Zachary DeVita

ECE 375

Lab#8 PreLab

*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 description: “If you press button X on the toy, it takes action Y (or makes sound Z)”.*

I have one of my kid’s dolls nearby. When I flip the power switch X on, then the LED turns on which signifies that the toy has power. When I press a button X on the doll, a song from the movie associated with the doll is played.

*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 explanation as 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.*

When the power switch is turned on, then commands in the program ‘Initialization’ turn the LED light on. The ‘Stack Pointer’ is initialized, and a port is initialized for output. The port is then hard coded with a 1 to turn the light on. Example assumes that the LEDs output is determined by the first bit in PORTB.

Example Code

INIT:

; Initialize the Stack Pointer

ldi mpr, low(RAMEND)

out SPL, mpr ; Load SPL with low byte of RAMEND

ldi mpr, high(RAMEND)

out SPH, mpr ; Load SPH with high byte of RAMEND

; Initialize Port B for output

ldi mpr, $FF ; Set Port B Data Register

out DDRA, mpr ; for output

ldi mpr, $01 ; Initialize Port B Data Register

out PORTA, mpr

The button which plays music signals an interrupt on the microcontroller. For this you need a port for input and a port for output, and an interrupt set up. The ‘main’ will simply loop until an interrupt is received. The Z pointer will need to be initialized so that it can point to the memory location of the sound byte. The mask and global interrupts must be set. I am not qualified for some of this coding but I will do my best.

Example Code

; Interrupt Vectors

.org $0000

rjmp INIT ; reset interrupt

.org $0002 ; {IRQ0 => pin0, PORTD}

rcall PLAYMUSIC

reti ; Return from interrupt

INIT:

; Initialize the Stack Pointer

ldi mpr, low(RAMEND)

out SPL, mpr ; Load SPL with low byte of RAMEND

ldi mpr, high(RAMEND)

out SPH, mpr ; Load SPH with high byte of RAMEND

; Configure I/O ports

; Initialize Port B for output

ldi mpr, $FF ; Set Port B Data Register

out DDRB, mpr ; for output

ldi mpr, $00 ; Initialize Port B Data Register

out PORTB, mpr ; so all Port B outputs are low

; Initialize Port D for input

ldi mpr, $00 ; Set Port D Data Direction Register

out DDRD, mpr ; for input

ldi mpr, $FF ; Initialize Port D Data Register

out PORTD, mpr ; so all Port D inputs are Tri-State

ldi ZH, high(clip<<1) ; Initialize Z pointer to music clip

ldi ZL, low(clip<<1)

; Configure External Interrupt Mask

ldi mpr, $0F

out EIMSK, mpr

; Enable global interrupts

ldi mpr, $AA ; set interrupts to falling edge

sts EICRA, mpr

sei ; set global interrupt flag

MAIN:

rjmp MAIN ; return to top of MAIN

; Subroutines

PLAYMUSIC:

lpm mpr, Z ; load address into mpr

; I would assume something magical must happen here

out PORTB, mpr ; output value to PORTB

ret ; End a function with RET

*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.*

Both of these flags indicate USART Transmitter state, and both flags can be used for generating interrupts. The UDRE flag indicates whether the transmit buffer is ready to receive new data, and the TXC flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer. The UDRE flag is set when the byte that was just written moves from UDR to the transmit shift register. This is so you can wait to write UDR until the UDRE is set. TXC is not set until the last bit in the transmit shift register is finished shifting, and TX is completed.

The UDRE flag gets set first because it can be set before any bits have been sent, opposed to the TXC flag which needs the entire frame to be shifted.

*3. Each ATmega128 USART module has one flag used to indicate its current receiver state (not including the error flags). 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.*

The Receive Enable (RXEN) bit indicates the USART Receiver state; it is Bit 4. The address associated with the RXEN interrupt vector is $0024.