Kyle Rosenthal

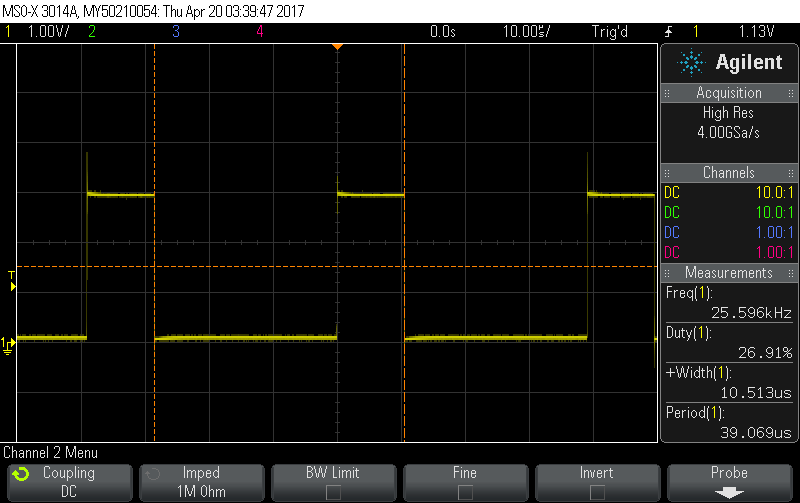
Assignment 4

# Step 1

Observed blinking LED.

# Step 2

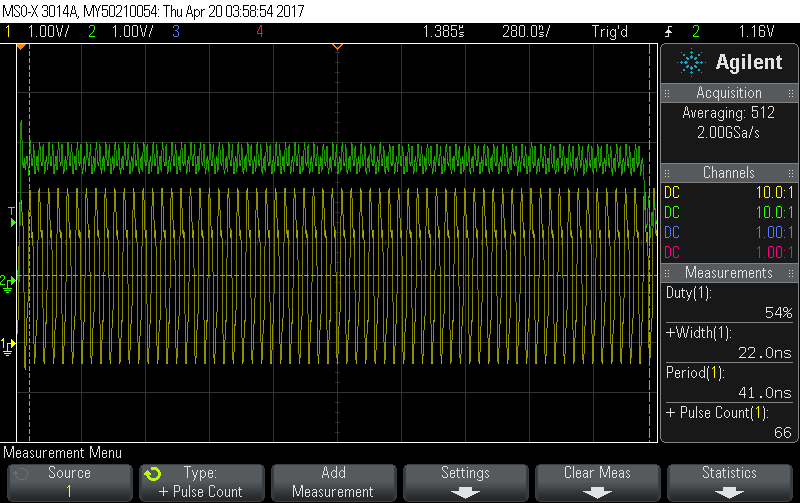
Period = 1 / 25000; CCR[0] = (1 / 25k) / (1 / 24M) / 4 then tweaked a bit to make it closer to perfect



# Step 3

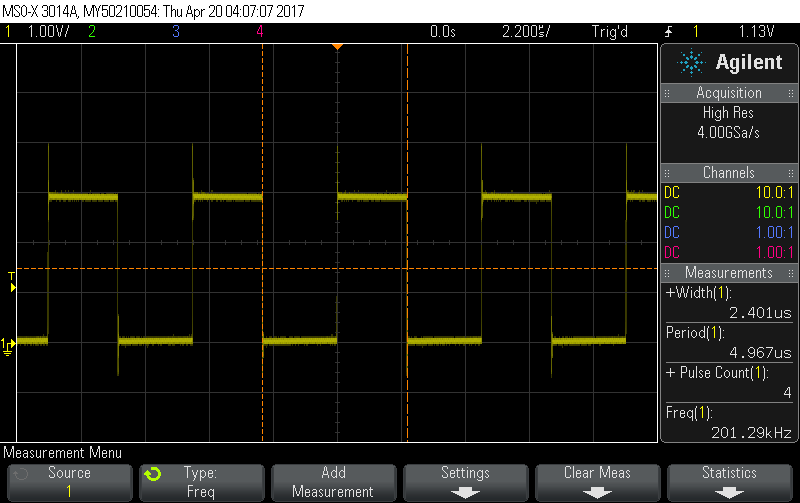
Double CCRO and toggle bit every time to raise duty cycle to 50%.

