Laboratory 1: Input Interfacing with the MSP432 Using Interrupts

EGR-326

Jack Lukomski & Zack Peters

September 8, 2022

Contents

1 Objective

The purpose of this laboratory is to develop a program for the MSP432 microcontroller that interfaces with pushbutton switches to control a red-green-blue (RGB) light-emitting diode (LED), utlize a debouncing method to have percice button inputs, and to use multiple pushbuttons to carry out functions.

2 Materials

2.1 Apparatus

1. Digital Multi-Meter
2. DC Power Supply

2.2 Components

1. $2x$ Pushbuttons
2. 100Ω Resistor
3. Common Cathode RGB LED
4. MSP432R Microcontroller
5. Breadboard
6. Assorted Wires

2.3 Software

1. Code-Composer Studio (CCS)

3 Libraries

Before we started creating the functional part of this laboratory, MSP432 libaries were created to make interfacing with the MSP432 much easier in the future.

3.1 pinsInit Library

3.1.1 Purpose

The purpose of this library is to create functions to make initlizing pins as GPIOs with pullup or pulldown resistors much easier by calling one simple function.

3.1.2 Header File

```c
 /*
 * pinsInint.h
 *
 *  Created on: Sep 2, 2022
 *      Author: jtluk
 */

#ifndef PINSININT_H_
#define PINSININT_H_
#include <stdint.h>
#include "msp.h"

typedef enum port2Pins_t
{
    pin0,
    pin1,
    pin2,
    pin3,
    pin4,
    pin5,
    pin6,
    pin7,
}port2Pins_t;

typedef enum port2IO_t
{
    input,
    output,
}port2IO_t;

typedef enum port2GPIOConfig_t
{
   pullup,
   pulldown,
}port2GPIOConfig_t;

typedef struct port2GPIO_t
{
    port2Pins_t e_IOpinNumber;
    port2GPIOConfig_t e_GPIOType;
    port2GPIOConfig_t e_IO;
}port2GPIO_t;

void vpinsInit_GPIO(port2GPIO_t * s_userGPIO_ptr, port2Pins_t e_userPin,
   port2IO_t e_userPortIO, port2GPIOConfig_t e_userGPIO);
```

```
static void vPrv_pinsInit_InitIO(port2Pins_t e_userPin, port2IO_t
   e_userPortIO);
static void vPrv_pinsInit_InitConfig(port2Pins_t e_userPin,
   port2GPIOConfig_t e_userGPIO);

#endif /* PINSININT_H_ */
```

3.1.3 Typedef and Function Descriptions

1.
   ```
   typedef enum port2Pins_t
   ```

   This is an enumeration of all avalible pins on MSP432's port 2.

2.
   ```
   typedef enum port2IO_t
   ```

   This is an enumeration of all avalible IO options on the MSP432 (Input and Output).

3.
   ```
   typedef enum port2GPIOConfig_t
   ```

   This is an enumeration to reprsent pulling up or pulling down the MSP432's internal resistor.

4.
   ```
   typedef struct port2GPIO_t
   ```

   This is a struture containing all enums listed above to encapsulate initlizing a pin.

5.
   ```
   void vpinsInit_GPIO(port2GPIO_t * s_userGPIO_ptr, port2Pins_t
   e_userPin, port2IO_t e_userPortIO, port2GPIOConfig_t e_userGPIO)
   ```

   This is a function used to initlize a single pin on the MSP432.

6.
   ```
   static void vPrv_pinsInit_InitIO(port2Pins_t e_userPin, port2IO_t
   e_userPortIO)
   ```

   This is a private helper function.

7.
   ```
   static void vPrv_pinsInit_InitConfig(port2Pins_t e_userPin,
   port2GPIOConfig_t e_userGPIO);
   ```

   This is a private helper function.

3.2 readInputs Library

3.2.1 Purpose

The purpose of this library is to create a simple function that will return 1 or 0 based on if a user input is pressed or not, automatically accounting for debouncing.

