow Model Development	Firmware Development	Run / Update Application On ARC EM9D AIoT DK
Tool	Anaconda Cygwin	Cygwin Metaware or ARC GNU VirtualBox (Ubuntu 20.04)	JTAG Himax-FT4222-GUI USB Cable
Language	Python 3	C language C++ language	

Run / Update Application on ARC EM9D AIoT DK



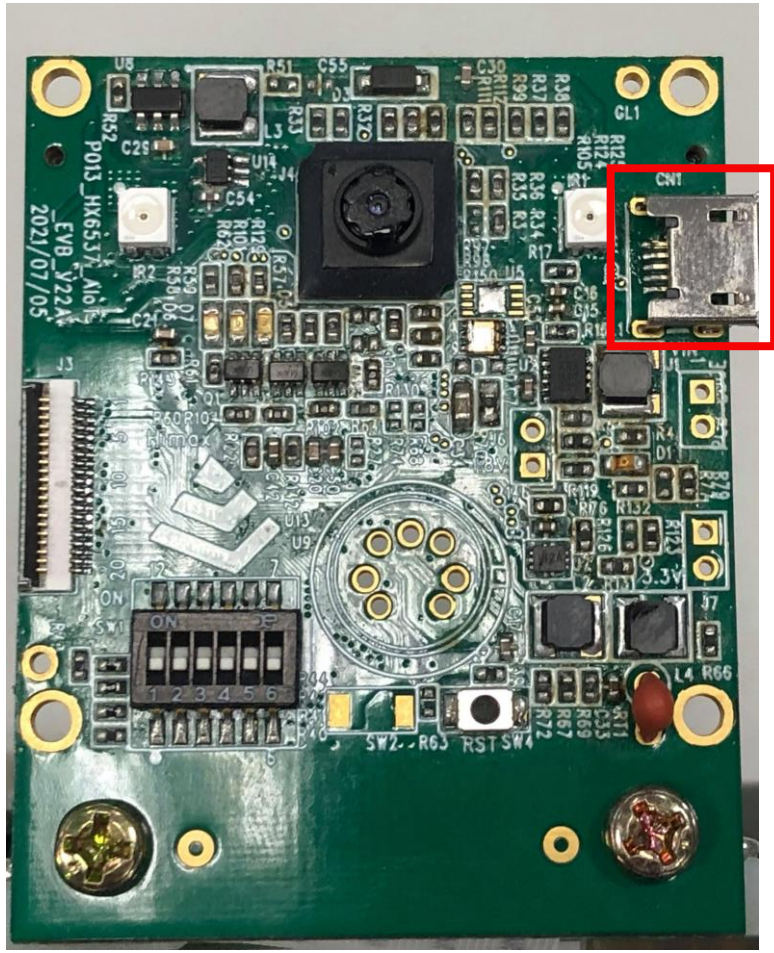
Update Application on ARC EM9D AIoT DK

1. Always modify SW1 as bellow
2. Short J20 and J11 for update application mode
   TESTMODE: 1
   BOOT_OPT0: 1
   BOOT_OPT1: 0
   BOOT_OPT2: 1
   BOOT_OPT3: 0
3. Turn power by S1
4. Connect USB Cable to J14



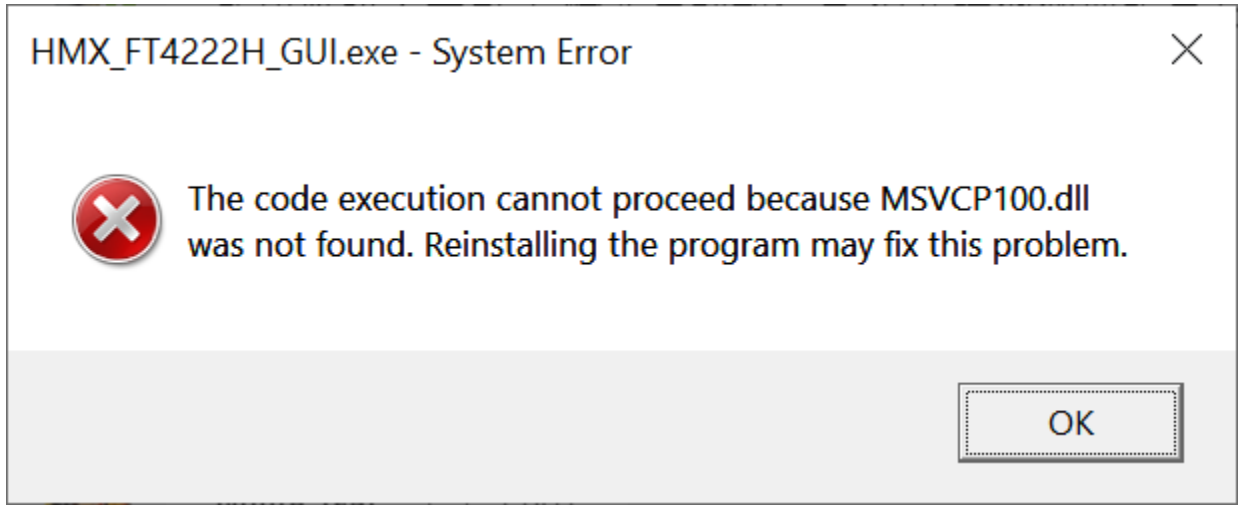
Update Application on ARC EM9D AIoT DK

5. Connect USB cable to CN1 for CPU board
   - This step is for some USB power source is lower than 5V.
   - If only connect USB cable to extension board, CPU may have fault during program flash.



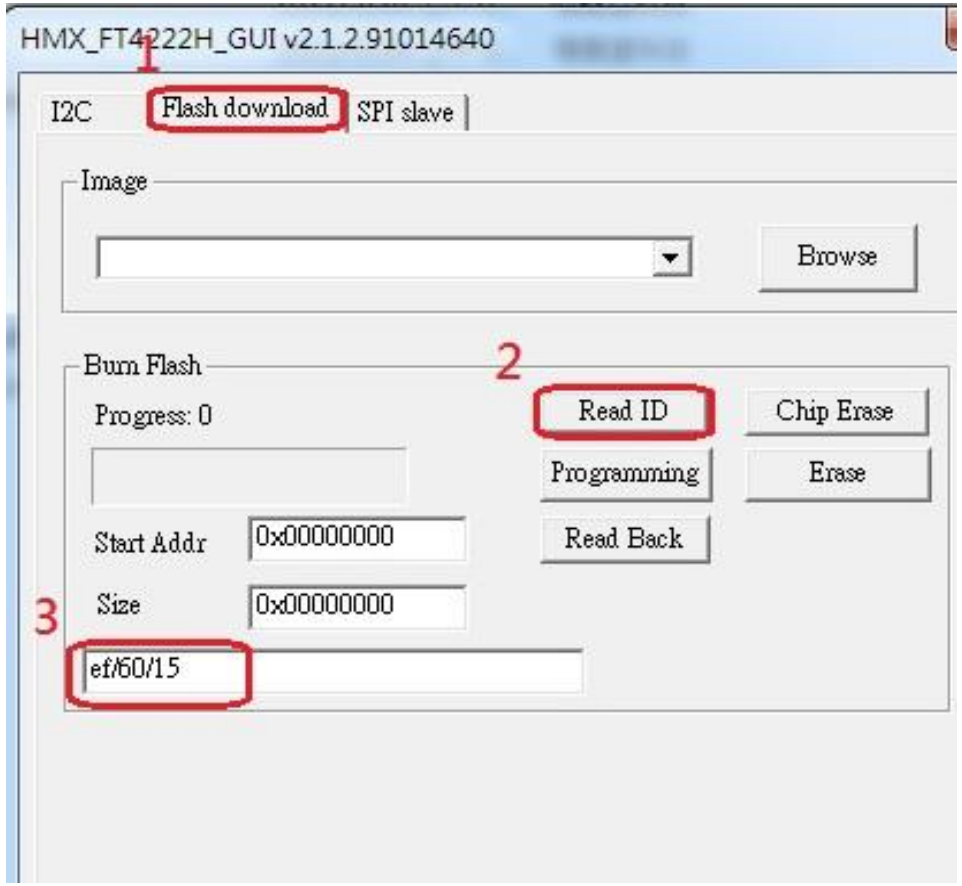
Update Application on ARC EM9D AIoT DK

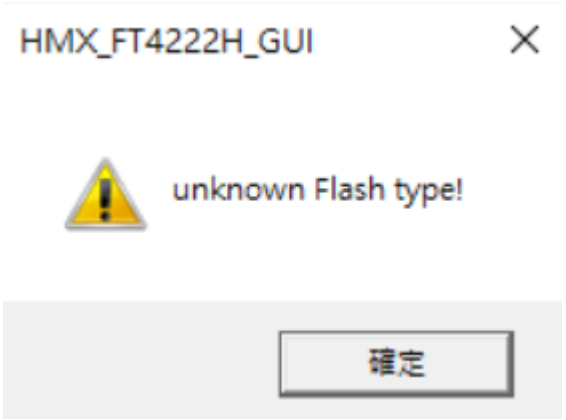
6. Execute *“HMX_FT4222H_GUI.exe“* in *“Synopsys_SDK_Vxx/tools/HMX_FT4222H_GUI/”*
   If you have a warning message as left bellow, make sure the ARC EM9D AIoT DK has been connected to PC, and USB device in PC is mounted.
   If USB device can’t find correct driver, please refer to Appendix-2&3.
   If your have DLL error as right bellow, please refer to Appendix-4.



Update Application on ARC EM9D AIoT DK

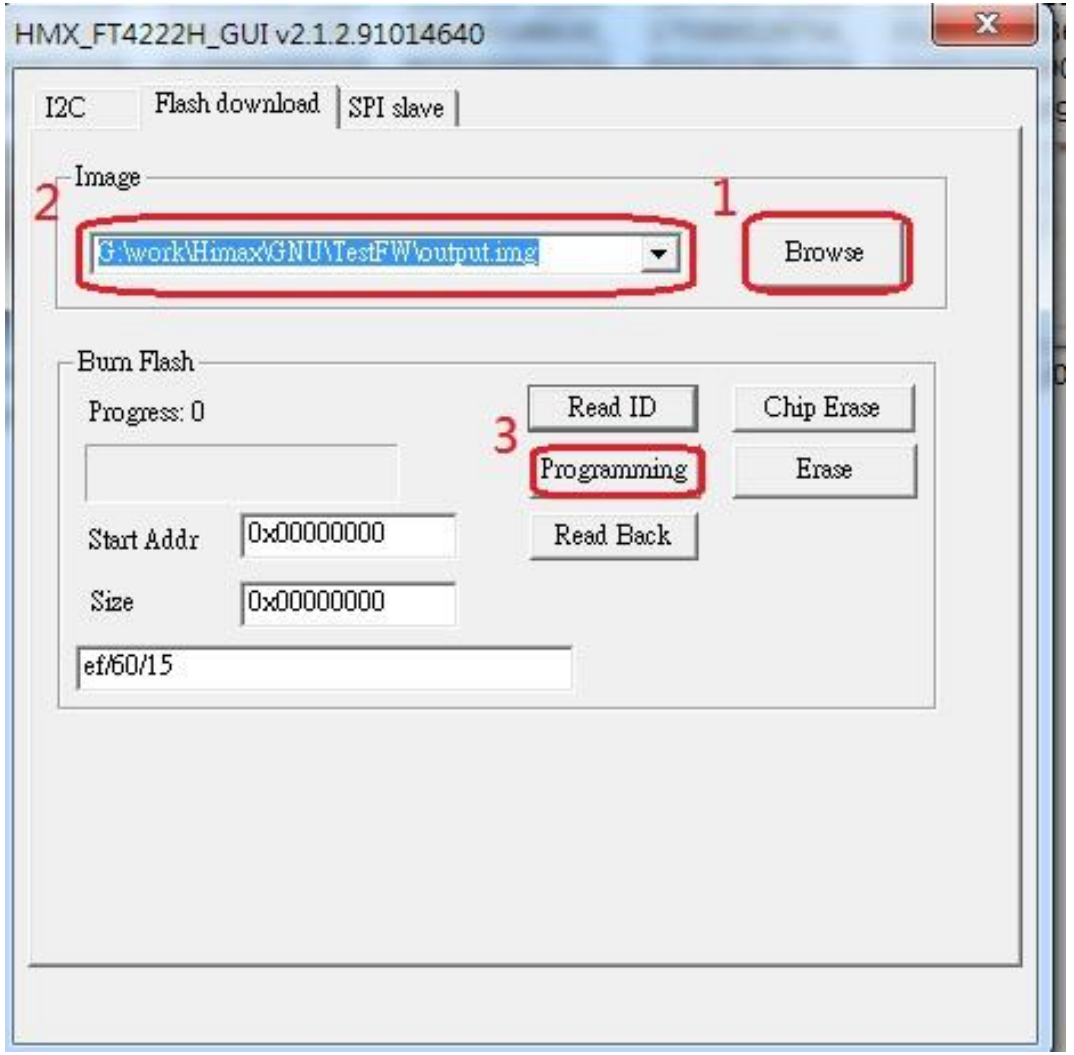
7. Select *“Flash download”* > *“Read ID”*
   You will get *“ef/60/15”* if MCU is ready.
   If the ID is incorrect or shows *“unknow Flash type!”* , make sure your boot-signals are right.
   After that, you can power-up and press reset switch, then close and execute *“HMX_FT4222H_GUI.exe“* again.





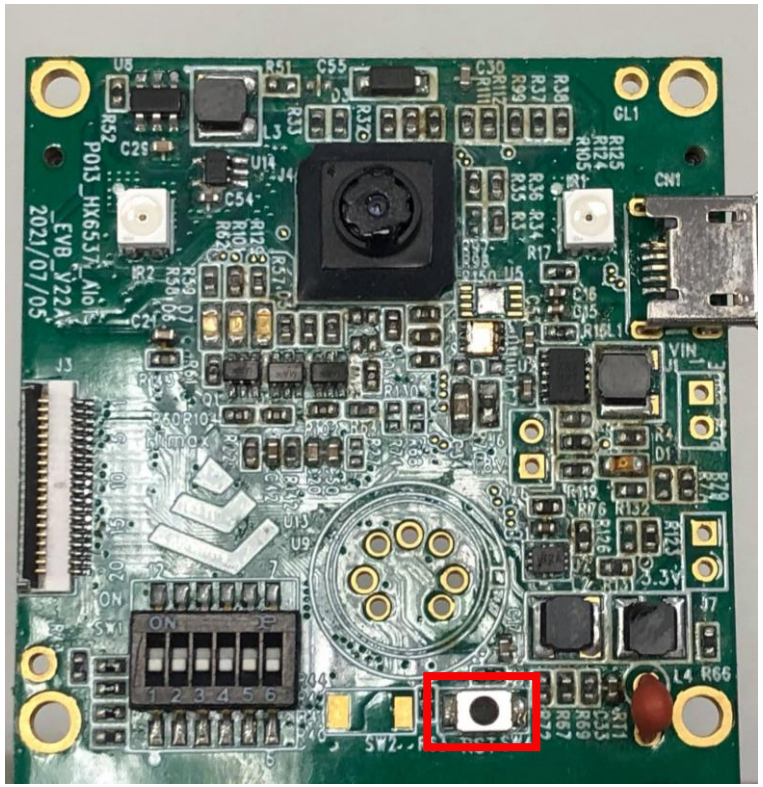
Update Application on ARC EM9D AIoT DK

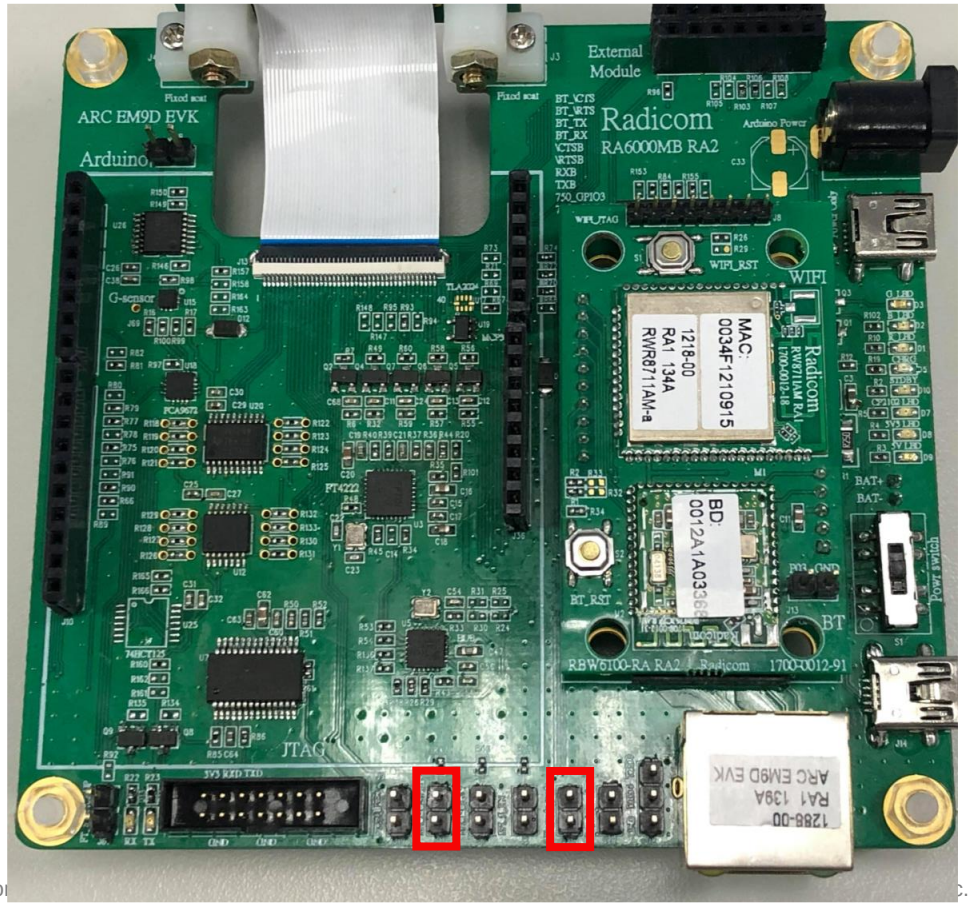
8. *Click "Browse"* to select your image file and *"Programming"*.
9. After programming finish, open J20 for running application.
   TESTMODE: 0
   BOOT_OPT0: 0
   BOOT_OPT1: 0
   BOOT_OPT2: 0
   BOOT_OPT3: 0



Run Application on ARC EM9D AIoT DK

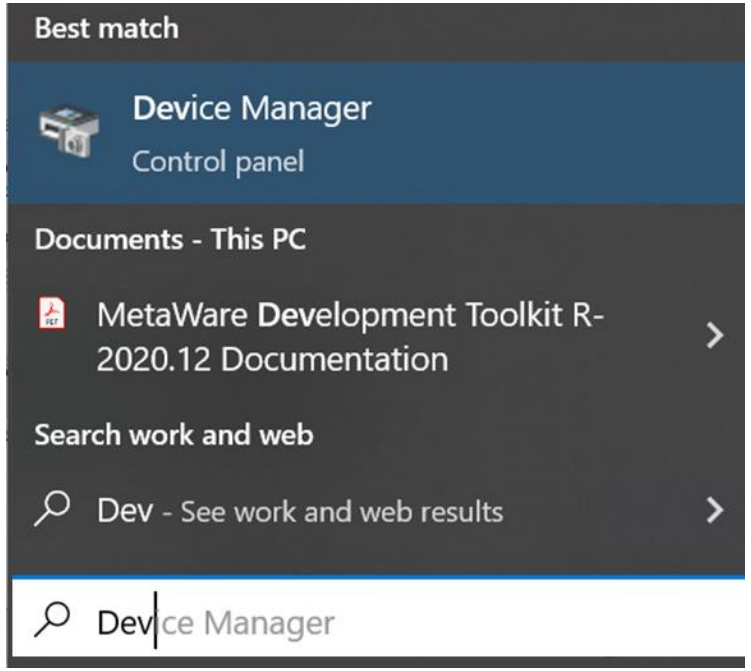
1. Short J20 and J11 for update mode, open J20 for run mode
2. Press reset button SW4. MCU will start to run the application.
   If MCU doesn't boot-up, make sure your boot-signals are set right, or you can try to power-up again.

