UCSD Extension

Embedded Controller Programming for Real-Time Systems

Course Number: ECE-40097  
Section ID: 145032

Final Project

Date: 3/17/2020

Chris Isabelle

christopher.j.isabelle@gmail.com

SID: U01136665

Contents

[Project Requirements 3](#_Toc33486459)

[Summary 4](#_Toc33486460)

[Code 5](#_Toc33486461)

[UART 5](#_Toc33486462)

[GPIO Callback 5](#_Toc33486463)

[numOnes 6](#_Toc33486464)

[sumOfSquares 6](#_Toc33486465)

[Process Character 7](#_Toc33486466)

[Test Data 8](#_Toc33486467)

[Main Menu 8](#_Toc33486468)

[Test g 8](#_Toc33486469)

[Test b 8](#_Toc33486470)

[Test Blue <USER> Button – GPIO interrupt callback 9](#_Toc33486471)

[Test d – Disable GPIO Interrupt 9](#_Toc33486472)

[Test e – Enable GPIO Interrupt 9](#_Toc33486473)

[Test v – Sum of Squares 10](#_Toc33486474)

[Test n – Number of Ones counter 11](#_Toc33486475)

[Test a – Disable Interrupts 11](#_Toc33486476)

[Test Reset 12](#_Toc33486477)

# Project Requirements

|  |  |  |
| --- | --- | --- |
| Requirement ID | Description | Compliance |
| **P** | **Mid-term prerequisites** |  |
| P-1 | Use Cube32MX to generate code without UART interrupt. | ✓ |
| P-2 | Expand generated code to use UART in **Interrupt** mode. | ✓ |
| P-3 | Enable GPIO PIN 13 interrupt, used for blue user switch using handler EXTI15\_10\_IRQHandler() | ✓ |
| P-4 | Enable Timer 2 and Timer 3, Timer 6 interrupts | ✓ |
| P-5 | Enable both IWDG and WWDG watchdog timers | ✓ |
| P-6 | Enable RTC | ✓ |
| **R1** | **UART Interrupt, (1 point)** |  |
| R1-1 | Implement logMsg() method to display a character on the terminal using interrupt. | ✓ |
| R1-2 | Implement logGetMsg() method to receive character from terminal using interrupt. | ✓ |
| R1-3 | If ‘g’ is entered on the terminal, toggle the green LED after a 1 second delay. | ✓ |
| R1-4 | If ‘b’ is entered on the terminal, toggle the red LED after a 1 second delay. | ✓ |
| R1-5 | For any other character, print “unknown character received”. | ✓ |
| R1-6 | Comment out MX\_IWDG\_Init(), MX\_WWDG\_Init() & MX\_RTC\_Init() | ✓ |
| **R2** | **Create a software interrupt and use one of the non-used IRQ, (5 points)** |  |
| R2-1 | Use FMC\_IRQn = 48 //FMC Global Interrupt | ✓ |
| R2-2 | Enable FMC\_IRQa in MX\_GPIO\_Init() | ✓ |
| R2-3 | Create a menu item ‘s’ which will generate a software interrupt and print “SW Interrupt detected” | ✓ |
| R2-3a | When ‘s’ is selected enable STIR using FMC\_IRQn | ✓ |
| R2-3b | ISR callback function is called | ✓ |
| R2-3c | Use a global flag to let main menu display the required message | ✓ |
| **R3** | **Create the method myDelay1() using timer2 with input in msec, (5 points)** |  |
| R3-1 | Use myDelay1() as the delay method for 1 second delay requirement in R1-3 | ✓ |
| **R4** | **Create the method myDelay2() using SysTick with input in msec, (5 points)** |  |
| R4-1 | Use myDelay2() as the delay method for 1 second delay requirement in R1-4 | ✓ |
| **R5** | **Use Timer 3 to count events, (6 points)** |  |
| R5-1 | Program the timer in MX\_TIM3\_Init() so that it will expire every 1 second. |  |
| R5-2 | Create a new menu item ‘t’ to start timer3 |  |
| R5-3 | Implement HAL\_TIM\_PeriodElapsedCallback() to set a flag indicating that the 1 sec timer has elapsed. |  |
| R5-4 | In the main loop count 10 events and stop timer3 after 10 iterations |  |
| R5-5 | Print the log message, “Total counted timer3 events = %d\r\n” |  |
| **R6** | **Test IWDG, (6 points)** |  |
| R6-1 | Program prescaler, window and reload value for a timeout of 500 msec. | ✓ |
| R6-2 | Pet the watchdog in main() | ✓ |
| R6-3 | Verify not watchdog event and no reset | ✓ |
| R6-4 | Create a new menu item ‘w’ that will add a 1sec delay and trip the watchdog. | ✓ |
| R6-2 | Verify that the device resets due to the watchdog timeout. | ✓ |
| **R7** | **Test MX\_RTC, (6 points)** |  |
| R7-1 | Set the alarm for hour 0, minute 1 |  |
| R7-2 | Implement a callback to set the flag and detect the alarm. |  |
| R7-3 | Display the message “RTC alarm A detected” |  |
| R7-4 | Verify that message is printed 1 minute following power on or reset. |  |

