**ECEN 5813**

**Chutao Wei**

**Curry Buscher**

**PES Project 4 Code pdf**

**readme.md**

**# cu-ecen-5813-project-4**

**\*\*Title:\*\***

PES Project 4 Readme **<**br/**>**

**\*\*Name:\*\***

Curry Buscher, Chutao Wei **<**br/**>**

**\*\*Repository Comments:\*\*** **<**br/**>**

In documents folder: **<**br/**>**

There are PES Project 4.pdf, and state machine diagram.jpg**<**br/**>**

In source folder: **<**br/**>**

main.c: main function wrapper has two versions. One runs the test script without command line, one require user to put in command in console.**<**br/**>**

memory\_utility.c/h: contains all memory utility functions.**<**br/**>**

pattern\_gen.c/h: generate random byte array using linear feedback shit register**<**br/**>**

led.c/h: contains RGB LED control functions**<**br/**>**

timer.c/h: contains only blocking delay function for now**<**br/**>**

gpio.c/h: contains gpio control functions**<**br/**>**

state.c/h: state machine function**<**br/**>**

touch\_sen.c/h: contains touch sensor printing function**<**br/**>**

mma8451.c/h: contains mma8451 accelerameter function**<**br/**>**

test.c: contains test function for uCUnit testfunction**<**br/**>**

uCUnit.c/h: uCUnit test function**<**br/**>**

System.c/h: System for uCUnit**<**br/**>**

(see more details in PES Project 4.pdf) **<**br/**>**

**\*\*Project Comments:\*\***

Please use semihost **<**br/**>**

**### \*\*Installation/Execution/Editing Notes:\*\*<br/>**

**\*\*Language:\*\***

C**<**br/**>**

**\*\*Compiler:\*\***

GCC version 7.4.0**<**br/**>**

**\*\*IDE:\*\***

MCUExpresso**<**br/**>**

**\*\*Build Environment:\*\***

Ubuntu 16 or up**<**br/**>**

**\*\*Target Environment:\*\***

KL25Z/Linux**<**br/**>**

**\*\*License:\*\***

MIT**<**br/**>**

**project\_5.c (main.c)**

/\*

\* Copyright 2016-2020 NXP

\* All rights reserved.

\*

\* Redistribution and use in source and binary forms, with or without modification,

\* are permitted provided that the following conditions are met:

\*

\* o Redistributions of source code must retain the above copyright notice, this list

\* of conditions and the following disclaimer.

\*

\* o Redistributions in binary form must reproduce the above copyright notice, this

\* list of conditions and the following disclaimer in the documentation and/or

\* other materials provided with the distribution.

\*

\* o Neither the name of NXP Semiconductor, Inc. nor the names of its

\* contributors may be used to endorse or promote products derived from this

\* software without specific prior written permission.

\*

\* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND

\* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED

\* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE

\* DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR

\* ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES

\* (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;

\* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON

\* ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT

\* (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS

\* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*/

/\*\*

\* @file Project\_4.c

\* @brief Application entry point.

\*/

**#include** <stdio.h>

**#include** "board.h"

**#include** "peripherals.h"

**#include** "pin\_mux.h"

**#include** "clock\_config.h"

**#include** "MKL25Z4.h"

**#include** "fsl\_debug\_console.h"

**#include** "i2c.h"

**#include** "gpio.h"

**#include** "led.h"

**#include** "mma8451.h"

**#include** "touch\_sen.h"

**#include** "timer.h"

**#include** "state.h"

**#include** "logger.h"

**#include** "command\_parser.h"

**#include** "buffer.h"

**#include** "test.h"

/\*

\* @brief Application entry point.

