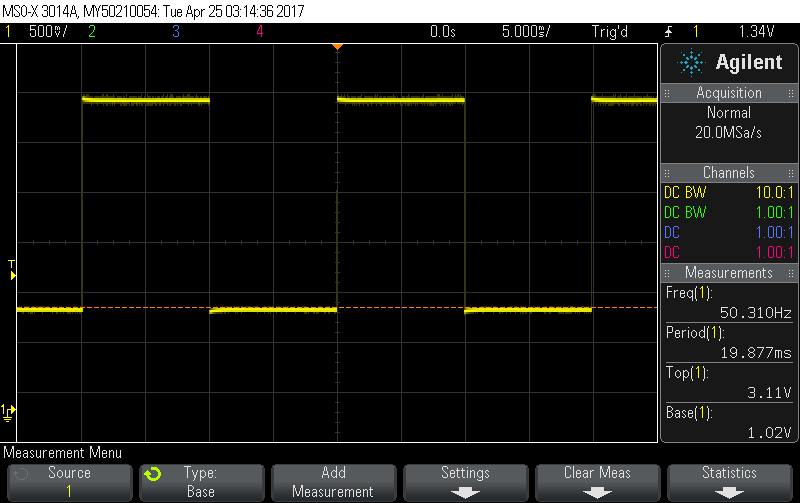
Kyle Rosenthal

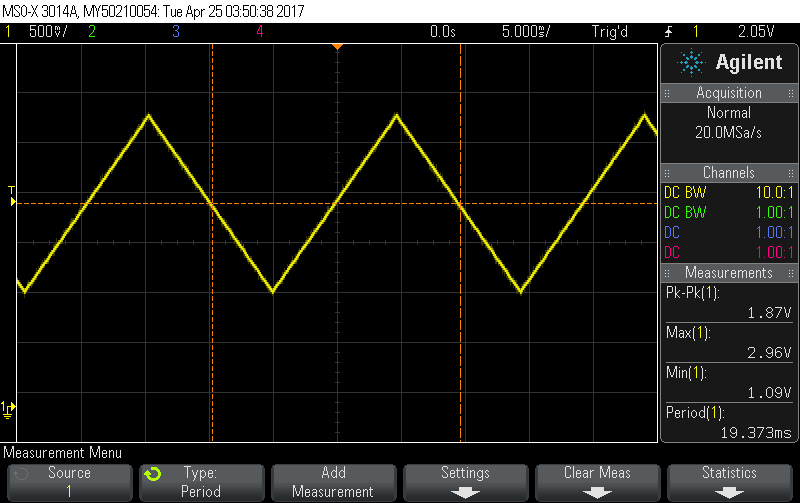
Assignment 5

# Square Wave – 2Vpp 1V offset



This was formed with an interrupt that switched the DAC between the two values.

# Square Wave – 2Vpp 1V offset



This was formed by an interrupt moving the DAC output up and down at each interrupt.

# Youtube link

<https://www.youtube.com/watch?v=ru4SxrP7f8k>

# Code

## main.c for square

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// DriveDAC.C

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**#include** "msp.h"

**#include** <stdint.h>

**void** **Drive\_DAC**(**unsigned** **int** level);

**volatile** **unsigned** **int** TempDAC\_Value = 0;

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

WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD; // Stop watchdog timer

// DCO = 24 MHz, SMCLK and MCLK = DCO

CS->KEY = CS\_KEY\_VAL;

CS->CTL0 = 0;

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

CS->CTL1 = CS\_CTL1\_SELA\_2 | CS\_CTL1\_SELS\_3 | CS\_CTL1\_SELM\_3;

CS->KEY = 0;

// Configure port bits for SPI

P4->DIR |= BIT1; // Will use BIT4 to activate /CE on the DAC

P1SEL0 |= BIT6 + BIT5; // Configure P1.6 and P1.5 for UCB0SIMO and UCB0CLK

P1SEL1 &= ~(BIT6 + BIT5); //

// SPI Setup

EUSCI\_B0->CTLW0 |= EUSCI\_B\_CTLW0\_SWRST; // Put eUSCI state machine in reset

EUSCI\_B0->CTLW0 = EUSCI\_B\_CTLW0\_SWRST | // Remain eUSCI state machine in reset

EUSCI\_B\_CTLW0\_MST | // Set as SPI master

EUSCI\_B\_CTLW0\_SYNC | // Set as synchronous mode

EUSCI\_B\_CTLW0\_CKPL | // Set clock polarity high

EUSCI\_B\_CTLW0\_MSB; // MSB first

EUSCI\_B0->CTLW0 |= EUSCI\_B\_CTLW0\_SSEL\_\_SMCLK; // SMCLK

EUSCI\_B0->BRW = 0x01; // divide by 16, clock = fBRCLK/(UCBRx)

EUSCI\_B0->CTLW0 &= ~EUSCI\_B\_CTLW0\_SWRST; // Initialize USCI state machine, SPI

// now waiting for something to

// be placed in TXBUF

EUSCI\_B0->IFG |= EUSCI\_B\_IFG\_TXIFG; // Clear TXIFG flag

//setup interupt

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

TIMER\_A0->CCR[0] = 3000;

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

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

TIMER\_A\_CTL\_ID\_\_8;

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);

**while** (1) {

\_\_sleep();

\_\_no\_operation(); // increment by 100 mv

}

} // end of main

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

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

TIMER\_A0->CCR[0] += 0x20;

**static** **unsigned** **int** i = 0x0280;

Drive\_DAC(i);

i = (i == 0x0280) ? 0x0EFF : 0x0280;

}

**void** **Drive\_DAC**(**unsigned** **int** level){

**unsigned** **int** DAC\_Word = 0;

**int** i;

DAC\_Word = (0x1000) | (level & 0x0FFF); // 0x1000 sets DAC for Write

// to DAC, Gain = 2, /SHDN = 1

// and put 12-bit level value

// in low 12 bits.

