

Dokumentation des
Werkstättenprojektes
4BHELI 2007/08

XY – Plotter

Michael Stocker
Bernhard Wintersperger

Inhaltsverzeichnis

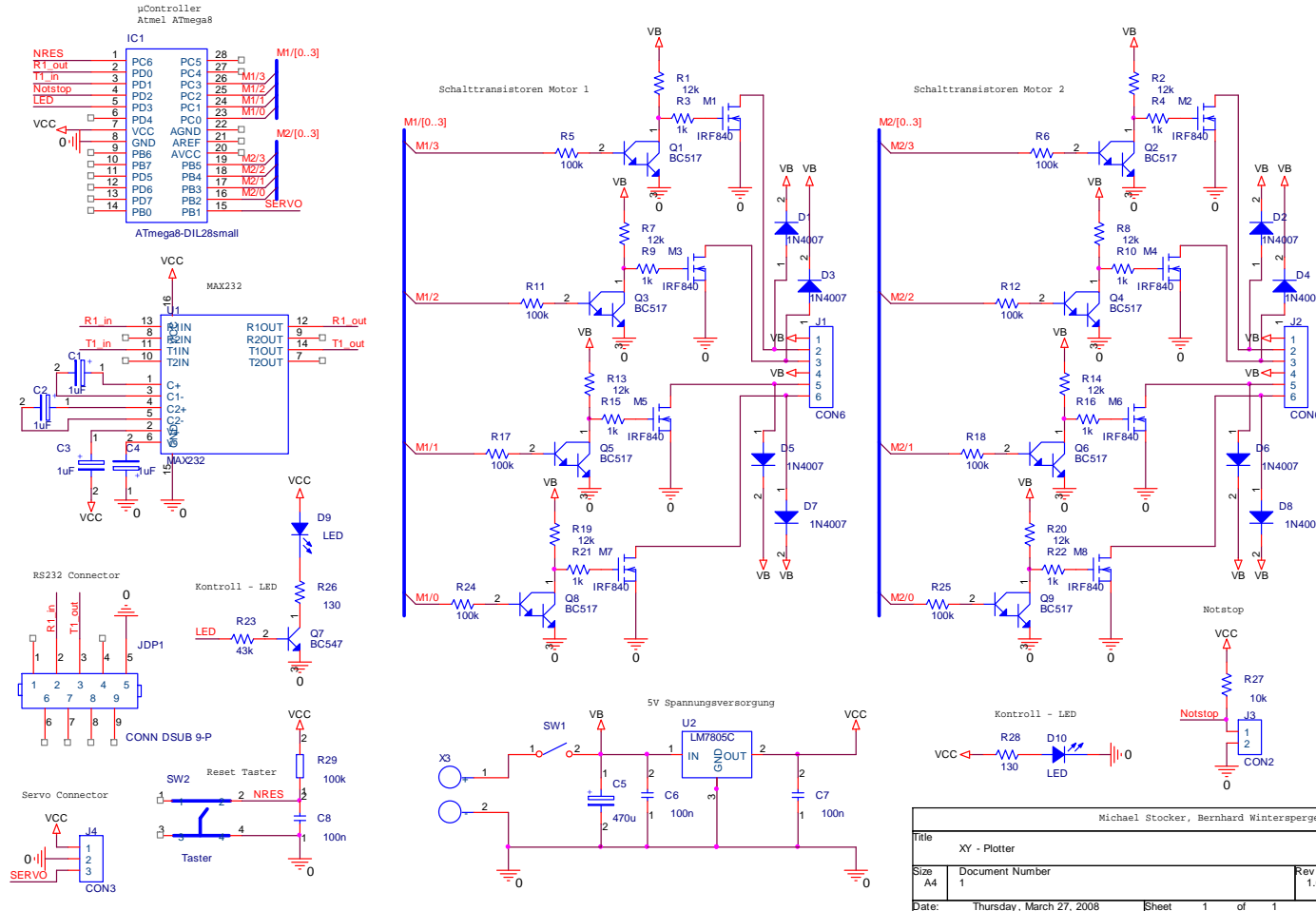
| | | |
|-------|---|----|
| 1 | Allgemeines..... | 3 |
| 2 | Hardware..... | 4 |
| 2.1 | Schaltplan: | 4 |
| 2.2 | Der μ Controller:..... | 5 |
| 2.3 | Spannungsversorgung und Reset:..... | 6 |
| 2.3.1 | Spannungsversorgung:..... | 6 |
| 2.3.2 | Reset – Schaltung:..... | 7 |
| 2.4 | MAX232: | 8 |
| 2.5 | Schalttransistoren:..... | 10 |
| 2.5.1 | MOSFET – Transistoren | 10 |
| 2.5.2 | Darlington Transistor:..... | 11 |
| 2.5.3 | Freilaufdioden: | 12 |
| 2.6 | Spannungskontroll-LED, Kontroll-LED und Notstop:..... | 13 |
| 2.6.1 | Spannungskontroll-LED | 13 |
| 2.6.2 | Kontroll-LED..... | 13 |
| 2.6.3 | Notstop:..... | 13 |
| 2.7 | Sonstiges: | 14 |
| 3 | Software μ C..... | 14 |
| 3.1 | Anforderungen | 14 |
| 3.2 | Einleitung | 15 |
| 3.3 | Hauptprogramm | 16 |
| 3.3.1 | Blockschaltbild | 16 |
| 3.3.2 | Ablauf..... | 16 |
| 3.4 | Empfangs- Interrupt..... | 17 |
| 3.4.1 | Blockschaltbild | 17 |
| 3.4.2 | Ablauf..... | 17 |
| 3.5 | Timerinterrupt..... | 17 |
| 3.5.1 | Blockschaltbild | 17 |
| 3.5.2 | Ablauf..... | 18 |
| 3.6 | Funktionen | 18 |
| 3.6.1 | setBuffer() | 18 |
| 3.6.2 | getBuffer() | 18 |
| 4 | Software PC | 18 |
| 4.1 | Initialisierung | 18 |
| 4.2 | Methoden | 19 |
| 4.3 | Screenshots | 19 |

