C Programming Cheatsheet for Microcontrollers

To set bit *B* in a variable (or register):

variable |= (1<<*B*); or variable = variable | (1<<*B*);

Example, set Bit 5 in PORTA:

PORTA |= (1<<5);

Explanation:
The << operator shifts a value to the left by the specified amount. That is, 1<<1 becomes 0b10, and 1<<7 becomes 0b10000000. The | operator is a bitwise OR of two variables. For example, variable1 | variable 2 looks like this:

	variable1	0b1100	So,		Variable1	0b1100
OR	variable2	0b0110		OR	(1<<1)	0b0010
		---				---
Result:		0b1110		Result:		0b1110

This results in the bit *B* being set without modifying the other bits.

To clear bit *B* in a variable (or register):

variable &= ~(1<<*B*); or variable = variable & ~(1<<*B*);

Example, clear Bit 6 in TRISA:

TRISA &= ~(1<<6);

Explanation:
The << operate shifts a value to the left by the specified amount. That is, 1<<1 becomes 0b10, and 1<<7 becomes 0b10000000. The ~ operator inverts all bits in a variable. That is, ~(0b1100) becomes (0b0011). The & operator is a bitwise AND of two variables. For example, variable1 & variable 2 looks like this:

	variable1	0b1100	So,		Variable1	0b1110	0b1110
AND	variable2	0b0110		AND	~(1<<1)	~(0b0010)	0b1101
		---					---
Result:		0b0100			Result:		0b1100

This results in the bit being cleared in the original variable without modifying the other bits.

Configuring Inputs and Outputs:

1. First, properly set the tristate or the data direction register for either input or output. Set TRISx bit B to a '1' for input or '0' for output
2. For outputs, initialize the output to some value. Set LATx bit B to a '1' to output VDD or a '0' to output GND.
3. For inputs, set ANSELx bit B to a '0' for digital input or a '1' for analog input. They are set to '1' by default, causing all digital reads of the pin to read '0.'

Input pin Example

```
TRISA |= 1<<5;
ANSELA &= ~(1<<5);
```

Output Pin Example

```
TRISA &= ~(1<<5);
LATA |= 1<<5;
```

Reading Pin Input Status

When a pin is configured as an input, it's current status can be read through the PORTA variable. It is common to need to branch conditionally on the current status of a pin, but this can pose some problems. When accessing the PORTA variable, the status of *all* the pins in the port is read. To correct this, the input has to be masked. Otherwise, you may satisfy an if-statement even when your selected pin is not activated. Here is an example of an if-conditional branch when bit B5 is set.

```
if (PORTB & (1<<5))
{
        //Branch code
}
```

What this does is mask away all the bits that are *not* bit 5. Here's what the code codes

```
      PORTB       0b1110110     0b1100110
AND   (1<<5)      0b0010000     0b0010000
                  -------------  -------------
                  0b0010000     0b0000000
```

As you can see, we now have ONLY the status of pin 5. If this resulting value is nonzero, the pin is reading a '1'. If it is zero, the pin is reading a '0'.