Diplomarbeit 1993

Zeitdiskreter Regler mit Stellglied

Kandidaten:

Oliver Gossel
Hochschule für Technik, Wirtschaft und Sozialwesen, Zittau/Görlitz (FH)

Roger Meier
Technikum Winterthur Ingenieurschule (HTL), Klasse 6Ec

Dozent:

Prof. Dr. J. Golder

Im Dezember 1993

Inhaltsverzeichnis

1. Einleitung

1.1 Aufgabenstellung

TECHNIKUM WINTERTHUR
INGENIEURSCHULE DES KANTONS ZÜRICH

Schulleitung

Abteilung E
Vertiefungsfach RT
Jahr 1993
Experten:

Bachofner Werner
Märki Andreas

Praktische Prüfung

Kandidaten Meier Roger Gossel Oliver	Klasse 6Ec Fachhochschule Zittau/Görlitz	Lehrer Golder

Ausgabe der Aufgabe	27. Oktober 1993
Abgabetermin	6. Dezember 1993 1130

Thema

Zeitdiskreter Regler mit Stellglied

Gr 93/1
(DA_93I01.PRF)

Spezifikationen

Übersicht:

Die Arbeit baut auf eine im Frühsommer 1993 angefertigte Projektarbeit zum Mikrocontroller Motorola 68332 auf, umfasst den Entwurf einer PWM-H-Brücke sowie geeigneter Regleralgorithmen, vorerst im Hinblick auf ein Wippenmodell für das Labor. Dabei sind auch die Fragen um Abtastzeit- sowie Auflösungseinflüsse zu klären. Dazu sollen vorhandene Simulationswerkzeuge (MatLab) eingesetzt werden.

Aufgabe:

Der zeitdiskrete Regler, implementiert im Mikrocontroller, soll über seinen PWM-Ausgang an den kontinuierlichen Prozess angeschlossen werden. Die dazu erforderliche geschaltete Leistungsbrücke ist unter weitestgehender Verwendung integrierter Bauteile zu entwerfen und aufzubauen. Dabei ist insbesondere auf Betriebssicherheit bezüglich Strombegrenzung, überspannungsfestigkeit, Kurzschlussschutz zu achten. Die Schaltung soll ferner die Messung der Stromstärke im Modell-Eingangskreis ermöglichen. Die Anzahl der analogen Eingänge ist entsprechend zu erhöhen durch Erweiterung des vorhandenen Konzepts.

Zu untersuchen sind die Möglichkeiten einer Kaskadenregelung mit unterlagertem Winkellage-Regler sowie ein Zustandsregler mit unvollständigen Rückführungen: die Rollgeschwindigkeit der Kugel fehlt als gemessene Grösse.

Verlangt sind aufgebaute und funktionsbereite Schaltungsteile (Eingangs- und Ausgangsseite, Format Europakarte mit 64poligem VG-Stecker, ferner Modellanalysen und Simulationen, jedoch noch kein Echtzeitbetrieb am wirklichen Modell zur Abklärung der Einflüsse insbesondere der Abtastzeit, der Auflösungen am Eingang und am Ausgang (Grenzschwingungen). Ferner werden geprüfte und dokumentierte Programmteile für den Controller als Regler erwartet.

Literatur:

Projektarbeit Oechslin/Gahlinger Sommer 1993 in den Fächern MC/RT
Dokumentation Motorola 68332 Mikrocontroller
Unterlagen C-Entwicklungsumgebung für 68332
Leigh: "Applied Digital Control", Prentice Hall, New York, 1992

Kopien am Ausgabetag an die Direktion (ohne Beilagen): 1 für Archiv
je 1 für beteiligte Experten
ferner: je 1 an Prof. Dr. Rähder und Prof. Dr. Karbaum, FH Zittau/Görlitz

2. Hardware

2.1 Mikrokontroller Motorola MC68332

2.1.1 Übersicht

Der Motorola **MC68332** ist ein 32-Bit Mikrokontroller mit verschiedenen internen Baugruppen, die dem Anwender zur Verfügung stehen. Er ist das erste Mitglied der Kontrollerfamilie **M68300**, die in CMOS-Technologie gefertigt werden. Der Kontroller basiert intern auf dem Prozessor **MC68020** der Firma Motorola. Extern besitzt er aber nur einen 16-bit Datenbus und 24 Adressleitungen. Alle Befehle des **MC68010** und die meisten des **MC68020** sind auch im Befehlssatz des **MC68332** enthalten.

Im folgenden sind die wichtigsten Merkmale des Kontrollers **MC68332** kurz aufgeführt:

- Leistungsaufnahme max. 600 mW und 500 µW im Standby-Mode
- Chipfrequenz 16.78 MHz
- modulare Architektur auf dem Chip
- neue Befehle für Kontrolleranwendungen
- intelligente Zeiverarbeitungseinheit TPU mit
  - -16 vollständigen Kanälen
  - verschiedenen Timefunktionen (z.B. PWM-Signal oder Output-Compare)
  - zwei Zählerregistern
  - einstellbaren Kanalprioritäten
- zwei verschiedene serielle I/O-Einheiten
  - serial Communications Interface (SCI)
  - serial Peripheral Interface (SPI)
- 2k-Bytes RAM on Chip
- Chip-Select-Logic mit bis zu 12 Kanälen on Chip
- bis zu 32 diskrete Ein- und Ausgänge

Die *Abb. 2.1.1* zeigt eine Übersicht der einzelnen Moduln des **MC68332** mit ihren Ein- und Ausgangssignalen. Dabei ist zu bemerken, dass auf Grund der begrenzten Pin-Anzahl (132) nicht jede Signalleitung einzeln nach aussen geführt ist und deshalb bei der Anwendung Kompromisse eingegangen werden müssen (siehe Abschnitt Chip-Select-Decodierung).

Die einzelnen Moduln TPU, die QSM, das System Integration Module (SIM) und der 2 kByte RAM bilden zusammen mit der CPU32 ein leistungsfähiges Gebilde für die Anwendungsbereiche des Kontrollers **MC68332**.

Mögliche Anwendungsgebiete des Kontrollers auf Grund seiner Leistungsfähigkeit sind:

- alle Bereiche der Automatisierungstechnik
- Antriebstechnik (Steuerung von Motoren)
- Automobilindustrie (z.B. in Fahrzeugen der Firma SAAB)
- Sicherheitstechnik
- Konsumgüterindustrie

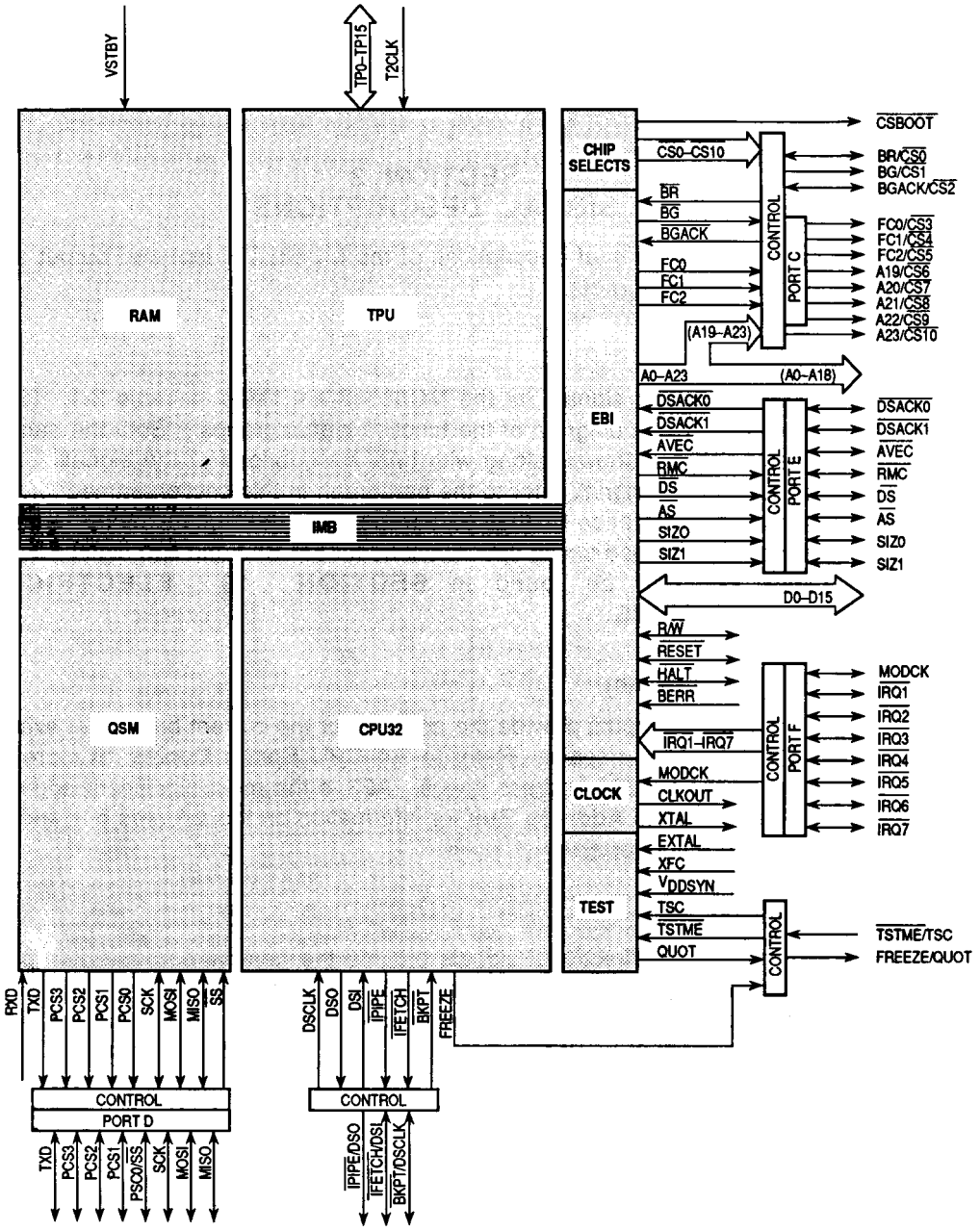


*Abb. 2.1.1: Blockschaltbild des Kontrollers **MC68332** (Quelle: [4])*

[4]: Motorola, MC68332 User's Manual, Seite 2-2

2.1.2 Time Processor Unit (TPU)

In den meisten Anwendungsfällen der Mikrokontrollertechnik sind zeitrelevante Verarbeitungen nötig. Deshalb werden normalerweise die Kontroller mit einer internen Zeitverarbei-tungseinheit ausgerüstet. Das dies nicht nur bei Produkten der Firma Motorola der Fall ist, beweist die Intel-Kontroller Serie **i8051**.

Der im vorliegenden Fall angewendete Mikrokontroller **MC68332** beinhaltet eine sehr gut ausgebaute integrierte Einheit (Übersicht über Aufbau der TPU in *Abb. 2.1.2*). Sie besitzt 16 völlig gleichwertige Zeitkanäle, die je durch einen Pin mit der Aussenbeschaltung verbunden werden können. Jeder Kanal kann mit einem der beiden 16-Bit Zähler verbunden werden. Ein Zähler kann mit dem Systemtakt synchronisiert werden und besitzt eine Auflösung von minimal 500 ns. Der andere Zähler basiert auf der an dem Pin T2CLK anlegbaren externen Taktquelle und kann minimal 250 ns zählen.

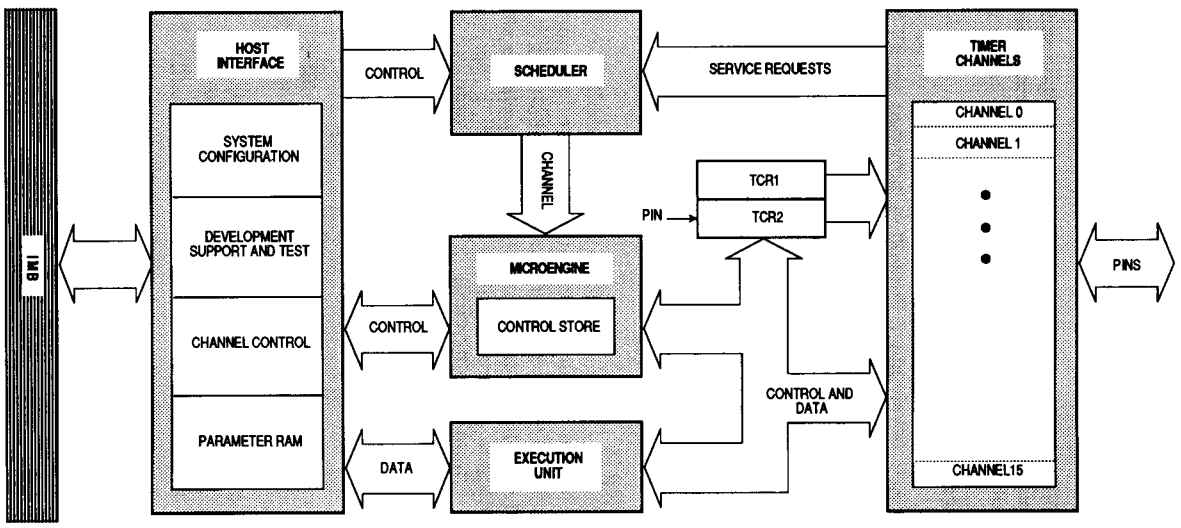


Abb. 2.1.2: TPU Block-Diagramm (Quelle: [11])

Wie bereits erwähnt, besitzt die TPU einige Zeitfunktionen, die bereits integriert sind und im folgenden kurz beschrieben werden.

Discrete Input/Output (DIO)

Diese Funktion ermöglicht eine diskrete Ein- oder Ausgangsfunktion an jedem beliebigen Pin. Das TPU-Register PIN_LEVEL enthält dabei die letzten 16 Zustände des Pins (Bit 15 entspricht dabei dem aktuellen Zustand). Bei einer Eingabefunktion kann festgelegt werden, wann der Inhalt des Registers erneuert werden soll. Entweder bei jedem Pegelwechsel am Pin oder bei einer Anfrage durch die CPU. Die CPU kann sich jederzeit ein Up-Date des Registers holen. Wenn der Pin als Ausgang deklariert wurde, kann die CPU den Pegel durch einen Zugriff auf das HSR-Register festlegen.

[11]: Motorola, TPU Reference Manual, Seite 1-6

Input Capture/ Input Transition Counter (ITC)

Diese Funktion dient der Überwachung von Zuständen am Pin. Dabei kann entweder jede Änderung am Pin ausgewertet werden (Input Capture) oder die Änderungen am Pin gezählt werden. Diese Funktion kann zum Auslösen von Interrupts benötigt werden, z.B. für die Überwachung von Schutzschaltungen in der Anwendung des Kontrollers (siehe Abschnitt 2.3.4 Überstromschutz des Stellgliedes). In der Konfiguration der Funktion kann festgelegt werden, ob die weitere Arbeit des Kanals nach der ersten Auslösung sofort oder erst bei einer neuen Initialisierung erfolgen soll.

Output Compare (OC)

Mit dieser Zeitfunktion können am Kanalausgang fallende oder steigende Flanken generiert werden. Ausserdem ist es möglich einen Takt zu erzeugen, der durch Teilung des Prozessortaktes erreicht wird. Allerdings ist es nur möglich eine Frequenz von gleich oder weniger als 2 MHz zu erreichen.
Da für den Betrieb des A/D-Wandlers eine Frequenz von 8 MHz nötig ist (siehe Abschnitt 2.2.4), kann diese Funktion leider nicht für die Teilung verwendet werden.

Pulse-Width Modulation (PWM)

Die TPU kann für den Anwender ohne grossen Aufwand ein PWM-Signal mit einer Tastrate zwischen 0 und 100 % erzeugen. Dazu muss die Periodendauer und die Dauer der High-Zeit programmiert werden.

Synchronized Pulse-Width Modulation (SPWM)

Der Unterschied zur vorangegangenen Funktion liegt darin, dass das PWM-Signal zusätzlich von einem anderen Kanal synchronisiert werden kann.

Stepper Motor (SM)

Mit Hilfe der SM-Funktion lassen sich problemlos Schrittmotoren im Halbschritt- (4 Kanäle) oder im Vollschrittbetrieb (2 Kanäle) betreiben. Dabei steuern die Ausgänge der TPU z.B. H-Brücken an, die dann wiederum die Spulen des Schrittmotors speisen.

Period/Pulse-Width Accumulator (PPWA)

In bestimmten Anwendungsfällen kann es notwendig sein, die Perioden oder Pulswellen zu zählen. Dazu kann ein in dieser Funktion programmierter Kanal dienen. Er kann eine Anzahl zwischen 0 und 255 zählen, und bei Erreichen eines voreingestellten Wertes einen anderen Kanal triggern (auslösen).

Period Measurement with Additional Transition Detection (PMA)

Diese Funktion dient der periodischen Messung eines Signals. Beim Auftreten eines zusätzlichen Übergangs wird eine entsprechende Funktion, z.B. Linking auf einen anderen Kanal, ausgelöst.

Period Measurement with Missing Transition Detection (PMM)

Wie in der vorangegangenen Zeitfunktion wird auch hier ein Signal überwacht, z.B. tastet ein optischer oder magnetischer Sensor eine Scheibe mit codierten Informationen ab. Die Auslösung einer Funktion geschieht hier aber bei Auftreten eines fehlenden Übergangs.

Position-Synchronized Pulse Generator (PSP)

Jeder Kanal kann von der TPU generiert werden, einen Übergang oder einen Impuls am Ausgang zu liefern. Als Zeitgrundlage dient dabei einer der anderen Kanäle, der zuvor aktiviert wurde (z.B. durch eine PMA-Funktion).

Das Zusammenspiel der drei letztgenannten Zeitfunktionen ermöglicht eine erstaunliche Anwendung, die allerdings vom Entwickler des Kontroller durchaus gewollt ist. Wie bereits erwähnt, besteht ein grosses Anwendungsgebiet des Kontrollers in der Automobilindustrie. Hier kann die Verwendung der TPU zur Steuerung der gesamten Motorelektronik genutzt werden. Über einen Sensor, der die Kurbelwelle überwacht, werden mit Hilfe der PMA- oder PMM-Funktion entsprechend viele andere Kanäle der TPU angesteuert. Diese dienen nun, programmiert in der PSP-Funktion, als Zeitkanäle zur elektronischen Steuerung der Zündkerzen und der Benzineinspritzung.

Für genauere Angaben zu den verschiedenen Funktionen der Time-Processor-Unit und ihrer Programmierung soll hier auf die User's Manuals der Firma Motorola [4] und [11] hingewiesen werden.

[4]: Motorola, MC68332 User's Manual
[11]: Motorola, TPU Reference Manual

2.1.3 System Integration Module (SIM)

Der SI-Modul dient der Steuerung des Mikrokontrollers, z.B. dem System Startup, der Konfiguration und der Zeitbasiserzeugung. In *Abb. 2.1.3* sind an Hand des Block-Diagramms die einzelnen Funktionen ersichtlich.

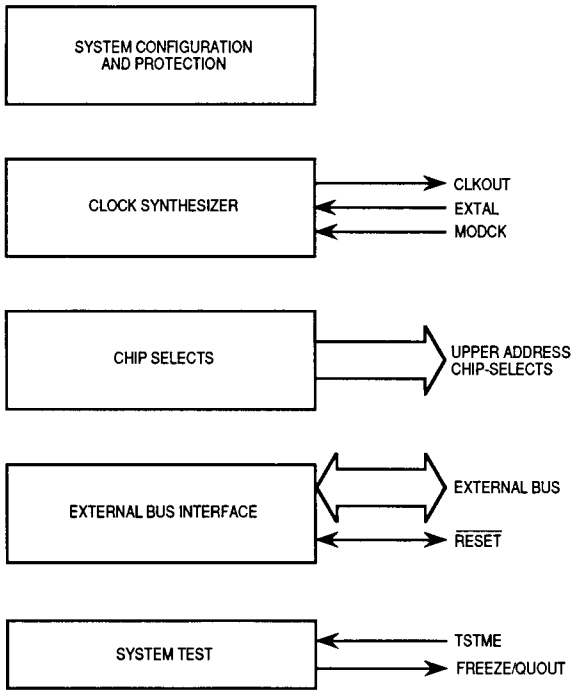


Abb. 2.1.3: SIM-Blockdiagramm (Quelle: [4])

Das Systemkonfigurationsmodul enthält alle wichtigen Register, die zum Verhalten der CPU notwendig sind. Dazu zählen Informationen ob ein externer Clock angeschlossen wurde, wie das Verhalten in Master-Slave-Systemen zu generieren ist und welche RESET-Quelle angenommen werden soll. Ausserdem sind die Interrupteingänge zu konfigurieren.

Chip-Select-Modul

Die typischen Mikrokontroller benötigen zur Generierung der Chip-Select-Signale für die Peripherie eine externe Hardware. Dieser Kontroller hat diese bereits integriert. Die Chip-Select-Signale können als Output-Enable, als Read/Write oder als Interrupt-Acknowledge programmiert werden.

[4]: Motorola, MC68332 User's M;anual, Seite 4-1

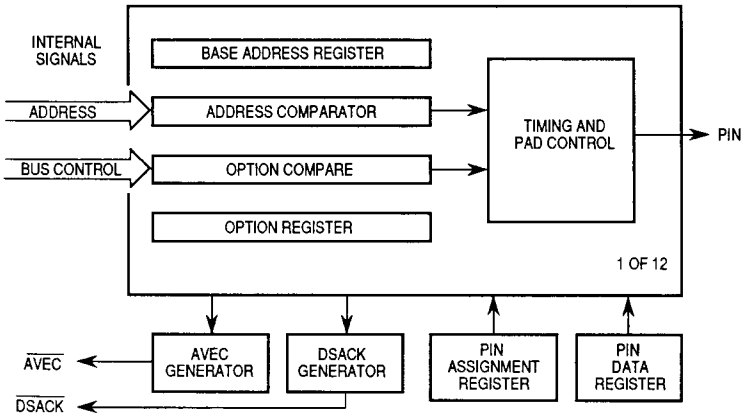


Abb. 2.1.4: Blockdiagramm der Chip-Select-Schaltung (Quelle: [4])

Insgesamt stehen 12 Chip-Select-Signale zur Verfügung, die Ausgangspins können jedoch gleichzeitig durch andere Signale genutzt werden. Die Funktion der entsprechenden Pins kann mit Hife der CSPARx-Register eingestellt werden, sodass sowohl die Alternativfunktionen als auch die Chip-Select-Signale genutzt werden können. Zur Erläuterung der Funktionsweise soll die *Abb. 2.1.4* dienen, in der das Blockdiagramm der Chip-Select-Schaltung aufgeführt ist.

Die Adressbereiche, mit denen das Chip-Select-Signal abgedeckt werden soll, ist variabel. Dabei kann zwischen einer Blockgrösse von 2K, 8K, 16K, 64K, 128K, 256K, 512K oder 1MByte gewählt werden. Dadurch werden die entsprechenden Adressleitungen gleichgültig (z.B. Blockgrösse = 2Kbyte -> A11 bis A23 gleichgültig). Ausserdem kann mit Hilfe des CSBARx-Registers die notwendige Basisadresse eingestellt werden.

Um die verschiedenen Anwendungsmöglichkeiten zu konfigurieren, steht ein Chip-Select-Option-Register CSORx für jeden Kanal zur Verfügung. Um eine Chip-Select-Logik zu generieren, muss der Aufbau des Registers *(Abb. 2.1.5)* beachtet werden. Deshalb soll im folgenden darauf kurz eingegangen werden.

CSOR0–CSOR10 — Chip-Select Option Registers **$YFFA4E–YFFA76**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE	BYTE		R/$\overline{W}$		STRB	$\overline{DSACK}$				SPACE		IPL			$\overline{AVEC}$
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Abb. 2.1.5: Aufbau des CSORx-Registers (Quelle: [4])

[4]: Motorola, MC68332 User's Manual, Seiten 4-29 und 4-37

In *Tab. 2.1.1* sind die Optionen aufgeführt, die zur Funktion der Chip-Select-Logik notwendig sind.

Mode	Byte	R/$\overline{W}$	STRB	$\overline{DSACK}$	Space	IPL	$\overline{AVEC}$
0 = ASYNC	00 = Off	00 = Rsvd	0 = $\overline{AS}$	0000 = 0 WAIT	00 = CPU SP	000 = All	0 = Off
1 = SYNC	01 = Lower	01 = Read	1 = $\overline{DS}$	0001 = 1 WAIT	01 = User SP	001 = Level 1	1 = On
	10 = Upper	10 = Write		0010 = 2 WAIT	10 = Supv SP	010 = Level 2	
	11 = Both	11 = Both		0011 = 3 WAIT	11 = S/U SP	011 = Level 3	
				0100 = 4 WAIT		100 = Level 4	
				0101 = 5 WAIT		101 = Level 5	
				0110 = 6 WAIT		110 = Level 6	
				0111 = 7 WAIT		111 = Level 7	
				1000 = 8 WAIT			
				1001 = 9 WAIT			
				1010 = 10 WAIT			
				1011 = 11 WAIT			
				1100 = 12 WAIT			
				1101 = 13 WAIT			
				1110 = F term			
				1111 = External			

Tab 2.1.1: Optionen des CSORx-Registers (Quelle: [4])

In Bit 15 kann zwischen asynchroner und synchroner Arbeitsweise ausgewählt werden. Für die meisten Anwendungsgebiete ist die asynchrone Arbeitsweise nötig, da hier eine Synchronisierung durch die Signale AS# oder DS# vorgenommen wird. Das Byte-Feld entscheidet, ob bei einem 16-Bit-Port nur das untere, das obere oder beide Bytes genutzt werden sollen. Mit dem R/W-Feld kann festgelegt werden, ob mit dem Chip-Select ein Port nur gelesen, nur geschrieben oder ob beides erlaubt sein soll. In Bit 10 wird bei einer asynchronen Arbeitsweise (siehe Bit 15) festgelegt, ob eine Synchronisierung der Verarbeitung mit dem Adressstrobe AS# oder dem Datenstrobe DS# erfolgen soll. Mit den vier Bits des DSACK#-Feldes können die Anzahl der Waitstates programmiert werden, die vom Prozessor abgewartet werden sollen, bis am Bus ein Data-Strobe-Acknowledge anliegt. Diese Funktion ist nur in der asynchronen Arbeitsweise aktiv und ein Waitstate entspricht einem Taktzyklus.

Während der Programmierung der A/D-Wandlerkarte traten einige Zeitprobleme auf, die jedoch durch eine Generierung von 13 Waitstates für den DSACK#-Impuls gelöst werden konnten. Wahrscheinlich ist die Verarbeitung des A/D-Wandlers und des nachfolgenden GAL-Bausteins zu langsam für die sonst schnelle Arbeitsweise des Kontrollers. Damit zeigt sich wieder, dass nicht der Kontroller die Verarbeitungsgeschwindigkeit des Systems bestimmt, sondern die an ihn geschaltete Peripherie.

Mit den Space-Bits wird der Adressraum überprüft, der durch das Chip-Select angesprochen wird. Dabei wird ein Vergleich mit den beiden Bits vorgenommen. Im IPL-Feld wird die Interruptpriorität in einem Interrupt-Acknowledge-Zyklus festgelegt. Das Bit 0 dient zur Angabe des Interruptvektors.

[4]: Motorola, MC68332 User's Manual, Seite 4-37

Im Betriebssystem für die A/D-Wandlerkarte wurden zwei verschiedene CS-Signale generiert. Nach langer Suche in den bereits vorliegenden Softwaresystemen zum Evaluationkit (siehe Abschnitt 2.1.6) und des Monitorprogramms wurde festgestellt, dass bereits einige CS-Signale festgelegt wurden (einen Ausschnitt aus dem Monitor-Initprogramm zeigt dazu *Abb. 2.1.6*). Das CS3-Signal wurde durch das Monitorprogramm des Evaluationkit gebunden und für den Bus-Error verwendet. Da es in der Anwendung zu Konflikten mit unserer Soft- bzw. Hardware kam, wurde dieser CS3 desaktiviert. Für die Anwendung in der Hardware stehen in der vorhandenen Entwicklungsumgebung nur noch zwei CS-Siganle (CS6 und CS7) zur Verfügung. Als CS-Signal wurde CS6 verwendet, dessen Basisadresse auf 20000h festgelegt wurde. Die Blockgrösse ist mit 2Kbyte vereinbart wurden (siehe Softwarebeschreibung Abschnitt 3).

```
MOVE.L   #0xFFFFFFFF, (A1)+   ; Chip select and pin assignment
MOVE.L   #0x080468B0, (A1)+   ; Boot cs
MOVE.L   #0x0003503E, (A1)+   ; cs0 : bcc RAM write (MSB)
MOVE.L   #0x0003303E, (A1)+   ; cs1 : bcc RAM write (LSB)
MOVE.L   #0x0003683E, (A1)+   ; cs2 : bcc RAM read (word access)
MOVE.L   #0x000778B0, (A1)+   ; cs3 : Not used (avoid bus error)
MOVE.L   #0xFFF8680F, (A1)+   ; cs4 : Abort auto vector
MOVE.L   #0xFFE8783F, (A1)+   ; cs5 : cd for MC68881/882
MOVE.L   #0x00000000, (A1)+   ; cs6 : Not used
MOVE.L   #0x00000000, (A1)+   ; cs7 : Not used
MOVE.L   #0x01036870, (A1)+   ; cs8 : RAM U1/U3 (word read access)
MOVE.L   #0x01033030, (A1)+   ; cs9 : RAM U1 (LSB)
MOVE.L   #0x01035030, (A1)+   ; cs10: RAM U1 (MSB)
```

Abb. 2.1.6: Ausschnitt aus dem Monitorprogramm

2.1.4 Interner RAM

Zur Speicherung von Daten enthält der **MC68332** einen statischen RAM mit einer Grösse von 2 Kbyte. Der Zugriff kann zum Lesen oder Schreiben mit wahlweise 8, 16 oder 32 Bit erfolgen. Der RAM kann entweder von der CPU32 oder von der TPU benutzt werden. Bei Nutzung durch die TPU ist die CPU32 nicht berechtigt einen Zugriff auf den RAM-Bereich durchzuführen. Zur Steuerung der Speicherarbeit sind 3 Register vorhanden.

2.1.5 Queued Serial Module (QSM)

Dieser Modul (siehe *Abb. 2.1.7*) dient der Kommunikation mit externen Baugruppen unter Nutzung von zwei völlig getrennten seriellen Schnittstellen (QSPI und SCI).

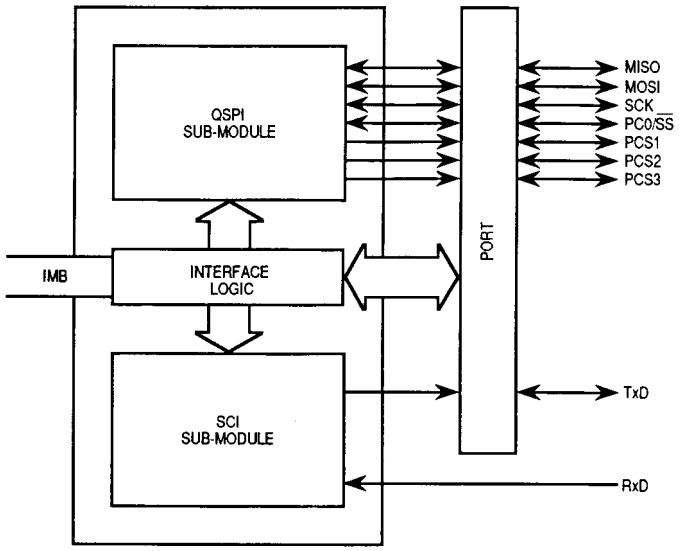


Abb. 2.1.7: Blockschaltbild QSM (Quelle: [4])

Die synchrone Schnittstelle QSPI ist eine serielle voll-duplex-fähige Schnittstelle zur Kommunikation mit anderen Mikrokontrollern oder externen Peripheriegeräten. Sie ist voll kompatibel zu den SPI-Schnittstellen von anderen Motorola-Schaltkreisen der **M68HC11**- und **M68HC05**-Familie.

Im Gegensatz dazu, arbeitet die andere Schnittstelle SCI asynchron. Sie ist für die Verbindung mit externen Geräten über einen asynchronen Bus zuständig. Diese Schnittstelle dient in der vorliegenden Entwicklungsumgebung zur Verbindung mit dem PC, der zur Softwareentwicklung benötigt wurde.

[4]: Motorola, MC68332 User's Manual, Seite 5-2

2.1.6 Entwicklungsumgebung

Evaluationkit M68332EVK

Zur Entwicklung der Hard- und Software stand ein Evaluationkit **M68332EVK** der Firma Motorola zur Verfügung. Es dient zur Erarbeitung und Testung der Hard- bzw. Softwarelösung eines bestimmten Problems. Damit kann dem Entwickler die Entscheidung, ob ein System zur Lösung des Problems geeignet ist, erleichtert werden. Den Aufbau des Evaluationboard zeigt *Abb. 2.1.8* .

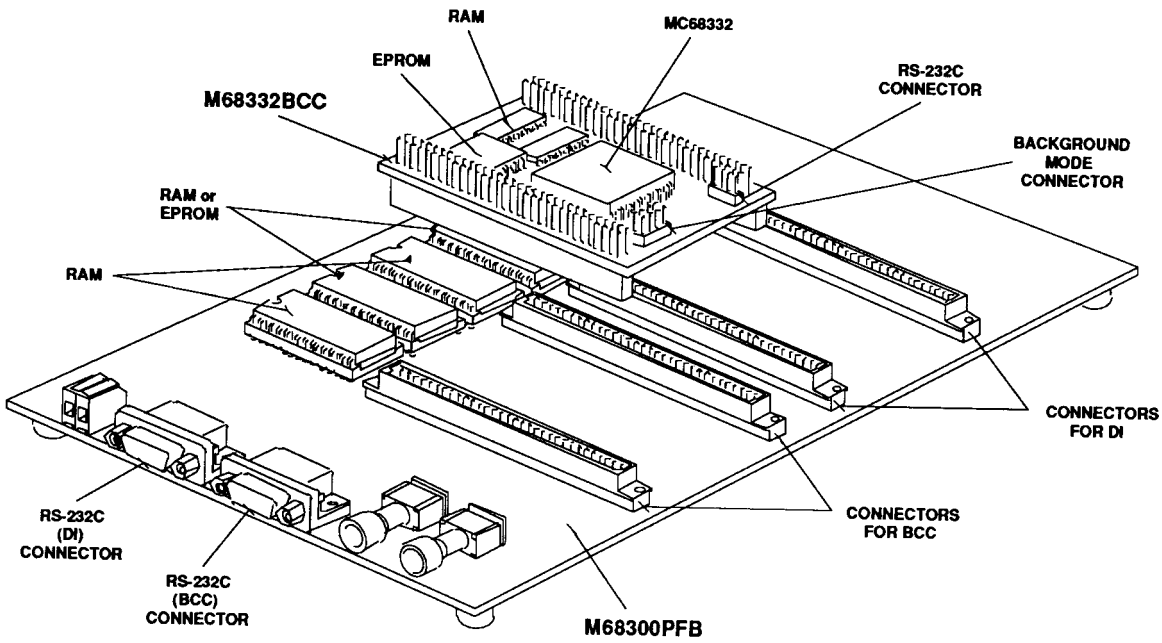


*Abb. 2.1.8: Aufbau Evaluationboard **M68332EVK** (Quelle: [5])*

Das Board besteht im wesentlichen aus der Hauptplatine (Platform Board), der Prozessorplatine (Business Card Computer) und dem Debug-Modul (CPU32Bug). Auf den Aufbau soll hier nicht weiter eingegangen werden, für eventuelle Fragen steht ihnen das User's Manual des EVK [5] zur Verfügung.

Die Verbindung zur Programmierumgebung (einem PC) wird über eine RS-232C-Schnittstelle (BCC-Connector) hergestellt. Als Spannungsversorgung werden +5V benötigt. Die Stromaufnahme beträgt min. 700mA.

[5]: Motorola, M68332 Evaluation Kit User's Manual, Seite 2-2

C-Entwicklungsumgebung HICROSS

Für die komfortable Softwareentwicklung mit dem EVK stellt die Firma HIWARE eine C-Entwicklungsumgebung HICROSS zur Verfügung. Diese unter WINDOWS laufende Software besteht aus verschiedenen Tools. Zum einen wird ein Compiler, ein Linker und ein Debugger/Downloader angeboten, zum anderen kann noch eine Simulationssoftware genutzt werden. Da der Debugger, im Gegensatz zur Simulationssoftware, die Software auf dem Board testet, muss eine Verbindung über die serielle Schnittstelle hergestellt werden.