1 Allgemeines

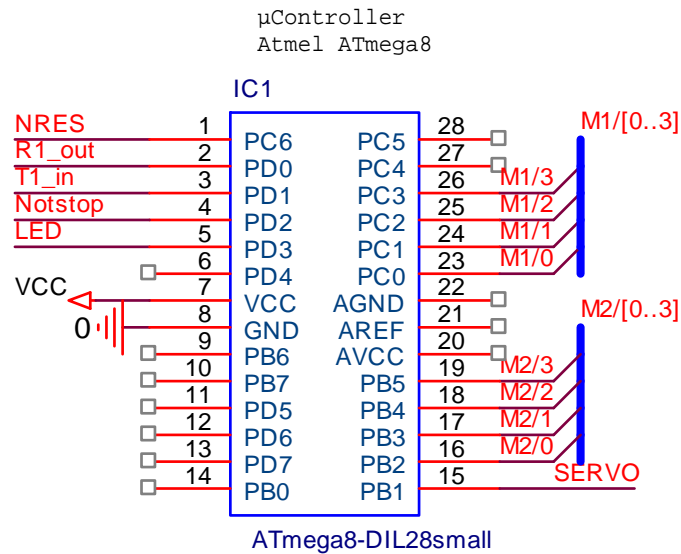
Die Idee zur Entwicklung unseres Plotters entstammt einer anderen Idee für eine Diplomarbeit. Am Beginn des Schuljahres 2007/08 kam bei Herrn Stocker und mir die Idee auf, als Diplomarbeit eine automatische Platinenbohrmaschine zu entwickeln. Ein wesentlicher Punkt hierbei war die Schrittmotoransteuerung. Nachdem wir bei unserem letztjährigen (2006/07) Werkstättenprojekt an dieser per Ansteuerungschip gescheitert waren, wollten wir es unbedingt mit einer selbst entwickelten Ansteuerungsschaltung wieder versuchen. Und so kam uns das diesjährige (2007/08) Werkstättenprojekt sehr gelegen. Einerseits konnten wir so an unserer Schrittmotoransteuerungsschaltung arbeiten und gleichzeitig in Richtung Diplomarbeit „vorarbeiten“, da viele Aspekte der technischen Herausforderungen bei einem Plotter und einer automatischen Platinenbohrmaschine dieselben sind.

2 Hardware

2.1 Schaltplan:



2.2 Der µController:



Bei der Wahl eines µControllers haben wir uns nach längerem Überlegen für einen Atmel ATmega8 entschieden. Entscheidend war die Möglichkeit zur RS232 (USART) Kommunikation, ein interner Quarz und das kleine Gehäuse (DIL28) bzw. die geringe Anzahl der IO – Ports. Nach ersten Tests (Stand: 11.06.2008) sind wir sehr optimistisch, dass der µController eine richtige Wahl war.

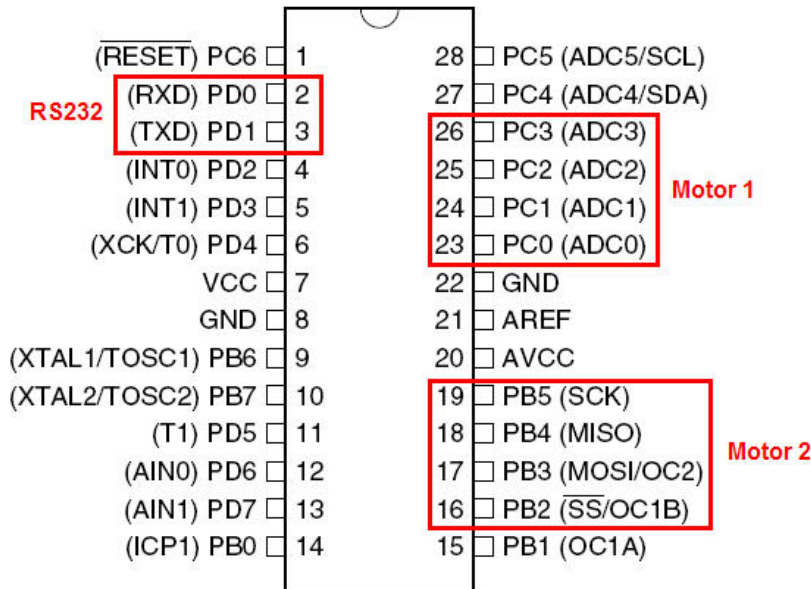


Abbildung der Pin Konfiguration mit Markierung der von uns eingesetzten Pins.

Der Atmel ATmega8 selbst, ist ein µController im RISC (Reduced Instruction Set Computing) – Design. Das bedeutet, dass bei der Entwicklung des Befehlssatzes des µControllers absichtlich auf komplexe Befehle verzichtet wurde. Dafür kann er die Befehle seines begrenzten Befehlssatzes umso schneller ausführen.¹ Das Fehlen komplexer Befehle stellt höhere Ansprüche an Assembler- und Compiler Entwickler. Durch mittlerweile gut ausgereifte Compiler wie den CodevisionAVR Compiler der

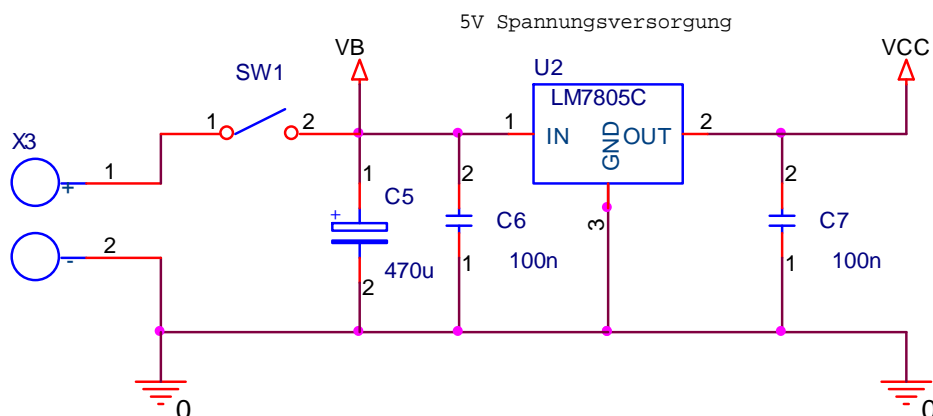
¹ Vgl. <http://de.wikipedia.org/wiki/RISC> vom 11.06.2008

Firma HP InfoTech oder das Open Source Projekt AVR GCC lässt sich der ATmega8 auch sehr bequem in C programmieren.

Neben dem RISC – Befehlssatz enthält der ATmega8 8k Byte Flash, sowie 512 Byte EEPROM Speicher. Er hat 23 programmierbare IO Ein- und Ausgänge, einen 8- und einen 16 Bit Timer. Versorgt werden kann er mit einer Spannung von 4,5 – 5,5V.

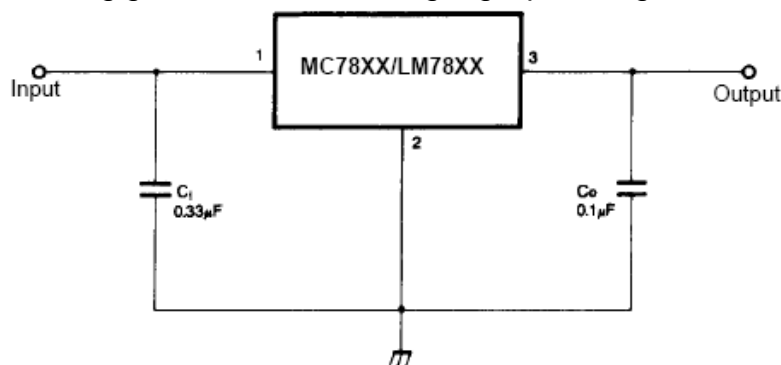
2.3 Spannungsversorgung und Reset:

2.3.1 Spannungsversorgung:



Das Herzstück der Spannungsversorgung ist ein LM7805 Spannungsregler. Dieser sorgt dafür, dass der μ Controller, (fast) unabhängig von der Eingangsspannung, konstant mit 5V versorgt wird. Die Kondensatoren dienen der Spannungsstabilisierung. Auf der fertigen Platine wurde statt dem 470 μ F Elko² ein 2200 μ F Elko verbaut. Dadurch ist die Platine gegen kurzzeitige Spannungsschwankungen geschützt. Die kleineren Folien-, bzw. Keramikkondensatoren dienen hauptsächlich dem Schutz des LM7805. Durch diese Stützkondensatoren wird der sichere Betrieb des ICs sichergestellt.

