Microcontrollers 1

Port I/O

Christiaen Slot – c.g.m.slot@saxion.nl
Hans Stokkink – j.s.d.stokkink@saxion.nl

Kom verder. Saxion.

Kom verder. Saxion.

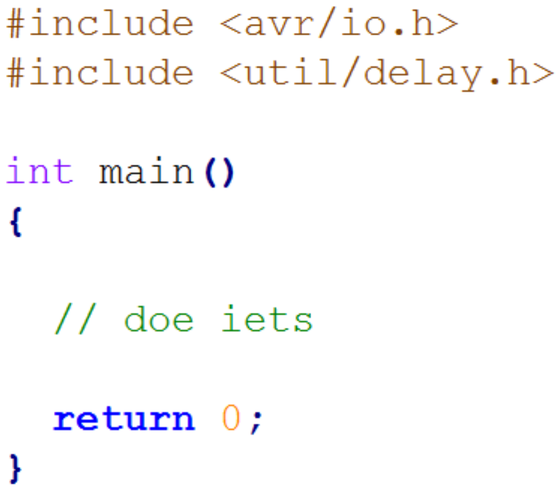
Content

- Difference arduino code and atmel Code
- What is a port
- I/O: Direct port manipulation

Direct C instead of Arduino

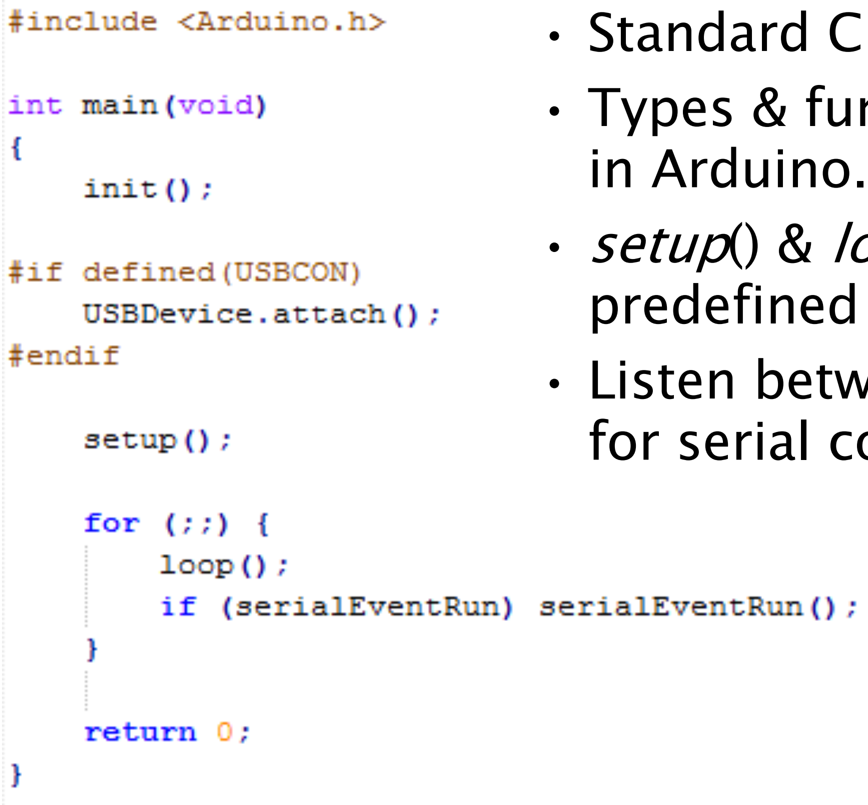
- No setup andloop
- No arduino libraries

- ‘c’ libraries
- main loop
- Direct port manipulation



```
#include <avr/io.h>
#include <util/delay.h>

int main()
{

   // doe iets

   return 0;
}
```

Arduino “main”



```
#include <Arduino.h>

int main(void)
{
    init();

#if defined(USBCON)
    USBDevice.attach();
#endif

    setup();

    for (;;) {
        loop();
        if (serialEventRun) serialEventRun();
    }

    return 0;
}
```