Zum Editieren der zu erstellenden Software dient das Programm WinEdit. Dieses, nicht aus dem Hause HIWARE stammende, Programm lässt auch den Aufruf der anderen Tools über Icon's recht komfortabel zu.

Leider entstanden durch die Installation einer neuen Programmversion von HICROSS V2.6 grosse Probleme bei der Zusammenarbeit mit dem Evaluationboard. Ein downloaden der Software war nicht mehr möglich, da wahrscheinlich die vom Programm gewählte Baudrate zu hoch eingestellt war. Leider liess sich diese Einstellung nicht mehr, im Gegensatz zur älteren Version, anpassen. Da nach langer Fehlersuche noch keine Lösung gefunden wurde, ist die ältere Version wieder zum Einsatz gekommen. Die Lösung des Problems wurde auf einen späteren Zeitpunkt verschoben.

Zur Arbeit mit der Entwicklungssoftware HICROSS kann das User Manual [12] empfohlen werden.

[12]: HIWARE, User Manual HICROSS V2.5

2.2 A/D-Wandlung

2.2.1 Übersicht über die A/D-Wandlerkarte

In der Aufgabenstellung zu dieser Arbeit wurde eine Erweiterung der A/D-Wandlerkanäle gefordert. Das bisher vorliegende Ergebnis der Projektarbeit Oechslin/Gahlinger [10] bestand aus einer Europakarte mit je einem A/D- bzw. D/A-Wandlerkanal. Da der Aufbau dieser Karte jedoch einige Mängel aufweist und ausserdem dieses Hardwarekonzept nicht erweiterbar ist, wurde eine völlig neue Hardwarestruktur der A/D-Wandlerkarte geschaffen. Da das später zu regelnde Wippenmodell insgesamt vier messbare Zustandsgrössen aufweist, ist mindestens eine Hardware mit vier Analog-Digital-Wandlerkanälen zu schaffen.

Im weiteren Entwurf der Hardware, wurde ein Konzept erarbeitet, mit dem eine einfache Erweiterung der gesamten Schaltung ermöglicht wird. Das Hardwaresystem besteht aus universellen Europakarten mit jeweils zwei A/D-Wandlerkanälen, realisiert mit dem Schaltkreis **MAX120** und vorgeschaltenem Antialiasing-Filter, den Eingangsverstärkern und der Chip-Select-Logik.

Die Chip-Select-Logik wurde so ausgelegt, dass eine Erweiterung der Hardware auf 8 Kanäle, also vier Europakarten, nichts im Wege steht. Es wurde angestrebt, nur ein Print für die Herstellung der A/D-Wandlerkarten zu entwerfen. Dadurch entstand eine relativ einfache Hardware auf der Europakarte, die jederzeit durch Nachnutzer um jeweils 2 Kanäle erweiterbar ist.

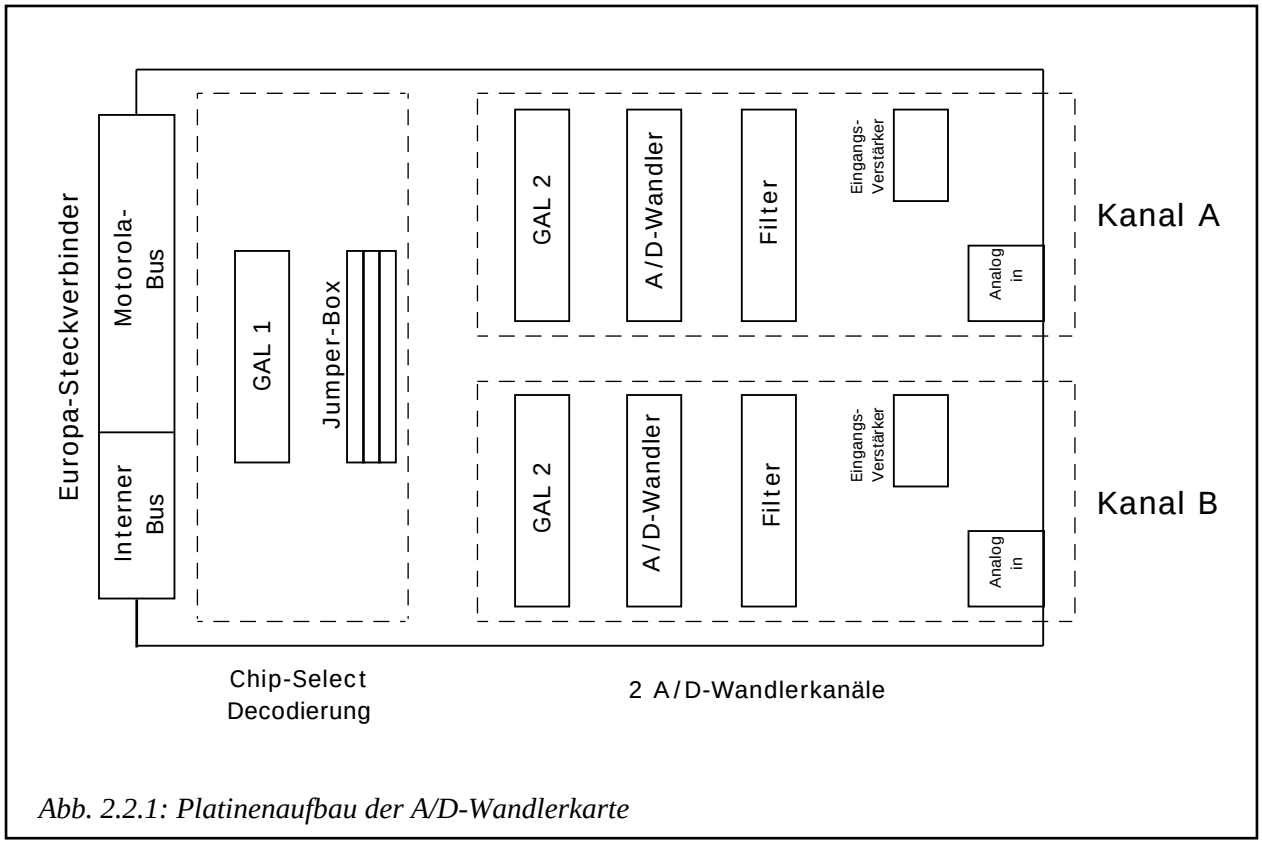


Abb. 2.2.1: Platinenaufbau der A/D-Wandlerkarte

[10]: TWI; Joachim Oechslin/Daniel Gahlinger, Projektarbeit I/93, Digitale Regelung mit dem Motorola MC68332

Der grundsätzliche Aufbau der Europakarte ist in *Abb. 2.2.1* dargestellt. Wie ersichtlich ist, wurde die CS-Decodierung auf dem Print mit integriert. Da bei bis zu 8 Kanälen aber nur eine Chip-Select-Decodierung notwendig ist, wird nur auf einer Karte dieser Hardware bestückt und beschaltet. Auf den weiteren Europakarten müssen an dieser Stelle nur noch wenige Brücken gesetzt werden. Das vom CS-Decoder gebildete CONV-Signal (Adresse 20000h) und die einzelnen READ-Leitungen (Adressen 20002h - 20010h) werden über den internen Bus zu allen A/D-Karten geführt. Dort können die Adressbereiche je nach Anwendung mittels Jumper gesteckt werden. Somit sind die Adressen für die einzelnen Kanäle frei wählbar und können jederzeit geändert werden. Wie aus *Tabelle 2.2.1* ersichtlich ist, werden hardwareseitig nur die Adressleitungen A1 bis A10 decodiert. Um sicher auf den Adressbereich zuzugreifen, muss mittels der CS-Generierung, die intern im **MC68332** vorhanden ist, der Anwender-Adressbereich eingestellt werden. Mehr dazu erfahren sie im Abschnitt 2.2.3 und in der Beschreibung der Software im Abschnitt 3.

Bei einer externen CS-Decodierung hätte man eine zusätzliche Platine schaffen müssen, die zudem bei weitem nicht ausgelastet wäre. Sicherlich ist diese Hardwarestruktur noch nicht die technisch günstigste, jedoch stellt sie eine erhebliche Verbesserung gegenüber der bisher, oben kurz erwähnten, vorliegenden Technik dar.

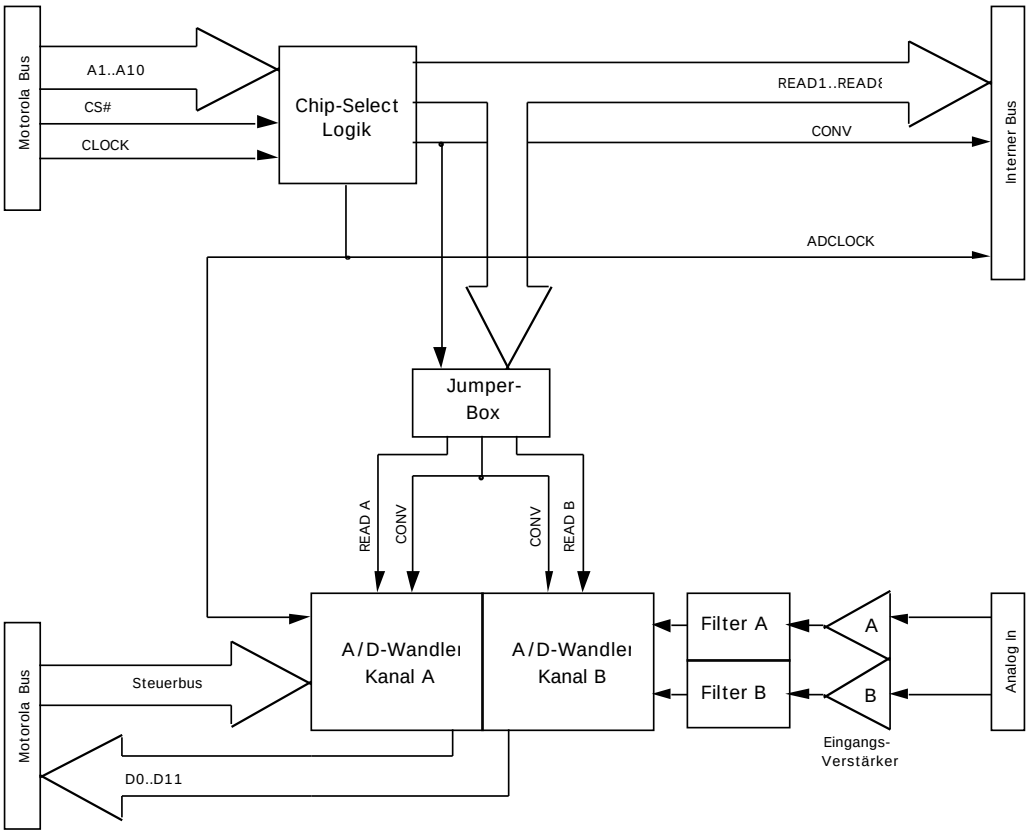
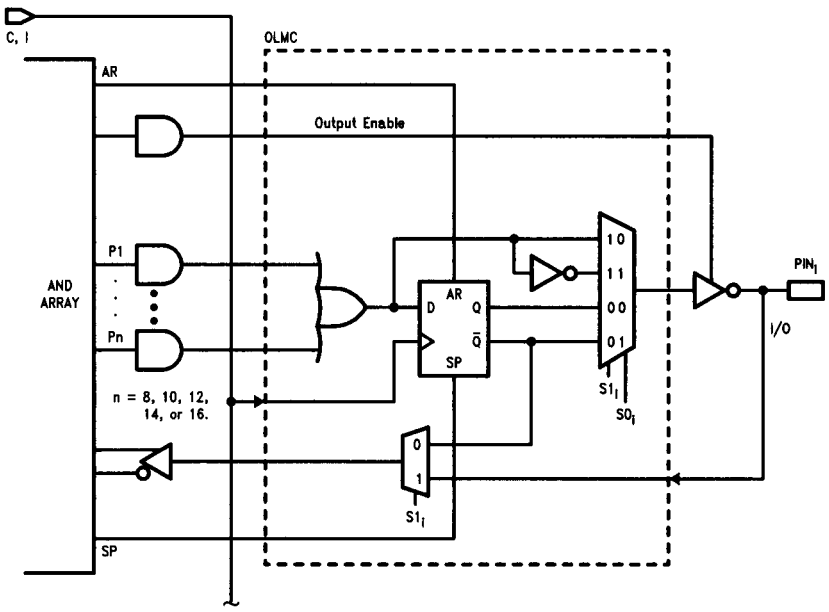


Abb. 2.2.2: Blockschaltbild der A/D-Wandler-Karte

Um sich einen Einblick in die Funktionsweise der A/D-Wandlerkarte verschaffen zu können, ist in *Abb. 2.2.2* das Blockschaltbild der Analog-Digital-Wandlung mit all seinen Komponenten dargestellt. Zum besseren Verständnis der einzelnen Baugruppen sei hier auf die entsprechenden Kapitel zur CS-Decodierung, der Filterung und der A/D-Wandlung hingewiesen.

2.2.2 Einführung in die GAL-Programmierung

Um einen minimalen Schaltkreisaufwand mit den A/D-Wandlerkarten zu erreichen, wurde die Möglichkeit der Verwendung von Generic-Array-Logic Bausteinen (GAL) geprüft. Dabei stand im Vordergrund, die umfangreiche Chip-Select-Decodierung über 24 Bit und die komplizierte Steuerung des A/D-Wandlers **MAX120** mit den elektrisch programmierbaren und löschbaren Schaltkreisen (EEPROM-Technologie) zu ermöglichen.
Die Grundarchitektur der GAL's entspricht denen der Programmable Logic Devices (PLD's), eine Kombination von UND- und ODER-Matrix. Eine wesentliche Erweiterung stellen aber die sogenannten Macrozellen (OLMC) dar. Diese befinden sich an den Ausgängen der ODER-Matrix und können flexibel programmiert werden. Aus dem Logik-Diagramm, dass in *Abb. 2.2.3* dargestellt ist, lässt sich der Aufbau der Ausgangszelle verdeutlichen.



S1	S0	Output Configuration
0	0	Registered/Active Low
0	1	Registered/Active High
1	0	Combinatorial/Active Low
1	1	Combinatorial/Active High

Abb. 2.2.3: GAL 22V10: Logik-Diagramm der Ausgangszelle (Quelle: [3])

Es lassen sich so praktisch alle bekannten Arten von PAL's nachempfinden. Die Ausgangszellen der GAL-Schaltkreise lassen sich durch vier Multiplexer programmieren. Damit erreicht man folgende Möglichkeiten an der GAL-Ausgangszelle:

- kombinatorischer Ein-/Ausgang mit programmierbaren "Output Enable"
- kombinatorische Signalausgabe mit programmierbaren "Output Enable"
- kombinatorische Ausgabe mit programmierbarer Polarität
- Nutzung eines Ein-/Ausgabepins als zusätzlichen Eingang
- Registerausgang mit Tri-State-Möglichkeit und aktivem Low- oder High-Pegel

[3]: National Semiconductor, Programmable Logic Devices Databook and Design Guide, Seite 2-89

Somit bieten sich mit Hilfe der GAL's weite Möglichkeiten in der Mikroprozessortechnik zur Realisierung von kombinatorischen und sequentiellen Schaltungen.

Die Schaltzeiten des auf der A/D-Wandlerkarte zum Einsatz gekommenen Types **GAL 22V10** liegen zwischen 10 und 30 ns. Somit ist er für den Einsatz in Mikroprozessorschaltungen geeignet. Das Blockdiagramm des **GAL 22V10** ist in *Abb. 2.2.4* dargestellt.

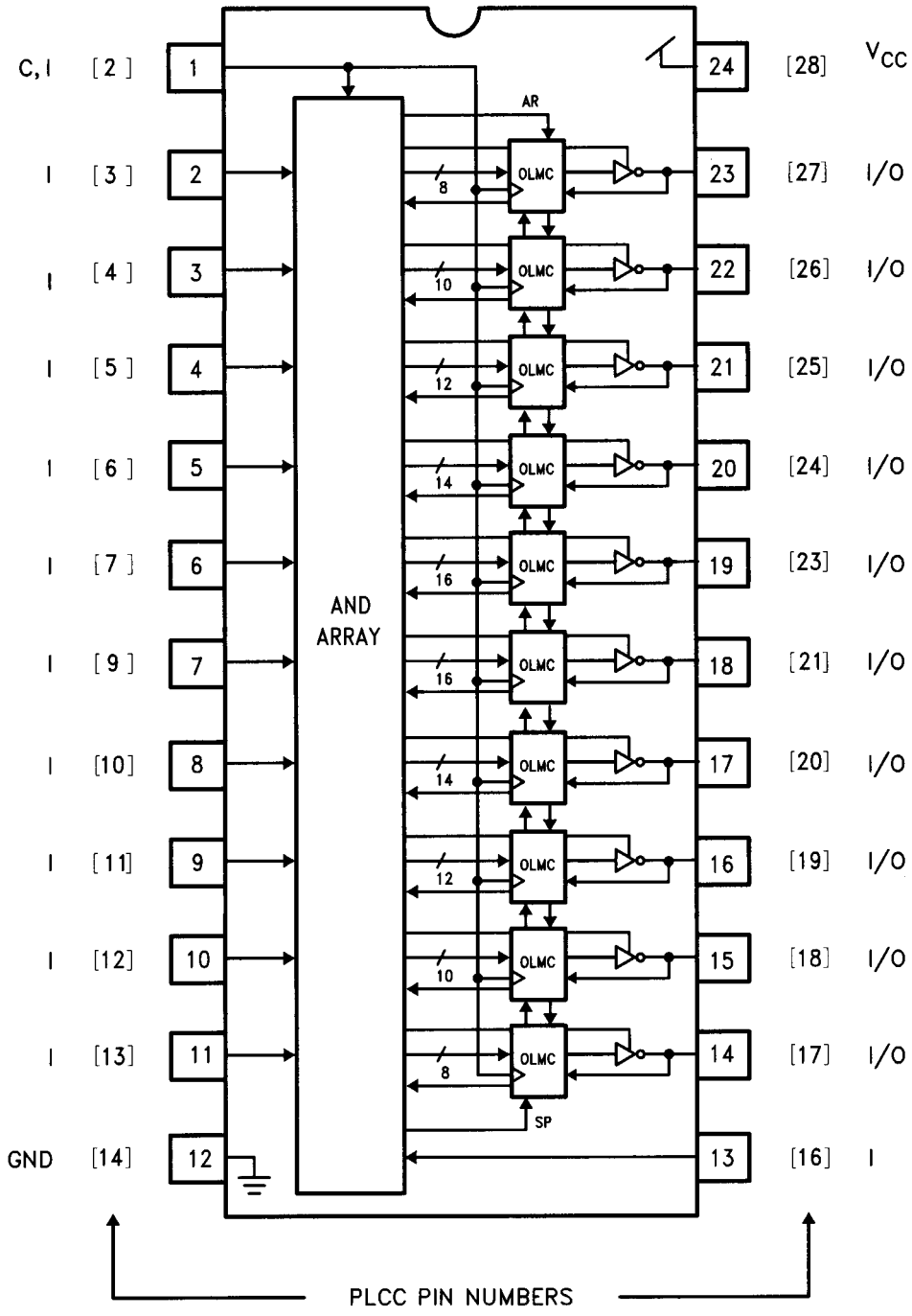


Abb. 2.2.4: GAL 22V10 Blockschaltbild (Quelle: [3])

[3]: National Semiconductor, Programmable Logic Devices Databook and Design Guide, Seite 2-84

Dabei lassen sich die Pin's mit der Bezeichnung "I" als Eingänge und mit "I/O" wahlweise als Ein- oder Ausgänge benutzen. Der Pin 1 ist für den CLOCK als Eingang reserviert.

Ein GAL-Baustein lässt sich einige hundertmal löschen und programmieren. Bei der Programmierung des GAL-Bausteins werden die einzelnen Matrix-Verbindungspunkte und die Konfiguration der Macrozellen definiert. Für die Programmierung der GAL's werden Logic-Compiler und universelle Programmiergeräte eingesetzt. Die Strukturen der vorliegenden GAL's wurden mit Hilfe des PLD-Compiler aus dem ORCAD-Programmpaket entworfen. Die erzeugte *.JED Datei kann im ORCAD-System simuliert werden und dann mit Hilfe eines Programmiergerätes auf die GAL-Bausteine 'gebrannt' werden.

Für Anwender der GAL-Bausteine und zum Verständnis der Programmiersoftware sei auf die ORCAD-Softwarebeschreibung und auf Literatur [6] hingewiesen.

Auf der A/D-Wandlerkarte wurden für die Adressdecodierung und für die Steuerung des A/D-Wandlers je ein **GAL 22V10** verwendet.

2.2.3 Chip-Select-Logik

Ein angestrebtes Ziel dieser Arbeit bestand darin, eine universell einsetzbare Analog/Digital-Wandlerkarte mit mindestens 4 Kanälen zu entwickeln. Dazu wurde eine Chip-Select-Logik aufgebaut, die es ermöglicht bis zu 8 A/D-Wandlerschaltkreise anzusprechen. Zu Beginn der Arbeit am Mikrokontrollersystem wurde eine vollständige Adressdecodierung mit 24 Bit angestrebt, um die A/D-Wandlerkarte auch in anderen Mikrokontrollersystemen anzuwenden. Dazu wurde eine Chip-Select-Decodierung mit einem GAL und einem anschliessenden 1 aus 8 -Decoder **74HC138** entwickelt.

Die aufgebaute Hardware funktionierte mit Hilfe der angelegten Adressleitungen, sodass einer Nutzung im Kontrollersystem nichts mehr im Wege stand. Da die Zusammenarbeit im System jedoch nicht funktionierte, begann eine aufwendige Fehlersuche. Dabei wurde festgestellt, dass bereits das Monitorprogramm des Evaluationkit von Motorola einige Chip-Select-Signale generiert und nutzt.

Die Pins des **MC68332** mit denen die Adressleitungen A19 - A23 auf das Board geführt werden, können auch zur Bildung der Chip-Select-Logik mit den dazugehörigen Chip-Select-Signalen CS6 bis CS10 benutzt werden (siehe Blockschaltbild auf *Abb. 2.1.1*). Mit Hilfe der CSPARx-Register wird die Nutzung der Pin's voreingestellt. Dabei wird mit Hilfe einer 0- oder 1-Belegung festgelegt, ob das Pin als Adresse oder als Chip-Select benutzt werden soll.

Vom Entwickler des EVK wird dem CS3-Signal eine Funktion zugeordnet, die leider aus den fehlerhaften Hard- und Softwaremanuals des Herstellers nicht klar ersichtlich ist. Deshalb wurde mit Bedauern auf die vollständige und deshalb sichere Adressdecodierung verzichtet und die interne Chip-Select-Generierung des Motorola **MC68332** genutzt (siehe Kap. 2.1). Um keine Konflikte mit dem Monitorprogramm enstehen zu lassen, wurde der CS3 desaktiviert (siehe Betriebssystem der AD-Wandlerkarte im Anhang A). Zur Nutzung unserer Hardware wurde der CS6 generiert, da im Monitorprogramm keine Verwendung vorgesehen ist. An dieser Stelle soll nochmals betont werden, dass die vorliegenden Manuals und Betriebshandbücher der Firma Motorola nicht vollständig und exakt ausgeführt sind. Es sollte bei jeder weiteren Anwendung des Evaluationkit **M68332EVK** dieser Umstand bedacht werden, um aufwendige Fehlersuche in der Anwendersoft- bzw. Anwenderhardware zu vermeiden.

[6]: Technikum Winterthur; Prof. J. Zeman, MC-Stoff, Kapitel PLD's

Auf Grund der vorliegenden Hardware ist der in *Abb. 2.2.5* aufgezeigte Speicheraufbau für den Anwender existent. Für die Arbeit der einzelnen A/D-Wandlerkanäle wurde der Adressbereich ab 20000h gewählt.

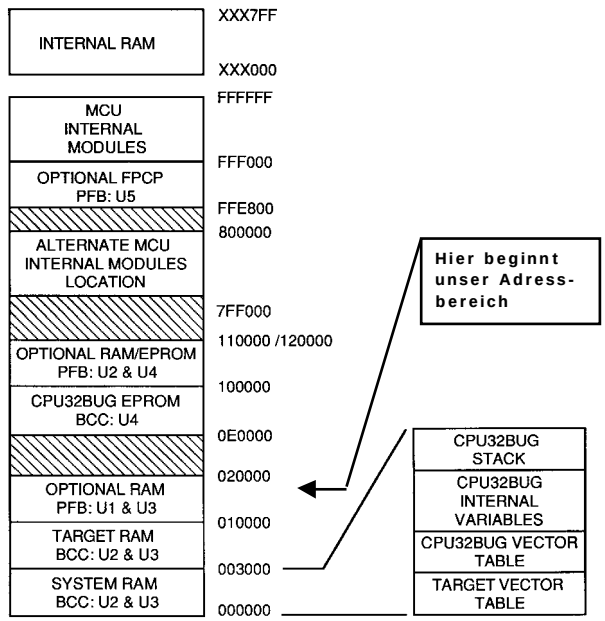


Abb. 2.2.5: M68332 EVK Memory Map (Quelle: [5])

Die einzelnen Adressen wurden wie folgt zugeteilt:

Adresse	Funktion	Adressbitbelegung										
		A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
20000h	A/D-Konversion starten	0	0	0	0	0	0	0	0	0	0	0
20002h	1. A/D-Wandler	0	0	0	0	0	0	0	0	0	1	0
20004h	2. A/D-Wandler	0	0	0	0	0	0	0	0	1	0	0
20006h	3. A/D-Wandler	0	0	0	0	0	0	0	0	1	1	0
20008h	4. A/D-Wandler	0	0	0	0	0	0	0	1	0	0	0
2000Ah	5. A/D-Wandler	0	0	0	0	0	0	0	1	0	1	0
2000Ch	6. A/D-Wandler	0	0	0	0	0	0	0	1	1	0	0
2000Eh	7. A/D-Wandler	0	0	0	0	0	0	0	1	1	1	0
20010h	8. A/D-Wandler	0	0	0	0	0	0	1	0	0	0	0

Tab. 2.2.1: Adressbelegung

[5]: Motorola, M68332EVK Evaluation Kit User's Manual, Seite 4-7

Die Adressleitung A0 besitzt immer den logischen Wert 0, da im 16-Bit-System nur gerade Adressen auftauchen können. Die Adresse 20000h dient dem A/D-Wandler **MAX120** als Adresse zum Einlesen des analogen Wertes in den Schaltkreis. Es muss also vor jedem Analog-Digital-Wandelvorgang diese Adresse angesprochen werden. Das vorliegende Testprogramm zum Einlesen eines A/D-Kanals und zur anschliessenden Ausgabe eines zugehörigen PWM-Signals nutzt die Adresse 20002h.

Durch die Verknüpfung der Adressleitungen A1 bis A10 und dem Chip-Select 6 werden die entsprechenden Adressen der einzelnen A/D-Wandler gebildet. Ausser der Decodierung der Adressleitungen wird der GAL 1 auch zur Bildung eines Taktsignals benötigt. Da über die Bussteckverbinder des Evaluationkit nur der Sytemtakt mit 16 MHz verfügbar ist, der A/D-Wandler **MAX120** aber nur mit maximal 8 MHz arbeiten kann, ist ein Teiler im GAL 1 programmiert worden. Damit ergibt sich das in *Abb 2.2.6* gezeigte Blockschaltbild.

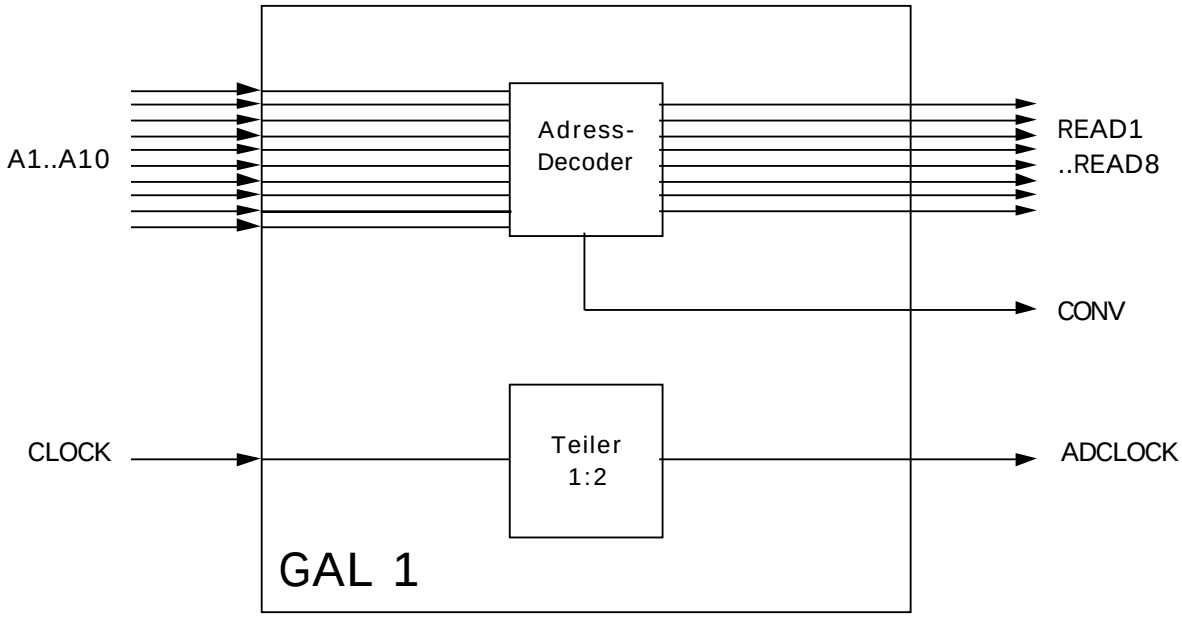


Abb. 2.2.6: GAL 1 Blockschaltbild

Für die Programmierung des GAL 1 wurde ein Zustandsautomat entworfen. Er besitzt zwei Zustände, die immer durchlaufen werden. Dadurch ergibt sich der halbierte Prozessortakt mit 8 MHz am Ausgang ADCLOCK des GAL 1. Die *Abb. 2.2.7* zeigt den Zustandsgraph des Zustandsautomaten GAL 1.

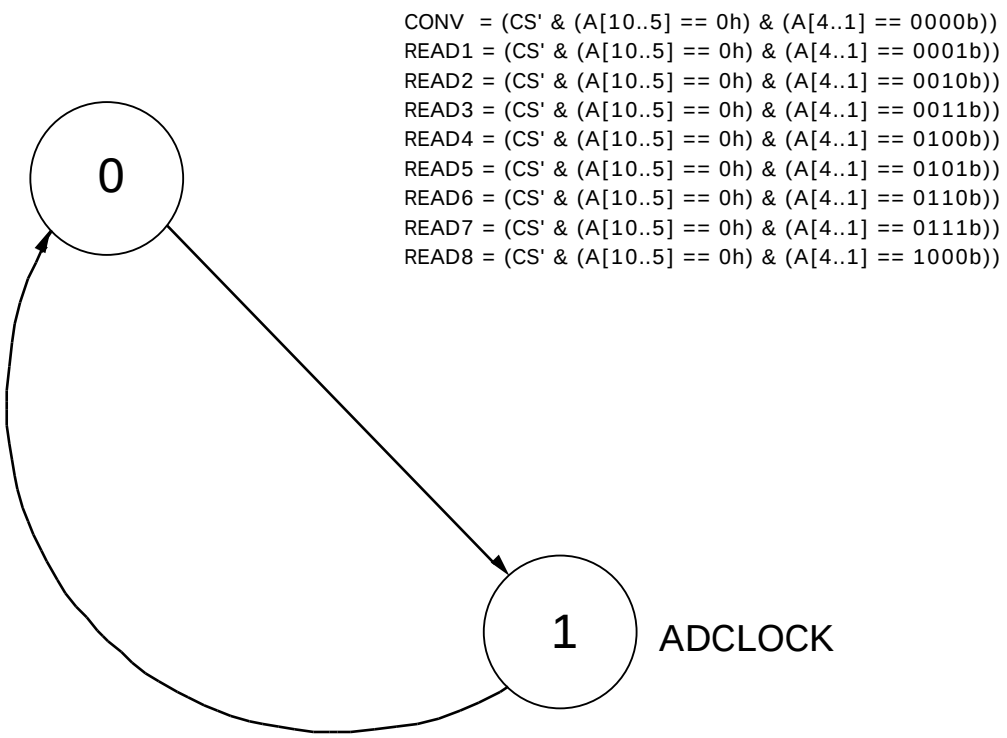


Abb. 2.2.7: GAL 1 Zustandgraph

Programmierung des GAL 1

Mit Hilfe eines beliebigen Editors und der anschliessenden Einbindung in den PLD-Compiler des ORCAD-Programmpaketes kann die erforderliche JEDEC-Datei erzeugt werden. Nach dem Übersetzen des Source-File erzeugt der ORCAD-PLD-Compiler eine Datei *.LST (*Abb. 2.2.8*), aus der die Verbindungsliste, die Zustandsmaschine und die Pinbelegung des programmierten GAL ersichtlich ist. Wobei der Programmierer die Zuordnung der Pin's sowohl vorgeben, als auch die Vergabe dem Compiler überlassen kann. Wenn die Belegung der Pin's durch den ORCAD-Compiler vorgenommen wird, besitzt dieser natürlich einen grösseren Spielraum bei der Gestaltung der Hardware im GAL, jedoch erfährt der Anwender erst aus dem Listing die resultierende Zuordnung der Pin's.

```
OrCAD PLD COMPILER  V4.00 6/19/92  (Source file DIPGAL1.PLD)

  1  |GAL22V10 2:A1,3:A2,4:A3,5:A4,6:A5,7:A6,8:A7,9:A8,10:A9,11:A10,13:CS,
  2  |         14:CONV,15:READ1,16:READ2,17:READ3,18:READ4,
  3  |         19:READ5,20:READ6,21:READ7,22:READ8,23:ADCLOCK,
  4  |         clock:CLOCK
  5  |
  6  |
  7  |TITLE: " Adressdecodierung        "
  8  |       " dipgal1.PLD    Version:3 "
  9  |       " rogoli 26.11.93          "
 10  |
 11  |
 12  |CONV  = (CS' & (A[10..5] == 0h) & (A[4..1] == 0000b))
 13  |READ1 = (CS' & (A[10..5] == 0h) & (A[4..1] == 0001b))
 14  |READ2 = (CS' & (A[10..5] == 0h) & (A[4..1] == 0010b))
 15  |READ3 = (CS' & (A[10..5] == 0h) & (A[4..1] == 0011b))
 16  |READ4 = (CS' & (A[10..5] == 0h) & (A[4..1] == 0100b))
 17  |READ5 = (CS' & (A[10..5] == 0h) & (A[4..1] == 0101b))
 18  |READ6 = (CS' & (A[10..5] == 0h) & (A[4..1] == 0110b))
 19  |READ7 = (CS' & (A[10..5] == 0h) & (A[4..1] == 0111b))
 20  |READ8 = (CS' & (A[10..5] == 0h) & (A[4..1] == 1000b))
 21  |
 22  |
 23  |CONDITIONING:  CLOCK // ADCLOCK
 24  |
 25  |PROCEDURE: ADCLOCK
 26  |{
 27  |  STATE0. ->STATE1
 28  |  STATE1. ->STATE0
 29  |}
 30  |
 31  | Vectors:
 32  |{
 33  |  Display "A[10..1]: ",A[10..1],\
 34  |          "  CONV: ",CONV,"  READ[1..8]: ",READ[1..8]
 35  |
 36  |  TEST A[10..1]=0h;CS=1
 37  |  TEST A[10..1]=0h;CS=0
 38  |  TEST A[10..1]=1h;CS=1
 39  |  TEST A[10..1]=1h;CS=0
 40  |  TEST A[10..1]=2h;CS=1
 41  |  TEST A[10..1]=2h;CS=0
 42  |  TEST A[10..1]=3h;CS=1
 43  |  TEST A[10..1]=3h;CS=0
 44  |  TEST A[10..1]=4h;CS=1
 45  |  TEST A[10..1]=4h;CS=0
 46  |  TEST A[10..1]=5h;CS=1
 47  |  TEST A[10..1]=5h;CS=0
 48  |  TEST A[10..1]=6h;CS=1
 49  |  TEST A[10..1]=6h;CS=0
 50  |  TEST A[10..1]=7h;CS=1
 51  |  TEST A[10..1]=7h;CS=0
 52  |  TEST A[10..1]=8h;CS=1
 53  |  TEST A[10..1]=8h;CS=0
 54  |
 55  |  END
 56  |}
```

```
STATE TABLE FOR ADCLOCK

                              State Number
State Label                 --------------------
                            Decimal Binary Level

STATE0                         0      0      L
STATE1                         1      1      H


RESOLVED EXPRESSIONS (Reduction 2)

Signal name      Row   Terms

CONV             123   A1' A2' A3' A4' A5' A6' A7' A8' A9' A10' CS'

READ1            112   A1  A2' A3' A4' A5' A6' A7' A8' A9' A10' CS'

READ2             99   A1' A2  A3' A4' A5' A6' A7' A8' A9' A10' CS'

READ3             84   A1  A2  A3' A4' A5' A6' A7' A8' A9' A10' CS'

READ4             67   A1' A2' A3  A4' A5' A6' A7' A8' A9' A10' CS'

READ5             50   A1  A2' A3  A4' A5' A6' A7' A8' A9' A10' CS'

READ6             35   A1' A2  A3  A4' A5' A6' A7' A8' A9' A10' CS'

READ7             22   A1  A2  A3  A4' A5' A6' A7' A8' A9' A10' CS'

READ8             11   A1' A2' A3' A4  A5' A6' A7' A8' A9' A10' CS'

ADCLOCK            2   ADCLOCK'


SIGNAL ASSIGNMENT
                                        Rows
 Pin    Signal name   Column     --------------    Activity
                                Beg Avail Used

  1.     CLOCK           0        -    -    -        High    (Clock)
  2.     A1              4        -    -    -        High
  3.     A2              8        -    -    -        High
  4.     A3             12        -    -    -        High
  5.     A4             16        -    -    -        High
  6.     A5             20        -    -    -        High
  7.     A6             24        -    -    -        High
  8.     A7             28        -    -    -        High
  9.     A8             32        -    -    -        High
 10.     A9             36        -    -    -        High
 11.     A10            40        -    -    -        High
 13.     CS             42        -    -    -        High
 14.     CONV           38      122    9    1        High    (Three-state)
 15.     READ1          34      111   11    1        High    (Three-state)
 16.     READ2          30       98   13    1        High    (Three-state)
 17.     READ3          26       83   15    1        High    (Three-state)
 18.     READ4          22       66   17    1        High    (Three-state)
 19.     READ5          18       49   17    1        High    (Three-state)
 20.     READ6          14       34   15    1        High    (Three-state)
 21.     READ7          10       21   13    1        High    (Three-state)
 22.     READ8           6       10   11    1        High    (Three-state)
 23.     ADCLOCK         3        1    9    1        High    (Registered)
 25.     -               -        0    1    0
 26.     -               -      131    1    0
                                ---- ----
                                 132   10  (8%)


I200  No fatal errors found in source code.
I201  No warnings.
```

```
OrCAD PLD
Type:       PAL22V10
Title:      Adressdecodierung
            dipgal1.PLD    Version:3
            rogoli 26.11.93
*
QP24* QF5828* QV1024*
F0*
L0044 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L0088 11 01 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L0440 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L0484 11 11 10 11 10 11 10 11 01 11 10 11 10 11 10 11 10 11 10 11 10 10 *
L0924 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L0968 11 11 01 11 01 11 01 11 10 11 10 11 10 11 10 11 10 11 10 11 10 10 *
L1496 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L1540 11 11 10 11 01 11 01 11 10 11 10 11 10 11 10 11 10 11 10 11 10 10 *
L2156 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L2200 11 11 01 11 10 11 01 11 10 11 10 11 10 11 10 11 10 11 10 11 10 10 *
L2904 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L2948 11 11 10 11 10 11 01 11 10 11 10 11 10 11 10 11 10 11 10 11 10 10 *
L3652 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L3696 11 11 01 11 01 11 10 11 10 11 10 11 10 11 10 11 10 11 10 11 10 10 *
L4312 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L4356 11 11 10 11 01 11 10 11 10 11 10 11 10 11 10 11 10 11 10 11 10 10 *
L4884 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L4928 11 11 01 11 10 11 10 11 10 11 10 11 10 11 10 11 10 11 10 11 10 10 *
L5368 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L5412 11 11 10 11 10 11 10 11 10 11 10 11 10 11 10 11 10 11 10 11 10 10 *
L5808 10 11 11 11 11 11 11 11 11 11 *
C6793*

I202  11/26/93  3:12 pm  (Friday)
I203  Memory utilization 2742/21767 (13%)
I204  Elapsed time 5 seconds
```