Summary

Difference between a blocking UART TX and an interrupt UART TX is that the logMsg() method had to have a blocking poll for UART Transfer Complete (UART\_FLAG\_TC).

Was unable to resolve STM32CubeMX assembler directives for having a separate .s file. Implemented sumOfSquares as an in-line assembly function.

Added a special case to the input char process code that prompted and captured a user input for the sumOfSquares method.

Used a random number generator to determine the input 32 bit word that is sent to the numOnes() method.

Excellent Project – It provided great exposure to assembly and interrupts. Sincere Thanks!

# Code

## UART

**void** **logMsg**(UART\_HandleTypeDef \*huart, **char** \_out[])

{

//Use STM32L4xx\_HAL\_Driver for UART TX

uint32\_t numTxChar = **strlen**(\_out);

HAL\_UART\_Transmit\_IT(huart, (uint8\_t \*)\_out, numTxChar);

//need to stall and wait for all characters to be transmitted

**while**(!\_\_HAL\_UART\_GET\_FLAG(huart, UART\_FLAG\_TC));

}

// logGetMsg method

uint8\_t **logGetMsg**(UART\_HandleTypeDef \*huart)

{

uint8\_t rxChar;

HAL\_UART\_Receive\_IT(&huart1, &rxChar, **sizeof**(uint8\_t));

**return** (rxChar);

}

// generic USART handler

**void** **USART1\_IRQHandler**(**void**)

{

HAL\_UART\_IRQHandler(&huart1);

}

## GPIO Callback

//C callback function to trap interupt for Blue <USER> button press

**void** **HAL\_GPIO\_EXTI\_Callback**(uint16\_t GPIO\_Pin)

{

**if**(GPIO\_Pin == BUTTON\_EXTI13\_Pin)

{

logMsg(&huart1, "\n\r\nBlue Button Pressed \r\n");

}

}

## numOnes

//numOnes assembly implementation

**int** **numOnes**(**int** i)

{

**int** res = 0;

**\_\_asm** ( //r0 holds input variable i

"MOV r1, #1\n\t" //Initialize r1 to 1 - r1 holds the bit mask

"MOV r3, #0\n\t" //Initialize r3 to 0 - r3 holds the bit count

"LoopBitChecker:\n\t" //Start of LoopBitChecker:

"AND r2, r0, r1\n\t" //Bit-wise AND of r0 and r1 result goes to r2

"CMP r1, r2\n\t" //Compare the result if there's a 1 in that bit position in r0

"BEQ IncrementCount\n\t" //Branch to Increment Count

"B ShiftBitMask\n\t" //Else Branch to Shift mask

"IncrementCount:\n\t" //Start IncrementeCount:

"ADD r3, #1\n\t" //Increment r3 which holds the resultant bit count

"ShiftBitMask:\n\t" //Start of ShiftBitMask:

"LSL r1, #1\n\t" //Shift the bit mask left one step

"CMP r1, #0\n\t" //Compare the bit mask to 0

"BEQ DoneNumOnes\n\t" //If bit mask is zero, then all bits have been tested, branch to Done

"B LoopBitChecker\n\t" //Else repeat bit checker loop

"DoneNumOnes:\n\t" //Start of DoneNumOnes:

"MOV r0, r3\n\t" //Move the final result to r0

"BX lr \n\t" //Branch terminate

);