3.2.2 Header File

```c
/*
 * readInputs.h
 *
 *  Created on: Sep 2, 2022
 *      Author: jtluk
 */

#ifndef READINPUTS_H_
#define READINPUTS_H_

#define MIN_DEBOUNCE_VAL 0xe000
#define MAX_DEBOUNCE_VAL 0xf000
#include "pinsInint.h"
#include <stdbool.h>

 bool xReadInputs_ReadPin(port2GPIO_t e_userPin);
 static bool xPrv_ReadInputs_read(port2GPIO_t e_userPin);

#endif /* READINPUTS_H_ */
```

3.2.3 Function Descriptions

1.
   ```c
   bool xReadInputs_ReadPin(port2GPIO_t e_userPin)
   ```

   This is function that debounces a user button press and returns 1 for a user input, and a 0 if there is no user input.

2.
   ```c
   static bool xPrv_ReadInputs_read(port2GPIO_t e_userPin)
   ```

   This is a private function that returns the raw, un-debounced button input from the user.

3.3 Toggle Library

3.4 Purpose

The purpose of this libary is for a user to call a simple function with a single parameter to toggle a pin from high to low, or low to high.

3.4.1 Header File

```c
/*
 * cycleRGB.h
 *
 *  Created on: Sep 2, 2022
 *      Author: jtluk
 */

#ifndef TOGGLE_H_
#define TOGGLE_H_
#include "msp.h"
#include "pinsInint.h"

 void vToggle_turnPinHigh(port2GPIO_t e_userColor);
```

```
void vToggle_turnPinLow(port2GPIO_t e_userColor);

#endif /* TOGGLE_H_ */
```

3.4.2 Function Descriptions

1.
   ```
   void vToggle_turnPinHigh(port2GPIO_t e_userColor)
   ```

   This function toggles a pin to high, it accounts for the pin being a internal pullup, or pulldown resistor.

2.
   ```
   void vToggle_turnPinLow(port2GPIO_t e_userColor)
   ```

   This function toggles a pin to low, it accounts for the pin being a internal pullup, or pulldown resistor.

4 Procedure

4.1 Part I – Sequencing colors of a RGB LED using a pushbutton switch

4.1.1 Description

Part one of this laboratory consisted of creating a program to control a RGB LED using the MSP432R microcontroller and a pushbutton for user input. On reset, all LEDs should be off. On the first button press, the red LED should be turned on, on the second press, the green LED should turn on and the red LED should turn off. on the third button press, the blue LED should turn on and the green LED should turn off, after the forth button press, the cycle continues starting back at the red LED.

4.1.2 Soulution

To complete this part of the laboratory, five functions were created in the *main.c* file for each color of the LED, turing the led off, and initlizing each pin.

-
  ```
  void vMain_InitOutputs(void)
  ```

  ```
  void vMain_TurnRedLedOn(void)
  ```

-
  ```
  void vMain_TurnGreenLedOn(void)
  ```

  ```
  void vMain_TurnBlueLedOn(void)
  ```

-
  ```
  void vMain_TurnAllOff(void)
  ```

were the functions used to complete this. These functions use the libaries described in the Libraries section.
After this, a function pointer, *vMain_LedState_ptr* was created and typedefed to create a way to refrence each of the funcitons listed above. A table was then created consisting of the type *vMain_LedState_ptr*, as shown below.

```
static const vMain_LedState_ptr ledState_t[NUM_STATES] =
{
 vMain_TurnRedLedOn,
 vMain_TurnGreenLedOn,
 vMain_TurnBlueLedOn,
};
```

This table contains all functions needed to cycle the RGB led, and is easy to iterate through with a simple counter. In the main function, a *if* statment that is true when the pushbutton is pressed, contains the counter that goes from $0 - 2$ to iterate through the *ledState_t* table.

4.1.3 Result

After compiling and running the code, the program works as expected.

4.2 Part II – Sequencing colors of a RGB LED using two switches with reverse direction

4.2.1 Description

For this part of the laboratory, another button is added to the circuit to have the RGB LED go in the reverse direction of what was completed in Part 1 of this labratorty.

