Microprocessor Systems

ELE 271

Laboratory 6:

Clocks

Due 27MAR2023

By: Nick Bean ,Taylor Brookes, Toussaint Falangani

**Introduction**

In micro processing the driving pace of a system is the system clock, this essentially sets the speed for all functions performed. The objective of this experiment is to manipulate the speed of the system clock to achieve a greater level of control over the cadence of the processor's functions.

**Part one** of the lab focuses on observing the default system clock via GPIO output pin. We will use the PA8 pin with the microcontroller clock output (MCO) alternate function (AF) to observe the system clock via logic analyzer.

**Part two** of the experiment focuses on manipulating the system clock to a cadence other than the default 4MHz. To achieve this goal we will select MSI as the system clock and then change its frequency by setting the MSIRANGE bits in the RCC\_CR register. This will allow for greater control over the system processes in future use of the microprocessor.

**Part 1**

To begin the experiment, we start with the standard pin configuration for PA.8 this time instead of PA.5 for the LED. This means we will be using 8 and 16 for 1 and 2 bit shift locations respectively instead of 5 and 10. The only difference between the clock pin configuration and the LED pin configuration is that we set MODER to Alternate Function (10) instead of output (01). This allows us to change settings to output the clock speed of the processor. We use PA.8 because it contains the microcontroller clock output (MCO) alternate function (AF0), which allows us to read the processor’s current clock speed and display it as an output on the oscilloscope.

[Configuration of Clock Pin Part 1]

void configure\_Clock\_pin(){

// Enable the clock to GPIO Port A

RCC -> AHB2ENR |= RCC\_AHB2ENR\_GPIOAEN;

RCC -> AHB2ENR |= 0x00000001;

// output

GPIOA -> MODER &= ~(3UL<<16);

GPIOA -> MODER |= 2UL<<16; // 01 general output , 10 alternate function

GPIOA -> OTYPER &= ~(1UL<<8);

GPIOA -> OSPEEDR &= ~(3UL<<16);

GPIOA -> PUPDR &= ~(3<<16);

For the second half of the clock configuration method, we introduce new instructions. First, we configure the AFR (alternate function register). The AFR is configured as either AFR low (AFR[0]) or AFR high (AFR[1]). AFR[0] deals with pins 0-7, while AFR[1] deals with pins 8-15. Because we are using pin PA.8, we will utilize AFR[1]. AFR controls 3 bits, so we take the complement with bitwise AND operator of 111 to give us 000. We are addressing AF[0] which starts on bit 0. AF0 allows us to use the MCO. Our second step is to select the MSI clock as our system clock. We access bits 24-27 to do so, and set the value to 0010 after setting all the bits to 0000. The MSI clock signal is the internal RC oscillator’s frequency, which can be changed by changing the MSI range in Part 3 of our clock pin configuration method. We noticed while writing our report that we failed to initially reset the values for the CFGR and AFR, but didn’t notice any odd results occurring from not doing this.

[Configuration of Clock Pin Part 2]

