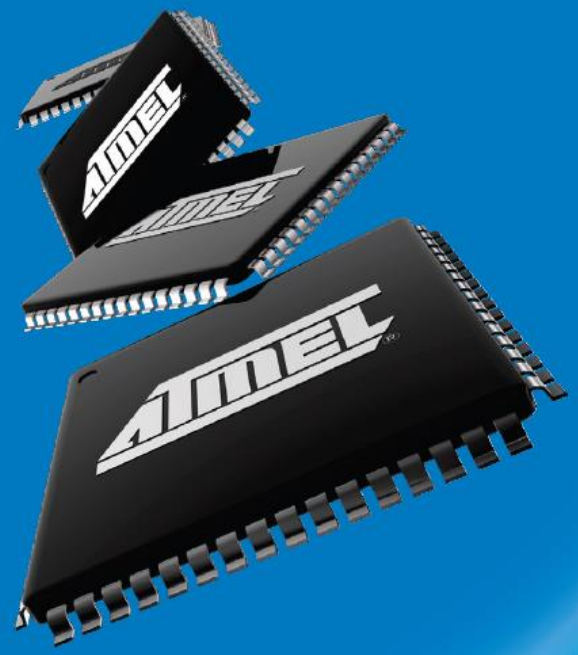
AVR®

8-bit Microcontrollers

AVR®32

32-bit Microcontrollers and Application Processors



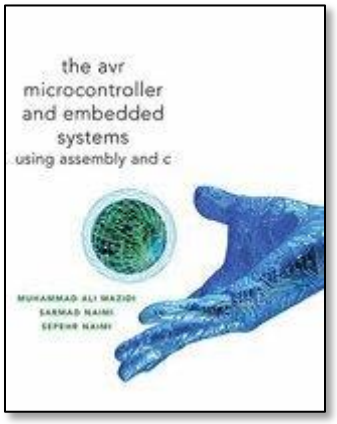
AVR Peripherals

February 2009

Everywhere You Are®

General-Purpose Input/Output

READING



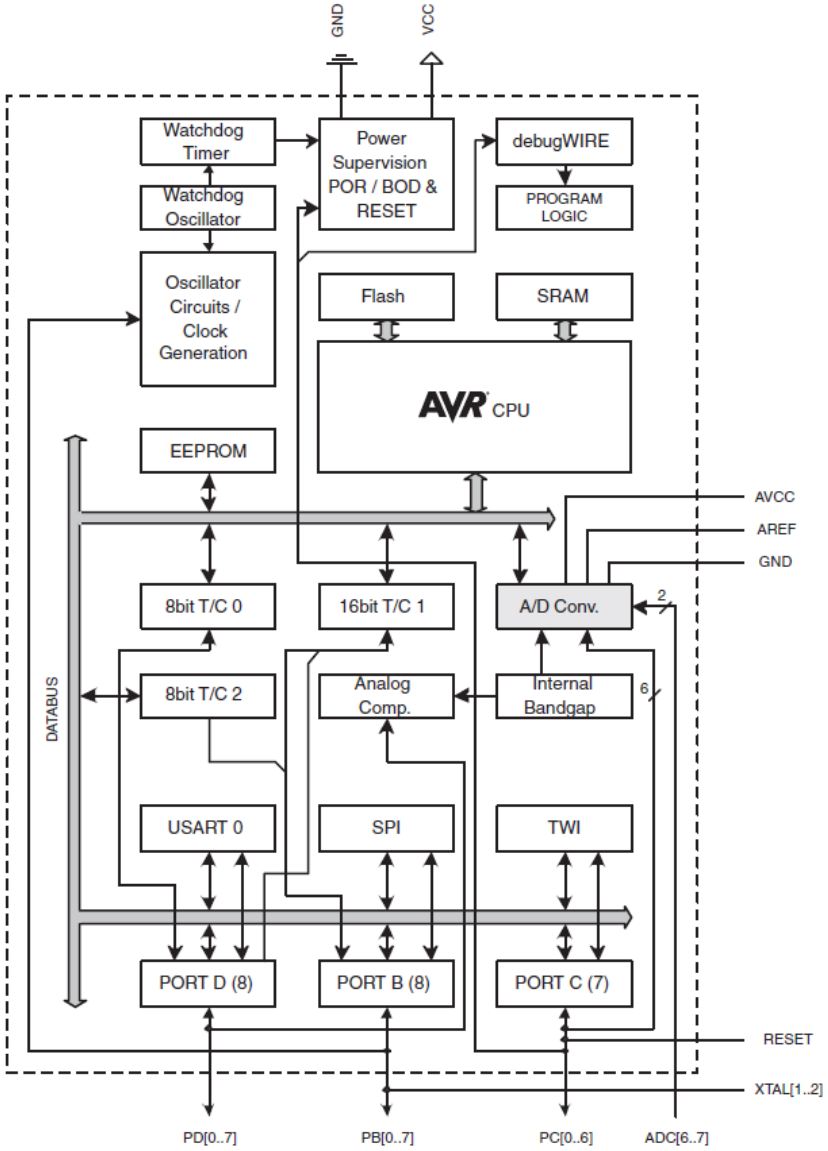
The AVR Microcontroller and Embedded Systems using Assembly and C)
by Muhammad Ali Mazidi, Sarmad Naimi, and Sepehr Naimi

