Praktikum Informations-Systemtechnik II

Ubiquitous Computing

WS 2006/2007

Mikroprozessorlabor
Institut für Betriebssysteme und Rechnerverbund
Technische Universität Braunschweig

Ziel und Aufgabenstellung

Ziel des Praktikums ist es das Zusammenwirken einzelner Hard- und Softwarekomponenten zu einem Gesamtsystem darzustellen. Innerhalb dieser Aufgabenstellung ist es erforderlich, durch Messungen mit unterschiedlichen Messgeräten die Funktionalität der Praktikumshardware besser zu verstehen und gleichzeitig den Einsatz und Umgang unterschiedlicher Messgeräte zumindest einmal kennengelernt zu haben.
Die Gesamtaufgabe umfasst die Inbetriebnahme des Praktikumsarbeitsplatzes und die Einarbeitung in die hier verwendete Programmiersprache C und die Entwicklungsumgebung. Zum Verständnis der Hardware werden einzelne Messungen mit dem Logikanalysator, dem digitalen Speicheroszilloskop, dem Frequenzzähler und dem Multimeter durchgeführt.
Schrittweise werden einzelne Funktionsbausteine des Gesamtsystems in Betrieb genommen. Hier soll versucht werden, selbständig eigene Softwaretreiber zu schreiben, die später modular in eigenen Projekten verwendet werden können. Für komplexere und zeitaufwendigere Treiber können fertige Funktionsmodule eingebunden werden.

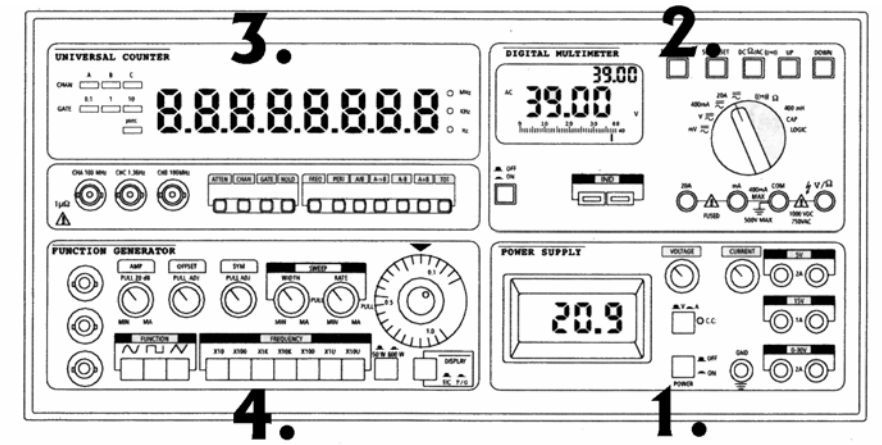
Schließlich sollen die neu erworbenen Kenntnisse dazu verwendet werden, ein eigenes Projekt zu realisieren. Hierbei stehen grundsätzlich alle Möglichkeiten offen – so wäre es denkbar, ein Szenario zu erstellen, in dem zwei miteinander kommunizierende Fahrzeuge eine Fahrstrecke bewältigen und dabei in der Lage sind, auftauchenden Hindernissen auszuweichen. Für ein derartiges Projekt ist dann die Kreativität der Teilnehmer gefordert.

Zur Bewältigung dieser Aufgabe ist es notwendig erstens die vorhandenen Werkzeuge kennenzulernen, zweitens sich mit der Programmierung der Einzelkomponenten auseinander zu setzen um drittens das Gesamtsystem anwenden zu können.

Der Praktikumsarbeitsplatz (Übersicht)

Der Arbeitsplatz besteht aus einem PC mit einer C51 Entwicklungsumgebung der Firma Keil, der Messstation, die Multimeter, Frequenzzähler, Funktionsgenerator und Stromversorgung für die Praktikumshardware umfasst, dem Logikanalysator, Speicheroszilloskop und natürlich der Mikrocontrollerplatine selbst.

Die Mess-Station



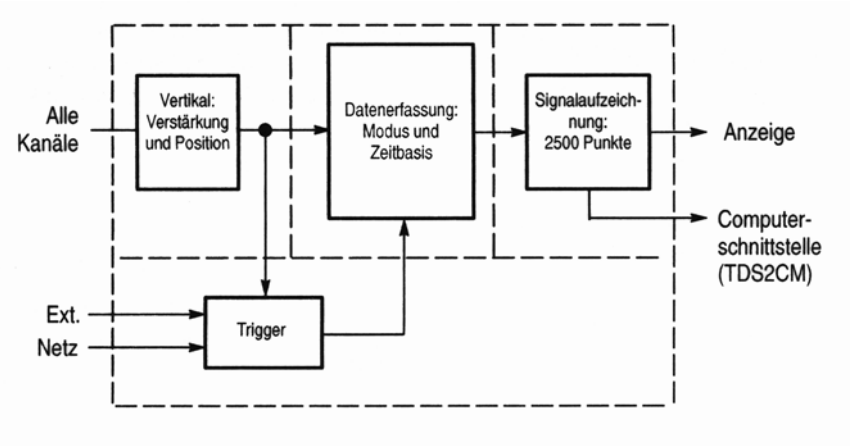
Das Gleichspannungsnetzgerät (1.) hat zwei Festspannungsausgänge mit 5V / 2A und 15V / 1A, sowie einen regelbaren Ausgang mit 0-30V / 0-3A. Da sich auf der Praktikumsplatine ein Spannungsregler des Typs 7805 befindet, der eine Mindestspannung von etwa 7 V verlangt, ist zum Anschluss der Mikrocontrollerplatine der regelbare Ausgang zu verwenden – hier ist jedoch darauf zu achten, dass möglichst nicht mehr als 9V angelegt werden, da die überschüssige Energie in Wärme umgewandelt wird, was zur Zerstörung des Spannungsreglers und umliegender Wertgegenstände führen kann.
Mit dem Digitalmultimeter (2.) können Gleichspannungen, Wechselspannungen und Ströme sowie Kapazitäten und Widerstände gemessen werden.
Der Frequenzzähler (3.) dient zur Messung von Frequenzen (nomen est omen), wobei es für unterschiedliche Messbereiche und Impedanzen verschiedene Eingangsbuchsen gibt. Es ist weiterhin möglich anstatt der Frequenz auch die Periodendauer, das Zeitintervall, das Verhältnis oder die Summe zweier Kanäle zu messen. Der Frequenzzähler ist auch als reiner Impulszähler verwendbar.
Als letzte Komponente ist noch ein Funktionsgenerator (4.) vorhanden, mit dessen Hilfe man verschiedene Kurvenformen (Sinus, Dreieck, Rechteck) einer vorgegebenen Frequenz erzeugen kann. Die Frequenz kann intern vom Frequenzzähler angezeigt werden. Auch das Frequenzwobbeln ist möglich.
Vor der Benutzung jeder Einzelkomponente sollte das Handbuch ausführlich zu Rate gezogen werden; inwieweit im Rahmen des Praktikums von den einzelnen Geräten Gebrauch gemacht wird, hängt von der jeweiligen Gruppe ab.

