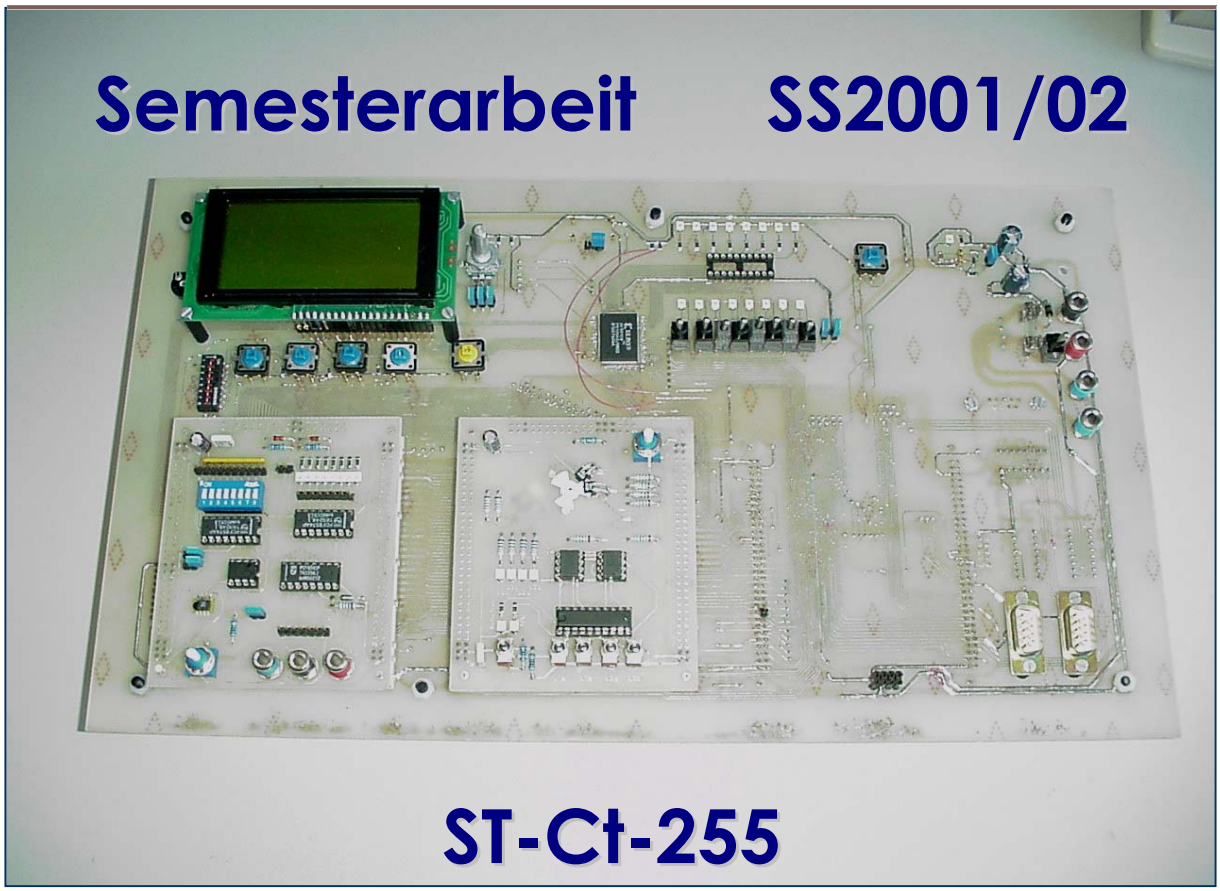
CT - lab Infrastructure



von

Claudio Foscan,
Anthony Sibler
und
Stefan Mächler

1 Grobpflichtenheft

1.1 Einführung

Das Grobpflichtenheft dient dazu, die zentralen Elemente und Funktionen der Semesterarbeit zu definieren. Dies vor allem um den Aufwand besser abschätzen zu können. In erster Linie geht es um die Festlegung der zu entwickelnden Hardware. Im Vordergrund stehen die Hauptplatine als Interface für den MEGA332 und die dazugehörigen Module. Es ist uns ein Anliegen, dass die zukünftigen Benutzer der Evaluationskonsole einen breiten Einblick in die Welt der eingebetteten Systeme erhalten.
Ein weiterer Bestandteil der Semesterarbeit besteht in der Konzipierung einer neuen Programmierungsumgebung auf der Ebene von C und C++. Für Schulungszwecke erscheint es uns sinnvoll, eine komfortable Entwicklungsumgebung zu verwenden, die nebst dem Editieren von Texten noch weitere Möglichkeiten, wie z.B. das *debuggen* von Applikationen ermöglicht.

1.2 Hinweise

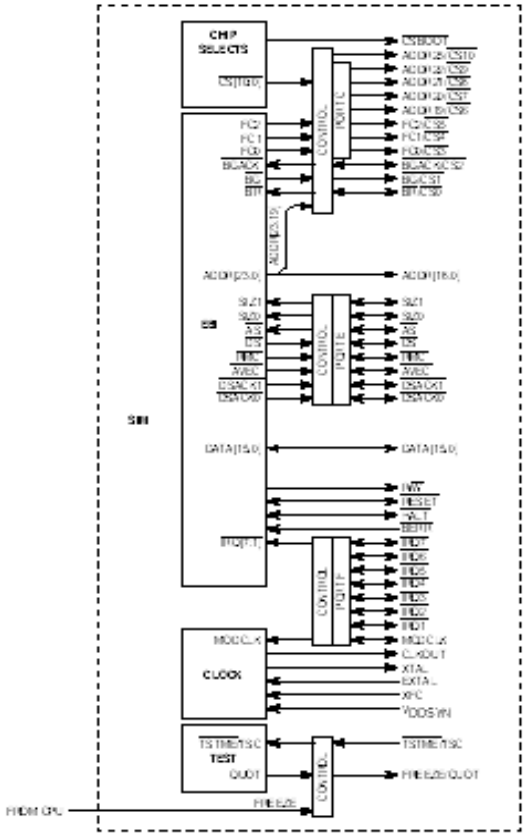
Der Inhalt des Dokumentes beruht auf den mündlichen Besprechungen mit Herrn Brändle. Das Grobpflichtenheft ist Bestandteil des Gesamtberichts. Kursiv geschriebene Wörter werden im Anhang (Glossar) näher erläutert.

1.3 Ausgangslage

Zur Ausbildung hat die HSR im Jahr 1999 zwanzig MEGA 332 Einplatinencomputer angeschafft. Mit dabei waren auch die Lizenzen für die Programmieroberfläche WinECO-C.
Drei Jahre dienten die Mega-Boards nicht nur der Grundausbildung im Fach Computertechnik, sondern wurden auch in diversen Semester- und Diplomarbeiten verwendet.

1.4 Funktionale Anforderungen

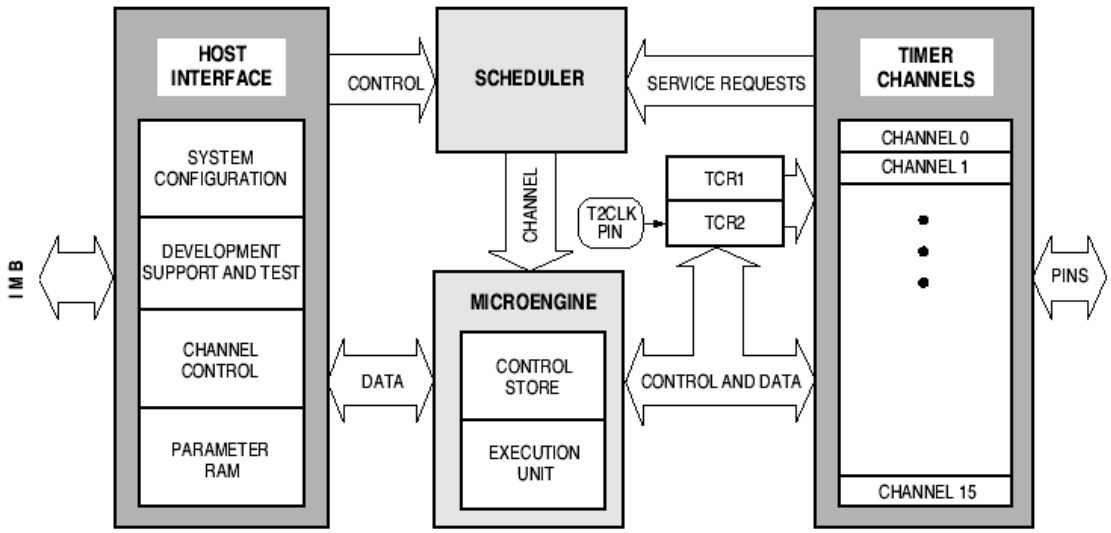
Die im CT-Labor vorhandenen MEGA-332 Boards besitzen einen leistungsfähigen Motorola MC86332 Controller. Er besteht aus einer Recheneinheit (CPU) sowie einem System Integration Modul (*SIM*), einer Time processing Unit (*TPU*) und einem Queued Serial Module (*QSM*) . Die CPU basiert auf dem legendären MC68000er Prozessor. Einstellungen für die Systemkonfiguration erfolgen entweder durch die externe Beschaltung mit Widerständen oder durch beschreiben der entsprechenden SIM-Register im Programm.
Bei externer Beschaltung des Controllers werden die Register des SIM beim einschalten automatisch beschrieben. Während der Laufzeit lassen sich die entsprechenden Register jederzeit neu beschreiben.
Die Konfiguration umfasst die Einstellung der *Chipselect* Bereiche, der Bus-Koordination und die *Interrupt*-Aktivierung bzw. Deaktivierung.



Grafik 1.4-1 : SIM Ein- und Ausgänge

Das QSM enthält zwei Schnittstellen für die serielle Kommunikation. Die Konfiguration erfolgt ebenfalls über die entsprechenden QSM-Register.

Mit der TPU lassen sich bis zu 16 Ein- oder Ausgänge direkt programmieren. Für die Ansteuerung komplexer Ein- und Ausgabeformate wie *PWM, QDEC, UART* usw. sind die entsprechenden Routinen bereits imlementiert und stehen dem Anwender zur Verfügung.



Grafik 1.4-2 : TPU

Der Einplatinencomputer MEGA332 besitzt weitere Hardwarekomponenten wie AD-Wandler, RTC, 1Mbyte Flash-ROM, DUART sowie einen CAN-Controller.
Der Prozessor selbst besitzt 24 Adress- und 16 Datenleitungen, von denen 20 Adress- und 16 Datenleitungen nach aussen individuell nutzbar sind. Von den sieben Interrupt-Request Anschlüssen des MC68332 Controllers dürfen drei für externe Anwendungen verwendet werden. Die restlichen vier werden für die MEGA interne Peripherie verwendet.
Die neue Hauptplatine wird mit einer LCD-Anzeige und vier Schaltern bestückt. Sie lassen sich durch den Anwender frei programmieren. Dazu kommen noch ein Encoder, ein PLD sowie ein Interruptcontroller. Die Speisespannung der Mega-Platine beträgt 5V ±5% bei 200...500mA.

Um der Entwicklungsumgebung die bestmöglichste Flexibilität zu verleihen, lässt sich die Hauptplatine mit Modulen erweitern. Auf der Hauptplatine sind Steckplätze für zwei Module vorhanden. Die jeweiligen Module unterscheiden sich durch den Einsatz von verschiedenen Pheriperiebauteilen wie z.B. IC's, LED's, Schalter und Sensoren. Sie werden speziell für die spätere Anwendung in der CT-Ausbildung entwickelt. Es ist dabei wichtig, ein möglichtst breites Spektrum von Anwendungen zu berücksichtigen.
Die Komposition mit den verschiedenen Modulen lässt sich mit einem PC programmieren. Zu Testzwecken werden für die Komponenten entsprechende C Programme entworfen. Sie sollen später zukünftigen Programmierern als hilfreiche Beispiele für den Einstieg in die „CT-Welt" dienen.

1.5 Spezielles

Die Plattform nicht nur für Prozessoren mit einer 32-Bit Architektur kompatibel sein. Dies bedeutet, dass eine universelle Lösung gefunden werden muss, damit Adress- und Datenbus sowie sämtliche Steuerleitungen auch von anderen Rechnerarchitekturen unterstützt werden können, vorausgesetzt die Prozessorplatine arbeitet mit einer Betriebsspannung von 5V. Eine weitere Herausforderung ist die Wahl der Module. Es besteht durchaus die Möglichkeit, im Gegensatz zum MEGA-Board, welches bereits einen D/A-Wandler besitzt, eine Platine ohne Wandler einzusetzen. In diesem Fall soll das entsprechende Modul das analoge Signal entweder unbearbeitet an den Mega weitergeben, oder das Signal mit einem A/D-Wandler digitalisieren und es anschliessend über den Datenbus dem Controller auf der Platine übergeben.
Solche und weitere Probleme sind in einer ersten Phase genau zu Analysieren.
Weitere Meilensteine der Arbeit sind die Programmierung des PLDs zur Peripheriesteuerung und die Auswahl von Standardbausteinen auf den Modulen.

1.6 Zusätzliche Anforderungen

Damit die Stromversorgung für die Plattform gewährleistet ist, sind Netzteile für den maximal benötigten Strom von einem Ampere notwendig. Der Stromverbrauch der Module ist noch nicht berücksichtigt.
Ebenfalls sind auch noch mechanische Einrichtungen für die Arretierung der Hauptplatine nötig.

1.7 Labor Infrastruktur

Das CT-Labor ist mit PC's ausgestattet. Die auf ihnen installierte Software erlaubt es, Text und Code zu schreiben, den Code zu kompilieren und dann auf die Boards herunterzuladen.

2 Arbeitsaufteilung / Terminplanung

2.1 Zielsetzung

Eine grobe Zielsetzung ist im Pflichtenheft aufgeführt worden. In diesem ist der ungefähre Umfang der Arbeit festgelegt worden. Aufgrund der Vielzahl von Möglichkeiten Module mit verschiedensten Hardwarekomponenten zu entwickeln, sind Konzeptänderungen und Erweiterungen im Verlauf der Arbeit nicht ausgeschlossen. Ebenfalls in betracht zu ziehen sind Herausforderungen über welche am Anfang noch keine Indizien vorhanden sind. Für diese Fälle erachten wir es als notwendig, fundamentale Richtungsentscheide immer in Absprache mit unserem Betreuer zu fällen.

2.2 Verfahren

Unsere Gruppe besteht aus drei Personen. Wir legten eine Strategie fest, bei der je ein Gruppenmitglied einen Teilbereich der Arbeit übernimmt. Die Teilbereiche wurden folgendermassen aufgeteilt:

Claudio Foscan:	Bereich Konzipierung und Fertigung der Grundplatte mit den entsprechenden Hardwarekomponenten
Antony Sibler:	Entwickeln und produzieren der Module mit den entsprechenden Hardwarekomponenten
Stefan Mächler:	Software für die zu entwickelnde Infrastruktur sowie die Einführung einer neuen Programmierumgebung

Wichtig bei diesem Vorgehen ist, das von Zeit zu Zeit gegenseitige Informationsaustausche stattfinden.

2.3 Termine

- 18. März 2002: Start der Semesterarbeit
- 10/11.Juli.2002: Präsentation
- 12.Juli.2002: Abgabe der Studienarbeitarbeit und des Berichts

2.4 Sitzungen

Die Sitzungen mit Hr. Brändle finden jeweils Montags von 10.00 bis 11.00 Uhr statt. Ausgenommen davon sind die Unterrichtsfreien Tage (Ostermontag, Pfingstmontag)

2.5 Bericht

Es ist vorgesehen dass der Bericht laufend aktualisiert wird.

2.6 Zeitplan

Woche	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Claudio Foscan																
Konzept							Pfingsten									
Schema Grundplatine							Auffahrt									
											1					
Layout Grundplatine																
											2					
Xilinx Programm																
											3					
Aufbau und Test																
															4	
Stefan Mächler																
Konzept																
CodeWarrior																
									5							
Testprogramme																
Inbetriebnahme Module																
											6				7	
Anthony Sibler																
Konzept																
Modul 1 I2C																
Modul 2 Stepping Motor																
Modul 3 ADC/DAC																

Tabelle 2.6-1: individueller Zeitplan

Woche	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Allgemein																
Dokumentation																
Milestone																
Test																
Präsentation																

1	Schema fertig
2	Layout version 1.0 fertig -> Prototyp in Auftrag(HSR)
3	Xilinx Pinout Definitif
4	Erfolgreicher Test der Trägerplatine
5	Anpassung CodeWarrior problematisch
6	Schrittmotormodulansteuerung OK
7	I²C-Modulansteuerung OK

Tabelle 2.6-2: Gruppenzeitplan

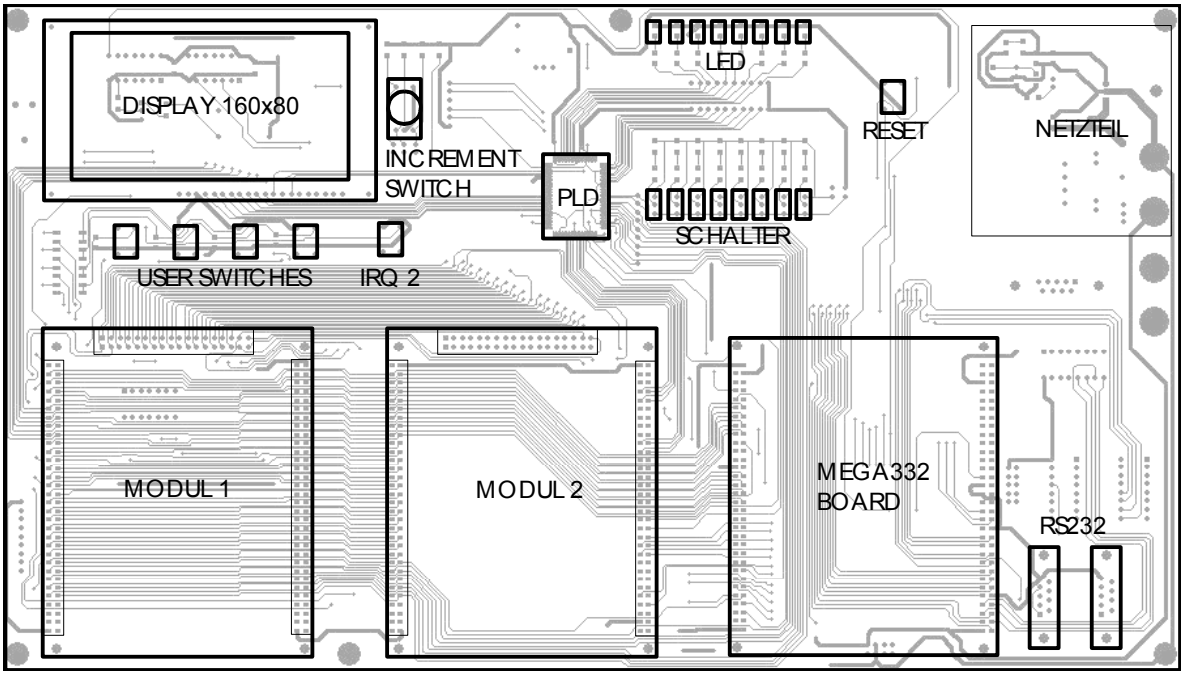
2.7 Projektlog

Wärend der Arbeit wurde ein detailliertes Projektlog geführt. Es ist im Anhang C zu finden.

3 Trägerplatine

3.1 Aufbau

Die Trägerplatine besitzt auf der Rückseite einen Terminal für die Prozessorplatine. Ein PLD (Programmable Logic Device) sorgt dafür, dass die Steuersignale an den richtigen Adressenten gelangen. Mit dem auf der Vorderseite vorhandenen Display besteht die Möglichkeit, Menueinstellungen anzuzeigen oder mit den vier vorhandenen Schaltern, Menüeinstellungen vorzunehmen. Rechts neben dem Display ist zudem noch ein Inkrementalgeber angebracht.



Grafik 3.1-1 : Layout Frontseite

Eine Reihe mit acht Leuchtdioden und eine Reihe mit acht Schaltern lassen sich für beliebige Anwendungen individuell programmieren. In der unteren Hälfte der Printplatte sind Steckplätze für zwei Erweiterungsmodule vorhanden. Zwei serielle Schnittstellen befinden sich in der rechten unteren Ecke der Trägerplatine. Das Netzteil ist auf der Rückseite der Trägerplatine angebracht.

