

# CPE403/ECG603/ECG710 – Advanced

## Embedded Systems/Real-Time Embedded Systems

### Design Assignment 4

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

Name: Joshua Martinez

Email: [marti87@unlv.nevada.edu](mailto:marti87@unlv.nevada.edu)

Github Repository link (root): [https://github.com/JoshuaMa2003/submission\\_da\\_0011](https://github.com/JoshuaMa2003/submission_da_0011)

Youtube Playlist link (root):

[https://www.youtube.com/playlist?list=PLZ09MA\\_uFt61Vw2JCAtw-8vAdmoCKXpZ](https://www.youtube.com/playlist?list=PLZ09MA_uFt61Vw2JCAtw-8vAdmoCKXpZ)

**Follow the submission guideline to be awarded points for this Assignment.**

Submit the following for all Assignments:

1. In the document, for each task submit the modified or included code (from the base code) with highlights and justifications of the modifications. Also include comments. If no base code is provided, submit the base code for the first task only.
2. Create a private Github repository with a random name (no CPE403/603/710, Lastname, Firstname). Place all labs under the root folder MSP432E4/CC1352, sub-folder named Assignment1, with one document and one video link file for each lab, place modified c files named as asng\_taskxx.c.
3. If multiple c files or other libraries are used, create a folder asng1\_t01 and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) with startup\_ccs.c and other include files, c) text file with youtube video links (see template).
5. Submit the doc file in canvas before the due date. The root folder of the github assignment directory should have the documentation and the text file with youtube video links.
6. Organize your videos as a playlist under the name “ADVEMBSYS”. The playlist should

have the video sequence arranged as submission or due dates.

7. Only submit the PDF. Upload this document in the github repository and in canvas.

1. **Code for tasks: For each task, submit the relevant modified/section or included code (from the base code) with highlights and justifications of the modifications. Also include the comments. If no base code is provided, submit the initialization and execution section of each task separately. Use a separate page for each task.**

### **Task 1 – Base Project Zero Application (Unmodified Reference)**

- **Original Behavior:**
  - The original *project\_zero.c* comes from TI's Project Zero example.
  - It already:
    - Initializes GAP, GATT, Device Information Service.
    - Adds three Bluetooth services:
      1. LED Service
      2. Button Service
      3. Data Service
    - Registers callbacks for:
      1. LED Writes
      2. Button notifications config
      3. Data Services writes / CCCD
    - Handles buttons presses locally by updating the Button Service characteristic in *ProjectZero\_handleButtonPress()* and letting the BLE stack send notifications.
  - No ADC or joystick code exists in the original file; all hardware interactions are just two LEDs and two push-buttons.
- **Unchanged Base Code:**
  - Service registrations in *ProjectZero\_init*:

```
LedService_AddService(selfEntity);
ButtonService_AddService(selfEntity);
DataService_AddService(selfEntity);
```
  - Callback Registration:

```
LedService_RegisterAppCBs(&ProjectZero_LED_ServiceCBs);
ButtonService_RegisterAppCBs(&ProjectZero_Button_ServiceCBs);
DataService_RegisterAppCBs(&ProjectZero_Data_ServiceCBs);
```

## **Task 2 – Add an ADC-based Joystick Input**

### **2.1 SysConfig / Hardware Changes**

- In SysConfig → ADC, I enabled a single ADC instance:

- Name: Config\_ADC\_0
- Reference Source: Fixed
- Reference Voltage: 3300000 uV
- Sampling Duration: 2.7 uS
- ADC Peripheral: Any (ADC0)
- ADC Pin: Any (DI023/2 Header)

- In TIDrivers section, I made sure the ADC Driver was enabled

### **2.2 Code Changes: Initialize and Read ADC**

1. Included the ADC driver header at the top with the other TIDriver Includes:

```
#include <ti/drivers/ADC.h>
#include "ti_drivers_config.h"
```

2. Added global variables for the joystick ADC:

```
static ADC_Handle    adcJoystickHandle;
static ADC_Params    adcJoystickParams;
static uint16_t       adcJoystickRaw = 0;
```

