Jeffrey Steele

September 2024

3-2 Milestone Two: Enhancement One – Software Design and Engineering

In the revised version of the PWM control code, I focused on enhancing modularity, readability, and error handling to improve overall code management and robustness. To achieve this, I introduced a separate function called `setupPWM`, which is dedicated to initializing the PWM parameters, opening the PWM channel, and starting it. This function not only encapsulates the PWM setup logic but also simplifies the main thread by abstracting away the detailed setup steps. This approach makes the code more organized and easier to understand.

For error handling, I included a loop within the `setupPWM` function that continuously checks if the PWM handle returned is null, indicating a failure to open the PWM channel. If such a failure occurs, the system halts, preventing further execution with an uninitialized state, thereby safeguarding the application against unpredictable behaviors.

I defined all duty cycles, periods, and sleep times as constants at the start of the `mainThread` function. This not only clarifies their use throughout the code but also facilitates easier modifications and tuning of the PWM parameters. In the main control loop, I streamlined the logic to focus exclusively on toggling the PWM duty cycles between the predefined settings. This clarified the intended behavior of blinking LEDs and made the control sequence more straightforward to follow.

These modifications not only make the code cleaner and more maintainable but also enhance its stability by ensuring all components are correctly initialized and managed.

***Original Code:***

/\*

\* Copyright (c) 2015-2019, Texas Instruments Incorporated

\* All rights reserved.

\*

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

\* modification, are permitted provided that the following conditions

\* are met:

\*

\* \* Redistributions of source code must retain the above copyright

\* notice, this list of conditions and the following disclaimer.

\*

\* \* 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.

\*

\* \* Neither the name of Texas Instruments Incorporated 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 OWNER 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.

\*/

/\*

/\* For usleep() \*/

#include <unistd.h>

#include <stddef.h>

/\* Driver Header files \*/

#include <ti/drivers/PWM.h>

/\* Driver configuration \*/

#include "ti\_drivers\_config.h"

/\*

\* ======== mainThread ========

\* Task periodically increments the PWM duty for the on board LED.

\*/

void \*mainThread(void \*arg0)

{

/\* Period and duty in microseconds \*/

uint16\_t pwmPeriod = 3000;

uint16\_t pwm1DutyOn = 2700; // 90% of pwmPeriod (3000)

uint16\_t pwm1DutyOff = 0; // Completely off

uint16\_t pwm2DutyOn = 2700; // 90% of pwmPeriod (3000)

uint16\_t pwm2DutyOff = 300; // 10% of pwmPeriod (3000)

/\* Sleep time in microseconds \*/

uint32\_t time = 1000000; // Pause for 1 second (1 sec = 1000000 microsec)

PWM\_Handle pwm1 = NULL;

PWM\_Handle pwm2 = NULL;

PWM\_Params params;

/\* Call driver init functions. \*/

PWM\_init();

PWM\_Params\_init(&params);

params.dutyUnits = PWM\_DUTY\_US;

params.dutyValue = 0;

params.periodUnits = PWM\_PERIOD\_US;

params.periodValue = pwmPeriod;

pwm1 = PWM\_open(CONFIG\_PWM\_0, &params);

if (pwm1 == NULL) {

/\* CONFIG\_PWM\_0 did not open \*/

while (1);

}

PWM\_start(pwm1);

pwm2 = PWM\_open(CONFIG\_PWM\_1, &params);

if (pwm2 == NULL) {

/\* CONFIG\_PWM\_1 did not open \*/

while (1);

}

PWM\_start(pwm2);

/\* Loop forever incrementing the PWM duty \*/

while (1) {

// LEDs blink - switching between the yellow and green LEDs

PWM\_setDuty(pwm1, pwm1DutyOn); // Yellow on

PWM\_setDuty(pwm2, pwm2DutyOff); // Green off

usleep(time); // Pause

PWM\_setDuty(pwm1, pwm1DutyOff); // Yellow off

PWM\_setDuty(pwm2, pwm2DutyOn); // Green on

usleep(time); // Pause

}

}

***New Code:***

/\*

\* Copyright (c) 2015-2019, Texas Instruments Incorporated

\* All rights reserved.

\*

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

\* modification, are permitted provided that the following conditions

\* are met:

\*

\* \* Redistributions of source code must retain the above copyright

\* notice, this list of conditions and the following disclaimer.

\*

\* \* 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.

\*

\* \* Neither the name of Texas Instruments Incorporated 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 OWNER 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.

\*/

/\*

/\* For usleep() \*/

#include <unistd.h>

#include <stddef.h>

/\* Driver Header files \*/

#include <ti/drivers/PWM.h>

/\* Driver configuration \*/

#include "ti\_drivers\_config.h"

/\*

\* Function to initialize and start PWM.

\*/

PWM\_Handle setupPWM(uint32\_t pwmIndex, uint32\_t dutyCycle, uint16\_t period) {

PWM\_Handle pwm;

PWM\_Params params;

PWM\_Params\_init(&params);

params.dutyUnits = PWM\_DUTY\_US;

params.dutyValue = dutyCycle;

params.periodUnits = PWM\_PERIOD\_US;

params.periodValue = period;

pwm = PWM\_open(pwmIndex, &params);

if (pwm == NULL) {

/\* Handle PWM open error \*/

while(1);

}

PWM\_start(pwm);

return pwm;

}

/\*

\* ======== mainThread ========

\* Task periodically increments the PWM duty for the on board LED.

\*/

void \*mainThread(void \*arg0) {

const uint16\_t pwmPeriod = 3000; // PWM period in microseconds

const uint16\_t pwm1DutyOn = 2700; // 90% of pwmPeriod

const uint16\_t pwm2DutyOn = 2700; // 90% of pwmPeriod

const uint16\_t pwm1DutyOff = 0; // LED off

const uint16\_t pwm2DutyOff = 300; // 10% of pwmPeriod

const uint32\_t sleepTime = 1000000; // Pause for 1 second

/\* Initialize PWM \*/

PWM\_init();

/\* Setup PWM for two channels \*/

PWM\_Handle pwm1 = setupPWM(CONFIG\_PWM\_0, pwm1DutyOff, pwmPeriod);

PWM\_Handle pwm2 = setupPWM(CONFIG\_PWM\_1, pwm2DutyOff, pwmPeriod);

/\* Main control loop \*/

while (1) {

PWM\_setDuty(pwm1, pwm1DutyOn); // Turn PWM1 to 90%

PWM\_setDuty(pwm2, pwm2DutyOff); // Turn PWM2 off

usleep(sleepTime); // Pause

PWM\_setDuty(pwm1, pwm1DutyOff); // Turn PWM1 off

PWM\_setDuty(pwm2, pwm2DutyOn); // Turn PWM2 to 90%

usleep(sleepTime); // Pause

}

}