3.2 Display und Schalter

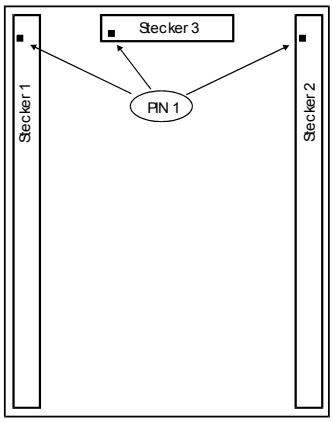
Mit dem Grafik-Display können Menüs, Variablen, Programmabläufe und vieles mehr angezeigt werden. Technische Details zum Display sind auf der CD zu finden. Unterhalb der grafischen Anzeige sind vier Schalter so angebracht, dass auf dem Display ein *Softmenu* realisiert werden kann. Die zusätzlichen Schalter und LED's sind als Ein- und Ausgabeport gedacht. Der LED Port erscheint in zwei Varianten, entweder nur beschreibbar oder schreib- und lesbar.
Die Adressierung ist in der Memory-Map (Grafik 1.6-1) ersichtlich. Im weiteren wurde die Trägerplatine noch mit einem Interrupt- und einem Resettaster versehen. Durch betätigen des Interrupttasters wird ein direktes Interruptsignal an das MEGA332-Board gesendet. Die dazugehörigen Behandlungsroutinen lassen sich im entsprechenden Softwareprojekt individuell programmieren.

3.3 PLD-Baustein

Der PLD Baustein verfügt über 108 Makrozellen und 2400 Gatter. In ihm wird die Adressdecodierung für die CS Signale (siehe 3.6 : Memorymap), die Ein- und Ausgaberegister für die 8 LEDs und 8 Schalter und die Piepsersteuerung vorgenommen. Der dazugehörige VHDL-Code ist auf der CD zu finden. Getaktet wird der Baustein über einen externen Taktgenerator oder über den CPU-Takt vom Prozessor-Board. Benötigt wird der Takt für den Piepserzähler und die State-Machine.

3.4 Modul-Sockel ExBus

Die Trägerplatine ist mit zwei Sockeln für je ein Erweiterungsmodul versehen. Die Stecker 1 & 2 der Module sind universell spezifiziert, d.h. alle möglichen Signale des Controllers stehen den Modulen zur Verfügung. Dabei ist zu erwähnen, dass aufgrund des verwendeten MEGA332 Board nicht alle Signale des ExBus bedient werden können, da sie nicht herausgeführt sind. Die speziellen Signale für die Timer Ports, die analogen Eingänge und CAN-Leitungen des MEGA332 Boards sind auf den Stecker 3 geführt. Die Sockelbelegung sieht folgendermassen aus :



Stecker 1

Pin Nr.		Pin Nr.	
GND	1	2	VCC
D 0	3	4	D 1
D 2	5	6	D 3
D 4	7	8	D 5
D 6	9	10	D 7
D 8	11	12	D 9
D 10	13	14	D 11
D 12	15	16	D 13
D 14	17	18	D 15
D 16	19	20	D 17
D 18	21	22	D 19
D 20	23	24	D 21
D 22	25	26	D 23
D 24	27	28	D 25
D 26	29	30	D 27
D 28	31	32	D 29
D 30	33	34	D 31
INT 0	35	36	INT 1
INT 2	37	38	INT 3
INT 4	39	40	INT 5
INT 6	41	42	INT 7
IPL0	43	44	IPL1
IPL2	45	46	AVEC*
IPEND*	47	48	IRQ 2*
CS 0*	49	50	CS 1*
CS 2*	51	52	CS 3*
CS 4*	53	54	CS 5*
CS 6*	55	56	CS 7*
	57	58	
MI 0	59	60	MI 1
MI 2	61	62	MI 3
GND	63	64	VCC

Stecker 2

Pin Nr.		Pin Nr.	
VCC	1	2	GND
A 0	3	4	A 1
A 2	5	6	A 3
A 4	7	8	A 5
A 6	9	10	A 7
A 8	11	12	A 9
A 10	13	14	A 11
A 12	15	16	A 13
A 14	17	18	A 15
A 16	19	20	A 17
A 18	21	22	A 19
A 20	23	24	A 21
A 22	25	26	A 23
A 24	27	28	A 25
A 26	29	30	A 27
A 28	31	32	A 29
A 30	33	34	A 31
R / W*	35	36	AS*
DS*	37	38	DSACK 0*
SIZ 0	39	40	DSACK 1*
SIZ 1	41	42	BERR*
MISO	43	44	MOSI
SS*	45	46	SCK
BR	47	48	BG
BACK	49	50	
FC 0	51	52	FC 1
FC 2	53	54	
HALT*	55	56	
RESET*	57	58	CPUCLK
	59	60	
V -15	61	62	V +15
VCC	63	64	GND

Stecker 3

Pin Nr.		Pin Nr.	
RxD 2	1	2	CTS 2*
TxD 2	3	4	RTS 2*
AIN 0	5	6	AIN 1
AIN 2	7	8	AIN 3
AIN 4 / CAN RxD0	9	10	AIN 5 / CAI
AIN 6 / CAN RxD1	11	12	AIN 7 / CAI
AIN 8	13	14	AIN 9
AIN 10 / RxD1	15	16	CTS 1*
TxD1	17	18	RTS 1*
STDP*	19	20	
TP 0	21	22	TP 1
TP 2	23	24	TP 3
TP 4	25	26	TP 5
TP 6	27	28	TP 7
TP 8	29	30	TP 9
TP 10	31	32	TP 11
TP 12	33	34	TP 13
TP 14	35	36	TP 15

Grafik 3.4-1: Sockel und Pinbelegung

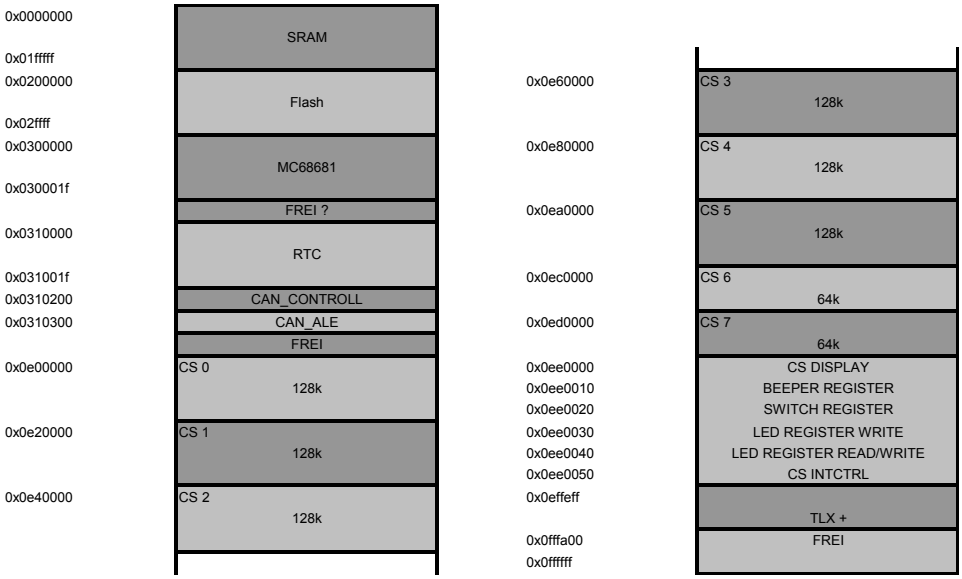
3.5 Interrupt Controller

Die Trägerplatine ist auf der Rückseite mit einem Priority *Interruptcontroller* 82C59 von Intel bestückt. Dieser Baustein erweitert die drei Interruptleitungen des MEGA332 Boards mit 8 zusätzlichen Interrupts. Diese Interrupts lassen sich entsprechend den Eingangsleitungen priorisieren.

3.6 Memory-MAP

Der Adressbereich von 0x0000000 bis 0x02FFFFF ist für das RAM und ROM (Flash) auf dem MEGA332 Board definiert. Auf den darauf folgenden Adressen liegen die Hardwarekomponenten, die ebenfalls auf dem MEGA Board vorhanden sind. Zwischen 0x0310300 und 0x0DFFFFF befindet sich ein freier Adressraum. Für die folgenden sechs 128kB und zwei 64kB grossen Blöcke werden ChipSelect Signale vom PLD geliefert. Die restlichen Adressen sind für die Peripherie der Trägerplatine und der Module bestimmt.

In der nachfolgenden Grafik sind die einzelnen Adressblöcke mit den zugehörigen Registerbezeichnungen oder Chipselectsignalen aufgeführt.



Grafik 3.6-1: Memory Map

4 Grafikdisplay AG-16080A

4.1 Display Daten

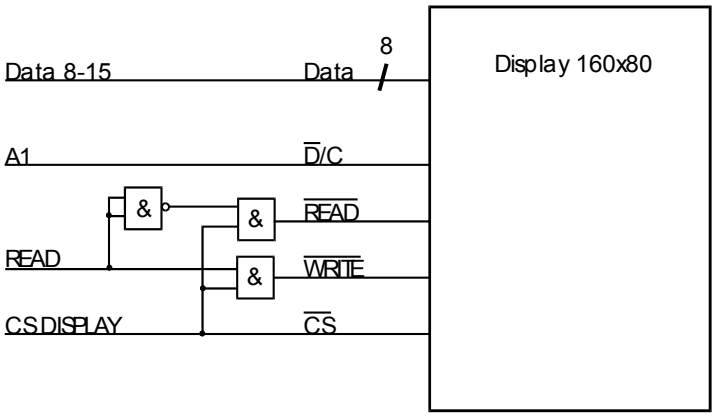
Wir haben uns für ein Display der Firma Marelcom entschieden. Das Display hat eine Auflösung von 160 x 80 Pixel. Die Anbindung an unser Mikroprozessorsystem erfolgt via Parallelbus und einigen Steuerleitungen. Dank des integrierten Charaktergenerators muss dem Displaycontroller für ein Zeichen lediglich ein zweistelliger Hexcode übermittelt werden. Der Hexcode entspricht dem ASCII-Zeichensatz, ist aber mit einem Offset von 4 Byte behaftet. Das Display verfügt über 64kByte RAM und einen Controllerchip von Toshiba (Typ T6963C). Die Datenblätter zum Controller sind auf der CD zu finden.

4.2 Schnittstelle

Zur Schnittstelle des Display-Controllers gehören folgende Leitungen:

- 8 Datenleitungen
- ChipSelect
- Command / Data Leitung
- Read und Write Leitungen

Das Chipselect Signal für das Display wird direkt aus der Adresse 0x0ee000...2 gewonnen. Um signalisieren zu können, ob Instruktionen oder Daten übertragen werden sollen, wurde der Command/Data Anschluss des Displays auf die Adressleitung A1 gelegt. Das bedeutet das sich das Data Register mit Adresse **0x0ee0000** und das Command Register mit **0x0e0002** in Kombination mit dem Chipselect auswählen lassen.

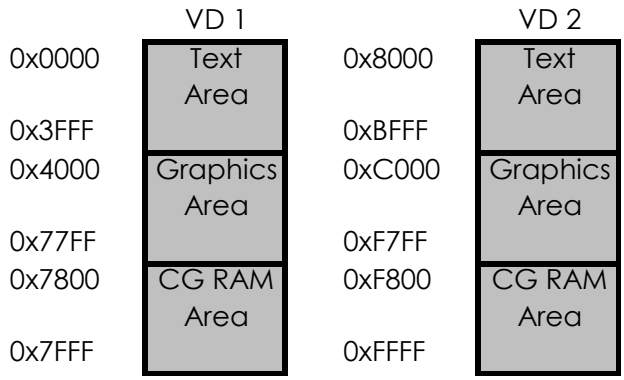


Grafik 4.2-1 : Display Anschlüsse

Die Read und Write Signale sind jeweils mit dem Chipselect Signal verknüpft, wobei das Write Signal lediglich durch die Invertierung des Readsignals erzeugt wird.

4.3 Interner Aufbau

Wie schon erwähnt ist das Display mit einem Controller der Firma Toshiba ausgestattet. Dieser Controller ermöglicht verschieden textbasierende und grafische Darstellungen. Intern verfügt das Display über 64kByte RAM. Da der Controller eine Auflösung von 160 x 80 Pixel nicht als ganzes Verwalten kann, ist das Display in zwei virtuelle Displays unterteilt. Jeweils die Hälfte des RAMs ist einem virtuellen Display zugeteilt. Für den Zugriff auf das Display ändert das aber nichts.



Grafik 4.3-1 : Interner Displayaufbau

4.4 Modi

Der Controller beherrscht drei verschiede Modi, in denen nur Text, nur Grafik oder Text und Grafik gemischt dargestellt werden kann. Die gemischte Anzeigen erfolg über logische Verknüpfungen zwischen Text und Grafik (EXOR, AND usw...). Mehrere Bytes können mit einer Adressierung (ohne jedes mal den Adresspointer zu inkrementieren zu müssen) gesendet werden. Im Auto-Mode Betrieb wird nach jedem Datenbyte der Adresspointer automatisch erhöht.

4.5 Zugriffszyklen

Um überhaupt auf das Display zugreifen zu können, müssen spezielle Zugriffsmuster eingehalten werden. Da die Kommunikation mit dem Displaycontroller asynchron verläuft (der Controller hat intern Empfangsregister für Daten und Commands) sind beim Lesen und Schreiben gewisse Bedingungen einzuhalten. So muss zum Beispiel vor jedem Datentransfer, der Inhalt des Statusregisters gelesen und überprüft werden, ob der Controller bereit ist Daten zu senden oder zu empfangen.

Das Statusregister ist wie folgt aufgebaut :

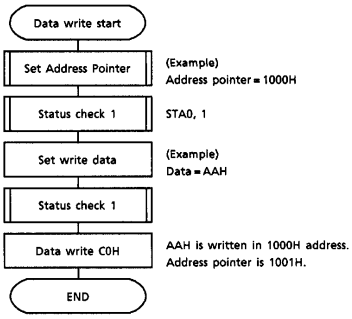
MSB LSB

STA 7 D 7	STA 6 D 6	STA 5 D 5	STA 4 D 4	STA 3 D 3	STA 2 D 2	STA 1 D 1	STA 0 D 0

STA 0	Check command execution capability	0: disable 1: enable
STA 1	check data read/write capability	0: disable 1: enable
STA 2	check auto mode data read capability	0: disable 1: enable
STA 3	check auto mode data write capability	0: disable 1: enable
STA 4	not used	
STA 5	check controller operation cpability	0: disable 1: enable
STA 6	error flag. Used for screen peek and screen copy commands	0: no error 1: error

Grafik 4.5-1 : Statusregister

Ablauf um ein Datenbyte zu senden :



Grafik 4.5-2 : Sende Ablauf

Detaillierte Angaben zu den verschiednen Abläufen finden sich ebenfalls im Datenblatt des Controllers auf der CD.

4.6 Zeichensatz :

CHARACTER CODE MAP
ROM code 0101

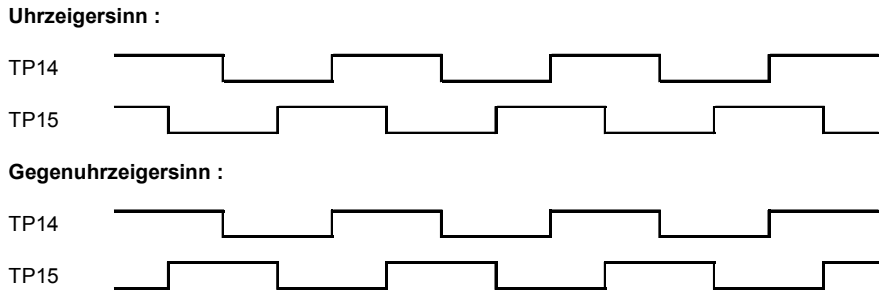
LSB / MSB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2																
3																
4																
5																
6																
7																

Tabelle 4.6-1: Display-Zeichensatz

5 Schalter auf der Trägerplatine

5.1 INCREMENT SWITCH

Rechts neben dem Display befindet sich ein Encoder der Firma Marelcom. Der Encoder verfügt nebst den *Drehimpulsgeber* zusätzlich über einen Drucktaster. Angeschlossen wird er an einen der Timer Eingänge des MC68332. Die Decodierung der Signale wird in einer bereits implementierten Quadraturdecoder-Routine innerhalb der TPU vorgenommen.
In folgenden Zeitdiagramm sind die Signale am TP14 und TP15 zu sehen :



Grafik 5.1-1:Zeitdiagramm Encoder

Es können Richtung und Umdrehungen detektiert werden. Der Encoder besitzt eine Auflösung von 12° oder 30 Schritte pro Umdrehung. Die CPU kann den Zählerwert jederzeit aus dem Parameter-Ram der TPU auslesen. Der Drucktaster ist an den Timer Port 13 gelegt.

5.2 USER SWITCHES

Die sog. USER SWITCH's können entweder an den Timerport (TP 9-12) des MEGA332 Boards, oder an die vier Interruptleitungen (INT 0-3) des Interruptcontrollers auf der Hauptplatine gelegt werden. Mit USER SWITCH's sind die vier Taster gemeint, welche sich unmittelbar unter dem Display befinden.
Da die Interruptleitungen des 82C59A Controllers HIGH aktiv sind, müssen die Schalter über Pulldown- Widerstände auf Masse gelegt werden. Das gleiche gilt ebenfalls für die Timerportkanäle. Dabei zählt eine Null als ausgeschaltet und eine Eins als eingeschaltet.

5.3 IRQ 2 SWITCH

Um einen echten Interrupt auslösen zu können, haben wir einen Schalter direkt an die Interruptleitung 2 (IRQ2*) des MEGA332 Boards gelegt. Die Interruptleitungen am MC68332 sind LOW aktiv.

5.4 RESET SWITCH

Durch betätigen des Resettasters wird ein Hardwarereset (RESET*) ausgeführt. Dieser Reset betrifft im wesentlichen alle Bausteine inkl. den Resetleitungen auf den Modulsteckern.

6 ExBus & MEGA332_Ex_Port

Bei der Planung der Modulschnittstelle haben wir besonders auf eine universelle Spezifikation geachtet. Die Weiterverwendbarkeit der Module bei einem allfälligen CPU-Wechsel sollte gewährleistet sein.

6.1 ExBus

Bei der Definition des „ExBuses“ galt es folgenden Aspekten zu berücksichtigen:

- Universalität
- Möglichst viele Standartsignale
- Stabilität der Steckverbindung
- Gute Stromversorgung

Damit die Steckverbindung nicht schon nach wenigem Umstecken Wackelkontakte aufweisen, wurden stabile Stecker vom Typ DIN42612 a+b eingesetzt. Im Ganzen stehen 128 Kontakte zur Verfügung, die wie folgt belegt sind :

Stecker 1

	Pin Nr.		Pin Nr.	
GND	1		2	VCC
D 0	3		4	D 1
D 2	5		6	D 3
D 4	7		8	D 5
D 6	9		10	D 7
D 8	11		12	D 9
D 10	13		14	D 11
D 12	15		16	D 13
D 14	17		18	D 15
D 16	19		20	D 17
D 18	21		22	D 19
D 20	23		24	D 21
D 22	25		26	D 23
D 24	27		28	D 25
D 26	29		30	D 27
D 28	31		32	D 29
D 30	33		34	D 31
INT 0	35		36	INT 1
INT 2	37		38	INT 3
INT 4	39		40	INT 5
INT 6	41		42	INT 7
IPL0	43		44	IPL1
IPL2	45		46	AVEC*
IPEND*	47		48	IRQ 2*
CS 0*	49		50	CS 1*
CS 2*	51		52	CS 3*
CS 4*	53		54	CS 5*
CS 6*	55		56	CS 7*
	57		58	
MI 0	59		60	MI 1
MI 2	61		62	MI 3
GND	63		64	VCC

Stecker 2

	Pin Nr.		Pin Nr.	
VCC	1		2	GND
A 0	3		4	A 1
A 2	5		6	A 3
A 4	7		8	A 5
A 6	9		10	A 7
A 8	11		12	A 9
A 10	13		14	A 11
A 12	15		16	A 13
A 14	17		18	A 15
A 16	19		20	A 17
A 18	21		22	A 19
A 20	23		24	A 21
A 22	25		26	A 23
A 24	27		28	A 25
A 26	29		30	A 27
A 28	31		32	A 29
A 30	33		34	A 31
R / W*	35		36	AS*
DS*	37		38	DSACK 0*
SIZ 0	39		40	DSACK 1*
SIZ 1	41		42	BERR*
MISO	43		44	MOSI
SS*	45		46	SCK
BR	47		48	BG
BACK	49		50	
FC 0	51		52	FC 1
FC 2	53		54	
HALT*	55		56	
RESET*	57		58	CPUCLK
	59		60	
V -15	61		62	V +15
VCC	63		64	GND

Grafik 6.1-1:ExBus

Folgende Leitungen sind im ExBus integriert :

- Datenbus 32 Bit breit
- Adressbus 32 Bit breit
- Steuerleitungen für Buszugriffe
- 8 Interruptleitungen & Interruptsteuerinterface
- 8 Chipselect Leitungen
- DMA Interface
- SPI Interface
- Speisung
- Modulidentifikation

6.2 MEGA332_Ex_Port

Der MEGA332_Ex_Port besteht aus allen Signalen, die neben den Standartbussignalen vom MEGA332 Board noch zusätzlich zu Verfügung gestellt werden.