# Step 4



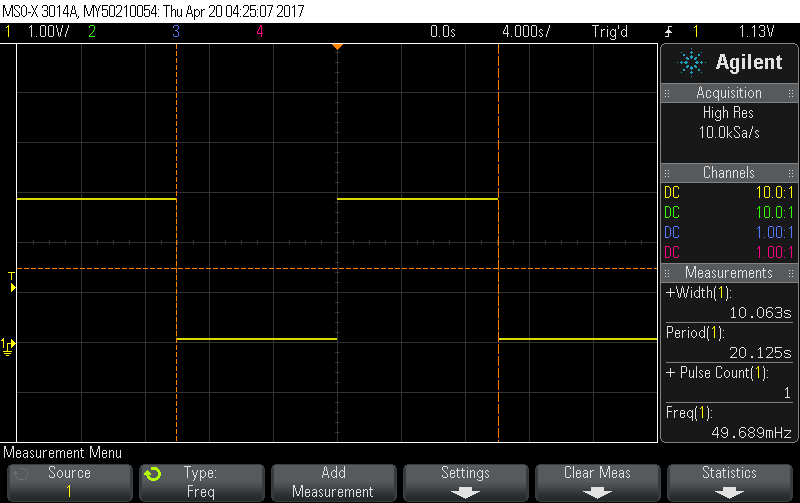
66 cycles for the interrupt.

# Step 5

Lowest value was 60, which is in line with the cycle count from step 4. 

# Step 6

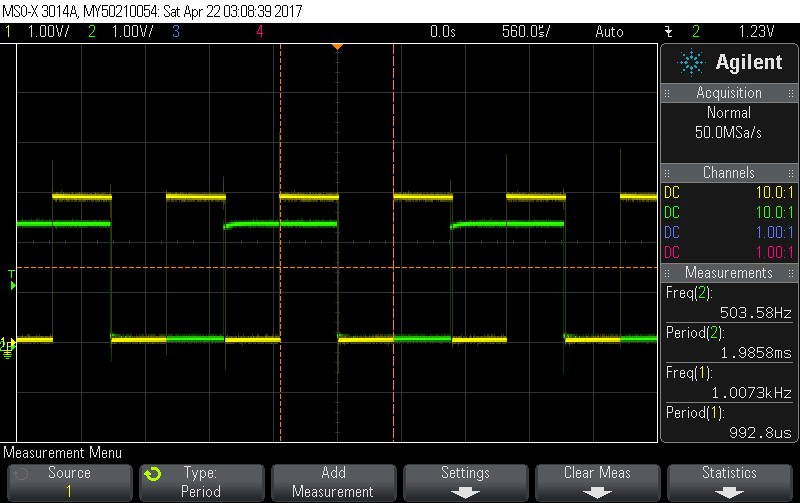
1.5Hz with CCRO of 0xFFFF gets a 86ms period, to get that to 20 seconds we need to only toggle the bit every 465 times (20 secs/ 86 ms).



# Step 7

Created two interrupts with one toggling both output bits, and one only toggling one of them.

# Step 8



The waveforms effectively form a 2-bit counter.

# Youtube Link

## 20 second clock

<https://www.youtube.com/watch?v=UT6FJ-BboNk>

## Reflex

<https://www.youtube.com/watch?v=JcMbWuh2dOM>

# Clock Calculation

# Code

## 25KHz main.c

**#include** "msp.h"

**#define** FREQ\_24\_MHz 24000000

**int** **main**(**void**) {

WDT\_A->CTL = WDT\_A\_CTL\_PW | // Stop WDT

WDT\_A\_CTL\_HOLD;

CS->KEY = CS\_KEY\_VAL; // unlock CS registers

CS->CTL0 = 0; // clear register CTL0

CS->CTL0 = CS\_CTL0\_DCORSEL\_4; //set 24 MHz

CS->CTL1 = CS\_CTL1\_SELA\_2 | CS\_CTL1\_SELS\_3 | CS\_CTL1\_SELM\_3; // select clock sources