3. In ProjectZero\_init(), after the button GPIO initialization, I opened the ADC:

```

// --- Joystick ADC initialization ---
ADC_init();
ADC_Params_init(&adcJoystickParams);
adcJoystickParams.isProtected = false;
adcJoystickHandle = ADC_open(CONFIG_ADC_0, &adcJoystickParams);

if (adcJoystickHandle == NULL)
{
    Log_error0("Error initializing joystick ADC (CONFIG_ADC_0)");
}
else
{
    Log_info0("Joystick ADC initialized on CONFIG_ADC_0");
}

```

**4. Created a small helper function to read the joystick:**

```

static void ProjectZero_readJoystick(void)
{
    int_fast16_t res;

    if (adcJoystickHandle == NULL)
    {
        return;
    }

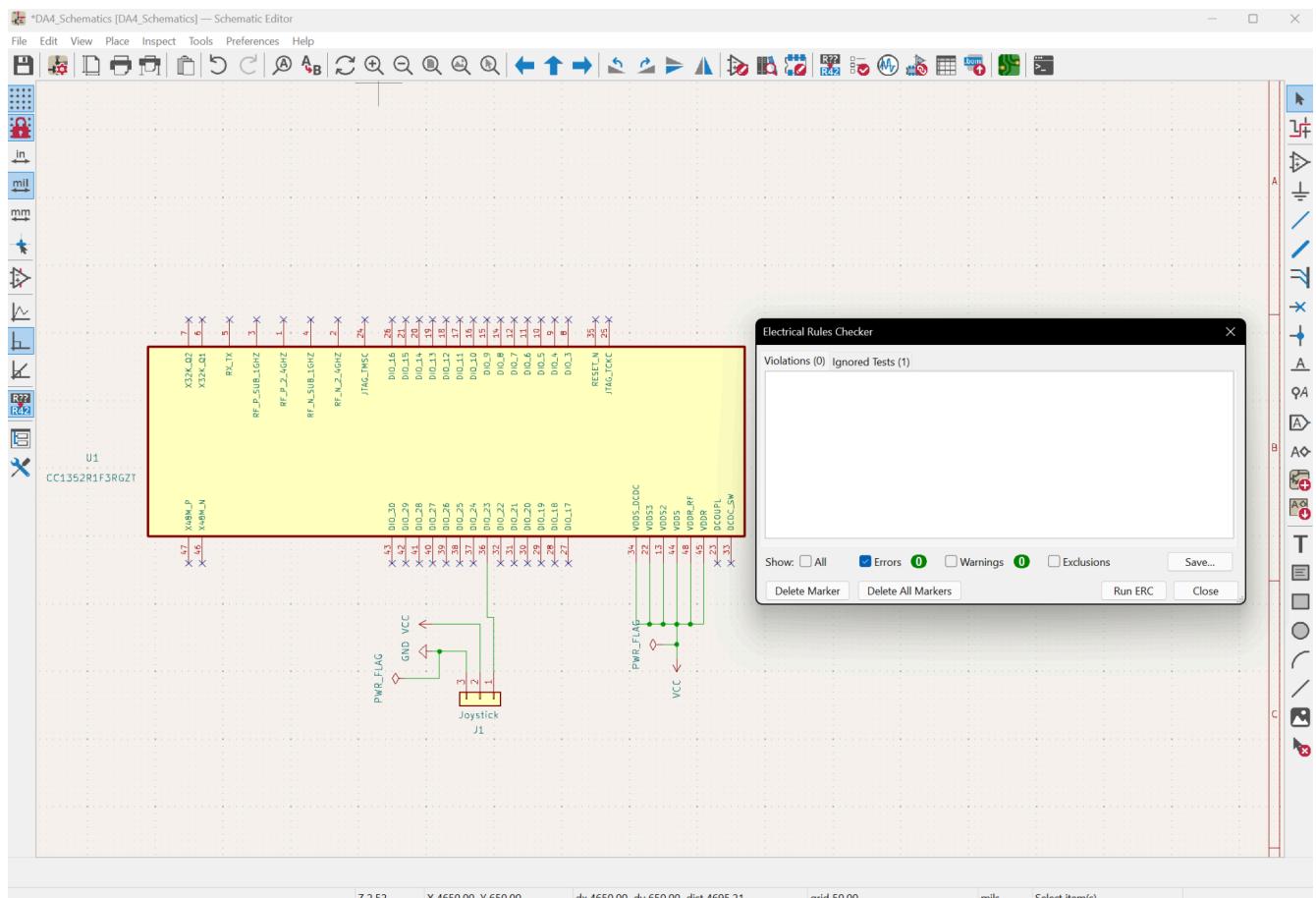
    res = ADC_convert(adcJoystickHandle, &adcJoystickRaw);

    if (res == ADC_STATUS_SUCCESS)
    {
        // adcJoystickRaw now holds a 12-bit value 0-4095
        Log_info1("Joystick ADC raw = %d", adcJoystickRaw);
    }
    else
    {
        Log_error1("ADC_convert failed (%d)", res);
    }
}