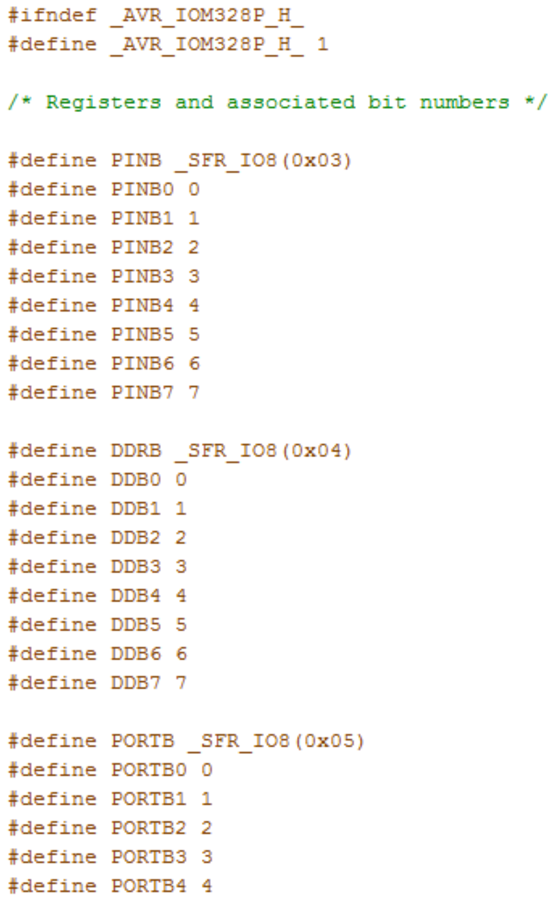
- Standard C code
- Types & functions in Arduino.h
- *setup*() & *loop*() predefined
- Listen between ‘loops’ for serial communication

io.h & iom328.h

Defines …

- Ports
- Constants
- Registers
- Etc.

For example:
#define F_CPU 16000000UL

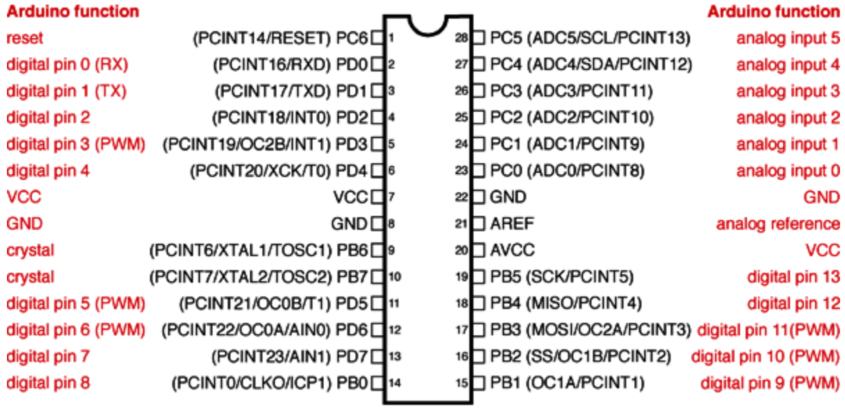


```
#ifndef _AVR_IOM328P_H_
#define _AVR_IOM328P_H_ 1

/* Registers and associated bit numbers */

#define PINB _SFR_IO8(0x03)
#define PINB0 0
#define PINB1 1
#define PINB2 2
#define PINB3 3
#define PINB4 4
#define PINB5 5
#define PINB6 6
#define PINB7 7

#define DDRB _SFR_IO8(0x04)
#define DDB0 0
#define DDB1 1
#define DDB2 2
#define DDB3 3
#define DDB4 4
#define DDB5 5
#define DDB6 6
#define DDB7 7

#define PORTB _SFR_IO8(0x05)
#define PORTB0 0
#define PORTB1 1
#define PORTB2 2
#define PORTB3 3
#define PORTB4 4
```

