Homework 4

ECE2534 CRN:12927

Jacob Abel

March 14, 2018

Problem 1: Consider a UART with the following setup:

- 8 data bits
- EVEN parity
- 2 stop bits
- LSB first.

a) If the receiver gets the following frame: 1111001110110011111, has there been any data corruption in this communication? Explain and show your work to a reasonable extent.

Start Bit: 0, Data: 01110110, Parity Bit: 0, Stop Bits: 11, Hamming Weight: 5 (Odd)

The start and stop bits are correct however the data has the incorrect parity. There was likely data corruption.

b) If the baudrate is 9600 baud/sec, approximately how long does it take to transfer one character using this UART?

$\frac{1}{9600} s \approx 104 \ \ s$

c) If one end wishes to send 'G' to another, what will be the full data frame including start, data, parity and stop bits?

Start Bit: 0, Data: 0x47 : 1110001, Hamming Weight: 4 (Even), Parity Bit: 0, Stop Bits: 11

01110001011

Problem 2: Consider a UART with the following setup:

- 8 data bits
- Odd parity
- 1 stop bit
- LSB first
- BRCLK (Baudrate Source Clock) is 2.88 MHz
- Baudrate is 460800 bit/sec

a) For the MSP432 UART calculate the values of BCRx and BCRSx. (Based on the footnote below the Table 24-4 of MCU User Guide, if the fraction part of the division factor falls between two values in the table, we need to pick the lower one. In the Lab 2 starter code, we made a small mistake when by choosing 0xAA instead of 0x55. For this problem, please follow the proper method for selecting BCRSx)

$$N = \frac{2{:}88MHz}{460800bit/sec} = 6{:}25$$
$$OS16 = 0$$
$$UCBRx = 6$$
$$UCBRSx = 0x22$$

b)

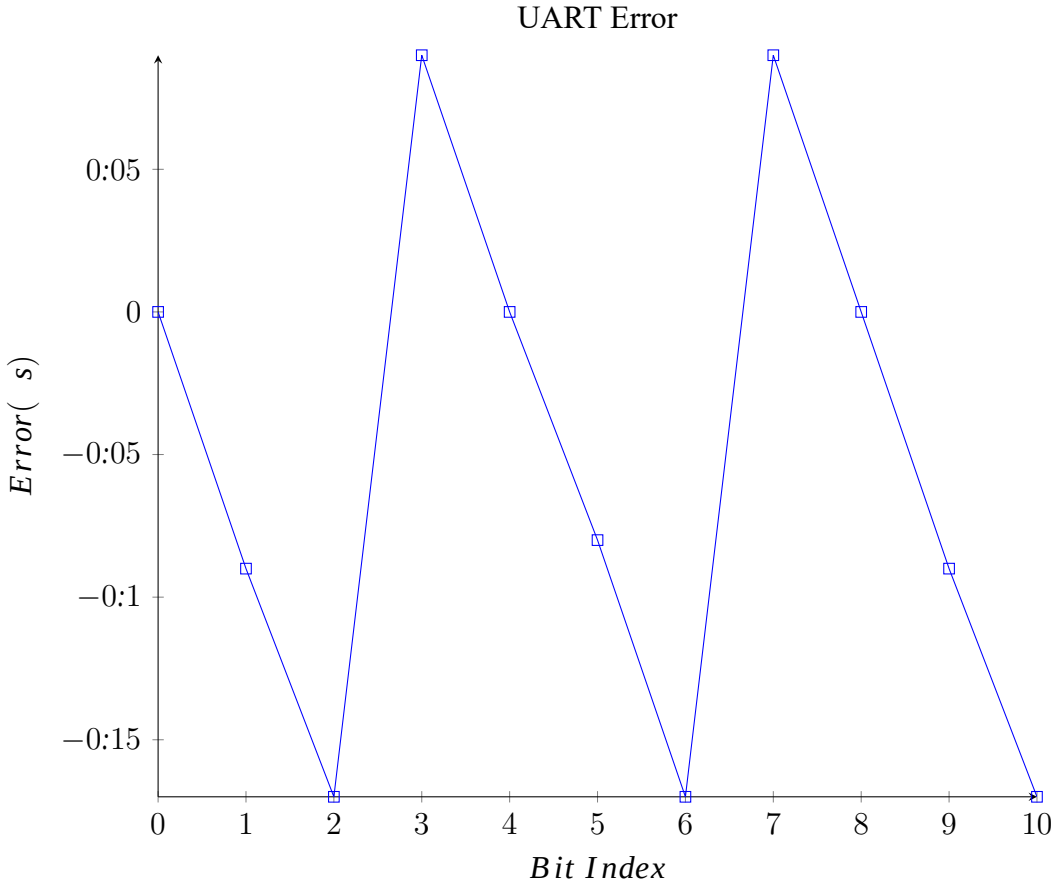
b) If we mark the middle of the start bit as beginning of time (0 s), then,

i) When is the ideal moment for sampling each of the following bits?
ii) What is the counter value that UART should load at each sampling moment (for waiting until next sampling moment)?
iii) What is the total ”clock ticks” UART has waited up to each sampling moment?
iv) What is the actual sampling moment for each bit?
v) What is the error (difference) between the ideal and actual sampling times?