Der IC selbst ist in einem TO-220 Gehäuse untergebracht. Dieses kann aufgrund dessen Bauform und Größe relativ einfach mit Kühlkörpern versehen werden, wodurch die Leistungsfähigkeit des ICs steigt. Üblicherweise leistet ein LM7805 500mA – 1,5A, abhängig von Modell und Eingangsspannung.



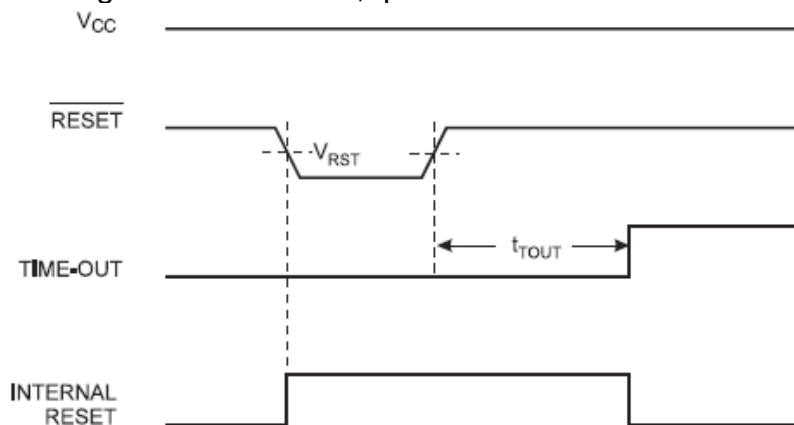
Typical Application Vorschlag aus dem Datenblatt

² Abkürzung für Elektrolytkondensator, vgl. <http://de.wikipedia.org/wiki/Elektrolytkondensator> vom 12.06.2008

2.3.2 Reset – Schaltung:

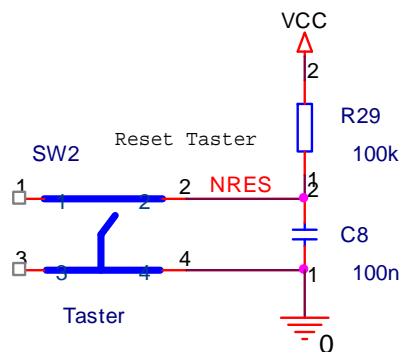
Da es sich bei dem Reset – Pin am μ Controller um einen Invertierten Pin handelt, muss folglich auch der Reset invertiert ausgeführt werden. Invertiert bedeutet in diesem Fall, dass die Spannungsänderung nicht, wie üblich, von 0V auf Reset Spannung durchgeführt wird, sondern von V_{CC} auf Reset Spannung. Die Schwelle der Reset Spannung liegt laut Datenblatt zwischen 0,2 und 0,9-mal der V_{CC} Spannung.

Wie im Reset – Zeitdiagramm relativ einfach ersichtlich ist, muss die Spannungsänderung nicht nur invertiert erfolgen, sondern auch eine gewisse Zeit t_{RST} gehalten werden. Die minimale Zeitdauer für einen Reset ist dem Datenblatt zu entnehmen und beträgt in diesem Fall $1,5\mu s$.



Reset Zeitdiagramm aus dem Datenblatt

Da auch die Steilheit der Flanken vorgegeben ist und die Reset Spannung mindestens t_{RST} ($1,5\mu s$) gehalten werden muss, eignet sich zur Erfüllung dieser Bedingungen eine RC – Schaltung sehr gut.

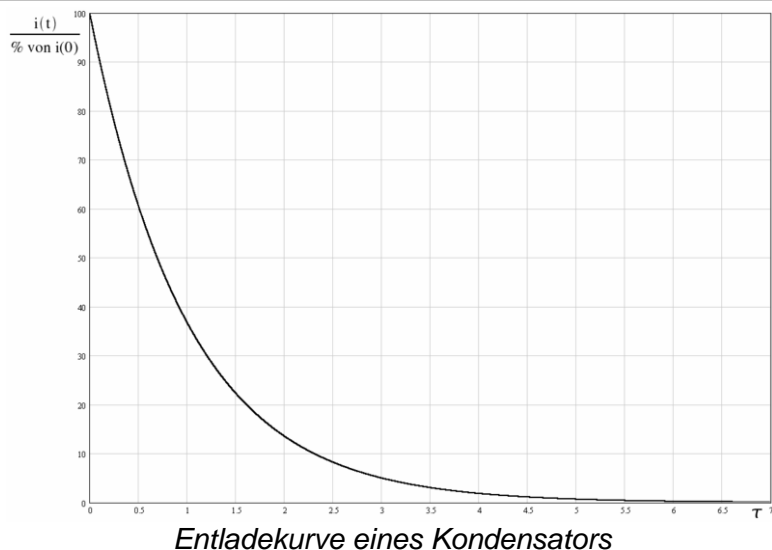


Reset Schaltung

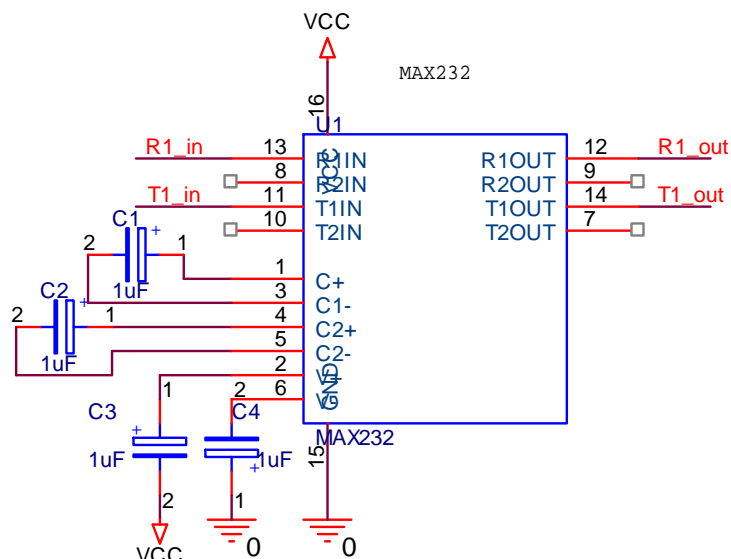
Beim Einschalten der Schaltung lädt sich der Kondensator auf und NRES hat somit das Potenzial von V_{CC} ; in unserem Fall 5V. Wenn nun der Taster gedrückt wird, entlädt sich der Kondensator und das Potenzial von NRES sinkt ab. Durch die Entladekurve des Kondensators ist gewährleistet, dass die Reset Spannung für mindestens die vorgeschriebene Zeit unterschritten wird. Die Zeit für, die die Reset Spannung unterschritten wird, lässt sich auch relativ einfach über Tau berechnen.

$$\tau = R \cdot C = (100 \cdot 10^3) \cdot (100 \cdot 10^{-9}) = 0,01s \hat{=} 10ms$$

Mit den von uns eingesetzten Bauteilen wird die Reset Spannung 10ms lang unterschritten und ein zuverlässiger Reset dadurch gesichert.



