Review of C-Programming:
Bitwise Operation (for Embedded Controller)

Prof. Young-Keun Kim

mod .2023.08.19

Firmware programming example

- Example of firmware programming in C

#Define

Pointer Typecasting

Enum

Structures

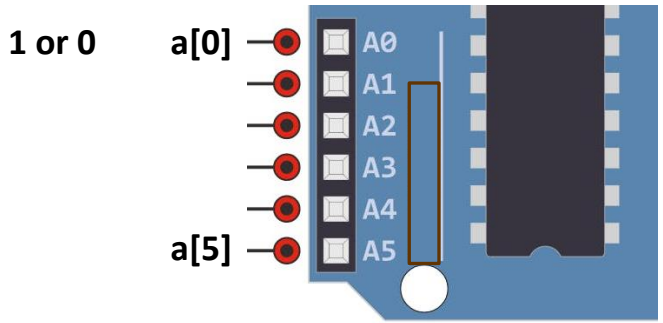
Bitwise

```
#ifndef __EC_GPIO_H
#define __EC_GPIO_H

#define INPUT 0x00
#define GPIOA  ((GPIO_TypeDef *) GPIOA_BASE)

typedef enum
{
    /******  Cortex-M4 Processor Exceptions Numbers ***                */
    RCC_IRQn = 5,      /*!< RCC global Interrupt                       */
    EXTI0_IRQn = 6,     /*!< EXTI Line0 Interrupt                      */
    EXTI1_IRQn = 7,     /*!< EXTI Line1 Interrupt                      */
}

typedef struct
{
    __IO uint32_t MODER;    /*!< GPIO port mode register,          Address offset: 0x00     */
    __IO uint32_t OTYPER;   /*!< GPIO port output type register,     Address offset: 0x04     */
    __IO uint32_t ODR;      /*!< GPIO port output data register,     Address offset: 0x14     */
} GPIO_TypeDef;


// GPIO Mode : Input(00), Output(01), AlterFunc(10), Analog(11, reset)
void GPIO_mode(GPIO_TypeDef* Port, int pin, int mode) {
    Port->MODER &= ~(3UL << (2 * pin));
    Port->MODER |= mode << (2 * pin);
}
```

Bitwise Operation

- Online Resource for Bitwise Operation

- Bitwise operation
  - Shift, Toggle
  - AND/OR/XOR/NOT (Boolean vs Logical)



- Why use Bitwise operation?

unit8_t a=5

a has 8-bits memory (NOT 8-bytes)

memory **a**	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
value **5**	0	0	0	0	0	1	0	1

How to read or set only the bit at **a[5]**? // it is not 1D array index

MCU I/O pin is binary (1 or 0).
Each pin is connected to one bit (not bytes) → Need to access and modify bits
e.g. Turn on only the LED on pin **a[5]**

Shift Operator

- **Shift**
  - For controlling MCU, we only use logical shift (not arithmetic)

  - Logical shift: Fill the blanks with '0', regardless of sign
    s<<2: shift to left by two digits
    s>>2: shift to right by two digits

unsigned s=00100 (4)

(LSL) S<<2 → 10000 (16)

(LSR) S>>2 → 00001 (1)

s=1111 0011

(LSR) S>>2→ 0011 1100

(LSL) S<<2 → 1100 1100

- Multiply or divide by 2^n for unsigned number

```
(VAL<<4);       // Multiply VAL by 2^4=16

(VAL>>4);       // Divide VAL by 2^4=16
```

* Careful for unsigned or signed numbers

Shift Operator

- **Shift** operator in MCU:
  - See Tutorial: Bitwise Operation

  - Check / Set / Reset of a bit(s) at target positions

    - Examples:

```
PA=PA | (1<<5);          //set PA5 (as High) and mask others
PA=PA & ~(1<<5);         //reset PA5 (as LOW) and mask others
Bit = PA & (1<<5);       //check the bit5. bit=1 if  PA5 is 1
```

Bit Operator vs Boolean Operator

- Need to know how to read and control individual bits of I/O memory
  - Check / Set / Reset / Toggle bit(s)

- Bit operators
  - Bit Operators: (&, |, ~)
  - Boolean Operators: (&& ,||, !) // True or False

A && B	Boolean AND	A & B	Bitwise AND
A||B	Boolean OR	A|B	Bitwise OR
!B	Boolean NOT	~B	Bitwise NOT
		A^B	Bitwise XOR

Code	Result	Comment
b10 & b01	b00	Bitwise
b10 && b01	b01	Boolean
~(0x01)		Bitwise
!(0x01)		Boolean
0x11^0x01		Bitwise

Bit Read

- **Read the bit at k^{th} index**: **bit = a & (1<<k)**
  - Shift 'bit 1' left by k starting from LSB

1	**&**	**1**	= 1
0	**&**	**1**	= 0

Example: read bit at a[5] from 8-bit number **a**

[Method 1]

	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
a								
1 << 5	0	0	1	0	0	0	0	0
a & (1<<5)	0	0	a_5	0	0	0	0	0

[Method 2]
(recommend)

	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
a								
a>>5	0	0	0	0	0	a_7	a_6	a_5
(a >>5)& (1)	0	0	0	0	0	0	0	a_5

Bit Read

- **Reading one bit**:
  Exercise: Read a[6] from port **a**

  uint8_t bit= a_____ & _____; // check bit of a6

- **Reading multiple bits**:
  (warning) read value is not binary anymore. 2bits --> (0~3)

  Exercise: Read a[6] and a[5] (2-bits) from port **a**

a	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
val								

uint8_t val=_____&_____; // check bits of a6, a5, (0~3)

