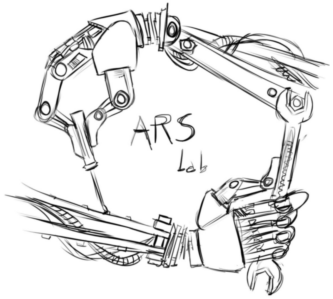
Bitwise Operations in C

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it



L.S.M. Course

Bit Manipulation Operations

- To perform operations on a SFR we need to **manipulate single bits**
- Set (to 1) a specific bit.
- Clear (set to 0) a specific bit.
- “Toggle” a specific bit.
- Test a specific bit.
- These operations are performed using bit-mask

Example: the GPIO MODER Register

8.4.1 GPIO port mode register (GPIOx_MODER) (x = A..E and H)

Address offset: 0x00

Reset values:

- 0x0C00 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O direction mode.

00: Input (reset state)

01: General purpose output mode

10: Alternate function mode

11: Analog mode

Setting a bit

- Make an **OR** operation with a constant bit pattern formed as follows:
  - The bit to be set is “1”
  - All the other bits are “0”
- Example: setting the bit 3 of the (8-bit) variable `A`:

A	B7	B6	B5	B4	B3	B2	B1	B0	OR
mask	0	0	0	0	1	0	0	0	=
A	B7	B6	B5	B4	1	B2	B1	B0	

```
A = A | 0x08;
A |= 0x08;
```

Clearing a bit

- Make an **AND** operation with a constant bit pattern formed as follows:
  - The bit to be cleared is “0”
  - All the other bits are “1”
- Example: clearing the bit 6 of the (8-bit) variable `A`:

A	B7	B6	B5	B4	B3	B2	B1	B0	AND
mask	1	0	1	1	1	1	1	1	=
A	B7	0	B5	B4	B3	B2	B1	B0	

```
A = A & 0xbf;
A &= 0xbf;
```

Clearing a bit (using negation)

- Make an **AND** operation with a constant bit pattern formed as follows:
  - The bit to be cleared is “0”
  - All the other bits are “1”
- Example: clearing the bit 6 of the (8-bit) variable `A`:

```
A = A & ~0x40;
A &= ~0x40;
```

	B7	B6	B5	B4	B3	B2	B1	B0	
~	0	1	0	0	0	0	0	0	=
	1	0	1	1	1	1	1	1	

A	B7	B6	B5	B4	B3	B2	B1	B0	AND
mask	1	0	1	1	1	1	1	1	=
A	B7	0	B5	B4	B3	B2	B1	B0	

Toggling a bit

- Make an **XOR** operation with a constant bit pattern formed as follows:
  - The bit to be set is “1”
  - All the other bits are “0”
- Example: toggling the bit 4 of the (8-bit) variable `A`:

A	B7	B6	B5	B4	B3	B2	B1	B0	XOR
mask	0	0	0	1	0	0	0	0	=
A	B7	B6	B5	$\overline{B4}$	B3	B2	B1	B0	

```
A = A ^ 0x10;
A ^= 0x10;
```

Testing a bit

- Make an **AND** operation with a constant bit pattern formed as follows:
  - The bit to be tested is “1”
  - All the other bits are “0”
- Check if the result is zero or non-zero
- Example: testing the bit 5 of the (8-bit) variable `A`:

A	B7	B6	B5	B4	B3	B2	B1	B0	AND
mask	0	0	1	0	0	0	0	0	=
	0	0	B5	0	0	0	0	0	

```
if ((A & 0x20) != 0) ...   // non-zero
```

An Example

Given A a 8-bit variables as $\{B7, B6, B5, B4, B3, B2, B1, B0\}$, determine the result of the following program:

```
A |= 0x30;
A ^= 0x0c;
A &= ~0x80;
```

Given A a 16-bit variables as $\{B15, B14, B13, B12, B11, B10, B9, B8, B7, B6, B5, B4, B3, B2, B1, B0\}$, determine the result of the following program:

```
A &= 0xf3ff;
A ^= 0x0100;
A |= 0x8000;
```

Example: Setting the GPIO MODER Register

We want to set the MODE of PIN12 of GPIOA as “output”:

8.4.1 GPIO port mode register (GPIOx_MODER) (x = A..E and H)

Address offset: 0x00

Reset values:

- 0x0C00 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O direction mode.
00: Input (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

1. Clear bits 24-25: `GPIOA->MODER &= 0xfcffffff;`
2. Set bit 24: `GPIOA->MODER |= 0x01000000;`

Setting a bit with bit-shift

- We can built bit pattern by using left-shift with a shift count **equal to the bit number to manipulate**
- Example: setting the bit 3 of the (8-bit) variable `A`:

```
A = A | (1 << 3);
A |= (1 << 3);
```

1	0	0	0	0	0	0	0	1	<< 3 =
	0	0	0	0	1	0	0	0	0x08

A	B7	B6	B5	B4	B3	B2	B1	B0	OR
mask	0	0	0	0	1	0	0	0	=
A	B7	B6	B5	B4	1	B2	B1	B0	

Clearing a bit with bit-shift

- Example: Clearing the bit 4 of the (8-bit) variable `A`:

```
A = A & ~(1 << 4);
A &= ~(1 << 4);
```

1	0	0	0	0	0	0	0	1	<< 4 =
	0	0	0	1	0	0	0	0	~ =
	1	1	1	0	1	1	1	1	0xEF

A	B7	B6	B5	B4	B3	B2	B1	B0	OR
mask	1	1	1	0	1	1	1	1	=
A	B7	B6	B5	0	B3	B2	B1	B0	

Example: Setting the GPIO MODER Register

We want to set the MODE of PIN12 of GPIOA as “output”:

8.4.1 GPIO port mode register (GPIOx_MODER) (x = A..E and H)

Address offset: 0x00

Reset values:

- 0x0C00 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O direction mode.
00: Input (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

1. Clear bits 24-25: `GPIO->MODER &= ~(int32_t)(3 << 24);`
2. Set bit 24: `GPIO->MODER |= (1 << 24);`

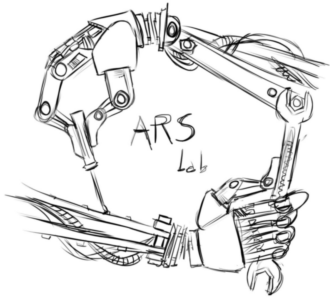
Additional C Integer Types usefull from MCU Programs

- `int8_t` 8 bit signed integer
- `uint8_t` 8 bit unsigned integer
- `int16_t` 16 bit signed integer
- `uint16_t` 16 bit unsigned integer
- `int32_t` 32 bit signed integer
- `uint32_t` 32 bit unsigned integer

Bitwise Operations in C

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it



L.S.M. Course