Mikrocontroller

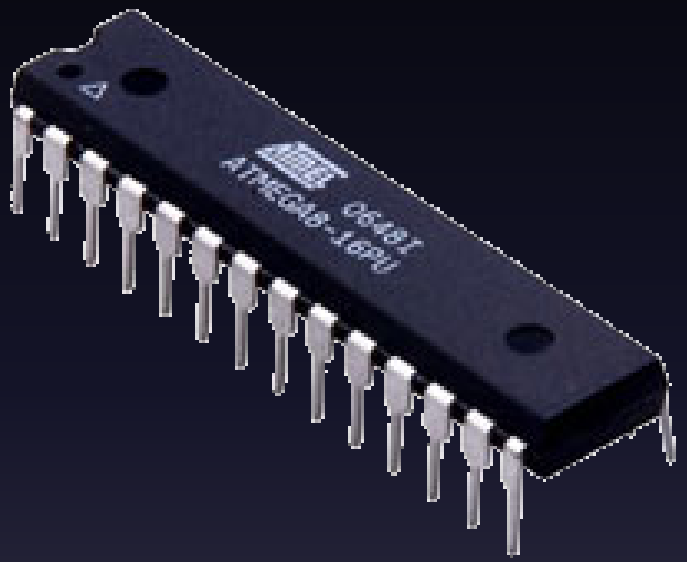
AVR-uC programmieren mit C

Thaddäus Krönert

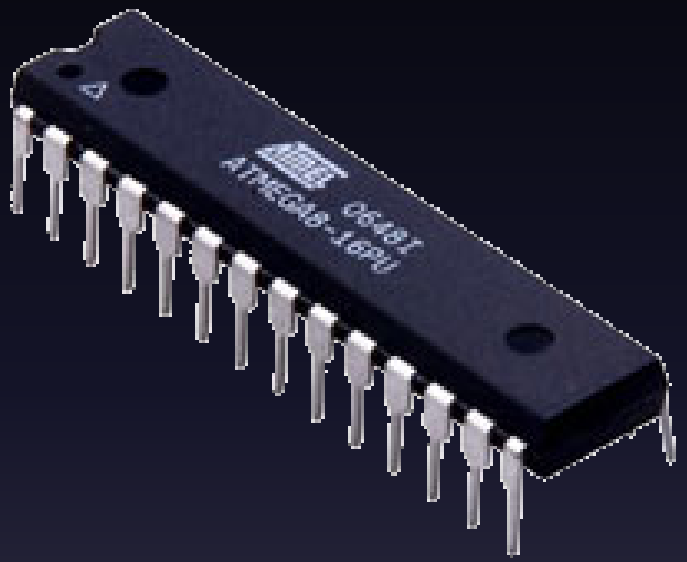
PROJEKT LABOR

Der Mikrocontroller

Der Mikrocontroller

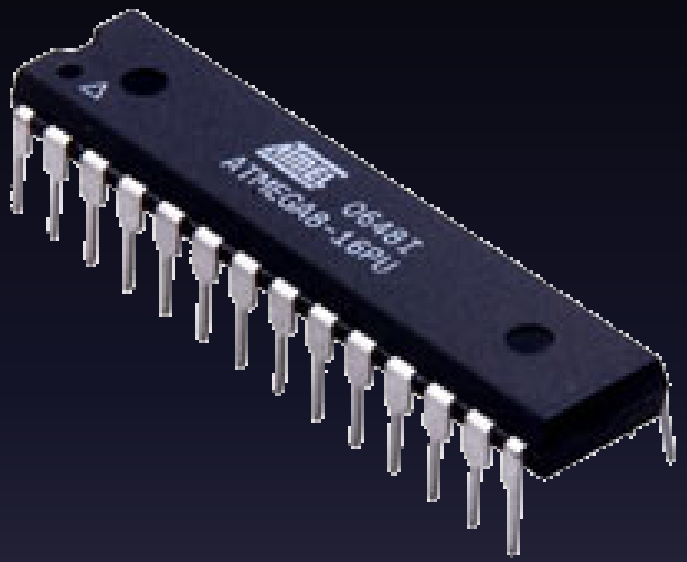


Der Mikrocontroller



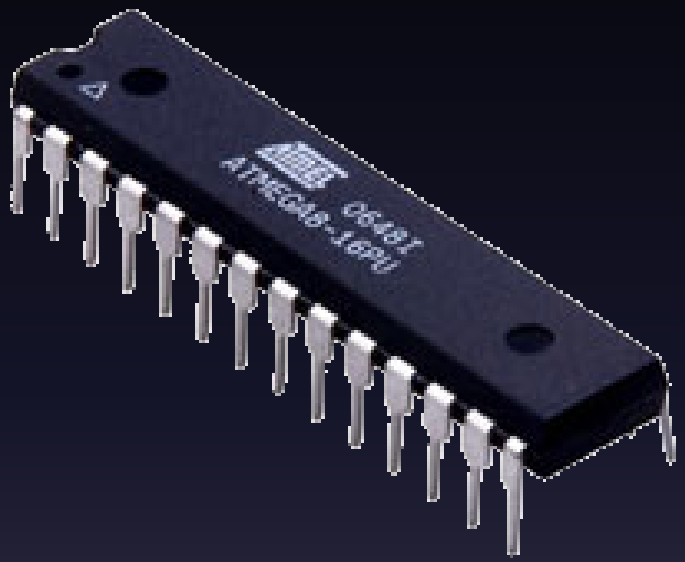
- Bauformkompatible zu Logic ICs

Der Mikrocontroller



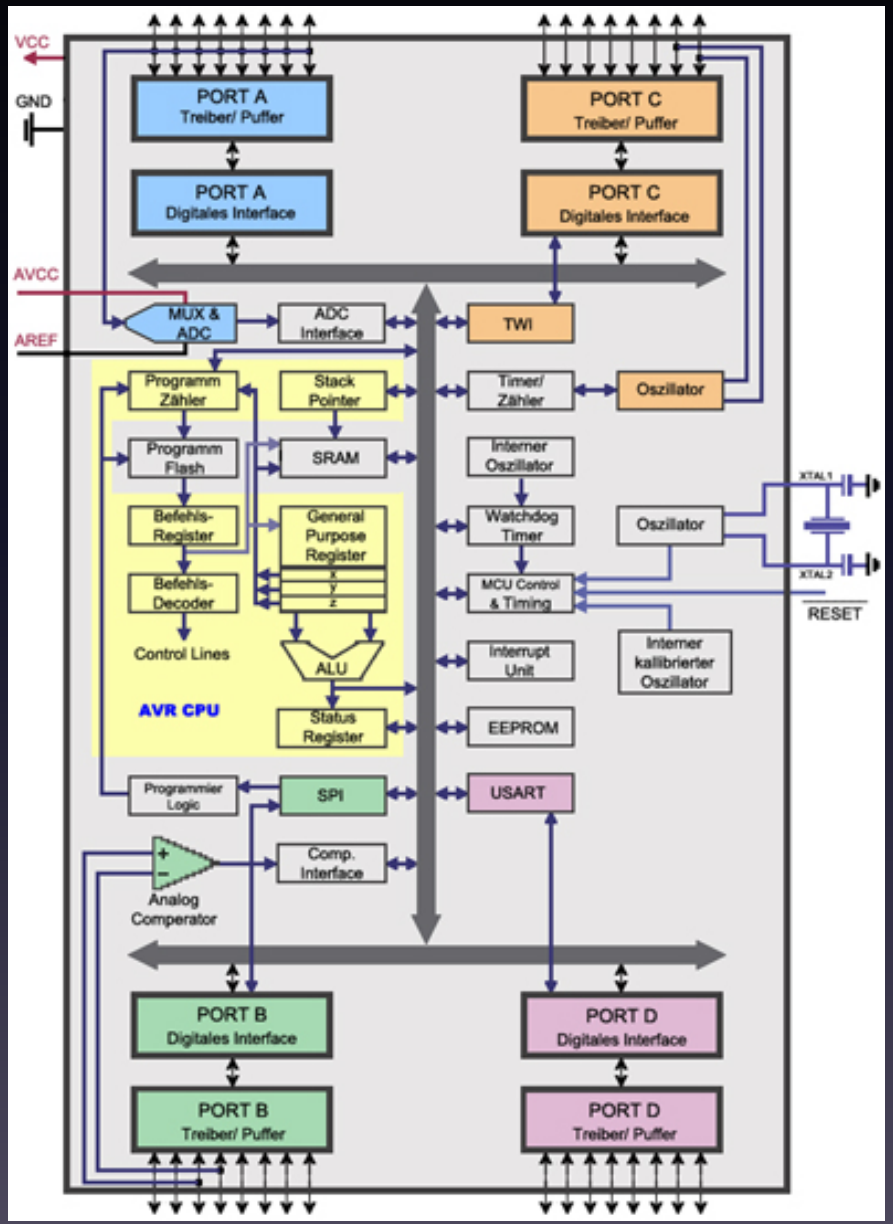
- Bauformkompatible zu Logic ICs
- (meist) auch selber Logic Pegel (5V)

Der Mikrocontroller



- Bauformkompatible zu Logic ICs
- (meist) auch selber Logic Pegel (5V)
- Eigenständiger mini-PC ;)

Im Mikrocontroller



Kenndaten eines Atmel AVR Atmega

- 8-bit RISC

Kenndaten eines Atmel AVR Atmega

- 8-bit RISC
- 32 Register

Kenndaten eines Atmel AVR Atmega

- 8-bit RISC
- 32 Register
- Speicher
  - 1-256 kB Flash-ROM

Kenndaten eines Atmel AVR Atmega

- 8-bit RISC
- 32 Register
- Speicher
  - 1-256 kB Flash-ROM
  - 0.5-16 kB RAM

Kenndaten eines Atmel AVR Atmega

- 8-bit RISC
- 32 Register
- Speicher
  - 1-256 kB Flash-ROM
  - 0.5-16 kB RAM
- 2 -20MHz

