A2 - Blinking LED, Clock Control, an Software Delay on MSP432

Students: Zach Bunce & Garrett Maxon
Class: CPE329-03
Professor: Gerfen, Jeffrey

Youtube video Demonstration of one-second pulse using three system clock frequencies:
https://youtu.be/FsSP_aTBuSc

Documentation of accuracy of the one-second pulses with three different clock frequencies
Table 1: Data taken from oscilloscope showing edges of pulses

Time (s)	Volts (V)
-2	0
-1.93502	0
-1.93502	3.3
-0.9311	3.3
-0.9311	0
-0.67263	0
-0.67262	3.3
0.331565	3.3
0.331568	0
0.590131	0
0.590134	3.3
1.59353	3.3
1.593533	0
1.851866	0
1.851869	3.3
1.98424	3.3
1.98425	0
2	0

First Pulse at 1.5 MHz
(-0.9311)- (-1.93502) = 1.00392
$\% \, error = \frac{Experimental - Actual}{Actual} x100 = \frac{1.00392 - 1}{1} x100 = .392\%$

Second Pulse at 6 MHz
(0.331565) - (-0.6762)= 1.00419 $\% \, error = \frac{Experimental - Actual}{Actual} x100 = \frac{1.00419 - 1}{1} x100 = .419\%$

Third Pulse at 24 MHz
(1.593533)- (.590134) = 1.0034 $\% \, error = \frac{Experimental - Actual}{Actual} x100 = \frac{1.0034 - 1}{1} x100 = .34\%$

C Code Used for 1s Three Pulse Test

```
int main(void)
{
    WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;    // Stop WDT

    // P1.0 set as GPIO
    P1->SEL0 &= ~BIT0;              //Clear bit 0 of the P1->SEL0 register
    P1->SEL1 &= ~BIT0;              //Clear bit 0 of the P1->SEL1 register

    P1->DIR |= BIT0;                //P1.0 set as output

    //Creates three 1000 ms pulses using 1.5 MHz, 6 MHz, and 24 MHz
    while (1)                       //Continuous Loop
    {
        set_DCO(F_1p5_MeHz);        //Sets the DCO to 1.5 MHz
        P1->OUT |= BIT0;            //Turn on P1.0 LED
        delay_ms(1000, F_1p5_MeHz); //Waits for 1000 ms
        P1->OUT &= ~BIT0;            //Turn off P1.0 LED
        delay_ms(250, F_1p5_MeHz);  //Wait a half pulse

        set_DCO(F_6_MeHz);          //Sets the DCO to 1.5 MHz
        P1->OUT |= BIT0;            //Turn on P1.0 LED
        delay_ms(1000, F_6_MeHz);   //Waits for 1000 ms
        P1->OUT &= ~BIT0;            //Turn off P1.0 LED
        delay_ms(250, F_6_MeHz);    //Wait a half pulse

        set_DCO(F_24_MeHz);         //Sets the DCO to 1.5 MHz
        P1->OUT |= BIT0;            //Turn on P1.0 LED
        delay_ms(1000, F_24_MeHz);  //Waits for 1000 ms
        P1->OUT &= ~BIT0;            //Turn off P1.0 LED
        delay_ms(250, F_24_MeHz);   //Wait a half pulse
    }
}
```