Pinmapping Atmega328

- Port B0-B5 → pin 8 – pin 13
- Port C0-C5 → pin A0 – pin A5
- Port D0-D7 → pin 0 – pin 7

- Port B6/7 → oscilator
- Port C6 → reset



Arduino function					Arduino function
reset	(PCINT14/RESET) PC6	1	28	PC5 (ADC5/SCL/PCINT13)	analog input 5
digital pin 0 (RX)	(PCINT16/RXD) PD0	2	27	PC4 (ADC4/SDA/PCINT12)	analog input 4
digital pin 1 (TX)	(PCINT17/TXD) PD1	3	26	PC3 (ADC3/PCINT11)	analog input 3
digital pin 2	(PCINT18/INT0) PD2	4	25	PC2 (ADC2/PCINT10)	analog input 2
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3	5	24	PC1 (ADC1/PCINT9)	analog input 1
digital pin 4	(PCINT20/XCK/T0) PD4	6	23	PC0 (ADC0/PCINT8)	analog input 0
VCC	VCC	7	22	GND	GND
GND	GND	8	21	AREF	analog reference
crystal	(PCINT6/XTAL1/TOSC1) PB6	9	20	AVCC	VCC
crystal	(PCINT7/XTAL2/TOSC2) PB7	10	19	PB5 (SCK/PCINT5)	digital pin 13
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5	11	18	PB4 (MISO/PCINT4)	digital pin 12
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6	12	17	PB3 (MOSI/OC2A/PCINT3)	digital pin 11(PWM)
digital pin 7	(PCINT23/AIN1) PD7	13	16	PB2 (SS/OC1B/PCINT2)	digital pin 10 (PWM)
digital pin 8	(PCINT0/CLKO/ICP1) PB0	14	15	PB1 (OC1A/PCINT1)	digital pin 9 (PWM)

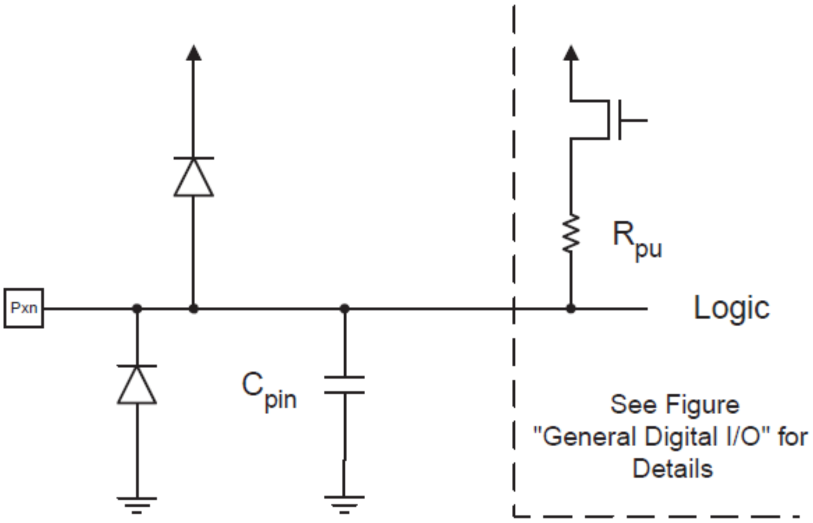
Digital Pins 11,12 & 13 are used by the ICSP header for MISO, MOSI, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.

Content

- Difference arduino code and atmel Code
- **What is a port**
- I/O: Direct port manipulation

Port overview

- Safety diodes/capacitors (spikes)
- Internal Pull up resistor
- Pin logic for input, output, tristate