Kenndaten eines Atmel AVR Atmega

- 8-bit RISC
- 32 Register
- Speicher
  - 1-256 kB Flash-ROM
  - 0.5-16 kB RAM
- 2 -20MHz
- 1,8-5,5 V

Kenndaten eines Atmel AVR Atmega

- 8-bit RISC
- 32 Register
- Speicher
  - 1-256 kB Flash-ROM
  - 0.5-16 kB RAM
- 2 -20MHz
- 1,8-5,5 V
- Peripherie: AD-Wandler, Timer, PWM, SPI, UART
- 8 - 256 Pins

Kenndaten eines Atmel AVR Atmega

- 8-bit RISC
- 32 Register
- Speicher
  - 1-256 kB Flash-ROM
  - 0.5-16 kB RAM
- 2 -20MHz
- 1,8-5,5 V
- Peripherie: AD-Wandler, Timer, PWM, SPI, UART
- 8 - 256 Pins
- JTAG, debugWire

Warum AVR?

1. Konstenlose Programmierumgebung
   - AVR-Studio (von Atmel)
   - avr-gcc
   - BASCOM (BASIC)

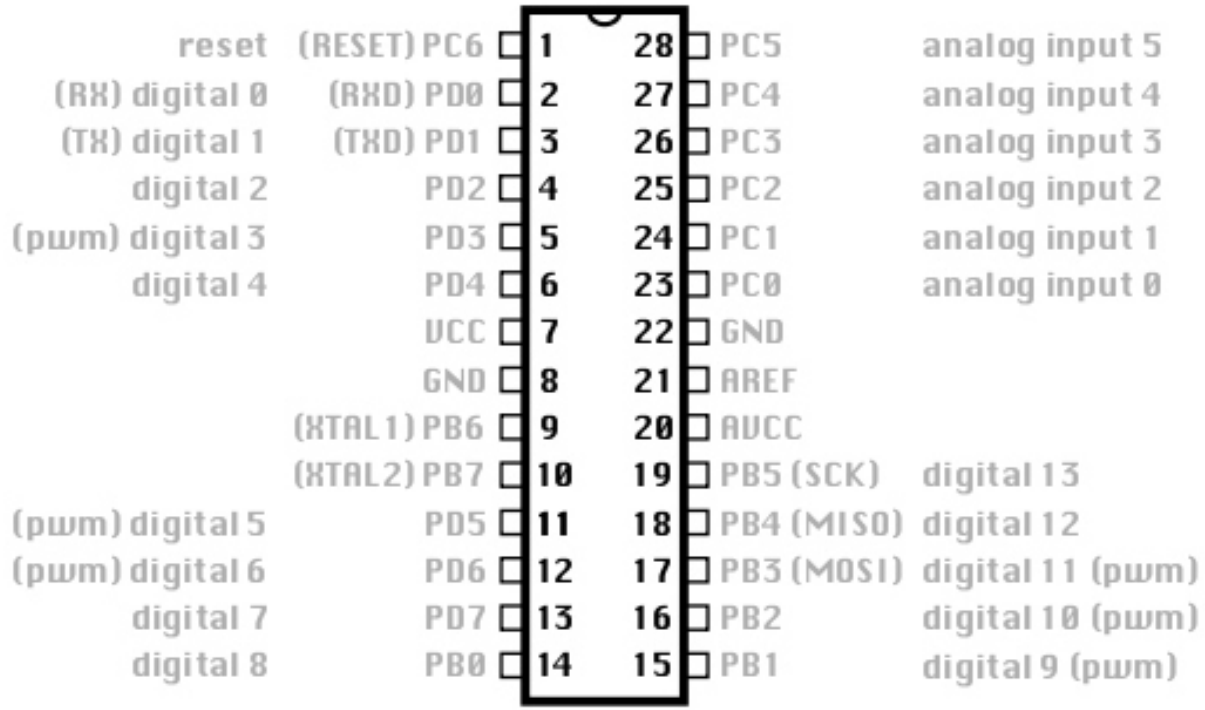
Warum AVR?

1. Konstenlose Programmierumgebung
   - AVR-Studio (von Atmel)
   - avr-gcc
   - BASCOM (BASIC)
2. Günstig
   - Programmer: 5 Euro (zum selberloeten)
   - Mikrocontroller: ab 2.50 Euro

Warum AVR?

