Diplomarbeit

Fachhochschule Wilhelmshaven
- Fachbereich Elektrotechnik -
Wintersemester 1999/2000

Entwicklung, Aufbau und Erprobung eines Low-Cost-Steuergerätes mit Mikrocontroller für den universellen Einsatz zur Steuerung von Drehstrom-Asynchronmaschinen

Prof. Dipl. Ing. A. Burgholte
- Erstprüfer –

Prof. Dr. R. Geyer
- Zweitprüfer –

Erklärung

Hiermit erkläre ich, dass ich die Arbeit selbstständig und ohne Verwendung anderer als der angegebenen Hilfsmittel verfasst habe. Die aus fremden Quellen entnommenen Gedanken sind als solche kenntlich gemacht.

Die vorliegende Arbeit wurde bisher keiner anderen Prüfungsbehörde in gleicher oder ähnlicher Form eingereicht und auch nicht veröffentlicht.

20.12.1999

(Christian Finger)

Gliederung

1. Einführung

2. Grundlagen

2.1 Leistungshalbleiter und Halbleiterschalter
2.2 Wichtige Parameter von Thyristoren und Triacs

2.2.1 Kritische Spannungssteilheit (d*u*/d*t*)
2.2.2 Kritische Stromsteilheit d*i*/d*t*
2.2.3 Durchlassverluste
2.2.4 Sperrspannungsbelastung

2.3 Definition des Steuerwinkels
2.4 Spannungssteuerung durch Phasenanschnitt

2.4.1 Einphasige Systeme (Wechselstromsteller), ohmsche Last
2.4.1.1 Steuerkennlinie des Wechselstromstellers
2.4.2 Phasenanschnitt mit ohmsch-induktiver Last

2.5 Phasenanschnitt im Drehstromsystem

2.5.1 Simulationsumgebung
2.5.2 Zündimpulsschema
2.5.3 Stromverlauf an der Last
2.5.4 Unsymmetrien im Stromfluß

2.6 EMV-Problematiken

2.6.1 Entstehung von Steuerblindleistung
2.6.2 Netzrückwirkungen durch Phasenanschnitt
2.6.3 EMV Gesetzgebung

2.6.3.1 Störemisssion
2.6.3.2 Störfestigkeit

3 Schaltungsrealisierung

3.1 Anforderungen an die zu realisierende Schaltung

3.1.1 Einstellmöglichkeiten

3.2 Systemstruktur
3.3 Entwicklung der Schaltungsteile

3.3.1 Das Netzteil
3.3.1.1 Primäre Schutzglieder
3.3.1.2 Sekundäre Schutzglieder
3.3.1.3 Gleichspannungserzeugung
3.3.2 Nulldurchgangsdetektion
3.3.3 Spannungsmessung
3.3.4 Messung weiterer äußerer Größen
3.3.5 Ausgangsstufen
3.3.6 Buskonzept
3.3.6.1 Bedienungskonzept
3.3.6.2 Speicherung der Konfigurationsdaten
3.3.6.3 Zusätzliche Schalt-Ein- und Ausgänge
3.3.6.4 Struktur des I^2C-Busses
3.3.7 Die Controllerplatine

3.4 Entwicklung der Leiterplatten

4.4.1 Entwicklung der Controllerplatine

4. Softwareentwicklung

4.1 Strategie zur Zündimpulserzeugung
4.1.1 Errechnung der Ladewerte für die Timer 1 und 2 :
4.2 Einleitung eines Hoch- oder Auslaufens
4.3 Menüerzeugung
4.4 Steuerung des Displays

4.4.1 Zeichenausgabe auf dem Display
4.4.2 Kommandoübertragung zum Display
4.4.3 Cursorpositionierung

4.5 Steuerung der Tastatur
4.6 Schreiben und Auslesen der Konfigurationsdaten
4.7 Programmierdetails

ANHANG

A Literatur

- Bücher
- Zeitschriften
- Datenblätter
- Sonstige Schriften

B Listings der Simulationen

- Simulation einphasiger Anschnittsteller mit ohmscher Last
- Simulation einphasiger Anschnittsteller mit ohmsch-induktiver Last
- Simulation dreiphasiger Anschnittsteller mit ohmscher Last
- Simulation Spannungsmessung über Optokoppler
- Simulation Nulldurchgangsdetektor

C Stücklisten der Platinen

- Stückliste der Controllerplatine
- Stückliste der Hauptplatine

D Schaltpläne

- Schaltplan der Controllerplatine
- Schaltplan der Hauptplatine

E Platinenlayouts

- Layout der Controllerplatine
- Layout der Hauplatine

F Listing der Software

1. Einführung

Die Arbeitsweise jedes Stromrichters besteht in periodischen Schaltvorgängen, die von Stromventilen vorgenommen werden.
Zur Steuerung von Asynchronmaschinen eignen sich grundsätzlch zwei Arten von Stromrichtern. Man unterscheidet zwischen Umrichtern mit Speicherkreis und solchen ohne Zwischenspeicherung der elektrischen Energie.
Stromrichter mit Speicherkreis erzeugen das notwendige Drehfeld zum Antrieb einer Asynchronmaschine selbst, man spricht daher auch von selbstgeführten Stromrichtern, da diese unabhängig sind von der Spanung und der Frequenz des Versorgungsnetzes.
In dieser Diplomarbeit werden Stromrichter mit Speicherkreis nicht behandelt. Zentrales Thema ist ein sogenannter fremdgeführter, das heißt mit der Netzfrequenz synchron arbeitender, Stromrichter.

2. Grundlagen

2.1 Leistungshalbleiter und Halbleiterschalter

Halbleiterschalter, also Stromventile, sind die Basis für jeden modernen Stromrichter. Zur Verfügung stehen hier als Grundelemente Dioden, Triacs, Thyristoren und Transistoren im Schalterbetrieb.
Zusammenschaltungen dieser Grundfunktionen ermöglichen den Aufbau von Leistungsmodulen mit verschiedenen Eigenschaften.
Zum Einsatz kommen in dieser Anwendung Leistungsmodule aus antiparallel geschalteten Thyristoren.

2.2 Wichtige Parameter von Thyristoren und Triacs

Zu den wichtigesten Parametern von Thyristoren und Triacs zählen die maximale Sperrspannungsbelastung, die kritische Spannungssteilheit ($\mathrm{d}u/\mathrm{d}t$), die kritische Stromsteilheit ($\mathrm{d}i/\mathrm{d}t$) sowie der maximale Durchlassstrom.
Als Vergleich dienen hier der Thyristor BSTC03 (600V, 12 A, Siemens) sowie der Triac TXC01 (600 V, 7.5 A, Siemens).

2.2.1 Kritische Spannungssteilheit (d*u*/d*t*)

Dies ist der größte Wert der Spannungssteilheit in Vorwärtsrichtung, bei dem der Thyristor oder Triac ohne Steuerimpuls noch nicht vom sperrenden in den leitenden Zustand übergeht. Für den genannten Thyristor gilt laut Datenblatt $(du/dt)_{krit}$=200V/µs, für den Triac hingegen nur $(du/dt)_{krit}$=50V/µs.

2.2.2 Kritische Stromsteilheit d*i*/d*t*

Dies ist die höchstzulässige Anstiegssteilheit des Stromes beim Einschalten, bei der noch keine Beeinträchtigung der elektrischen Eigenschaften des Thyristors oder Triacs eintritt. Dieser Wert ist abhängig von der Betriebsfrequenz.
Für den genannten Thyristor gilt laut Datenblatt $(di/dt)_{krit}$=150 A/µs, für den Triac hingegen nur $(di/dt)_{krit}$=20A/µs.

2.2.3 Durchlassverluste

Dieser Parameter bestimmt vor allem den für die Kühlung der Bauteile notwendigen Aufwand. Dabei gilt für den Durchlastverlust P_T

$$P_T = U_{T0} * I_{TAV} + r_T * I^2_{TRMS}$$

mit U_{T0}=Schleusenspannung, I_{TAV}=Mittelwert des Laststromes, r_T=differentielle Widerstand und I_{TRMS}=Effektivwert des Laststromes. Im Vergleich liegen die genannten Bauteile bei

Parameter	Thyristor BSTC03	Triac TXC01
U_{TO} [V]	1	1.04
r_T [mΩ]	33	57

Tabelle 1. Parameter zur Verlustrechnung

2.2.4 Sperrspannungsbelastung

Im gesperrten Zustand darf die anliegende Spannung die zulässigen Grenzwerte nicht überschreiten, da der Halbleiter sonst in in Sperrrichtung in den leitenden Zustand übergehen kann (sogenanntes Überkopf-Zünden). Für den genannten Thyristor ist dieser Wert gleich dem des Triac (U_{RRM}=600V). Aus Sicherheitsgründen legt man diesen Parameter mit einem Faktor s=2..3 über der maximalen Betriebsspannung aus.

2.3 Definition des Steuerwinkels

Als Steuer- oder Zündwinkel wird der von der Speisespannung eines Wechselspannungssystems überstrichene Winkel $\alpha=\omega t$ bezeichnet, an welchem das zugehörige Stromventil aufgesteuert (gezündet) wird :

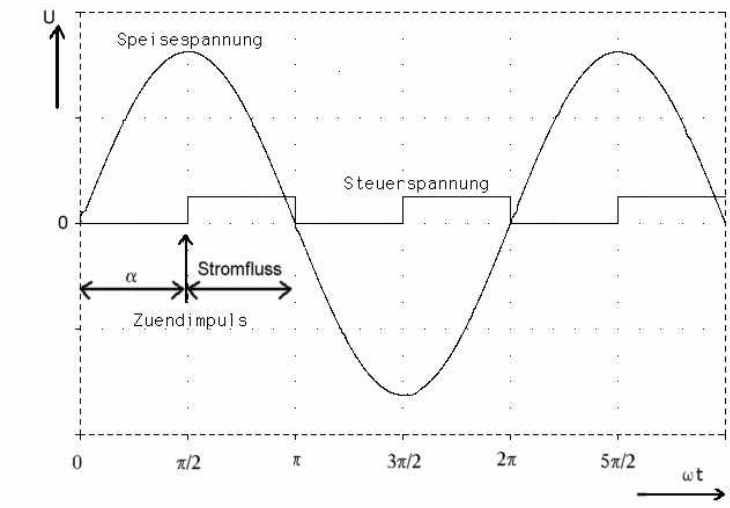


Bild 1. Definition des Steuerwinkels

Hier überstreicht die Speisespannung nach ihrem Nulldurchgang den Winkel $\omega t=\pi/2$, bis das Stromventil mit der positiven Flanke der Steuerspannung aufgesteuert wird. Damit wird in diesem Fall der Steuerwinkel $\alpha=90°$, welcher im Bereich von 0° bis 180° (Entsprechend 0-π) variieren kann. Die tatsächliche Dauer des Stromflusses durch das Ventil hängt vom Laststrom ab.

2.4 Spannungssteuerung durch Phasenanschnitt

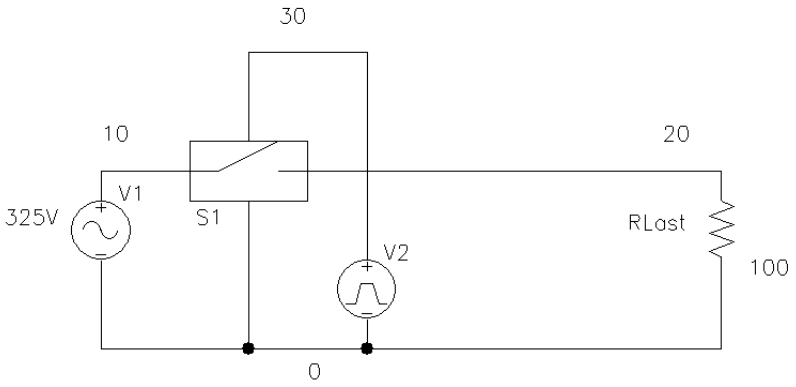
Ein Halbleitersteller für den Phasenanschnitt sieht ein in jeder Halbwelle der Speisespannung wiederholtes Einschalten vor, wobei der Strom jeweils vom Zündzeitpunkt bis zu seinem natürlichen Nulldurchgang fliesst.
Durch dieses Verfahren läßt sich die effektive Spannung und damit die Leistung am Verbraucher von Null bis zum Maximum stufenlos einstellen.
Der Effektivwert der Spannung an einer ohmschen Last berechnet sich nach der Definition des Effektivwertes einer Wchselspannung zu

$$U_{eff} = \sqrt{\frac{1}{p}\int_{a}^{p} (\hat{u} * \sin(wt)^2 \, dwt}$$

Gleichung 1. Effektive Spannung in Abhängigkeit vom Steuerwinkel beo ohmscher Last

2.4.1 Einphasige Systeme (Wechselstromsteller), ohmsche Last

Um die wichtigen theoretischen Grundlagen zu erarbeiten soll eine ohmsche Last R_{Last} an einer Wechselspannungsquelle V_1 durch einen Anschnittsteller angesteuert werden. Dieser Steller besteht idealisiert aus einem spannungsgesteuerten Schalter S_1, welcher durch die Steuerspannungsquelle V_2 geöffnet bzw. geschlossen wird.
Die Ansteuerung erfolgt nach der Definition des Steuerwinkels (siehe *Definition des Steuerwinkels*) nach Überstreichen des Zündwinkels α und läßt den Schalter bis zum nächsten Nulldurchgang des Laststromes geschlossen.



Schaltbild 1. Schaltbild zur Simulation Phasenanschnittsteller für Wechselstrom

Die Simulation des Phasenanschnitts mit Steuerwinkel α=90° ergibt folgende Spannung über R_{Last} und den Effektivwert (RMS) der Spannung :

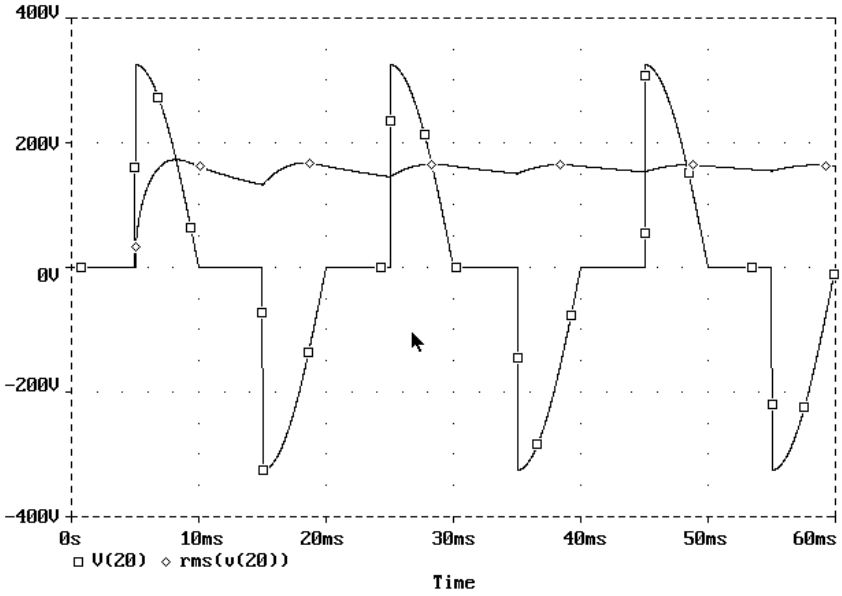


Bild 2. Phasenanschnitt mit Steuerwinkel α=90°

2.4.1.1 Steuerkennlinie des Wechselstromstellers

Die effektive Spannung über R_{Last} nimmt mit steigendem Steuerwinkel α ab. Die effektive Spannung am Verbraucher ist

$$U_{eff} = \sqrt{\frac{1}{p}\int_a^p \left(\hat{u}\sin wt\right)^2 dwt}$$

Daraus folgt nach Umstellung

$$\frac{u^2{}_{eff}}{\hat{u}^2} p = \int_a^p \sin^2 wt\, dwt$$

Das bestimmte Intergral ergibt nach der Lösung mit dem Additionstheorem

$$\sin a * \cos a = \frac{1}{2}\sin 2a$$

sowie der Beziehung

$$\hat{u} = U_N * \sqrt{2}$$

die endgültige Form Ueff=f(α [rad]) mit

$$U_{eff} = U_N \sqrt{1 - \frac{a}{p} + \frac{1}{2p} \sin 2a}$$

Gleichung 2. Steuerkennlinie Phasenanschnittsteller

und ergibt die Steuerkennlinie eines einphasigen Wechselstromstellers mit ohmscher Last (mit α=0..180°):

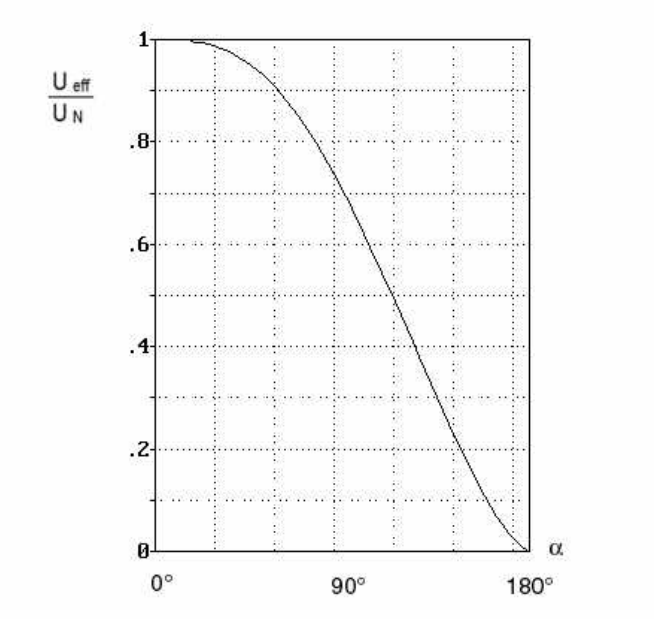
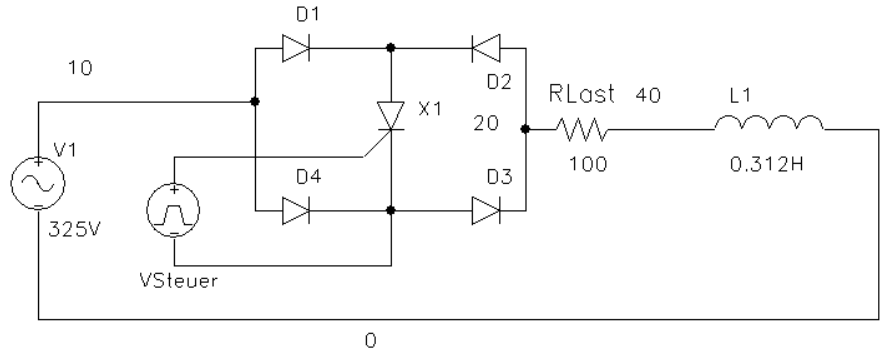


Bild 3. Steuerkennlinie eines einphasigen Anschnittstellers mit ohmscher Last

2.4.2 Phasenanschnitt mit ohmsch-induktiver Last

Eine Anschnittmodell mit rein ohmscher Last entspricht nicht der Realität, in der Antriebstechnik findet sich eine gemischte Belastung aus ohmschem Widerstand und Induktivität am häufigsten.
Durch den Phasenverschiebungswinkel φ eines RL-Gliedes verlängert sich der Stromfluß durch das Stromventil über den Nulldurchgang der Speisespannung hinaus. Zur Simulation wurde hier kein spannungsgesteuerter Schalter mehr verwendet, sondern dieser durch ein Thyristormodell erweitert.
Zur Verdeutlichung wurde die Induktivität so gewählt, daß sich ein Phasenwinkel von φ=45° ergibt :



Schaltbild 2. Schaltung zur Simulaton Phasenanschnitt mit ohmsch-induktiver Last

Für einen Steuerwinkel von 90° ergibt sich folgender Strom/Spannungsverlauf :

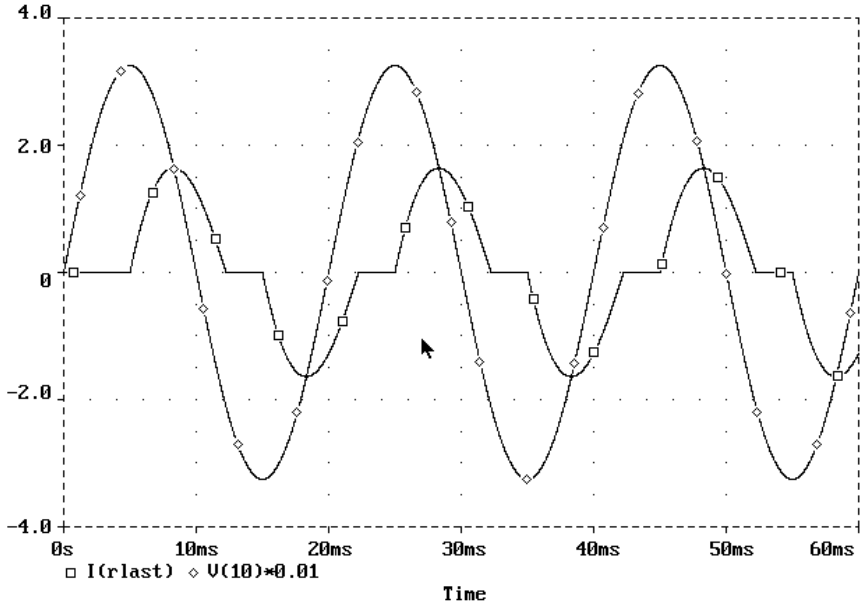


Bild 4. Phasenanschnitt mit α=90° bei ohmsch-induktiver Last

Bei einem Steuerwinkel gleich dem Phasenwinkel der Last findet bereits Vollaussteuerung statt, das heißt der Aussteuerungsbereich verringert sich um den Phasenwinkel der Last ($\varphi \leq \alpha \leq 180°$) .

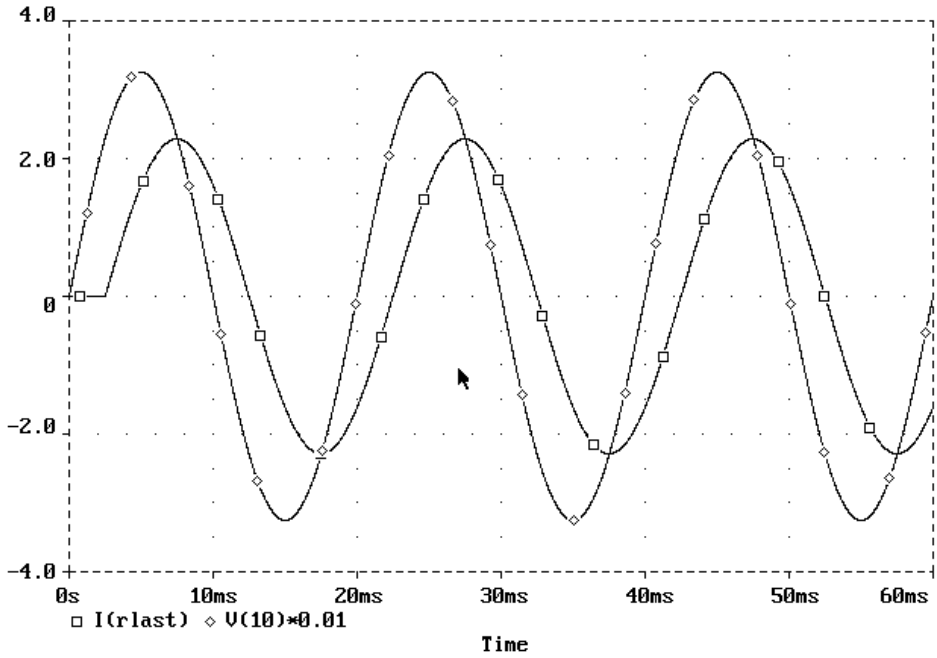


Bild 5. Phasenanschnitt mit $\alpha=\varphi$ bei ohmsch-induktiver Last

Im Nulldurchgang des Laststromes (hier mit Faktor 10^3 rot dargestellt), das heißt im Moment des Abschaltens des Thyristors, entsteht durch die Induktivität der Last am Thyristor eine hohe Induktionsspannung (grün) :

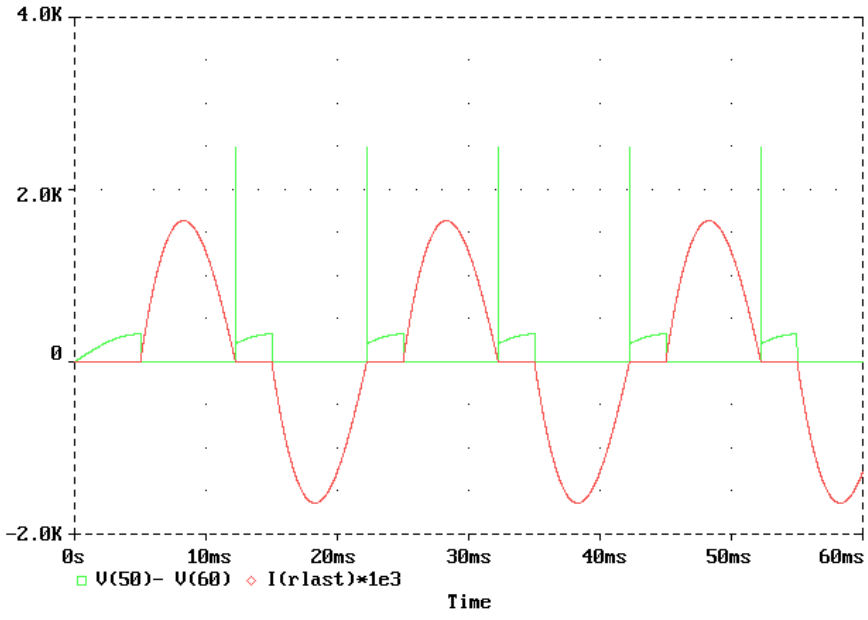


Bild 6. Spanung am Thyristor bei Phasenanschnitt mit ohmsch-induktiver Last

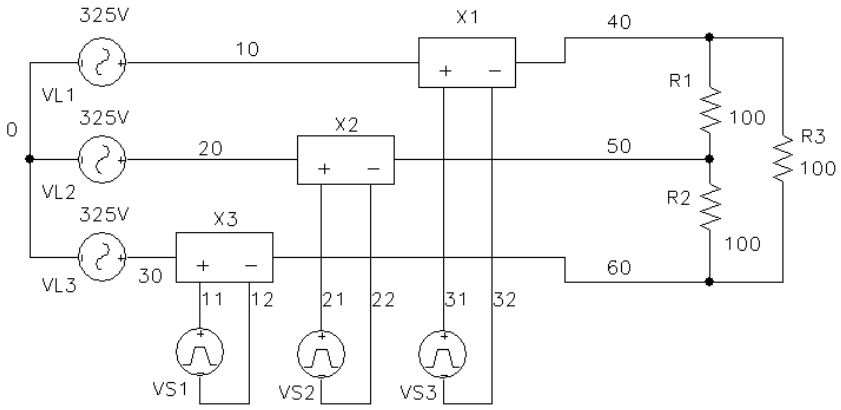
Diese Spannungsspitzen liegen im Kilovolt-Bereich mit sehr großen Steilheiten, so daß der kritische Parameter *du/dt* des Thyristors überschritten wird, falls keine Schutzmaßnahmen ergriffen werden.

2.5 Phasenanschnitt im Drehstromsystem

2.5.1 Simulationsumgebung

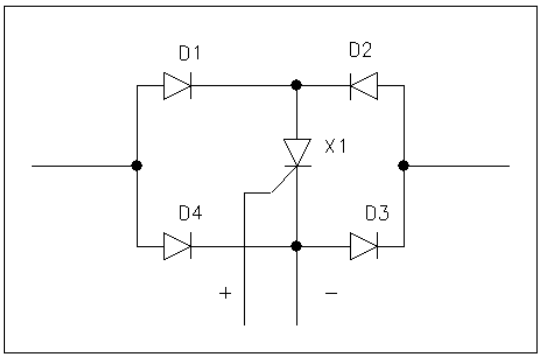
Im Drehstromsystem ist ein Phasenanschnitt um ein vielfaches komplexer als im einphasigen System. Zum Einsatz kommen soll ein symmetrisches System, d.h. alle 3 Phasen werden durch jeweils ein Stromventil gesteuert. Zusätzlich soll von einer symmetrischen, im Dreieck geschalteten Last, das heißt ohne Nulleiter, ausgegangen werden.

Zur Simulation dient dabei folgende Schaltung :



Schaltbild 3. Simulation Phasenanschnitt im Drehstromsystem

Die Wechselstromventile wurden mit einer Gleichrichterschaltung und einem Thyristiormodell realisiert.



Schaltbild 4. Unterschaltung Stromventil

Die 3 Spannungen V_{L1}, V_{L2} und V_{L3} erzeugen ein klassisches Drehstromsystem :

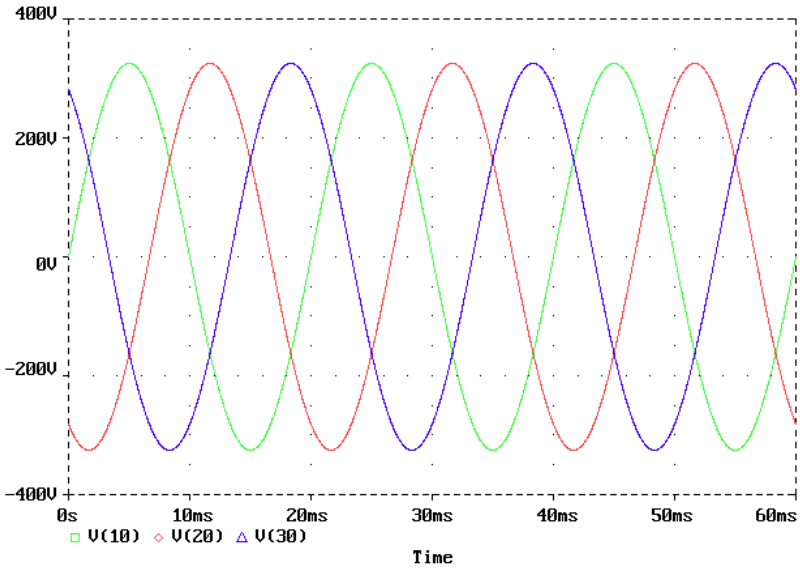


Bild 7. Speisespannungen V_{L1}-V_{L3} des Drehstromsystems

Aus diesen 3 Speisespannungen werden die drei Aussenleiterspannungen :

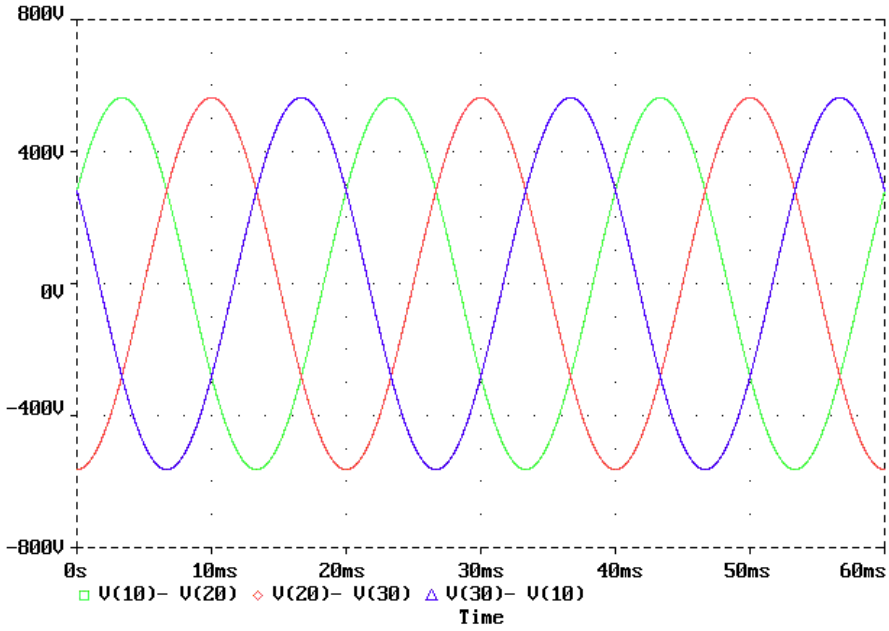


Bild 8. Aussenleiterspannungen des Drehstromsystems

2.5.2 Zündimpulsschema

Der Steuerwinkel für die Stromventile wird ab Nulldurchgang der jeweiligen Speisespannung (bezogen auf den Mittelleiter (Hinweis Bild) gemessen und die Zündimpulsdauer berechnet sich jeweils bis zum nächsten Nulldurchgang dieser Spannung.