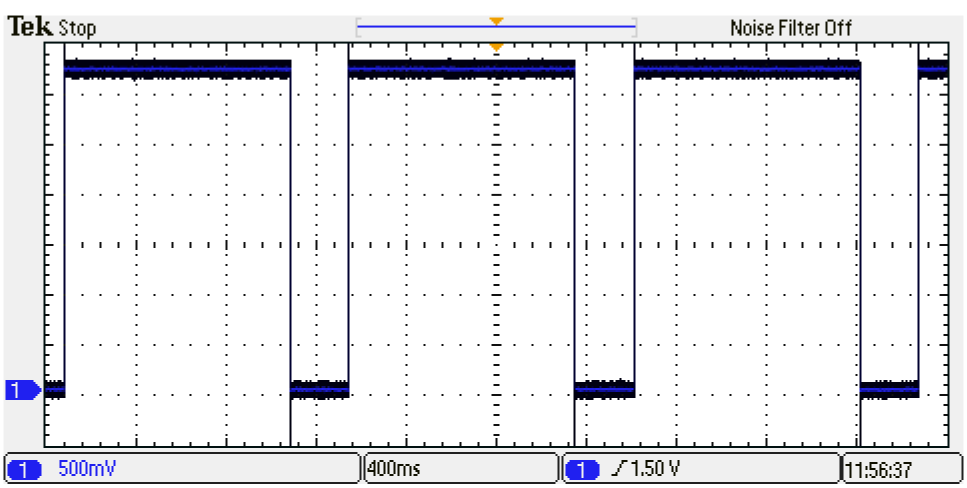

Figure 1: Oscilloscope Capture of the three 1s delay_ms pulses at 1.5 MHz, 6 MHz, and 24 MHz

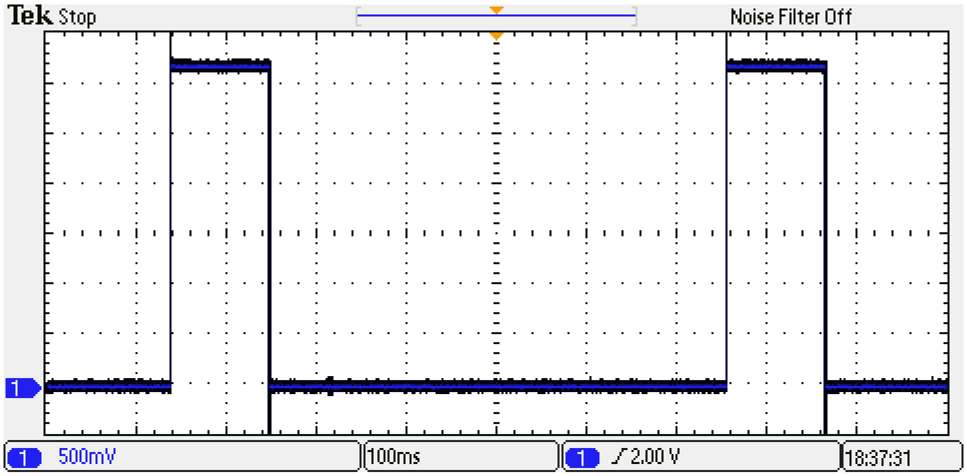

Figure 2: Oscilloscope capture of the 2 100 µs pulses produced with delay_us

C Code Used for the two 100 µs Pulse Test

```
int main(void)
{
    WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;    // Stop WDT

    // P1.0 set as GPIO
    P1->SEL0 &= ~BIT0;          //Clear bit 0 of the P1->SEL0 register
    P1->SEL1 &= ~BIT0;          //Clear bit 0 of the P1->SEL1 register

    P1->DIR |= BIT0;            //P1.0 set as output

    set_DCO(F_24_MeHz);         //Sets the DCO to 24 MHz
    P1->OUT |= BIT0;            //Turn on P1.0 LED
    delay_ms(100, F_24_MeHz);   //Pulse for 100 us
    P1->OUT &= ~BIT0;           //Turn off P1.0 LED
    delay_ms(500, F_24_MeHz);   //Wait 500 us

    P1->OUT |= BIT0;            //Turn on P1.0 LED
    delay_ms(100, F_24_MeHz);   //Pulse for 100 us
    P1->OUT &= ~BIT0;           //Turn off P1.0 LED
}
```