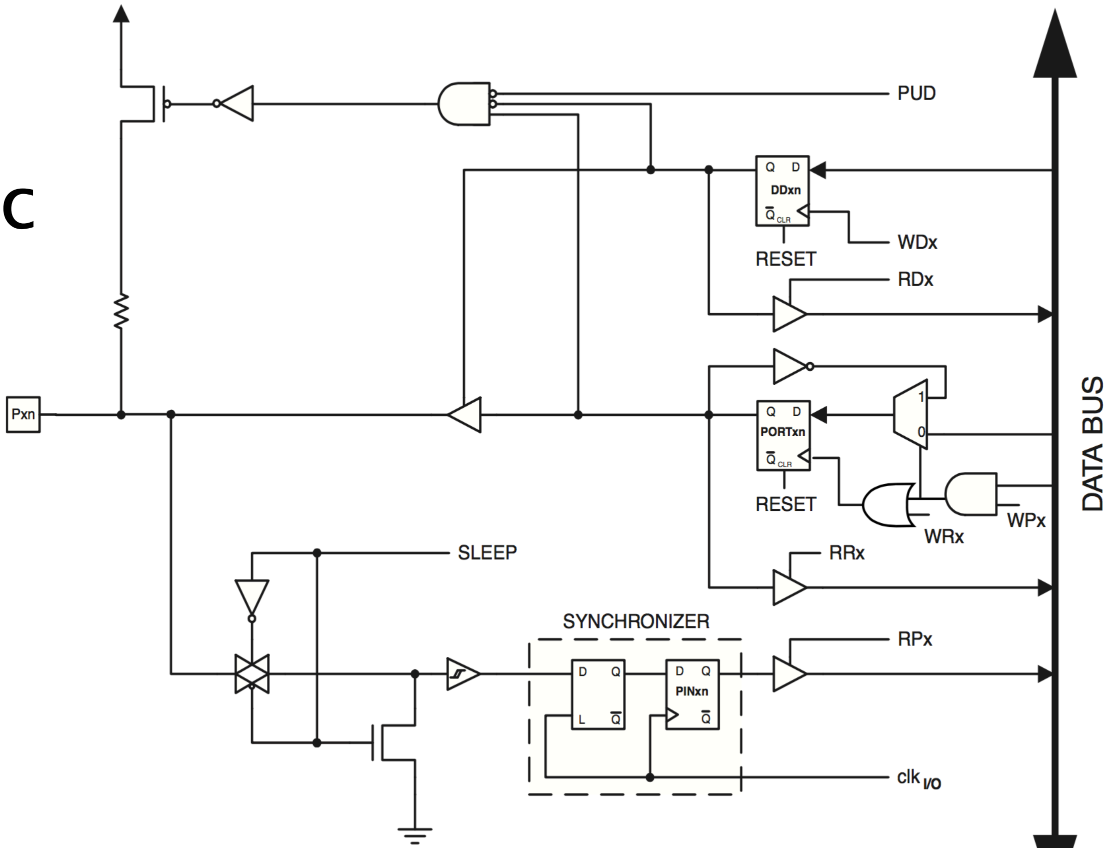
Port logic

Table 13-1. 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)

- Note: most pins have alternate functions, f.i. the SPI and I^{2}C interface or interrupts.

