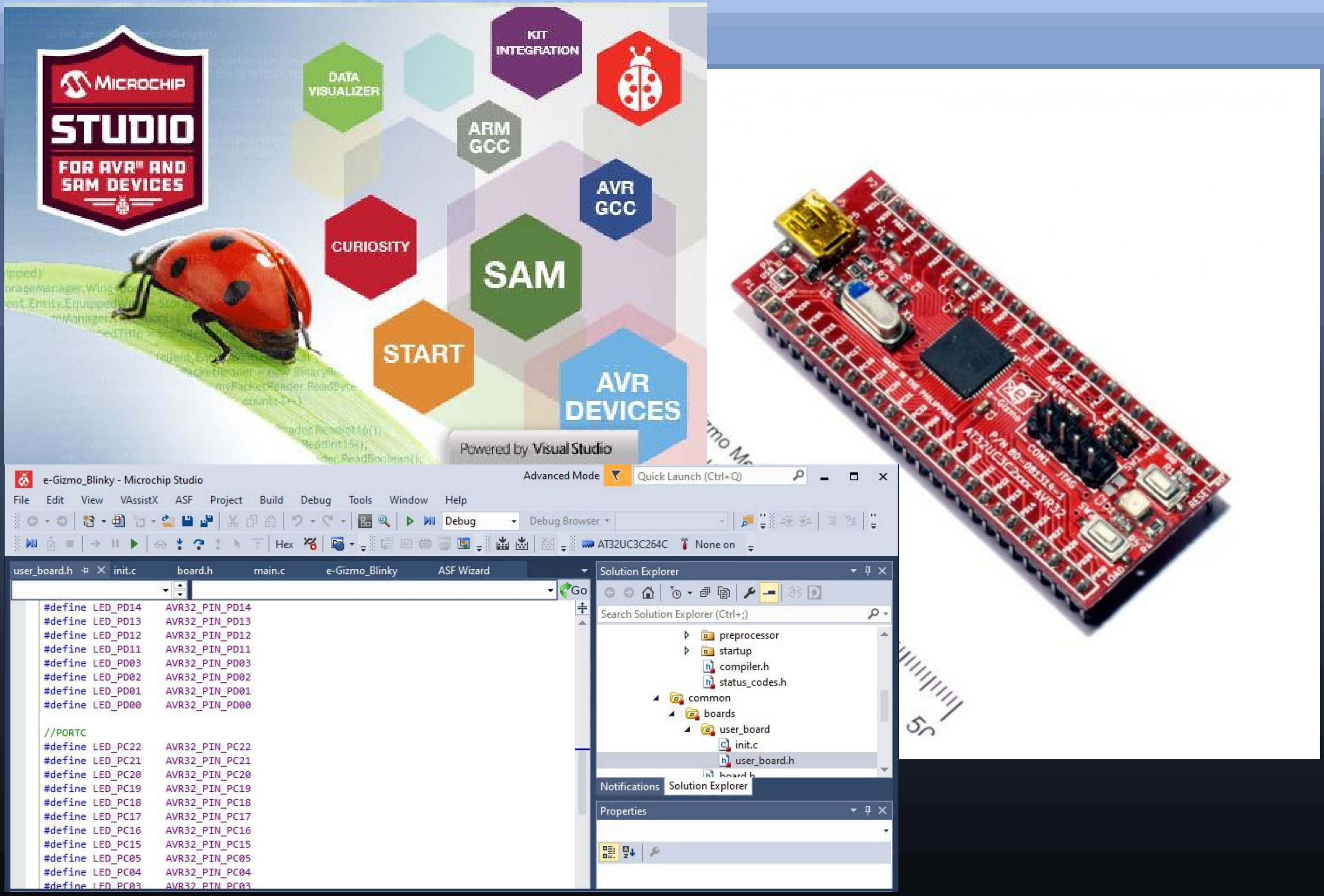
e-Gizmo
AVR 32-bit UC3C2 MCU board



```
#define LED_PD14    AVR32_PIN_PD14
#define LED_PD13    AVR32_PIN_PD13
#define LED_PD12    AVR32_PIN_PD12
#define LED_PD11    AVR32_PIN_PD11
#define LED_PD03    AVR32_PIN_PD03
#define LED_PD02    AVR32_PIN_PD02
#define LED_PD01    AVR32_PIN_PD01
#define LED_PD00    AVR32_PIN_PD00

//PORTC
#define LED_PC22    AVR32_PIN_PC22
#define LED_PC21    AVR32_PIN_PC21
#define LED_PC20    AVR32_PIN_PC20
#define LED_PC19    AVR32_PIN_PC19
#define LED_PC18    AVR32_PIN_PC18
#define LED_PC17    AVR32_PIN_PC17
#define LED_PC16    AVR32_PIN_PC16
#define LED_PC15    AVR32_PIN_PC15
#define LED_PC05    AVR32_PIN_PC05
#define LED_PC04    AVR32_PIN_PC04
#define LED_PC03    AVR32_PIN_PC03
```

Description

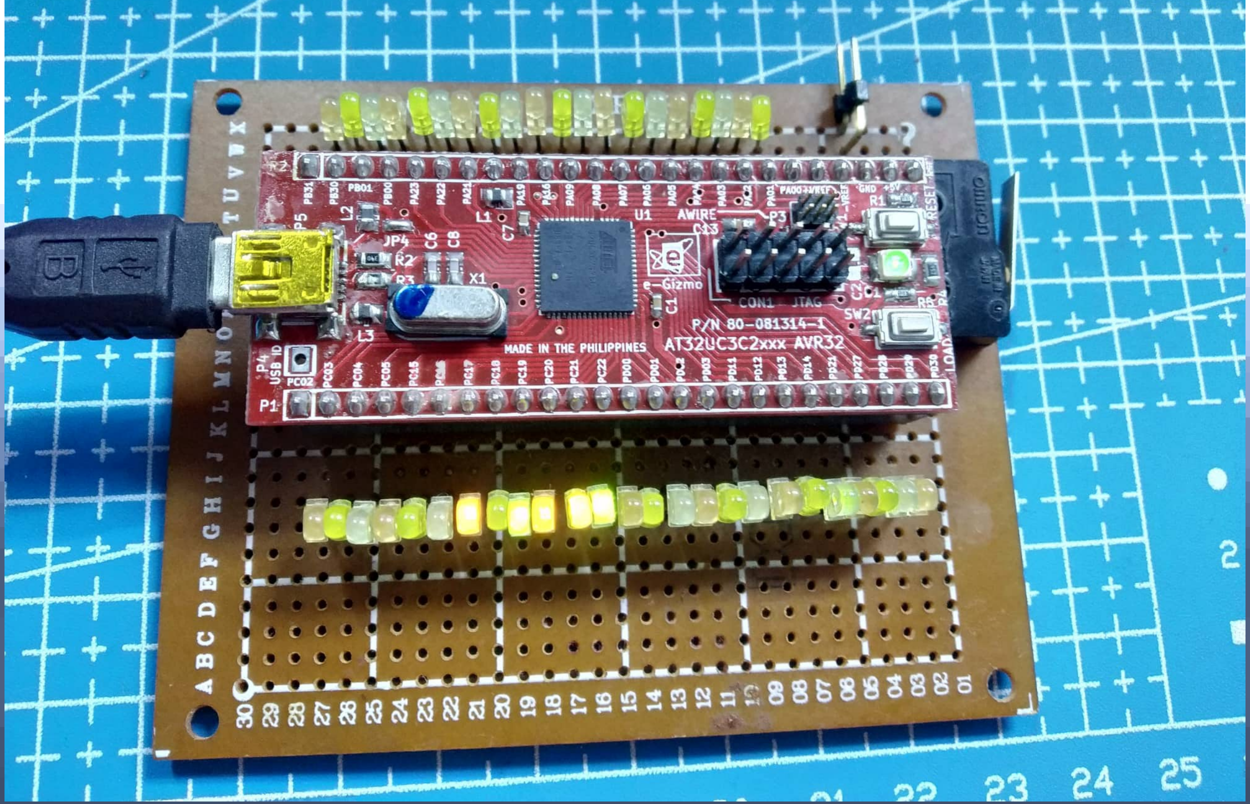
- The AT32UC3C is a complete System-On-Chip microcontroller based on the AVR32UC RISC processor running at frequencies up to 66MHz.
- Is a high-performance 32-bit RISC microprocessor core, designed for cost-sensitive embedded applications, with particular emphasis on low power consumption, high code density and high performance.

F.Y.I

This family of 32-bit MCUs from Atmel as the name implies it is a descendant of the AVR MCUs, which are both come from Atmel and you can use the same Atmel Studio IDE with free C/C++ compiler.

- Arduino chose the ARM cortex M0 based SAMD21 device instead of the AVR32 for their next generation 32-bit Arduino Zero board.
- Nevertheless there are advantages of the AVR32, unlike the SAMD21, the AVR32 comes preprogrammed with a bootloader. So yo can load your program without a programmer. AVR32 comes with USB host and device interface and an Ethernet MACB interface. So it is closer than the Arduino Zero to implementing IOT projects.

According to motion55 @elab.ph forum “The AVR32 Tutorials”



- It is basically an AT32UC3C264C or AT32UC3C2128C mounted on a small board with connectors to access all pins (breakout), a JTAG connector for debugging and programming and a mini USB (OTG) connector. There is also a 12MHz crystal needed for USB clocking.
- The I/O connectors are standard 0.1” pitch connector so you can in turn mount the mini board over a prototyping board for your projects.