Abb. 2.2.8: Datei DIPGAL1.LST

Die entworfene Logik kann dann im ORCAD-PLD-Programm auf einfache Weise getestet werden. Dazu ist dem Source-File ein TEST-Vector anzufügen, der übersetzt wird und dann mit dem ORCAD-TEST-VECTOR-GENERATOR simuliert werden kann. Dabei entsteht als Ergebnis eine Datei mit der Erweiterung *.LOG, aus der die Abläufe der Zustandsmaschine und die jeweiligen gesetzten Ausgangszustände der OUTPUT-Pin's ersichtlich sind (siehe *Abb. 2.2.9*).

```
OrCAD TEST VECTOR GENERATOR  V4.00 6/19/92

|{
|  Display "A[10..1]: ",A[10..1],\
|          "  CONV: ",CONV,"  READ[1..8]: ",READ[1..8]

|
|  TEST A[10..1]=0h;CS=1
A[10..1]: 0000000000  CONV: 0  READ[1..8]: 00000000

|  TEST A[10..1]=0h;CS=0
A[10..1]: 0000000000  CONV: 1  READ[1..8]: 00000000

|  TEST A[10..1]=1h;CS=1
A[10..1]: 0000000001  CONV: 0  READ[1..8]: 00000000

|  TEST A[10..1]=1h;CS=0
A[10..1]: 0000000001  CONV: 0  READ[1..8]: 10000000

|  TEST A[10..1]=2h;CS=1
A[10..1]: 0000000010  CONV: 0  READ[1..8]: 00000000

|  TEST A[10..1]=2h;CS=0
A[10..1]: 0000000010  CONV: 0  READ[1..8]: 01000000

|  TEST A[10..1]=3h;CS=1
A[10..1]: 0000000011  CONV: 0  READ[1..8]: 00000000

|  TEST A[10..1]=3h;CS=0
A[10..1]: 0000000011  CONV: 0  READ[1..8]: 00100000

|  TEST A[10..1]=4h;CS=1
A[10..1]: 0000000100  CONV: 0  READ[1..8]: 00000000

|  TEST A[10..1]=4h;CS=0
A[10..1]: 0000000100  CONV: 0  READ[1..8]: 00010000

|  TEST A[10..1]=5h;CS=1
A[10..1]: 0000000101  CONV: 0  READ[1..8]: 00000000

|  TEST A[10..1]=5h;CS=0
A[10..1]: 0000000101  CONV: 0  READ[1..8]: 00001000

|  TEST A[10..1]=6h;CS=1
A[10..1]: 0000000110  CONV: 0  READ[1..8]: 00000000

|  TEST A[10..1]=6h;CS=0
A[10..1]: 0000000110  CONV: 0  READ[1..8]: 00000100

|  TEST A[10..1]=7h;CS=1
A[10..1]: 0000000111  CONV: 0  READ[1..8]: 00000000

|  TEST A[10..1]=7h;CS=0
A[10..1]: 0000000111  CONV: 0  READ[1..8]: 00000010

|  TEST A[10..1]=8h;CS=1
A[10..1]: 0000001000  CONV: 0  READ[1..8]: 00000000

|  TEST A[10..1]=8h;CS=0
A[10..1]: 0000001000  CONV: 0  READ[1..8]: 00000001

|
|  END
```

Abb. 2.2.9: Datei DIPGAL1.LOG

2.2.4 Der Analog-Digital-Wandlerkanal

Ein A/D-Wandlerkanal besteht im wesentlichen aus vier Komponenten. Nach dem am Eingang befindlichen Verstärker wurde ein Antialiasing-Filter realisiert (siehe Abschnitt 2.2.5). Neben dem A/D-Wandlerschaltkreis **MAX 120** ist eine umfangreiche sequentielle Schaltung notwendig, die die Zusammenarbeit zwischen dem Kontroller und dem A/D-Wandler ermöglicht. Dabei wurde auf die guten Erfahrungen, die bei der Verwendung eines GAL-Bausteines in der Adressdecodierung gemacht wurden, zurückgegriffen. Im folgenden soll auf den Aufbau des A/D-Wandlers und seiner Funktionsweise im Mikrokontrollersystem eingegangen werden.

A/D-Wandler MAX 120

Bei dem A/D-Wandler **MAX 120** handelt es sich um einen Schaltkreis der Firma **MAXIM** mit einer 12-Bit Auflösung und eingebauter Track/Hold-Logik. Er bietet sich für die Anwendung im Mikrokontrollersystem an, da er mit einer Taktfrequenz von bis zu 8 MHz arbeiten kann. Die Konversionszeit vom Auslösen der Wandlung bis zur Ausgabe des 12-Bit-Wertes auf den Datenbus beträgt bei 8 MHz ca. 2 µs. Die interne Track/Hold-Logik wird durch verschiedene Eingänge gesteuert. Die Ein- und Ausgänge sind kompatibel zur CMOS/TTL-Logik und können deshalb direkt mit dem Mikrokontrollersystem verbunden werden. Die Datenausgänge sind für den Busbetrieb mit Tri-State-Stufen ausgerüstet. Das Funktionsdiagramm in *Abb. 2.2.10* stellt den internen Aufbau des Schaltkreises dar.

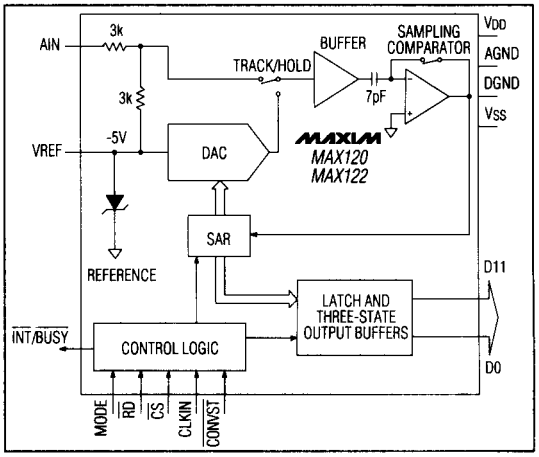


*Abb. 2.2.10: Funktionsdiagramm **MAX120** (Quelle: [7])*

Der A/D-Wandler lässt sich in verschiedenen Modi betreiben. In der Anwendung im Sytem mit dem **MC68332** wurde der FULL-CONTROL-MODE (Pin MODE an V_{DD}) ausgewählt und realisiert. In dieser Betriebsart steht der **MAX120** unter vollständiger Kontrolle durch den Mikrokontroller und wird durch den Eingang CONVST# ("Konversion starten") zum Einlesen des analogen Wertes bewegt (Timingdiagramm *Abb. 2.2.11*). Die Wandlung ist nach ca. 1.6 µs abgeschlossen und wird mit der Generierung des Ausgangs INT# quittiert. Die Übernahme der Daten kann nun durch den Prozessor erfolgen. Dazu muss durch die Steuerleitungen CS# und RD# der Zugriff eingeleitet werden.

[7]: Maxim, 1993 New Releases Databook Volume II, Seite 7-9

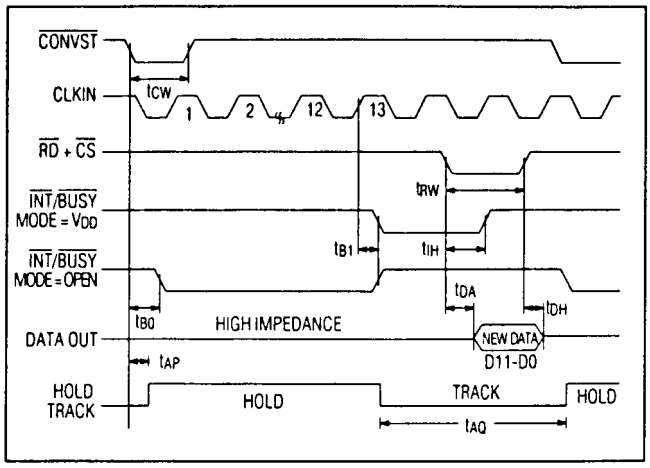


*Abb. 2.2.11: Timing-Diagramm **MAX120** im Full Control Mode (Quelle: [8])*

Der A/D-Wandler nimmt das INT#-Signal zurück und legt die gewandelten Daten als 12-Bit-Wert auf den Datenbus des Systems. Nach dem automatischen Zurücksetzen der Track/Hold-Logik durch den Wandler kann eine neue Konversion eingeleitet werden. Die Steuerung der einzelnen Anforderungs- und Quittierungsleitungen übernimmt in der Anwendung mit dem **MC68332** ein **GAL 22V10** auf den im folgenden Abschnitt näher eingegangen wird. Erwähnenswert ist sicherlich das Datenformat des gewandelten Wertes, welches in *Abb 2.2.12* dargestellt wird. Dabei liefert das 12. Bit den Vorzeichenwert des im Bereich von +5V und -5V liegenden analogen Wertes.

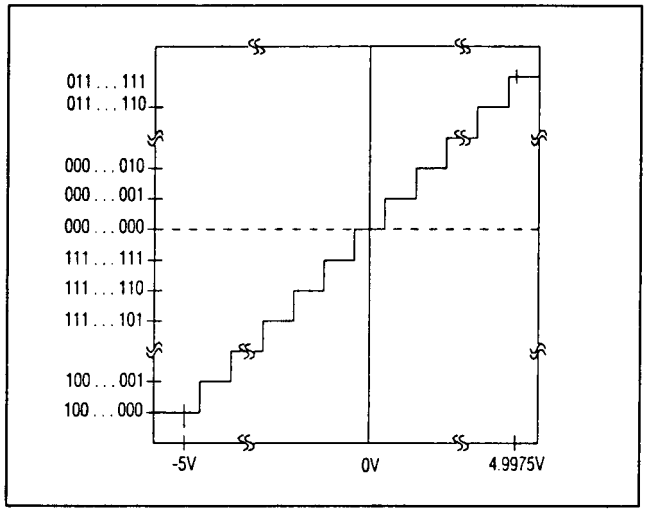


*Abb. 2.2.12: Transferformat des **MAX120** (Quelle: [8])*

[8]: Maxim, Data Pack Nr. 7; A/D-Converters Design Guide, Seiten 7 und 12

Um ein optimales Sytem zu erhalten, sollte auf dem Print die analoge Masse von der digitalen strikt getrennt werden. Die zwei Massekreise sollten erst so nahe wie möglich an der Spannungsversorgung zusammengeführt werden, wie *Abb. 2.2.13* verdeutlicht. Ausserdem sollten keine analogen Leitungen parallel zu digitalen Verbindungen plaziert werden.

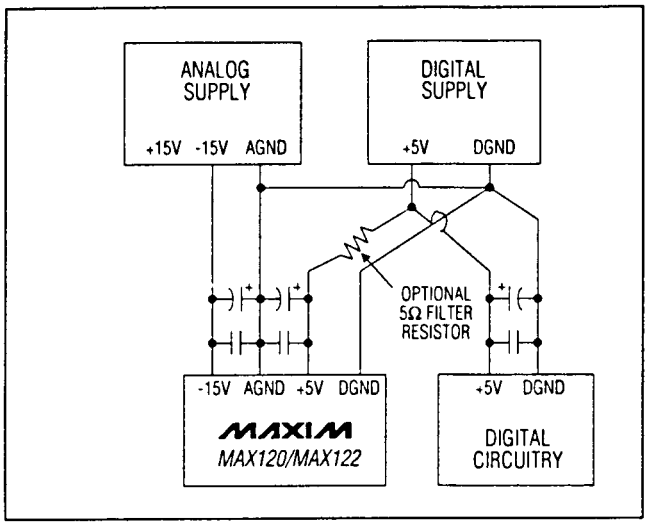


Abb. 2.2.13: Massen-Verbindung (Quelle: [8])

Um keine Störeinflüsse durch die Spannungsversorgung auf den A/D-Wandler wirken zu lassen, sollte die Versorgung mit einem 0.1µF (MKT) und einem 10µF (ELKO) Kondensator abgeblockt werden. Für die Spannungsversorgung des **MAX120** ist eine positive Spannung V_{DD} von 5 V und eine negative V_{SS} zwischen -12 und -15 V notwendig.

Programmierung des GAL 2

Wie bereits erwähnt wurde, kommt zur Steuerung der Zusammenarbeit zwischen A/D-Wandler und dem Mikrokontrollersystem ein programmierter GAL-Baustein zur Anwendung. Dieser beinhaltet zwei Zustandsmaschinen. Die erste dient zum Auslösen des Starts der Konversion und die andere zur Steuerung der Übernahme der gewandelten Daten durch den Controller.

Nachdem im vorangegangenen Abschnitt die Funktionsweise des A/D-Wandlers im FULL-CONTROL-MODE erläutert wurde, kann nun auf den programmierten Ablauf im GAL 2 der A/D-Wandlerkarte eingegangen werden. Aufgrund der möglichst strikten Trennung bei der Aufgabenverteilung auf die GAL's, dient der GAL 2 ausschliesslich der Steuerung der A/D-Wandlung, während der im Abschnitt 2.2.3 beschriebene GAL-Baustein 1 nur zur Adressdecodierung verwendet wird. Dieser Fakt ist bei der Erweiterung der Hardware auf mehr als 2 Kanäle überaus wichtig.

Der erste Zustandsautomat erzeugt das für den A/D-Wandler notwendige Start-Signal CONVST#. Um einen Start auszulösen, muss zuvor die entsprechende Adresse 20000h angesprochen werden und gleichzeitig das Adressgültigkeitssignal AS# des Prozessors anliegen (siehe *Abb. 2.2.14*).

[8]: Maxim, Data Pack Nr. 7; A/D-Converters Design Guide, Seite 11

Um zu Beginn der Zusammenarbeit mit dem A/D-Wandler in einen definierten Zustand zu gelangen, muss der Zustandsautomat durch das RESET# des Prozessors generiert werden. Da aber die Syntax des ORCAD-Compiler dafür nur High-aktive Signale zulässt, musste der Prozessor-Reset durch den GAL negiert werden.

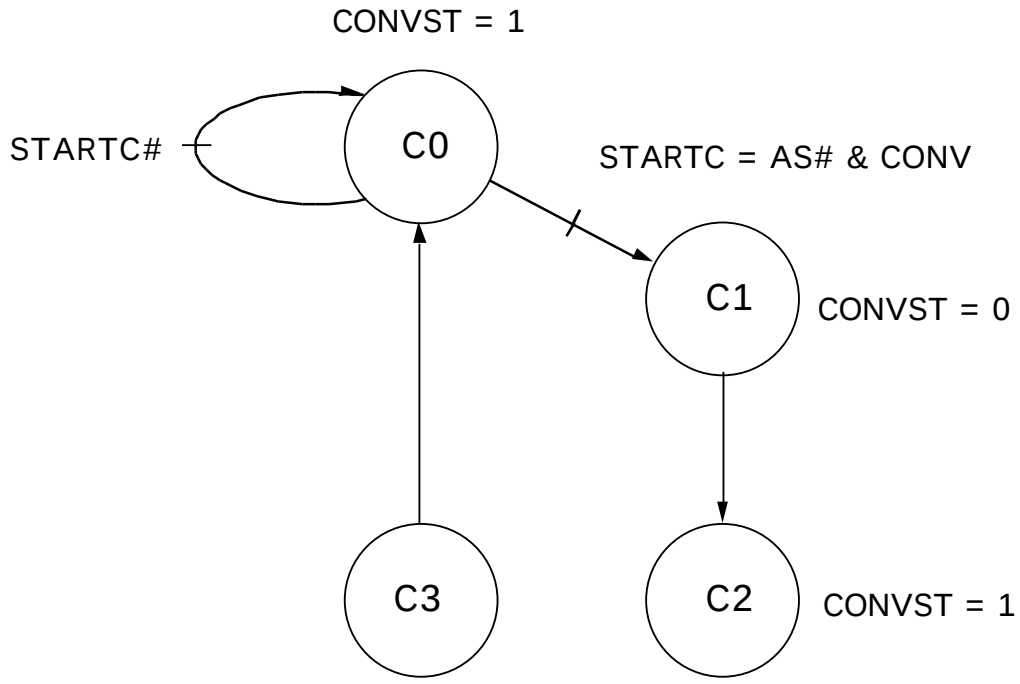


Abb. 2.2.14: Zustandgraph des Automaten CONVST

Wie aus dem Zustandsgraph in *Abb. 2.2.14* ersichtlich ist, gibt es keine Übergangsbedingung, die in den Zustand C3 führen kann, da er nicht benötigt wird. Um die Sicherheit des Systems zu erhöhen, sollte jedoch ein Übergang zu Zustand C0 vorgesehen werden.

Innerhalb der zweiten Zustandsmaschine READ wird mit Hilfe des INT#-Ausgangs des **MAX120** das Signal DSACK1# generiert. Es dient dem Mikrocontroller als Hinweis, dass sich auf dem Datenbus gültige Signale befinden und sie in den Speicher des **68332** übernommen werden können. Dabei ist allerdings zu beachten, dass dieses Signal in den Tri-State geschalten werden muss, da es am BUS der Kontrollerkarte als Ausgang anliegt. Im vorliegenden Fall wird DSACK1# in den Zuständen R0 und R1 in den Tri-State geschalten. Der ebenfalls in dem Automaten READ generierte Ausgang CS_RD# dient als Anforderungssignal für den A/D-Wandler. Im Zustand R0 wird das Signal CS_RD# auf High gesetzt und mit erfolgter STARTR-Bedingung in den Zustand R1 gewechselt. Dort wird auf die Aktivierung des Eingangs INT# gewartet, um danach in den Zustand R2 überzugehen.

Im Zustand R2 wird der CS_RD-Ausgang auf Low-Pegel und das DSACK1#-Signal auf High-Pegel geschaltet und damit die Ausgabe der Daten vom A/D-Wandler auf den Datenbus gestartet. Sobald der Eingang INT# vom **MAX 120** wieder desaktiviert wurde, wechselt die Zustandsmaschine in den Zustand R3. Hier wird nun das Daten-Übernahmesignal DSACK1# für den Prozessor auf Low-Pegel gesetzt. Der gesamte Ablauf der Zustandsmaschine ist in *Abb. 2.2.15* verdeutlicht.

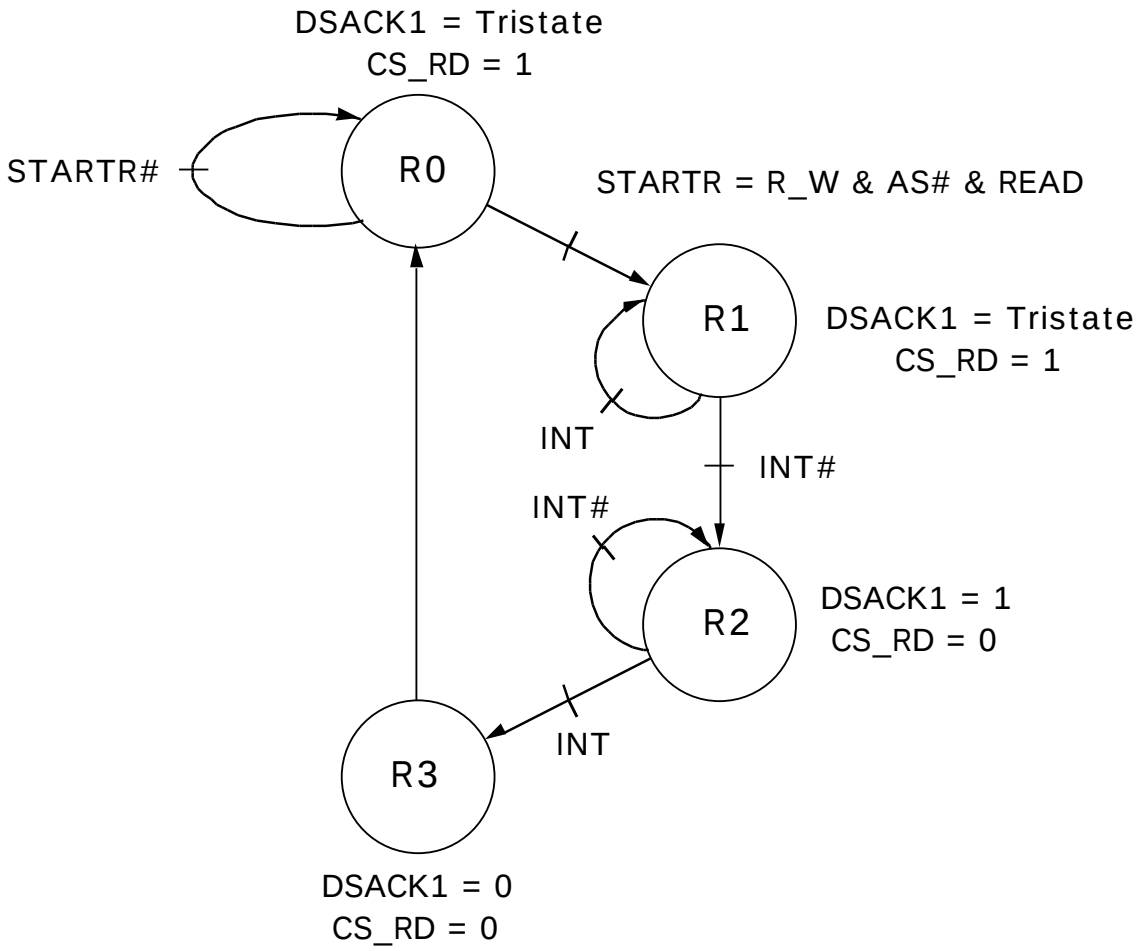


Abb. 2.2.15: Zustandgraph des Automaten READ

In *Abb. 2.2.16* ist mit Hilfe eines KO-Plots der Signalverlauf analysiert worden.

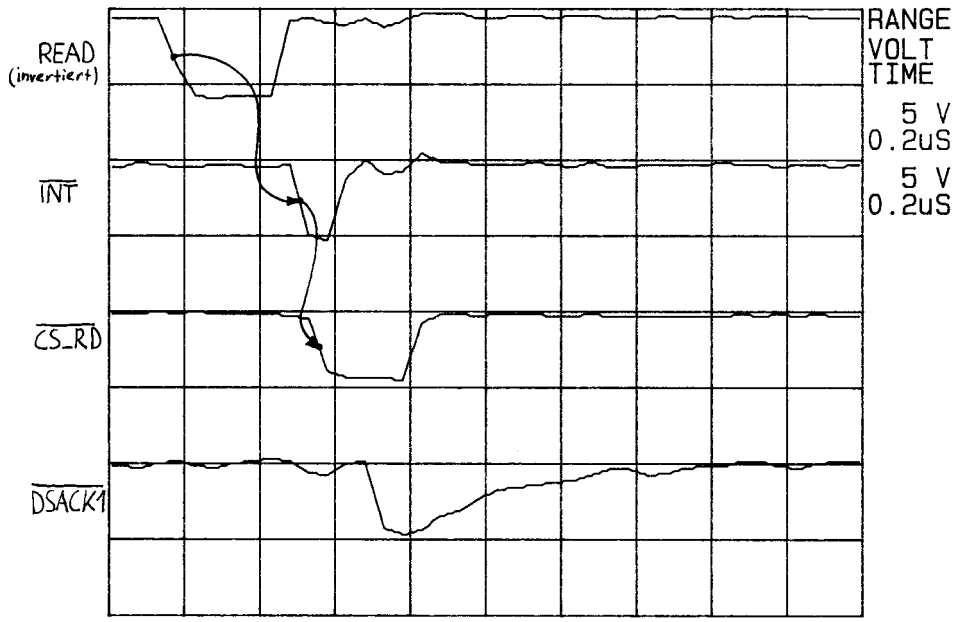


Abb. 2.2.16: KO-Plot Timing

Achtung: READ ist kein LOW-Aktives Signal. Durch eine fehlerhafte KO-Einstellung wurde dieses Signal invertiert dargestellt.

Bei der Arbeit mit Zustandsmaschinen in Mikrokontrollersystemen ist unbedingt zu beachten, dass eine Synchronisierung der entsprechenden Eingänge im GAL erfolgt. In der erwähnten Projektarbeit kam es auf Grund der nicht synchronisierten Eingänge zu Fehlverhalten der Zustandsmaschinen. Bei den Tests der vorliegenden Hardware traten durch die synchrone Arbeitsweise in den Zustandsautomaten keinerlei Probleme mehr auf. Alle Wechsel zwischen den einzelnen Zuständen werden durch die Low-High-Flanke des durch GAL 1 halbierten Prozessortakts gestartet. Die im GAL 2 gebildeten Zustandsmaschinen bilden das in *Abb. 2.2.17* gezeigte Blockschaltbild.

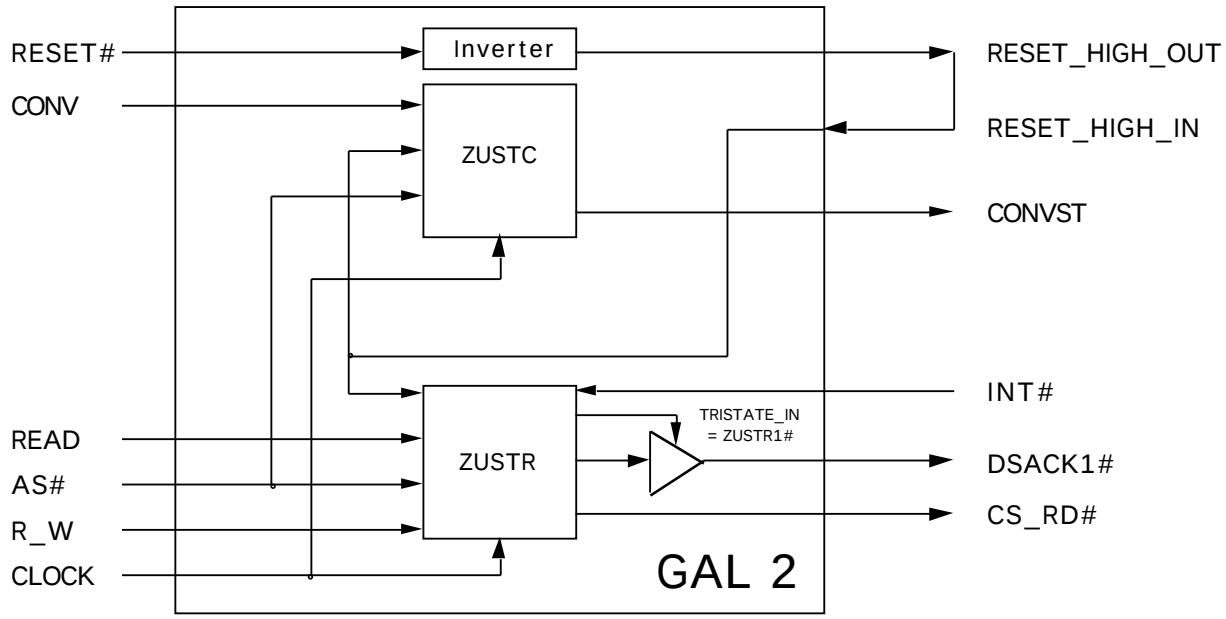


Abb. 2.2.17: Blockschaltbild GAL 2

Daraus lässt sich eine Source-Datei DIPGAL2.PLD erzeugen und mittels Editor für die Verarbeitung im ORCAD-PLD-System vorbereiten. Nach erfolgreicher Übersetzung durch ORCAD wurde vom Compiler eine Datei DIPGAL2.LST *(Abb. 2.2.18)* erzeugt, die der Information des Anwenders dient. Darin sind die einzelnen im vorangegangenen Text erläuterten Abläufe nachvollziehbar.

```
OrCAD PLD COMPILER  V4.00 6/19/92  (Source file DIPGAL2.PLD)

  1  |GAL22V10 14:TRISTATE_IN,
  2  |         in:(CONV,READ,R_W,AS,RESET,RESET_HI_IN),
  3  |         io:(CS_RD,CONVST,DSACK1,INT,RESET_HI_OUT,ZUSTC[1..0],ZUSTR[1..0]),
  4  |         clock:CLOCK
  5  |
  6  |
  7  |TITLE: " STEUERUNG DER A/D-WANDLER "
  8  |       " dipgal2.PLD    Version:3  "
  9  |       " rogoli,  26.11.93          "
 10  |
 11  |STARTC = (AS' & CONV)
 12  |STARTR = (R_W  & AS' & READ)
 13  |RESET_HI_OUT = RESET'
 14  |
 15  |
 16  |CONDITIONING:  CLOCK // ZUSTC[1..0]
 17  |CONDITIONING:  CLOCK // ZUSTR[1..0]
 18  |CONDITIONING:  TRISTATE_IN ?? DSACK1
```

```
19  |
20  |REGISTERS: INT
21  |
22  |PROCEDURE: RESET_HI_IN,ZUSTC[1..0]
23  |{
24  |  STATEC0. CONVST = 1
25  |           STARTC ? ->STATEC1
26  |                    ->STATEC0
27  |
28  |  STATEC1. CONVST = 0
29  |                    ->STATEC2
30  |
31  |  STATEC2. CONVST = 1
32  |                    ->STATEC0
33  |
34  |  STATEC3.          ->STATEC0
35  |}
36  |
37  |PROCEDURE: RESET_HI_IN,ZUSTR[1..0]
38  |{
39  |  STATER0. DSACK1 = 1
40  |           CS_RD  = 1
41  |           STARTR ? ->STATER1
42  |                    ->STATER0
43  |
44  |  STATER1. DSACK1 = 1
45  |           CS_RD  = 1
46  |           INT' ?   ->STATER2
47  |                    ->STATER1
48  |
49  |  STATER2. DSACK1 = 1
50  |           CS_RD  = 0
51  |           INT ?    ->STATER3
52  |                    ->STATER2
53  |
54  |  STATER3. DSACK1 = 0
55  |           CS_RD  = 0
56  |                    ->STATER0
57  |}
58  |
59  | Vectors:
60  |{
61  |  Display "clk:",CLOCK,"RESETHIOUT:",RESET_HI_OUT,"CONV:",CONV,"READ:",READ,\
62  |  "R_W:",R_W,"AS:",AS,"CS_RD:",CS_RD,"CONVST:",CONVST,"DSACK1:",DSACK1,"INT:",INT,\
63  |  "ZUSTC:",ZUSTC[1..0],"ZUSTR:",ZUSTR[1..0]
64  |
65  |  TEST RESET=0;RESET_HI_IN=1;CLOCK=1(0,1)
66  |  TEST RESET=1;RESET_HI_IN=0;CLOCK=1(0,1)
67  |  TEST CONV=0;READ=0;R_W=0;AS=1;INT=1;TRISTATE_IN=0;CLOCK=1(0,1)
68  |  TEST CONV=0;READ=0;R_W=0;AS=0;INT=1;CLOCK=1(0,1)
69  |  TEST CONV=1;READ=0;R_W=0;AS=1;INT=1;CLOCK=1(0,1)
70  |    TEST CONV=1;READ=0;R_W=0;AS=0;CLOCK=1(0,1)
71  |    TEST CLOCK=1(0,1)
72  |    TEST CLOCK=1(0,1)
73  |  TEST CONV=0;READ=0;R_W=0;AS=1;CLOCK=1(0,1)
74  |  TEST CONV=0;READ=0;R_W=1;AS=1;CLOCK=1(0,1)
75  |  TEST CONV=0;READ=0;R_W=1;AS=0;CLOCK=1(0,1)
76  |  TEST CONV=0;READ=1;R_W=1;AS=1;CLOCK=1(0,1)
77  |    TEST CONV=0;READ=1;R_W=1;AS=0;CLOCK=1(0,1)
78  |  TEST INT=1;CLOCK=1(0,1)
79  |    TEST INT=0;TRISTATE_IN=1;CLOCK=1(0,1)
80  |  TEST INT=0;CLOCK=1(0,1)
81  |    TEST INT=1;CLOCK=1(0,1)
82  |    TEST CLOCK=1(0,1)
83  |  END
84  |}
```

```
STATE TABLE FOR ZUSTC

                                State Number
State Label                 ----------------------
                            Decimal Binary   Level

STATEC0                        0      00      LL
STATEC1                        1      01      LH
STATEC2                        2      10      HL
STATEC3                        3      11      HH


STATE TABLE FOR ZUSTR

                                State Number
State Label                 ----------------------
                            Decimal Binary   Level

STATER0                        0      00      LL
STATER1                        1      01      LH
STATER2                        2      10      HL
STATER3                        3      11      HH


RESOLVED EXPRESSIONS (Reduction 2)

Signal name      Row   Terms

RESET_HI_OUT      50   RESET'

CONVST            11   RESET_HI_IN' ZUSTC0'

ZUSTC0            84   CONV  AS' RESET_HI_IN' ZUSTC1' ZUSTC0'

ZUSTC1            67   RESET_HI_IN' ZUSTC1' ZUSTC0

DSACK1            21   TRISTATE_IN
                  22   RESET_HI_IN' ZUSTR1'
                  23   RESET_HI_IN' ZUSTR0'

CS_RD              2   RESET_HI_IN' ZUSTR1'

ZUSTR0           112   READ  R_W  AS' RESET_HI_IN' ZUSTR1' ZUSTR0'
                 113   RESET_HI_IN' INT  ZUSTR1' ZUSTR0
                 114   RESET_HI_IN' INT  ZUSTR1  ZUSTR0'

ZUSTR1            99   RESET_HI_IN' INT' ZUSTR1' ZUSTR0
                 100   RESET_HI_IN' ZUSTR1  ZUSTR0'
```

```
SIGNAL ASSIGNMENT
                                        Rows
 Pin    Signal name   Column      --------------    Activity
                                 Beg Avail Used

  1.     CLOCK           0        -    -    -        High    (Clock)
  2.     CONV            4        -    -    -        High
  3.     READ            8        -    -    -        High
  4.     R_W            12        -    -    -        High
  5.     AS             16        -    -    -        High
  6.     RESET          20        -    -    -        High
  7.     RESET_HI_IN    24        -    -    -        High
  8.     -              28        -    -    -
  9.     -              32        -    -    -
 10.     -              36        -    -    -
 11.     -              40        -    -    -
 13.     -              42        -    -    -
 14.     TRISTATE_IN    38      122    9    0        High    (Registered)
 15.     ZUSTR0         35      111   11    3        High    (Registered)
 16.     ZUSTR1         31       98   13    2        High    (Registered)
 17.     ZUSTC0         27       83   15    1        High    (Registered)
 18.     ZUSTC1         23       66   17    1        High    (Registered)
 19.     RESET_HI_OUT   18       49   17    1        High    (Three-state)
 20.     INT            14       34   15    0        High    (Registered)
 21.     DSACK1         10       21   13    3        High    (Three-state)
 22.     CONVST          6       10   11    1        High    (Three-state)
 23.     CS_RD           2        1    9    1        High    (Three-state)
 25.     -               -        0    1    0
 26.     -               -      131    1    0
                                  ---- ----
                                  132   13  (10%)


I200  No fatal errors found in source code.
I201  No warnings.



OrCAD PLD
Type:       PAL22V10
Title:      STEUERUNG DER A/D-WANDLER
            dipgal2.PLD    Version:3
            rogoli,  26.11.93
*
QP24* QF5828* QV1024*
F0*
L0044 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L0088 11 11 11 11 11 11 11 11 11 11 11 11 10 11 11 01 11 11 11 11 11 11 11 11 *
L0440 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L0484 11 11 11 11 11 11 11 11 11 11 11 11 10 01 11 11 11 11 11 11 11 11 11 11 *
L0924 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 01 11 11 *
L0968 11 11 11 11 11 11 11 11 11 11 11 11 10 11 11 01 11 11 11 11 11 11 11 11 *
L1012 11 11 11 11 11 11 11 11 11 11 11 11 10 11 11 11 11 01 11 11 11 11 11 11 *
L2156 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L2200 11 11 11 11 11 11 11 11 11 11 10 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L2904 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L2948 11 11 11 11 11 11 11 11 11 11 11 01 10 10 11 11 11 11 11 11 11 11 11 11 *
L3652 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L3696 11 11 01 11 11 11 11 11 10 11 11 01 10 01 11 11 11 11 11 11 11 11 11 11 *
L4312 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L4356 11 11 11 11 11 11 11 10 11 11 11 11 10 11 11 01 11 11 10 11 11 11 11 11 *
L4400 11 11 11 11 11 11 11 11 11 11 11 11 10 11 11 10 11 01 11 11 11 11 11 11 *
L4884 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 *
L4928 11 11 11 11 01 11 01 11 10 11 11 11 10 11 11 01 11 01 11 11 11 11 11 11 *
L4972 11 11 11 11 11 11 11 11 01 11 11 11 11 10 11 11 01 11 10 11 11 11 11 11 *
L5016 11 11 11 11 11 11 11 11 01 11 11 11 11 10 11 11 10 11 01 11 11 11 11 11 *
L5808 11 11 11 11 11 10 10 10 10 11 *
C6998*

I202  11/26/93  3:51 pm  (Friday)
I203  Memory utilization 2748/21767 (13%)
I204  Elapsed time 6 seconds
```

