ATMega128 I/O Ports

The AVR I/O ports are **the** path to the outside world.

Understand how to use them and life is good.

Failure to understand how the ports are used **will cause grief and possibly cost $'s.**

An abused I/O port is fairly easy to burn out with excessive current or static damage.
- -most all the I/O ports are floating inputs that can build up large static charge

Never carry your AVR board in a non static-dissipative bag.
- -dry fall days are perfect for creating conditions for ESD damage
- -practice safe electronics, use the pink bag

Using proper port software conventions will keep code transportable, readable and more bug free. i.e.,

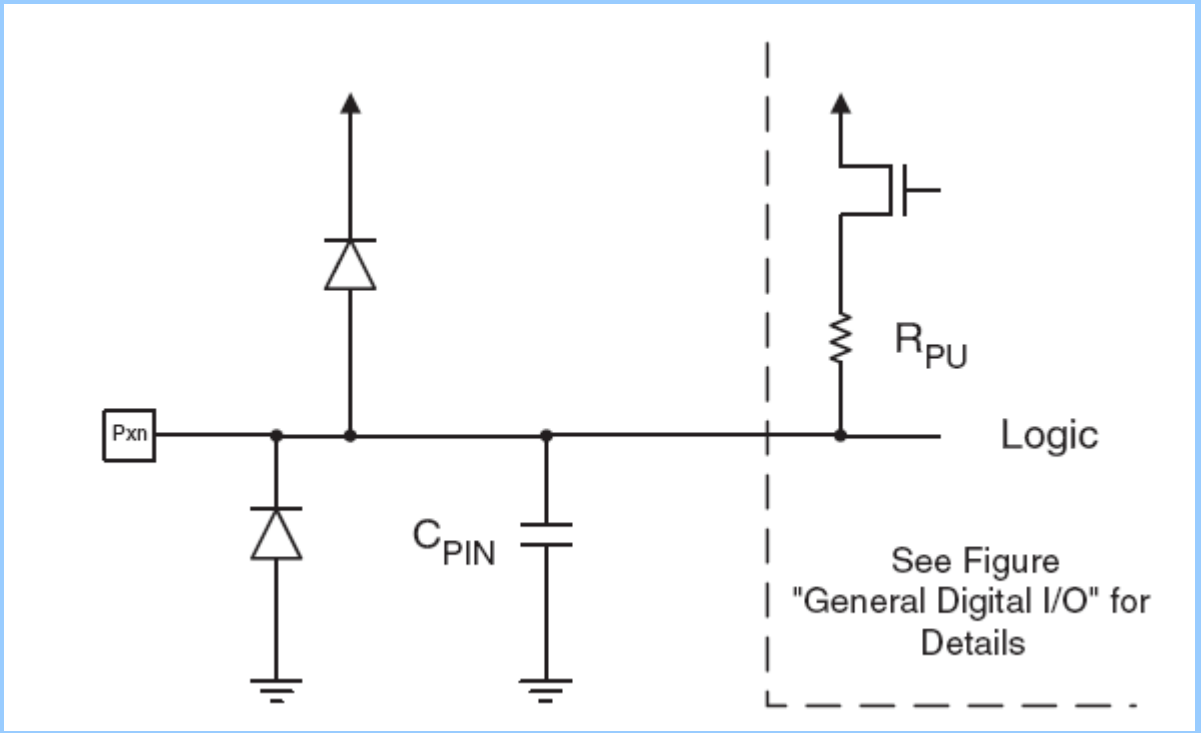
```
TIMSK |= (1<<TOIE0); //sets a bit
```

Versus

```
TIMSK = b01000000; //clobbers the register
```

ATMega128 I/O Ports

I/O Port input structure



-protection diodes
-programmable pull-up resistor
-what happens if voltages exceeding Vcc are applied to an I/O pin?
-can you power a chip from an I/O pin?

ATMega128 I/O Ports

All ports.....

-have bit-selectable pull-up resistors
-have bit-selectable tri-state outputs (what are these?)
-have schmitt trigger input buffers (what is that?) (can you draw one?)
-are synchronized to the system clock to prevent metastability (what is that?)
-have symmetrical DC drive capability (what does that mean?)