Der Logikanalysator

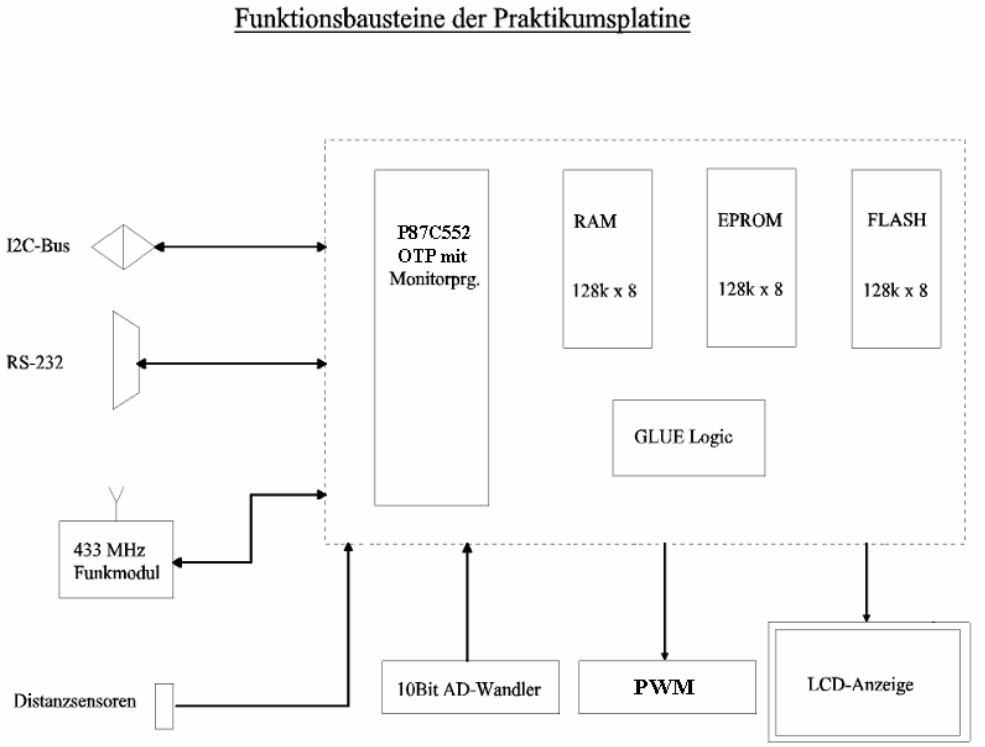
Mit dem Logikanalysator lassen sich insgesamt 34 digitale Kanäle (32 plus 2 für Clocksignale) messen und darstellen. Dabei ist es möglich, entweder in einem festen Zeitraster aufzunehmen und darzustellen oder die Messungen von einem externen Signal triggern zu lassen und somit auch asynchrone Quellen auszumessen. Die maximal messbare Taktfrequenz beträgt 250 Mhz, was für unsere Zwecke mehr als ausreichend ist.
Die Messanschlüsse sind in zwei sog. Pods zu je 16+1 Kanäle aufgeteilt, die sich direkt auf die Praktikumsplatine stecken lassen. Es ist aber auch möglich, einzelne Messspitzen aufzustecken, um Signale einzeln aufzunehmen.
Der Analysator wird über eine auf dem Praktikums-PC installierte grafische Benutzeroberfläche gesteuert. Weitere Informationen befinden sich im Anhang.

Das digitale Speicheroszilloskop

Mit einem digitalen Speicheroszilloskop ist es einerseits möglich, analoge periodische Signalverläufe genau wie mit einem analogen Oszilloskop darzustellen – darüber hinaus können durch die Speicherfunktion auch nichtperiodische Signale, wie sie im Rahmen dieses Praktikums häufig auftreten, „eingefroren“ und angezeigt werden. Außerdem stehen weiter Funktionen wie die direkte Anzeige von Parametern periodischer (Frequenz, Amplitude) und nichtperiodischer (High-/Lowtime, Impulsbreite) Signale, komplexe Triggermöglichkeiten, automatischer Messbereichseinstellung und manueller Messung von Parametern durch Cursor zur Verfügung.
Das hier vorhandene Zweikanal-Digitaloszilloskop verfügt auf jedem Kanal über eine Bandbreite von 100 MHz bei einer Abtastrate von 100 GigaSamples/s. Bei Bedarf können die Messergebnisse auch auf einen PC übertragen und dort ausgewertet werden.

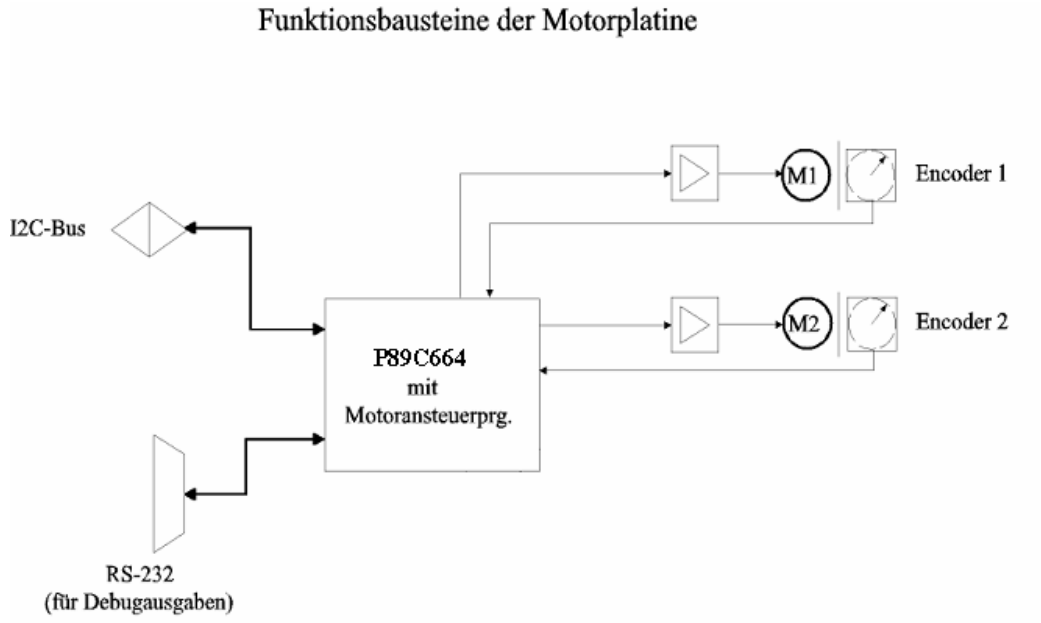


Die Praktikumsplatine



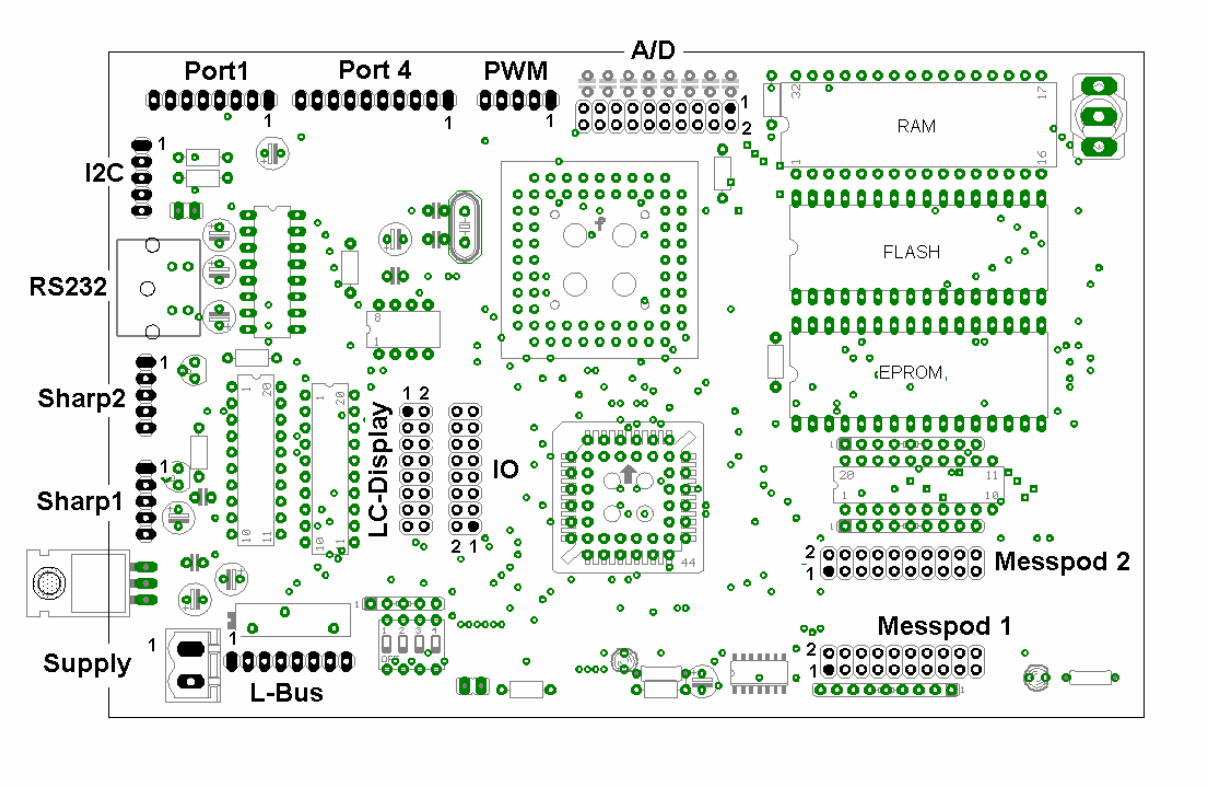
Auf der Praktikumsplatine befindet sich der 8-Bit Mikrocontroller P87C552 der Firma Philips. Dieser Mikrocontroller basiert auf der sehr weit verbreiteten 8051 Architektur. Er kann maximal 64 kByte Programmspeicher und 64 kByte Datenspeicher adressieren.
Bei der hier vorhandenen Realisierung werden 8 kByte interner Programmspeicher des Mikrocontrollers für ein Monitorprogramm verwendet, das die Kommunikation mit der Entwicklungsumgebung auf dem Praktikums-PC kontrolliert. Die möglichen Speicherkonfigurationen werden bei der Prozessorbeschreibung im Detail erläutert. Desweiteren enthält die Praktikumsplatine eine serielle RS232-Schnittstelle für die Kommunikation mit dem Praktikums-PC. Hierüber können Programme in den Speicher der Praktikumsplatine geladen und gestartet werden. Ebenso kann hierüber Kontroll- und Statuskommunikation von der Praktikumsplatine zur Entwicklungsumgebung und zurück erfolgen. Die Praktikumsplatine besitzt weiterhin eine Schnittstelle für ein LC-Display für autonome Ausgaben, ein I2C-Interface zur Kommunikation mit der zugehörigen Motorplatine, eine Schnittstelle zum Anschluss von Distanzsensoren, einen 10 Bit AD-Wandler für die Messung analoger Spannungsgrößen sowie zwei Pulsweiten modulierte Ausgänge (PWM), mit denen z.B. Servomotoren angesteuert werden können. Für projektbezogene Anwendungen und Erweiterungen gibt es außerdem noch einige Steckerleisten, die mit verschiedenen Signalen belegt sind (Adressen, Daten, Ports, etc.). Die notwendige "Glue"-Logik zur Ansteuerung der externen Speicherbausteine, des "Memory Mapped IO" etc., ist in einem rekonfigurierbaren Logikbaustein, ähnlich denen des IST-1-Praktikums, integriert.
Die Beschreibung der einzelnen Steckerleisten sowie eine Übersicht des Philips P87C552 befindet sich im Anhang.
Auf den Platinen ist ein Spannungsregler integriert, der Spannungen im Bereich von 7 bis 30 V auf die von der Elektronik benötigten 5 V herunterregelt. Man sollte den Bereich von 7-9V jedoch nicht verlassen, um eine Überhitzung des Spannungsreglers zu vermeiden.