Bit Read

- **Reading multiple bits**:

Example: read CKD[1:0] from TIM1_CR1

TIM1 and TIM8 control register 1 (TIMx_CR1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

uint16_t ClkDiv= (TIM1_CR1>>8) & 0x03

uint16_t ClkDiv=

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Reserved						CKD[1:0]	
														rw	rw

Bit Set

- **Set(HIGH) a bit** in a number

a = a | (1 << k) or a |= (1 << k)

X	|	1 = **1**
X	|	0 = X ← masking

Example: **a =a | (1<<5); // set a5**

	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
a								
1 << 5	0	0	**1**	0	0	0	0	0
a | (1 << 5)	a_7	a_6	**1**	a_4	a_3	a_2	a_1	a_0

- Exercise: Turning ON LEDs of Port A(PA)
  - assume 8 LEDs are connected to Digital Out pins of PA

PA=b00001111; //LED0 is LSB, Set to turn on LED

PA|=_____; // turn ON LED4

PA|=_____; // turn ON LED4 and LED5

Bit Clear

- **Clear(LOW) a bit in a number**

a &= ~(1<<k)

X & 0 = 0
X & 1 = X ← masking

Example: a &=~(1<<5); // clear bit a5

~(1<<5) → ~(0010 0000) → (1101 1111)

a	a_7	a_6	**a_5**	a_4	a_3	a_2	a_1	a_0
~(1 <<5k)	1	1	**0**	1	1	1	1	1
a & ~(1<<5)	a_7	a_6	**0**	a_4	a_3	a_2	a_1	a_0

- Exercise: Turning off LEDs of Port A(PA)

PA=b00001111; //LED0 is LSB, Set to turn on LED

PA&=_____; // turn off LED2

PA&=_____; // turn off LED0 and LED2

Bit Clear

- **Toggle a bit**

  a ^= 1<<k

X^1 =~X ← (XOR)
X^0 =X ← masking

Without knowing the initial value, a bit can be toggled by XORing it with a "1"

Example: Blink LED5 on and off, **k = 5.**

a	a_7	a_6	**a_5**	a_4	a_3	a_2	a_1	a_0
1 << k	0	0	**1**	0	0	0	0	0
a ^= 1<<k	a_7	a_6	NOT(**a_5**)	a_4	a_3	a_2	a_1	a_0

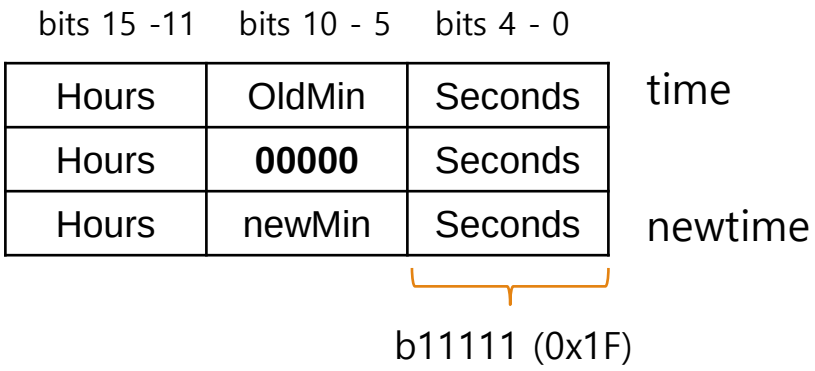
a_5	1	$a_5 \oplus 1$
0	1	1
1	1	0

Bit Replace

- **Replacing multiple bits**

TIP: Always (1) Reset first then (2) Write

Example: replace oldMin by newMin at time[5-10]




```
newtime = time & ~(0x1F<<5)   //  First, reset bit (10~5)

newtime | = (newMin & 0x1F)<<5  // Then, write bits
```

Bitwise Operation: Summary

- Set(HIGH) a bit in a number
  a |= (1 << k)

- Clear(LOW) a bit in a number
  a &= ~(1<<k)

- Read/Check a bit:
  bit = (a>>k) & 1

- Toggle a bit
  a ^= 1<<k

X & 0 = 0
X & 1 = X

X | 1 = **1**
X | 0 = X

1 **&** **1** = 1
0 **&** **1** = 0

X^1 =~X ← (XOR)

// Macro defined in stm32fxx.h

```
#define SET_BIT(REG, BIT)     ((REG) |= (BIT))
#define CLEAR_BIT(REG, BIT)((REG) &= ~(BIT))
#define READ_BIT(REG, BIT)((REG) & (BIT))
```

Exercise: Bitwise operation

Exercise: bitwise operation

- Read instruction in the given source file.
- Apply bitwise operations (Set HIGH, Toggle, Reset etc) as instructed.

C_bitwise_exercise4.c

```
118 as binary = 01110110
result1 = 0
result2 = 11110110
result3 = 11110010
result4 = 11111110
result5 = 11111111
result6 = 01111110
```

```c
#include <stdio.h>

void main() {

    unsigned int a = 118;
    printf("\n118 as binary = ");
    dec2bin(a);

    //read 8th bit of a :   a[7]
    unsigned int bit_check = a & (1 << 7);
    printf("\nresult1 = %d", bit_check);


    //set 8th (a[7])  bit as HIGH
    // [YOUR CODE GOES HERE]

    printf("\nresult2 = ");
    dec2bin(a);
    ..

}
```

```c
void dec2bin(unsigned int n) {
    unsigned int a = 0x80;
    for (int i = 0; i < 8; i++) {
        if ((a & n) == a)
                printf("1");
        else
                printf("0");
        a = a>> 1;
    }
}
```