Embedded Systems: Homework #5

Yousef Alaa Awad

November 25, 2025

1

A)

Given: *For a daisy-chained SPI with 4 devices ($\mu C \rightarrow$ Device #1 $\rightarrow$ Device #2 $\rightarrow$ Device #3 $\rightarrow$ Device #4 $\rightarrow$), the SPI master intends to send the following bytes to the devices in sequence:*

- *0x5A to Device #1*
- *0x6B to Device #2*
- *0x7C to Device #3*
- *0x8D to Device #4*

A) After 32 clock cycles, what will be the contents of the SPI shift registers of devices?

Right at the start, the contents of the shift registers will be the following:

Stage	**Source**	**Step 1**	**Step 2**	**Step 3**	**Step 4**
Unit	μC	Device #1	Device #2	Device #3	Device #4
Value	0xFF	0xFF	0xFF	0xFF	0xFF

And, when 8 cycles occur, then the first data will be placed into Device #1, meaning it takes 8 cycles per placing of each data into the device...

Stage	**Source**	**Step 1**	**Step 2**	**Step 3**	**Step 4**
Unit	μC	Device #1	Device #2	Device #3	Device #4
Value	0x7C	0x8D	0xFF	0xFF	0xFF

And now, after 16 clock cycles, it means that it will be 2 devices in, and preloading the 3rd data point.

Stage	**Source**	**Step 1**	**Step 2**	**Step 3**	**Step 4**
Unit	μC	Device #1	Device #2	Device #3	Device #4
Value	0x6B	0x7C	0x8D	0xFF	0xFF

And... guess what... now it's going to be 32 clock cycles!!! (Since I think you can guess where its going).

Stage	**Source**	**Step 1**	**Step 2**	**Step 3**	**Step 4**
Unit	μC	Device #1	Device #2	Device #3	Device #4
Value	N/A	0x5A	0x6B	0x7C	0x8D

B) If the SPI clock operates at 500 kHz, how long (in milliseconds) will it take for the SPI master to complete the transmission to all devices and receive back the first byte sent?

Now, each bit specifically takes the following to transmit:

$$\frac{1}{500,000KHz} = 2\mu s$$

Therefore, for 32 bits to transmit, it would take...

$$32 * 2\mu s = 64\mu s = 0.064ms$$

C) Suppose the master sends the byte sequence with an incorrect clock frequency of 250 kHz, but the devices expect 500 kHz. What would be the potential impact on the data received by Device #4, and why?

At 250KHz, each bit would take the following amount of time to transmit:

$$\frac{1}{250,000KHz} = 4\mu s$$

However, since the master has a slower clock speed, creating a mismatch in what both the master and slave expect for timing, this will cause Device #1 to sample every bit that the master sends twice (instead of once as it should be). This timing issue would then cause Device #4 to recieve incomplete/corrupted data since the clock on both devices (that being, again, master and slave) being unsynchronized.

D) If Device #2 is configured to reverse the bits of the data it receives before passing it on, what will be the final byte received by the master after 32 clock cycles?

If Device #2 is configured to reverse all bits of data that it recieves, and then pass it on, the final byte that is recieved by the master after 32 cycles wil be 0xFF.

2

Given: *You are designing a system to control the brightness of an LED using timer-based Pulse Width Modulation (PWM) on a microcontroller.*

A) Calculate the period of a PWM signal with a frequency of 1 kHz in microseconds. The microcontroller's clock frequency is 16 MHz, and the timer uses a prescaler of 64. Calculate the value to be loaded into the timer's register (TA0CCRx value) to achieve a 1 kHz PWM frequency.

The period of a PWM signal is calculated as follows:

$$Period = \frac{1}{frequency} = \frac{1}{1KHz} = \frac{1}{1000} = 1000\mu s = 1ms$$

Now, to calculate the value that you must load into TA0CCRx to get a 1KHz frequency you then need to do the following math:

$$TA0CCRx = \frac{Clock\ Frequency}{Prescalar * PWM\ Frequency} - 1 = \frac{16MHz}{64 * 1KHz} = \frac{16,000,000}{64 * 1,000} - 1 = 250 - 1 = 249$$

B) Given the calculated TA0CCRx value from part (1), compute the TA0CCRx for the 50% and 75% duty cycle.