2.4 MAX232:



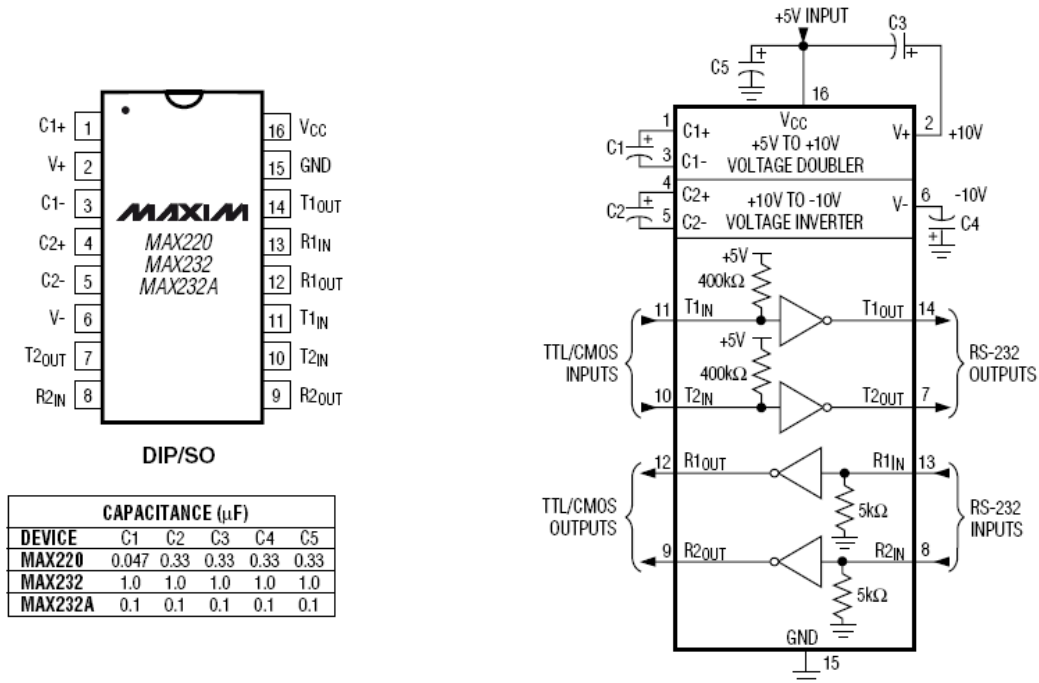
Der MAX232 Baustein der Firma Maxim ist ein Pegelwandler für die RS232 Kommunikation. Üblicherweise wird für eine solche Kommunikation ein Signalpegel von $\pm 12V$ benötigt. Viele Schaltungen arbeiten aber mit einer Versorgungsspannung von 5V. Um nun trotzdem mit einer solchen Schaltung eine RS232 Kommunikation durchführen zu können, benötigt man einen Pegelwandler wie den MAX232.

Dieser verwendet das Ladungspumpen – Prinzip, um die Spannung zu erhöhen. Bei diesem Prinzip laden sich verschiedene Kondensatoren über Dioden auf und erhöhen dadurch die Spannung am Ausgang.³ Um größere Spannungen zu speichern, braucht man auch größere Kondensatoren. Da es unmöglich ist, solch große Kondensatoren auf dem Chip selbst unterzubringen, sind im Datenblatt gewisse Kondensatoren für den Betrieb vorgeschrieben.

Da es sich bei der Schaltung für den MAX232 um eine doppelte Ladungspumpe handelt, wird die Spannung verdoppelt. Weiters wird über einen, ebenfalls mit dem

³ Vgl. <http://de.wikipedia.org/wiki/Ladungspumpe> vom 12.06.2008 und <http://www.sprut.de/electronic/switch/schalt.html#pumpe> vom 12.06.2008

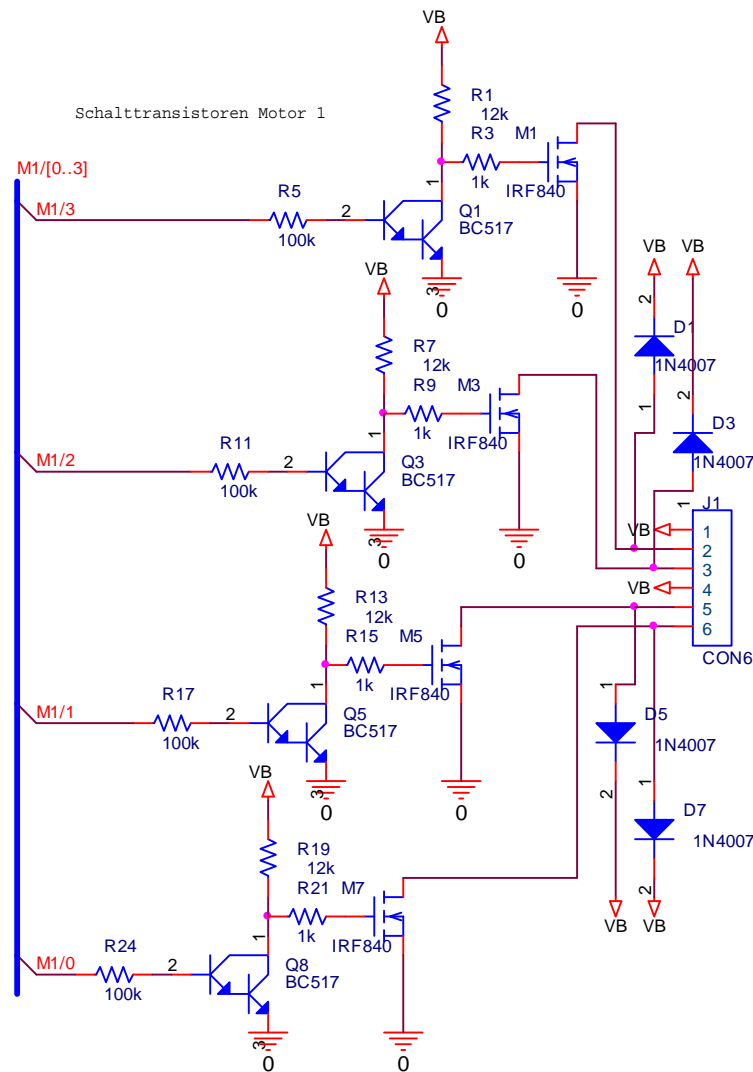
Ladungspumpen – Prinzip arbeitenden Spannungsinverter die, für die Kommunikation benötigte, negative Spannung erzeugt. Somit erhält man aus +5V $\pm 10V$. Eigentlich wird für eine RS232 Kommunikation ein Signalpegel von $\pm 12V$ benötigt, aber da $\pm 10V$ noch innerhalb des Toleranzfeldes des EIA-232 Standards liegen, sind Übertragungen somit kein Problem.⁴