P4->OUT &= ~BIT1; // Clear P4.1 (drive /CS low on DAC)

// Using a port output to do this for now

EUSCI\_B0->TXBUF = (**unsigned** **char**) (DAC\_Word >> 8); // Shift upper byte of DAC\_Word

// 8-bits to right

**while** (!(EUSCI\_B0->IFG & EUSCI\_B\_IFG\_TXIFG)); // USCI\_A0 TX buffer ready?

EUSCI\_B0->TXBUF = (**unsigned** **char**) (DAC\_Word & 0x00FF); // Transmit lower byte to DAC

**while** (!(EUSCI\_B0->IFG & EUSCI\_B\_IFG\_TXIFG));

**for**(i = 200; i > 0; i--); // Delay 200 16 MHz SMCLK periods

//to ensure TX is complete by SIMO

P4->OUT |= BIT1; // Set P4.1 (drive /CS high on DAC)

**return**;

}

## main.c for triangle

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// DriveDAC.C

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**#include** "msp.h"

**#include** <stdint.h>

**void** **Drive\_DAC**(**unsigned** **int** level);

**volatile** **unsigned** **int** TempDAC\_Value = 0;

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

WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD; // Stop watchdog timer

// DCO = 24 MHz, SMCLK and MCLK = DCO

CS->KEY = CS\_KEY\_VAL;

CS->CTL0 = 0;

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

CS->CTL1 = CS\_CTL1\_SELA\_2 | CS\_CTL1\_SELS\_3 | CS\_CTL1\_SELM\_3;

CS->KEY = 0;

// Configure port bits for SPI

P4->DIR |= BIT1; // Will use BIT4 to activate /CE on the DAC

P1SEL0 |= BIT6 + BIT5; // Configure P1.6 and P1.5 for UCB0SIMO and UCB0CLK

P1SEL1 &= ~(BIT6 + BIT5); //

// SPI Setup

EUSCI\_B0->CTLW0 |= EUSCI\_B\_CTLW0\_SWRST; // Put eUSCI state machine in reset

EUSCI\_B0->CTLW0 = EUSCI\_B\_CTLW0\_SWRST | // Remain eUSCI state machine in reset

EUSCI\_B\_CTLW0\_MST | // Set as SPI master

EUSCI\_B\_CTLW0\_SYNC | // Set as synchronous mode

EUSCI\_B\_CTLW0\_CKPL | // Set clock polarity high

EUSCI\_B\_CTLW0\_MSB; // MSB first

EUSCI\_B0->CTLW0 |= EUSCI\_B\_CTLW0\_SSEL\_\_SMCLK; // SMCLK

EUSCI\_B0->BRW = 0x01; // divide by 16, clock = fBRCLK/(UCBRx)

EUSCI\_B0->CTLW0 &= ~EUSCI\_B\_CTLW0\_SWRST; // Initialize USCI state machine, SPI

// now waiting for something to

// be placed in TXBUF

EUSCI\_B0->IFG |= EUSCI\_B\_IFG\_TXIFG; // Clear TXIFG flag

//setup interupt

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

TIMER\_A0->CCR[0] = 500;

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

TIMER\_A\_CTL\_MC\_\_UP |

TIMER\_A\_CTL\_ID\_\_4;

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);

P6->DIR |= BIT0;

**while** (1) {

\_\_sleep();

\_\_no\_operation(); // increment by 100 mv

}

} // end of main

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

P6->OUT |= BIT0;

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

//TIMER\_A0->CCR[0] += 200;

**int** x = TIMER\_A0->CCR[0];

**static** **unsigned** **int** i = 0x02F1;

**static** **int** dir = 11;

Drive\_DAC(i);

**if** (i <= 0x02F0 || i >= 0x077F) dir = -dir;

i += dir;

P6->OUT &= ~BIT0;

}

**void** **Drive\_DAC**(**unsigned** **int** level){

**unsigned** **int** DAC\_Word = 0;

**int** i;

DAC\_Word = (0x1000) | (level & 0x0FFF); // 0x1000 sets DAC for Write

// to DAC, Gain = 2, /SHDN = 1

// and put 12-bit level value

// in low 12 bits.

P4->OUT &= ~BIT1; // Clear P4.1 (drive /CS low on DAC)

// Using a port output to do this for now

EUSCI\_B0->TXBUF = (**unsigned** **char**) (DAC\_Word >> 8); // Shift upper byte of DAC\_Word

// 8-bits to right

**while** (!(EUSCI\_B0->IFG & EUSCI\_B\_IFG\_TXIFG)); // USCI\_A0 TX buffer ready?

EUSCI\_B0->TXBUF = (**unsigned** **char**) (DAC\_Word & 0x00FF); // Transmit lower byte to DAC

**while** (!(EUSCI\_B0->IFG & EUSCI\_B\_IFG\_TXIFG));

**for**(i = 200; i > 0; i--); // Delay 200 16 MHz SMCLK periods

//to ensure TX is complete by SIMO

P4->OUT |= BIT1; // Set P4.1 (drive /CS high on DAC)

**return**;

}