Bit Manipulation Cheat Sheet

Note1: "REGX" is not an actual Atmel register, rather "REGX" could be any register, ex. DDRB, PORTC, etc.
Note2: There is nothing special about changing bits 3, 5, or 7, those are simply used as examples here, these numbers could be replaced with any bit in the register

Setting / Clearing / Toggling

```
REGX |= (1 << 3);                    // set 3rd bit
REGX |= (1 << 3) | (1 << 5);         // set 3rd and 5th bits
REGX &= ~(1 << 3);                   // clear 3rd bit
REGX &= ~((1 << 3) | (1 << 5));      // clear 3rd and 5th bits
REGX ^= (1 << 3);                    // toggle 3rd bit
REGX ^= ((1 << 3) | (1 << 5));       // toggle 3rd and 5th bits

// if desired, macros can be used in place of the above, as follows:
#define SET_BIT(byte, bit) (byte |= (1 << bit))
#define CLEAR_BIT(byte, bit) (byte &= ~(1 << bit))
#define TOGGLE_BIT(byte, bit) (byte ^= (1 << bit))

// after using the above macros, we could then do . . .
SET_BIT(REGX, 3);              // set 3rd bit
CLEAR_BIT(REGX, 3);            // clear 3rd bit
TOGGLE_BIT(REGX, 3);           // toggle 3rd bit
```

Conditionals

```
if(REGX & (1 << 3)) {          // if REGX bit 3 is set . . .
if(!(REGX & (1 << 7))) {       // if REGX bit 7 is clear . . .

                  // if REGX bits 3 and 5 are set and REGX bit 7 is clear . . .
if((REGX & (1 << 3)) && (REGX & (1 << 5)) && !(REGX & (1 << 7))) {

// if desired, macros can be used in place of the above, as follows:
#define BIT_IS_SET(byte, bit) (byte & (1 << bit))
#define BIT_IS_CLEAR(byte, bit) (!(byte & (1 << bit)))

// after using the above macros, we could then do . . .
if (BIT_IS_SET(REGX, 3)) {           // if REGX bit 3 is set . . .
if (BIT_IS_CLEAR(REGX, 7)) {         // if REGX bit 7 is clear . . .

                  // if REGX bits 3 and 5 are set and REGX bit 7 is clear . . .
if (BIT_IS_SET(REGX, 3) && BIT_IS_SET(REGX, 5) && BIT_IS_CLEAR(REGX, 7)) {
```