Chapter 7: AVR Programming in C

Section 7.2: I/O Programming in C

TABLE OF CONTENTS

Figure 2-1. Block Diagram

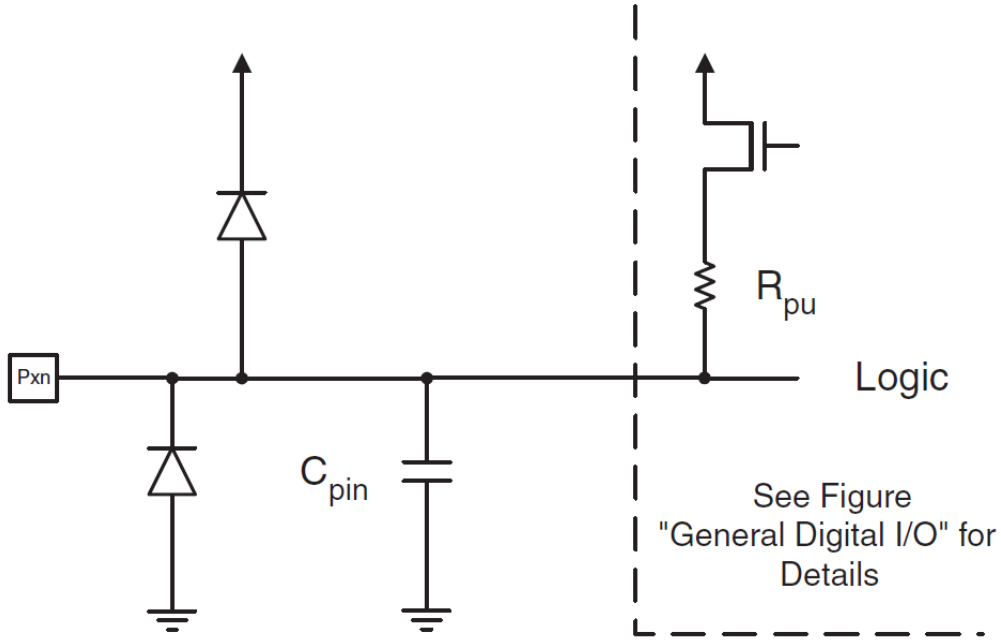


Source: ATmega328P Data Sheet http://www.atmel.com/dyn/resources/prod_documents/8161S.pdf page 5

