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
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
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

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 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
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 \( A \):

<table>
<thead>
<tr>
<th>( A )</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>mask</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>=</td>
</tr>
<tr>
<td>( A )</td>
<td>B7</td>
<td>B6</td>
<td>B5</td>
<td>B4</td>
<td>1</td>
<td>B2</td>
<td>B1</td>
<td>B0</td>
<td></td>
</tr>
</tbody>
</table>

\[
A = A | 0x08; \\
A |= 0x08;
\]
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`:

<table>
<thead>
<tr>
<th>A</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
<th>AND</th>
</tr>
</thead>
<tbody>
<tr>
<td>mask</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

```
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 $A$:

  $$A = A \ & \ \sim 0x40;$$
  $$A \ &=\ \sim 0x40;$$

<table>
<thead>
<tr>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sim$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
<th><strong>AND</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>mask</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>=</td>
</tr>
<tr>
<td>A</td>
<td>B7</td>
<td>0</td>
<td>B5</td>
<td>B4</td>
<td>B3</td>
<td>B2</td>
<td>B1</td>
<td>B0</td>
<td>=</td>
</tr>
</tbody>
</table>
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`:

<table>
<thead>
<tr>
<th></th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
<th>XOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>mask</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>=</td>
</tr>
<tr>
<td>A</td>
<td>B7</td>
<td>B6</td>
<td>B5</td>
<td>B4</td>
<td>B3</td>
<td>B2</td>
<td>B1</td>
<td>B0</td>
<td></td>
</tr>
</tbody>
</table>

\[
A = A \ ^\ 0x10;
\]
\[
A \ ^= 0x10;
\]
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 \( A \):

<table>
<thead>
<tr>
<th>A</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
<th>AND</th>
</tr>
</thead>
<tbody>
<tr>
<td>mask</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>=</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>B5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{if } \((A \& 0x20) \neq 0) \ldots \text{ // non-zero}\]
An Example

Given $A$ a 8-bit variables as $\{B_7, B_6, B_5, B_4, B_3, B_2, B_1, B_0\}$, determine the result of the following program:

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

Given $A$ a 16-bit variables as $\{B_{15}, B_{14}, B_{13}, B_{12}, B_{11}, B_{10}, B_9, B_8, B_7, B_6, B_5, B_4, B_3, B_2, B_1, B_0\}$, determine the result of the following program:

```
A &= 0xf3ff;
A ^= 0x0100;
A |= 0x8000;
```
We want to set the MODE of PIN12 of GPIOA as “output”:

1. Clear bits 24-25: `GPIOA->MODER &= 0xfcffffff;
2. Set bit 24: `GPIOA->MODER |= 0x01000000;`
We can build 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 \( A \):

\[
A = A | (1 << 3);
\]

\[
A |= (1 << 3);
\]

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>&lt;&lt; 3 =</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0x08</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>mask</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>=</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Example: Clearing the bit 4 of the (8-bit) variable \( A \):

\[
A = A \& \sim (1 \ll 4) ;
\]

\[
A \&= \sim (1 \ll 4) ;
\]

<table>
<thead>
<tr>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\sim = 0xEF
\]

<table>
<thead>
<tr>
<th>A</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>&lt;&lt; 4</td>
</tr>
</tbody>
</table>

\[
<< 4 =
\]

<table>
<thead>
<tr>
<th>A</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0xEF</td>
</tr>
</tbody>
</table>

OR

\[
A \mid\mid mask =
\]

<table>
<thead>
<tr>
<th>A</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

=
Example: Setting the GPIO MODER Register

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

1. Clear bits 24-25: \( \text{GPIO} \rightarrow \text{MODER} \ &= \ \sim (\text{int32.t})(3 \ll 24); \)
2. Set bit 24: \( \text{GPIO} \rightarrow \text{MODER} \ |= (1 \ll 24); \)

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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
Additional C Integer Types useful 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