\*/

//#define DEBUG\_MODE

//#define NORMAL\_MODE

**#define** TEST\_MODE

//#define ECHO\_SUBMODE

//#define APPL\_SUBMODE

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

/\* Init board hardware. \*/

BOARD\_InitBootPins();

BOARD\_InitBootClocks();

BOARD\_InitBootPeripherals();

/\* Init FSL debug console. \*/

BOARD\_InitDebugConsole();

/\* Init Systick \*/

Init\_SysTick();

/\* Init LED \*/

init\_LED();

LOG\_INFO("Hello, PES Project 5\n");

LOG\_INFO("LED will blink green for a sec to indicate start\n");

turn\_LED\_green(*on*);

mdelay(1000);

turn\_LED\_green(*off*);

/\* Enter an infinite loop \*/

**while**(1)

{

// DEBUG MODE

**#ifdef** DEBUG\_MODE

**#define** INCLUDE\_LOG\_DEBUG 1

**#ifdef** ECHO\_SUBMODE

**#endif**

**#ifdef** APPL\_SUBMODE

command\_parser();

**#endif**

**#endif**

// NORMAL MODE

**#ifdef** NORMAL\_MODE

**#ifdef** ECHO\_SUBMODE

**#endif**

**#ifdef** APPL\_SUBMODE

command\_parser();

**#endif**

**#endif**

// TEST MODE

**#ifdef** TEST\_MODE

Test();

**#endif**

}

**return** 0 ;

}

**timer.c (main.c)**

/\*

\* timer.c

\*

\* Created on: Feb 11, 2020

\* Author: chutao

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Include \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#include** <logger.h>

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "state.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Define \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#define** BLOCK\_WAITING

**#define** CPU\_FREQ\_MHZ (48)

**#define** NUM\_ASSE\_FOR (7)

**#define** DELAY\_MS\_TO\_LOOP\_COUNT(msec)\

((uint32\_t)((msec\*(CPU\_FREQ\_MHZ\*1000))/(NUM\_ASSE\_FOR)))

//#define BLOCKWAITING

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Global Variables \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**const** uint32\_t delay\_look\_up\_table[] = {

DELAY\_MS\_TO\_LOOP\_COUNT(500),

DELAY\_MS\_TO\_LOOP\_COUNT(1000),

DELAY\_MS\_TO\_LOOP\_COUNT(2000),

DELAY\_MS\_TO\_LOOP\_COUNT(3000)};

uint64\_t msec\_count = 0;

uint64\_t target\_msec\_count = 0;

bool delay\_flag = 0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Interrupt Hanlder \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **SysTick\_Handler**(**void**)

{

msec\_count ++;

**if** (delay\_flag == true)

{

**if** (msec\_count == target\_msec\_count)

{

delay\_flag = false;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Function \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **Init\_SysTick**(**void**) {

SysTick->LOAD = (48000L-1L);// count 1 msec

NVIC\_SetPriority(*SysTick\_IRQn*, 4); // enable NVIC

SysTick->VAL = (480000L-1L); // reset count value

SysTick->CTRL = SysTick\_CTRL\_CLKSOURCE\_Msk | SysTick\_CTRL\_TICKINT\_Msk | SysTick\_CTRL\_ENABLE\_Msk;

}

**#ifdef** BLOCK\_WAITING

// Block waiting function abandoned for now

**void** **mdelay**(uint32\_t msec)

{

LOG\_DEBUG("Blocking wait for %d msec", msec);

uint32\_t i = 0;

uint32\_t delay\_count = 0;

**if** (msec == 500)

{

delay\_count = delay\_look\_up\_table[0];

}

**else** **if** (msec == 1000)

{

delay\_count = delay\_look\_up\_table[1];

}

**else** **if** (msec == 2000)

{

delay\_count = delay\_look\_up\_table[2];

}

**else** **if** (msec == 3000)

{

delay\_count = delay\_look\_up\_table[3];

}

**else**

{

LOG\_ERROR("Unexpected msec value, has to be 500, 1000, 2000, 3000");

}

**for**(i=0; i<delay\_count; i++);

}

**#else**

// Interrupt waiting function4

**void** mdelay(uint32\_t msec)

{

LOG\_DEBUG("Interrupt wait for %d msec", msec);

// read current count

target\_msec\_count = msec\_count + msec;

delay\_flag = true;