Abb. 2.2.18: Datei DIPGAL2.LST

Es sei hier nochmals auf die Möglichkeit verwiesen, dass bestimmte Pinbelegungen von vornherein festgelegt werden können. Im vorliegenden Fall erwies es sich als vorteilhaft, den Ausgang ZUSTR1 der Zustandsmaschine READ so nah wie möglich neben den Eingang TRISTATE_IN zu legen. Dieser Eingang dient zur Steuerung der Tri-State-Ausgangsstufe des Signals DSACK1# und wird deshalb wie bereits erwähnt, mit dem Ausgang ZUSTR1 verbunden.

Daraus ergibt sich die Erkenntnis, dass bei der Programmierung der GAL's bereits klare Vorstellungen über die Gestaltung des später zu schaffenden Prints von grossem Nutzen seien können.

Um den ordnungsgemässen Ablauf der zwei Zustandsautomaten zu simulieren, wurden auch hier im Source-File bereits die erwähnten Testvektoren programmiert. Nach erfolgter Compilierung konnte mit dem ORCAD-Simulator die Funktionsweise geprüft werden. Die entstandene Datei DIPGAL2.LOG ist in *Abb. 2.2.19* aufgelistet. Daran lässt sich das Verhalten der einzelnen Signale sehr gut nachvollziehen. Hier nur ein Beispiel dazu: Wie bereits erwähnt, muss das Signal DSACK1# in den Zuständen R0 und R1 als Tri-State-Ausgang geschalten werden. In der Simulationsdatei wird dies durch die Angabe "Z" dargestellt. Nach Wechsel in Zustand R2 besitzt der Ausgang den gewünschten High-Pegel (siehe auch KO-Plot in *Abb. 2.2.16*).

```
OrCAD TEST VECTOR GENERATOR  V4.00 6/19/92

|{
|  Display "clk:",CLOCK,"RESETHIOUT:",RESET_HI_OUT,"CONV:",CONV,"READ:",READ,\
|  "R_W:",R_W,"AS:",AS,"CS_RD:",CS_RD,"CONVST:",CONVST,"DSACK1:",DSACK1,"INT:",INT,\
|  "ZUSTC:",ZUSTC[1..0],"ZUSTR:",ZUSTR[1..0]

|
|  TEST RESET=0;RESET_HI_IN=1;CLOCK=1(0,1)
clk:0RESETHIOUT:1CONV:0READ:0R_W:0AS:0CS_RD:0CONVST:0DSACK1:ZINT:0ZUSTC:UUZUSTR:UU
clk:1RESETHIOUT:1CONV:0READ:0R_W:0AS:0CS_RD:0CONVST:0DSACK1:ZINT:0ZUSTC:00ZUSTR:00

|  TEST RESET=1;RESET_HI_IN=0;CLOCK=1(0,1)
clk:0RESETHIOUT:0CONV:0READ:0R_W:0AS:0CS_RD:1CONVST:1DSACK1:ZINT:0ZUSTC:00ZUSTR:00
clk:1RESETHIOUT:0CONV:0READ:0R_W:0AS:0CS_RD:1CONVST:1DSACK1:ZINT:0ZUSTC:00ZUSTR:00

|  TEST CONV=0;READ=0;R_W=0;AS=1;INT=1;TRISTATE_IN=0;CLOCK=1(0,1)
clk:0RESETHIOUT:0CONV:0READ:0R_W:0AS:1CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:00
clk:1RESETHIOUT:0CONV:0READ:0R_W:0AS:1CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:00

|  TEST CONV=0;READ=0;R_W=0;AS=0;INT=1;CLOCK=1(0,1)
clk:0RESETHIOUT:0CONV:0READ:0R_W:0AS:0CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:00
clk:1RESETHIOUT:0CONV:0READ:0R_W:0AS:0CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:00

|  TEST CONV=1;READ=0;R_W=0;AS=1;INT=1;CLOCK=1(0,1)
clk:0RESETHIOUT:0CONV:1READ:0R_W:0AS:1CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:00
clk:1RESETHIOUT:0CONV:1READ:0R_W:0AS:1CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:00

|    TEST CONV=1;READ=0;R_W=0;AS=0;CLOCK=1(0,1)
clk:0RESETHIOUT:0CONV:1READ:0R_W:0AS:0CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:00
clk:1RESETHIOUT:0CONV:1READ:0R_W:0AS:0CS_RD:1CONVST:0DSACK1:ZINT:1ZUSTC:01ZUSTR:00

|    TEST CLOCK=1(0,1)
clk:0RESETHIOUT:0CONV:1READ:0R_W:0AS:0CS_RD:1CONVST:0DSACK1:ZINT:1ZUSTC:01ZUSTR:00
clk:1RESETHIOUT:0CONV:1READ:0R_W:0AS:0CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:10ZUSTR:00

|    TEST CLOCK=1(0,1)
clk:0RESETHIOUT:0CONV:1READ:0R_W:0AS:0CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:10ZUSTR:00
clk:1RESETHIOUT:0CONV:1READ:0R_W:0AS:0CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:00

|  TEST CONV=0;READ=0;R_W=0;AS=1;CLOCK=1(0,1)
clk:0RESETHIOUT:0CONV:0READ:0R_W:0AS:1CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:00
clk:1RESETHIOUT:0CONV:0READ:0R_W:0AS:1CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:00
```

```
|  TEST CONV=0;READ=0;R_W=1;AS=1;CLOCK=1(0,1)
clk:0RESETHIOUT:0CONV:0READ:0R_W:1AS:1CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:00
clk:1RESETHIOUT:0CONV:0READ:0R_W:1AS:1CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:00

|  TEST CONV=0;READ=0;R_W=1;AS=0;CLOCK=1(0,1)
clk:0RESETHIOUT:0CONV:0READ:0R_W:1AS:0CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:00
clk:1RESETHIOUT:0CONV:0READ:0R_W:1AS:0CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:00

|  TEST CONV=0;READ=1;R_W=1;AS=1;CLOCK=1(0,1)
clk:0RESETHIOUT:0CONV:0READ:1R_W:1AS:1CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:00
clk:1RESETHIOUT:0CONV:0READ:1R_W:1AS:1CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:00

|    TEST CONV=0;READ=1;R_W=1;AS=0;CLOCK=1(0,1)
clk:0RESETHIOUT:0CONV:0READ:1R_W:1AS:0CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:00
clk:1RESETHIOUT:0CONV:0READ:1R_W:1AS:0CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:01

|  TEST INT=1;CLOCK=1(0,1)
clk:0RESETHIOUT:0CONV:0READ:1R_W:1AS:0CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:01
clk:1RESETHIOUT:0CONV:0READ:1R_W:1AS:0CS_RD:1CONVST:1DSACK1:ZINT:1ZUSTC:00ZUSTR:01

|    TEST INT=0;TRISTATE_IN=1;CLOCK=1(0,1)
clk:0RESETHIOUT:0CONV:0READ:1R_W:1AS:0CS_RD:1CONVST:1DSACK1:1INT:0ZUSTC:00ZUSTR:01
clk:1RESETHIOUT:0CONV:0READ:1R_W:1AS:0CS_RD:0CONVST:1DSACK1:1INT:0ZUSTC:00ZUSTR:10

|  TEST INT=0;CLOCK=1(0,1)
clk:0RESETHIOUT:0CONV:0READ:1R_W:1AS:0CS_RD:0CONVST:1DSACK1:1INT:0ZUSTC:00ZUSTR:10
clk:1RESETHIOUT:0CONV:0READ:1R_W:1AS:0CS_RD:0CONVST:1DSACK1:1INT:0ZUSTC:00ZUSTR:10

|    TEST INT=1;CLOCK=1(0,1)
clk:0RESETHIOUT:0CONV:0READ:1R_W:1AS:0CS_RD:0CONVST:1DSACK1:1INT:1ZUSTC:00ZUSTR:10
clk:1RESETHIOUT:0CONV:0READ:1R_W:1AS:0CS_RD:0CONVST:1DSACK1:0INT:1ZUSTC:00ZUSTR:11

|    TEST CLOCK=1(0,1)
clk:0RESETHIOUT:0CONV:0READ:1R_W:1AS:0CS_RD:0CONVST:1DSACK1:0INT:1ZUSTC:00ZUSTR:11
clk:1RESETHIOUT:0CONV:0READ:1R_W:1AS:0CS_RD:1CONVST:1DSACK1:1INT:1ZUSTC:00ZUSTR:00

|  END
```

Abb. 2.2.19: Datei DIPGAL2.L0G

2.2.5 Antialiasing-Filterung

Der A/D-Converter wandelt, wie bereits in Abschnitt 2.2.4 erläutert, die am Eingang als analoge Werte anliegenden Signale, in für den Kontroller verständliche, digitale Signale um. Er erfüllt aber gleichzeitig die Funktion des Abtasters (Sampler). Dabei kann es vorkommen, dass der Abtaster das ursprüngliche Frequenzspektrum des analogen Signals so verändert, dass eine weitere Verarbeitung sinnlos wird. Dieser Effekt wird als "Aliasing" bezeichnet. Um den abgetasteten Wert eines analogen Signals zu erhalten, wird es mit einer Gewichtsfunktion multipliziert. Dieser Vorgang wird Faltung genannt. Dabei ist darauf zu achten, dass die Frequenz der Gewichtsfunktion kleiner ist, als die halbe Abtastfrequenz (Nyquist-Frequenz). Sobald die Frequenz zu gross gewählt ist, wird das Signal verfälscht, es entsteht der sogenannte Alias-Effekt. Dies lässt sich wie folgt begründen. Da jetzt weniger als zwei Messwerte pro Periode abgetastet werden, entspricht das resultierende gefaltete Signal nicht mehr dem originalen analogen Eingangssignal.

Die einzige Möglichkeit, diesen Effekt zu verhindern besteht darin, einen Anti-Aliasing-Filter vorzuschalten. Der günstigste Fall wäre dabei, wenn der Filter im Frequenzbereich zwischen 0 und der halben Abtastfrequenz eine Verstärkung von 1 hätte und über diesen Bereich hinaus keine Verstärkung aufweisen würde. Dieser ideale Tiefpass lässt sich aber nicht realisieren. Deshalb muss man eine optimale Näherung unter den verschiedenen Filtertypen finden.

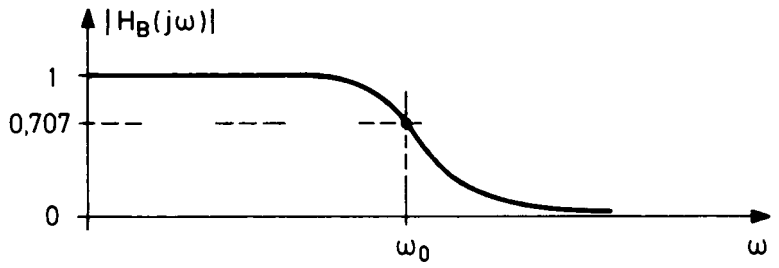


Abb. 2.2.20: Butterworth-Filtercharakteristik

Für unsere Anwendung bietet der Butterworth-Tiefpass eine günstige Charakteristik (siehe *Abb 2.2.20*). Sie werden als optimal flache Filter bezeichnet, da ihr Verstärkungsfaktor sehr nahe bei 1 liegt. Oberhalb der Grenzfrequenz fällt die Verstärkung mit n mal 20dB (n: Filterordnung) pro Dekade ab. Um einen möglichst einfachen und dennoch mit sehr guter Charakteristik ausgestatteten Filterbaustein zu erhalten, wurde der Schaltkreis **MAXIM274** ausgewählt. Für unsere Anwendung wurde die Abtastfrequenz mit 1200Hz festgelegt und durch die TPU des Mikrokontrollers **MC68332** zur Verfügung gestellt.

Filterbaustein **MAX274**

Bei dem Schaltkreis **MAX274** handelt es sich um einen analogen Filter mit 4 Sektionen, wobei eine Sektion jeweils einen Filter 2. Ordnung darstellt. Durch Kaskadierung lässt sich somit mit einem Schaltkreis ein Filter 8.Ordnung aufbauen (siehe Funktionsplan *Abb. 2.2.21*) Aus der oben erwähnten Abtastfrequenz ergibt sich unter Beachtung des Abtasttheorems eine Grenzfrequenz für den Filter von maximal 600Hz.

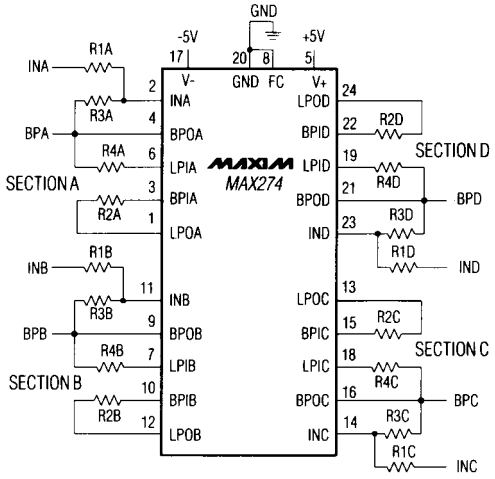


*Abb. 2.2.21: Funktionsplan **MAX274** (Quelle: [14])*

Dieser Baustein benötigt zur Funktion eines Filters nur die Aussenbeschaltung mit einigen Widerständen. Die erforderlichen Kapazitäten sind bereits integriert. Da diese jedoch verständlicherweise sehr klein sind (ca. 80 pF), ergeben sich sehr hohe Widerstandswerte. Zur Berechnung der einzelnen Filterwerte stellt der Hersteller MAXIM freundlicherweise ein Programm FILTER.EXE zur Verfügung. Damit lässt sich die gewünschte Filtercharakteristik mit ihren entsprechenden Werten einstellen (in *Abb. 2.2.22* sind die Vorgaben für das gewünschte Verhalten im A/D-Wandlersystem enthalten). Das Programm berechnet verschiedene Filtertypen mit Ordnung und Frequenzverläufen.

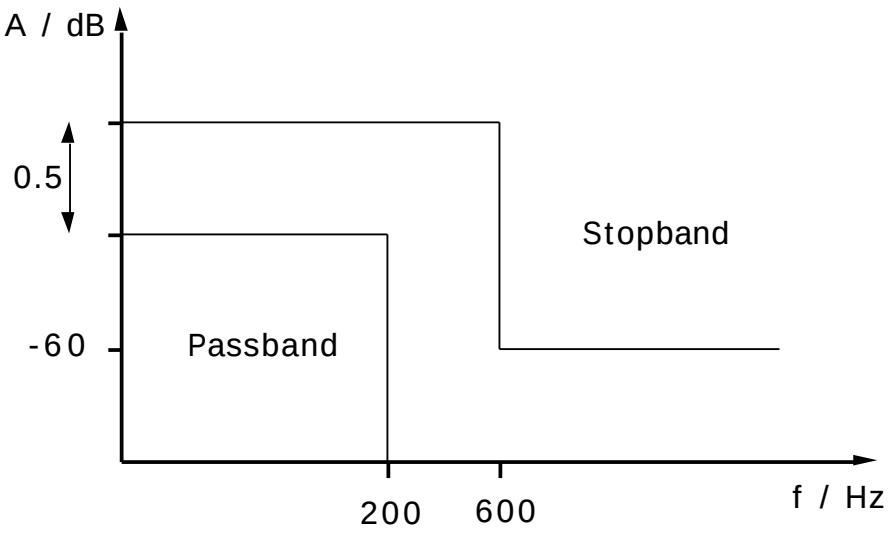


Abb. 2.2.22: Filtercharakteristik des Antialiasingfilters

[14]: Maxim, 4th and 8th-Order Continuous-Time Active Filters, Seite 10

Mit Hilfe der Vorgaben, die für das Antialiasing-Problem gemacht wurden, berechnete das Programm einen Butterworth-Filter 8.Ordnung. Dieser Aufbau ist mit genau einem Chip pro Kanal zu realisieren. Im Gegensatz zu anderen Filtertypen (z.B. Tschebyscheff-Filter) wurde beim Butterworth eine realisierbare Aussenbeschaltung mit Widerständen berechnet. Die MAXIM-Software ermittelte die in *Abb. 2.2.23* aufgelisteten Werte der Filterübertragungsfunktion und in *Abb 2.2.24* ist das Frequenzverhalten des Filters in einer Grafik dargestellt.

```
Maxim Integrated Products Filter Design Software
Version 1.01
*------------------------------------------------------------------------*



FILTER DESCRIPTION:
    TYPE: Butterworth Lowpass
    ORDER: 8


    Corner frequency:  200.000 Hz    * upper passband limit *
    Stop frequency:    600.000 Hz    * lower stopband limit *


    AMAX:  500.000mdB    * maximum attenuation in the passband *
    AMIN:   67.204 dB    * minimum attenuation in the stopband *


    POLE            Q               Zero
    ------------    ------------    ------------
     228.034 Hz     509.796m
     228.034 Hz     601.345m
     228.034 Hz     899.976m
     228.034 Hz       2.563




LOWPASS PROTOTYPE:
    * filter transformed to lowpass with corner frequency = 1 rad/sec *


    POLE            Q               Zero
    ------------    ------------    ------------
       1.140        509.796m
       1.140        601.345m
       1.140        899.976m
       1.140          2.563
```

```
QUADRATIC EQUATIONS FOR FILTER:

                                  (2.05285717E+06)
H1(s) = Holp1 X ------------------------------------------------
                    S^2 + (2.81049814E+03)S + (2.05285717E+06)

                                  (2.05285717E+06)
H2(s) = Holp2 X ------------------------------------------------
                    S^2 + (2.38262527E+03)S + (2.05285717E+06)

                                  (2.05285717E+06)
H3(s) = Holp3 X ------------------------------------------------
                    S^2 + (1.59201931E+03)S + (2.05285717E+06)

                                  (2.05285717E+06)
H4(s) = Holp4 X ------------------------------------------------
                    S^2 + (5.59042838E+02)S + (2.05285717E+06)

Where Holp = lowpass gain at DC



OVERALL FILTER EQUATION:


  H(s) = H1(s) X H2(s) X H3(s) X H4(s)
```

*Abb. 2.2.23: Filterberechnung durch **MAXIM**-Software*

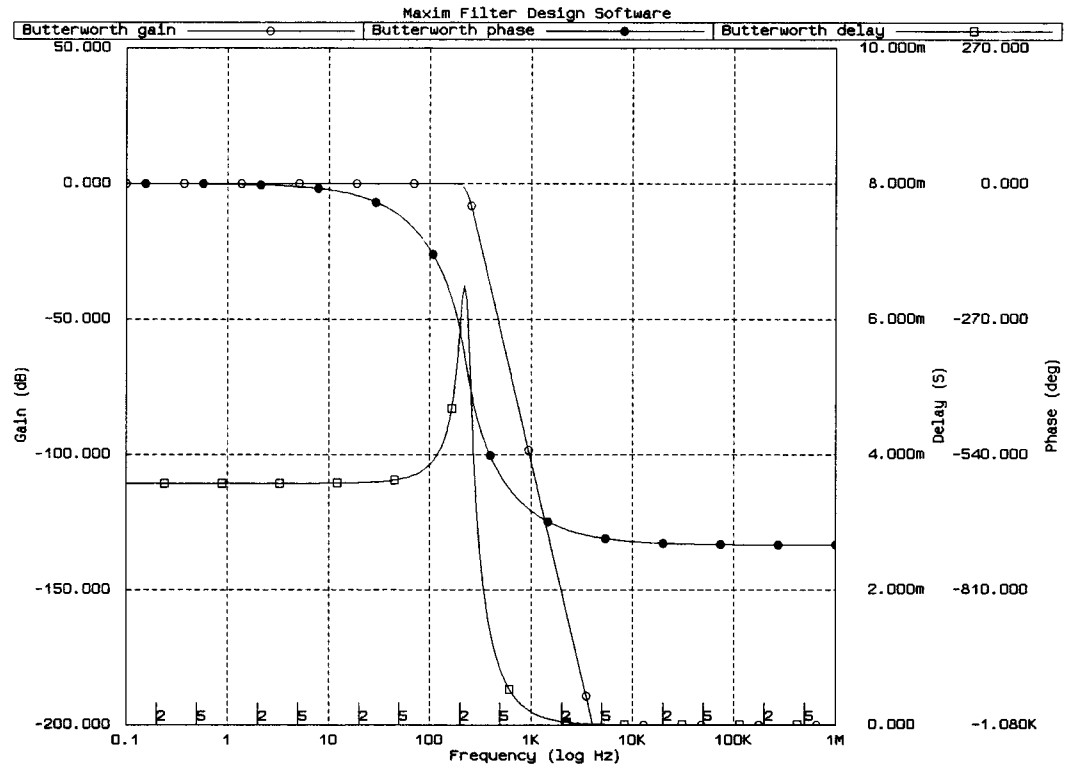
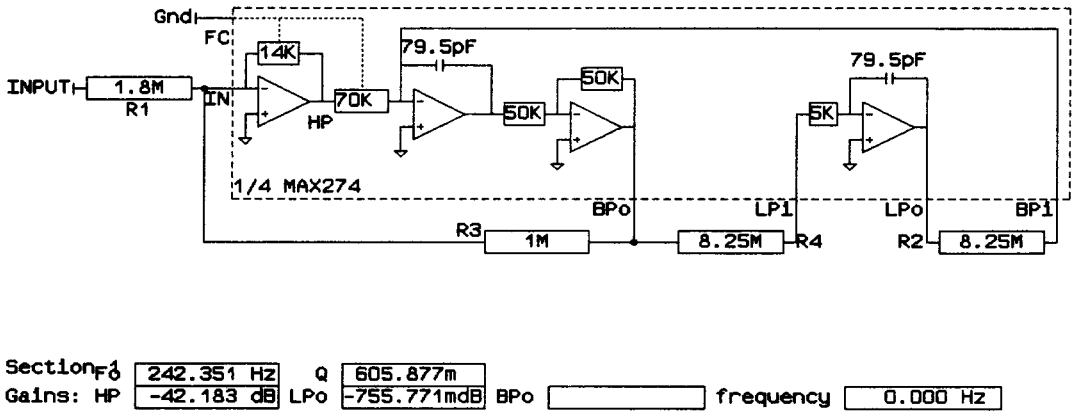


Abb. 2.2.24: Frequenzverhalten des gesamten Filters

Nach Auswahl der gewünschten Filtercharakteristik lassen sich alle zur Aussenbeschaltung vorgesehenen Widerstände berechnen und an entsprechende standartisierte E-Reihen anpassen. Dazu dient der zweite Teil der Software. In der Abb. 2.2.25 sind die vier Filterstufen mit ihren angepassten Widerstände enthalten. Dabei ist im Vergleich zu den vorangemachten Berechnungen zu bemerken, dass sich die Werte z.B. für die Resonanzfrequenz Fo geringfügig ändern. Dies entsteht durch die Anpassung der Widerstandswerte.



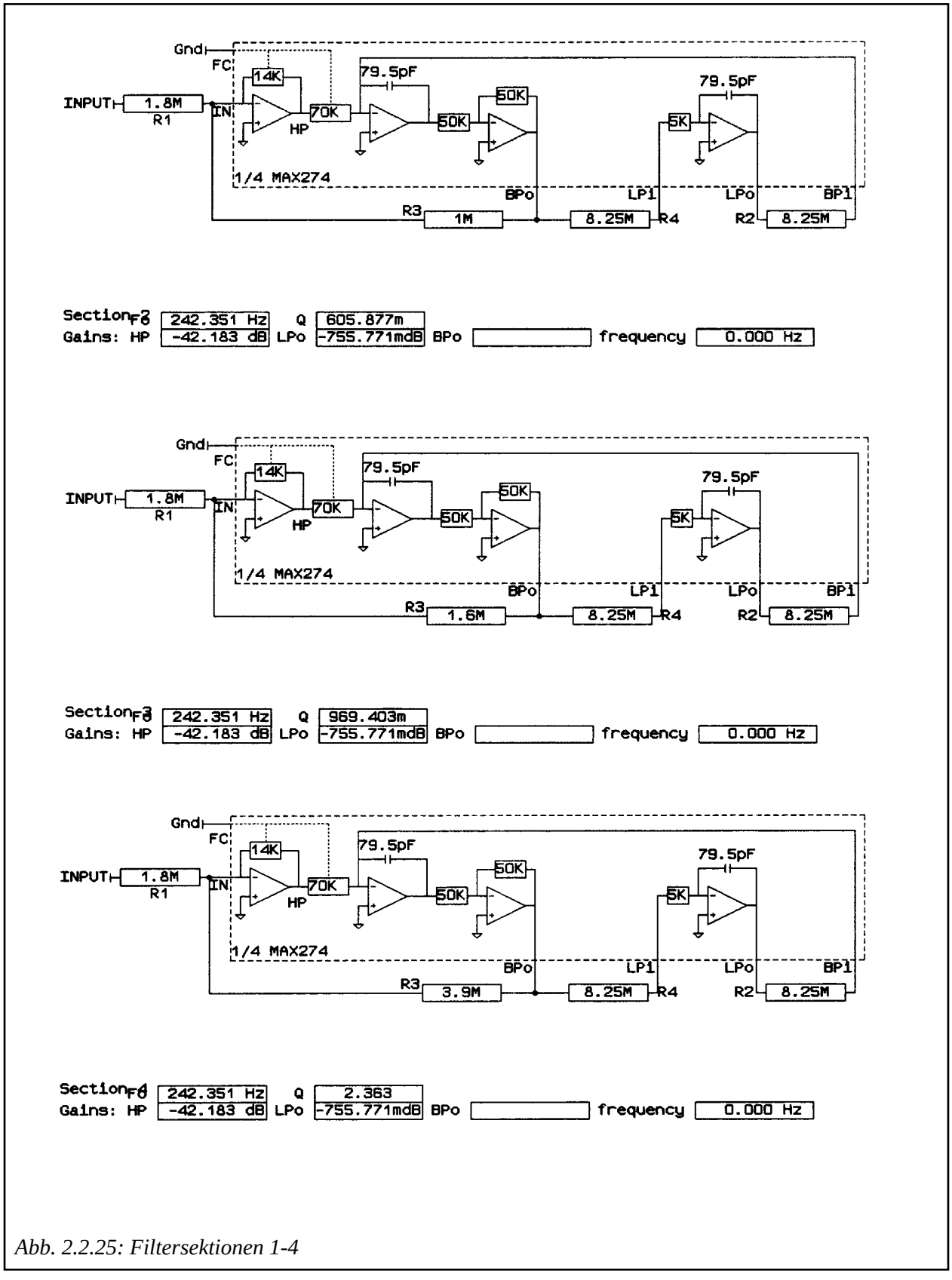


Abb. 2.2.25: Filtersektionen 1-4

Jede Sektion besteht aus drei Eingängen und zwei Ausgängen, wobei bei einem Tiefpassfilter der Pin IN den Sektionseingang und LPo den Ausgang darstellt. Um einen Filter 8.Ordnung zu erreichen, werden die Ausgänge der Sektionen LPoX über den Widerstand R1X mit dem Eingang der nächsten Sektion INX verbunden. Somit ergibt sich für die vorliegende Anwendung als Filtereingang der Widerstand R1A an Pin INA und als Ausgang des Butterworth-Filter der Pin LPoD.

In *Abb. 2.2.26* ist der Betrag des Amplitudengangs und in *Abb. 2.2.27* der Betrag des Phasengangs der einzelenen Sektionen dokumentiert. Dabei ist ersichtlich, dass jede Sektion eine Phasenverschiebung von 180° liefert und somit das Ausgangssignal am Filter in gleicher Phasenlage zum Eingang liegt, jedoch mit einer entsprechenden Verzögerungszeit.

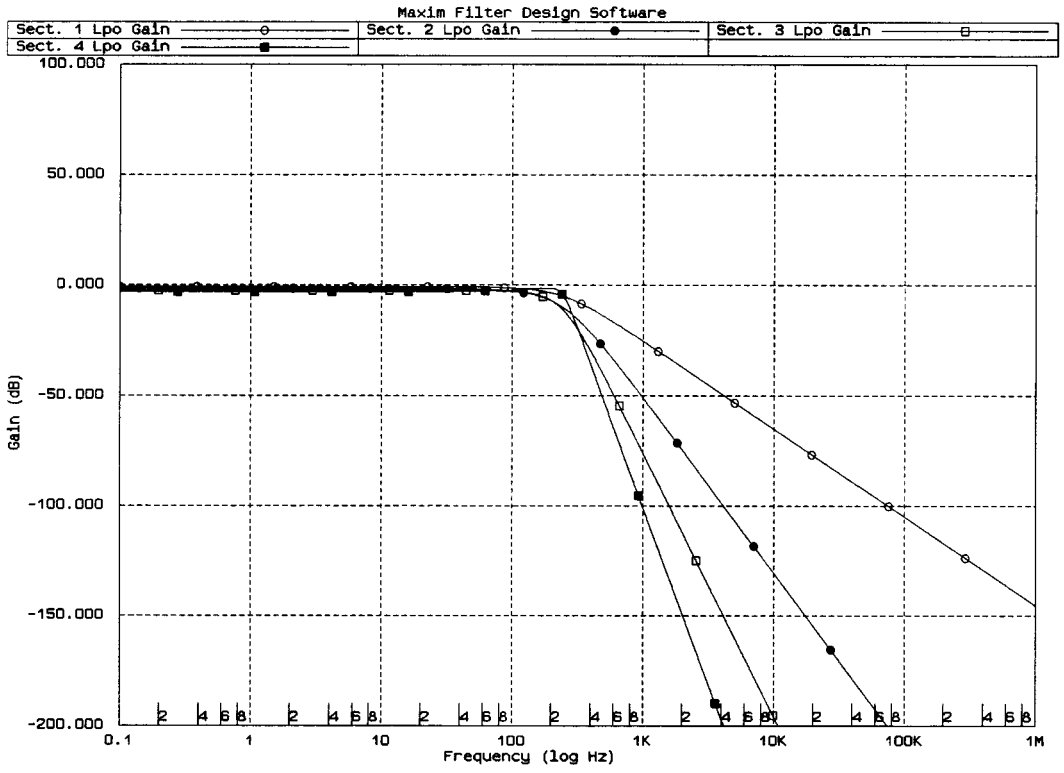


Abb. 2.2.26 Amplitudengang der einzelnen Sektionen

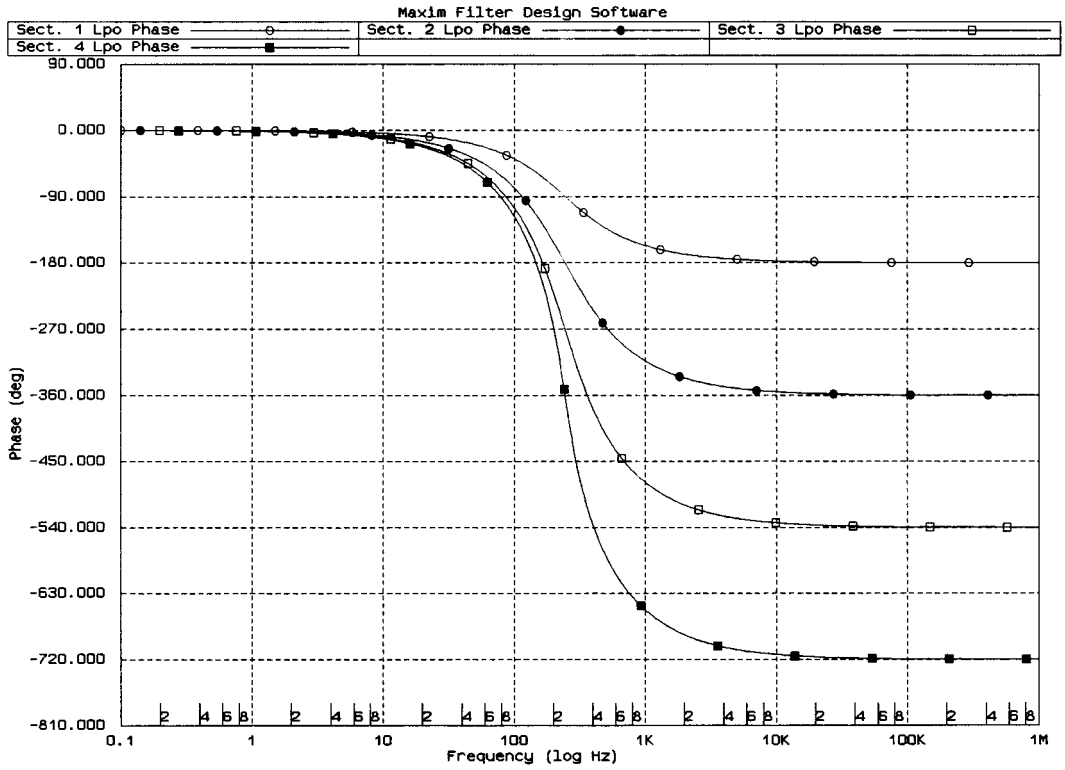


Abb. 2.2.27 Phasengang der einzelnen Sektionen

Zum Lieferumfang der Design-Software gehört auch ein Evaluationkit zum Aufbau des konfigurierten Filters. Nach der Komplettierung der Filterschaltung mit externen Widerständen wurde die Baugruppe einem Test mit dem **Gain-Phase-Analyzer Schlumberger** unterzogen (siehe *Abb. 2.2.28*).