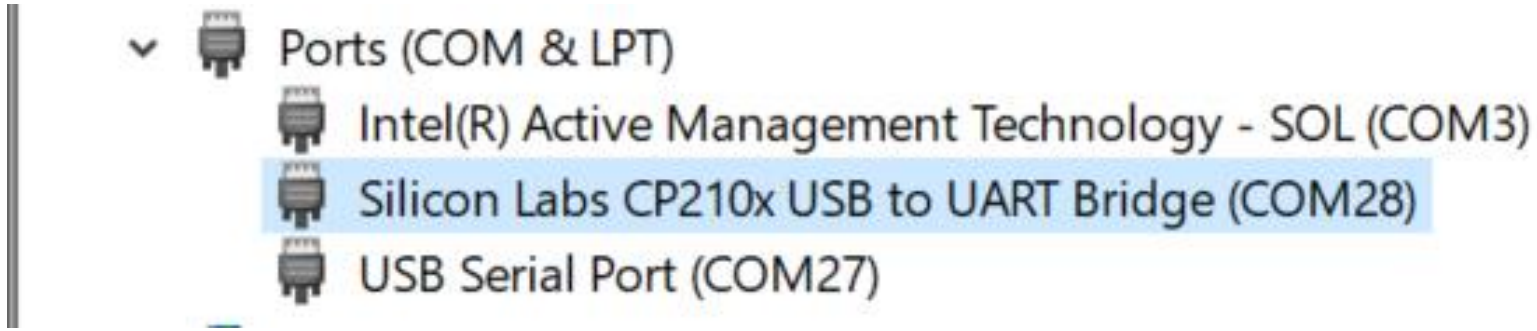




Run Application on ARC EM9D AIoT DK

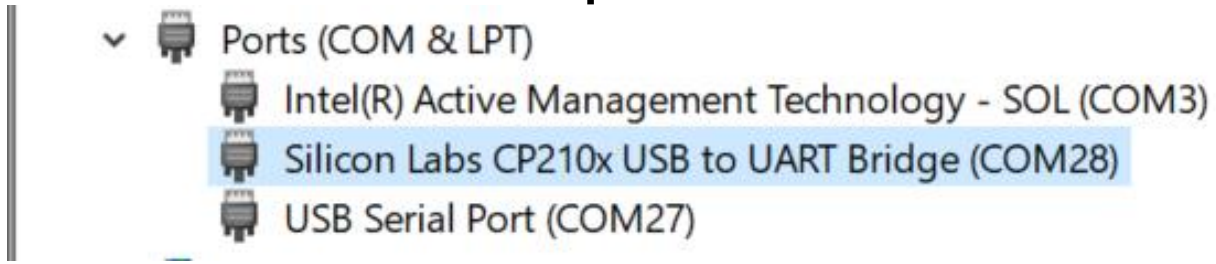
3. You can also use USB VCP to receive package from ARC EM9D AIoT DK.
4. Connect ARC EM9D AIoT DK and PC by USB Cable





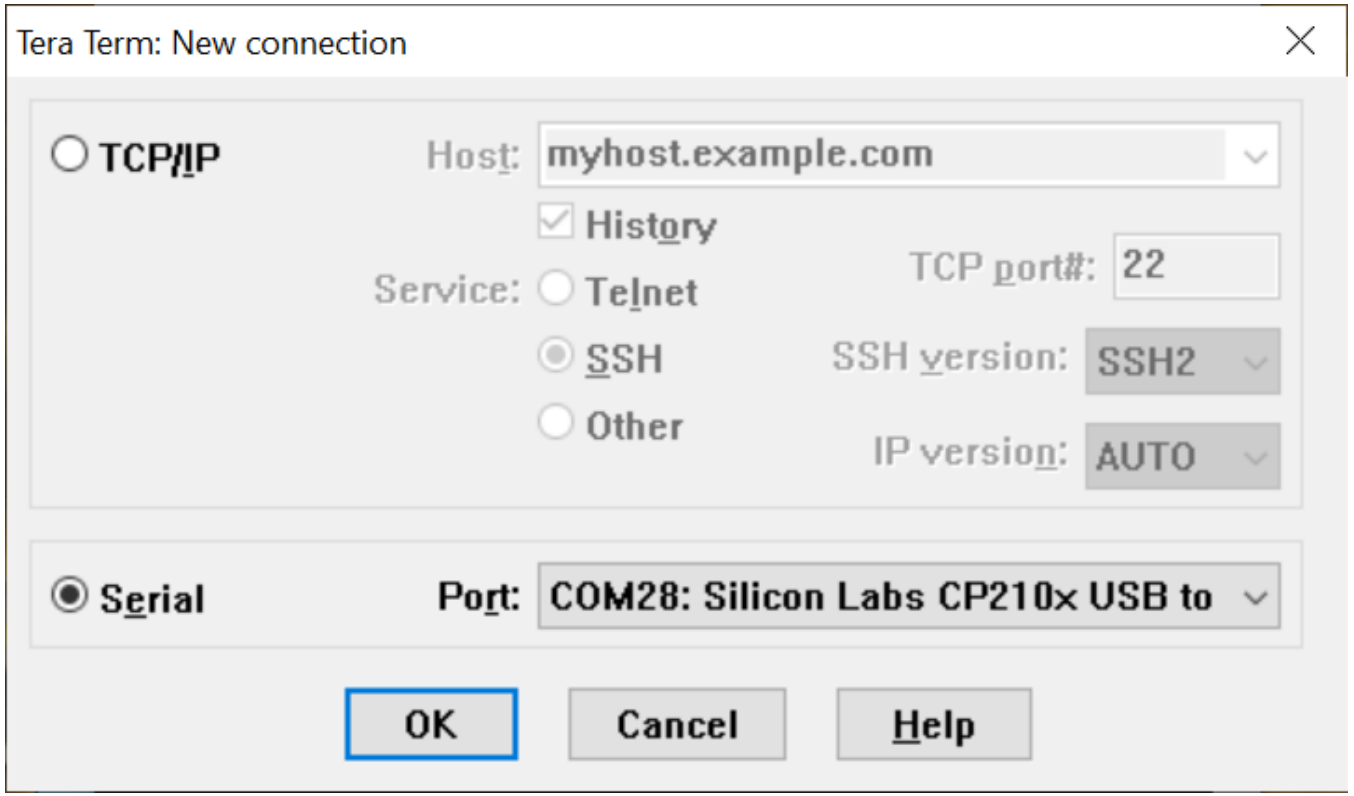
Run Application on ARC EM9D AIoT DK

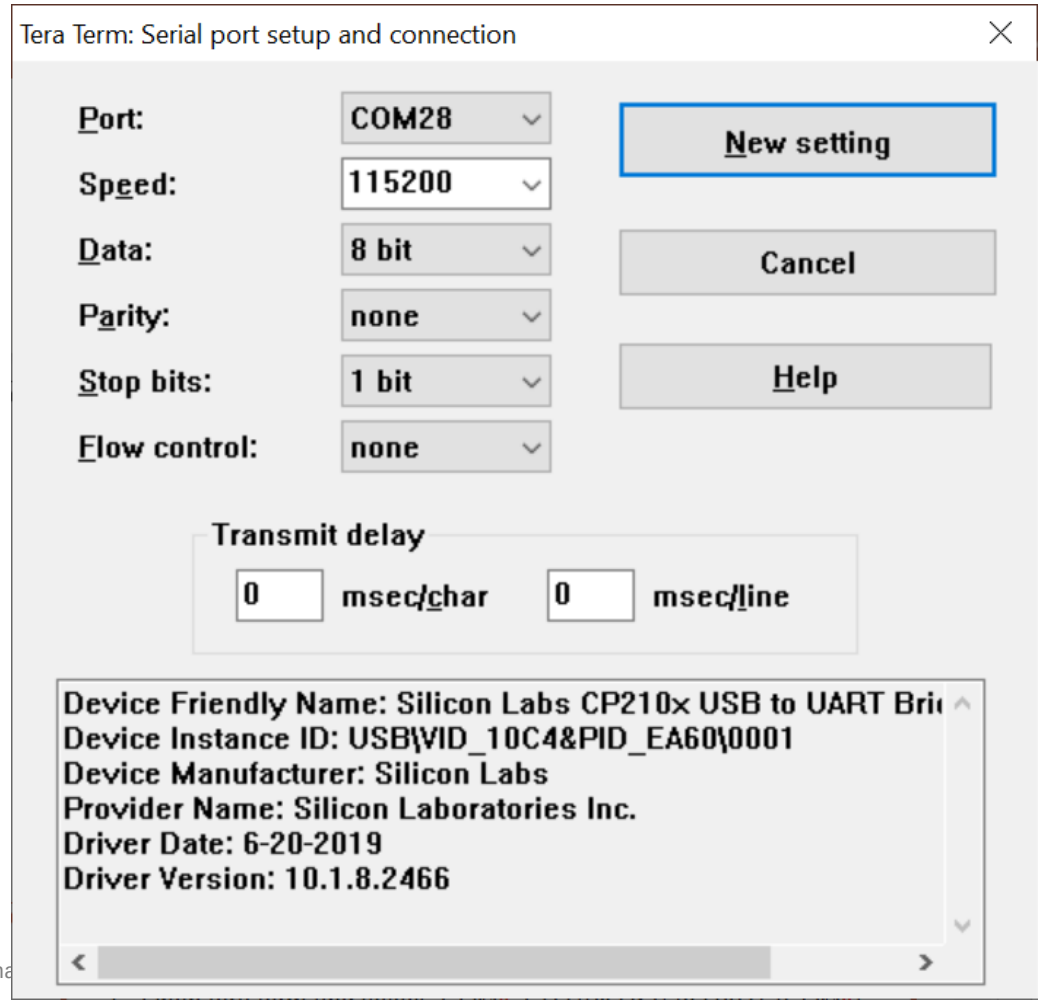
5. Check your ARC EM9D AIoT DK USB port number
   We have 2 USB ports on ARC EM9D AIoT DK .
   Uart0_CP2102: Silicon Labs CP210x USB to UART Bridge (COMx)
   Uart1_FT232: USB Serial Port (COMx)
   (If USB Serial Port is not shown here, please refer to Appendix-2 and Appendix-3)
6. Device Manager> Ports(COM & LPT) > CP210x USB to UART (COMx)
   x: This is your ARC EM9D AIoT DK USB port number
   Please choose the USB serial port with CP2102 by default.



Run Application on ARC EM9D AIoT DK

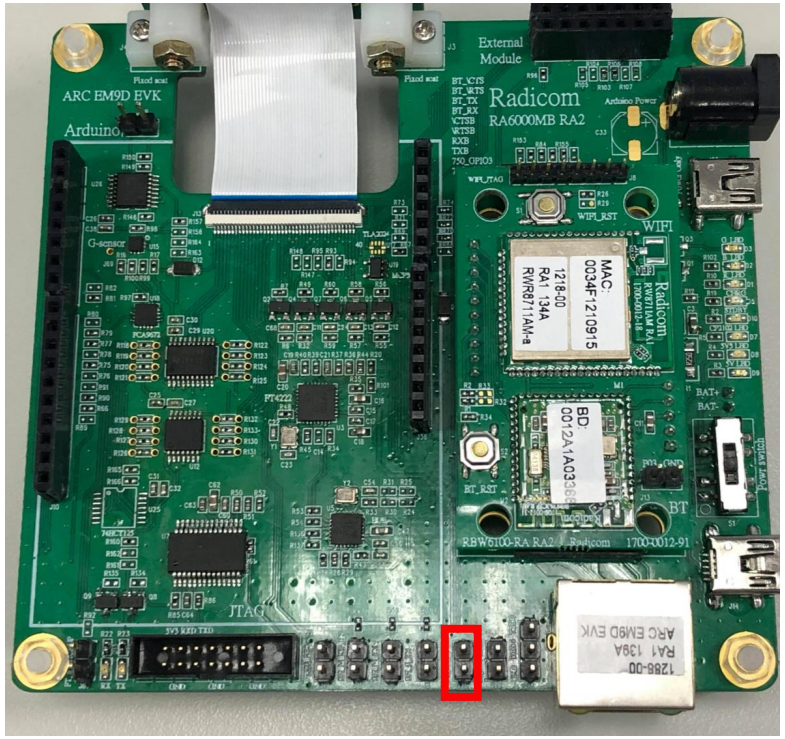
7. Open tera term and select “COMx: CP210x USB to UART (COMx)
8. Tera term *“Setup”* > *“Serial Port”* > *Change Baud to 115200*
   No need to change other settings.

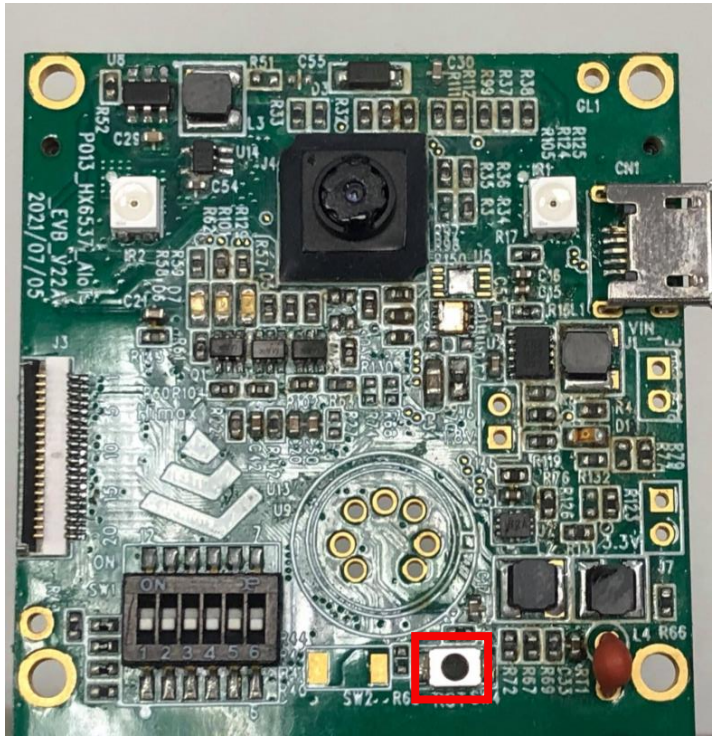




Run Application on ARC EM9D AIoT DK

9. Open J20 for run mode
10. Press reset button SW4. MCU will reset and run the application





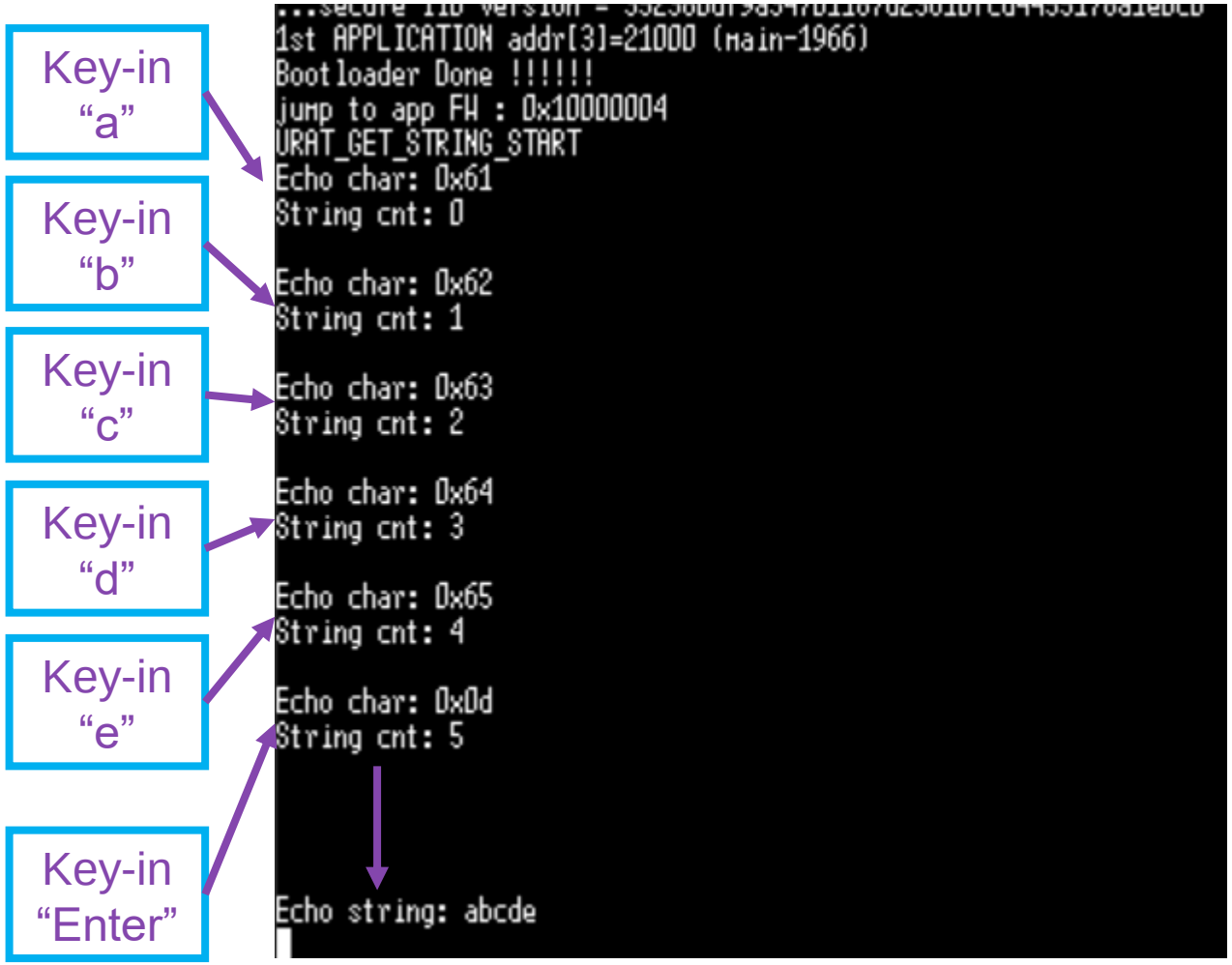
```
--------------------------------------------------------------
 Himax WEI Boot loader
--------------------------------------------------------------

embARC Build Time: Jan  4 2021, 13:44:14
Compiler Version: Metaware, 4.2.1 Compatible Clang 8.0.1
Boot loader Version : 1.4.4 (Date:Jan  4 2021)
chip version : 0x8535a1
cpu speed : 400000000 hz
spi speed : 50000000 hz
wake up evt:4
...secure lib version = 352380df9a347b1187d2361bfcd4455178a1ebcb
1st APPLICATION addr[3]=21000 (main-1966)
Bootloader Done !!!!!!
jump to app FW : 0x10000004
12 bytes lost due to alignment. To avoid this loss, please make su
HM0360 RevB,C,D Config
person score:-2 no person score 2
person score:-6 no person score 6
```

