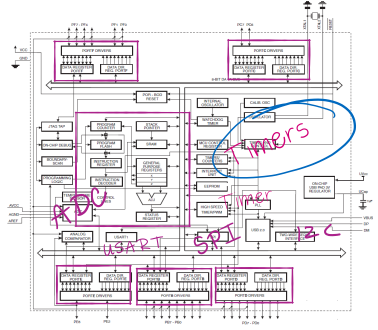
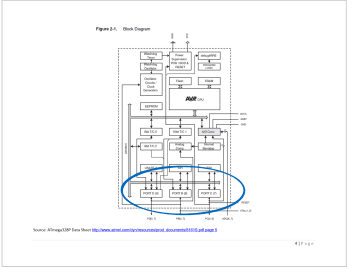
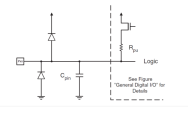


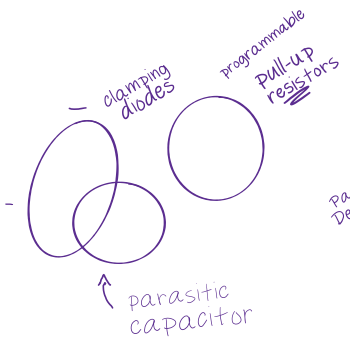


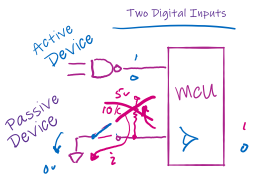
ATMEGA GENERAL PURPOSE DIGITAL I/O PORTS

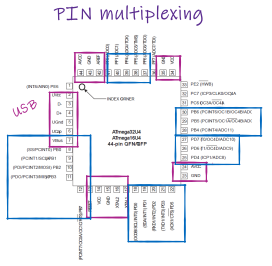
Reading: Section 4.1 I/O port pins and their functions

- The ATmega328P has 23 General Purpose Digital I/O Pins assigned to 5 **GPIO Ports** (8-bit Ports ~~B, D and Not Port C~~) B,C,D,E,F
- Each I/O port pin may be configured as an **output** with symmetrical drive characteristics. Each pin driver is strong enough (**20 mA**) to drive LED displays directly.
- Each I/O port pin may be configured as an **input** with or without a **pull-up resistors**. The values for the pull up resistors can range from 20 - 50 K ohms.
- Each I/O pin has **clamping diodes** to protect input circuit from undervoltage/overvoltage and ESD conditions.



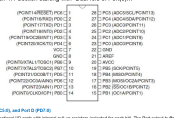






DUAL ROLE OF PORTS B, C AND D OF THE ATMEGA328P

Reading: Replaces each 4.1 Section starting with "Dual role of Port(x)..."



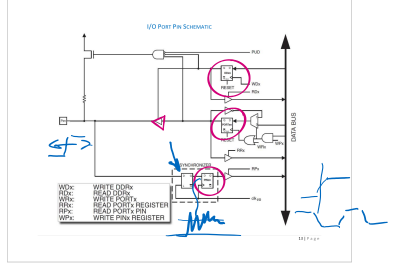
I/O Ports B (PB7:0), Port C (PC6:0), and Port D (PD7:0):
Ports B, C, and D are bi-directional I/O ports with internal pull-up resistors (selected for each bit). The Port output buffers have symmetrical drive characteristics with both high sink and source capability.

Interrupts (INT1, INT0, PCINT23..0)
External 4-ban pin, are triggered by the INT0 and INT1 pins or any of the PCINT23..0 pins. Observe that, if enabled, the interrupts will trigger even if the INT0 and INT1 or PCINT23..0 pins are configured as outputs. This feature provides a way of generating a software interrupt.

AVCC
AVCC is the supply voltage pin for the A/D Converter. It should be externally connected to VCC, if the ADC is used, it should be connected to VCC through a low-pass filter.

AREF
AREF is the analog reference pin for the A/D Converter.