Pin Konfiguration, logischer Aufbau sowie Tabelle mit je nach Modell vorgeschriebenen Kondensatoren

⁴ Vgl. <http://de.wikipedia.org/wiki/EIA-232> vom 12.06.2008

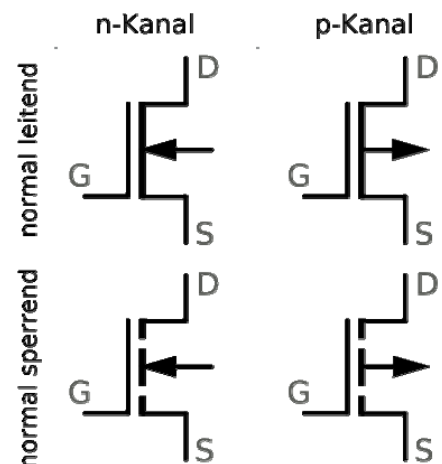
2.5 Schaltransistoren:



2.5.1 MOSFET – Transistoren

Um Schrittmotoren von einem μ Controller aus steuern zu können, bedarf es einigem zusätzlichem Schaltungsaufwand. Der Ausgang eines AVR μ Controllers kann maximal in etwa 10-15mA Strom liefern. Um einen kleineren Schrittmotor zu betreiben, werden allerdings 500mA – 2A benötigt. Es gilt also eine Möglichkeit zu finden, bei der die einzelnen Spulen eines Schrittmotors von den Ausgängen eines μ Controllers gesteuert werden können, ohne dass der μ Controller Schaden nimmt. Aufgrund der schnellen Schaltfrequenzen fallen Relais von vornherein aus. Eine gute Möglichkeit um hohe Ströme mit höheren Frequenzen schalten zu können, sind MOSFET Transistoren.

Diese Transistoren arbeiten mit dem Prinzip der CMOS Technologie und sind bestens zum Schalten größerer Ströme geeignet. Im Gegensatz zu den Bipolar – Transistoren hängt die schaltbare Last nicht vom Basisstrom, sondern von der Gatespannung ab. Es gilt lediglich am Gate eine entsprechende Spannung

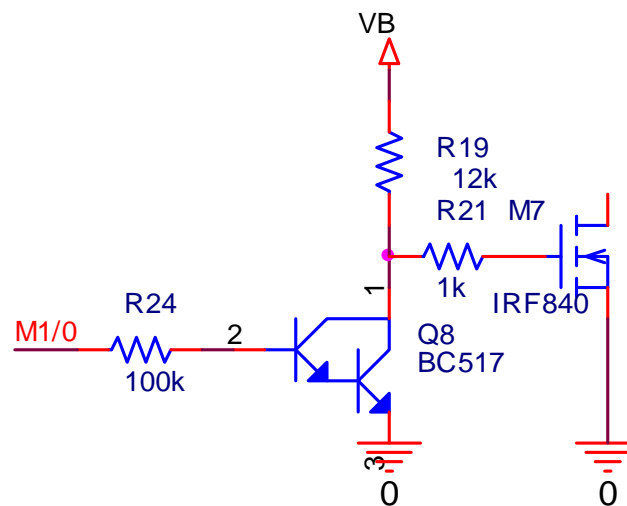


anzulegen und die Last kann über die Drain – Source Strecke quasi verlustfrei geschaltet werden.

Um nun Motoren schalten zu können gilt, es allerdings die Induktivität deren Spulen nicht zu vergessen. Aus diesem Grund haben wir für unsere Schaltung einen MOSFET – Transistor mit integrierter Freilaufdiode eingesetzt und zusätzlich, zum Schutz der restlichen Schaltung, weitere Freilaufdioden verbaut.

2.5.2 Darlington Transistor:

Da der μ Controller an einem seiner Ausgänge nur maximal 5V liefern kann und zum Schalten der Motoren eine höhere Gatespannung benötigt wird, wurde zwischen dem μ Controller und dem MOSFET – Transistor ein Darlington Bipolar Transistor geschaltet. Ursprünglich war an dieser Stelle ein normaler BC547 Bipolar Transistor eingeplant. Aufgrund dessen niedriger Stromverstärkung wäre es allerdings nicht möglich gewesen, mit dem vom μ Controller Ausgang zur Verfügung stehenden Basisstrom, den MOSFET ausreichend zum Durchsteuern zu bringen. Durch den Einsatz eines BC517 Darlington Transistors ist dies nun problemlos möglich.



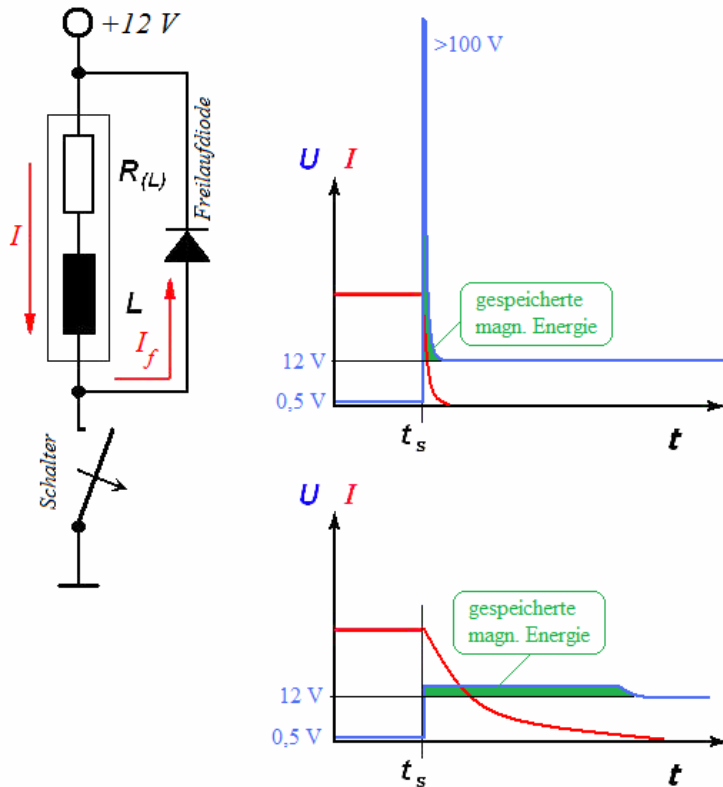
Vergrößerte Abbildung einer „Schalteinheit“

Anders als in der Abbildung zu sehen wurde bei der fertigen Platine, aufgrund der einfacheren Verfügbarkeit, statt eines IRF840 ein IRL3103 MOSFET verwendet. Dieser hat statt eines maximalen Drainstroms von 8A einen maximalen Drainstrom von 64A. Beide sind von der Anschlussbelegung her identisch und haben intern beide eine Freilaufdiode integriert. Der Kollektorwiderstand beim Darlington Transistor ist so dimensioniert das bei einer Spannung V_B von 12V ein Kollektor – Emitter Strom von 1mA fließt. Der Basis – Vorwiderstand des BC517 ist relativ klein dimensioniert um sicherzustellen, dass der Transistor beim Einschalten an der Kollektor – Emitter Strecke voll durchsteuert.

