Jeffrey Steele

September 2024

4-2 Milestone Three: Enhancement Two - Algorithms and Data Structure

In revising the main function for an embedded system utilizing Texas Instruments' NoRTOS, I prioritized enhancements that bolster the system's robustness, efficiency, and maintainability. My primary focus was on instituting robust error handling mechanisms right after the board initialization. This change is critical as it ensures the system halts if hardware setup fails, thus preventing further execution that might rely on potentially uninitialized peripherals. Such a feature is essential for reliability, especially in production environments where failing gracefully is preferable to errant behavior.

To address potential logic errors during the development phase, I incorporated an assertion check to verify if the mainThread() unexpectedly returns—a scenario that should ideally never occur in a continuously operating embedded application. This assertion acts as a safety net, catching critical issues during testing before the software is deployed.

Recognizing the importance of power efficiency, particularly for battery-operated devices, I integrated a power management strategy. Instead of an idle loop consuming unnecessary processing power, I implemented a call to Power\_sleep(), which significantly reduces power consumption by putting the microcontroller into a low-power standby mode when inactive. This adaptation not only extends the operational lifespan of battery-powered applications but also aligns with best practices for energy-sensitive device programming.

I enhanced the clarity and usefulness of the code through comprehensive comments and structured documentation. These comments elucidate the purpose and functionality of each code segment, facilitating a deeper understanding for future developers, including those who might not be familiar with the project's nuances. This practice of detailed documentation is crucial for maintaining the code's accessibility and ease of use over time.

While the current code snippet remains relatively compact, I laid the groundwork for potential modularization. As the application scales, functions such as initialization and error handling can be segregated into distinct modules. This approach not only streamlines the codebase, making it cleaner and more navigable but also simplifies maintenance and future enhancements.

Through these targeted improvements, I've aimed to enhance the application's operational reliability, reduce its energy footprint, and ensure its long-term maintainability and scalability. This process has been a valuable exercise in applying best practices in embedded systems programming and in understanding the critical balance between functionality and system resources.

***Original Code:***

/\*

\* Copyright (c) 2017-2020, 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.

\*/

/\*

\* ======== main\_nortos.c ========

\*/

#include <stdint.h>

#include <stddef.h>

#include <NoRTOS.h>

#include <ti/drivers/Board.h>

extern void \*mainThread(void \*arg0);

/\*

\* ======== main ========

\*/

int main(void)

{

Board\_init();

/\* Start NoRTOS \*/

NoRTOS\_start();

/\* Call mainThread function \*/

mainThread(NULL);

while (1) {}

}

***New Code:***

/\*

\* Copyright (c) 2017-2020, 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.

\*/

/\*

\* ======== main\_nortos.c ========

\*/

#include <stdint.h>

#include <stddef.h>

#include <assert.h>

#include <ti/drivers/Power.h>

#include <ti/drivers/Board.h>

#include <NoRTOS.h>

extern void \*mainThread(void \*arg0);

/\*

\* Main entry point for NoRTOS-based applications. Initializes the board hardware,

\* starts the NoRTOS scheduler, and runs the main application thread.

\* The system is designed to run indefinitely; mainThread should not return.

\* A halt in the loop indicates unexpected behavior.

\*/

int main(void) {

// Initialize all configured peripherals

int initStatus = Board\_init();

if (initStatus != Board\_SOK) {

// Handle initialization error (e.g., halt, try to recover, log error)

while(1); // Halts on error

}

// Start NoRTOS environment

NoRTOS\_start();

// Run main application thread

void\* threadResult = mainThread(NULL);

// Assuming mainThread should never return

assert(threadResult == NULL); // Use assert to handle unexpected return

// Infinite loop to safely park the processor in low power mode

while (1) {

Power\_sleep(PowerCC26XX\_STANDBY); // Proper include and configuration needed

}

}

/\* 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

}

}