Lab1: UART

- This example project will echo every word you key-in
- After you key-in “Enter”

  It will return the string you key-in

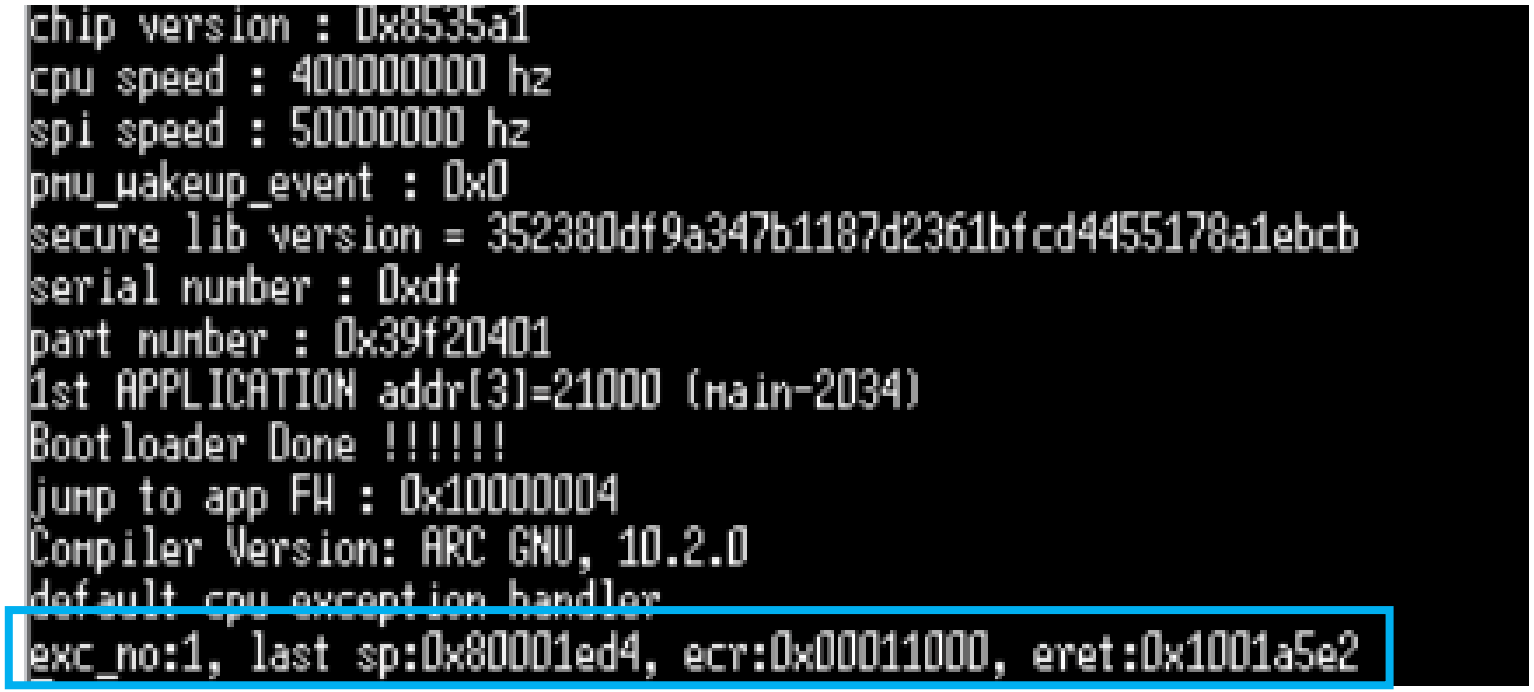


Lab1: UART

- If you get this error message after program and reset CPU.

  You can try to turn off all power and turn on again.

  If it still shows error message, please program your flash again!!!



Hands-on (Lab 1): GPIO (Not Arduino Extension GPIO)

Lab1: GPIO

- Header File: hx_drv_iomux.h
- Initialize GPIO

```
IOMUX_ERROR_E hx_drv_iomux_init(void);
// GPIO initial API, should be called first before you use GPIO

e.g.   hx_drv_iomux_init(); //Initialize GPIO
```

Lab1: GPIO

- Set direction

```
    IOMUX_ERROR_E hx_drv_iomux_set_pmux(IOMUX_INDEX_E aIndex, uint8_t aConfig);
// (IOMUX_INDEX_E aIndex) options are bellow
    IOMUX_PGPIO0     /* Select GPIO number 0  */
    IOMUX_PGPIO1     /* Select GPIO number 1  */
                  |
    IOMUX_PGPIO14    /* Select GPIO number 14  */

// (uint8_t aConfig) options are bellow
    aConfig = 2      /* Set direction is Input  */
    aConfig = 3      /* Set direction is Output  */

e.g    hx_drv_iomux_set_pmux(IOMUX_PGPIO1, 2); //Set CPU Board IR Sensor GPIO Input
       hx_drv_iomux_set_pmux(IOMUX_PGPIO6, 3); //Set CPU Board LED GPIO Output
```

Lab1: GPIO

- Set GPIO stage

```
   IOMUX_ERROR_E hx_drv_iomux_set_outvalue(IOMUX_INDEX_E aIndex, uint8_t aValue);
// (IOMUX_INDEX_E aIndex) options are bellow
   IOMUX_PGPIO0     /* Select GPIO number 0  */
   IOMUX_PGPIO1     /* Select GPIO number 1  */
                 |
   IOMUX_PGPIO14    /* Select GPIO number 14  */

// (uint8_t aValue) options are bellow
   aValue = 0       /* Set GPIO output Low */
   aValue = 1       /* Set GPIO output High */

e.g.   hx_drv_iomux_set_outvalue(IOMUX_PGPIO6, 0); //Set GPIO6 = Low
       hx_drv_iomux_set_outvalue(IOMUX_PGPIO6, 1); //Set GPIO6 = High
```

Lab1: GPIO

- Get GPIO stage

```
   IOMUX_ERROR_E hx_drv_iomux_get_invalue(IOMUX_INDEX_E aIndex,uint8_t * aValue);
// (IOMUX_INDEX_E aIndex) options are bellow
   IOMUX_PGPIO0     /* Select GPIO number 0  */
   IOMUX_PGPIO1     /* Select GPIO number 1  */
                |
   IOMUX_PGPIO14    /* Select GPIO number 14  */

// (uint8_t * aValue) options are bellow
   *aValue = 0      /* Get GPIO stage is Low */
   *aValue = 1      /* Get GPIO stage is High */

e.g.   uint8_t io_buf;
       hx_drv_iomux_get_invalue(IOMUX_PGPIO1, &io_buf); //Read GPIO1 to io_buf
       if(io_buf == 0)  print("Gpio is Low");
       else             print("Gpio is High");
```

Lab1: GPIO

Conclusion

- Header File: hx_drv_iomux.h
- GPIO need initialize and set direction.

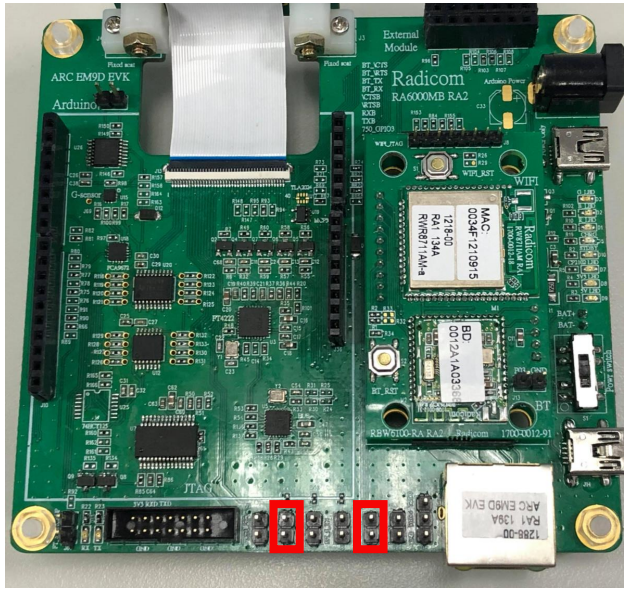
```
IOMUX_ERROR_E hx_drv_iomux_init(void);
IOMUX_ERROR_E hx_drv_iomux_set_pmux(IOMUX_INDEX_E aIndex, uint8_t aConfig);
```

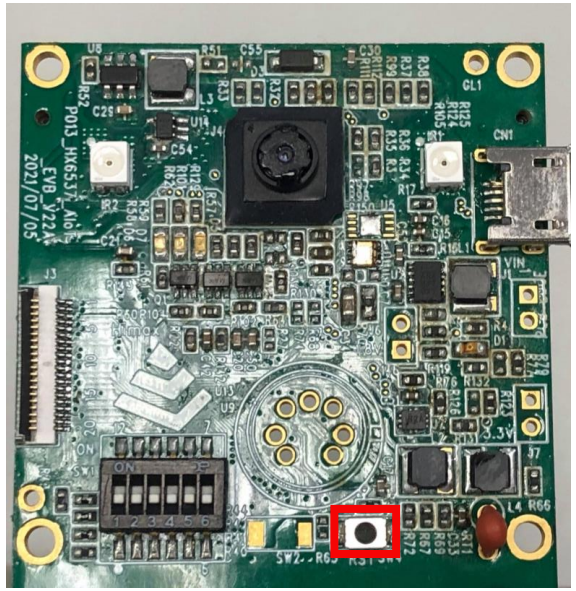
- After you initialize GPIO, you can set or get GPIO stage.

```
IOMUX_ERROR_E hx_drv_iomux_set_outvalue(IOMUX_INDEX_E aIndex, uint8_t aValue);
IOMUX_ERROR_E hx_drv_iomux_get_invalue(IOMUX_INDEX_E aIndex,uint8_t * aValue);
```

Lab1: GPIO

1. Short J20 and J11 for update application mode
2. Download image file to CPU
3. Open J20 for run mode
4. Press reset button SW4. MCU will reset and run the application





Lab1: GPIO

- This example project will print IR_GPIO_1 input stage
- LEDs on extension board will toggle.

Hands-on (Lab 2): I2C Master

Lab2: I2C Master

- Header File: hx_drv_iic_m.h
- Create I2C structural pointer

```
DEV_IIC * iic_x_ptr;
```

- Set I2C structural pointer to I2Cx

```
DEV_IIC_PTR hx_drv_i2cm_get_dev(USE_SS_IIC_E iic_id);
```

```
// (USE_SS_IIC_E iic_id) options are bellow
   USE_SS_IIC_0 = SS_IIC_0_ID       /* Select I2C Master 0*/
   USE_SS_IIC_1 = SS_IIC_1_ID       /* Select I2C Master 1  */
   USE_SS_IIC_2 = SS_IIC_2_ID       /* Select I2C Master 2  */
   USE_SS_IIC_ALL = SS_IIC_ALL_ID   /* Select I2C Master All  */

e.g.   DEV_IIC * iic0_ptr; //Create a structural pointer
       iic0_ptr = hx_drv_i2cm_get_dev(USE_SS_IIC_0); //Pointer will be I2C Master 0
```

Lab2: I2C Master

- Open I2C and set mode

```
    int32_t (*iic_open) (uint32_t mode, uint32_t param);

// (uint32_t mode) options are bellow
    DEV_MASTER_MODE         /* I2C select master mode */
    DEV_SLAVE_MODE          /* I2C select slave mode */

// (uint32_t param) options are bellow when mode is Master
    IIC_SPEED_STANDARD      /* I2C bit rate is 100kbps*/
    IIC_SPEED_FAST          /* I2C bit rate is 400kbps*/
    IIC_SPEED_FASTPLUS      /* I2C bit rate is 1Mbps*/

e.g.   iic0_ptr->iic_open(DEV_MASTER_MODE, IIC_SPEED_STANDARD);
       //Open I2C master with 100kbps
```

Lab2: I2C Master

- Write data to slave

```
int32_t hx_drv_i2cm_write_data(uint8_t iic_id, uint8_t slave_addr_sft, uint8_t
addr[], uint32_t addr_len, uint8_t data[], uint32_t data_len);
// (uint8_t iic_id)            Select I2C which you want to use
// (uint8_t slave_addr_sft)    Slave address
// (uint8_t addr[])            Don't need to use
// (uint32_t addr_len)         Don't need to use
// (uint8_t data[])            Write buffer pointer
// (uint32_t data_len)         How many byte need to write

e.g.   uint8_t data[3] = {slave_reg, data1, data2};
       hx_drv_i2cm_write_data(USE_SS_IIC_0, slave_addr, &data[0], 0, &data[0], 3);
   //Use I2C0 to write slave_addr, and data length is 2, start from slave_reg
```

Lab2: I2C Master

- Read data from slave

```
   int32_t hx_drv_i2cm_read_data(uint8_t iic_id, uint8_t slave_addr_sft, uint8_t
data[], uint32_t len);
// (uint8_t iic_id)           Select I2C which you want to use
// (uint8_t slave_addr_sft)   Slave address
// (uint8_t data[])           Read buffer pointer
// (uint32_t data_len)        How many byte need to read
// You need to use write API to configure which register will start to read
e.g.   uint8_t read_buf[10];
       uint8_t write_buf = reg_addr;
       hx_drv_i2cm_write_data(USE_SS_IIC_0, slave_addr, &write_buf, 0, &write_buf, 1);
   //Use I2C0 to write slave_addr, configure the register address = reg_addr

       hx_drv_i2cm_read_data(USE_SS_IIC_0, slave_addr, &read_buf[0], 6);
   //Read data from slave_addr, and data length is 6, start from reg_addr
```

Lab2: I2C Master

Conclusion

- Header File: hx_drv_iic_m.h
- I2C need to create a structural pointer and set pointer to I2Cx

```
DEV_IIC * iic_x_ptr;
DEV_IIC_PTR hx_drv_i2cm_get_dev(USE_SS_IIC_E iic_id);
```

- Should initialize and set mode, bit rate before send and get I2C data

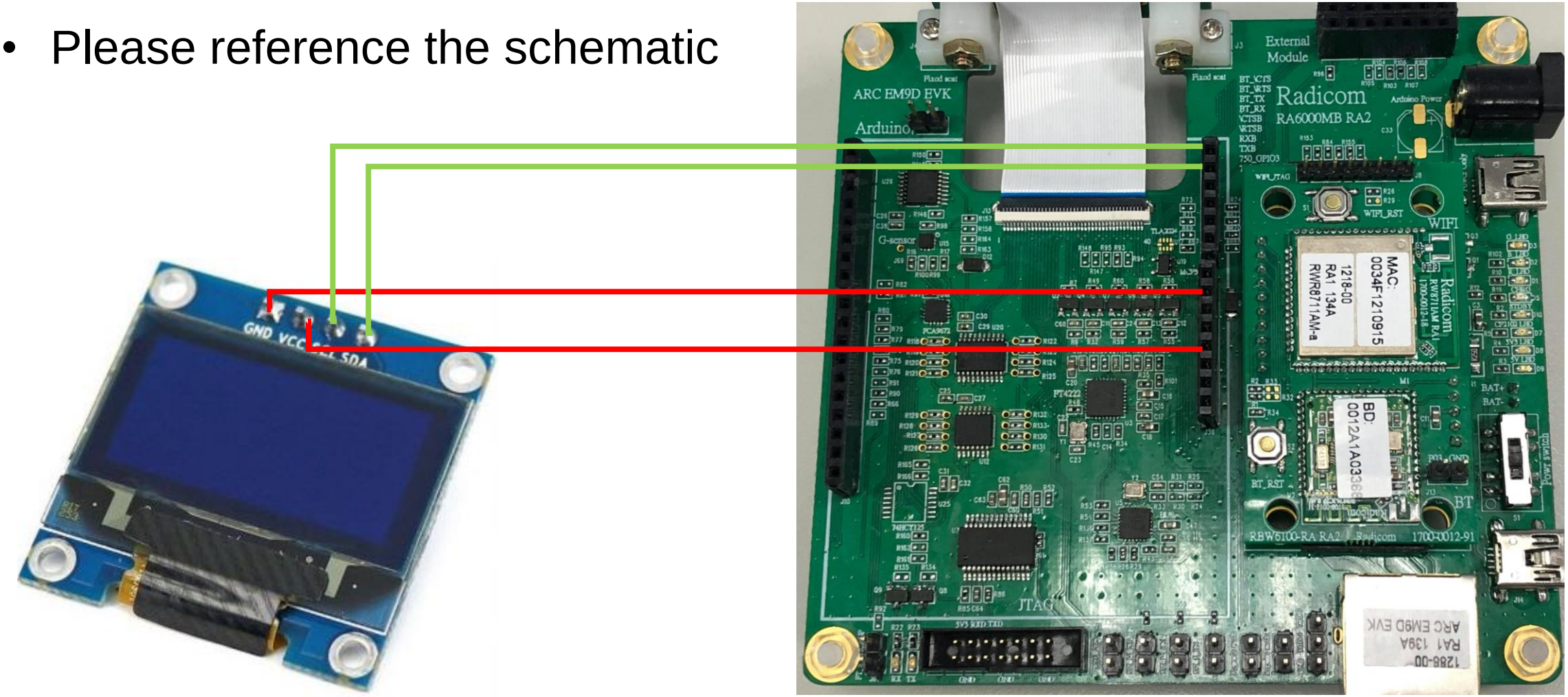
```
int32_t (*iic_open) (uint32_t mode, uint32_t param);
```

- After you initialize I2C, you can send and get I2C data

```
int32_t hx_drv_i2cm_write_data(uint8_t iic_id, uint8_t slave_addr_sft, uint8_t addr[], uint32_t addr_len, uint8_t data[], uint32_t data_len);
int32_t hx_drv_i2cm_read_data(uint8_t iic_id, uint8_t slave_addr_sft, uint8_t data[], uint32_t len);
```

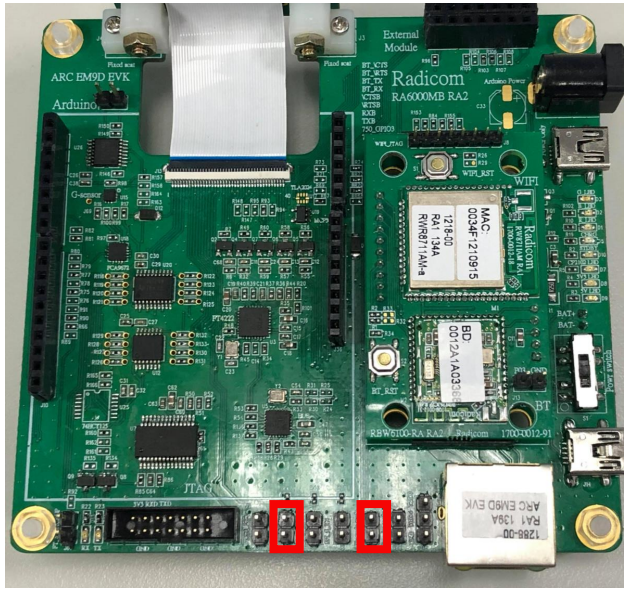
Lab2: I2C Master (OLED)