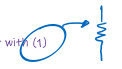
ADC5:0
These pins serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

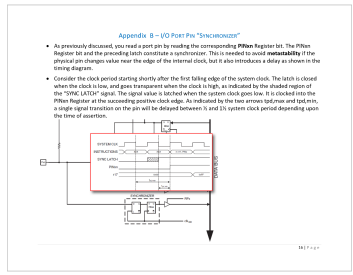


I/O PORT PIN CONFIGURATIONS

Inputs		Outputs		
DDRXn	PORTXn	I/O	Pull-up	Comments
0	0	Input	No	Read "Synchronized" PINXn
0	1	Input	Yes	
1	X	Output	N/A	Write bit to PORTXn

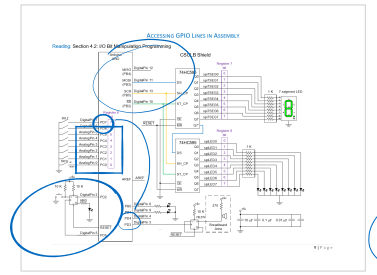
DDR: Is the pin an input (0) or an output (1)
PORT: If an input do you want it with (1) or without (0) a pull-up resistor.





ACCESSING GPIO LINES IN ASSEMBLY

Reading: Section 4.2 I/O Bit Manipulation Programming

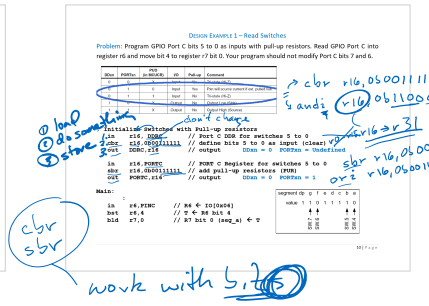


DESIGN EXAMPLE 1 – READ SWITCHES

Problem: Program GPIO Port C bits 5 to 0 as inputs with pull-up resistors. Read GPIO Port C into register r6 and move bit 4 to register r7 bit 0. Your program should not modify Port C bits 7 and 6.

DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

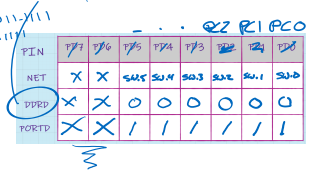
```
;Initialize Switches with Pull-up resistors
    in   r16, DDRC      ; Port C DDR for switches 5 to 0
    cbr  r16,0b00111111 ; Define bits 5 to 0 as input (clear) I/O
    out  DDRC,r16       ; output          DDxn = 0  PORTxn = Undefined

    in   r16,PORTC      ; PORT C Register for switches 5 to 0
    sbr  r16,0b00111111 ; add pull-up resistors (PUD)
    out  PORTC,r16      ; output          DDxn = 0  PORTxn = 1

Main:
    .
    .
    in   r6,PINC        ; R6 <- IO(0x06)
    bst  r6,4           ; T <- R6 bit 4
    bld  r7,0           ; R7 bit 0 (msg_a) <- T
```



work with 5,7,6

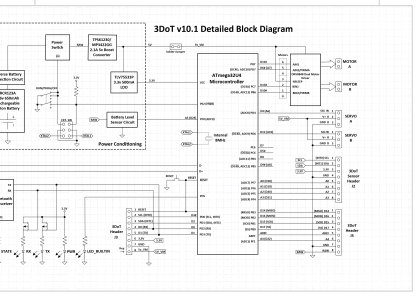
Modify all 8-bits

```
ldi  r16,0b00111111
out  PORTD,r16
```



I/O PORT PIN CONFIGURATIONS

Inputs		Outputs		
DDRXn	PORTXn	I/O	Pull-up	Comments
0	0	Input	No	Read "Synchronized" PINXn
0	1	Input	Yes	
1	X	Output	N/A	Write bit to PORTXn



Port B

PIN	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
					J5-1	J5-2	J5-3	J5-4
NET	AIN1	BIN2	NC	NC	MISO	MOSI	SCK	SS
DDRB	1	1	0	0	X	X	X	X
PORTB	0	0	1	1	X	X	X	X

```
/* undefine pins are not changed */
/* pins that are not connected are defined as inputs with pull-up resistors */
DDRB  &= 0x3F;  // clear most significant nibble
DDRB  |= 0xC0;  // set bit pattern
PORTB &= 0x3F;  // clear most significant nibble
PORTB |= 0xC0;  // set bit pattern
```