}

**#endif**

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

\* @brief

\* get the time had been run since powered up

\*

\* @note

\* Please #include config.h

\* Now only based on 1000 Hz clock,

\*

\* @return

\* runtime [in unit ms]

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

uint64\_t **timerGetRunTimeMilliseconds**(**void**)

{

**return** msec\_count;

}

**buffer .c**

**#include** "buffer.h"

errors\_t error;

errors\_t **initBuffer**(**struct** circular\_buffer \*cbuff){

error=*no\_error*;

cbuff->buff\_ptr= **malloc**(BUFFER\_SIZE \* **sizeof**(**char**));

**if** (!verifyValidPointer(cbuff)){

**for** (**int** i=0; i<BUFFER\_SIZE; i++){

cbuff->buff\_ptr[i]='0';

}

**if** (!verifyInit(cbuff)){

cbuff->head=0;

cbuff->tail=0;

cbuff->count=0;

}

}

**return** error;

}

errors\_t **verifyInit**(**struct** circular\_buffer \*cbuff){

error=*no\_error*;

**for** (**int** i=0; i<BUFFER\_SIZE; i++){

**if** (cbuff->buff\_ptr[i]!='0'){

error=*error\_buf\_init*;

}

}

**return** error;

}

errors\_t **verifyValidPointer**(**struct** circular\_buffer \*cbuff){

error=*no\_error*;

**if** (cbuff->buff\_ptr==NULL){

error=*error\_buff\_ptr*;

}

**return** error;

}

errors\_t **destroyBuffer**(**struct** circular\_buffer \*cbuff){

error=*no\_error*;

**free**(cbuff->buff\_ptr);

**return** error;

}

errors\_t **addItem**(**struct** circular\_buffer \*cbuff, **char** item){

error=*no\_error*;

**if** (!isFull(cbuff)){

cbuff->buff\_ptr[cbuff->head]=item;

cbuff->head=(cbuff->head+1)%BUFFER\_SIZE;

cbuff->count++;

}

**return** error;

}

errors\_t **removeItem**(**struct** circular\_buffer \*cbuff, **char** \*item){

error=*no\_error*;

**if** (!isEmpty(cbuff)){

\*item=cbuff->buff\_ptr[cbuff->tail];

cbuff->tail=(cbuff->tail+1)%BUFFER\_SIZE;

cbuff->count--;

}

**return** error;

}

errors\_t **isFull**(**struct** circular\_buffer \*cbuff){

error=*no\_error*;

**if** (cbuff->count==BUFFER\_SIZE){

error=*error\_full*;

}

**return** error;

}

errors\_t **isEmpty**(**struct** circular\_buffer \*cbuff){

error=*no\_error*;

**if** (cbuff->count==0){

error=*error\_empty*;

}

**return** error;

}

**gpio.c**

/\*

\* gpio.c

\*

\* Created on: Feb 11, 2020

\* Author: chutao

\*

\* Minic the functions from fsl\_gpio.c

\* Still use MKL25Z4.h for hardware addresses

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Include \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#include** <stdio.h>

**#include** <stdint.h>

**#include** "gpio.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Function \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **set\_GPIO\_Pinout**(GPIO\_Type \*port, uint32\_t pin)

{

port->PSOR = (0x1 << pin);

}

**void** **clear\_GPIO\_Pinout**(GPIO\_Type \*port, uint32\_t pin)

{

port->PCOR = (0x1 << pin);

}

**void** **toggle\_GPIO\_Pinout**(GPIO\_Type \*port, uint32\_t pin)

{

port->PTOR = (0x1 << pin);

}

**void** **init\_GPIO\_Pin**(GPIO\_Type \*port, uint32\_t pin,

gpio\_pin\_direct\_t pin\_direction, uint8\_t pin\_data)

{

**if** (pin\_direction == *GPIO\_DigitalInput*)

{

// Set pin to input direction

port->PDDR &= ~(0x1 << pin);

}

**else** **if** (pin\_direction == *GPIO\_DigitalOutput*)

{

// Set pin to output direction

port->PDDR |= (0x1 << pin);

