Jeffrey Steele

October 2024

5-2 Milestone Four: Enhancement Three - Databases

In the revised version of the main function for an embedded system utilizing Texas Instruments' NoRTOS, I introduced several critical enhancements aimed at bolstering system reliability, efficiency, and maintainability.

I implemented rigorous error handling right at the beginning with the board initialization process. By checking the return status of Board\_init(), I ensure that if any part of the hardware setup fails, the system immediately halts. This preemptive halt prevents further execution that might rely on potentially uninitialized peripherals, thereby safeguarding the system against unpredictable behaviors and hardware damage. Such a step is essential in embedded systems, where proper hardware initialization is crucial for stable operations.

I also incorporated an assert statement to enforce that the mainThread should never return under normal operational conditions. In embedded applications designed to run perpetually, the main thread's unexpected termination could signal a severe flaw. Utilizing assert helps detect such critical issues during the development phase, allowing for timely corrections before deployment.

Recognizing the importance of energy efficiency, especially in battery-powered devices, I replaced the previously idle loop with a call to Power\_sleep(). This function significantly reduces the microcontroller's power consumption when it's not actively managing tasks, thereby extending the device's operational life and adhering to energy conservation standards.

I then enhanced the clarity and usefulness of the code through comprehensive inline comments. Each segment of the code is now accompanied by detailed explanations of its purpose and the logic behind its implementation. These annotations are invaluable for future maintenance, making the code accessible and understandable to other developers, including those who might not be familiar with its intricacies.

With these improvements, the main function will be more robust, ensuring that it not only effectively manages the application's core operations but also does so in a way that is sustainable and considerate of system resources. This refined approach significantly enhances the reliability and maintainability of the application, demonstrating best practices in embedded system programming.

***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 <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

}

}

***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. This function is responsible for

\* initializing the hardware, starting the NoRTOS environment, and ensuring continuous

\* operation through the main application thread. It employs strict error handling and

\* power management to enhance reliability and efficiency.

\*/

int main(void) {

// Initialize all configured peripherals

int initStatus = Board\_init();

if (initStatus != Board\_SOK) {

// Log initialization failure if possible and halt

// Log\_error("Board initialization failed");

while(1); // Halts on initialization error

}

// Start the NoRTOS environment

NoRTOS\_start();

// Run the main application thread

void\* threadResult = mainThread(NULL);

// Assuming mainThread should never return, enforce this assumption with an assert

assert(threadResult == NULL); // Use assert to catch unexpected behavior during development

// Implement a safer infinite loop. If somehow mainThread returns, the system should enter a low-power state.

while (1) {

// Enter low-power standby mode when not processing

Power\_sleep(PowerCC26XX\_STANDBY); // Replace PowerCC26XX\_STANDBY with the appropriate power mode for your hardware

}

}