ATMEGA GENERAL PURPOSE DIGITAL I/O PORTS

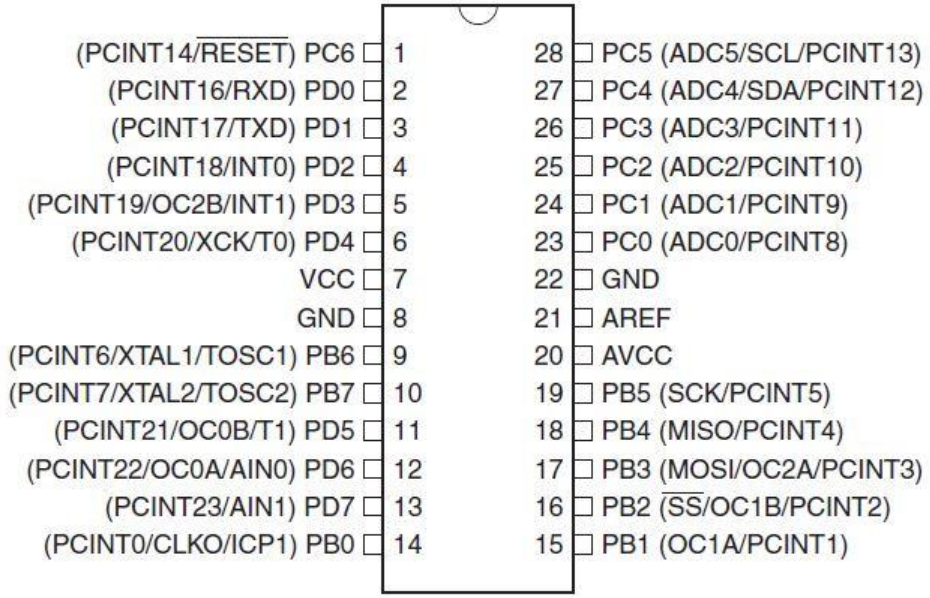
Reading: Section 6.1.1 Introduction

- The ATmega328P has 23 General Purpose Digital I/O Pins assigned to 3 Ports (8-bit Ports B, D and 7-bit Port C)
- 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 protection diodes to both VCC and Ground



PIN DESCRIPTION OF THE ATMEGA328P

Reading: Section 6.1.3 Pin-Muxing



I/O Ports B (PB7:0), Port C (PC5:0), 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 Interrupts 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

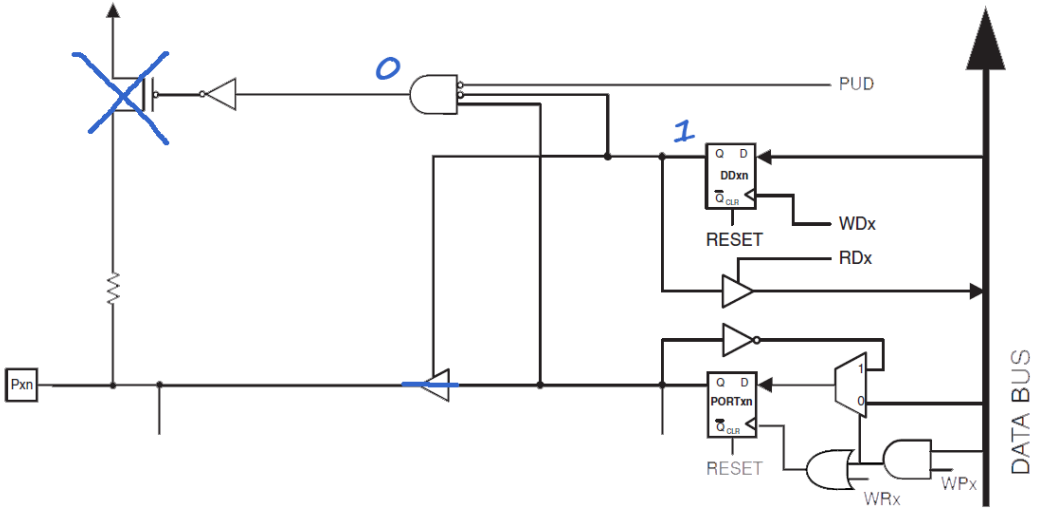
These pins serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

I/O Port Pin as an Output

Reading: Section 6.1.2 Basic Operation

- To configure a Port (x) pin as an output set corresponding bit (n) in the Data Direction Register (**DDxn**) to 1. Once configured as an output pin, you control the state of the pin (1 or 0) by writing to the corresponding bit (n) of the **PORTxn** register.

- Writing (signal **WPx**) a logic one to **PINxn toggles** the value of PORTxn, independent on the value of DDxn. Note that the SBI instruction can be used to toggle one single bit in a port.