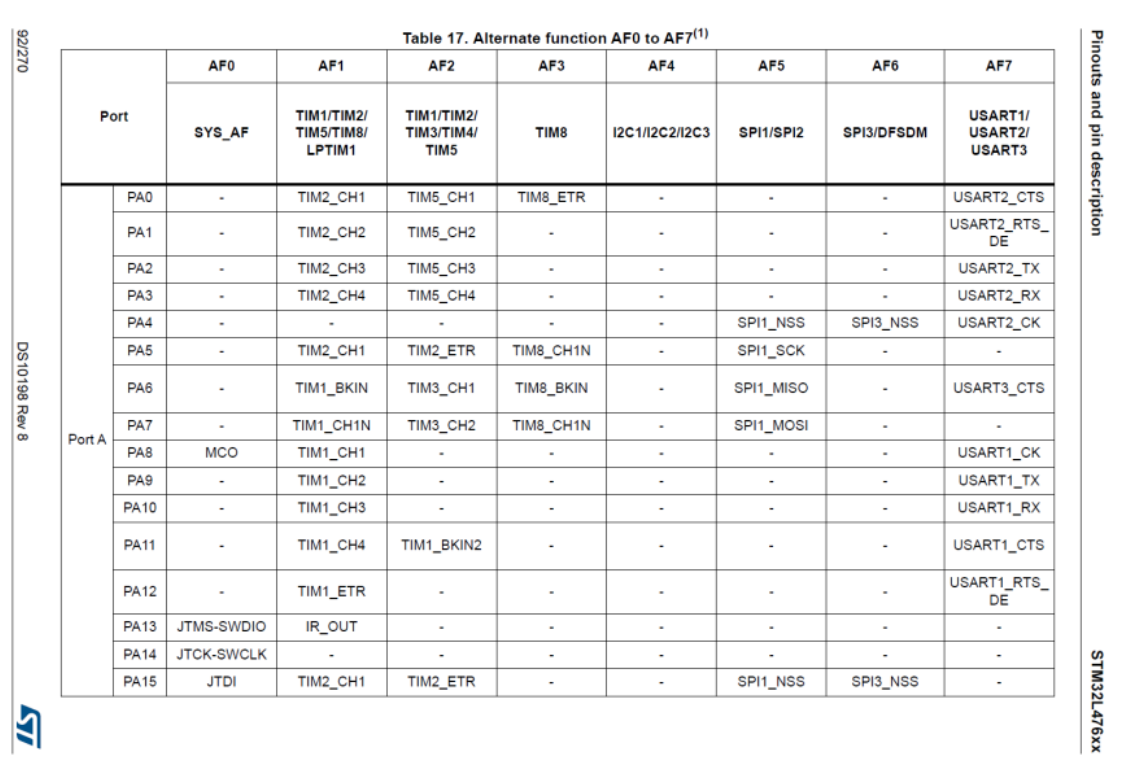
GPIOA -> AFR[1] &= ~(7<<0); // fill value with 0000 (for AF0) (pg311/1903)

RCC -> CFGR &= ~(7<<24);

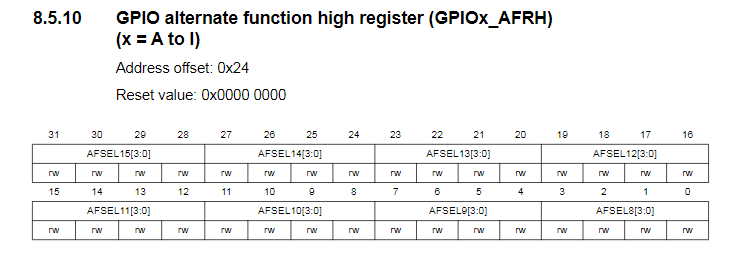
RCC -> CFGR |= (2<<24); // MCOSEL[2:0] = 0010 // MSI clock selected (pg227/1903)

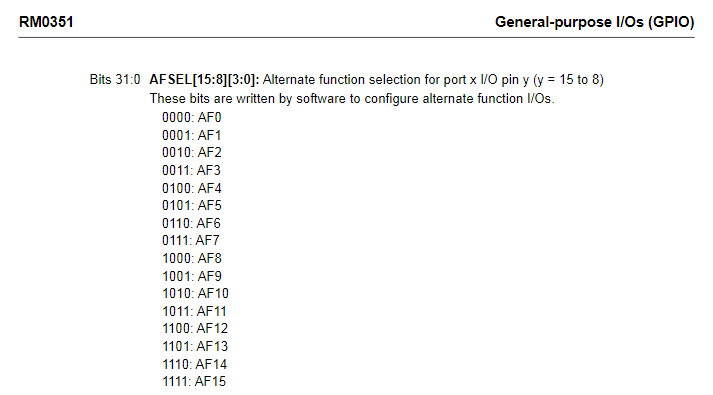
}

[PA8 AF0 usage]