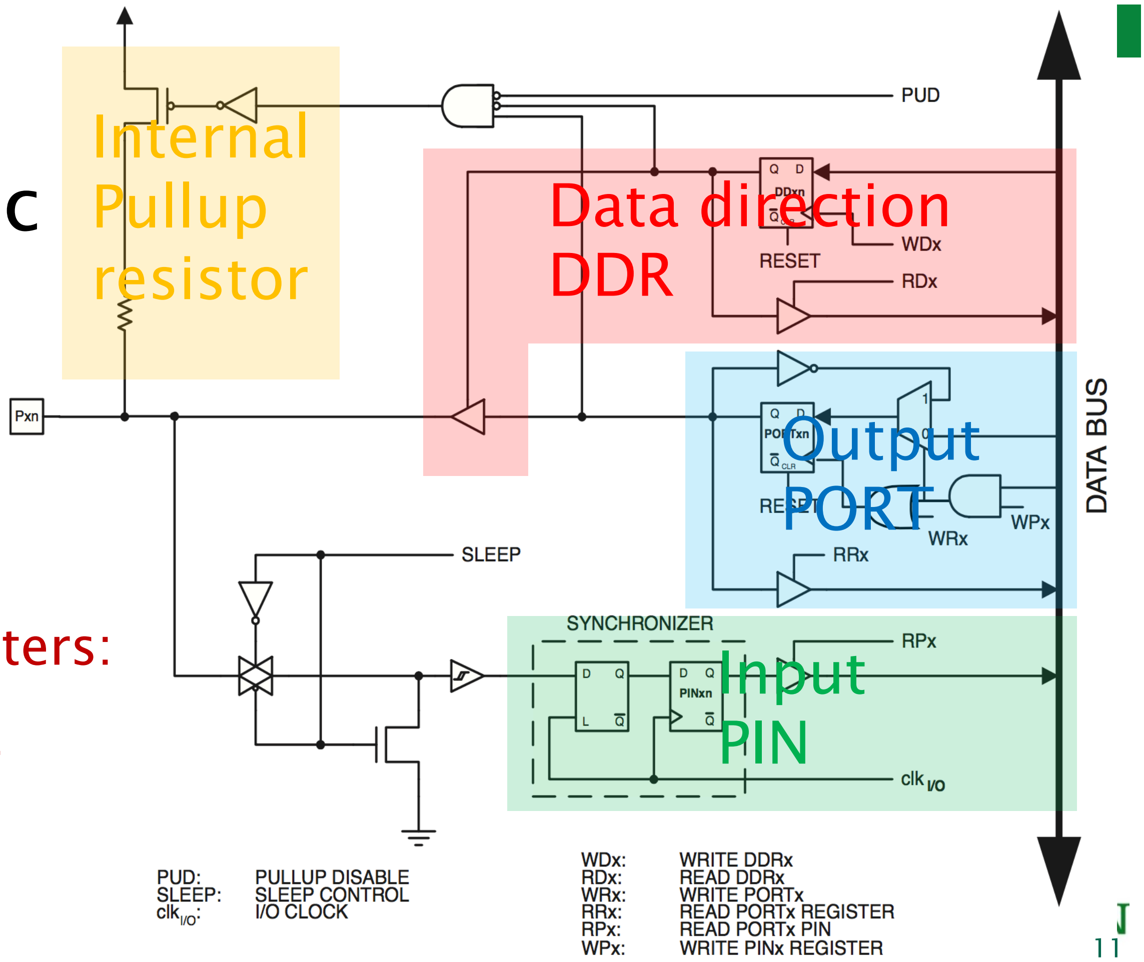
Port Logic



PUD: PULLUP DISABLE
SLEEP: SLEEP CONTROL
clk$_{I/O}$: I/O CLOCK

WDx: WRITE DDRx
RDx: READ DDRx
WRx: WRITE PORTx
RRx: READ PORTx REGISTER
RPx: READ PORTx PIN
WPx: WRITE PINx REGISTER

Port Logic



Internal Pullup resistor

Data direction DDR

Output PORT

Input PIN

Registers:
DDR
PORT
PIN

Content

- Difference arduino code and atmel Code
- What is a port
- **I/O: Direct port manipulation**

Direct port manipulation

What is needed?

