Homework - Mastermind Report Draft

Submitted: September 22, 2017

Sabbir Ahmed

Description

This project emulates a variant of the game "mastermind" on the AVR Dragon. The user will take guesses at the code using the joystick with combinations of up-down-left-right. A wrong input will sound a buzzer (piezoelectric element) and reset the user's game. Completing the code will light an LED. At this point the game can be reset by pressing the push-button. All inputs will be logged over a serial interface (UART) to a computer reporting the state of the game.

State Machine Implementation

The game is implemented as a state machine, where each successful inputs progresses through the states.

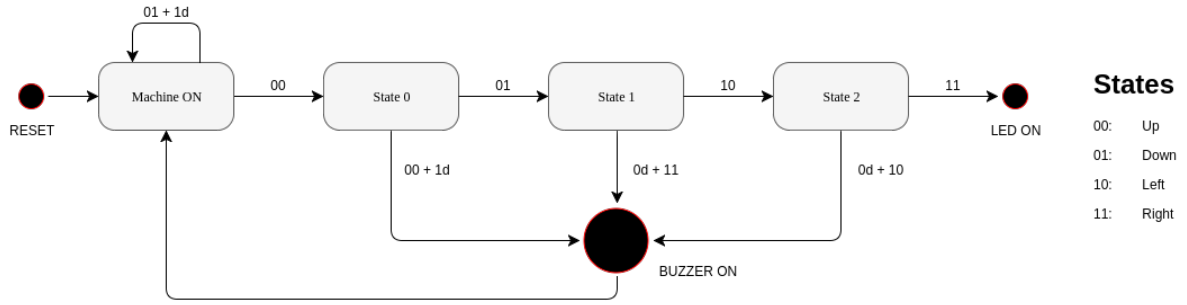

Figure 1: State Diagram Implementation of the Game

Code

The assembly code used for the implementation has been attached and provided at the end of the report.

mastermind.asm

Contains the state machine implementation of the game.