Use the table below to enter your answers. For sampling times and error, enter the numbers in microseconds with two decimal points accuracy. The first three rows are completed to guide you.

	Ideal time	Counter value	Total ticks

d) Graph the error values with respect to bits. Do you see a pattern? Explain in a few words.



The error is very periodic and always corrects every 4 bits.

Problem 3: Consider a 32-bit Timer that is ticking with a 10MHz clock. Answer the following questions:

a) In the one-shot mode, to measure 500 ms, what is the value that needs to be loaded to the counter of this timer?

$10MHz \times 500ms = 5000000$

b) In the one-shot mode, if we load 1000 in the counter, how long does it take before the timer expires?

$\frac{1000}{10MHz} = 100 \ \ s$

c) What is the longest amount of time that can be measured in the one-shot mode using this timer?

$\frac{2^{32}-1}{10MHz} = 429{:}5s$

d) In the periodic mode with N = 10,000

i) If we read the counter, and it is 3000, if we wait for 7000 clock ticks, and read the counter again, what is the expected counter value (approximately)?

It should be at 6000 as it burnt the 3000 ticks, reset back to 10000, and burnt the remaining 4000 ticks.

ii) If we read the counter, and it is 4000, what is the counter value after 0.5 ms?

$10MHz \times 0{:}5ms = 5000 \implies 4000 - 5000 = -1000 \implies 10000 - 1000 = 9000$

The counter will be at 9000.

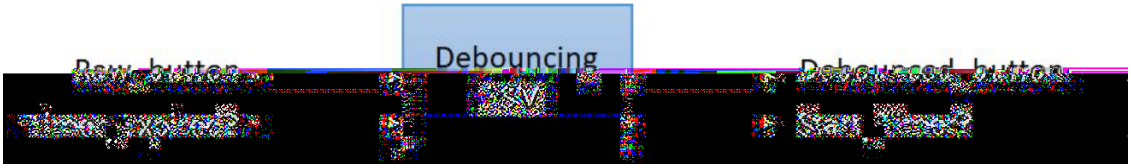
iii) If we read the counter value, and it is 2000 , and then we read it again, and it is 4000, and we know for sure the clock has rolled over only once during this time, how long has passed?

$2000 + (10000 - 4000) = 8000 \implies \frac{8000}{10MHz} = 800 \ \ s$

Problem 4: The below snippet of code starts TIMER32_0_BASE in free running mode with the max possible period. You need to write a function that uses this function to measure a given amount of time in microseconds. The details of the function are provided below.

```
// 16-bit timer ticking at 3,000,000 Hz
// Free running (periodic mode) with N equal to 2^16 -1
Timer32_initModule(TIMER32_0_BASE, TIMER32_PRESCALER_1,
                   TIMER32_16BIT, TIMER32_FREE_RUN_MODE);
Timer32_setCount(TIMER32_0_BASE, UINT16_MAX);
Timer32_startTimer(TIMER32_0_BASE, false);
/*
 * This function returns the next value of the
 * Timer32_0_BASE when time_in_microseconds has elapsed.
 * If the time is too long to be properly measured,
 * the function returns -1.
 * Example1: If counterWhenTimeHasPassed(1000) is called when
 * the TIMER32_0_BASE holds the value 1000, the returned
 * value will be UNIT16_MAX-2000
 * Example2: counterWhenTimeHasPassed(100000) always returns -1
 * Example3: If counterWhenTimeHasPassed(1000) is called when
 * the TIMER32_0_BASE holds the value 10000, the returned
 * value will be 7000.
 */
int counterWhenTimeHasPassed(int time_in_microseconds)
{
    int t = time_in_microseconds * 3;
    int t_cur = Timer32_getValue(TIMER32_0_BASE);
    if (t > UINT16_MAX) {
        return -1;
    }
    if (t > t_cur) {
        return UINT16_MAX + t_cur - t;
    }
    return t_cur - t;
}
```

Problem 5: Consider the button debouncing FSM shown below:



A 100ms timer is used to measure the debouncing window, and the FSM controls the start time of this timer, and looks at its expiration as one of its inputs. The below figure shows the starting point of the FSM's drawing. For each transition, the inputs are shown above the line: Raw_button (0, 1 or D for Don't care) followed by Timer_Expired (T for True, F for False or D for Don't care). The outputs are shown below the line, the debounced button (0 or 1) followed by Start_timer (T for True or F for False). Add the missing transition arrows as well as all the input/output values for each transition.