Schematic Diagram

- https://e-gizmo.net/oc/kits%20documents/AVR32/New%20AVR32/The%20%20AVR32/schematic.gif

Software Downloads

- **Microchip Studio** for AVR and SAM devices
- https://www.microchip.com/en-us/development-tools-tools-and-software/microchip-studio-for-avr-and-sam-devices
- Microchip official website
- https://www.microchip.com/

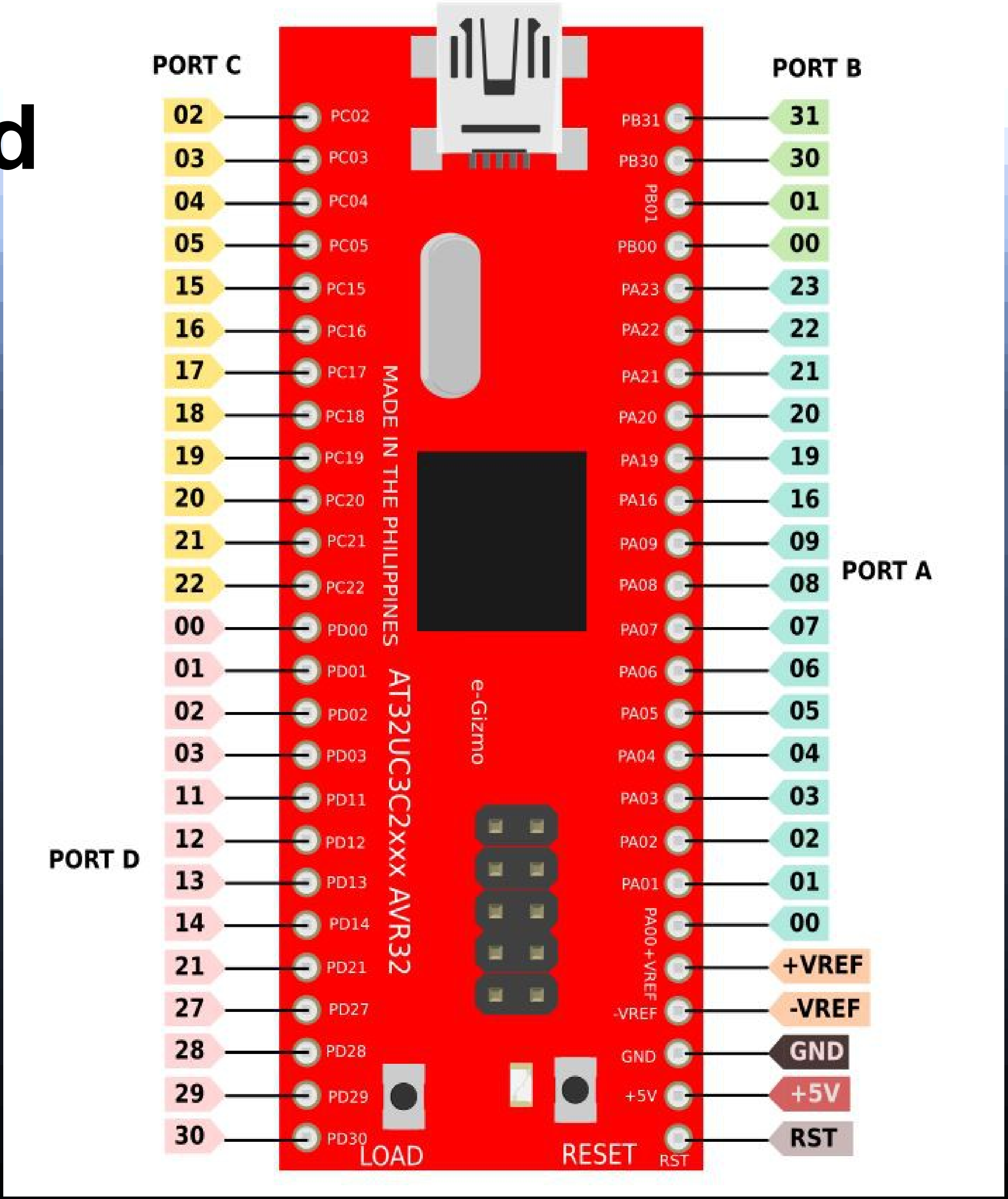
FLIP 3.4.7

- FLIP installer (to use the bootloader)
- https://e-gizmo.net/oc/kits %20documents/AVR32/New%20AVR32/The %20%20AVR32/

AVR32 Breakoutboard

- Variants available
- AT32UC3C264
- AT32UC3C2128C
- AT32UC3C2512C

QFN 64Pins



Specifications

MCU boards	Program Memory	SRAM	I/O pins	Features
AT32UC3C264	64KB	16KB	45	TWI,USART,SPI,I2C,2MSPS,ADC,DAC,ETHERNET,USB (device + OTG)
AT32UC3C2128C	128KB	32KB	45	
AT32UC3C2512C	512KB	64KB	45	

- Datasheet
- http://ww1.microchip.com/downloads/en/DeviceDoc/doc32117.pdf

GPIO Controller Functions Multiplexing

AVR32_PIN_PAxx GPIO function

QFN	PIN	A	B	C	D	E	F
1	**PA00**		CANIF-TZXLINE[1]				
2	**PA01**		CANIF-RXLINE[1]	PEVC-PAD_EVT[0]			
3	**PA02**	SCIF-GCLK[0]		PEV-PAD_EVT[1]			
4	**PA03**	SCIF-GLCK[1]	EIC-EXTINT[1]				
7	**PA04**	**ADCIN0**	USBC-ID	ACIFA0-ACAOUT			
8	**PA05**	**ADCIN1**	USBC-VBOF	ACIFA0-ACBOUT			
9	**PA06**	**ADCIN2**	AC1AP1	PEVC-PAD_EVT[2]			
10	**PA07**	**ADCIN3**	AC1AN1	PEVC-PAD_EVT[3]			
11	**PA08**	**ADCIN4**	AC1BP1	EIC-EXTIN[2]			
12	**PA09**	**ADCIN5**	AC1BN1				
13	**PA16**	**ADCREF0**		DACREF			
14	**ADC REFP**						
15	**ADC REFN**						

* see the GPIO Function summary

GPIO Function

AVR32_PIN_PAxx/PBxx GPIO function

QFN	PIN	A	B	C	D	E	F
16	**PA19**	**ADCIN8**	EIC-EXTINT[1]				
19	**PA20**	**ADCIN9**	AC0AP0	AC0AP0 or DAC0A			
20	**PA21**	**ADCIN10**	AC0BN0	AC0BN0 or DAC0B			
21	**PA22**	**ADCIN11**	AC0AN0	PEVC-PAD_EVT[4]		MACB-SPEED	
22	**PA23**	**ADCIN12**				MACB-WOL	
62	**PB00**	**USART0-CLK**	CANIF-RXLINE[1]	EIC-EXTINT[8]	PEVC-PAD_EVT[10]		
63	**PB01**		CANIF-TXLINE[1]		PEVC-PAD_EVT[11]		
31	**PB30**						
32	**PB31**						

* see the GPIO Function summary

GPIO Function

AVR32_PIN_PCxx

QFN	PIN	A	B	C	D	E	F
33	**PC02**	**TWIMS0-TWD**	SPI0-NPCS[3]	USART2-RXD	TC1-CLK1	MACB-MDC	
34	**PC03**	**TWIMS0-TWCK**	EIC-EXTINT[1]	USART2-TXD	TC1-B1	MACB-MDIO	
37	**PC04**	**TWIMS1-TWD**	EIC-EXTINT[3]	USART2-TXD	TC0-B1		
38	**PC05**	**TWIMS1-TWCK**	EIC-EXTINT[4]	USART2-RXD	TC0-A2		
39	**PC15**	**PWM-PWMH[1]**	SPI0-NPCS[0]	EBI-SDWE	USART0-RXD	CANIF-RXLINE[1]	
40	**PC16**	**PWM-PWML[1]**	SPI0-NPCS[1]	EBI-CAS	USART0-TXD	CANIF-TXLINE[1]	
41	**PC17**	**PWM-PWMH[0]**	SPI0-NPCS[2]	EBI-RAS	IISC-ISDO		USART3-TXD
42	**PC18**	**PWM-PWML[0]**	EIC-EXTINT[5]	EBI-SDA10	IISC-ISDI		USART3-RXD
43	**PC19**	**PWM-PWML[2]**	SCIF-GCLK[0]	EBI-DATA[0]	IISC-IMCK		USART3-CTS
44	**PC20**	**PWM-PWMH[2]**	SCIF-GCLK[1]	EBI-DATA[1]	IISC-ISCK		USART3-RTS
45	**PC21**	**PWM-EXT_FAULTS[0]**	CANIF-RXLINE[0]	EBI-DATA[2]	IISC-IWS		
46	**PC22**	**PWM-EXT_FAULTS[1]**	CANIF-TXLINE[0]	EBI-DATA[3]		USART3-CLK	

* see the GPIO Function summary

AVR32_PIN_PDxx

GPIO Function

QFN	PIN	A	B	C	D	E	F
47	**PD00**	**SPI0-MOSI**	TC1-CLK0	EBI-DATA[13]	QDEC0-QEPI	USART0-TXD	
48	**PD01**	**SPI0-MISO**	TC1-A0	EBI-DATA[14]	TC0-CLK1	USART0-RXD	
49	**PD02**	**SPI0-SCK**	TC0-CLK2	EBI-DATA[15]	QDEC0-QEPA		
50	**PD03**	**SPI0-NPCS[0]**	TC0-B2	EBI-ADDR[0]	QDEC0-QEPB		
53	**PD11**	**USART1-TXD**	USBC-ID	EBI-ADDR[8]	PEVC-PAD_EVT[6]	MACB-TXD[0]	
54	**PD12**	**USART1-RXD**	USBC-VBOF	EBI-ADDR[9]	PEVC-PAD_EVT[7]	MACB-TXD[1]	
55	**PD13**	**USART1-CTS**	USART1-CLK	EBI-SDCK	PEVC-PAD_EVT[8]	MACB-RXD[0]	
56	**PD14**	**USART1-RTS**	EIC-EXTINT[7]	EBI-ADDR[10]	PEVC-PAD_EVT[9]	MACB-RXD[1]	
57	**PD21**	**USART3-TXD**	EIC-EXTINT[0]	EBI-ADDR[17]	EBI-ADDR[17]	QDEC1-QEPI	
58	**PD27**	**USART0-TXD**	CANIF-RXLINE[0]	EBI-NCS[1]	TC0-A0	MACB-RX_ER	
59	**PD28**	**USART0-RXD**	CANIF-TXLINE[0]	EBI-NCS[2]	TC0-B0	MACB-TX_CLK	
60	**PD29**	**USART0-CTS**	EIC-EXTINT[6]	USART0-CLK	TC0-CLK0	MACB-TX_CLK	
61	**PD30**	**USART0-RTS**	EIC-EXTINT[3]	EBI-NWAIT	TC0-A1	MACB-TX_EN	

* see the GPIO Function summary

*GPIO Function in summary

PORT A #	PIN
PA00 - 05	
PA06	ADC4
PA07	ADC5
PA08	ADC6
PA09	ADC7
PA16	
PA19	ADC0
PA20	ADC2
PA21	ADC3
PA22	ADC1
PA23	GPIO

PORT B #	PIN
PB00	
PB01	
PB30	
PB31	

PORT C #	PIN
PC02	TWI SDA
PC03	TWI SCL
PC04	TWI SDA
PC05	TWI SCL
PC15	PWM0
PC16	PWM1
PC17	UART TX
PC18	UART RX
PC19	
PC20	
PC21	CAN-RX
PC22	CAN-TX

PORT D #	PIN
PD00	SPI MOSI
PD01	SPI MISO
PD02	SPI SCK
PD03	SPI CS4
PD11	UART TX
PD12	UART RX
PD13	GPIO
PD14	GPIO
PD21	GPIO
PD27	SPI MOSI
PD28	SPI MISO
PD29	SPI SCK
PD30	SPI CS1

