CG2271: Real-Time Operating Systems

Lab 5: UART

In this lab, you will be exploring the UART capabilities of the FRDM-KL25Z microcontroller. The Lecture Notes have given a good overview of the registers and how they can be configured. WE will now see them in action.

Part 1: Polling

In Part 1, we will be exploring the UART2 module of the KL25Z microcontroller.

A. Initializing the UART2 Module

The following screenshot shows the initUART2() code to initialize the PWM module.

```
13 /* Init UART2 */
14 void initUART2(uint32_t baud_rate)
15 {
16   uint32_t divisor, bus_clock;
17
18   SIM->SCGC4 |= SIM_SCGC4_UART2_MASK;
19   SIM->SCGC5 |= SIM_SCGC5_PORTE_MASK;
20
21   PORTE->PCR[UART_TX_PORTE22] &= ~PORT_PCR_MUX_MASK;
22   PORTE->PCR[UART_TX_PORTE22] |= PORT_PCR_MUX(4);
23
24   PORTE->PCR[UART_RX_PORTE23] &= ~PORT_PCR_MUX_MASK;
25   PORTE->PCR[UART_RX_PORTE23] |= PORT_PCR_MUX(4);
26
27   UART2->C2 &= ~((UART_C2_TE_MASK) | (UART_C2_RE_MASK));
28
29   bus_clock = (DEFAULT_SYSTEM_CLOCK)/2;
30   divisor = bus_clock / (baud_rate * 16);
31   UART2->BDH = UART_BDH_SBR(divisor >> 8);
32   UART2->BDL = UART_BDL_SBR(divisor);
33
34   UART2->C1 = 0;
35   UART2->S2 = 0;
36   UART2->C3 = 0;
37
38   UART2->C2 |= ((UART_C2_TE_MASK) | (UART_C2_RE_MASK));
39 }
```

REVIEW QUESTIONS:

Q1. Explain the need for the following line of code.

```
18   SIM->SCGC4 |= SIM_SCGC4_UART2_MASK;
19   SIM->SCGC5 |= SIM_SCGC5_PORTE_MASK;
20
```

Q2. Explain why this is necessary.

```
27   UART2->C2 &= ~((UART_C2_TE_MASK) | (UART_C2_RE_MASK));
28
```

Q3. Why do we need to perform the (baud_rate * 16) in the configuration as shown below?

```
29   bus_clock = (DEFAULT_SYSTEM_CLOCK)/2;
30   divisor = bus_clock / (baud_rate * 16);
31   UART2->BDH = UART_BDH_SBR(divisor >> 8);
32   UART2->BDL = UART_BDL_SBR(divisor);
```

B. UART2 Transmit Function

We will now explore the UART2 Transmit Function.

REVIEW QUESTIONS:

Q4. Explain the need for the while() loop in the function shown below.

```
41 /* UART2 Transmit Poll*/
42 void UART2_Transmit_Poll(uint8_t data)
43 {
44   while(!(UART2->S1 & UART_S1_TDRE_MASK));
45   UART2->D = data;
46 }
```

C. UART2 Receive Function

We will now explore the UART2 Receive Function.

REVIEW QUESTIONS:

Q5. Explain the need for the while() loop in the function shown below.

```
48 /* UART2 Receive Poll */
49 uint8_t UART2_Receive_Poll(void)
50 {
51   while(!(UART2->S1 & UART_S1_RDRF_MASK));
52   return (UART2->D);
53 }
```

D. Completing the Transmit

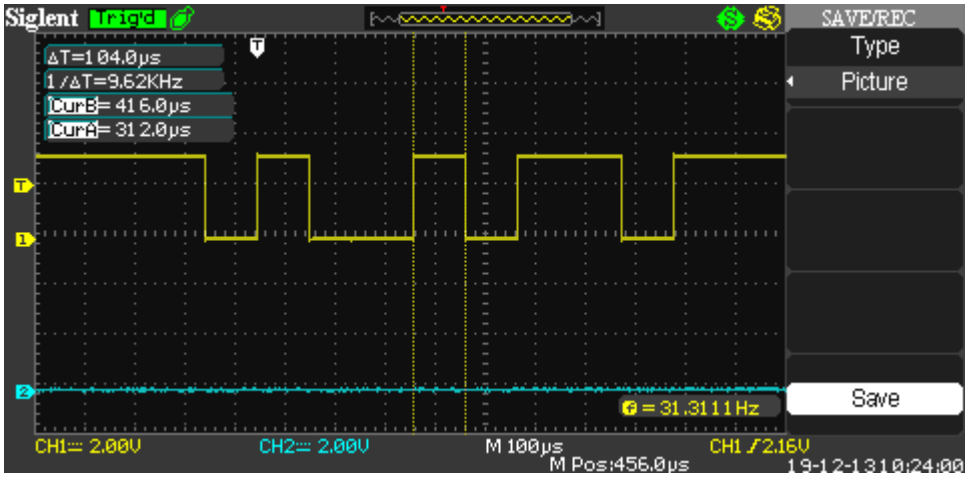
We are now ready to complete the code. The missing pieces are shown below.

```
 8 #define BAUD_RATE 9600
 9 #define UART_TX_PORTE22 22
10 #define UART_RX_PORTE23 23
11 #define UART2_INT_PRIO  128
```

```
55 /* Delay routine */
56 static void delay(volatile uint32_t nof) {
57   while(nof!=0) {
58     __asm("NOP");
59     nof--;
60   }
61 }
62
63 /* MAIN function */
64 int main(void)
65 {
66   uint8_t rx_data = 0x69;
67
68   SystemCoreClockUpdate();
69   initUART2(BAUD_RATE);
70
71   while(1)
72   {
73     /* Rx and Tx*/
74     UART2_Transmit_Poll(0x69);
75     delay(0x80000);
76   }
```

DEMO REQUIREMENT:

Try to monitor the UART signal on the TX pin of the microcontroller. You should be able a waveform similar to the one shown below.



REVIEW QUESTIONS:

Q6. Explain how the waveform above relates to the data being transmitted by the UART module.

Part2: UART over WiFi

For this part, you need to complete the setup and configuration of the ESP32. Refer to the ESP32 related documents in this order:

1. Getting Started with ESP32
2. App Controlled ESP32 Webserver
3. App_ESP32_Webserver_KL25Z

To get started with the AppInventor, you can refer to the following videos on the basics:

http://www.docr.sg/apps.html

As you are using WiFi to control the robot, you can choose to implement the controller in any form. IT can be an App or a webpage, or anything else that you can et working.

There are no marks allocated to the design of the controller. The main objective should be to design something that makes it easy for you to control the robot.

Part3: Interrupts

Modify the code in Part 2 to implement the UART using ISR's. Look back at the Lecture Slides for some sample code. You don't have to use circular buffers for now. You can use a fixed value that will be sent out through the TX pin.