Vorläufige Arbeitskopie!

Übersetzen von Schrittmotorprotokollen

Entwurf eines Hardwareübersetzers

Praxisbericht

im Fachgebiet Mess- und Sensortechnik

vorgelegt von:	Johannes Dielmann
Studienbereich:	Technik
Matrikelnummer:	515956
Erstgutachter:	Prof. Dr. Carstens-Behrens

Inhaltsverzeichnis

Abkürzungsverzeichnis

ADC	analog digital convertor
ASCII	American Standard Code for Information Interchange
AVRISP	AVR in system programmer
CAD	(engl. computer-aided design) Computer gestützter Entwurf
CF	Compact Flash
CPU	central processing unit
DAC	digital analog convertor
DIL	dual in line package
IC	integrated cuircuit
LCD	(engl. liquid crystal display) Flüssigkristall Display
MHz	megaherz
RS	Radio Sector
SCSI	Small Computer System Interface
USB	universal serial bus
V	Volt

Abbildungsverzeichnis

Tabellenverzeichnis

Codeverzeichnis

1. Einleitung

(TODO: Die Einleitung ist sehr WICHTIG!!!) (TODO: Ausbauen? Neu schreiben!) Die 3D-Lasererfassung bietet zahlreiche Anwendungsgebiete. Von der Erfassung kleiner Objekte über die Erkennung von

(TODO: Klarstellen der Begrifflichkeiten) In der CAD-Entwicklung kommt es vor das für ein real existierendes Objekt eine Erweiterung konstruiert werden muss. Um die Erweiterung sinnvoll konstruieren zu können müssen dazu die Abmessungen des Objektes möglichst genau bekannt sein. Das übertragen der Abmessungen, kann insbesondere für komplexe Objekte, sehr aufwendig sein. Abhilfe soll ein Laserscanner schaffen der das Objekt aus mehreren Richtungen vermisst und aus diesen Informationen ein genaues 3D-Modell davon generiert.

1.1. Motivation

Im Projekt soll nun mit einer Kombination aus einem Lasererfassungssystem, einem Drehtisch und der dazugehörigen Software auf einfachem Wege ein 3D-Modell erfasst werden. Dieses soll dann in einer CAD-Software wie *Solidworks* nutzbar sein.

1.2. Ziel der Arbeit

Die Kommunikation zwischen Software und Drehtisch soll ermöglicht werden. Dazu werden die Befehle der Software mit einem Mikrocontroller ausgewertet und in für den Drehtisch verständliche Befehle übersetzt.
Um den Drehtisch manuell bedienen zu können und den aktuellen Status des Drehtisch anzuzeigen sind noch ein LC-Display und mehrere Taster vorgesehen.
Die Ansteuerung des Drehtisches ist als Einschub für ein 19Rack realisiert. Daher wird die Platine für den Mikrocontroller auch als 19Einschub realisiert.

1.3. Aufbau der Arbeit

Der Aufbau der Arbeit gliedert sich im Wesentlichen in die Entwicklung neuer und die Nutzung vorhandener Hardware, sowie in die Entwicklung der Software für den Mikrocontroller.
Zur Hardware gehören die Auswahl des Mikrocontroller, die Endschalter, die Schrittmotorkarten, die Schrittmotoren, sowie die verwendeten PCs.
Zur Software gehören die Entwicklungsumgebungen und die 3D-Erfassungssoftware.

(TODO: Überblick verschaffen. Kommunikation mit Schrittmotor-Karte von PC aus. Aufbauen des STK500. Einarbeiten in uC Entwicklung. Steuern der Schrittmotor Karte vom uC aus. Erarbeiten der Protokolle (Reverse Engineering). C-Programm zum verstehen eingehender Befehle. Übersetzen der Befehle. Neuer Mikrocontroller. Umgebungswechsel. LC-Display. Endschalter. Max232. Platinenlayout.)

2. Hardware

(TODO: Besseres Bild! Schema? Bessere Beschriftung!)