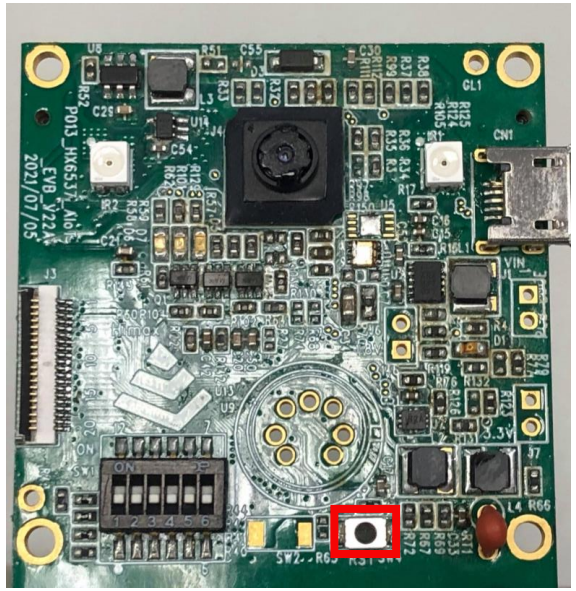
- Connect OLED1306 and ARC EM9D AIoT DK by 2.54 header
- Please reference the schematic



Lab2: I2C Master (OLED)

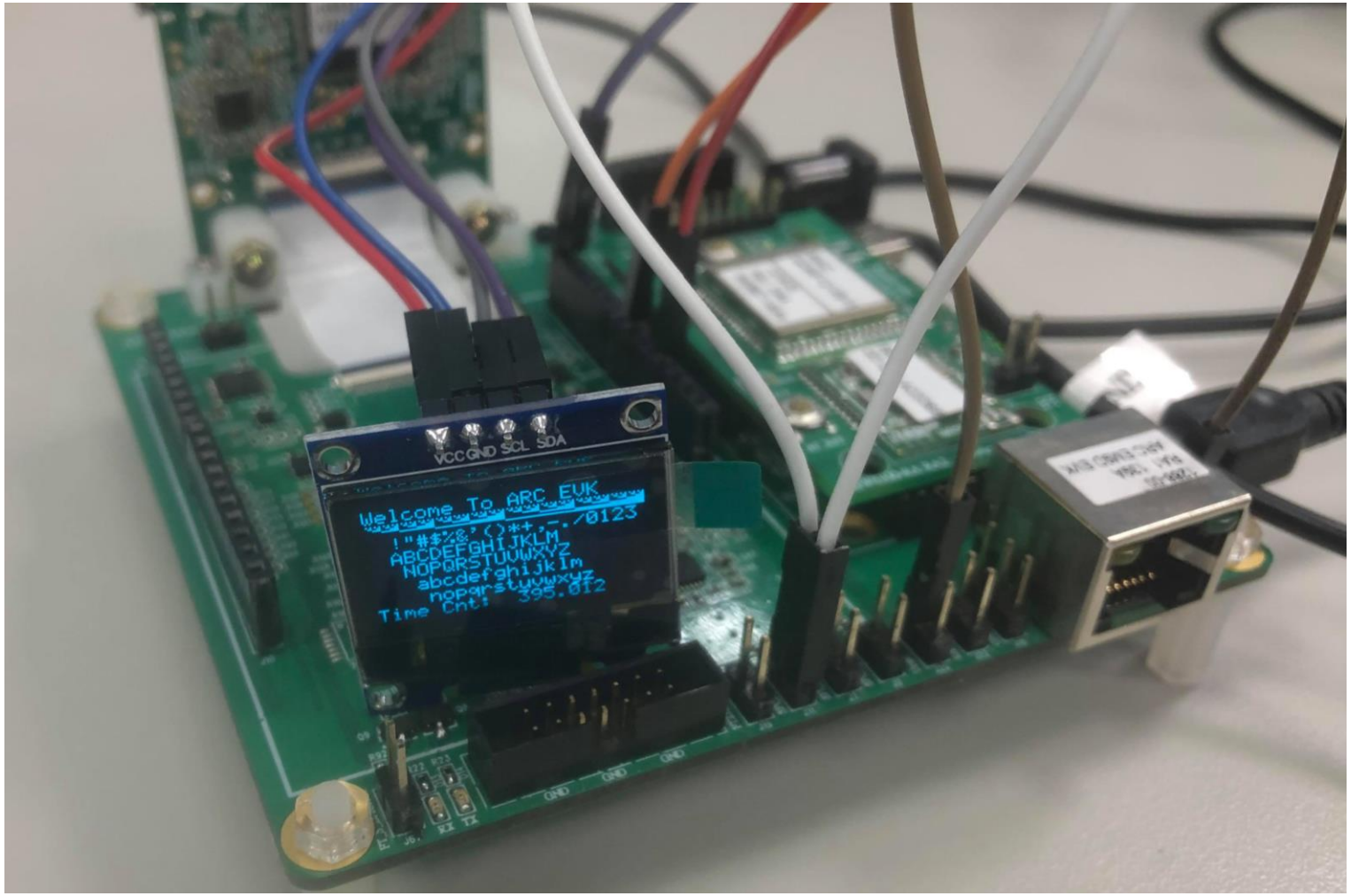
1. Short J20 and J11 for update application mode
2. Download image file to CPU
3. Open J20 for run mode
4. Press reset button SW4. MCU will reset and run the application





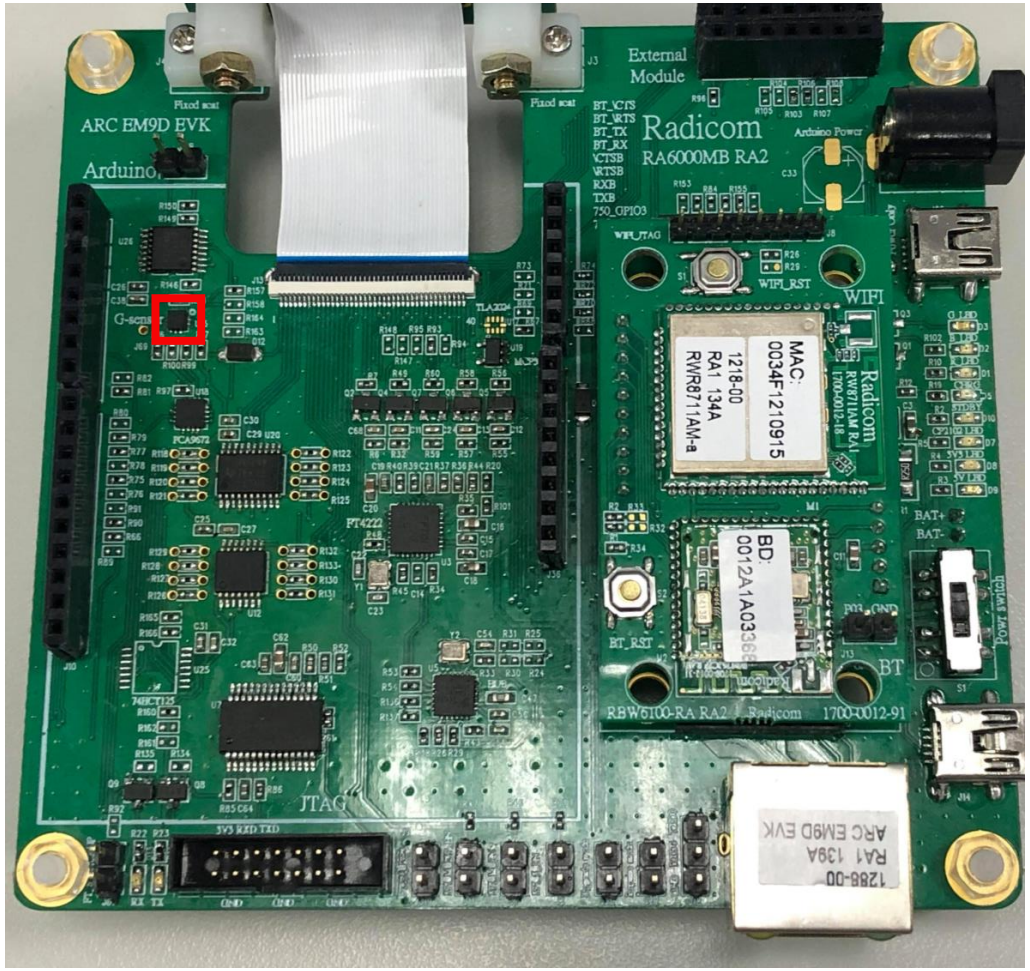
Lab2: I2C Master (OLED)

- This example project will print string on OLED1306.



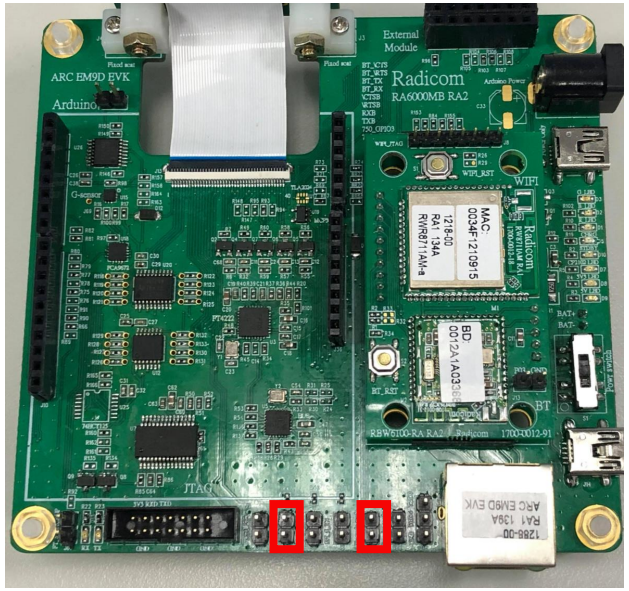
Lab2: I2C Master (Accelerometer)

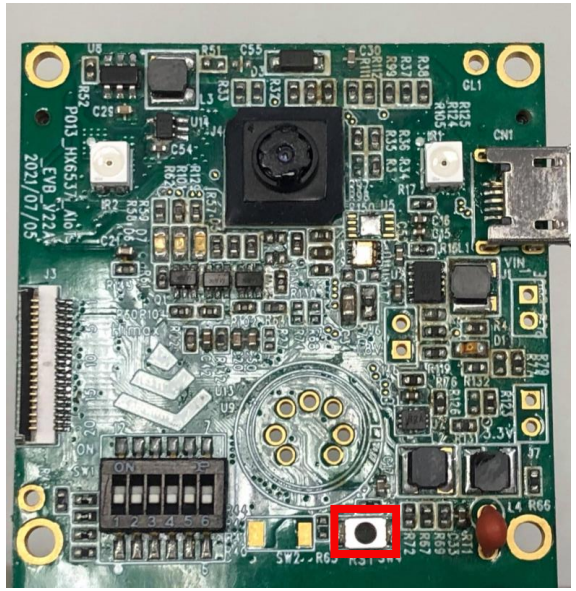
- Accelerometer is already mounted on board



Lab2: I2C Master (Accelerometer)

1. Short J20 and J11 for update application mode
2. Download image file to CPU
3. Open J20 for run mode
4. Press reset button SW4. MCU will reset and run the application





Lab2: I2C Master (Accelerometer)

- This example project will show 3-axis acceleration and temperature

Hands-on (Lab 2): SPI Master (Arduino Extension GPIO)

Lab2: SPI Master

- Header File: hx_drv_spi_m.h
- Create SPI structural pointer

```
DEV_SPI * spi_x_ptr;
```

- Set SPI structural pointer to SPIx

```
DEV_SPI_PTR hx_drv_spi_mst_get_dev(USE_DW_SPI_MST_E spi_id);
```

```
// (USE_DW_SPI_MST_E spi_id) options are bellow
   USE_DW_SPI_MST_0 = DW_SPI_0_ID      /* Select SPI Master 0*/
   USE_DW_SPI_MST_1 = DW_SPI_1_ID      /* Select SPI Master 1*/
   USE_DW_SPI_MST_ALL = DW_SPI_MST_NUM /* Select SPI Master All  */

e.g.   DEV_SPI * spi_ptr; //Create a structural pointer
       spi_ptr = hx_drv_spi_mst_get_dev(USE_DW_SPI_MST_1);
//Pointer will be SPI master 1
```

Lab2: SPI Master

- Open SPI and set mode

```
    int32_t (*spi_open) (uint32_t mode, uint32_t param);

// (uint32_t mode) options are bellow
    DEV_MASTER_MODE         /* SPI select master mode */
    DEV_SLAVE_MODE          /* SPI select slave mode */

// (uint32_t param) options are bellow when mode is Master
     param  = bit_rate

e.g.   spi_ptr->spi_open(DEV_MASTER_MODE, 1000000);
       //Open SPI master with 100kbps
```

Lab2: SPI Master

- Configure SPI protocol

```
   int32_t (*spi_control) (uint32_t ctrl_cmd, void *param);
// (uint32_t ctrl_cmd) select SPI register to configure
// (void *param)       configure date to SPI register

e.g.   spi_ptr->spi_control(SPI_CMD_SET_CLK_MODE, SPI_CLK_MODE_0);
   //IDLE CLK = Low, CLK toggles in middle of first data bit

      spi_ptr->spi_control(SPI_CMD_SET_CLK_MODE, SPI_CLK_MODE_1);
   //IDLE CLK = Low, CLK toggles at start of first data bit

      spi_ptr->spi_control(SPI_CMD_SET_CLK_MODE, SPI_CLK_MODE_2);
   //IDLE CLK = High, CLK toggles in middle of first data bit

      spi_ptr->spi_control(SPI_CMD_SET_CLK_MODE, SPI_CLK_MODE_3);
   //IDLE CLK = High, CLK toggles at start of first data bit
```

Lab2: SPI Master

- Send data by SPI Master (Half duplex mode)

```
int32_t (*spi_write) (const void *data, uint32_t len);

// (const void *data)          Write buffer pointer
// (uint32_t len)       How many byte need to write

e.g.   uint8_t data_buf[2] = {0xaa, 0x01};
       spi_ptr->spi_write(&data_buf[0], 2);
       //Use SPI Master to write 2 bytes
```

Lab2: SPI Master

- Read data by SPI Master (Half duplex mode)

```
int32_t (*spi_read) (void *data, uint32_t len);

// (const void *data)          Write buffer pointer
// (uint32_t len)       How many byte need to write

e.g.   uint8_t data_buf[2] = {0};
       spi_ptr->spi_read(&data_buf[0], 1);
       //Use SPI Master to read 1 byte
```

Lab2: SPI Master

Conclusion

- Header File: hx_drv_spi_m.h
- SPI need to create a structural pointer and set pointer to SPIx

```
DEV_SPI * spi_x_ptr;
DEV_SPI_PTR hx_drv_spi_mst_get_dev(USE_DW_SPI_MST_E spi_id);
```

- Should initialize and set mode, bit rate before send and get SPI data

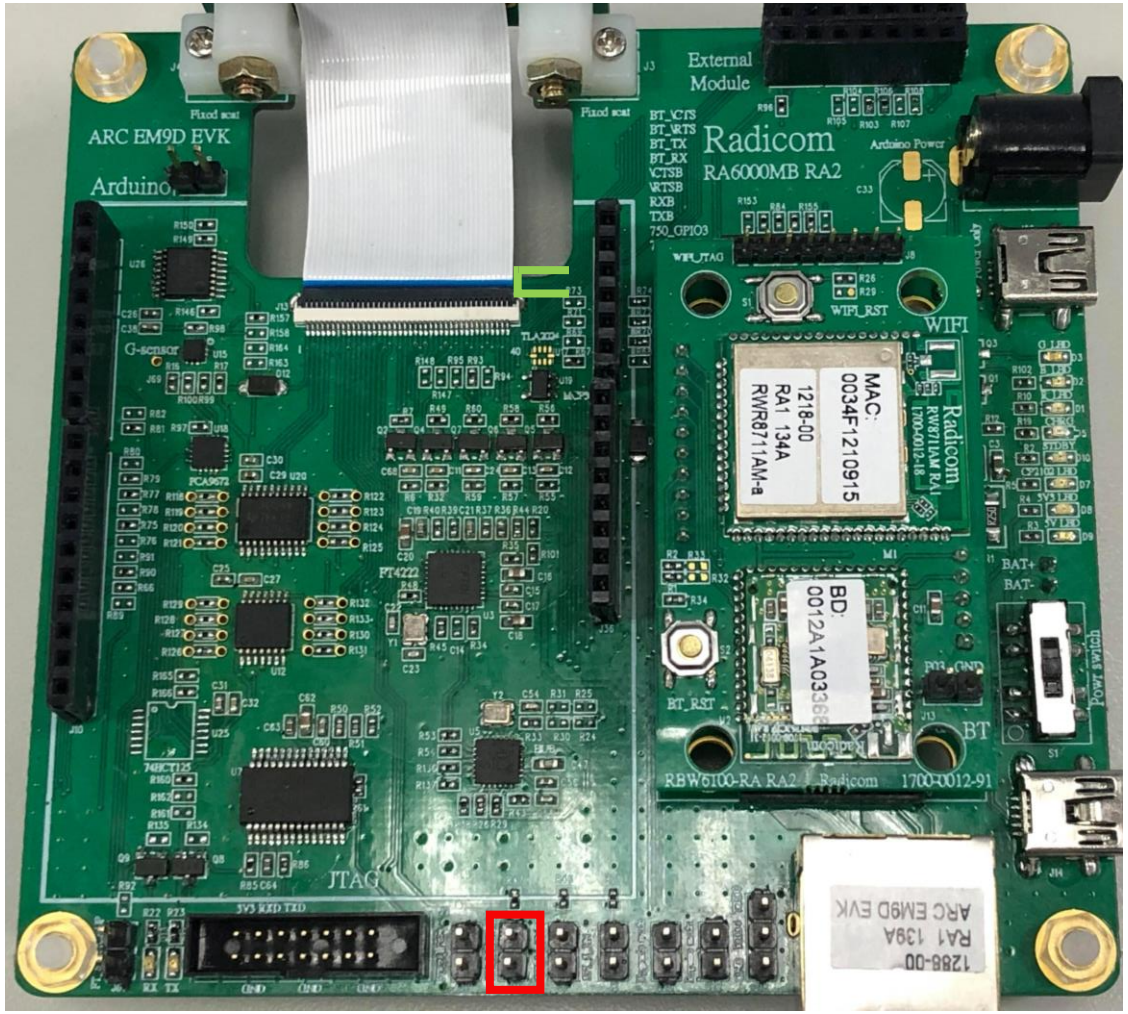
```
int32_t (*spi_open) (uint32_t mode, uint32_t param);
int32_t (*spi_control) (uint32_t ctrl_cmd, void *param);
```

- After you initialize SPI, you can send and get SPI data

```
int32_t (*spi_write) (const void *data, uint32_t len);
int32_t (*spi_read) (void *data, uint32_t len);
```

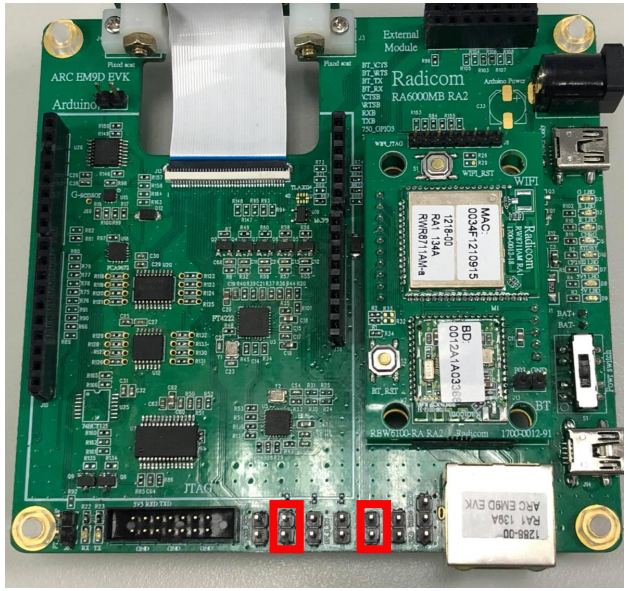
Lab2: SPI Master (Arduino Extension GPIO)

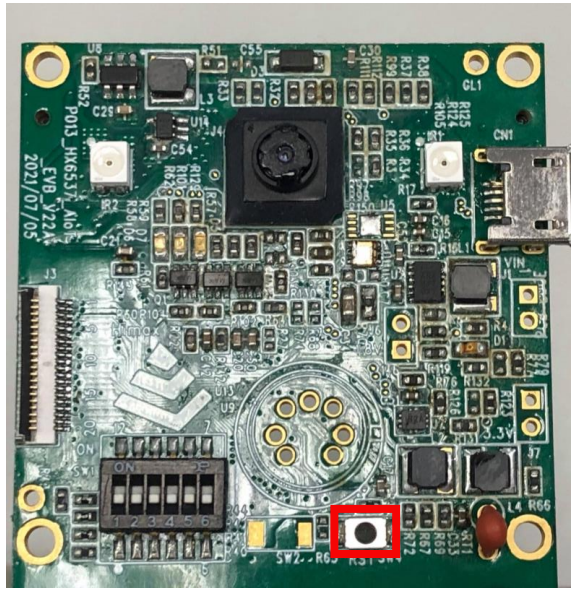
- Open J11 for SPI master control SC16IS752
- Short 752_GPIO7 and 752_GPIO6
- 752_GPIO7 is GPIO output
- 752_GPIO6 is GPIO output
- Please reference the schematic
- User can reference the example APIs and modify 752 serial GPIO according to your application.