Given the TA0CCRx value of 249, as calculated in the previous part, if we wanted to get the TA0CCRx for respective duty cycles we'd do the following math...

$$TA0CCRx = Duty\ Cycle * TA0CCRx$$

Of which, for a 50% duty cycle, we'd then get..

$$TA0CCRx = 50\% * 249 \approx 125$$

And for a 75% duty cycle, we'd get...

$$TA0CCRx = 75\% * 249 \approx 188$$

C) Assume the timer is in 8-bit mode (255 max). Calculate the minimum and maximum achievable PWM frequencies using the 16 MHz clock and a prescaler of 64. Determine the resolution of the duty cycle as a percentage.

Now, for an 8-bit timer, you'd have a maximum TA0CCRx value of 255 (as given). To calculate the minimum and maximum frequencies, then, you'd simply need to do the following math:

$$MIN(freq) = \frac{clock}{prescalar * (TA0CCRx + 1)} = \frac{16,000,000}{64 * (255 + 1)} \approx 976.56Hz$$

$$MAX(freq) = \frac{clock}{prescalar * 2} = \frac{16,000,000}{64 * 2} = 125,000Hz = 125KHz$$

Now, to get the resolution of the duty cycle, you'd need to do the following:

$$Resolution = \frac{1}{TA0CCRx + 1} * 100 = \frac{1}{255 + 1} * 100 = 0.39\%$$

D) Suppose you switch to a 16-bit timer with the same 16 MHz clock and a prescaler of 64. Calculate the maximum TA0CCRx value. Calculate the minimum achievable PWM frequency.

Now, to find the minimum and maximum achievable PWM frequencies using a 16MHz clock, with a prescalar of 64, we'd do the following math...

$$MAX(TA0CCRx) = 2^{bit\ width\ of\ timer} = 2^{16} = 65,535$$

$$MIN(TA0CCRx)_{frequency} = \frac{Clock\ Frequency}{Prescalar * (TA0CCRx + 1)} = \frac{16,000,000}{64 * (65,535 + 1)} \approx 3.81Hz$$

3

Given: *A push button is used to control an LED based on these rules: (1) Every time the push button is pressed, the LED turns ON for 5 seconds; (2) If the button is pressed again within the 5-second window, the timer resets, and the LED stays ON for another 5 seconds from the latest press. A user interacts with the button as follows:*

- *Press 1: Time = 0s*
- *Press 2: Time = 3s*
- *Press 3: Time = 8s*
- *Press 4: Time = 12s*

A) Calculate the total time the LED stays ON based on the button press sequence. Provide a timeline showing the ON and OFF states of the LED.

The total time that the LED stays on would be the addition of the following:

- Press 1: LED is on from 0 to 5s (+5s)
- Press 2: LED is reset and is now on from 3s to 8s (-2s + 5s)
- Press 3: LED is now on from 8s to 13s (+5s)
- Press 4: LED resets again and is now on from 12s to 17s (-1s + 5s)

The total time that the LED is on would simply be, since it never turned off in all 4 presses, be 17 seconds. This is also confirmed by the following math:

$$5 - 2 + 5 + 5 - 1 + 5 = 17$$

B) Explain if there are any periods where pressing the button has no effect and why that happens.

The only time that the button press has no effect is if the press of the button is after the 5 seconds of LED being on. This, therefore, means that in the case of our 4 presses, there are no ignored presses.

C) Consider a scenario where the button is pressed every 4 seconds starting at Time = 0 seconds. How long would the LED remain ON by the 20th press?

If the button is pressed every 4 seconds starting at 0 seconds, the LED would be on indefinetely. This specifically occurs since you keep resetting the 5 second timer, every 4 seconds, thereby getting rid of any chance for the LED to turn off. This therefore means that, by the 20th press it would have 19 four second presses, plus the addition 5 seconds (assuming you stop pressing after) of LED on giving you...

$$4 * 19 + 5 = 81 \ seconds$$