THE CODES

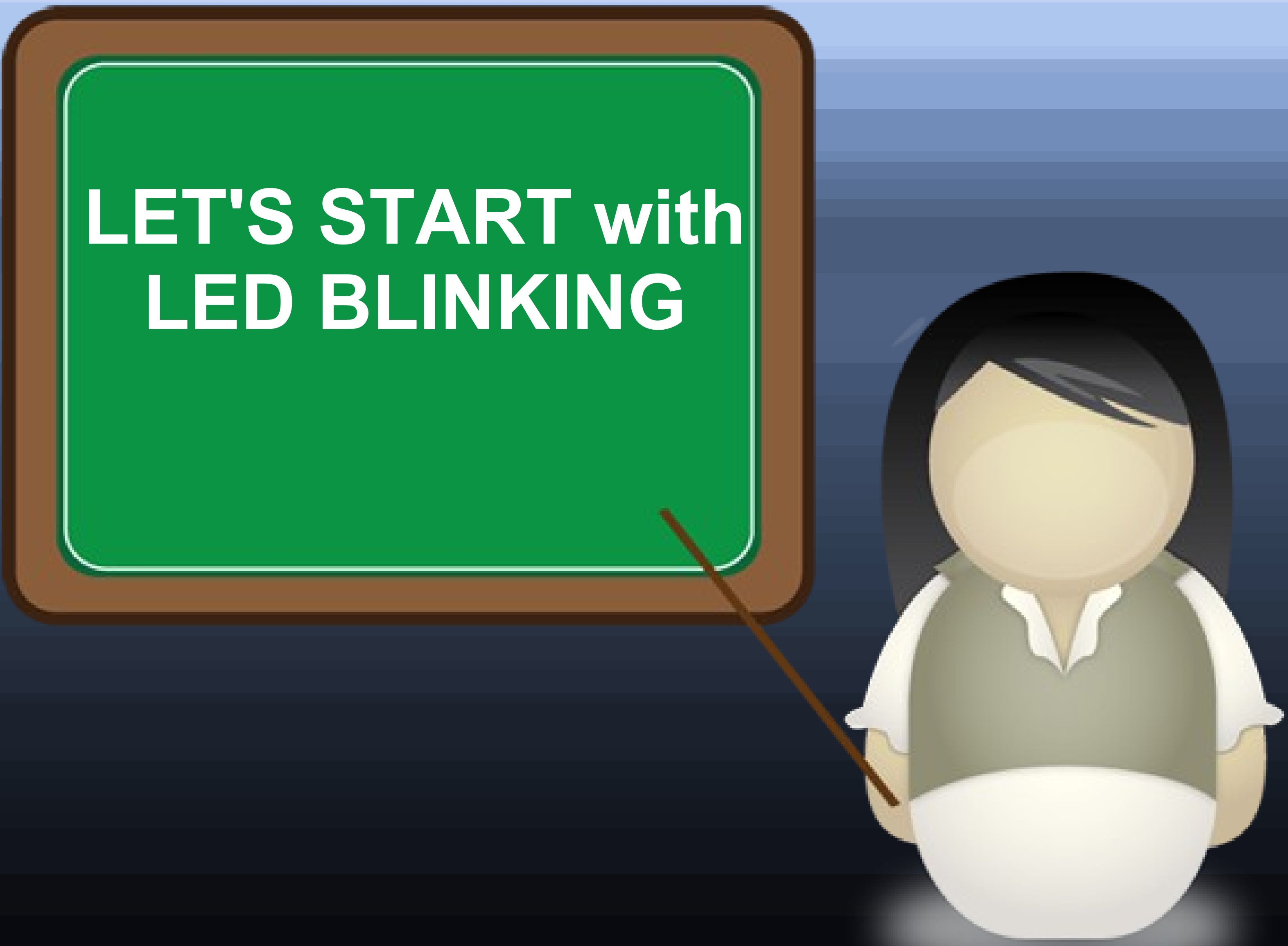
Pin assignment

- **AVR32_PIN_P*A00 – 09,16,19,20 – 23***
- **AVR32_PIN_P*B00 – 01,30 - 31***
- **AVR32_PIN_P*C02 – 05,15 – 22***
- **AVR32_PIN_P*D00 – 03,11 – 14,21,27 - 30***

- These are the GPIO mapping

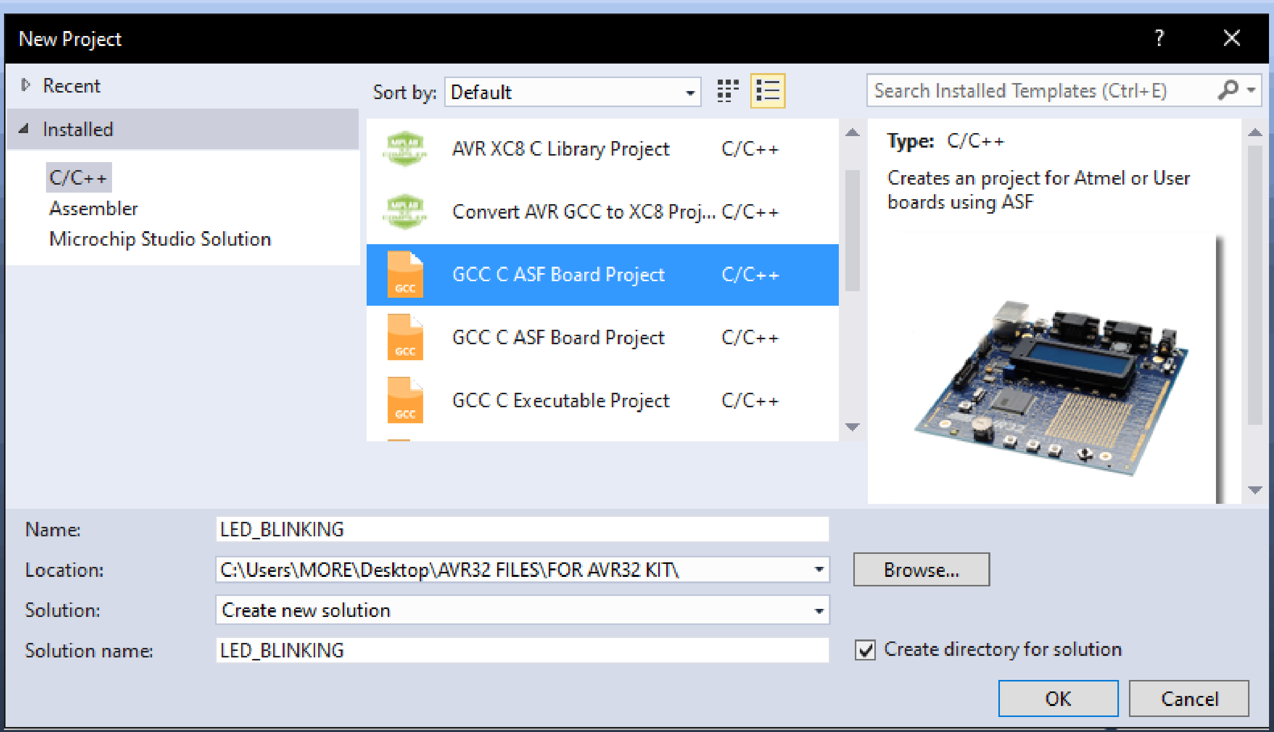
MCU frequency

- #define BOARD_OSC0_HZ 12000000
- #define BOARD_OSC0_STARTUP_US 50000
- #define BOARD_OSC0_IS_XTAL true



LET'S START with LED BLINKING

Creating a new AVR32 project


New Project
?
Recent
Installed
C/C++
Assembler
Microchip Studio Solution
Search Installed Templates (Ctrl+E)
Browse...
OK
Cancel

Sort by: Default

AVR XC8 C Library Project	C/C++
Convert AVR GCC to XC8 Proj...	C/C++
GCC C ASF Board Project	C/C++
GCC C ASF Board Project	C/C++
GCC C Executable Project	C/C++

Type: C/C++

Creates an project for Atmel or User boards using ASF

Name: LED_BLINKING

Location: C:\Users\MORE\Desktop\AVR32 FILES\FOR AVR32 KIT\

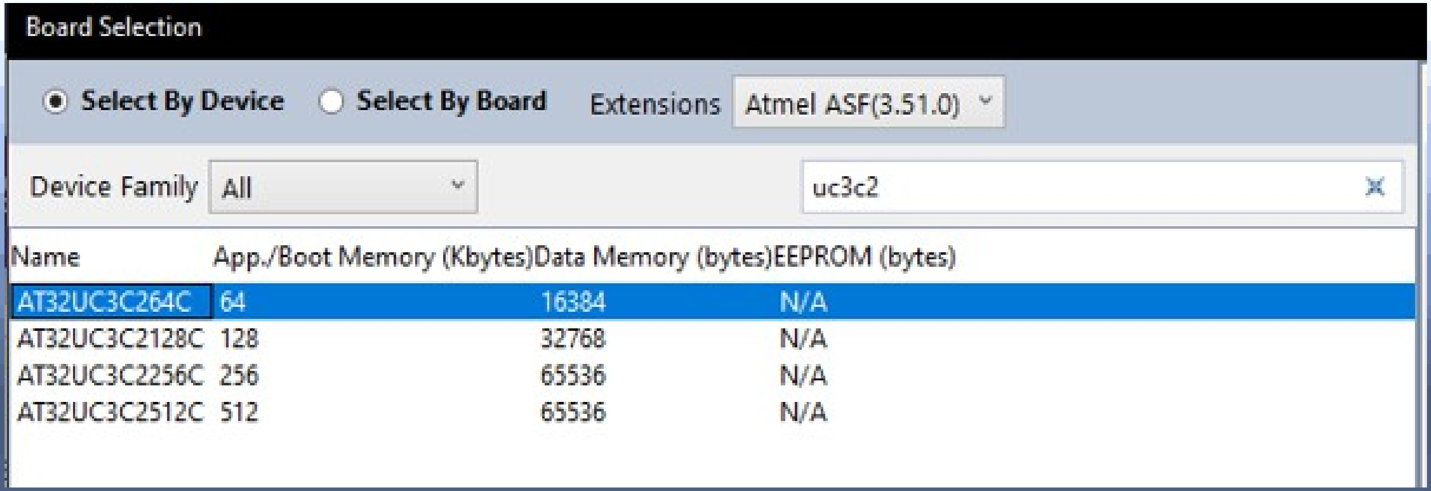
Solution: Create new solution

Solution name: LED_BLINKING

☑ Create directory for solution

- Step 1. Launch Microchip* Studio and on the menu select **File > New > Project.** The new project dialog will appear. On the dialog, **select "GCC C ASF Board Project"**. Fill in the name **"LED_BLINKING"** for the project and press OK.

*from ATMEL to Microchip brand name


Board Selection

● Select By Device ○ Select By Board Extensions: Atmel ASF(3.51.0)

Device Family: All

uc3c2

Name	App./Boot Memory (Kbytes)	Data Memory (bytes)	EEPROM (bytes)
AT32UC3C264C	64	16384	N/A
AT32UC3C2128C	128	32768	N/A
AT32UC3C2256C	256	65536	N/A
AT32UC3C2512C	512	65536	N/A

- Step 2. After Choosing “GCC C ASF Board Project”, select the device. For the board sample I have, I select the **AT32UC3C264C**.
- The Microchip Studio will then create the project LED_BLINKING in the desired location. So far this is standard procedure when using the Microchip Studio.
- The next step will be to run ASF wizard to add the modules you will be using on the project.

