Subject: Storing / Inserting Data per BIT
Posted by Wolfgang on Sun, 22 Apr 2012 14:02:44 GMT
View Forum Message <> Reply to Message

Hi,

got a little question:

I try to get data stored as bit formats, need to store it in this way:

<9 bit empty><10 bit as a number 0-1023><10 bit as a number 0-1023><....>
I need 16 Byte in complete... just told you the first 29 BIT.

But i don't know how to realize that.

Thanks for help!

Subject: Re: Storing / Inserting Data per BIT
Posted by Lance on Sun, 22 Apr 2012 15:02:55 GMT
View Forum Message <> Reply to Message

check bit-wise operation

you will need
bit or |
bit and &
bit negation ~
and bit shift <<, >>

eg, if you want to set 2 most significant bits of a int16 without changing other bits, you can:

```
void test_bit()
{
    uint16 target=0;
    DUMP(target);

    // set 2 most significant bits

    target |= ((uint16)0x3 << 14);
    // 0x3=0000 0000 0000 0011b right shift by 14 bits becomes 1100 0000 0000 0000b
    // or 0xC000
    DUMP(target);

    // and set the 3 LSBs
    target |= 0x7;
    DUMP(target);
    // if you instead want to unset the 2 least significant bits
    // you can do something like:
    target &= ~0x3;
    // 0x3 is 0000 0000 0000 0011b
    // ~0x3 becomes 1111 1111 1111 1100b
    // bitwise-and this number with target result in the
    // 2 LSB being unset.
    DUMP(target)
}
```

Subject: Re: Storing / Inserting Data per BIT
Posted by Lance on Sun, 22 Apr 2012 15:12:35 GMT
View Forum Message <> Reply to Message

Depending on your situation, you may find bit-field more handy in your case:

```
//<9 bit empty><10 bit as a number 0-1023><10 bit as a number 0-1023><....>
//I need 16 Byte in complete... just told you the first 29 BIT.

struct MyData
{
   unsigned dummy:9;
   unsigned number1:10;
   unsigned number2:10;
   //....
};
```

Then you can modify each field as you do a normal variable, and load/store the whole 16 bytes together.

Subject: Re: Storing / Inserting Data per BIT
Posted by Wolfgang on Mon, 23 Apr 2012 19:00:12 GMT
View Forum Message <> Reply to Message

Thank you very much for help, I've done it by bit-wise operations...

just if someone wants to see the code:

```
struct canFrame
{
   unsigned int sAdress:10;
```

```
    unsigned int dAdress:10;
    char buffer[16];

    char* getBuffer() {
 return buffer;
    }
    void setAdress(const int& s, const int& d) {
 sAdress = s;
 dAdress = d;
 buffer[0] = sAdress;
 buffer[1] = (dAdress << 2);
 buffer[1] += sAdress/256;
 buffer[2] = (dAdress >> 6);
    }
    String getAdresses() {
 return (String)AsString(sAdress) + " :: " + AsString(dAdress);
    }
    canFrame() {
 sAdress = 0b0000000000;
 dAdress = 0b0000000000;
 for (int i=0;i<16;i++)
     buffer[i] = 0x0;
    }
};
```

Hope this code is "ok"