**return**(res);

}

## sumOfSquares

//sumOfSquares assembly implementation

**int** **sumOfSquares**(**int** i)

{

**int** res = 0;

**\_\_asm** ( //r0 holds input variable i

"MOV r1, #2\n\t" //Initialize r1 to 2 - r1 holds the root to be squared

"MOV r2, #0\n\t" //Initialize r2 to 0 - r2 holds the sum of the squares

"LoopSquareAndAccumulate:\n\t" //Start of LoopSquareAndAccumulate:

"MLA r2, r1, r1, r2\n\t" //Add r1^2 to r2

"CMP r0, r1\n\t" //Compare r1 to r0

"BEQ DoneSumOfSquares\n\t" //If equal branch to Done

"ADD r1, #1\n\t" //Increment root to be squared

"B LoopSquareAndAccumulate\n\t" //Repeat square and accumulate

"DoneSumOfSquares:\n\t" //Start of DoneSumOfSquares:

"MOV r0, r2\n\t" //Move the final result to r0

"BX lr \n\t" //Branch terminate

);

**return**(res);

}

## Process Character

//process character

**switch**(rxChar)

{

**case**('g'): logMsg(&huart1, "\n\r\nreceived character = g \r\n");

HAL\_GPIO\_TogglePin(GRN\_LED\_GPIO\_Port, GRN\_LED\_Pin);

logMsg(&huart1, "Toggle Green LED \r\n");

**break**;

**case**('b'): logMsg(&huart1, "\n\r\nreceived character = b \r\n");

HAL\_GPIO\_TogglePin(BLUE\_LED\_GPIO\_Port, BLUE\_LED\_Pin);

logMsg(&huart1, "Toggle Blue LED \r\n");

**break**;

**case**('v'): logMsg(&huart1, "\n\r\nreceived character = v \r\n");

squareSumInProcess=1;

**break**;

**case**('n'): logMsg(&huart1, "\n\r\nreceived character = n \r\n");

//generate random number

**int** randomNum = **rand**()\*0xffffffff;

**sprintf**(tempStr, "calling numOnes() with a random number = 0x%08x\r\n", randomNum);

logMsg(&huart1, tempStr);

//call numOnes()

**int** result = numOnes(randomNum);

**sprintf**(tempStr, "result = %i ones counted\r\n", result);

logMsg(&huart1, tempStr);

**break**;

**case**('d'): logMsg(&huart1, "\n\r\nreceived character = d \r\n");

//Use ICER to clear interrupt EXTI15\_10\_IRQn

//EXTI15\_10\_IRQn = 40

NVIC->ICER[(40 >> 5)] = (1 << (40 & 0x1F));

logMsg(&huart1, "Disabled Blue <USER> push button interrupt \r\n");

**break**;

**case**('e'): logMsg(&huart1, "\n\r\nreceived character = e \r\n");

//Use ISER to set interrupt EXTI15\_10\_IRQn

//EXTI15\_10\_IRQn = 40

NVIC->ISER[(40 >> 5)] = (1 << (40 & 0x1F));

logMsg(&huart1, "Enabled Blue <USER> push button interrupt \r\n");

**break**;

**case**('a'): logMsg(&huart1, "\n\r\nreceived character = a \r\n");

logMsg(&huart1, "Disabling all interrupts \r\n");

logMsg(&huart1, "Depress <RESET> push button to restore functionality \r\n");

//suspend all interrupts (note this is not portable to M0!)

\_\_ASM **volatile** ("cpsid i");

**break**;

**default**: **if**(squareSumInProcess==1)

{

squareSumInProcess=0;

**if**(isdigit(rxChar))

{

**sprintf**(tempStr, "Sum of squares to n=%i is ", **atoi**((**char** \*)&rxChar));

logMsg(&huart1, tempStr);

**int** summation=sumOfSquares(**atoi**((**char** \*)&rxChar));

**sprintf**(tempStr, "%i\r\n", summation);

logMsg(&huart1, tempStr);

**break**;

}

}

logMsg(&huart1, "\n\r\nunknown character received \r\n");

}

//loop

}

# Test Data

## Main Menu

Welcome to <Embedded Controller Programming for Real-Time Systems>

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~ Assignment: Mid-Term Project ~

~ Course Number: ECE-40097 ~

~ Section ID: 145032 ~

~ Student Name: Chris Isabelle ~

~ SID: U01136665 ~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Main Menu

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter g - Toggle the Green LED

Enter b - Toggle the Blue LED

Enter v - Find the sum of squares of a number entered into the terminal

Enter n - Find number of set bits in a random integer

Enter d - Disable the Blue <USER> push button switch interrupt

Enter e - Enable the Blue <USER> push button switch interrupt

Enter a - Disable all interrupts.