Diese sind :

- 2 Serielle Schnittstellen
- 16 TPU Kanäle
- 10 Analog Eingänge
- 2 CAN Interfaces
- 1 Realtime Clock Timer Ausgang

Bei den seriellen Schnittstellen handelt es sich um die zwei Kanäle des Doppel-UART – Bausteins MC68681, welcher via *SPI* an den MC68332 angebunden ist. Die 16 TPU Kanäle werden direkt vom MC68332 herausgeführt. Der von Texas Instruments stammende Analog/Digital Wandler auf dem MEGA-Board besitzt 10 Kanälen. Sein Arbeitsbereich liegt, bei 12 Bit Auflösung, zwischen 0 und 4,096 Volt. Der A/D-Wandler ist via SPI mit den MC68332 verbunden. Der Realtime Clock Baustein auf dem MEGA332-Board besitzt einen Alarm-Ausgang STDP*, der den Ablauf des internen Timers signalisiert. Der CAN-Controller von Philips SJA1000T bietet 2 Kanäle .Die Pinbelegung des MEGA332_Ex_Port sieht folgendermassen aus:

Stecker 3

	Pin Nr.		Pin Nr.	
RxD 2	1		2	CTS 2*
TxD 2	3		4	RTS 2*
AIN 0	5		6	AIN 1
AIN 2	7		8	AIN 3
AIN 4 / CAN RxD0	9		10	AIN 5 / CAN TxD0
AIN 6 / CAN RxD1	11		12	AIN 7 / CAN TxD1
AIN 8	13		14	AIN 9
AIN 10 / RxD1	15		16	CTS 1*
TxD1	17		18	RTS 1*
STDP*	19		20	
TP 0	21		22	TP 1
TP 2	23		24	TP 3
TP 4	25		26	TP 5
TP 6	27		28	TP 7
TP 8	29		30	TP 9
TP 10	31		32	TP 11
TP 12	33		34	TP 13
TP 14	35		36	TP 15

Grafik 6.2-1:Mega332 ExPort

Aufgrund der Doppelbelegung einzelner Leitungen auf Seitens des MEGA332-Boards, können nicht alle Funktionen gleichzeitig verwendet werden!

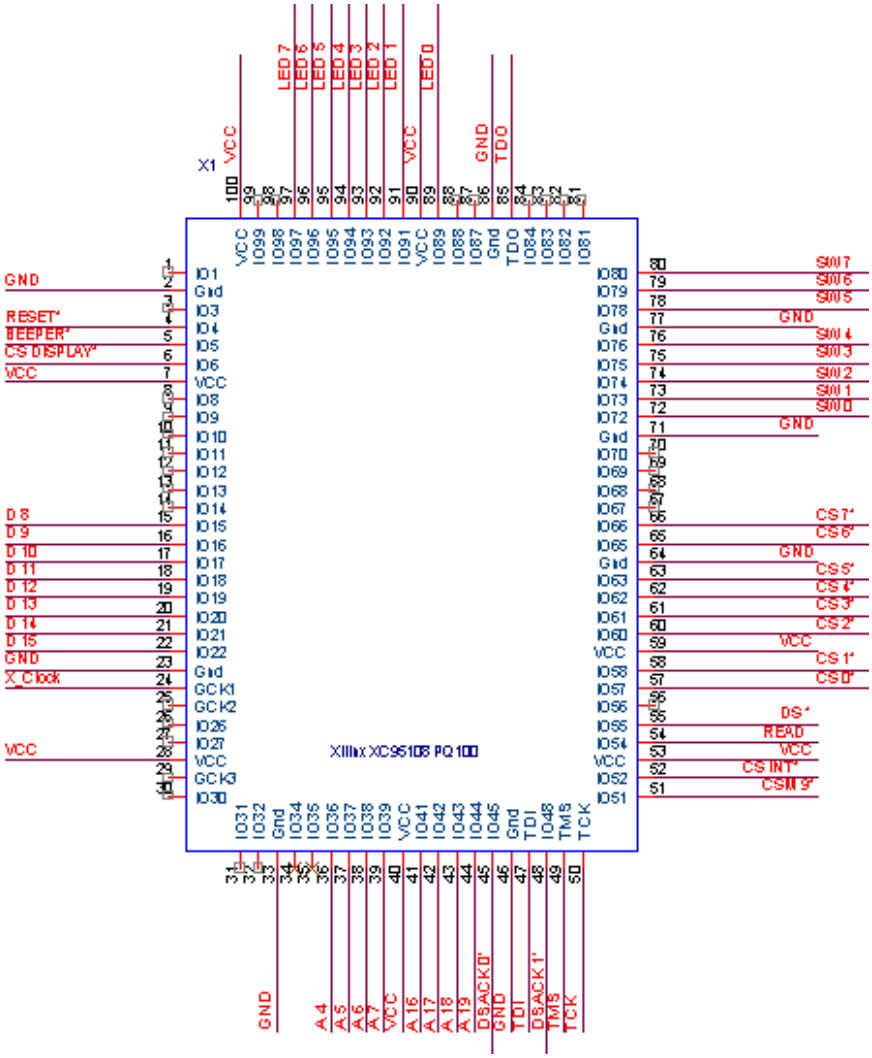
7 Xilinx 95108 PQ100 PLD

7.1 Beschreibung

Der Xilinx 95108 PQ100 Baustein ist ein sogenanntes **P**rogrammable **L**ogic **D**evice (PLD). Intern verfügt der Baustein über ca. 2500 Logikgatter und eine programmierbare Switch-Matrix, welche die Gatter untereinander verbindet. Die Gatter sind in 108 Blöcke (Macrozellen) unterteilt. Über eine *JTAG* Schnittstelle lässt sich der PLD komfortabel programmieren.
Auf der Trägerplatine übernimmt der PLD folgende Aufgaben:

- Adressdecodierung und Chipselect-Signal erzeugung für das Display, den Interruptcontroller und die 8 CS* Signale auf ExPort
- Bufferung und Bussteuerung für den Zugriff auf die 8 Schalter und LEDs
- Die komplette Beeper Logik

Um diese Aufgaben zu integrieren wird VHDL verwendet. VHDL ist eine Beschreibungssprache für programmierbare Logikbausteine. Den kompletten VHDL Code ist auf der CD zu finden. Angeschlossen ist der Baustein wie folgt :



Grafik 7.1-1

7.2 Adressen und Register

Folgende Adressen werden decodiert :

- CS 0 : 0x0e00000 - 0x0e1ffff
- CS 1 : 0x0e20000 - 0x0e3ffff
- CS 2 : 0x0e40000 - 0x0e5ffff
- CS 3 : 0x0e60000 - 0x0e7ffff
- CS 4 : 0x0e80000 - 0x0e9ffff
- CS 5 : 0x0ea0000 - 0x0ebffff
- CS 6 : 0x0ec0000 - 0x0ecffff
- CS 7 : 0x0ed0000 - 0x0edffff
- CS DISPLAY : 0x0ee--0-
- BEEPER REGISTER : 0x0ee--1-
- SWITCH REGISTER : 0x0ee--2-
- LED REG R : 0x0ee--3-
- LED REG R/W : 0x0ee--4-
- CS INT CONTROLLER : 0x0ee--5-

Beim Schreibzugriff werden die Signale CS 0-7, CS DISPLAY und CS INT CONTROLLER direkt umgesetzt, das heisst sobald eine gültige Adresse am Eingang anliegt wird das entsprechende CS Signal auf LOW gezogen. Für Lesezugriffe auf den Interruptcontroller werden zusätzlich noch die *DSACK* Signale erzeugt. Ebenso bei Zugriffen auf die SWITCHs, LEDs und den BEEPER.

7.3 Programmierung

Wie schon erwähnt werden die Chipselect Signale direkt decodiert. Das heisst sie sind hardwaremässig „verdrahtet“. Die diskrete Logik decodiert die Adresssignale und schaltet die Chipselect Signale mit einer maximalen Verzögerung von 20 ns. In VHDL sieht das wie folgt aus :

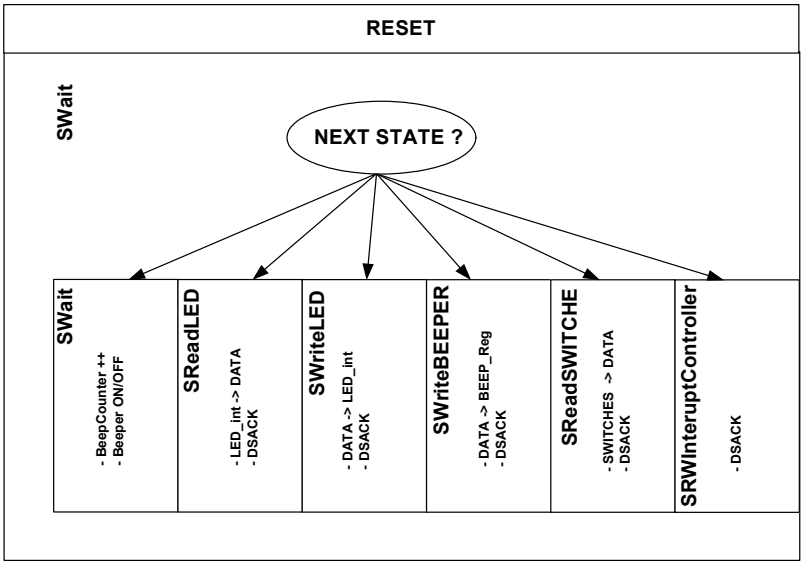
```
begin

Adressdecoding :
process(a19,a18,a17,a16,a7,a6,a5,a4,cs)
VARIABLE peripherie_vec : std_logic_vector(7 downto 0) ;
VARIABLE cs_vec               : std_logic_vector(2 downto 0) ;
begin
      peripherie_vec := (a19, a18, a17, a16, a7, a6, a5, a4) ;
      cs_vec         := (a19, a18, a17) ;

      if ((cs_vec = CS_0_ADDRESS) AND (cs='0')) then
            cs_0 <= '0' ;
      else
            cs_0 <= '1' ;
      end if;
```

Die Logik, hier mit der IF-Bedingung beschrieben, ist im Baustein Hardwaremässig verknüpft.

Anders sieht es bei der Erzeugung der DSACK Signalen und den Zugriffen auf die internen Register aus. Diese Aktionen werden von einer State-Machine gesteuert. Den Takt für die State Machine bezieht das PLD vom MEGA332 Board. Es ist der selbe Takt (16MHz) den die CPU auf dem Board benutzt.
Die State Machine ist wie folgt aufgebaut :

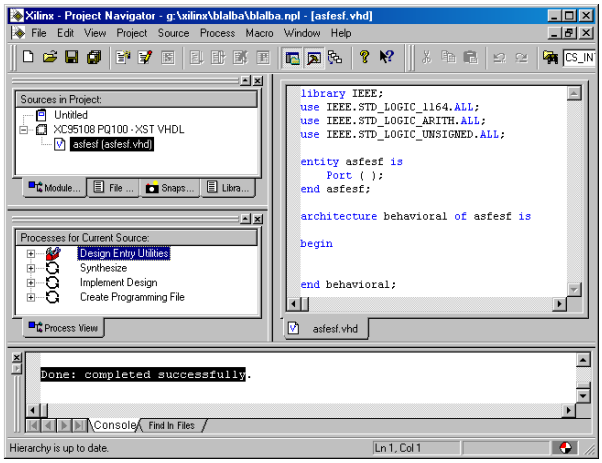


Zu den einzelnen States :

1. **Swait**
   In diesem State wartet die State Machine bis eine gültige Adresse (inkl. READ, CS und DS Signalen). Sobald diese anliegt, springt sie in den dazugehörigen State. Zusätzlich wird noch die BEEPER Logik gesteuert. Das heisst, der Zähler für den Intervall und die Dauer werden mit jedem Taktimpuls aktualisiert. Der BEEPER Ausgang wird entsprechend gesetzt.
2. **SReadLED**
   Falls vom LED Register gelesen wird, hat das zur Folge dass in diesem State das die Daten des internen Registers auf den Datenbus gelegt und die DSACK Signale entsprechend dem Datenformat gesetzt werden. Nach einem Takt springt die State Machine wieder in den SWait State
3. **SWriteLED**
   Die Daten werden vom Datenbus in das interne Register übernommen und die DSACK Signale werden erzeugt.
4. **SReadSWITCHES**
   Die Schalter werden direkt eingelesen und auf den Datenbus gelegt und die DSACK Signale werden erzeugt.
5. **SWriteBEEPER**
   In diesem State werden die Daten auf dem Datenbus in das interne BEEPER Regeister geschrieben. Ebenfalls werden die DSACK Signale erzeugt.
6. **SRWInterruptController**
   Dieser State sorgt einzig für die Erzeugung der DSACK Signale. Die Daten auf dem Datenbus kommen direkt vom Interruptcontroller.

Programiert wird der Baustein mittels einer JTAG Schnittstelle. Die Programierumgebung kann gratis vom Internet *www.xilinx.com* heruntergeladen werden. Für den 95108-PQ100 von Xilinx wird das WebPACK Version 3.3 benötigt. Die neueren Versionen funktionieren für den Baustein nicht. Programmiert wird in VHDL. Die Pinzuordnung kann grafisch oder per Script erfolgen.

Erstellen eines Projektes mit WebPACK :



1. Installieren von WebPACK
2. Neues Projekt erstellen
    1. Projektname wählen
    2. Projektverzeichnis wählen
    3. Device Family : Xilinx 9500 CPLD
    4. Device : XC95108 PQ100
    5. Synthesis Tool : XST VHDL
3. Projekt-> Add Source oder New Source ; fügt VHDL File zu Projekt hinzu
   Im Fenster Sources in Project werden alle Sourcefiles angezeigt.
4. VHDL Code erstellen
5. Im Fenster Processes for Current Source unter dem Punkt
   Design Entry Utilitys -> User Constrains -> Pin Assignment Chip View
   Können die IO Pins grafisch zugeordnet werden.
6. Sind alle Pins zugeordnet und der VHDL Code vollständig, wird das VHDL-File compiliert. Unter dem Punkt Synthesis -> Einstellungen kann die Optimierungsmethode angegeben werden. Speed für möglichst schnelles Design, Area für grosse Designs die nur knapp in das PLD passen.
7. Sind alle Einstellungen gemacht, kann unter dem Punkt Create Programming File -> Run das Binärfile zur Programmierung erzeugt werden. Ist alles in ordung erscheint im Statusfenster „Done: completed successfully“. Im Untermenu Implement Design -> Fitter -> Fitterreport kann die Statistik über die Logik und den Platzverbrauch auf dem Chip eingesehen werden.
8. Nun kann der PLD Programmiert werden : Das Programmiertool startet man direkt aus dem WebPACK. Unter Create Programming File -> Launch JTAG Programmer.
9. Nach der Programmierung und Verifizierung ist der Baustein Einsatzbereit.

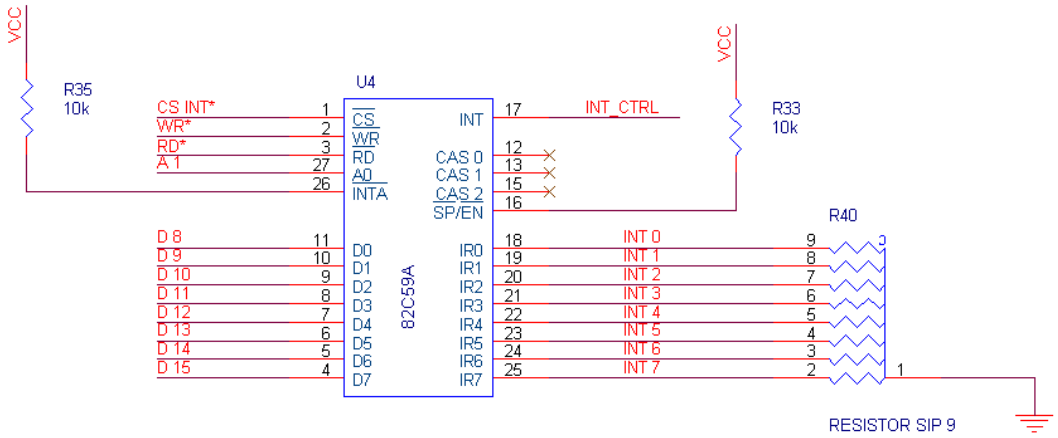
Das VHDL-Sourcefile ist unter cs_and_beep.vhd für das VHDL-Webpack oder unter cs_and_beep.txt für das Öffnen mit Texteditor zu finden.

8 Interruptcontroller

8.1 Allgemeines

Auf der Trägerplatine befindet sich ein Interruptcontroller Typ 82C59 von Intel. Aufgrund der verschiedenen Architekturen können Intel- und Motorola –Produkte nicht beliebig kombiniert werden. Der Datenaustausch ist , da die standardisierten Routinen der beiden Hersteller nicht identisch sind, nur über gewöhnliche Read- und Write-Zyklen möglich. Der zweite Grund weshalb die Interruptabarbeitung nicht über Standardzyklen stattfinden kann liegt daran, dass das MEGA332-Board die für die Interrupt-Acknowledge-Zyklen benötigten Signale gar nicht zur Verfügung stellt.
Der Interrupt-Controller selbst kann 8 Prioritätsebenen unterscheiden. Intern besitzt er zwei Register, das Interruptrequest Register und das Inservice Register.

8.2 Anschlussschema :

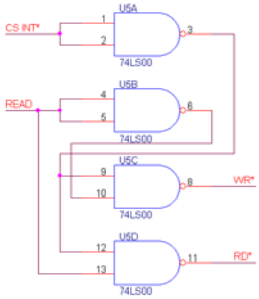


Grafik 8.2-1:Anschlussschema

Da der Controller auf die von Intel standardisierten Anschlüsse ausgelegt ist, sind die Interrupteingänge im High-Zustand aktive. Unglücklicherweise arbeiten Motorola Controller mit Interrupteingänge welche im High-Zustand als inaktiv detektiert werden. Das bedeutet, dass die Signale zwischen den beiden Controllern invertiert werden müssen.
INT0 bis INT7 sind die 8 nach Priorität geordneten Eingänge. INT7 hat dabei die höchste und INT0 die niedrigste Priorität. Pin 17 (**INT_CTRL**) ist der Signalisierungsausgang zum MEGA332-Board auf **IRQ4***. Das Chipselect Signal CS_INT* wird vom PLD direkt aus den Adressen **0x0ee0050** und **0x0ee0052** erzeugt. Das Interuptbestätigungssignal INTA* wird vom MC68332 nicht erzeugt. Deshalb ist kein echter Bestätigungszyklus möglich. WR* und RD* werden bei einem Zugriff aus dem READ Signal des MEGA332-Boardes erzeugt.