**if** (pin\_data)

{

set\_GPIO\_Pinout(port,pin);

}

**else**

{

clear\_GPIO\_Pinout(port,pin);

}

}

**else**

{

**#ifdef** LOGGING\_DEBUG

// **TODO**: Debug message

**#endif**

}

}

**led.c**

**/\***

\* led.c

\*

\* Created on: Feb 11, 2020

\* Author: chutao

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Include \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#include** <logger.h>

**#include** <stdint.h>

**#include** "gpio.h"

**#include** "led.h"

**#include** "timer.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Global Varibles \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

led\_color\_t color = *red*;

**const** **char** \* led\_color\_string[3] ={"off","on","toggle"};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Function \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **init\_LED**(**void**)

{

init\_GPIO\_Pin(LED3\_RED\_PORT, LED3\_RED\_PIN, *GPIO\_DigitalOutput*, 1);

init\_GPIO\_Pin(LED3\_GREEN\_PORT, LED3\_GREEN\_PIN, *GPIO\_DigitalOutput*, 1);

init\_GPIO\_Pin(LED3\_BLUE\_PORT, LED3\_BLUE\_PIN, *GPIO\_DigitalOutput*, 1);

}

**void** **turn\_LED**(led\_state\_t LED\_state)

{

**if** (color == *red*)

{

turn\_LED\_red(LED\_state);

}

**else** **if** (color == *green*)

{

turn\_LED\_green(LED\_state);

}

**else** **if** (color == *blue*)

{

turn\_LED\_blue(LED\_state);

}

**else**

{

LOG\_ERROR("Unexpected led\_state\_t");

}

}

**void** **change\_LED\_color**(led\_color\_t LED\_color)

{

color = LED\_color;

}

**void** **turn\_LED\_red**(led\_state\_t LED\_state)

{

LOG\_DEBUG("Turn LED red %s",led\_color\_string[LED\_state]);

color = *red*;

**if** (LED\_state == *off*)

{

set\_GPIO\_Pinout(LED3\_RED\_PORT, LED3\_RED\_PIN);

}

**else** **if** (LED\_state == *on*)

{

clear\_GPIO\_Pinout(LED3\_RED\_PORT, LED3\_RED\_PIN);

}

**else** **if** (LED\_state == *toggle*)

{

toggle\_GPIO\_Pinout(LED3\_RED\_PORT, LED3\_RED\_PIN);

}

**else**

{

LOG\_ERROR("Unexpected led\_state\_t");

}

}

**void** **turn\_LED\_green**(led\_state\_t LED\_state)

{

LOG\_DEBUG("Turn LED green %s",led\_color\_string[LED\_state]);

color = *green*;

**if** (LED\_state == *off*)

{

set\_GPIO\_Pinout(LED3\_GREEN\_PORT, LED3\_GREEN\_PIN);

}

**else** **if** (LED\_state == *on*)

{

clear\_GPIO\_Pinout(LED3\_GREEN\_PORT, LED3\_GREEN\_PIN);

}

**else** **if** (LED\_state == *toggle*)

{

toggle\_GPIO\_Pinout(LED3\_GREEN\_PORT, LED3\_GREEN\_PIN);

}

**else**

{

LOG\_ERROR("Unexpected led\_state\_t");

}

}

**void** **turn\_LED\_blue**(led\_state\_t LED\_state)

{

LOG\_DEBUG("Turn LED blue %s",led\_color\_string[LED\_state]);

color = *blue*;

**if** (LED\_state == *off*)

{

set\_GPIO\_Pinout(LED3\_BLUE\_PORT, LED3\_BLUE\_PIN);

}

**else** **if** (LED\_state == *on*)

{

clear\_GPIO\_Pinout(LED3\_BLUE\_PORT, LED3\_BLUE\_PIN);

}

**else** **if** (LED\_state == *toggle*)

{

toggle\_GPIO\_Pinout(LED3\_BLUE\_PORT, LED3\_BLUE\_PIN);

}

**else**

{

LOG\_ERROR("Unexpected led\_state\_t");

