STM32duino GPIO Registers and programming

Sources:

http://embedded-lab.com/blog/stm32-gpio-ports-insights/
http://hertaville.com/stm32f0-gpio-tutorial-part-1.html

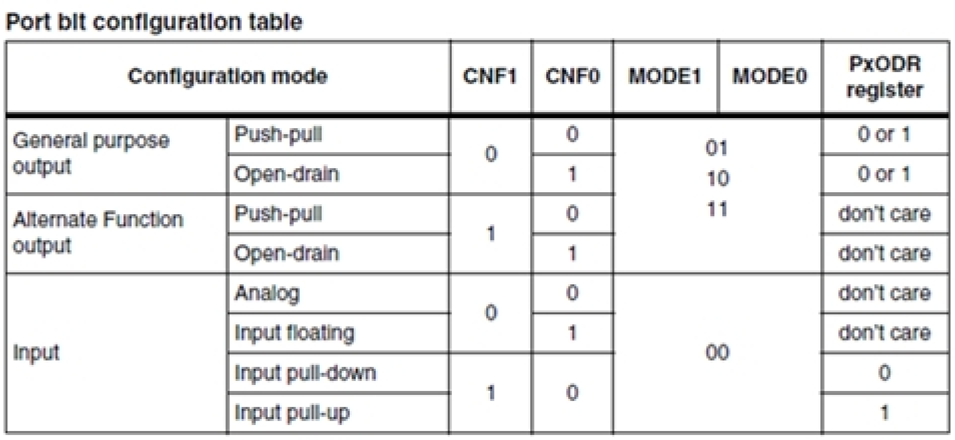
The libmaple libraries, on which STM32duino is based, provides access to registers by the syntax:

```
GPIOA->regs->REG
```

where REG can be one of the following:

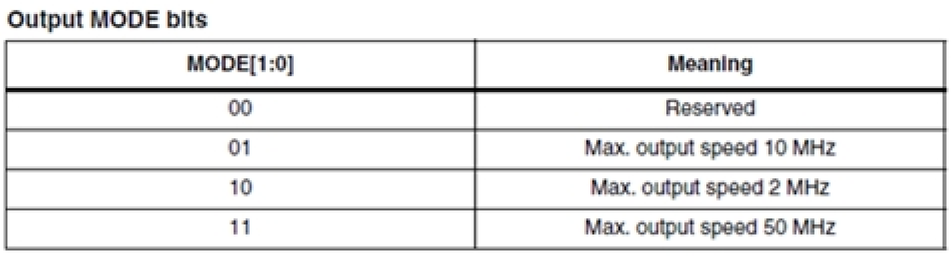
CRH and CRL

CRH is used to set type/and or speed of pins 8-15 of the port **CRL** is used to set type/and or speed of pins 0-7 of the port Accessed as a 32 bit word, with 4 bits representing the state of each pin. Out of these 4 bits, the low 2 bits are MODE, and high 2 bits are CNF.



Port bit configuration table

Configuration mode		CNF1	CNF0	MODE1	MODE0	PxODR register
General purpose output	Push-pull	0	0	01 10 11		0 or 1
	Open-drain		1			0 or 1
Alternate Function output	Push-pull	1	0			don't care
	Open-drain		1			don't care
Input	Analog	0	0	00		don't care
	Input floating		1			don't care
	Input pull-down	1	0			0
	Input pull-up					1



Output MODE bits

MODE[1:0]	Meaning
00	Reserved
01	Max. output speed 10 MHz
10	Max. output speed 2 MHz
11	Max. output speed 50 MHz

The 4 bits for each pin can be set to:
0b0011 (binary) or 0x3 (HEX) - Corresponds to setting pin as output, same as pinMode()
0b1000 or 0x8 - Corresponds to setting pin as input, same as pinMode()

Say I want to set PORTA pins 0, 3 and 4 to OUTPUT and 1, 6, 7 to INPUT, and leave pins 2 and 5 in their original state. The code is:

```
PORTA->regs->CRL = (PORTA->regs->CRL & 0x00F00F00) | 0x88000080 |0x00033003;
//0x00F00F00 is bitmask to retain value of pins 2 and 5 in original state
//0x88000080 is bitmask to set inputs
//0x00033003 is bitmask to set outputs
```

IDR - Input Data Register

Used to read input of entire 16 pins of port at once. Accessed as a 32 bit word whose lower 16 bits represent each pin. The pins being read must be set to INPUT mode by using CRL/CRH or pinMode() before using this.

Say I want to read pins A2. The code is:

```
bool result = GPIOA->regs->IDR & 0x0004; //returns true if A2 is HIGH
```

```
//0x0004 is 0b0000000000000100
```

ODR - Output Data Register

Used to write output to entire 16 pins of port at once. Accessed and written as a 32 bit word whose lower 16 bits represent each pin. The pins being read must be set to OUTPUT mode by using CRL/CRH or pinMode() before using this.

Say I want to set pins A2, A12 and A13, and **reset (clear)** all other pins in the 16 pin bus. The code is:

```
GPIOA->regs->ODR = 0b0011000000000100; //note,  binary
```

Now if I want to set and clear A2, A12 and A13 **without** altering other pins, the code is:

```
//Set A2, A12, A13 (HIGH)
GPIOA->regs->ODR |= 0b0011000000000100;
//Clear A2, A12, A13 (LOW)
GPIOA->regs->ODR &= ~(0b0011000000000100);
```

but notice how, if we want to touch only some pins, we have to READ, MASK and WRITE. That's why there is BRR and BSRR

BRR - Bit Reset Register

32 bit word. Lower 16 bits have 1's where bits are to be set to "LOW". Upper 16 bits have 1's where bits are to be set "HIGH". **0's mean ignore**

Now, to set and clear A2, A12, A13 while preserving the state of all other pins in the port, the code is:

```
//Set A2, A12, A13 (HIGH)
GPIOA->regs->BRR = 0b0011000000000100 << 16; //move to upper 16 bits
//Clear A2, A12, A13 (LOW)
GPIOA->regs->BRR = 0b0011000000000100;
```

BSRR - Bit Set Reset Register

BSRR is like the complement of BRR. It's also a 32 bit word. Lower 16 bits have 1's where bits are to be set to "HIGH". Upper 16 bits have 1's where bits are to be set "LOW". **0's mean ignore**

In this case, to set and clear A2, A12, A13 while preserving the state of all other pins in the port, the code is:

```
//Set A2, A12, A13 (HIGH)
GPIOA->regs->BSRR = 0b0011000000000100;
//Clear A2, A12, A13 (LOW)
GPIOA->regs->BSRR = 0b0011000000000100 << 16; //move to upper 16 bits
```

Combination of BRR and BSRR

Since BRR and BSRR are opposite of each other, you can use both if you don't want to do the bit shift left operation .

In this case, to set and clear A2, A12, A13 while preserving the state of all other pins in the port, the code is:

```
//Set A2, A12, A13 (HIGH)
GPIOA->regs->BSRR = 0b0011000000000100; //lower 16 bits
//Clear A2, A12, A13 (LOW)
GPIOA->regs->BRR = 0b0011000000000100; //lower 16 bits
```