In der Übersicht für alle 3 Stromventile ergebt sich ein vollständiges Zündschema. Hier beispielhaft für α=45°, α=90° und α=135° :

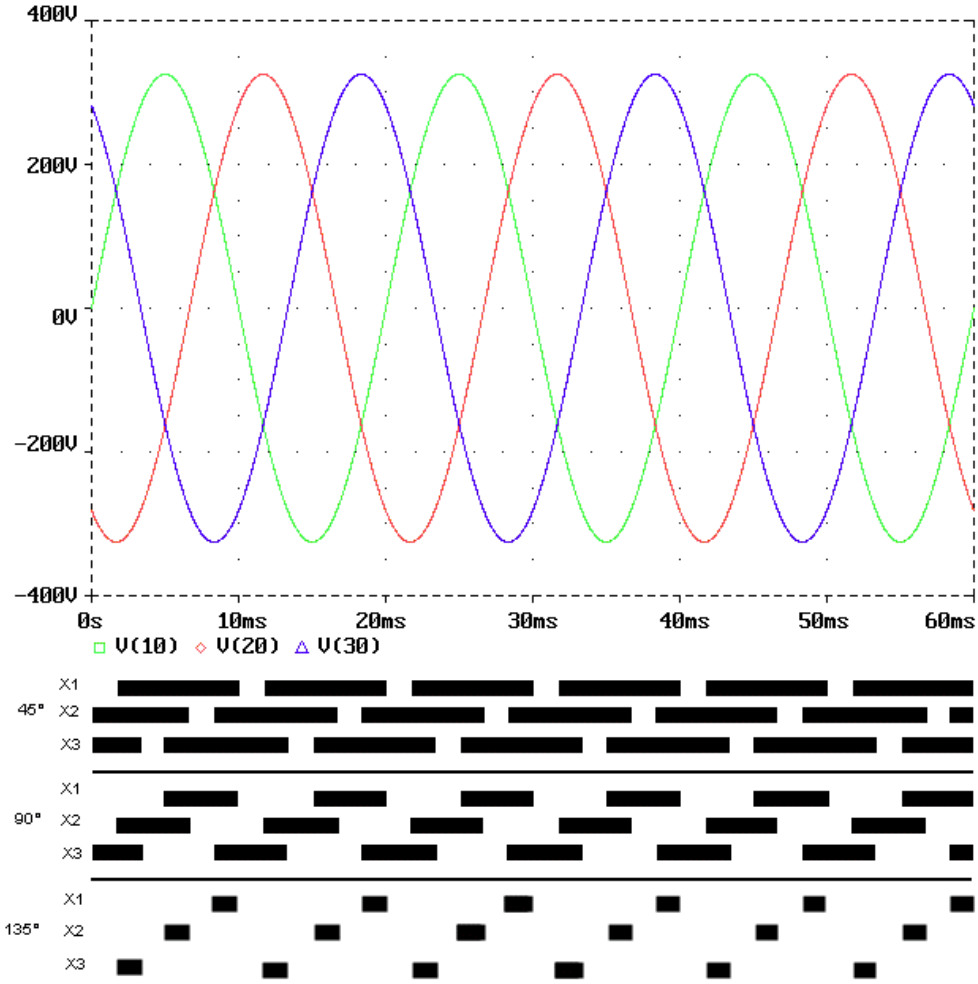


Bild 9. Vollständiges Zündschema für 3 verschiedene Steuerwinkel

2.5.3 Stromverlauf an der Last

Die Spannungen an der Last sind durch das Fehlen des Mittelleiters anders geformt als beim Wechselstrom-Phasenanschnitt. Ein Steuerwinkel von 45° hat folgenden Stromverlauf zur Folge :

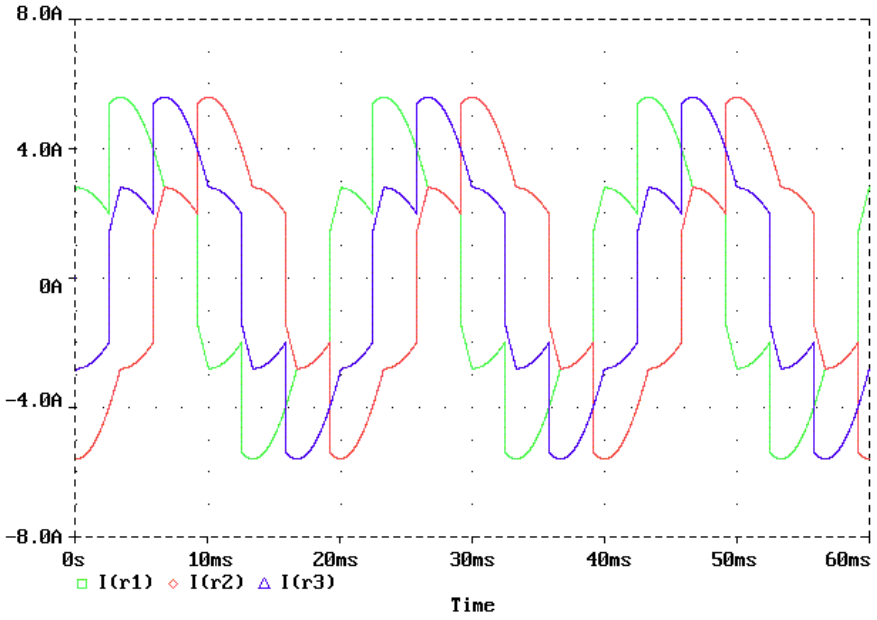


Bild 10. Stromverlauf bei Steuerwinkel 45°

Ein Steuerwinkel von 65° liefert folgenden Stromverlauf :

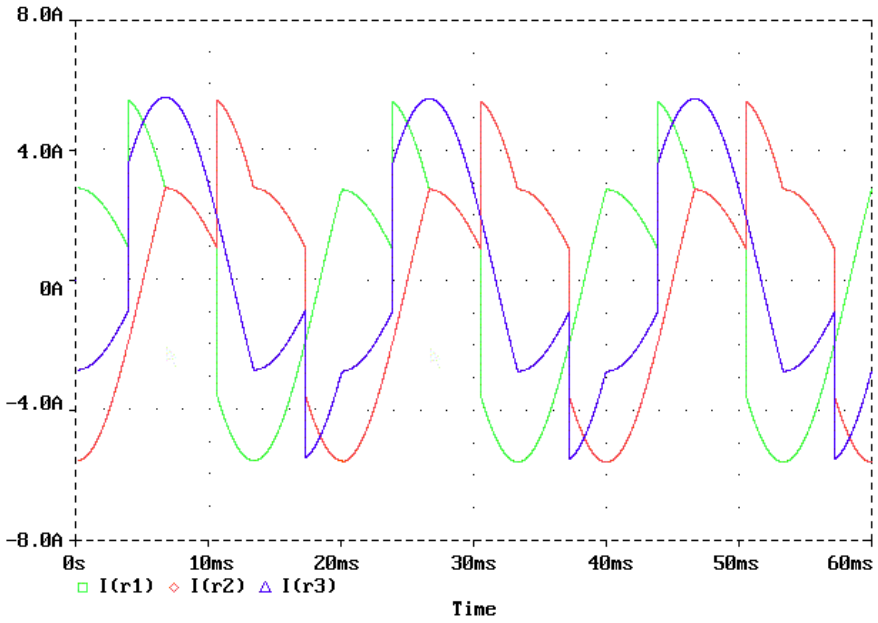


Bild 11. Stromverlauf bei Steuerwinkel 65°

Ab einem Steuerwinkel von 90° treten Lücken im Strom auf, da nur noch jeweils ein Ventil aufgesteuert wird :

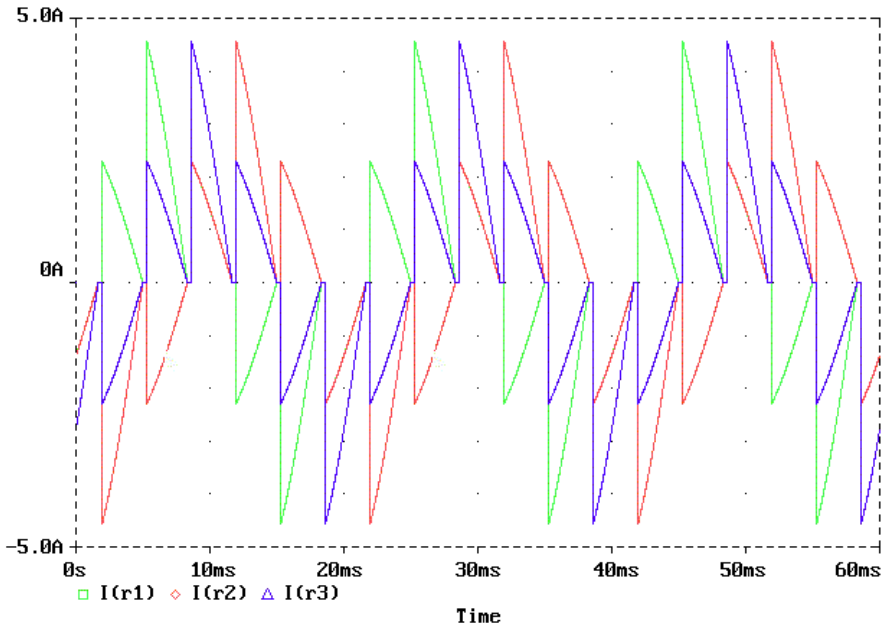


Bild 12. Stromverlauf bei Steuerwinkel 90°

Ab einem Steuerwinkel von 150° ist kein Stromfluß mehr möglich, der Strom durch die Last und damit die Leistung wird Null.

Es findet beim Einschalten ein Einschwingvorgang statt, da die Steuerimpulse der Reihe nach zugeschaltet werden. Der Laststrom beginnt zu fliessen, sobald zwei der drei Ventile aufgesteuert werden :

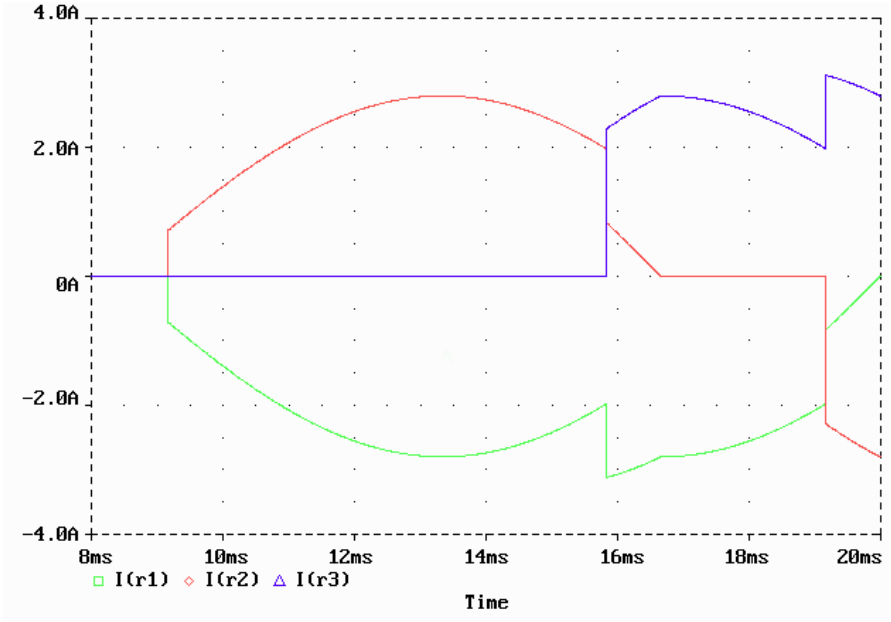


Bild 13. Laststrom während des Einschaltvorgangs mit α=45°

2.5.4 Unsymmetrien im Stromfluß

Die Simulationen eröffnen ebenfalls Unsymmetrien im Stromfluß in bestimmten Bereichen des Steuerwinkels. Um die Interpretation einfacher zu gestalten wurde die Last von einer Dreieck- zu einer Sternschaltung umgestaltet, um den Sternmittelpunkt (Knoten 70) zu Symmetriebetrachtungen heranziehen zu können.

Bis zu einem Steuerwinkel von 60° ist der Stromfluß symmetrisch, d.h. der Mittelwert der Mittelpunktspannung ist 0.

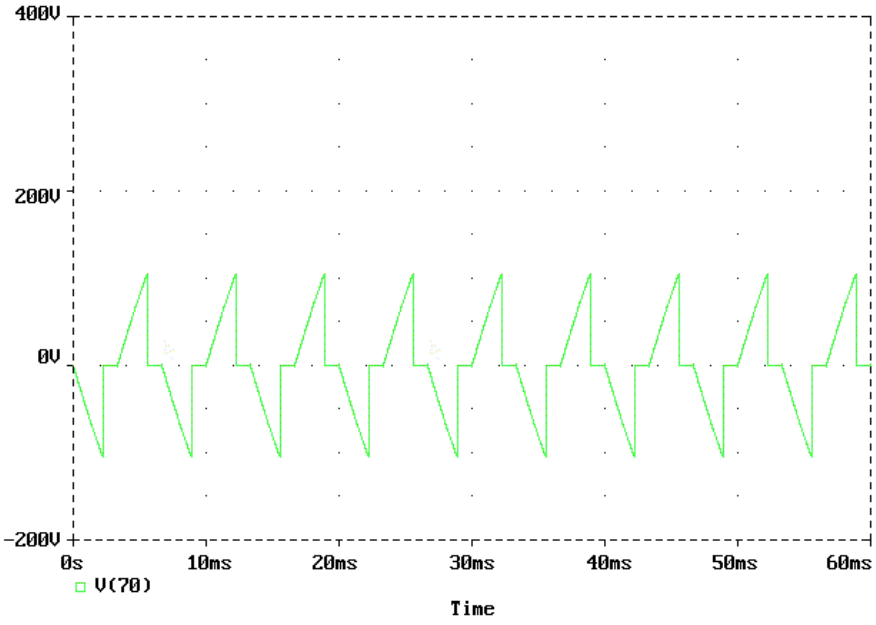


Bild 14. Spannung am Mittelpunkt der Last bei Steuerwinkel α=45°

Ab einem Steuerwinkel von 60° bis 90° wird das Drehstromsystem gestört, d.h. die Mittelpunktspannung ist ungleich 0 :

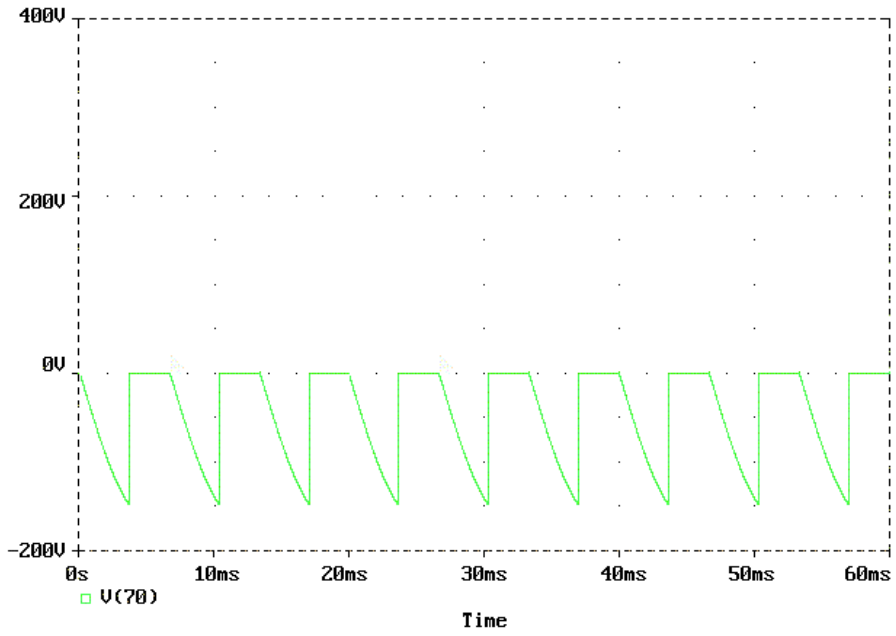


Bild 15. Mittelpunktspannung bei Steuerwinkel α=90°

Oberhalb von einem Steuerwinkel von 90° wird der Stromfluß wieder symmetrisch, d.h. das Drehstromsystem ist wieder ungestört :

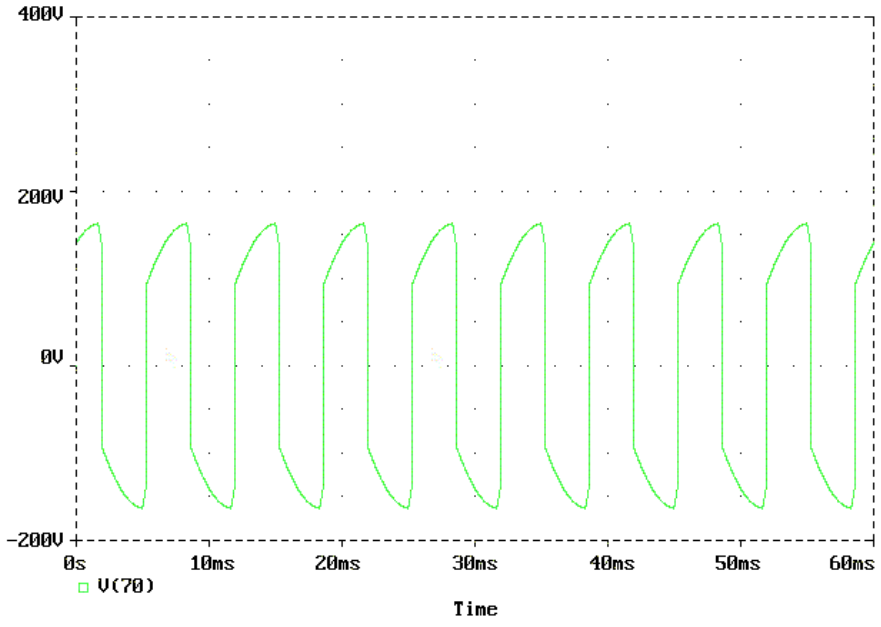


Bild 16. Mittelpunktspannung bei Steuerwinkel α=120°

Der Stromverlauf läßt sich abschnittsweise analytisch beschreiben. Hierfür sei auf [101, Seite 24ff] verwiesen.

2.6 EMV-Problematiken

Durch den Phasenanschnit entstehen nicht-sinusförmige Strom- und Spannungsverläufe, welche vielfältige Rückwirkungen erzeugen.

2.6.1 Entstehung von Steuerblindleistung

Bei Phasenanschnitt entsteht auch bei rein ohmscher Last eine Verschiebung zwischen Speisepannung und dem Strom durch die Last. Betrachtet man die erste Grundschwingung des Laststromes, so eilt diese der Speisespannung nach, es entsteht also eine induktive Grundschwingungsblindleistung, die sogenannte Steuerblindleistung.
So liefert PSPICE für einen Anschnittwinkel von 135° eine Verschiebung der ersten Grundschwingung des Laststromes von –60.25°.

FOURIER COMPONENTS OF TRANSIENT RESPONSE I(rlast)

DC COMPONENT = 1.148790E-03

HARMONIC NO	FREQUENCY (HZ)	FOURIER COMPONENT	NORMALIZED COMPONENT	PHASE (DEG)	NORMALIZED PHASE (DEG)
1	**5.000E+01**	**5.957E-01**	**1.000E+00**	**-6.025E+01**	**0.000E+00**
2	1.000E+02	2.298E-03	3.857E-03	1.800E+02	2.402E+02
3	1.500E+02	5.171E-01	8.682E-01	5.667E-04	6.025E+01

Bild 17. Ausschnitt aus der PSPICE-Logdatei

2.6.2 Netzrückwirkungen durch Phasenanschnitt

Durch den Phasenanschnitt wird bei einem Anschnittwinkel von 0°<α<180° ein nicht-sinusförmiger Strom durch die Last erzeugt, welcher nach Fourier in eine Summe aus sinusförmigen Strömen zerlegt werden kann [102, Seite 65ff].
Transformiert man bei einem Phasenanschnitt den Lastrom in den Frequenzbereich, so zeigt sich, daß neben der netzfrequenten Grundschwingung nur Oberschwingungen mit ungeraden Ordnungszahlen entstehen.

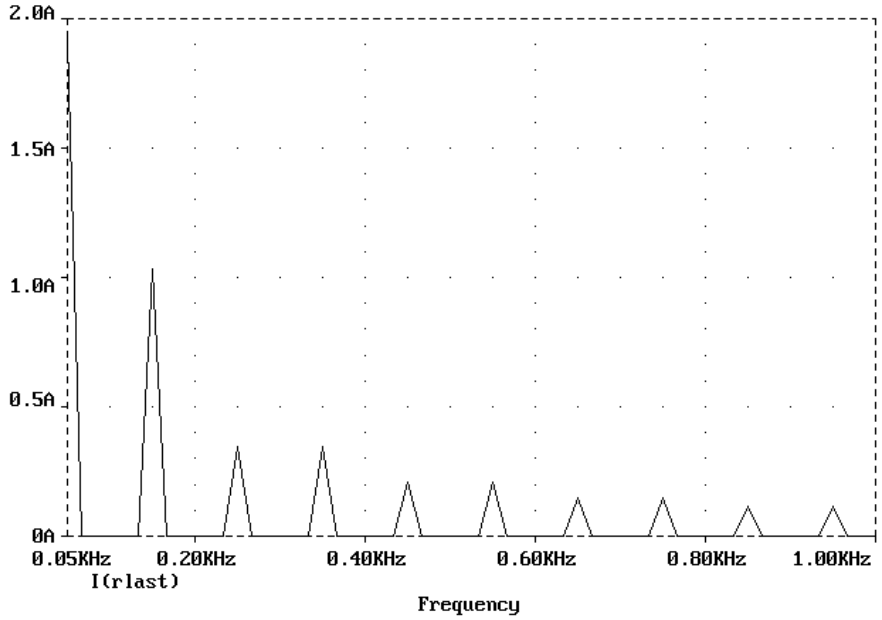


Bild 18. Fouriertransformation des Laststromes bei α=90 (Siehe Lastrom bei α=90°)

Beim gezeigten Betriebsfall halber Aussteuerung (α=90°) besitzt zum Beispiel die dritte Oberschwingung ein ausgeprägtes Maximum. So gibt es für jede Oberschwingung (Nummer i) bei bestimmten Anschnittwinkeln Höchstwerte, die durch den Faktor

$$C_{i\,\max} = \frac{I_{i\,\max}}{I_N}$$

Gleichung 3. Oberschwingungsfaktor

gekennzeichnet werden können.

Daraus entsteht eine Liste [102, Seite 59ff] mit den ungünstigsten Anschnittwinkeln :

Ordnungszahl i	Ungünst. Anschnittwinkel [°]	C_{imax}
3	90 (π/2)	0.3183
5	60 (π/3), 120 (2π/3)	0.1378
7	90 (π/2)	0.1061
9	72 (2π/5), 108 (3π/5)	0.0758
11	90 (π/2)	0.0637
13	77 (3π/7) ,26 (5π/)	0.0517
15	90 (π/2)	0.0455

Tabelle 2. Ungünstigste Anschnittwinkel für Oberschwingungen

2.6.3 EMV Gesetzgebung

Das EMV-G definiert die elektromagnetische Verträglichkeit als die Fähigkeit eines Gerätes oder eines Systems, in einer elektromagnetischen Umgebung zufriedenstellend zu funktionieren und andere Geräte nicht zu stören.
Die europaweit gültige Vorgabe für die EMV ist im EMVG (EMV-Gesetz) definiert und alle entsprechenden Vorgaben besitzen Rechtsgültigkeit.
Der Nachweis über die Einhaltung für die Störemission und die Störfestigkeit eines Systems erfolgt durch die Zertifizierung mit dem CE-Zeichen.
Die RegTP (Regulierungsbehörde für Telekommunikation und Post) listet alle aktuellen europäisch harmonisierten Normen, die für das bestimmte Gerät anzuwenden sind, auf.
Man unterscheidet zwischen den Fachgrundnormen (unterteilt in Wohn/Geschäfts- und Industriebereich) und den Produktenormen oder Produktfamlienenormen. Diese unterteilen sich wieder in Normen zur Störfestigkeit sowie zur Störemission.
Aus diesen umfangreichen Werken muß die zum jeweiligen Gerät passende Kombination ermittelt werden. Sind in den Produktenormen keine Definitionen enthalten, muß auf die jeweilige Fachgrundnorm zurückgegriffen werden.

In diesem Fall kommen folgende Regelwerke zur Störfestigkeit in Betracht :

Norm/Dokument	Titel
Fachgrundnormen :	
EN61000-4-2	„Prüfung der Störfestigkeit gegen elektrostatische Entladungen (ESD)“
EN61000-4-4	„Prüfung der Störfestigkeit gegen schnelle elektrische Transienten (Burst)“
EN61000-4-5	„Prüfung der Störfestigkeit gegen Stoßspannungen (Surge)“
EN61000-4-8	„Prüfung der Störfestigkeit gegen Magnetfelder mit Netzfrequenz“
EN61000-4-11	„Prüfung der Störfestigkeit gegen Spannungseinbrüche, Kurzzeitunterbrechungen und Spannungsschwankungen“
Produktfamilienormen :	
EN55014-2	„Anforderungen an die Störfestigkeit von Haushaltsgeräten, Elektrowerkzeugen und ähnlichen Elektrogeräten"

Tabelle 3. EMV-Normen zur Störfestigkeit

Zur Beurteilung der Störemissionen kommen folgende Normen in Betracht :

Norm/Dokument	Titel
Fachgrundnormen :	
EN61000-3-2	„Grenzwerte für Oberschwingungsströme aller Geräte mit einem Geräteeingangsstrom≤16A pro Leiter“
EN61000-3-3	„Grenzwerte für Spannungsschwankungen und Flicker in Niederspannungsnetzen für alle Geräte mit einem Eingangsstrom ≤16A pro Leiter“
EN61000-3-4	„Grenzwerte für Oberschwingungsströme für alle Geräte mit einem Geräteeingangsstrom >16A und ≤75A pro Leiter“
Produktfamilienormen :	
EN55014-1	„Grenzwerte und Meßmethoden für Funkstörungen von Haushaltsgeräten, Elektrowerkzeugen und ähnlichen Elektrogeräten"

Tabelle 4. EMV-Normen zur Störemission

2.6.3.1 Störemisssion

Es definiert die EN 55014-1 („Grenzwerte und Meßverfahren für Funkstörungen von Haushaltsgeräten, Elektrowerkzeugen und ähnlichen Elektrogeräten“), deren Anwendungsbereich sich auf Geräte mit motorischer Hauptfunkton und auf Halbleiter-Stellglieder bezieht, folgende Grenzwerte für Störungen im Frequenzbereich 148.5 kHz bis 30 MHz :

Frequenzbereich [MHz]	Spannung am Netzanschluss Quasipeak [db µV]	Spannung am Netzanschluss Mittelwert [db µV]	Spannung am Verbraucheranschluss Quasipeak [db µV]	Spannung am Verbraucheranschluss Mittelwert [db µV]
0.15-0.5	Mit dem log. Der Frequenz linear fallend von 66 bis 56	Mit dem log. Der Frequenz linear fallend von 59 bis 46	80	70
0.5-5	56	46	74	64
5-30	60	50	74	64

Tabelle 5. Grenzewerte Funkstörspannungen nach EN 55014-2

Im Frequenzbereich 30-300 MHz unterliegen Anschnittsteller wie der hier behandelte keinen Grenzwerten.

Die EN 61000-3-2 („Grenzwerte für Oberschwingungsströme aller Geräte mit einem Geräteeingangsstrom≤16A pro Leiter“) legt die absoluten Obergrenzen für die vom Gerät erzeugten Stromoberschwingungen fest :

Oberschwingungsordnung n	Zul. Höchstwert des Stromes [A]
2	1.08
3	2.3
4	0.43
5	1.14
6	0.3
7	0.77
$8 \leq n \leq 40$ (gerade Ordnung)	0.15 * 15/n
9	0.4
11	0.33
13	0.21
$15 \leq n \leq 39$ (ungerade Ordnung)	0.23 * 8/n

Tabelle 6. Grenzwerte der EN 61000-3-2

Diese Zuordnung gehört zur Geräteklasse A („symmetrisches, dreiphasiges Betriebsmittel“), der für diesen Anwendungsfall geltenden Auswahl. Dir Prüfung kann sowohl im Meßbetrieb mit ohmscher Last, als auch mit einem rechnerischen Ansatz, also z.B. einer Simulation, bestimmt werden.

Die EN 61000-3-3 („Grenzwerte für Spannungsschwankungen und Flicker in Niederspannungsnetzen für alle Geräte mit einem Eingangsstrom ≤16A pro Leiter“) legt Grenzwerte für Spannungsänderungen fest, die durch ein Gerät im Netz erzeugt werden. Es gelten im folgenden :

Typ	Grenzwert	Bedeutung
d_{max}	≤ 4%	Größte relative Spannungsänderung
d(t)	≤ 3% für max. 200 ms	Relativer Spannungsänderungsverlauf
d_c	≤ 3%	relative Konstante Spannungsabschächung

Tabelle 7. Grenzwerte der EN 61000-3-3

Zur Verdeutlichung liefert die Vorlage folgende Grafik :

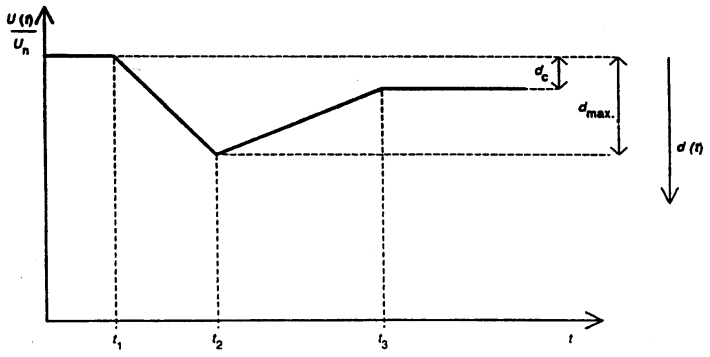


Bild xy. Verdeutlichung der Grenzwerte der EN 61000-3-3

2.6.3.2 Störfestigkeit

Die EN 55014-2 („Anforderungen an die Störfestigkeit von Haushaltsgeräten, Elektrowerkzeugen und ähnliche Elektrogeräte“) legt eine ganze Reihe von Prüfungen fest. Die zu Prüfenden Geräte werden unterteilt in die Kategorien I-III, in denen sich die Prüfkriterien unterscheiden. Im einzelnen sind dies :

Kategorie I

Betriebsmittel, die keine elektronischen Steuerungen enthalten (z.B. motorische Geräte, Spielzeuge, Elektrowerkzeuge)

Kategorie II

Netzbetrtiebene motorische geräte, Elektrowerkzeuge, Elektrowärmegeräte und ähnliche elektrische Betriebsmittel, die keine elektronischen Steuerungen mit einer Takt- oder Oszillatorfrequenz oberhalb 15 MHz enthalten.

Kategorie III

Batteriebetriebene Geräte, die bei der üblichen Benutzung nicht mit dem Stromversorgungsnetz verbunden sind und keine elektronischen Steuerungen mit einer Takt- oder Oszillatorfrequenz oberhalb 15 MHz enthalten.

Nach Kategorie II („netzbetriebene, motorische Geräte [...], die keine elektronischen Steuerungen mit einer Takt – oder Oszillatorfrequenz oberhalb von 15 MHz“) ist für den Anschnittsteller also eine Reihe von Prüfungen zu absolvieren :

Entladung statischer Elektrizität mit Bewertungskriterium B

- 8 kV Luftentladung
- 4 kV Kontaktentladung

Schnelle Transienten mit Bewertungskriterium B

- Eingekoppelt auf Signal- und Steuerleitungen 0.5 kV Spitze 5/50ns T_r/T_h mit 5 Khz
- Eingekoppelt auf Wechselstrom-Netzein- und Ausgänge 1 kV Spitze 5/50ns T_r/T_h mit 5 Khz

Eingespeiste Ströme bis 230 MHz mit Bewertungskriterium A

- Eingespeiste Ströme auf Signal- und Steuerleitungen $1V_{eff}$ unmoduliert, 150 Ω Quellipedanz
- Eingespeiste Ströme auf Wechselstrom-Netzein- und Ausgänge $3V_{eff}$ unmoduliert, 150 Ω Quellipedanz

Stoßspannungen (langsame, energiereiche Impulse) mit Bewertungskriterium B

- Eingekoppelt auf Wechselstrom-Netzeingänge 1.2/50 (8/20) Tr/Th µs mit 2kV Spitze (Phase-Nulleiter-Schutzleiter) und 1kV Spitze (Phase-Phase)

Spannungseinbrüche und Unterbrechungen mit Bewertungskriterium C

Für die Prüfung von Spannungseinbrüchen gelten die folgenden Parameter :

Umgebungsphänomene	Prüfstörgrößen in % von U_N	Dauer (in Perioden der Netzfrequenz)
Unterbrechungen	0	0.5
Einbrüche in % von U_N		
60	40	10
30	70	50

Tabelle 8. Angaben zur Prüfung auf Spannungseinbrüche nach EN 55014-2

Die Bewertungskriterien A bis C erlauben eine Unterscheidung verschiedener Betriebsqualitäten des Prüflings. Im folgenden gelten :

Bewertungskriterium A

Das Gerät muß während der Prüfung ordnungsgemäß arbeiten. Eine Beeinträchtigung des Betriebsverhaltens unterhalb einer vom Hersteller beschriebenen minimalen Betriebsqualität ist nicht zulässig.

Bewertungskriterium B

Das Gerät muß nach der Prüfung ordnungsgemäß arbeiten. Eine Beeinträchtigung des Betriebsverhaltens unterhalb einer vom Hersteller beschriebenen minimalen Betriebsqualität aber ist während der Prüfung zulässig, darf jedoch keinen Verlust gespeicherter Daten oder die Änderung von Einstellungen beinhalten.

Bewertungskriterium C

Während der Prüfung ist ein zeitweiliger Funktionsausfall erlaubt. Die Funktion muß jedoch nach der Prüfung selbstständig, oder durch vom Hersteller in der Bedienungsanleitung beschriebene Maßnahmen wiederherstellbar sein.

3. Schaltungsrealisierung

3.1 Anforderungen an die zu realisierende Schaltung

Dieses Gerät soll die folgenden allgemeinen Eigenschaften aufweisen :

- Auslegung für einphasen- und dreiphasen-Systeme
- Spannungssteuerung durch symmetrischen Anschnitt aller 3 Phasen
- Berücksichtigung der EMV-Normen DIN EN-55014-1 und –2
- Auslegung des Steuergerätes für den Anschluß beliebiger Leistungsmodule

Weiter sollen folgende Details im Aufbau Berücksichtigung finden :

- Parametrierbarkeit des Gerätes über Tasten und ein Klartextdisplay
- Funktionen zur Überwindung von Losbrechmomenten
- Steuerspannungseingang zur Umschaltung zwischen Hoch- und Auslaufbetrieb
- Steuerspannungseingang zur Steuerung von Not-Aus-Funktionen
- Steuerspannungsausgang zur Ansteuerung eines Überbrückungsschützes für die Leistungsteile
- Steuerspannungsausgang zur Ansteuerung eines Motorschützes
- Optional die Erkennung von Netzfehlern (Unter/Überspannung) und der Netzfrequenz

Einstellmöglichkeiten

Für den Benutzer des Gerätes sollen folgende Parameter über ein Menü einstellbar sein :

- Freigabe einer Kickstartfunktion
- Dauer des Kickstarts (0-2 Sekunden, Schrittweite 0.1s)
- Höhe des Kickstarts (0-100 %)
- Dauer der Hochlaufzeit (0-20s, Schrittweite 1s)
- Startwert des Hochlaufs (0-100%)
- Endwert des Hochlaufs (0-100%)
- Dauer des Auslaufes (0-200s, Schrittweite 1s)
- Überbrückung der Leistungsteile nach Ende des Hochlaufes (Ja/Nein)
- Ein- und Ausschalten eines Motorschützes (Ja/Nein)

Als Optionen sollen zur Verfügung stehen :

- Detektion von Überspannung (Ja/Nein)
- Detektion von Unterspannung (Ja/Nein)
- Schwellenwerte zur Über/Unterspannungserkennung (0 - ±20% der Nennspannung)
- Verzögerungszeit für Wiedereinschalten nach Fehler (0-100s, Schrittweite 1s oder manueller Reset)
- Ansprechzeit bei Vorliegen eines Fehlers (0-20s, Schrittweite 1s)
- Schnellstart, d.h. kein Sanftanlauf (Ja/Nein)

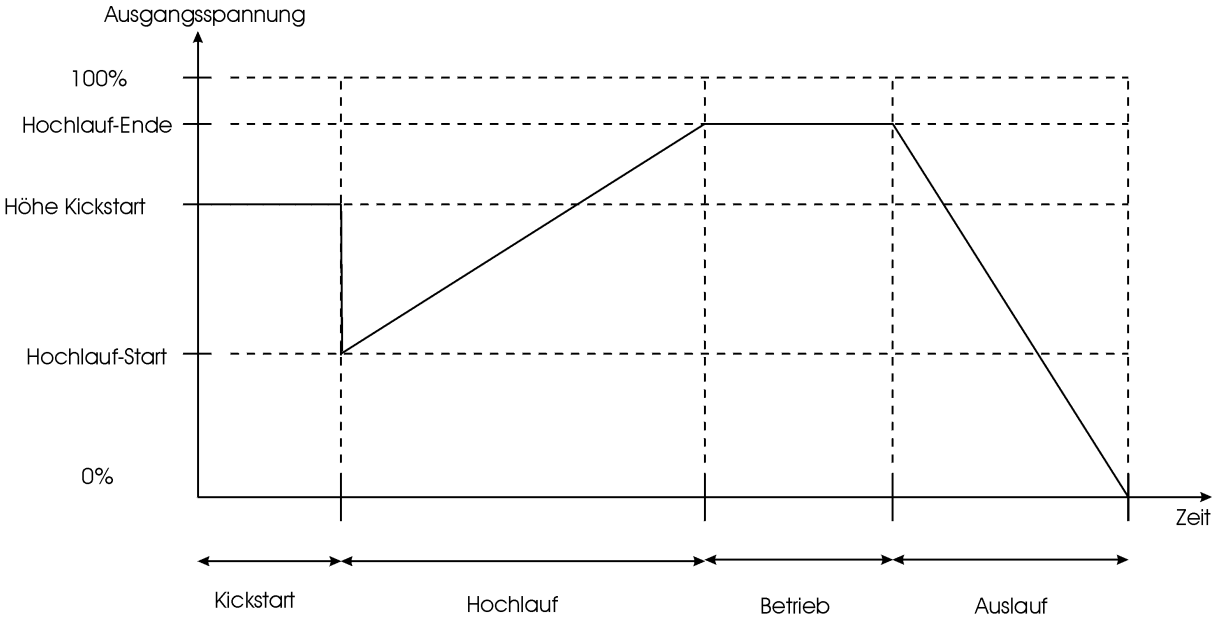


Bild 19. Ausgangsspannungsverlauf und einstellbare Parameter

3.2 Systemstruktur

Das gesamte System, das heißt das Modul mit Leistungsteilen und der weiteren Peripherie läßt sich auf einzelne Funktionsblöcke zurückführen. Das folgende Blockschaltbild gibt die Struktur wieder (die gestrichelte Umrandung verdeutlicht den Umfang des Steuermoduls) :

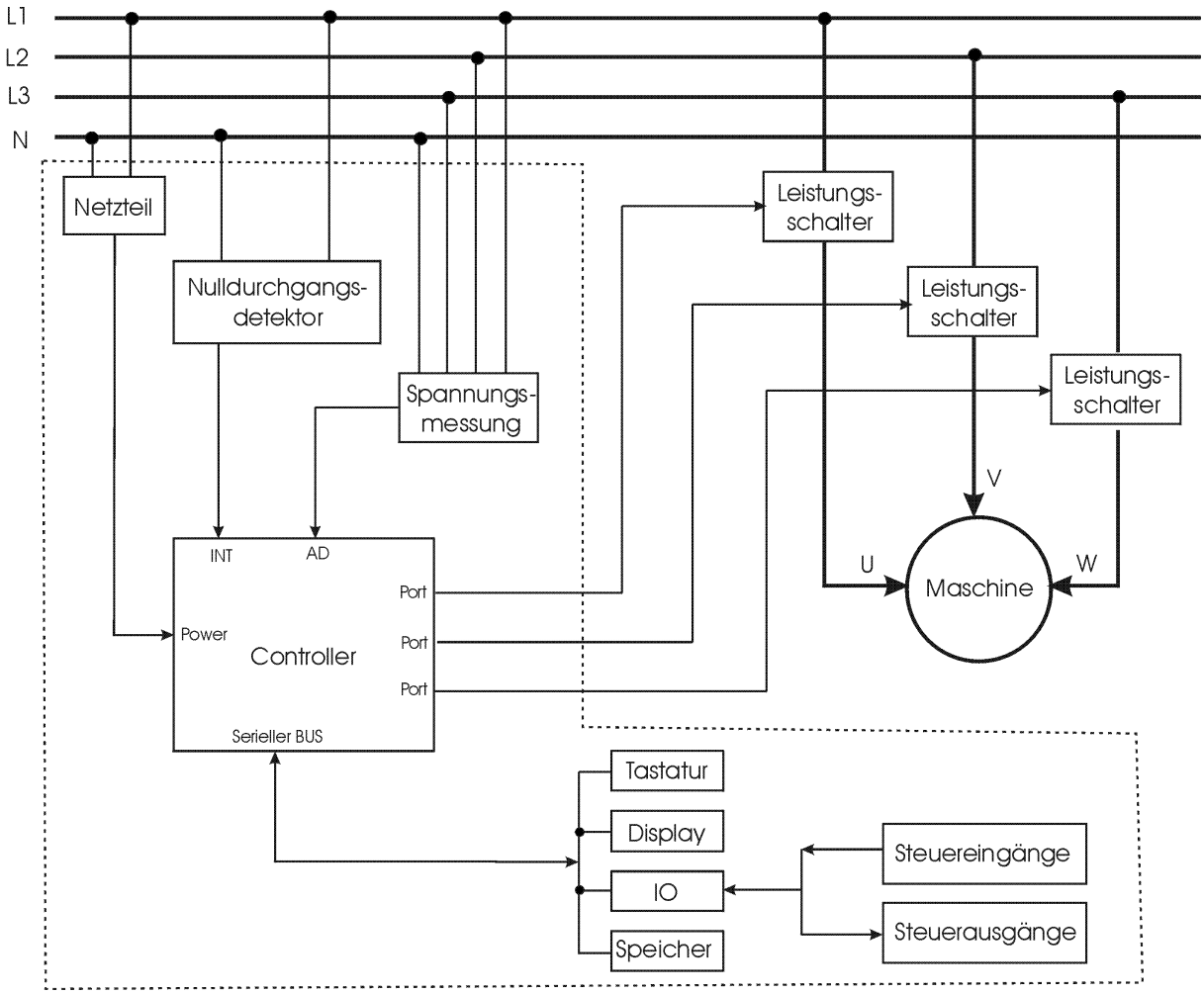


Bild 20. Blockschaltbild des Systems

Die Spannungsversorgung des Systems wird über die Phase L_1 und N entnommen und über ein Netzteil die erforderlichen Gleichspannungen erzeugt. Der Nulldurchgang der Phase L1 wird detektiert und löst jeweils einen Interrupt im Controller aus, so daß die über IO-Ports direkt angesteuerten Leistungsschalter mit dem Netzverlauf synchronisiert werden können. Diese sind symmetrisch angeordnet, das heißt alle 3 Phasen werden geschaltet.