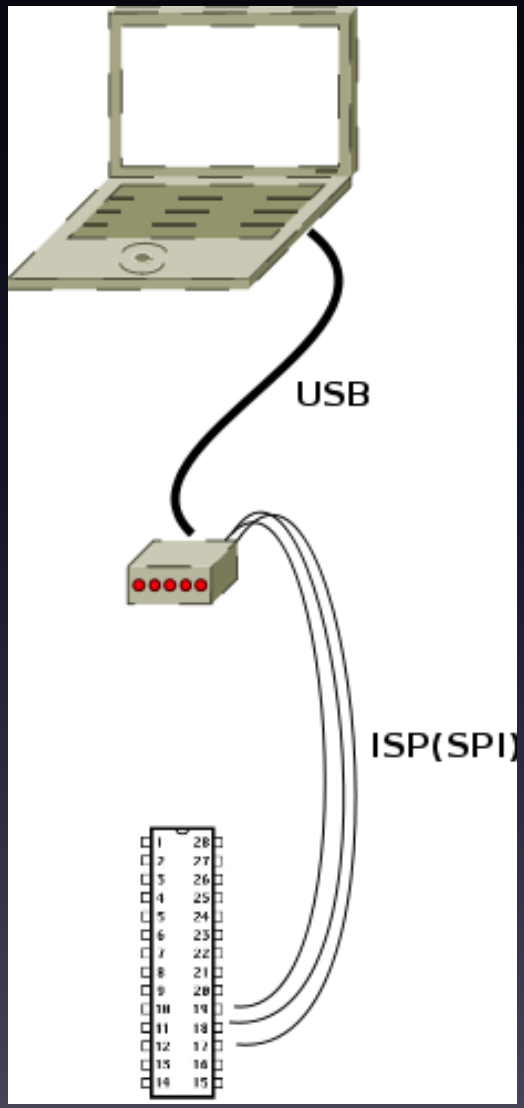
1. Konstenlose Programmierumgebung
   - AVR-Studio (von Atmel)
   - avr-gcc
   - BASCOM (BASIC)
2. Günstig
   - Programmer: 5 Euro (zum selberloeten)
   - Mikrocontroller: ab 2.50 Euro
3. Gut Dokumentiert
   - sehr gute Datenblätter mit Beispielcode
   - sehr grosse Community

Natürlicher Lebensraum eines uC

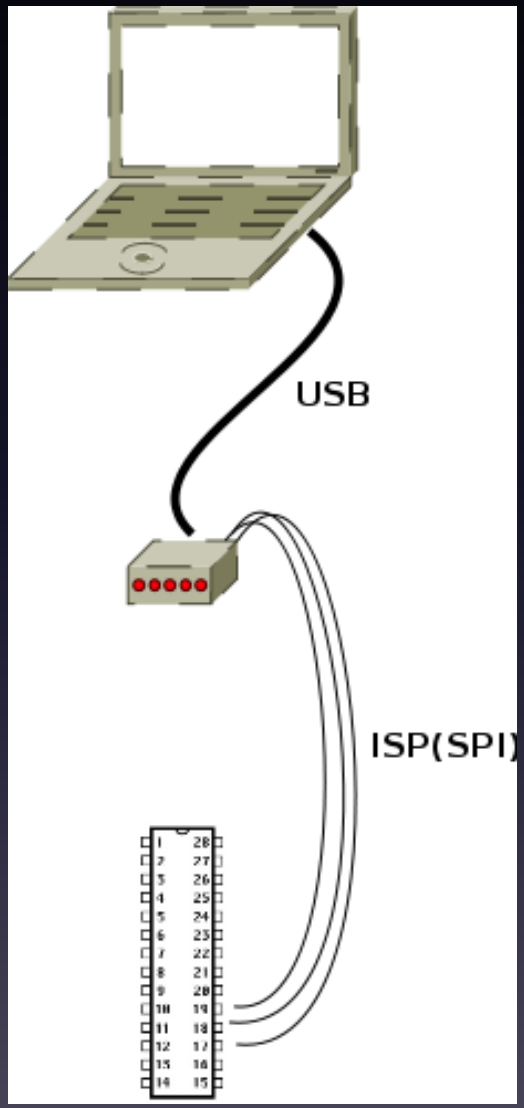


Erste Schritte

Zutaten:

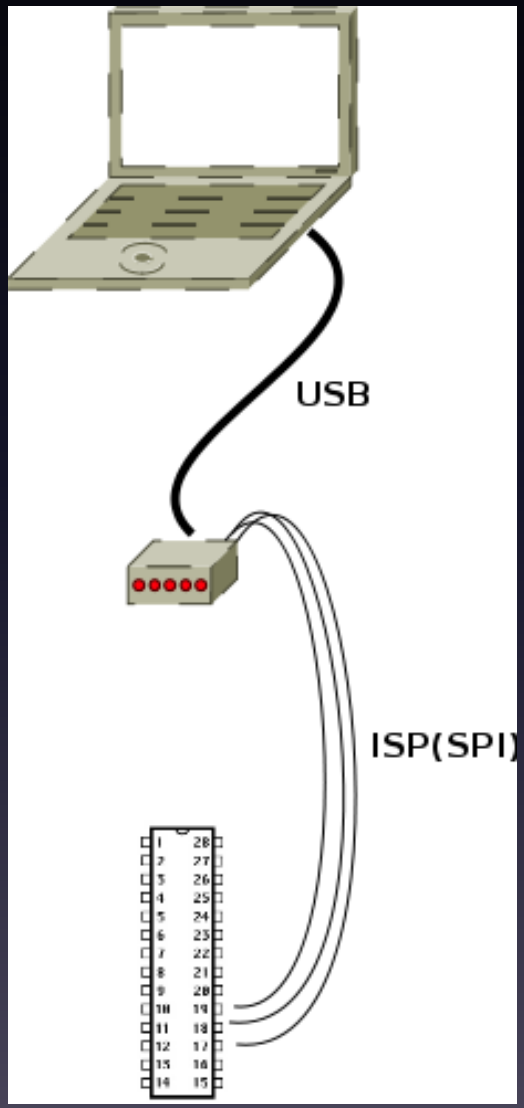


Zutaten:



