# Real-Time Programming TI00AA55

**Bit operations** 

Jarkko.Vuori@metropolia.fi

# Bit-wise operations 1

- We already know:
  - Type int is used to express logical information
  - Operations for logical information are AND (&&), OR (||) and NOT (!)
- The smallest addressable unit in memory is byte. Sometimes however we want to access one specific bit (or bit field). To modify a bit, the procedure is a three phase operation:
  - 1. The byte containing a bit is moved to the register in the processor
  - 2. Apply the operations to the register so, that only the specified bit is changed
  - 3. Move the byte back to the memory (device register)

- Operations we often need are:
  - Test the current value of the bit (is it originally On or OFF?)
  - Set the bit (set it to ON-state (1))
  - Clear the bit (set it to OFF-state(0))
  - Invert the bit
- In addition to these operation we often need manipulation of bit fields
  - Determine the value of a bit field
  - Set a value to the bit field

### Bit-wise operations 2

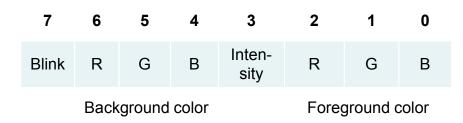
- Bit-wise operations and comparison to logical operations:
  - Remark. Both operations are defined for an integer

Operation	Logical oper.	Bit-wise operation oper.
AND	&&	&
OR	II	1
XOR		٨
NOT	!	~

#### How these operators work in practice

# How to use them in practice 1

 Our example is the attribute byte of the display of PC in text mode. The format of attribute byte in display memory is:



 Lets now assume that variable attr contains the attribute byte of a certain screen location. Variable attr is defined in the following way: unsigned char attr;

#### 1. Testing a bit position

```
- Example. Is blinking bit set?
  #define BLINK 0x80
  if (attr & BLINK)
    printf("\nBlinking is on")
  else
    printf("\nBlinking is off")
```

#### How to use them in practice 2

- 2. Setting a bit on (set it ON or set it to 1)
  - Example. Set the intensity bit ON
     #define INTENSITY 0x08
     ...
     attr = attr | INTENSITY;
- 3. Clearing a bit (set it OFF or set it to 0)
  - Example. Set intensity OFF
     #define INTENSITY 0x08
    ...
    attr = attr & ~INTENSITY;

- 4. Inversion of the bit
  - Example. Invert the state of the INTENSITY bit.

```
#define INTENSITY 0x08
...
attr = attr ^ INTENSITY;
```

- In all of these operations you have to choose:
  - 1. Mask
  - 2. operation

### How to use them in practice 3

- To make sure your operation works correctly check the following:
  - The specified bit position
    - 1. becomes to required if it originally is 0
    - 2. becomes to required if it originally is 1
  - All other bits remain the same
    - 3. 0 remains the same
    - 4. 1 remains the same
- Original Χ Χ Χ Χ Χ Χ X data Mask S S S S Result X Χ Χ Χ Χ Χ Χ from from original original ult

- Sometimes we need an access to a bit field consisting of several bits
- This is the case for example if we have to answer the question:
  - What is the colour number of background colour?
  - or if we have to set background colour to red (colour no 6)
  - Then we need shift operations

s "preserves" m modifies t is the result

TI00AA55/JV

#### Shift operations

- Shift left << and shift right >>
- Is the shift right operation arithmetic or logical?
  - It is not defined in the standard →
     It is better to use it only for
     unsigned data or mask the bits
     that are fed in from the left

- Find out the value of the bit field
  - Example. What is the background color ... bcolor = ( attr >> 4) & 0x07;
- Set a value into the bit field
  - Example. Set brown as the background color

```
#define BACKGCOLOR 0x70
#define FOREGCOLOR 0x07
...
bcolor = 6; // brown for example
attr = (attr & ~BACKGCOLOR) | (bcolor
<< 4)

erase old value set new value</pre>
```