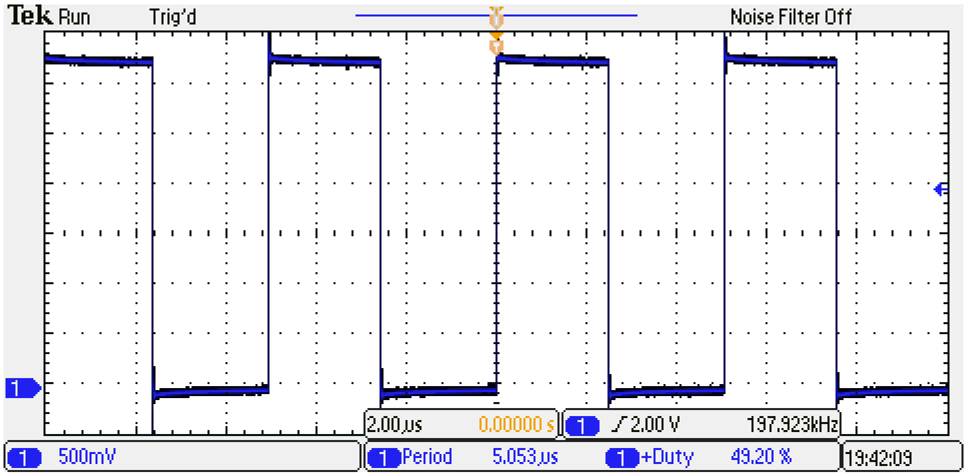

Figure 3: Shortest pulse possible, 5 µs, with a desired delay of 30 µs

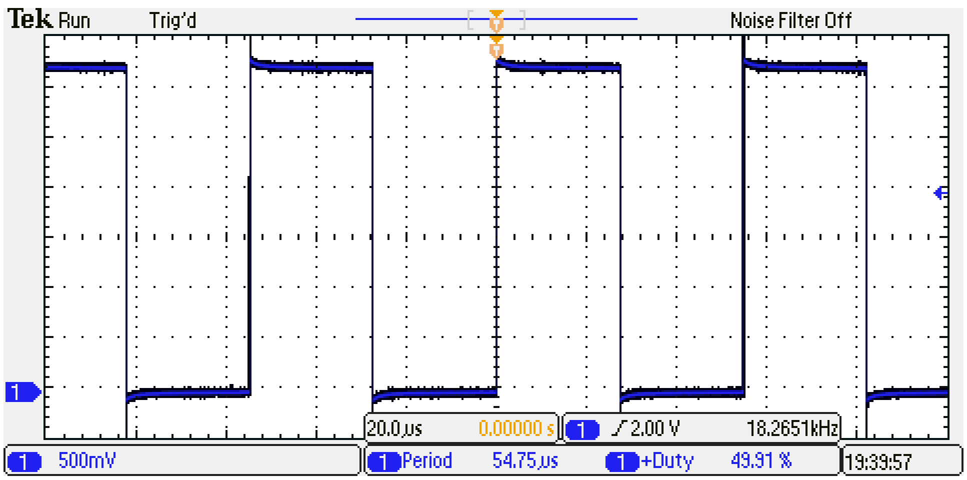

Figure 4: Shortest accurate pulse possible, 54.75 µs, with a desired delay of 55 µs