Bei weiteren Messungen zeigte sich ein Problem auf, das in der Beschreibung des Schaltkreises vom Hersteller leider unterschlagen wurde. Statt einer maximalen Offsetspannung am Ausgang von +/-200 mV wurden mit verschiedenen Schaltkreisen Offsetwerte zwischen -0.9V und +2.3V gemessen!
Glücklicherweise wird in Literatur [14] ein Verfahren zur Beseitigung dieser Offsetspannung *(Abb. 2.2.9)* erwähnt.

[14]: MAXIM, 4th and 8th-Order Continuous-Time Active Filters, Seite 20

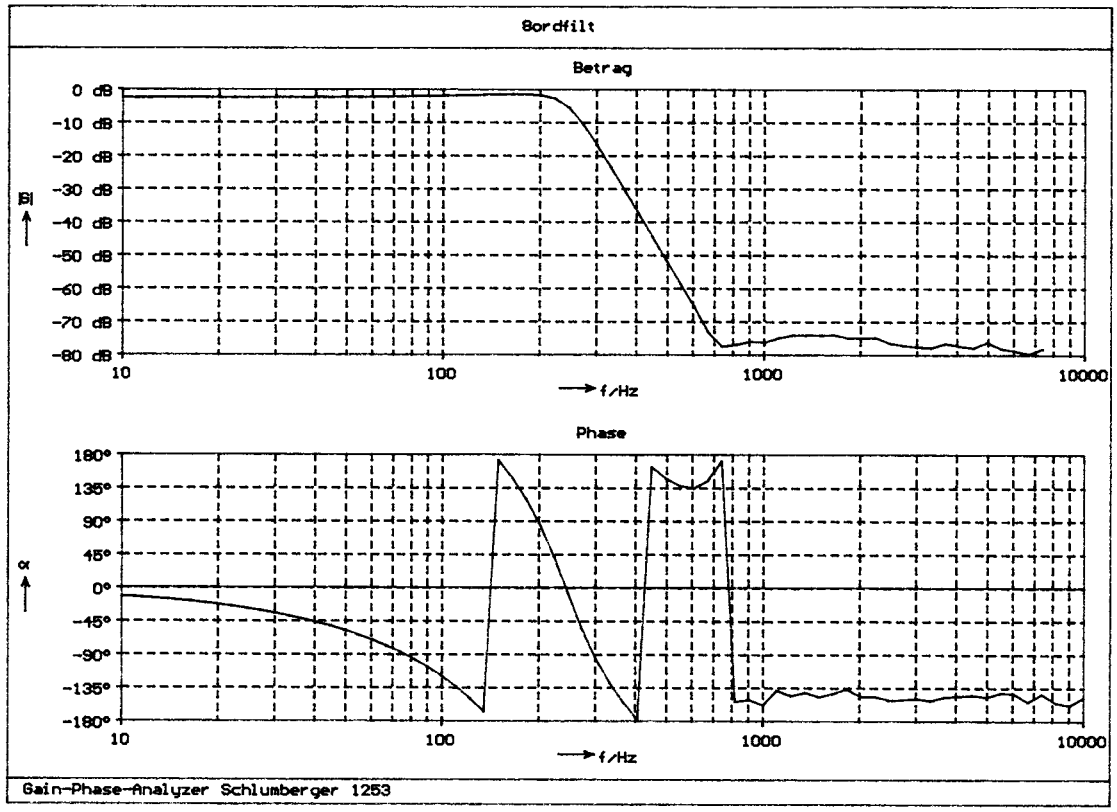


Abb. 2.2.28 Schlumberger-Plot

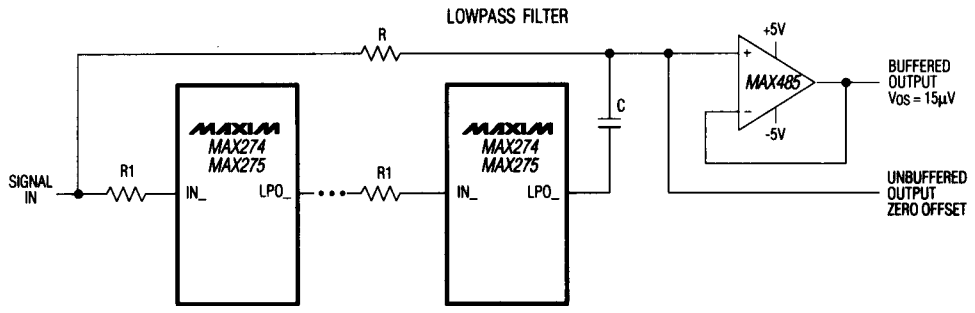


Abb. 2.2.29: DC-Offset-Kompensierung

Für das Signal SIGNAL_IN bedeutet die Zuschaltung des Widerstands R und des Kondensators C ein Tiefpassfilter 1. Ordnung. Für den Filterausgang bedeutet diese Zuschaltung ein Hochpassfilter 1. Ordnung. Die Grenzfreuenz dieses RC-Gliedes soll laut Literatur [14] mindestens 1 bis 2 Dekaden unter derjenigen des aktiven Filters liegen.
Bei tiefen Frequenzen hat SIGNAL_IN den direkten Einfluss auf den Ausgang. Der Filterausgang hingegen ist hochohmig vom OP-Amp am Ausgang getrennt. So hat ein allfälliger DC-Offset am Filterausgang keinerlei Einfluss auf das Ausgangssignal der gesamten Schaltung.
Bei hohen Frequenzen ist der Kondensator C gegenüber dem Widerstand R so hochohmig, dass nur noch das Filter-Ausgangssignal das Ausgangssignal der gesamten Schaltung bestimmt.

Die Grenzfrequenz der RC-Schaltung wird für unsere Anwendung auf 20Hz festgelegt. Daraus ergeben sich die Komponentenwerte R = 8kΩ und C = 10µF.

Da aber die Offsetspannung filterintern immer noch vorhanden ist, muss der Bereich der Eingangsspannung in das Intervall [-2.5V..+2.5V] fallen. Dadurch gerät der aktive Filter nicht in den Übersteuerungsbereich. Um nach dem Filter trotzdem Spannungen im Bereich [-5V..+5V] zu erhalten, wird der OP-Amp der Offset-Kompensierung gerade als nichtinvertierender Verstärker mit Verstärkung 2 verwendet. Somit kann die ganze Auflösung des A/D-Wandlers **MAX120** genutzt werden.

2.2.6 Eingangsverstärker

Da man im Unklaren darüber ist, wie gross der Ausgangswiderstand (welcher bei bestimmten Sensoren oder ähnlichem sehr gross sein kann) der Signalquelle ist, ist es erforderlich, das Eingangssignal über einen Impedanzwandler an den aktiven Filter zu schalten.

Gleichzeitig ist aber auch die Tatsache zu beachten, dass ein aktives Tiefpassfiler nicht unendlich schnell sein kann, und deshalb hochfrequente Störungen verstärken kann.

Darum wird dem Impedanzwandler ein Passiver RC-Tiefpass vorgeschaltet. Für unsere Anwendung wird die Grenzfrequenz dieses Tiefpasses auf etwa 10kHz festgelegt.

Damit nicht der RC-Tiefpass allein als Last für den Signalgeber wirkt, was sich nachteilig auswirken könnte, wird mit einem 100kΩ Widerstand gegen Masse die Eingangsimpedanz des A/D-Wandlerkanals bestimmt. *Abb. 2.2.30* zeigt die Realisierung des Eingangsverstäkers.

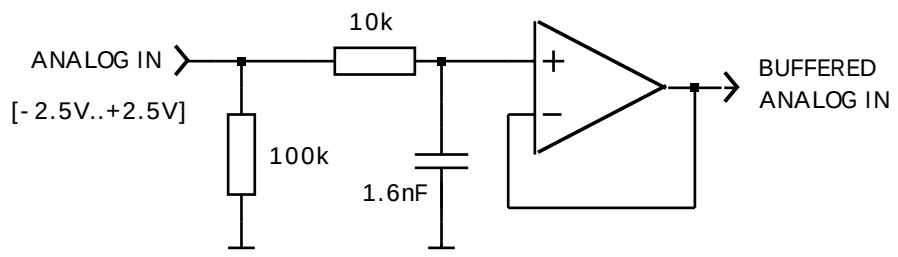


Abb. 2.2.30: Prinzipschema Eingangsverstärker

2.2.7 Technische Spezifikationen, Fertigungsunterlagen

Stromversorgung

Die A/D-Wandlerkarte muss mit den Spannungen +15V, -15V, +5V, -5V versorgt werden. Die Karte wird nicht über den Europastecker, sondern über Klemmleisten mit den Speisespannungen versorgt.

Analoge Eingänge A und B

Die beiden analogen Eingänge A und B sind für Signale mit Amplituden von bis 2.5V und Frequenzen bis 200Hz spezifiziert.
Die analogen Eingangssignale werden über Koaxial-buchsen an die Hardware gebracht.

Kartenstecker

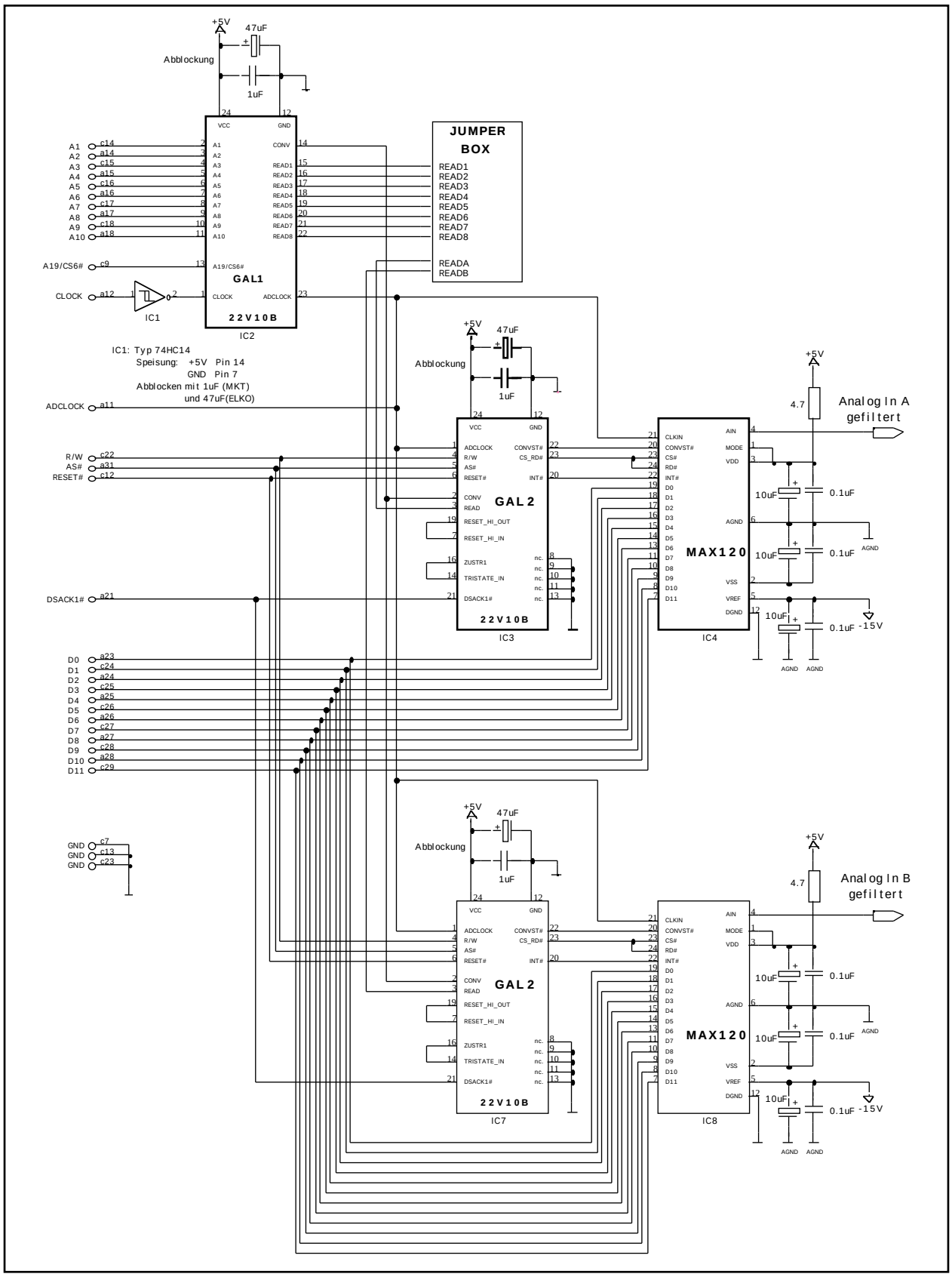
Sämtliche benötigten Bus-Signale des Kontrollers, die internen Bus-Signale (CONV, READ1..8) sowie die digitalen Steuersignale des Stellgliedes (PWM_OUT, Stop out, BridgeReset und ThermFlag) sind am Europa-Kartenstecker abgreifbar. Die Pin-Nummern sind den Gesamtschemata der A/D-Wandlerkarte*(Abb. 2.2.31)* und des Stellgliedes *(Abb. 2.3.19)* zu entnehmen.

Gesamtschema

Das Gesamtschema der A/D-Wandlerkarte ist in *Abb. 2.2.31* abgebildet.

Layout, Bestückungszeichnung

Leider hat die Zeit nicht mehr gereicht, vor Abgabe dieser Arbeit die A/D-Wandlerkarte anzufertigen. Daher sind auch noch keine Unterlagen wie Layout und Bestückungszeichnung vorhanden.


+5V
47uF
Abblockung
1uF
JUMPER BOX
READ1
READ2
READ3
READ4
READ5
READ6
READ7
READ8
READA
READB
A1 A2 A3 A4 A5 A6 A7 A8 A9 A10
A19/CS6#
CLOCK
GAL1
22V10B
IC1
IC2
IC1: Typ 74HC14
Speisung: +5V Pin 14
GND Pin 7
Abblocken mit 1uF (MKT)
und 47uF(ELKO)
ADCLOCK
R/W
AS#
RESET#
DSACK1#
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11
GND GND GND
GAL 2
IC3
IC7
MAX120
IC4
IC8
4.7
Analog In A gefiltert
Analog In B gefiltert
10uF
0.1uF
AGND
-15V

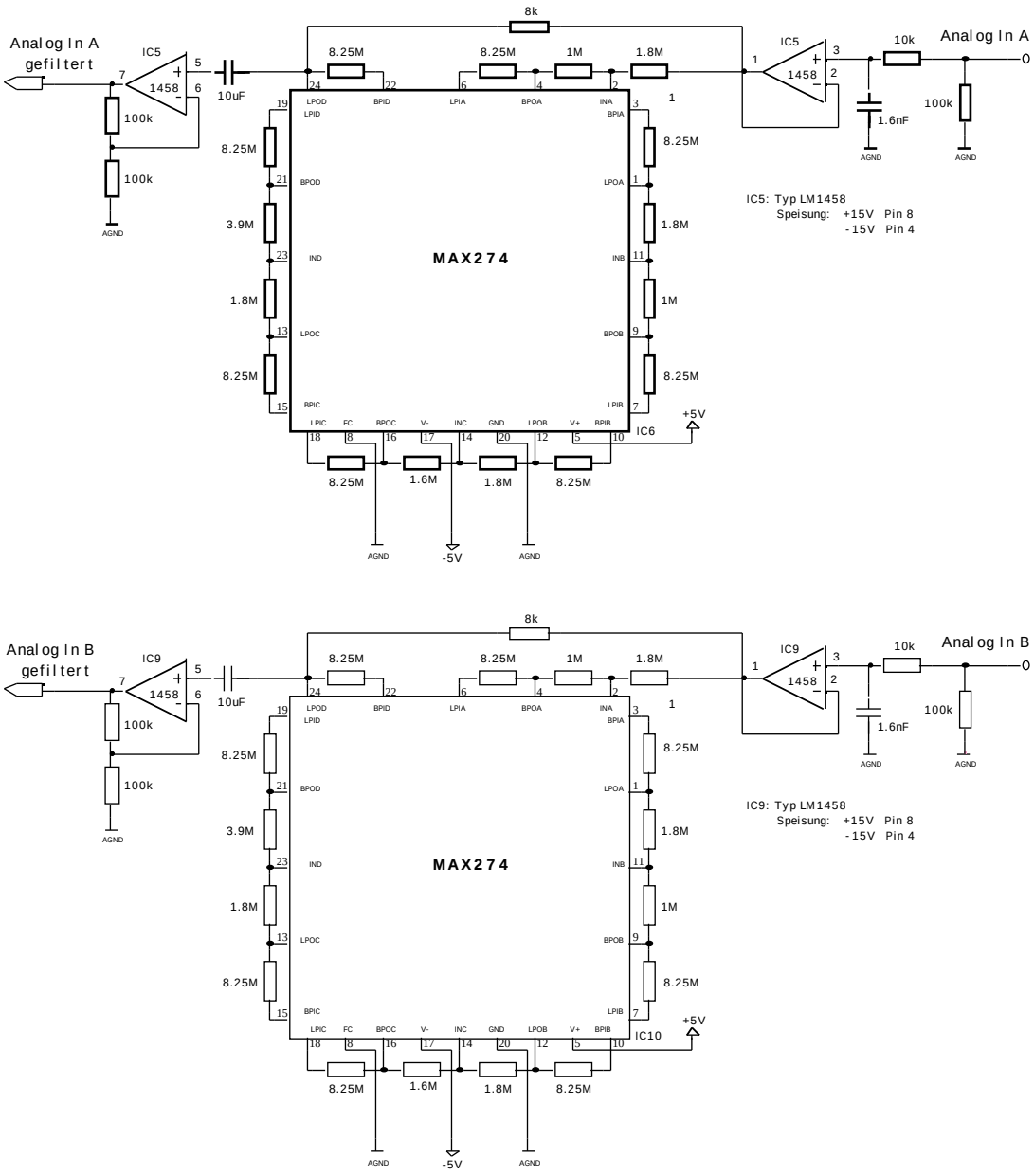


Abb. 2.2.31: Gesamtschema A/D-Wandlerkarte

2.3 Stellglied

Laut Aufgabenstellung soll es sich beim Stellglied um eine geschaltete Leistungsbrücke handeln. Es sei insbesondere auf Betriebssicherheit bezüglich Strombegrenzung, Überspannungsfestigkeit und Kurzschlussschutz zu achten. Ausserdem soll die Schaltung die Messung der Stromstärke im Modell-Eingangskreis ermöglichen.

2.3.1 Die H-Brücke

Eine H-Brücke eignet sich sehr gut als Antrieb für Gleichstrommotoren oder als Speisung für beliebige induktive Lasten. Das Kugelwippenmodell des Regelungstechniklabors, das es letzten Endes zu regeln gilt, entspricht im Modell-Eingangskreis dem elektrischen Ersatzschaltbild eines Gleichstrommotors mit Permanentmagnet als Fremd-Erregung *(Abb. 2.3.1).*

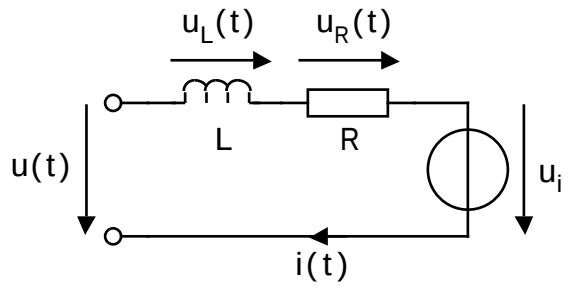


Abb. 2.3.1: Ersatzschaltbild eines Gleichstrommotors mit konstanter Fremderregung

Grundlagen zur H-Brückenschaltung

Eine H-Brückenschaltung besteht im Prinzip aus vier Schaltern, welche durch eine Ansteuerlogik geöffnet und geschlossen werden können *(Abb. 2.3.2).* Je nach Schalterstellung liegt an den Klemmen die positive oder die negative Speisespannung an.

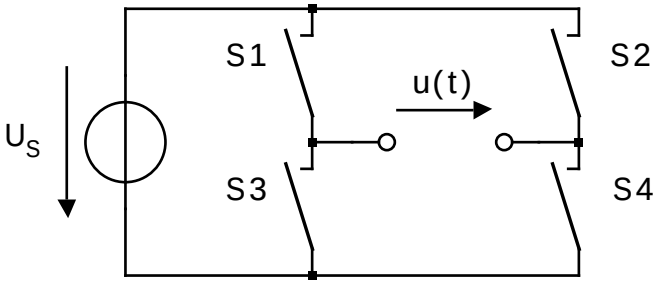


Abb. 2.3.2: Das H-Brücken-Prinzip

Es werden immer nur diagonal gegenüberliegende Schalter gleichzeitig geschlossen.
Während einer Zeit t_{ein} seien Schalter 1 und 4 geschlossen und Schalter 2 und 3 geöffnet. Während einer Zeit (T-t_{ein}) seien Schalter 1 und 4 geöffnet und Schalter 2 und 3 geschlossen. Das heisst, dass u(t) die Form einer Rechteckschwingung hat und sich mit *Gl. 2.3.1* beschreiben lässt. *Abb. 2.3.3* zeigt den zeitlichen Verlauf von u(t).

$$u(t) = \begin{cases} U_S & \text{für } 0 \le t < t_{ein} \\ -U_S & \text{für } t_{ein} \le t < T \end{cases} \qquad \textit{Gl. 2.3.1}$$

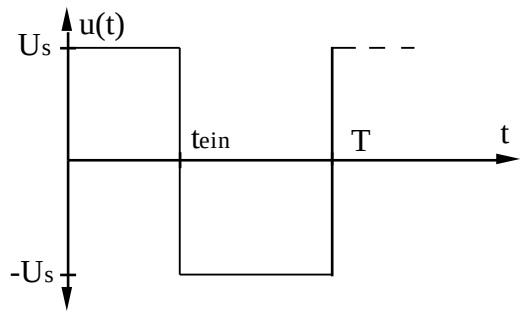


Abb. 2.3.3: Zeitlicher Verlauf von u(t)

Theoretische Vorarbeit, Wahl der geeigneten H-Brücke

Um für einen bestimmten Verwendungszweck eine geeignete H-Brücke auswählen zu können, interessiert nicht nur die maximale Brückenspannung (hier in diesem Fall ±Us), sondern auch der Strom i(t). Dieser Strom i(t) fliesst ausser auch immer in einem der Brückenpfade. Das heisst, dass man den maximal möglichen Brückenstrom kennen muss.

Bestimmung der zu schaltenden Brückenspannung

In früheren Versuchen wurde die Kugelwippe mit maximal 24V betrieben. Wir übernehmen diesen Erfahrungswert und legen die Spannung Us auf 24V fest.

Bestimmung des Brückenstromes i(t)

Da die Last in Form der Kugelwippe kein statisches sondern ein dynamisches Gebilde ist, muss eine Differentialgleichung (*Gl. 2.3.2*) gelöst werden, um i(t) zu erhalten. Diese erhält man, indem man das Kirchhoff'sche Maschengesetz auf die Schaltung in *Abb. 2.3.1* anwendet.

$$u(t) = L\frac{di}{dt} + R\ i(t) + u_i(\omega) \qquad \textit{Gl. 2.3.2}$$

In *Gl. 2.3.2* fällt die Grösse $u_i(\omega)$ auf. Sie entspricht der induzierten Spannung und ist proportional zur Winkelgeschwindigkeit der Wippe. Da diese Spannung im allgemeinen nicht einfach so bekannt und u(t) nicht stetig ist, ist es sehr schwierig die Differentialgleichung zu lösen.

Ein anderer Weg, Informationen über i(t) zu erhalten, führt über Methode, die Signale als Überlagerung von Gleich- und Wechselanteilen zu betrachten.
Näherungsweise, kann i(t) als Überlagerung eines Gleichstromes und eines Wechselstromes mit der Form einer Dreieckschwingung betrachtet werden, da sich der Kugelwippen-Eingangskreis *(Abb. 2.3.4)* ähnlich wie ein Abwärtswandler verhält (siehe auch [1]).

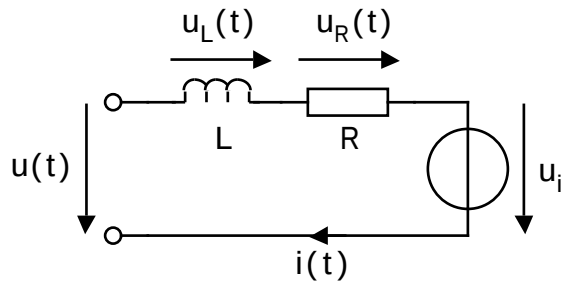


Abb. 2.3.4: Ersatzschaltbild des Kugelwippen-Eingangskreises

Daraus ergeben sich folgende Beziehungen für die Gleichgrössen:

$i(t) \rightarrow \bar{I}$ = konstant

$u_L(t) = L\frac{di}{dt} \rightarrow L\frac{d\bar{I}}{dt} = 0$

$u_R(t) = R\ i(t) \rightarrow R\ \bar{I}$ = konstant

$u_i(\omega) \rightarrow \overline{U_i}$

aus *Gl. 2.3.1* folgt für u(t): $u(t) \rightarrow \overline{U} = \frac{t_{ein}}{T}U_S - \frac{T - t_{ein}}{T}U_S = (2\frac{t_{ein}}{T} - 1)\ U_S$

Damit wird aus *Gl. 2.3.2*:

$$u(t) = L\frac{di}{dt} + R\ i(t) + u_i(\omega) \quad \rightarrow \quad (2\frac{t_{ein}}{T} - 1)\ U_S = R\ \bar{I} + \overline{U_i} \qquad \textit{Gl. 2.3.3}$$

Nun wird *Gl. 2.3.3* nach $\bar{I}$ aufgelöst und $x = \frac{t_{ein}}{T}$ substituiert

$$\bar{I} = (2x - 1)\frac{U_S}{R} - \frac{U_i}{R} \quad \text{für } 0 \le x \le 1 \qquad \textit{Gl. 2.3.4}$$

Gl. 2.3.4 beschreibt den Gleichstromanteil von i(t). Die Grösse x heisst Tastverhältnis.

[1]: Tietze-Schenk, Halbleiter-Schaltungstechnik, 9. Auflage, Seite 563 ff.

Beziehungen der Wechselgrössen:

Gehen wir von der bereits erwähnten Annahme aus, dass der Wechselanteil von i(t) eine Dreieckschwingung ausführt *(Abb. 2.3.5)*, so kann die Differenz ΔI berechnet werden.

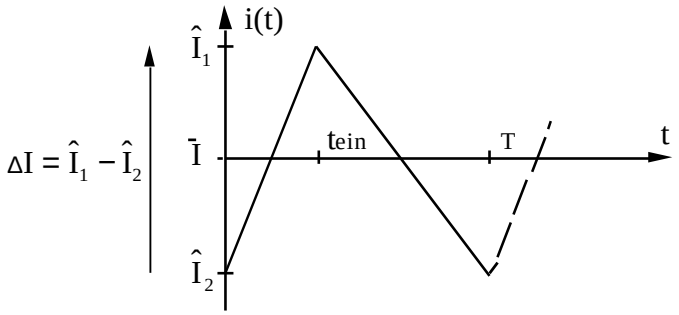


Abb. 2.3.5: Zeitlicher Verlauf von i(t)

Während der Zeit $0 \le t < t_{ein}$ gilt :

$u(t) = U_S$

$u_R(t) = R\ \bar{I}$

$u_L(t) = L\frac{\Delta I}{\Delta t} = L\frac{\Delta I}{t_{ein}}$

Daraus wird mit *Gl. 2.3.2*:

$$u(t) = L\frac{di}{dt} + R\ i(t) + u_i(\omega) \quad \rightarrow \quad U_S = L\frac{\Delta I}{t_{ein}} + R\ \bar{I} + \overline{U}_i \qquad \textit{Gl. 2.3.5}$$

Nun wird *Gl. 2.3.5* nach Δ*I* aufgelöst

$$\Delta I = \frac{t_{ein}}{L}\left(U_S - U_i - R\ \bar{I}\right) \qquad \textit{Gl. 2.3.6}$$

ΔI kann aber auch für die Zeit $t_{ein} \le t < T$ berechnet werden. Für diese Zeit gilt:

$u(t) = -U_S$

$u_L(t) = L\frac{\Delta I}{\Delta t} = L\frac{\Delta I}{T - t_{ein}}$

daraus lässt sich wiederum ΔI berechnen

$$\Delta I = -\frac{T - t_{ein}}{L}\left(U_S + U_i + R\ \bar{I}\right) \qquad \textit{Gl. 2.3.7}$$

Gl. 2.3.6 und *Gl. 2.3.7* beschreiben den Wechselstomanteil von i(t). ΔI entspricht der Stromänderung. Während der Zeit $0 \le t < t_{ein}$ ist diese Änderung positiv *(Gl. 2.3.6)* und während der Zeit $t_{ein} \le t < T$ ist sie negativ *(Gl. 2.3.7).*

Aber betragsmässig stimmen die beiden ΔI überein. Das heisst, dass deren Summe 0 ergeben muss. Anhand dieser Tatsache kann jetzt die Gültigkeit der bisherigen Annahmen und Näherungen überprüft werden. *Gl. 2.3.6* und *Gl. 2.3.7* werden addiert, was zur Folge hat, dass die Wechselanteile ΔI verschwinden und sich eine Beziehung für Gleichgrössen herauskristallisiert:

$$0 = \frac{t_{ein}}{L}\left(U_S - U_i - R\ \bar{I}\right) - \frac{T - t_{ein}}{L}\left(U_S + U_i + R\ \bar{I}\right)$$

$$0 = U_S\left(\frac{t_{ein}}{L} - \frac{T - t_{ein}}{L}\right) - U_i\left(\frac{t_{ein}}{L} + \frac{T - t_{ein}}{L}\right) - R\ \bar{I}\left(\frac{t_{ein}}{L} + \frac{T - t_{ein}}{L}\right)$$

$$0 = U_S\ \left(2t_{ein} - T\right) - U_i\ T - R\ \bar{I}\ T$$

$$0 = U_S\ \left(\frac{2t_{ein}}{T} - 1\right) - U_i - R\ \bar{I}$$

Diese Gleichung wird jetzt nach $\bar{I}$ aufgelöst und $x = \frac{t_{ein}}{T}$ substituiert

$$\bar{I} = (2x-1)\frac{U_S}{R} - \frac{U_i}{R} \quad \text{für } 0 \le x \le 1 \qquad \textit{Gl. 2.3.8}$$

Die Gleichung *Gl. 2.3.8* ist dieselbe wie Gleichung *Gl. 2.3.4.* Daher lässt sich sagen, dass die Annahmen und Näherungen über den zeitlichen Verlauf von i(t) durchaus Gültigkeit haben.

Nachdem die theoretische Vorarbeit verrichtet ist, soll anhand von Zahlenwerten eine geeignete H-Brücke ausgewählt werden.

Bei der Vorführung wurde die Wippe harmonisch angeregt, und es floss ein maximaler Strom von etwa 3A durch den Modell-Eingangskreis. Bei einer Lage-Regelung der Kugel sind aber keine so hohen Beschleunigungen nötig, sodass der Gleichstromanteil diese 3A nicht erreichen wird. Da das Stellglied aber getaktet ist, wird der Spitzenstrom noch um ΔI/2 grösser als der Gleichstromanteil.

Messungen am Wippenmodell haben ergeben: R=2.67Ω, L=2.24mH

Aus *Gl. 2.3.6* soll für den Fall eines Stillstandes (Stillstand: U_i=0) der Wippe und für t_{ein}=T/2 (U_i=0 und x=1/2 bedeutet: I=0) das ΔI/2 berechnet werden. Für das Wippenmodell werden noch neue Spulen mit der doppelten Windungszahl und dickerem Draht gewickelt. Das heisst, dass die Induktivität vier mal grösser wird und der Wicklungswiderstand R etwa gleich bleiben wird.
Es wurde eine Taktfrequenz von mindestens 20kHz gewählt, damit keine störenden Pfeifgeräusche entstehen. Das heisst: T ≤ 50μs

Mit $L = 9mH,\ R = 2.7\Omega, U_S = 24V, t_{ein} = 2.5\mu s$ folgt aus *Gl. 2.3.6*: $\frac{\Delta I}{2} = 33mA$

Wir entschieden uns deshalb für die von Herrn Golder vorgeschlagene H-Brücke des Typs **LMD 18200** von National Semiconductor, die sich für unser Vorhaben bestens eignet.

Ein Auszug aus dem Datenblatt befindet sich auf den nächsten fünf Seiten.

Datenblatt LMD 18200 (entnommen aus [2])

National Semiconductor

LMD18200 3A, 55V H-Bridge

General Description

The LMD18200 is a 3A H-Bridge designed for motion control applications. The device is built using a multi-technology process which combines bipolar and CMOS control circuitry with DMOS power devices on the same monolithic structure. Ideal for driving DC and stepper motors; the LMD18200 accommodates peak output currents up to 6A. An innovative circuit which facilitates low-loss sensing of the output current has been implemented.