Xilinx →

MEGA 332 →

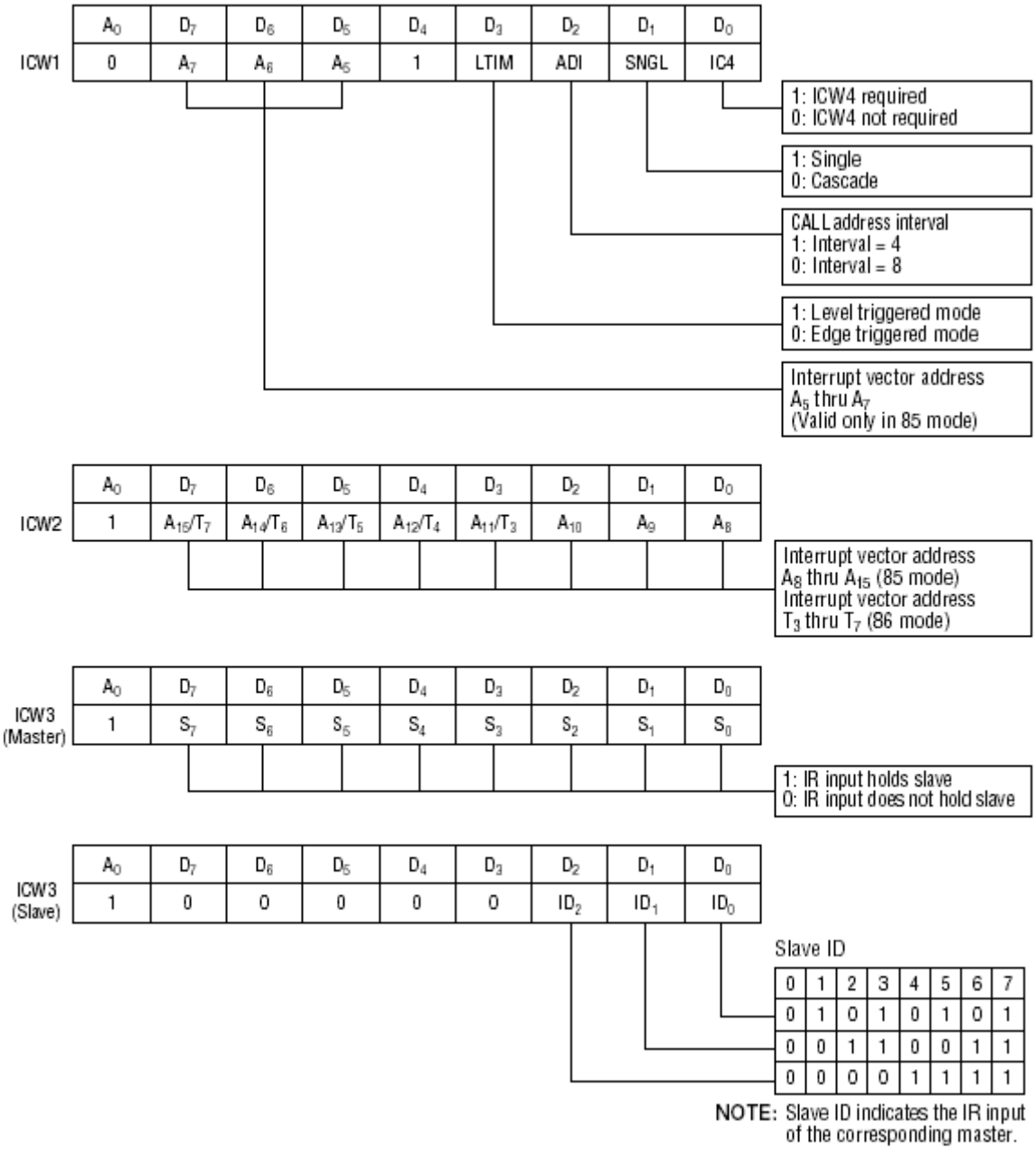


← Interruptcontroller

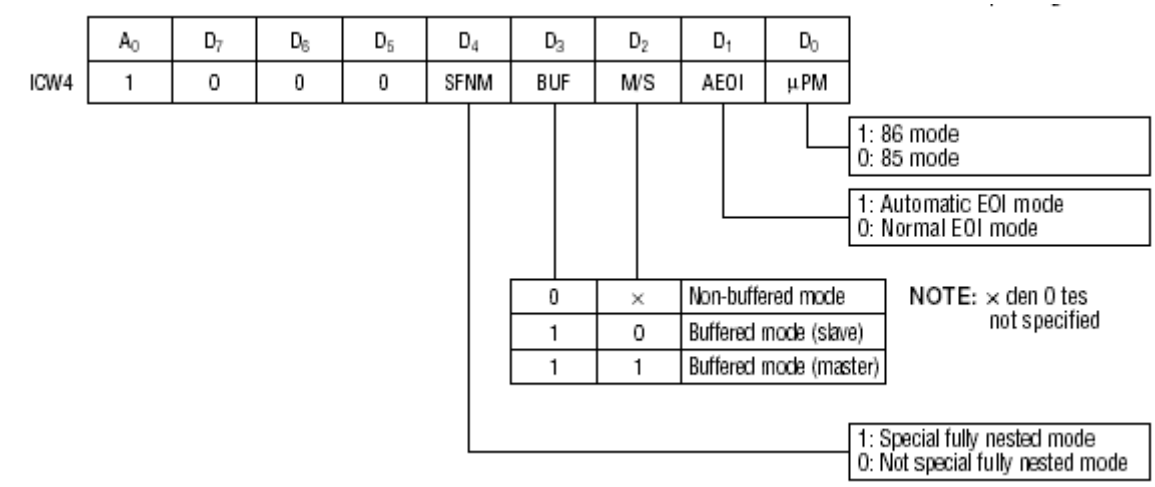
Grafik 8.2-2 :Read- und Writesignale

8.3 Initialisierung

Für die Initialisierung des Controllers sind normalerweise 4 Initialisation Words (ICW1- 4) notwendig. Sie werden hintereinander an den Controller übermittelt. Für die Single-Mode Betriebsart werden nur ICW1, ICW2 und ICW4 benötigt. Für den Einsatz auf der Trägerplatine muss der Controller im Single Modus initialisiert werden. Die Initialisierungsworte sind wie folgt aufgebaut:

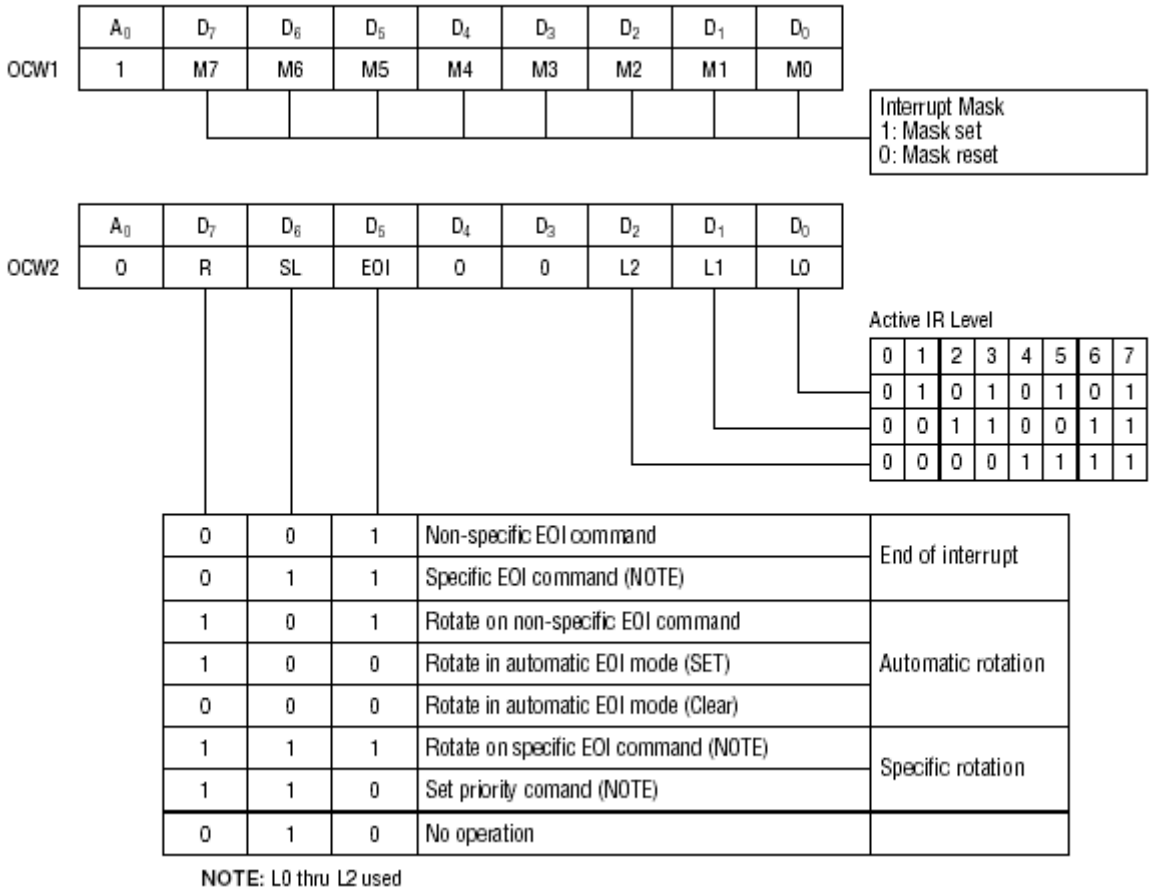


Grafik 8.3-1 : Initialisierung ICW1...3

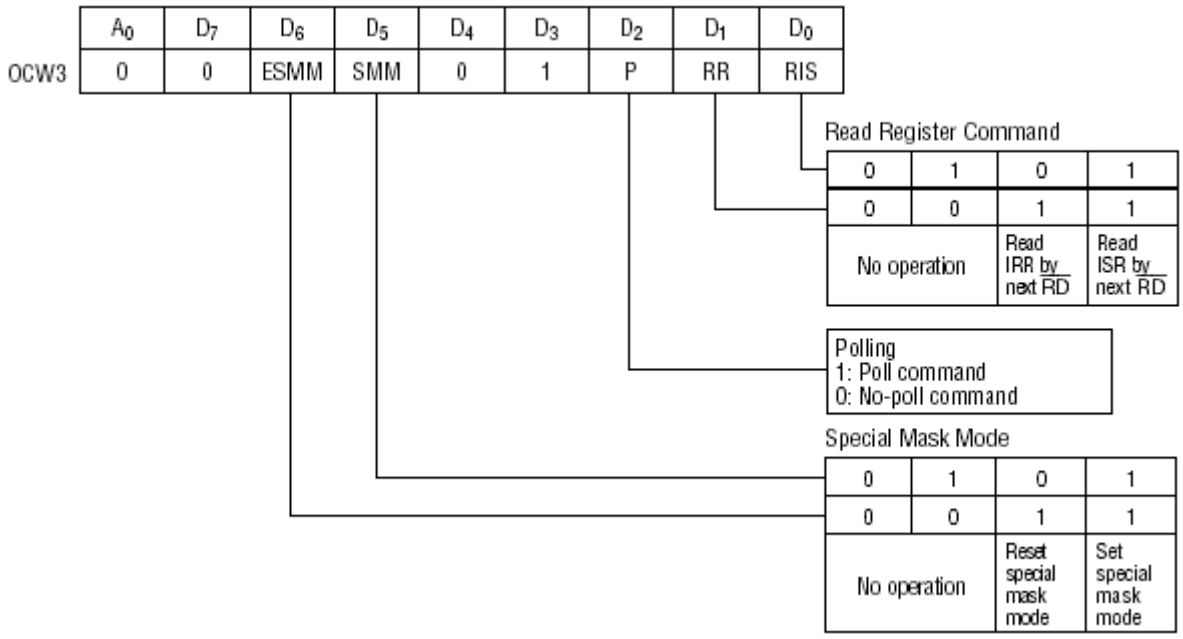


Grafik 8.3-2 : Initialisierung ICW4

Des weiteren sind noch folgende Operation Command Wörter OCW1 – 3 vorhanden:



Grafik 8.3-3 Betrieb OCW1..2



Grafik 8.3-4 Betrieb OCW3

Bevor ein Interrupt richtig erkannt werden kann muss die Interrupt Maske gesetzt und der Interruptpriority Level bestimmt werden. Mittels OCW1 können die Bits in der Interrupt Maske gesetzt oder gelöscht werden. Ein gesetztes Bit bedeutet, dass der Interupteingng deaktiviert ist. Als zweites muss der aktuelle Interruptpriority Level gesetzt werden. Dies geschieht mit den Bits 0-2 vom OCW2. Wenn der Level auf 0 gesetzt ist reagieren alle Interrupt Eingänge, für den Level 3 z. B. reagieren nur die Interrupts von 3 an aufwärts.

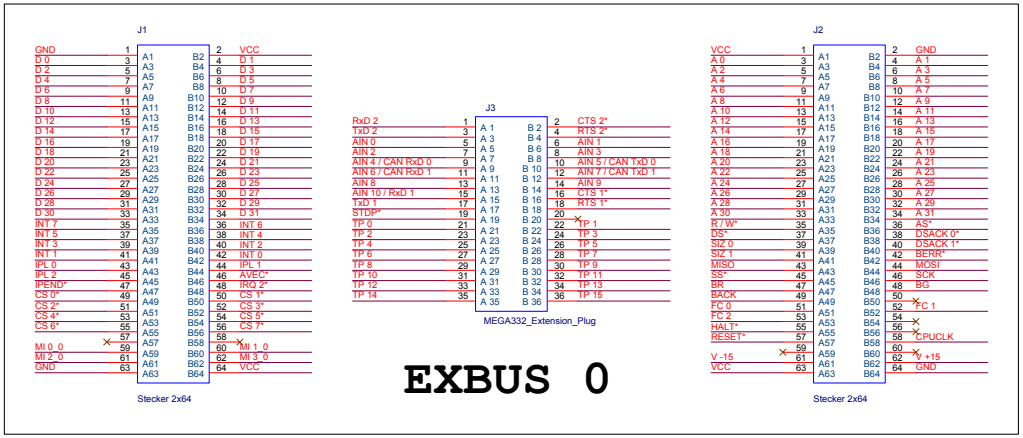
8.4 Interrupt Ablauf :

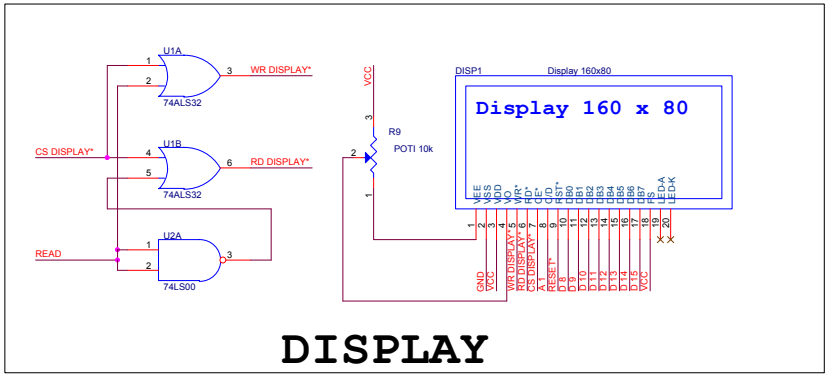
Sobald der Interruptcontroller initialisiert ist, können Interrupts erkannt und vom MEGA332 Board verarbeitet werden. Wenn ein Interrupt ausgelöst wird, setzt der Controller in seinem IRQ Register das entsprechende Bit und signalisiert dem MEGA332 Board über die IRQ4* Leitung , das ein Interupt anliegt. Sobald ein Interrupt via IRQ4* angezeigt wird, holt die Interruptserviceroutine des MEGA332 Boardes (Queltext im Anhang) das IRQ Register und das ISR Register des Interruptcontrollers. Mit diesen Informationen kann die CPU nun auf jeden der 8 Interuptleitungen reagieren. Die Priorisierung geschieht in diesem Fall auf Softwareebenen.

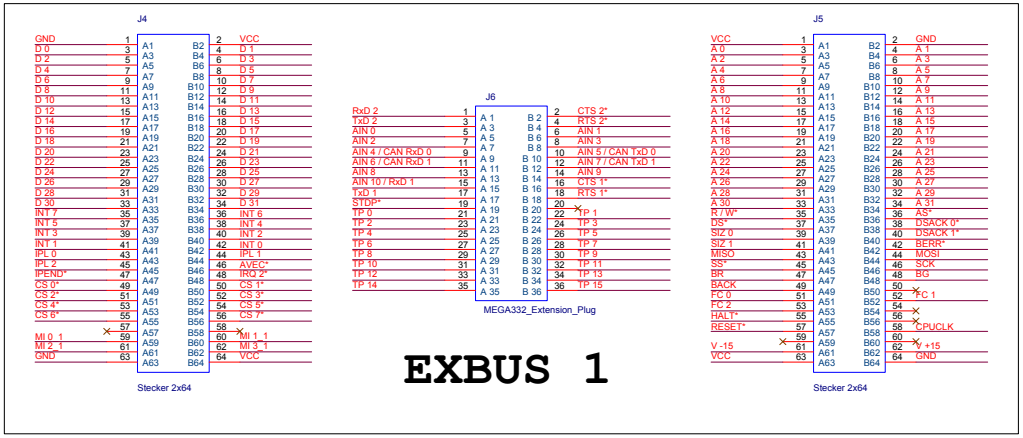
Schritt für Schritt :

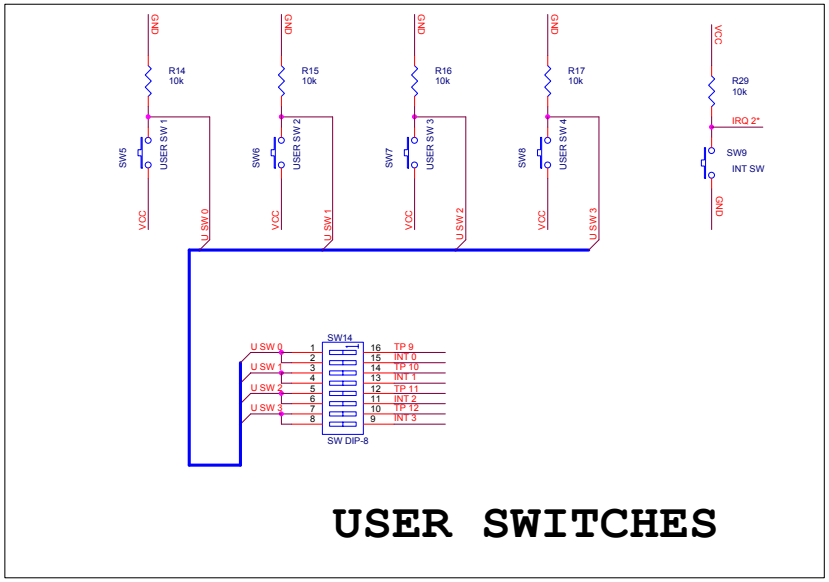
1. Interupt wird ausgelöst
2. Der Interruptcontroller setzt entsprechendes Bit im IRR Register und im ISR Register. Danach signalisiert er dem MC68332 Controller dass ein Interrupt anliegt indem er das IRQ4* Signal auf LOW zieht.
3. CPU holt sich Register IRR und ISR aus dem Controller
4. CPU arbeitet entsprechende ISR ab.