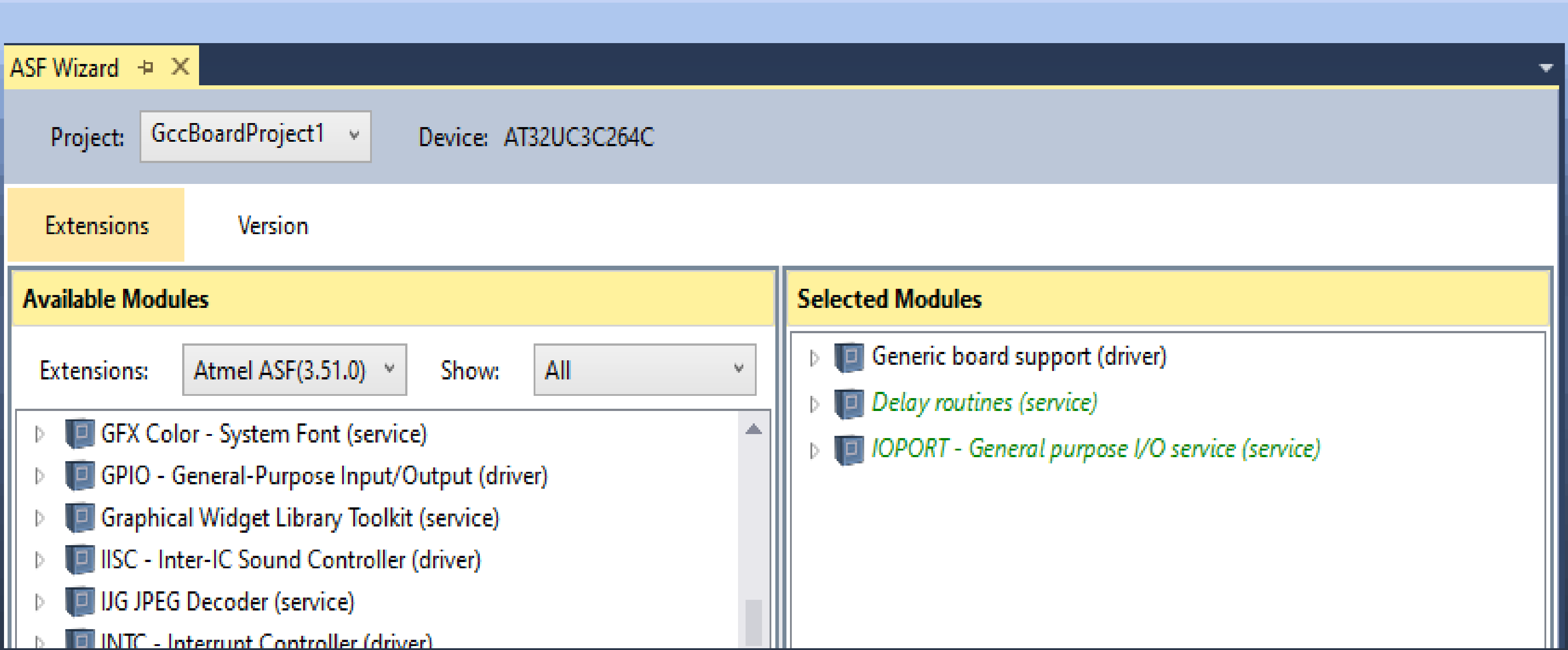
Using the ASF Wizard

- After creating the project, take time to examine the contents.
- On the right side, is the "**Solution Explorer**". Although there's nothing to do on this project in terms of execution, there's an organization and file structure as shown. The center, you can see the contents of the code. In this case it's the **main.c** file. Except for the call to **board_init()**, it is blank at this point. If you build (compile) this project, **it will compile without errors.**
- Remember that the Microsoft company who created this IDE. So if you want to add or cut the files. You can cut and paste from Windows Explorer to the Solution Explorer will work.
- Another way is the ASF Wizard will allow you to automatically add code developed by Atmel that can utilize the features of the MCU. It is True,that you can also write your own code to access the registers directly to operate the peripherals.

Use ASF wizard

- Let's now use the ASF (Atmel Software Framework) Wizard. In menu select **ASF → ASF Wizard.**
- On the left box, you can select the modules that you will need. There's no case if you select modules that you don't need. It can occupy space on the MCU but if it is fit don't you worry.
- Besides the linker might remove unused modules. Let's add the following with BLINKING.
- ***1. Delay routes (service)***
- ***2. IOPORT – General purpose I/O (service)***
- Note: If you compile the code at this point there wil be an error occurred and clearly we need to organize it.

ASF Wizard view



Project: GccBoardProject1 Device: AT32UC3C264C

Available Modules

Extensions: Atmel ASF(3.51.0) Show: All

- GFX Color - System Font (service)
- GPIO - General-Purpose Input/Output (driver)
- Graphical Widget Library Toolkit (service)
- IISC - Inter-IC Sound Controller (driver)
- IJG JPEG Decoder (service)
- INTC - Interrupt Controller (driver)

Selected Modules

- Generic board support (driver)
- *Delay routines (service)*
- *IOPORT - General purpose I/O service (service)*

- Select and Click **Add>>** to select modules.
- After you added the module click **Apply.**

Completing the Blinky Project

- 1. Modify the **user_board.h***file to include the following lines.

```
#define BOARD_OSC0_HZ 12000000
#define BOARD_OSC0_STARTUP_US  50000
#define BOARD_OSC0_IS_XTAL true


#define BLINK_LED   AVR32_PIN_PD30
```

*use the search solution explorer to go to file.

- 2. Modify the board_init() function in the **init.c** file in the user_board folder as follows:

```
void board_init (void)

{

ioport_init(); // This must be called before any other ioport function

ioport_set_pin_dir (BLINK_LED, IOPORT_DIR_OUTPUT); //make pin an output

}
```

- The code initialize the I/O port. The LED is connected to the PD30 pin.

- 3. Modify the **main.c** file as follows:

```
#include <asf.h>
int main (void)
{
        sysclk_init();
        board_init();

        while (true)
        {
        ioport_toggle_pin_level (BLINK_LED);
        delay_ms(500);
        }
}
```

- An LED in series with a 470 ohms resistor connected to the PD30 and to GND on the other end.

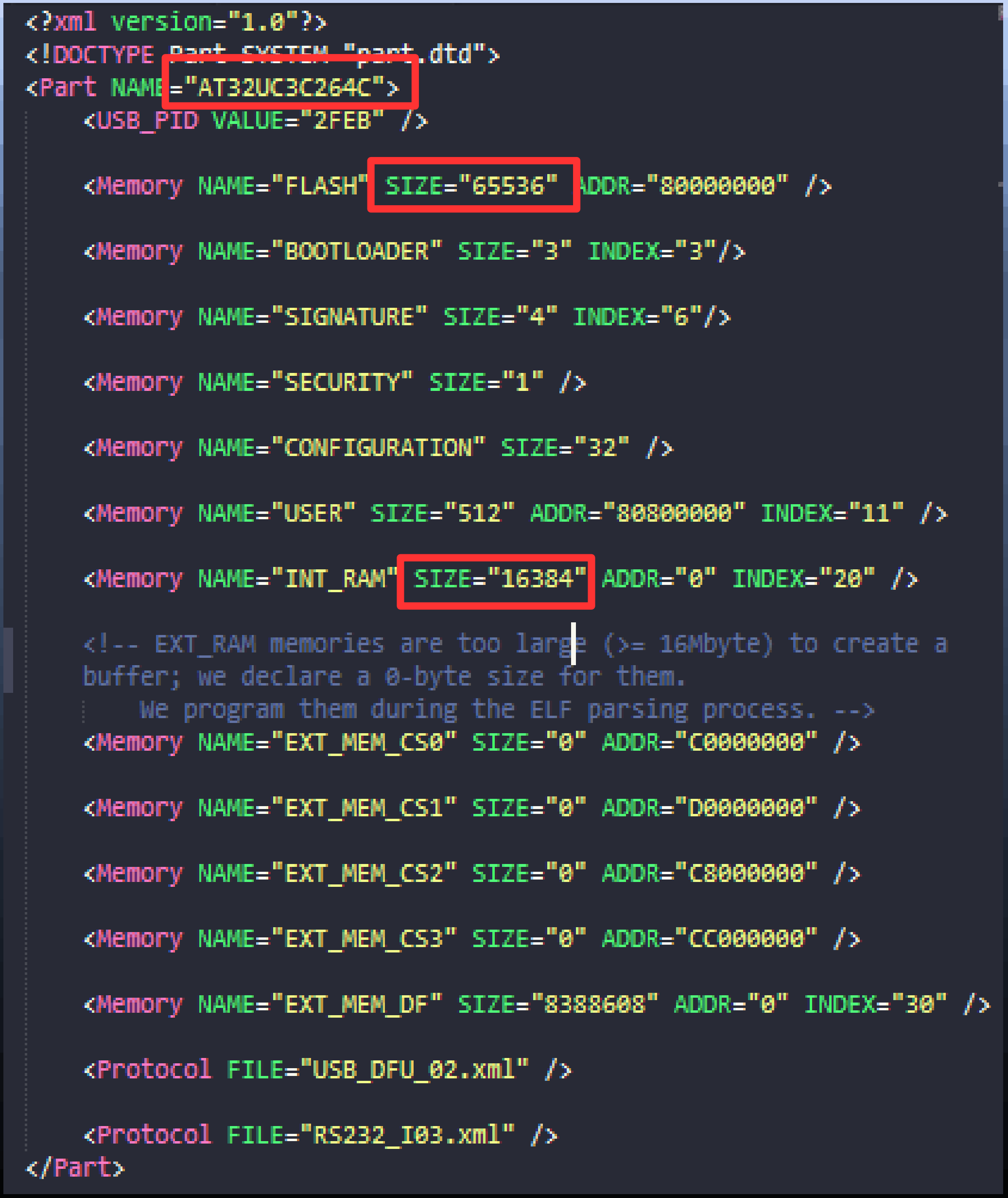
Using the Bootloader on the Mini Board

- Some of us we don't have any JTAGICE debug tool for program flashing to AVR32 Mini board. Luckily, all AVR32 has preprogrammed bootloader. Immediately after reset, the bootloader checks an I/O pin. Usually the I/O pin is connected to 1 switch, push button or jumper pin. If the button is pressed, then the bootloader will enter to load the program from the USB port.
- We need a program from PC to download the program to the AVR32. This is called the **batchsip.exe** program. It is part of the FLIP v3.4.7 programmer. It can be downloaded from this link.
- https://www.microchip.com/en-us/development-tool/flip

Create a new device description file

- Unfortunately, the last version 3.4.7 of FLIP has **no AT32UC3C264C** on the device that will be recognize. We need to create a device description file for AT32UC3C264C.
- Instruction to add AT32UC3C264C part description file.
- 1. From the <***Flip install path>\bin\PartDescriptionFiles folder,*** **copy** the file **AT32UC3C2128C.xml** file to your desktop. **Rename** the file to **AT32UC3C264C.xml.**
- 2. Using a text editor, edit the file. Change the Part Name from “AT32UC3C2128C” to “AT32UC3C264C”.
- 3. Change FLASH size from “131072” to “65536”.
- 4. Change INT_RAM size form “32768” to “16384”.
- 5. After saving, copy the file back to the <Flip install path>\bin\PartDescriptionFiles folder.