Final C Code Used to Oscillate Square Waves to Test functions

```
/*
 *  A2 Main
 *  Implements various led pulses.
 *
 *  Date: April 4, 2018
 *  Authors: Zach Bunce, Garrett Maxon
 */

#include "msp.h"
#include "delays.h"
#include "set_DCO.h"

#define F_1p5_MeHz  15      //Defines various frequency values in almost MHz (10^5)
#define F_3_MeHz    30      //MeHz labels are used to indicate this
```

```c
#define F_6_MeHz    60      //Blame data type truncation
#define F_12_MeHz   120
#define F_24_MeHz   240
#define F_48_MeHz   480

void set_DCO(int);
void delay_ms(int, int);
void delay_us(int, int);

int main(void)
{
    WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;    // Stop WDT

    // P1.0 set as GPIO
    P1->SEL0 &= ~BIT0;              //Clear bit 0 of the P1->SEL0 register
    P1->SEL1 &= ~BIT0;              //Clear bit 0 of the P1->SEL1 register

    P1->DIR |= BIT0;                //P1.0 set as output

    //Creates three 1000 ms pulses using 1.5 MHz, 6 MHz, and 24 MHz
    while (1)                       //Continuous loop
    {
        set_DCO(F_1p5_MeHz);        //Sets the DCO to 1.5 MHz
        P1->OUT |= BIT0;            //Turn on P1.0 LED
        delay_ms(1000, F_1p5_MeHz); //Waits for 1000 ms
        P1->OUT &= ~BIT0;            //Turn off P1.0 LED
        delay_ms(1000, F_1p5_MeHz);  //Wait for 1000 ms

        set_DCO(F_6_MeHz);          //Sets the DCO to 1.5 MHz
        P1->OUT |= BIT0;            //Turn on P1.0 LED
        delay_ms(1000, F_6_MeHz);   //Waits for 1000 ms
        P1->OUT &= ~BIT0;            //Turn off P1.0 LED
        delay_ms(1000, F_6_MeHz);    //Wait for 1000 ms

        set_DCO(F_24_MeHz);         //Sets the DCO to 1.5 MHz
        P1->OUT |= BIT0;            //Turn on P1.0 LED
        delay_ms(1000, F_24_MeHz);  //Waits for 1000 ms
        P1->OUT &= ~BIT0;            //Turn off P1.0 LED
        delay_ms(1000, F_24_MeHz);   //Wait for 1000 ms
    }
}
```

Delay Functions

```
/*
 *  delays.c
 *  Code file for both the ms and us delay functions.
 *
 *  Date: April 4, 2018
 *  Authors: Zach Bunce, Garret Maxon
 */

#define F_1p5_MeHz  15      //Defines various frequency values in almost MHz (10^5)
#define F_3_MeHz    30      //MeHz labels are used to indicate this
#define F_6_MeHz    60      //Blame data type truncation
#define F_12_MeHz   120
#define F_24_MeHz   240
#define F_48_MeHz   480

//Takes in desired time delay in ms and clock frequency in MeHz
void delay_ms(int time_ms, int freq_MeHz)
{
    int i;
    unsigned int j;
    //Unit Conversion: MeHz * ms = 10^5 * 10^-3 = 10^2 = 100
    int time_fix = time_ms / 10 + 1;
    unsigned int freq_cnv = freq_MeHz * 100;  //Multiplies in unit conversion to stable variable
    for (i = time_fix; i > 0; i--) {         //Wait the amount of ms specified
        for (j = freq_cnv; j > 0; j--);//    //Wait the amount of MeHz of the system clk for each ms
    }
}

//Takes in desired time delay in us and clock frequency in MeHz
void delay_us(int time_us, int freq_MeHz)
{
    int i;
    int time_fix;
    //Unit Conversion: MeHz * us = 10^5 * 10^-6 = 10^-1 = 0.1; Accounted for in decrement
    switch (freq_MeHz)
    {
    case F_1p5_MeHz:
        time_fix = time_us * freq_MeHz / 100 - 13;  //Fixes the count
        for (i = time_fix; i > 0; i--);             //Wait the amount of us specified
        break;
    case F_3_MeHz:
        time_fix = time_us * freq_MeHz / 100 - 16;  //Fixes the count
        for (i = time_fix; i > 0; i--);             //Wait the amount of us specified
        break;
    case F_6_MeHz:
        time_fix = time_us * freq_MeHz / 100 - 24;  //Fixes the count
        for (i = time_fix; i > 0; i--);             //Wait the amount of us specified
        break;
    case F_12_MeHz:
        time_fix = time_us * freq_MeHz / 100 - 39;  //Fixes the count
        for (i = time_fix; i > 0; i--);             //Wait the amount of us specified
        break;
    case F_24_MeHz:
        time_fix = time_us * freq_MeHz / 100 - 72;  //Fixes the count
        for (i = time_fix; i > 0; i--);             //Wait the amount of us specified
        break;
```

```
    case F_48_MeHz:
        time_fix = time_us * freq_MeHz / 100 - 402; //Fixes the count
        for (i = time_fix; i > 0; i--);             //Wait the amount of us specified
        break;
    }
}
```

