Seite 75 bis 93 befasst sich mit dem Kapitel 13, welches zu lesen ist

Aufgabe 5.1

* Erklären Sie, wie in den Funktionen pinMode, digitalWrite und delay auf die Hardware zugegriffen wird.

PinMode( ):

**// definiert einen digitalen Pin entweder als Eingang oder Ausgang**

void pinMode(uint8\_t pin, uint8\_t mode)

{

// pin ist der Pin, welcher als Ein- oder Ausgang festgelegt wird

// mode ist der Modus der pin festlegt

uint8\_t bit = digitalPinToBitMask(pin);

uint8\_t port = digitalPinToPort(pin);

volatile uint8\_t \*reg, \*out;

// Ist der übergebene pin ein gültiger Pin? Nein:Ende! Ja: Weiter gehts

if (port == NOT\_A\_PIN) return;

// JWS: can I let the optimizer do this?

// Registrieren des Ports auf die Pointer reg&out

// ***portModeRegister*** liefert ein mode Register des übergebenen Ports zurück, das den Modus des Ports kontrolliert

// ***portOutputRegister*** liefert einen outputPort des angegeben Ports

reg = portModeRegister(port);

out = portOutputRegister(port);

// Ab hier ist die Modusbehandlung

if (mode == INPUT) {

uint8\_t oldSREG = SREG;

cli(); // cli(); hält alle Interrups an! (auch delay()!)

\*reg &= ~bit; // **~** -Zeichen Bitwise not Operator (1->0 & 0->1)

\*out &= ~bit;

SREG = oldSREG; // Setzt das SREG wieder zurück(um da weiterzumachen, wo er aufgehört hat)

} else if (mode == INPUT\_PULLUP) {

uint8\_t oldSREG = SREG;

cli();

\*reg &= ~bit;

\*out |= bit; // **|** -Zeichen Bitwise OR Operator

SREG = oldSREG;

} else {

uint8\_t oldSREG = SREG;

cli();

\*reg |= bit;

SREG = oldSREG;

}

}

digitalWrite(Pin,Wert):

**// schaltet einen, zuvor mit pinMode() als OUTPUT deklarierten, digitalen Kanal auf HIGH (5V+) oder LOW (GND).**

void digitalWrite(uint8\_t pin, uint8\_t val)

{

uint8\_t timer = digitalPinToTimer(pin);

uint8\_t bit = digitalPinToBitMask(pin);

uint8\_t port = digitalPinToPort(pin);

volatile uint8\_t \*out;

if (port == NOT\_A\_PIN) return;

// If the pin that support PWM output, we need to turn it off

// before doing a digital write.

if (timer != NOT\_ON\_TIMER) turnOffPWM(timer);

out = portOutputRegister(port);

uint8\_t oldSREG = SREG;

cli();

if (val == LOW) {

\*out &= ~bit;

} else {

\*out |= bit;

}

SREG = oldSREG;

}

delay(millis)

void delay(unsigned long ms)

{

uint16\_t start = (uint16\_t)micros();

while (ms > 0) {

yield(); //Pennt exakt eine Microsekunde lang

if (((uint16\_t)micros() - start) >= 1000) {

ms--;

start += 1000;

}

}

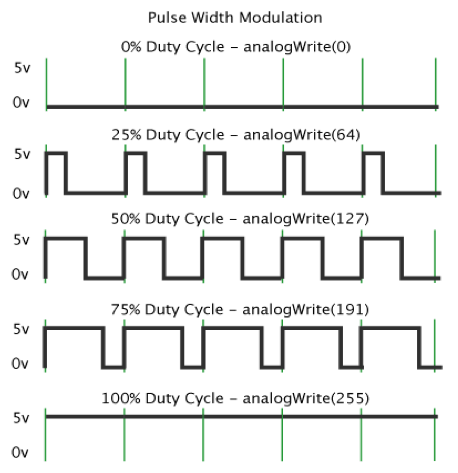
}

// YIELD: Passes control to other tasks when called.

Zusätzliches:

Was ist PWM(Puls Width Modulation) -> Ist ein Verfahren um Spannungen regulieren:

Auf dem Arduino-Board gibt es sechs Kanäle, die PWM-Signale erzeugen können – Siehe Analog Out



PinMode:

* http://garretlab.web.fc2.com/en/arduino/inside/arduino/wiring\_digital.c/pinMode.html
* Tieferes verständnis von digitalPinToBitMask(pin):

http://urbanhonking.com/ideasfordozens/2009/05/18/an\_tour\_of\_the\_arduino\_interna

So ist das Grundgerüst um einen ISR anzuhalten

Interessantes zu Interrups und cli():

void ISR\_anhaltenn () {

uint8\_t sreg = SREG;

**cli();**

//irgendwas machen

SREG = sreg;

}

Gutes Deutsch Tutorial Arduino:

http://www.arduino-tutorial.de/2010/06/pwm/