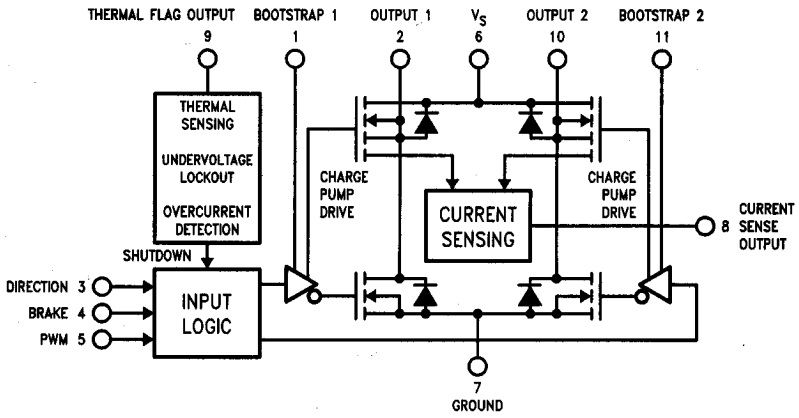
Features

- Delivers up to 3A continuous output
- Operates at supply voltages up to 55V
- Low $R_{DS}(ON)$ typically 0.3Ω per switch
- TTL and CMOS compatible inputs
- No "shoot-through" current
- Thermal warning flag output at 145°C
- Thermal shutdown (outputs off) at 170°C
- Internal clamp diodes
- Shorted load protection
- Internal charge pump with external bootstrap capability

Applications

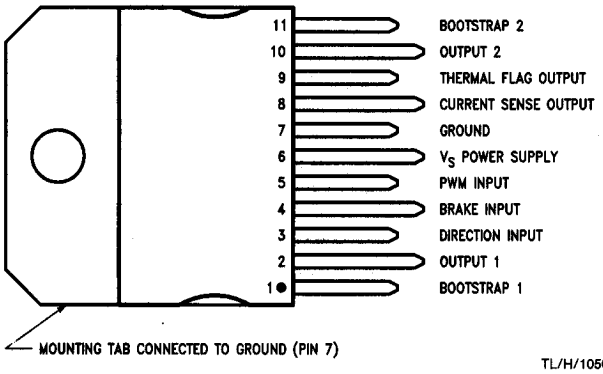
- DC and stepper motor drives
- Position and velocity servomechanisms
- Factory automation robots
- Numerically controlled machinery
- Computer printers and plotters

Functional Diagram



TL/H/10568–1

Connection Diagram and Ordering Information



TL/H/10568–2

Top View

Order Number LMD18200T
See NS Package TA11B

[2]: National Semiconductor, Power IC's Databook, 1993 Edition, Seite 4-61 bis 4-65

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Total Supply Voltage (V_S, Pin 6)	60V
Voltage at Pins 3, 4, 5, 8 and 9	12V
Voltage at Bootstrap Pins (Pins 1 and 11)	V_{OUT} + 16V
Peak Output Current (200 ms)	6A
Continuous Output Current (Note 2)	3A
Power Dissipation (Note 3)	25W
Power Dissipation (T_A = 25°C, Free Air)	3W
Junction Temperature, $T_{J(max)}$	150°C
ESD Susceptibility (Note 4)	1500V
Storage Temperature, T_{STG}	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Operating Ratings (Note 1)

Junction Temperature, T_J	−40°C to +125°C
V_S Supply Voltage	+12V to +55V

Electrical Characteristics

The following specifications apply for V_S = 42V, unless otherwise specified. **Boldface** limits apply over the entire operating temperature range, −40°C ≤ T_J ≤ +125°C, all other limits are for T_A = T_J = 25°C. (Note 5)

Symbol	Parameter	Conditions	Typ	Limit	Units
$R_{DS}(ON)$	Switch ON Resistance	Output Current = 3A (Note 6)	0.33	0.4/**0.6**	Ω (max)
$R_{DS}(ON)$	Switch ON Resistance	Output Current = 6A (Note 6)	0.33	0.4/**0.6**	Ω (max)
V_{CLAMP}	Clamp Diode Forward Drop	Clamp Current = 3A (Note 6)	1.2	1.5	V (max)
V_{IL}	Logic Low Input Voltage	Pins 3, 4, 5		**−0.1** **0.8**	V (min) V (max)
I_{IL}	Logic Low Input Current	V_{IN} = −0.1V, Pins = 3, 4, 5		**−10**	μA (max)
V_{IH}	Logic High Input Voltage	Pins 3, 4, 5		**2** **12**	V (min) V (max)
I_{IH}	Logic High Input Current	V_{IN} = 12V, Pins = 3, 4, 5		**10**	μA (max)
	Current Sense Output	I_{OUT} = 1A (Note 8)	377	325/**300** 425/**450**	μA (min) μA (max)
	Current Sense Linearity	1A ≤ I_{OUT} ≤ 3A (Note 7)	±6	±9	%
	Undervoltage Lockout	Outputs turn OFF		9 11	V (min) V (max)
T_{JW}	Warning Flag Temperature	Pin 9 ≤ 0.8V, I_L = 2 mA	145		°C
$V_F(ON)$	Flag Output Saturation Voltage	T_J = T_{JW}, I_L = 2 mA	0.15		V
$I_F(OFF)$	Flag Output Leakage	V_F = 12V	0.2	10	μA (max)
T_{JSD}	Shutdown Temperature	Outputs Turn OFF	170		°C
I_S	Quiescent Supply Current	All Logic Inputs Low	13	25	mA (max)
t_{Don}	Output Turn-On Delay Time	Sourcing Outputs, I_{OUT} = 3A Sinking Outputs, I_{OUT} = 3A	300 300		ns ns
t_{on}	Output Turn-On Switching Time	Bootstrap Capacitor = 10 nF Sourcing Outputs, I_{OUT} = 3A Sinking Outputs, I_{OUT} = 3A	 100 80		 ns ns
t_{Doff}	Output Turn-Off Delay Times	Sourcing Outputs, I_{OUT} = 3A Sinking Outputs, I_{OUT} = 3A	200 200		ns ns
t_{off}	Output Turn-Off Switching Times	Bootstrap Capacitor = 10 nF Sourcing Outputs, I_{OUT} = 3A Sinking Outputs, I_{OUT} = 3A	 75 70		 ns ns
t_{pw}	Minimum Input Pulse Width	Pins 3, 4 and 5	1		μs
t_{cpr}	Charge Pump Rise Time	No Bootstrap Capacitor	20		μs

Electrical Characteristics Notes

Note 1: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions.

Note 2: See Application Information for details regarding current limiting.

Note 3: The maximum power dissipation must be derated at elevated temperatures and is a function of $T_{J(max)}$, θ_{JA}, and T_A. The maximum allowable power dissipation at any temperature is $P_{D(max)} = (T_{J(max)} - T_A)/\theta_{JA}$, or the number given in the Absolute Ratings, whichever is lower. The typical thermal resistance from junction to case (θ_{JC}) is 1.0°C/W and from junction to ambient (θ_{JA}) is 30°C/W. For guaranteed operation $T_{J(max)}$ = 125°C.

Note 4: Human-body model, 100 pF discharged through a 1.5 kΩ resistor. Except Bootstrap pins (pins 1 and 11) which are protected to 1000V of ESD.

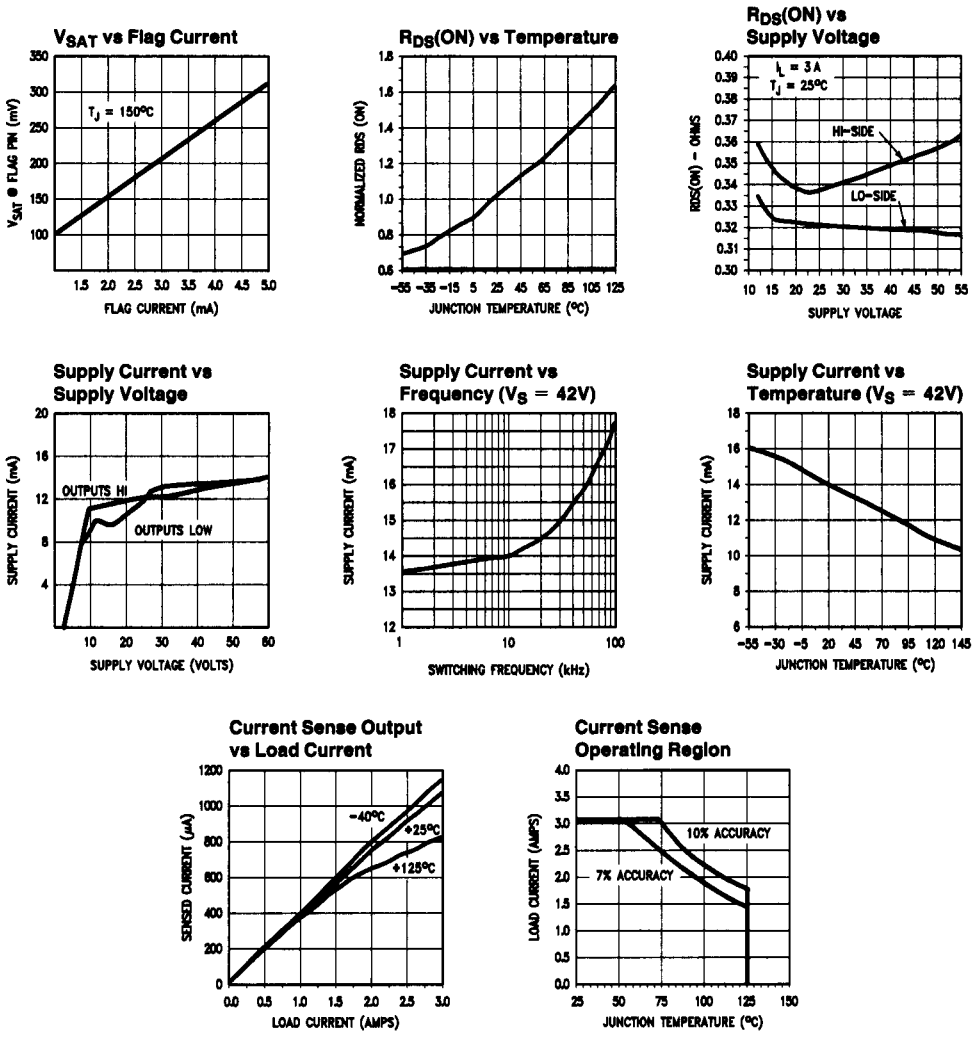
Note 5: All limits are 100% production tested at 25°C. Temperature extreme limits are guaranteed via correlation using accepted SQC (Statistical Quality Control) methods. All limits are used to calculate AOQL, (Average Outgoing Quality Level).

Note 6: Output currents are pulsed (t_W < 2 ms, Duty Cycle < 5%).

Note 7: Regulation is calculated relative to the current sense output value with a 1A load.

Note 8: Selections for tighter tolerance are available. Contact factory.

Typical Performance Characteristics



Pinout Description (See Connection Diagram)

Pin 1, BOOTSTRAP 1 Input: Bootstrap capacitor pin for half H-bridge number 1. The recommended capacitor (10 nF) is connected between pins 1 and 2.

Pin 2, OUTPUT 1: Half H-bridge number 1 output.

Pin 3, DIRECTION Input: See Table I. This input controls the direction of current flow between OUTPUT 1 and OUTPUT 2 (pins 2 and 10) and, therefore, the direction of rotation of a motor load.

Pin 4, BRAKE Input: See Table I. This input is used to brake a motor by effectively shorting its terminals. When braking is desired, this input is taken to a logic high level and it is also necessary to apply logic high to PWM input, pin 5. The drivers that short the motor are determined by the logic level at the DIRECTION input (Pin 3): with Pin 3 logic high, both current sourcing output transistors are ON; with Pin 3 logic low, both current sinking output transistors are ON. All output transistors can be turned OFF by applying a logic high to Pin 4 and a logic low to PWM input Pin 5; in this case only a small bias current (approximately −1.5 mA) exists at each output pin.

Pin 5, PWM Input: See Table I. How this input (and DIRECTION input, Pin 3) is used is determined by the format of the PWM Signal.

Pin 6, V_S Power Supply

Pin 7, GROUND Connection: This pin is the ground return, and is internally connected to the mounting tab.

Pin 8, CURRENT SENSE Output: This pin provides the sourcing current sensing output signal, which is typically 377 μA/A.

Pin 9, THERMAL FLAG Output: This pin provides the thermal warning flag output signal. Pin 9 becomes active-low at 145°C (junction temperature). However the chip will not shut itself down until 170°C is reached at the junction.

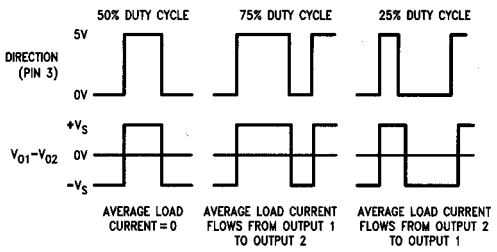
Pin 10, OUTPUT 2: Half H-bridge number 2 output.

Pin 11, BOOTSTRAP 2 Input: Bootstrap capacitor pin for Half H-bridge number 2. The recommended capacitor (10 nF) is connected between pins 10 and 11.

TABLE I. Logic Truth Table

PWM	Dir	Brake	Active Output Drivers
H	H	L	Source 1, Sink 2
H	L	L	Sink 1, Source 2
L	X	L	Source 1, Source 2
H	H	H	Source 1, Source 2
H	L	H	Sink 1, Sink 2
L	X	H	NONE

Locked Anti-Phase PWM Control



TL/H/10568–4

Application Information

TYPES OF PWM SIGNALS

The LMD18200 readily interfaces with different forms of PWM signals. Use of the part with two of the more popular forms of PWM is described in the following paragraphs.

Simple, locked anti-phase PWM consists of a single, variable duty-cycle signal in which is encoded both direction and amplitude information. A 50% duty-cycle PWM signal represents zero drive, since the net value of voltage (integrated over one period) delivered to the load is zero. For the LMD18200, the PWM signal drives the direction input (pin 3) and the PWM input (pin 5) is tied to logic high.

Sign/magnitude PWM consists of separate direction (sign) and amplitude (magnitude) signals. The (absolute) magnitude signal is duty-cycle modulated, and the absence of a pulse signal (a continuous logic low level) represents zero drive. Current delivered to the load is proportional to pulse width. For the LMD18200, the DIRECTION input (pin 3) is driven by the sign signal and the PWM input (pin 5) is driven by the magnitude signal.

USING THE CURRENT SENSE OUTPUT

The CURRENT SENSE output (pin 8) has a sensitivity of 377 μA per ampere of output current. For optimal accuracy and linearity of this signal, the value of voltage generating resistor between pin 8 and ground should be chosen to limit the maximum voltage developed at pin 8 to 5V, or less. The maximum voltage compliance is 12V.

It should be noted that the recirculating currents (free wheeling currents) are ignored by the current sense circuitry. Therefore, only the currents in the upper sourcing outputs are sensed.

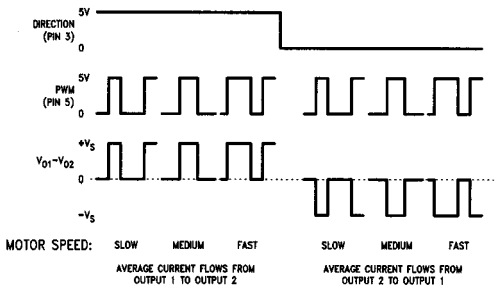
USING THE THERMAL WARNING FLAG

The THERMAL FLAG output (pin 9) is an open collector transistor. This permits a wired OR connection of thermal warning flag outputs from multiple LMD18200's, and allows the user to set the logic high level of the output signal swing to match system requirements. This output typically drives the interrupt input of a system controller. The interrupt service routine would then be designed to take appropriate steps, such as reducing load currents or initiating an orderly system shutdown. The maximum voltage compliance on the flag pin is 12V.

SUPPLY BYPASSING

During switching transitions the levels of fast current changes experienced may cause troublesome voltage transients across system stray inductance.

Sign/Magnitude PWM Control



Application Information (Continued)

It is normally necessary to bypass the supply rail with a high quality capacitor(s) connected as close as possible to the V_S Power Supply (Pin 6) and GROUND (Pin 7). A 1 μF high-frequency ceramic capacitor is recommended. Care should be taken to limit the transients on the supply pin below the Absolute Maximum Rating of the device. When operating the chip at supply voltages above 40V a voltage suppressor (transorb) such as P6KE62A is recommended from supply to ground. Typically the ceramic capacitor can be eliminated in the presence of the voltage suppressor. Note that when driving high load currents a greater amount of supply bypass capacitance (in general at least 100 μF per Amp of load current) is required to absorb the recirculating currents of the inductive loads.

CURRENT LIMITING

Current limiting protection circuitry has been incorporated into the design of the LMD18200. With any power device it is important to consider the effects of the substantial surge currents through the device that may occur as a result of shorted loads. The protection circuitry monitors this increase in current (the threshold is set to approximately 10 Amps) and shuts off the power device as quickly as possible in the event of an overload condition. In a typical motor driving application the most common overload faults are caused by shorted motor windings and locked rotors. Under these conditions the inductance of the motor (as well as any series inductance in the V_{CC} supply line) serves to reduce the magnitude of a current surge to a safe level for the LMD18200. Once the device is shut down, the control circuitry will periodically try to turn the power device back on. This feature allows the immediate return to normal operation in the event that the fault condition has been removed. While the fault remains however, the device will cycle in and out of thermal shutdown. This can create voltage transients on the V_{CC} supply line and therefore proper supply bypassing techniques are required.

The most severe condition for any power device is a direct, hard-wired ("screwdriver") long term short from an output to ground. This condition can generate a surge of current through the power device on the order of 15 Amps and require the die and package to dissipate up to 500 Watts of power for the short time required for the protection circuitry to shut off the power device. This energy can be destructive, particularly at higher operating voltages (>30V) so some precautions are in order. Proper heat sink design is essential and it is normally necessary to heat sink the V_{CC} supply pin (pin 6) with 1 square inch of copper on the PCB.

INTERNAL CHARGE PUMP AND USE OF BOOTSTRAP CAPACITORS

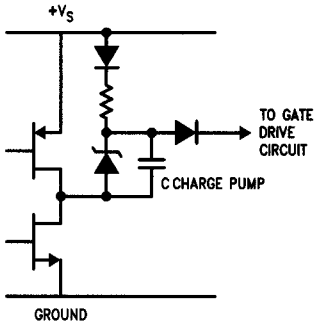
To turn on the high-side (sourcing) DMOS power devices, the gate of each device must be driven approximately 8V more positive than the supply voltage. To achieve this an internal charge pump is used to provide the gate drive voltage. As shown in *Figure 1*, an internal capacitor is alternately switched to ground and charged to about 14V, then switched to V supply thereby providing a gate drive voltage greater than V supply. This switching action is controlled by a continuously running internal 300 kHz oscillator. The rise time of this drive voltage is typically 20 μs which is suitable for operating frequencies up to 1 kHz.

For higher switching frequencies, the LMD18200 provides for the use of external bootstrap capacitors. The bootstrap principle is in essence a second charge pump whereby a large value capacitor is used which has enough energy to quickly charge the parasitic gate input capacitance of the power device resulting in much faster rise times. The switching action is accomplished by the power switches themselves *(Figure 2)*. External 10 nF capacitors, connected from the outputs to the bootstrap pins of each high-side switch provide typically less than 100 ns rise times allowing switching frequencies up to 500 kHz.

INTERNAL PROTECTION DIODES

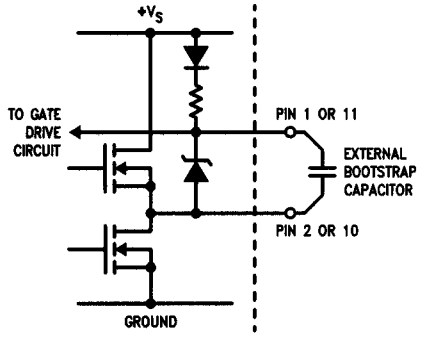
A major consideration when switching current through inductive loads is protection of the switching power devices from the large voltage transients that occur. Each of the four switches in the LMD18200 have a built-in protection diode to clamp transient voltages exceeding the positive supply or ground to a safe diode voltage drop across the switch.

The reverse recovery characteristics of these diodes, once the transient has subsided, is important. These diodes must come out of conduction quickly and the power switches must be able to conduct the additional reverse recovery current of the diodes. The reverse recovery time of the diodes protecting the sourcing power devices is typically only 70 ns with a reverse recovery current of 1A when tested with a full 6A of forward current through the diode. For the sinking devices the recovery time is typically 100 ns with 4A of reverse current under the same conditions.



TL/H/10568–6

FIGURE 1. Internal Charge Pump Circuitry



TL/H/10568–7

FIGURE 2. Bootstrap Circuitry

Aus der Wahrheitstabelle Table I im Datenblatt wird ersichtlich, dass für diese Anwendung nur gerade die ersten beiden Zustände massgebend sind. In diesen Zuständen hat das PWM-Signal einen HIGH-Pegel und je nach Pegel des Signals DIR wird der eine oder der andere Brückenpfad eingeschalten. Bei einer H-Brücke ist zu beachten, dass in einem Pfad die Transistoren nicht mehr leiten, bevor der andere Pfad durchgeschaltet wird. Es muss also bei jedem Wechsel des DIR-Signals eine Verzögerungszeit einbezogen werden, damit die Transistoren Zeit haben, um alle Ladungen abzubauen. Beim **LMD18200** ist diese Verzögerung bereits eingebaut und betrifft den Anwender nicht. Gemäss der von uns gewünschten Funktion der Brücke ergibt sich die Beschaltung in *Abb. 2.3.6* .

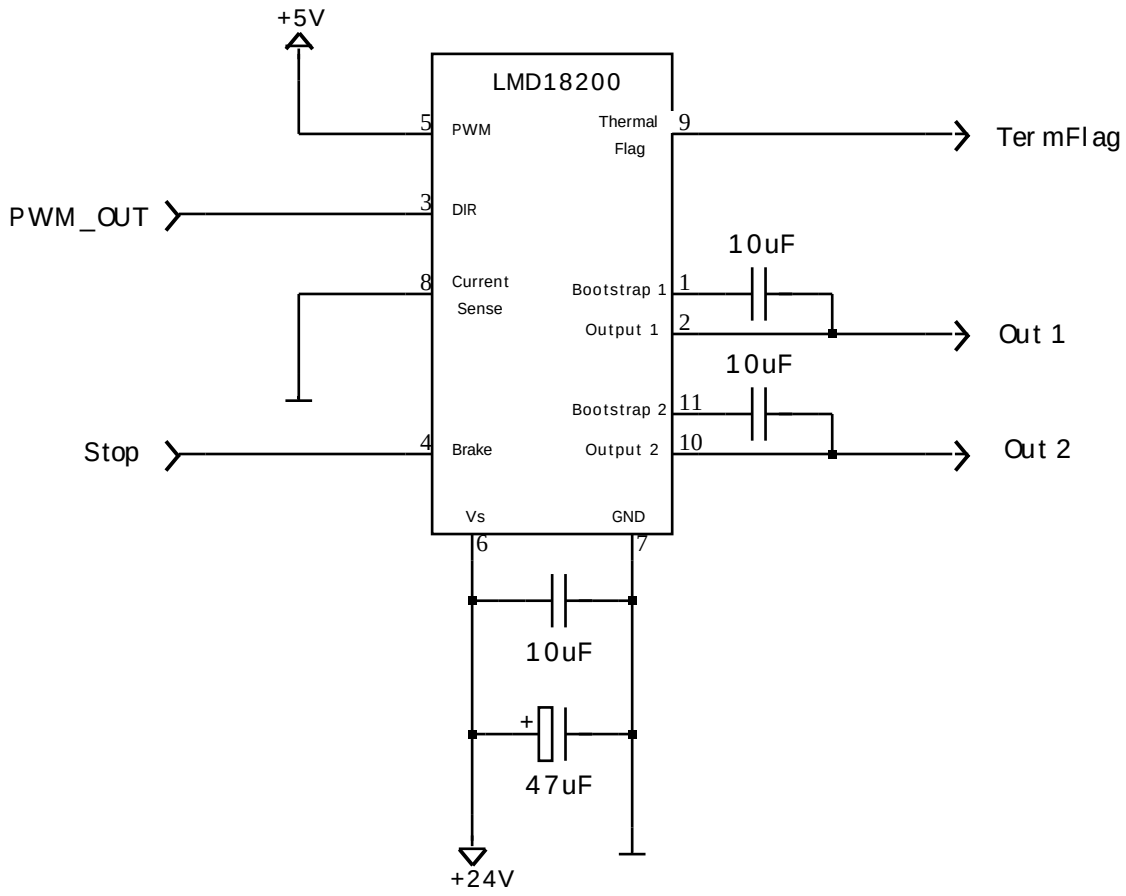


*Abb. 2.3.6: Beschaltung des **LMD18200***

Speisung

Bei der Speisung ist ganz besonders darauf zu achten, dass sie abgeblockt wird. Dies kann mit einem 10µF (MKT) und einem 47µF (ELKO) Kondensator erreicht. Es ist wichtig, dass diese beiden Kondensatoren möglichst nahe bei den Pins des **LMD18200** angebracht werden.

Wenn diese Regeln beachtet werden, können die Störeinflüsse der raschen Stromänderungen auf den Speiseleitungen minimal gehalten werden. Hier sei noch auf die Literatur [9] verwiesen.

Bootstrap-Kondensatoren

Für Frequenzen höher als 1kHz empfiehlt National Semiconductor die Zuschaltung von 10nF-Kondensatoren. Mit Hilfe dieser Kondensatoren wird eine rise-time (Anstiegszeit) von weniger als 100ns am Ausgang garantiert.

Current Sense

Dieser Ausgang liefert einen zum Betrag des Brückenstromes proportionalen Strom. Da bei unser verlangten Strommessung auch das Vorzeichen des Brückenstromes gemessen werden muss, wird dieser Pin von uns nicht benutzt. Damit der Sense-Strom aber weiterhin fliessen kann, wird der Current-Sense-Anschluss auf Masse geschalten.

Thermal Flag

Bei Kristall-Temperaturen von über 145°C geht das Thermal-Flag Signal von HIGH nach LOW. Bei dieser Anwendung soll das Signal abgreifbar sein, damit bei eventuellen späteren Anwendungen des Stellglieds die Möglichkeit einer Temperatur-Überwachung vorhanden ist.
Der **LMD18200** hat als Feature noch einen sogenannten 'thermal shutdown'. Dies bedeutet, dass der Baustein bei Temperaturen von über 170°C den Betrieb sofort unterbricht.

Brake

Das Brake-Eingangssignal erlaubt dem Anwender, den Betrieb der Brücke zu stoppen. Der Eingang ist High-aktiv und bewirkt, dass die Klemmen über der Last kurzgeschlossen werden. Dieses Signal wird in Kapitel 2.3.3 als Stop-Signal bei der Überstromschutzschaltung verwendet.

[9]: TWI; Andreas Graf/Roger Meier, Projektarbeit I/93; Geschaltete Konzepte

2.3.2 Stromdetektor

Der Eingangs-Strom des Wippenmodells stellt eine Zustandsgösse des Systems dar und muss mit der A/D-Wandler-Hardware gemessen werden können. Mit der vorliegenden A/D-Wandler-Hardware können Spannungen von -2.5V bis +2.5V verarbeitet werden. Daher soll eine Schaltung entwickelt werden, die eine Spannung in diesem Bereich generiert, welche proporional zum Brückenstrom des Stellglieds ist.

LEM

Eine sehr komfortable Möglichkeit der Srommessung bietet der 'Multi-range current transducer' der Firma **LEM**, kurz: **LEM Module LA 25-NP**. Der zu messende Strom fliesst durch eine Wicklung im innern des Bausteines. Je nach zu messender Stromstärke, kann diese Wicklungszahl im Bereich 1 bis 5 variiert werden. Der Eingangsstrom I_n erzeugt einen Fluss im Kern, auf den die Windungen aufgewickelt sind. Dieser Fluss wird mit einer in den magnetischen Kreis integrierten Hall-Sonde gemessen. Aufgrund dieser Messung wird in einen zweiten Spule, die auf den selben Kern gewickelt ist, ein Strom I_s eingestellt. Dieser Strom erzeugt im Kern einen entgegengesetzten Fluss. Eine Regelung regelt diesen Strom so, dass der Fluss im Kern verschwindet. Damit ist gewährleistet, dass I_s sich direkt proportional zum Eingangsstrom verhält. Die Funktionsweise des **LEM LA 25-NP** ist in *Abb. 2.3.7* dargestellt.

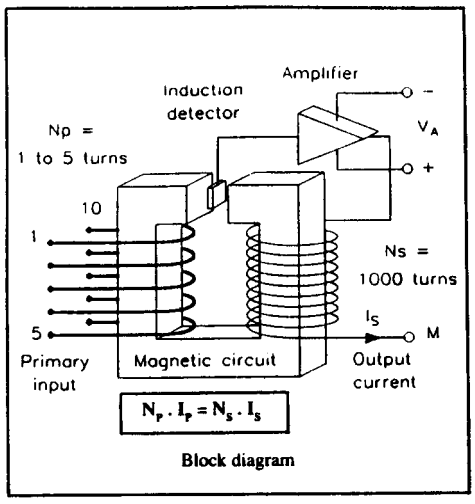


*Abb. 2.3.7: Funktionsdiagramm **LEM LA 25-NP***

Abb. 2.3.8 zeigt einen Auszug aus dem Datenblatt des **LEM LA 25-NP**.

Electrical characteristics

Nominal current	I_N	25	[A.t rms]
Measuring range	I_P	0 to +/- 36	[A.t]
Load resistance	R_M	min. 100	[Ohm]
		max. 190	[Ohm]
Maximum error at +25°C	e	±0.6	[% I_N]
Nominal output current	I_S	25	[mA]
Supply voltage	V_A	±15 (±5%)	[V]
Turn ratio		1-2-3-4-5 / 1000	
Dielectric strength		2.5 kVrms/50 Hz/1min	

General characteristics

Operating Temperature	0 to +70	[°C]
Storage Temperature	-25 to +85	[°C]
Current Consumption	10 + I_S (output current)	[mA]
Secondary Internal Resistance	110 (at +70°C)	[Ohm]
Primary Internal Resistance	<1.25	[mOhm/turn]
Insulation Resistance	>1500 (at 500V and +25°C)	[MOhm]
Weight	18	[g]
Construction	potted in insulated self-extinguishing plastic case	
Current direction	a positive output current is obtained on terminal "M" when the primary current flows from terminals 1, 2, 3, 4 and 5 to terminals 10, 9, 8, 7 and 6.	

Accuracy - Dynamic performance

Parameter	Symbol	Conditions	Typical	Maximum	Unit
Offset	I_{OS}	I_P = 0A, T = +25°C		±0.05	mA
Residual current *	I_{rt}	I_P = 0A, T = +25°C		±0.08	mA
Offset current drift	d I_{OS}	I_P = 0A, T = 0°C to +25°C	±0.20	±0.30	mA
with temperature	d I_{OS}	I_P = 0A, T = +25°C to +70°C	±0.25	±0.60	mA
Linearity	ε_L	I_{os} = 0mA		±0.2	% I_N
Delay time	td	I_P = 25A.t (see graph)		1	µS
Bandwidth	f	I_P = 25A.t at -1dB	DC to 150		kHz

* Result of the coercive field of the magnetic circuit

Number of primary turns	Primary current nominal I_N [A]	Primary current maximum I_P [A]	Nominal output current I_S [mA]	Turn ratio	Primary resistance [mOhm]	Primary insertion inductance [µH]	Recommended connections
1	25	36	25	1/1000	0.3	0.023	
2	12	18	24	2/1000	1.1	0.09	
3	8	12	24	3/1000	2.5	0.21	
4	6	9	24	4/1000	4.4	0.37	
5	5	7	25	5/1000	6.3	0.58	

*Abb. 2.3.8: Datenblatt **LEM LA 25-NP***

Da unser Brückenstrom maximal etwa 3A betragen wird, entscheiden wir uns für 5 Primärwicklungen am **LEM**. Dabei ist ein maximaler Eingangsstrom I_n von 5A zulässig. Das Stromverhältnis beträgt in dieser Betriebsart 200:1.

Um eine für den A/D-Wandler messbare Grösse zu erhalten, muss die Schaltung noch erweitert werden. Da der Wandler nur Spannungen verarbeitet, liegt es nahe, den Strom durch einen Widerstand fliessen zu lassen um eine Spannung zu erhalten. Für die Regelung ist aber nur der Gleichstromanteil des Brückenstromes massgebend. Daher wird dem Widerstand noch ein Filter nachgeschaltet, um den hochfrequenten Wechselanteil herauszufiltern. Dabei muss das aktive Filter das Signal so verstärken, dass ein Brückenstrom von 3A eine Messspannung von 2.5V ergibt, um die volle Auflösung des A/D-Wandlerkanals ausnützen zu können. *Abb. 2.3.9* zeigt das Prinzipschema der Stromdetektorschaltung.

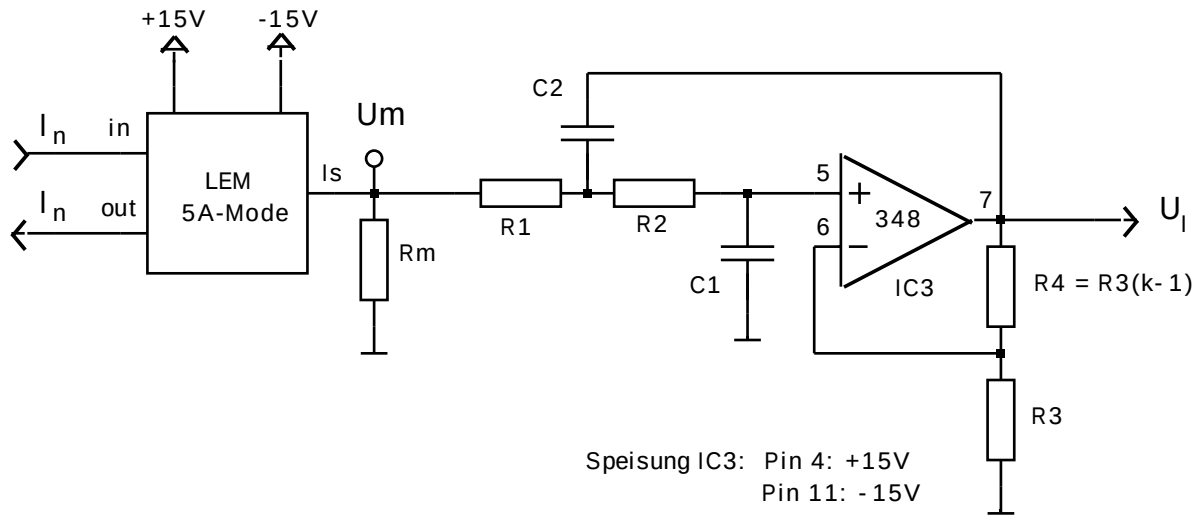


Abb. 2.3.9: Prinzipschema Stromdetektor

Dimensionierung der Stromdetektorschaltung

Für R_m wird ein Widerstand von $100\,\Omega$ eingesetzt. Daraus ergibt sich für $G_1(s) = \frac{U_m(s)}{I_n(s)} = 0.5\frac{A}{V}$

Die Übertragungsfunktion des Filters sieht folgendermassen aus :

$$G_2(s) = \frac{U_m(s)}{U_I(s)} = \frac{k}{1 + P\ \omega_g(C_1(R_1 + R_2) + R_1C_2(1-k)) + P^2\ \omega_g{}^2 R_1R_2C_1C_2} \qquad \textit{Gl. 2.3.9}$$

Die Gesamtübertragungsfunktion $G(s) = \frac{U_I(s)}{I_n(s)}$ lautet somit : $G(s) = G_1(s)\ G_2(s)$

Wie schon erwähnt, soll für tiefe Frequenzen die Ausgangspannung U_I =2.5V betragen, wenn ein Strom I_n =3A fliesst. Es wird eine neue Grösse K eingeführt, die das Übertragungsmass von I_n zu U_I beschreibt.

Bei tiefen Frequenzen gilt : $s \to 0$ $\quad G_2(s=0)=k \quad G(s=0)=k \;\; 0.5\frac{\mathrm{V}}{\mathrm{A}} = K$

für K gilt aber auch : $K = \frac{2.5\mathrm{V}}{3\mathrm{A}}$ daraus lässt sich k berechnen : $k = \frac{5}{3}$

Aus der Schaltung in *Abb. 2.3.9* geht hervor : $\frac{R_4}{R_3} = k - 1 = \frac{2}{3}$ $\quad \underline{\underline{R_3 = 300\mathrm{k}\Omega,\ R_4 = 200\mathrm{k}\Omega}}$

Die A/D-Wandler-Kanäle sind für Frequenzen bis 200Hz ausgelegt. Bei höheren Frequenzen beginnen die Antialiasing-Filter zu dämpfen. Die Grenzfrequenz des aktiven Filters soll also mindestens 200Hz betragen. Wir entschieden uns für 300Hz. Ausserdem soll das Filter Butterworthcharakter haben.

$G_2(s)$ hat diese Form : $G_2(s) = \frac{A_0}{1 + P\ a + P^2\ b}$ $\quad$ wobei $A_0 = k$

Jetzt werden die Werte für a und b aus der entsprechenden Tabelle in [1] herausgelesen, und mittels Koeffiezientenvergleich die beiden Nennerpolynome verglichen. Es entsteht ein Gleichungssystem mit 2 Unbekannten. Es sind aber 4 unbekannte Grössen zu bestimmen. Daher werden die Werte für R1 und R2 schon von vornherein festgelegt.