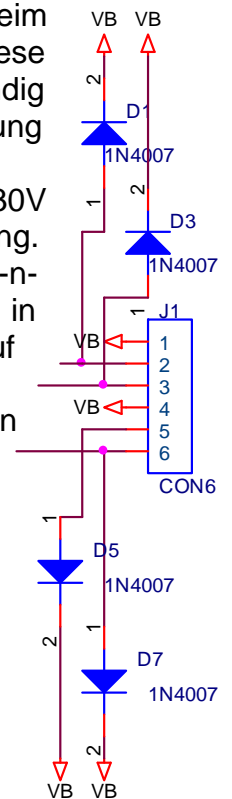
Da wir für unseren Plotter ein X – und eine Y – Achse benötigen, ist auf der Platine der auf Seite 10 abgebildete Schaltplan zwei Mal vorhanden.

2.5.3 Freilaufdioden:

Eine Freilaufdiode dient Grundsätzlich zum Schutz vor Überspannungen beim Ausschalten einer induktiven Gleichspannungslast. In unserem Fall sind diese Dioden sehr wichtig, da bei der Ansteuerung der Schrittmotoren ständig induktive Lasten ein- und ausgeschaltet werden. Anders als in der Abbildung zu sehen haben wir uns auf unserer Platine für SSB43L Schottky – Dioden in DO-214 SMD Bauform entschieden. Diese Diode leistet maximal 30V Sperrspannung und 4A I_F bei nur 0,38V Spannungsabfall in Durchlassrichtung. Schottky Dioden gelten allgemein als schnelle Dioden da sie statt einem p-n-Übergang einen Halbleiter-Metall-Übergang verwenden. Sie werden in Hochfrequenzanwendungen bis hin zum Mikrowellenbereich eingesetzt.⁵ Auf unserer Schaltung sind die Dioden parallel zu den induktiven Lasten verbaut und schützen dadurch alle anderen Bauteile vor gefährlichen Spannungsspitzen.



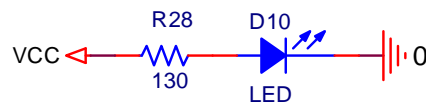
Darstellung der Schutzfunktion einer Freilaufdiode



⁵ Vgl. <http://de.wikipedia.org/wiki/Schottky-Diode> vom 12.06.2008

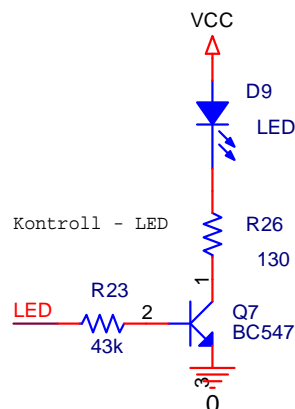
2.6 Spannungskontroll-LED, Kontroll-LED und Notstop:

2.6.1 Spannungskontroll-LED



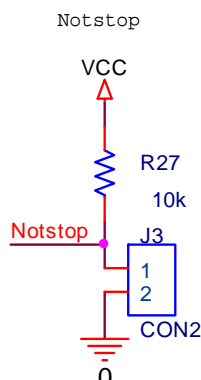
Zu Kontrolle ob die Schaltung überhaupt mit Spannung versorgt wird, wurde eine Normale LED-Diode mit Vorwiderstand vorgesehen. Der Widerstand ist hier so dimensioniert, dass die LED mit 3V und 15mA versorgt wird.

2.6.2 Kontroll-LED



Die Kontroll-LED lässt sich von Port D.2 vom μ Controller aus ansteuern. Sie dient zur Kontrolle der Funktion des μ Controlles. Theoretisch wäre es zwar mögliche eine LED direkt vom Ausgang des μ Controlles aus zu treiben, dies ist allerdings riskant und könnte eine Beschädigung nach sich ziehen. Aus diesem Grund wird die LED über einen BC547 Bipolartransistor geschaltet. Der Vorwiderstand der LED ist wie bei der Spannungskontroll-LED so dimensioniert, dass die LED mit 3V und 15mA versorgt wird.

2.6.3 Notstop:

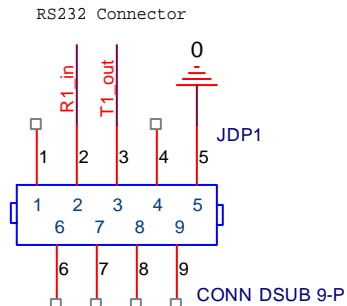


Zur Sicherheit wurde die Möglichkeit eines Notstop – Tasters auf dem Schaltplan integriert. Die Schaltung für den Taster selbst wurde der Beschreibung zum STK500 Entwicklungsboard entnommen. Durch den Pull-up Widerstand liegt in nicht gedrücktem Zustand ein logisches High am μ Controller Eingang an. Als Eingang am μ Controller wurde ein externer Interrupt Port gewählt. Dadurch ist sichergestellt das

bei einem Druck auf den Notstop – Taster das aktuell ausgeführte Programm im μ Controller unterbrochen wird.

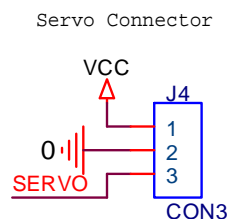
2.7 Sonstiges:

Neben den bereits beschriebenen Komponenten gibt es zwei weitere die auf der Platine verbaut sind.



Zum einen ist das ein RS232 9-Poliger DSUB Stecker. Der Stecker wurde nach dem EIA-232 Standard angeschlossen. Pin 3 ist der TxD Pin und wird zum senden von Daten verwendet. Pin 2 ist der RxD Pin und wird zum empfangen von Daten verwendet. Pin 5 ist der Masse Pin und wird benötigt um auf beiden, miteinander kommunizierenden, Seiten ein gemeinsames Masse Potential herzustellen.

Zum anderen ist es ein Connector zum Anschluss eines Servomotors.



Durch dieses Connector ist der Betrieb eines 5V Servomotors an der Schaltung möglich. Die SERVO Leitung ist am μ Controller an einem PWM (Pulsweitenmodulation) – Pin angeschlossen. Bei entsprechender Programmierung des μ Controllers ist es somit problemlos möglich einen Servomotor als Z – Achse, Beispielsweise für Bohrungen, zu betreiben.