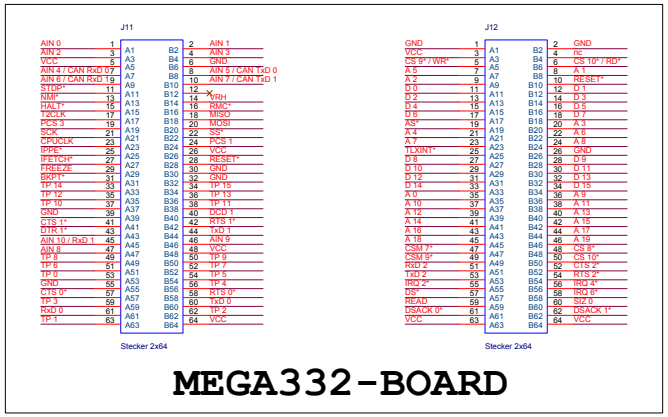
9 Schemas & Layout



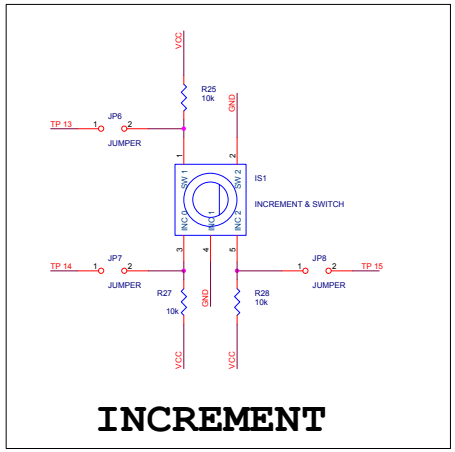




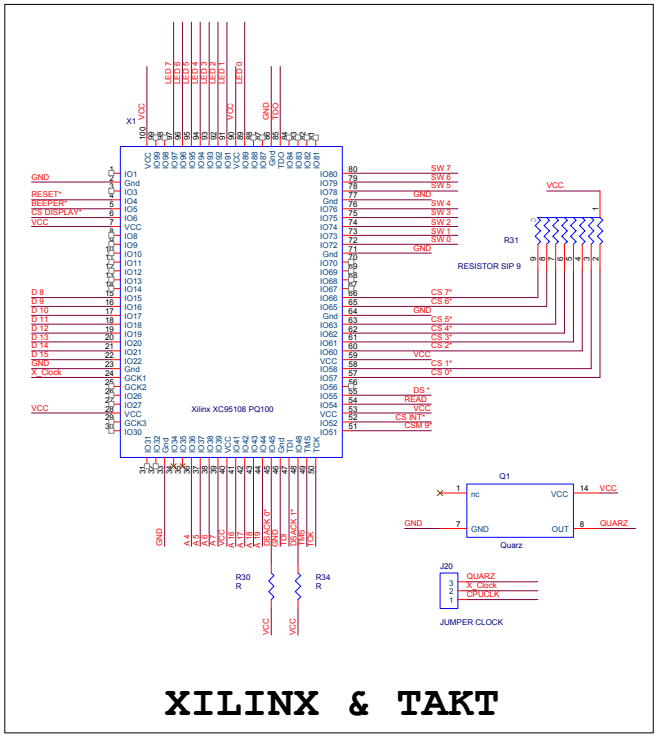




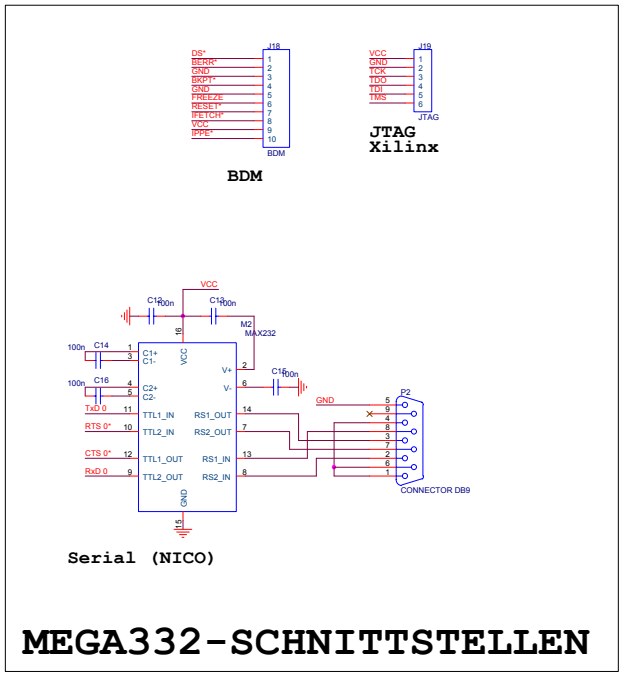
MEGA332-BOARD



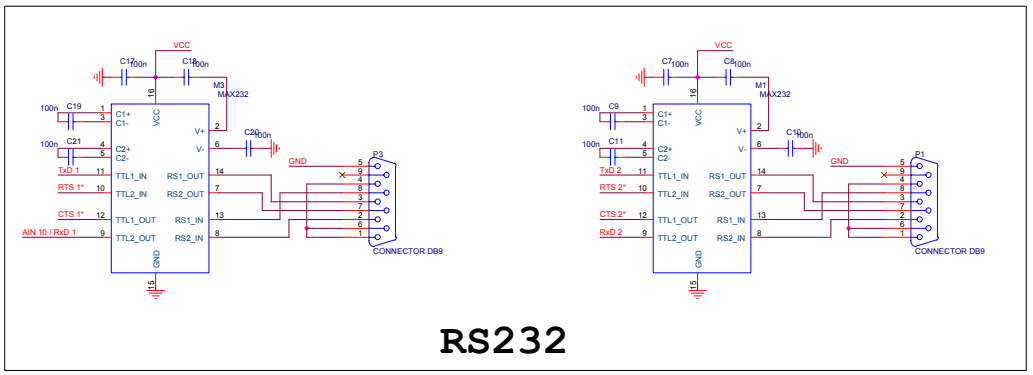
INCREMENT



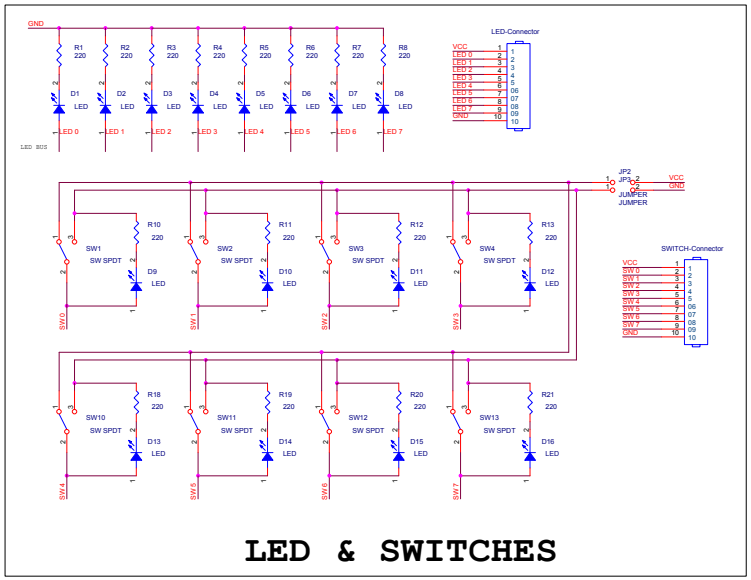
XILINX & TAKT

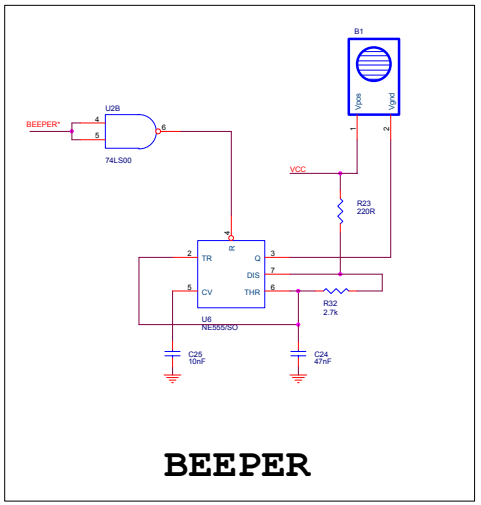


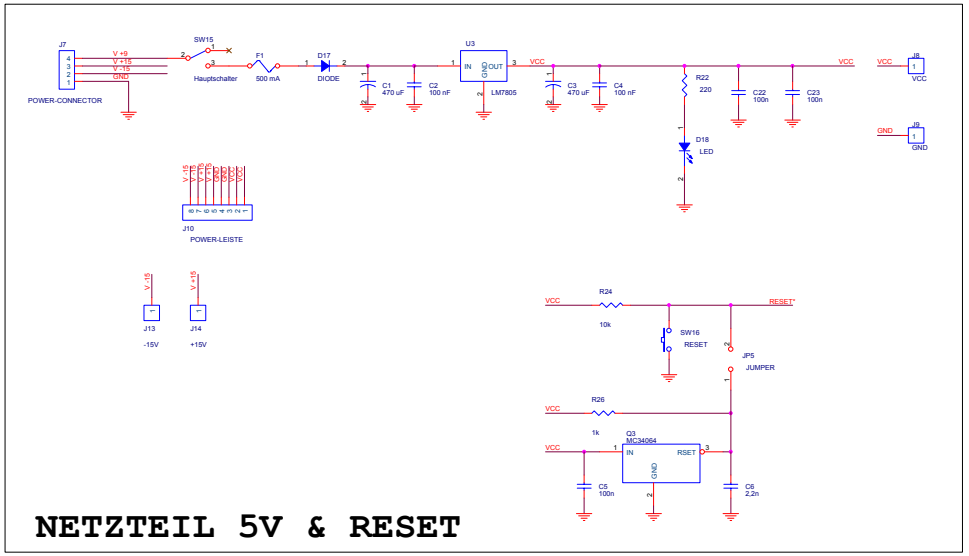
MEGA332-SCHNITTSTELLEN

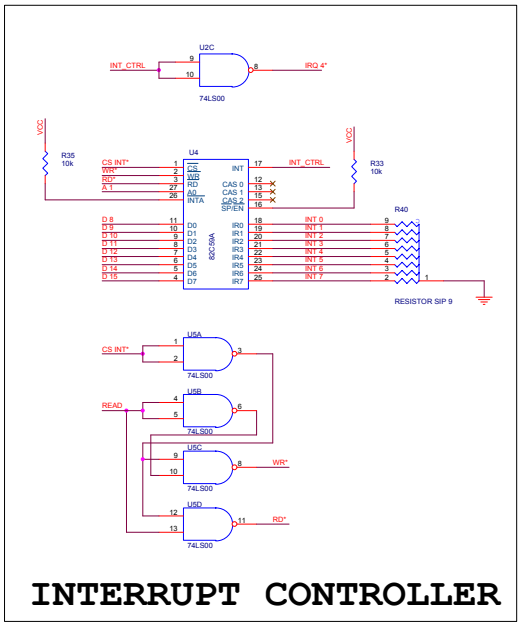


RS232









10 Module

10.1 Allgemein

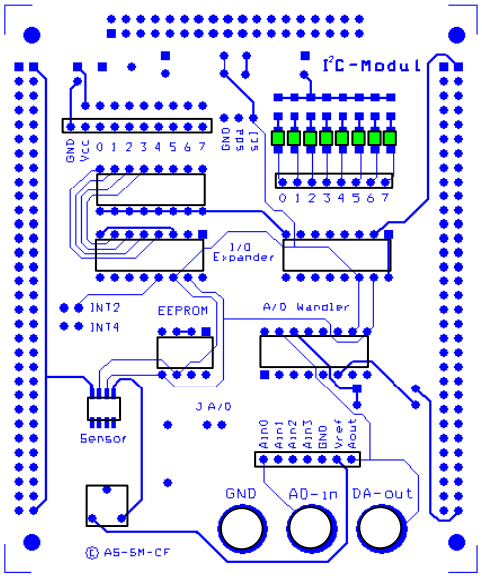
Für die Trägerplatine waren verschiedene Module zu entwerfen. Die Modul-Prints haben eine Abmessung von 80x100mm was gerade der grösse des MEGA-Boards entspricht. Die Belegungen der beiden Steckerlisten sind durch die Definition des Ex_Bus-Stecker gegeben.

11 I²C-Modul

11.1 Allgemeine Beschreibung

Das I²C-Modul wurde mit den folgenden Bausteinen bestückt:

- AD/DA - Wandler
- Zwei 8-Bit I/O-Expander
- EEPROM
- Feuchtigkeits- und Temperatursensor



Grafik 11.1-1 I²C -Print

Mit diesem Modul kann in einfachen Schritten die Funktionsweise der I²C-Komunikation beigebracht werden. Mit den oben aufgeführten Bausteinen lassen sich aber auch andere Anwendungen üben. Beispielsweise ist der Zugriff auf das EEPROM standardmässig, d.h. der zeitliche Ablauf um Daten zu schreiben und zu lesen ist bei den meisten anderen EEPROM's derselbe. Die I/O-Expander sind zum Einlesen und Ausgeben von 8-Bit, in paralleler Form, ideal. Um mit analogen Werten zu arbeiten, ist der AD/DA-Wandler ein interessanter Baustein. Mit einem Poti kann auf einem Kanal ein Analogwert vorgegeben werden. Es können aber auch vier unterschiedliche Analogwerte auf die Eingänge des AD-Wandlers gegeben werden, die differentiell bearbeitet auch gleich auf den einzigen Analogausgang gegeben werden können. Das faszinierende am Feuchtigkeits- und Temperatursensor ist die kleine Bauweise und die Genauigkeit der gemessenen Werte.
Weiter werden die einzelnen Bausteine kurz beschrieben, wobei bei spezifischen Anwendungen immer noch das Datenblatt studiert werden sollte.

11.2 I²C-Bus

Der I²C Bus von Philips ist ein sehr einfacher Seriellbus, der nur zwei Leitungen besitzt, SDA (seriel data) und SCL (seriel clock line). Er wurde vor beinahe 20 Jahren entwickelt und hat sich bis heute weltweit etabliert. Er wird in verschiedensten Anwendungen im Rahmen von eingebetteten Systemen, wie zum Beispiel in Telekommunikations-Applikationen als Kontroll-, Diagnose- und Energiemanagementbus, eingesetzt. Obwohl der I²C Bus nur zwei Leitungen besitzt, ist er multimasterfähig. Die genauere Funktionsweise des I²C Busses kann aus den Datenblättern auf der CD entnommen werden.

11.3 AD/DA-Wandler

Wir haben den Baustein PCF8591P von Philips verwendet. Dieser 8-Bit-Wandler, besitzt vier A/D-Kanäle und einen D/A-Kanal. Die A/D-Kanäle können einzeln oder differentiell eingelesen werden. An den ersten Eingang ist ein Poti über einen Jumper angeschlossen. Mit diesem Poti können Analogwerte von 0 bis 4.7Volt eingestellt werden. Das zu wandelnde Spannungssignal sollte die Referenzspannung von 4.7Volt nicht überschreiten.
Der Wandler hat drei Hardwareadresspins A_0...A_2. Dabei setzt sich das Adressbyte aus vier fix vorgegebenen Bits und den Adresspins zusammen:

S	1	0	0	1	A2	A1	A0	R/$\overline{W}$	A

Grafik 11.3-1 : Adressbyte AD-DA-Wandler

11.4 8-Bit I/O-Expander

Hier wurde der PCF8574AP von Philips eingesetzt. Wie es sein Name schon sagt, kann dieser Expander 8-Bit-Werte ausgeben bzw. einlesen. Die Ausgänge vermögen genug Strom zu liefern um pro Ausgang je eine LED anzusteuern. Aus Sicherheitsgründen haben wir auf dem Modul zwei Expander positioniert, um zu verhindern dass eine Ausgabe unbeabsichtigt mit den Schaltern auf dem Modul verfälscht wird. Dabei dient der eine Expander nur als Eingangsbaustein und der andere nur als Ausgabebaustein. Der Eingabebaustein lässt sich entweder über die DIL-Schalterreihe oder über die Anschlusspins des Moduls ansteuern. Die Datensignale des Ausgabebausteins werden auf eine Reihe mit acht LED's sowie auf eine PIN-Reihe auf dem Modul geführt . Die Ansteuerung erfolgt mit negativer Logik, d.h. wenn ein DIL-Schalter ON ist, ist der Eingang auf null und falls ein logischer Ausgang auf null ist leuchtet die jeweilige LED.

Hardwareadresse zum Schreiben: **A_0=1**, **A_1=0**, **A_2=0**

S	0	1	1	1	A2	A1	A0	0	A

Grafik 11.4-1 : Adressbyte (I)/O-Expander

S	0	1	1	1	A2	A1	A0	1	A

Hardwareadresse zum Lesen: **A_0=0**, **A_1=0**, **A_2=0**

Grafik 11.4-2 : Adressbyte I/(O)-Expander

11.5 EEPROM

Das eingesetzte M24C01ist ein Produkt von ST-Thomson. Das 256Byte grosse EEPROM lasst sich ohne grossen Aufwand mit Daten beschreiben. Auch das Auslesen von gespeicherten Daten ist kein Problem. Die Schreib- und Lesenzyklen können entweder von der aktuellen Adresse, oder von einer bestimmter Adresse vorgenommen werden.

Beides erlaubt ein mehrfaches schreiben oder lesen, wobei nach jedem Zugriff die aktuelle Adresse inkrementiert wird.
Die Adressleitungen A_0...A_2 sind wie folgt gesetzt: **A_0=0**, **A_1=0**, **A_2=0** .
Das Adressbyte
lautet:

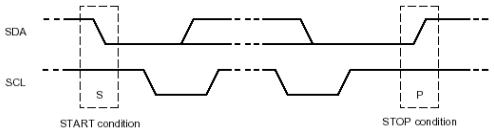
S	1	0	1	0	A2	A1	A0	1	A

Grafik 11.5-1: Adressbyte EEPROM

11.6 Feuchtigkeits- und Temperatursensor

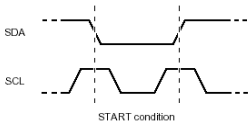
Der verwendete SHT11 Baustein von Sensirion ist ein Singlechip Sensor, der die relative Luftfeuchtigkeit und die Umgebungstemperatur misst. Er arbeitet mit einem leicht modifiziertem I²C- Signal. Der Unterschied zum I²C besteht darin, dass er eine andere Startkondition voraussetzt und kein Stopsignal benötigt. Die restliche Kommunikation verläuft nach dem I²C-Standard.

Standard I²C Signal:



Grafik 11.6-1 : Start/Stop im I²C-Signal

Modifizierte Startkondition des Sensors:



Grafik 11.6-2 : Startkondition des Sensors

Die Temperaturmessungen sind bis zu einer 14-Bit Auflösung möglich, Feuchtigkeits-Messungen bis zu 12-Bit. Der Fehler bei der Feuchtigkeitsmessung liegt bei maximal ±5%. Bei der Temperatur gilt eine Toleranz von ±1°C.
Bei der Ansteuerung des Sensors wird beim ersten Byte ein Command mitgeschickt. Mit diesem Befehl erfolgt die Auswahl der verschiedenen Modi, wie z.B. Temperaturmessen oder einen Reset durchführen. Der Sensor verfügt noch über einige andere Features die aus dem Datenblatt zu entnehmen sind.
Das Adressbyte des Sensors enthält zugleich den Command (C1 – C4). Der Command ist ebenfalls aus Datenblatt zu entnehmen.

Somit lautet das Adressbyte:

‚S'	0	0	0	C0	C1	C2	C3	C4	A

Grafik 11.6-3 : Adressbyte mit Command für den Sensor

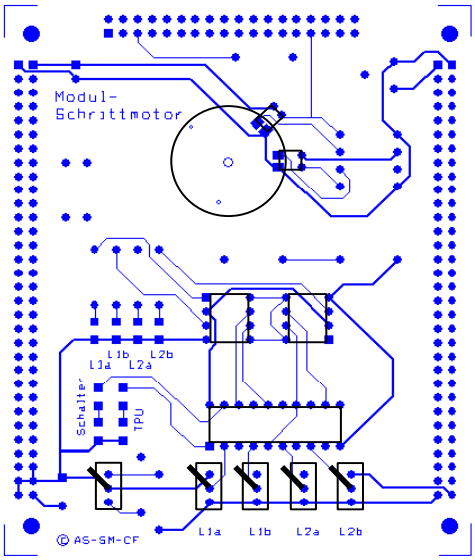
11.7 Entwicklung

Die wichtigsten Schritte beim Entwickeln des I²C –Moduls waren das evaluieren der einzelnen Bausteine, die Festlegung der Modulgrösse (100x80mm), das Layouten und das Testen des Moduls. Am längsten hat sicher das Layouten gedauert, da es das erste Modul war das entwickelt wurde.

12 Schrittmotor Modul

12.1 Allgemeine Beschreibung

Mit dem Schrittmotormodul haben wir das zweite Modul gefertigt. Auf diesem Modul sind ein kleiner Schrittmotor und zwei SMD-Gabellichtschranken montiert. Der Schrittmotor ist von Hand oder mit der TPU des Mega332 ansteuerbar. Zur Handsteuerung sind Kippschalter vorhanden. Damit die Schalter die TPU nicht beeinflussen können, ist ein 2*4Bit TTL-Treiber eingebaut, der wahlweise über einen Kippschalter, die TPU oder die Schalter treibt. Als optische Unterstützung sind die vier Steuersignale für den Schrittmotor auf LED geführt. Ebenfalls wird die jeweils ausgewählte Steuerquelle mit einer LED angezeigt.