Abbildung 2.1.: Überblick des Arbeitsplatz

(TODO: Fussnoten für Komponenten, Herst. u. Websites)
(TODO: Komponenten auf Abbildung erwähnen!)

2.1. Lasererfassungssystem VI-900

(TODO: Bild) Das Lasererfassungssystem *VI-900* der Firma *Minolta* besteht aus einem *Lasertriangulator* und einer Kamera. Das System lässt sich über eine *SCSI*-Schnittstelle ansprechen und konfigurieren. Zur mobilen Nutzung kann das Gerät

auch auf der Rückseite bedient werden. Aufgenommene Daten können auf einer *CF-Karte* gespeichert werden. Im Projekt wurde jedoch lediglich die Ansteuerung via SCSI genutzt.

2.2. Ansteuerung für den Drehtisch

2.2.1. Drehtisch

Der Drehtisch ist eine Eigenkonstruktion der Werkstatt des RheinAhrCampus. Er besteht aus einer massiven Edelstahl Arbeitsplatte, welche auf 4 Füßen ruht. Aus dieser ist ein Rechteck mit aufgesetztem Halbkreis ausgeschnitten. In diesem Ausschnitt befindet sich der Drehtisch. Er ist auf einem Schienensystem gelagert. Mit dem Schienensystem lässt der Drehtisch sich in der Vertikalen positionieren. Mit einem Schrittmotor lässt sich der Drehtisch sich zusätzlich in der Höhe verstellen. Die Höhenverstellung wird mit einem Schneckengetriebe realisiert. Ein weiterer Schrittmotor ist für die Drehung des Tisches zuständig. Der Tisch ist über ein *Harmonic-Drive-Getriebe* mit dem Schrittmotor verbunden. Das Übersetzungsverhältnis beträgt 1:50.

2.2.2. Spannungsversorgung

(TODO: Verkabelung Steckbar und universell gemacht) Die Schrittmotorkarten werden von einem PC-Netzteil gespießt. Die Kabel waren direkt an die Verbindungsleisten gelötet. Um den Aufbau modular und erweiterbar zu machen, ersetzte ich die feste Lötverbindung durch eine Standard PC-Netzteil Verbindung. Dadurch kann das Netzteil einfach ausgebaut werden, bzw. das System leicht mit neuen Einschubkarten erweitert werden.

2.2.3. Schrittmotoren

(TODO: Motoren beschreiben! Technische Daten! Schritte, Spannungen. Verdrahtung.)

2.2.4. Schrittmotorkarten

Die Ansteuerung für die Schrittmotoren sind als 19Einschübe realisiert. Für jeden Schrittmotor wird ein Einschub benötigt. Die Einschübe sind Produkte der Firma R+S. Mittels RS-232 Schnittstelle lassen sich die Karten konfigurieren und ansteuern.

Die Konfiguration und Ansteuerung erfolgt über einen vorgegeben *ASCII* Befehlssatz. Der Befehlssatz befindet sich im Kapitel 3.3.8.Es können 2 oder mehr Karten als *Daisy-Chain* [1] in Reihe geschaltet werden.

2.2.5. Motorverkabelung

Die Schrittmotoren benötigen ein mindestens 4-adriges Kabel. Das Kabel für den Schrittmotor der für die Rotation zuständig ist war bereits gefertigt. Das Kabel für den Schrittmotor der für die Höhenverstellung zuständig ist habe ich selbst gefertigt. Hier wurden 3 weitere Adern für die beiden Endschalter benötigt. **(TODO: Schemazeichnung Kabel!)**

