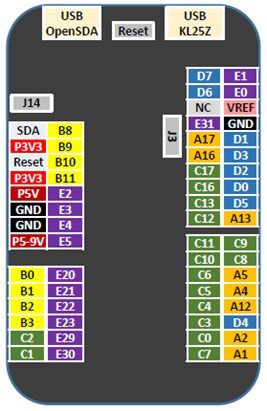
Robbie Decker, Tobias Martin, Nathan Mayton

EE260-01

4/14/20

Project 7

**Project Goals:** To get a servo motor to operate with an interrupt driven system. In one project this interrupt should be a TOF interrupt that happens every 20 milliseconds, and in another project the interrupt should use an ADC conversion complete interrupt.



**Description:**

To get the interrupt working with a TOF interrupt we first needed to initialize a TPM timer. To do this we first needed to figure out what timer and channel that we needed to use. Since the motor was using pin E20 we decided to go with a corresponding timer with that which was TPM1 Channel 0. We then initialized the clock to TPM1, set up its peripherals, and then made sure to use IRQ18 since that corresponds with TPM1. We then had an interrupt handler function which activated the motor and printed the voltage used to a terminal using UART. The main part of this function was getting the ADC conversion to run again. To do this we made sure to enable the COCO flag, the interrupt enable, and the differential mode.

To get the interrupt working with an ADC conversion complete interrupt, many of the same things were done (like how we got the ADC conversion to run again), but instead added some lines of code to the ADC0 initialize function. We added a line to enable the interrupt enable, and we made sure to enable interrupt bit 15 since that corresponds to ADC0. We then had an ADC0 interrupt handler function which acted almost the same as our TPM interrupt handler.

**Code:**

**Project with TOF interrupt**

//Analog input to FRDM KL25Z pin E20 (PTE20)

//PWM output from FRDM KL25Z pin C2 (PTC2)

#include "MKL25Z4.h"

#include <stdio.h>

#include "board.h"

void UART0\_init(void);

void UART0Tx(char c);

void UART0\_puts(char\* s);

void ADC0\_init(void);

void PWM\_init(void);

void LED\_init(void);

void LED\_set(int s);

void TPM1\_IRQHandler(void);

void TPM\_init(void);

void delayMs(int n);

short int result;

short int voltage;

char buffer[20];

int main(void) {

\_\_disable\_irq();

TPM\_init();

BOARD\_InitBootClocks();

UART0\_init();

LED\_init(); /\* Configure LEDs \*/

ADC0\_init(); /\* Configure ADC0 \*/

PWM\_init(); /\* Configure PWM \*/

\_\_enable\_irq();

ADC0->SC1[0] &= ~0x1F; //turns on COCO flag, diff, and enables interrupts

while (1) {

// // ADC0->SC1[0] = 0; /\* start conversion on channel 0 \*/

//

// while(!(ADC0->SC1[0] & 0x80)) { } /\* wait for conversion complete \*/

// result = ADC0->R[0]; /\* read conversion result and clear COCO flag \*/

//

//// For LED

// LED\_set(result >> 7); /\* display result on LED with bits 2, 1, & 0 after shift \*/

//

//// For Servo

// TPM0->CONTROLS[1].CnV = 1500 + result\*3/2; /\* Set up channel value between 2.5% and 12.5%\*/

// For the Buzzer

// TPM0->MOD = 6000 - 5\*(result/4); /\* Frequencies from 500 Hz to 3000 Hz\*/

// TPM0->CONTROLS[1].CnV = TPM0->MOD/2; /\* Set up channel value between 50%\*/

// For DC motor

// TPM0->CONTROLS[1].CnV = result\*14; /\* Set up channel value between 0% - 93%\*/

}

}

void TPM\_init(void){

//SIM->SOPT2 |= 0x01000000; /\* use MCGFLLCLK as timer counter clock \*/

SIM->SCGC6 |= 0x02000000; /\* enable clock to TPM0 \*/

TPM1->SC = 0; /\* disable timer while configuring \*/

TPM1->SC = 0x07; /\* prescaler /128 \*/

TPM1->MOD = 0xFFFF; /\* max modulo value \*/

TPM1->SC |= 0x80; /\* clear TOF \*/

TPM1->SC |= 0x40; /\* enable timeout interrupt \*/

TPM1->SC |= 0x08; /\* enable timer \*/

//NVIC->ISER[0] |= 0x00020000; /\* enable IRQ17 (bit 17 of ISER[0]) \*/

NVIC\_EnableIRQ(18);