DCO Frequency Set Function

```
/*
 *  set_DCO.c
 *  Code file for the DCO frequency change function.
 *
 *  Date: April 6, 2018
 *  Authors: Zach Bunce, Garret Maxon
 */

#include "msp.h"
#define F_1p5_MeHz  15      //Defines various frequency values in almost MHz (10^5)
#define F_3_MeHz    30      //MeHz labels are used to indicate this
#define F_6_MeHz    60      //Blame data type truncation
#define F_12_MeHz   120
#define F_24_MeHz   240
#define F_48_MeHz   480

void set_DCO(int freq)
{
    CS->KEY = CS_KEY_VAL;                                  //Unlocks CS registers
    CS->CTL0 = 0;                                          //Clears CTL0 register

    switch (freq)
    {
    case F_1p5_MeHz:
        CS->CTL0 = CS_CTL0_DCORSEL_0;                      //Sets DC0 to 1.5 MHz
        CS->CTL1 = CS_CTL1_SELA_2 | CS_CTL1_SELS_3 | CS_CTL1_SELM_3; //Sets the clock references
        break;
    case F_3_MeHz:
        CS->CTL0 = CS_CTL0_DCORSEL_1;                      //Sets DC0 to 3 MHz
        CS->CTL1 = CS_CTL1_SELA_2 | CS_CTL1_SELS_3 | CS_CTL1_SELM_3; //Sets the clock references
        break;
    case F_6_MeHz:
        CS->CTL0 = CS_CTL0_DCORSEL_2;                      //Sets DC0 to 6 MHz
        CS->CTL1 = CS_CTL1_SELA_2 | CS_CTL1_SELS_3 | CS_CTL1_SELM_3; //Sets the clock references
        break;
    case F_12_MeHz:
        CS->CTL0 = CS_CTL0_DCORSEL_3;                      //Sets DC0 to 12 MHz
        CS->CTL1 = CS_CTL1_SELA_2 | CS_CTL1_SELS_3 | CS_CTL1_SELM_3; //Sets the clock references
        break;
    case F_24_MeHz:
        CS->CTL0 = CS_CTL0_DCORSEL_4;                      //Sets DC0 to 24 MHz
        CS->CTL1 = CS_CTL1_SELA_2 | CS_CTL1_SELS_3 | CS_CTL1_SELM_3; //Sets the clock references
        break;
    case F_48_MeHz:
        // Transition to VCORE Level 1: AM0_LDO --> AM1_LDO
        while ((PCM->CTL1 & PCM_CTL1_PMR_BUSY))
            ;
        PCM->CTL0 = PCM_CTL0_KEY_VAL | PCM_CTL0_AMR_1;
        while ((PCM->CTL1 & PCM_CTL1_PMR_BUSY));

        // Configure Flash wait-state to 1 for both banks 0 & 1
        FLCTL->BANK0_RDCTL = (FLCTL->BANK0_RDCTL
                & ~(FLCTL_BANK0_RDCTL_WAIT_MASK)) | FLCTL_BANK0_RDCTL_WAIT_1;
        FLCTL->BANK1_RDCTL = (FLCTL->BANK0_RDCTL
                & ~(FLCTL_BANK1_RDCTL_WAIT_MASK)) | FLCTL_BANK1_RDCTL_WAIT_1;
```

```
        CS->CTL0 = CS_CTL0_DCORSEL_5;                       //Sets DC0 to 48 MHz
        CS->CTL1 = CS->CTL1
                & ~(CS_CTL1_SELM_MASK | CS_CTL1_DIVM_MASK)| CS_CTL1_SELM_3; //Sets MCLK to DCO
        break;
    default:
        break;
    }
    CS->KEY = 0;                                        //Locks CS registers
}
```