Bit Manipulation using C (Register Level Programming)

Date: 15-02-2025

This document explains commonly used bit manipulation techniques used in Embedded C and ARM microcontroller programming.
Bit manipulation is widely used while working with **hardware registers, GPIO control, status flags, and peripheral configuration**.

Helper Macro

```
#define BV(n) (1 << (n))
```

This macro generates a bit mask with the **nth bit set to 1**.
It improves code readability and avoids hard-coded values.

Example:

```
BV(0) → 0000 0001
BV(3) → 0000 1000
```

Check nth Bit of Register

```
if (regr & BV(n))
{
    // nth bit is 1
}
else
{
    // nth bit is 0
}
```

Example: Checking nth bit

```
Check 3rd bit:

regr : 0x4A : 0100 1010
BV(3):        0000 1000
---------------------
Result:       0000 1000  → bit is 1 (true)
```

```
Check 4th bit:

regr : 0x4A : 0100 1010
BV(4):        0001 0000
---------------------
Result:       0000 0000  → bit is 0 (false)
```

Used in:

- Checking input pin status
- Checking peripheral ready / busy flag

Set nth Bit of Register

```
regr = regr | BV(n);
// or
regr |= BV(n);
```

This operation sets the **nth bit to 1** while keeping all other bits unchanged.

Example: Set nth bit

```
Set 4th bit:

regr : 0x4A : 0100 1010
BV(4):        0001 0000
---------------------
Result:       0101 1010
```

Used in:

- Enabling features in control registers
- Turning ON output pins
- Configuring peripheral settings

Clear nth Bit of Register

```
regr = regr & ~BV(n);
// or
regr &= ~BV(n);
```

Example: Clear nth bit

```
Clear 3rd bit:

regr : 0x4A : 0100 1010
BV(3):        0000 1000
~BV(3):       1111 0111
----------------------
Result:       0100 0010
```

Used in:

- Disabling features
- Turning OFF output pins
- Resetting configuration bits

Toggle nth Bit of Register

```
regr = regr ^ BV(n);
// or
regr ^= BV(n);
```

Example: Toggle nth bit

```
Toggle 4th bit:

regr : 0x4A : 0100 1010
BV(4):        0001 0000
----------------------
Result:       0101 1010
```

Used in:

- Toggling output pins
- Switching modes
- Flipping status flags

Set Bits 12 to 15 of Register

```
regr = regr | BV(12) | BV(13) | BV(14) | BV(15);
// or
regr |= BV(12) | BV(13) | BV(14) | BV(15);
```

Example: Setting bits 12 to 15

```
BV(12): 0000 0000 0000 0000 0001 0000 0000 0000
BV(13): 0000 0000 0000 0000 0010 0000 0000 0000
BV(14): 0000 0000 0000 0000 0100 0000 0000 0000
BV(15): 0000 0000 0000 0000 1000 0000 0000 0000
-----------------------------------------------
Mask :  0000 0000 0000 0000 1111 0000 0000 0000
```

```
regr :  0000 0000 0000 0000 0100 1010 0000 0000
-----------------------------------------------
Result:0000 0000 0000 0000 1111 1010 0000 0000
```

Clear Bits 17 to 20 of Register

```
regr &= ~(BV(17) | BV(18) | BV(19) | BV(20));
```

Example: Clearing bits 17 to 20

```
BV(17): 0000 0000 0000 0010 0000 0000 0000 0000
BV(18): 0000 0000 0000 0100 0000 0000 0000 0000
BV(19): 0000 0000 0000 1000 0000 0000 0000 0000
BV(20): 0000 0000 0001 0000 0000 0000 0000 0000
-----------------------------------------------
OR   :  0000 0000 0001 1110 0000 0000 0000 0000
~    :  1111 1111 1110 0001 1111 1111 1111 1111
```

```
regr :  0000 0000 0000 0100 1010 0000 0000 0000
-----------------------------------------------
Result:0000 0000 0000 0000 1010 0000 0000 0000
```

Read Value from Bits 19 to 24 of Register

```
value = (regr >> 19) & 0x0000003F;
```

Example

```
regr : 0x104A0000
0001 0000 0100 1010 0000 0000 0000 0000

>> 19
0000 0000 0000 0000 0000 0010 0000 1001

& 0x3F
0000 0000 0000 0000 0000 0000 0011 1111
---------------------------------------
Result:
0000 0000 0000 0000 0000 0000 0000 1001
```

Write Value to Bits 8 to 15 of Register

```
regr &= ~(BV(8) | BV(9) | BV(10) | BV(11) |
          BV(12) | BV(13) | BV(14) | BV(15));

regr |= (value << 8);
```

Example

```
value : 0x52
0000 0000 0000 0000 0000 0000 0101 0010

value << 8
0000 0000 0000 0000 0101 0010 0000 0000
```

```
regr before:
0001 0000 0000 0100 1010 0000 0000 0000

After clear:
0001 0000 0000 0100 0000 0000 0000 0000

Final result:
0001 0000 0000 0100 0101 0010 0000 0000
```

`

`

Wait While Bit 4 of Register is 0

```
while (!(regr & BV(4)))
{
    // wait until bit 4 becomes 1
}
```

Used in:

- Waiting for peripheral ready
- Polling status flag

Wait While Bit 4 of Register is 1

```
while (regr & BV(4))
{
    // wait until bit 4 becomes 0
}
```

Used in:

- Waiting for operation completion
- Busy-wait polling