$f_g = 300\,\mathrm{Hz}, \quad \omega_g = 2\pi \;\; f_g = 1885\,s^{-1}$

Festlegung : $\underline{\underline{R_1 = R_2 = 100\,\mathrm{k}\Omega}}$ $\quad$ aus Tabelle : $a = \sqrt{2},\ b = 1$

Koeffizientenvergleich : $\left| \begin{array}{c} a = \omega_g (C_1(R_1 + R_2) + R_1 C_2 (1-k)) \\ b = \omega_g{}^2 R_1 R_2 C_1 C_2 \end{array} \right|$ $\quad C_1 = 5.46\,\mathrm{nF}, C_2 = 5.15\,\mathrm{nF}$

Kondensatoren aus der E - Reihe : $\underline{\underline{C_1 = C_2 = 4.7\,\mathrm{nF}}}$

Messungen

Um den Stromdetektor zu testen, wurde die H-Brücke mit einem kleinen 24V-Gleichstrom-Motor aus dem Regelungstechniklabor belastet. Als PWM-Geber wurde ein Pulsbreitenmodulator des Typs **SG3731** verwendet. Wie dieser Baustein zu beschalten ist, kann der Literatur [9] entnommen werden. Der **SG3731** ist so dimensioniert worden, dass er eine PWM mit der Frequenz von ca. 22kHz liefert.

Bei den nachfolgenden Messungen 1 bis 3 *(Abb. 2.3.10 bis Abb. 2.3.12)* wurde das Tastverhaltnis so variiert, dass sich verschiedene Werte für den Gleichstromanteil von $I_n(t)$ einstellten.

- Auf dem KO-Kanal CH1 wurde jeweils Eingangspannung $U_{PWM}(t)$ am Pin 3 der H-Brücke **LMD18200** gemessen.
- Auf dem KO-Kanal CH2 wurde die Spannung $U_m(t)$ gemessen. Sie ist exakt proportional zum Brückenstrom $I_n(t)$. Ein U_m von 1V entspricht einem Strom I_n von 2A.
- Auf dem KO-Kanal CH3 wurde die Mess-Spannung $U_I(t)$ gemessen.

[1]: Tietze-Schenk, Halbleiter-Schaltungstechnik, 9. Auflage, Seite 391 ff.
[9]: TWI; Andreas Graf/Roger Meier, Projektarbeit I/93; Geschaltete Konzepte

Messung 1 *(Abb. 2.3.10)* : Tastverhältnis x=0.5: Mittelwert von $I_n(t)$ beträgt 0A.

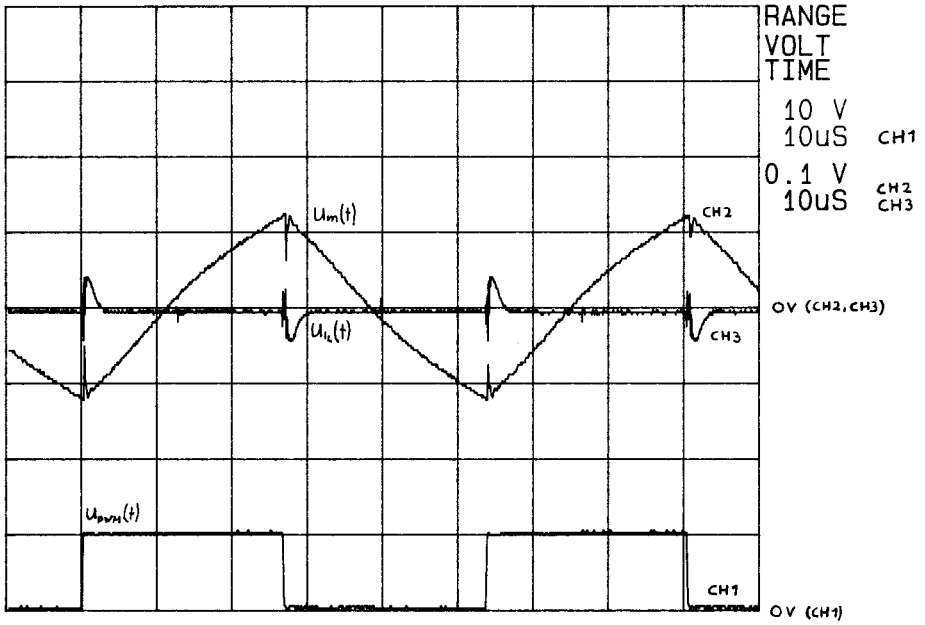


Abb. 2.3.10: Messung 1

Messung 2 *(Abb. 2.3.11)* : Tastverhältnis x>0.5: Mittelwert von $I_n(t)$ ist positiv

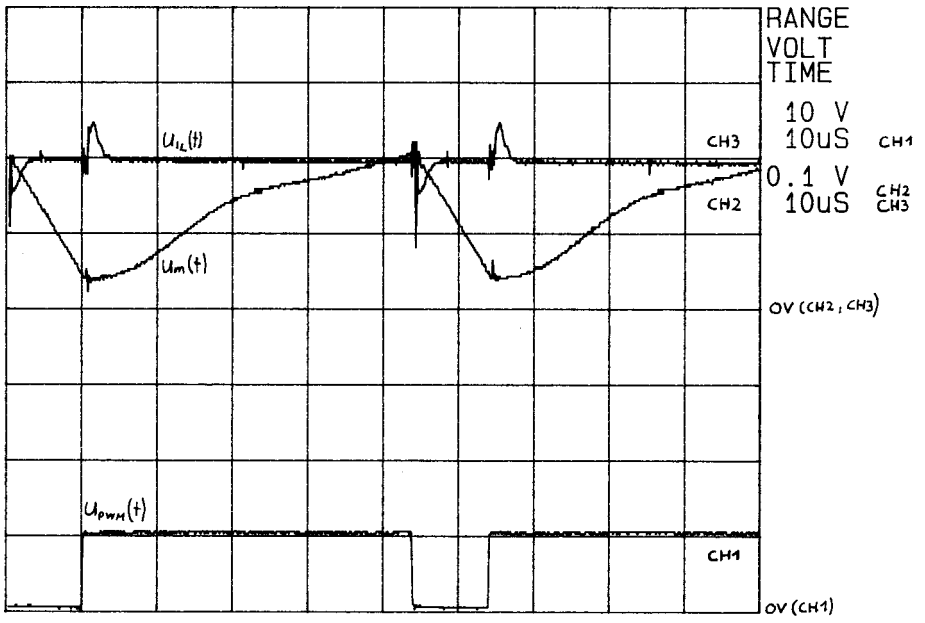


Abb. 2.3.11: Messung 2

Messung 3 *(Abb. 2.3.12)* : Tastverhältnis x<0.5: Mittelwert von $I_n(t)$ ist negativ

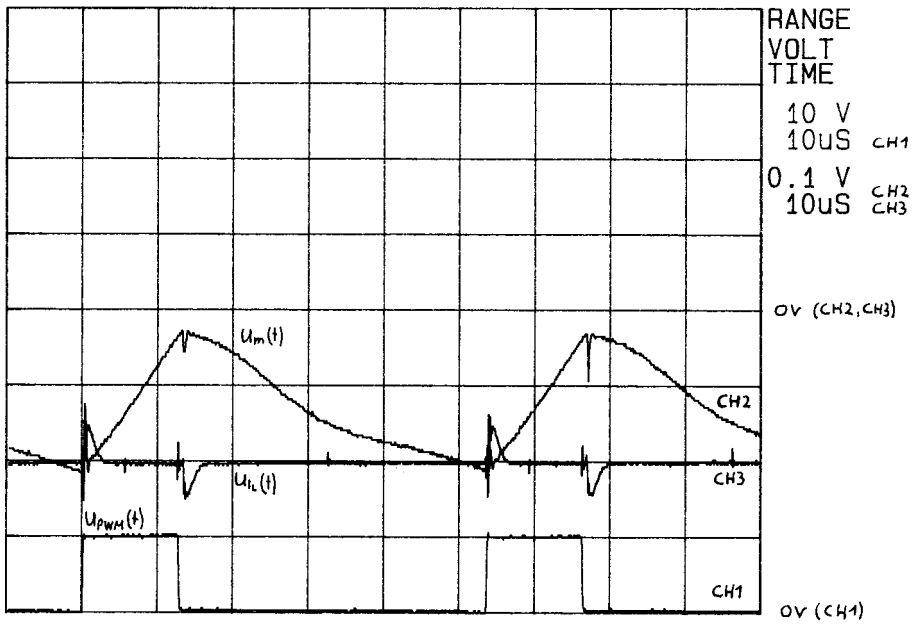


Abb. 2.3.12: Messung 3

Fazit der Messungen 1 bis 3: Bis auf kleine Unebenheiten liefert die Schaltung wie gewünscht eine fast saubere Gleichspannung, die dem Mittelwert des Brückenstromes entspricht. Die kleinen Unebenheiten im Bereich zwischen 40 und 60 mV können vernachlässigt werden.

2.3.3 Überstromschutz

Laut Aufgabenstellung soll bei der Realisierung des Stellgliedes auch eine Strombegrenzung realisiert werden. Auf grund der vorhandenen Hardware liegt es nahe, eine Überstromschutzschaltung zu realisieren, welche bei zu hohen Brückenströmen das Brake-Eingangssignal der H-Brücke **LMD18200** nutzt.

Strommessung

Diese Aufgabe wird von dem schon im Kapitel 2.3.2 erwähnten **LEM LA 25-NP** übernommen. Somit steht uns die zum Brückenstrom direkt proportionale Signalgrösse $U_m(t)$ zu Verfügung.

Betragsbildung

Da die Spannung $U_m(t)$ positiv oder negativ sein kann, eignet sie sich nicht, um mit einem Schwellwert verglichen zu werden. Es muss der Betrag $Abs(U_m(t))$ dieser Spannung gebildet werden. Dieser kann anschliessend mit einer einstellbaren Schwellenspannung verglichen werden, um so zu erfahren, ob der Brückenstrom zu gross ist oder nicht.
Die Betragsbildung wird mit einer Präzisionsgleichrichter-Schaltung realisiert *(Abb. 2.3.13)* .

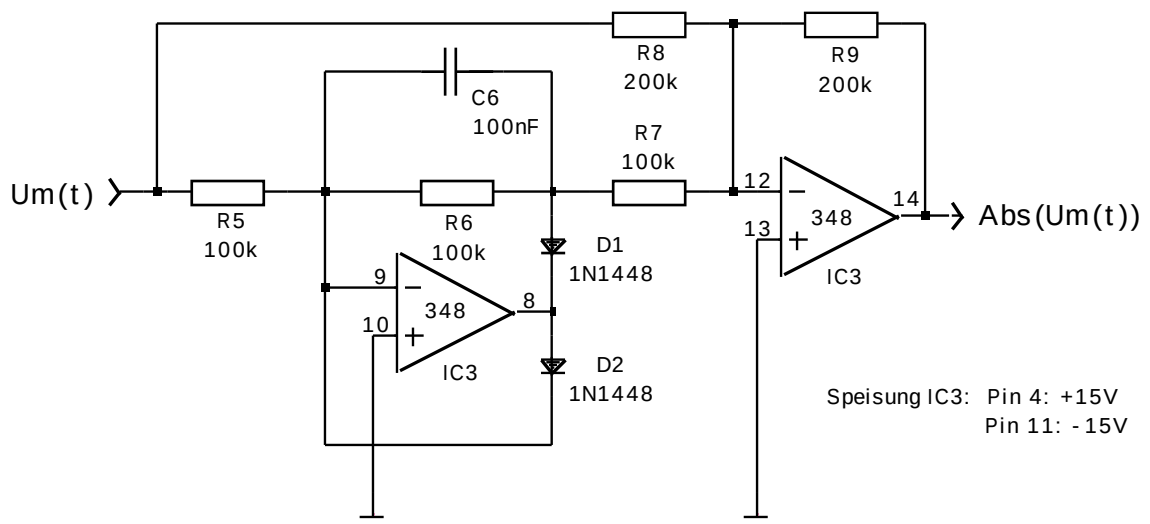


Abb. 2.3.13: Prinzipschema der Betragsbildungsschaltung

Wenn $U_m(t)$ **positiv** ist, dann wirkt der erste OP-Amp als invertierender Verstärker. Seine Ausgangsspannung beträgt $-U_m$. Der zweite OP-Amp wirkt als invertierende Addierstufe. Sie wichtet die Spannung $-U_m$ am Ausgang des ersten OP-Amp mit dem Faktor -2. Und sie wichtet die Eingangsspannung U_m mit dem Faktor -1. Diese beiden gewichteten Spannungen werden addiert und bilden $Abs(U_m)$. So hat die Spannung $Abs(U_m)$ den selben Wert wie U_m.
Wenn $U_m(t)$ hingegen **negativ** ist, so hat der erste OP-Amp 0V als Ausgangsspannung. Der zweite OP-Amp wirkt daher nur als invertierender Verstärker. So hat die Spannung $Abs(U_m)$ den selben Wert wie $-U_m$.

Messungen zur Betragsbildungs-Schaltung:

Anhand von zwei Messungen *Abb. 2.3.14* soll nun gezeigt werden, dass diese Schaltung auch in der Praxis funktioniert. Bei Messung 4 ist $U_m(t)$ immer positiv, und bei Messung 5 ist die Spannung immer negativ. Am KO-Kanal CH1 liegt jeweils die Spannung $U_m(t)$ und an CH2 die Spannung $Abs(U_m(t))$. Es ist zu erkennen, dass die Schaltung problemlos funktioniert.

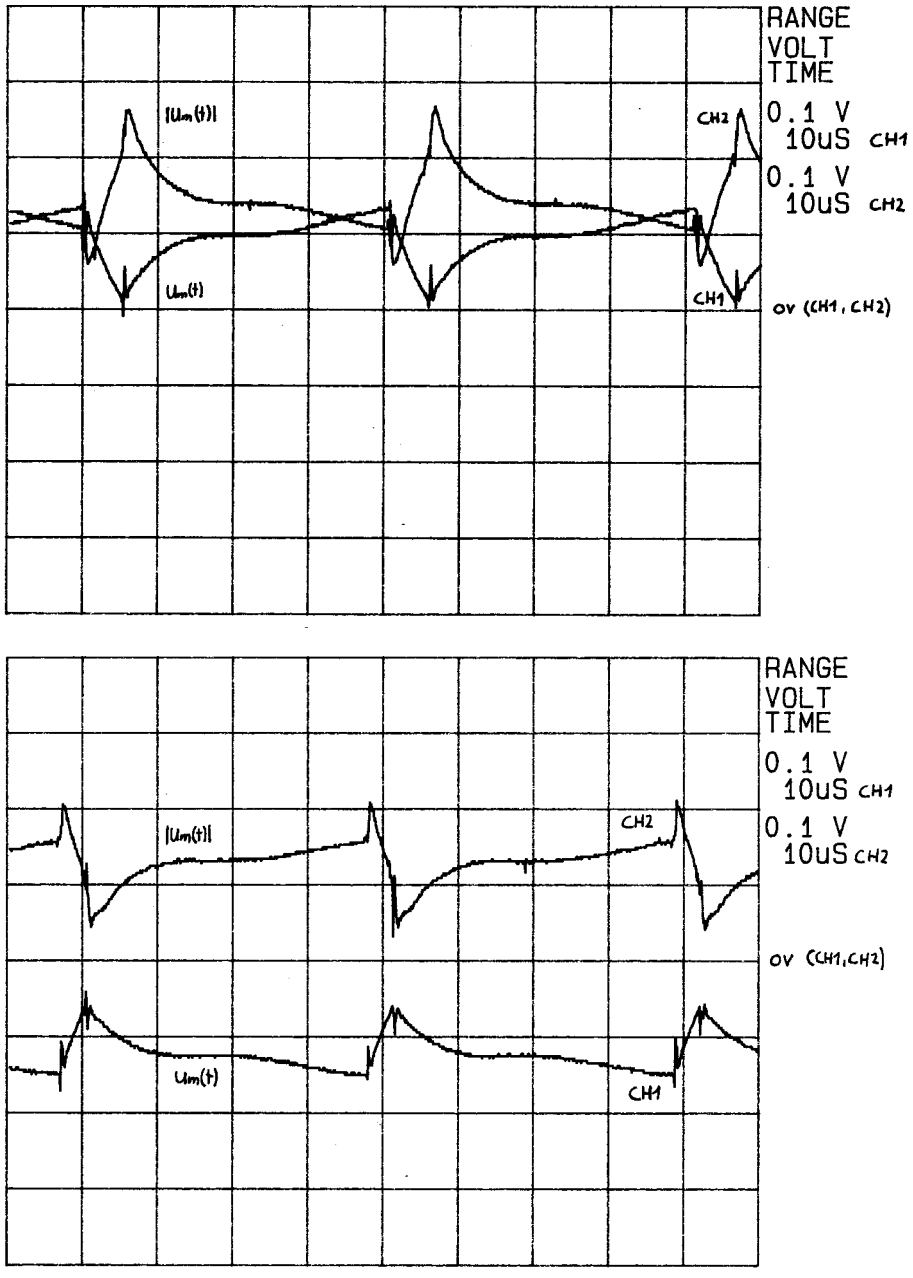


Abb. 2.3.14: Messung 4 und 5

Schwellwert-Schalter

Die mit der Betragsbildungs-Schaltung *(Abb. 2.3.13)* erzeugte Spannung Abs(U_m(t)) gibt die betragsmässige Grösse des Brückenstromes an. Es soll jetzt ein Schwellwertschalter *(Abb 2.3.15)* realisiert werden, welcher mit einem digitalen Signal angibt, wann der Brückenstrom zu gross ist.

Der maximal zulässige Brückenstrom des **LMD18200** beträgt 3A. Um zu verhindern, dass die H-Brücke an dieser oberen Grenze betrieben wird, soll der Schwellwert-Schalter bei Strömen grösser als 2.5A ansprechen.

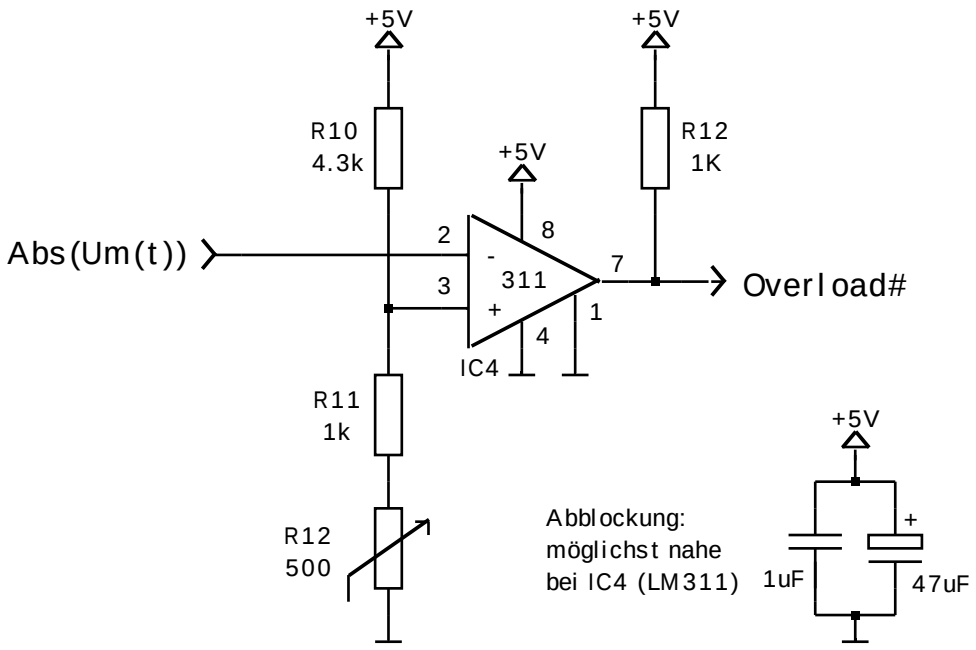


Abb. 2.3.15: Prinzipschema des Schwellwert-Schalters

Die Betragsbildungs-Schaltung liefert bei I_n = 2.5A eine Spannung Abs(U_m) von 1.25V. Das heisst, dass das Potentiometer R_{12} solange verstellt werden muss, bis die Schwellen-Spannung am nichtinvertierenden Eingang des Komparators **LM311** 1.25V beträgt.

Der Schwellwertschalter ist dann so eingestellt, dass das Signal Overload# bei zu hohen Strömen einen LOW-Pegel führt. Andernfalls führt es einen HIGH-Pegel.

Da auch bei dieser Schaltung schnelle Änderungen stattfinden können ist die Speisespannung unbedingt abzublocken.

Messung: Die H-Brücke wird (zusätzlich zum Gleichstrommotor) mit einer verstellbaren ohmschen Last beschalten, sodass Spitzenströme von über 2.5A auftreten können.
Am KO-Kanal CH1 liegt die Spannung Abs(U_m(t)), am Kanal CH3 die Schwellenpannung von 1.25V und am Kanal CH2 liegt Ausgangssignal Overlad# an. Das Resultat der Messung 6 ist in *Abb. 2.3.16* abgebildet.

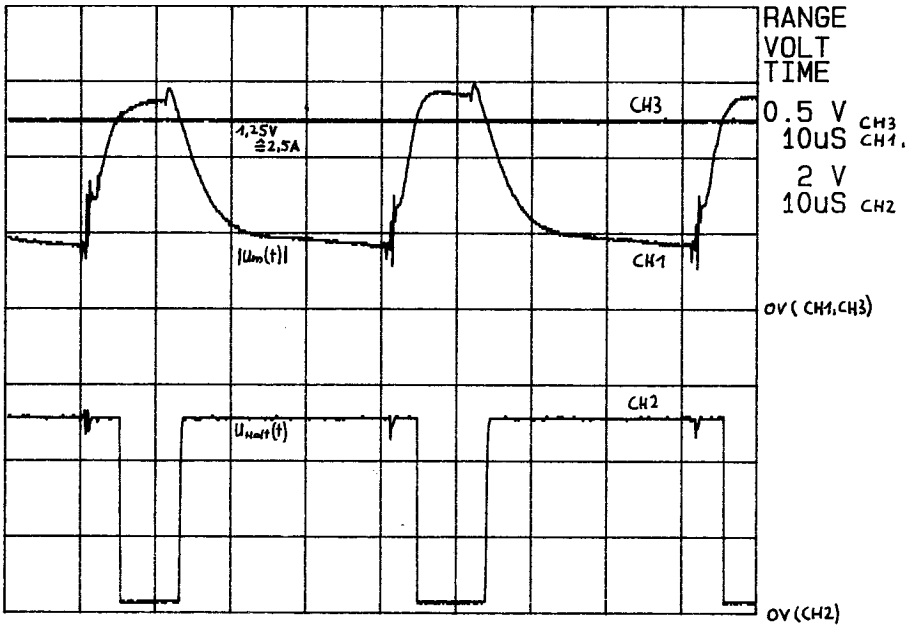


Abb. 2.3.16: Messung 6

Deutlich ist zu erkennen, dass das Signal Overload# (in *Abb. 2.3.16* mit $U_{Halt}(t)$ bezeichnet) immer dann LOW-aktiv wird, wenn der Pegel von Abs($U_m(t)$) die Schwellenspannung von 1.25V überschreitet.

Brücken-Abschaltung, Bedienung durch den Kontroller

Die Aufgabe der Brücken-Abschaltung besteht darin, bei Auftreten eines zu grossen Brückenstromes den Betrieb der Brücke augenblicklich einzustellen. Durch einen Reset-Taster soll der Betrieb wieder aufgenommen werden können.
Zusätzlich soll ein Signal generiert werden, welches den Betriebszustand der Brücke anzeigt und vom Kontroller abgefragt werden kann. Dazu soll der Konroller oder eine andere externe Baugruppe die Möglichkeit haben, die Brücken-Abschaltung rückzusetzen. *(Abb.2.3.17)* zeigt die Realisierung dieser Aufgabe.

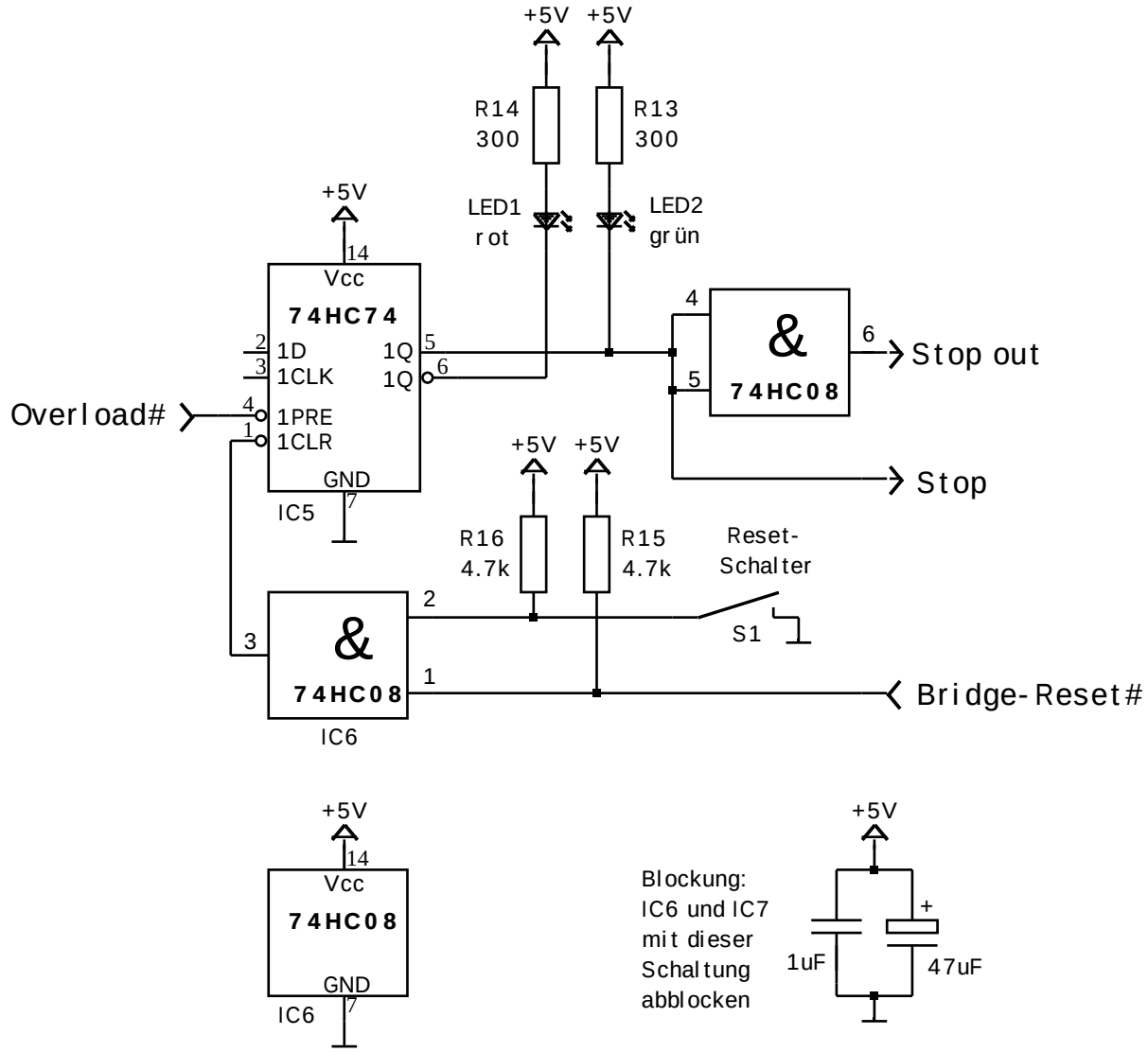


Abb. 2.3.17: Prinzipschema der Brückenabschaltung

Sobald das Signal Overload# auf LOW wechselt, kippt Das Flip-Flop **74HC74** und setzt den Ausgang Stop auf HIGH. Dieser Ausgang ist mit dem Brake-Eingang der H-Brücke **LMD18200** verbunden. Somit wird der Betrieb der Brücke gestoppt und die rote LED leuchtet. Der Ausgang Stop out stellt das getriebenen Stop-Signal dar und kann von anderen Applikationen (wie zum Beispiel dem Kontroller) genutzt werden.
Mit dem Reset-Taster oder mit dem LOW-aktiven Signal Bridge-Reset# kann nun das Flip-Flop zurückgesetzt werden und die grüne LED leuchtet.

2.3.4 Überspannungs-Schutz

Bei einer Belastung des Stellgliedes mit induktiven Lasten können bei raschen Stromänderungen unerwünschte Überspannungen auftreten. Wenn zum Beispiel die stromdurchflossene Last vom Stellglied getrennt wird, können Spannungsspitzen von mehreren kV auftreten. Das kann zu einer Zerstörung der H-Brücke führen. Deshalb soll die Brücke vor Überspannungen geschützt werden.

Hier bietet sich ein Spannungsabhängiger Widerstand (VDR) an. Es muss ein VDR ausgesucht werden, der folgende Bedingungen erfüllt: Er muss bei Betriebsspannung so hochohmig wie möglich sein, damit er das Stellglied nicht unnötig belastet. Andererseits muss er bei Überspannungen, die durch induktive Lasten entstehen können, sehr niederohmig sein, um das Stellglied zu schützen.

Anhand einer V/R-Kennlinie *(Abb. 2.3.18)* soll nun die Charakteristik eines VDR veranschaulicht werden. In diesem Beispiel handelt es sich um einen Metalloxid-Varistor des Typs **SIOV-B60K250** der Firma Siemens-Matsushita Components.

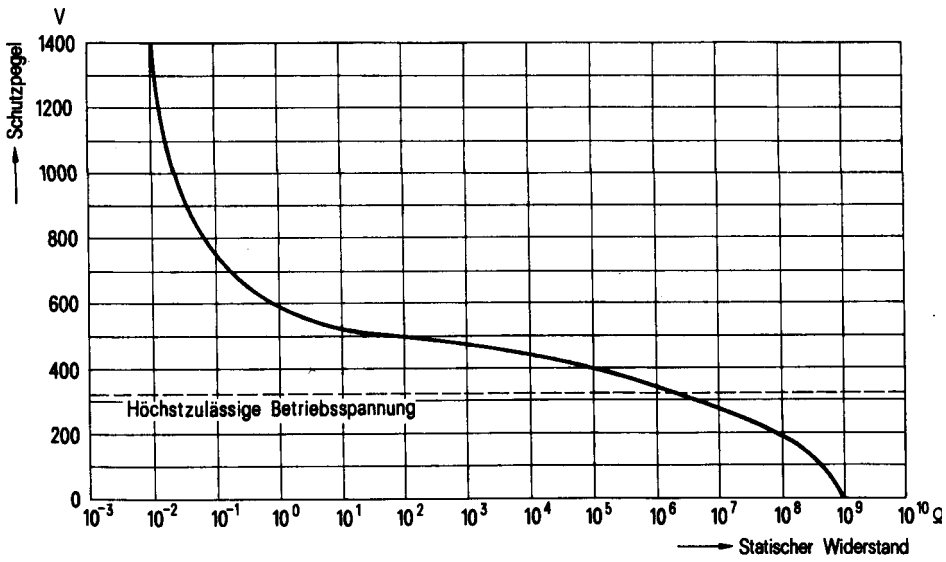


*Abb. 2.3.18: V/R-Kennlinie des VDR **SIOV-B60K250** (Quelle: [13])*

Die H-Brücke **LMD18200** ist für Spannungen bis 55V ausgelegt. Um den Baustein nicht bis an diese Grenze betreiben zu müssen, wird der Schutzpegel auf etwa 40 bis 45V festgelegt. Die Betriebsspannung der Brücke beträgt 24V.

Anhand dieser Angaben haben wir den Varistoren **SIOV-S05K14** von Siemens-Matsushita ausgewählt. Laut V/I-Kennlinen-Feld in [13] fliesst bei der Betriebsspannung 24V ein Strom von etwa 1mA. Das heisst, dass der VDR bei Betriebspannung einen statischen Widerstand von etwa 24kΩ aufweist. Bei Schutzpegel 43V fliessen hingegen schon etwa 3A, was einen statischen Widerstand von 14Ω bedeutet. Bei Spannungen über 43V steigt der Strom sehr rasch auf mehrere 100A.

[13]: Siemens-Matsushita Components, SIOV Metalloxid-Varistoren, Seiten 17 und 136

2.3.5 Technische Spezifikationen, Fertigungsunterlagen

Stromversorgung

Der Stellglied-Print muss mit den Spannungen +15V, -15V, +5V, -5V versorgt werden. Die Karte wird nicht über den Europastecker, sondern über Klemmleisten mit den Speisespannungen versorgt.

Kartenstecker

Die digitalen Steuersignale des Stellgliedes (PWM_OUT, Stop out, BridgeReset und ThermFlag) sind am Europa-Kartenstecker abgreifbar. Die Pin-Nummern sind dem Gesamtschema des Stellgliedes *(Abb. 2.3.19)* zu entnehmen.

Gesamtschema

Das Gesamtschema der A/D-Wandlerkarte ist in *Abb. 2.3.19* abgebildet.

Layout, Bestückungszeichnung

Leider hat die Zeit nicht mehr gereicht, vor Abgabe dieser Arbeit den Stellglied-Print anzufertigen. Daher sind auch noch keine Unterlagen wie Layout und Bestückungszeichnung vorhanden.

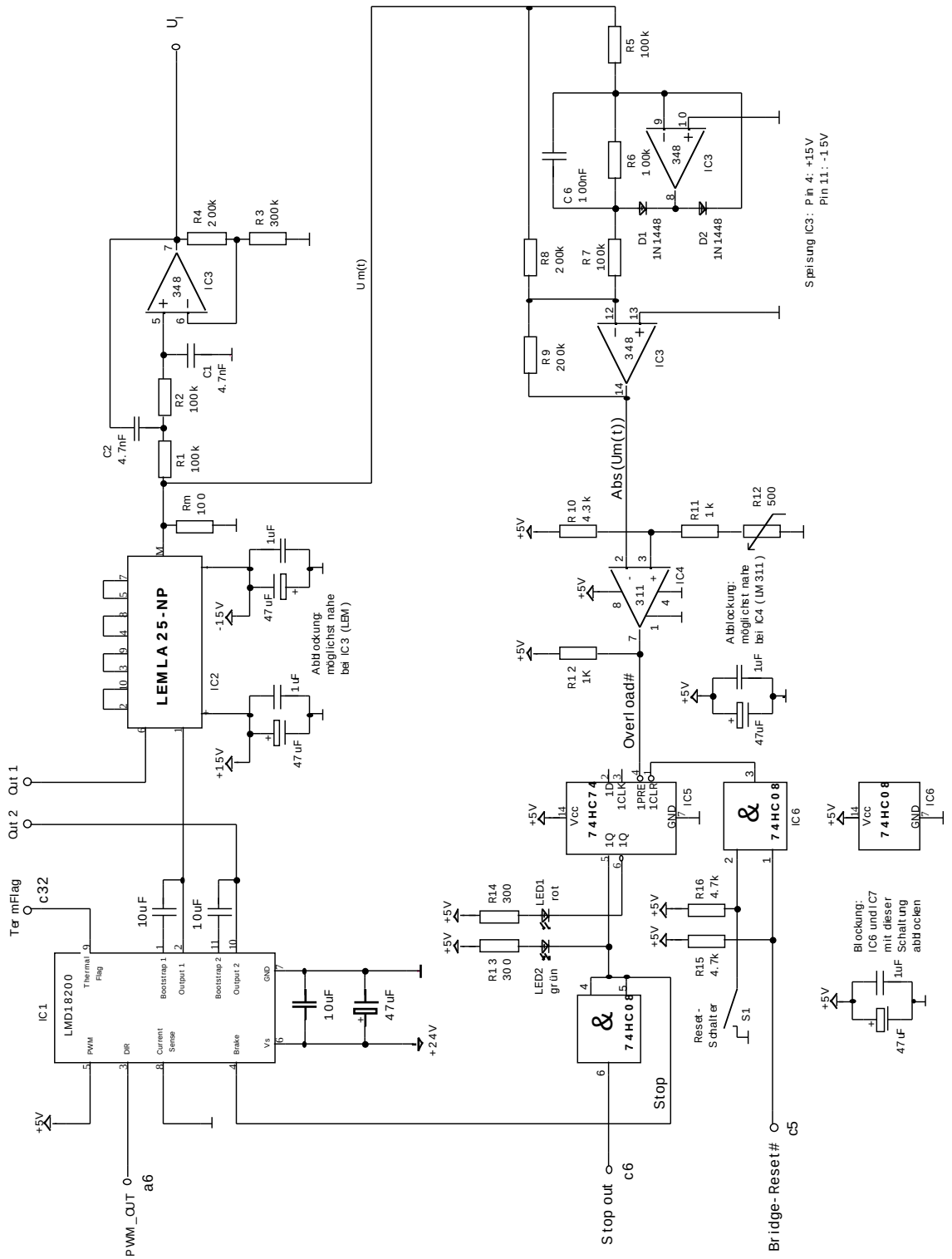


Abb. 2.3.19 : Gesamtschema Stellglied

3. Software

3.1 Betriebssystem für die A/D-Wandlerkarte

Zum Betreiben der A/D-Wandlerkarte ist ein Betriebssystem entwickelt worden, das auf dem Programm ADSYS beruht. Dieses Betriebssystem ist Inhalt der Projektarbeit [10]. Um eine Arbeit mit der vorliegenden Hardware und eventueller Erweiterungen zu ermöglichen, wurden einige Änderungen vorgenommen.