Depress Blue <USER> push button to test GPIO interrupt

## Test g

received character = g

Toggle Green LED

Main Menu

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter g - Toggle the Green LED

Enter b - Toggle the Blue LED

Enter v - Find the sum of squares of a number entered into the terminal

Enter n - Find number of set bits in a random integer

Enter d - Disable the Blue <USER> push button switch interrupt

Enter e - Enable the Blue <USER> push button switch interrupt

Enter a - Disable all interrupts.

Depress Blue <USER> push button to test GPIO interrupt

received character = g

Toggle Green LED

Main Menu

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter g - Toggle the Green LED

Enter b - Toggle the Blue LED

Enter v - Find the sum of squares of a number entered into the terminal

Enter n - Find number of set bits in a random integer

Enter d - Disable the Blue <USER> push button switch interrupt

Enter e - Enable the Blue <USER> push button switch interrupt

Enter a - Disable all interrupts.

Depress Blue <USER> push button to test GPIO interrupt

## Test b

received character = b

Toggle Blue LED

Main Menu

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter g - Toggle the Green LED

Enter b - Toggle the Blue LED

Enter v - Find the sum of squares of a number entered into the terminal

Enter n - Find number of set bits in a random integer

Enter d - Disable the Blue <USER> push button switch interrupt

Enter e - Enable the Blue <USER> push button switch interrupt

Enter a - Disable all interrupts.

Depress Blue <USER> push button to test GPIO interrupt

received character = b

Toggle Blue LED

Main Menu

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter g - Toggle the Green LED

Enter b - Toggle the Blue LED

Enter v - Find the sum of squares of a number entered into the terminal

Enter n - Find number of set bits in a random integer

Enter d - Disable the Blue <USER> push button switch interrupt

Enter e - Enable the Blue <USER> push button switch interrupt

Enter a - Disable all interrupts.

Depress Blue <USER> push button to test GPIO interrupt

## Test Blue <USER> Button – GPIO interrupt callback

Blue Button Pressed

Blue Button Pressed

## Test d – Disable GPIO Interrupt

received character = d

Disabled Blue <USER> push button interrupt

Main Menu

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter g - Toggle the Green LED

Enter b - Toggle the Blue LED

Enter v - Find the sum of squares of a number entered into the terminal

Enter n - Find number of set bits in a random integer

Enter d - Disable the Blue <USER> push button switch interrupt

Enter e - Enable the Blue <USER> push button switch interrupt

Enter a - Disable all interrupts.

Depress Blue <USER> push button to test GPIO interrupt

## Test e – Enable GPIO Interrupt

received character = e

Blue Button Pressed

Main Menu

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter g - Toggle the Green LED

Enter b - Toggle the Blue LED

Enter v - Find the sum of squares of a number entered into the terminal

Enter n - Find number of set bits in a random integer

Enter d - Disable the Blue <USER> push button switch interrupt

Enter e - Enable the Blue <USER> push button switch interrupt

Enter a - Disable all interrupts.

Depress Blue <USER> push button to test GPIO interrupt

Blue Button Pressed

Main Menu

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter g - Toggle the Green LED

Enter b - Toggle the Blue LED

Enter v - Find the sum of squares of a number entered into the terminal

Enter n - Find number of set bits in a random integer

Enter d - Disable the Blue <USER> push button switch interrupt

Enter e - Enable the Blue <USER> push button switch interrupt

Enter a - Disable all interrupts.

Depress Blue <USER> push button to test GPIO interrupt

## Test v – Sum of Squares

received character = v

Square Sum selected, enter an integer between 0 and 9

Sum of squares to n=3 is 13

Main Menu

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter g - Toggle the Green LED

Enter b - Toggle the Blue LED

Enter v - Find the sum of squares of a number entered into the terminal

Enter n - Find number of set bits in a random integer

Enter d - Disable the Blue <USER> push button switch interrupt

Enter e - Enable the Blue <USER> push button switch interrupt

Enter a - Disable all interrupts.