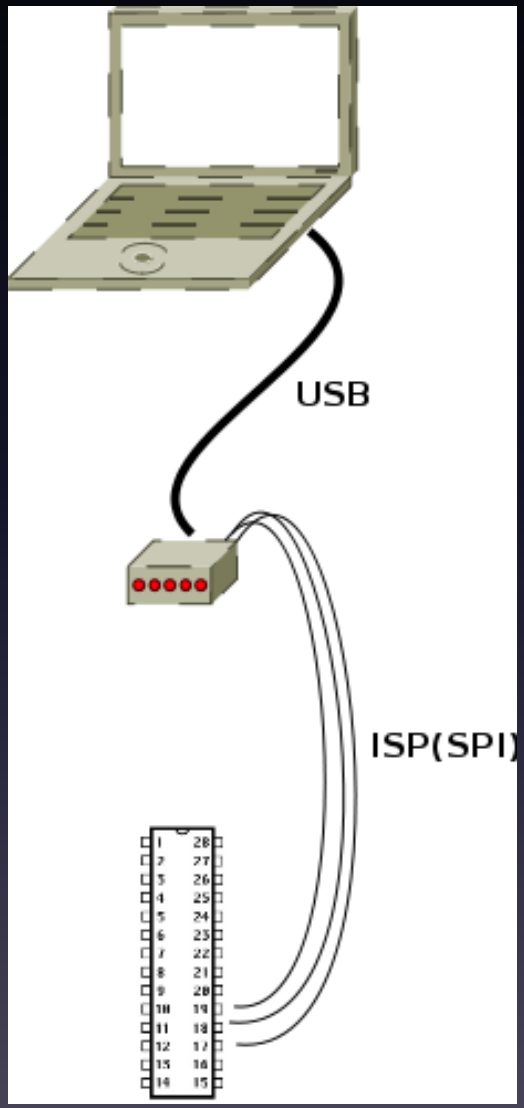
- PC
  - AVR-Studio
  - WinAVR
  - KontrollerLAB
  - avr-gcc + avrdude
  - Arduino (fuer Arduinos)

Zutaten:



- PC
  - AVR-Studio
  - WinAVR
  - KontrollerLAB
  - avr-gcc + avrdude
  - Arduino (fuer Arduinos)
- Programmer
  - USBasp, Ullis: usbasp-smd
  - Dragon (60Euro)

Zutaten:



- PC
  - AVR-Studio
  - WinAVR
  - KontrollerLAB
  - avr-gcc + avrdude
  - Arduino (fuer Arduinos)
- Programmer
  - USBasp, Ullis: usbasp-smd
  - Dragon (60Euro)
- AVR-Startsets
  - Steckbrett (breadini)
  - STK 500
  - noname-Board
  - Arduino (ohne Programmer, dank usb-bootloader)

Hello World!

OUTPUT

```
1  #include <avr/io.h>
2  void main (void) {
3
4
5
6
7    while(1); //endlosschleife
8  }
```

OUTPUT

```
1  #include <avr/io.h>
2  void main (void) {
3    DDRB = 0b00000001; //1==Ausgang
4
5
6
7    while(1); //endlosschleife
8  }
```

OUTPUT

```
1  #include <avr/io.h>
2  void main (void) {
3    DDRB = 0b00000001; //1==Ausgang
4
5    PORTB= 0b00000000; //1== HIGH, 0 == LOW
6
7    while(1); //endlosschleife
8  }
```

INPUT

```
1  #include <avr/io.h>
2  void main (void) {
3
4
5
6    int   i; // int ist 8-bit gross auf einem Atmega
7
8
9
10
11
12   while(1); //endlosschleife
13 }
```

INPUT

```
1  #include <avr/io.h>
2  void main (void) {
3    DDRB = 0b00000000; //als Eingang ; 1==Ausgang
4    PORTB= 0b00000010; //Pullup aktiv; 0==ohne Pullup
5
6    int  i; // int ist 8-bit gross auf einem Atmega
7
8
9
10
11
12   while(1); //endlosschleife
13 }
```

INPUT

```
1  #include <avr/io.h>
2  void main (void) {
3    DDRB = 0b00000000; //als Eingang ; 1==Ausgang
4    PORTB= 0b00000010; //Pullup aktiv; 0==ohne Pullup
5
6    int  i; // int ist 8-bit gross auf einem Atmega
7
8
9      i = PINB; //Lesen des gesammten PORTs B
10
11
12   while(1); //endlosschleife
13 }
```

INPUT

```
1  #include <avr/io.h>
2  void main (void) {
3    DDRB = 0b00000000; //als Eingang ; 1==Ausgang
4    PORTB= 0b00000010; //Pullup aktiv; 0==ohne Pullup
5
6    int  i; // int ist 8-bit gross auf einem Atmega
7
8    do {
9      i = PINB; //Lesen des gesammten PORTs B
10   while (!(i & (0b00000010)));
11
12   while(1); //endlosschleife
13 }
```