}

}

**command\_parser.c**

**/\***

\* .c

\*

\* Created on: Mar 9, 2020

\* Author: Curry

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Include \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#include** "command\_parser.h"

**#include** <stdio.h>

**#include** <stdint.h>

**#include** <string.h>

**#include** <stdlib.h>

**#include** "led.h"

**#include** "logger.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Define \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#define** MAX\_USER\_BUF 64

**#define** MAX\_USER\_ARG 4

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Function \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **command\_parser**(**void**)

{

// Initialize char array

uint8\_t str[MAX\_USER\_BUF];

uint8\_t i = 0;

turn\_LED\_blue(*on*);

//takes all the characters until enter is pressed

**while**((str[i]=getchar())!='\n'){

//increment the index of the character array

i++;

// protect against long input

**if**(i == MAX\_USER\_BUF-1)

{

str[MAX\_USER\_BUF-1] = '\n';

LOG\_ERROR("user input too long, the rest of the string will be counted next time \n");

**break**;

}

}

turn\_LED\_blue(*off*);

i = 0;

uint8\_t char\_count[255];

**memset**(char\_count,0,255);

// go through the string

**while**(str[i]!='\n')

{

// count which character is incremented

char\_count[str[i]]++;

i++;

}

turn\_LED\_green(*on*);

// print them out

i = 'A';

**while**(i<='Z')

{

**if**(char\_count[i]!=0)

{

printf("%c - %u; ",i,char\_count[i]);

}

i++;

}

i = 'a';

**while**(i<='z')

{

**if**(char\_count[i]!=0)

{

printf("%c - %u; ",i,char\_count[i]);

}

i++;

}

printf("\n");

printf("\n");

turn\_LED\_green(*off*);

}

**test.c**

**/\***

\* test.c

\*

\* Created on: Apr 6, 2020

\* Author: user

\*/

**#include** "uCUnit.h"

**#include** "buffer.h"

//adapted from

//https://mcuoneclipse.com/2018/08/26/tutorial-%CE%BCcunit-a-unit-test-framework-for-microcontrollers/

**void** **Test**(**void**) {

**struct** circular\_buffer cbuff;

**char** item;

UCUNIT\_Init(); /\* initialize framework \*/

UCUNIT\_TestcaseBegin("Buffer Tests");

UCUNIT\_CheckIsEqual(0, initBuffer(&cbuff));

UCUNIT\_CheckIsEqual(0, destroyBuffer(&cbuff));

initBuffer(&cbuff);

cbuff.head=BUFFER\_SIZE-1;

cbuff.tail=BUFFER\_SIZE-1;

UCUNIT\_CheckIsEqual(0, addItem(&cbuff,'a'));

UCUNIT\_CheckIsEqual(0, cbuff.head);

UCUNIT\_CheckIsEqual(0, addItem(&cbuff,'b'));

UCUNIT\_CheckIsEqual(0, addItem(&cbuff,'c'));

UCUNIT\_CheckIsEqual(0, removeItem(&cbuff, &item));

printf("getitem: %s", item);

UCUNIT\_CheckIsEqual(0, cbuff.tail);

UCUNIT\_CheckIsEqual(0, removeItem(&cbuff, &item));

printf("getitem: %s", item);

UCUNIT\_CheckIsEqual(0, removeItem(&cbuff, &item));

printf("getitem: %s", item);

destroyBuffer(&cbuff);

initBuffer(&cbuff);

**for** (**int** i=0; i<BUFFER\_SIZE; i++){

addItem(&cbuff,'a');

}

UCUNIT\_CheckIsEqual(1U, addItem(&cbuff,'a'));

**for** (**int** i=0; i<BUFFER\_SIZE; i++){

printf("%d, %s", removeItem(&cbuff, &item),item);

}

UCUNIT\_CheckIsEqual(2U, removeItem(&cbuff, &item));

destroyBuffer(&cbuff);

UCUNIT\_TestcaseEnd();

/\* finish all the tests \*/

UCUNIT\_WriteSummary();

UCUNIT\_Shutdown();

}