Zusätzlich können über einen Meßumformer und den Analog-Digitalwander des Controllers die Netzspannungen bezüglich Frequenz sowie Über- und Unterspannung ausgewertet werden. Die Netzfrequenz ist insbesondere zur Berechnung der Nulldurchgänge der Phasen L2 und L3 wichtig.
Alle weiteren peripheren Ein- und Ausgaben für die Bedienungselemente (Tastatur und Display), der dauerhaften Speicherung von Konfigurationsdaten und der Realisierung von Steuerein- und Ausgängen werden über ein serielles Bussystem (Hier I^2C) abgewickelt. Der Controller beinhaltet ein komplettes System von Prozessor, Ram, Rom und IO.

3.3 Entwicklung der Schaltungsteile

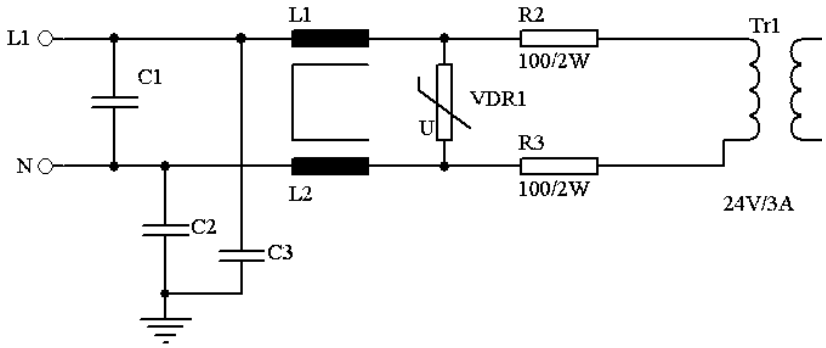
Soweit die einzelnen Schaltungsteile mit Simulationen unterstützt werden, sind die Schaltpläne der Windows-Version von Pspice entnommen. Alle anderen wurden mit Protel Adv. Schematics erstellt. Der vollständige Schaltplan mit Layouts und Bestückungslisten wird später im Anhang aufgeführt sein.

3.3.1 Das Netzteil

Die Netzspannung von L_1 zum Mittelleiter wird über einen handelsüblichen Transformator auf 24 herunter transformiert und mit einem Gleichrichter und linearen Spannungsreglern auf die erforderlichen Werte geregelt.

3.3.1.1 Primäre Schutzgliede

Um das Modul vor von der Netzseite einlaufenden Störungen zu schützen, ist folgende Schaltunge vorgesehen :

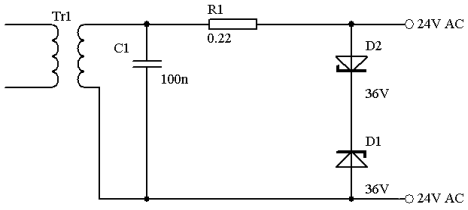


Schaltbild 5. Primärseitige Schutzschaltungen des Netzteils

C_1, C_2 und C_3, sowie die Induktivitäten L_1 und L_2 sind die Bestandteile eines fertigen Filtermoduls der Firma Schaffner (2*10mH, 15nF (X), 2.2nF (Y)), welches einlaufende Störungen dämpfen soll.
R_2 und R_3 begrenzen den Strom durch die Primärwicklung des Transformators bei länger anhaltenden Überspannungen. VDR_1 wird bei Überspannung leitend.

3.3.1.2 Sekundäre Schutzglieder

Sekundär auftretende Störspannungen werden durch folgende Schutzmaßnahmen bedämpft :

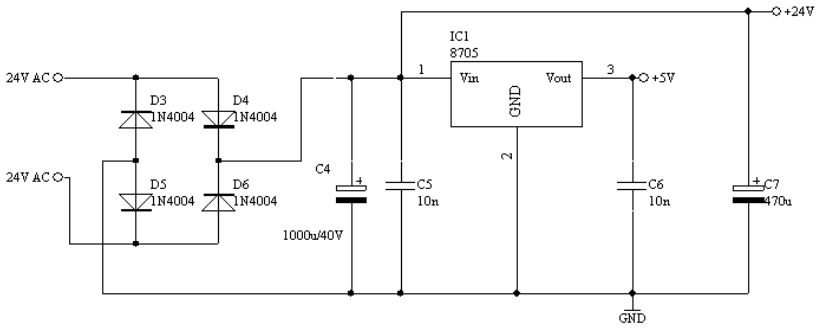


Schaltbild 6. Sekundärseitige Schutzschaltungen des Netzteils

C_3 bedämpft auch hier symmetrische, hochfrequente Störungen, R_5 begrenzt den Laststrom, falls das antiparallel geschaltete Z-Diodenpaar D_1/D_2 auftretende Spannungsspitzen kurzschliesst.

3.3.1.3 Gleichspannungserzeugung

Aus den 24V Wechselspannung generiert eine Kombination aus Brückengleichrichter und Siebelko eine Gleichspannung von ca. 30V.

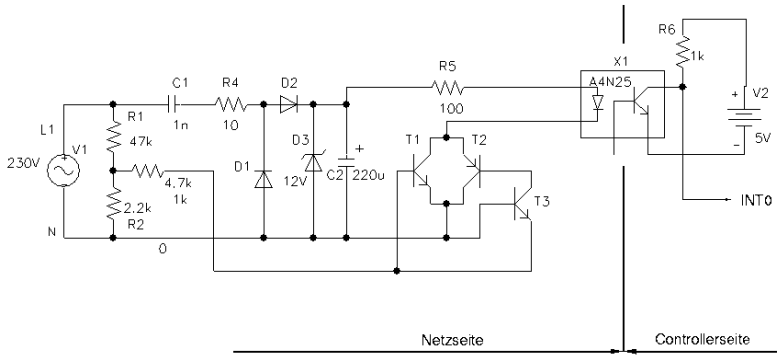


Schaltbild 7. Gleichspannungserzeugung des Netzteils

Hieraus gewinnt der Festpannungsregler IC_1 Dämpfungskondensatoren C_5/C_6, +5V für die Logik. Die unstabilisierte Gleichspannung von +30 V steht für die Zündimpulserzeugung zur Verfügung.

3.3.2 Nulldurchgangsdetektion

Um die Stromventile synchron zur Netzspannung führen zu können [siehe *Zündschema*], ist es notwendig, die Nulldurchgänge der Netzspannung detektieren zu können. Dies soll in einer Hardware realisiert werden.
Prinzipiell wäre dies auch über eine digitale Verarbeitung der im Controller sowieso schon vorliegenden Spannungswerte möglich. Die Hardwarelösung besitzt aber den Vorteil, über die Auslösung eines externen Interrupts bei Detektion eines Nulldurchgangs den Controller zu entlasten.



Schaltbild 8. Nulldurchgangsdetektor

Die Schaltung detektiert den Nulldurchgang von L_1 in Bezug zum Mittelleiter, also der Sternspannung.
Zunächst erzeugt C_1 mit R_4 eine Phasenverschiebung von fast 90° zwischen Strom und Spannung, so das an R_4 nur eine Wirkleistung von ca. 30mW abfällt. Die Kombination aus den Dioden D_1 bis D_3 erzeugt dann an C_2 eine Hilfs-Gleichspannung von etwa 12 V. Aus dieser wird die LED des Optokopplers über R_5 gespeist, so daß auf der Sekundärseite eine galvanische Trennung gewährleistet ist.
R_1 und R_2 teilen die Netzspannung auf etwa 20 Volt herunter, R_3 begrenzt den von diesem Spannungsteiler entnommenen Strom. Ist die Spannung an R_3 positiv (positive Halbwelle der Netzspannung), so leitet T_1, und es fliesst Strom durch die LED des Optokopplers.
Ist die Spannung an R_3 negativ (während der negativen Halbwelle der Netzspannung), so leitet T_2, es fließt ebenfalls Strom durch die LED. Nur im Nulldurchgang ist keiner der Transistoren leitend, zu diesem Zeitpunkt fließt kein Strom durch die LED.

Aus dieser Information wird über den Spannungsabfall über R_6 ein Spannungsimpuls am INT0-Pin des Controllers generiert:

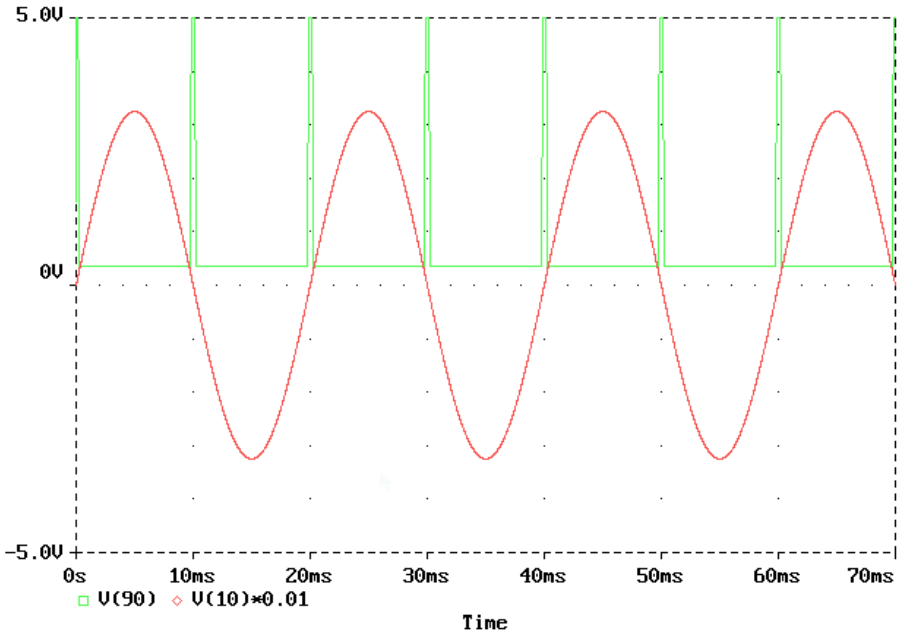


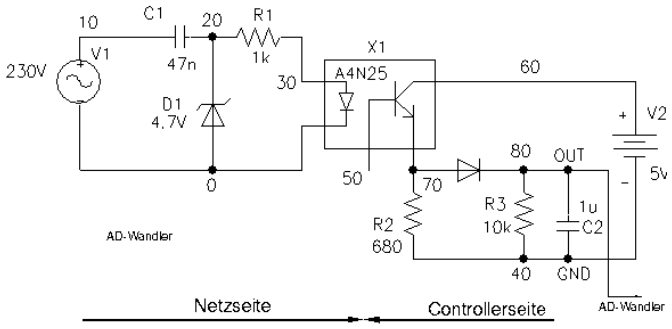
Bild 21. Simulation des Nulldurchgangsdetektors

Der erzeugte Spannungsimpuls generiert einen externen Interrupt über die INT0-Leitung des Controllers [301]. Aus dieser Information können dann die Nulldurchgänge der zwei anderen Spannungen in einer Interrupt-serviceroutine errechnet werden (vereinfachtes Listing der Simulation siehe *Anhang*).

3.3.3 Spannungsmessung

Der Controller kann über seinen AD-Wandler [301] ebenfalls zur Überwachung der Netzspannungen eingesetzt werden. Daraus kann die Qualität der Versorgung, das heißt ein Vorliegen von Über- oder Unterspannung oder eines Drehrichtungsfehlers, beurteilt werden, und zusätzliche Schutzmaßnahmen für die angeschlossenen Maschinen wären nicht mehr notwendig, sondern könnten direkt von diesem Modul eingeleitet werden. Gemessen werden die Sternspannungen.
Das Modul muß dabei vor zu hohen Spannungen geschützt werden, also ist eine galvanische Trennung notwendig. Dies soll mit Optokopplern realisiert werden. Wichtigstes Kriterium ist hierbei die Linearität des Strom-übertragungsverhältnisses CTR, das heißt des Quotienten aus Ausgangs- und Eingangsstrom [106].

Hierzu dient die folgende Schaltung :



Schaltbild 9. Netzspannungsmessung mit galvanischer Trennung.

C_1 sorgt für eine Phasenverschiebung zwischen Spannung und Strom von nahezu 90°, so daß auf der Primärseite dieser Schaltung kaum Verlustleistung anfällt. D_1 begrenzt die Spannung an R_1 und der LED des Optokopplers auf ungefährliche Werte.
Durch dieses Vorgehen ist zwar keine phasenrichtige Spannungsmessung möglich, diese ist aber nur wichtig im Bezug zu den den jeweils anderen Phasen. Außerdem reicht der lineare Bereich der Messung nur so weit herab, wie Strom durch die LED des Opptokopples fließt. Ebenso gerät man bei zu hoher Eingangsspannung in den Bereich der Sättigung.
Diese Schaltungsvariante eignet sich also nur zur Beurteilung eines relelativ schmalen Bereches der Netzspannung. Dies ist zur Beurteilung von Unter- oder Überspannung im Bereich von ±20% der Nennspannung oder der Gewinnung einer Drehrichtungsinformation ausreichend.
Auf der Sekundärseite integrieren D_1, R_3 und C_2 die Spannung um, Störimpulse aus dem Netz zu dämpfen.
Die Simulation dieser Schaltung liefert folgende Ergebnisse (Listing der Simulation siehe *Anhang*) :

Für Eingangsspannung=Nennspannung (230 V) liefert die Schaltung einen maximalen Ausgangspegel von 1.5 Volt :

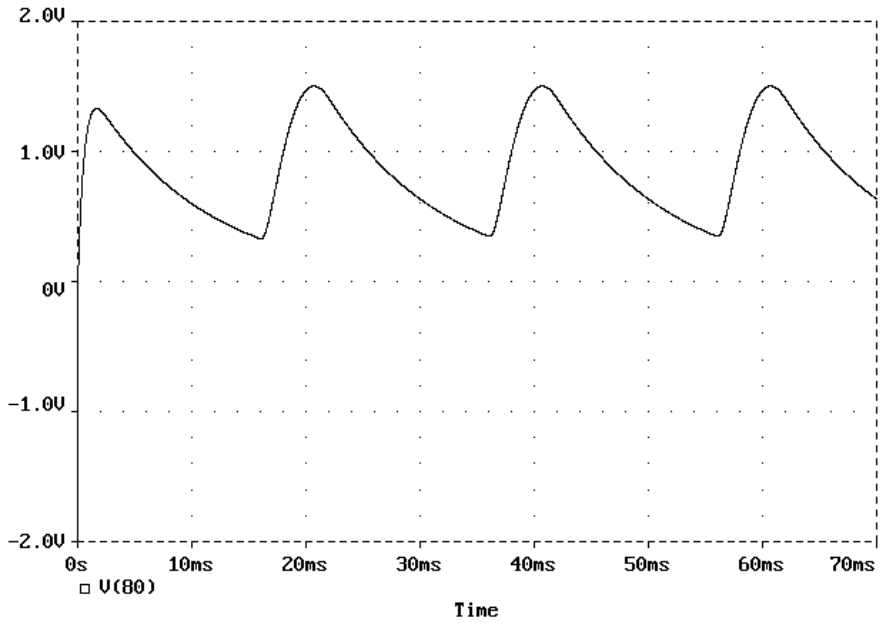


Bild 22. Spannungsmessung bei Nennspannung

Für Eingangsspannung=Nennspannung-20% liefert die Schaltung einen maximalen Ausgangspegel von 1.2 Volt :

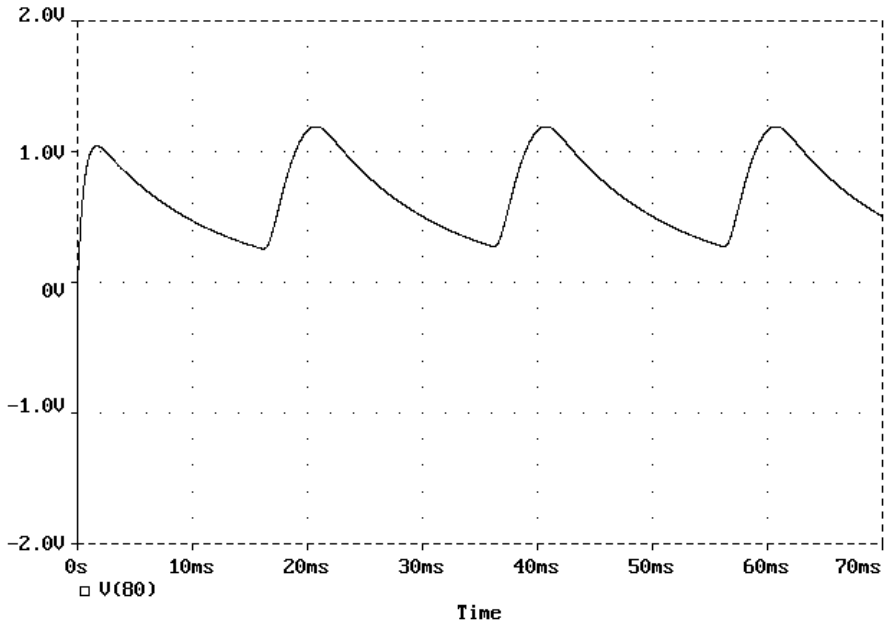


Bild 23. Spannungsmessung bei Nennspannung-20%

Für Eingangsspannung=Nennspannung+20% liefert die Schaltung einen maximalen Ausgangspegel von 1.8 Volt :

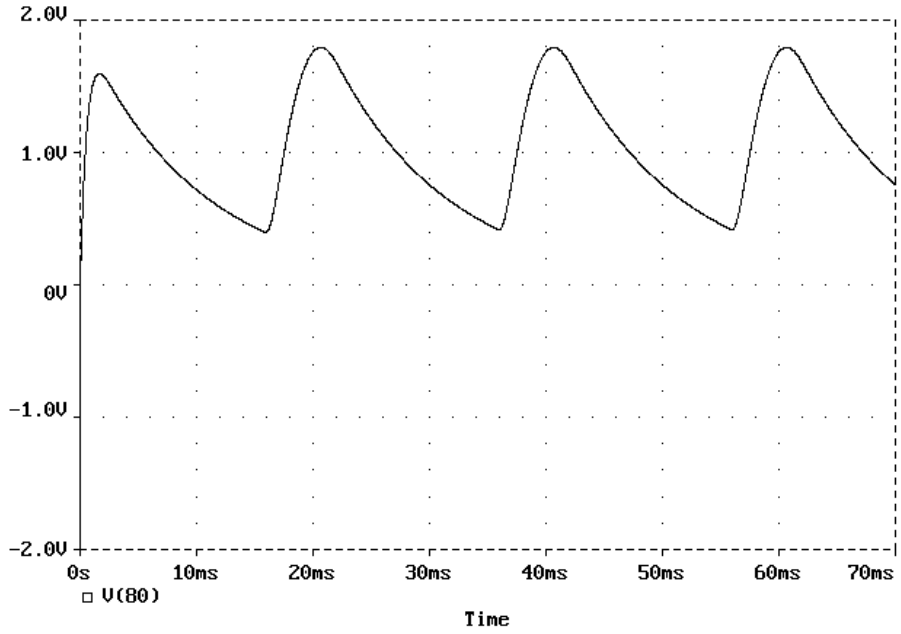


Bild 24. Spannungsmessung bei Nennspannung+20%

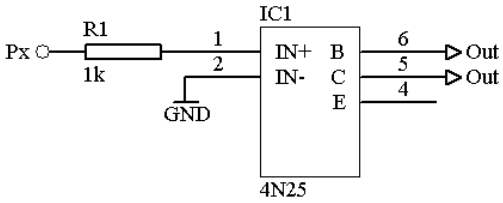
Praktische Messungen haben eine große Exemplarstreuung ergeben, das heißt, die im Datenblatt zum 4N25 angegebenen CTR-Werte sind nicht zur Kalibrierung geeignet. Dies muß also später vor der Inbetriebnahme in einer Software-Routine geschehen und das Stromübertragungsverhältniss im verwendeten Bereich als hinreichend linear angenommen werden.

3.3.4 Messung weiterer äußerer Größen

Für eine weiterführende Verwendung des Steuermoduls sind eventuell noch weitere analoge Eingänge nötig. Über diese könnten zum Beispiel Strangströme, Maschinentemperaturen oder Drehzahlen eingekoppelt und verarbeitet werden. Hierzu werden die verbleibenden Eingänge nach aussen geführt, um später weiter beschaltet zu werden.

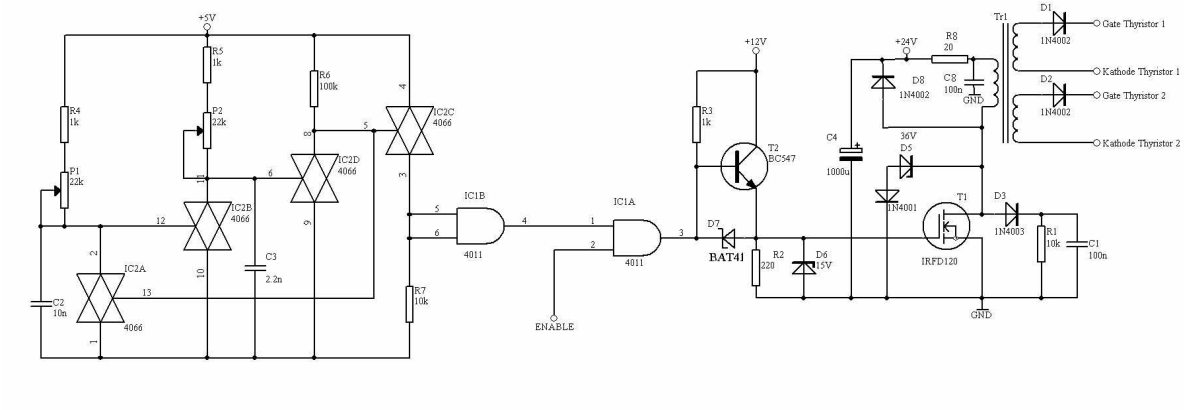
3.3.5 Ausgangsstufen

Um das Modul so universell wie möglich zu gestalten und auch die Ansteuerung von 6-pulsigen Leistungsbrücken zu ermöglichen, sind zusätzlich sechs über Optokoppler von der Aussenwelt galvanisch getrennte Schaltausgänge vorgesehen. Diese werden von den Portpins P4.0 bis P4.5 des Controllers [301] angesteuert.



Schaltbild 10. Ausgangsstufen zur universellen Ansteuerung von Leistungsteilen

Zusätzlich soll die direkte Ansteuerung von antiparallel geschalteten Thyristorpaaren ermöglicht werden. Um die Zündübertrager möglichst klein zu halten, ist ein gepulster Betrieb vorgesehen mit einer Taktfrequenz von etwa 30 KHz und einem Tastverhältniss von 0.5 vorgesehen. Da der Controller mit dieser Aufgabe mit seiner Zykluszeit von 1 µs sehr viel Rechenzeit nur für die Erzeugung der Zündimpulse benötigen würde, werden diese über einen Taktgenerator erzeugt.



Schaltbild 11. Steuerendstufe zur Thyristorzündung mit Takterzeugung

IC_2 enthält 4 analog-Schalter, welche über eine Seuerspannung geöffnet oder geschlossen werden können. Ist Schalter B geöffnet, findet sich am Steuerausgang von SchalterD eine logische 1 (Schalter geschlossen) und deshalb eine logische 0 an den Steuereingängen von Schalter A und C. C_2 kann sich nun über P_1 und R_4 aufladen, bis die Schaltschwelle von Schalter B erreicht ist. Dieser schliesst sich und entlädt C_3.

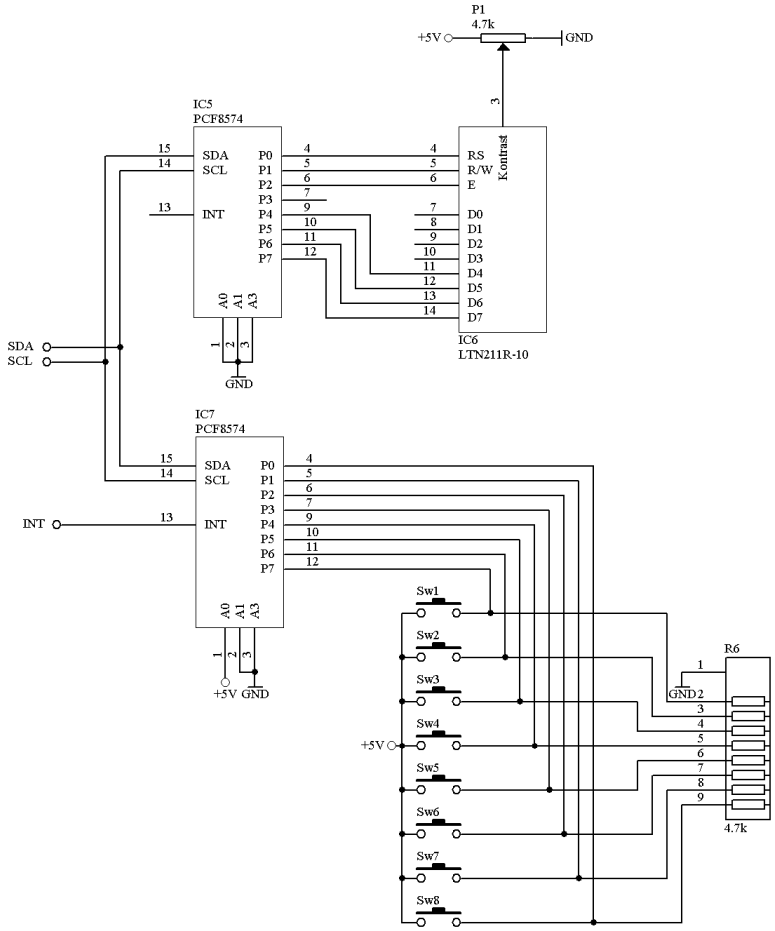
Gleichzeitig wird der Eingang von Schalter D low, so daß die Steuereingänge von Schalter A und C aktiv werden. Schalter A entlädt C_2, so daß sich Schalter B wieder öffnet. Über P_1 und P_2 sind die Impuls- und Pausenzeiten getrennt einstellbar.
Das als Inverter geschaltetet NAND-Gatter IC_{1B} erzeugt die nötige Flankensteilheit. Über das zweite NAND-Gatter IC_{1A} werden die Impulse über ENABLE vom Portpin freigeschaltet.
Die Zündendstufe selbst besteht aus einen Totem-Pole-Treiber (T_2, R_3, D_7), welcher den TTL-Pegel des Taktgenerators auf +12V anhebt und den erforderlichen Strom für die Ladung des Gates von T_1 liefert. D7
R_2 sorgt nach dem Abschalten für eine rasche Entladung des Gates. D_4, D_5 und D_6 bilden die Schutzbeschaltung des MOSFET T_1 und die Parallelschaltung von R_1 und C_1 bedämpft entstehende Spannungsspitzen. D_3 verhindert dabei, das C_1 beim Einschalten über T_1 entladen werden muß.
Der Kondensator C4 liefert die notwendige Energie beim Einschalten des Übertragers mit einem Wicklungsverhältniss von 3:1:1, der über D_1 und D_2 den Zündstrom für die Thyristoren liefert. D_8 sorgt für die Entmagnetisierung des Übertragers über R_8, welcher den Einschaltstrom begrenzt.

3.3.6 Buskonzept

Der gewählte Controller besitzt nicht genug Portpins, um alle angestrebten Funktionen bedienen zu können. Aus diesem Grund werden periphere Funktionen wie Tastatur, Display, Konfigurationspeicher und externe IO über den I^2C-Bus angesprochen [301, 303]. Dieser wird vom Controller direkt unterstützt.
Dies ermöglicht auch die spätere Erweiterung des Moduls ohne aufwendige Änderungen an der Hardware.

3.3.6.1 Bedienungskonzept

Die Bedienung des Gerätes durch den Anwender soll über eine kleine Tastatur und ein Klartextdisplay erfolgen. Da der Controller nicht genügend Portpins zur Verfügung stellt, soll dies über Portexpander über den I^2C-Bus geschehen [407]. Dazu dienen zwei IO-Expander PCF 8574 von Phillips [303]. Das Display ist ein LC-Standardmodell, wie es von vielen Herstellern (hier Hitachi LTN211R-10) [401] hergestellt wird.



Schaltbild 12. Tastatur und Display am I^2C-Bus

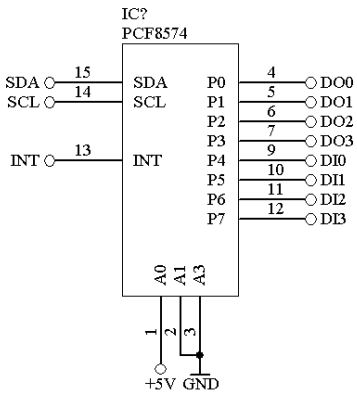
IC_6 bedient das LC-Display im Nibble-Modus [401], P_1 dient zur Kontrasteinstellung. IC_7 übernimmt die Abfrage der Tastatur aus 8 Tastern, deren Pegel sich bei Betätigung durch R6 von Low auf High ändert.
Neben den I^2C-Busleitungen wird auch die Interruptleitung von IC_7 genutzt, welche zur Generierung eines Tastaturinterrupts dient.

3.3.6.2 Speicherung der Konfigurationsdaten

Die vom Bediener des Gerätes vorgegebenen Parameter müssen dauerhaft gespeichert werden. Hierzu soll ein E^2PROM der Firma Phillips dienen. Der PCA8581 [302] speichert 128*8Bit und ist über den I^2C-Bus [407] ansprechbar (siehe *Buskonzept*).

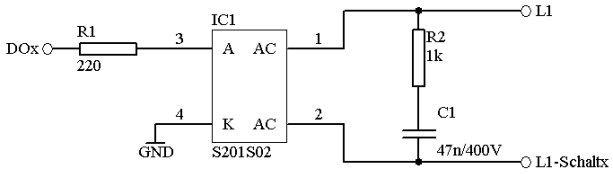
3.3.6.3 Zusätzliche Schalt-Ein- und Ausgänge

Das Modul soll zusätzliche Steuerspannungs-Ein- und Ausgänge erhalten [siehe *Schaltungsumfang*]. Diese werden über einen weiteren I^2C-Bus-Portexpander PCF8574 realisiert, welcher jeweils vier (Di_x) digitale Ein- und vier digitale Ausgänge (Do_x) zur Verfügung stellt [303] :



Schaltbild 13. Zusätzlicher IO-Port

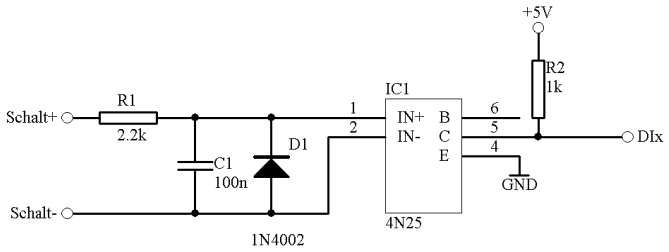
Zum Einschalten des Motorschützes und eines eventuellen Überbrückungsschützes für die Leistungsteile sollen darüber hinaus noch zwei Schaltspanungen von 230 V zur Verfügung stehen. Diese werden über zwei Solid-State-Relais erzeugt. Um Induktive Lasten, wie ein Schütz, schalten zu können, müssen diese mit einer RC-Schaltung erweitert werden :



Schaltbild 15. 230V-Schaltspannungsausgänge

R_1 begrenzt den Strom durch die Diode des integrierten Optokopplers, R_2 und C_1 bilden die oben genannte Schutzbeschaltung. Somit schaltet diese Relaisschaltung eine Last von L_1 nach N bis maximal 1A.

Des weiteren müssen Steuerspannungseingänge vorgesehen werden, um von außen zwischen Betriebszuständen umschalten zu können. In diesem Fall sind dies NOT-Aus und die Umschaltung zwischen Hoch- und Auslaufbetrieb.
Dies wird über das Anlegen einer Schaltspannung von 24V, ein im Industriebbereich gängiges Verfahren, angezeigt.
Um diese Spannung galvanisch vom Modul zu trennen und in TTL-Pegel umzusetzen, werden auch hier Optokoppler eingesetzt :



Schaltbild xy. Steuerspannungseingänge

R_1 begrenzt den Strom durch die LED des Optokopplers IC_1, D_1 verhindert zu hohe Sperrspannungen bei möglicher Verpolung und C_1 bedämpft hochfrequente Störungen. Auf der sekundären Seite erzeugt der Pull-up-Widerstand R_2 zusammen mit dem Schalttransistors des Optokopplers den erforderlichen TTL-Pegel.

3.3.6.4 Struktur des I²C-Busses

Die Struktur des internen I²C-Busses setzt sich somit wie folgt zusammen :

Gerät	Realisierung	Slaveadresse
Tastatur	PCF8574	66d
Display	PCF8574	64d
Konfigurationsspeicher	PCA8581	160d
IO-Ports	PCF8574	68d

Tabelle 9. Adressen des I²C-Busses

3.3.7 Die Controllerplatine

Der Controller im Blockschaltbild des Systems beinhaltet einen kompletten Kleinrechner. Basis ist der Mikrokontroller (80C552 von Phillips [301], einem zu Intels 8052-kompatiblen, aber um folgende Funktionen erweiterten Prozessor :

- I^2C-Schnittstelle
- Integrierter 10Bit-AD-Wandler mit 8 Eingängen
- 2 PWM-Ausgänge
- 2 zusätzliche Timer

Er bedient ein 32k-Eprom als Festwertspeicher sowie 32k statisches Ram, welche, um die Adressdekodierung so einfach wie möglich zu gestalten, direkt hintereinander folgen :

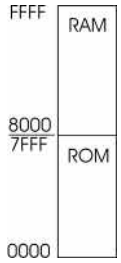


Bild 25. Speicherraum der Controllerplatine

Für die Anwendung als Anschnittsteller ist ein RAM von 32k-Bytes zu groß dimensioniert. Dies dient als Option für eine weiterführende Verwendung der Controllerplatine.