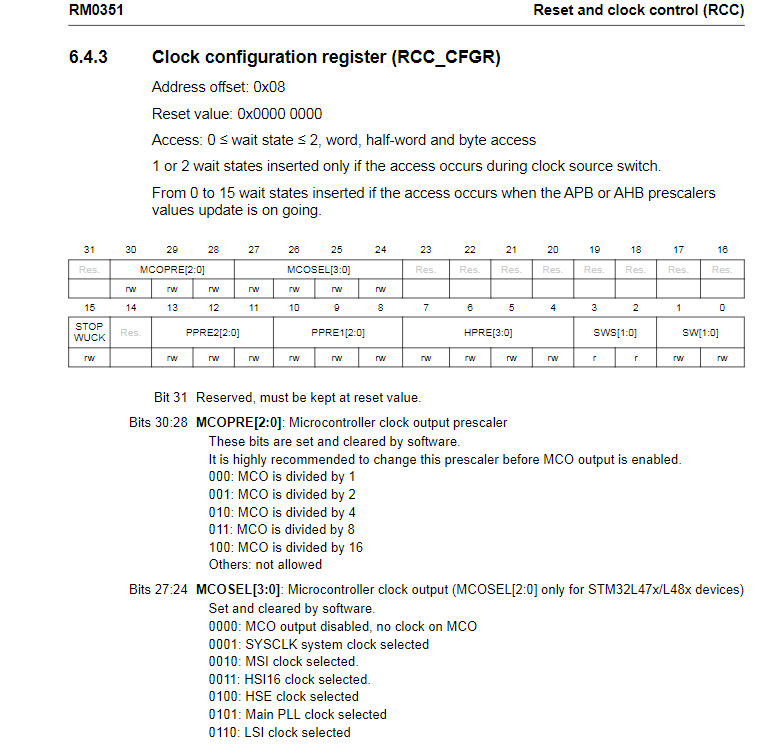


[PA.8 AF set to AF0]





[MCOSEL set to MSI Clock]



Our main method remains pretty bare, consisting of just calling the configuration function and then creating a dead while loop so we can continuously monitor the clock speed of the processor to allow us to get a continuous output on the oscilloscope.

[Configuration of Main Method]

int main(void){

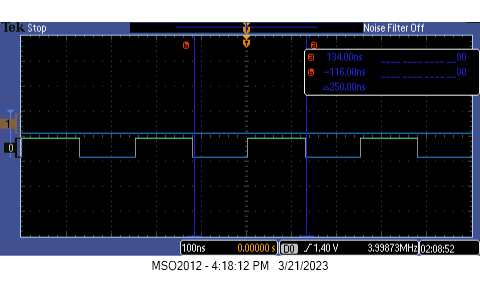
configure\_Clock\_pin();

while(1);

}

After building and loading our main.c file, we received the desired 4MHz system clock frequency of our microprocessor, as output below.

[4MHz frequency Oscilloscope display]



**Part 2**  The second part of the experiment is for us to switch between different clock frequencies. First, we reset all the CR values by setting it to the reset value. We then proceeded by selecting MSI as the system clock and then changed its frequency by setting the MSIRANGE bits [bits 4 to 7] in the RCC\_CR register. To change the frequency of the clock we must first ensure that the MSION [bit 0] is set to OFF [0], and that the MSIRDY [bit1] is set to a ready state [1], and that the MSIRGSEL [bit 3] is set to [1] so that the timing data will be provided by MSIRANGE(3:0). With these previous functions, the value of the MSIRANGE (3:0) can be manipulated to achieve a variety of frequencies from 100KHz to 48MHz.

[Configuration of Clock Pin Part 3]

RCC -> CR = 0x00000063; // reset value for CR (223/1903)

RCC -> CR &= (0<<0); // MSION is OFF, bit 0 (226/1903)

RCC -> CR |= (1<<1); // MSI is ready, bit 1 (225/1903)

RCC -> CR |= (1<<3); // bit 3, set to 1: MSIRGSEL, MSI Range is provided by MSIRANGE[3:0] in the RCC\_CR register (225/1903)

RCC -> CR &= ~(15<<4); // MSIRANGE, bits 4 - 7, range 0 around 100kHz when MSIRGSEL is set (225/1903)