INPUT

```
1  #include <avr/io.h>
2  void main (void) {
3    DDRB = 0b00000000; //als Eingang ; 1==Ausgang
4    PORTB= 0b00000010; //Pullup aktiv; 0==ohne Pullup
5
6    int  i; // int ist 8-bit gross auf einem Atmega
7
8    do {
9      i = PINB; //Lesen des gesammten PORTs B
10   //while (!(i & (0b00000010)));
11   while(!(i & (_BV(PB1)))); //<- Alternative!
12   while(1); //endlosschleife
13 }
```

INPUT

```
1  #include <avr/io.h>
2  void main (void) {
3    DDRB = 0b00000000; //als Eingang ; 1==Ausgang
4    PORTB= _BV(PB1); //Pullup aktiv; 0==ohne Pullup
5
6    int  i; // int ist 8-bit gross auf einem Atmega
7
8    do {
9      i = PINB; //Lesen des gesammten PORTs B
10   //while (!(i & (0b00000010)));
11   while(!(i & (_BV(PB1)))); //<- Alternative!
12   while(1); //endlosschleife
13 }
```

Beispielvorgehen: Timer

Anforderung

- Timer0 (8-bit Timer)
- Ausführen von Code bei überlauf
- Prescaler: IO-CLK / 1024

Aus dem Datenblatt:

Timer/Counter Control Register – TCCR0

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	CS02	CS01	CS00	TCCR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 2:0 – CS02:0: Clock Select**

The three clock select bits select the clock source to be used by the Timer/Counter.

Table 34. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Timer/Counter Interrupt Mask Register – TIMSK

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	TIMSK
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	

- **Bit 0 – TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is written to one, and the I-bit in the Status Register is set (one), the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

Timer

```
1  #include <avr/interrupt.h>
2  ISR () //Timer0 overflow vector
3  {
4    // insert interrupt-handler here
5  }
6  void main (void)
7  {
8
9
10     sei();
11     while (1);
12 }
```

Timer

```
1  #include <avr/interrupt.h>
2  ISR () //Timer0 overflow vector
3  {
4    // insert interrupt-handler here
5  }
6  void main (void)
7  {
8    TCCR0 |= ( _BV(CS00) | _BV(CS02) ); //clk/1024
9
10    sei();
11    while (1);
12 }
```

Timer

```
1  #include <avr/interrupt.h>
2  ISR () //Timer0 overflow vector
3  {
4    // insert interrupt-handler here
5  }
6  void main (void)
7  {
8    TCCR0 |= ( _BV(CS00) | _BV(CS02) ); //clk/1024
9    TIMSK |= _BV(TOIE0);
10    sei();
11    while (1);
12 }
```

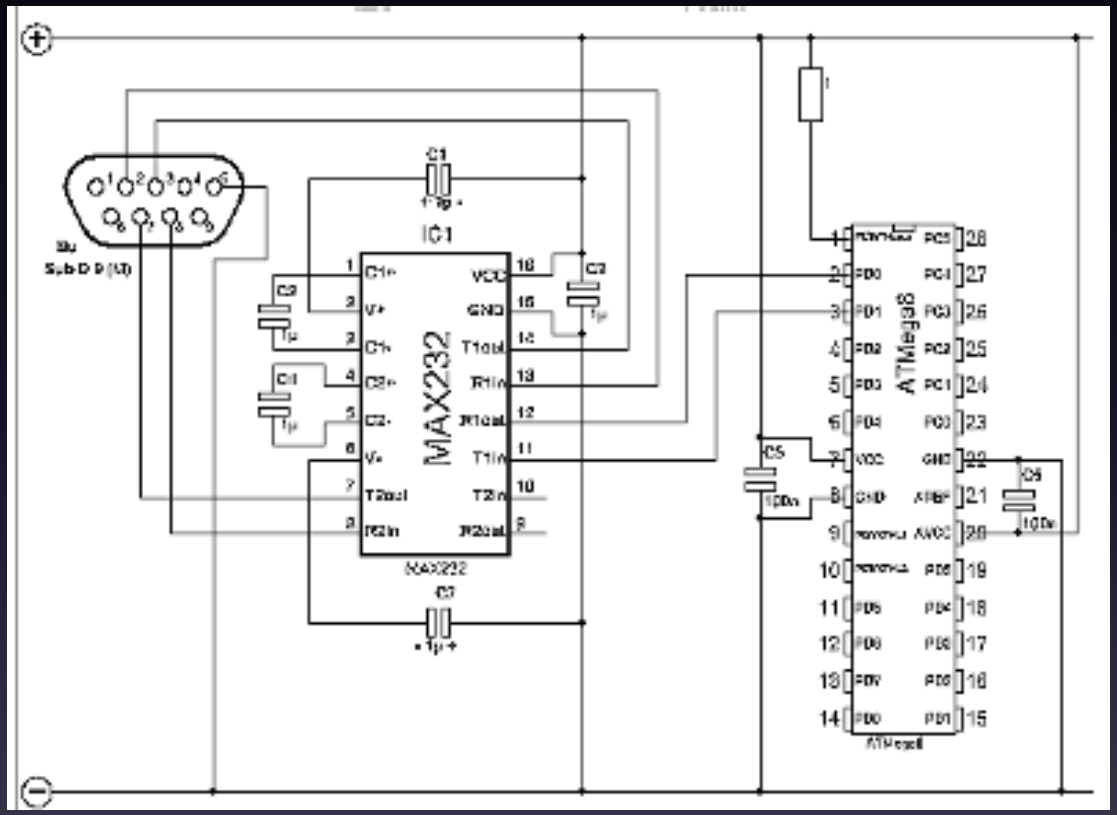
Timer

```
1  #include <avr/interrupt.h>
2  ISR (TIMER0_OVF_vect) //Timer0 overflow vector
3  {
4    // insert interrupt-handler here
5  }
6  void main (void)
7  {
8    TCCR0 |= ( _BV(CS00) | _BV(CS02) ); //clk/1024
9    TIMSK |= _BV(TOIE0);
10    sei();
11    while (1);
12 }
```