uart.asm

Contains the specifications for the UART interface.

```
/* mastermind.asm
State machine implementation of Mastermind
*/

; Include the butterfly definitions for the M169P.
.INCLUDE "M169PDEF.INC" ;(BUTTERFLY DEFINITIONS)

; initialize the stack and also define the port functionality
.DEF PORTDEF                    = R22
; Counter 1 and 2 are used to waste time so that the buzzer sound is hearable
.DEF COUNTER                    = R23
.DEF COUNTER2                   = R24

; secret code to win the game (UP, DOWN, LEFT, RIGHT)
.EQU secret                     = 0b00011011

.ORG $0000
    RJMP START

; Initializes the stack.
START:

    LDI PORTDEF, HIGH(RAMEND)   ; Upper byte
    OUT SPH, PORTDEF            ; to stack pointer
    LDI PORTDEF, LOW(RAMEND)    ; Lower byte
    OUT SPL, PORTDEF            ; to stack pointer

; PORT_INIT initializes the ports for I/O. It is configured so
; that any of the PORT D Pin 7 is used for the push button (input)
; and PIN 1 on PORT B can be used for the LED (output)
; The buzzer is also connected to PORT B (Pin 5) so that is initialized as well
PORT_INIT:

    LDI PORTDEF, 0b00100010
    OUT DDRB, PORTDEF
    LDI PORTDEF, 0b10000000
    OUT PORTD, PORTDEF

; STATE0:
; The very first state where no input has been entered. Will directly call
; READINPUT and wait for user to press the push button. Once the button is
; pressed, turns on the LED and jumps to next state.
STATE0:

    RCALL READINPUT
    RCALL TURNONLED

; STATE1:
```

```
; The second state. Does the same thing as State 0 but this time beeps the for
; successful push button press.
STATE1:

    RCALL READINPUT
    RCALL BEEPNOISE

; STATE2:
; Once a button is pressed, turns off the LED and resets
; Return to state 0
STATE2:

    RCALL READINPUT
    LDI PORTDEF, 0b00000000
    OUT PORTB, PORTDEF
    JMP STATE0


; Routine to turn on the LED
TURNONLED:
    LDI PORTDEF, 0b00000000
    OUT PORTB, PORTDEF
    RET

; Beepnoise sets the buzzer high (at PortB, Pin 5) and then sits in a loop so
; that the buzzer is low enough frequency to be hearable to the human ear.
BEEPNOISE:
    LDI PORTDEF, 0b00100000
    OUT PORTB, PORTDEF
    ; Wastetime is a counter that counts to 255 and then returns.
    RCALL WASTETIME
    ; Once wastetime is finished, the buzzer is turned off
    LDI PORTDEF, 0b11011111
    OUT PORTB, PORTDEF
    ; Wastetime is called again to make the period of the soundwave even lower
    RCALL WASTETIME
    ; This whole thing is done 255 times to make the buzzer hearable.
    DEC COUNTER2
        BRNE BEEPNOISE
    RET

; sits in loop and waits for user to press PIND 7
; if pressed, debounces
READINPUT:
    SBIS PIND, 7
        RJMP DEBOUNCE
    RJMP READINPUT
```

```
; Waits for user to stop pressing and then returns.
DEBOUNCE:
    SBIC PIND, 7
        RET
    RJMP DEBOUNCE

; Used to make the buzzer sound hearable. Used
; to lower frequence enough so that the sound from the buzzer is hearable
WASTETIME:
    CLR COUNTER

CONTWASTETIME:
    NOP
    DEC COUNTER
    BRNE CONTWASTETIME
        RET

; definitions for the joystick inputs
SBIS PINB, 6                    ; joystick up
    RJMP JOYUP
SBIS PINB, 7                    ; joystick down
    RJMP JOYDOWN
SBIS PINE, 2                    ; joystick left
    RJMP JOYLEFT
SBIS PINE, 3                    ; joystick right
    RJMP JOYRIGHT
```

```
/* uart.asm
specifications for the UART interface
*/

; Simple program to send "Hello! World" using AVR Butteryfly @ default 8Mhz
; runs 4800 buad with 2 stop bits and no parity
.DEF TEMP                       = R16
.DEF TEMP0                      = R17
.DEF TEMP1                      = R18

; Initialize Stack Pointer to end of RAM
LDI TEMP, HIGH(RAMEND)          ; Upper byte
OUT SPH, TEMP                   ; to stack pointer
LDI TEMP, LOW(RAMEND)           ; Lower byte
OUT SPL, TEMP                   ; to stack pointer

USART_INIT:

    ; Load UBRRH with 0 and UBRRL with 103
    ; in other words FOSC/16/BAUD-1
    ; to set a baud rate of about 4800 at 8MHz
    LDI TEMP, 00

    ; MOV UBRRH, TEMP ; can't do this since UBRRH is in the _extended_ i/o
    STS UBRRH, TEMP
    LDI TEMP, 103
    STS UBRRL, TEMP

    ; Clear all error flags
    LDI TEMP, 00
    STS UCSRA, TEMP

    ; Enable Transmission and Reception
    LDI TEMP, (1 << RXEN0) | (1 << TXEN0)
    STS UCSRB, TEMP

    ; Set frame format: 8data, 2stop bit
    LDI TEMP, (1 << USBS0) | (3 << UCSZ00)
    STS UCSR0C, TEMP

; Transmit Data
USART_TRANSMIT_H:

    ; Wait for empty transmit buffer
    LDS TEMP0, UCSR0A
    SBRS TEMP0, UDRE
    RJMP USART_TRANSMIT_H
```

```
    ; send the data
    LDI TEMP, 0x48
    STS UDR0, TEMP

USART_TRANSMIT_E:

    ; Wait for empty transmit buffer
    LDS TEMP0, UCSR0A
    SBRS TEMP0, UDRE
    RJMP USART_TRANSMIT_E

    ; send the data
    LDI TEMP, 0x45
    STS UDR0, TEMP

USART_TRANSMIT_L:

    ; Wait for empty transmit buffer
    LDS TEMP0, UCSR0A
    SBRS TEMP0, UDRE
    RJMP USART_TRANSMIT_L

    ; send the data
    LDI TEMP, 0x4C
    STS UDR0, TEMP

USART_TRANSMIT_O:

    ; Wait for empty transmit buffer
    LDS TEMP0, UCSR0A
    SBRS TEMP0, UDRE
    RJMP USART_TRANSMIT_O

    ; send the data
    LDI TEMP, 0x4F
    STS UDR0, TEMP

USART_TRANSMIT_EXCLAMATION:

    ; Wait for empty transmit buffer
    LDS TEMP0, UCSR0A
    SBRS TEMP0, UDRE
    RJMP USART_TRANSMIT_EXCLAMATION

    ; send the data
    LDI TEMP, 0x21
    STS UDR0, TEMP
```

```
; now send the data using a function call
DONE:
    RJMP DONE

; assumes data is in register TEMP
USART_TRANSMIT:

    ; Wait for empty transmit buffer
    ;sbis UCSR0A,UDRE
    LDS TEMP0, UCSR0A
    SBRS TEMP0, UDRE
    RJMP USART_TRANSMIT

    ; send the data
    STS UDR0, TEMP
    RET
```