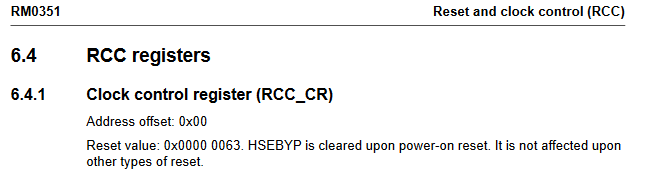
// and MSION is OFF and/or MSI is ready

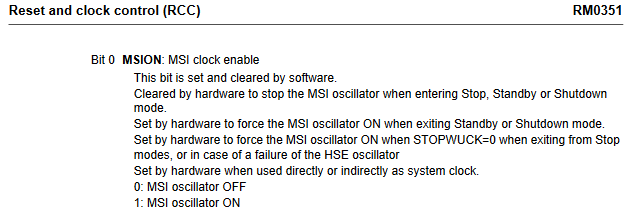
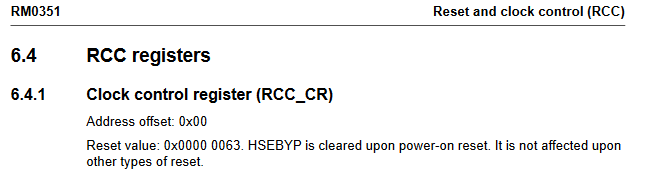
// lowest frequency at 0

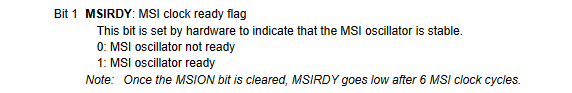
RCC -> CR |= (11<<4);

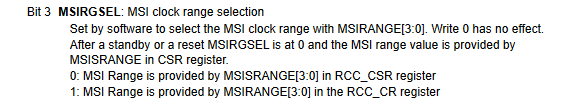
// highest frequency at 11, need to do autoset for it to show, with no single

[CR Bit Configuration]

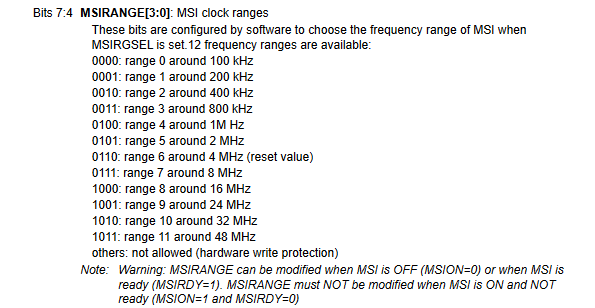






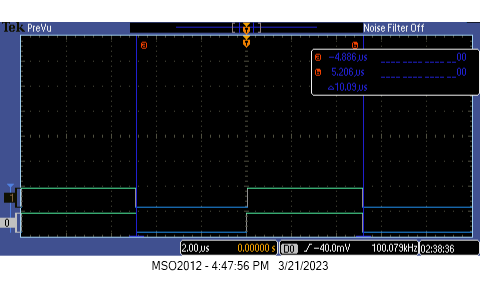


**[ Documentation for MSIRANGE Values]**

\

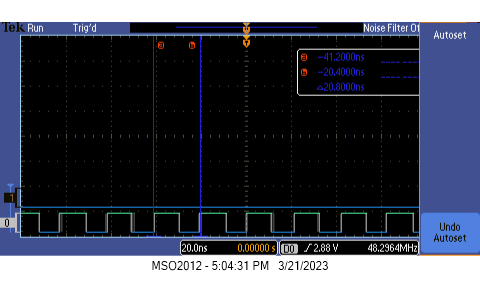
To achieve the lowest available modified system clock value, we set the MSIRANGE values to [0000] with the complement function and bitwise AND operator applied to the value 15. This method was successful in manipulating the clock to a frequency of Aprox. 100KHz.

[MSIRANGE[3:0] to 100kHz, Lowest Frequency]



Similar to the previous method used to slow the system clock frequency, we manipulated MSIRANGE to a value of [1011] to select the highest preset frequency for the system clock, achieving a frequency of Approx. 48MHz.

[MSIRANGE[3:0] to 48MHz, Highest Frequency]



**Conclusion**

Our goal in this lab was to maneuver the speed of the system clock in order to get a higher level of control over the rhythm of the functions of the processor. We first began taking notice of the default system clock(MSI) through the GPIO output pin and displaying it on the oscilloscope. We made use of the PA8 pin with the output clock (MCO) alternate function in order to have an observation of the system clock through the oscilloscope. We were able to observe from the oscilloscope a 4MHz system clock frequency of our microprocessor as an output. Moving forward in the experiment, we had the goal to switch between different frequencies of the clock, different from the obtained results above. We used MSI as the system clock and modified its frequencies by MSIRANGE bits in the RCC\_CR register. We were able to capture the lowest and highest frequencies, along with the system clock with the oscilloscope.