AT32UC3C264C.xml view



```
<?xml version="1.0"?>
<!DOCTYPE Part SYSTEM "part.dtd">
<Part NAME="AT32UC3C264C">
    <USB_PID VALUE="2FEB" />

    <Memory NAME="FLASH" SIZE="65536" ADDR="80000000" />

    <Memory NAME="BOOTLOADER" SIZE="3" INDEX="3"/>

    <Memory NAME="SIGNATURE" SIZE="4" INDEX="6"/>

    <Memory NAME="SECURITY" SIZE="1" />

    <Memory NAME="CONFIGURATION" SIZE="32" />

    <Memory NAME="USER" SIZE="512" ADDR="80800000" INDEX="11" />

    <Memory NAME="INT_RAM" SIZE="16384" ADDR="0" INDEX="20" />

    <!-- EXT_RAM memories are too large (>= 16Mbyte) to create a
    buffer; we declare a 0-byte size for them.
        We program them during the ELF parsing process. -->
    <Memory NAME="EXT_MEM_CS0" SIZE="0" ADDR="C0000000" />

    <Memory NAME="EXT_MEM_CS1" SIZE="0" ADDR="D0000000" />

    <Memory NAME="EXT_MEM_CS2" SIZE="0" ADDR="C8000000" />

    <Memory NAME="EXT_MEM_CS3" SIZE="0" ADDR="CC000000" />

    <Memory NAME="EXT_MEM_DF" SIZE="8388608" ADDR="0" INDEX="30" />

    <Protocol FILE="USB_DFU_02.xml" />

    <Protocol FILE="RS232_I03.xml" />
</Part>
```

- Modified

File name reference .xml

Part Name	FLASH size (bytes in binary)	INT_RAM
AT32UC3C264C	65536	16384
AT32UC3C2128C	131072	32768
AT32UC3C2512C	524288	65536

Create External Tools menu

- ***Before using batchisp***, it will easier to add it to the external tools menu. From the menu, select Tools > External Tools. Press the Add button and edit the following boxes:

Title: *DFU AT32UC3C264C*

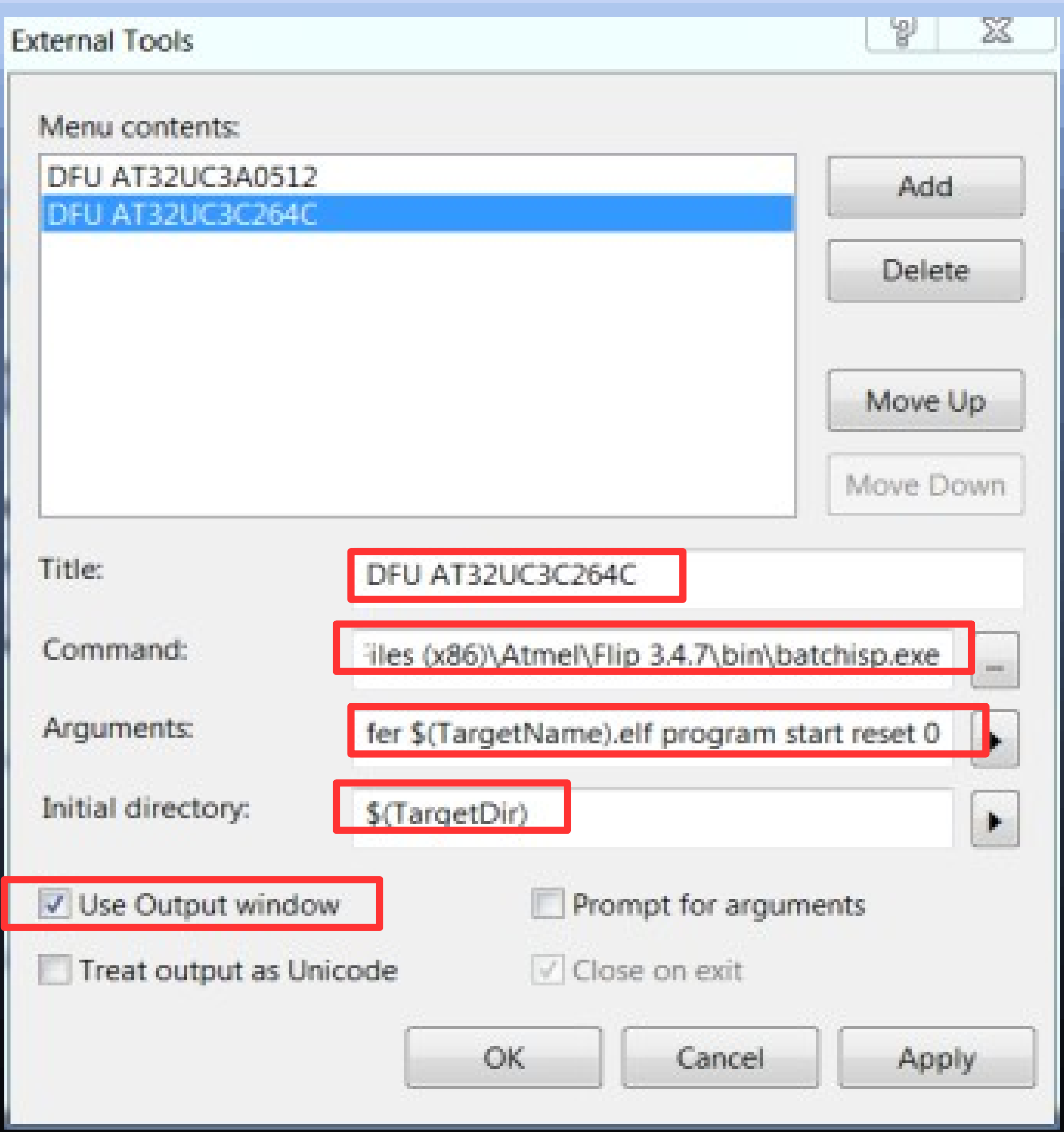
Command: *C:\Program Files (x86)\Atmel\Flip 3.4.7\bin\batchisp.exe*

Arguments: *-device AT32UC3C264C -hardware usb -operation onfail abort memory FLASH erase F loadbuffer*

$(TargetName).elf program start reset 0

Initial directory: *$(TargetDir)*

External tools View



External Tools

Menu contents:

- DFU AT32UC3A0512
- DFU AT32UC3C264C

Title: DFU AT32UC3C264C

Command: iles (x86)\Atmel\Flip 3.4.7\bin\batchisp.exe

Arguments: fer $(TargetName).elf program start reset 0

Initial directory: $(TargetDir)

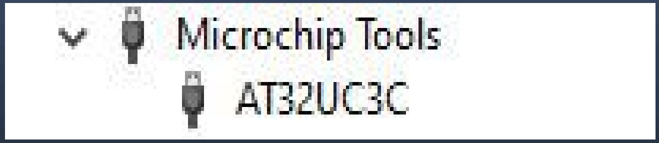
☑ Use Output window

□ Prompt for arguments

□ Treat output as Unicode

☑ Close on exit

We are now ready to download

- 1. Connect the mini board to the PC using the USB cable.
- 2. Press the **LOAD button** and momentarily press the **RESET button.** After you hear a beep on your PC, you may now release the load button.
- 3. Check if the device ***Microchip Tools > AT32UC3C*** appears in the device manager.
- 

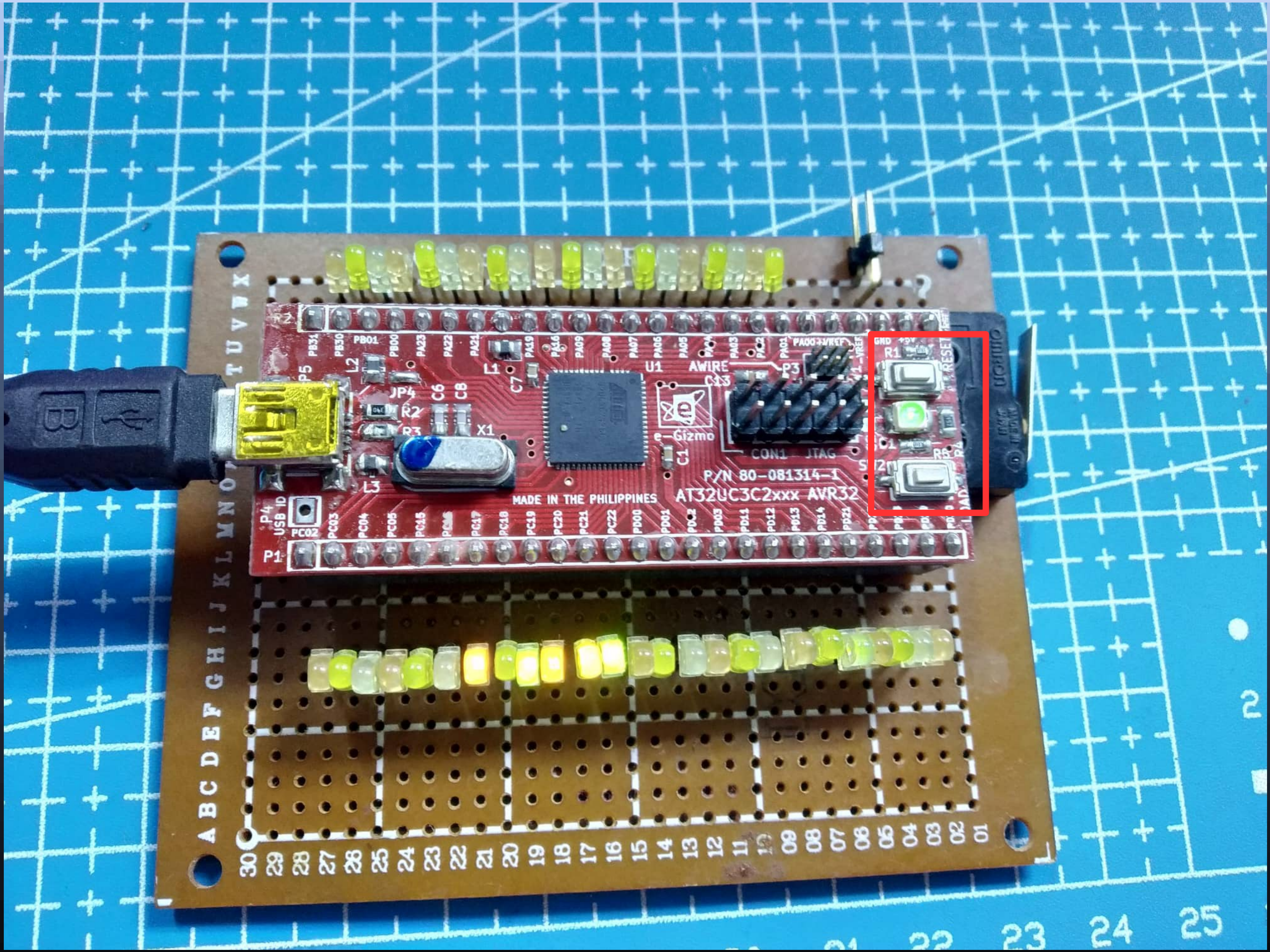
  Microchip Tools

  AT32UC3C
-
- 4. From the menu, select Tools> DFU AT32UC3C264C. This should initiate downloading to the mini board.

