__FALL 2025 EEL4742__ – Embedded Systems - Quiz 4 Solution - Time: 30 Minutes

1. (5 points) The following code configures UART on MSP430FR6989.

   (a) Fill it correctly in a way that it is using SMCLK = 1,048,576 Hz for 57600 baud communication **without** oversampling.

   (b) Fill it correctly in a way that it is using SMCLK = 4,000,000 Hz for 19200 baud communication **with** oversampling.

```
void Initialize_UART(void) {
    P3SEL1 _______________; // Divert pins to UART functionality
    P3SEL0 _______________; // Divert pins to UART functionality
    UCA1CTLW0 |= UCSWRST;
    UCA1CTLW0 |= ________;  // Select clock source

    // Compute baud rate settings
    UCA1BRW = ________;
    UCA1MCTLW = (________ << 8) | ________ | UCOS16;

    UCA1CTLW0 &= ~UCSWRST;
}
```

Pin number	x	Function	P3SEL1.x	P3SEL0.x
40	4	P3.4	0	0
		UCA1TXD	0	1
		...	...	...
41	5	P3.5	0	0
		UCA1RXD	0	1
		...	...	...

UCA1TXD: UART Transmit
UCA1RXD: UART Receive

UCPEN		UCMSB	UC7BIT	UCSPB	UCMODEx		UCSYNC	UCSSELx							UCSWRST
bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

UCWRST (Resetting the eUSCI_A)
0: eUSCI is operational
1: eUSCI is in reset

UCSSEL (clock source for eUSCI_A)
01: from ACLK
10: from SMCLK

UCSYNC (Selecting of async or sync)
0: asynchronous mode
1: synchronous mode

UCMODEx (operation mode for eUSCI_A)
00: UART Mode
01: Idle line (multiprocessor mode)
10: Address bit (multiprocessor mode)

UCPEN (Parity Enable)
0: disable
1: enable

UCSPB (number of stop bits)
0: one stop bit
1: two stop bit

UC7BIT (character length)
0: 8-bit
1: 7-bit

UCMSB (Endianness)
0: LSB first
1: MSB first

UCBRSx								UCBRFx							UCOS16
bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

UCBRSx (fraction part) **UCBRFx (fraction part)**

UCBRx															
bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

UCBRx (integer part)

Fractional Portion of N	UCBRSx[1]		Fractional Portion of N	UCBRSx[1]
0.0000	0x00		0.5002	0xAA
0.0529	0x01		0.5715	0x6B
0.0715	0x02		0.6003	0xAD
0.0835	0x04		0.6254	0xB5
0.1001	0x08		0.6432	0xB6
0.1252	0x10		0.6667	0xD6
0.1430	0x20		0.7001	0xB7
0.1670	0x11		0.7147	0xBB
0.2147	0x21		0.7503	0xDD
0.2224	0x22		0.7861	0xED
0.2503	0x44		0.8004	0xEE
0.3000	0x25		0.8333	0xBF
0.3335	0x49		0.8464	0xDF
0.3575	0x4A		0.8572	0xEF
0.3753	0x52		0.8751	0xF7
0.4003	0x92		0.9004	0xFB
0.4286	0x53		0.9170	0xFD
0.4378	0x55		0.9288	0xFE

Answer (part a): Given SMCLK=1,048,576 Hz, baud rate=57,600 baud, and NO oversampling (UCOS16 = 0):

```
void Initialize_UART(void) {
    P3SEL1 &= ~(BIT4 | BIT5);        // Divert pins
    P3SEL0 |=  (BIT4 | BIT5);

    UCA1CTLW0 |= UCSWRST;            // Hold eUSCI in reset
    UCA1CTLW0 |= UCSSEL_2;           // Use SMCLK

    UCA1BRW   = 18;                  // 1.048 MHz / 57600 = 18.204 --> INT(18.204)
    UCA1MCTLW = (0x11 << 8) | 0x0 | 000 | 0; // No oversampling (UCOS16=0) with fractional mapping

    UCA1CTLW0 &= ~UCSWRST;           // Release reset
}
```