- Input, output or tristate?
- Write a value out
- Read a value in

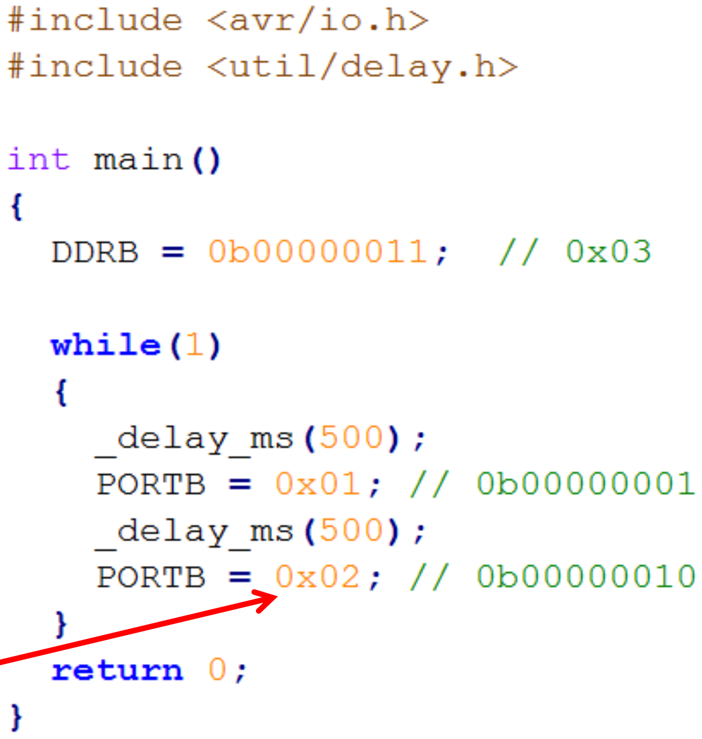
Registers

- DDRx
- PORTx
- PINx

All this information is stored in registers (= RAM memory) and accessible through programming

Direct Port manipulation

- Data Direction (DDRA, DDRB, etc)
- Writing (PORTA, PORTB, etc)
- Reading (PINA, PINB, etc)



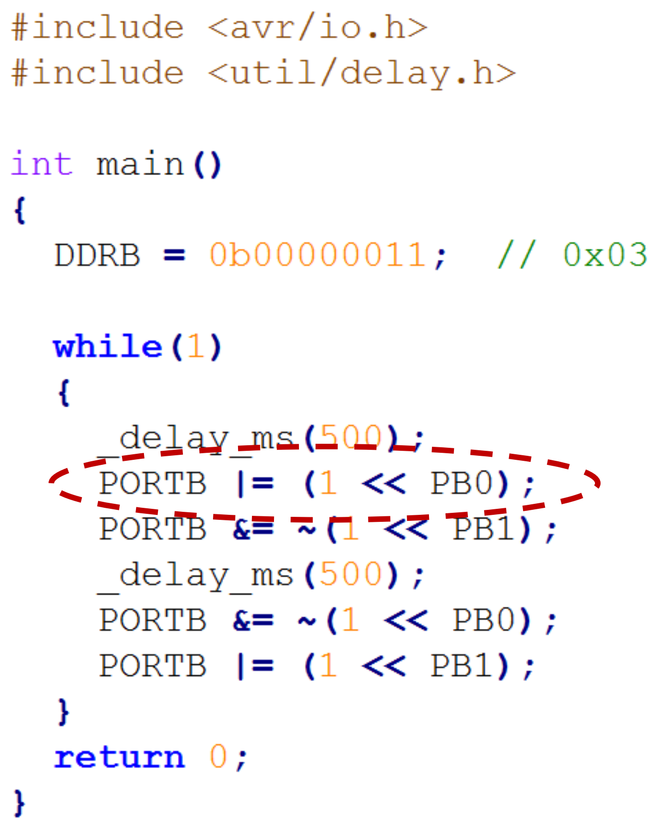
```
#include <avr/io.h>
#include <util/delay.h>

int main()
{
  DDRB = 0b00000011;   // 0x03

  while(1)
  {
    _delay_ms(500);
    PORTB = 0x01; // 0b00000001
    _delay_ms(500);
    PORTB = 0x02; // 0b00000010
  }
  return 0;
}
```

What is disadvantage?

Direct Port manipulation

- Data Direction
  (DDRA, DDRB, etc)
- Writing
  (PORTA, PORTB, etc)
- Reading
  (PINA, PINB, etc)

Bit on: logical or
Bit off: logical and



```
#include <avr/io.h>
#include <util/delay.h>

int main()
{
  DDRB = 0b00000011;  // 0x03

  while(1)
  {
    _delay_ms(500);
    PORTB |= (1 << PB0);
    PORTB &= ~(1 << PB1);
    _delay_ms(500);
    PORTB &= ~(1 << PB0);
    PORTB |= (1 << PB1);
  }
  return 0;
}
```