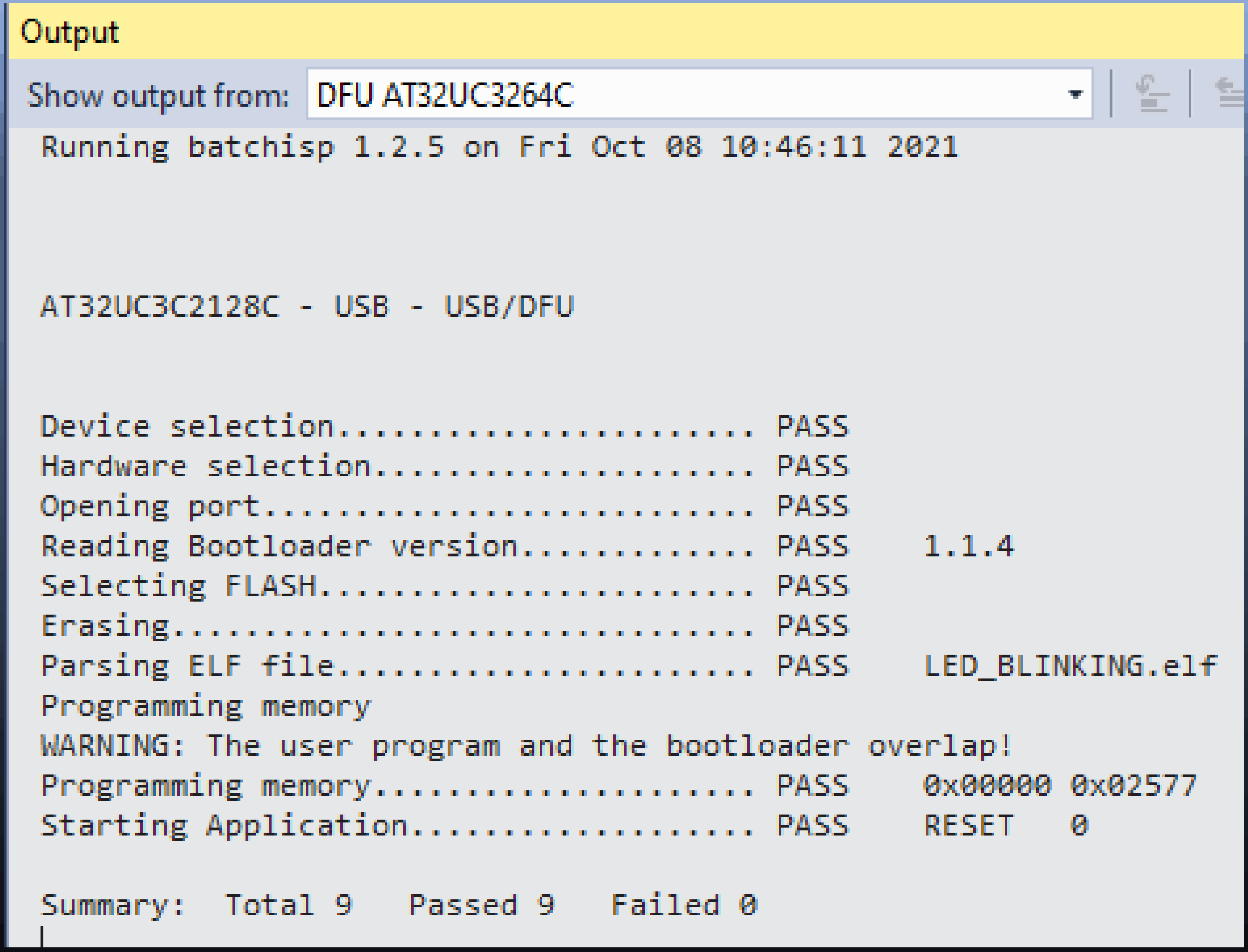
Installing DFU usb

- **If the program did not proceed.**
- **If DFU does not exist.**
- Go to Device manager > Microchip Tools> AT32UC3C (Right-Click update driver)
- Select “Browse my computer for drivers”
- Click “Let me pick from a list of available drivers on my computer”
- Click “Have a Disk...”
- Click “Browse...”
- Find the location >C:\ Program Files (x86)\Atmel\Flip 3.4.7\usb
- Select the “atmel_usb_dfu.inf” and click Open and OK.
- Click “ Next” and wait until the driver successfully installed.

Board view


AT32UC3C2xxx AVR32
P/N 80-081314-1
MADE IN THE PHILIPPINES
e-Gizmo
AWIRE
CON1
JTAG
RESET
USB ID
X1
U1
P1
P4
P5

Successfully uploaded



```
Running batchisp 1.2.5 on Fri Oct 08 10:46:11 2021



AT32UC3C2128C - USB - USB/DFU



Device selection....................... PASS
Hardware selection..................... PASS
Opening port........................... PASS
Reading Bootloader version............. PASS    1.1.4
Selecting FLASH........................ PASS
Erasing................................ PASS
Parsing ELF file....................... PASS    LED_BLINKING.elf
Programming memory
WARNING: The user program and the bootloader overlap!
Programming memory..................... PASS    0x00000 0x02577
Starting Application................... PASS    RESET   0

Summary:  Total 9   Passed 9   Failed 0
```

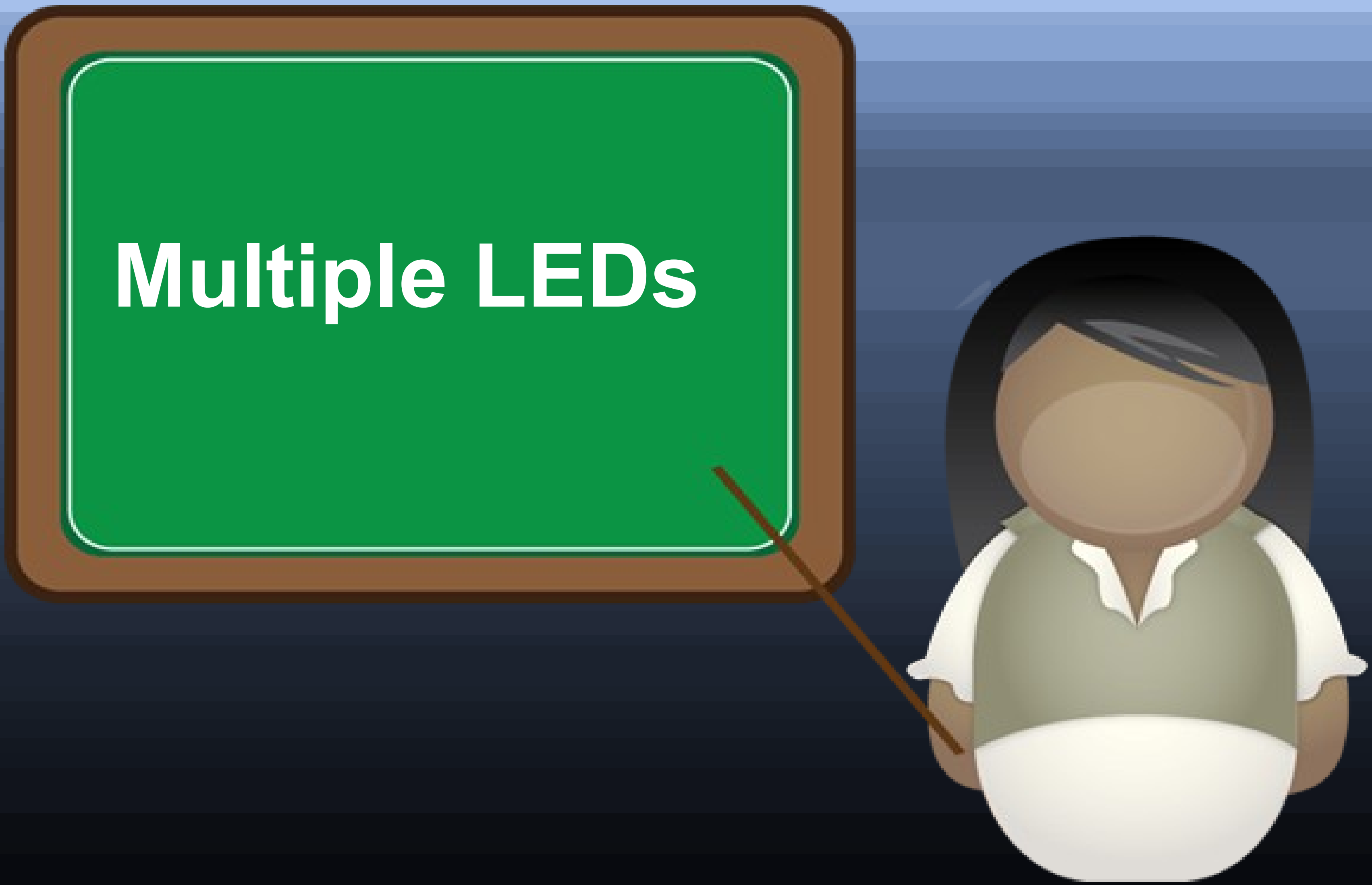
Demo

- Watch Demo video Blinking
- Watch Demo video Multiple LED

Troubleshoot

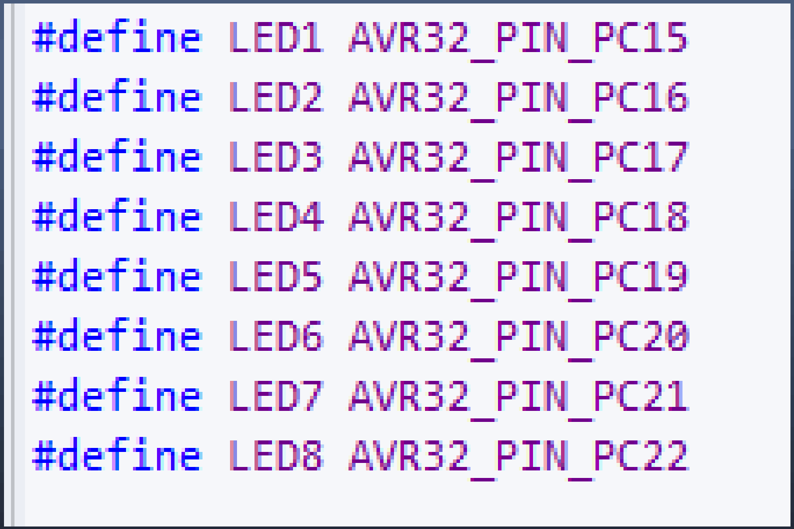
- **Output error after the external tools applied:**
- Opening port........ FAIL Could not open USB device.
- If AtLibUsbDfu not found / ISP done.
- Solution:
- 1. Go to Tools > External tools … > Arguments
- - device (Must be the same device)...
- 2. Press the Load and Reset button on board and release.
- The Microchip Tools > AT32UC3C should appear on the device manager.
- Now try it again. It will work!