Herausgeführt werden aus Gründen der Vereinfachung lediglich die folgenden Anschlüsse :

- Port 4 (8 Bit IO)
- der I^2C-Bus (SDA und SCL)
- die asynchrone Schnittstelle (TxD und RxD)
- die 8 Wandlereingänge
- die 2 PWM-Ausgänge
- die externen Interrupteingänge (INT0 und INT1)
- der Reset-Eingang

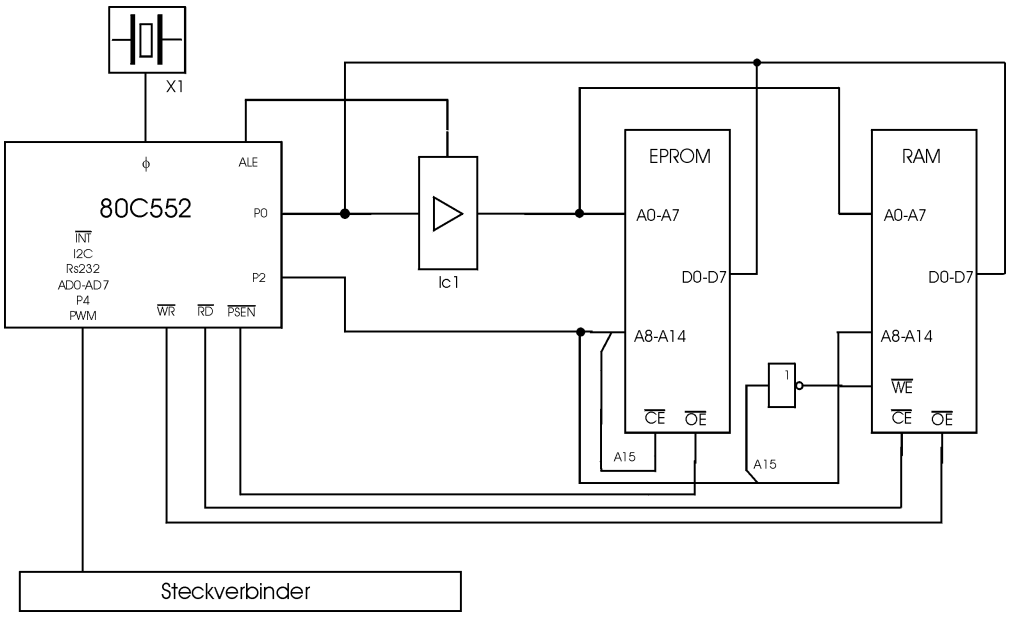
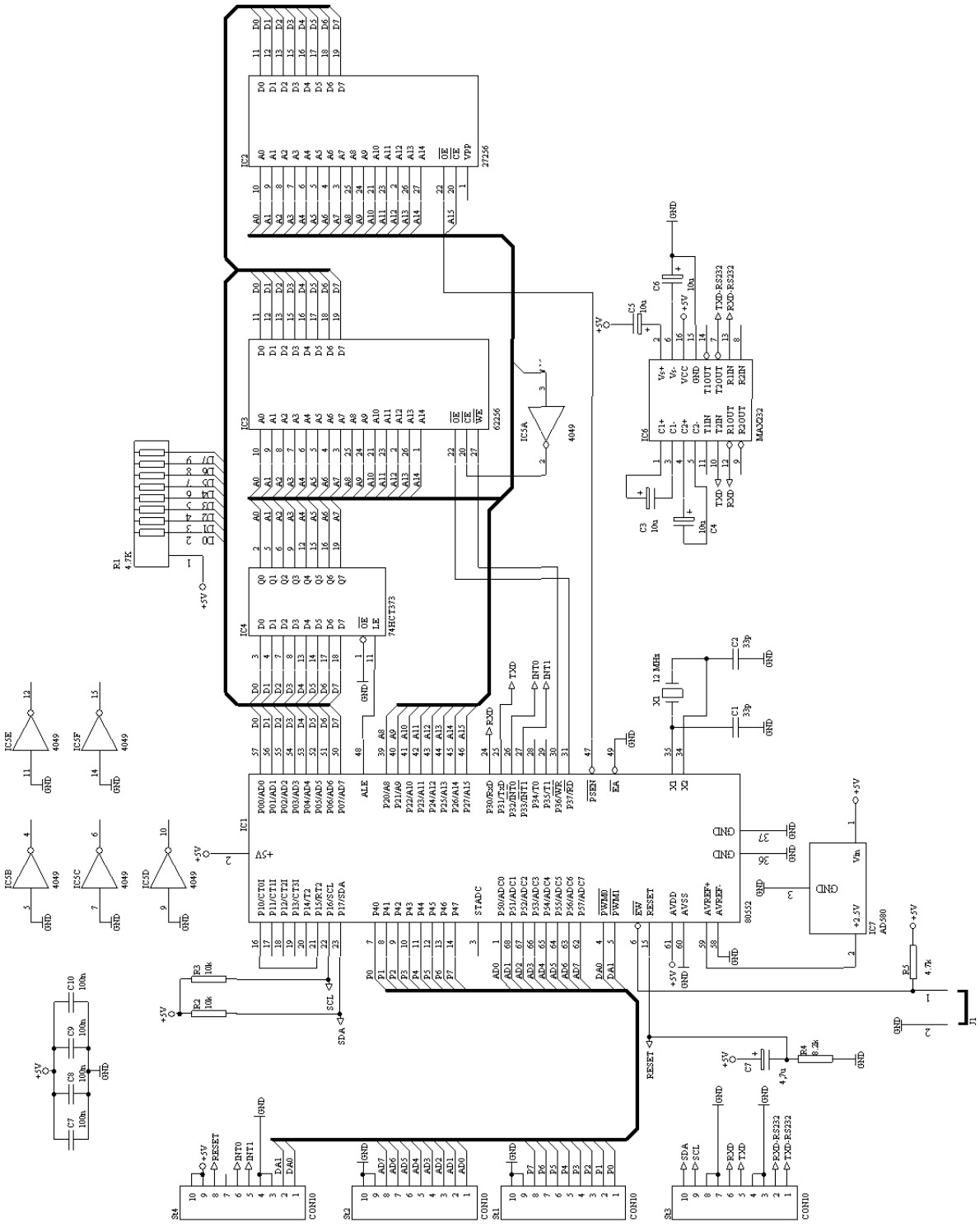


Bild 26. Struktur der Kontrollerplatine

Das Adress- und Datenmultiplexing übernimmt Adresslatch IC_1, gesteuert über die ALE-Leitung des 80C552. Den Takt erzeugt ein Quarz mit 12 MHz. Die Selektion des Adressraumes geschieht über A_{15} des Controllers, so daß sich das oben genannte Adresschema ergibt. P1.0 dient als Reset für Timer 2 (P1.5).Über einen zentralen Steckverbinder werden die wichtigen Signale herausgeführt.
Durch diese Anordnung der Hardware werden mit dem Vorteil der einfachen Struktur einige Nachteile erzeugt :

- Erweiterungen der Hardware sind nur über den I^2C-Bus sinnvoll möglich, da Adress- und Datenbus nicht herausgeführt sind
- Die Anzahl der Portpins ist begrenzt
- Das Ausführen von Programm im RAM ist nicht möglich, die Verwendung von Monitorprogrammen somit ausgeschlossen

Aus diesen Überlegungen entstand ein eigener, kleiner Einplatinenrechner, welcher auf einer eigenen Leiterplatte aufgebaut wird, um ihn für andere Anwendungen nutzen zu können :



Schaltbild 16. Schaltbild der Kontrollerplatine

3.4 Entwicklung der Leiterplatten

3.4.1 Entwicklung der Controllerplatine

Um den Kleinrechner auch in anderen Projekten weiterverwenden zu können, liegt der Schwepunkt der Leiterplattenentwicklung hier auf einer möglichst effizienten Nutzung des Raumes. Alle Steckverbinder werden nach unten herausgeführt, so daß die Platine Huckepack montiert werden kann. Die Wiedergabe ist hier nicht maßstäblich (siehe *ANHANG*) :

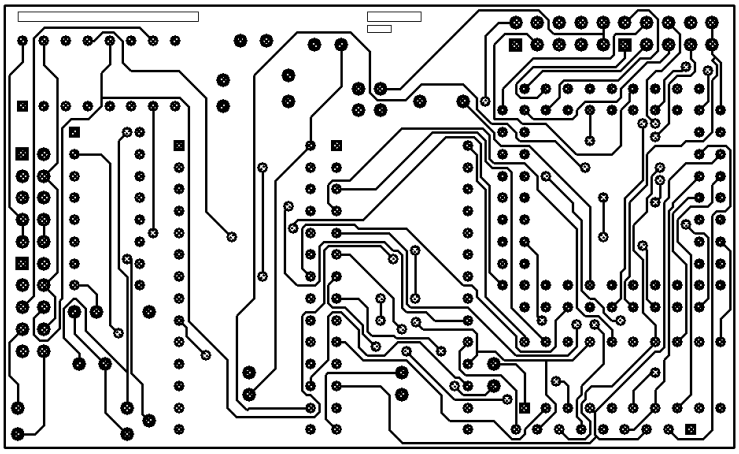


Bild 27. Bestückungsseite der Controllerplatine

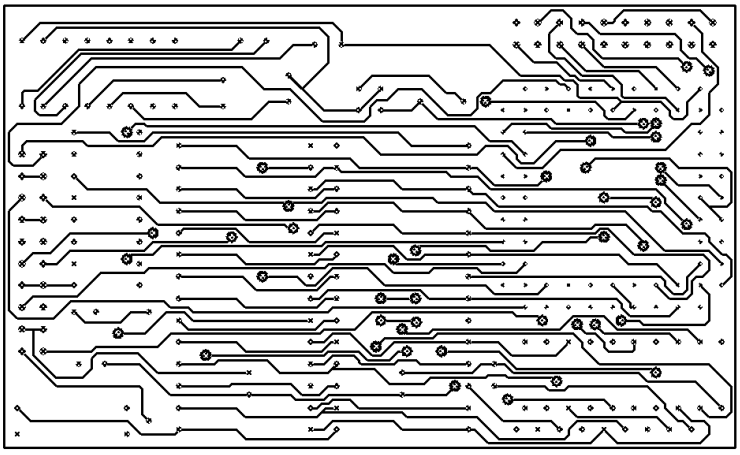


Bild 28. Unterseite der Controllerplatine

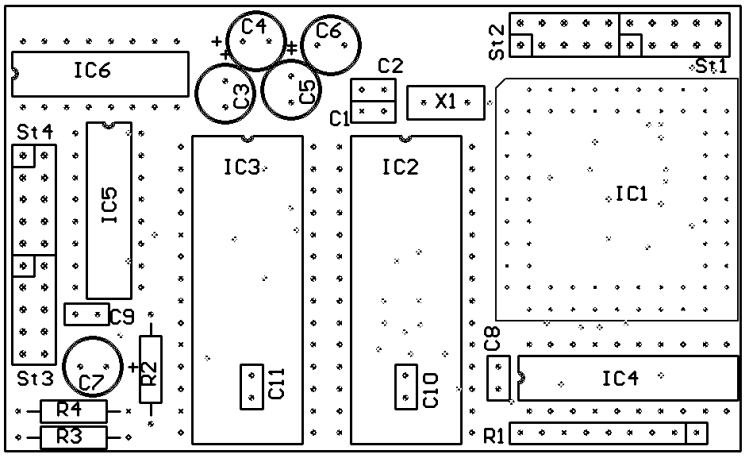
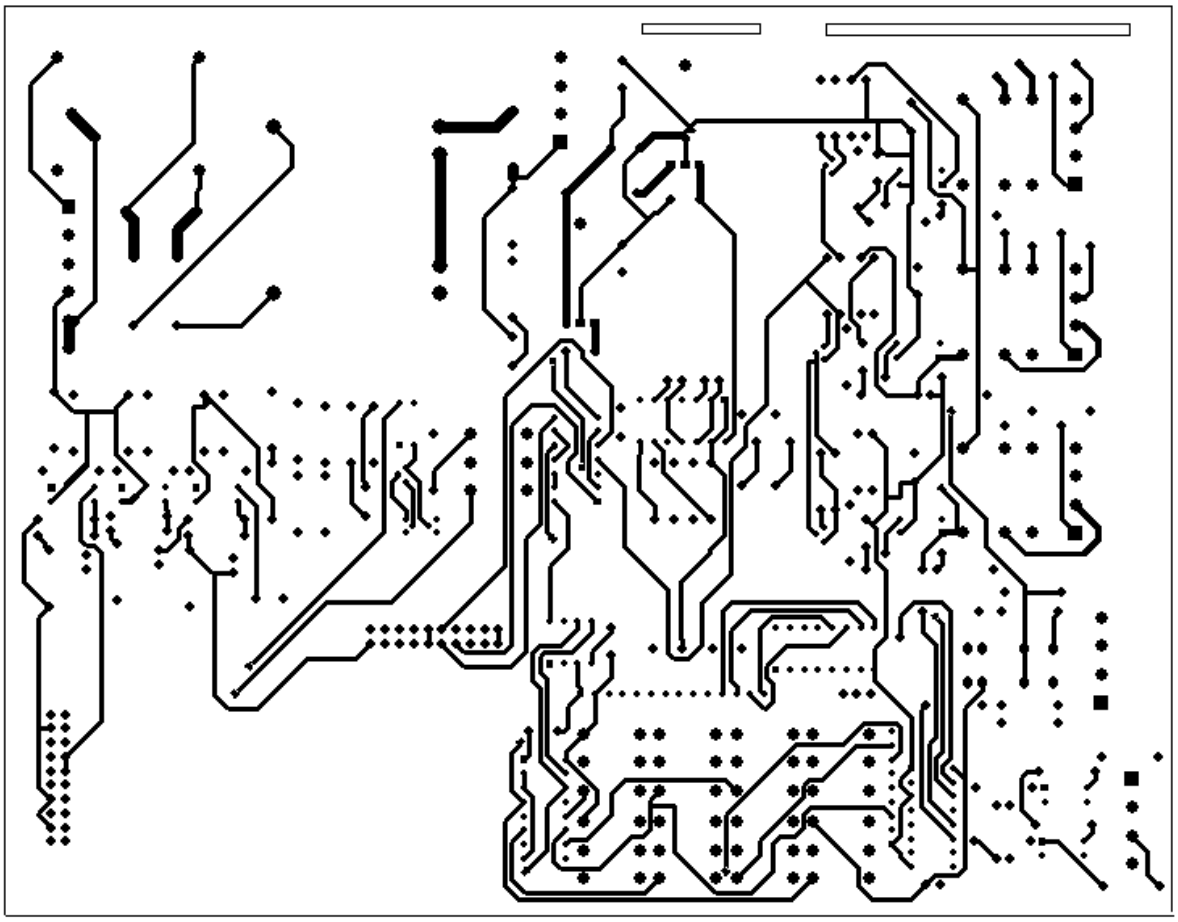


Bild 29. Bestückung der Controllerplatine

3.4.2 Entwicklung der Hauptplatine

Die Hauptplatine nimmt alle Bauteile ausser der Thyristoren und deren Schutzbeschaltung auf. Bei Bedarf können dann die Bedienelemente und die Anzeige über Flachbandkabel nach aussen geführt werden.



ild 30. Oberseite der Hauptplatine

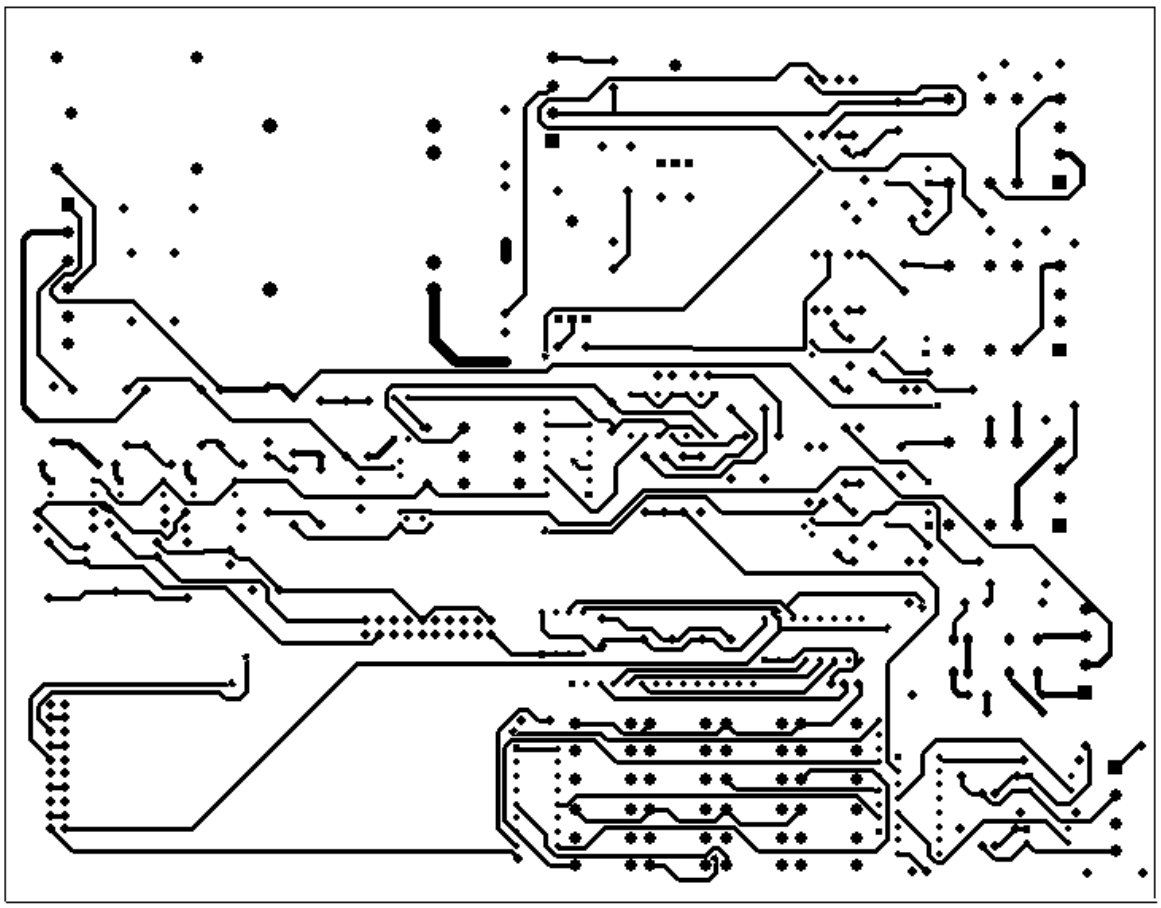


Bild 31. Unterseite der Hauptplatine

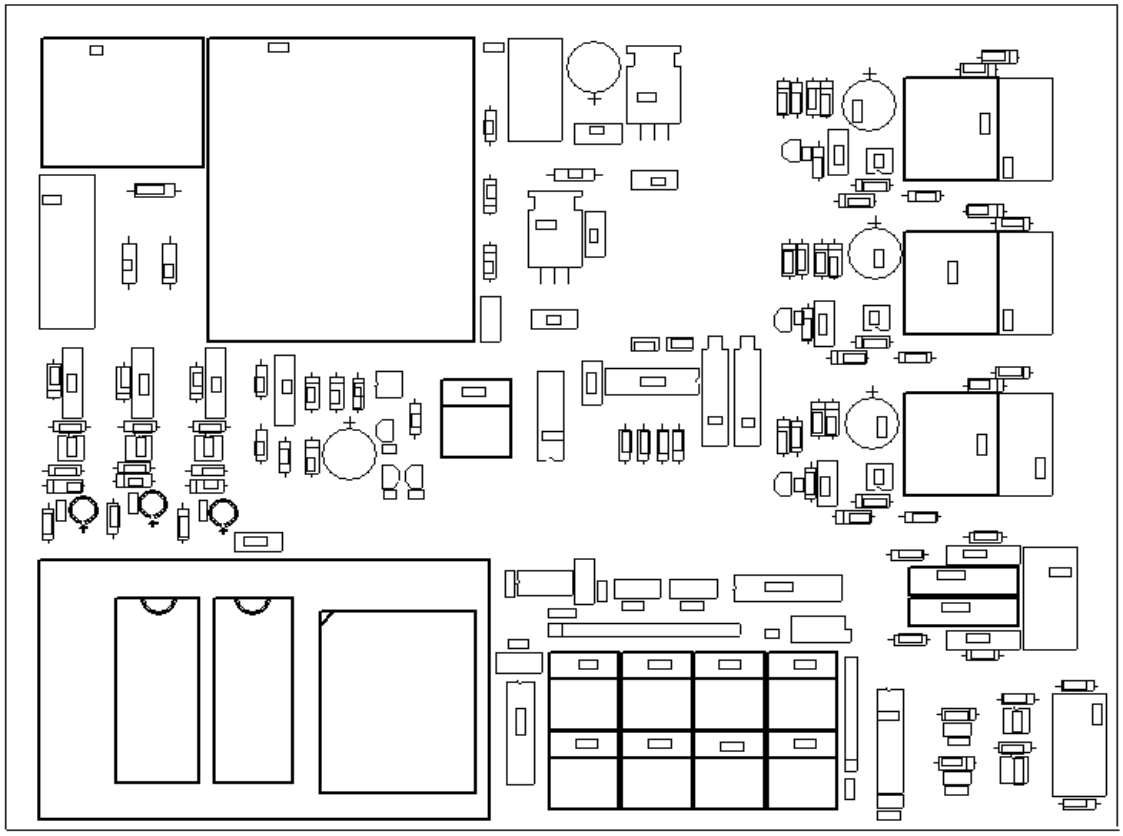
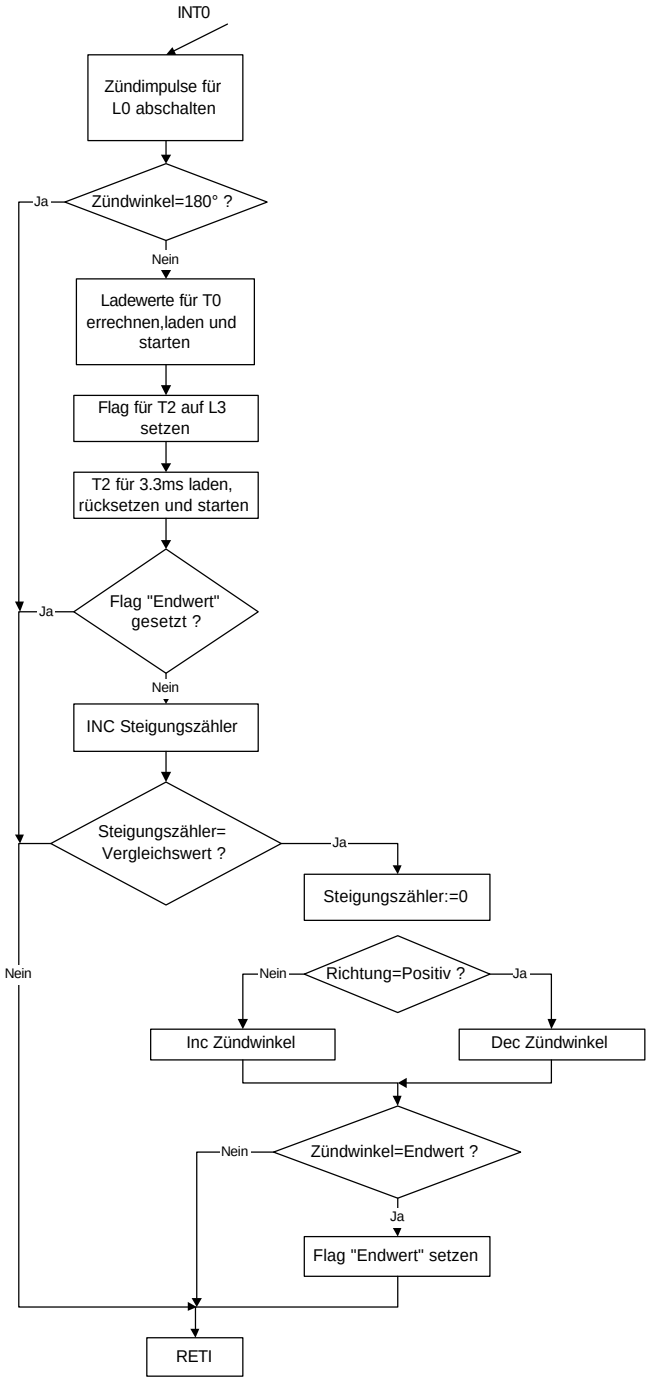


Bild 32. Bestückung der Hauptplatine

4. Softwareentwicklung

4.1 Strategie zur Zündimpulserzeugung

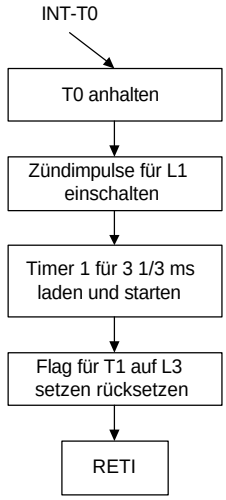
Bei Nulldurchgang von L_1, angezeigt durch den externen Interrupt INT0, wird Timer 0 mit dem Wert für den Zündwinkel α vorgeladen und gestartet. Die Zündimpulse für L1 werden abgeschaltet und T2 für 3.3ms (entsprechend 60° lektrisch bei einer Netzfrequnz von 50 Hz) gestartet mit einem auf L3 zeigenden Flag:



Struktogramm 1. ISR für INT0.

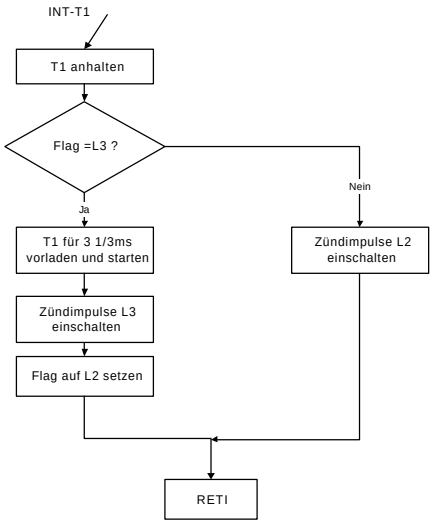
Ist der Zündwinkel=180°, so wird die ISR nach abschalten der Zündimpulse sofort wieder verlassen, weil keine weiteren Impulse generiert werden müssen.
Ist dies nicht der Fall und soll eine Spannungsrampe generiert werden, so wird in vorher zu berechnenden Abständen der Zündwinkel erhöht oder erniedrigt (siehe *Errechnung der Ladewerte, Erzeugung einer Spannungsrampe*).

Nach Ablauf von Timer 0 (Interrupt INT-T0) werden in der zugehörigen Interrupt-Service-Routine die Zündimpulse für L_1 eingeschaltet und Timer 1 für 3 1/3 ms vorgeladen (entspricht 60° elektrisch) und gestartet mit einem auf L_3 zeigenden Flag.



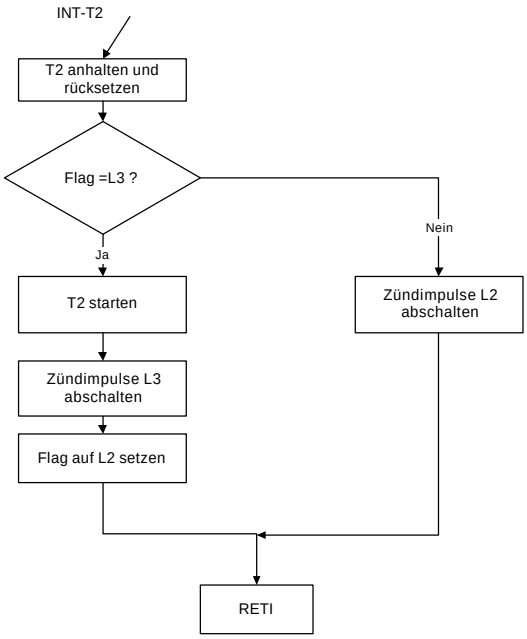
Struktogramm 2. ISR für INT-T0

Nach dessen Ablauf (Interrupt INT–T1) werden die Zündimpulse für L_3 eingeschaltet und Timer 1 erneut mit dem gleichen Wert vorgeladen und gestartet mit dem Flag für L_2. Nach seinem zweitem Ablauf werden dann die Zündimpulse für L_2 eingeschaltet :



Struktogramm 3. ISR für INT-T1

Timer 2 schaltet in seiner ISR die jeweligen Zündimpulse nach dem gleichen Schema wie ISR-T1 wieder ab :



Struktogramm 4. IST für INT-T2

Timer 2 besitzt bei diesem Controller 80C552 eine Besonderheit. Er verfügt über keine direkte Lade- oder Autoreload-Möglichkeit. Dies hat zur Folge, das der Interrupt über Compare-Funktionen erzeugt und T2 über Portpin P 1.0 an Pin T2R rückgesetzt werden muß (Siehe *Schaltplan der Controllerplatine*).

Daraus ergibt sich folgendes Interrupt-und Timerschema (α=90°) :

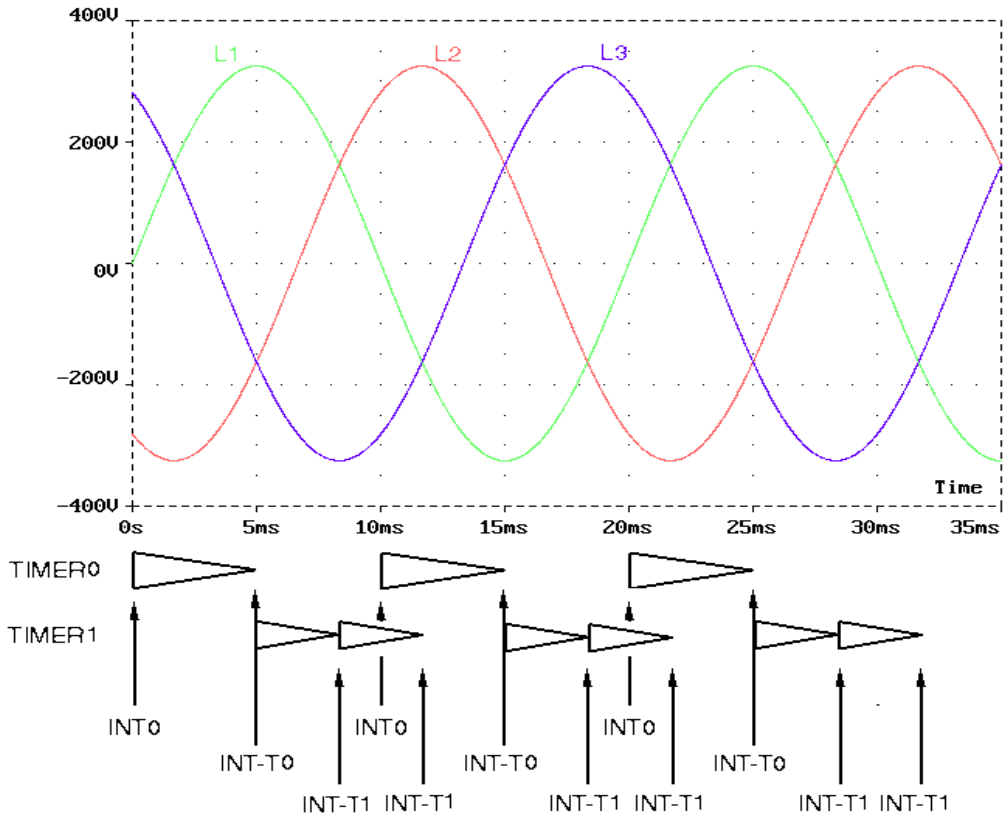


Bild xy. Zündimulsschema mit Timerwerten

Timer T_2 erfüllt die gleiche Funktion wie Timer T_1, nur das die zugehörige ISR die Zündimpulse ab- statt einschaltet.

4.1.1 Errechnung der Ladewerte für die Timer 1 und 2 :

Timer 0 muß bei einer Netzfrequenz von 50 Hz maximal 10 Millisekunden laufen (8 1/3 ms bei 60 Hz) . Daraus folgt ein maximaler Ladewert von 10^4 bei einer Taktfrequenz von 1 MHz. Dieser Ladewert entspricht einem Zündwinkel von 180°. Daraus folgt :

$$Ladewert0 := \left| \frac{10.000}{180} * a \right| + 1$$

Zum Ladewert wird eine eins addiert, da sonst ein Ladewert von 0 möglich wird, so daß Timer 1 dann von 2^{16} ab rückwärts laufen würde. Für eine Netzfrequenz von 60 Hz gilt

$$Ladewert0 := \left| \frac{8333}{180} * a \right| + 1$$

Timer 1 und Timer 2 müssen immer eine feste Zeit von 3 1/3 Millisekunden überbrücken. Bei der gleichen Taktrate von 1 MHz folgt somit für Timer 1 :

$$Ladewert1 = 2^{16} - 3333 = 62203$$

Timer 2 wird über seine Compare-Register betrieben, diese werdebn also mit 3333 geladen (2778 bei 60 Hz).

4.2 Einleitung eines Hoch- oder Auslaufens

Um die Erzeugung der Zündimpulse zu kontrollieren ist es nur noch notwendig, Start- und Endwinkel, sowie die Dauer einer möglichen Rampe festzulegen. Hieraus wird der in der ISR-INT0 verwendete Vergleichswert errechnet :

*Vergleichswert=Dauer der Rampe [s] * 100 / Zu übertreichender Winkelbereich [°]*

Da die ISR-INT0 bei Nulldurchgang der Phase L_1, also alle 10 ms aufgerufen wird, kann bei jedem Durchlauf ein Zähler erhöht und mit diesem Vergleichswert vergleichen werden. Sind beide Werte gleich, kann der Zündwinkel entsprechend um den Wert 1 korregiert werden, sofern der angestrebte Endwert noch nicht erreicht wurde. Dann wird ein Flag gesetzt und der Zündwinkel stabil gehalten. Für eine Netzfrequenz von 60 Hz muß also der Wert 100 auf 120 erhöht werden.
Dieses Vorgehen ermöglicht es, die Rampengenerierung in die Interruptroutine zu verlegen und durch das Fehlen aufwendiger Fließkommarithmetik einen erheblichen Gewinn an Rechenzeit zu erzeugen.