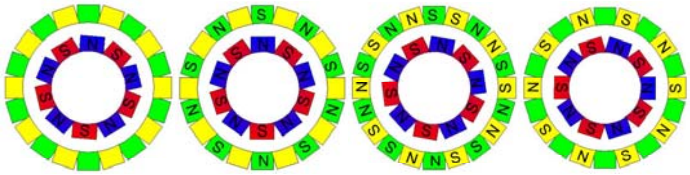
Grafik 12.1-1 Schrittmotor-Print

Dem Schrittmotor wurde eine Codierscheibe aufgesteckt, um mit den zwei Lichtschranken die Drehrichtung und die Umdrehungszahl des Motors messen zu können. Die Signale der Lichtschranken wurden wiederum auf einen TP gelegt. Zusätzlich hat es noch ein Potentiometer, mit dem eine Analogspannung zwischen 0 und 5 Volt auf den Ain0 Anschluss des Mega332 gegeben werden kann.

12.2 Schrittmotor

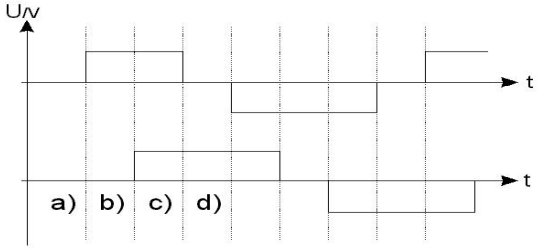
Der verwendete Motor ist ein Permanentmagnet-Schrittmotor, genauer genommen ein Klauenpol-Motor, mit zwei Statorspulen. Klauenpol-Motoren haben einen kleineren Schrittwinkel als einfache Permanentmagnet-Motoren und sind zudem billig herzustellen. Unser Motor macht eine Drehung von 18° pro Vollschritt und 9° pro Halbschritt. Dies entspricht 20 Voll- und 40 Halbschritten pro Umdrehung.
Der Motor ist folgendermassen aufgebaut: Er hat zwei Spulen, die je zehn „Zähne“ besitzen. In der Grafik 12.2-1 sind die Zähne der einen Spule gelb und die der anderen grün abgebildet. Wenn Strom in der Spule fliesst, werden die Zähne der jeweiligen Spule abwechslungsweise magnetisiert (Bild b)).
In der Grafik 12.2-1 ist der Ablauf eines Schrittes im Uhrzeigersinn aufgezeichnet. Im Bild a) fliesst in keiner Spule Strom und der *Anker* ist dadurch noch frei beweglich. In b) wird die grüne Spule durchflossen, was den Anker ausrichten lässt. c) zeigt den Halbschritt, bei dem in beiden Spulen ein Strom fliesst. Der Anker richtet sich dabei nach den

Magnetfeldern beider Statoren aus. Der Drehsinn ist, wie auch beim Vollschritt, durch die Polarisierung der Statoren gegeben. Im Bild d) ist die grüne Spule wieder ausgeschaltet und nur die gelbe Spule aktiv. Dar Anker richtet sich also nach dem gelben Stator aus. Bei einem Vollschritt wird der Schritt c) ausgelassen.



Grafik 12.2-1 a) b) c) d)

Passend zur Grafik 12.2-1 ist der Spannungsverlauf an den Spulen in der Grafik 12.2-2. Der Spannungsverlauf ist für Halbschritte aufgezeichnet. Bei Vollschritten sieht der Spannungsverlauf sehr ähnlich aus. Der einzige Unterschied besteht darin, dass an den Spulen nie gleichzeitig eine Spannung liegt.



Grafik 12.2-2 Spannungsverlauf an den Spulen

12.3 Entwicklung

Das Schrittmotormodul war das zweite Modul, das wir fertigten. Die anfänglichen Probleme beim Layouten fielen also weg, dafür gab es mehr elektronischen Aufwand. Erste Versuche den Schrittmotor mit einem einfachen TTL-Treiberbaustein anzusteuern schlugen fehl, da der Treiber zu schwach war. Dieses Problem war mit einer OP-Treiberstufe schnell behoben.
Der gebrauchte Schrittmotor hatte das CT-Labor mal als Geschenk in sehr hoher Stückzahl bekommen. Die Anschlüsse des Motors sind mit einem Flexiprintkabel herausgeführt, was uns dazu veranlasste eine passende Buchse zu finden. Eine solche Buchse zu finden war nicht ganz einfach, weshalb wir auf Herrn Naef's Hilfe zurückgreifen mussten.
Was wir bis anhin noch nicht finden konnten war eine passende Codierscheibe für den Encoder. Für das Modul haben wir deshalb selber eine Scheibe angefertigt, die wir auf die Welle des Schrittmotors aufstecken konnten.

13 DCF77 Empfänger-Modul

13.1 Allgemeine Beschreibung

Der DCF77, oder Funkuhrempfänger wurde als fertige Einheit gekauft und auf das Modul gepasst.
Der DCF77-Sender ist der in Zentraleuropa einzige zuverlässig und mit einfachen Mitteln empfangbare Zeitzeichensender (der Aufwand zum Empfang der von GPS-Satelliten ausgesendeten Normalfrequenz ist ungleich höher und teurer).
Die gesendete Zeit ist für die Bundesrepublik Deutschland gültig. Die Zeitnormale wird von der Physikalisch-Technische Bundesanstalt bereitgestellt.
Der Sender DCF77 steht in Hessen bei Mainflingen (nahe Frankfurt am Main) und ist aufgrund seiner zentralen Lage mit einer Sendeleistung von 50 kW und einer abgestrahlten Leistung von 30 kW im Umkreis von ca. 2000 km gut zu empfangen:

Grafik 13.1-1 Empfangsradius des DCF77

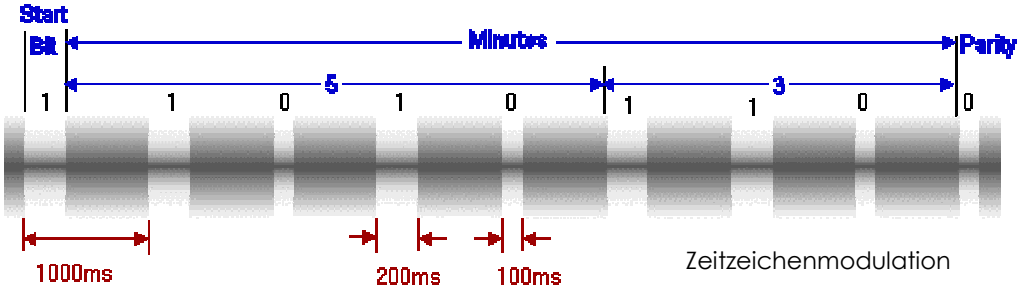
Die Trägerfrequenz beträgt 77,5 kHz (daher auch der Name 'DCF**77**') und ist aus einem Atomfrequenznormal abgeleitet. Ihre relative Abweichung vom Nennwert beträgt im Mittel über 100 Tage weniger als $2 \cdot 10^{-13}$ Sekunden. Frequenzschwankungen sind also nur aufgrund der für diesen Frequenz-Bereich (Langwelle) typischen ausbreitungsbedingten Phasenschwankungen des Trägers zu erwarten. Damit ist die Trägerfrequenz auch zur Synchronisierung von Frequenzoszillatoren einsetzbar.
Die für uns interessanten Informationen ist die binärcodierte DCF77 Zeitangabe.
Zum Empfang und zur Umwandlung des Trägersignals in für Computer nötige Digitalsignale benötigt man einen DCF77-Empfänger

13.2 Der Zeitcode

Das vom DCF77-Empfänger gelieferte digitale Signal kann nun von einem Programm ausgewertet werden. Die unterschiedliche Dauer der Sekundenmarke von 100 ms oder 200ms wird dazu benutzt, im BCD-Kode Uhrzeit und Datum zu übertragen. Dabei entsprechen Sekundenmarken mit einer Dauer von 100 ms der logischen 0 und solche mit einer Dauer von 200 ms der logischen 1.
Insgesamt werden also in einer Minute 58 (oder 59) einzelne 0 und 1 Bits übertragen, welche die Uhrzeit und das Datum der folgenden Minute beinhalten.
Grafik 13.3-1 zeigt das Schema der Kodierung und die Zuordnung der einzelnen Sekundenmarken auf die übertragene Zeitinformation (die Wertigkeit der logischen Informationen).
Als Beispiel für die Kodierung bei DCF77 ist in Grafik 13.3-1ein Ausschnitt der Modulation des Trägers für eine Minute dargestellt.

13.3 Trägermodulation

Ausschnitt Trägermodulation für die Minute 35:
In der hochstabilen Normalträgerfrequenz 77,5 kHz des DCF77-Signals sind die Bitinformationen amplitudenmoduliert eingebettet. Ein Bit wird durch Absenkung der Trägerwelle auf 25% Amplitude dargestellt.
Grafik 13.3-1 zeigt den Ausschnitt einer Trägermodulation in der 34. Minute (für die 35. Minute) mit vorherigem Startbit und Minutenparitätsbit. Die Wertigkeit der Bits sind in der Zeitcode Tabelle aufgeführt:

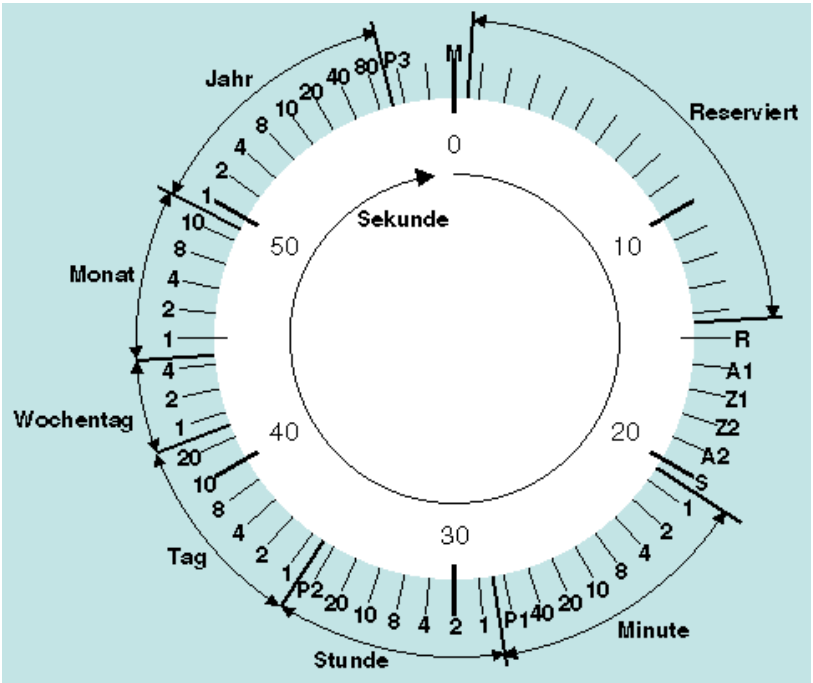


Grafik 13.3-1 Zeitzeichenmodulation

13.4 Zeitcode Schema

Schema der kodierten Zeitinformation

Grafik 13.4-1 Zeitcode-Schema



13.5 Zeitcode Tabelle

Die Bedeutung der einzelnen Bits im Bit-Informationsfenster:

Sekunde	Bez.	Wert	Pegel	Bedeutung
0	M		0	Minutenanfang
1..14			0	Reserviert
15	R		?	Reserveantenne aktiv
16	A1		?	Ankündigung Sommerzeit
17	Z1		?	Zeitzonenbit 1
18	Z2	2	?	Zeitzonenbit 0
19	A2	1	0	Schaltsekunde
20	S		1	Zeitkodierung Startbit
21		1	?	Minute Einer
22		2	?	" "
23		4	?	" "
24		8	?	" "
25		10	?	Minute Zehner
26		20	?	" "
27		40	?	" "
28	P1		?	Prüfbit 1
29		1	?	Stunde Einer
30		2	?	" "
31		4	?	" "
32		8	?	" "
33		10	?	Stunde Zehner
34		20	?	" "
35	P2		?	Prüfbit 2
36		1	?	Tag Einer
37		2	?	" "
38		4	?	" "
39		8	?	" "
40		10	?	Tag Zehner
41		20	?	"
				"
42		1	?	Wochentag Einer
43		2	?	" "
44		4	?	" "
45		1	?	Monat Einer
46		2	?	" "
47		4	?	" "
48		8	?	" "
49		10	?	Monat Zehner
50		1	?	Jahr Einer
51		2	?	" "
52		4	?	" "
53		8	?	" "
54		10	?	Jahr Zehner
55		20	?	" "
56		40	?	" "
57		80	?	" "
58	P3		?	Prüfbit 3
59	-	-	-	Sync (Kein Impuls)

Tabelle 13.5-1 : Zeitcode-Tabelle

13.6 Entwicklung

Für das Modul brauchte man einen fertigen Empfänger richtig zu beschalten und dessen Signal zu verstärken. Unglücklicherweise hat das Modul im CT-Labor nicht genügend Empfang, weshalb für den späteren Einsatz im Unterricht eine andere Version entwickelt werden müsste. Dabei wäre es möglich ein einziger Empfänger an einer geeigneten Stelle zu positionieren und das Signal über eine Infrarotschnittstelle weiter zu senden.

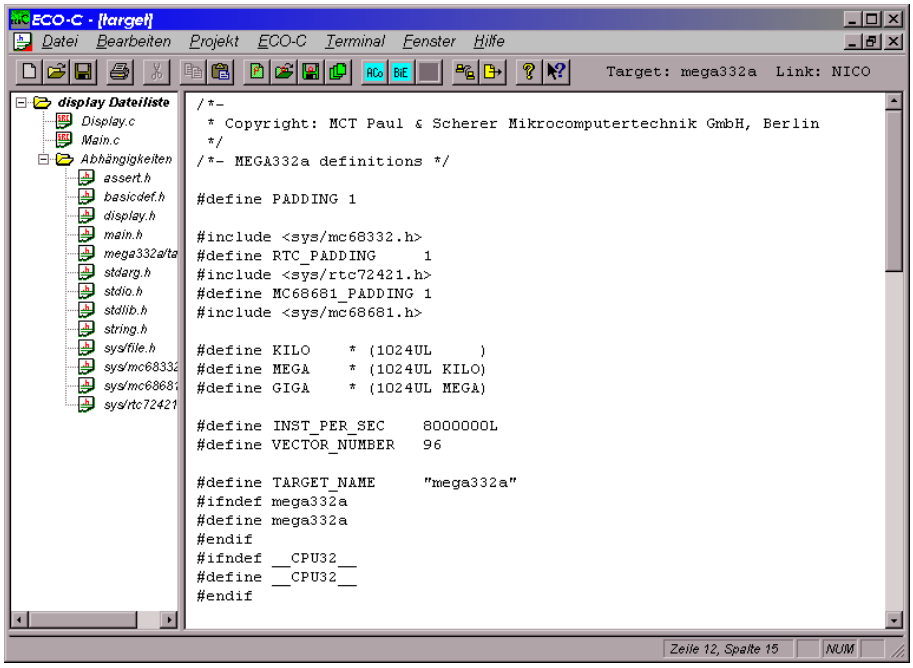
14 MCT Entwicklungsumgebungen

14.1 Allgemeines

Bis anhin erfolgte die Programmierung in den CT-Übungen mit dem beim Kauf der MEGA-Platine mitgelieferten Programmierumgebung ECO C . Auch bei Semester- und Diplomarbeiten, bei welchem der MEGA zum Einsatz kam, diente der ECO C als Entwicklungsumgebung. ECO C ist im Vergleich zu anderen auf dem Markt erhältlichen Entwicklungsumgebungen wie z.B. CodeWarrior oder µVision schlechter ausgestattet. Diese Tatsache gab den Ausschlag, für die Ausbildungsplattform ebenfalls eine neue Entwicklungsumgebung anzuschaffen.
Das CT-Lab ist nebst dem ECO C noch im Besitz weiterer Entwicklungsinfrastrukturen für den MC68332 Controller. So zum Beispiel der sog. PC-Fant und ein auf MS-DOS basierendes Softwarepaket mit dem Namen EDB. Diese stammten ebenfalls aus dem gleiche Haus wie das MEGA-Board. Die Schule hatte dies jedoch für den Einsatz in Kombination mit dem „SCOTTY“, einem weiteren Board von MCT, gekauft. Ein etwas jüngeres Produkt welches ebenfalls zur CT-Lab Inventar gehört, ist ein HI-WAVE Entwicklungspaket für CPU32 Controller welches in Kombination mit einem Adapter mit dem Targetboard kommunizieren kann.

14.2 WinECO-C mit Targetmonitor

Mit der Hilfe von WinECO-C, der bisherigen Entwicklungsumgebung für den MEGA332 können Projekte mit Quell-Code erstellt und bearbeitet werden.



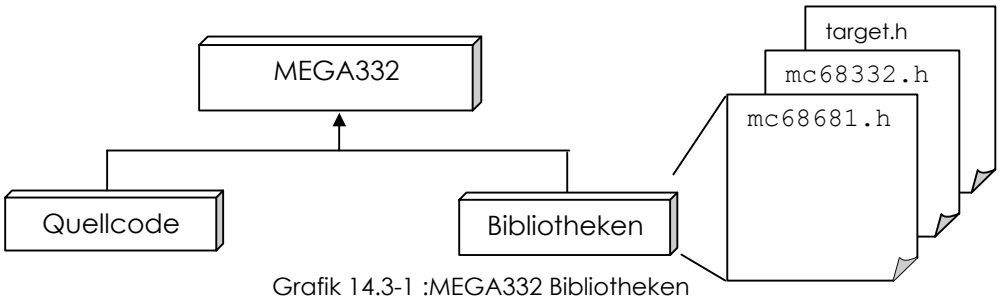
Grafik 14.2-1:WinECO-C

Ein 68k GNU-Compiler sorgt für die Erstellung des ausführbaren Programms.
Der NICO-Targetmonitor sorgt für die Verbindung zwischen dem Host-PC und der Experimentierplatine. Der NICO ist eine Applikation, die sich im ROM des Targetboard befindet und dient als Schnittstelle zum Host-PC. In den Übungen kam es schon öfters vor, dass während dem Herunterladen des Programms irrtümlicherweise die Resettaste betätigt wurde. Die folge war der Verlust des Monitors d.h. NICO wurde überschrieben.

Der Monitor musste in der Folge wieder neu ins Flash geladen werden. Anwendungen lassen sich mit dem Monitor starten und stoppen. Im weiteren stehen diverse andere Optionen zur Verfügung um Applikationen zu testen.

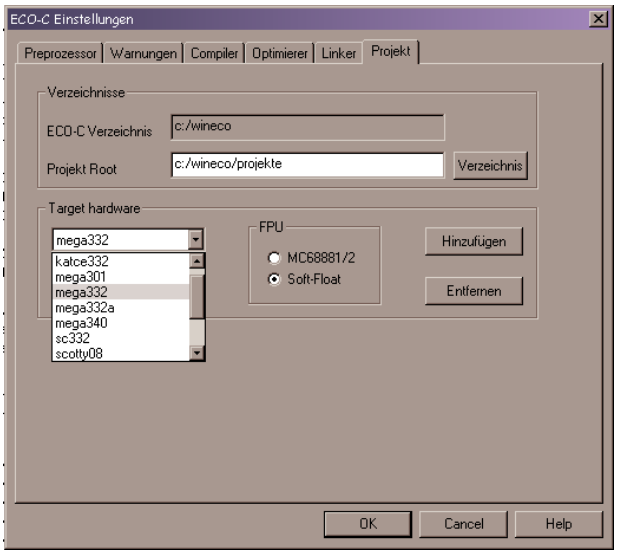
14.3 Bibliotheken

WinECO-C ist eine Entwicklungsumgebung für alle Einplatinencomputer von MCT mit einem Motorola Prozessor. Für die Lauffähigkeit der Boards sind diverse Bibliotheken, die jeweils zu eigenen Anwendungen dazugelinkt werden, notwendig. Dies sind in erster Linie Definitionen der im Controller und der Peripherie vorhanden Register in Form von Header-Files.