2.2.6. Endschalter

Die Schrittmotorkarten unterstützen das Abschalten der Motoren wenn ein sogenannter Endschalter ausgelöst wird. Dies sind im allgemeinen mechanische Schalter die ausgelöst werden wenn der Tisch sich dem Ende des Arbeitsbereiches nähert. Dies verhindert eine Beschädigung des Aufbaus.
Im Aufbau waren bereits induktive Endschalter der Firma Pepperl+Fuchs verbaut. Normalerweise unterstützt die Schrittmotorkarte nur mechanische Endschalter. Durch geschickte Verdrahtung ließen sich die induktiven Endschalter verwenden. Hierzu musste über einen Spannungsteiler die Spannung herabgesetzt werden und konnte somit direkt an den Optokoppler der Schrittmotorkarte angeschlossen werden. **(TODO: Schemazeichnung der Verdrahtung)**
Am Drehtisch war ein Metallstutzen angebracht der den Endschalter auslösen sollte. Dieser war jedoch ungeeignet da er nicht dicht genug an den Induktiven Schalter heran kam, obwohl der Tisch schon in der Endposition war.
Abhilfe schaffte ein längerer Metallstutzen der von der Werkstatt gefertigt wurde.
Wenn der Tisch sich in der Endposition befindet soll dies auch auf dem Mikrocontroller angezeigt werden. Die Signale der Endschalter liegen auf der Rückseite **(TODO: Zeichnung der Anschlüsse referenzieren.)** am Verbindungsstecker an. Es muss also nur eine Brücke zu den entsprechenden Pins des Verbindungsstecker des Mikrocontroller gelötet werden.
Auf der Mikrocontroller Platine sind diese Pins mit 2 Pins des Mikrocontroller verbunden. Die beiden Pins werden im Mikrocontroller als Interrupts definiert. Die

[1] Als Daisy Chain (englisch, wörtlich „Gänseblümchenkette“) bezeichnet man eine Anzahl von Hardware-Komponenten, welche in Serie miteinander verbunden sind (meist in sogenannten Bussystemen in der Automatisierungstechnik).Wikipedia [2012a]

Interrupt-Service-Routine ist im Kapitel (TODO: Software Kapitel referenzieren) beschrieben.

2.3. Mikrocontroller

Ein Mikrocontroller vereint, in einem IC, die wichtigsten Komponenten um komplexe technische Probleme leicht lösen zu können. Dazu gehören z.B. CPU, Flash-Speicher, Arbeitsspeicher, Register, Ports, ADC, DAC und mehr. Einen schematischen Überblick über die Komponenten eines Mikrocontroller bietet das Blockdiagramm in Abbildung 2.2.
In einer Programmierumgebung lässt sich dann für den Mikrocontroller ein Programm schreiben. Diese Programme können Signale an Pins des Mikrocontroller auswerten und Signale über andere Pins ausgeben. Eingehende Signale können binär ausgewertet werden oder mit einem ADC die Spannungshöhe bestimmt werden. Ausgehende Signale können auch binär oder mit einem DAC analog ausgegeben werden. Binäre Signale können zur Steuerung von LEDs oder Peripherie Geräten genutzt werden. Auch LC-Displays und Serielle Schnittstellen können so angesteuert werden. Für unterschiedliche Aufgaben sind verschiedene Mikrocontroller geeignet. Zu Beginn stand ein ATmega 8515Atmel [2012b] im DIL-Gehäuse zur Verfügung. Dieser hatte 8 Kbyte Flash, 3 externe Interrupts, 1 Serielle Schnittstelle und konnte mit bis zu 16 MHz betrieben werden. Dieser war geeignet sich in die Programmierung mit C ein zu finden und eine Serielle Schnittstelle an zu steuern.
Für dieses Projekt sind jedoch 2 externe Schnittstellen nötig. Der ATmega 324A erfüllt diese Voraussetzung. Atmel [2012a] Er ist dem ATmega 8515 recht ähnlich, bietet jedoch die benötigten 2 seriellen Schnittstellen. Des weiteren hat er 32 Kbyte Flash. (TODO: mehr schreiben??)