4.2.2 Soulution

The solution for this part of the labratory consisted of having another *if* statment in the *while* loop that checked if the second pushbutton is pressed. If it is pressed, the counter is subtracted by one thus causing the state to go back one.

4.2.3 Result

After compiling and running the code, this part of the laboratory worked.

5 Conclusion

In conclusion, this labratory took off the rust for programming the MSP432. One thing we focused on this labratorty is creating libaries that can be used later labs to make life much easier.

6 Code

6.1 main.c

```c
#include "Toggle.h"
#include "msp.h"
#include "pinsInint.h"
#include "readInputs.h"
#include "pinsInint.h"
#include "port3PinsInit.h"
#include "port3ReadInputs.h"

#define NUM_STATES 3
#define PART_TWO 1

/**
 * main.c
 *
 * PIN 2.4 B
 * PIN 2.5 G
 * PIN 2.6 R
 */

port2GPIO_t s_redLed;
port2GPIO_t s_greenLed;
port2GPIO_t s_blueLed;
port2GPIO_t s_fowardPushButton;
port3GPIO_t s_backwardsPushButton;

typedef void(*vMain_LedState_ptr)(void);

void vMain_InitOutputs(void);
void vMain_TurnAllOff(void);
void vMain_TurnRedLedOn(void);
void vMain_TurnGreenLedOn(void);
void vMain_TurnBlueLedOn(void);

void vMain_GoFoward(int8_t * currButtonState);
void vMain_GoBackwards(int8_t * currButtonState);

static const vMain_LedState_ptr ledState_t[NUM_STATES] =
{
 vMain_TurnRedLedOn,
 vMain_TurnGreenLedOn,
 vMain_TurnBlueLedOn,
};

void main(void)
{
  WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;    // stop watchdog timer

     vpinsInit_GPIO(&s_fowardPushButton, pin7, input, pullup);
```

```
      #if PART_TWO
      vpinsInit_3GPIO(&s_backwardsPushButton, pin32, input3, pullup3);
      #endif

    int8_t currentButtonState = 0;

    while(1)
    {
        if(xReadInputs_ReadPin(s_fowardPushButton) != false)
        {
            vMain_GoFoward(&currentButtonState);
        }

          #if PART_TWO
        if(xReadInputs_3ReadPin(s_backwardsPushButton) != false)
        {
            vMain_GoBackwards(&currentButtonState);
        }
          #endif
    }
}

void vMain_GoFoward(int8_t * currButtonState)
{
    ledState_t[*currButtonState]();
    if(++(*currButtonState) >= 3)
    {
        *currButtonState = 0;
    }
}

void vMain_GoBackwards(int8_t * currButtonState)
{
    if(--(*currButtonState) < 0)
    {
        *currButtonState = 2;
    }
    ledState_t[*currButtonState]();
}


void vMain_InitOutputs(void)
{
    vpinsInit_GPIO(&s_greenLed, pin5, output, pullup);
    vpinsInit_GPIO(&s_redLed, pin6, output, pullup);
    vpinsInit_GPIO(&s_blueLed, pin4, output, pullup);
}

void vMain_TurnRedLedOn(void)
{
    vMain_InitOutputs();
    vToggle_turnPinLow(s_blueLed);
    vToggle_turnPinLow(s_greenLed);
    vToggle_turnPinHigh(s_redLed);
}

void vMain_TurnGreenLedOn(void)
{
    vMain_InitOutputs();
    vToggle_turnPinLow(s_redLed);
    vToggle_turnPinLow(s_blueLed);
    vToggle_turnPinHigh(s_greenLed);
}

void vMain_TurnBlueLedOn(void)
{
    vMain_InitOutputs();
    vToggle_turnPinLow(s_greenLed);
    vToggle_turnPinLow(s_redLed);
    vToggle_turnPinHigh(s_blueLed);
}

void vMain_TurnAllOff(void)
{
    vToggle_turnPinLow(s_greenLed);
    vToggle_turnPinLow(s_redLed);
    vToggle_turnPinLow(s_blueLed);
}
```