Multiple LEDs



The code

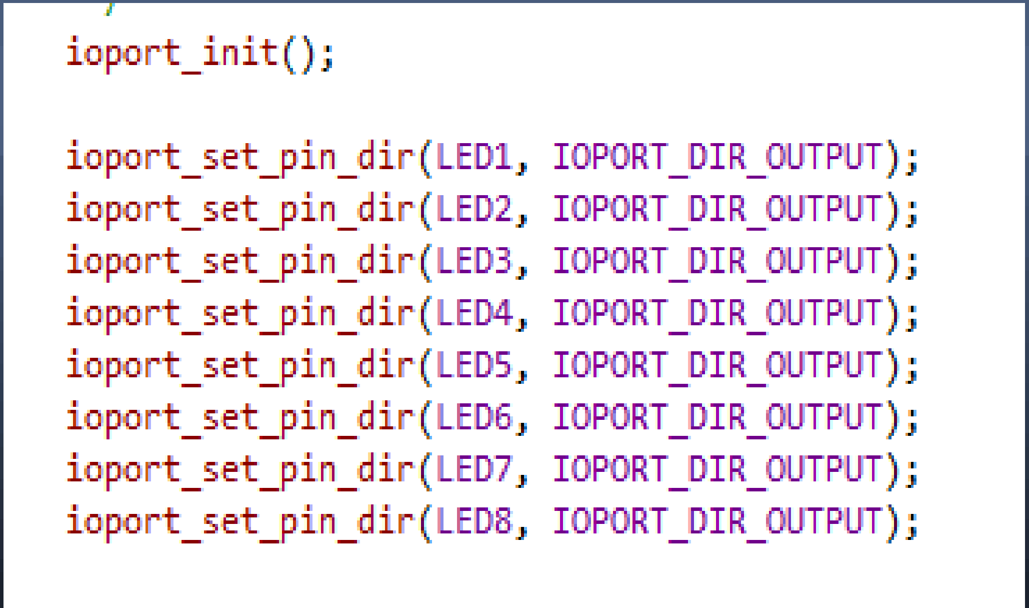
- user_board.h



```
#define LED1 AVR32_PIN_PC15
#define LED2 AVR32_PIN_PC16
#define LED3 AVR32_PIN_PC17
#define LED4 AVR32_PIN_PC18
#define LED5 AVR32_PIN_PC19
#define LED6 AVR32_PIN_PC20
#define LED7 AVR32_PIN_PC21
#define LED8 AVR32_PIN_PC22
```

The code

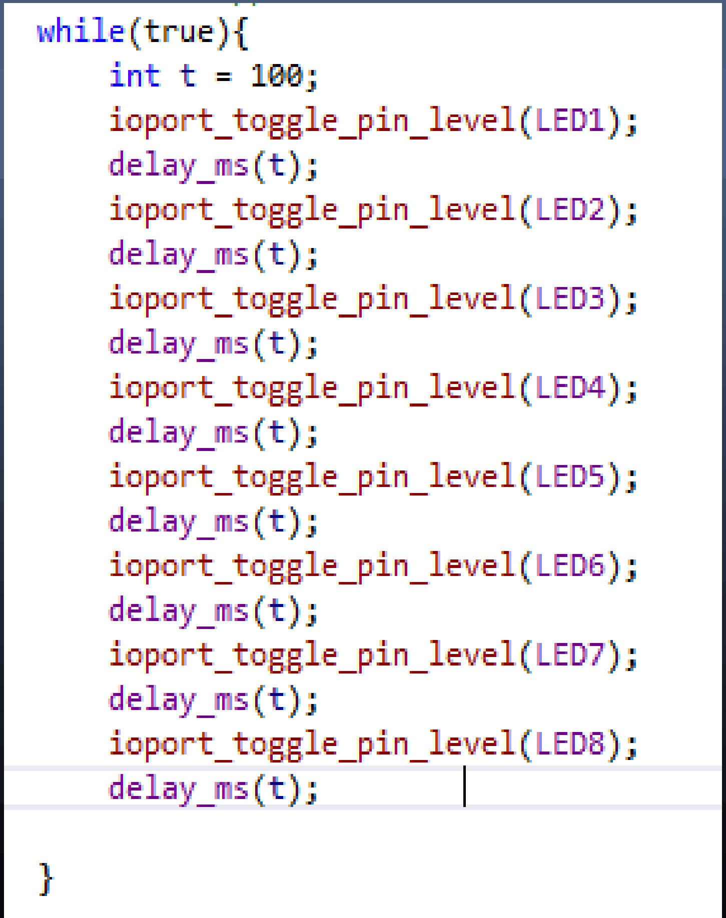
- Init.c



```
ioport_init();

ioport_set_pin_dir(LED1, IOPORT_DIR_OUTPUT);
ioport_set_pin_dir(LED2, IOPORT_DIR_OUTPUT);
ioport_set_pin_dir(LED3, IOPORT_DIR_OUTPUT);
ioport_set_pin_dir(LED4, IOPORT_DIR_OUTPUT);
ioport_set_pin_dir(LED5, IOPORT_DIR_OUTPUT);
ioport_set_pin_dir(LED6, IOPORT_DIR_OUTPUT);
ioport_set_pin_dir(LED7, IOPORT_DIR_OUTPUT);
ioport_set_pin_dir(LED8, IOPORT_DIR_OUTPUT);
```

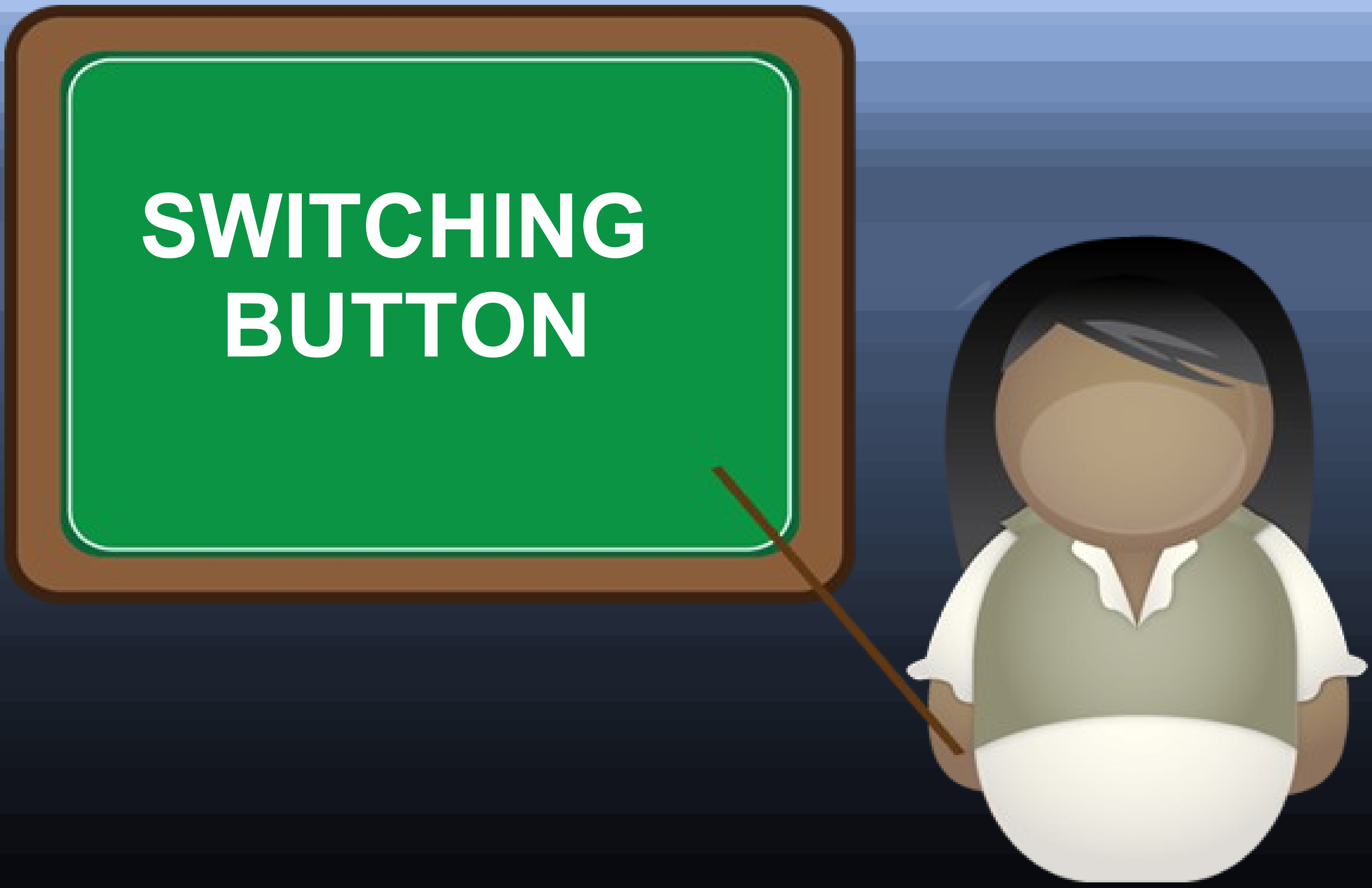
The code

- main.c



```
while(true){
    int t = 100;
    ioport_toggle_pin_level(LED1);
    delay_ms(t);
    ioport_toggle_pin_level(LED2);
    delay_ms(t);
    ioport_toggle_pin_level(LED3);
    delay_ms(t);
    ioport_toggle_pin_level(LED4);
    delay_ms(t);
    ioport_toggle_pin_level(LED5);
    delay_ms(t);
    ioport_toggle_pin_level(LED6);
    delay_ms(t);
    ioport_toggle_pin_level(LED7);
    delay_ms(t);
    ioport_toggle_pin_level(LED8);
    delay_ms(t);

}
```