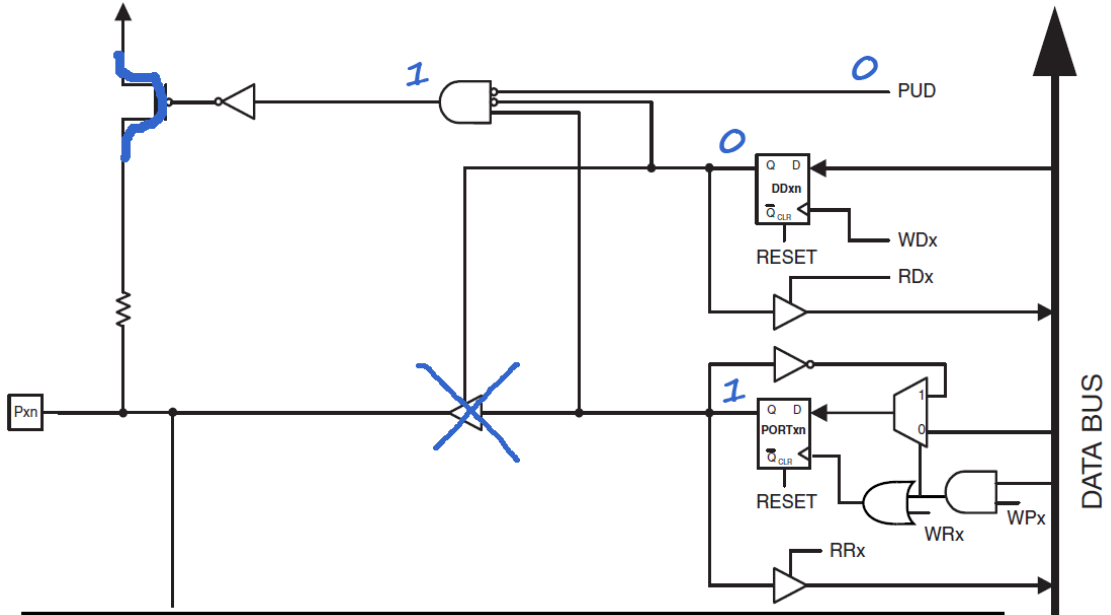


DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

I/O Port Pin as an Input

Reading: Section 6.2 Input

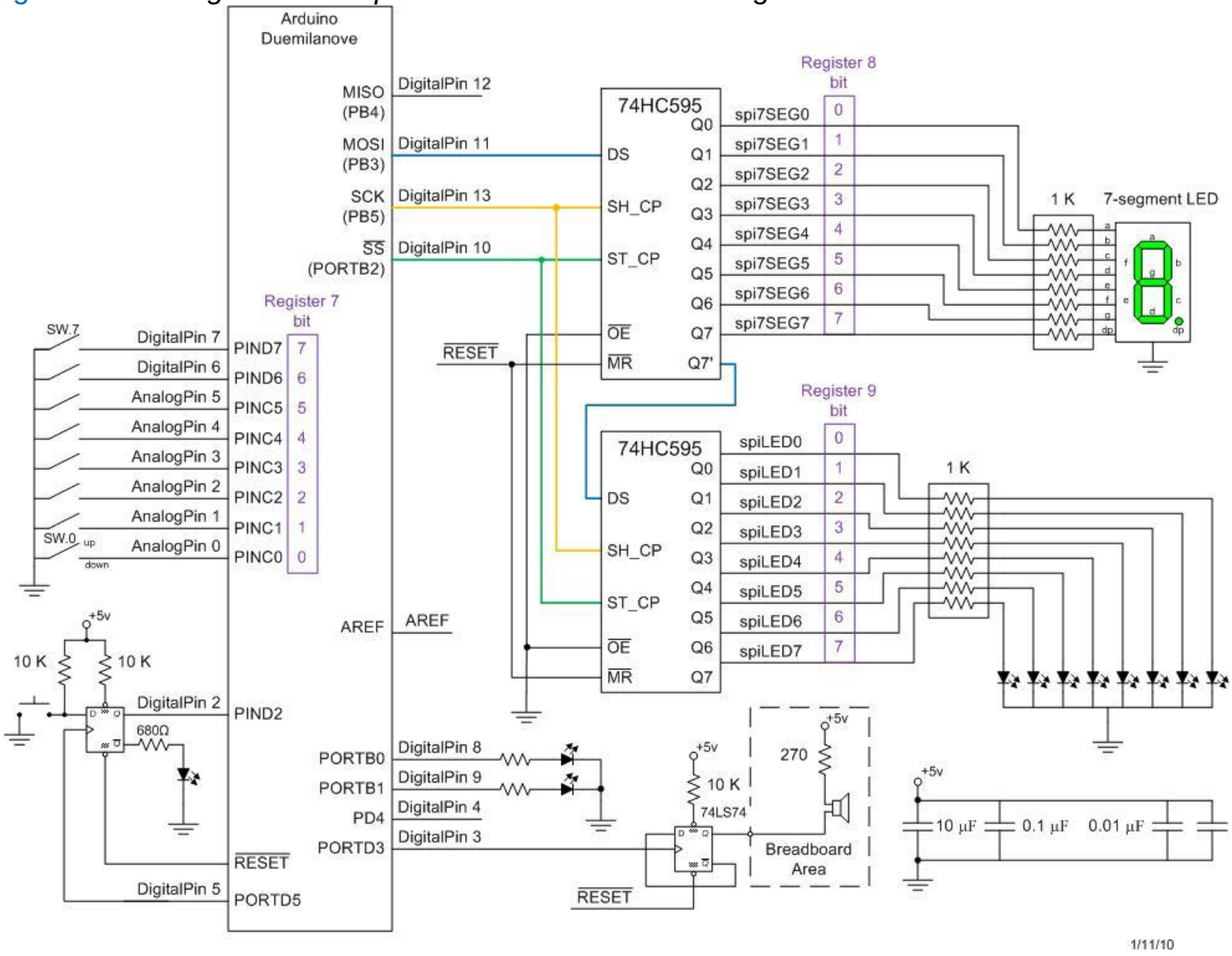
- To configure a Port (x) pin as an input set corresponding bit (n) in the Data Direction Register (**DDxn**) to 0. To add a pull-up resistor set the corresponding bit (n) of the **PORTxn** register to 1 (see illustration).
- You can now read the state of the input pin by reading the corresponding bit (n) of the **PINxn** register.