2.3.1. Entwicklerboard STK500

Um Mikrocontroller zu programmieren und die Programmierung zu überprüfen kann das *Entwicklerboard* STK500 der Firma ATMEL verwendet werden. Das Board enthält mehrere Mikrocontroller Steckplätze, 2 Serielle Schnittstellen, 8 Taster, 8 LEDs, 2 Erweiterungsports, ein *ISP* (TODO: besserer Name!) und mehrere Jumper zum konfigurieren des Boards.
Von den beiden seriellen Schnittstellen kann die eine zur Programmierung des Mikrocontroller verwendet werden. Die andere kann zur Kommunikation mit dem Mikrocontroller genutzt werden.

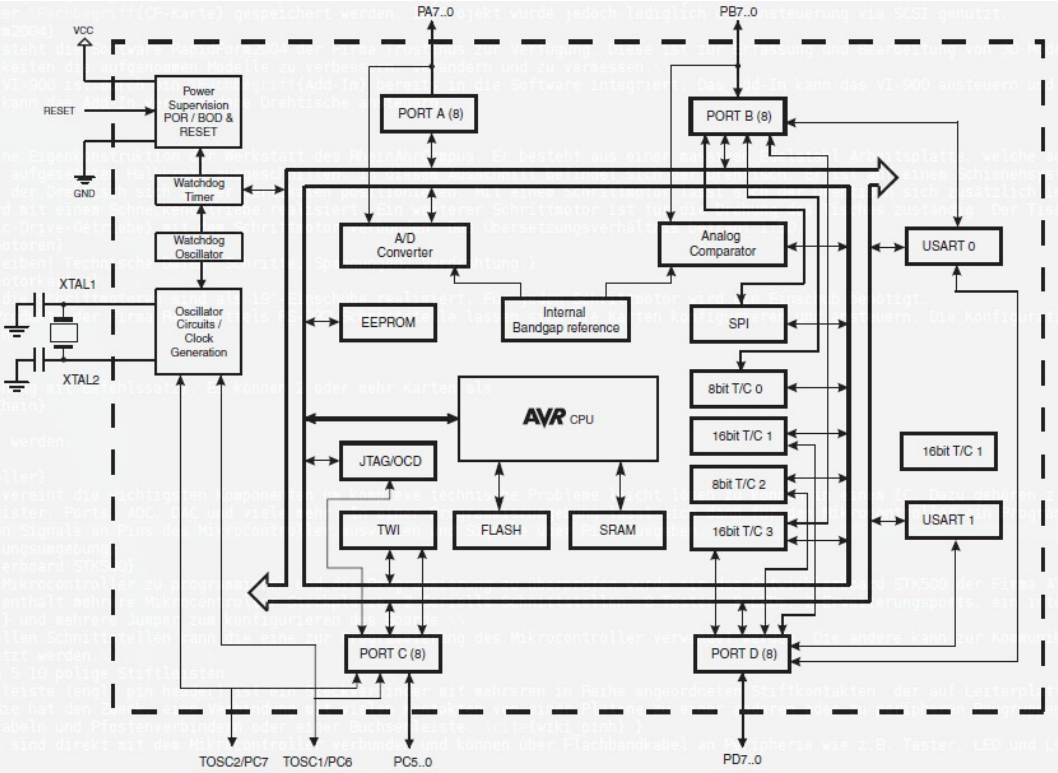

Abbildung 2.2.: Block Diagram eines Mikrocontroller
[Atm 2011]

Auf dem Board stehen 5 10 polige Stiftleisten zur Verfügung. Diese sind direkt mit dem Mikrocontroller verbunden und können über Flachbandkabel an Peripherie wie z.B. Taster, LED und LC-Displays angeschlossen werden.

2.3.2. AVRISP mkII

Das AVRISP mkII ist ein USB-basiertes In-System-Programmiersystem. Dieses kann anstelle des RS-232 basierten Programmiersystem des STK500 verwendet werden.
Die Übertragungsgeschwindigkeit des AVRISP mkII ist wesentlich höher als die über die Serielle Verbindung. Desweiteren wurde der ATmega324A nicht mehr vom STK500 internen ISP unterstützt.
Der AVRISP mkII lässt sich einfach an den Programmierport, eine 6-Polige Stiftleiste, des STK500 anschließen.