SWITCHING BUTTON



Create new project

- Name: SWITCHING_BUTTON
- user_board.h

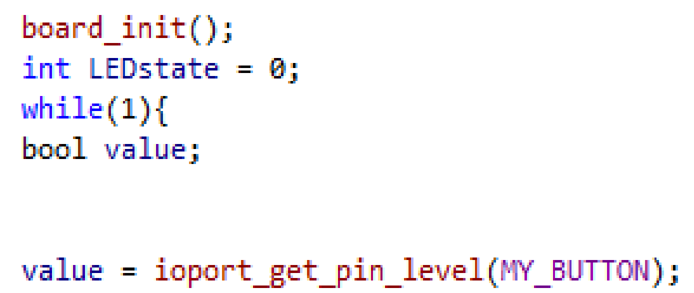
```
#define BOARD_OSC0_HZ 12000000
#define BOARD_OSC0_STARTUP_US  50000
#define BOARD_OSC0_IS_XTAL true


#define MY_LED   AVR32_PIN_PD30
#define MY_BUTTON   AVR32_PIN_PC02
```

The code

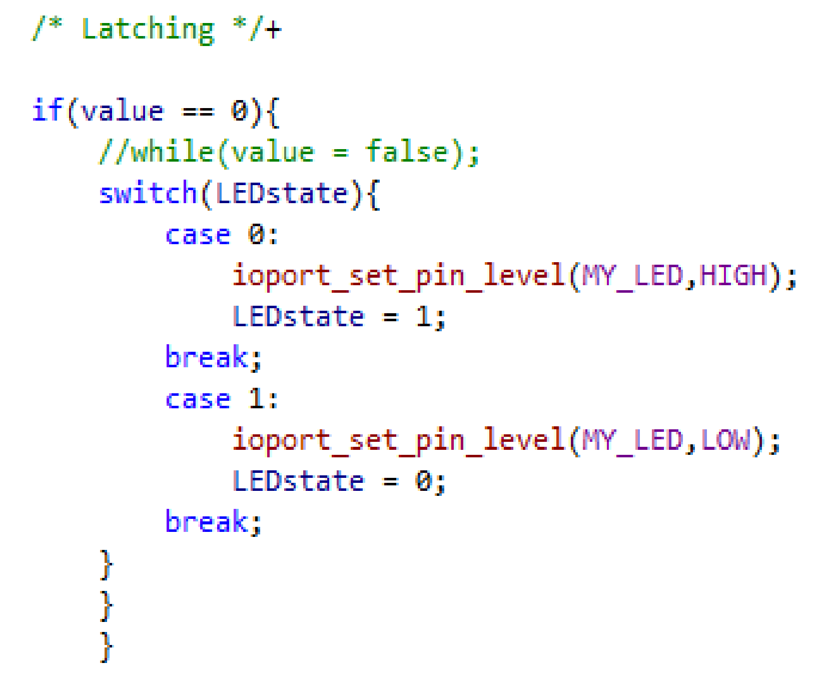
- Init.c

```
ioport_init();


ioport_set_pin_dir(MY_LED,IOPORT_DIR_OUTPUT);

ioport_set_pin_dir(MY_BUTTON, IOPORT_DIR_INPUT);

//ioport_set_pin_mode(MY_BUTTON,
IOPORT_MODE_PULLUP);
```

The main (latching switch)

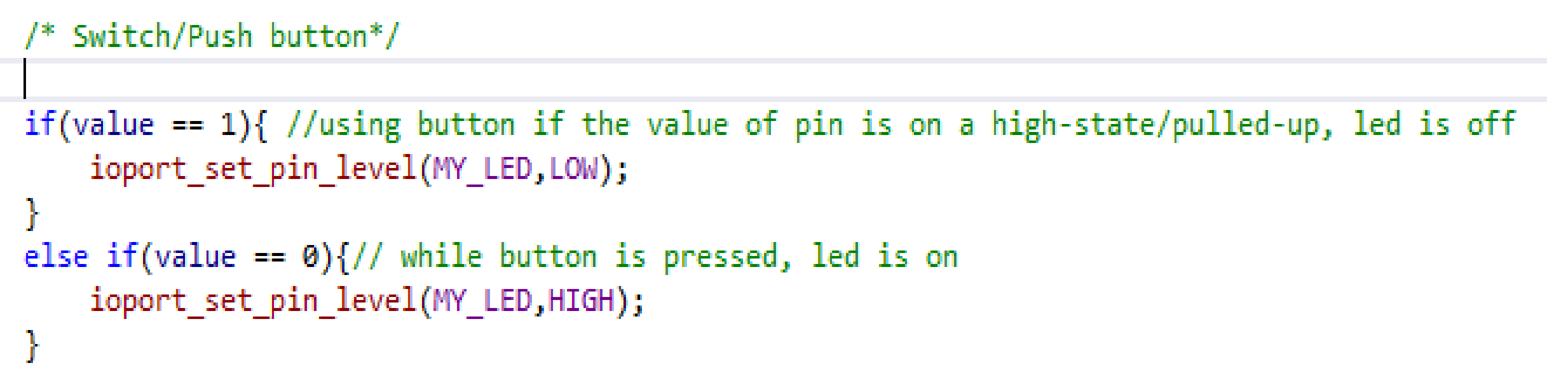


```
board_init();
int LEDstate = 0;
while(1){
bool value;


value = ioport_get_pin_level(MY_BUTTON);
```



```
/* Latching */+

if(value == 0){
    //while(value = false);
    switch(LEDstate){
        case 0:
            ioport_set_pin_level(MY_LED,HIGH);
            LEDstate = 1;
        break;
        case 1:
            ioport_set_pin_level(MY_LED,LOW);
            LEDstate = 0;
        break;
    }
    }
    }
```

The main (Push button)



```
/* Switch/Push button*/

if(value == 1){ //using button if the value of pin is on a high-state/pulled-up, led is off
    ioport_set_pin_level(MY_LED,LOW);
}
else if(value == 0){// while button is pressed, led is on
    ioport_set_pin_level(MY_LED,HIGH);
}
```

Demo

- Watch Demo video Push switch
- And Latching switch

FILES LOCATION

Uc3c2128c.h location

- /* PAD-> GPIO bits mapping */
- #define AVR32_PIN_PA00 … PD30

ioport.h location and descriptions

- *\ brief Set direction for a single IOPORT pin
- *\param pin IOPORT pin to configure
- *\param dir Direction to set for the specified pin
- ioport_set_pin_dir(ioport_pin_t pin, enum ioport_direction dir);
-
- *\ brief Toggle the value of an IOPORT pin, which has previously configured as an output.
- *\param pin IOPORT pin to toggle
- ioport_toggle_pin_level(ioport_pin_t pin);
- **Select the function, right-click Goto Implementation (Alt+G);**

Common IOPORT service main header file for AVR, UC3 and ARM architectures (Basics)

- IOPORT_DIR_INPUT *\ input pin direction
- IOPORT_DIR_OUTPUT *\ output pin direction
- IOPORT_PIN_LEVEL_LOW *\ pin value low
- IOPORT_PIN_LEVEL_HIGH *\pin value high
- ioport_init(); *\ initialize the IOPORT service, ready for use.
- *\ This function must be called before using any other functions in the IOPORT service.
- ioport_enable_pin(ioport_pin_t pin); *\ Enable an IOPORT pin
- ioport_enable_port(ioport_port_t port, ioport_port_mask_t mask); *\param mask Mask of pins within the port to enable
- ioport_disable_pin(ioport_pin_t_pin); *\ Disable IOPORT pin
- ioport_disable_port(ioport_port_t port, ioport_port_mask_t mask); *\param mask Pin mask of pins to disable

Common IOPORT service main header file for AVR, UC3 and ARM architectures (Basics)

- ioport_set_port_mode (ioport_port_t port, ioport_port_mask_t mask, ioport_mode_t mode); *\Set multiple pin modes in a single IOPORT, such as pull-up, pull-down, etc. config;param mode Mode masks to configure for the specified pin
- ioport_set_pin_mode (ioport_pin_t pin, ioport_mode_t mode); *\Set pin mode for one single IOPORT pin;
- ioport_reset_port_mode(ioport_port_t port, ioport_port_mask_t mask); *\ Reset multiple pin modes in a specified IOPORT port to defaults; param Mask of pins whose mode configuration is to be reset.
- ioport_reset_pin_mode(ioport_pin_t pin); *\ Reset pin mode configuration for a single IOPORT pin; pin to configure
- ioport_set_port_dir(ioport_port_t port, ioport_port_mask_t mask, enum ioport_direction dir); *\ Set I/O direction for a group of pins in a single IOPORT.
- ioport_set_pin_dir***
- ioport_set_pin_level(ioport_pin_t pin, bool level); *\ Set an IOPORT pin to a specified logical value.

Common IOPORT service main header file for AVR, UC3 and ARM architectures (Basics)

- ioport_set_port_level(ioport_port_t port, ioport_port_mask_t mask, enum ioport_value level); *\ Set a group of IOPORT pins in a single port to a specified logical value
- ioport_get_pin_level(ioport_pin_t pin); *\Get current value of an IOPORT pin, which has been configured as an input
- ioport_get_port_level(ioport_pin_t port, ioport_port_mask_t masl); *\Get current value of several IOPORT pins in a single port, which have been configured as an inputs.
- ioport_toggle_pin_level***
- ioport_toggle_port_level(ioport_port_t port, ioport_port_mask_t mask); *\Toggle the values of serveral IOPORT pins located in a single port.
- ioport_set_pin_sense_mode(ioport_pin_t pin, enum ioport_sense pin_sense); *\Set the pin sense mode of a single IOPORT pin;param pin_sense Edge to sense for the pin
- ioport_set_port_sense_mode(ioport_port_t port, ioport_port_mask_t mask, enum ioport_sense pin_sense); *\ Set the pin sense mode of a multiple IOPORT pins on a single port

Common IOPORT service main header file for AVR, UC3 and ARM architectures (Basics)

- ioport_pin_to_port_id(ioport_pint_t pin); *\Convert a pin ID into a its port ID;param pin IOPORT pin ID to cenvert; retval Port ID for the given pin ID
- ioport_pin_to_mask(ioport_pin_t pin); *\Convert a pin ID into a bitmask mask for the given pin on its port; param pin IOPORT pin ID to convert;retval Bitmask with a bit set that corresponds to the given pin ID in its port.

Common IOPORT service main header file for AVR, UC3 and ARM architectures (Basics)

- #define MY_LED IOPORT_CREATE_PIN(PORTA,5)
- #define MY_BUTTON IOPORT_CREATE_PIN(PORTA,6)
- ioport_init();
-
- ioport_set_pin_dir(MY_LED, IOPORT_DIR_OUTPUT);
- ioport_set_pin_dir(MY_BUTTON, IOPORT_DIR_INPUT);
- ioport_set_pin_mode(MY_BUTTON, IOPORT_MODE_PULLUP);
-
- Bool value;
- Value = ioport_get_pin_level(MY_BUTTON);
- ioport_set_pin_level(MY_LED, value);

Go to
SWITCHING_BUTTON
Project

Reference

A very big Thank you! To motion55 @elab.ph forum for the “The AVR32 Tutorial”.

- Product page: https://www.e-gizmo.net/oc/index.php?route=product/product&search=AT32&product_id=1418