Die Motorplatine



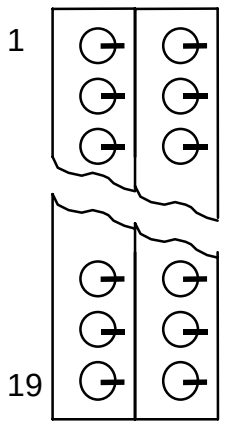
Das Fahrmodul besitzt zwei modifizierte Servomotoren. Die Motoren verfügen über einen Encoder, so dass sich ein vollständiger Wirkungskreis ergibt. Zur Ansteuerung der Motoren, und evtl. weiterer Funktionen befindet sich die Mototorplatine auf dem Fahrmodul. Das zugehörige Programm ist in dem Flash des P89C664, eines ebenfalls 8051-basierten Mikrocontrollers, gespeichert. Damit ist es möglich, über den I2C-Bus der Motorplatine Kommandos für Geschwindigkeit und Richtung der Motoren zu geben. Das Verständnis und die Handhabung des I2C-Busses wird mittels eigens dafür konzipierter Teilaufgaben erarbeitet.

Schnittstellenbeschreibung der Praktikumsplatine



Schnittstelle A/D:

Analog/Digital - Wandler (male)



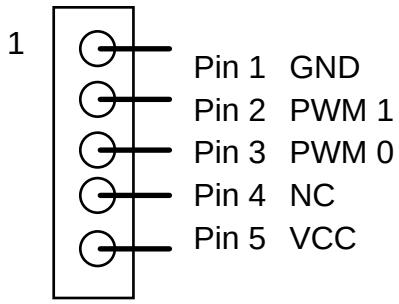
2

Pin 1	GND	Pin 2	ADC 7
Pin 3	GND	Pin 4	ADC 6
Pin 5	GND	Pin 6	ADC 5
Pin 7	GND	Pin 8	ADC 4
Pin 9	GND	Pin 10	ADC 3
Pin 11	GND	Pin 12	ADC 2
Pin 13	GND	Pin 14	ADC 1
Pin 15	GND	Pin 16	ADC 0
Pin 17	GND	Pin 18	GND
Pin 19	GND	Pin 20	STADDC

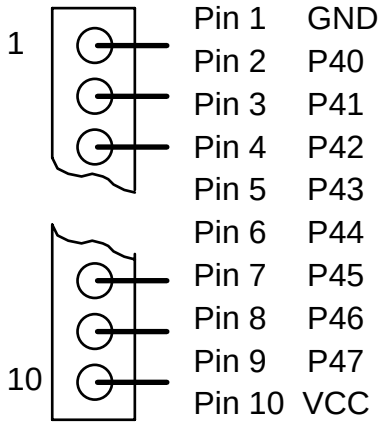
20

Schnittstelle PWM:

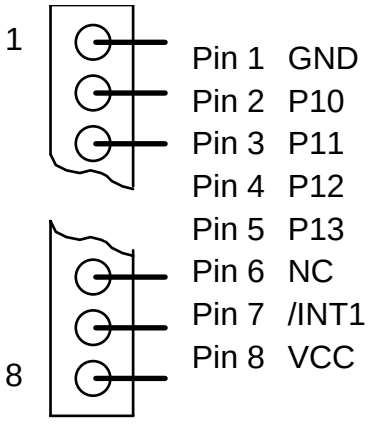
PWM Output (male)



Schnittstelle Port4:

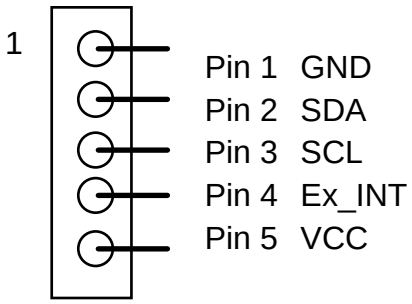


Schnittstelle Port1:



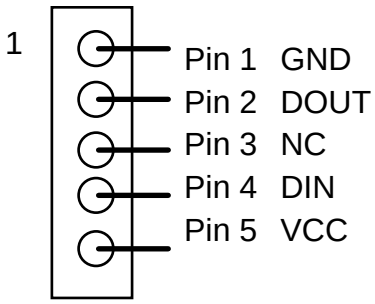
Schnittstelle I2C:

I2C-Bus Interface (male)

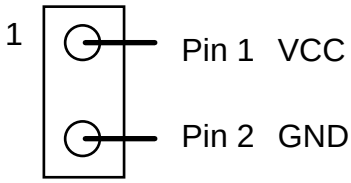


Schnittstelle SHARP1 & SHARP2:

SHARP Entfernungssensor 1 + 2 (male)

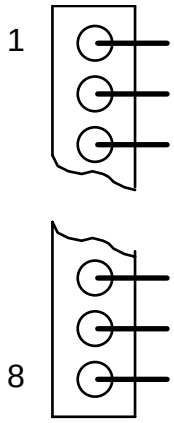


Schnittstelle Supply:



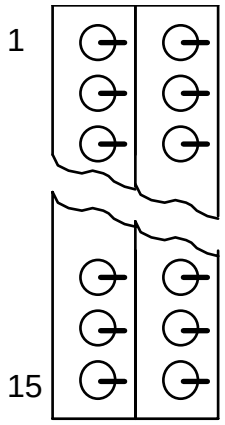
Schnittstelle L-BUS:

4 Bit Output Port (74LS574) (male)



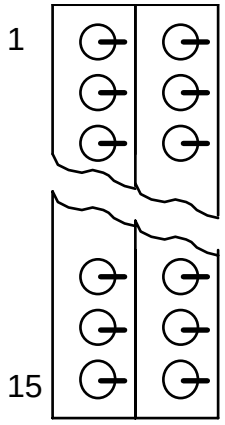
Pin 1	GND		
Pin 2	NC		
Pin 3	D4		
Pin 4	D5		
Pin 5	D6		
Pin 6	D7		
Pin 7	NC		
Pin 8	VCC		

Schnittstelle LC-Display:



Pin 1	GND	Pin 2	VCC
Pin 3	Kontrast	Pin 4	A0
Pin 5	A1	Pin 6	SEL_IO2
Pin 7	D0	Pin 8	D1
Pin 9	D2	Pin 10	D3
Pin 11	D4	Pin 12	D5
Pin 13	D6	Pin 14	D7
Pin 15	NC	Pin 16	NC

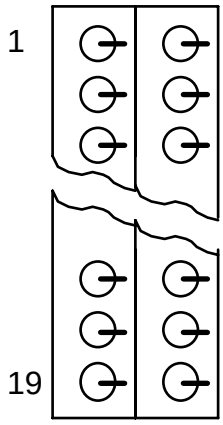
Schnittstelle IO:



