ECE 375 PRELAB 8

Lab Time: Tuesday 4-6

Owen Markley 2/23

QUESTIONS

1. In this lab, you will be given a set of behaviors/actions that you need to have a proof-of-concept "toy" perform. Think of a toy you know of (or look around online for a toy) that is likely implemented using a microcontroller and describe the behaviors it performs. Here is an example behavior: "If you press button X on the toy, it takes action Y (or makes sound Z)"

   A RC car would have a microcontroller in it. Moving a lever forwards to increase would increase the speed of the motor on the car. Pushing another lever to the side would adjust the front axle of the car either left or right depending on the input. Releasing either of these levers would respectively slow the engine down to a stop, or return the axle to its resting / original position.

2. For each behavior you described in the previous question, explain which microcontroller feature was likely used to implement that behavior, and give a brief code example indicating how that feature should be configured. Make your explanations as the ATmega128-specific as possible (e.g., discuss which I/O registers would need to be configured, and if any interrupts will be used), and also mention if any additional mechanical and/or electronic devices are needed.

   First the car would need several definitions to be made. We want variables for Forward, Reverse, Right, Left, and Stop. We should be able to put these all on DDRC / PORTC. Based off of these, we have the different states to define. Forwards would load the left and right motor pins, Reverse would change the direction of these pins, right would adjust the motors to move the axle right, and Left would do the same in the opposite direction. Another port possibly PORTD can be used to take in inputs. I'm not yet sure how input is taken in from a wireless source, possibly another piece of hardware must be added to the chip. There are 4 signals that can be taken in by the wireless input / remote. I would use signals so that actions can occur during other processes, for example. A left turn can be taken while moving the car forwards.

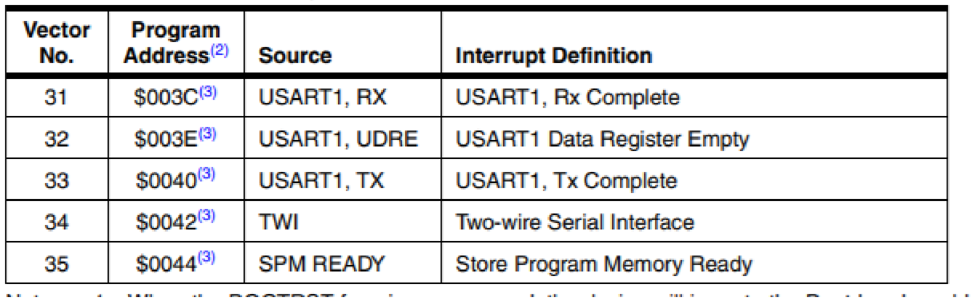
3. Each ATmega128 USART module has two flags used to indicate its current transmitter state: the Data Register Empty (UDRE) flag and Transmit Complete (TXC) flag. What is the difference between these two flags, and which one always gets set first as the transmitter runs? You will probably need to read about the *Data Transmission* process in the datasheet (including looking at any relevant USART diagrams) to answer this question.

   The UDRE flag indicates if the transmit buffer is empty and is ready to receive new data to transmit. When data has not been moved to the shift register and is in the transmit buffer then the bit is set. The TXC flag is set when the transmit shift register has been shifted out and there is no new data in the transmit buffer. This is cleared when a transmit complete interrupt is executed. The UDRE flag should be set first, seeing as the data can be transmitted from the buffer without having been transmitted yet.

4. Each ATmega128 USART module has one flag used to indicate its current receiver state (not including the error flags). For USART1 specifically, what is the name of this flag, and what is the interrupt vector address for the interrupt associated with this flag? This time, you will probably need to read about *Data Reception* in the datasheet to answer this question.

This would be the RXC flag or receive complete flag. Should this flag be set, the USART Receive Complete Interrupt will be executed. This is enabled by the Receive Complete Interrupt Enable (RXCIE). The interrupt vector address of this is $003C

Table 23. Reset and Interrupt Vectors



Vector No.	Program Address[2]	Source	Interrupt Definition
31	$003C[3]	USART1, RX	USART1, Rx Complete
32	$003E[3]	USART1, UDRE	USART1 Data Register Empty
33	$0040[3]	USART1, TX	USART1, Tx Complete
34	$0042[3]	TWI	Two-wire Serial Interface
35	$0044[3]	SPM READY	Store Program Memory Ready

Reference