Lab2: SPI Master (Arduino Extension GPIO)

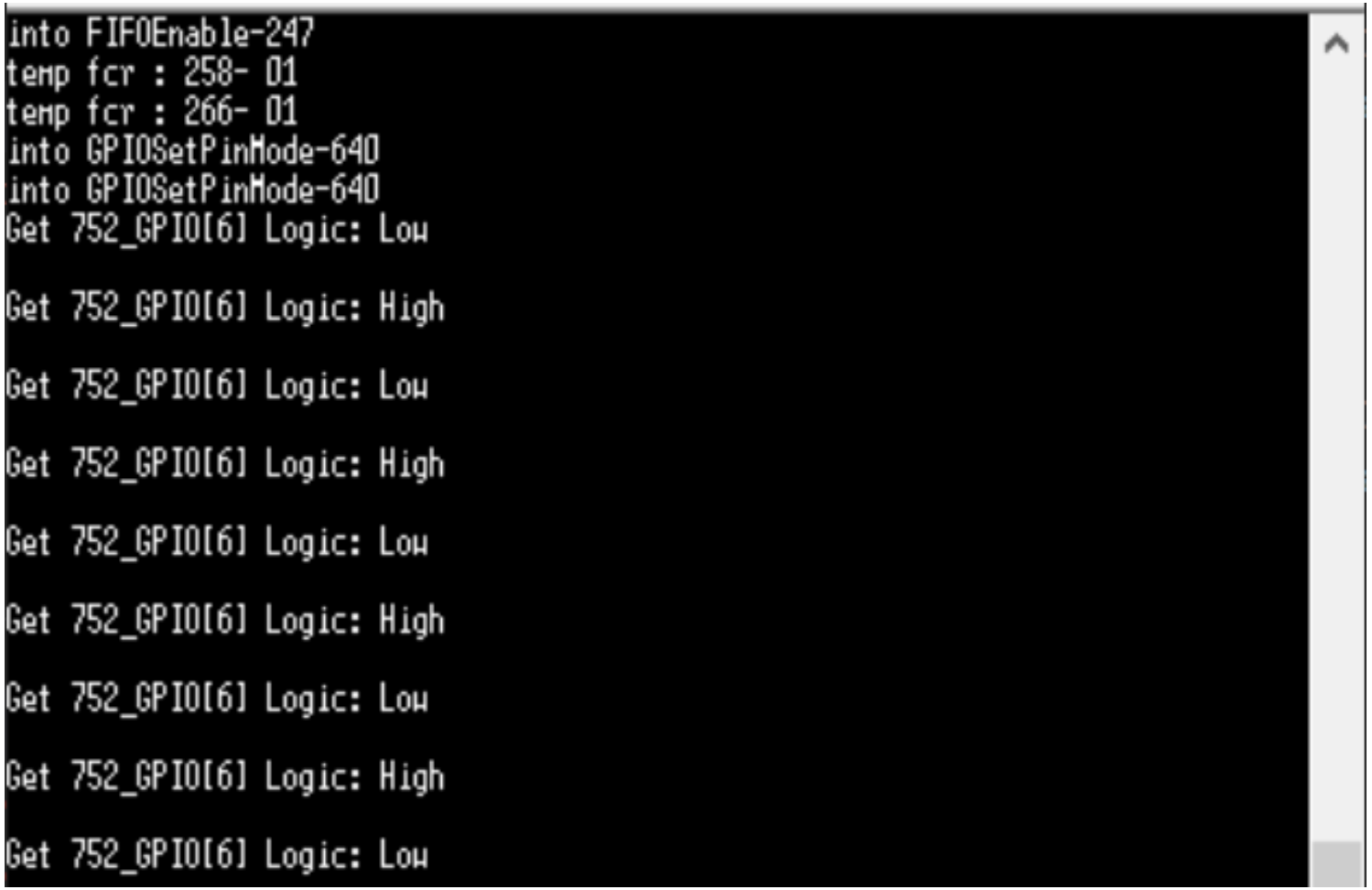
1. Short J20 and J11 for update application mode
2. Download image file to CPU
3. Open J20 for run mode, open J11 for SPI master control SC16IS752
4. Press reset button SW4. MCU will reset and run the application





Lab2: SPI Master (Arduino Extension GPIO)

- 752_GPIO[7] will toggle, and 752_GPIO[6] will get GPIO stage



Hands-on (Lab 3): PDM Microphone

Lab3: PDM Microphone

- Header File: aud_lib.h
- Initialize PDM interface for microphone

```
void hx_lib_audio_set_if(uint32_t aud_if);
AUDIO_ERROR_E hx_lib_audio_init(void);
```

- Register PDM event callback function

```
AUDIO_ERROR_E hx_lib_audio_register_evt_cb(AUD_ISR_CB aud_evt_cb);
```

```
// When hardware finish each block, CPU will jump to callback function.
// It is for user to process data and finite state machine(FSM).

e.g.   hx_lib_audio_set_if(AUDIO_IF_PDM);
       hx_lib_audio_init();
       hx_lib_audio_register_evt_cb(pdm_rx_callback_fun);
```

Lab3: PDM Microphone

- Configure PDM specification, DMA and start

```
audio_config_t aud_pdm_cfg;
AUDIO_ERROR_E hx_lib_audio_start(audio_config_t *aud_cfg);
```

```
e.g.   aud_pdm_cfg.sample_rate = AUDIO_SR_16KHZ;
       aud_pdm_cfg.buffer_addr = (uint32_t *) (0x20000000+36*1024);//0x2000A000;
       aud_pdm_cfg.block_num = (16 + 1);
       aud_pdm_cfg.block_sz = 1024 * 4;
       aud_pdm_cfg.cb_evt_blk = 2;
       hx_lib_audio_start(&aud_pdm_cfg);

// PDM microphone is dual channels, each channel is 16-bits (2bytes).
// Block size = 1024 * 4, means it has 1024 audio data each block.
// Each block audio length = 1024 / 16kHz = 0.064 second
// Total block = 16, and 1 block is for FSM.
// Total audio length each time = 0.064 * 16 = 1.024 second.
```

Lab3: PDM Microphone

- FSM and data process in event callback function
- Read the callback function stage

```
AUDIO_ERROR_E hx_lib_audio_request_read(uint32_t *address, uint32_t *block_num);
```

e.g.

```
uint32_t pdm_buf_addr, block;
uint32_t block;
hx_lib_audio_request_read(&pdm_buf_addr, &block);

{process audio data and copy to the other buffer}

if (block >= (aud_pdm_cfg.block_num - 1))
{
   last_block = 0;  //Re-count how many block is already processed.
   audio_flag = 1;  //Audio data ready
}
```

Lab3: PDM Microphone

Conclusion

- Header File: aud_lib.h
- PDM need to initialize, and register PDM event callback function

```
void hx_lib_audio_set_if(uint32_t aud_if);
AUDIO_ERROR_E hx_lib_audio_init(void);
AUDIO_ERROR_E hx_lib_audio_register_evt_cb(AUD_ISR_CB aud_evt_cb);
```

- Configure PDM specification, DMA and start

```
audio_config_t aud_pdm_cfg;
AUDIO_ERROR_E hx_lib_audio_start(audio_config_t *aud_cfg);
```

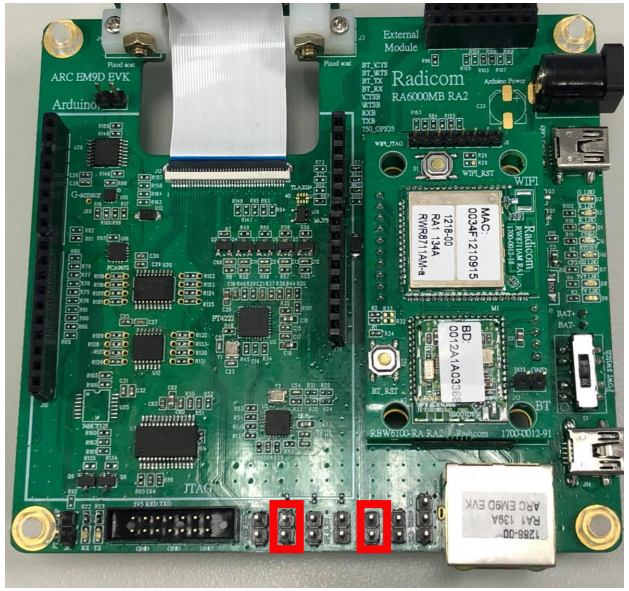
- Configure PDM specification, DMA and start

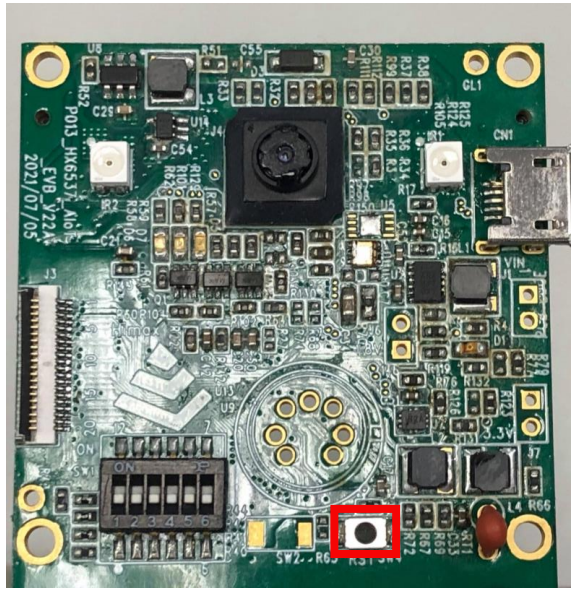
```
AUDIO_ERROR_E hx_lib_audio_request_read(uint32_t *address, uint32_t *block_num);
```

- When audio_flag == 1 means data is ready, and clean it manually

Lab3: PDM Microphone

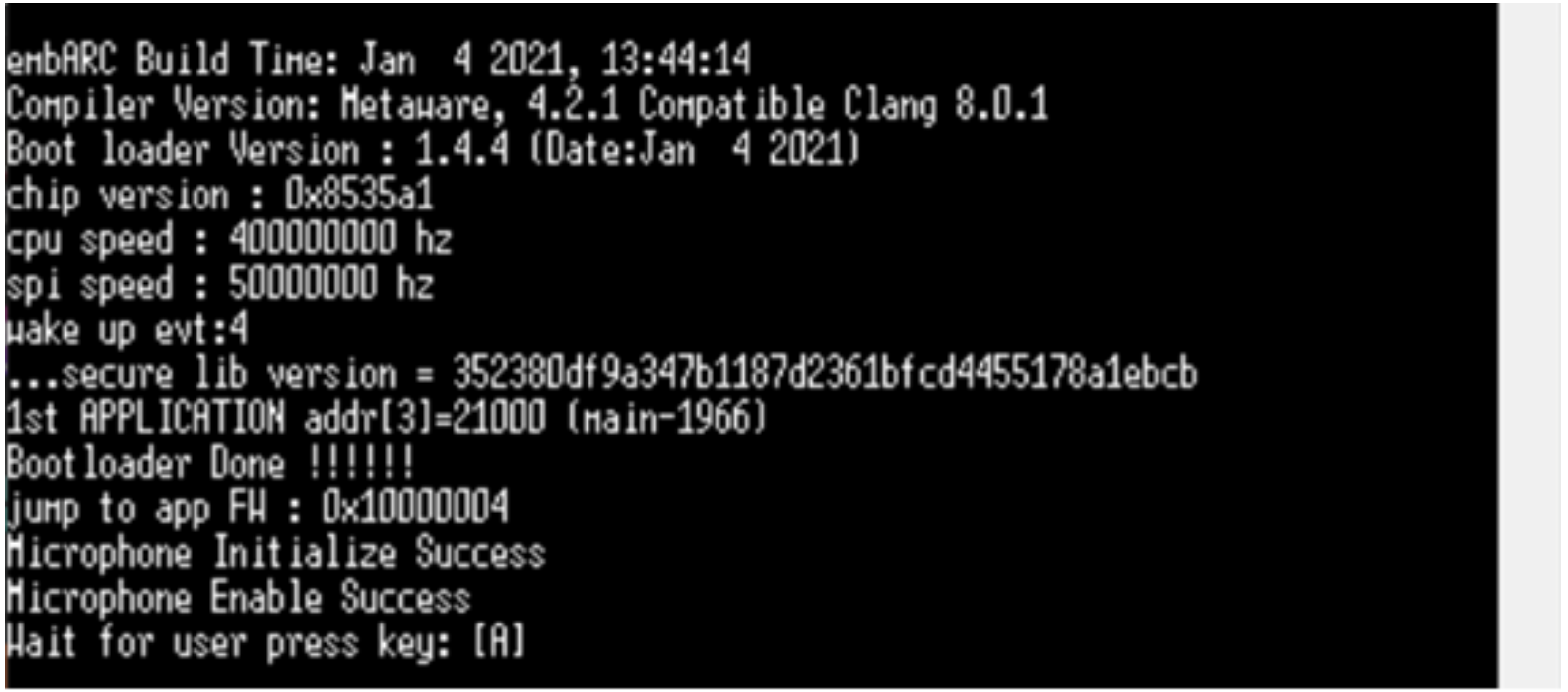
1. Short J20 and J11 for update application mode
2. Download image file to CPU
3. Open J20 for run mode
4. Press reset button SW4. MCU will reset and run the application





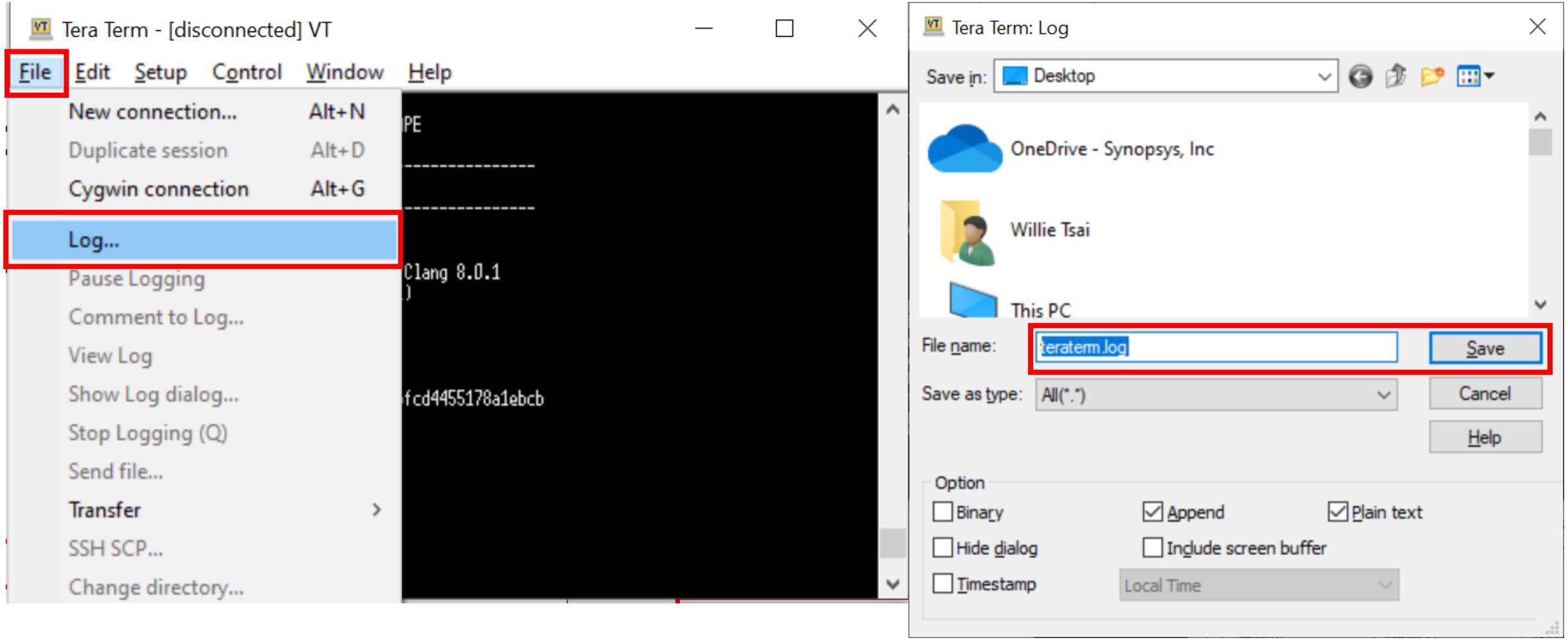
Lab3: PDM Microphone

- This example project will wait user key-in “A”, and then recode and send 10 seconds dual channel audio data. (Length = AUD_STEP_CNT * 1.024)
- You can save dual channel audio data by terminal log function