4.3 Menüerzeugung

Um eine möglichst einfache Parametrierung zu ermöglichen, ist eine Menüführung vorgesehen. Diese muß sich ändernden Anforderungen anpassbar sein, also eine wiederverwendbare Struktur besitzen.
Diese besteht aus 3 Ebenen. Ebene 1 enthält die Oberbegriffe, Ebene 2 deren Ausdifferentierungen und Ebene 3 die durch Ebene 1 und Ebene 2 referenzierten Parameter, so daß sich ein Satz von maximal 8*8=64 Parametern ergibt :

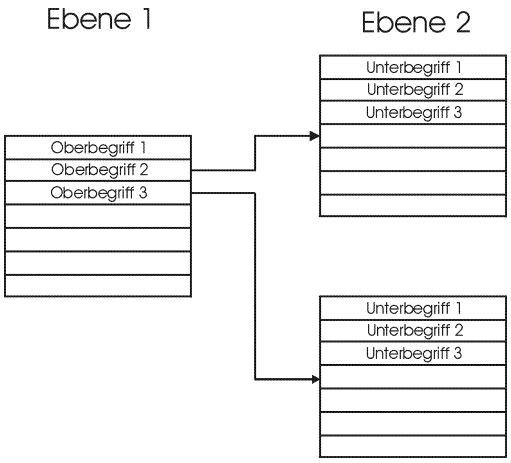


Bild xy. Struktur Menüebene 1 und 2

Jeder der 9 Blöcke (einer für Ebene 1 und 8 für Ebene 2) wird durch einen Satz aus 8 Textfeldern (11 Byte, 0-terminiert) und einem Zahlenfeld (1 Byte) dargestellt. Das Zahlenfeld gibt die Anzahl der benutzten Felder im Block an. Daraus ergibt sich eine problemlos erweiterbare Tabelle im Speicher, aus welcher die Menü´s aufgebaut werden.
Aus der Position des Bedieners in Ebene 1 (Pos1) und Ebene 2 (Pos2) wird dann eine weitere Tabelle referenziert, welche die notwendigen Informationen zur Klassifizierung der Parameter enthält. Jeder Parameter erhält dort einen eigenen Satz, so daß nach den Tabellen für Ebene 1 und 2 noch eine Tabelle mit 64 Feldern im Speicher liegt :

Parameter	Bedeutung
Typ	Art des Parameters (Byte, Wort, Wahrheitswert usw.)
Obergrenze	Oberster einstellbarer Wert
Untergrenze	Unterster einstellbarer Wert
Schrittweite	Schrittweite, in der der Parameter einstellbar ist
Einheit	Text von maximal 3 Zeichen, 0-terminiert
Adresse	Zeiger auf den Parameter im nichtfl. Speicher

Tabelle xy. Klassifizierung der Parameter in Menüebene 3

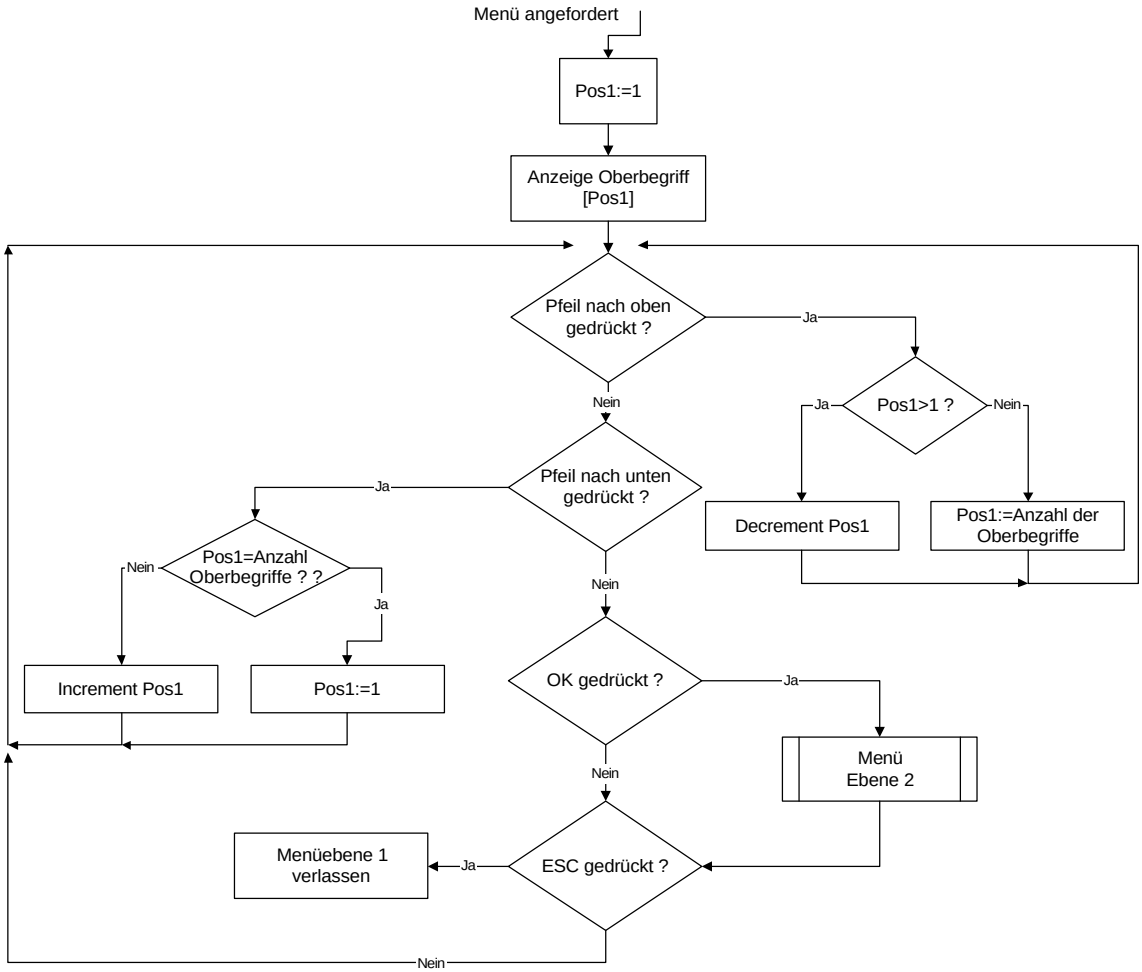
Der Typ des Parameters wird wie folgt kodiert :

Typ	Kodierung/Bedeutung
1	Wahrheitswert (1=Wahr, 0=Falsch)
2	Byte (0-FFh)
3	Wort (2 Byte) (0-FFFFh)
4	Festkommawort (2 Byte Vor-, 2 Byte Nachkomma)

Tabelle xy. Kodierung der Parameter

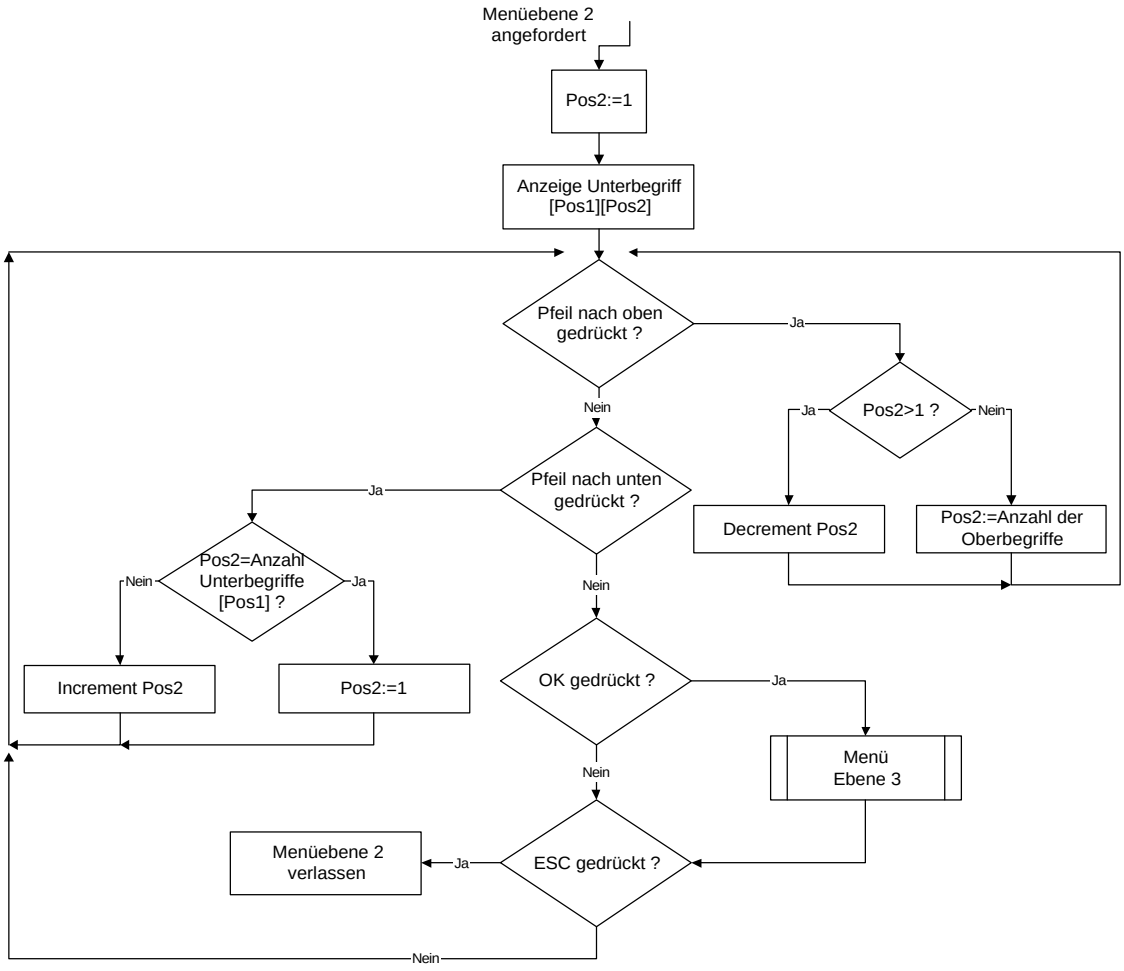
Über die Tasten Ý und ß kann dann im Menü in den Begriffen navigiert werden. Die Taste **OK** schaltet zur nächsten Ebene, die Taste **ESC** eine Ebene zurück. In Ebene 3 können die Parameter mit den Taste + und – nach oben und unten verändert werden. Die Taste **OK** speichert diesen Wert, **ESC** stellt den ursprünglichen Wert wieder her.

Menüebene 1 wird durch folgende Struktur aufgebaut :



Struktogramm 5. Menüebene 1

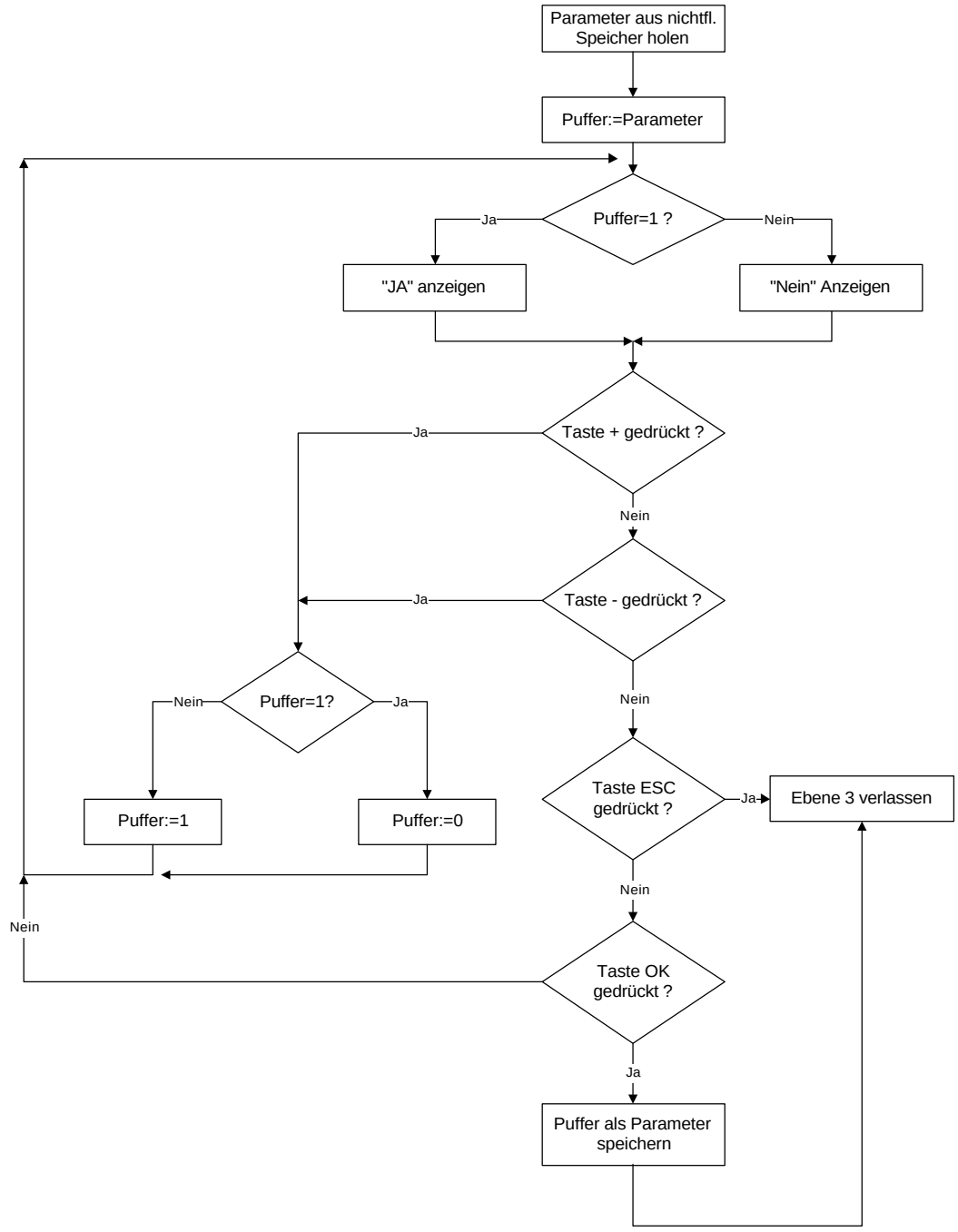
Menüebene 2 gestaltet sich sehr ähnlich, nur werden hier die jeweiligen durch Pos1 referenzierten Unterbegriffe über einen zweiten Zähler Pos2 ausgewählt :



Struktogramm 6. Menüebene 2

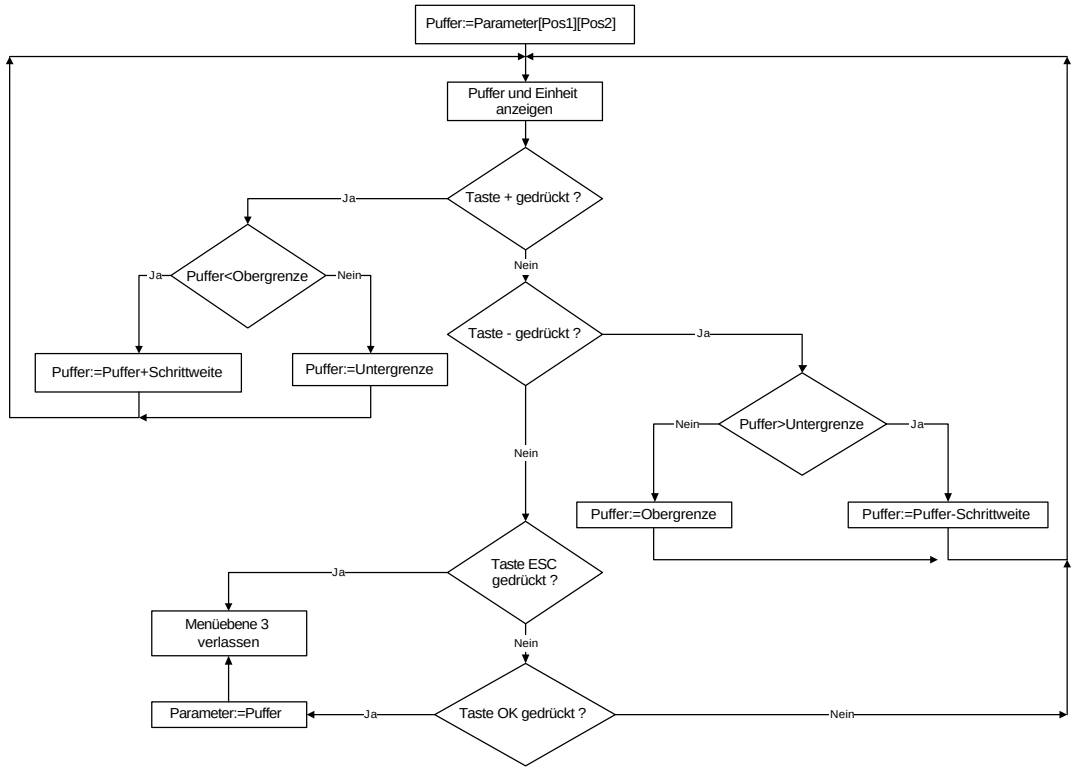
Menüebene 3 gestattet das Ändern der eingetragenen Parameter. Ein Druck auf die Taste **OK** führt in das Bearbeitungsmenü. Dort wird der Typ des jeweiligen Parameters festgestellt und in eine passende Bearbeitungsroutine verzweigt. Für diese Anwendung werden nur Wahrheitswerte und Werte im Zahlenbereich 0-200 verwendet, also sind nur 2 Routinen erforderlich.

Wahrheitswerte lassen sich durch die Zustände „Ja“ und „Nein“, etwa zur Frage der Einschatung eines Motorschützes, darstellen :



Struktogramm 7. Ebene 3-Menü zur Änderung eines Wahrheitswertes

Die Ebene 3-Funktion zur Änderung eines Zahlenwertes ist nur geringfügig umfangreicher :



Struktogramm 8. Ebene 3-Menü zur Änderung eines Zahlenwertes

Für die gegebene Anwemdung des Moduls zum Aufbau eines Sanftanlaufgerätes ergibt sich somit folgende Menüstruktur :

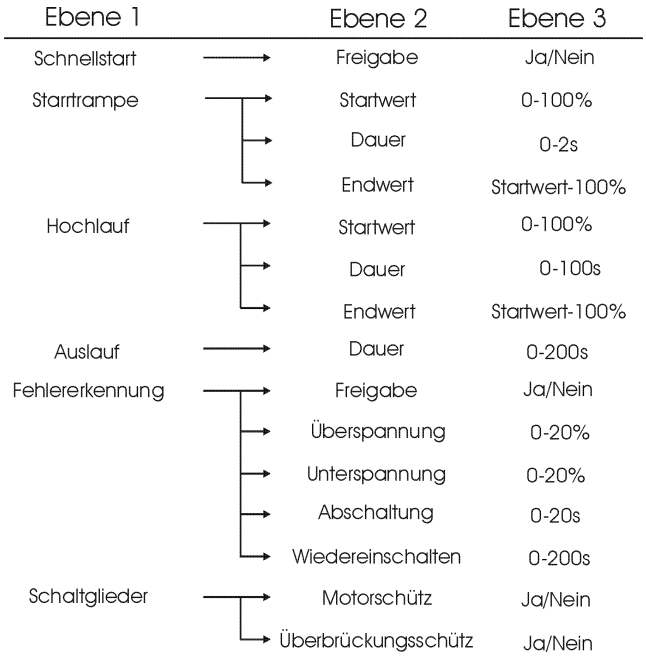
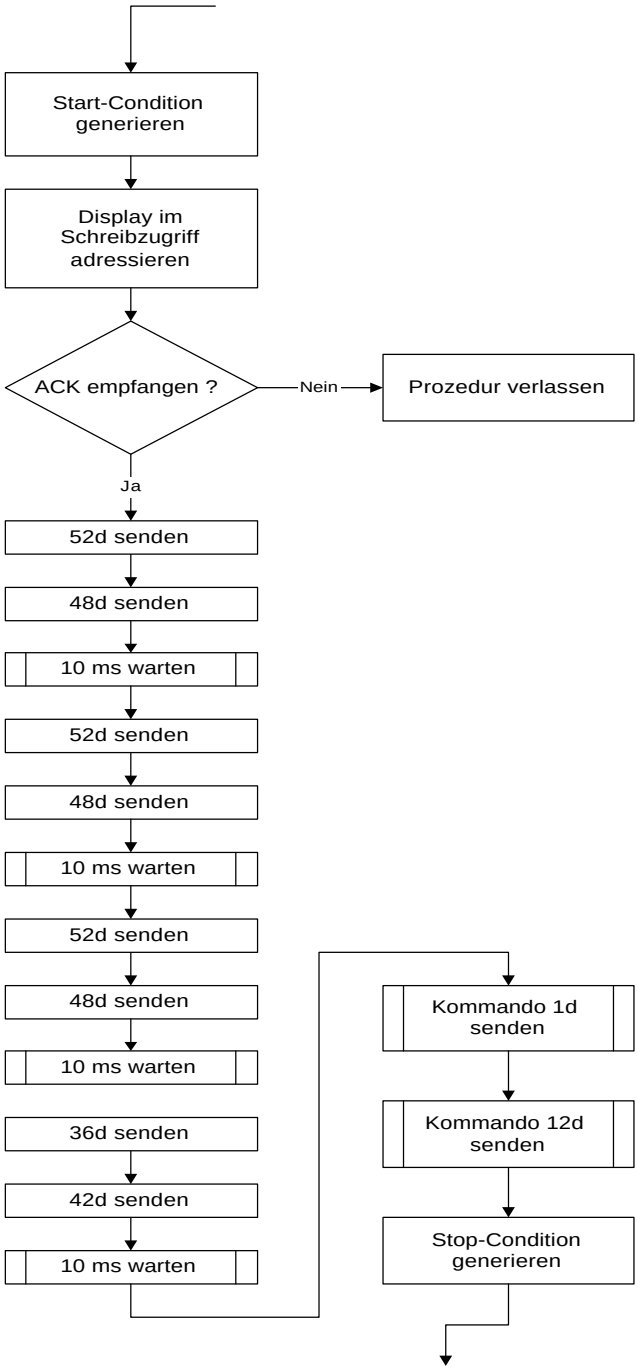


Bild xy. Menübaum des Sanftanlaufes

4.4 Steuerung des Displays

Da die Klartextanzeige am I²C-Bus über einen Portexpander [303] angesteuert wird, ist einiger Aufwand zur Zeichendarstellung notwendig [401].
Das Display muß nach dem Einschalten des Gerätes initialisiert werden. Dazu muß eine Reihe von festgelegten Werten zum Display übertragen werden [401].

Folgende Prozedur schaltet die Anzeige in den Nibble-Modus mit abgeschaltetem Cursor und einem 5*11 Zeichensatz :

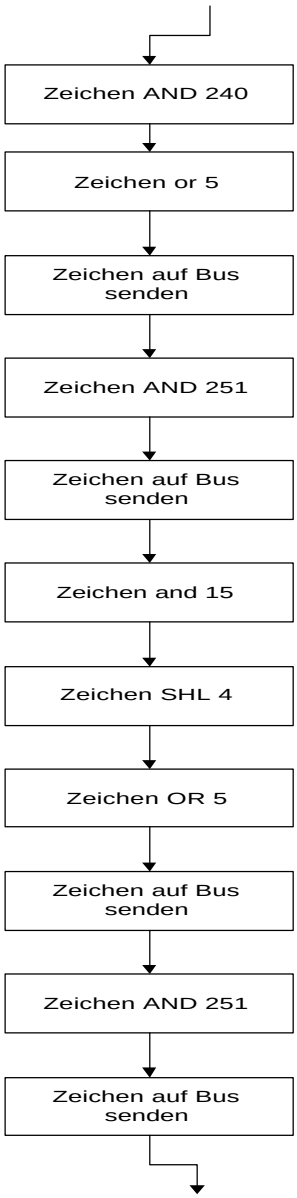


Struktogramm 9. Initialisierung des Displays

4.4.1 Zeichenausgabe auf dem Display

Anschließend muß jedes darzustellende Zeichen und jedes Kommando in seine Nibbles zerlegt und diese nacheinander übertragen werden, nachdem der Portexpander wieder im Schreibmodus adressiert wurde.

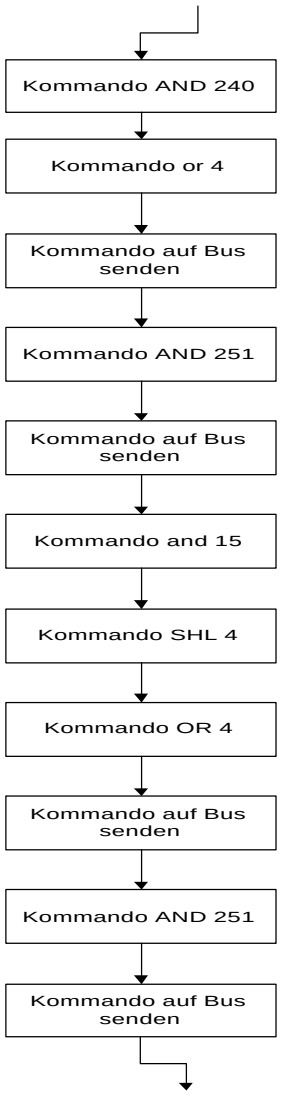
Dann legt der logische Zustand der Leitung RS des Displays fest, ob es sich um ein Zeichen oder ein Kommando handelt. Um ein Zeichen an der Cursorposition auszugeben dient folgende Prozedur :



Struktogramm 10. Ausgabe eines Zeichens auf dem Display

4.4.2 Kommandoübertragung zum Display

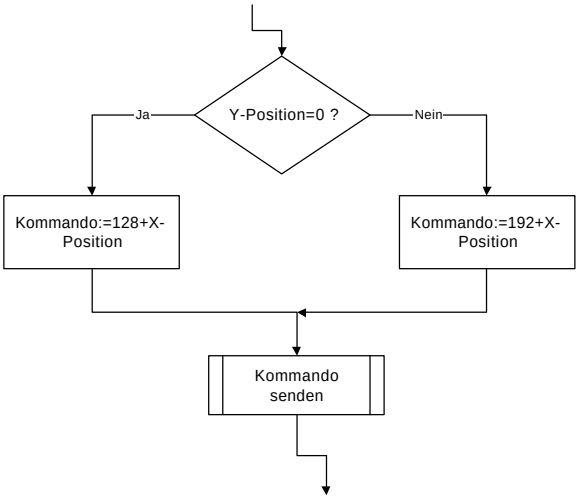
Ein Kommando zu übermitteln übemimmt folgende Prozedur (natürlich wieder bei korrekt adressiertem Portexpander) :



Struktogramm 11. Kommandoübermittlung zum Display

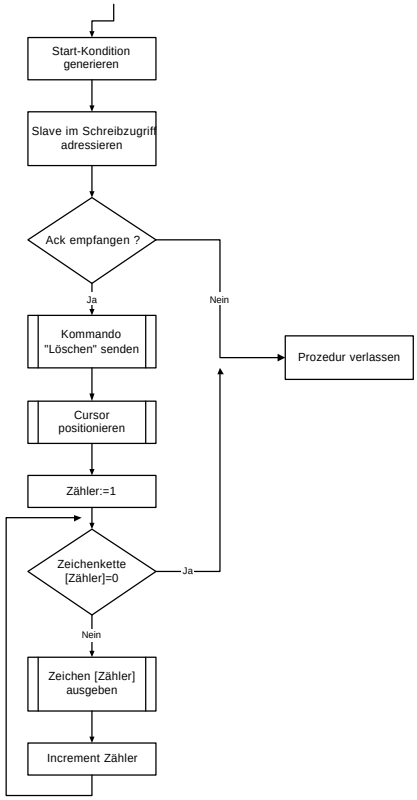
4.4.3 Cursorpositionierung

Um eine Zeichenkette an einer definierten Position ausgeben zu können, ist es notwendig, den Cursor positionieren zu können. Dies erfordert etwas Berechnung, da der Speicher der Anzeige nur innerhalb einer Zeile linear verläuft [401] :



Struktogramm 12. Positionierung des Cursors im Display

Somit ergibt sich die folgende Prozedur, um eine Zeichenkette auf der Anzeige auszugeben. Hier liegt Sie als Null-terminierte Zeichenkette vor :

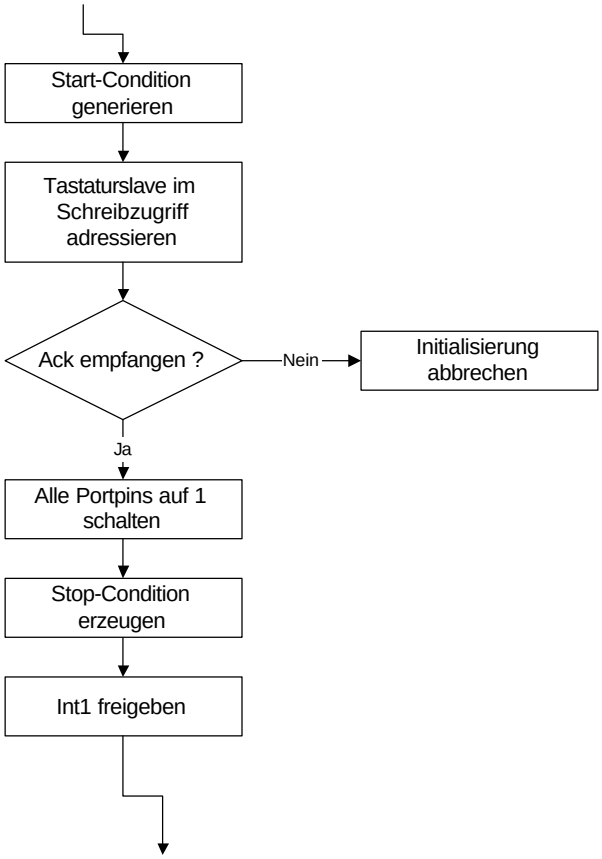


Struktogramm 13. Ausgabe einer Zeichenkette auf dem Display

Für weitere Details sein auf die „*Mini-LCD FAQ*“ [401] und auf den kommentierten Quellcode verwiesen.

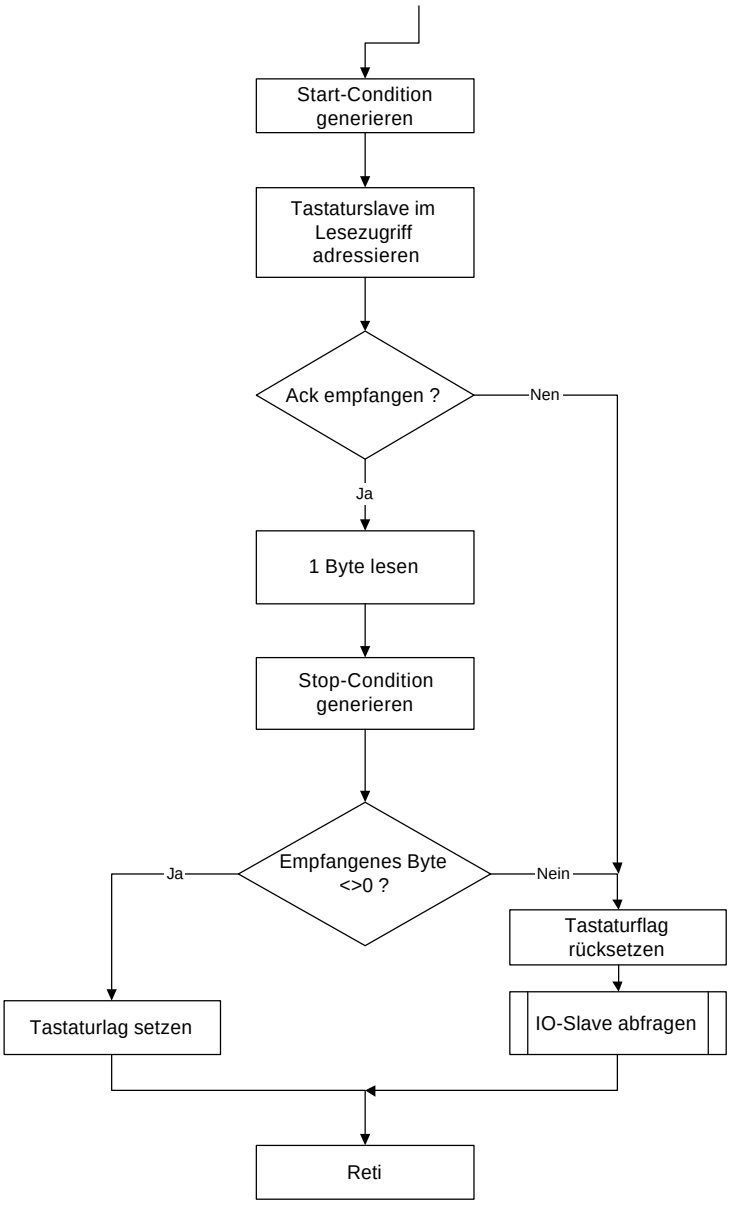
4.5 Steuerung der Tastatur

Auch die Tastatur wird über eine Portexpander [303] im I^2C-Bus betrieben. Um die Pegel abfragen zu können, müssen die Portpins des Expanders auf 1 geschaltet und der externe Interrupt 1 freigegeben werden, ebenso die Eingänge des IO-Expanders für die Steuerspanungseingänge (P_4-P_7) :



Struktogramm 14. Initialisierung der Tastatur

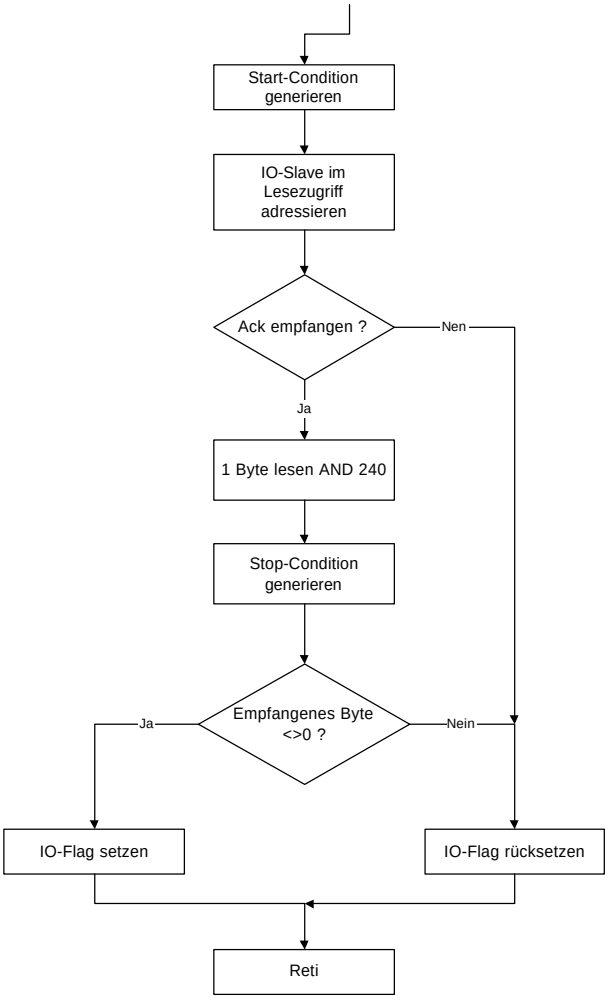
Dann kann der Tastenwert direkt ausgelesen werden :



Struktogramm 15. Tastaturabfrage

Wurde kein Tastendruck festgestellt, ist es möglich, das einer der Steuerspanungseingänge (siehe *3.3.6.3 Zusätzliche Schalt-Ein- und Ausgänge*) seinen Pegel gewechselt hat.

Dies muß in einer Extraroutine festgestellt werden :

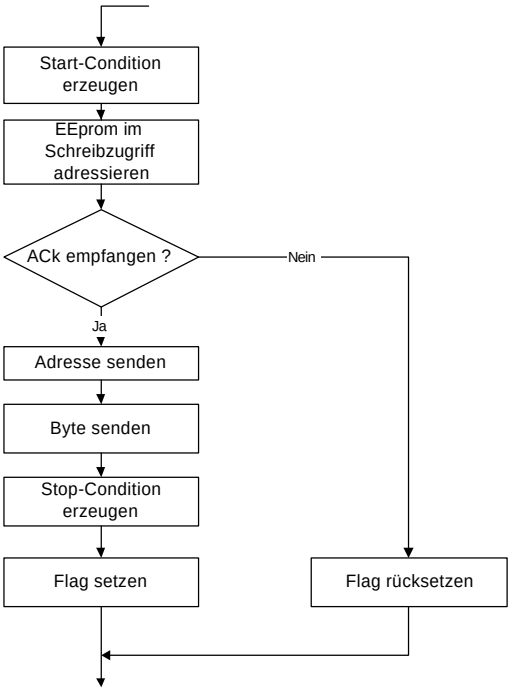


Struktogramm 16. Steuerspannungseingänge abfragen

4.6 Schreiben und Auslesen der Konfigurationsdaten

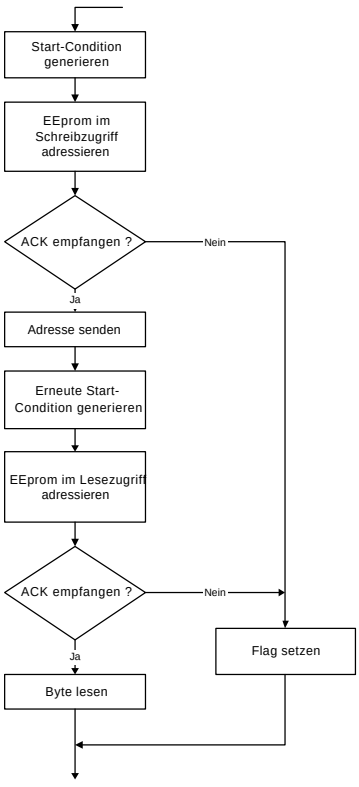
Die Konfigurationsdaten werden in einem über den I^2C-Bus ansprechbaren EEProm abgelegt [302]. Dieses stellt eine Speicherkapazität von 128 Byte zur Verfügung. Durch die Busstruktur ist es auch später problemlos möglich, diesen Speicher durch den Einsatz eines anderen IC´s zu erweitern.

Bei einem Schreibzugriff wird direkt nach der Adresse das zu schreibende Byte übertragen :



Struktogramm 17. Schreiben einesr Konfigurationsbytes

Das Auslesen eines Bytes ist nur geringfügig aufwendiger. Hierzu wird zuerst im Schreibzugeriff die Adresse übertragen und anschliessend in einem Lesezugriff das Byte ausgelesen :



Struktogramm 18. Lesen eines Konfigurationsbytes

4.7 Programmierdetails

4.7.1 Externe Interrupts

Um INT0 und INT1 bei fallender Flanke auszulösen, müssen im SFR TCON die Bits IT0 (TCON.0) und IT1 (TCON.2) auf 1 gestzt werden. Die Freigabe erfolgt durch das Setzen der Interrupt-Enable-Bits EX0 (SFR IE0.0) und EX1 (SFR IE0.2) und natürlich der globalen Interrupt-Freigabe EA (SFR IE0.7).

4.7.2 Timer-Interrupts

Die übergelaufenen Zähler setzen die jeweiligen Request-Flags. Ihre Freigabe erfolgt durch die Bits ET0 (SFR IE0.1) und ET1 (SFR IE0.3). Bis auf Timer 2 werden diese durch die Hardware zurückgesetzt.

4.7.3 Prioritätstruktur der Interrupts

Es exisiteren 2 Prioritätebenen. Der Einfachheit halber werden alle Interrupts über SFR IP0 auf Priorität 1, das heißt keine gegenseitige Unterbrechung, gesetzt (IP0.0 – IP0.6 auf 1 setzen).

4.7.4 Konfiguration der Timer

Die Funktionsweise von T0 und T1 wird bestimmt durch das SFR TMOD. In diesem Fall arbeiten beide als Timer, also muß das jeweilige Bit C/F (SFR TMOD.6 und TMOD.2 auf 0 und das Bit Gate (SFR TMOD.7 und TMOD.3) ebenfalls auf 0 gesetzt werden. Die Modusbits M0 (SFR TMOD.4 und TMOD.0) und M1 (SFR TMOD.5 und TMOD.1) legen die Betriebsart mit M1:=0 und M0:=1 als 16-Bit-Zeitgeber fest.
Durch die Bits TR0 (SFR TCON.4) und TR1 (SFR TCON.6) werden die Timer T1 und T2 angehalten und gestartet und zählen bei einem Systemtakt von 12 MHz mit einer Frequenz von 1 MHz.

4.7.5 Stackpointer

Der Stackpointer zeigt nach dem Reset auf die Adresse 07h des internen Rams. Dies verhindert die Nutzung der 4 Registerbänke, also muß er auf einen höheren Wert gesetzt werden.