Grafik 14.3-1 :MEGA332 Bibliotheken

Damit die richtigen Dateien dazugebunden werden ist es wichtig anzugeben mit welchem Board gearbeitet wird.



Grafik 14.3-2 :Targeteinstellungen

Wie bereits erwähnt besteht der MC86332 aus den Komponenten CPU, SIM, TPU und des QSM. Die Programmierung der TPU erfolgt über Registereinträge. Da ein Register bis zu acht Kanäle enthalten kann, lassen sich die einzelne Kanäle nur durch Maskieren konfigurieren.

```
...
/* Beispiel einer Maskierung eines Registers */
register = register && 0x11a1;
register = register || 0x00a1;
...
```

Grafik 14.3-3 : Bitmaskierung

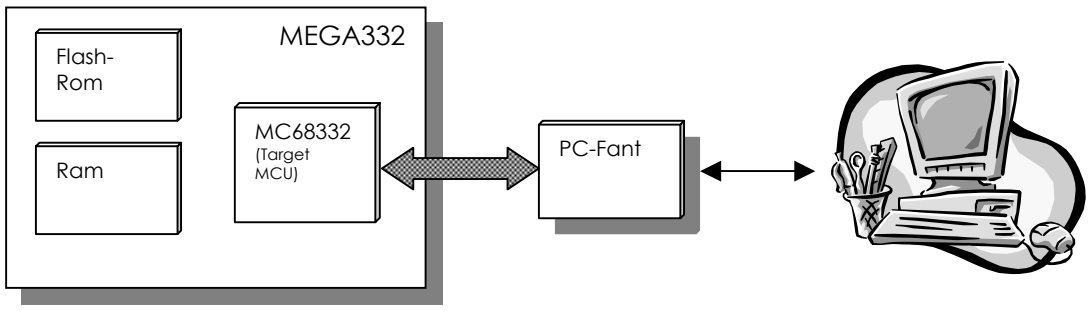
Für das Erstellen von Testprogrammen für die Module wurde ein zusätzliches Header-File mit dem Namen tpu.h geschrieben, bei dem die Register als Bitfelder deklariert sind. Bitfelder ermöglichen den Zugriff auf einzelne Bits innerhalb des Registers und machen den C-Code wesentlich übersichtlicher. Auch lassen sich Konfigurationsänderungen oder Ergänzungen mit zusätzlichen Kanäle ohne grossen aufwand durchführen.

```
...
/************** Single-Bit  Value ******************/
typedef struct
{
    volatile unsigned ch15  :1;
    volatile unsigned ch14  :1;
    ...
    volatile unsigned ch0   :1;
}t_bit1_r0;
...
```

Grafik 14.3-4 : Bitfelder

14.4 Background Debugging Mode mit EDB

Der Background Debugging Modus ermöglicht es, nebst den Visualisierungen sämtlicher Register des Prozessors, den gesamten Speicherbereich oder einzelne Speicherstellen zu beschreiben oder nur zu lesen. Dies dient in erster Linie zum Test von Softwareprogrammen auf der Prozessorplattform.
MCT-Mikrocomputertechnik bietet eine Lösung an, mit der es möglich ist, auf dem MEGA332-Board zu debuggen.



Grafik 14.4-1:EDB

Die Lösung besteht im wesentliche aus dem sog. „PC-Fant“ und der EDB-Treibersoftware. Der „PC-Fant“ wird auf der PC Seite mit einem Parallelkabel verbunden. Der MEGA332 wird vom PC-Fant aus mit einem 8-Poligen Flachkabel angesprochen.
Wir gingen davon aus dass mit Hilfe dieses Werkzeuges sich auch der „Flash-Loader“ für die Verwendung des Nico-Monitors ins Flash-ROM schreiben lässt. Das hat den Vorteil

dass der „Flash-Loader“, falls er irrtümlicherweise überschrieben wurde, mit Hilfe des BDM problemlos wieder geladen werden kann.
Leider hat dies nicht Funktioniert, da nicht alle Signale die der Host-PC benötigt hätte, vom MEGA-Board zur Verfügung standen. Auch der Versuch, die fehlenden Signale Manuell zu überbrücken, war kein Erfolg.

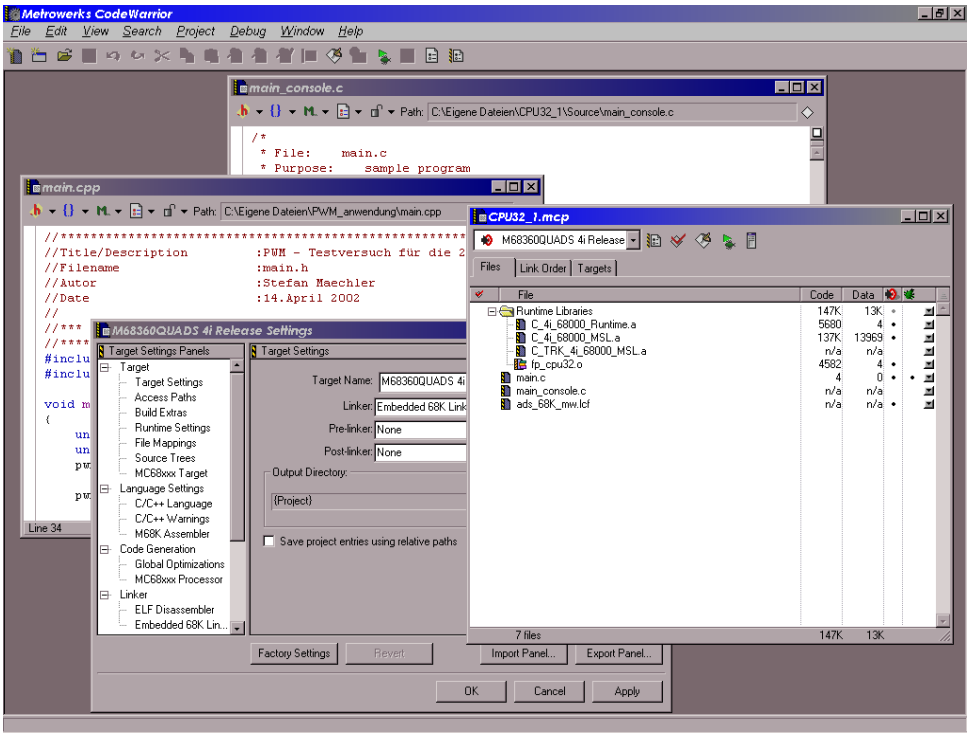
15 CodeWarrior IDE

15.1 Allgemeines

CodeWarrior ist eine in der Industrie weitverbreitete Softwareentwicklungsumgebung für eingebettete Systeme, basierend auf Motorola Controllern. Dies rührt in erster Linie daher, dass Motorola Inhaber der Metrowerks AG in Basel ist. Metrowerks bietet günstige Evauationsboards und Entwicklungssoftwarepakete für Schulen und Studenten an. Durch die Tatsache, dass der MEGA332 auf einem Motorola Produkt basiert, ist es möglich, den CodeWarrior in die neue Entwicklungsumgebung einzubinden.

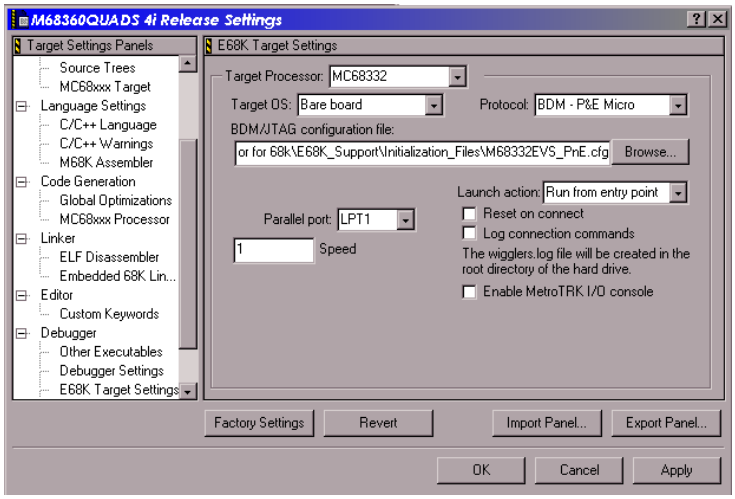
15.2 Source code editor

Der CodeWarrior bietet die Möglichkeit, Projekte zu erstellen und zu bearbeitet. Ein zum CodeWarrior gehörender Source Code Editor ermöglicht das Erstellen und Editieren von C- bzw. C++ Code wie es beim WinECO C ebenfalls der Fall war. Der Projektmanager verwaltet alle zum Projekt gehörenden Quellcode- und Bibliothekenfiles.
Eine weitere Besonderheit ist der *Class Browser* für die objektorientierte Programmierung. Was auf keinen Fall fehlen darf sind der ANSI C/C++ Compiler sowie der *Embedded 68k Linker* für das Erstellen von Anwendungen.
Das Lizenzpaket des CodeWarriors erlaubt die Nutzung der Metrowerks Standard Libraries. Dies sind in erster Linie vorkompilierte C, C++, oder EC++ Bibliothekenfiles die der Entwicklung von Anwendungen auf eingebetteten Systemen dienen.
Die Entwicklungsumgebung unterstütz nebst dem MC68332 Controller eine ganze Reihe von Motorola Controllern und Prozessoren. Für einzelne Evaluaton-Boards mit den entsprechenden Controller sind sogar die ganzen Target-Einstellungen bereits implementiert. Der Benutzer braucht nur noch den Code hinzuzufügen.



Grafik 15.2-1:Projekt-Manager und Editorfenster

Um Fehlern bei der Kompilation oder beim Linken zu beheben, steht eine umfangreiche Hilfe-Bibliothek zur Verfügung. Auch die C/C++ Standards sind in einem Manual beschrieben.
Die durch den Linker erstellte ausführbare Datei lässt sich auf eine Prozessorplatine, in unserem Fall auf den Mega332, herunter laden.
Bevor das herunterladen jedoch möglich ist, müssen im Projekt noch die entsprechenden „Targetboard“ und Linkeinstellungen angepasst werden.



Grafik 15.2-2 : Projekt und Targeteinstellungen

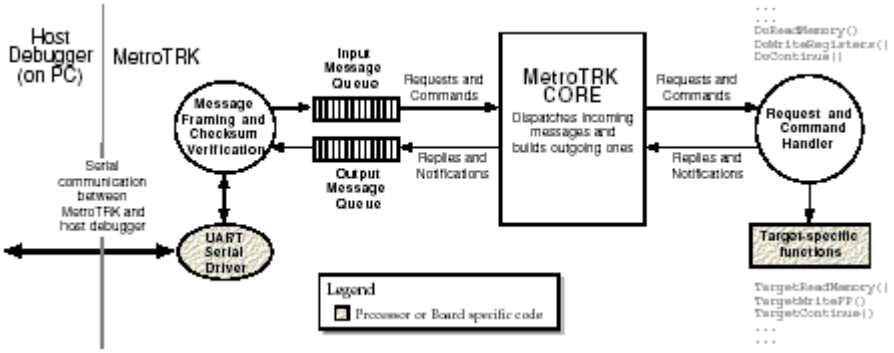
Sind unwissentlich oder versehentlich Konfigurationseinstellungen gemacht worden, besteht jederzeit die Möglichkeit auf die Werkseinstellungen zurückzugreifen.

15.3 Debugger

Ein ebenfalls zur Entwicklungsumgebung gehörender Debugger ermöglicht das Testen einer Applikation. Sämtliche Zeilen im C und Assembler Quellcode lassen sich dabei Schritt für Schritt aufrufen, um Variablen und die Inhalte von Registern zu überwachen. In unserem Fall kann dies auf zwei Arten, entweder mit dem MetroTRK oder über den P&E Micro, geschehen. Welcher der beiden Debuggermodus verwendet werden soll, kann in der Targeteinstellung selektiert werden.

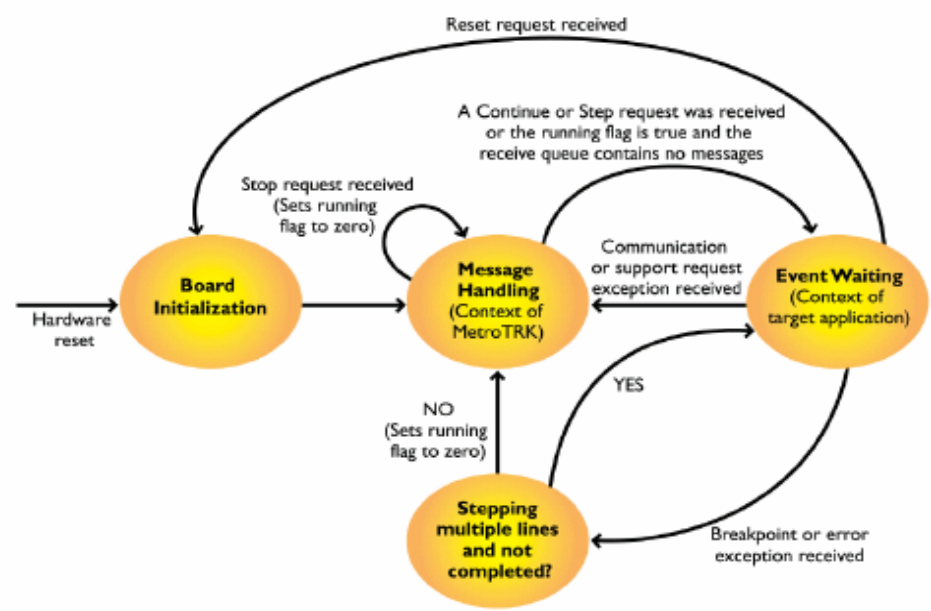
15.4 MetroTRK Debugging

MetroTRK (**T**arget **R**esident **K**ernel) ist ein Monitor der es erlaubt, Anwendungen auf ein Targetbord zu laden, ähnlich wie es bei NICO der Fall war. Als Kommunikations-schnittstelle dient ein serielles Kabel.



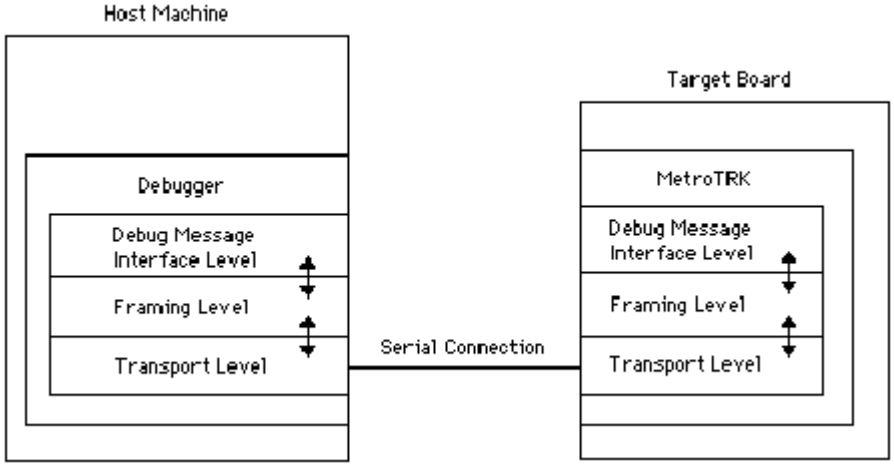
Grafik 15.4-1 Metro TRK debugging

Wird das Board eingeschaltet oder ist ein Reset erfolgt, beginnt der TRK mit der Initialisierung der Hardware auf dem Board, bevor er in den „Message-Handling“ Zustand wechselt. Im Betrieb kann sich der TRK innerhalb der Zustände „Message-Handling“ „Event Waiting“ oder „Stepping multiple lines“ bewegen.



Grafik 15.4-2 : TRK Statusdiagramm

Der Austausch von Daten zwischen dem Host-Rechner und Evaluierplatine erfolgt über verschiedene Schichten. Dabei werden einzelne Mitteilungen „verpackt“ und gleichzeitig mit einem Start- , einem Stop- und einem Paritätsbit versehen. Anschliessend wird die modifizierte Mitteilung gesendet.



Grafik 15.4-3 : TRK Ebenen

Als Beispiel simulieren wir einmal den Fall, eine MessageID 0x12 zu senden. Die original Debug-Mitteilung hat folgende Struktur

0x12	0x00	0x0065	0x007e
Message ID	Options	firstRegister	lastRegister

Grafik 15.4-4 : TRK Message

Für den TRK wird er in das folgend Format verwandelt und mit einer Checksumme versehen

0x12 + 0x00 + 0x00 + 0x65 + 0x00 + 0x7e = 0xf5; the complement of 0xf5 = 0x0a

0x12 Message Byte #1	0x00 Message Byte #2	0x00 Message Byte #3	0x65 Message Byte #4	0x00 Message Byte #5	0x7e Message Byte #6	0x0a 1-byte checksum

Grafik 15.4-5 : TRK Checksumme

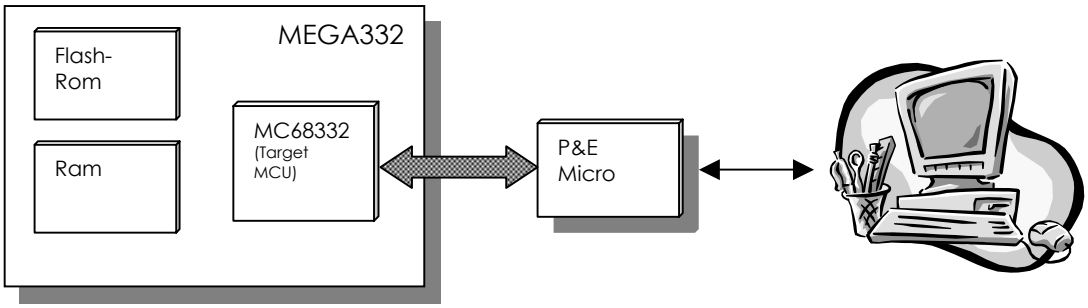
und an mit eine Escape Character und je einem Start und einem Stopbit ergänzt.

0x7e Start flag	0x12 Message Byte #1	0x00 Message Byte #2	0x00 Message Byte #3	0x65 Message Byte #4	0x00 Message Byte #5	0x7d Escape char	0x5e Message Byte #6 XOR 0x20	0x0a 1-byte checksum	0x7e End flag

Grafik 15.4-6 : TRK Escape-, Start und Stopbit

15.5 P&E Micro debugging

P&E ist eine Firma die sich auf Hard- und Softwarelösungen zum Debuggen von Applikationen über das Background Debugging Interface spezialisiert. Das Prinzip ist dem BDI von MCT ziemlich ähnlich. Anstelle des PC-Fanten kam diesmal ein P&E Micro Interface für CPU16/32 zum Einsatz. In diesem Fall kann das Programm jedoch als Konsolenapplikation, als Anwendung im Konsolen-Modus, im Debugging-Modus oder im Release-Modus gestartet werden. Bei der Konsolenapplikation besitzt der Host-Rechner die volle Kontrolle über den Programmablauf in dem er dem Targetrechner einen Befehl nach dem anderen zur Ausführung übergibt. Bei den beiden anderen Betriebsarten wird das Programm ins RAM geladen und von dort aus gestartet.



Grafik 15.5-1 : P&E Micro debugging

Bevor das erste Programm, egal in welchem Modus, heruntergeladen wird ist es wichtig zuvor die nötigen Zieladresseinstellungen vorzunehmen. Ansonsten besteht die Gefahr dass versucht wird, Code und Daten in nichtbeschreibbare Adressbereiche abzulegen.

15.6 Systemanforderung

Voraussetzung ist ein Pentium 166MHz Host Rechner mit minimal 64MB RAM und 170MB freier Festplattenspeicher. Mit dem MetroTRK erfolgt das debuggen über die Serieschnittstelle. Der P&E Micro Adapter wird über ein Parallelkabel mit dem PC verbunden.
Für die Nutzug des MetroTRK sind auf dem Anwendungsboard 8KB RAM für den Stack sowie 6KB RAM für globale Danten notwendig.