3 Software μ C

3.1 Anforderungen

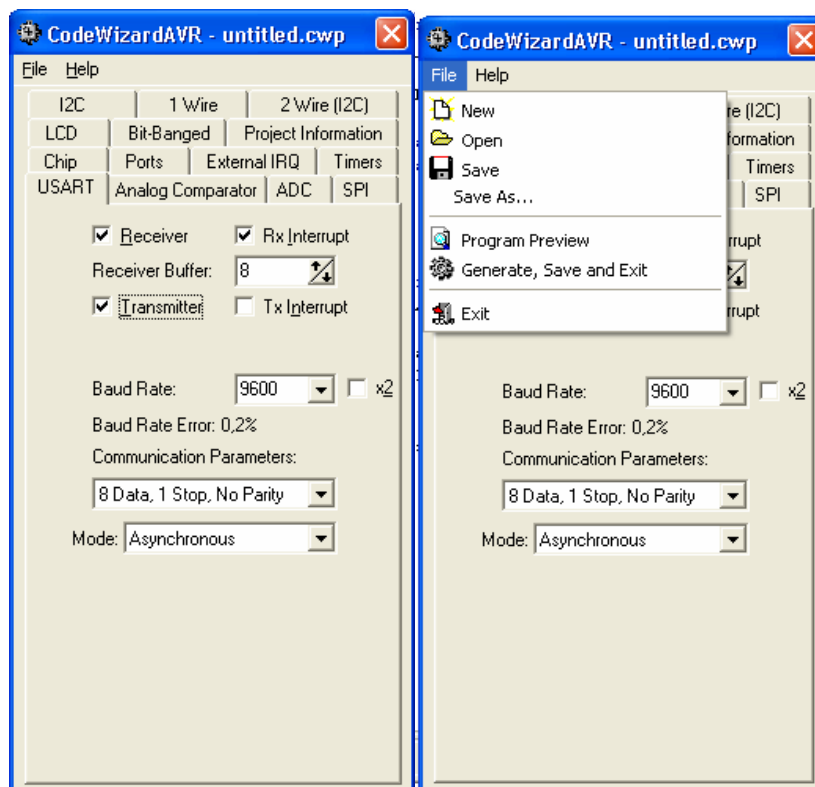
Die Anforderungen an das μ C Programm sind das Empfangen von Positionsdaten und das drauf folgende richtige Ansteuern der Schrittmotoren. Außerdem soll ein Softwareseitiger Notaus implementiert werden, der bei Eintreffen eines bestimmten Zeichens den Timer stoppt.

Eine weitere wichtige Funktion, ist das Ändern der Geschwindigkeit während der Plotter arbeitet. Dies wird durch das Ändern des Timers realisiert und wird ausgeführt sobald ebenfalls ein bestimmtes Zeichen am μ C eintrifft.

3.2 Einleitung

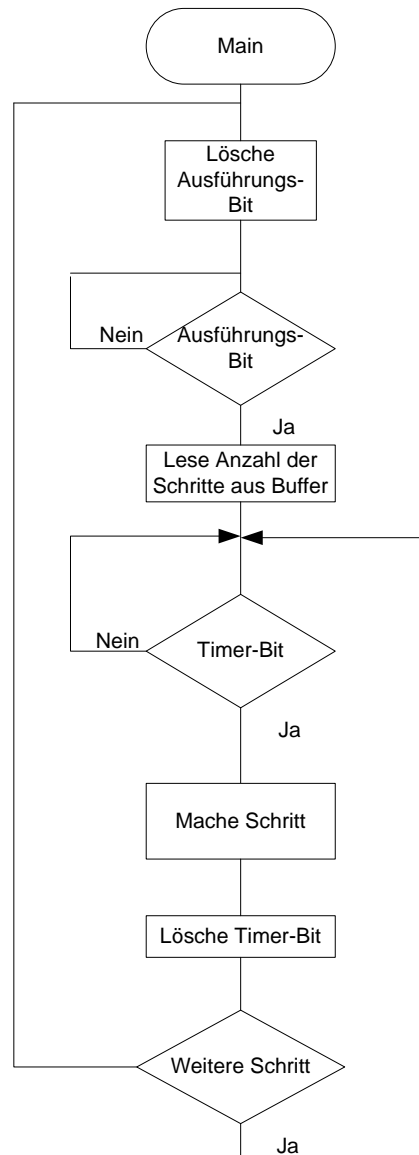
Da ein μC kein zu Grunde liegendes Betriebssystem besitzt, muss das abzuarbeitende Programm auf dem μC so konzipiert sein, dass es auch noch auf Aktionen vom PC reagiert.

Prinzipiell gibt es daher zwei Verfahren, um Daten zu empfangen (Polling und Interrupt). Beim Pollingverfahren werden die Daten mit der `scanf()` Funktion eingelesen. Beim Interruptverfahren wird ein Interrupt ausgelöst, sobald neue Daten im Empfangsbuffer vorhanden sind. Der Nachteil bei dieser Methode liegt darin, dass nur jeweils ein `char` Zeichen bearbeitet werden kann und nicht wie bei der Polling-Methode auch ein ganzer String empfangen wird. Da es in unserem Programm jedoch auch wichtig ist, im laufenden Betrieb die Maschine softwareseitig zu stoppen oder die Geschwindigkeit zu ändern, greifen wir auf die Interrupt-Methode zurück. Um einen Receive-Interrupt in CodeVision mit dem CodeWizardAVR zu aktivieren, sind folgende Schritte nötig. Als erstes muss der CodeWizard unter Tools \rightarrow CodeWizard oder mithilfe der Tastenkombination „Umschalt + F2“ gestartet werden. Unter dem Unterpunkt USART wird danach das Häkchen bei Receive, RX-Interrupt und Transmitter gesetzt. Der so generierte Code enthält neben den Konfigurationseinstellungen auch schon eine einfache Interruptroutine, die entsprechend unseren Bedürfnissen angepasst wurde.