```

**2. Block diagram and/or Schematics showing the components, pins used, and interface. You can use KiCAD/Eagle/Altium to get the schematics. KiCAD Symbol libraries for TI uCs are @ [https://kicad.github.io/symbols/MCU\\_Texas.html](https://kicad.github.io/symbols/MCU_Texas.html) and <https://www.snapeda.com/part/CC1352P1F3RGZT/Texas%20Instruments/view-part/>**

### **KiCad Design Schematics (ERC Compliant):**



**3. Screenshots of the IDE, physical setup, debugging process – Provide screenshot of successful compilation, screenshots of registers, variables, graphs, etc.**

**IDE Workspace:**

The screenshot shows the Code Composer Studio IDE interface with the following windows open:

- Project Explorer:** Shows the project structure for "CPE403\_DA4" including source files like "main.c", header files like "project\_zero.h", and configuration files like "Board.html".
- Debug:** Displays the connection to a Texas Instruments XDS110 USB Debug Probe/Cortex\_M4\_0 (SUSPENDED). It shows memory dump information at address 0x0000100A4, including the value of variable "\_c\_int00" at boot\_cortex.m:64 (0x0001CA40).
- Variables:** Shows a table with one entry: "params" of type struct UART2\_Params with value "readMode=UART2..." at location 0x20013FFC.
- Registers:** A tab labeled "Registers" is visible in the top right of the main window area.
- Code Editor:** The main code editor window displays the "main.c" file with the following content:

```

109 extern void AssertHandler(uint8_t assertCause,
110                           uint8_t assertSubcause);
111
112 //*****
113 * @fn      Main
114 *
115 * @brief    Application Main
116 *
117 * @input parameters
118 *
119 * @param     None.
120 *
121 * @output parameters
122 *
123 * @param     None.
124 *
125 * @return    None.
126 */
127 int main()
128 {
129     /* Register Application callback to trap asserts raised in the Stack */
130     RegisterAssertCb(AssertHandler);
131
132     Board_initGeneral();
133
134 #if !defined( POWER_SAVING )
135     /* Set constraints for Standby, powerdown and idle mode */
136     // PowerCC26XX_SB_DISALLOW may be redundant
137     Power_setConstraint(PowerCC26XX_SB_DISALLOW);
138     Power_setConstraint(PowerCC26XX_IDLE_PD_DISALLOW);
139 #endif // POWER_SAVING
140
141     /* Update User Configuration of the stack */
142     user0Cfg.appServiceInfo->timerTickPeriod = Clock_tickPeriod;
143     user0Cfg.appServiceInfo->timerMaxMillisecond = ICall_getMaxMsecs();
144
145     /* Initialize the RTOS Log formatting and output to UART in Idle thread.
146      * Note: Define xdc_runtime_Log_DISABLE_ALL and remove define UARTLOG_ENABLE
147      *       to remove all impact of Log statements.
148      * Note: NULL as Params gives 115200,8,N,1 and Blocking mode */
149
150     // Initialize UART2 parameters
151     UART2_Params params;
152     UART2_Params_init(&params);

```

**Terminal:** Shows the output of the build process:

```

Cortex_M4_0: GEL Output: Memory Map Initialization Complete.
Cortex_M4_0: GEL Output: Memory Map Initialization Complete.
Cortex_M4_0: GEL Output: Board Reset Complete.

```

**Physical Setup:**



## Debugging Process:

The screenshot shows the Code Composer Studio interface during the debugging process of a project named "CPE403\_DA4".