4.7.6 Watchdog

Der Controller bestitzt einen eigenen Watchdog-Timer, welcher verwendet werden soll. Hierzu dient Timer 3, welcher durch den Portpin EW freigegeben wird. T3 muß dann innerhalb seiner Laufzeit nachgeladen werden, damit kein Überlauf stattfindet und damit ein Reset des Controllers ausgelöst wird.
Die maximale Laufzeit beträgt mit 12 MHz Systemtakt und einem Ladewert von 00h 512ms. Wird T3 also während dieser Zeit nicht wieder auf 0 gesetzt, wird ein Reset ausgelöst. Um T3 währed des betriebes zu laden, muß zuvor das Bit WLE (SFR PCON.4) gesetzt werden.

4.7.7 I^2C-Bus-Kommunikation

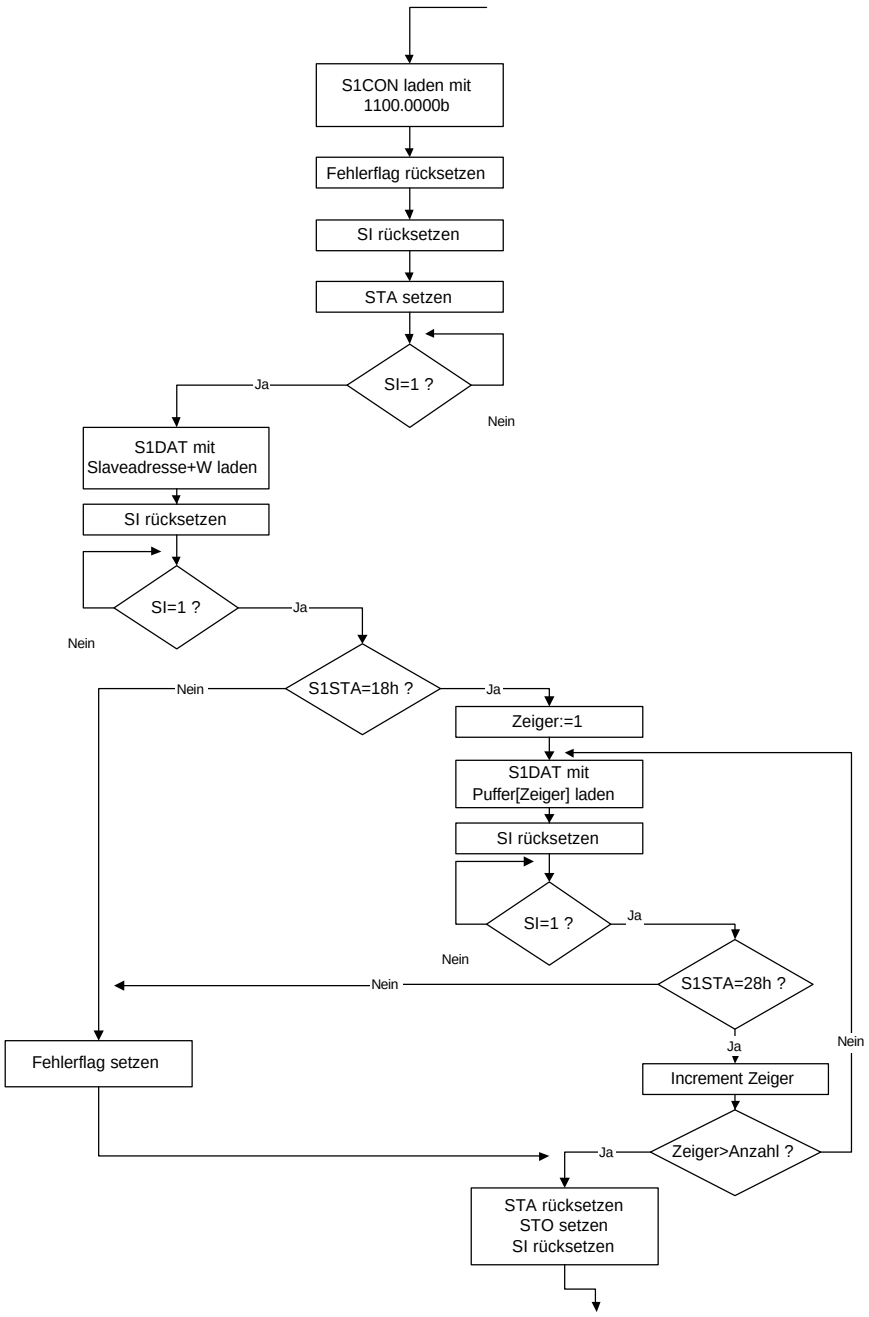
Zur Freigabe der SIO1-Funktionen [301] müssen die Portpins P1.6 (SDA) und P1.7 (SLC) auf 1 gesetzt werden. Die Kontrolle der Funktionen geschieht dann über 4 SFR´s :

Zur Freigabe des I^2C-Busses muß Bit ENS1 (SFR S1CON.6) gesetzt werden. Bit SI (SFR S1CON.3) zeigt dann eine beendete Übertragung an (gütlige Daten im SFR S1DAT) und muß per Software zurückgesetzt werden.
Eine Startcondition wird durch das Setzen des Bit STA (SFR S1CON.4) ausgelöst. Gleiches gilt für eine Stop-Condition (Bit STO, SFR S1CON.4). Das Bit AA (SFR S1CON.2) muß gesetzt werden, wenn für ein empfamngenes Byte ein Acknowledge generiert werden soll.
Die Taktrate wird über die Bits CR2, CR1 und CR0 (SFR S1CON.7, S1CON.1 und S1CON.0) gewählt zu 12.5 KHz mit CR2:=1, CR1:=CR0:=0.
In diesem Fall (Controller als Single-Master) folgen zwei mögliche Betrebsmodi :

- Der Controller arbeitet im Master-Transmitter-Mode
- Der Controller arbeitet im Master-Receiver-Mode

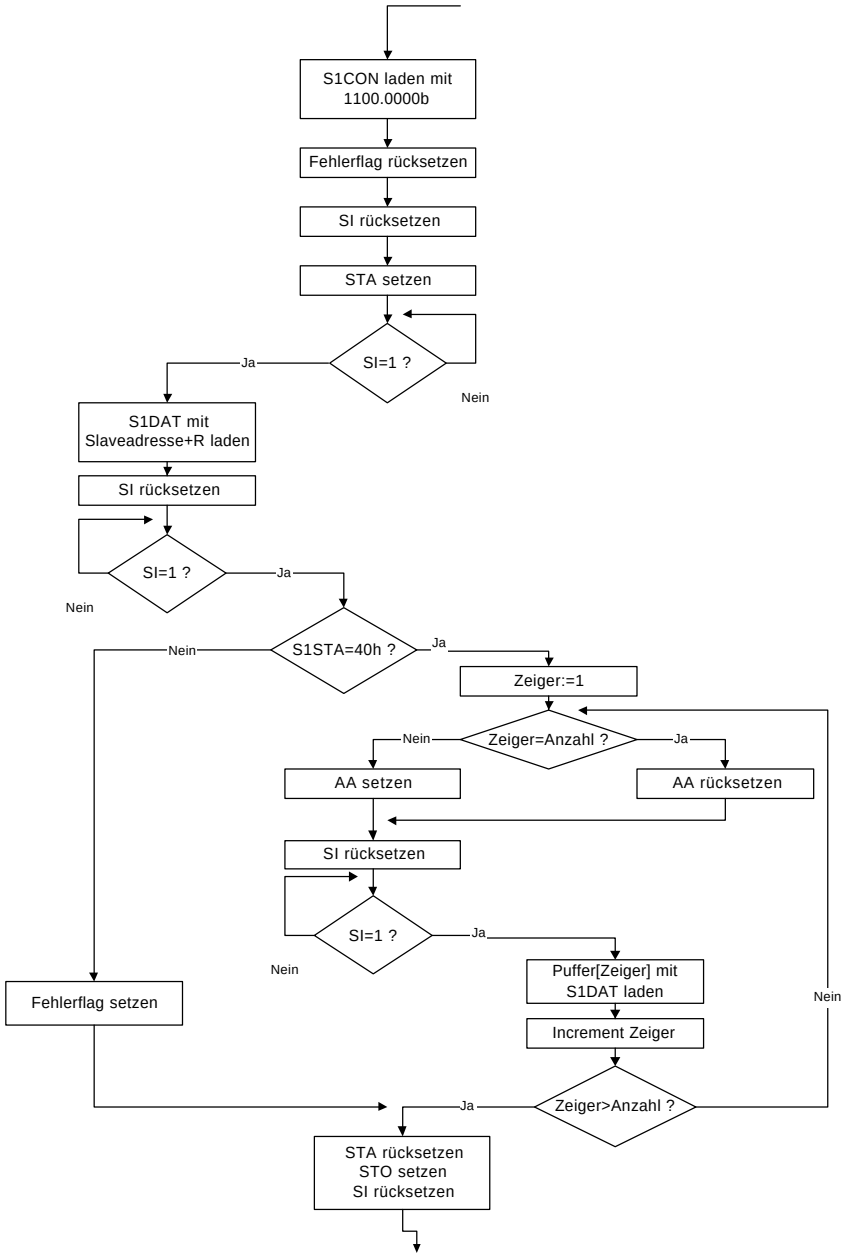
Für den Sende- und den Empfangsbetrieb wird ein Puffer eingerichtet, in welchem die empfangegen oder die zu sendenden Zeichen übergeben werden. Zusätzlich muß ein Zähler übergeben werden, welcher die Anzahl der zu übertragenen Zeichen angibt und somit den Puffer adressiert.

Damit folgt für den Master-Transmitter-Mode :



Struktogramm 19. Master-Transmitter-Mode im I^2C-Bus

Für den Master-Receiver-Modus folgt damit folgende Prozedur :

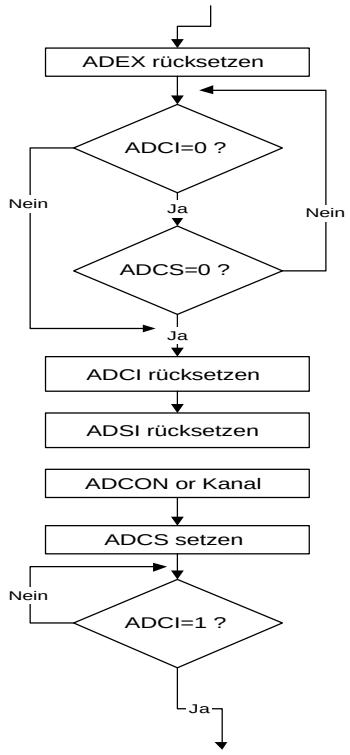


Struktogramm 20. Master-Receiver-Mode im I^2C-Bus

Beide Prozeduren arbeiten somit ohne Interrupt-Gebrauch.

4.7.8 AD-Wandlung

Die AD-Wandlung beansprucht 50 Taktzyklen, dauert also bei 12 MHz Systemtakt genau 50 µs.In der angebeben Konfiguration (siehe *3.3.7 Die Controllerplatine*) beträgt der Eingangsspannungsbreich 0-2.5V, entsprechend einem Wandlerwert von 0-1024. abgetastet wird nach dem Verfahren der suxessiven Approximation. Über die unteren 3 Bits des SFR ADCON wird der Kanal (0-7) ausgwählt.
Nach der Wandlung stehen die oberen 8 Bit des Ergebnisses im SFR ADCH. Die unteren 2 Bits liegen oben im SFR ADCON. Eine Wandlung wird durch die folgende Prozedur ermöglicht :

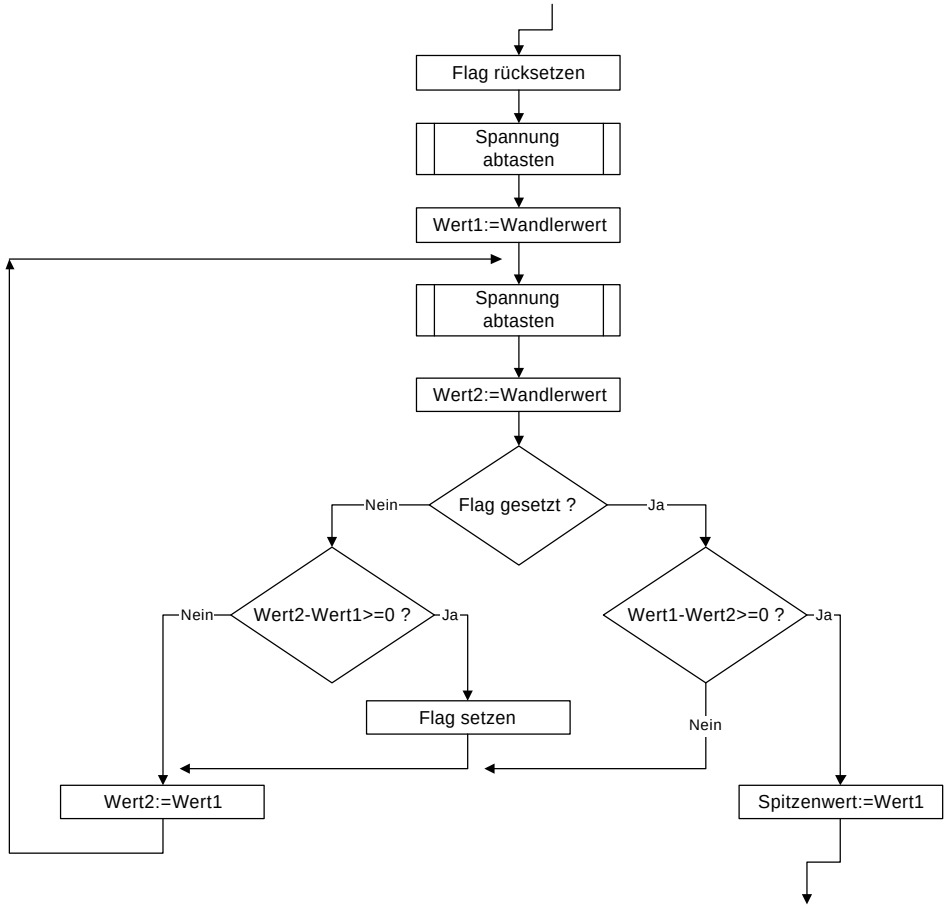


Struktogramm 21. Bedienung des AD-Wandlers

4.7.8.1 Spannungsmessung

Um über den AD-Wandler die Spitzenwerte der Eringangsspannung zu ermitteln, muß der Spannungsverlauf abgetastet und differentiert werden Aus der Abfolge positive Steigung ⇒ negative Steigung ergibt sich der Spitzenwert. Dazu werden zwei aufeianderfolgende Wandlerwerte voneinander abgezogen. Wird eine positive Steigung festgestellt, wird solange weiter abgetastet, bis sich wieder eine negative Steigung ergibt.

Der Spitzenwert ist dann derjenige, der zuletzt von dem Auftreten der negativen Steigung ermittelt wurde :



Struktogramm 22. Ermittelung des Spitzenwertes der Eingangsspannung

4.7.8.2 Kalibierung der Spannungsmessung

Durch die Exemplarstreuungen der Optokoppler muß die Spannungsmessung kalibiert werden. Hierzu wird die Eingangsspannung zunächst um 20% der Nennspannung gesenkt. Durch Messung dieser Sapnnung wird die Untere Grenze (ADMIN) festgelegt. Dieses Vorgehen wird mit einer um 20% erhöhten Eingangsspannung wiederholt und damit die obere Grenze der Eingangsspannung festgelegt (ADMAX).

Über die Form der Geradengleichung

$$Y = mX + b$$

werden die Parameter

$$m = \frac{ADMAX - ADMIN}{(Nennspannung + 20\%) - (Nennspannung - 20\%)}$$

und

$$b = \frac{ADMIN}{m * (Nennspannung - 20\%)}$$

ermittelt um später die Wandlerwerte in eine reale Spannung umrechnen zu können.

4.7.8.3 Ermittelung der realen Eingangsspannung

Die aus der Kalibierung gewonnen Geradenparamter ergeben nach Ermittelung des Spitzenwertes der Eingangsspannung die folgende reale Eingangsspannung :

$$Eingangsspannung := \frac{Wandlerwert - b}{m}$$

Ist der Wandlerwert ober- oder Unterhalb der +-20%-Grenzen, so muß ein Flag zur Erkennung einer Fehlmessung gesetzt werden.
Die maximale Slew-Rate des Eingangssignals des AD-Wandlers darf 10V/ms nicht überchreiten, um eine krasse Fehlmessung auszuschliessen. Ausgehend von einem sinusförmigen Eingangssignal mit

$$u(t) = U * \sin(v t + j)$$

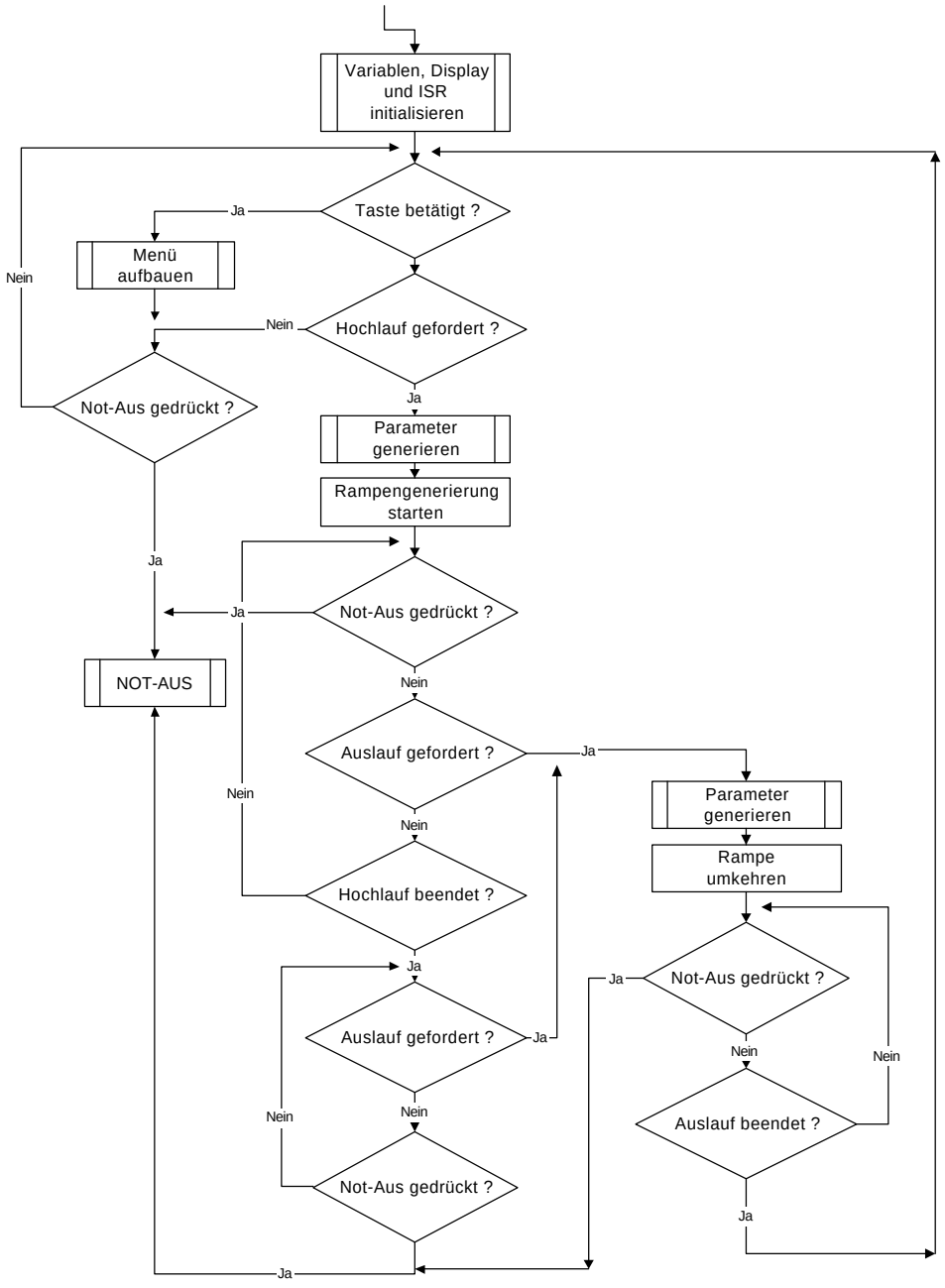
folgt für U=2.5V, $\omega=50s^{-1}$ und t=50µs für eine maximale Steigung im Nulldurchgang eine Spannungsänderung von

$$\frac{39{,}3mV}{50us} = 785V / s$$

Damit ist eine Wandlung ohne externes sample-and-hold-Glied möglich, zumal eine Bedämpfung der Eingangsspannung durch einen Kondendator am Wandlereingang gegeben ist.

4.8 Das Hauptptogramm

Da alle wichtigen Funktionen zur Zündimpulserzeugung in die ISR der Timer verlegt wurden, besteht die Aufgabe des Hauptprogrammes nur noch in der zyklischen Abfrage der Tastatur und den Steuerspannungseingängen, um bei Bedarf die entsprechenden Werte für den angeforderten Betriebszustand zu errechnen und das Menü zur Parametrierung zu erzeugen :

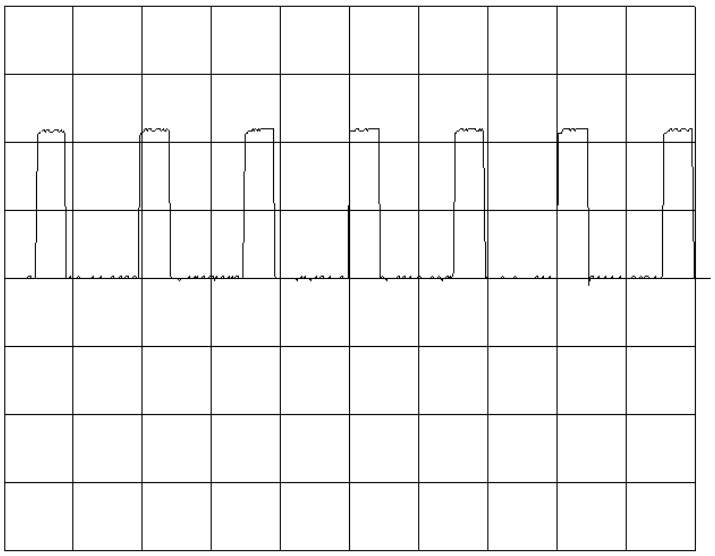


Struktogramm 32. Hauptprogramm der Anschnittsteuerung

5. Erprobung

5.1 Abgleich

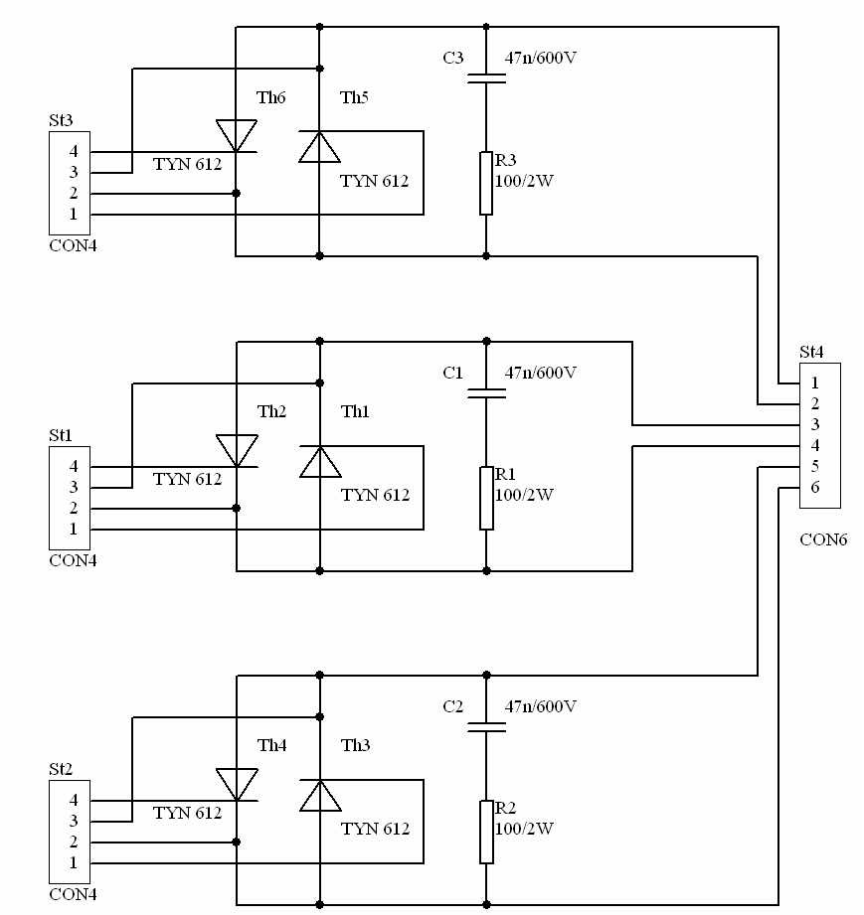
Der Abgleich des Moduls beschränkt sich auf die Einstellung der Takterzeugung um IC_{11}. Mit dem Trimmer P_1 wird die Pausenzeit (0 V) des Rechtecksignals an Pin 11 von IC_8 auf 20 µs und mit P_2 die Impulszeit (5+V) auf 10 µs eingestellt. Damit ist der Abgleich abgeschlossen.



Oszillogramm 1. Takterzeugung für Impulsübertrager. 20 µs/2 V / DIV

5.2 Inbetriebnahme

Um das Modul praxisnah erproben zu können, wurde es um eine kleine Thyristorendstufe ergänzt :



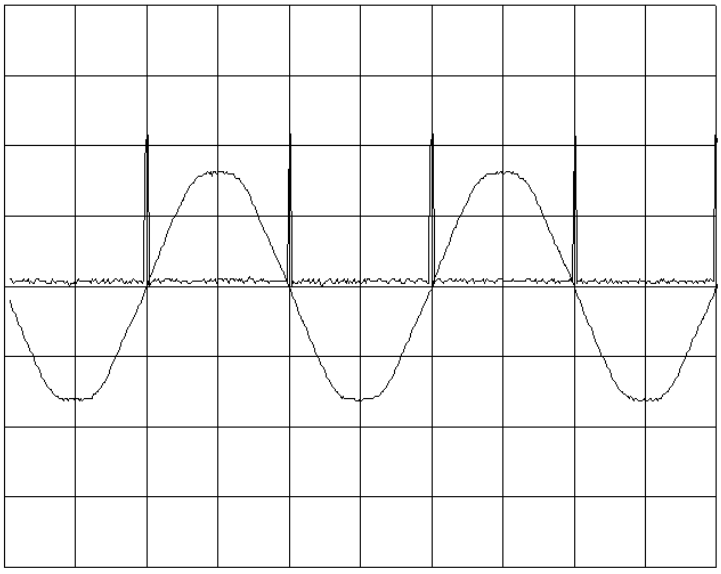
Schaltbild xy. Thyristorendstufe zur Erprobung

Diese kann direkt an die 3 Phasen und die Zündimpulsübertrager angeschlossen werden und erlaubt eine Leistungsübertragung von etwa 5 KW, sofern die Thyrsitoren nicht gekühlt werden.
Die RC-Glieder bilden einen einfachen Überspannungsschutz für die Thyristoren um ein „Über-Kopf-Zünden“ während der Kommutierung zu verhindern.
St_1-St_3 der Thyristorplatine werden direkt mit St_2, St_5 und St_6 der Hauptplatine verbunden. St_1 der Hauptplatine nimmt die 3 Phasen sowie den Nulleiter des Versorgungsnetzes auf.
St_3 steuert das Motor- sowie das Überbrückungsschütz an und über St_4 werden die Steuerspannungen für Not-Aus und die Steuerungskontrolle eingespeist.

5.2.1 Meßprotokolle

5.2.1.1 Nulldurchgangsdetektion

Der Nulldurchgangsdetektor liefert am INT0-Pin der Controllerplatine ein der Simulation entsprechechendes Signal :

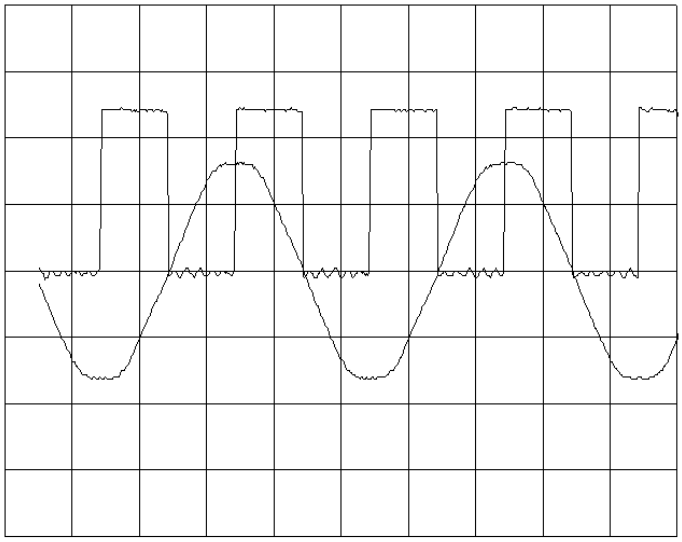


Oszillogramm 2. Ausgangssignal der Nulldurchgangsdetektion Ch1=200V/Div, Ch1=2 V/Div, 5 ms/Div

Die Sinuskurve von Kanal 1 zeigt den Verlauf der Netzspanung von L_1 gegen N, Kanal 2 die Impulsflanken beim Nulldurchgang (Pin 4 IC_3 der Hauptplatine).

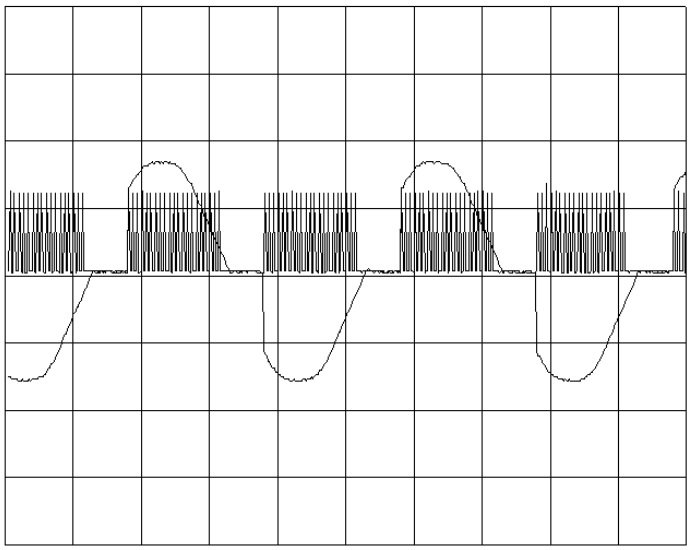
5.2.1.2 Ansteuerung der Takterzeugung

Die Takterzeugung zur Ansteuerung der Thyristoren erfolgt über die Portpins P_0-P_2 der Controllerplatine an IC_8. Für einen Anschnittwinkel von 90° elektrisch wurden an diesen 3 Pins folgende Spannungen gemessen :



Oszillogramm 3. Ansteuerung der Takterzeugung. Ch1=200V/Div, Ch2=2V/Div, 5 ms/Div

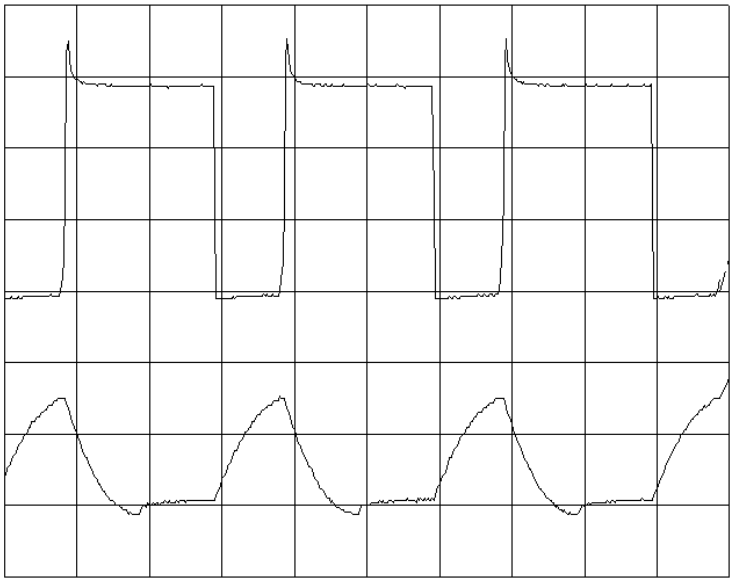
Die sinusförmige Spannung ist die Spannung der jeweiligen Phase, zu der die Impulserzeugung über IC_8 der Hauptplatine an- (Pegel=High) oder abgeschaltet (Pegel=Low) wird.
An den Eingängen der Impulsübertrager-Endstufen ergibt sich damit für jede der drei Phasen das folgende Bild. Erkennbar ist das Anschalten der Impulse kurz vor dem Nulldurchgang der Phase :



Oszillogramm 4. Takterzeugung an den Übertrager-Endstufen Ch1=200V/Div, Ch2=2V/Div, 5 ms/Div

5.2.1.3 Zündstromerzeugung

Die Zündendstufen müssen den Gatestrom für die Thyristoren liefern, um diese sicher zu zünden. Das folgende Oszillogramm zeigt im oberen Breich die Drain-Source-Spannung eines der MosFet´s und im unteren Bereich den zugehörigen Gatestrom :



Oszillogramm 5. Gatestrom der Thyristoren. Ch1=10V/Div, Ch2=400mA/Div, 10µs/Div

Erkennbar ist die zu geringe Steigung der Stromimpulse beim Einschalten des Transistors. Zum Teil wird dies verursacht durch die langen Zuleitung vom Übertrager zum Thyristor. Für den gegebenen Aufbau sind die Impulshöhe und die Steilheit jedoch ausreichend.

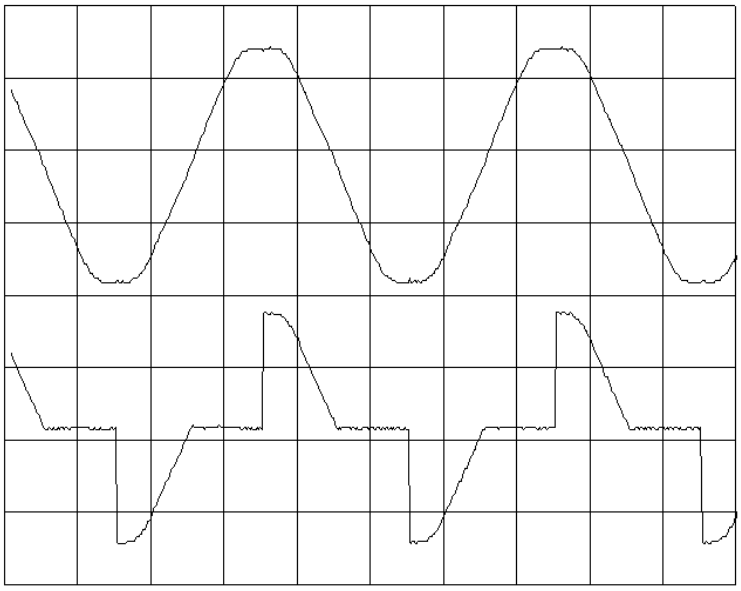
Aus dem Aufbau ergibt sich jedoch eine weitere Schwierigkeit. Der Controller schaltet über die Und-Gatter (IC_8) die Zündimpulse ein und aus, ohne die Impulslängen zu berücksichtigen. Dadurch können beim Schaltvorgang kurze Impulse erzeugt werden :

Oszillogramm 6. Angeschnittener Zündimpuls. Einstellungen wie Oszillogramm 5

Die hierdurch entstehenden Zündstrome können unter Umständen zu niedrig sein, um den Thyristor sicher durchzusteuern. Dies kann seine Zerstörung zur Folge haben. Für eine weitere Entwicklung muß diesem Umstand Rechnung getragen werden.
Zum Beispiel können die Steuerimpulse von IC_8, Pin 11 auf einem freien Portpin des Controllers eingekoppelt werden. Dieser muß dann, bevor in den Interrupt-Routinen die Zündimpulse ein- oder ausgeschaltet werden, solange warten, bis das Signal logisch 0 wird. Im schlimmsten Fall bedeutet dies eine Wartezeit von 10 µs.