Ausblick

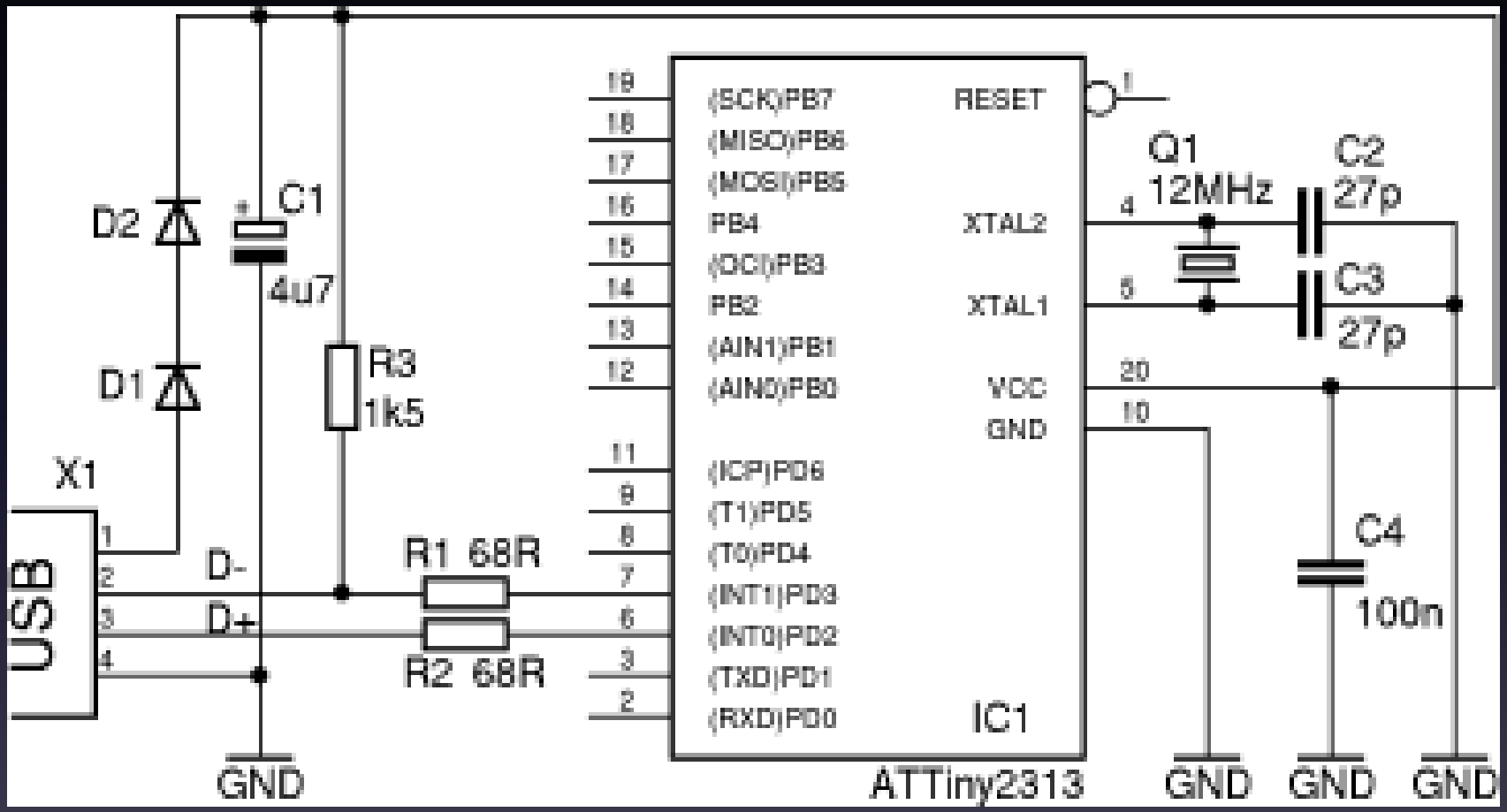
Seriell

- RS232



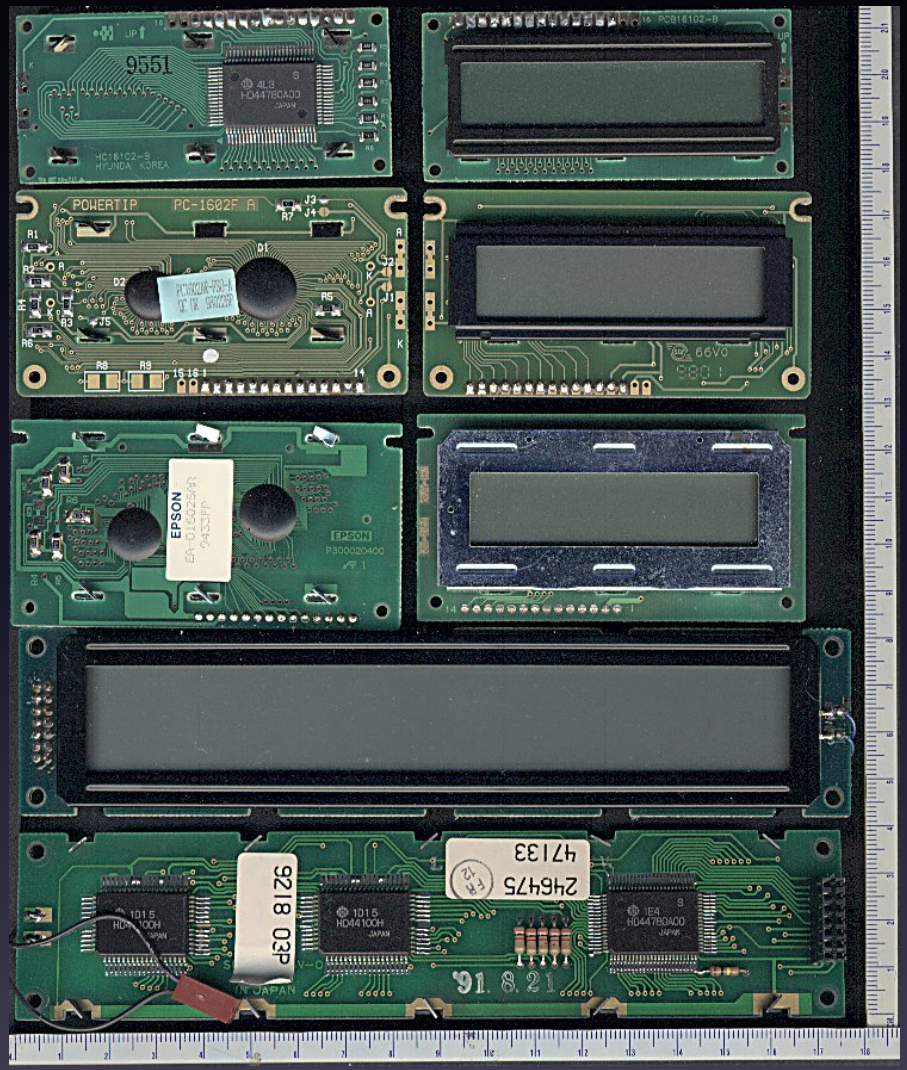
- MAX232 um den Pegel anzupassen
- UART-Pins TX/RX direkt am Atmega

USB



- www.obdev.at/vusb

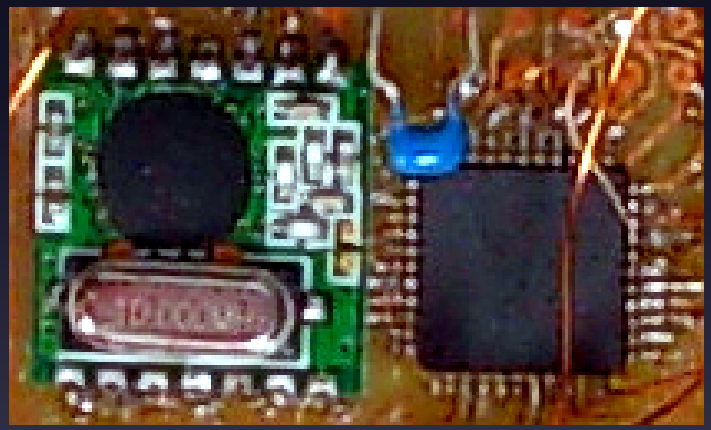
Display



- HD44780 - Displays
- Verbindung ueber 7 Pins

Funk

- RFM12 - Modul
  - 433Mhz
  - 3Euro
  - SPI in/out



- www.das-labor.orgwikiDatenfunk_mit_dem_AVR

Danke!

Fragen?

Quellen & Links

- www.mikrocontroller.net
- tinkerlog.com
- wiki.ullihome.de
- elk.informatik.fh-augsburg.de/da/da-21/Tutorial/intro/intro.html
- Logos:
  - projektlabor.ee.tu-berlin.de
  - www.tu-berlin.de - Pressestelle
- Erstellt mit wiki2beamer ;)
- Style+Makefile: Sebastian Pipping <sebastian@pipping.org>