2.3.3. MAX232

Die Spannungspegel des Mikrocontroller(typ. 0-5 V) sind nicht kompatibel zu den Spannungspegeln des RS-232 Standards (typ. -12-+12 V). Daher wird der *Pegelumsetzer* MAX232 genutzt. Dieser wandelt mit internen Operationsverstärkern die Spannungspegel auf den richtigen Wert.(TODO: Beschaltung?)

2.4. Platinenlayout

Für den Mikrocontroller und seine Peripherie wurde ein Platinenlayout entwickelt. Dieses wurde in der Opensource Software KiCad entwickelt.
Dazu wurden die Schaltungen wie auf dem STK500 in den Schaltplan übernommen und dort das Layout entwickelt. (TODO: Schaltplan und Layout Bild einbinden.)

3. Software

(TODO: Einführung schreiben) (TODO: Weitere Software in Begriffen erklären. Minolta)

3.1. RapidForm2004

Zur Erfassung am PC steht die Software RapidForm2004 der Firma TrustInus zur Verfügung. Diese ist zur Erfassung und Bearbeitung von 3D-Modellen gedacht. Sie bietet umfangreiche Möglichkeiten die aufgenommen Modelle zu verbessern, verändern und zu vermessen.
Die Ansteuerung des VI-900 ist durch ein *Add-In* bereits in die Software integriert. Das Add-In kann das VI-900 ansteuern und die Aufgenommenen Daten auslesen. Weiterhin kann das Add-In verschiedene Drehtische ansteuern.

3.2. Entwicklungsumgebung

Als Entwicklungsumgebung wird eine Software bezeichnet die es dem Anwender erleichtert Programme für den Mikrocontroller zu schreiben. Im allgemeinen bestehen Entwicklungsumgebungen aus einem Editor, dem Compiler und einer Programmiersoftware. Der Editor bietet dabei meist Komfortfunktionen wie Syntaxhighlighting, Autovervollständigung und Projektmanagement. (TODO: besser schreiben!)

3.2.1. AVR Studio 5

(TODO: AVR Studio Eclipse Bug Defekte Biblio?)

3.2.2. Eclipse

3.3. Mikrocontroller

(TODO: Codebeispiele sind zusammengefasst. Vollständiger Code im Anhang.) (TODO: Backup anlegen und cleanen!)

3.3.1. Fuses

Als Fuses werden Register bezeichnet mit denen sich, auf Hardwareebene, das Verhalten des Mikrocontroller verändern lässt. (TODO: Fuses Tabellen aus Datenblatt!)

CKSEL	
SUT	•
CKDIV8	•
CKOUT	•
CKOPT	•
RSTDISBL	•
SPIEN	•
JTAGEN	•
DWEN	•
OCDEN	•
EESAVE	•
BODEN	•
BODLEVEL	•
WDTON	•
BOOTRST	•
BOOTSZ	•
Compatibility Bits	•
SELFPRGEN	•
HWBEN	•

3.3.2. LEDs

Das Codebeispiel 3.1 zeigt ein einfaches Beispiel mit dem sich die Funktionalität der LEDs leicht überprüfen lässt. Bei jedem Aufruf der Funktion wird der aktuelle Status des LED Port abgefragt und der Hexwert um 1 Bit verschoben. Dadurch wird die daneben liegende LED eingeschaltet und die aktuelle aus geschaltet. Wird ein bestimmter Wert überschritten wird der Port wieder auf den Anfangszustand zurück gesetzt.