5.2.1.4 Phasenanschnitt gegen Neutralleiter

Mit einer ohmschen Last von 100 Ω gegen den Neutralleiter wurden an allen 3 Ausgängen der Thyristorendstufen (U,V,W) identische Stromverläufe für einen Anschnittwinkel von 90° elektrisch gemessen :

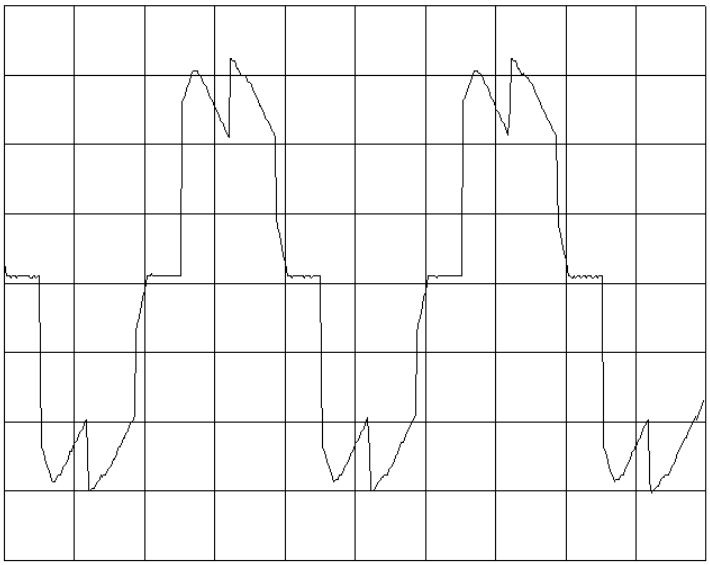


Oszillogramm 7. Phasenanschnitt mit ohmscher Last gegen N, Ch1=Ch2=200V/Div, 5ms/Div

Die obere Kurve zeigt den Verlauf der jeweiligen Phase des Netzes, die untere den Spannungsverlauf an der Last.

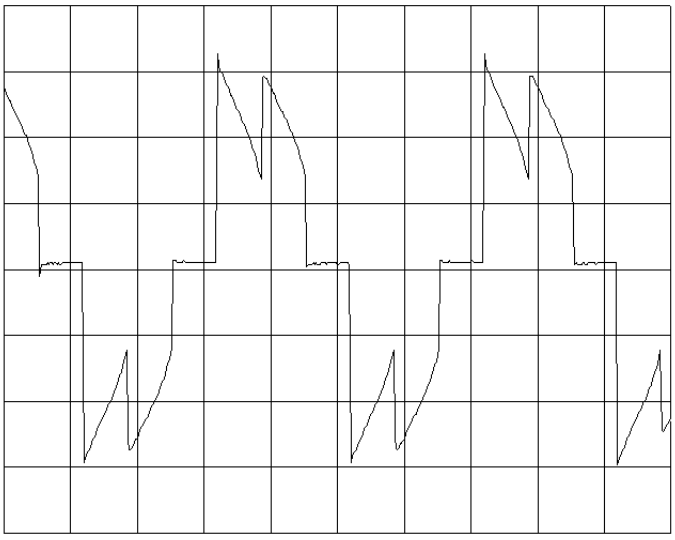
5.2.1.5 Phasenanschnitt mit symmetrischer Last

Mit einer symmetrischen, sternförmigen, ohmschen Last von jeweils 100 Ω ergeben sich die bereits aus der Simulation bekannten Lastströme. Für einen Anschnittwinkel von 45° :



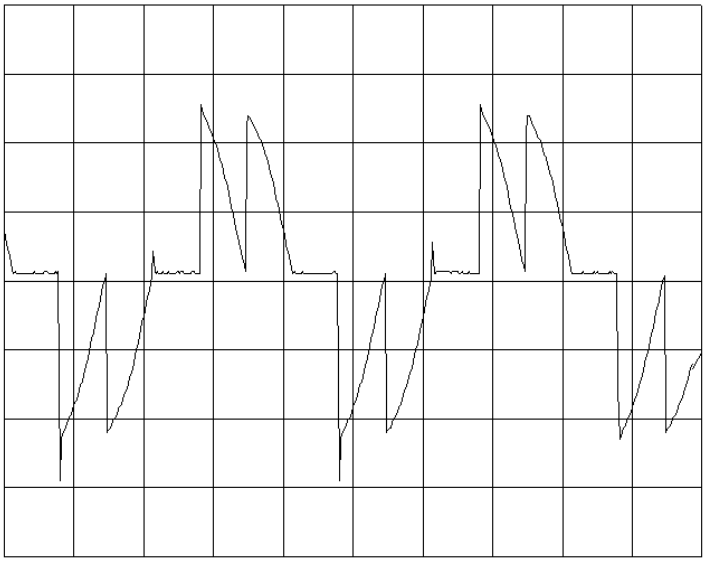
Oszillogramm 8, Laststrom für Anschnittwinkel φ=45°, Ch1=100 V/Div, 5ms/Div

Für einen Anschnittwinkel von 65° ergibt sich der folgende Laststrom :



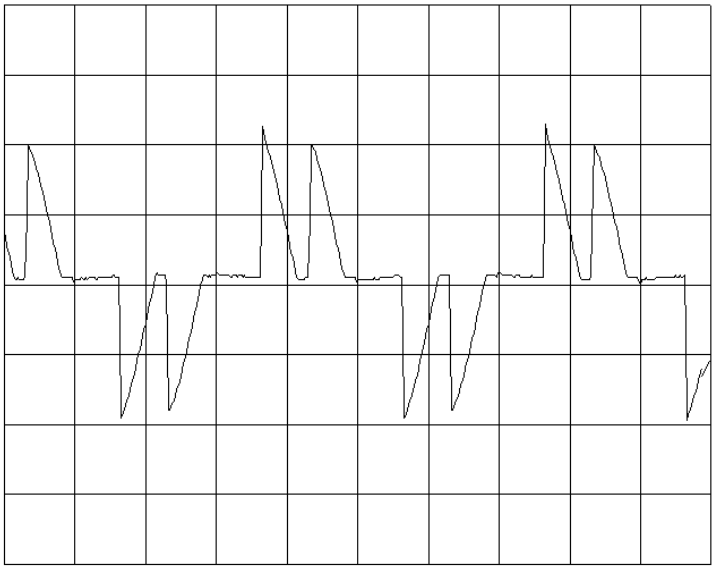
Oszillogramm 9, Laststrom für Anschnittwinkel φ=65°, Ch1=100 V/Div, 5ms/Div

Für einen Anschnittwinkel von 90° ergibt sich :



Oszillogramm 10, Laststrom für Anschnittwinkel φ=90°, Ch1=100 V/Div, 5ms/Div

Bei 105° ergibt sich :



Oszillogramm 12, Laststrom für Anschnittwinkel φ=105°, Ch1=100 V/Div, 5ms/Div

Anhang

A Literatur

Bücher

[101] Erich Eder - Stromrichter zur Drehzahlsteuerung von Drehfeldmaschinen Teil 2, Siemens, 1974
[102] Wolfgang Keuter, Das Stellen und Schalten von Wechselgrößen Hüthig Verlag 1982
[103] M. Meyer, Leistungselektronik, Einführung, Grundlagen, Überblick Springer Verlag 1990
[104] 305 Schaltungen, 80C552-Microcontroller, Elektor Verlag 1996
[105] 302 Schaltungen, Richtungsanzeiger, Schaltung 123, Elektor Verlag
[106] Wirsum, Optoelektronik, Franzis Praxisbuch, Franzis Verlag 1990
[107] Otmar Feger, Die 8051-Mikrocontrollerfamilie, Markt&Technik 1987
[108] Andreas Roth, Das Mikrocontrollerkochbuch, IWT 1989

Zeitschriften

[201] Elektor Februar 1990, Entwicklungshelfer mit 8032, Seite 60ff
[202] Elektor April 1991, 8052-Compuboard, Seite 17ff

Datenblätter

[301] 80C552 Single Chip Microcontroller, Phillips IC20, 1998
[302] PCA 8581 128*8-Bit EEPROM with I^2C-Bus-Interface, Phillips IC12, 1997
[303] PCF 8574 8-Bit I/O-Expander for I^2C-Bus, Phillips IC20, 1997
[304] LTN 111R-10, Datenblatt Toshiba LCD-Module

Sonstige Schriften

[401] Mini-LCD-FAQ, http://www.hut.fi/~electonics
[402] Interfacing to an LCD-Module, Microchip AppNote AN 587, 1996
[403] Using the ADC of the 8XC552-Microcontroller, Phillips AN93017 1994
[404] 80C51 family programmers guide, Phillips 1997
[405] 80C51 family hardware description, Phillips 1997
[406] 80C51 family derivates, Phillips 1996
[407] The I^2C-Bus and how to use it, Phillips 1995

B Listings der Simulationen

Simulation einphasiger Anschnittsteller mit ohmscher Last

```
Simulation Einphasensystem, ohmsche Last, angeschnitten

* Einschwinganalye
.tran 10u 60m 0 10u UIC
.four 50 3 i(rlast)
.lib eval.lib

* Speisespannung
v1 0 10 sin (0 325 50 0 0 180)

* Lastwiderstand
rlast 20 0 100

* Stellerteil
s1 10 20 30 0 schalter
.model schalter vswitch (von=0.5 voff=0.1 ron=10m ROFF=800K)

* Steuerspannung fuer den Steller
vsteuer 30 0 PULSE 0 10 5m 1n 1n 5m 10m

* Grafische Ausgabe der Ergebnisse
.probe

.end
```

Simulation einphasiger Anschnittsteller mit ohmsch-induktiver Last

```
Simulation Einphasensystem, ohmsch-induktive Last, angeschnitten

* Einschwinganalye
.tran 10u 60m 0 10u UIC
.inc scrneu.mod

* Speisespannung
v1 0 10 sin (0 325 50 0 0 180)

* Lastwiderstand und Induktivität
rlast 20 40 100
rkor 40 0 1e9
l1 40 0 0.318


* Shunt zur Strommessung
rmess 10 0 100

* Stellerteil
x1 10 20 11 12 ac_schalter

* Steuerspannung fuer den Steller
vsteuer 11 12 PULSE 0 10 2.5m 1n 1n 5m 10m

* Unterschaltung Leistungsschalter
*                  Eing. Ausg. +  -
.subckt ac_schalter  1    2    3  4
d1 1 5 dmod
d2 2 5 dmod
```

```
d3 4 2 dmod
d4 4 1 dmod
x1 5 3 4 scrneu
.model dmod d
.ends

* Grafische Ausgabe der Ergebnisse
.probe

.end
```

Simulation dreiphasiger Anschnittsteller mit ohmscher Last

```
2. Simulation Drehstromsystem, ohmsche Last, Dreieck, angeschnitten

* Einschwinganalye
.tran 10u 60m 0m 10u UIC
.lib eval.lib
.inc d:\diplom\simula\sim3\scrneu.mod

* Angabe des Steuerwinkels in Grad
.param alpha=125

* Umrechnung der Steuerwinkel in eine jeweilige Zeitangabe
.param verzoeg1={1m*alpha/18}
.param verzoeg2={6.666m+1m*alpha/18}
.param verzoeg3={6.666m+6.666m+1m*alpha/18}

* Pulsbreite der Ansteuerimpulse
.param pulsbreite={10m-verzoeg1-0.1m}

* Spannungsquellen, erzeugen dreiphasiges 400V-Drehstomsystem
vl1 0 10 sin (0 325 50 0 0 180)
vl2 0 20 sin (0 325 50 0 0 60)
vl3 0 30 sin (0 325 50 0 0 -60)

* Passive, ohmsche Dreieckschaltung mit jeweils 100 Ohm
*r1 40 50 100
*r2 50 60 100
*r3 40 60 100

* Passive, ohmsche Sternschaltung mit Mittelleiter mit jeweils 100 Ohm
* r1 40 0 100
* r2 50 0 100
* r3 60 0 100

* Passive, ohmsche Sternschaltung ohne Mittelleiter mit jeweils 100 Ohm
r1 40 70 100
r2 50 70 100
r3 60 70 100

* Schalterteil
x1 10 40 11 12 ac_schalter
x2 20 50 21 22 ac_schalter
x3 30 60 31 32 ac_schalter

* Steuerteil, Impulse jeweils 120° verzögert, Konstante=6.6ms

* Steuerspannung Strang 1
vs1 11 12 PULSE 0 100 {verzoeg1} 10n 10n {pulsbreite} 10m
* Steuerspannung Strang 2
vs2 21 22 PULSE 0 100 {verzoeg2} 10n 10n {pulsbreite} 10m
```

```
* Steuerspannung Strang 3
vs3 31 32 PULSE 0 100 {verzoeg3} 10n 10n {pulsbreite} 10m

* Unterschaltung Leistungsschalter
*                  Eing. Ausg. +  -
.subckt ac_schalter  1    2    3  4
d1 1 5 dmod
d2 2 5 dmod
d3 4 2 dmod
d4 4 1 dmod
x1 5 3 4 scrneu
.model dmod d
.ends


* Grafische Ausgabe der Ergebnisse
.probe

.end
```

Simulation Spannungsmessung über Optokoppler

```
* Simulation einer Netzspannungsmessung über einen Optokoppler

* Einschwinganalye
.tran 10u 70m 0m 10u UIC
.lib c:\mseval61\eval.lib

* Spannungsquelle, erzeugt 230 V AC
v1 0 10 sin (0 325 50 0 0 180)

* Primärseite
c1 10 20 47n
r1 20 30 47
d1 0 20 D1N750

* Optokoppler
x1  30    0    70   60   50 A4N25


* Zur Berechnung notwendige galvanische Verbindungen (hochohmig)
r4 0 40 1e6
r5 40 50 1e6

* Sekundärseite
r2 70 40 680
c2 80 40 1u
r3 80 40 10k
d2 70 80 D1N4148

* 5V-Versorgung
v2 60 40 DC 5

* Grafische Ausgabe der Ergebnisse
.probe

.end
```

Simulation Nulldurchgangsdetektor

```
Simulation eines Nulldurchgangsetektors

* Einschwinganalye
.tran 10u 70m 0m 10u UIC
.lib c:\mseval61\eval.lib

* Spannungsquelle, erzeugt 230 V AC
v1 0 10 sin (0 325 50 0 0 180)

* Spannungsteiler
r1 10 20 47k
r2 20 0 2.2k
r3 20 30 4.7k

* Zur Berechnung notwendige galvanische Verbindungen (hochohmig)
r5 0 40 1e-6

* Schaltungsteil
r4 80 60 470

q1 50 30 40 npn_tran
q2 40 70 50 pnp_tran
q3 70 40 30 npn_tran

.model npn_tran NPN
.model pnp_tran PNP

* 5V-Versorgung
v2 80 40 DC 5

* Optokoppler
x1 60 50  40 90 110  A4N25
r6 110 40 1e6
r7 90 80 1k

* 6-pin DIP: pin #1   #2   #4   #5   #6

* Grafische Ausgabe der Ergebnisse
.probe

.end
```

C Stücklisten der Platinen

Stückliste der Controllerplatine :

Halbleiter :

IC1	80C552
IC2	27256
IC3	62256
IC4	74HC373
IC5	4049
IC6	MAX232
IC7	AD 580

Widerstände :

R1	4.7K,8-fach Array
R2	10k, 1/4 W
R3	10k, 1/4 W
R4	8.2k, 1/4 W
R5	4.7k /1/4 W

Kondensatoren :

C1	33p
C2	33p
C3	10u
C4	10u
C5	10u
C6	10u
C7	4.7u
C8	100n
C9	100n
C10	100n
C11	100n

Steckverbinder :

St1-St4	CON10, Pfosten

Sonstiges :

X1	12.000 MHz
J1	Jumper

Stückliste der Hauptplatine :

Halbleiter :

D1/D2	ZD 36V
D3/D4	1N4004
D4	1N4004
D5	ZD12
D6-D8	1N40148
D9-D11	ZD4.7
D12-D14	36V
D15-D17	1N4002
D18-D20	1N4003
D21-D23	1N4001
D24-D26	15V
D27-D29	BAT41
D30-D34	1N4002

Br1 B40C5000

IC1	7805
IC2	7812
IC3-IC7	PC817
IC8	4081
IC9	PCF8574
IC10	S201S02
IC11	4066
IC12	PC817
IC13	S201S02
IC14	PCF8574
IC15	PCF8574
IC16	LTN211R-10
IC17	EPC552, Einplatinencomputer
IC18	PCA8581

T1-T6	BC547
T7-T9	IRFD120

Kondensatoren :

C1	100n
C2	1000u/40V
C3-C6	10n
C7	220n/400V
C8	220u/16V
C9-C11	220n/400V
C12-C14	1u/16V
C15	10n
C16	2.2n
C17-C19	100n
C20-C22	1000u/40V
C23	100n
C24	47n/400V
C25	100n
C26	47n/400V
C27-C33	100n

Widerstände :

R1/R2	100/2W
R3	0.22
R4	47/2W
R5	47k
R6	2.2k
R7	4.7k
R8	100
R9	1k
R10-R12	100
R13-R15	680
R16-R18	10k
R19/R20	1k
R21	10k
R22	100k
R23-R25	1k
R26-R28	10k
R29-R31	220
R32/R33	2.2k
R34-R37	1k
R38/R39	220
R40	4.7k
P1/P2	22k
P3	4.7k

Klemmverbinder :

St1	NETZ
St2	Thy-L1
St3	SCHÜTZE
St4	STEUER
St5	Thy-L2
St6	Thy-L3

Taster :

Sw1-Sw9	DigiTast

Transformatoren :

Tr1	230V/24V/16 VA
Tr2-Tr4	ZKB472/106

Sonstiges :

VDR1	VARISTOR
F1	FILTER

D Schaltpläne

Schaltplan der Controllerplatine :

Schaltplan der Hauptplatine :

E Platinenlayouts

Layout der Controllerplatine :

Layout der Hauplatine :