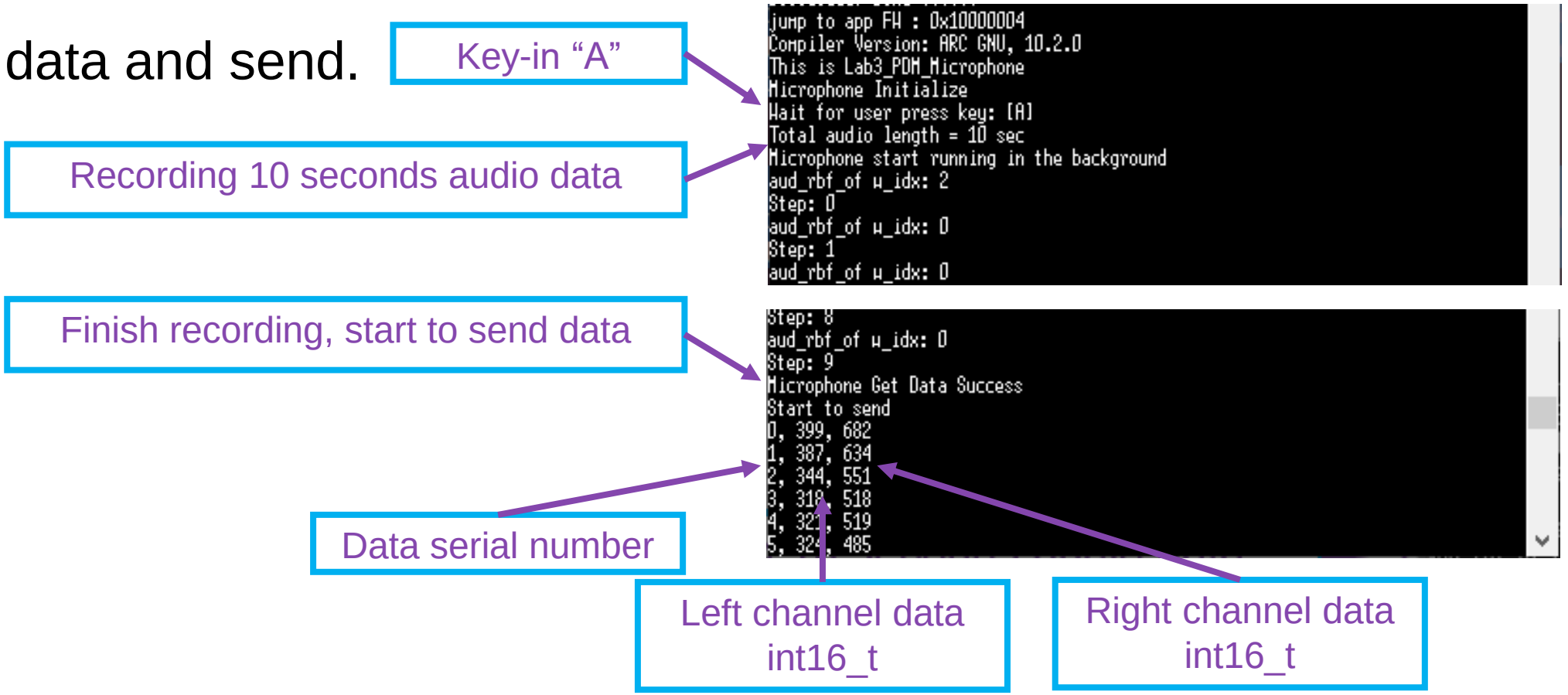
Lab3: PDM Microphone

- You can save terminal text to a log file



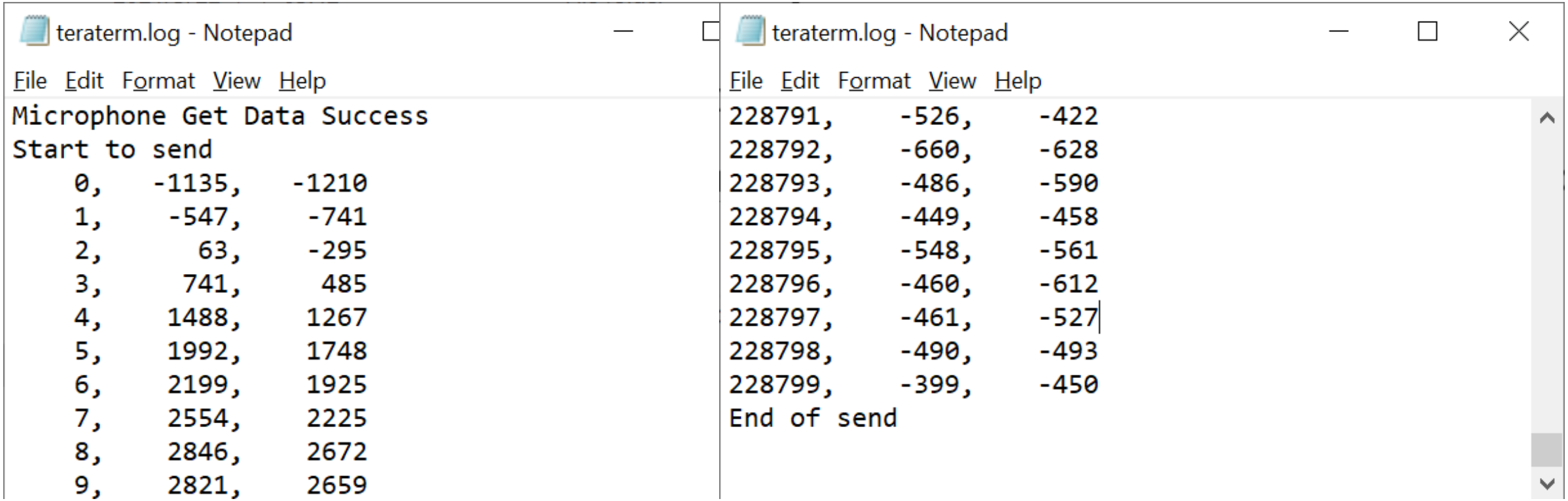
Lab3: PDM Microphone

- Now, it is recoding and saving log file.
- After key-in “A”, ARC EM9D AIoT DK starts to recode 10 seconds audio data and send.



Lab3: PDM Microphone

- It will take a lot of time to send data.
- After ARC EM9D AIoT DK finished sending, close terminal log function and you can open log file.

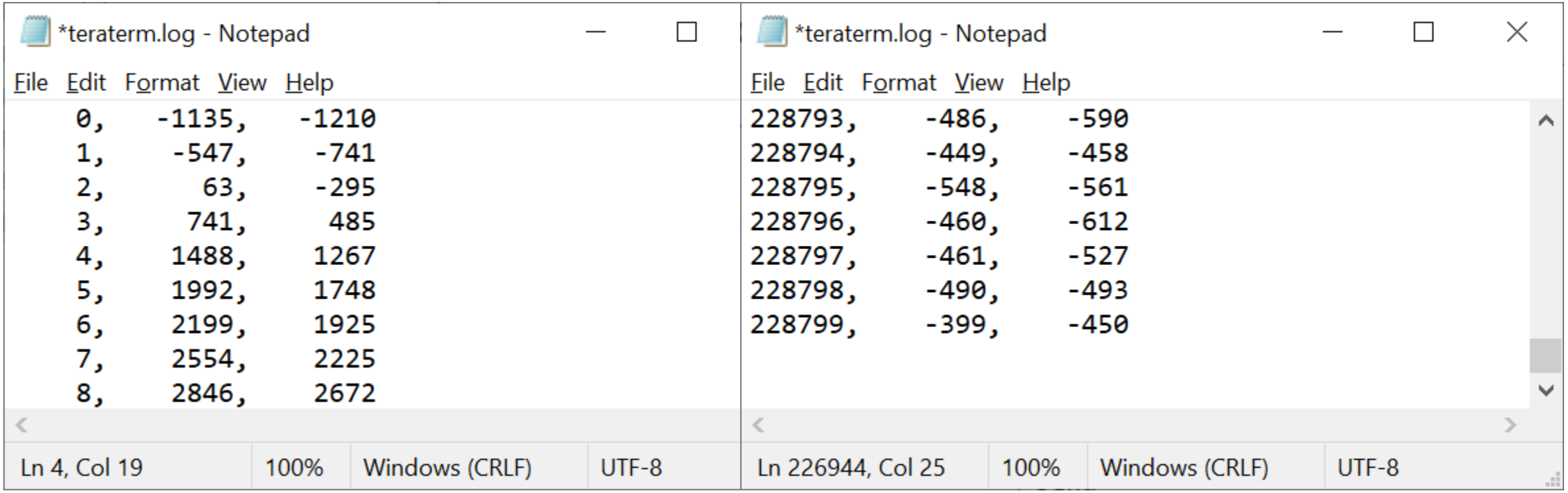


Lab3: PDM Microphone

- We also provide python code, help you convert log file to wav file

1. Delete non-audio-data message.

   (The top and the end of the log file)



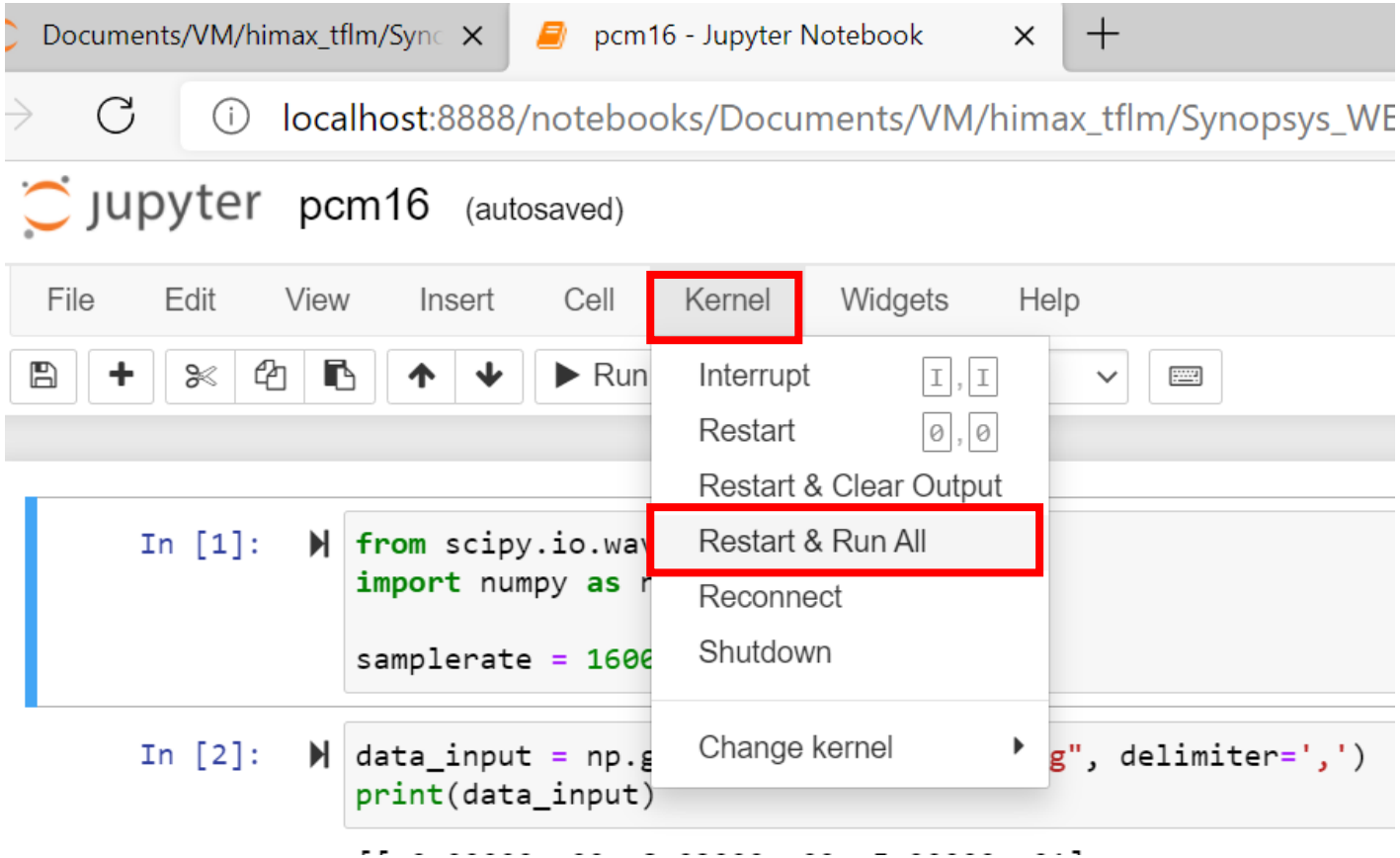
Lab3: PDM Microphone

2. Copy log file to “./LabPY_raw2wav/”
3. Rename log file to “pdm_dual.log”
4. Open Jupyter Notebook
5. Open “./LabPY_raw2wav/raw2wav.ipynb”

Lab3: PDM Microphone

6. Click “Restart and Run All”, then wav file will be saved

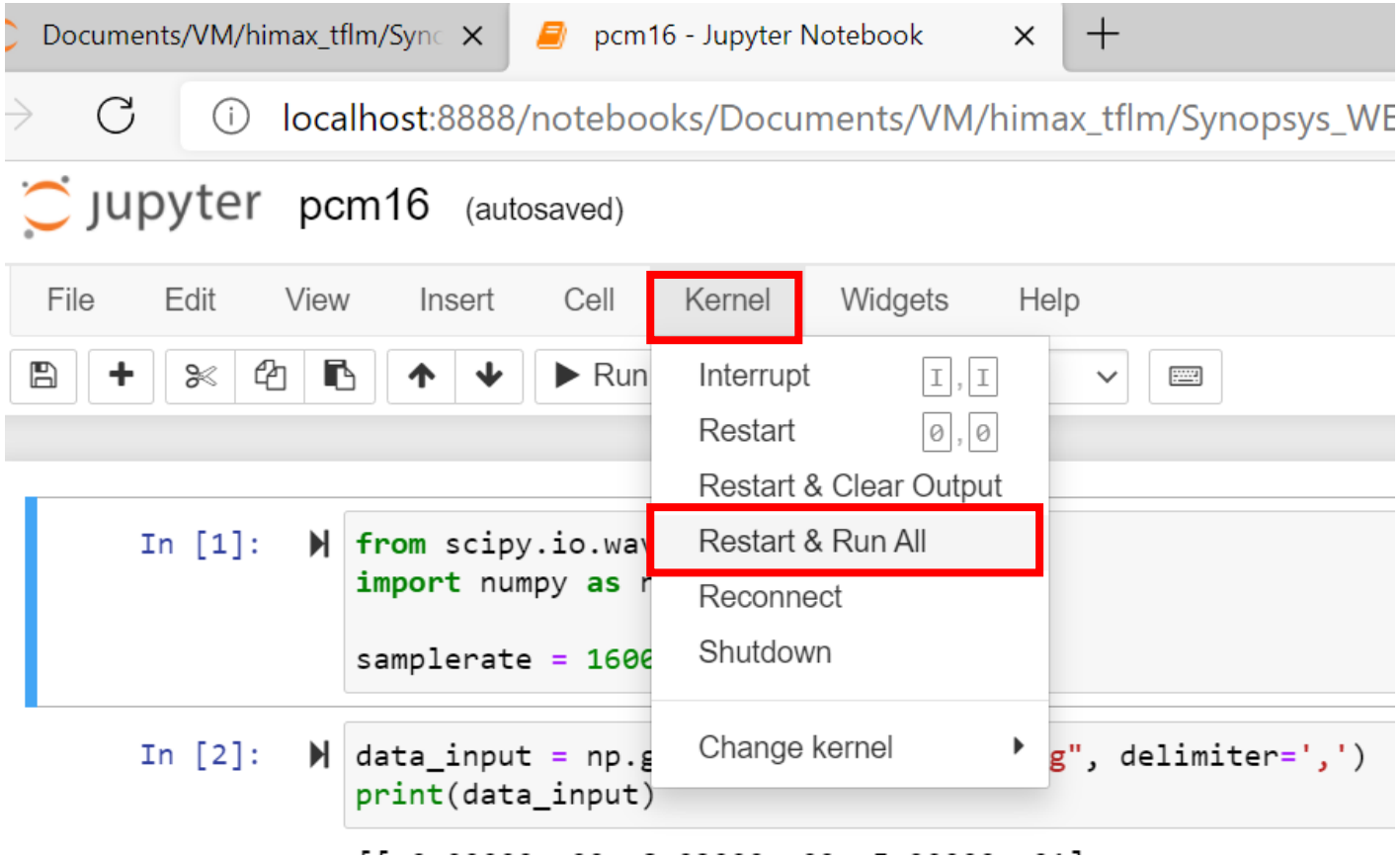
   path: “LabPY_raw2wav/pdm16_example.wav”



Lab3: PDM Microphone

7. Now you can play “pdm16_example.wav”

   You will know what microphone recorded in 10 seconds



Hands-on (Lab 3): Grayscale Camera

Lab3: Grayscale Camera

- This version project of Camera is using DMA to get image pixel.

  The program architecture is more complex, we suggest user modify example project for your application.
- Header File: synopsys_sdk_camera_drv.h & others…
- Initialize grayscale camera

```
void synopsys_camera_init (void);
// Image sensor initialization, query one JPEG and one RAW frame to target address.
// Current image sensor use is HM0360, image resolution is 640x480.
// Each pixel data is 8-bit.

e.g.   synopsys_camera_init();
```

Lab3: Grayscale Camera

- Get grayscale image (640x480)

```
void synopsys_camera_start_capture (void);
uint8_t * img_ptr = (uint8_t *) g_wdma2_baseaddr;
```

```
// Image data will in (uint8_t *) g_wdma2_baseaddr
// Need a dealy for waiting DMA

e.g.   synopsys_camera_start_capture();
       //Delay

       uint8_t * img_ptr;
       uint32_t img_width = 640;
       uint32_t img_height = 480;
       img_ptr = (uint8_t *) g_wdma2_baseaddr;
```

Lab3: Grayscale Camera

- Downscale if you need, it uses average pooling.

```
    void synopsys_camera_down_scaling (uint8_t * input_image, uint32_t input_width,
uint32_t input_height, uint8_t * output_image, uint32_t output_width, uint32_t output_height)
// (uint8_t * input_image)    Input image pointer
// (uint32_t input_width)     Input image width
// (uint32_t input_height)    Input image height
// (uint8_t * output_image)   Output image pointer
// (uint32_t output_width)    Output image width
// (uint32_t output_height)   Output image height

e.g.   synopsys_camera_down_scaling(&img_ptr[0], img_width, img_height,
&output_img[0], output_width, output_height);
```

Lab3: PDM Microphone

Conclusion

- Header File: synopsys_sdk_camera_drv.h & others…
- Camera need to initialize, and register event callback function

```
void synopsys_camera_init (void);
```

- Get grayscale image (640x480), and need to delay for waiting DMA

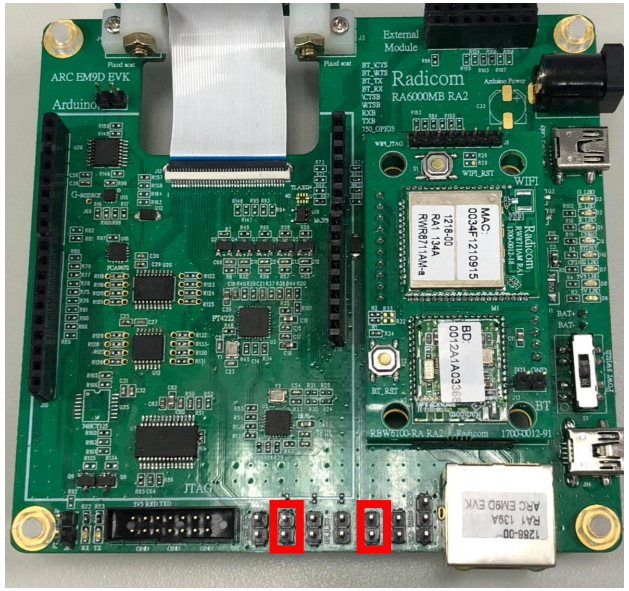
```
void synopsys_camera_start_capture (void);
uint8_t * img_ptr = (uint8_t *) g_wdma2_baseaddr;
```

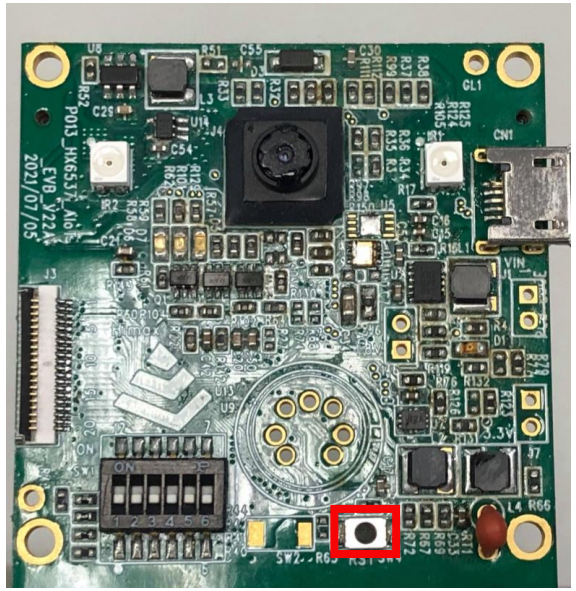
- Downscale if you need, it uses average pooling

```
void synopsys_camera_down_scaling (uint8_t * input_image, uint32_t input_width,
uint32_t input_height, uint8_t * output_image, uint32_t output_width, uint32_t output_height)
```

