EE 354
Notes GPIO for the ARM Cortex M4 Discovery Board

The STM32F407VG has nine ports which are called Port A through Port I. All of these port pins are accessible only on the chip which has 176 pins. We are using the 100 pin version of the chip only Ports A through Port E are available and each of these has 16 pins labeled Px0 to Px15.

Each port has ten associated registers which configure the port since each pin is capable of multiple functions. The ten registers and their function is shown in Table 1.

<table>
<thead>
<tr>
<th>Register</th>
<th>Abbreviation</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>GPIOA_MODER</td>
<td>Sets pin to input, output, analog, or alternate function.</td>
</tr>
<tr>
<td>Pull up/Pull down</td>
<td>GPIOA_PUPDR</td>
<td>Sets pin to have a pull up, a pull down, or floating.</td>
</tr>
<tr>
<td>Speed</td>
<td>GPIOA_OSPEEDER</td>
<td>If the pin is output it can be set to low, medium, fast, or high speed.</td>
</tr>
<tr>
<td>Output type</td>
<td>GPIOA_OTYPER</td>
<td>If the pin is output, it can be set to push-pull or open drain.</td>
</tr>
<tr>
<td>Input Data Register</td>
<td>GPIOA_IDR</td>
<td>Holds input data</td>
</tr>
<tr>
<td>Output Data Register</td>
<td>GPIOA_ODR</td>
<td>Holds output data</td>
</tr>
<tr>
<td>Output bit set/reset register</td>
<td>GPIOA_BSRR</td>
<td>Writing a one to this register allows individual port bits to be set or reset.</td>
</tr>
<tr>
<td>Alt Function Register low</td>
<td>GPIOA_AFRL</td>
<td>Sets pins for alternate functions such as UART, Timers, CAN, SPI, etc.</td>
</tr>
<tr>
<td>Alt Function Register high</td>
<td>GPIOA_AFRH</td>
<td></td>
</tr>
<tr>
<td>Lock register</td>
<td>GPIOX_LCKR</td>
<td>Allows a bit to be locked – it cannot be changed except by a reset.</td>
</tr>
</tbody>
</table>


All of the ports are synchronous to a clock. This means that the user must first enable to clock for the port before it can be used for input or output. There are three clock registers related to peripherals: RCC_AHB1ENR, RCC_AHB2ENR, and RCCAHB3ENR. All of the GPIO ports are on RCC_AHB1ENR. Bits 0, 1, 2, 3, and 4 enable/disable the clock for GPIO ports A, B, C, D, and E respectively. A 1-bit enable the clock and a 0-bit disables the clock.

The ARM processor does not support the bit data type so there are no instructions that directly operate on bits. You can use the AND and OR operators or you can use the bit set/reset register.

Using AND and OR operators to do bit I/O
Suppose we want to set bit 3 on Port A to a 1. We could do this like this: GPIOA_ODR = GPIOA_ORD | 0x4;
Which can be written as:
```c
GPIOA_ODR |= 0x4;
```
Or it can be written as:
```c
GPIOA_ODR |= (1 << 3);  //<< is the shift operator in C
```

Likewise, we can use the AND operator to clear a bit to zero. To clear bit 3 of Port A to zero we can write any of the following:
```c
GPIOA_ODR = GPIOA_IRD & 0xFFFB;  //0xFFFB = 1111 1111 1111 1011
GPIOA_ODR &= 0xFFFB;
GPIOA_ODR &= ~(1 << 3);
```

**Using the bit set/reset register to do I/O**

Each port has a bit set/reset register named GPIOx_BSRR. Each port is only 16-bits long but the GPIOx_BSRR register is 32 bits long. Placing a zero in the BSRR register has no effect. Placing a 1 in bit-\(n\) in the BSRR register where \(0 \leq n \leq 15\) causes bit-\(n\) in the ODR register to be a 1. Placing a 1 in bit-\(n\) in the BSRR register where \(16 \leq n \leq 31\) causes bit \(n-16\) to be a zero in the ODR register.

Suppose we want to set bit 3 on Port A to a 1. We could do this like this:
```c
GPIOA_BSRR |= (1 << 3);
```
If we want to clear bit 3 on Port A we could do this:
```c
GPIOA_BSRR |= (1 << 19);  //19 = 16 + 3 so bit 3 is cleared.
```
If you simultaneously set a bit to both zero and a one the one will win out.

The program on the next page shows how to do digital I/O.
/*GPIOInOut.c              May 15, 2015
This program toggle pin PA7. It also inputs from pin PB2 and output
to PA9 as fast as possible
*/
#include "stm32f407vg.h"

int main()
{
int i, tmp;
//Clock bits
RCC_AHB1ENR |= 1;         //Bit 0 is GPIOA clock enable bit
RCC_AHB1ENR |= 2;         //Bit 1 is GPIOB clock enable bit
//I/O bits PA7
GPIOA_MODER |= 0x4000;    //Bits 15-14 = 01 for digital output on PA7
//OTYPER register resets to 0 so it is push/pull by default
GPIOA_OSPEEDER |= 0xC000;  //Bits 15-14 = 11 for high speed on PA7
//PUPDR defaults to no pull up no pull down
//I/O bits PA9
GPIOA_MODER |= 0x40000;    //Bits 18-19 = 01 for digital output on PA9
//OTYPER register resets to 0 so it is push/pull by default
GPIOA_OSPEEDER |= 0xC0000;  //Bits 18-19 = 11 for high speed on PA9
//PUPDR defaults to no pull up no pull down
//I/O bits PB2
GPIOB_OSPEEDER |= 0xC;     //PB2 to high speed
//All bits are input by default.
//Main program loop
tmp = 0;
while(1)
{
if(tmp == 0)
    GPIOA_ODR &= ~(1 << 7);  //Toggle PA7 depending on tmp
else
    GPIOA_ODR |= (1 << 7);
    tmp = ~tmp;
//Input bit PB2 and shift it 7 places to output to PA9
if((GPIOB_IDR & 0x4) == 0)
    GPIOA_ODR |= (1 << 9);
else
    GPIOA_ODR &= ~(1 << 9);
    //for(i=0;i<10000;i++);  //Put this in to slow down toggle
}