F Listing des Hauptprogramms

```
; ************************************************************************
; * Diplomarbeit Christian Finger                                        *
; *                                                                      *
; * Wintersemster 99/00                                                  *
; *                                                                      *
; * Sanftanlauf, 3 Phasen                                                *
; *                                                                      *
; ************************************************************************
;
; Prozessor auswählen
$MOD552
;
; Titelzeite für LST-Datei angeben
$TITLE (Sanftanlauf)

; Keine Symboldatei erzeugen
$NODEBUG


; ********************************************************************
; * Variablen-Deklarationen                                          *
; ********************************************************************

; Serielle Kommunikation (Adressen)
;----------------------
ser_puffer            EQU  80h  ; Sende- und Empf.puffer (16 Bytes Länge 80-90)
ser_anzahl            EQU  91h  ; Anzahl zu bearbeitender Bytes
ser_slave_adr         EQU  92h  ; Slave-Adresse
ser_flag              EQU  93h  ; Kommunikations-Flagregister

; Slave-Adressen (Werte)
;----------------
display_adresse       EQU  64   ; Adresse des Display-Slaves
tastatur_adresse      EQU  66   ; Adresse des Tastatur-Slaves
io_adresse            EQU  68   ; Adresse des IO-Slaves
eeprom_adresse        EQU  160  ; Adresse des EEProms

; Mathematische Variablen  (Adressen)
;------------------------
byte_puffer1          EQU 9Ah   ; Puffer für 1 Byte
byte_puffer2          EQU 9Bh   ; Puffer für 1 Byte
byte_puffer3          EQU 9Ch   ; Puffer für 1 Byte
obergrenze            EQU 9dh   ; Max. Einstellbarer Wert des aktuellen Par.
untergrenze           EQU 9eh   ; Min. Einstellbarer Wert des aktuellen Par.
schrittweite          EQU 9Fh   ; Schrittweite des aktuellen Par.

; Variablen zur Steigungsberechnung
;----------------------------------
startwert             EQU 160  ; Startwert der Rampe (A0)
endwert               EQU 161  ; Endwert der Rampe   (A1)
dauer                 EQU 162  ; Dauer der Rampe     (A2)
steigungszaehler_msb  EQU 163  ; Zähler für Steigung MSB (A3)
steigungszaehler_lsb  EQU 164  ; Zähler für Steigung LSB (A4)
steigung_msb          EQU 165  ; Vergleichswert für Steigung MSB (A5)
steigung_lsb          EQU 166  ; Vergleichswert für Steigung LSB (A6)
bereich               EQU 167  ; Zu überstreichender Bereich der Rampe  (A7)
richtung              EQU 168  ; Flag für pos. oder neg. Steigung (A8)
endwert_flag          EQU 169  ; Flag für erreichten Endwert der Rampe  (A9)


; Wertübernahme (Adressen)
;----------------

tastatur_wert         EQU 94h  ; Empfangspuffer Tastatur
io_wert               EQU 95h  ; Empfamgspuffer IO

; Menüdefinitionen (Adressen)
;-----------------
```

```
menue_offset          EQU 6000h ; Adresse der Menütabellen
anzahl_ebene1         EQU   96h ; Anzahl der Begriffe in Ebene 1
anzahl_ebene2         EQU   97h ; Anzahl der Begriffe in Ebene 2
menue_pos1            EQU   98h ; Aktuelle Pos. in Ebene 1
menue_pos2            EQU   99h ; Aktuelle Pos. in Ebene 2


; Anschnitterzeugung (Adressen)
;------------------------------
zuendwinkel           EQU  70h  ; Aktueller Zündwinkel (direkt adressierbar)
t1_flag               EQU  71h  ; Flag für T1
t2_flag               EQU  72h  ; Flag für T2
timer_low             EQU  73h  ; Timerladewert Low-Byte
timer_high            EQU  74h  ; Timerladewert High-Byte

; Anschnittsteuerung (Werte)
;------------------------------
l1                    EQU 1     ; Flagwert für L1
l2                    EQU 2     ; Flagwert für L2
l3                    EQU 4     ; Flagwert für L3

; Systemuhr (Adressen)
;----------------------
ticker                EQU  75h  ; Wird alle 10ms incrementiert

; Rechnungspuffer (Adressen)
; Vor allem für die Divisionsroutine
;-----------------------------------

ergebniss_msb         data 60h
ergebniss_lsb         data 61h

arbeitsreg_msb        data 62h
arbeitsreg_lsb        data 63h

bitzaehler            data 64h
zwischensp            data 65h

dividend_msb          data 66h
dividend_lsb          data 67h

divisor_msb           equ  68h
divisor_lsb           equ  69h

; ********************************************************************
; * Reset-Einsprung                                                  *
; ********************************************************************

ORG     0000H                         ; Controller startet bei 0
ljmp    programm_start                  ; Sprung zum Hauptprogramm

; ********************************************************************
; * Interrupt-Einsprungadressen                                      *
; ********************************************************************

ORG 0003H                               ; Einsprung für INT0
ljmp isr_int0                           ; Sprung zur ISR

ORG 000BH                               ; Einsprung für T0
ljmp isr_T0                             ; Sprung zur ISR

ORG 0013H                               ; Einsprung für INT1
ljmp isr_int1                           ; Sprung zur ISR

ORG 001BH                               ; Einsprung für T1
ljmp isr_T1                             ; Sprung zur ISR

ORG 005BH                               ; Einsprung für T2 Compare CM0
ljmp isr_T2                             ; Sprung zur ISR

; ********************************************************************
; * Interrupt-Service-Routinen
; ********************************************************************

$INCLUDE (isr.asm)

; ********************************************************************
; * Start des Hauptprogrammes                                        *
```

```
; ********************************************************************
ORG    200H                         ; Start nach I-Einsprungtabelle

programm_start:                     ; Hier beginnt alles

               mov sp,#17h          ; Stackpointer raufsetzen damit
                                    ; 3 Registerbänke zur Verfügung stehen

               lcall display_init   ; Display initialiseren
               lcall isr_init       ; ISR initialisieren
haupt_start:   lcall startmeldung   ; Startmeldung ausgeben

               ; Jetzt kann die Tastatur abgefragt werden

hauptprog_tast:
               ; Tastatur adressieren und auslesen
               ; Anzahl zu lesender Bytes festlegen
               mov r0,#ser_anzahl       ; r0 mit Adresse laden
               mov @r0,#1               ; Anzahl auf 1 festlegen

               ; Adresse der Tastatur angeben
               mov r0,#ser_slave_adr    ; r0 mit Adresse laden
               mov @r0,#tastatur_adresse  ; Adresse festlegen

               ; Daten reinholen einlesen
               lcall I2C_master_receiver

               ; Ist der Tastaturwert=0 ?
               mov r0,#ser_puffer       ; r0 mit Adresse d. Puffers laden
               mov a,@r0                ; Tastaturwert in a schieben
               jz hauptprog_io          ; Wenn Null, Tastatur links
                                        ; liegen lassen

               ; Hier gelandet war da ein Tastendruck
               mov r4,a             ; Taste in r4 bunkern

               ; Jetzt noch einmal warten, bis der Dummuser die
               ; außerordentliche Gnade erweist, seinen fettigen
               ; Finger wieder von der Taste zu nehmen....
               ; seine unegalen Wixgriffel von der Taste genommen hat

hauptprog_tast_nicht:

               ; Daten reinholen einlesen
               lcall I2C_master_receiver

               ; Ist der Tastaturwert=0 ?
               mov r0,#ser_puffer          ; r0 mit Adresse d. Puffers laden
               mov a,@r0                   ; Tastaturwert in a schieben
               jnz hauptprog_tast_nicht    ; Wenn <> Null, Tastatur links
                                           ; liegen lassen

               ; Jetzt kann die Abfrage der Tasten erfolgen

               ; OK gedrückt für ein Menü ?
               mov a,r4             ; Taste in a schieben
               anl a,#1             ; mit 1  (OK) AND
               jz hauptprog_io      ; OK gedrückt -> Menü anzeigen

               ; Hier gelandet soll das Menü aufgebaut werden
               lcall generiere_menue    ; Menü aufbauen
               lcall startmeldung       ; Wieder den Starttext ausgeben

               ; Ab hier erfolgt die Abfrage der IO-Klamotten
hauptprog_io:
               ; Anzahl festlegen
               mov r0,#ser_anzahl       ; r0 mit Adresse laden
               mov @r0,#1               ; Anzahl auf 1 festlegen
               ; Adresse des IO-Slaves angeben
               mov r0,#ser_slave_adr    ; r0 mit Adresse laden
               mov @r0,#io_adresse      ; Adresse festlegen
               ; Und den IO-Slave abfragen
               lcall  I2C_master_receiver

               ; Jetzt sind die Portdaten da (P1=Lauf, P0=Not-Aus)
               mov r0,#ser_puffer          ; r0 mit Adresse d. Puffers laden
               mov a,@r0                   ; IOwert in A schieben
               mov r4,a                    ; A in r4 bunkern
```

```
                ; Abfrage, ob evtl. Not-Aus
                mov a,r4                    ; A wiederherstellen
                anl a,#1                    ; P0 ausmaskieren
                jnz hauptprog_hoch          ; Wenn P0=1 -> Nix machen

                ; Hier gelandet Not-Aus auslösen
                lcall not_aus

hauptprog_hoch:

                ; Abfrage, ob evtl. Hochlauf
                mov a,r4                    ; A wiederherstellen
                anl a,#2                    ; P1 ausmaskieren
                jz hauptprog_tast           ; Wenn P1=0 -> Nix machen

                ;-------------------------------------------------------
                ; Hier gelandet ist P1=1, das heißt Hochlauf angefordert..
                ;-------------------------------------------------------

                ; Zuerst feststellen, ob das Motorschütz eingeschaltet werden
                ; soll, dazu den Wert aus dem EEPROM holen (Adresse 32)

                mov a,#32                   ; Adresse 32 in a schieben
                lcall get_eeprom_data       ; Wert aus dem EEPROm holen

                ; Ist a.0=1, soll das Motorschütz eingeschaltet werden
                jnb acc.0,hauptprog_motorsch_nicht

                ; Hier gelandet soll das Schütz eingeschaltet werden

                ; Anzahl festlegen
                mov r0,#ser_anzahl        ; r0 mit Adresse laden
                mov @r0,#1                ; Anzahl auf 1 festlegen

                ; Adresse des IO-Slaves angeben
                mov r0,#ser_slave_adr     ; r0 mit Adresse laden
                mov @r0,#io_adresse       ; Adresse festlegen

                mov r0,#ser_puffer          ; r0 mit Adresse d. Puffers laden
                mov @r0,#127                ; P7 auf 0 setzen
                lcall i2c_master_transmitter ; Und ausgeben

hauptprog_motorsch_nicht:

                ; Falls Schnellstart gefordert, wird die Anlauframpe
                ; übersprungen
                mov a,#0                    ; Adresse 0 in a schieben
                lcall get_eeprom_data       ; Wert aus dem EEPROm holen

                ; Ist a.0=1, ist Schnellstart angesagt
                jnb acc.0,hauptprog_kein_schnellstart

                ; Hier gelandet ist Schnellstart angesagt, also Zündwinkel
                ; gleich auf Endwert des Hochlaufs, keine Rampe

                ; Endwertflag setzen (keine Rampe)
                mov r0,#endwert_flag            ; r0 als zeiger
                mov @r0,#1                      ; Flag setzen
                mov a,#18                       ; Adresse Endwert Hochlauf
                lcall get_eeprom_data           ; Wert rausholen

                ; Jetzt aus dem Wert den Winkel generieren
                mov b,#2                ; Divisor:=2
                mov r0,a                ; A in r0 bunkern
                div ab                  ; a/b
                add a,r0                ; Und den Startwert dazu
                mov r0,a                ; Und wieder speichern
                mov a,#150              ; 150 in A schieben
                clr cy                  ; Carry-Flag rücksetzen
                subb a,r0               ; Startwert abziehen
                mov zuendwinkel,a       ; Und reinschieben

                ; Jetzt läuft die Sache....
                ljmp hauptprog_hoch_ende

hauptprog_kein_schnellstart:
```

```
                ; Jetzt feststellen, ob ein Kickstart (8) vorgesehen ist
                mov a,#8                  ; Adresse 8 in A
                lcall get_eeprom_data      ; Wert aus dem EEPROm holen

                ; Ist a.0=1, ist Kickstart vorgesehen
                jnb acc.0,hauptprog_kickstart_nicht

                ; Hier gelandet ist ein Kickstart vorgesehen
                lcall kick_meldung        ; Meldung ausgeben

                ; Zuerst die Dauer des Kickstart aus dem EEPROM holen
                mov a,#9                  ; Adresse 9 in a
                lcall get_eeprom_data     ; Wert aus dem EEPROm holen

                ; Jetzt diesen Wert*100
                lcall bin2bcd             ; Wert aufspalten
                mov a,r6                  ; Vorkommawert in A
                clr cy                    ; Carry löschen
                subb a,#48                ; Von ASCII in BIN wandeln
                mov b,#100                ; 100 in B
                mul ab                    ; A*B
                mov r6,a                  ; Ergebniss in r6 bunkern
                mov a,r7                  ; Nachkommawert in A
                clr cy                    ; Carry löschen
                subb a,#48                ; Von ASCII in BIN wandeln
                mov b,#10                 ; 10 in B
                mul ab                    ; A*B
                add a,r6                  ; Ergebniss bilden
                mov r7,a                  ; Ergebniss in r7 bunkern

                ; Jetzt den Zündwinkel errechnen
                ; Zuerst den Wert aus dem EEPROM holen
                mov a,#10                 ; Adresse 10 in a
                lcall get_eeprom_data     ; Wert aus dem EEPROm holen
                ; Umrechnung von % in einen Zündwinkel  (*1.5)
                mov r6,a                 ; Wert in r6 bunkern
                mov b,#2                  ; Divisor:=2
                div ab                    ; a/b
                add a,r6                  ; Und den Wert dazu
                mov  r6,a                 ; in r6 bunkern
                mov a,#150                ; 15 in A
                clr cy                    ; Carry rücksetzen
                subb a,r6                 ; Winkel bilden (150°-wert*1.5)

                ; Flag für Endwert setzen, damit keine Rampengenerierung
                ; erfolgt
                mov r0,#endwert_flag            ; r0 als zeiger
                mov @r0,#1                      ; Flag setzen

                ; Uhr rückstellen
                ; Erst warten, bis ISR vorbei
                mov r7,ticker             ; Ticker in R7 bunkern
warte_isr0:     mov a,ticker              ; Ticker in A schieben
                clr cy                    ; Carry rücksetzen
                subb a,r7                 ; Ticker-Altwert
                jz warte_isr0             ; Warte, solange gleich

                clr ticker                ; ticker auf 0 setzen

                ; Impulserzeugung einschalten
                mov zuendwinkel,a         ; Zündwinkel setzen

hauptprog_kickstart:

                ; Wenn ticker=dauer (r7), ist der Kickstart beendet
                mov a,ticker              ; Ticker in A schieben
                clr cy                    ; Carry rücksetzen
                subb a,r7                 ; Dauer abziehen
                jnz hauptprog_kickstart   ; Warten, wenn <>

hauptprog_kickstart_nicht:

                ; Ab hier kann der normale Hochlauf generiert werden
                ; Zuerst die Parameter aus dem RAM holen
                mov a,#16                  ; Adresse Startwert
                lcall get_eeprom_data      ; Wert reinholen
                mov r0,#startwert          ; r0 als zeiger
```

```
                mov @r0,a                ; Wert übergeben

auslauf_abbruch:   ; Dieser EInsprung kommt, falls der Auslauf abgebrochen
                   ; und wieder hochgefahren werden soll

                mov a,#17                ; Adresse Dauer
                lcall get_eeprom_data    ; Wert reinholen
                mov r0,#dauer            ; r0 als zeiger
                mov @r0,a                ; Wert übergeben
                mov a,#18                ; Adresse Endwert
                lcall get_eeprom_data    ; Wert reinholen
                mov r0,#endwert          ; r0 als zeiger
                mov @r0,a                ; Wert übergeben

                ; Generierung der Parameter für die Rampe
                lcall rampen_param

                ; Jetzt kann der Hochlauf angeleiert werden
                lcall hochlauf_meldung     ; Sach ma Bescheid...

                ; Endwertflag rücksetzen
                mov r0,#endwert_flag           ; r0 als zeiger
                mov @r0,#0                     ; Flag rücksetzen

                ; Rampengenerierung starten
                mov r0,#startwert              ; r0 als Zeiger
                mov zuendwinkel,@r0            ; Startwinkel übergeben

hauptprog_hoch_rampe:

                ; In dieser Schleife wird ein mögliches wieder
                ; herunterfahren implementiert

                ; Zuerst die IO-Port abfragen

                ; Anzahl festlegen
                mov r0,#ser_anzahl       ; r0 mit Adresse laden
                mov @r0,#1               ; Anzahl auf 1 festlegen
                ; Adresse des IO-Slaves angeben
                mov r0,#ser_slave_adr    ; r0 mit Adresse laden
                mov @r0,#io_adresse      ; Adresse festlegen
                ; Und den IO-Slave abfragen
                lcall  I2C_master_receiver

                ; Jetzt sind die Portdaten da (P1=Lauf, P0=Not-Aus)
                mov r0,#ser_puffer         ; r0 mit Adresse d. Puffers laden
                mov a,@r0                  ; IOwert in A schieben
                mov r4,a                   ; A in r4 bunkern

                ; Abfrage, ob evtl. Not-Aus
                mov a,r4                   ; A wiederherstellen
                anl a,#1                   ; P0 ausmaskieren
                jnz hauptprog_hoch_1       ; Wenn P0=1 -> Nix machen

                ; Hier gelandet Not-Aus auslösen
                lcall not_aus

hauptprog_hoch_1:

                ; Jetzt kontrollieren, ob vielleicht Auslauf
                mov a,r4                   ; A wiederherstellen
                anl a,#2                   ; P1 ausmaskieren
                jnz hochlauf_schleife      ; Wenn P1=1 -> Nix machen

                ; Hier gelandet ist während des Hochlaufes wieder
                ; umgeschaltet worden, also wieder herunterfahren...

                ; Aus dem aktuellen Zündwinkel muß ein Startwert generiert
                ; werden (Startwert:=(150-Winkel)*2/3
                mov a,#150                  ; 150 in A schieben
                clr cy                      ; Carry rücksetzen
                subb a,zuendwinkel          ; 150-Winkel
                mov b,#2                    ; 2 in B
                mul ab                      ; A*B
                mov b,+3                    ; 3 in B
                div ab                      ; a/b
                mov r0,#startwert           ; r0 als Zeiger
                mov @r0,a                   ; Neuen Startwert speichern
```

```
                ljmp hochlauf_abbruch      ; Und abwärts....

hochlauf_schleife:

                ; Jetzt kontrollieren wir noch, ob der Hochlauf
                ; schon abgeschlossen ist...
                mov r0,#endwert_flag            ; r0 als zeiger
                mov a,@r0                       ; Flag in A schieben
                jz hauptprog_hoch_rampe         ; Warten bis fertig

hauptprog_hoch_ende:

                ; Jetzt ist der Hochlauf abgeschlossen
                lcall betrieb_meldung           ; Machn se Meldunk

                ; Eventuell muß jetzt noch das šberbrückungsschütz
                ; eingeschaltet werden
                ; Dazu den Wert aus dem EEPROM holen (Adresse 33)

                mov a,#33                   ; Adresse 41 in a schieben
                lcall get_eeprom_data       ; Wert aus dem EEPROm holen

                ; Ist a.0=1, soll das šberbrückungsschütz eingeschaltet werden
                jnb acc.0,hauptprog_brueck_nicht

                ; Hier gelandet soll das Schütz eingeschaltet werden

                ; Anzahl festlegen
                mov r0,#ser_anzahl        ; r0 mit Adresse laden
                mov @r0,#1                ; Anzahl auf 1 festlegen

                ; Adresse des IO-Slaves angeben
                mov r0,#ser_slave_adr     ; r0 mit Adresse laden
                mov @r0,#io_adresse       ; Adresse festlegen

                lcall i2c_master_receiver ; Portstatus reinholen
                mov r0,#ser_puffer          ; r0 mit Adresse d. Puffers laden
                mov a,@r0                   ; Port in A schieben
                anl a,#191                  ; P6 ausmaskieren
                mov @r0,a                   ; Zurückschieben
                lcall i2c_master_transmitter ; Und ausgeben

                ; Und weil jetzt das Schütz die Thyristoren überbrückt,
                ; können die Zündimpulse abgeschaltet werden
                mov zuendwinkel,#180

                ; Jetzt steht normaler Betrieb an, eigentlich muß
                ; jetzt nicht passieren, bis ein Auslauf angefordert wird

hauptprog_brueck_nicht:

                ; Also die IO-Ports abfragen....

                ; Anzahl festlegen
                mov r0,#ser_anzahl        ; r0 mit Adresse laden
                mov @r0,#1                ; Anzahl auf 1 festlegen

                ; Adresse des IO-Slaves angeben
                mov r0,#ser_slave_adr     ; r0 mit Adresse laden
                mov @r0,#io_adresse       ; Adresse festlegen

                lcall i2c_master_receiver ; Portstatus reinholen

                ; Jetzt sind die Portdaten da (P1=Lauf, P0=Not-Aus)
                mov r0,#ser_puffer          ; r0 mit Adresse d. Puffers laden
                mov a,@r0                   ; IOwert in A schieben
                mov r4,a                    ; A in R4 bunkern

                ; Abfrage, ob evtl. Not-Aus
                mov a,r4                    ; A wiederherstellen
                anl a,#1                    ; P0 ausmaskieren
                jnz hauptprog_betrieb_1     ; Wenn P0=1 -> Nix machen

                ; Hier gelandet Not-Aus auslösen
                lcall not_aus

hauptprog_betrieb_1:
```

```
                mov a,r4                    ; A wiederherstellen
                anl a,#2                    ; P1 ausmaskieren
                jnz hauptprog_brueck_nicht  ; Wenn P1=1 -> Nix machen

                ; Hier gelandet soll also ein Auslauf gestartet werden

                ; Erst einmal die Rampenparameter generieren
                mov a,#18                   ; Adresse Endwert Hochlauf
                lcall get_eeprom_data       ; Endwert reinholen
                mov r0,#startwert           ; r0 als Zeiger
                mov @r0,a                   ; Neuer Startwert

hochlauf_abbruch: ; Dieser Einsprung kommt, wenn der Hochlauf abgebrochen
                  ; wurde. In diesem Fall muß vom aktuellen Zündwinkel
                  ; wieder heruntergefahren werden

                mov r0,#endwert             ; r0 als Zeiger
                mov @r0,#0                  ; Neuer Endwert 0

                mov a,#24                   ; Adresse Auslaufdauer
                lcall get_eeprom_data       ; Wert holen
                mov r0,#dauer               ; r0 als Zeiger
                mov @r0,a                   ; A in Dauer schieben
                lcall rampen_param          ; Parameter generieren

                lcall auslauf_meldung       ; Meldung ausgeben

                ; šberbrückungsschütz abschalten
                ; Anzahl festlegen
                mov r0,#ser_anzahl        ; r0 mit Adresse laden
                mov @r0,#1                ; Anzahl auf 1 festlegen

                ; Adresse des IO-Slaves angeben
                mov r0,#ser_slave_adr     ; r0 mit Adresse laden
                mov @r0,#io_adresse       ; Adresse festlegen

                lcall i2c_master_receiver ; Portstatus reinholen
                mov r0,#ser_puffer          ; r0 mit Adresse d. Puffers laden
                mov a,@r0                   ; Port in A schieben
                orl a,#64                   ; P6 ausmaskieren
                orl a,#1
                orl a,#2
                mov @r0,a                   ; Zurückschieben
                lcall i2c_master_transmitter ; Und ausgeben

                ; Neuen Zündwinkel übergeben
                mov r0,#startwert           ; r0 als Zeiger
                mov zuendwinkel,@r0         ; Winkel:=Startwert

                ; Endwertflag rücksetzen, Startet neue Rampe
                mov r0,#endwert_flag            ; r0 als zeiger
                mov @r0,#0                      ; Flag rücksetzen

hauptprog_ausl_rampe:

                ; Zuerst die IO-Port abfragen

                ; Anzahl festlegen
                mov r0,#ser_anzahl        ; r0 mit Adresse laden
                mov @r0,#1                ; Anzahl auf 1 festlegen
                ; Adresse des IO-Slaves angeben
                mov r0,#ser_slave_adr     ; r0 mit Adresse laden
                mov @r0,#io_adresse       ; Adresse festlegen
                ; Und den IO-Slave abfragen
                lcall  I2C_master_receiver

                ; Jetzt sind die Portdaten da (P1=Lauf, P0=Not-Aus)
                mov r0,#ser_puffer          ; r0 mit Adresse d. Puffers laden
                mov a,@r0                   ; IOwert in A schieben

                ; Abfrage, ob evtl. Not-Aus
                anl a,#1                    ; P0 ausmaskieren
                jnz hauptprog_ausl_1        ; Wenn P0=1 -> Nix machen

                ; Hier gelandet Not-Aus auslösen
                lcall not_aus
```

```
hauptprog_ausl_1:

               ; Hier gelandet ist während des Hochlaufes wieder
               ; umgeschaltet worden, also wieder herunterfahren...

               ; Aus dem aktuellen Zündwinkel muß ein Startwert generiert
               ; werden (Startwert:=(150-Winkel)*2/3
               ;mov a,#150                   ; 150 in A schieben
               ;clr cy                       ; Carry rücksetzen
               ;subb a,zuendwinkel           ; 150-Winkel
               ;mov b,#2                     ; 2 in B
               ;mul ab                       ; A*B
               ;mov b,+3                     ; 3 in B
               ;div ab                       ; a/b
               ;mov r0,#startwert            ; r0 als Zeiger
               ;mov @r0,a                    ; Neuen Startwert speichern

               ;ljmp auslauf_abbruch        ; Und wieder aufwärts....

auslauf_schleife:

               ; Noch kontrollieren, ob der Auslauf schon abgeschlossen
               ; wurde
               mov r0,#endwert_flag            ; r0 als zeiger
               mov a,@r0                       ; Flag in A schieben
               jz hauptprog_ausl_rampe         ; Warten bis fertig

               ; Jetzt ist der Auslauf abgeschlossen

               ; Motorschütz abschalten
               ; Anzahl festlegen
               mov r0,#ser_anzahl        ; r0 mit Adresse laden
               mov @r0,#1                ; Anzahl auf 1 festlegen

               ; Adresse des IO-Slaves angeben
               mov r0,#ser_slave_adr     ; r0 mit Adresse laden
               mov @r0,#io_adresse       ; Adresse festlegen

               mov r0,#ser_puffer          ; r0 mit Adresse d. Puffers laden
               mov @r0,#255                ; Port abschalten
               lcall i2c_master_transmitter ; Und ausgeben

               ; Zündimpulse abschalten
               mov zuendwinkel,#180

               ljmp haupt_start         ; Schleife wieder schliessen


test:
               ; Ewige Schleife, hier bleibt die Kiste stehen
               ljmp test

; ---------------------------------------------------------------------
; - Ende des Hauptprogrammes                                          -
; ---------------------------------------------------------------------

               ljmp programm_start      ; Und wieder von vorne

; *********************************************************************
; * Unterprogramme                                                    *
; *********************************************************************

; ---------------------------------------------------------------------
; - UP NOT-Aus, schaltet alles ab und legt die Kiste auf Eis
; ---------------------------------------------------------------------

not_aus:     ; Zündimpulse abschalten
             mov zuendwinkel,#180

             ; Schütze abschalten

             ; Anzahl festlegen
             mov r0,#ser_anzahl        ; r0 mit Adresse laden
             mov @r0,#1                ; Anzahl auf 1 festlegen

             ; Adresse des IO-Slaves angeben
             mov r0,#ser_slave_adr     ; r0 mit Adresse laden
             mov @r0,#io_adresse       ; Adresse festlegen
```

```
             mov r0,#ser_puffer          ; r0 mit Adresse d. Puffers laden
             mov @r0,#255                ; Port abschalten
             lcall i2c_master_transmitter ; Und ausgeben

             ; Not-Aus anzeigen
             lcall notaus_meldung

             ; Endlose Schleife anlegen
not_ende:    ljmp not_ende

             ret                      ; Und nix wie weg...

; ---------------------------------------------------------------------
; - UP Rampen_param
; - Berechnet aus Start- und Endwert [in %] und der Dauer [in s]
; - die Parameter zur Erzeugung einer Spannungsrampe
; - Diese sind :
; -
; -         Startwert = Zündwinkel zu Beginn der Rampe
; -         Endwert   = Zündwinkel zum Ende der Rampe
; -         steigung_lsb = LSB des Wertes, wie oft ISR0 aufgerufen werden
; -                        muß, bis der Winkel um 1 geändert werden kann
; -         steigung_msb = MSB des Wertes, wie oft ISR0 aufgerufen werden
; -                        muß, bis der Winkel um 1 geändert werden kann
; -         steigung     = 0, falls der Winkel rauf- 1 falls runtergezählt
; -                        werden soll
; ---------------------------------------------------------------------

rampen_param:

               ; Jetzt zunächst Start- und Endwert von Prozent in
               ; Zündwinkel umsetzen (Wert*1.5). Kann auch mit einer Tabelle
               ; gemacht werden, falls die Steuerkennlinie bekannt ist

               ; Startwert:=150-Startwert*1.5
               mov b,#2                ; Divisor:=2
               mov r0,#startwert       ; r0 als Zeiger
               mov a,@r0               ; Startwert in A
               div ab                  ; a/b
               add a,@r0               ; Und den Startwert dazu
               mov @r0,a               ; Und wieder speichern
               mov a,#150              ; 150 in A schieben
               clr cy                  ; Carry-Flag rücksetzen
               subb a,@r0              ; Startwert abziehen
               mov @r0,a               ; Und wieder speichern


               ; Endwert:=150-Endwert*1.5
               mov b,#2                ; Divisor:=2
               mov r0,#endwert         ; r0 als Zeiger
               mov a,@r0               ; Endwert in A
               div ab                  ; a/b
               add a,@r0               ; Und den Endwert dazu
               mov @r0,a               ; Und wieder speichern
               mov a,#150              ; 150 in A schieben
               clr cy                  ; Carry-Flag rücksetzen
               subb a,@r0              ; Endwert abziehen
               mov @r0,a               ; Und wieder speichern

               ; Berechnung der notwendigen Steigung um eine
               ; Spannungsrampe generieren zu können

               ; Richtung setzen
               mov r0,#richtung         ; r0 als Zeiger
               mov @r0,#0               ; Richtung ist zunächst positiv

               ; Bereich:=abs (Startwert-Endwert)
               mov r0,#startwert        ; r0 als Zeiger
               mov a,@r0                ; Startwert in a laden
               mov r0,#endwert          ; r0 als Zeiger
               clr cy                   ; carry löschen
               subb a,@r0               ; Startwert-Endwert

               ; Jetzt muß noch überprüft werden, ob das Ergebniss
               ; negativ ist, also Carry gesetzt ist
               jc bereich_pos
```

```
                ; Hier gelandet war der Bereich negativ, also 2er-
                ; Komplement bilden
                cpl a                   ; Alle Bits invertieren
                add a,#1                ; +1

                ; Richtung setzen
                mov r0,#richtung         ; r0 als Zeiger
                mov @r0,#1               ; Richtung ist negativ

bereich_pos:
                ; Bereich in Variable speichern
                mov r0,#bereich         ; r0 als Zeiger
                mov @r0,a               ; Bereich speichern

                ; Steigung:=dauer*100/Bereich

                ; Jetzt Dauer*100 bilden
                mov r0,#dauer           ; r0 als Zeiger
                mov a,@r0               ; Dauer in A schieben
                mov b,#100              ; 100 in B
                mul ab                  ; A*B

                ; Jetzt durch Bereich teilen

                ; Ergebniss in den Zähler schieben
                mov dividend_msb,b
                mov dividend_lsb,a

                ; Bereich in den Nenner schieben
                mov divisor_msb,#0    ; MSB ist 0
                mov r0,#bereich       ; r0 als Zeiger
                mov divisor_lsb,@r0   ; LSB übergeben

                lcall division        ; Teilung ausführen

                ; Das Ergebniss der ganzen Geschichte umsetzen
                mov r0,#steigung_msb             ; r0 als Zeiger
                mov @r0,ergebniss_msb
                mov r0,#steigung_lsb             ; r0 als Zeiger
                mov @r0,ergebniss_lsb

                ; Jetzt kann es vorkommen, das das Ergebniss 0 ist.
                ; In diesem Fall würde der Zähler in der ISR0 bis 2^16
                ; raufzählen. Deshalb wird das Ergbeniss dann 1 gesetzt
                mov a,ergebniss_msb             ; MSB in A
                jnz rampen_param_ende           ; Raus, falls <>0
                mov a,ergebniss_lsb             ; LSB in A
                jnz rampen_param_ende           ; Raus, falls <>0

                ; Hier gelandet war das Ergebniss leider 0
                mov r0,#steigung_lsb             ; r0 als Zeiger
                mov @r0,#1                       ; Auf 1 setzen


rampen_param_ende:

                ret                     ; Mach' die verdammte Tür zu....

; ***********************************************************************
; * Kleine Hilfsroutinen
; ***********************************************************************

;-------------------------------------------------------------------------
; UP get_eeprom_data
; Holt den Wert an Adresse A aus dem EEPROM und gibt diesen in A zurück
;-------------------------------------------------------------------------

get_eeprom_data:
                  ; Adresse des EEproms angeben
                  mov r0,#ser_slave_adr     ; r0 mit Adresse laden
                  mov @r0,#eeprom_adresse   ; Adresse festlegen

                  ; Anzahl der zusendenen Bytes auf 1 setzen
                  mov r0,#ser_anzahl        ; r0 mit Adresse laden
                  mov @r0,#1                ; Anzahl auf 1 festlegen

                  ; Zu sendende Adresse festlegn
                  mov r0,#ser_puffer        ; r0 mit Adresse d. Puffers laden
```

```
                mov @r0,a               ; Adresse in 1. Stelle laden

                ; EEPROM ansprechen und Adresse übermitteln
                lcall I2C_master_transmitter

                ; Der nöchste Lesezugriff liefert jetzt das adressierte
                ; Byte
                lcall I2C_master_receiver

                ; Byte aus Puffer auslesen
                mov r0,#ser_puffer      ; r0 mit Adresse d. Puffers laden
                mov a,@r0               ; Wert in A schieben

                ret                     ; Und wieder raus....

; ---------------------------------------------------------------------
; - UP Startmeldung, gibt eine Meldung nach dem Einschalten aus
; ---------------------------------------------------------------------

startmeldung:

             mov a,#01h                 ; 1 in A schieben
             lcall display_kommando     ; Display löschen

             mov dptr,#startanzeige_zeile1  ; Zeiger auf Startmeldung

             ; Zähler für 15 Zeichen generieren
             mov r7,#15             ; r7 als Zähler mit 15 laden
             mov r6,#0              ; 0 in A schieben

startmeldung_ausgabe1:

              mov a,r6              ; Adresse in A schieben
              inc r6                ; Inc Adresse
              movc a,@a+dptr        ; Zeichen Oberbegriff [r2] holen

              lcall display_Ausgabe  ; Zeichen ausgeben

              djnz r7,startmeldung_ausgabe1

              ; Jetzt den Cursor auf die 2. Zeile positionieren
              mov a,#10h                ; Y=1 in A
              lcall display_cursor      ; Und setzen

              ; Jetzt die 2. Zeile ausgeben
              mov dptr,#startanzeige_zeile2  ; Zeiger auf Startmeldung

             ; Zähler für 15 Zeichen generieren
             mov r7,#15             ; r7 als Zähler mit 15 laden
             mov r6,#0              ; 0 in A schieben

startmeldung_ausgabe2:

              mov a,r6              ; Adresse in A schieben
              inc r6                ; Inc Adresse
              movc a,@a+dptr        ; Zeichen Oberbegriff [r2] holen

              lcall display_Ausgabe  ; Zeichen ausgeben

              djnz r7,startmeldung_ausgabe2

             ret                    ; Raus ausse Kartoffeln

startanzeige_zeile1:

            DB 'Sanftanlauf 1.0'       ; Der Spruch auf den Weg...

startanzeige_zeile2:

            DB 'OK : Menue     '       ; Der Spruch auf den Weg...

; ---------------------------------------------------------------------
; - UP notaus_meldung, Ende der Durchsage sozusagen.....
; ---------------------------------------------------------------------

notaus_meldung:

              mov a,#01h                ; 1 in A schieben
```

```
             lcall display_kommando     ; Display löschen

             mov dptr,#notaus_zeile1  ; Zeiger auf Startmeldung

             ; Zähler für 15 Zeichen generieren
             mov r7,#15            ; r7 als Zähler mit 15 laden
             mov r6,#0             ; 0 in A schieben

notaus_ausgabe1:

              mov a,r6               ; Adresse in A schieben
              inc r6                 ; Inc Adresse
              movc a,@a+dptr         ; Zeichen Oberbegriff [r2] holen

              lcall display_Ausgabe  ; Zeichen ausgeben

              djnz r7,notaus_ausgabe1

             ret                       ; Raus ausse Kartoffeln

notaus_zeile1:

            DB 'NOT - AUS      '       ; Der Spruch auf den Weg...

; ---------------------------------------------------------------------
; - UP hochlauf_meldung, gibt eine Meldung nach dem Einschalten aus
; ---------------------------------------------------------------------

hochlauf_meldung:

             mov a,#01h                ; 1 in A schieben
             lcall display_kommando     ; Display löschen

             mov dptr,#hochlauf_zeile1  ; Zeiger auf Startmeldung

             ; Zähler für 15 Zeichen generieren
             mov r7,#15            ; r7 als Zähler mit 15 laden
             mov r6,#0             ; 0 in A schieben

hochlauf_ausgabe1:

              mov a,r6               ; Adresse in A schieben
              inc r6                 ; Inc Adresse
              movc a,@a+dptr         ; Zeichen Oberbegriff [r2] holen

              lcall display_Ausgabe  ; Zeichen ausgeben

              djnz r7,hochlauf_ausgabe1

             ret                       ; Raus ausse Kartoffeln

hochlauf_zeile1:

            DB 'Hochlauf...    '       ; Der Spruch auf den Weg...

; ---------------------------------------------------------------------
; - UP betrieb_meldung, gibt eine Meldung "Betrieb" aus
; ---------------------------------------------------------------------

betrieb_meldung:

             mov a,#01h                ; 1 in A schieben
             lcall display_kommando     ; Display löschen

             mov dptr,#betrieb_zeile1  ; Zeiger auf Startmeldung

             ; Zähler für 15 Zeichen generieren
             mov r7,#15            ; r7 als Zähler mit 15 laden
             mov r6,#0             ; 0 in A schieben

betrieb_ausgabe1:

              mov a,r6               ; Adresse in A schieben
              inc r6                 ; Inc Adresse
              movc a,@a+dptr         ; Zeichen Oberbegriff [r2] holen

              lcall display_Ausgabe  ; Zeichen ausgeben
```

```
              djnz r7,betrieb_ausgabe1

              ret                       ; Raus ausse Kartoffeln

betrieb_zeile1:

             DB 'Betrieb...     '       ; Der Spruch auf den Weg...

; ---------------------------------------------------------------------
; - UP kick_meldung, gibt eine Meldung "Kickstart" aus
; ---------------------------------------------------------------------

kick_meldung:

              mov a,#01h                ; 1 in A schieben
              lcall display_kommando    ; Display löschen

              mov dptr,#kick_zeile1  ; Zeiger auf Startmeldung

              ; Zähler für 15 Zeichen generieren
              mov r7,#15            ; r7 als Zähler mit 15 laden
              mov r6,#0             ; 0 in A schieben

kick_ausgabe1:

               mov a,r6              ; Adresse in A schieben
               inc r6                ; Inc Adresse
               movc a,@a+dptr        ; Zeichen Oberbegriff [r2] holen

               lcall display_Ausgabe  ; Zeichen ausgeben

               djnz r7,kick_ausgabe1

              ret                       ; Raus ausse Kartoffeln

kick_zeile1:

             DB 'Kickstart...   '       ; Der Spruch auf den Weg...

; ---------------------------------------------------------------------
; - UP Auslauf_meldung, gibt eine Meldung aus
; ---------------------------------------------------------------------

Auslauf_meldung:

              mov a,#01h                ; 1 in A schieben
              lcall display_kommando    ; Display löschen

              mov dptr,#auslauf_zeile1  ; Zeiger auf Startmeldung

              ; Zähler für 15 Zeichen generieren
              mov r7,#15            ; r7 als Zähler mit 15 laden
              mov r6,#0             ; 0 in A schieben

auslauf_ausgabe1:

               mov a,r6              ; Adresse in A schieben
               inc r6                ; Inc Adresse
               movc a,@a+dptr        ; Zeichen Oberbegriff [r2] holen

               lcall display_Ausgabe  ; Zeichen ausgeben

               djnz r7,auslauf_ausgabe1

              ret                       ; Raus ausse Kartoffeln

auslauf_zeile1:

             DB 'Auslauf...     '       ; Der Spruch auf den Weg...


; ---------------------------------------------------------------------
; - UP division, Ergebniss:=Dividend/Divisor (16Bit ohne Vorzeichen)
; - Verändert a,psw
; - Die Quelle dieser Routine ist unbekant, dem Stil nach wohl
; - Data Becker (Würg!)
; ---------------------------------------------------------------------
division:
```

```
                ; Erst einmal alles auf 0 setzen
                mov ergebniss_msb,#0
                mov ergebniss_lsb,#0
                mov arbeitsreg_msb,#0
                mov arbeitsreg_lsb,#0

                ; Bitzähler auf 16
                mov bitzaehler,#10h

d10p:
                mov a,dividend_lsb
                rlc a
                mov dividend_lsb,a
                mov a,dividend_msb
                rlc a
                mov dividend_msb,a
                mov a,arbeitsreg_lsb
                rlc a
                mov arbeitsreg_lsb,a
                mov a,arbeitsreg_msb
                rlc a
                mov arbeitsreg_msb,a
                clr c
                mov a,arbeitsreg_lsb
                subb a,divisor_lsb
                mov zwischensp,a
                mov a,arbeitsreg_msb
                subb a,divisor_msb
                jc d20p
                mov arbeitsreg_msb,a
                mov arbeitsreg_lsb,zwischensp
d20p:

                cpl c
                mov a,ergebniss_lsb
                rlc a
                mov ergebniss_lsb,a
                mov a,ergebniss_msb
                rlc a
                mov ergebniss_msb,a

                djnz bitzaehler,d10p                        ; Bitzähler-1

                ret

; ---------------------------------------------------------------------
; - Up delay_10ms
; - Macht eine Pause von ca. 10 ms
; - Verändert : r6,r7,a,PSW
; ---------------------------------------------------------------------

delay_10ms:
                    mov r6,#50              ; r6 laden mit 50
outer_loop:

                    mov r7,#255             ; r7 laden mit 255
inner_loop:         djnz r7,inner_loop      ; dec r7

                    djnz r6,outer_loop      ; dec r6

                    ret                     ; Zurück zur Natur

; ---------------------------------------------------------------------
; - UP bin2bcd
; - Setzt einen 8Bit-Binärwert in A in 3 BCD-Ziffern um (R5-R7)
; - Verändert : A, PSW, B, R0, R1, R2
; - Achtung : Für ASCII-Darstellung jeweils 48 hinzuaddieren
; ---------------------------------------------------------------------

bin2bcd:
                  mov b,#100              ; B mit 100 laden
                  div ab                  ; a : b

                  ; Ist die erste Zahl eine Null, kann stattdessen
                  ; ein Leerzeichen eingefügt werden
                  mov r5,#20h             ; In r5 ein Leerzeichen schieben
                  jz bin2bcd_lead
```

```
                    ; War wohl doch nicht Null.......
                    add a,#48              ; Für ASCII hinzuaddieren
                    mov r5,a               ; Ergebniss in r5

bin2bcd_lead:       mov a,b                ; Rest in A schieben
                    mov b,#10              ; B mit 10 laden
                    div ab                 ; a : b
                    add a,#48              ; Für ASCII hinzuaddieren
                    mov r6,a               ; Ergebniss in R6
                    mov a,b                ; Rest in A schieben
                    add a,#48              ; Für ASCII hinzuaddieren
                    mov r7,a               ; Rest in R7
                    ret                    ; Take me back to earth...

; ---------------------------------------------------------------------
; - UP isr_init, gibt die Interrupts frei etc...
; - Verändert :
; ---------------------------------------------------------------------

isr_init:
                    ; Port 4 auf 0 setzen
                    mov p4,#0

                    ; Pin P1.0 auf low setzen, damit T2 laufen kann
                    anl p1,#254

                    mov zuendwinkel,#180     ; Erst einmal alles aus...

                    ; Zähler und Flags rücksetzen
                    mov r0,#endwert_flag          ; r0 als zeiger
                    mov @r0,#0                    ; Flag löschen
                    mov r0,#steigungszaehler_msb  ; r0 als zeiger
                    mov @r0,#0                    ; Zaehler löschen
                    mov r0,#steigungszaehler_lsb  ; r0 als zeiger
                    mov @r0,#0                    ; Zaehler löschen

                    ; Jetzt den IO-Slave so initialisieren, das 4 Ein-
                    ; und 4 Ausgänge zur Verfügung stehen

                    ; Zeichen in 1. Pufferstelle schieben
                    mov r0,#ser_puffer        ; r0 mit Adresse d. Puffers laden
                    mov @r0,#11111111b        ; P7-P4 Ausgang
                    ; Anzahl festlegen
                    mov r0,#ser_anzahl        ; r0 mit Adresse laden
                    mov @r0,#1                ; Anzahl auf 1 festlegen
                    ; Adresse des IO-Slaves angeben
                    mov r0,#ser_slave_adr     ; r0 mit Adresse laden
                    mov @r0,#io_adresse       ; Adresse festlegen
                    ; Und raus zum IO-Slave damit
                    lcall  I2C_master_transmitter


                    ; T0 und T1 als 16-Bit-Timer konfigurieren
                    mov tmod,#00010001b

                    ; T2 konfigurieren
                    ; Zuerst Compare-Register CM0 vorladen für 3.3ms
                    ; entspricht 3333d=0d05h
                    mov CMH0,#0dh           ; High-Byte beschreiben
                    mov CML0,#05h           ; Low-Byte beschreiben

                    ; Kein Vorteiler, kein Overflow-Int, T2 an 1/12 fosz
                    ; Reset über Pin T2ER möglich, verbunden mit P1.0
                    mov tm2con,#00100000b

                    ; Externe Ints konfigurieren
                    setb ex0                ; INT0 freigeben
                    setb it0                ; INT0 auf fallende Flanke stellen
                    ;setb it1               ; INT1 auf neg. Flanke stellen
                    ;setb ex1               ; INT1 freigeben

                    ; Timer-Ints konfigurieren
                    setb et0                ; T0-Int freigeben
                    setb et1                ; T1-Int freigeben
                    setb ecm0               ; T2-Int Compare 0 freigeben

                    ; Und jetzt kommt die Sache ins Rollen (Rolling home)....
```

```
                    setb ea                 ; Globale interrupt-Freigabe

                    ret                     ; Take me to your dealer

; ********************************************************************
; * Menü-Funktionen
; ********************************************************************

$INCLUDE (men_gen.asm)

; ********************************************************************
; * Display-Steuerung
; ********************************************************************

$INCLUDE (display.asm)

; ********************************************************************
; * Serielle IO-Routinen
; *
; * Speichernutzung im internen Ram :
; *
; * 80h-90h : Sende- und Empfangspuffer
; * 91h     : Anzahl der zu verabreichenden Bytes
; * 92h     : Slaveadresse
; * 93h     : Flagregister für IO-Klamotten
; *           Bit 0 : Fehler auf I2C-Bus
; ********************************************************************

; --------------------------------------------------------------------
; - UP I2C_master_transmitter
; - Sendet Bytes als Master-Transmitter auf dem I2C-Bus
; - Erwartet als šbergabe die Anzahl der zu sendenden Bytes,
; - Daten im Puffer im internen RAM und die Slaveadresse
; - Liefert ein Flag als Fehlerkenung zurück
; - Registerverwendung : R0 = Zeiger auf den Datenpuffer
; -                      R1 = Universeller Zeiger
; -                      R2 = Zähler für die Anzahl der ges. Bytes
; - Verändert : PSW, A, R0, R1, R2
; --------------------------------------------------------------------

I2C_master_transmitter:

               ; Taktrate setzen auf 47 kHz
               clr  cr2
               clr  cr1
               clr  cr0

               ; Fehlerflag rücksetzen
               mov R1,#ser_flag       ; r2 als Zeiger auf Flagregister
               mov a,@R1              ; Flagresister in A laden
               anl a,#254             ; Bit 0 rücksetzen im Akku
               mov @R1,a              ; Akku in Flagregister zurückladen

               ; šbertragung einleiten
               setb ens1              ; SIO1 freigeben
               clr si                 ; Interruptflag rücksetzen
               setb sta               ; Start-Condition erzeugen

bus_start_s:   jnb si,bus_start_s           ; Warten, bis Bus frei ist
               mov r1,#ser_slave_adr        ; R1 als Zeiger auf Slave-
                                            ; adresse laden
               mov s1dat,@r1                ; Ausgaberegister mit der
                                            ; Slaveadresse laden
               clr si                       ; Interruptflag rücksetzen
adress_send_s: jnb si,adress_send_s         ; Warten, bis Adresse gesendet
                                            ; wurde
               ; Jetzt muß im Statusregister 18h liegen, sonst ist ein Fehler aufgetreten
               mov A,s1sta                  ; Statusregister in Akku schieben
               cjne a,#18h,fehler_s         ; Akku<>18h, dann Sprung

               ; Hier gelandet wurde Ack empfangen und es kann weitergehen
               ; im Text, d.h. Daten aus dem Puffer können gesendet werden

               clr sta                  ; Start-Cond. zurücknehmen
               mov r0,#ser_anzahl       ; r0 als Zeiger auf Anzahl
               mov a,@r0                ; Anzahl in Akku schieben
               mov r2,a                 ; r2 mit Anzahl laden
               mov r0,#ser_puffer      ; r0 mit 1. Adresse des Puffers laden
```

```
sendeschleife:

               mov s1dat,@r0            ; Ausgabe mit dem jeweiligen Byte
                                        ; aus dem Puffer laden
               clr si                   ; Interruptflag rücksetzen
data_send_s:   jnb si,data_send_s       ; Warten, bis Byte gesendet

               ; Jetzt muß kontrolliert werden, ob ein Ack empfangen wurde
               ; oder nicht (Status=28h)
               mov A,s1sta              ; Statusregister in Akku schieben
               cjne a,#28h,fehler_s     ; Wenn Akku<>28h, dann Sprung
               ; Es wurde also ein Ack empfangen, es kann also weitergehen
               inc r0                   ; r0 erhöhen um 1
               djnz r2,sendeschleife    ; Wenn Anzahl der zu sendenden
                                        ; Bytes noch nicht erreicht, nächstes
                                        ; Byte holen und senden

               ; Hier gelandet wurden alle Bytes gesendet, die Sache wird
               ; zum Abschluss gebracht
               ljmp data_send_ende


               ; Hier gelandet ist ein Fehler aufgetreten, also
               ; Fehlerflag setzen
fehler_s:
                mov r1,#ser_flag        ; r1 als Zeiger auf Flagregister
                mov a,@r1               ; Flagregister in A laden
                orl a,#01h              ; Bit 0 setzen im Akku
                mov @r1,a               ; Akku in Flagregister zurückladen

data_send_ende:

               ; šbertragung abschliessen
               clr sta                  ; Start-Condition rücksetzen
               setb sto                 ; Stop-Condition erzeugen
               clr si                   ; Interruptflag rücksetzen

               ; Rücksprung zum Rest der Geschichte
               ret


; ---------------------------------------------------------------------
; - UP I2C_master_receiver
; - Empfängt Bytes als Master-Receiver auf dem I2C-Bus
; - Erwartet als šbergabe die Anzahl der zu empfangenen Bytes.
; - Liefert ein Flag als Fehlerkenung zurück und Daten im Puffer
; - Registerverwendung : R0 = Zeiger auf den Datenpuffer
; -                      R1 = Universeller Zeiger
; -                      R2 = Geplante Anzahl der empf. Bytes
; -                      R3 = Zähler für empf. bytes
; - Verändert : PSW, A, R0, R1, R2, R3
; ---------------------------------------------------------------------

I2C_master_receiver:

               ;clr ex1              ; INT1 sperren

               ; Taktrate setzen auf 47 kHz
               clr  cr2
               clr  cr1
               clr  cr0

               ; Fehlerflag rücksetzen
               mov R1,#ser_flag       ; r1 als Zeiger auf Flagregister
               mov a,@R1              ; Flagresister in A laden
               anl a,#254             ; Bit 0 rücksetzen im Akku
               mov @R1,a              ; Akku in Flagregister zurückladen

               ; šbertragung einleiten
               setb ens1              ; SIO1 freigeben
               clr si                 ; Interruptflag rücksetzen
               setb sta               ; Start-Condition erzeugen

bus_start_r:   jnb si,bus_start_r           ; Warten, bis Bus frei ist
               mov r1,#ser_slave_adr        ; R1 als Zeiger auf Slave-
                                            ; adresse laden
               mov a,@r1                    ; Akku mit Slaveadresse laden
```

```
                mov r2,a                    ; R2 mit Slaveadresse laden
                inc r2                      ; Read-Bit setzen
                mov s1dat,r2                ; Register mit Slaveadresse laden
                clr si                      ; Interruptflag rücksetzen
adress_send_r:  jnb si,adress_send_r        ; Warten, bis Adresse gesendet
                                            ; wurde

                ; Jetzt muß im Statusregister 40h liegen, sonst ist ein Fehler aufgetreten
                mov A,s1sta                 ; Statusregister in Akku schieben
                cjne a,#40h,fehler_r        ; Akku<>40h, dann Sprung

                ; Hier gelandet wurde Ack empfangen und es kann weitergehen
                ; im Text, d.h. Daten können empfangen werden

                clr sta                  ; Start-Cond. zurücknehmen
                mov r0,#ser_anzahl       ; r0 als Zeiger auf Anzahl
                mov a,@r0                ; Anzahl in Akku schieben
                mov r2,a                 ; r2 mit Anzahl laden
                mov r0,#ser_puffer      ; r0 mit 1. Adresse des Puffers laden

                ; Zähler für empf. Bytes laden
                mov r3,#1                ; r3 mit 1 laden

empf_schleife:
                ; Ist der Zähler = Anzahl ? ( r3=r2 ?)
                mov a,r2                 ; r2 in Akku laden
                clr cy                 ; Carry rücksetzen
                subb a,r3                ; r3 von A abziehen
                jz reset_aa              ; Wenn gleich, AA rücksetzen

                ; Hier gelandet ist Zähler<>Anzahl
                setb aa                  ; ACK freigeben
                ljmp cls_aa

reset_aa:       clr aa                   ; ACK sperren

cls_aa:
                clr si                   ; Interruptflag rücksetzen

data_send_r:    jnb si,data_send_r       ; Warten, bis Byte empfangen

                ; Jetzt kann das Empfangene Byte in Puffer geschoben werden
                mov @r0,s1dat            ; Byte auslesen in Puffer
                inc r0                   ; Inc Zeiger
                inc r3                   ; Inc Zähler

                ; Jetzt muß festgestellt werden, ob schon genug Bytes
                ; empfangen wurden (r3>r2 ?)
                ;mov a,r3                 ; r3 in a laden
                ;clr cy                 ; Carry rücksetzen
                ;subb a,r2                ; r2 von a abziehen

                ; Hat jetzt ein Unterlauf stattgefunden, ist r3>r2
                ;jnb cy,empf_schleife    ; Wenn Anzahl der zu empfangenen
                                         ; Bytes noch nicht erreicht, nächstes
                                         ; Byte reinholen

                ; Hier gelandet wurden alle Bytes empfangen, die Sache wird
                ; zum Abschluss gebracht
                ljmp data_read_ende

                ; Hier gelandet ist ein Fehler aufgetreten, also
                ; Fehlerflag setzen
fehler_r:
                 mov r1,#ser_flag        ; r1 als Zeiger auf Flagregister
                 mov a,@r1               ; Flagregister in A laden
                 orl a,#01h              ; Bit 0 setzen im Akku
                 mov @r1,a               ; Akku in Flagregister zurückladen

data_read_ende:

                ; šbertragung abschliessen
                clr sta                  ; Start-Condition rücksetzen
                setb sto                 ; Stop-Condition erzeugen
                clr si                   ; Interruptflag rücksetzen

                ;setb ex1                ; INT1 freigeben
```

```
                ; Rücksprung zum Rest der Geschichte
                ret



; ---------------------------------------------------------------------
; EInbinden der Menü-Daten
; ---------------------------------------------------------------------
$INCLUDE (menue.asm)


; ---------------------------------------------------------------------
; - Ende des Sourcecodes                                              -
; ---------------------------------------------------------------------

                 END                    ; Ende des Sourecodes
```