Lab3: Grayscale Camera

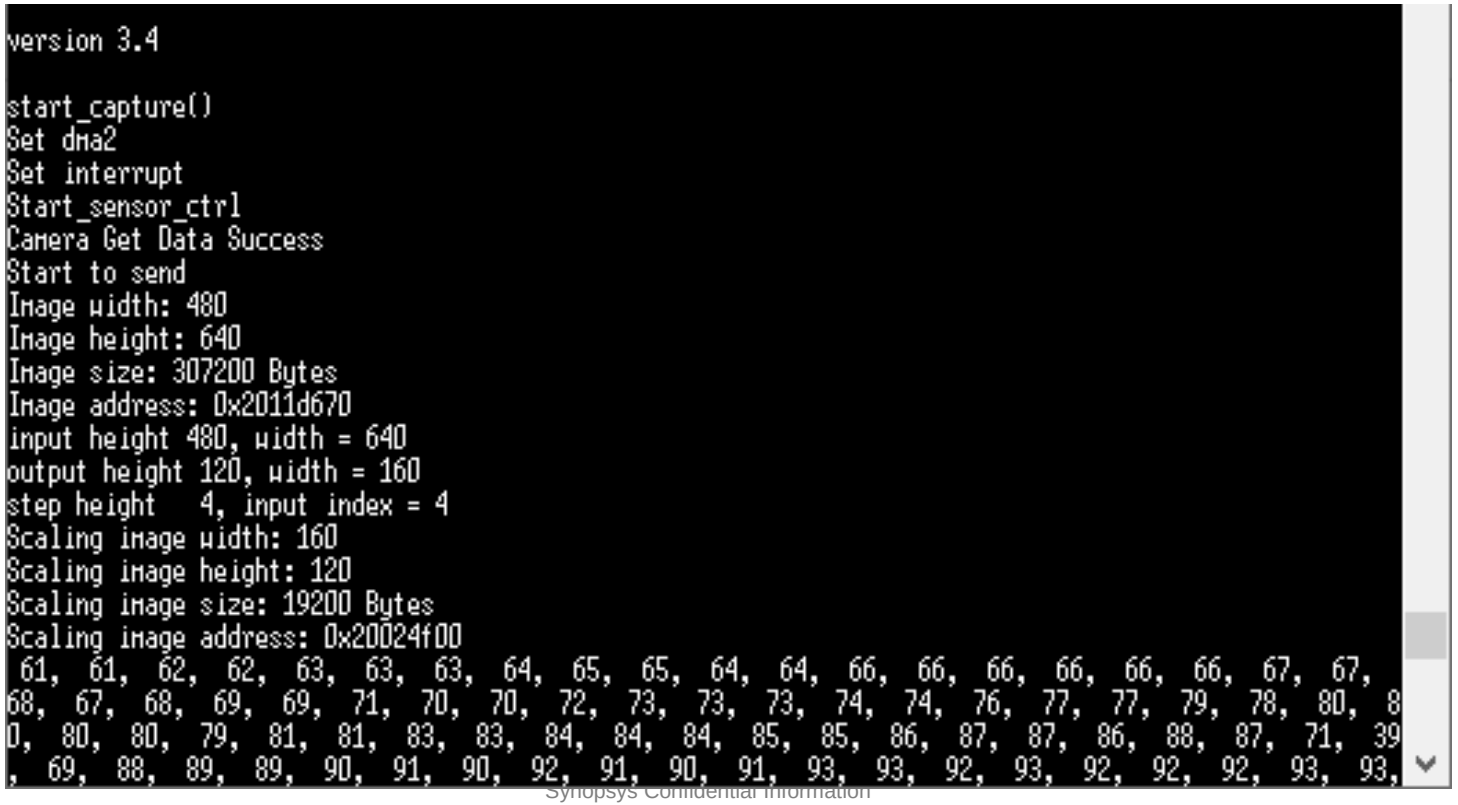
1. Short J20 and J11 for update application mode
2. Download image file to CPU
3. Open J20 for run mode
4. Press reset button SW4. MCU will reset and run the application





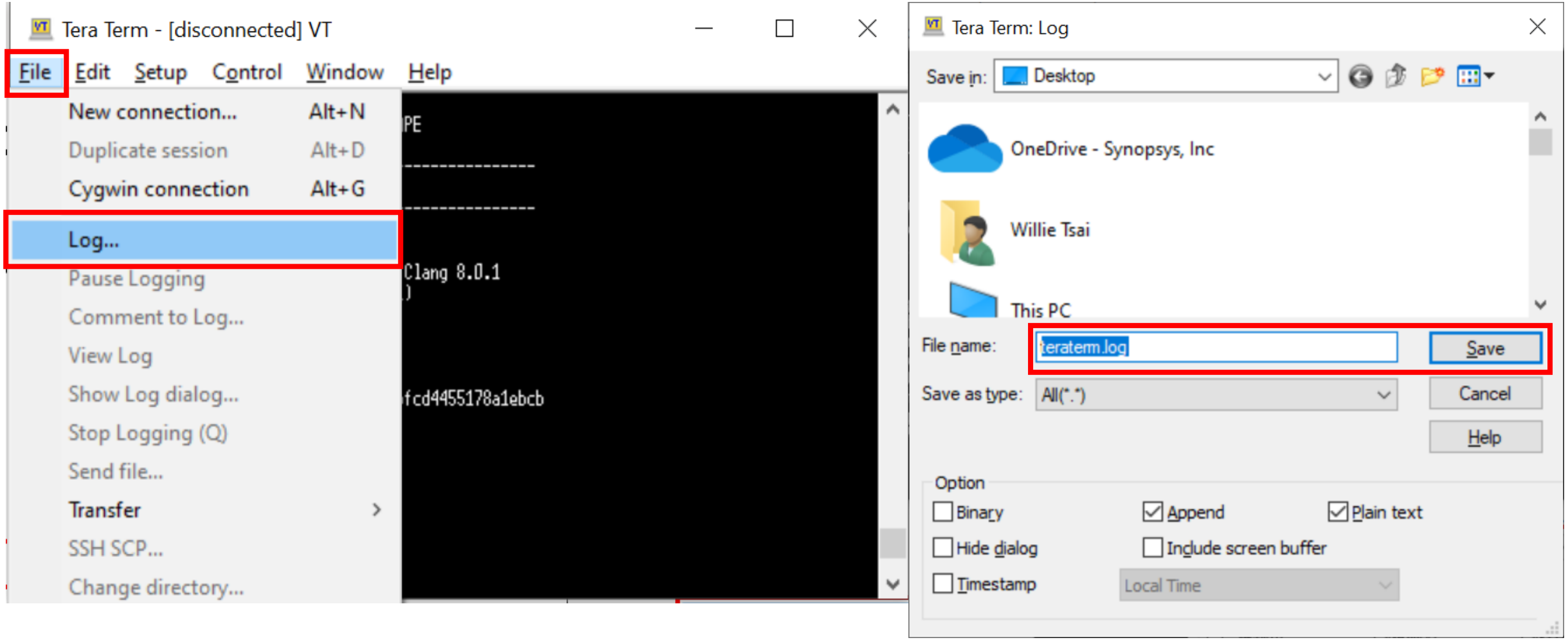
Lab3: Grayscale Camera

- This example project will wait until user key-in “A”, and then capture and send RAW image data.
- You can save RAW image data data by terminal log function



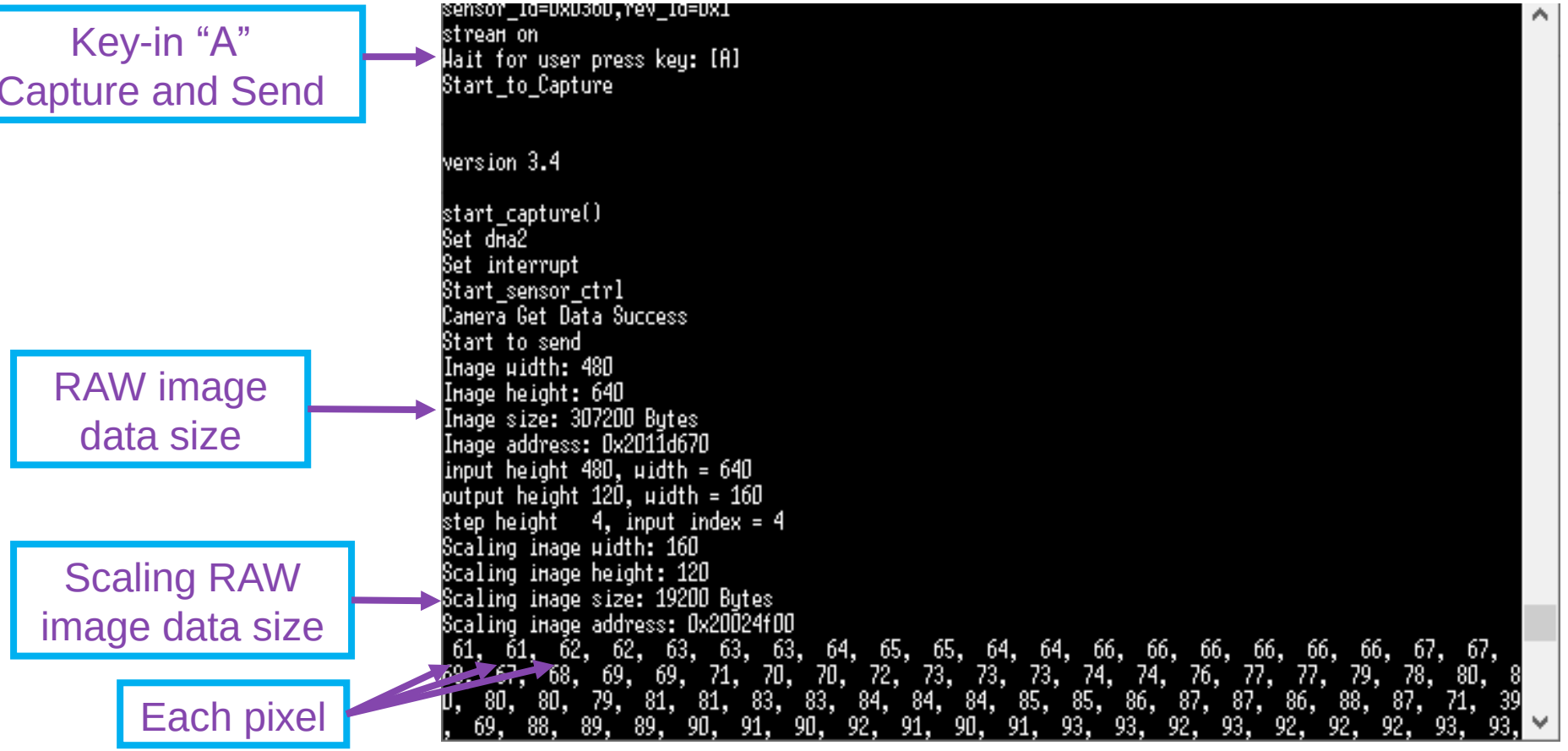
Lab3: Grayscale Camera

- You can save terminal text to a log file



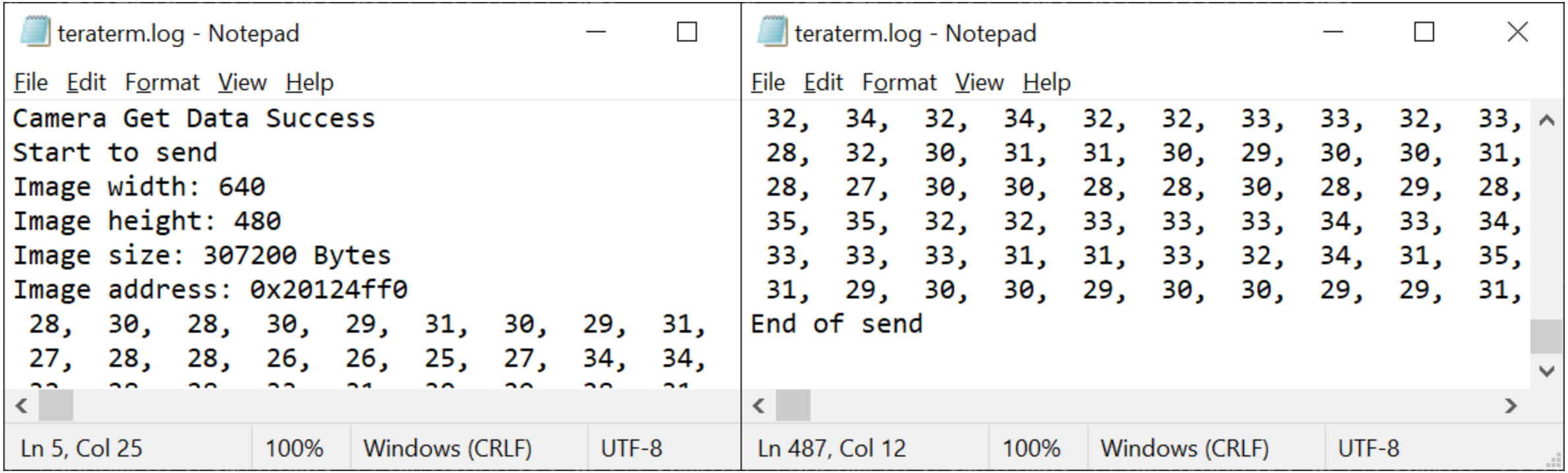
Lab3: Grayscale Camera

- Now, it is recoding and saving log file.
- After key-in “A”, ARC EM9D AIoT DK start to capture and send RAW image data.



Lab3: Grayscale Camera

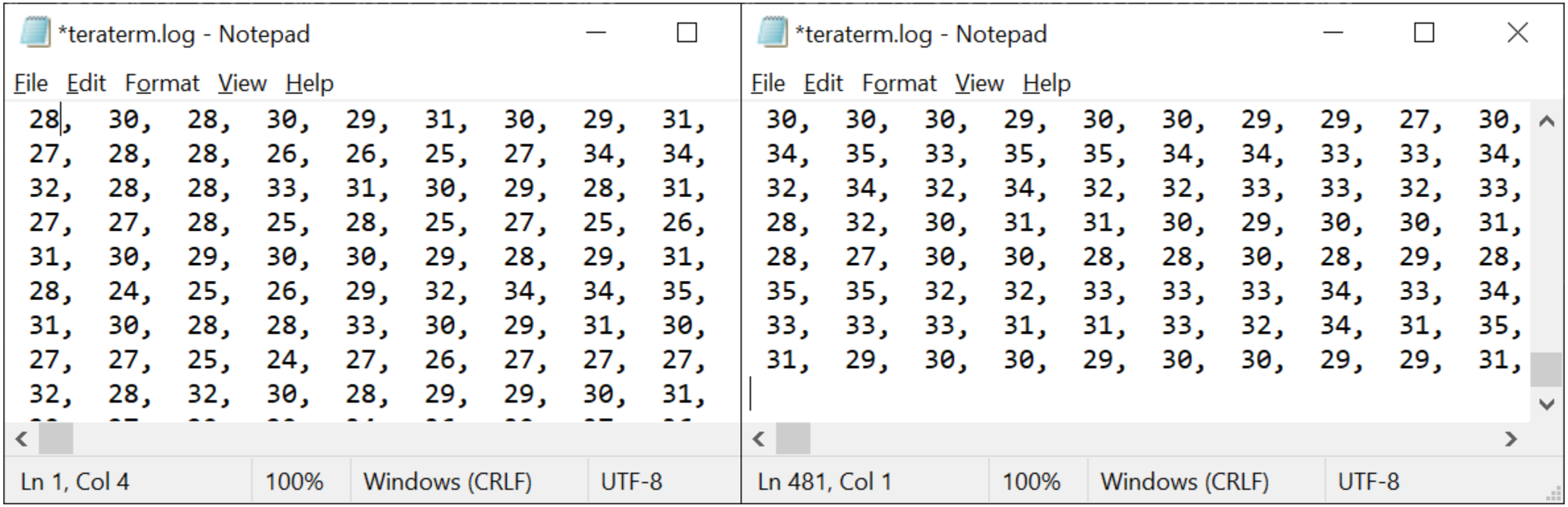
- It will take a lot of time to send data.
- After ARC EM9D AIoT DK finished sending, close terminal log function and you can open log file.