Listing 3.1: Funktion - Lauflicht

```
1
2 void led_lauflicht (void) {
3   uint8_t i = LED_PORT;
4   i = (i & 0x07) | ((i << 1) & 0xF0);
5   if (i < 0xF0)
```

```
6       i |= 0x08;
7     LED_PORT = i;
8   }
```

3.3.3. Taster

Listing 3.2: Taster

```
1  #include "Debounce.h"
2
3  void  debounce_init  (void) {
4    KEY_DDR &= ~ALL_KEYS; // configure key port for input
5    KEY_PORT |= ALL_KEYS; // and turn on pull up resistors
6    TCCR0B = (1 << CS02) | (1 << CS00); // divide by 1024
7    TCNT0 = (uint8_t) (int16_t) -(F_CPU / 1024 * 10 * 10e-3 + 0.5); // preload for 10ms
8    TIMSK0 |= 1 << TOIE0; // enable timer interrupt
9    sei();
10 }
11
12 if   (get_key_press(1 << KEY0) || get_key_rpt(1 << KEY0)){
13   lcd_puts("Betrete Menue!\n");
14   menu_enter(&menu_context, &menu_main);
15 }
```

3.3.4. LCD Bibliothek

Die meisten LC-Displays werden auf die selbe Art angesteuert. Hier gibt es fertige Bibliotheken die frei genutzt werden können. Im Projekt wird die von Peter Fleury**?** verwendet.
Dazu müssen die Dateien lcd.c und lcd.h in das Arbeitsverzeichnis kopiert werden und die Bibliothek mit *#include(lcd.h)* eingebunden werden.
Anschließend müssen noch in der lcd.h die Daten des Display eingegeben werden. Danach kann das Display mit den Befehlen aus Zeile 15-24 aus dem Codebeispiel 3.3 angesteuert werden.

Listing 3.3: Definitionen - LCD(Auszug)

```
1  #define LCD_CONTROLLER_KS0073 0 /**< Use 0 for HD44780 controller,
2                                  1 for KS0073 controller */
3
4  #define LCD_LINES         4     /**< number of visible lines of the display */
5  #define LCD_DISP_LENGTH 19 /**< visibles characters per line of the display */
6  #define LCD_LINE_LENGTH 0x40 /**< internal line length of the display  */
```

```
7
8  #define LCD_START_LINE1 0x00 /**< DDRAM address of first char of line 1 */
9  #define LCD_START_LINE2 0x40 /**< DDRAM address of first char of line 2 */
10 #define LCD_START_LINE3 0x14 /**< DDRAM address of first char of line 3 */
11 #define LCD_START_LINE4 0x54 /**< DDRAM address of first char of line 4 */
12
13 #define LCD_WRAP_LINES 1   /**< 0: no wrap, 1: wrap at end of  visibile  line */
14
15 extern void lcd_init(uint8_t dispAttr);
16 extern void lcd_clrscr(void);
17 extern void lcd_home(void);
18 extern void lcd_gotoxy(uint8_t x, uint8_t y);
19 extern void lcd_putc(char c);
20 extern void lcd_puts(const char *s);
21 extern void lcd_puts_p(const char *progmem_s);
22 extern void lcd_command(uint8_t cmd);
23 extern void lcd_data(uint8_t data);
24 #define lcd_puts_P(__s)        lcd_puts_p(PSTR(__s))
```

3.3.5. RS-232

Listing 3.4: Funktionen - RS-232

```
1  #define BAUD 9600
2  #include <util/setbaud.h>
3
4  void  uart_init   () {
5    UBRR0H = UBRRH_VALUE;  // UART 0 - IN (Rapidform Software/Terminal)
6    UBRR0L = UBRRL_VALUE;
7    UCSR0C = (3 << UCSZ00);
8    UCSR0B |= (1 << TXEN0);  //Transmitter Enabled
9    UCSR0B |= (1 << RXEN0);  // UART RX einschalten
10
11   UBRR1H = UBRRH_VALUE;  // UART 1 - OUT (Stepper Karte/Drehtisch)
12   UBRR1L = UBRRL_VALUE;
13   UCSR1C = (3 << UCSZ00);
14   UCSR1B |= (1 << TXEN1);  //Transmitter Enabled
15   UCSR1B |= (1 << RXEN1);  // UART RX einschalten
16 }
17 void  uart_put_charater (unsigned char c, int dir) {
18   if (dir == D_RapidForm) { // To Rapidform
19     while (!(UCSR0A & (1 << UDRE0))) {}//warten bis Senden moeglich
20     UDR0 = c;    // sende Zeichen
21   }
```

```
22    else {          // To Stepper
23      while (!(UCSR1A & (1 << UDRE1))) {}//warten bis Senden moeglich
24      UDR1 = c; // sende Zeichen
25    }
26  }
27  void  uart_put_string (char *s, int dir) {
28    while (*s) // so lange *s != '\0' also ungleich dem "String-Endezeichen(Terminator)" {
29      uart_put_charater(*s, dir);
30      s++;
31    }
32  }
```