In den zum Betriebssystem (siehe Anhang A) gehörenden Deklarationsdateien (Anhang C und D) wurden einige nichtbenutzte Definitionen entfernt. Ausserdem wurden neue zur Anwendung notwendige Vereinbarungen aufgenommen (z.B. Gleitpunktzahlen).

In der Adressdeklarationsdatei (Anhang B) sind Adressen für insgesamt 8 A/D-Wandlerkanäle enthalten. Diese befinden sich im Bereich von 20002h bis 20010h. Die Adresse 20000h dient dem Auslösen der Konversion (siehe Abschnitt 2.2.4 A/D-Wandler). Ebenfalls sind die Adressen (CS-Mode-Register und Basisadressregister) zur Bearbeitung der Chip-Select-Logik CS3 und CS6 vereinbart.

Die einzelnen Routinen des Betriebssystems wurden beibehalten. In der Funktion "StarteUebertragung" (siehe *Abb. 3.1.1*) wurde ein PWM-Signal mit dem Tastverhältnis 1:1 generiert, das mit einer Frequenz von 1200 Hz arbeitet.

```
void StarteUebertragung (void)
/*********************************
 Startet die Uebertragung.
 Diese Prozedure darf erst nach einem
 Aufruf von DefiniereUebertragung
 aufgerufen werden.
*********************************/
{
  DisableInterrupts;

  TPU_Grundeinstellung (1);           /* T=250 ns */
  TPU_InterruptEinstellungen (240,3); /* Grundint.-Nr.240 und Prioritität 3 */
  TPU_InterruptFreigeben (0);         /* Interrupt des Kanals 0 zulassen  */
  TPU_PWM (0, 3, 3333, 1667);         /* Welle mit f=1200 Hz für Int.-routine */
  TPU_EinfacherAusgang (3, 2);        /* an Kanal 3 wird angezeigt, ob eine
                                       Interrupt-Routine abgearbeitet wird. */

  *INT240  = (CARD32) &IntProc;       /* Interrupt-Vektor für Interrupt 240
                                         setzen */
  *BaseAddrCS3 = 0x0000;             /* CS3 desaktivieren, sonst immer aktiv!*/

  *ModeCS3     = 0x0000;

  *BaseAddrCS6 = 0x0200;              /* CS6 initialisieren */
  *ModeCS6     = 0x7B73;              /* mit 13 Waitstates fnr DSACK1 */

  EnableInterrupts;
}
```

Abb. 3.1.1: StarteUebertragung

[10]: TWI; Joachim Oechslin/Daniel Gahlinger, Projektarbeit I/93, Digitale Regelung mit dem Mototola MC68332

Es dient als Grundlage für die Abtastung des analogen Signals am Eingang des A/D-Wandlers. Mit Hilfe dieser Frequenz wird ein Interrupt an Kanal 0 ausgelösst. Danach wird mit Hilfe der Grundinterruptnummer in die Interruptroutine "IntProc" gesprungen.

In der Interruptroutine wird nun über die Adresse "*AD_WANDLER_INIT" die Konversion gestartet und anschliessend mit der Programmzeile:

```
Wert = *AD1_WANDLER;
```

der gewandelte Wert vom **MAX120** über den Datenbus der Variablen "Wert" zugewiesen. Dies erfordert einen hohen Hardwareaufwand, der in Abschnitt 2.2.4 beschrieben ist.

Bei einer Erweiterung der Hardware auf bis zu 8 Kanäle müssen nur noch im Betriebssystem die entsprechenden Werte, die über die Adressen "*AD2_WANDLER" bis "*AD8_WANDLER" eingelesen werden, einer beliebigen Variablen zugewiesen werden.

Danach erfolgen Wertebereichprüfungen und die Zuweisung des kontrollierten Wertes an die Variable "AbtastWert", die dann vom Hauptprogramm "HWTEST" verarbeitet werden kann.

Zur Unterstützung der Erprobungstätigkeit wird bei jedem Beginn der Interruptroutine der TPU-Ausgang Kanal 3 auf High gesetzt und bei jedem Verlassen wieder zurückgesetzt. Damit kann das Interruptverhalten des Kontrollers leicht überwacht werden.

Danach folgen im Beriebssystem weitere Routinen, die zur Initialisierung des Mikrokontrollers **MC68332** notwendig sind. Hierbei wird die Elementarzeit und die Interrupteinstellung programmiert. Anschliessend wurden Funktionen integriert, die die Nutzung von Zeitfunktionen der TPU durch den Anwender im Hauptprogramm erleichtern. Dies sind z.B. ein PWM-Signal oder ein einfacher Ausgang. In diesen Funktionen sind jeweils alle zu setzenden Register enthalten, sodass nur noch die Angabe des gewünschten Kanals und der Prioritätsebene erforderlich ist.

3.2 Hauptprogramm zum Testen der Hardware

Um die realisierte Hardware zu erproben, wurde ein Testprogramm geschrieben. Es ermöglicht, die Funktion der A/D-Wandlerkarte und die nachfolgende Ansteuerung des Stellgliedes zu generieren.

Die Funktion des Programms besteht in der Berechnung eines PWM-Signals, das gleichwertig zum vorher eingelesenen analogen Wert am A/D-Wandler ist:

+5V	Tastverhältnis	100%
:	:	
0V	Tastverhältnis	50%
:	:	
-5V	Tastverhältnis	0%

Dabei wird der abgetaste Wert eingelesen und eingangs geprüft, ob es ein negativer oder positiver Wert ist. Danach erfolgt eine Division um das Tastverhältnis zu berechnen. Anschliessend wird ein PWM-Signal mit der Grundfrequenz von 20 KHz (nicht hörbarer Bereich) und dem vorher berechneten Tastverhältnis ausgegeben. Der genaue Programmablauf ist im Struktogramm *Abb. 3.2.1* aufgeführt.

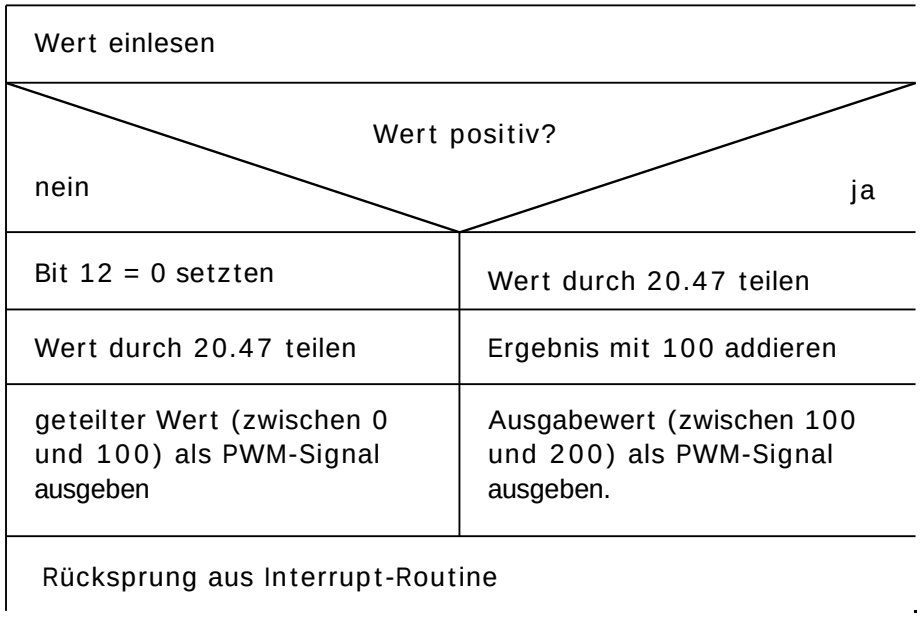


Abb. 2.3.1: Struktogramm Testprogramm

Für die weitere Arbeit mit dem Hardwaresystem, sind damit die gesamten Softwarevoraussetzungen geschaffen, sodass nur noch die Anpassung des Betriebssystems an die Anzahl der Kanäle zu erfolgen hat. Der jeweils zu programmierende Regelalgorithmus (das vorliegende Testprogramm beinhaltet einen P-Regler) ist dann nur noch in der "Wandelroutine" des Testprogramms zu editieren, zu compilieren und mit dem Betriebssystem (einschliesslich der Deklarationsdateien) zu linken. Dazu ist eine Linkdatei *.prm nötig, deren Aufbau an der Datei des Testprogramms "HWTEST.prm" in Anhang F ersichtlich ist.

Zur Information des Programmierers wird vom Linker eine Datei *.map erzeugt, die die verwendeten Programme, Routinen und Variablen darstellt. In Anhang G ist die Informationsdatei "HWTEST.map" enthalten.

Bei eventuell auftretenden Fragen zur Software und deren genaue Funktionsabläufe sei hiermit auf die Projektarbeit [10] hingewiesen.

[10]: TWI; Joachim Oechslin/Daniel Gahlinger, Projektarbeit I/93, Digitale Regelung mit dem Mototola MC68332

4. Schlusswort

Erst einmal möchten wir uns bei Herrn Prof. Dr. Golder und Herrn Kolb ganz herzlich für ihre unterstützende Mitarbeit bedanken.

Die Arbeit am Motorola-Kontroller MC68332 hat uns viele neue Einblicke in die Welt der Mikrokontroller gegeben. Im Vergleich zur Intel-Kontroller-Familie 8051 konnten interessante Unterschiede festgestellt werden.
Die grössten Probleme bereiteten uns die komplizierten und zum Teil sogar fehlerbehafteten Hard- und Software-Manuals.
Trotzdem gelang es uns, wenn auch mit grossem Aufwand, Hard- und Software auf einen gemeinsamen Nenner zu bringen. Das Resultat war ein einwandfrei funktionierender A/D-Wandlerkanal mit PWM-Ausgabe.

Die Konzeptionierung des geschalteten Stellgliedes war sehr interessant und lehrreich. Wir haben uns durch diese Arbeit fundiertes Wissen über H-Brücken und LEM aneignen können. Die Realisierung der Schutzmassnahmen gegen Überspannung und Überstrom brachte uns sehr viel schaltungstechnisches Know-how. Auch bei dieser Arbeit konnten wir den Erfolg einer funktionierenden Baugruppe verzeichnen.

Da uns das Fehlersuchen an Hard- und Software sehr viel Zeit kostete, war es uns leider nicht mehr möglich, komplette Fertigungsunterlagen zu den einzelnen Baugruppen zu schaffen. Auch die Herstellung der fertigen Leiterplatten musste deswegen auf die Zeit nach der Abgabe dieser Arbeit verschoben werden.

8400 Winterthur, 6. Dezember 1993

Oliver Gossel, Roger Meier

Literaturverzeichnis

[1]: Tietze-Schenk, Halbleiterschaltungstechnik, 9. Auflage
[2]: National Semiconductor, Power IC's Databook, 1993 Edition
[3]: National Semiconductor, Programmable Logic Devices Databook and Design Guide
[4]: Motorola, MC68332 User's Manual
[5]: Motorola, M68332EVK Evaluation Kit User's Manual
[6]: TWI; Prof. J. Zeman, MC-Stoff; Kapitel PLD's
[7]: MAXIM, 1993 New Releases Databook Volume II
[8]: MAXIM, Data Pack Nr. 7; AD-Converters Design Guide
[9]: TWI; Andreas Graf/Roger Meier, Projektarbeit I/93, Geschaltete Konzepte
[10]: TWI; Joachim Oechslin/Daniel Gahlinger, Projektarbeit I/93, Digitale Regelung mit ...
[11]: Motorola, TPU Reference Manual
[12]: HIWARE, User Manual HICROSS V2.5
[13]: Siemens-Matsushita Components, SIOV Metalloxid-Varistoren
[14]: MAXIM, 4th and 8th-Order Continuous-Time Active Filters

Anhang A: Betriebssystem für die A/D-Wandlerkarte

```
/*********************************
 Betriebssystem für die
 AD-Karte mit 1 Kanal,
 vorbereitet für 8 Kanäle !

 rogoli; 18.11. 1993
*********************************/

 #include "\source\types.h"
 #include "\source\const.h"
 #include "\source\rogoli.h"
 #include "hidef.h"


/*********************************
 External-Deklarationen
*********************************/
 extern INT16 AbtastWert[1];
 extern INT16 AusgangsWert[1];
 extern INT16   Anz_Werte;

/*********************************
 globale Variablen
*********************************/
 TUEBERTRAGUNG * Uebertragung;


void DefiniereUebertragung (TUEBERTRAGUNG * p)
/*********************************************
 Definieren der Prozedur, welche die
 Übertragungsfunktion (oder auch
 Rekursionsgl.) beinhaltet.
*********************************************/
{
  Uebertragung = p;
}


void LoescheVorgeschichte (void)
/*********************************
 L÷scht die Felder
 AbtastWert und AusgabeWert
*********************************/
{
  INT16  i;

  for (i=0; i<Anz_Werte; i++)         /* Eingangs- und Ausgangs-Array löschen */
   {
    AbtastWert[i] = 0;
    AusgangsWert[i] = 0;
   }
}

void StarteUebertragung (void)
/*********************************
 Startet die Uebertragung.
 Diese Prozedure darf erst nach einem
 Aufruf von DefiniereUebertragung
 aufgerufen werden.
*********************************/
{
  DisableInterrupts;

  TPU_Grundeinstellung (1);           /* T=250 ns */
  TPU_InterruptEinstellungen (240,3); /* Grundint.-Nr.240 und Prioritität 3 */
  TPU_InterruptFreigeben (0);         /* Interrupt des Kanals 0 zulassen  */
```

```
  TPU_PWM (0, 3, 3333, 1667);         /* Welle mit f=1200 Hz für Int.-routine */
  TPU_EinfacherAusgang (3, 2);        /* an Kanal 3 wird angezeigt, ob eine
                                       Interrupt-Routine abgearbeitet wird. */

  *INT240  = (CARD32) &IntProc;       /* Interrupt-Vektor für Interrupt 240
                                         setzen */
  *BaseAddrCS3 = 0x0000;             /* CS3 desaktivieren, sonst immer aktiv!*/

  *ModeCS3     = 0x0000;

  *BaseAddrCS6 = 0x0200;              /* CS6 initialisieren */
  *ModeCS6     = 0x7B73;              /* mit 13 Waitstates fnr DSACK1 */

  EnableInterrupts;
}


#pragma TRAP_PROC
void IntProc (void)
/*********************************
 Interrupt-Prozedur
*********************************/
{
  CARD16   Status;
  INT16    i;
  CARD16   Wert;

  Status = *CISR;
  *CISR = 0;

 TPU_AusgangHigh (3);

  *AD_WANDLER_INIT = 0;                    /*Konversion starten*/

  Wert = *AD1_WANDLER;                     /*AD-Wert einlesen*/

  i=Anz_Werte-1;
  do
   {
    i--;
    AbtastWert[i+1] = AbtastWert[i];           /*Anzahl Stuetzwerte*/
    AusgangsWert[i+1] = AusgangsWert[i];
   }
   while (i>0);

  AbtastWert[0] = (INT16)(Wert & 0x0FFF);      /*Abfrage ob Wertebereich */
  if (AbtastWert[0] & 0x0800)                  /*überschritten ist*/
    AbtastWert[0] = AbtastWert[0] | 0xF000;

  (*Uebertragung) ();

  TPU_AusgangLow (3);
}

int TPU_InterruptEinstellungen (int GrundIntNr, int Prior)
/*******************************************************
/ Initialisiert die Interrupt-Logik.
/
/ GrundIntNr:    Grund-Interrupt-Nummer
/                (0, 16, 32, .. , 240)
/ Prior:         Int.-Priorität der TPU
/                (0..7)
/
/ Return-Wert:   1: Fehler
/                0: Ok.
*******************************************************/
```

```
{
  CARD16   w;

  if ((GrundIntNr<0)||(GrundIntNr>255)) return 1;
  if ((Prior<0)||(Prior>7)) return 1;
  *TICR=0;
  w=GrundIntNr & 0x00f0;
  *TICR |= w;
  w=Prior * 256;
  *TICR |= w;

  return 0;
}


int TPU_InterruptFreigeben (int Kanal)
/*****************************************
/ Lässt die Interruptanforderung des gewählten
/ Kanals an die CPU
/
/ Kanal:         Kanal-Nummer (0..15)
/
/ Return-Wert:   1: Fehler
/                0: Ok.
*****************************************/
{
  CARD16   w;
  int      i;

  if ((Kanal<0)||(Kanal>15)) return 1;
  w=1;
  for (i=0; i<Kanal; i++)
    w*=2;
  *CIER |= w;
  return 0;
}


int TPU_InterruptSperren (int Kanal)
/*****************************************
/ Sperrt die Interruptanforderung des gewählten
/ Kanals
/
/ Kanal:         Kanal-Nummer
/
/ Return-Wert:   1: Fehler
/                0: Ok.
*****************************************/
{
  CARD16   w;
  int      i;

  if ((Kanal<0)||(Kanal>15)) return 1;
  w=1;
  for (i=0; i<Kanal; i++)
    w*=2;
  w=0xffff-w;
  *CIER &= w;
  return 0;
}
```

```
 int TPU_Grundeinstellung (int ZeitCode)
/*********************************************
/ Initialisierung der TPU.
/
/ ACHTUNG: Diese Prozedur sollte nur einmal
/ aufgerufen werden, jeder weitere Aufruf ist
/ sinnlos!!
/
/ Der Teiler fnr den externen Eingang (TCR2)
/ wird 1 gesetzt.
/ Die Grundzeit (oder Elementarzeit) fnr TCR1
/ kann mit dem Parameter ZeitCode folgendermassen
/ angegeben werden:
/
/ ZeitCode  | Elementarzeit
/ ----------|---------------
/ 1         | 250 ns
/ 2         | 500 ns
/ 3         |   1 us
/ 4         |   2 us
/ 5         |   4 us
/ 6         |   8 us
/ 7         |  16 us
/
/ Die Interrupt Arbitration ID ist immer 3.
/
/ Return-Wert:   1: Fehler
/                0: Ok.
*********************************************/
{

if ((ZeitCode<1)||(ZeitCode>7)) return 1;
switch (ZeitCode)
 {
  case 1: *TMCR = 0x0043; break;
  case 2: *TMCR = 0x2043; break;
  case 3: *TMCR = 0x4043; break;
  case 4: *TMCR = 0x0003; break;
  case 5: *TMCR = 0x2003; break;
  case 6: *TMCR = 0x4003; break;
  case 7: *TMCR = 0x6003; break;
 }
return 0;
}


int TPU_PWM (int Kanal, int Prior, CARD16 Per, CARD16 High)
/*********************************************************
/ Initialisierung eines Pulsbreitenmodulierten
/ Ausgangs
/ Kanal:         Kanal-Nummer (0..15)
/ Prior:         Priorität des Kanals (0..3)
/ Per:           Periode (0..0x8000)
/ High:          Hoch-Zeit (0..Per)
/
/ Return-Wert:   1: Fehler
/                0: Ok.
*********************************************************/
{
int    n, r, i;
CARD16 w1, w2;

CARD16 * CFSF   = (CARD16*) 0xfffe0c;
CARD16 * CPF    = (CARD16*) 0xfffe1c;
CARD16 * HSRF   = (CARD16*) 0xfffe18;
CARD16 * CCR    = (CARD16*) (0xffff00 + Kanal*0x0010);
```

```
CARD16 * PWMHI  = (CARD16*) (0xffff04 + Kanal*0x0010);
CARD16 * PWMPER = (CARD16*) (0xffff06 + Kanal*0x0010);

if ((Kanal<0)||(Kanal>15)) return 1;
if ((Prior<0)||(Prior>3)) return 1;
if (Per>0x8000) return 1;
if (High>Per) return 1;

/**********************************
 Setzen des channel control register,
 der Periodendauer und der Hoch-Zeit.
**********************************/

 *CCR     = 0x0092;
 *PWMPER  = Per;
 *PWMHI   = High;

/*********************************
 Setzen des channel function select
 register
*********************************/

 n=Kanal/4;
 r=Kanal-n*4;
 w1= 0x000f;
 w2= 0x0009;
 for (i=0; i<r; i++)
  {
  w1 *= 0x10;
  w2 *= 0x10;
  }
 w1=65535-w1;
 CFSF[3-n] &= w1;
 CFSF[3-n] ^= w2;

/*********************************
 Setzen des channel priority
 register
*********************************/

 n=Kanal/8;
 r=Kanal-n*8;
 w1= 0x0003;
 w2= Prior;
 for (i=0; i<r; i++)
  {
   w1 *= 0x4;
   w2 *= 0x4;
  }
 w1=65535-w1;
 CPF[1-n] &= w1;
 CPF[1-n] ^= w2;

/*********************************
 Setzen des host service request
 register
*********************************/

 w1= 0x0003;
 w2= 0x2;
 for (i=0; i<r; i++)
  {
   w1 *= 0x4;
   w2 *= 0x4;
  }
 w1=65535-w1;
```

```
 HSRF[1-n] &= w1;
 HSRF[1-n] ^= w2;


 return 0;
}


int TPU_EinfacherAusgang (int Kanal, int Prior)
/*****************************************************
/ Initialisierung eines einfachen Ausgangs
/ welcher die Zustände 0 und 1 annehmen kann.
/ Mit dem Aufruf der Prozeduren TPU_AusgangHigh(Kanal)
/ und TPU_AusgangLow(Kanal) kann der Pin beeinflusst
/ werden (aber nur wenn der Kanal als einfacher Ausgang
/ deklariert wurde).
/
/ Return-Wert:   1: Fehler
/                0: Ok.
*****************************************************/
{
int    n, r, i;
CARD16 w1, w2;

CARD16 * CFSF   = (CARD16*) 0xfffe0c;
CARD16 * CPF    = (CARD16*) 0xfffe1c;
CARD16 * HSRF   = (CARD16*) 0xfffe18;
CARD16 * CCR    = (CARD16*) (0xffff00 + Kanal*0x0010);

if ((Kanal<0)||(Kanal>15)) return 1;
if ((Prior<0)||(Prior>3)) return 1;

/*********************************
 Setzen des channel control register
*********************************/

 *CCR     = 0x0113;

/*********************************
 Setzen des channel function select
 register
*********************************/

 n=Kanal/4;
 r=Kanal-n*4;
 w1= 0x000f;
 w2= 0x0008;
 for (i=0; i<r; i++)
  {
  w1 *= 0x10;
  w2 *= 0x10;
  }
 w1=65535-w1;
 CFSF[3-n] &= w1;
 CFSF[3-n] ^= w2;


/*********************************
 Setzen des channel priority
 register
*********************************/

 n=Kanal/8;
 r=Kanal-n*8;
 w1= 0x0003;
 w2= Prior;
```

```
 for (i=0; i<r; i++)
  {
  w1 *= 0x4;
  w2 *= 0x4;
  }
 w1=65535-w1;
 CPF[1-n] &= w1;
 CPF[1-n] ^= w2;

return 0;
}


void TPU_AusgangHigh (int Kanal)
/*********************************************
/ Setzt den Ausgang des angewählten Kanals auf 1
/ (aber nur wenn der Kanal als einfacher Ausgang
/ deklariert wurde).
*********************************************/
{
int    n, r, i;
CARD16 w1, w2;

CARD16 * HSRF   = (CARD16*) 0xfffe18;

/*********************************
 Setzen des host service request
 register
*********************************/

 n=Kanal/8;
 r=Kanal-n*8;
 w1= 0x0003;
 w2= 0x1;
 for (i=0; i<r; i++)
  {
   w1 *= 0x4;
   w2 *= 0x4;
  }
 w1=65535-w1;
 HSRF[1-n] &= w1;
 HSRF[1-n] ^= w2;
}


void TPU_AusgangLow (int Kanal)
/*********************************************
/ Setzt den Ausgang des angewählten Kanals auf 0
/ (aber nur wenn der Kanal als einfacher Ausgang
/ deklariert wurde).
*********************************************/
{
 int    n, r, i;
 CARD16 w1, w2;

 CARD16 * HSRF   = (CARD16*) 0xfffe18;

/*********************************
 Setzen des host service request
 register
*********************************/

 n=Kanal/8;
 r=Kanal-n*8;
 w1= 0x0003;
 w2= 0x2;
```

```
 for (i=0; i<r; i++)
  {
   w1 *= 0x4;
   w2 *= 0x4;
  }
 w1=65535-w1;
 HSRF[1-n] &= w1;
 HSRF[1-n] ^= w2;
}

/* End of File */
```

Anhang B: Adressdeklarationen CONST. h

```
/*********************************
 Adressdeklarationen
 ROGOLI , 17/11/93
*********************************/


CARD16 * TICR    = (CARD16*) 0xFFFE08;
CARD16 * TMCR    = (CARD16*) 0xFFFE00;
CARD16 * CIER    = (CARD16*) 0xFFFE0A;
CARD16 * CISR    = (CARD16*) 0xFFFE20;

CARD32 * INT240  = (CARD32*) 0x0003C0;

CARD16 * AD_WANDLER_INIT = (CARD16*) 0x020000;
CARD16 * AD1_WANDLER     = (CARD16*) 0x020002;
CARD16 * AD2_WANDLER     = (CARD16*) 0x020004;
CARD16 * AD3_WANDLER     = (CARD16*) 0x020006;
CARD16 * AD4_WANDLER     = (CARD16*) 0x020008;
CARD16 * AD5_WANDLER     = (CARD16*) 0x02000A;
CARD16 * AD6_WANDLER     = (CARD16*) 0x02000C;
CARD16 * AD7_WANDLER     = (CARD16*) 0x02000E;
CARD16 * AD8_WANDLER     = (CARD16*) 0x020010;

CARD16 * BaseAddrCS3 = (CARD16*)0xFFFA58;
CARD16 * ModeCS3     = (CARD16*)0xFFFA5A;
CARD16 * BaseAddrCS6 = (CARD16*)0xFFFA64;
CARD16 * ModeCS6     = (CARD16*)0xFFFA66;
```

Anhang C: Deklarationen ROGOLI. h

```
/*********************************
 Deklarationen

 Daniel Gahlinger
 Joachim Oechslin
 3. Mai 1993
 *********************************/

/*********************************
 extern-Deklarationen
 *********************************/
  extern TUEBERTRAGUNG * Uebertragung;

/*********************************
 Prototypen-Deklarationen
 *********************************/
  void IntProc (void);
  void DefiniereUebertragung (TUEBERTRAGUNG * p);
  void LoescheVorgeschichte (void);
  void StarteUebertragung (void);

  int TPU_InterruptEinstellungen (int GrundIntNummer, int Prior);
  int TPU_InterruptFreigeben (int Kanal);
  int TPU_InterruptSperren (int Kanal);
  int TPU_Grundeinstellung (int ZeitCode);
  int TPU_PWM (int Kanal, int Prior, CARD16 Per, CARD16 High);
  int TPU_EinfacherAusgang (int Kanal, int Prior);
  void TPU_AusgangHigh (int Kanal);
  void TPU_AusgangLow (int Kanal);
```

Anhang D: Typendeklarationen TYPES. h

```
/*********************************
 Typendeklarationen
 rogoli, 18.11.93
*********************************/


#define CARD8     unsigned short int
#define CARD16    unsigned int
#define CARD32    unsigned long int
#define INT8      short int
#define INT16     int
#define INT32     long int
#define GLEIT     float


/*********************************
 Zeiger auf einen Prototyp einer
 benutzerdefinierten Funktion
 (einer Übertragungsfunktion)
*********************************/
typedef void TUEBERTRAGUNG (void);


/*********************************
 Abtastzeit in s
*********************************/
#define  T          0.0002
```

Anhang E: Hauptprogramm HWTEST. c

```
/************************************************************
Programm zum Einlesen einer Spannung über einen AD-Kanal
auf Adresse 20002 mit nachfolgender Ausgabe eines zugehörigen
PWM-Signal's über den TPU-Kanal 1

rogoli, 25.11.93
************************************************************/

  #include "\source\types.h"
  #include "\source\rogoli.h"
  #include "hidef.h"

  #define   STUETZWERTE 1


/*********************************
  Deklarationen
*********************************/

  INT16 AbtastWert [STUETZWERTE];
  INT16 AusgangsWert [STUETZWERTE];
  INT16 Anz_Werte = STUETZWERTE;

  GLEIT TV;
  GLEIT DIV=20.47;


/*********************************
  Wandelroutine
*********************************/

void wandel (void)
{
if (AbtastWert[0] & 0x0800)                 /* Test ob Abtastwert negativ? */
  {
   TV = AbtastWert[0] / DIV ;               /* wenn nein, dann teilen auf */
   TV = TV + 100;                           /* Wert zwische 100 und 200 */
  }

else
  {
   AbtastWert[0] = AbtastWert[0] | 2048;    /* wenn ja, dann Vorzeichenbit */
   TV = AbtastWert[0] / DIV ;               /* eliminieren und teilen */
  }

TPU_PWM (1,3,200,TV);                       /* Ausgabe des PWM-Signals */
}



/*********************************
 Hauptprogramm
*********************************/

void main (void)
{

EnableInterrupts;                            /* freigeben Interrupt */
```

```
 DefiniereUebertragung (&wandel);             /* definieren der Routine */
 LoescheVorgeschichte ();
 StarteUebertragung ();


   while (TRUE)                               /* Endlosschleife */
   {
   }
}
/* END OF FILE */
```

Anhang F: Linkdatei HWTEST. prm

```
 LINK HWTEST.abs

 NAMES  \source\HWTEST.o \source\rogoli.o startmc.o ansi32.lib
 END


 SECTIONS
    MY_RAM = READ_WRITE  0x4000 TO 0x43FF;
    MY_ROM = READ_ONLY   0x1000 TO 0x3FFF;


PLACEMENT
    DEFAULT_ROM, ROM_VAR, STRINGS  INTO  MY_ROM;
    DEFAULT_RAM                    INTO  MY_RAM;
END

STACKSIZE  0x200
```

Anhang G: Informationsdatei HWTEST. map

```
PROGRAM "HWTEST.abs"

****************************************************
TARGET SECTION
----------------------------------------------------


Processor: MOTOROLA, SMALL


****************************************************
FILE SECTION
----------------------------------------------------
\source\HWTEST.o
\source\rogoli.o
startmc.o
ansi32.lib (\lib\rm68k\rtsm68k3.o)

****************************************************
STARTUP SECTION
----------------------------------------------------
Linker generated code (at 0x1000) before calling __Startup:
MOVE #0x2700, SR
JMP 0x1816
_startupData is allocated at 100A and uses 50 Bytes

extern struct _tagStartup{
    unsigned flags                              0
    _PFunc   main                            115E (_main)
    unsigned dataPage                           0
    long     stackOffset                     423E
    int      nofZeroOuts                        1
    _Range   pZeroOut   ->
                          4000                  3E
    long     toCopyDownBeg                   103C
    _PFunc   mInits   ->
    _PFunc   libInits ->
} _startupData;

****************************************************
SEGMENT-ALLOCATION SECTION
----------------------------------------------------
Segmentname                     from             to

_PRESTART                       1000           1009
STARTUP                         100A           103B
COPY                            103C           10BD
FUNCS                           10BE           1A3F
DEFAULT_RAM                     4000           403D
SSTACK                          403E           423D

****************************************************
OBJECT-ALLOCATION SECTION
----------------------------------------------------
Type:               Name:                         Address:


MODULE              -- \source\HWTEST.o --
- PROCEDURES:
                    wandel                                10BE
                    main                                  115E
```

```
- VARIABLES:
                    AbtastWert                          4000
                    AusgangsWert                        4002
                    Anz_Werte                           4004
                    TV                                  4006
                    DIV                                 400A


MODULE              -- \source\rogoli.o --
- PROCEDURES:
                    DefiniereUebertragung               1184
                    LoescheVorgeschichte                1198
                    TPU_Grundeinstellung                11CE
                    TPU_InterruptEinstellungen          127C
                    TPU_InterruptFreigeben              12D6
                    TPU_PWM                             1314
                    TPU_EinfacherAusgang                148A
                    IntProc                             1580
                    StarteUebertragung                  1646
                    TPU_AusgangHigh                     16D4
                    TPU_AusgangLow                      1738

- VARIABLES:
                    TICR                                400E
                    TMCR                                4012
                    CIER                                4016
                    CISR                                401A
                    INT240                              401E
                    AD_WANDLER_INIT                     4022
                    AD1_WANDLER                         4026
                    BaseAddrCS3                         402A
                    ModeCS3                             402E
                    BaseAddrCS6                         4032
                    ModeCS6                             4036
                    Uebertragung                        403A


MODULE              -- startmc.o --
- PROCEDURES:
                    CallInitFunctions                   179C
                    LibInits                            17B4
                    Init                                17D0
                    _Startup                            1816

- VARIABLES:
                    _startupData                        100A


MODULE              -- \lib\rm68k\rtsm68k3.o --
- PROCEDURES:
                    _FADD                               1844
                    _FDIV                               190A
                    _FSFLOAT                            19B6
                    _FUTRUNC                            19EA


*************************************************
SEGMENT USE IN OBJECT-ALLOCATION SECTION
-------------------------------------------------
SEGMENT "FUNCS"
   wandel   main   DefiniereUebertragung   LoescheVorgeschichte   TPU_Grundeinstellung
TPU_InterruptEinstellungen
  TPU_InterruptFreigeben  TPU_PWM  TPU_EinfacherAusgang  IntProc  StarteUebertragung
  TPU_AusgangHigh  TPU_AusgangLow  CallInitFunctions  LibInits  Init
  _Startup  _FADD  _FDIV  _FSFLOAT  _FUTRUNC
```

```
SEGMENT "DEFAULT_RAM"
  AbtastWert  AusgangsWert  Anz_Werte  TV  DIV  TICR
  TMCR  CIER  CISR  INT240  AD_WANDLER_INIT
  AD1_WANDLER  BaseAddrCS3  ModeCS3  BaseAddrCS6  ModeCS6
  Uebertragung

****************************************************
UNUSED-OBJECTFILE-OBJECTS SECTION
----------------------------------------------------
NOT USED VARIABLES
\source\rogoli.o:
  AD2_WANDLER  AD3_WANDLER  AD4_WANDLER  AD5_WANDLER  AD6_WANDLER  AD7_WANDLER
  AD8_WANDLER
NOT USED PROCEDURES
\source\rogoli.o:
  TPU_InterruptSperren

****************************************************
OBJECT-DEPENDENCIES SECTION
----------------------------------------------------
wandel                    USES  _FUTRUNC _FADD _FDIV _FSFLOAT TPU_PWM DIV
                                TV AbtastWert
main                                USES   StarteUebertragung LoescheVorgeschichte wandel
DefiniereUebertragung
DefiniereUebertragung     USES  Uebertragung
LoescheVorgeschichte      USES  AusgangsWert AbtastWert Anz_Werte
TPU_Grundeinstellung      USES  TMCR
TPU_InterruptEinstellungen USES  TICR
TPU_InterruptFreigeben    USES  CIER
IntProc                USES  TPU_AusgangLow Uebertragung AusgangsWert AbtastWert Anz_Werte
AD1_WANDLER
                                AD_WANDLER_INIT TPU_AusgangHigh CISR
StarteUebertragung        USES  ModeCS6 BaseAddrCS6 ModeCS3 BaseAddrCS3 IntProc INT240
                                      TPU_EinfacherAusgang TPU_PWM TPU_InterruptFreigeben
TPU_InterruptEinstellungen TPU_Grundeinstellung

CallInitFunctions         USES  _startupData
LibInits                  USES  Init
Init                      USES  LibInits CallInitFunctions
_Startup                  USES  Init _startupData
```