Pin 1	VCC	Pin 2	VCC
Pin 3	D6	Pin 4	D7
Pin 5	D4	Pin 6	D5
Pin 7	D2	Pin 8	D3
Pin 9	D0	Pin 10	D1
Pin 11	/RD	Pin 12	SEL_IO1
Pin 13	/WR	Pin 14	SEL_IO0
Pin 15	GND	Pin 16	GND

Schnittstelle Messpod1:

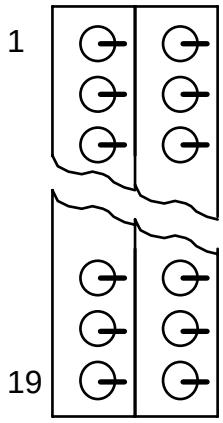
Logikanalysator Messpod I (male)



Pin 1	NC	Pin 2	NC
Pin 3	NC	Pin 4	A15
Pin 5	A14	Pin 6	A13
Pin 7	A12	Pin 8	A11
Pin 9	A10	Pin 10	A9
Pin 11	A8	Pin 12	A7
Pin 13	A6	Pin 14	A5
Pin 15	A4	Pin 16	A3
Pin 17	A2	Pin 18	A1
Pin 19	A0	Pin 20	GND

Schnittstelle Messpod2:

Logikanalysator Messpod II (male)



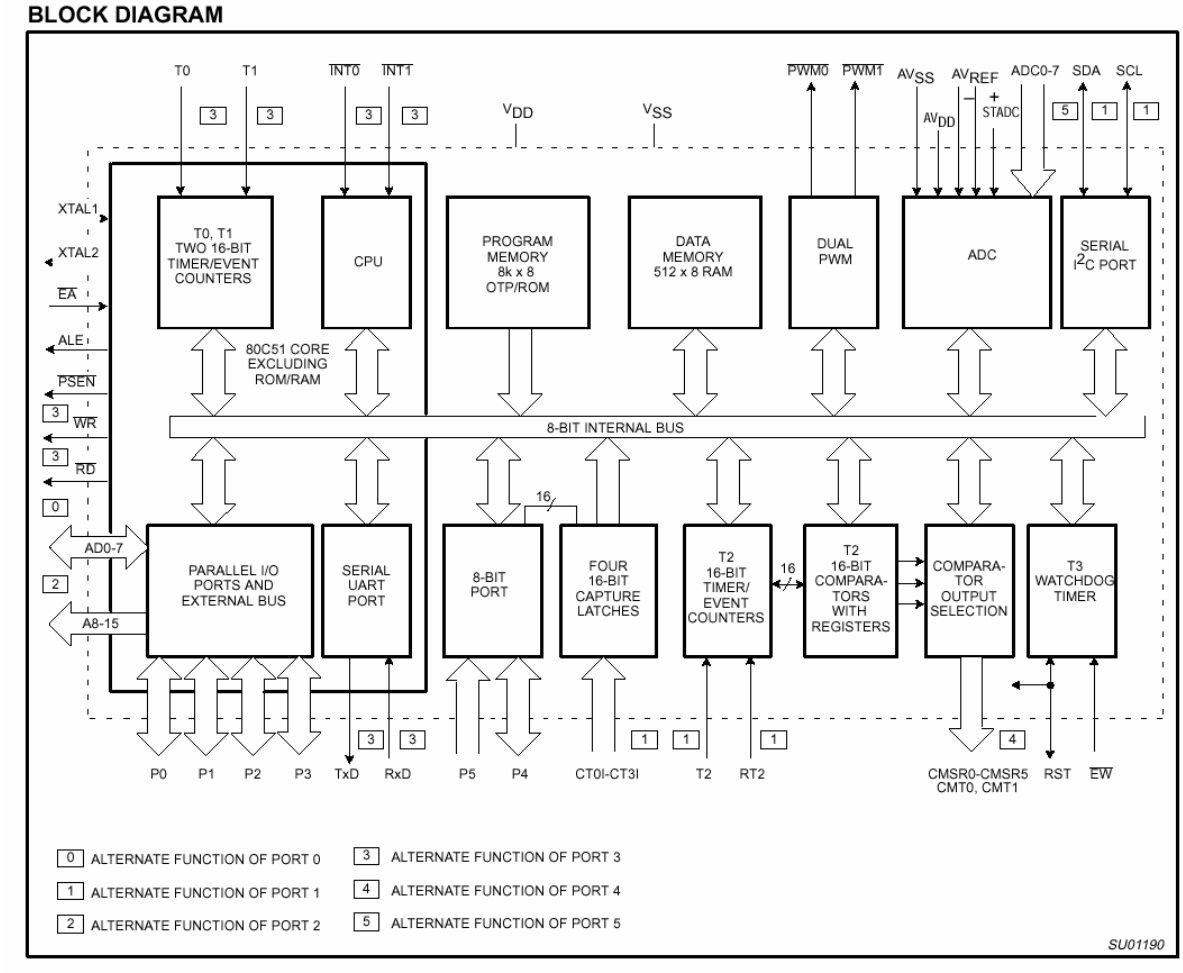
Pin 1	NC	Pin 2	NC
Pin 3	NC	Pin 4	NC
Pin 5	TxD	Pin 6	RxD
Pin 7	/RD	Pin 8	T0
Pin 9	/PSEN	Pin 10	/WR
Pin 11	ALE	Pin 12	AD7
Pin 13	AD6	Pin 14	AD5
Pin 15	AD4	Pin 16	AD3
Pin 17	AD2	Pin 18	AD1
Pin 19	AD0	Pin 20	GND

Beschreibung des verwendeten Mikrokontrollers

Der im Praktikum verwendete Mikrocontroller ist eine auf dem Intel 8051 basierende Variante der Firma Philips. Der Standard 8051-Kern wird beim hier eingesetzeten 80552 um einige zusätzliche Funktionalitäten erweitert. Der 80552-Mikrocontroller enthält u.a. :

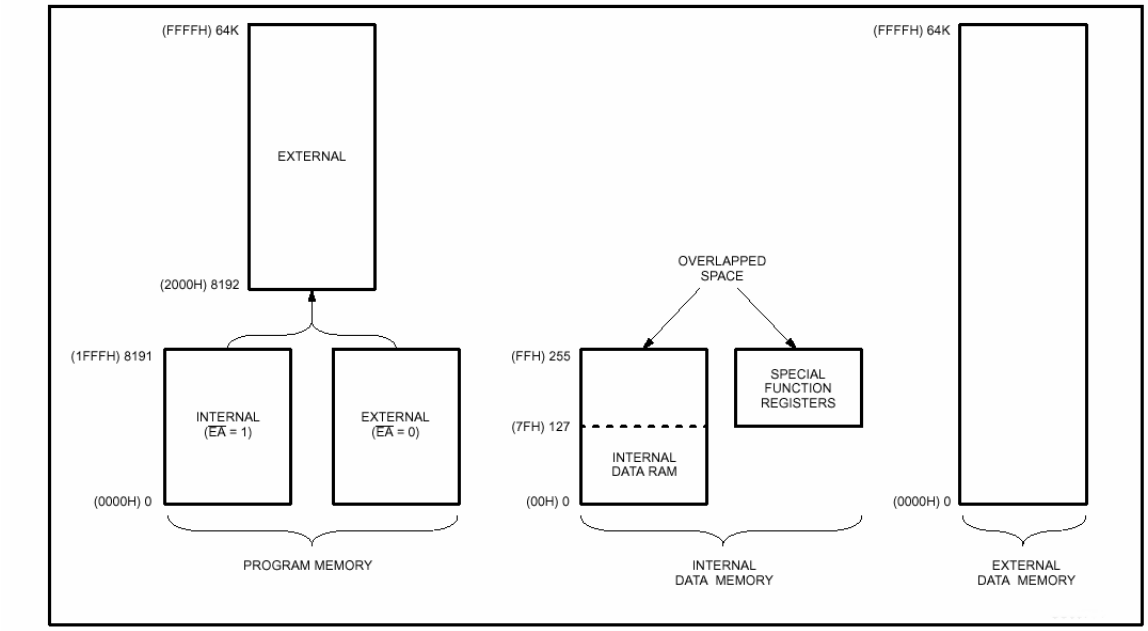
- 8051-kompatible CPU
- 3 16-Bit Timer
- 8K Byte internen Programmspeicher (extern bis auf 64K Byte erweiterbar)
- 256 Byte internes RAM (extern bis auf 64K Byte erweiterbar)
- 10Bit AD-Wandler mit 8 gemultiplexten Eingängen
- 2 PWM-Ausgänge
- On Chip Watchdog Timer
- I2C-Bus mit Master und Slave Funktionalität

weitere Details sind dem nachfolgenden Blockschaltbild oder dem Datenblatt zu entnehmen.