3.3.6. Menü Bibliothek

Der Drehtisch kann Manuell über Taster am Einschub bedient werden. Die Menü Bibliothek gestaltet dies einfach und Komfortabel. Mit den Tasten Zurück. Select, Hoch und Runter lässt sich durch die Einzelnen Menü Punkte Navigieren. **(TODO: Menü Baum erstellen!)**

3.3.7. Interrupts

Viele Mikrocontroller bieten die Möglichkeit zeitkritische Subroutinen auszuführen. Wenn einer der Interrupts ausgelöst wird, wird das Hauptprogramm unterbrochen und die Entsprechende Interrupt-Service-Routine ausgeführt. Nach Beendigung der ISR wird das Hauptprogramm an der vorherigen Stelle wieder aufgenommen.
ISR dürfen nur sehr wenige Befehle enthalten und müssen innerhalb weniger Clock-Cicles abgeschlossen sein.
Interrupts können z.B. der Überlauf eines internen Timer sein, oder ein externens Signal an einem Pin.
Im Projekt werden externe Interrupts, Timer-Überlauf Interrupts und der Watchdog Interrupt genutzt.

3.3.7.1. Endschalter

Die Endschalter sind über die Schrittmotorkarten und eine Brücke in der Steuerung mit der Mikrocontroller Platine Verbunden. Dort sind sie an 2 Interrupt Pins angeschlossen. Bei einem Flanken Wechsel an den Pins wird ein Interrupt ausgelöst.
Das Code-Listing 3.5 zeigt die ISR für die Endschalter.

Listing 3.5: ISR - Endschalter

```
PCMSK3 |= ( 1 << PCINT28 );  // Interrupts definierenPD4 als Interrupt zulassen
PCICR  |= ( 1 << PCIE3 );    // Pin Change Interrupt Control Register – PCIE3 setzen fuer
    PCINT30
ISR(PCINT3_vect){          // Endschalter Position  erreicht
  lcd_puts("Positive Enschalter Position  Erreicht!");
  LED_PORT ^= (1 << LED3);
}
ISR(PCINT2_vect){          // Endschalter Position  erreicht
  lcd_puts("Negative Enschalter Position Erreicht!");
  LED_PORT ^= (1 << LED3);
}
```

3.3.7.2. Watchdog

Der *Watchdog* ist eine Sicherungseinrichtung des Mikrocontroller. In regelmäßigen Abständen wird überprüft ob das Watchdog Bit gesetzt ist und anschließend zurück gesetzt. Das Bit muss innerhalb der voreingestellten Zeit immer wieder neu gesetzt werden. Ist das Bit nicht gesetzt, wird der Mikrocontroller zurückgesetzt. Dies geschieht z.B. bei nicht geplanten Endlosschleifen.
Wahlweise kann kurz vor dem Reset noch die Watchdog-ISR durchlaufen werden.
Im Projekt wird hier die Fehler LED eingeschaltet und eine Meldung auf dem LC-Display ausgegeben. Siehe hierzu auch das Code-Listing 3.6.

Listing 3.6: Watchdog

```
#include <avr/wdt.h>

void init_WDT(void) {
    cli();
    wdt_reset();
    WDTCSR |= (1 << WDCE) | (1 << WDE);
    WDTCSR = (1 << WDE) | (1 << WDIE) | (1 << WDP3) | (1 << WDP0); //Watchdog 8s
    //WDTCSR = 0x0F; //Watchdog Off
    sei();
}

ISR(WDT_vect){                    // Watchdog ISR
    LED_PORT &= ~(1 << LED4);  // LED5 einschalten
    lcd_puts("Something went \nterribly wrong!\nRebooting!");
}
```

3.3.8. Protokoll der Schrittmotorkarte

Tabelle 3.1 zeigt den ASCII Befehlssatz der Schrittmotorkarte.