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)

ACCESSING GPIO LINES IN ASSEMBLY

Reading: *The following material replaces* Section 6.3 Accessing GPIO In C



DESIGN EXAMPLE 1 – Read Switches

Problem: Program GPIO Port C bits 5 to 0 as inputs with pull-up resistors. For assembly version, read GPIO Port C into register r7 and move bit 4 to register r8 bit 0. Your programs should not modify Port C bits 7 and 6.

DDRxn	PORTxn	I/O	Pull-up	Comment
0	0	Input	No	
0	1	Input	Yes	
1	0	Output		Output Low (Sink)
1	1	Output		Output High (Source)

Arduino Script

```
// Analog pins0,1,2,3,4,5 map to Digital pins 14, 15, 16, 17, 18, 19
pinMode(14, INPUT_PULLUP);    // set pin to input with pullup resistor
```

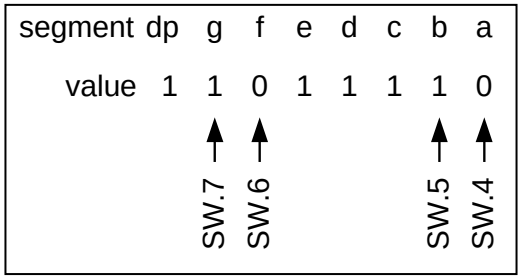
Repeat these 2 lines for pins 15, 16, 17, 18, and 19.

C++

```
DDRC &= ~0b00111111;
PORTC |= 0b00111111;
```

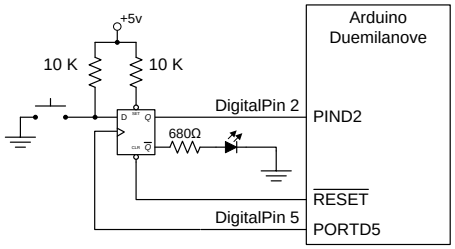
Assembly

```
; 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)
   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 (PUR)
   out   PORTC,r16        // output      DDxn = 0  PORTxn = 1

Main:
      :
   in    r7,0x06      // R7 ← PINC
   bst   r7,4         // T ← R7 bit 4
   bld   r8,0         // R8 bit 0 (seg_a) ← T
```



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.



DDRxn	PORTxn	I/O	Pull-up	Comment
0	0	Input	No	
0	1	Input	Yes	
1	0	Output		Output Low (Sink)
1	1	Output		Output High (Source)

Arduino Script

```
// Variable definitions
const int dff_clk = 5;       // D flip-flop clock wired to digital pin 5
const int dff_Q = 2;         // D flip-flop Q output wired to digital pin 2
// Initialize digital pin 5 (dff_clk) as an output and digital pin 2 (dff_Q) as an input
pinMode(dff_clk, OUTPUT);    // set pin as output
digitalWrite(dff_clk, LOW); // initialize to zero
pinMode(dff_Q, INPUT);       // set pin as input
```

C++

```
// Preprocessor directives
#define dff_clk PORTD5
#define dff_Q PIND2
// Initialize push-button debounce circuit
DDRD |= 1 << dff_clk;      // define bit 5 of Data Direction Register (DDR)PORT D as an output
PORTD &= ~(1 << dff_clk); // initialize to zero
DDRD &= ~(1 << dff_Q);     // define bit 2 of Data Direction Register (DDR)PORT D as an input
PORTD &= ~(1 << dff_Q);    // without a pull-up resistor
```

Assembly

```
; Assembly directives
.EQU dff_clk=PORTD5
.EQU dff_Q=PIND2
; Initialize push-button debounce circuit
  sbi   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    //                  DDRD2 = 0 PORTD2 = 0
```

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.

MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	–	BODS	BODSE	PUD	–	–	IVSEL	IVCE	MCUCR
Read/Write	R	R	R	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

13.4.2 PORTB – The Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

13.4.3 DDRB – The Port B Data Direction Register

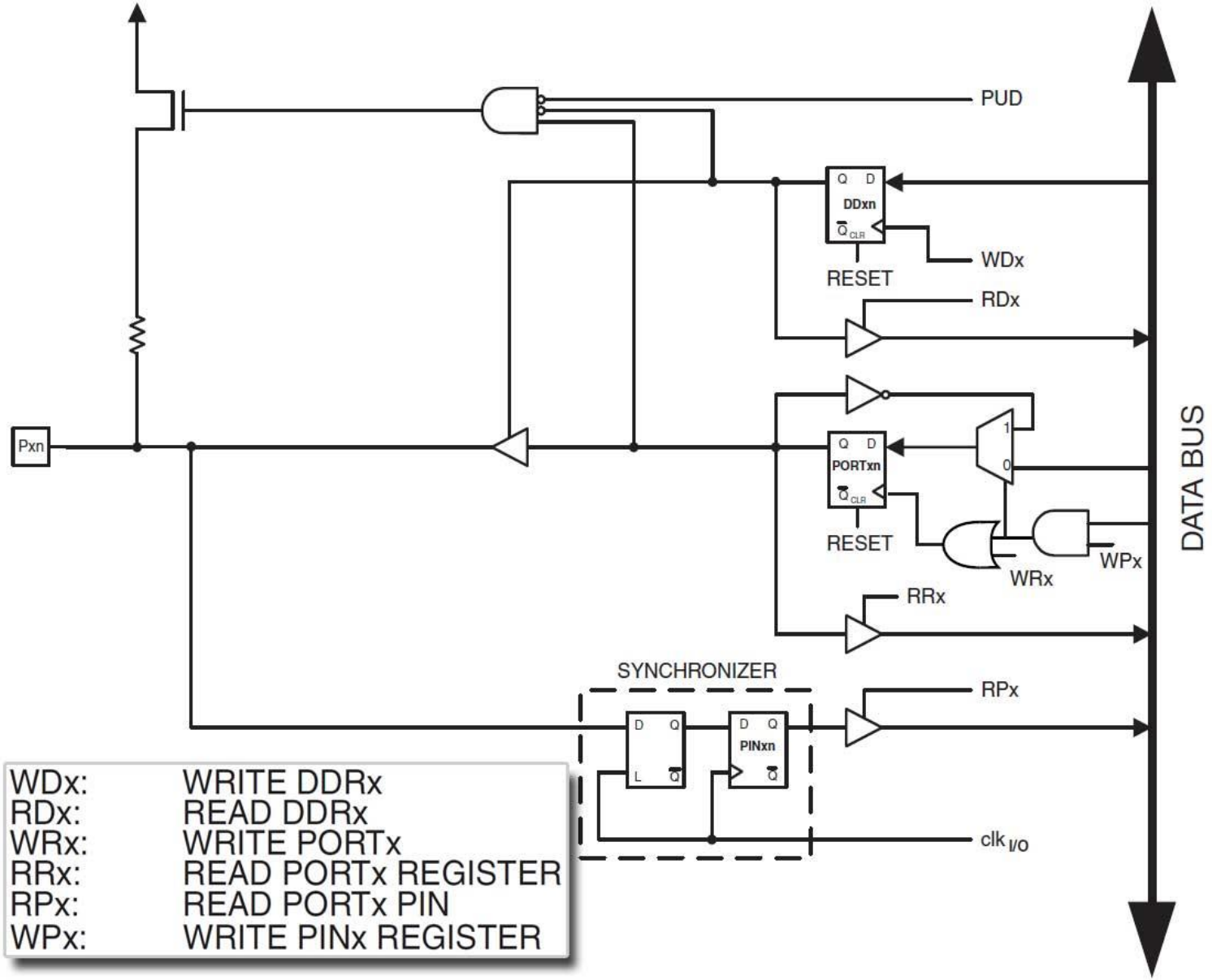
Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