6.2 pinsInint.c

```
 /*
 * pinsInint.c
 *
 *  Created on: Sep 2, 2022
 *      Author: jtluk
 */
#include "pinsInint.h"

void vpinsInit_GPIO(port2GPIO_t * s_userGPIO_ptr, port2Pins_t e_userPin,
   port2IO_t e_userPortIO, port2GPIOConfig_t e_userGPIO)
{
    s_userGPIO_ptr->e_GPIOType = e_userGPIO;
    s_userGPIO_ptr->e_IO = e_userPortIO;
    s_userGPIO_ptr->e_IOpinNumber = e_userPin;

    P2->SEL0 &= ~BIT(e_userPin);
    P2->SEL1 &= ~BIT(e_userPin);
    vPrv_pinsInit_InitIO(e_userPin, e_userPortIO);
    vPrv_pinsInit_InitConfig(e_userPin, e_userGPIO);
}

static void vPrv_pinsInit_InitIO(port2Pins_t e_userPin, port2IO_t
   e_userPortIO)
{
    if(e_userPortIO == output)
    {
        P2->DIR |= BIT(e_userPin);
    }
    else
    {
        P2->DIR &= ~BIT(e_userPin);
    }
}

static void vPrv_pinsInit_InitConfig(port2Pins_t e_userPin,
   port2GPIOConfig_t e_userGPIO)
{
    if(e_userGPIO == pullup)
    {
        P2->REN |= BIT(e_userPin); // pull up resistor
        P2->OUT |= BIT(e_userPin); //
    }
    else
    {
        P2->REN &= ~BIT(e_userPin); // pull down resistor
        P2->OUT &= ~BIT(e_userPin); //
    }
}
```

6.3 readInputs.c

```
#include "readInputs.h"
/*
 * readInputs.c
 *
 *  Created on: Sep 2, 2022
 *      Author: jtluk
 */
bool xReadInputs_ReadPin(port2GPIO_t e_userPin)
{
    bool b_retVal = false;
    static uint16_t currDebounceState_t = 0;

    currDebounceState_t = (currDebounceState_t << 1) | !(
  xPrv_ReadInputs_read(e_userPin)) | MIN_DEBOUNCE_VAL;

    if(currDebounceState_t == MAX_DEBOUNCE_VAL)
    {
        b_retVal = true;
    }

    return b_retVal;

}

static bool xPrv_ReadInputs_read(port2GPIO_t e_userPin)
{
    bool b_retVal;

    if(e_userPin.e_GPIOType == pullup)
    {
        b_retVal = ((bool)(P2->IN & BIT(e_userPin.e_IOpinNumber)));
    }
    else if (e_userPin.e_GPIOType == pulldown)
    {
        b_retVal = !((bool)(P2->IN & BIT(e_userPin.e_IOpinNumber)));
    }
    else
    {
        b_retVal = false;
    }

    return b_retVal;
}
```

6.4 Toggle.c

```
#include "Toggle.h"
/*
 * cycleRGB.c
 *
 *  Created on: Sep 2, 2022
 *      Author: jtluk
 */

void vToggle_turnPinHigh(port2GPIO_t e_userColor)
{
    if(e_userColor.e_GPIOType == pullup)
    {
        P2->OUT &= ~BIT(e_userColor.e_IOpinNumber);
    }
    else if(e_userColor.e_GPIOType == pulldown)
    {
        P2->OUT |= BIT(e_userColor.e_IOpinNumber);
    }
}

void vToggle_turnPinLow(port2GPIO_t e_userColor)
{
    if(e_userColor.e_GPIOType == pullup)
    {
        P2->OUT |= BIT(e_userColor.e_IOpinNumber);
    }
    else if(e_userColor.e_GPIOType == pulldown)
    {
        P2->OUT &= ~BIT(e_userColor.e_IOpinNumber);
    }
}
```