CS->KEY = 0; // lock the CS registers

// Configure GPIO

P1->DIR |= BIT0;

P1->OUT |= BIT0;

TIMER\_A0->CCTL[0] = TIMER\_A\_CCTLN\_CCIE; // TACCR0 interrupt enabled

TIMER\_A0->CCR[0] = 55;

TIMER\_A0->CTL = TIMER\_A\_CTL\_SSEL\_\_SMCLK | // SMCLK, continuous mode

TIMER\_A\_CTL\_MC\_\_CONTINUOUS |

TIMER\_A\_CTL\_ID\_\_1;

SCB->SCR |= SCB\_SCR\_SLEEPONEXIT\_Msk; // Enable sleep on exit from ISR

// Enable global interrupt

\_\_enable\_irq();

NVIC->ISER[0] = 1 << ((*TA0\_0\_IRQn*) & 31);

//MCLK out on pin 4.3

P4->SEL0 |= BIT3;

P4->DIR |= BIT3;

P6->DIR |= BIT0;

**while** (1)

{

\_\_sleep();

\_\_no\_operation(); // For debugger

}

}

// Timer A0 interrupt service routine

// Step 2 CCRO = 172 \* 2, 64 \*2

// STEP 4 CCRO = 256

// step 5 ccro = 60

**void** **TA0\_0\_IRQHandler**(**void**) {

//P6->OUT |= BIT0;

TIMER\_A0->CCTL[0] &= ~TIMER\_A\_CCTLN\_CCIFG;

**if** (P1->OUT & BIT0) {

TIMER\_A0->CCR[0] += 256;

P1->OUT &= ~BIT0;

} **else** {

TIMER\_A0->CCR[0] += 256;

P1->OUT |= BIT0;

}

//P6->OUT &= ~BIT0;

}

## 20-sec period main.c

**#include** "msp.h"

**#define** FREQ\_24\_MHz 24000000

**int** **main**(**void**) {

WDT\_A->CTL = WDT\_A\_CTL\_PW | // Stop WDT

WDT\_A\_CTL\_HOLD;

CS->KEY = CS\_KEY\_VAL; // unlock CS registers

CS->CTL0 = 0; // clear register CTL0

CS->CTL0 = CS\_CTL0\_DCORSEL\_0; //set 1.5 MHz

CS->CTL1 = CS\_CTL1\_SELA\_2 | CS\_CTL1\_SELS\_3 | CS\_CTL1\_SELM\_3; // select clock sources

CS->KEY = 0; // lock the CS registers

// Configure GPIO

P1->DIR |= BIT0;

P1->OUT |= BIT0;

TIMER\_A0->CCTL[0] = TIMER\_A\_CCTLN\_CCIE; // TACCR0 interrupt enabled

TIMER\_A0->CCR[0] = 0xFFFF;

TIMER\_A0->CTL = TIMER\_A\_CTL\_SSEL\_\_SMCLK | // SMCLK, continuous mode

TIMER\_A\_CTL\_MC\_\_CONTINUOUS |

TIMER\_A\_CTL\_ID\_\_1;

SCB->SCR |= SCB\_SCR\_SLEEPONEXIT\_Msk; // Enable sleep on exit from ISR

// Enable global interrupt

\_\_enable\_irq();

NVIC->ISER[0] = 1 << ((*TA0\_0\_IRQn*) & 31);

//MCLK out on pin 4.3

P4->SEL0 |= BIT3;

P4->DIR |= BIT3;

P6->DIR |= BIT0;

**while** (1)

{

\_\_sleep();

\_\_no\_operation(); // For debugger

}

}

// Timer A0 interrupt service routine

// Step 2 CCRO = 172 \* 2, 64 \*2

// STEP 4 CCRO = 256

// step 5 ccro = 60

**void** **TA0\_0\_IRQHandler**(**void**) {

**static** **int** x = 0;

x++;

//P6->OUT |= BIT0;

