/\*\*

\* \file

\*

\* \brief Empty user application template

\*

\*/

/\*\*

\* \mainpage User Application template doxygen documentation

\*

\* \par Empty user application template

\*

\* This is a bare minimum user application template.

\*

\* For documentation of the board, go \ref group\_common\_boards "here" for a link

\* to the board-specific documentation.

\*

\* \par Content

\*

\* -# Include the ASF header files (through asf.h)

\* -# Minimal main function that starts with a call to system\_init()

\* -# Basic usage of on-board LED and button

\* -# "Insert application code here" comment

\*

\*/

/\*

\* Include header files for all drivers that have been imported from

\* Atmel Software Framework (ASF).

\*/

/\*

\* Support and FAQ: visit <a href="https://www.microchip.com/support/">Microchip Support</a>

\*/

#include <asf.h>

#include "SerialConsole/SerialConsole.h"

#define CONF\_PWM\_MODULE TCC0

#define CONF\_PWM\_CHANNEL 2

#define CONF\_PWM\_OUTPUT 2

#define CONF\_PWM\_OUT\_PIN PIN\_PA10F\_TCC0\_WO2

#define CONF\_PWM\_OUT\_MUX MUX\_PA10F\_TCC0\_WO2

struct tcc\_module tcc\_instance;

struct tcc\_config config\_tcc;

static void configure\_tcc(uint32\_t newVal, uint32\_t period)

{

struct tcc\_config config\_tcc;

tcc\_get\_config\_defaults(&config\_tcc, CONF\_PWM\_MODULE);

config\_tcc.counter.period = period;

config\_tcc.compare.wave\_generation =

TCC\_WAVE\_GENERATION\_SINGLE\_SLOPE\_PWM;

config\_tcc.compare.match[CONF\_PWM\_CHANNEL] = newVal;

config\_tcc.pins.enable\_wave\_out\_pin[CONF\_PWM\_OUTPUT] = true;

config\_tcc.pins.wave\_out\_pin[CONF\_PWM\_OUTPUT] =

CONF\_PWM\_OUT\_PIN;

config\_tcc.pins.wave\_out\_pin\_mux[CONF\_PWM\_OUTPUT] =

CONF\_PWM\_OUT\_MUX;

tcc\_init(&tcc\_instance, CONF\_PWM\_MODULE, &config\_tcc);

tcc\_enable(&tcc\_instance);

}

#define MAX\_RX\_BUFFER\_LENGTH 5

volatile uint8\_t rx\_buffer[MAX\_RX\_BUFFER\_LENGTH];

volatile char buffer[7];

int main (void)

{

//Board Initialization -- Code that initializes the HW and happens only once

system\_init();

InitializeSerialConsole();

configure\_tcc(0xFFFFF/(2\*10.922)\*2, 0xFFFFF/1.0921);

SerialConsoleWriteString("ESE516 - CLI and Debug Logger\r\n"); //Order to add string to TX Buffer

delay\_init();

char string[] = "CLI starter code - ESE516\r\n";

/\*Simple DebugLogger Test\*/

setLogLevel(LOG\_INFO\_LVL);

LogMessage(LOG\_INFO\_LVL , "%s", string); //Test

setLogLevel(LOG\_ERROR\_LVL); //Sets the Debug Logger to only allow messages with LOG\_ERROR\_LVL or higher to be printed

LogMessage(LOG\_INFO\_LVL, "Performing Temperature Test…\r\n"); //This should NOT print

LogMessage(LOG\_FATAL\_LVL,"Error! Temperature over %d Degrees!\r\n", 55); //This should print

int pwmState = 0;

/\* This skeleton code simply sets the LED to the state of the button. \*/

while (1) {

/\* Is button pressed? \*/

if (port\_pin\_get\_input\_level(BUTTON\_0\_PIN) == BUTTON\_0\_ACTIVE) {

/\* Yes, so turn LED on. \*/

if (pwmState == 0)

{

for (int i = 0; i < 5; i++)

{

long newVal = 0xFFFFF/(2\*10.922)\*2 - (0xFFFFF/(2\*10.922) \* i/4);

tcc\_disable(&tcc\_instance);

delay\_ms(100);

configure\_tcc((uint32\_t) newVal, 0xFFFFF/1.0921);

pwmState = 1;

delay\_ms(1000);

}

} else {

for (int i = 4; i >= 0; i--)

{

long newVal = 0xFFFFF/(2\*10.922)\*2 - (0xFFFFF/(2\*10.922) \* i/4);

tcc\_disable(&tcc\_instance);

delay\_ms(100);

configure\_tcc((uint32\_t) newVal, 0xFFFFF/1.0921);

pwmState = 1;

delay\_ms(1000);

}

}

port\_pin\_set\_output\_level(LED\_0\_PIN, LED\_0\_ACTIVE);

} else {

/\* No, so turn LED off. \*/

port\_pin\_set\_output\_level(LED\_0\_PIN, !LED\_0\_ACTIVE);

}

//At the very end of the system, we tell the MCU to handle the text that is currently on the RX buffer

//Put a call to your state machine code that will handle the CLI by reading the data

//present on the RX buffer.

}

}