Discussion 2

STATE MACHINES AND AVR ASSEMBLY

CMPE 311 / FALL 2015

DR. MOHSENIN & MR. SMITH

GTAS: RUTHVIK KUKKAPALLI, AMEY KULKARNI

UTAS: JASON KLIMEK, ALI AHMAD

Objectives

Learn the basics of the design, structure and construction of state machines.

Learn some basic AVR Assembly Commands:

- LDI, RCALL, RJMP etc.
- Code required for initializing I/O ports
- Reading I/O pins, writing to them as well as reading specific bits in a register.

Design and implement a simple state machine using AVR Assembly and the ATMEL IDE.

AVR Assembly Commands

- AVR Assembly allows you to “map” registers – making variable names point to a specific register. Use the **.DEF** command for this:
  - `.DEF PORTDEF = R22`
- **LDI** (LoaD Immediate) Allows you to load a fixed value or constant into a register.
  - `LDI PORTDEF, 0b11111111`
- **MOV** (MOVe) Allows you to copy a value from one register to another register.
  - `.DEF NEWREG = R23`
  - `MOV NEWREG, PORTDEF`
- **RCALL** Calls a subroutine and can be returned to.
  - `STATE0:`
    `RCALL READINPUT`
- **RJMP** Jumps to a subroutine (typically can’t be returned to.)
  - `LOOPFOREVER:`
    `RJMP LOOPFOREVER`

AVR Assembly Commands (Compare and Specific Jumps)

- **CP** (ComPare) Sets a flag that allows you to do different kinds of jumps. ONLY WORKS WITH REGISTERS
  - `CP PORTDEF, NEWREG`
- **BREQ** Jumps if CP set Equal Flag.
  - ```
    CP PORTDEF, NEWREG
        BREQ LOOPFOREVER
    ```
- **BRNE** Jumps if CP set Not Equal Flag.
- **BRSH** Jumps greater equal (unsigned)
- **BRLO** Jumps if less (unsigned)
    ```

AVR Assembly Commands (Port/Register Conditionals)

- **SBRC** Sets flag if register bit clear.
  - ```
    SBRC PORTDEF, 7
        RJMP LOOPFOREVER
    ```
- **SBRS** Sets flag if register bit set
  - ```
    SBRC PORTDEF, 7
        RJMP LOOPFOREVER
    ```
- **SBIC** Sets flag if port pin clear
- **SBIS** Sets flag if port pin set

- Many of the simple commands needed for this discussion can be found at this webpage. For a more thorough explanation see the website.

Initializing I/O Ports

Use the DDRx and PORTx registers to initialize ports as I/O. They are initialized with the OUT command.

To initialize a pin as an input, make sure the DDR (Direction Register) for that pin number is set to 0 then activate the pull up resistor for that pin by writing a logic 1 to the PORT bit for that pin.

```
LDI PORTDEF, 0b00000001      ;Assuming DDRB0 == 0
    OUT PORTB, PORTDEF          ;Activates pullup for Port B, pin 0
```

To initialize a pin as an output, write a logic 1 to the DDR register for that pin bit.

```
LDI PORTDEF, 0b00000001
    OUT DDRD, PORTDEF           ;Port D, pin 0 is an output now
```

Refer to section 13.2 in the AVR datasheet for more details.

Peripheral definitions (Joystick and Piezo PORT/PIN definitions are in the User Guide. See sections 3.5/3.10

State Machine Structure for Today

The State Machine you are going to design will have 3 simple states:

- State 1 – Rest State – pressing button turns on LED goes to State 2
- State 2 – LED On – pressing button beeps and goes to State 3
- State 3 – LED On (Beeped) – pressing button turns off LED and Resets to State 1

The partial code is provided. Think about how you need to set up your code to produce the desired results.

Considerations:

- The switch buttons are imperfect. Sometimes pressing the button once may trigger as 2 presses and your state machine may look like it's not working properly. Test it out before deciding your code is wrong.
- Look at how the stack is initialized and figure out how that works on your own. It allows you to return to previous methods if you do a RCALL.

Let us know if you need any help! ☺