Die Architektur des Mikrocontroller basiert auf der sogenannten Harvard Architektur, d.h. es gibt einen getrennten Programm- und Datenspeicher. Durch entsprechende Beschaltung ist es aber möglich, das externe RAM sowohl als Programm- wie auch als Datenspeicher zu benutzen. Die Praktikums-Hardware macht davon Gebrauch, somit ist es möglich Anwendungsprogramme unter Kontrolle des sich im internen ROM befindlichen Monitorprogramms via serieller Schnittstelle downzuladen und ablaufen zu lassen.

Das Speicherkonzept des Mikrokontrollers:

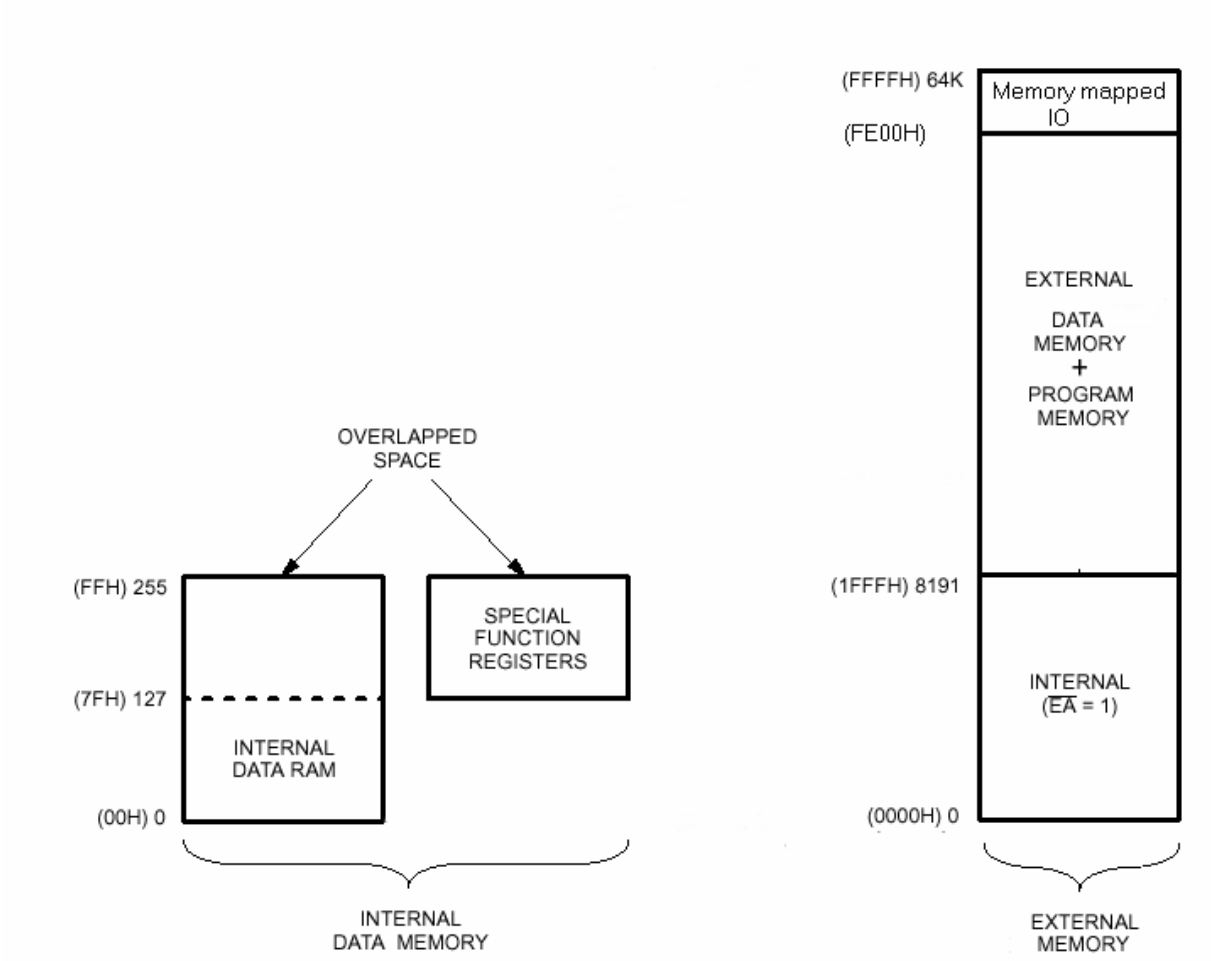


Memory Map

In der Abbildung ist die Speicheradressierung des Mikrokontrollers dargestellt. Man erkennt, dass der Programmspeicher intern auf 8K Byte begrenzt ist und extern auf 64K Byte erweiterbar. Durch Beschaltung ist es auch möglich, nur externen Programmspeicher zu benutzen. Der interne Datenspeicher beinhaltet die Prozessorregister, bitadressierbare Speicherstellen und die „Special Function Register“ mit deren Hilfe auf die interne Peripherie wie z.B. Timer0 zugegriffen werden kann. Eine genauere Übersicht der SFR folgt später.
Der externe Datenspeicher kann bis 64K Byte erweitert werden. Der Zugriff auf den internen Datenspeicher erfolgt durch direkte Adressierung während auf den externen Datenspeicher nur indirekt über Pointer zugegriffen werden kann. Die Übersicht zeigt, dass auf Programmspeicher und Datenspeicher getrennt zugegriffen wird. (Harvard Architektur)

Speicherkonzept des Mikrokontroller im Praktikum:

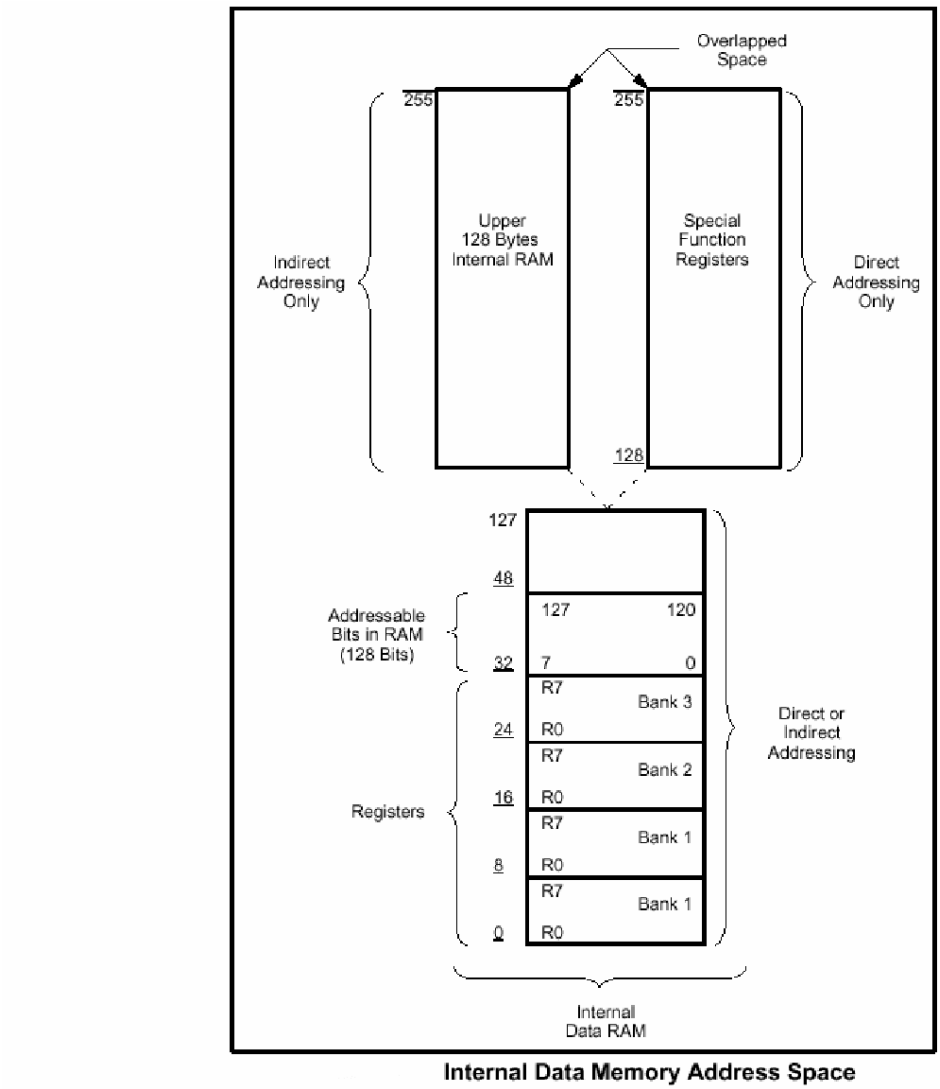
Das Speicherkonzept der Praktikumshardware sieht vor, dass von Adresse 0H bis 1FFFH der interne Programmspeicher des Mikrokontrollers mit dem darin befindlichen Monitorprogramm liegt. Ab Adresse 2000H bis FDFFH liegt der externe Programmspeicher(CDATA) und der externe Datenspeicher(XDATA), d.h. es ist sehr wichtig in der Keil-Entwicklungsumgebung die Anfangsadresse und die Länge des jeweiligen Speichersegmentes anzugeben und darauf zu achten, dass sich diese Bereiche nicht überlappen. Im Bereich von FE00H bis FFFFH liegt der sogenannte Memory-Mapped-IO Bereich, damit ist gemeint, dass über diese Speicherstellen auf zusätzliche nicht im Prozessor befindliche Peripherie zugegriffen wird. Die Adressierung erfolgt genau wie bei dem Zugriff auf den externen Datenspeicher indirekt über Pointer. Gegenwärtig wird für das Praktikum nur die intelligente LCD-Anzeige über diesen Bereich benutzt.