All ports have read-modify-write capability, i.e.,

-i.e., you can change pin *direction*, pin *value*, or pin *pull-up resistor* without effecting any other pins in the port

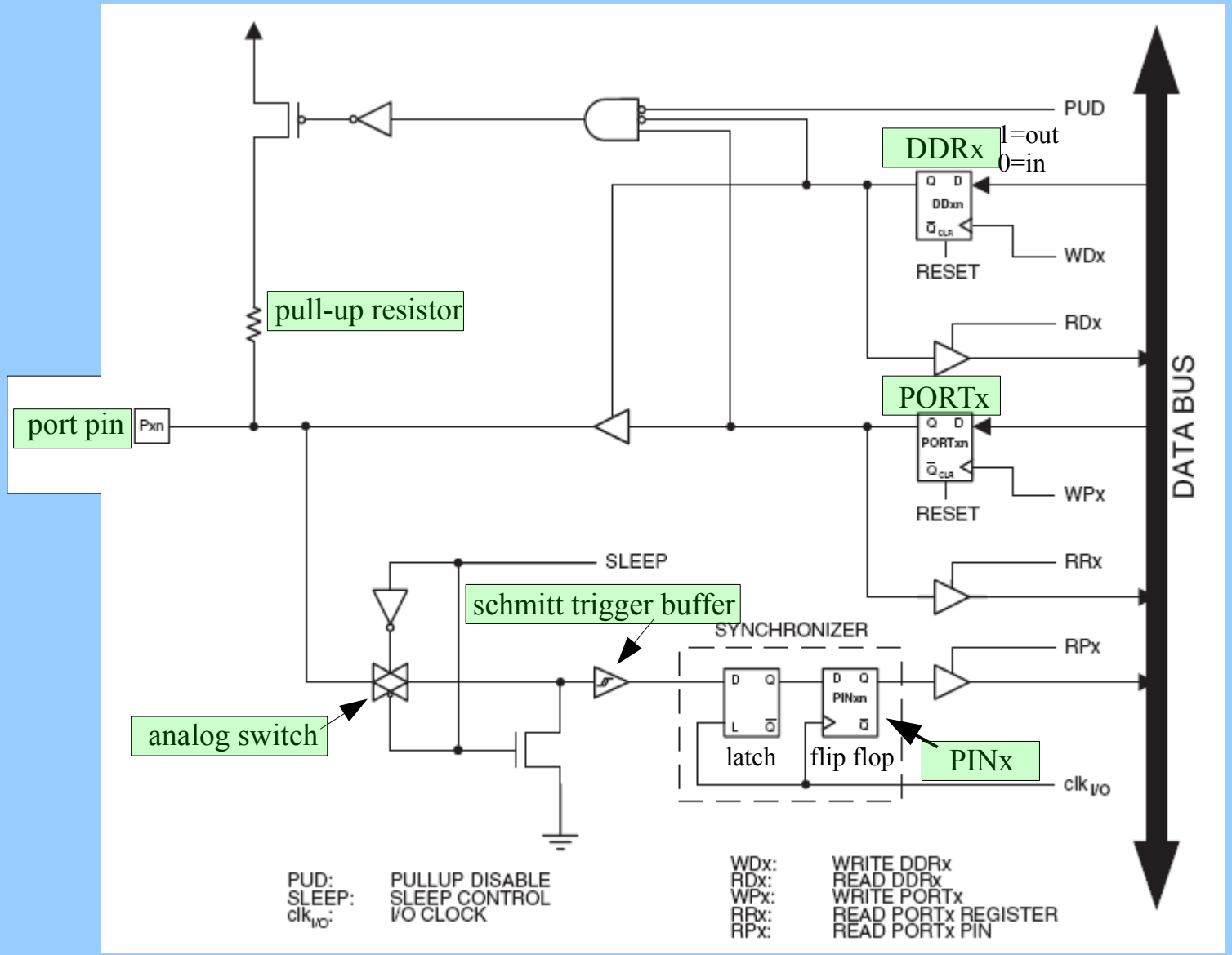
Control of all ports and pins is done with three registers

-DDRx (i.e., DDRB is *data direction register* port B)
-PORTx (i.e. PORTB is the *output register* for port B)
-PINx (i.e. PINB in the *input register* for port B)

All of these ports may be read. Writing the PINx register does nothing.

ATMega128 I/O Ports

AVR port architecture



ATMega128 I/O Ports

AVR I/O port usage

bit 0: output mode, logic '1' asserted, if PINB.0 is read, it returns a '1'
bit 1: output mode, logic '0' asserted, if PINB.1 is read, it returns a '0'
bit 2: input mode, no pullup resistor, if PINB2 read returns state of pin
bit 3: input mode, pullup resistor on, thus if PINB3 is read it.....
returns a '0' if pin is driven '0'
returns a '1' if the pin is not driven

	7	6	5	4	3	2	1	0
DDRB	0	0	0	0	0	0	1	1

	7	6	5	4	3	2	1	0
PORTB	0	0	0	0	1	0	0	1

	7	6	5	4	3	2	1	0
PIN B	?	?	?	?	?	?	0	1

ATMega128 I/O Ports

AVR I/O port usage

DDRx	selects pin direction (in or out)
PORTx	determines the driven pin value if the pin is an output determines if a pullup is present if the pin is an input
PINx	holds the value of the pin

Port usage fine points

Regardless of the setting of the DDRx register, the port pin can be read from PINx Thus, an driven output value in PORTx can always be read in PINx.