}

void TPM1\_IRQHandler(void){

result = ADC0->R[0];

voltage = result \* 3300/4096;

TPM0->CONTROLS[1].CnV = 1500 + result\*3/2; /\* Set up channel value between 2.5% and 12.5%\*/

sprintf(buffer, "\r\n voltage = %d mV", voltage);

UART0\_puts(buffer);

delayMs(100);

LED\_set(result >> 7);

ADC0->SC1[0] &= ~0x17;

}

void UART0\_init(void) {

SIM->SCGC4 |= 0x0400; /\* enable clock for UART0 \*/

SIM->SOPT2 |= 0x04000000; /\* use FLL output for UART Baud rate generator \*/

UART0->C2 = 0; /\* turn off UART0 while changing configurations \*/

UART0->BDH = 0x00;

UART0->BDL = 0xC; /\* 115200 Baud \*/

UART0->C4 = 0x0F; /\* Over Sampling Ratio 16 \*/

UART0->C1 = 0x00; /\* 8-bit data \*/

UART0->C2 = 0x08; /\* enable transmit \*/

SIM->SCGC5 |= 0x0200; /\* enable clock for PORTA \*/

PORTA->PCR[2] = 0x0200; /\* make PTA2 UART0\_Tx pin \*/

}

void UART0Tx(char c) {

while(!(UART0->S1 & 0x80)) {

} /\* wait for transmit buffer empty \*/

UART0->D = c; /\* send a char \*/

}

void UART0\_puts(char\* s) {

while (\*s != 0) /\* if not end of string \*/

UART0Tx(\*s++); /\* send the character through UART0 \*/

}

/\* Delay n milliseconds

\* The CPU core clock is set to MCGFLLCLK at 41.94 MHz in SystemInit().

\*/

void delayMs(int n) {

int i;

int j;

for(i = 0 ; i < n; i++)

for (j = 0; j < 3500; j++) {}

}

void PWM\_init(void)

{

SIM->SCGC5 |= 0x800; /\* enable clock to Port C\*/

PORTC->PCR[2] = 0x0400; /\* PTC2 used by TPM0 \*/

SIM->SCGC6 |= 0x01000000; /\* enable clock to TPM0 \*/

SIM->SOPT2 |= 0x01000000; /\* use MCGFLLCLK as timer counter clock \*/

TPM0->SC = 0; /\* disable timer \*/

// TPM0->CONTROLS[1].CnSC = 0x20|0x08; /\* edge-aligned PWM, pulse high MSB:MSA=10, ELSB:ELSA=10\*/

TPM0->CONTROLS[1].CnSC |= TPM\_CnSC\_MSB\_MASK |TPM\_CnSC\_ELSB\_MASK; //Enable TPM0\_CH1 as edge-aligned PWM

TPM0->MOD = 60000; /\* Set up modulo register for 50 Hz - 48.00 MHz \*/

TPM0->CONTROLS[1].CnV = 1500; /\* Set up channel value for 2.5% duty-cycle \*/

TPM0->SC |= 0x0C; /\* enable TPM0 with pre-scaler /16 \*/

}

void ADC0\_init(void)

{

//uint16\_t calibration;

SIM->SCGC5 |= 0x2000; /\* clock to PORTE \*/

PORTE->PCR[20] = 0; /\* PTE20 analog input \*/

SIM->SCGC6 |= 0x8000000; /\* clock to ADC0 \*/

ADC0->SC2 &= ~0x40; /\* software trigger \*/

/\* clock div by 4, long sample time, single ended 12 bit, bus clock \*/

ADC0->CFG1 = 0x40 | 0x10 | 0x04 | 0x00;

// ADC0->SC1[0] |= 0x40;

// NVIC->ISER[0] |= 0x8000;