Das verwendete Speicherkonzept des Praktikumsrechners

Beschreibung des internen Datenspeicher:

Wie in dem nachfolgenden Blockschaltbild ersichtlich ist der interne Datenspeicher in 3 Teile aufgeteilt. Die unteren 128 Byte sind sowohl direkt als auch indirekt adressierbar, in ihnen befinden sich die allgemeinen Register R0-R7 des Mikrocontrollers. Es ist möglich, z.B. in Unterprogrammen verschiedene Registersätze(Bänke) zu verwenden. Es gibt 4 Registerbänke mit jeweils den Registern R0-R7, allerdings können nicht zwei Registerbänke gleichzeitig benutzt werden. Weiterhin sind in diesem Speicherbereich 16 Byte(128 Bit) untergebracht, die bitadressierbar sind, damit ist es möglich boolsche, Aussagen platzsparend zu speichern. (Es gibt beim C51-Compiler auch den Datentyp BIT). Der restliche Speicher hat keine Sonderfunktion.
Der Bereich von 128-256 ist doppelt vorhanden und es wird durch die Adressierung unterschieden auf welchen Bereich man gerade zugreifen will. Der indirekt adressierte Bereich (beim C51-Compiler als IDATA bezeichnet) hat keine Sonderfunktion. Der direkt adressierbare Bereich beinhaltet die Special Function Register für die interne Peripherie und wird noch genauer aufgelistet.



Internal Data Memory Address Space

Übersicht der Special-Function-Register nach Funktionsgruppen:

ARITHMETIC REGISTERS:
ACCumulator,* B register,*
Program Status Word*

POINTERS:
Stack Pointer,
Data Pointer (High and Low)

PARALLEL I/O PORTS:
Port 5,* Port 4,*Port 3,*
Port 2,* Port 1,* Port 0*

INTERRUPT SYSTEM:
Interrupt Priority 0,*
Interrupt Priority 1,*
Interrupt Enable 0,*
Interrupt Enable 1*

PULSE WIDTH MODULATED O/Ps:
Pulse Width Modulation Prescaler
Pulse Width Modulation Register 0,
Pulse Width Modulation Register 1

SERIAL I/O PORTS:
Serial 0 CONtrol,* Serial 0 data BUFfer,
Serial 1 CONtrol,* Serial 1 DATa,
Serial 1 STAtus, Serial 1 ADDress, PCON

TIMERS:
Timer MODe, Timer CONtrol,*
Timer Low 0, Timer High 0,
Timer Low 1, Timer High 1,
TiMer T2 CONtrol, TiMer Low 2,
Timer High 2, Timer T3

CAPTURE AND COMPARE LOGIC:
CapTure CONtrol,
TiMer T2 Interrupt flag Register,
CapTure Low 0, CapTure High 0,
CapTure Low 1, CapTure High 1,
CapTure Low 2, CapTure High 2,
CapTure Low 3, CapTure High 3,
CoMpare Low 0, CoMpare High 0,
CoMpare Low 1, CoMpare High 1,
CoMpare Low 2, CoMpare High 2
SeT Enable, ReseT Enable

ADC
ADC cONtrol, ADC High byte

*NOTE: Bit and byte addressable

Special Function Registers

8XC552 Special Function Registers

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
ADCH#	A/D converter high	C6H									xxxxxxxxB
ADCON#	Adc control	C5H	ADC.1	ADC.0	ADEX	ADCI	ADCS	AADR2	AADR1	AADR0	xx000000B
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
CTCON#	Capture control	EBH	CTN3	CTP3	CTN2	CTP2	CTN1	CTP1	CTN0	CTP0	00H
CTH3#	Capture high 3	CFH									xxxxxxxxB
CTH2#	Capture high 2	CEH									xxxxxxxxB
CTH1#	Capture high 1	CDH									xxxxxxxxB
CTH0#	Capture high 0	CCH									xxxxxxxxB
CMH2#	Compare high 2	CBH									00H
CMH1#	Compare high 1	CAH									00H
CMH0#	Compare high 0	C9H									00H
CTL3#	Capture low 3	AFH									xxxxxxxxB
CTL2#	Capture low 2	AEH									xxxxxxxxB
CTL1#	Capture low 1	ADH									xxxxxxxxB
CTL0#	Capture low 0	ACH									xxxxxxxxB
CML2#	Compare low 2	ABH									00H
CML1#	Compare low 1	AAH									00H
CML0#	Compare low 0	A9H									00H
DPTR:	Data pointer (2 bytes)										
DPH	Data pointer high	83H									00H
DPL	Data pointer low	82H									00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IEN0*#	Interrupt enable 0	A8H	EA	EAD	ES1	ES0	ET1	EX1	ET0	EX0	00H
			EF	EE	ED	EC	EB	EA	E9	E8	
IEN1*#	Interrupt enable 1	E8H	ET2	ECM2	ECM1	ECM0	ECT3	ECT2	ECT1	ECT0	00H
			BF	BE	BD	BC	BB	BA	B9	B8	
IP0*#	Interrupt priority 0	B8H	–	PAD	PS1	PS0	PT1	PX1	PT0	PX0	x0000000B
			FF	FE	FD	FC	FB	FA	F9	F8	
IP1*#	Interrupt priority 1	F8H	PT2	PCM2	PCM1	PCM0	PCT3	PCT2	PCT1	PCT0	00H
P5#	Port 5	C4H	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	xxxxxxxxB
			C7	C6	C5	C4	C3	C2	C1	C0	
P4#	Port 4	C0H	CMT1	CMT0	CMSR5	CMSR4	CMSR3	CMSR2	CMSR1	CMSR0	FFH
			B7	B6	B5	B4	B3	B2	B1	B0	
P3*	Port 3	B0H	$\overline{RD}$	$\overline{WR}$	T1	T0	$\overline{INT1}$	$\overline{INT0}$	TXD	RXD	FFH
			A7	A6	A5	A4	A3	A2	A1	A0	
P2*	Port 2	A0H	A15	A14	A13	A12	A11	A10	A9	A8	FFH
			97	96	95	94	93	92	91	90	
P1*	Port 1	90H	SDA	SCL	RT2	T2	CT3I	CT2I	CT1I	CT0I	FFH
			87	86	85	84	83	82	81	80	
P0*	Port 0	80H	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	FFH
PCON#	Power control	87H	SMOD	–	–	WLE	GF1	GF0	PD	IDL	00xx0000B
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	F1	P	00H

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.