When the "pull-up disable bit" in the Special Function I/O Register (SFIOR) is set, all pull-ups are disabled regardless of the setting of DDRx and PORTx. Pullups are also disabled during reset.

Input pins have a 1.5 clock cycle delay before a new value can be read

-1 NOP instruction necessary to read updated pin

Use pull-ups on unused I/O pins to lower power consumption.

Using alternative functions of some port pins does not effect other pins.

ATMega128 I/O Ports

AVR I/O port programming

In the file, $install_path/avr/include/avr/iom128.h, define statements are used to make shorthand notation for ports and bits. For example.....

```
/*Data Register, Port B */
#define PORTB    _SFR_IO8(0x18)
```

and also.....

```
/*Port B Data Register  -  PORTB */
#define    PB7     7
#define    PB6     6
#define    PB5     5
#define    PB4     4
#define    PB3     3
#define    PB2     2
#define    PB1     1
#define    PB0     0
```

ATMega128 I/O Ports

AVR I/O port programming

These `#defines` allow us to write our programs in a more readable way.
What is better?

```
SPCR = 0x50; //what is going on here??
```

OR

```
SPCR = (1<<SPE)|(1<<MSTR); //SPI enable, Master mode
```

These generate identical code, but one is much clearer.
How about when bits don't have a named function, just a position?

```
DDRB = 0x05; //quick, what bits are set?
DDRB = (1<<DDB2) | (1<<DDB0); //clearer
DDRB = _BV(DDB2) | _BV(DDB0); //using a libc macro
```

`_BV(argument)` creates a bit vector that has the bit specified by arguement set to logic one.

ATMega128 I/O Ports

AVR I/O port programming

By using AND, OR and XOR, we can manipulate individual bits

```
//toggle bit 5
PORTB = PORTB ^ 0x20; // invert
PORTB ^= 0x20;          // invert again another way


//set bits 7 and 2, don't bother others
PORTB = PORTB | 0x84;
PORTB |= (1<<PB7)|(1<<PB2) ; //shorter, more portable


//clear bit 0 and 1, but nothing else
PORTB = PORTB & 0xfc; //one way
PORTB &= ~0x03;         //maybe clearer
PORTB &= ~((1<<PB0)|(1<<PB1)); //more portable
```

ATMega128 I/O Ports

AVR I/O port programming

By using a “mask” we can get the value of individual bits.

```
//looking for bit zero of port D to be a one
 if(PIND & 0x01){take_some_action();}
```

The value 0x01 as used here is called a “mask”. It allows us to zero out the other bits and determine if one particular bit is a one. We can look for a zero also....

```
//looking for bit 5 of port B to be a zero
 if(~PIND & 0x20){take_action();}
```

Best yet is to use the avr-libc functions *bit_is_set()* and *bit_is_clear()*:

```
if (bit_is_set(PINC, PC2) {return 0;}
while (bit_is_clear(SPSR,SPIF)) {}
```

ATMega128 I/O Ports

AVR I/O port programming

Assume (correctly!) that after reset, the following is true:
DDRD = 0x00
DDRB = 0x00

Assume a mega128 board has a normally open switch attached to port D bit zero. When the switch is closed, the port D bit is connected to ground.

Write code that reads port D bit zero, inverts its value and outputs that value to port B bit 0. Do not disturb the values of any other pins.
(5 minutes)

ATMega128 I/O Ports

AVR I/O port programming

Assume a mega128 board has a normally open switch attached to port D bit zero. When the switch is closed, the port D bit is connected to ground.

Write code that reads port D bit zero, inverts its value and outputs that value to port B bit 0. Do not disturb the values of any other pins.

```
#include <avr/io.h>
//read port d bit 0, invert and output to port b bit 0
int main(){
  DDRD  &= 0xFE;  //make port d bit zero input mode
  PORTD |= 0x01;  //turn on bit 0 pull-up
  DDRB  |= 0x01;  //set port b bit 0 to output mode
  while(1){
  if(PIND & 0x01){PORTB &= 0xFE;}
  else           {PORTB |= 0x01;}
  }//while
}//main
```

ATMega128 I/O Ports

AVR I/O port DC characteristics

The Absolute Maximum Ratings are not where you want to operate an IC

Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-0.5V to V_{CC}+0.5V
Voltage on $\overline{\text{RESET}}$ with respect to Ground	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0 mA
DC Current V_{CC} and GND Pins	200.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Where do these ratings come from?