}

void LED\_init(void) {

SIM->SCGC5 |= 0x400; /\* enable clock to Port B \*/

SIM->SCGC5 |= 0x1000; /\* enable clock to Port D \*/

PORTB->PCR[18] = 0x100; /\* make PTB18 pin as GPIO \*/

PTB->PDDR |= 0x40000; /\* make PTB18 as output pin \*/

PORTB->PCR[19] = 0x100; /\* make PTB19 pin as GPIO \*/

PTB->PDDR |= 0x80000; /\* make PTB19 as output pin \*/

PORTD->PCR[1] = 0x100; /\* make PTD1 pin as GPIO \*/

PTD->PDDR |= 0x02; /\* make PTD1 as output pin \*/

}

void LED\_set(int s) {

if (s & 1) /\* use bit 0 of s to control red LED \*/

PTB->PCOR = 0x40000; /\* turn on red LED \*/

else

PTB->PSOR = 0x40000; /\* turn off red LED \*/

if (s & 2) /\* use bit 1 of s to control green LED \*/

PTB->PCOR = 0x80000; /\* turn on green LED \*/

else

PTB->PSOR = 0x80000; /\* turn off green LED \*/

if (s & 4) /\* use bit 2 of s to control blue LED \*/

PTD->PCOR = 0x02; /\* turn on blue LED \*/

else

PTD->PSOR = 0x02; /\* turn off blue LED \*/

}

**Project with ADC conversion complete interrupt**

//Analog input to FRDM KL25Z pin E20 (PTE20)

//PWM output from FRDM KL25Z pin C2 (PTC2)

#include "MKL25Z4.h"

#include "board.h"

#include <stdio.h>

void UART0\_init(void);

void UART0Tx(char c);

void UART0\_puts(char\* s);

void ADC0\_init(void);

void PWM\_init(void);

void LED\_init(void);

void LED\_set(int s);

void ADC0\_IRQHandler(void);

void delayMs(int n);

short int result;

short int voltage;

char buffer[20];

int main(void) {

\_\_disable\_irq();

BOARD\_InitBootClocks();

UART0\_init();

LED\_init(); /\* Configure LEDs \*/

ADC0\_init(); /\* Configure ADC0 \*/

PWM\_init(); /\* Configure PWM \*/

\_\_enable\_irq();

ADC0->SC1[0] &= ~0x1F; //turns on COCO flag, diff, and enables interrupts

while (1) {

// // ADC0->SC1[0] = 0; /\* start conversion on channel 0 \*/

//

// while(!(ADC0->SC1[0] & 0x80)) { } /\* wait for conversion complete \*/

// result = ADC0->R[0]; /\* read conversion result and clear COCO flag \*/

//

//// For LED

// LED\_set(result >> 7); /\* display result on LED with bits 2, 1, & 0 after shift \*/

//

//// For Servo

// TPM0->CONTROLS[1].CnV = 1500 + result\*3/2; /\* Set up channel value between 2.5% and 12.5%\*/

// For the Buzzer

// TPM0->MOD = 6000 - 5\*(result/4); /\* Frequencies from 500 Hz to 3000 Hz\*/

// TPM0->CONTROLS[1].CnV = TPM0->MOD/2; /\* Set up channel value between 50%\*/

// For DC motor

// TPM0->CONTROLS[1].CnV = result\*14; /\* Set up channel value between 0% - 93%\*/

}

}

void ADC0\_IRQHandler(void){

result = ADC0->R[0];

voltage = result \* 3300/4096;

TPM0->CONTROLS[1].CnV = 1500 + result\*3/2; /\* Set up channel value between 2.5% and 12.5%\*/

sprintf(buffer, "\r\n voltage = %d mV", voltage);

UART0\_puts(buffer);

delayMs(100);

LED\_set(result >> 7);

ADC0->SC1[0] &= ~0x17;

}

void UART0\_init(void) {

SIM->SCGC4 |= 0x0400; /\* enable clock for UART0 \*/

SIM->SOPT2 |= 0x04000000; /\* use FLL output for UART Baud rate generator \*/

UART0->C2 = 0; /\* turn off UART0 while changing configurations \*/

UART0->BDH = 0x00;