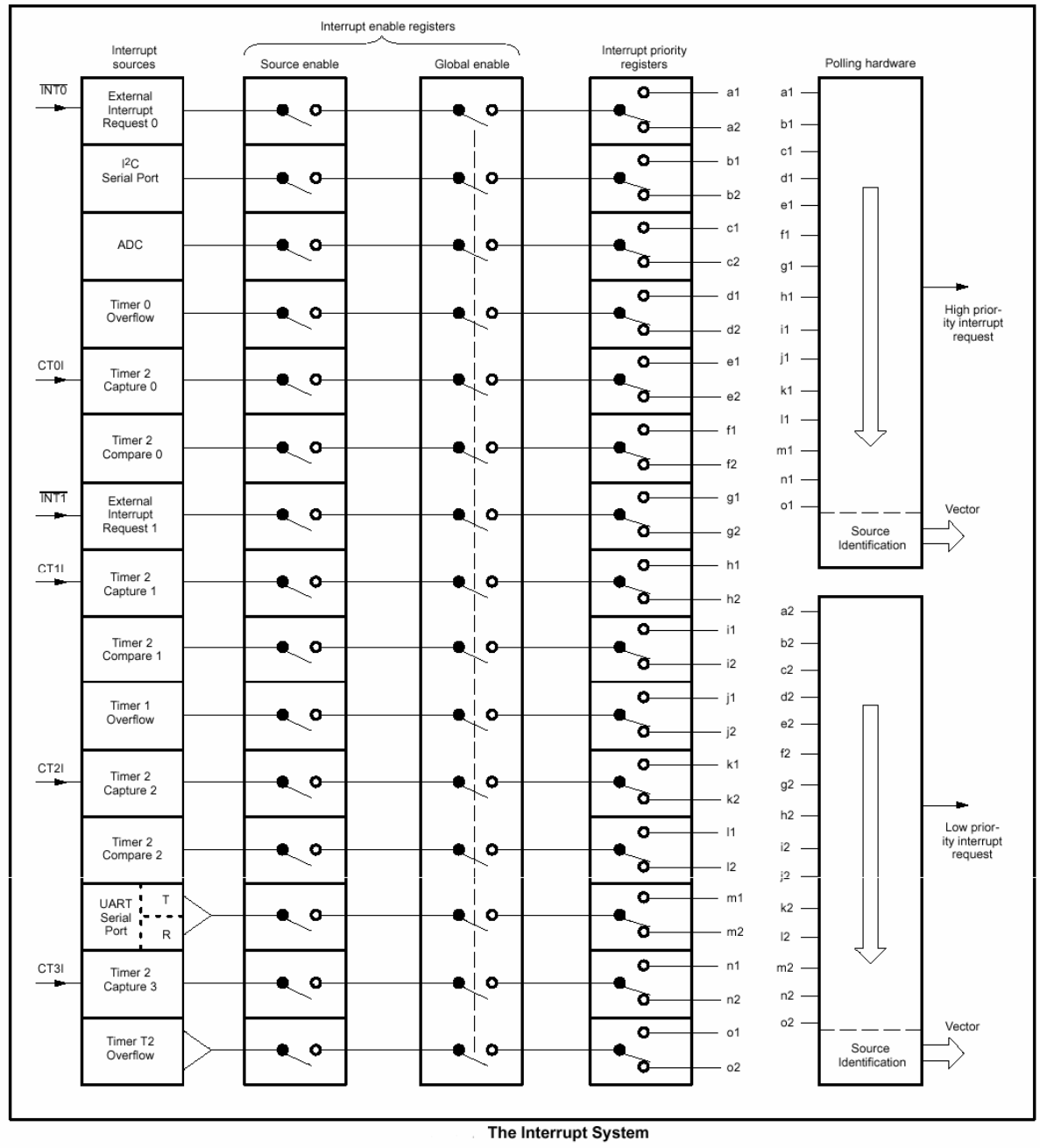
8XC552 Special Function Registers (Continued)

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
PWMP#	PWM prescaler	FEH									00H
PWM1#	PWM register 1	FDH									00H
PWM0#	PWM register 0	FCH									00H
RTE#	Reset/toggle enable	EFH	TP47	TP46	RP45	RP44	RP43	RP42	RP41	RP40	00H
SP	Stack pointer	81H									07H
S0BUF	Serial 0 data buffer	99H									xxxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
S0CON*	Serial 0 control	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
S1ADR#	Serial 1 address	DBH	SLAVE ADDRESS							GC	00H
SIDAT#	Serial 1 data	DAH									00H
S1STA#	Serial 1 status	D9H	SC4	SC3	SC2	SC1	SC0	0	0	0	F8H
			DF	DE	DD	DC	DB	DA	D9	D8	
SICON#*	Serial 1 control	D8H	CR2	ENS1	STA	ST0	SI	AA	CR1	CR0	00H
STE#	Set enable	EEH	TG47	TG46	SP45	SP44	SP43	SP42	SP41	SP40	C0H
TH1	Timer high 1	8DH									00H
TH0	Timer high 0	8CH									00H
TL1	Timer low 1	8BH									00H
TL0	Timer low 0	8AH									00H
TMH2#	Timer high 2	EDH									00H
TML2#	Timer low 2	ECH									00H
TMOD	Timer mode	89H	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0	00H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00H
TM2CON#	Timer 2 control	EAH	T2IS1	T2IS0	T2ER	T2B0	T2P1	T2P0	T2MS1	T2MS0	00H
			CF	CE	CD	CC	CB	CA	C9	C8	
TM2IR#*	Timer 2 int flag reg	C8H	T20V	CMI2	CMI1	CMI0	CTI3	CTI2	CTI1	CTI0	00H
T3#	Timer 3	FFH									00H

* SFRs are bit addressable.
\# SFRs are modified from or added to the 80C51 SFRs.

Zur Erläuterung der Tabelle ist zu sagen, dass die symbolischen Namen der einzelnen SFR oder einzelner Bits auch im C51-Compiler verwendet werden können (die Umsetzung der Adresse zum Symbol steht in der Datei Reg552.h). Das Programmstatuswort (PSW), der Stackpointer (SP) und der Akkumulator (ACC) werden ebenfalls in diesem Bereich gespeichert. Weiterhin ist zu sehen, dass die Alternativfunktion der Ports 0 bis 5 hier entsprechend programmiert wird. Die genaue Bedeutung der einzelnen Bits ergeben sich bei der Programmierung der entsprechenden Funktion und sind aus dem Datenblatt zu entnehmen.

Aufbau des Interruptsystems:



The Interrupt System

Interrupte können sowohl von der internen Peripherie z.B. Timer 2 als auch von einer externen Quelle wie z.B. /INT0 ausgelöst werden. Damit ein Interrupt wirksam wird muss zum einen das Interuptsystem global freigegeben sein und zum anderen die spezielle Interruptquelle freigegeben werden. Die Freigabe erfolgt in den SFR-Registern IEN0 und IEN1. Es muss weiterhin die Priorität des Interruptes festgelegen werden, dieses geschieht im SFR-Register IP. Ein niederwertig priorisierter Interrupt kann durch einen höher priorisierten Interrupt unterbrochen werden, umgekehrt jedoch nicht. Ist der entsprechende Interrupt freigegeben und aktiv, wird ein Interruptvektor gemäss nachfolgender Tabelle generiert. Im C51-Compiler erfolgt das Bearbeiten von Interrupten im C-Programm selber, so dass hier auf eine weitere Beschreibung des Interruptverarbeitung verzichtet werden kann.