13.4.4 PINB – The Port B Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Address	Name	Bit 7							Bit 0
0x1B (0x3B)	PCIFR	–							PCIF0
0x1A (0x3A)	Reserved	–							–
0x19 (0x39)	Reserved	–							–
0x18 (0x38)	Reserved	–							–
0x17 (0x37)	TIFR2	–							TOV2
0x16 (0x36)	TIFR1	–							TOV1
0x15 (0x35)	TIFR0	–							TOV0
0x14 (0x34)	Reserved	–							–
0x13 (0x33)	Reserved	–							–
0x12 (0x32)	Reserved	–							–
0x11 (0x31)	Reserved	–							–
0x10 (0x30)	Reserved	–							–
0x0F (0x2F)	Reserved	–							–
0x0E (0x2E)	Reserved	–							–
0x0D (0x2D)	Reserved	–							–
0x0C (0x2C)	Reserved	–							–
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
0x0A (0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
0x08 (0x28)	PORTC	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
0x07 (0x27)	DDRC	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0
0x06 (0x26)	PINC	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
0x04 (0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0

I/O PORT PIN SCHEMATIC



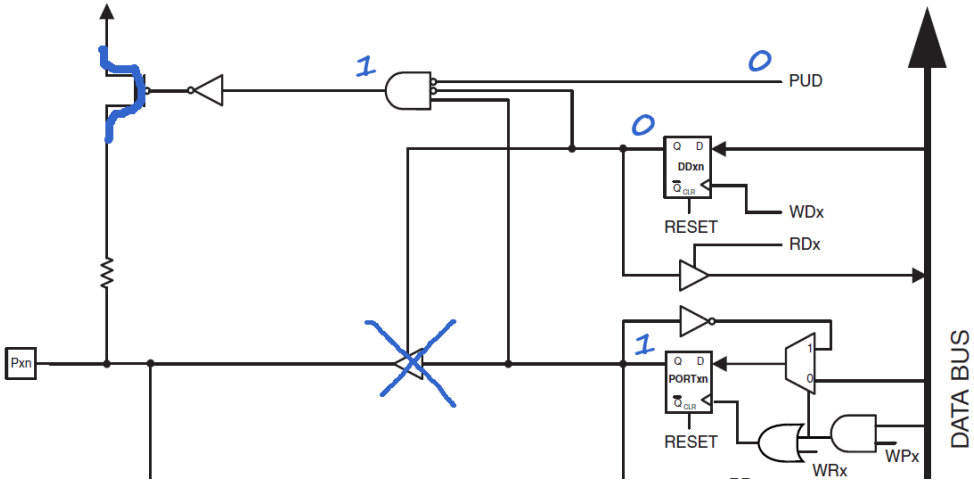
I/O Port Pin Configurations

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)

Simplified Version (PUD = 0)

DDRxn	PORTxn	I/O	Pull-up	Comment
0	**0**	**Input**	**No**	
0	**1**	**Input**	**Yes**	
1	**0**	**Output**		**Output Low (Sink)**
1	**1**	**Output**		**Output High (Source)**