ATMega128 I/O Ports

AVR I/O port DC output characteristics

Output Buffer characteristics (mega128 datasheet page 321)

V_{IH2}	Input High Voltage	RESET pin	0.85 V_{CC}[1]		V_{CC} + 0.5	V
V_{OL}	Output Low Voltage[3] (Ports A,B,C,D, E, F, G)	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 3V			0.7 0.5	V V
V_{OH}	Output High Voltage[4] (Ports A,B,C,D, E, F, G)	I_{OH} = -20 mA, V_{CC} = 5V I_{OH} = -10 mA, V_{CC} = 3V	4.2 2.4			V V

Always read the footnotes

2. Min means the lowest value where the pin is guaranteed to be read as high
3. Although each I/O port can sink more than the test conditions (20 mA at V_{CC} = 5V, 10 mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   TQFP and MLF Package:
   1] The sum of all IOL, for all ports, should not exceed 400 mA.
   2] The sum of all IOL, for ports A0 - A7, G2, C3 - C7 should not exceed 300 mA.
   3] The sum of all IOL, for ports C0 - C2, G0 - G1, D0 - D7, XTAL2 should not exceed 150 mA.
   4] The sum of all IOL, for ports B0 - B7, G3 - G4, E0 - E7 should not exceed 150 mA.
   5] The sum of all IOL, for ports F0 - F7, should not exceed 200 mA.
   If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
4. Although each I/O port can source more than the test conditions (20 mA at Vcc = 5V, 10 mA at Vcc = 3V) under steady state conditions (non-transient), the following must be observed:
   TQFP and MLF Package:
   1] The sum of all IOH, for all ports, should not exceed 400 mA.
   2] The sum of all IOH, for ports A0 - A7, G2, C3 - C7 should not exceed 300 mA.
   3] The sum of all IOH, for ports C0 - C2, G0 - G1, D0 - D7, XTAL2 should not exceed 150 mA.
   4] The sum of all IOH, for ports B0 - B7, G3 - G4, E0 - E7 should not exceed 150 mA.
   5] The sum of all IOH, for ports F0 - F7, should not exceed 200 mA.
   If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.

ATMega128 I/O Ports

AVR I/O port DC input characteristics

Input Buffer characteristics (mega128 datasheet page 321)

T_A = -40°C to 85°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{IL}	Input Low Voltage	Except XTAL1 and RESET pins	-0.5		0.2 V_{CC}(1)	V
V_{IL1}	Input Low Voltage	XTAL1 pin, External Clock Selected	-0.5		0.1 V_{CC}(1)	V
V_{IL2}	Input Low Voltage	RESET pin	-0.5		0.2 V_{CC}(1)	V
V_{IH}	Input High Voltage	Except XTAL1 and RESET pins	0.6 V_{CC}(2)		V_{CC} + 0.5	V
V_{IH1}	Input High Voltage	XTAL1 pin, External Clock Selected	0.7 V_{CC}(2)		V_{CC} + 0.5	V
V_{IH2}	Input High Voltage	RESET pin	0.85 V_{CC}(2)		V_{CC} + 0.5	V
	Output Low Voltage(3)					
I_{IL}	Input Leakage Current I/O Pin	Vcc = 5.5V, pin low (absolute value)			1.0	µA
I_{IH}	Input Leakage Current I/O Pin	Vcc = 5.5V, pin high (absolute value)			1.0	µA
R_{RST}	Reset Pull-up Resistor		30		60	kΩ
R_{PEN}	PEN Pull-up Resistor		30		60	kΩ
R_{PU}	I/O Pin Pull-up Resistor		20		50	kΩ
	Initialization Delay	V_{CC} = 5.0V			500	

Notes: 1. "Max" means the highest value where the pin is guaranteed to be read as low
2. "Min" means the lowest value where the pin is guaranteed to be read as high

ATMega128 I/O Ports

AVR I/O port interfacing

Pull-ups are handy for terminating switch terminals but watch the leakage current!

- -Quadrature encoder (90 deg. phased outputs)
- -How would you hook it up?
- -What about protection of output drivers?
- -What about protection for input buffers?

Led Drive Circuits

- -Highside or lowside drive
- -How do you determine the correct values of current limit resistors?
- -When do you need external drive help with a transistor?

Motor or relay drive circuits

- -catch diodes for inductive kickback

Interface directly to a speaker? Can it be done?

5V to 3.3V or 3.3V to 5V interfacing

- -resistor and protection diode,or two resistors, or resistor and zener diode?
- -74LVC244, TXB0108