- Project Explorer:** Shows the project structure with files like CPE403\_DA1, CPE403\_DA2, CPE403\_DA3, and CPE403\_DA4 (selected). It also lists Generated Source, Binaries, Includes, Application (containing project\_zero.c, project\_zero.h, rcosc\_calibration.c, rcosc\_calibration.h, sunlightService.c, sunlightService.h, temperatureService.c, temperatureService.h), common, Drivers, Call, CallBLE, Include, OAD, Profiles, Release, Startup, targetConfigs, cc13x2\_cc26x2\_app.tirtos7.cmd, Board.html, project\_zero.syscfg, and README.html.
- Debug View:** Displays assembly code for Cortex\_M4\_0. The current assembly instruction is:

```
109 extern void AssertHandler(uint8_t assertCause,  
110                           uint8_t assertSubcause);  
111  
112 ****  
113 * @fn      Main  
114 *  
115 * @brief    Application Main  
116 *  
117 * input parameters  
118 *  
119 * @param    None.  
120 *  
121 * output parameters  
122 *  
123 * @param    None.  
124 *  
125 * @return   None.  
126 */  
127 int main()  
128 {  
129     /* Register Application callback to trap asserts raised in the Stack */  
130     RegisterAssertCback(AssertHandler);  
131  
132     Board_initGeneral();  
133  
134 #if !defined( POWER_SAVING )  
135     /* Set constraints for Standby, powerdown and idle mode */  
136     // PowerCC26XX_SB_Disallow may be redundant  
137     Power_setConstraint(PowerCC26XX_SB_Disallow);  
138     Power_setConstraint(PowerCC26XX_IDLE_PD_Disallow);  
139#endif // POWER_SAVING  
140  
141     /* Update User Configuration of the stack */  
142     user0Cfg.appServiceInfo->timerTickPeriod = Clock_tickPeriod;  
143     user0Cfg.appServiceInfo->timerMaxMillisecond = ICall_getMaxMSecs();  
144  
145     /* Initialize the RTOS Log formatting and output to UART in Idle thread.  
146     * Note: Define xdc_runtime_log_DISABLE_ALL and remove define UARTLOG_ENABLE  
147     * to remove all impact of Log statements.  
148     * Note: NULL as Params gives 115200,8,N,1 and Blocking mode */  
149  
150     // Initialize UART2 parameters  
151     UART2_Params params;  
152     UART2_Params_init(&params);
```
- Variables View:** Shows a table with one entry: Name (params), Type (struct UART2\_Params), Value (readMode=UART2..., 0x20013FFC), and Location.
- Terminal View:** Displays the following log output:

```
Cortex_M4_0: GEL Output: Memory Map Initialization Complete.  
Cortex_M4_0: GEL Output: Memory Map Initialization Complete.  
Cortex_M4_0: GEL Output: Board Reset Complete.
```

## Successful Compilation:

The screenshot shows the Code Composer Studio interface with the following windows visible:

- Project Explorer**: Shows the project structure with several source files like CPE403\_DA1.c, CPE403\_DA2.c, CPE403\_DA3.c, and CPE403\_DA4.c.
- Debug**: Displays the code for CPE403\_DA4, specifically the main loop handling messages. It includes comments and annotations for the SunlightService and TemperatureService.
- Variables**: Shows a table for variables, currently empty.
- Terminal**: Shows the build logs:

```
**** Build of configuration Release for project CPE403_DA4 ****
C:\ti\ccs1281\ccs\utils\bin\gmake" -k -j 24 all -O
gmake[1]: 'CPE403_DA4.out' is up to date.
**** Build Finished ****
```

## **Fixed Identification Message:**

The screenshot shows a mobile application interface for managing a Bluetooth characteristic. At the top, there is a red header bar with the time "3:42" and signal strength indicators. Below the header, the word "Characteristic" is centered above a list of items.

**Service UUID:**  
F0001130-0451-4000-B000-000000000000

Format: **UTF-8**

**F0001131-0451-4000-B000-000000000000**  
Properties: Read Write

**Write**

---

**Read**

**CC1352R\_Device\_0**  
15:42:15

**F0001132-0451-4000-B000-000000000000**  
Properties: WriteNoRsp Notify

**Write**

---

**Notifications**  Enable

---

**4. Declaration**

**I understand the Student Academic Misconduct Policy -**  
**<http://studentconduct.unlv.edu/misconduct/policy.html>**

“This assignment submission is my own, original work”.

Joshua Martinez