_A	Motorstatus liefern
_C n	konstante Geschwindigkeit einstellen
_D n	Bezugswert definieren
_E n	Motorstrom einstellen
_F	Standardeinstellungen aktivieren
_H	Sanfter stop
_I	4-Bit-Eingang lesen
_J jdss	Joystickparameter einstellen
_L n	lokalen Modus aktivieren/beenden
_M n	n Schritte ausführen
_MA n	zu n bewegen
_MC n	mit konstanter Geschwindigkeit bewegen
_MCA n	MA mit konstanter Geschwindigkeit
_MCL n	MC zu Endschalterposition
_ML n	zur Endschalterposition bewegen
_N n	Zeilenvorschub (LF, hex. 0A) einfügen/löschen
_O n	n an 4-Bit-Ausgang senden
_P nnnn	Motorparameter einstellen
_Q	Parameter in EEROM speichern
_R n	Mikroschritteilung einstellen
_RL	Endschalterwerte lesen
_RS	verbleibende Schritte lesen
_S	Nothalt
_T n	Eingang n auslösen
_W	Position anfordern

Tabelle 3.1.: ASCII Befehlssatz R+S Schrittmotorsteuerung V9141 [2001]

3.3.9. Manueller Betrieb

3.3.10. Protokolle aus RapidForm

3.3.11. Übersetungs Logik

3.3.12. Automatische Protokollwahl

4. Fazit und Zukunft

4.1. Fazit

(TODO: Fazit schreiben!)

Literaturverzeichnis

Atm 2011
Atmel (Hrsg.): *ATmega164A/PA/324A/PA/644A/PA/1284/P Complete*. San Jose, CA 95131, USA: Atmel, 06 2011 2.2

Atmel 2012a
Atmel: *ATmega324A- Atmel Corporation*. http://www.atmel.com/devices/ATMEGA324A.aspx. Version: 2012. – [Online; Stand 11. Februar 2012] 2.3

Atmel 2012b
Atmel: *ATmega8515- Atmel Corporation*. http://www.atmel.com/devices/ATMEGA8515.aspx. Version: 2012. – [Online; Stand 11. Februar 2012] 2.3

V9141 2001
RS (Hrsg.): *Schrittmotor-Platine mit integriertem Treiber*. Mörfelden-Walldorf: RS, 03 2001 3.1

Wikipedia 2012a
Wikipedia: *Daisy chain — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Daisy_chain&oldid=98475104. Version: 2012. – [Online; Stand 11. Februar 2012] 1

Wikipedia 2012b
Wikipedia: *Stiftleiste — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Stiftleiste&oldid=99052435. Version: 2012. – [Online; Stand 11. Februar 2012]

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich den vorliegenden Bericht:

> *Übersetzen von Schrittmotorprotokollen*
> *Entwurf eines Hardwareübersetzers*

selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe.
Remagen, den 12. Februar 2012

Johannes Dielmann

A. Anhang

A.1. Schritt für Schritt Anleitung

Eine Schritt für Schritt Anleitung zum vollständigen Scannen und exportieren eines 3D-Objektes.

A.2. Vom Autor verwendete Software

Hier ist die verwendete Software aufgelistet. Soweit es möglich war, wurden Open-Source-Programme eingesetzt. (TODO: Überarbeiten!!!)

- **RapidForm2004**
  asdf
- **AVRStudio 5**
  Atmel.
  Website: http://www.atmel.com/
- **Eclipse**
  Eclipse mit CDT und AVRPlugin
  Website: http://www.eclipse.org/
- **AVRDude**
  Prorammer
  (TODO: Weitere?!)