APPENDIX A – PROGRAM I/O PORT AS AN INPUT USING MNEMONICS



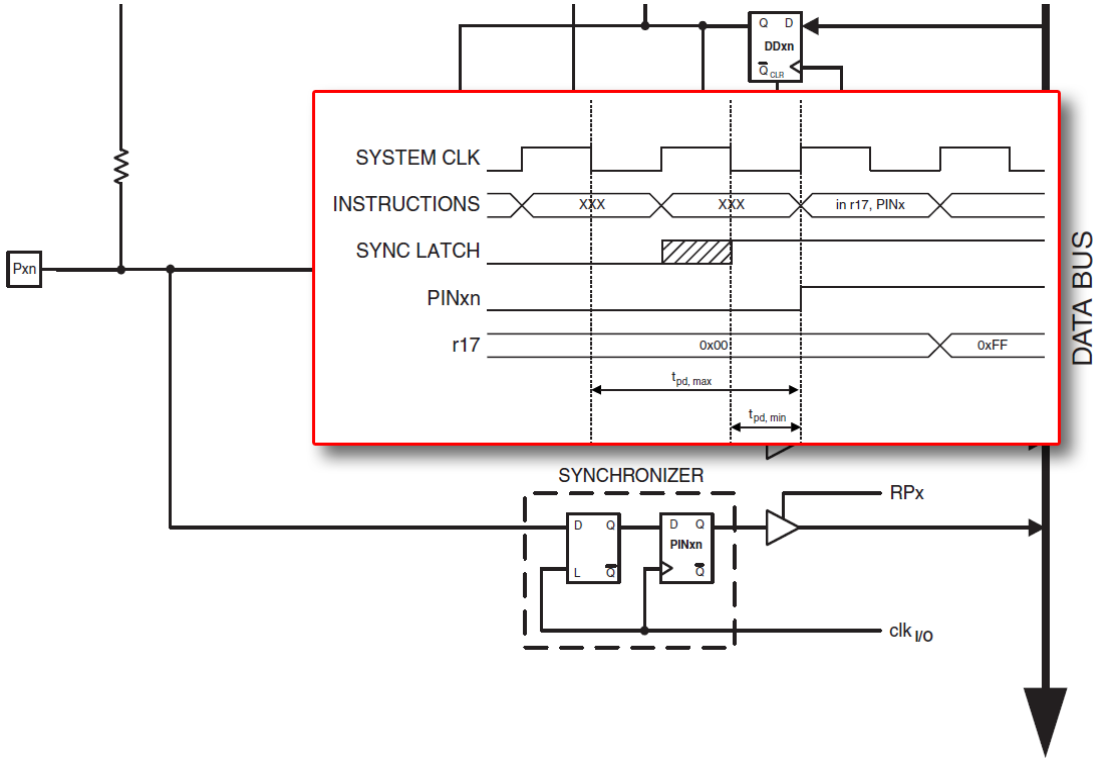

```
.INCLUDE <m328pdef.inc>
; C:\Program Files\Atmel\AVR Tools\AvrAssembler2\Appnotes\m328Pdef.inc

   in    r16,DDRC          // DDRC equated to 0x07 in m328Pdef.inc
   cbr   r16,(1<<PC5)|(1<<PC4)|(1<<PC3)|(1<<PC2)|(1<<PC1)|(1<<PC0)
   out   DDRC,r16          // output      DDxn = 0   PORTxn = Undefined
   in    r16,PORTC         // PortC equated to 0x08
   sbr   r16,(1<<PC5)|(1<<PC4)|(1<<PC3)|(1<<PC2)|(1<<PC1)|(1<<PC0)
   out   PORTC,r16         // output      DDxn = 0   PORTxn = 1

.INCLUDE "spi.inc"
```

The following Define and Equate Assembly Directives are defined in spi_shield.inc

```
.DEF spi7SEG=r8           // Text Substitution (copy-paste)
.DEF switch=r7
.EQU seg_a=0              // Numeric Substitution

   in    switch, PINC     // R7 ← PINC
   bst   switch,4         // T ← R7 bit 4
   bld   spi7SEG,seg_a    // R8 bit 0 ← T
```

Appendix B – I/O Port Pin "Synchronizer"

- As previously discussed, you read a port pin by reading the corresponding **PINxn** Register bit. The PINxn Register bit and the preceding latch constitute a synchronizer. This is needed to avoid **metastability** if the physical pin changes value near the edge of the internal clock, but it also introduces a delay as shown in the timing diagram.
- Consider the clock period starting shortly after the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the "SYNC LATCH" signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn Register at the succeeding positive clock edge. As indicated by the two arrows tpd,max and tpd,min, a single signal transition on the pin will be delayed between ½ and 1½ system clock period depending upon the time of assertion.



Appendix C – Switching Between I/O Port Pin Configurations

- When switching between tri-state ({DDxn, PORTxn} = 0b00) and output high ({DDxn, PORTxn} = 0b11), an intermediate state with either pull-up enabled ({DDxn, PORTxn} = 0b01) or output low ({DDxn, PORTxn} = 0b10) must occur.
- Switching between input with pull-up ({DDxn, PORTxn} = 0b01) and output low ({DDxn, PORTxn} = 0b10) generates the same problem. You must use either the tri-state ({DDxn, PORTxn} = 0b00) or the output high state ({DDxn, PORTxn} = 0b11) as an intermediate step.

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)