16 Beispielprogramme

16.1 Allgemeines

Um die Hauptplatine und die Module testen zu können, wurden zahlreiche Programme entwickelt. Schliesslich sollten die eingesetzten Bestandteile der Platinen auch funktionsfähig sein.
Am Anfang der Arbeit, bei dem die Planung der Hardwarekomponenten im Vordergrund stand, war das Ziel eine neue Entwicklungsumgebung auf des MEGA332-Board anzupassen. Später hätten die Programme auf der neuen Umgebung entworfen werden sollen. Da die CodeWarrior Entwicklungsumgebung nicht vollständig auf den MEGA angepasst werden konnte, wurden die Testprogramme auf dem bereits seit länger Zeit verwendeten ECO C verfasst.
Beim entwerfen der Programme wurde auf die allgemeine Wiederverwendbarkeit für weiter Softwareprojekte geachtet. Dabei solle es möglich sein, die Code-Files wie Bibliotheken in Programme einzubinden.

16.2 Display

Um auf Display etwas vernünftiges dargestellt werden kann, müssen dem Displaycontroller die gewünschten Daten übermittelt werden. Die Daten bestehen im wesentlichen aus den zu übertragenden Zeichen und den entsprechenden Steuerbefehlen. Die Übertragungszyklen müssen dabei genau eingehalten werden.
Die Codeseiten sind unter den folgenden Namen zu finden:

- basicdef.h
- display.h
- display.c

16.3 Interruptcontroller

Eine besondere Herausforderung war die Programmierung der Bedienungsroutinen für den Intel Interruptcontroller. Da die standardisierten Interrupt-Bedinungsroutinen der beiden Prozessorherstellern Intel und Motorola nicht übereinstimmen, bot sich für die Abarbeitung der Unterbrechungen folgende Variante an.
Nachdem der Interruptcontroller eine Interruptrequest ausgelöst hat, startet der MC68332 Controller eine Autovektor-Routine, welche die beiden Register des Intelcontrollers ausliest. Aus den Werten der beiden ausgelesenen Registern geht hervor welcher der sieben Interrupts zur Abarbeitung auf dem Interruptcontroller ausgelöst wurde.
Die Codeseiten sind unter den folgenden Namen zu finden:

- interrupt.c
- interrupt.h

16.4 Schrittmotormodul

Mit dem Schrittmotormodul kam erstmals die TPU des MC68332 Controllers zum Einsatz. Der Schrittmotormotor wird über vier Leitungen angesteuert. Die Ansteuerung erfolgt entweder über die vier Taster auf dem Modul oder mit der synchronen Pulsweitenmodulation der TPU. Dabei können Geschwindigkeit und Drehrichtung variiert werden. Die Drehzahl wird mit einem Encoder erfasst. Der Encoder ist mit zwei Lichtschranken und einer gefächerten Scheibe aufgebaut. Die Auswertung der vom Encoder gelieferten Signale erfolgt mit Hilfe der Quadratur Decodierung (QDEC). Dieselbe TPU-Funktion wurde auch für die Ansteuerung des Encoders auf der Hauptplatine verwendet. Die Codeseiten sind unter den folgenden Namen zu finden:

- schrittmotor.c

16.5 I²C Modul

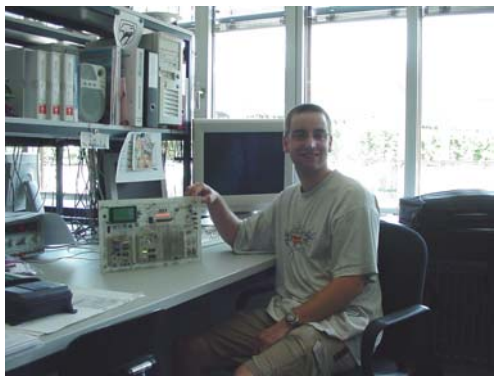
Wie es der Name schon erahnen lässt geht es darum die Komponenten des Moduls über einen I²C Bus anzusteuern. Für die Ansteuerung wurde auch in diesmal die TPU verwendet. Zwei diskrete Ausgänge sorgen dabei für die Adressierung und das versenden von Daten. Zwei Eingänge sind für das Einlesen und für die Kontrolle der Datenleitungen bestimmt. Die Kontrolle ist notwendig um feststellen zu können, ob sich noch ein anderer Master auf den Bus befindet.
Die Codeseiten sind unter den folgenden Namen zu finden:

- Ad-da.c
- DiscreteIO_e.h
- DiscreteIO_e.c
- I2C_sensor.h
- I2C_sensor.c
- I2C_Wait.h
- I2C_Wait.c

17 Fazit

17.1 Claudio Foscan (Platine)

Die Entwicklung einer so grossen Platine war für mich eine spezielle Herausforderung, da ich in meiner Lehrzeit nie mit Platinenlayout zu tun hatte und danach für meinen Arbeitgeber nur kleine Platinen von Hand entwickeln musste. Die Trägerplatine ist soweit den Erwartungen entsprechend fertig geworden. Es war mir auch möglich, das Layout von Hand auf zwei Layern zu gestalten, sodass wir in der Lage waren einen Prototypen vom CT-Labor Personal herstellen zu lassen. Mit diesem Prototypen war es uns möglich alle Komponenten der definitiven Lösung erfolgreich zu Testen.
Für mich war die Arbeit sehr Lehrreich in den Bereichen Layout, Schaltungsentwicklung und programmierbare Logik.



17.2 Anthony Sibler (Module)

Diese Semesterarbeit war für mich ein Erfolg. Bei der Entwicklung der Module hatten wir fast zu viele Ideen um konkrete Produkte auszuwählen, was mir die Möglichkeit gab, für mich interessante Module fertigen zu können. Da beim Entwickeln auch laufend weitere Wünsche aufkamen, gab es oft mehrere Durchgänge bei der Fertigung. Dies wirkte sich aber nur positiv auf das Schlussprodukt aus. Es war sehr angenehm den Umgang mit Orcad mit kleineren Projekten zu erlernen.

17.3 Stefan Mächler (Software)

Die Absicht, eine neue Entwicklungs-umgebung auf das vorhandene Board anzupassen war für mich eine neue Erfahrung. Dies beinhaltete auch das Arbeiten mit den im CT-Labor vorhanden Hilfsmitteln. Leider hat am Schluss aufgrund verschiedener Faktoren nicht alles Funktioniert, und somit waren die von uns gestellten Ziele im Bereich Software nicht ganz erreicht worden. Nichts desto Trotz waren die neu gewonnen Erkenntnisse in der hardwarenahen Programmierung sehr Wertvoll.

17.4 Gruppe

Die Zusammenarbeit in der Gruppe war sehr gut. Die einzelnen Teilbereiche wurden von den entsprechenden Gruppenmitgliedern sehr gut analysiert sodass aus unserer Sicht eine gute Arbeit entstanden ist. Wichtig war für uns auch der Austausch der wissenswerten Fakten, aus den verschiedenen Teilbereichen innerhalb der Gruppe. Dies hatte zur Folge dass diejenigen, die sich nicht Direkt mit den einzelnen Problemen befasst haben, trotzdem einen lehrreichen Einblick in die gesamte Arbeit bekamen.

17.5 Betreuung

Herr Brändle hat uns von Anfang an gut und engagiert betreut. Die Gruppe konnte von seiner Erfahrung im Bereich der eingebetteten Systeme viel profitieren. Was uns jeweils besonders motivierte war seine kooperative Art, wenn es darum ging grundlegende Entscheidungen zu Fällen.
Durch sein Engagement lief er jedoch ab und zu Gefahr, Dinge zu detailliert zu betrachten was zur Folge hatte, dass die wesentlichen Probleme ab und zu auf der Strecke blieben.
Insgesamt war es sehr Angenehm mit Herrn Bändle arbeiten zu dürfen.

18 Anhang A

18.1 Glossar

Begriff	Kurzbeschreibung
A	
AD-Wandler	Standardbaustein welcher analoge zu digitalen Signalen wandelt
Anker	Rotierender Teil im Elektromotor
Atomfreqzenznormal	Von Atomuhr generiertes Zeitsignal das extrem präzis ist
B	
C	
Charaktergenerator	Umsetzungstabelle für Codierte (char) Zeichen ins Displayformat (Pixel)
chipselect	Signal welches einen IC-Bautein (anhand seiner Adresse)aktiviert
Class Browser	Assistent für das erstellen und bearbeiten von Klassen in C++
D	
debuggen	austesten, Fehler suchen, "Bugs" beseitigen
differenziell	Bei I²C-AD-Wandler: Ein Eingangswert wird dem anderen abgezogen. Z.B. Ain0-Ain1
Drehimpulsgeber	
DSACK	Data strobe acknoledge
E	
Embedded 68k linker	Linker für 68k Controller von Motorola
F	
G	
Gabellischtschranken	kleiner U-förmiger Sensor, der erkennt ob sich etwas zwischen den Armen befindet
Gatter	Logik-Baustein
H	
I	
I²C Schnittstelle	Übergang zum I²C-Bus
interrupt	Unterbrechung, Ausnamebehandlung
Interruptcontroller	Erweiterungsbausten für zusätzliche Interrupts
Interruptserviceroutine	Programmunterbrechung in welcher eine Funktion (Routine) einen Interrupt abarbeitet
J	
JTAG	Joint Test Action Group, boundary-scan Industrie-Standard
K	
L	
M	
Makrozellen	
N	
O	
Offset	Bestimmte Anfangswert der aufaddiert werden muss
P	
Piepsersteuerung	
PLD	Programmierbarer Logikbaustein
PWM	Pusweiten-Modulation
Q	
QDEC	Quadratur-Decodierung
QSM	Queued serial module
R	
RTC	Echtzeit-Uhr

S	
SIM	System integratet module
Softmenu	Individuell programmierbare Schalter und Menue auf dem Disply
SPI	Seriall Peripherial Interface
State-Machine	Zustandsautomat / -maschine
T	
TPU	Time processing unit, autonomer Controller inderhalb des MC68332
U	
UART	Uniwerselle asynchrone Übertragungsschmittstelle
V	
VHDL-Code	Very high speed integrated circuit Hardware Description Language
W	
X	
Y	
Z	

Tabelle 18.1-1:Glossar

19 Anhang B

19.1 Quellenverzeichnis

- **MEGA332 Refenerence Manual**
  MCT ,Paul & Scherer Mikrocomputertechnik , Berlin, 1999
- **OrCAD Layout Footprint Lybraryes**
  OrCAD, 1996
- **Computertechnik Grundlagen & Vertiefung**
  Eduard Glatz & Rolf Schädler, HSR 1998
- **MC68332 User's Manual**
  Motorola, 1993
- **TPU Reference Manual**
  Motorola, 1993
- **Mikrocomputertechnik mit dem Controller 68332**
  G. Schmitt, 1998

19.2 Softwareunterstützung

- **Win ECO-C, Version 1.01**
  MCT ,Paul & Scherer Mikrocomputertechnik , Berlin, 1999
- **CodeWarrior IDE**
  Metroweks AG, Basel
- **OrCAD**
  MCT ,Paul & Scherer Mikrocomputertechnik , Berlin, 1999
- **WebPACK**
  MCT ,Paul & Scherer Mikrocomputertechnik , Berlin, 1999
- **Windows NT**
  Microsoft® Corporation 1993-2000
- **MS-Office 2000**
  Microsoft® Corporation

20 Anhang C

20.1 Logfile

Claudio		
Datum	**Text**	**File**
25.03.2002	AD-Wandler auf MegaBoard getestet	../eco-c/ad-wandler/ad.c
26.03.2002	Email an marelcom -> LCD Display	Typ AG16080 und AG12864
	Email an info@lcd-module.de -> LCD Dislay	Typ EA C160-6NLED
27.03.2002	Email an r.altherr@mpi.ch (Vertretung von EA)	Display unter 50.-
	MEGABoard hat 3 RS 232 schnittstellen !!!!!!!	
	Email an r.altherr@mpi.ch (Vertretung von EA)	Preisanfrage low cost EA GE128-6N3V24
	Email an ebrandle@hsr.ch Displayentscheid	
28.03.2002	Spezifikation ExtesionBus (Leitungen)	mit anthony
	Display Entscheid : 4x20	
02.04.2002	Display bestellt 4x20	Kaspar Naef
	E-Mail an e.braendle 5V/3.3V Problehm (MCORE)	
03.04.2002	Busspezifikation def.	../doku/exbus_spez.doc
	5V/3.3V (MCORE) Problem besprochen	
	Pinbelegung exbus	../doku/Pinout_exbus.xls
08.04.2002	Display ist gekommen ->	
	Anschlüsse kein 2.54 Raster !	
	Display funktioniert -Terstprogramm	../eco-c/display/displaytest.c
10.04.2002	Erste Versuche mit OrCad	../ct_lab_infrastructure/capture
		../ct_lab_infrastructure/layout
11.04.2002	Zeitplan	../administration/zeitplan.xls
	Doku Konzept	../doku/konzept.doc
	Stückliste Hauptplatine	../technische_doku/stückliste_Hauptplatine.xls
	Schema erweitert	

15.04.2002	Stückliste erweitert	../technische_doku/stückliste_Hauptplatine.xls
	WebPACK Downgeloaded	../xilinx/WebPACK
	Grob_Layout erstellt (noch nicht fertig)	../technische_doku/Layout_grob.doc
16.04.2002	Xilinx WebPAD installiert , erstets Beispiel gemacht	
17.04.2002	Nur aerger mit Xilinx-Testboard	
18.04.2002	Xilinx Testboard funktioniert ! (Chip war def!!!)	
	Adress-Decoder in VHDL und test auf Xilinx	../xilinx/addressdecoder/ad_decode.vhd
	Adress Switches verkabelt	
23.04.2002	Memorymap festgelegt	../technische_doku/memory_map.xls
24.04.2002	Adressbereiche für Peripherie festgelegt	
	Schnittstelle von Xilinx definiert	../xilinx/addressdecoder/ad_decode.vhd
	Mit Stefan das Display zum laufen gebracht	
	Schema erweitert	../capture/CT_LAB_INFRASTRUCTURE.DSN
25.04.2002	Schema erweitert	../capture/CT_LAB_INFRASTRUCTURE.DSN
	Sicherung erstellt	d:/sicherung per 020425/
29.04.2002	Schema erweitert	../capture/CT_LAB_INFRASTRUCTURE_01.DSN
	Layout vorbereitet	../layout/CT_LAB_INFRASTRUCTURE_02.MAX
03.04.2002	Xilinx Programm fertig -> passt nicht !!!!	
	Neuer Xilinx code nur mit LED Adressdecoder und beep	../xilinx/cs_and_beep/cs_and_beep.vhd
	Externer Interruptkontoller MSM8259A	
04.04.2002	Testaufbau mit Xilinx	
	Schema erweitert	../capture/CT_LAB_INFRASTRUCTURE_01.DSN
06.04.2002	Schema und Layout erweitert	../capture/CT_LAB_INFRASTRUCTURE_01.DSN
		../layout/CT_LAB_INFRASTRUCTURE_05_2_LAYER_HAND.MAX
23.05.2002	Zweiter Testaufbau mit Xilink	CS erfolgreich getestet
		LEDWRITE erfolgreich getestet
	VHDL-Code	../xilinx/cs_and_beep/cs_and_beep.vhd
06.06.2002	Schema und Layout fertigestellt	../capture/CT_LAB_INFRASTRUCTURE.DSN
		../layout/CT_LAB_INFRASTRUCTURE_05_2_LAYER_HAND_020606.MAX
	Stückliste Hauptplatine angefangen zu aktualisieren	../technische_doku/stückliste_Hauptplatine.xls
10.06.2002	Prototyp der Hauptplatine in auftrag gegeben	../layout/CT_LAB_INFRASTRUCTURE_05_2_LAYER_HAND_020610.MAX
	Stückliste fertig	../technische_doku/stückliste_Hauptplatine.xls

14.06.2002	Prototyp der Hauptplatine ist fertig.	
21.06.2002	Komponenten erfolgreich getestet	
	Peeper muss noch modifiziert werden	
24.06.2002	Layout gemäs Besprechung vom morgen angepasst	
31.6.2002	Doku , Lauyout produktionsfertig ohne Siebdruck	../layout/CT_LAB_INFRASTRUCTURE_05_2_LAYER_HAND_020631.MAX
04.07.2002	Layout Produktionsfertig inkl. Bestückungsplan und Siebdruck	../layout/CT_LAB_INFRASTRUCTURE_05_2_LAYER_HAND_020704.MAX
	Offerte eingeholt bei Walter Schoch AG	
	Offerte eingeholt bei Delectric	
05.07.2002	Offerte von Walter Schoch AG	Fax vom 5.7.2002
	Offerte von Delectric	Email vom 5.7.2002
	Etscheidung für Delectric	

Stefan

Datum	Text	File
bis 11.04.02	Grundgerüst der Doku	
	CodeWarrior: Anfragen an MetrioWerks & MCT	
	Terminplanung für Doku	
	Mcore Inbetriebname	
bis 18.04.02	Display:	
	- Datenblätter	
	- Anschlüsse	
	- Code Struktur (auf ECO-C)	
	CodeWarrior: MetroTRK und BDM	
bis 1.5.02	BDM mit EDB und PC-Fant	
	-- Probleme --	
	Bus-Error auf 86332 beim herunterladen	
	-->Telef. Absprache mit MCT (Hr. Paul & Hr. Scherrer)	
	Kommuniktion mit EDB sonst möglich	
bis 7.5.02	Encoder mit TPU (C- Programm)	
	Grobpflichtenheft (Entwurf)	

bis 21.5.02	Timer-Ports
	Interruptcontroller,Programm, Hardware
	Pegelanalysen mit Logikanalyzer
	Timing-Probleme
23.05.2002	LÄUFT NICHT - das ende des Interruptcontrollertraumes
24.05.2002	BDM Versuch mit HI-Wave & Abatron Modul
	BERR (BusError) auf MEGA nicht verfügbar
	Spezieller Anschluss (Steckerbrücke)
25.05.2002	Interruptcontroller LÄUFT DOCH (Juhu!)
27.05.2002	Grobpflichtenheft Endfassung (Unterzeichnung)
28.05.2002	MSP430 von TI analsysieren (gem. Hr Brändle)
	Datenblätter div. TI-Controller konsultieren
29.05.2002	Mail an Olimex.com; Anfrage wegen Lieferbarkeit
	CodeWarrior; Anpassungsanalysen auf MEGA
31.02.02	Testprogramm für Schrittmotor-Modul
bis 4.6.02	Schrittmotor und Encoder auf Modul läuft
10.06.2002	MSP430-Prints sind eingetroffen
11.06.2002	Dokumentation, Grobpflichtenheft angepasst
14.06.2002	Uberarbeitung der vorhanden Kapitel
15.06.2002	Gesammtdoku erstellt (unvollständige Version)
17.06.2002	P&E Micro Dongel eingetroffen
bis 25.6.02	Versuch, Programme ins Flash zu laden
03.07.2002	Mail von Jan Kobler wegen MetroTRK
bis 07.07.02	Dokumentation bereinigen

Anthony

Datum	Text	File
09/04/2002	Flash, S-RAM und I2C-Baustein suche und evaluierung	Div. \\Semesterarbeit\Datenblätter
10/04/2002	"	
11/04/2002	Auswahl verschiedener Bausteine	
15/04/2002	Entscheidung welche Module zuerst realisiert werden.	
	erstes Modul I2C mit AD/DA, 8Bit i/o, eeprom	g:/semesterarbeit/module/..
	zweites Modul Schrittmotor	
16/04/2002	Mit Schemazeichnen begonnen	
24/04/2002	Fertigung des ersten Printes	
26/04/2002	Testprogramm schreiben	g:/semesterarbet/c-code/i2cmodul.....
06/05/2002	Start zum Schrittmotormodul	
	Erste Schemas begonnen	
22/05/2002	Erweiterung des Modul_1	
23/05/2002	Erster Print von Modul_2 gefertigt	
27/05/2002	Nachbestellung von Bauteilen für Modul_1	
30/05/2002	Definitive Version von Modul_1 fertig	
07/06/2002	Begin Modul_3	
10/06/2002	Passende Flexiprintbuchse zum Schrittmotor bekommen	
11/06/2002	Layout zu Modul_2 angepasst	
12/06/2002	Definitive Version von Modul_2	
17/06/2002	Definitive Version von Modul_3	
26/06/2002	Endversion Doku	
28/06/2002	Alle Module nochmals überprüft	
02/07/2002	Korrektur der Doku	
04/07/2002	Gerberfiles für Module generiert	