3.3 Hauptprogramm

3.3.1 Blockschaltbild

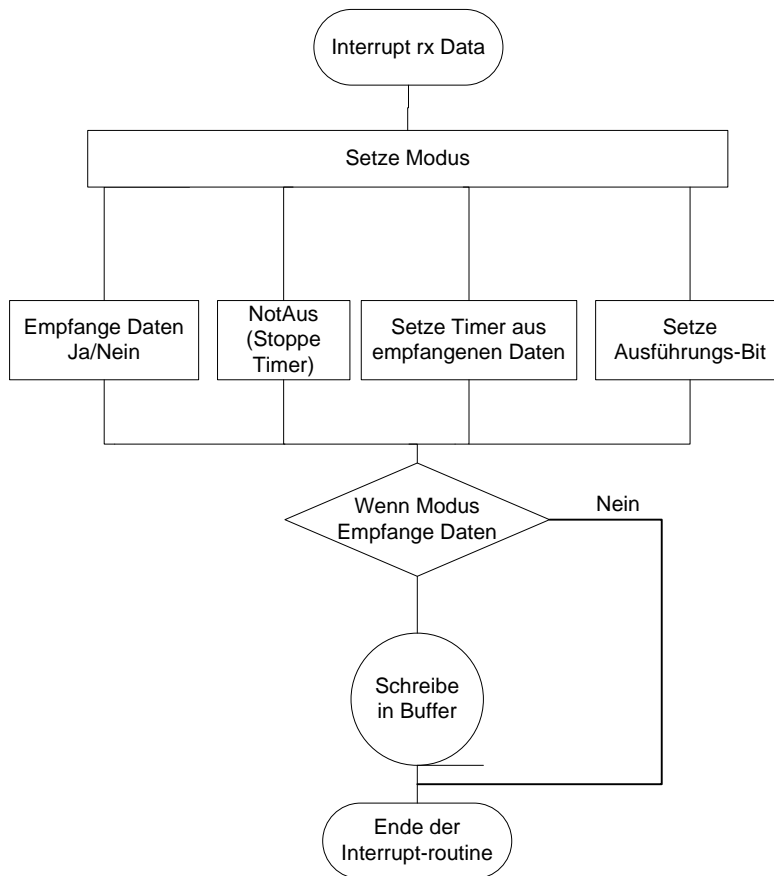


3.3.2 Ablauf

Sobald die das Ausführungsbit durch die RS232 Interrupt Funktion gesetzt wird, werden die Empfangenen Daten aus der Bufferfunktion ausgelesen mit einem Faktor multipliziert und in einer Variable abgespeichert. An der Überprüfung ob das Timerbit gesetzt ist, wird so lange gewartet bis das Timerbit durch den Timerinterrupt gesetzt wird. Sollte dies der Fall sein wird überprüft ob ein Schritt in x bzw. y Richtung nötig ist und gegebenenfalls ausgeführt. Danach wird wieder überprüft ob ein Schritt nötig ist und je nachdem an den Anfang des Programms gesprungen wo wieder so lange gewartet wird bis neue Daten Empfangen wurden und das Ausführungsbit gesetzt wird.

3.4 Empfangs- Interrupt

3.4.1 Blockschaltbild

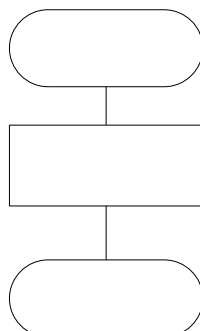


3.4.2 Ablauf

Der Empfangsinterrupt wird ausgelöst sobald Daten am μC eintreffen. In unserem Programm wird je nach empfangenen Zeichen entweder der Empfangsmodus aktiviert oder eine Aktion ausgeführt. Sollte der Empfangsmodus aktiviert werden, werden eingehenden Daten an die Bufferfunktion übergeben und abgespeichert. Danach wird durch das Senden eines weiteren Zeichens der Empfangsmodus deaktiviert und entweder das Ausführungsbit gesetzt oder die Empfangenen Daten als neue CTC Timerwerte verwendet (wodurch die Geschwindigkeit verändert wird).

3.5 Timerinterrupt

3.5.1 Blockschaltbild



3.5.2 Ablauf

Da der Timer in den CTC Modus gesetzt wurde, wird zu bestimmten Zeiten der Timer Interrupt ausgelöst. Der Timer Interrupt dient dem Setzen des Timerbits wodurch das Hauptprogramm veranlasst wird einen weiteren Schritt auszuführen.

3.6 Funktionen

Der Buffer dient dem zwischenspeichern eintreffender Werte und wird durch die Funktionen `setBuffer()` und `getBuffer()` realisiert. Er stellt einen FIFO – Stack Speicher dar.

3.6.1 `setBuffer()`

```
void setBuffer(char data) {  
    buffer[schreiben%buffersize] = data;  
    schreiben++;  
}
```

Wird der Funktion `setBuffer()` ein Wert übergeben, wird dieser in das Array `buffer` geschrieben. Dabei wird die Variable `schreiben` um eins erhöht damit der nächste Wert in das nächste Array Feld geschrieben wird. Damit der Wert `schreiben` nicht die definierte Arraygröße überschreitet, wird auf die Variable die Modulo Funktion angewandt. Wird zum Beispiel bei einer Buffergröße von 3 der Wert 2 erreicht, wird aufgrund der Modulo Division der Index 0 angesprochen. Wird der Wert 4 erreicht, wird das Array Feld 1 angesprochen.

3.6.2 `getBuffer()`

Die Funktion `getBuffer` liest die zuvor abgespeicherten Werte wieder aus dem Buffer-Array aus. Dabei wird bei jedem Aufruf die `lesen` Variable um eins erhöht, damit das nächste Array Feld ausgelesen wird. Um den `lesen` und `schreiben` Zeiger synchron zu halten, wird der Wert aus der `lesen` Variable der `schreiben` Variable zugewiesen.

```
char getBuffer() {  
    lesen++;  
    schreiben=lesen+1;  
    return buffer[lesen%buffersize];  
}
```

4 Software PC

Die Software auf dem PC wurde aufgrund der einfachen Handhabung in C# realisiert. Derzeit wurden jedoch nur zwei kleine Demonstrationsprogramme geschrieben weshalb hier auch nur auf die für die Kommunikation zuständige RS232 Klasse eingegangen werden soll.

4.1 Initialisierung

Die Klasse wird ohne Übergabeparameter initialisiert. Erst nach der Auswahl des COM Ports und setzen einiger anderen Einstellungen, kann die Verbindung mit den gegebenen Werten und der `open()` Methode geöffnet werden. Nach der erfolgreichen Initialisierung und Öffnung der Verbindung kann auf folgende Methoden zugegriffen werden. Sollte der übergebene COM Port bereits besetzt sein wird eine Fehlermeldung ausgegeben.

4.2 Methoden

public void open(string COM, int baudrate, int databits, bool rtenable)

Nach der Initialisierung der Klasse wird die COM Verbindung mit der open() Methode und den übergebenen Parametern geöffnet. Sollte der übergebenen COM Port bereits verwendet werden, wird eine Fehlermeldung ausgegeben.

public void writePosition(int x, int y)

Der Methode writePosition() können die zu absolvierenden Schritte übergeben werden. Danach werden diese an den μ C übertragen.

private void serialPort1_DataReceived ()

Die nicht öffentliche Methode **serialPort1_DataReceived()** wird dem rs232 Empfangs-Eventhandler zugewiesen und wird ausgeführt sobald neue Daten eintreffen. Wenn der Wert 245 empfangen wird, wird das ready Flag gesetzt und somit signalisiert, dass der μ C bereit für neue Positionsdaten ist.

public bool getReady

Über die Eigenschaft getReady kann abgefragt werden ob die letzte Positions-Änderung erfolgreich ausgeführt wurde.

4.3 Screenshots

Aufgrund von Zeitmangel konnte die fertige Klasse leider noch nicht in das bereits vorbereitete Interface implementiert werden.

Abschließend sind hier noch zwei Screenshots des Interfaces zur Ansteuerung des Plotters zu sehen.