Port C

PIN	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
NET	BIN1	NC	–	–	–	–	–	–
DDRC	1	0	–	–	–	–	–	–
PORTC	0	1	–	–	–	–	–	–

```
DDRC  &= 0x3F;  // clear most significant 2 bits
DDRC  |= 0x80; // mix bit pattern
PORTC &= 0x3F;  // clear most significant 2 bits
PORTC |= 0x40;
```

Port D

PIN	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
		Servo A	LED		Servo B	J1-5	J1-4	J3-3	J3-2
NET	AIN2	SIG	BUILTIN		SIG	TX	RX	SDA	SCL
DDRD	1	1	1		1	X	X	X	X
PORTD	0	0	0		0	X	X	X	X

```
DDRD  |= 0xF0; // mix most significant nibble
PORTD &= 0x0F; // clear most significant nibble
```

Port E

PIN	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
		S-1						
NET	–	NC	–	–	–	SW	–	–
DDRE	–	0	–	–	–	0	–	–
PORTE	–	1	–	–	–	0	–	–

```
DDRE  &= 0x0B;  // clear bits 6 and 2
PORTE &= 0x0B;  // clear bits 6 and 2
PORTE |= 0x40;  // set bit pattern
```

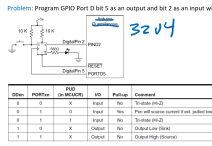
Port F

PIN	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
	J2-5	J2-6	J2-7	J2-8			J2-9	
NET	A0	A1	A2	A3	–	–	A4	BATT
DDRF	X	X	X	X	–	–	X	0
PORTF	X	X	X	X	–	–	X	0

```
DDRF |= 0xFE;
PORTF |= 0xFE;
```

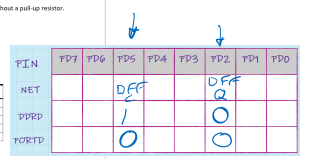
DESIGN EXAMPLE 2 – CONFIGURE D FLIP-FLOP

Problem: Program GPIO Port D bit 5 as an output and bit 2 as an input without a pull-up resistor.



DDxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	X	Input	No	Tri-state (Hi-Z)
0	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	Input	No	Tri-state (Hi-Z)
1	X	Output	No	Output Low (Sink)
1	X	Output	No	Output High (Source)

```
; Pushbutton debounce port D pins
  EQU dff_clk=PORTD5  ; clock of debounce flip-flop
  EQU dff_Q=PIND2     ; Q output of debounce flip-flop

; Initialize push-button debounce circuit
  cbi  DDRD,dff_clk   ; flip-flop clock,  DDRD5 = 1  PORTD5 = Undefined
  cbi  PORTD,dff_clk  ;                   DDRD5 = 1  PORTD5 = 0
  cbi  DDRD,dff_Q     ; flip-flop Q       DDRD2 = 0  PORTD2 = Undefined
  cbi  PORTD,dff_Q    ; flip-flop Q       DDRD2 = 0  PORTD2 = 0
```



0 - 31 addresses in the I/O Address space

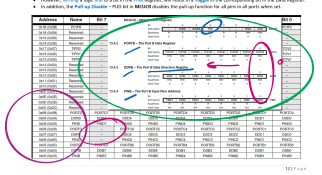
sbi cbi

works with a bit

REGISTER SUMMARY AND THE I/O PORT

Reading: Section 6.4 Pertinent Register Descriptions

- Three I/O memory address locations are allocated for each port, one each for the Data Register – **PORTx**, Data Direction Register – **DDRx**, and the Port Input Pins – **PINx**.
- The Port Input Pins I/O location **PINx** is Read Only, while the Data Register and the Data Direction Register are read/write.
- However, Writing a logic one to a bit in the **PINx** Register, will result in a Toggle in the corresponding bit in the Data Register.
- In addition, the **Pull-up Disable** – PUD bit in **MCUCR** disables the pull-up function for all pins in all ports when set.



convert to assembly

use the Simulator