Answer (part b): Given SMCLK=4,000,000 Hz, baud rate=19,200 baud, and oversampling (UCOS16 = 1):

```
void Initialize_UART(void) {
    P3SEL1 &= ~(BIT4 | BIT5);        // Divert pins
    P3SEL0 |=  (BIT4 | BIT5);

    UCA1CTLW0 |= UCSWRST;            // Hold eUSCI in reset
    UCA1CTLW0 |= UCSSEL_2;           // Use SMCLK

    UCA1BRW   = 13;                  // 4 MHz / 19200x4 = 13.0208 --> INT(13.0208)
    UCA1MCTLW = (0x49 << 8) | 0x0 | 000 | 1; // With oversampling (UCOS16=1) with fractional mapping

    UCA1CTLW0 &= ~UCSWRST;           // Release reset
}
```

2. (5 points) Complete the following code so that: (1) The function continuously (per every second) polls a light sensor over I2C; and (2) It only sends a new reading over UART when the value changes by more than 50 units (50 lux).

Answer: Based on provided function, here is the updated function reading the sensor every second and reflect it when the value changes by more than 50 units (50 lux).

```
void monitor_light(void) {
    Initialize_I2C();
    Initialize_UART();

    unsigned int current, previous = 0;

    while(1) {
        i2c_read_word(0x22, 0x70, &current); // Read the current light level from the I2C sensor

        if ( (current > previous + 50) || (previous > current + 50) ) { // Send only if change > 50 lux
            uart_write_uint16(current);
            previous = current;
        }

        __delay_cycles(1000000);  // delay for next reading
    }
}
```

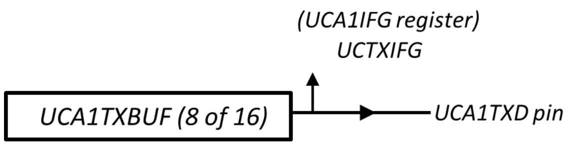
The helper functions below are available:

```
void Initialize_I2C(void);
void Initialize_UART(void);
int i2c_read_word(unsigned char i2c_address, unsigned char i2c_reg, unsigned int *data);
void uart_write_uint16(unsigned int data);
void __delay_cycles(int ms);
```

Let's assume I2C device address is 0x22 and the targeted register of I2C is 0x70.

3. (5 points) **Bonus**

You are debugging this UART transmission function on the MSP430. The code occasionally misses characters when sending strings rapidly. Assuming that baud rate is 9600, complete this code in a way that, with adding 20% write delay per character, the code ensures data is not overwritten before transmission completes
(**frequency is 1MHz = 1 us per operation**).




```
void uart_write_string(char *str) {
    while(*str) {
        // Wait for TX buffer to be ready
        while(________) == 0);    // (a)

        // Write next character
        ________ = *str++;

        ________;
        ________;
    }
}
```

Answer: At 9600 baud, one bit = 1 / 9600 = 104.17 µs. One character = 10 bits (1 start + 8 data + 1 stop) → 10 × 104.17 µs = 1.0417 ms per character. Adding 20% delay → 1.0417 × 1.2 ≈ 1.25 ms. If the system clock is 1 MHz, one loop iteration roughly equals 1 µs. So a delay loop of about 1250 iterations approximates 1.25 ms.

```
void uart_write_string(char *str) {
    while(*str) {
        // Wait for TX buffer to be ready
        while((UCA1IFG & UCTXIFG) == 0);    // (a) wait for buffer empty

        // Write next character
        UCA1TXBUF = *str++;

        // Add 20% delay per character (1.25 ms total at 9600 baud)
        for(volatile unsigned int i = 0; i < 1250; i++);    // (b)
    }
}
```