Depress Blue <USER> push button to test GPIO interrupt

received character = v

Square Sum selected, enter an integer between 0 and 9

Sum of squares to n=6 is 90

Main Menu

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter g - Toggle the Green LED

Enter b - Toggle the Blue LED

Enter v - Find the sum of squares of a number entered into the terminal

Enter n - Find number of set bits in a random integer

Enter d - Disable the Blue <USER> push button switch interrupt

Enter e - Enable the Blue <USER> push button switch interrupt

Enter a - Disable all interrupts.

Depress Blue <USER> push button to test GPIO interrupt

received character = v

Square Sum selected, enter an integer between 0 and 9

Sum of squares to n=8 is 203

Main Menu

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter g - Toggle the Green LED

Enter b - Toggle the Blue LED

Enter v - Find the sum of squares of a number entered into the terminal

Enter n - Find number of set bits in a random integer

Enter d - Disable the Blue <USER> push button switch interrupt

Enter e - Enable the Blue <USER> push button switch interrupt

Enter a - Disable all interrupts.

Depress Blue <USER> push button to test GPIO interrupt

received character = v

Square Sum selected, enter an integer between 0 and 9

Sum of squares to n=9 is 284

Main Menu

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter g - Toggle the Green LED

Enter b - Toggle the Blue LED

Enter v - Find the sum of squares of a number entered into the terminal

Enter n - Find number of set bits in a random integer

Enter d - Disable the Blue <USER> push button switch interrupt

Enter e - Enable the Blue <USER> push button switch interrupt

Enter a - Disable all interrupts.

Depress Blue <USER> push button to test GPIO interrupt

## Test n – Number of Ones counter

received character = n

calling numOnes() with a random number = 0xa7ae0bd3

result = 18 ones counted

Main Menu

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter g - Toggle the Green LED

Enter b - Toggle the Blue LED

Enter v - Find the sum of squares of a number entered into the terminal

Enter n - Find number of set bits in a random integer

Enter d - Disable the Blue <USER> push button switch interrupt

Enter e - Enable the Blue <USER> push button switch interrupt

Enter a - Disable all interrupts.

Depress Blue <USER> push button to test GPIO interrupt

received character = n

calling numOnes() with a random number = 0xbf4e7331

result = 19 ones counted

Main Menu

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter g - Toggle the Green LED

Enter b - Toggle the Blue LED

Enter v - Find the sum of squares of a number entered into the terminal

Enter n - Find number of set bits in a random integer

Enter d - Disable the Blue <USER> push button switch interrupt

Enter e - Enable the Blue <USER> push button switch interrupt

Enter a - Disable all interrupts.

Depress Blue <USER> push button to test GPIO interrupt

received character = n

calling numOnes() with a random number = 0xb44a09ba

result = 14 ones counted

Main Menu

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter g - Toggle the Green LED

Enter b - Toggle the Blue LED

Enter v - Find the sum of squares of a number entered into the terminal

Enter n - Find number of set bits in a random integer

Enter d - Disable the Blue <USER> push button switch interrupt

Enter e - Enable the Blue <USER> push button switch interrupt

Enter a - Disable all interrupts.

Depress Blue <USER> push button to test GPIO interrupt

received character = n

calling numOnes() with a random number = 0xb8fcced7

result = 21 ones counted

Main Menu

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter g - Toggle the Green LED

Enter b - Toggle the Blue LED

Enter v - Find the sum of squares of a number entered into the terminal

Enter n - Find number of set bits in a random integer

Enter d - Disable the Blue <USER> push button switch interrupt

Enter e - Enable the Blue <USER> push button switch interrupt

Enter a - Disable all interrupts.

Depress Blue <USER> push button to test GPIO interrupt

## Test a – Disable Interrupts

received character = a

Disabling all interrupts

Depress <RESET> push button to restore functionality

## Test Reset

Welcome to <Embedded Controller Programming for Real-Time Systems>

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~ Assignment: Mid-Term Project ~

~ Course Number: ECE-40097 ~

~ Section ID: 145032 ~

~ Student Name: Chris Isabelle ~

~ SID: U01136665 ~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Main Menu

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Enter g - Toggle the Green LED

Enter b - Toggle the Blue LED

Enter v - Find the sum of squares of a number entered into the terminal

Enter n - Find number of set bits in a random integer

Enter d - Disable the Blue <USER> push button switch interrupt

Enter e - Enable the Blue <USER> push button switch interrupt

Enter a - Disable all interrupts.

Depress Blue <USER> push button to test GPIO interrupt