Bit Manipulation

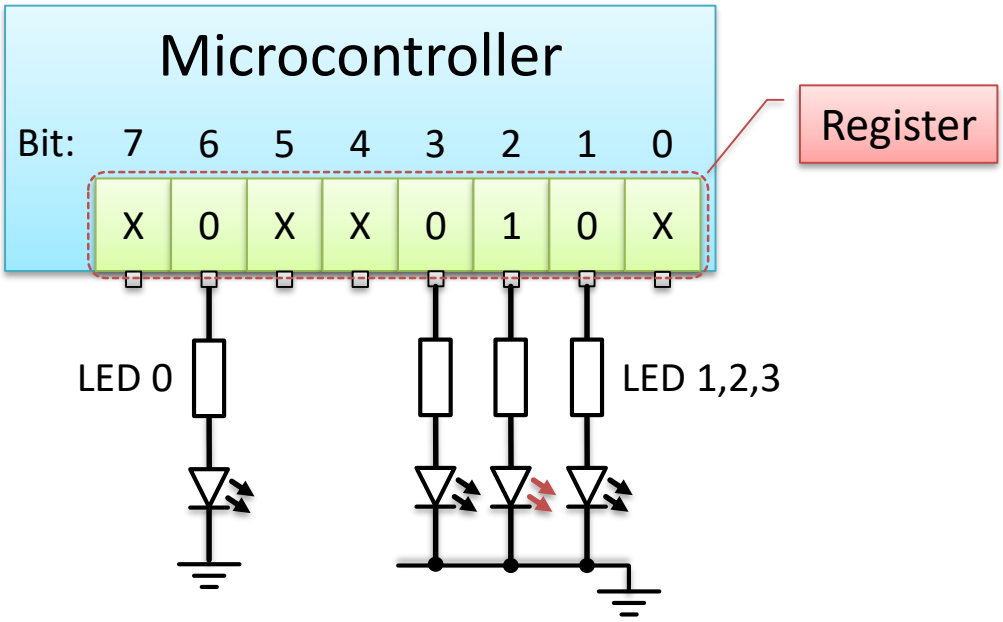
Networks and Embedded Software

Module 4.3.1

by Wolfgang Neff

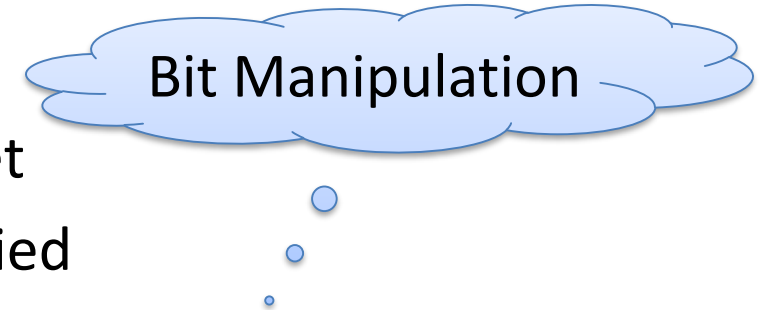
Motivation (1)

- Example: LED Control
  - Hardware Setup



Motivation (2)

- Example: LED Control (continued)
  - LEDs are connected to the pins
  - Output register controls pins
  - Registers can be changed on the whole, only
  - LED 0 shall be turned on
    - Bit on position 6 must be set
    - No other bit must be modified
    - New content of register has to be calculated



Bitwise Operators (1)

- Operator ~: bitwise NOT

a	0	0	1	1	0	1	1	0
~a	1	1	0	0	1	0	0	1

- Bitwise &: bitwise AND

a	0	0	1	1	0	1	1	0
b	0	0	0	0	1	1	1	1
a&b	0	0	0	0	0	1	1	0

Bitwise Operators (2)

- Operator |: bitwise OR

a	0	0	1	1	0	1	1	0
b	0	0	0	0	1	1	1	1
a|b	0	0	1	1	1	1	1	1

- Bitwise ^: bitwise XOR

a	0	0	1	1	0	1	1	0
b	0	0	0	0	1	1	1	1
a^b	0	0	1	1	1	0	0	1

Bitwise Operators (3)

- Operator <<: bitwise left shift

a	0	0	1	1	0	1	1	0
a<<2	1	1	0	1	1	0	0	0

- Bitwise >>: bitwise right shift

a	0	0	1	1	0	1	1	0
a>>2	0	0	0	0	1	1	0	1

  - Attention: implementation dependent

Bit Manipulation (1)

- Individual bits
  - Bit position (_bp)
    - Example: UART.CTRL

Bit	7	6	5	4	3	2	1	0
Name	CMODE		PMODE		SBMODE	CHSIZE		

      - UART_SBMODE_bp: 3
  - Bit mask (_bm)
    - Byte with one bit set at bit position
    - Example: UART_SBMODE_bm: 00001000

Bit Manipulation (2)

- Bits groups
  - Group position (_gp)
    - Start position of a bit group
    - Example: UART_PMODE_gp: 4
  - Group mask (_gm)
    - Byte with all bits of a bit group set
    - Example: UART_PMODE_gm: 0011 0000

Bit Manipulation (3)

- Bits groups (continued)
  - Group configuration (_gc)
    - A meaningful configuration of group bits
    - Example: PMODE configuration
      - UART_PMODE_NONE_gc: 00000000
      - UART_PMODE_EVEN_gc: 00100000
      - UART_PMODE_ODD_gc: 00110000
- Defines are declared in header file
  - Usually no need for user action

Bit Manipulation (4)

- Testing bits
  - Test if a specific bit is set
    - `if (REG & BIT_bm) { … }`
  - Test if a specific bit is cleared
    - `if (!(REG & BIT_bm)) { … }`

logical bitwise parenthesis

  - Test if a bit group has a specific configuration
    - `if ((REG & GROUP_gm) == GROUP_gc) { … }`

Bit Manipulation (5)

- Manipulation of individual bits
  - Setting one or more bits
    - `REG |= BIT_bm; // REG = REG | BIT_bm`
    - `REG |= (BIT1_bm | BIT2_bm | …);`
  - Clearing one or more bits
    - `REG &= ~BIT_bm; // REG = REG & ~BIT_bm`
    - `REG &= ~(BIT1_bm | BIT2_bm | …);`
  - Toggling one or more bits
    - `REG ^= BIT_bm; // REG = REG ^ BIT_bm`

Bit Manipulation (6)

- Manipulation of bit groups
  - Clearing a bit group
    - `REG = REG & ~GROUP_gm;`
  - Configuring a bit group
    - `REG = (REG & ~GROUP_gm) | GROUP_gc;`
    - First clear bit group then set configuration bits
  - Assigning a specific value to a bit group
    - `REG = (REG & ~GROUP_gm) | (value << GROUP_gp);`