UART0->BDL = 0xC; /\* 115200 Baud \*/

UART0->C4 = 0x0F; /\* Over Sampling Ratio 16 \*/

UART0->C1 = 0x00; /\* 8-bit data \*/

UART0->C2 = 0x08; /\* enable transmit \*/

SIM->SCGC5 |= 0x0200; /\* enable clock for PORTA \*/

PORTA->PCR[2] = 0x0200; /\* make PTA2 UART0\_Tx pin \*/

}

void UART0Tx(char c) {

while(!(UART0->S1 & 0x80)) {

} /\* wait for transmit buffer empty \*/

UART0->D = c; /\* send a char \*/

}

void UART0\_puts(char\* s) {

while (\*s != 0) /\* if not end of string \*/

UART0Tx(\*s++); /\* send the character through UART0 \*/

}

/\* Delay n milliseconds

\* The CPU core clock is set to MCGFLLCLK at 41.94 MHz in SystemInit().

\*/

void delayMs(int n) {

int i;

int j;

for(i = 0 ; i < n; i++)

for (j = 0; j < 3500; j++) {}

}

void PWM\_init(void)

{

SIM->SCGC5 |= 0x800; /\* enable clock to Port C\*/

PORTC->PCR[2] = 0x0400; /\* PTC2 used by TPM0 \*/

SIM->SCGC6 |= 0x01000000; /\* enable clock to TPM0 \*/

SIM->SOPT2 |= 0x01000000; /\* use MCGFLLCLK as timer counter clock \*/

TPM0->SC = 0; /\* disable timer \*/

// TPM0->CONTROLS[1].CnSC = 0x20|0x08; /\* edge-aligned PWM, pulse high MSB:MSA=10, ELSB:ELSA=10\*/

TPM0->CONTROLS[1].CnSC |= TPM\_CnSC\_MSB\_MASK |TPM\_CnSC\_ELSB\_MASK; //Enable TPM0\_CH1 as edge-aligned PWM

TPM0->MOD = 60000; /\* Set up modulo register for 50 Hz - 48.00 MHz \*/

TPM0->CONTROLS[1].CnV = 1500; /\* Set up channel value for 2.5% duty-cycle \*/

TPM0->SC |= 0x0C; /\* enable TPM0 with pre-scaler /16 \*/

}

void ADC0\_init(void)

{

//uint16\_t calibration;

SIM->SCGC5 |= 0x2000; /\* clock to PORTE \*/

PORTE->PCR[20] = 0; /\* PTE20 analog input \*/

SIM->SCGC6 |= 0x8000000; /\* clock to ADC0 \*/

ADC0->SC2 &= ~0x40; /\* software trigger \*/

/\* clock div by 4, long sample time, single ended 12 bit, bus clock \*/

ADC0->CFG1 = 0x40 | 0x10 | 0x04 | 0x00;

ADC0->SC1[0] |= 0x40;

NVIC->ISER[0] |= 0x8000;

}

void LED\_init(void) {

SIM->SCGC5 |= 0x400; /\* enable clock to Port B \*/

SIM->SCGC5 |= 0x1000; /\* enable clock to Port D \*/

PORTB->PCR[18] = 0x100; /\* make PTB18 pin as GPIO \*/

PTB->PDDR |= 0x40000; /\* make PTB18 as output pin \*/

PORTB->PCR[19] = 0x100; /\* make PTB19 pin as GPIO \*/

PTB->PDDR |= 0x80000; /\* make PTB19 as output pin \*/

PORTD->PCR[1] = 0x100; /\* make PTD1 pin as GPIO \*/

PTD->PDDR |= 0x02; /\* make PTD1 as output pin \*/

}

void LED\_set(int s) {

if (s & 1) /\* use bit 0 of s to control red LED \*/

PTB->PCOR = 0x40000; /\* turn on red LED \*/

else

PTB->PSOR = 0x40000; /\* turn off red LED \*/

if (s & 2) /\* use bit 1 of s to control green LED \*/

PTB->PCOR = 0x80000; /\* turn on green LED \*/

else

PTB->PSOR = 0x80000; /\* turn off green LED \*/

if (s & 4) /\* use bit 2 of s to control blue LED \*/

PTD->PCOR = 0x02; /\* turn on blue LED \*/

else

PTD->PSOR = 0x02; /\* turn off blue LED \*/

}

