AVR IO Ports

General Purpose I/Os, Pull-Up Resistors, Programming IOs

Credit to Dr. Robucci for slide information
ATMega169P Chip
I/O Ports

- All AVR Ports have true Read-Modify-Write functionality
  - Each pin on a port can be modified without unintentionally modifying any other pin
- Three I/O memory address locations allocated for each port
  - Data Register – PORTx (Read/Write)
  - Data Direction Register – DDRx (Read/Write)
  - Port Input Pins – PINx (Read)
All ATmega169P I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.

When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses.
I/O Ports

- ATMega 169P has 7 ports
  - A,B,C,D,E,F,G
- Pxn represents nth bit in Port x
  - E.g. PA6 == 6th bit of Port A (can be used only in C)
- If DDxn is a:
  - 1 – Pxn is configured to be an output pin
  - 0 – Pxn is configured to be an input pin
- If DDxn is configured as output and PORTxn is:
  - 1 – Pxn is driven high (1)
  - 0 – Pxn is driven low (0)
- Note: “writing” a logic 1 to a bit in the PINx Register will *toggle* the corresponding bit in the data
Programming I/O Ports - Registers

- Each port is controlled by 3 8-bit registers that control specific functionality of that port.
- $\text{DDR}x$ – Controls the Data Direction flow of the port.
- $\text{PORT}x$ – Controls the output value of an output pin as well pull-up resistor status of input pin.
- $\text{PIN}x$ – Can be read in to see the value of any pin on a port.
## I/O PORT

### DDRx Reg.
Controls the direction of data flow in both modes.

<table>
<thead>
<tr>
<th>1</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Input</td>
</tr>
</tbody>
</table>

### PORTx Reg.

<table>
<thead>
<tr>
<th>Input Mode</th>
<th>Output Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controls the pull up resistor connected to pin.</td>
<td>Controls output value of pin.</td>
</tr>
<tr>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td>1</td>
<td>On</td>
</tr>
</tbody>
</table>

### PINx Reg.
Can be read to see the value at any of the 8 pins on the port.
Programming I/O Ports - Assembly

- Using CBI and SBI to write to ports
  - SBI DDRB, 1 ;make bit 1 as output bit on PORTB //SBI DDRB, 0b00000001
  - CBI PORTB, 1 ;make PORTB bit 1 as "0"
  - SBI PORTB, 1 ;make PORTB bit 1 as "1"

- Using OUT instruction to write to ports
  - LDI R18, 0b00010000
  - OUT DDRB, R18 ;make bit 4 as output bit on PORTB
  - LDI R18, 0b00000000
  - OUT PORTB, R18 ;make PORTB bit 4 as "0"
  - LDI R18, 0b00010000
  - OUT PORTB, R18 ;make PORTB bit 4 as "1"
Programming I/O Ports - Assembly

• ;INPUT EXAMPLE
• IN R18, PINB //Reads all 8bits, For example, reading Push button value
• ;set pin 4 of B port as output
• ; without affecting other bits
• IN R18, DDRB //reading DDRB register value and storing in R18 can be any other register like R20
• ORI R18, 0b00010000
• OUT DDRB, R18

• We don’t need to use R18 for all three cases. Basically you can use any register for each of these three examples

• ;set pin 4 of B port to 1
• ; without affecting other bits
• IN R18, PORTB
• ORI R18, 0b00100000
• OUT PORTB, R18 //Since in previous example this bit was as output through DDRB, then this line means that we are making pin 4 port 4 to be one. For example, turning n LED On (or off)
Programming I/O Ports - Assembly

- ;clear pin 4 of B port to 0
- ; without affecting other bits
- IN R18, PORTB
- ANDI R18, 0b11101111
- OUT PORTB, R18  //In this example if we don’t set the bit4 direction as output then this value doesn’t get set to the output port

- ;set pin 7,3 of B port to 1 at same time
- ; without affecting other bits
- IN R18, PORTB
- ORI R18, 0b10001000
- OUT PORTB, R18
Programming I/O Ports - Assembly

- ;toggle pin 1 of B (no eori available)
- ; without affecting other bits
- IN R18,PORTB
- LDI R19,0b00000010
- EOR R18, R19
- OUT PORTB, R18 //Inverts Pin 1 by Xoring it or inverting it

- ;toggle pin 1 of B using PINB "input write trick"
- OUT PINB, 0b00000010
Ports on Butterfly Board

Figure 3-1. Connectors

Figure 3-7. PORT B and PORT D
Review of Bit masking

- Controlling Port I/O makes bit masks invaluable
  - Allows control of single pins without affecting others
- Using OR as mask to bring up a pin
  - ORI A o’b00000001
    - Only makes the LSB become 1, leaves others unaffected
- Using AND as mask to bring a pin down
  - ANDI A o’b1111110
    - Only makes LSB become 0, leaves others unaffected