Lab3: Grayscale Camera

- We also provide python code, help you convert log file to png file

1. Delete non-pixel-data message. (The top and the end of the log file)



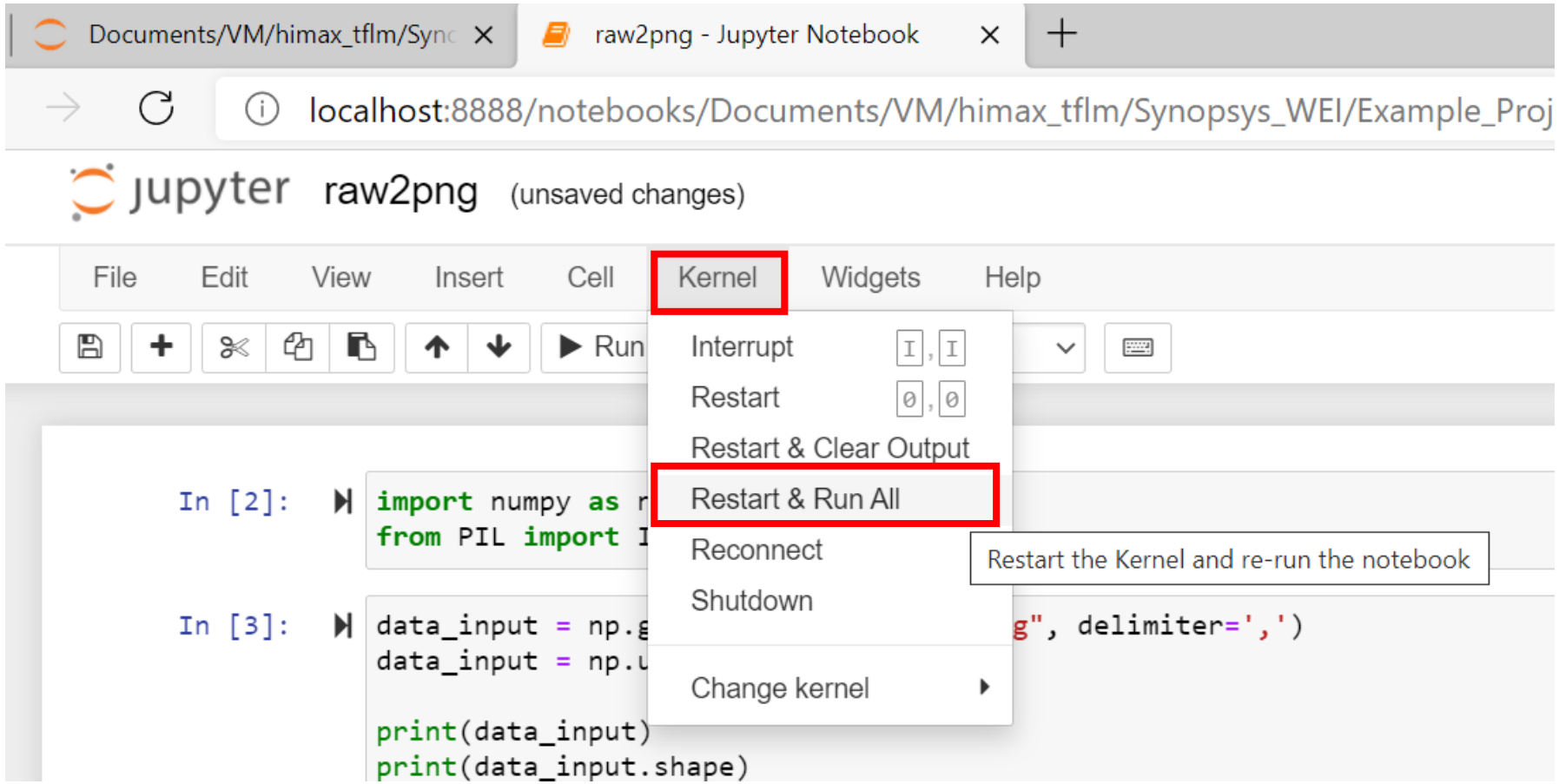
Lab3: Grayscale Camera

2. Copy log file to “./LabPY_raw2png/”
3. Rename log file to “camera_y.log”
4. Open Jupyter Notebook
5. Open “./LabPY_raw2png/raw2png.ipynb”

Lab3: Grayscale Camera

6. Click “Restart and Run All”, then png file will be saved

   path: “LabPY_raw2png/my_gray.png”



Lab3: Grayscale Camera

7. Now can show “my_gray.png”

   It will show what camera captured

Appendix-2: Troubleshooting - FTDI VCP Driver

Troubleshooting – FTDI VCP Driver

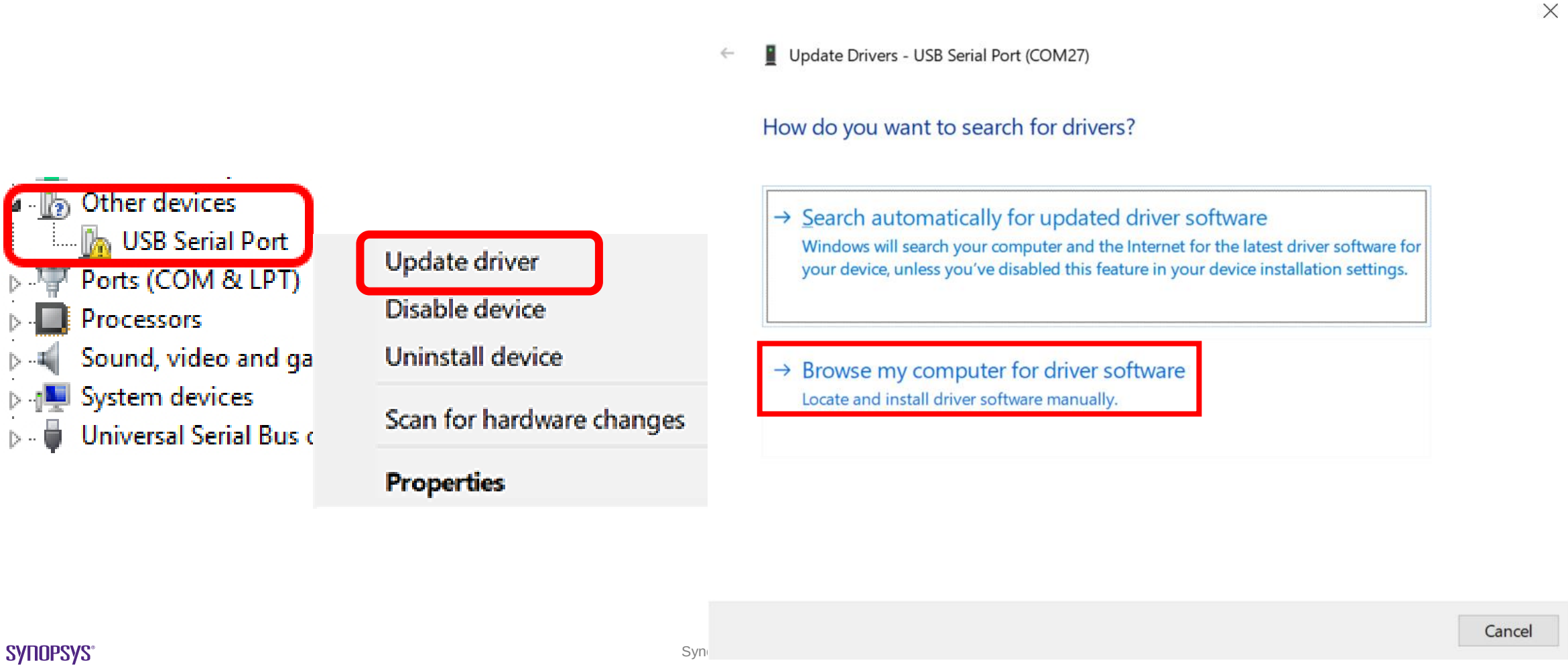
If the USB serial port is not shown in Ports (COM & LPT):

1. Download VCP driver: https://ftdichip.com/drivers/vcp-drivers/
   Select Windows/X64 version
2. Unzip the downloaded file (CDM v2.XX.XX WHQL Certified)

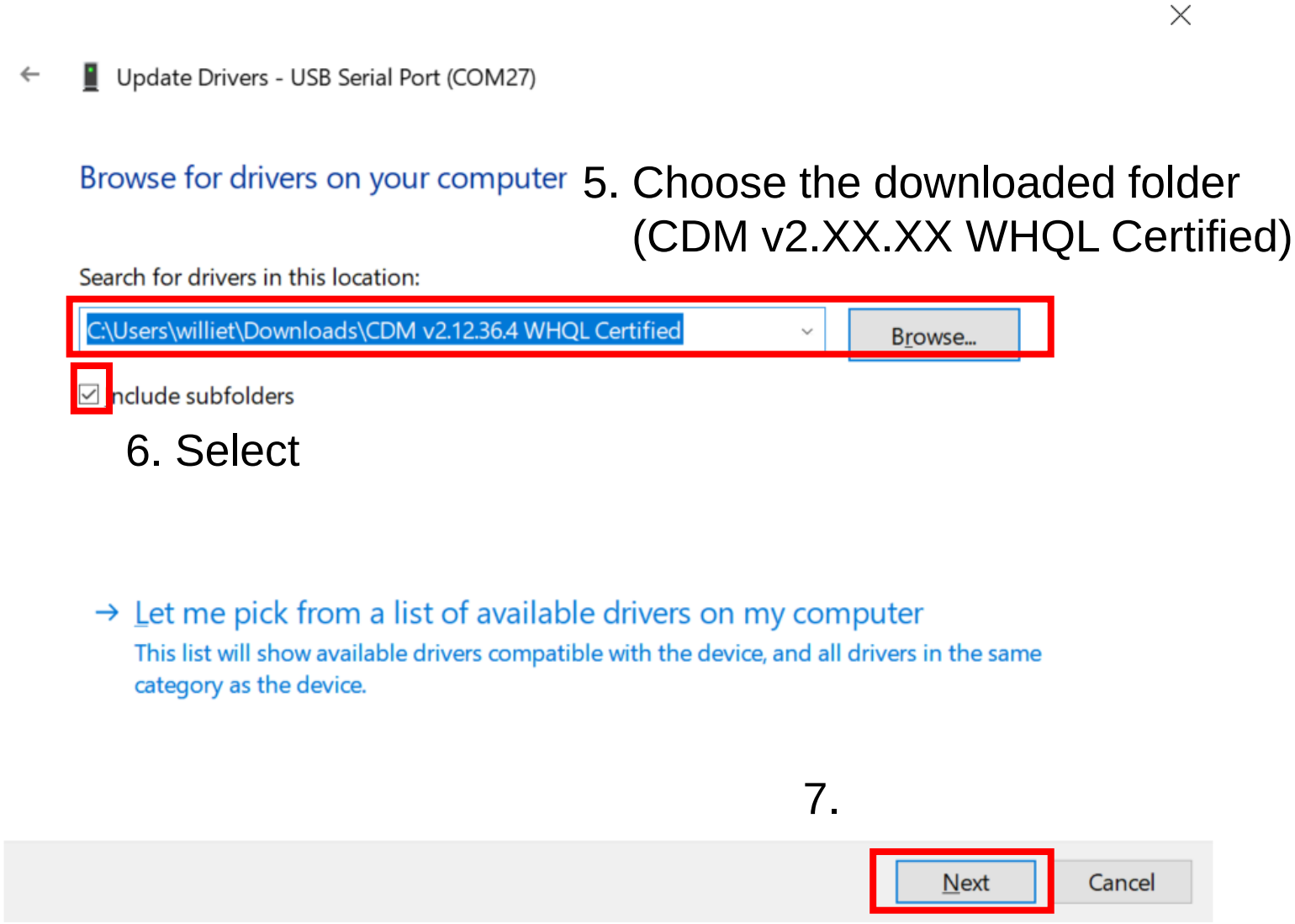
		Processor Architecture				
Operating System	**Release Date**	**X86 (32-Bit)**	**X64 (64-Bit)**	**PPC**	**ARM**	**MIPSII**
Windows*	2017–08-30	2.12.28	2.12.28	–	–	–
Linux	–	–	=	–	–	–

Troubleshooting – FTDI VCP Driver

3. Click Device Manager > Other devices > USB Serial Port > Update driver
4. Choose “Browse my computer for driver software”

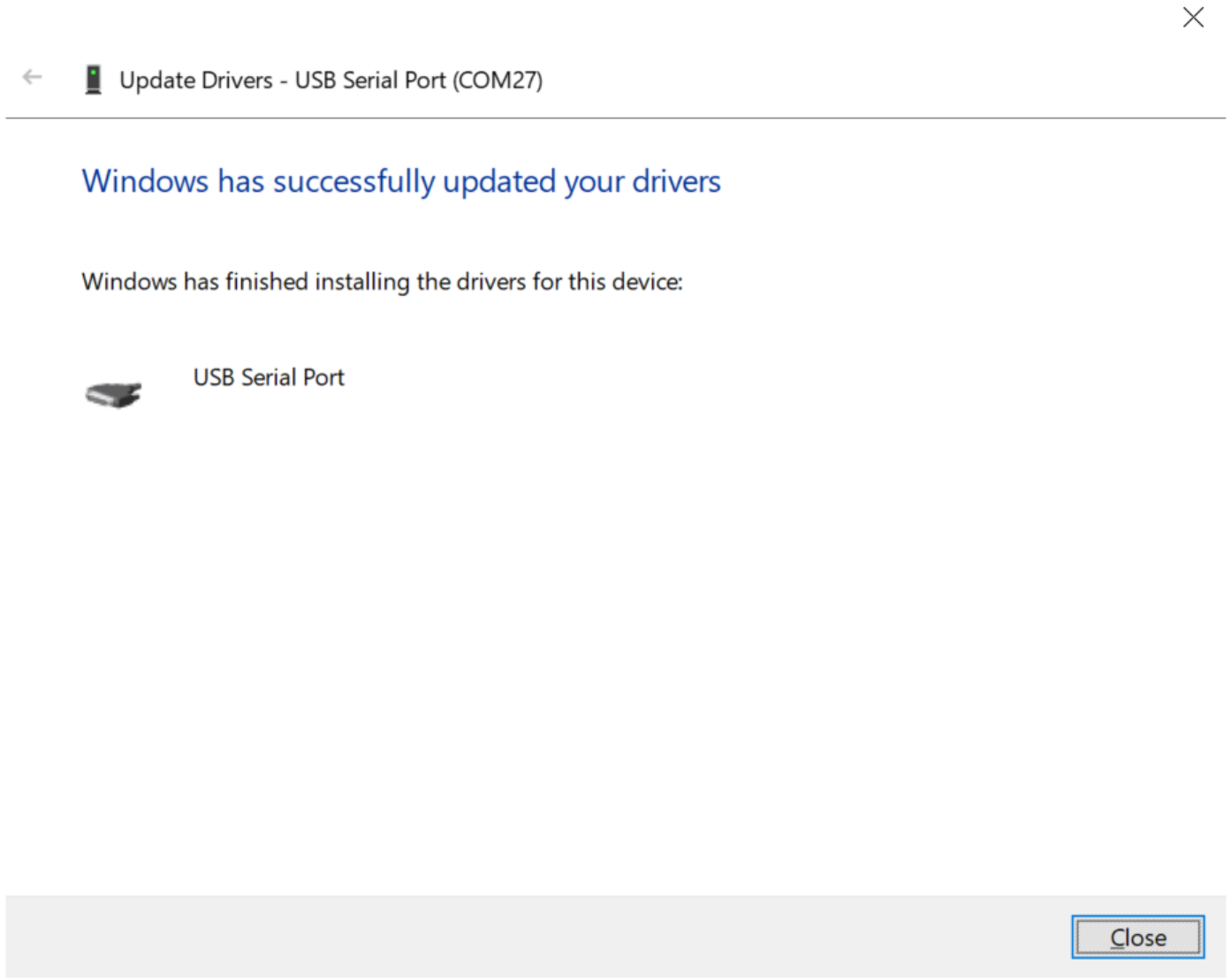


Troubleshooting – FTDI VCP Driver



Troubleshooting – FTDI VCP Driver

8. Finish

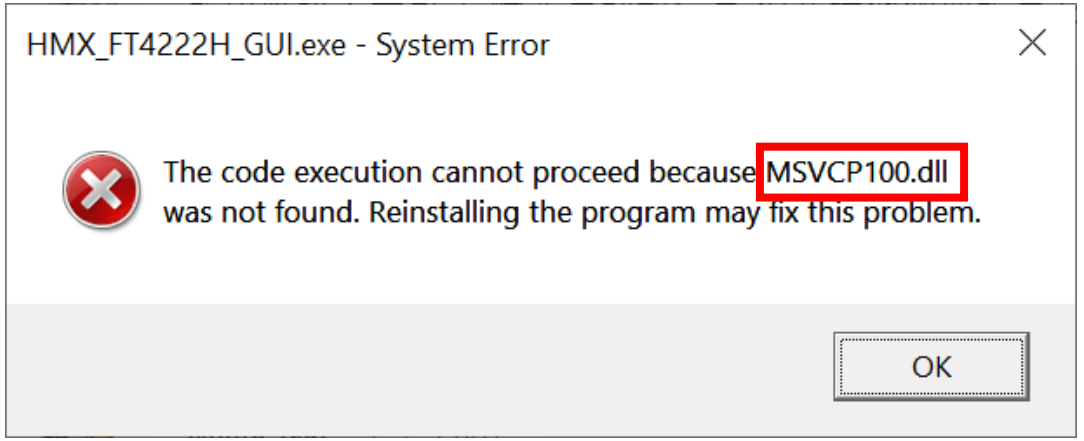


Appendix-4: Troubleshooting – HMX_FT4222H_GUI.exe DLL File Missing

Troubleshooting – DLL File Missing

If the DLL file missing is ftd2xx.dll

- Download FTDI D2XX Drivers install it.
  https://ftdichip.com/drivers/d2xx-drivers/



Currently Supported D2XX Drivers:

Subscribe to Our Driver Updates

		Processor Architecture					
Operating System	**Release Date**	**X86 (32-Bit)**	**X64 (64-Bit)**	**ARM**	**MIPS**	**SH4**	**Comments**
Windows (Desktop)*	2021-07-15	2.12.36.4	2.12.36.4	2.12.36.4A*****	–	–	WHQL Certified. Includes VCP and D2XX. Available as a setup executable Please see the Release Notes and Installation Guides.
Windows (Universal)****	2021-11-12	2.12.36.4U	2.12.36.4U	–	–	–	WHQL Certified. Includes VCP and D2XX.

Troubleshooting – DLL File Missing

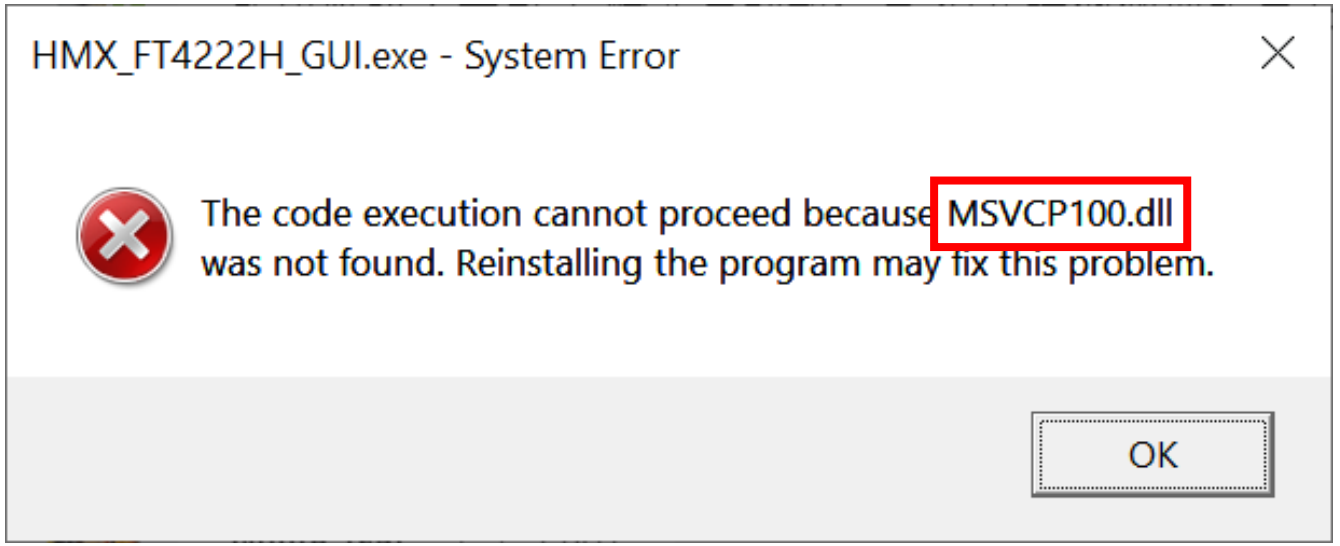
If the DLL file missing is MSVCP100.dll or MSCR100.dll

- Download Microsoft Visual C++ 2010 and install it.
  https://www.microsoft.com/en-us/download/details.aspx?id=26999

If the DLL file missing is MFC140.dll

- Download Microsoft Visual C++ 2015 and install it.
  https://www.microsoft.com/en-us/download/details.aspx?id=48145

If missing other DLL files, please search on the internet



Troubleshooting – DLL File Missing

If the DLL file missing is ftd2xx.dll

- Download FTDI D2XX Drivers install it.
  https://ftdichip.com/drivers/d2xx-drivers/

Currently Supported D2XX Drivers:

Subscribe to Our Driver Updates

		Processor Architecture					
Operating System	**Release Date**	**X86 (32-Bit)**	**X64 (64-Bit)**	**ARM**	**MIPS**	**SH4**	**Comments**
Windows (Desktop)*	2021-07-15	2.12.36.4	2.12.36.4	2.12.36.4A*****	–	–	WHQL Certified. Includes VCP and D2XX. Available as a setup executable Please see the Release Notes and Installation Guides.
Windows (Universal)****	2021-11-12	2.12.36.4U	2.12.36.4U	–	–	–	WHQL Certified. Includes VCP and D2XX.