TIMER\_A0->CCTL[0] &= ~TIMER\_A\_CCTLN\_CCIFG;

**if** (x >= 232 && P1->OUT & BIT0) {

//TIMER\_A0->CCR[0] += 256;

P1->OUT &= ~BIT0;

x = 0;

} **else** **if** (x >= 232) {

//TIMER\_A0->CCR[0] += 256;

P1->OUT |= BIT0;

x = 0;

}

//P6->OUT &= ~BIT0;

}

## 2-bit counter main.c

**#include** "msp.h"

**#define** FREQ\_24\_MHz 24000000

**int** **main**(**void**) {

WDT\_A->CTL = WDT\_A\_CTL\_PW | // Stop WDT

WDT\_A\_CTL\_HOLD;

CS->KEY = CS\_KEY\_VAL; // unlock CS registers

CS->CTL0 = 0; // clear register CTL0

CS->CTL0 = CS\_CTL0\_DCORSEL\_0; //set 1.5 MHz

CS->CTL1 = CS\_CTL1\_SELA\_2 | CS\_CTL1\_SELS\_3 | CS\_CTL1\_SELM\_3; // select clock sources

CS->KEY = 0; // lock the CS registers

// Configure GPIO

P1->DIR |= BIT0;

P1->OUT |= BIT0;

TIMER\_A0->CCTL[0] = TIMER\_A\_CCTLN\_CCIE; // TACCR0 interrupt enabled

TIMER\_A0->CCR[0] = 1500;

TIMER\_A0->CCTL[1] = TIMER\_A\_CCTLN\_CCIE; // TACCR1 interrupt enabled

TIMER\_A0->CCR[1] = 750;

TIMER\_A0->CTL = TIMER\_A\_CTL\_SSEL\_\_SMCLK | // SMCLK, continuous mode

TIMER\_A\_CTL\_MC\_\_CONTINUOUS |

TIMER\_A\_CTL\_ID\_\_1;

SCB->SCR |= SCB\_SCR\_SLEEPONEXIT\_Msk; // Enable sleep on exit from ISR

// Enable global interrupt

\_\_enable\_irq();

NVIC->ISER[0] = 1 << ((*TA0\_N\_IRQn*) & 31);

NVIC->ISER[0] = 1 << ((*TA0\_0\_IRQn*) & 31);

//MCLK out on pin 4.3

//P4->SEL0 |= BIT3;

//P4->DIR |= BIT3;

P6->DIR |= BIT0;

P6->OUT |= BIT0;

**while** (1)

{

\_\_sleep();

\_\_no\_operation(); // For debugger

}

}

// Timer A0 interrupt service routine

// Step 2 CCRO = 172 \* 2, 64 \*2

// STEP 4 CCRO = 256

// step 5 ccro = 60

**void** **TA0\_0\_IRQHandler**(**void**) {

//P6->OUT |= BIT0;

TIMER\_A0->CCTL[0] &= ~TIMER\_A\_CCTLN\_CCIFG;

**if** (P1->OUT & BIT0) {

TIMER\_A0->CCR[0] += 1500;

P1->OUT &= ~BIT0;

// x = 0;

} **else** {

TIMER\_A0->CCR[0] += 1500;

P1->OUT |= BIT0;

// x = 0;

}

}

**void** **TA0\_N\_IRQHandler**(**void**) {

**if**(TIMER\_A0->CCTL[1]&TIMER\_A\_CCTLN\_CCIFG)

{

TIMER\_A0->CCTL[1] &= ~TIMER\_A\_CCTLN\_CCIFG;

**if** (P6->OUT & BIT0) {

TIMER\_A0->CCR[1] += 1500;

P6->OUT &= ~BIT0;

// x = 0;

} **else** {

TIMER\_A0->CCR[1] += 1500;

P6->OUT |= BIT0;

// x = 0;

}

**if** (P1->OUT & BIT0) {

P1->OUT &= ~BIT0;

// x = 0;

} **else** {

P1->OUT |= BIT0;

// x = 0;

}

}

//P6->OUT &= ~BIT0;

}