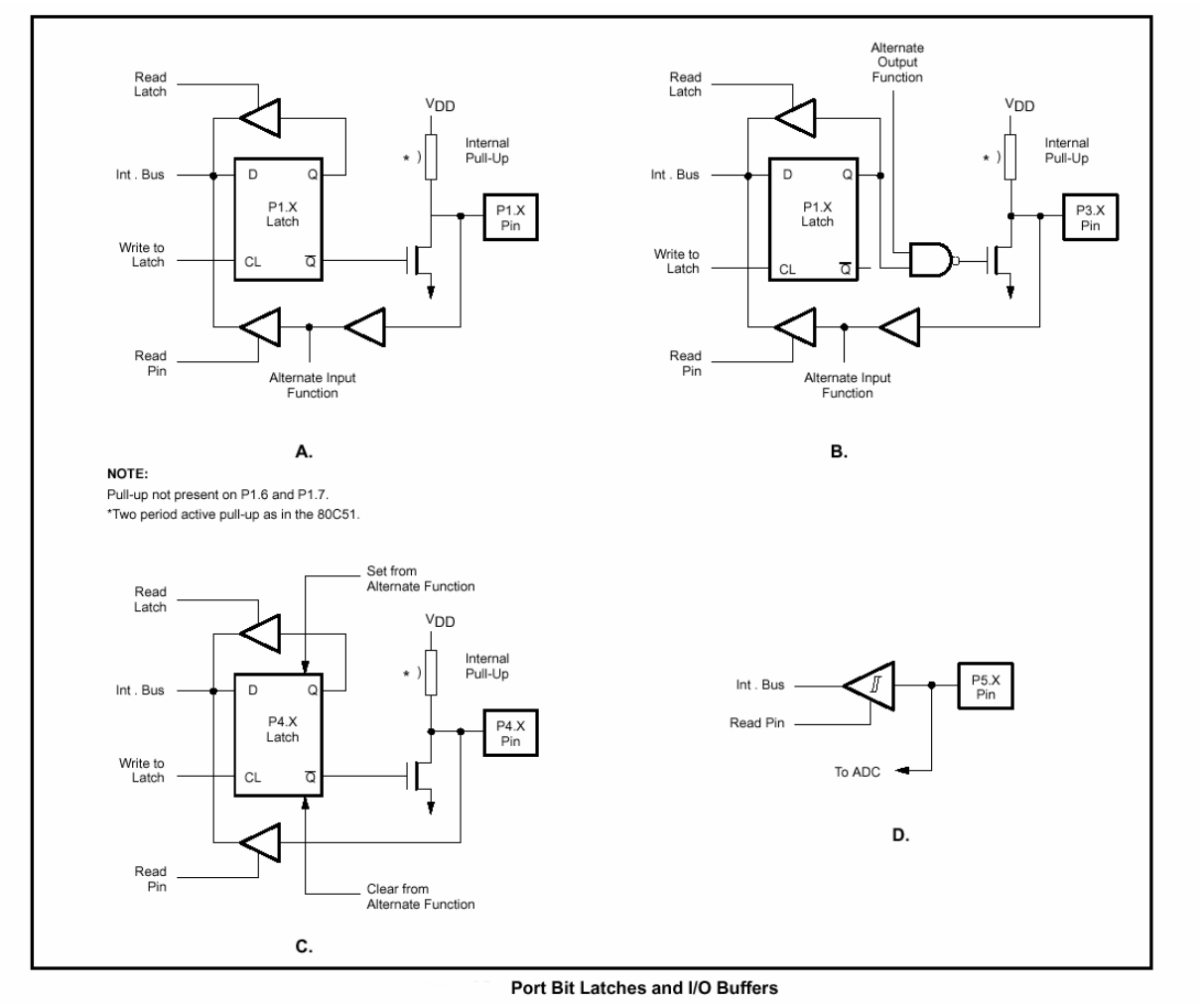
Interrupt Vector Addresses

SOURCE	NAME	VECTOR ADDRESS
External interrupt 0	X0	0003H
Timer 0 overflow	T0	000BH
External interrupt 1	X1	0013H
Timer 1 overflow	T1	001BH
SIO0 (UART)	S0	0023H
SIO1 (I^2C)	S1	002BH
T2 capture 0	CT0	0033H
T2 capture 1	CT1	003BH
T2 capture 2	CT2	0043H
T2 capture 3	CT3	004BH
ADC completion	ADC	0053H
T2 compare 0	CM0	005BH
T2 compare 1	CM1	0063H
T2 compare 2	CM2	006BH
T2 overflow	T2	0073H

```
/* Interrupthandler fuer Timer 0 */
void inttime_0(void) interrupt 1
{
   int  sendbyte;

   /* Timer 0 wieder anstarten */
   TR0 = 0;                    /* Timer anhalten */
   TH0 = TIM0H;    /* Timer neu laden */
   TL0 = TIM0L;
   TR0 = 1;                    /* Timer starten */

sendbyte=(int)buffer[pointer];        /* Datenbyte adressieren */

                    usw.
```

Beispiel eines Interupthandlers in C

Beschaltung der IO-Buffer



Port Bit Latches and I/O Buffers

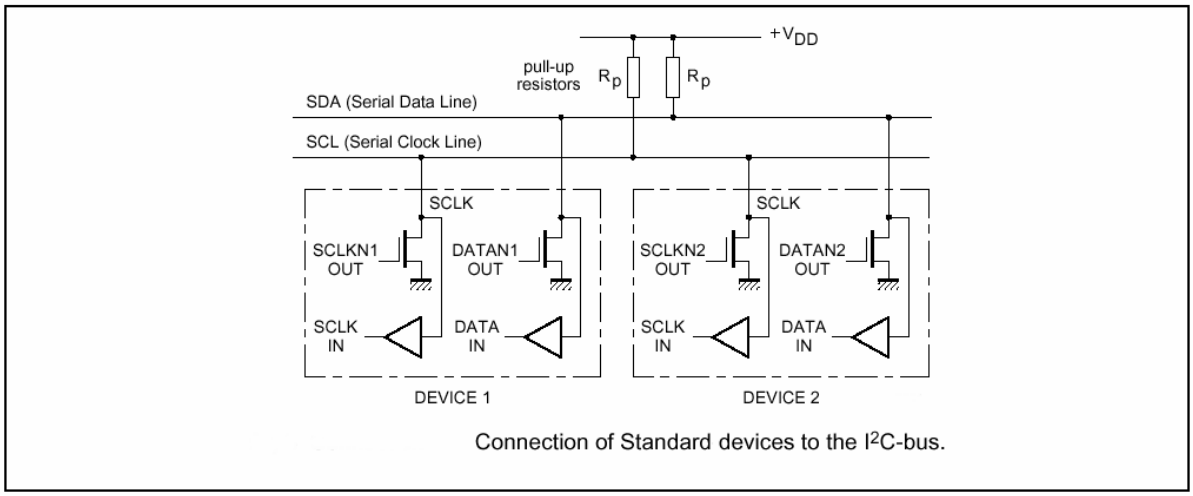
Bei der Programmierung von Port Ein/Ausgabe Funktionalitäten ist beim Einlesen des Ports darauf zu achten, dass zuvor ein High in den Ausgabebuffer geschrieben worden ist (siehe Beispiel). Der Grund liegt in dem Aufbau der IO-Buffer begründet. Die Ports haben eine quasibidirektionale Struktur und man kann in der schematischen Darstellung des Portbuffers (Abbildung B) erkennen, dass bei der Portausgabe der High-Pegel durch den internen Pullup Widerstand realisiert wird, der LOW-Pegel durch das Durchschalten des Transistors. Ist bei Port-eingabe der Ausgangsbuffer auf Low-Pegel programmiert, beeinflusst dies das extern anliegende Signal und es wird ein falscher Wert eingelesen.

```
P4= 0xff;        /* Port 4 für Einlesen vorbereiten */
input= P4;       /* Port 4 einlesen */
```

Beispiel Port Ein-Ausgabe

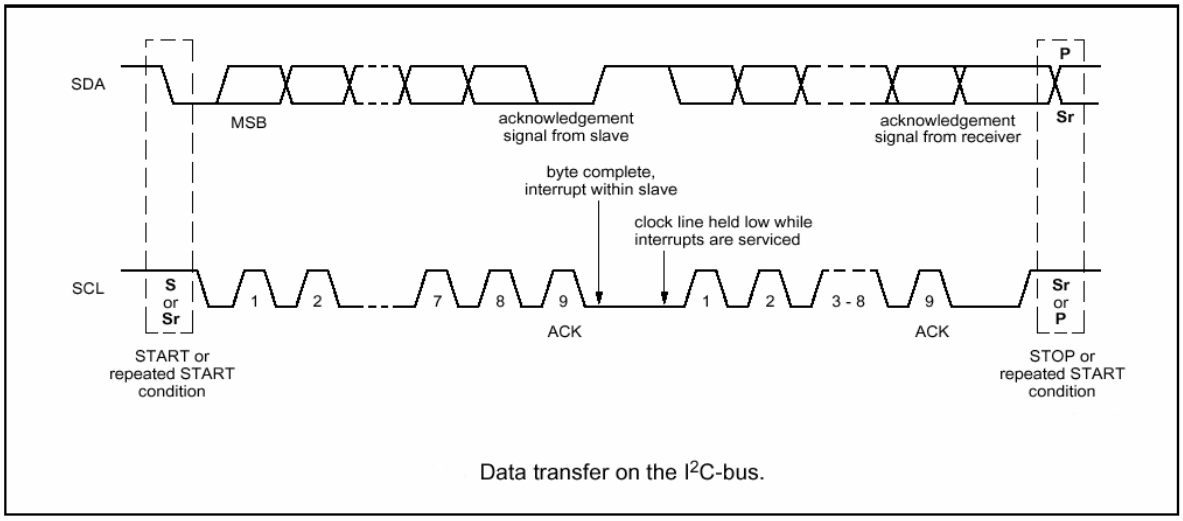
Beschreibung des I2C-Busses

Der I2C-Bus ist ein 2-draht Bus, der in vielen Geräten zum Einsatz kommt (z.B. Fernseher, Video etc.). Es gibt auch eine ganze Anzahl von „General-purpose“ Bausteine mit I2C-Bus Anschluss wie z.B. Ram, Eeprom, Clock/Calender. Der Bus besteht aus einer Clockleitung (SCL) und einer Datenleitung (SDA). Beide Leitungen sind bidirektional und werden extern über einen Pullup Widerstand mit der positiven Versorgungsspannung verbunden. Im Ruhezustand liegen beide Leitungen auf High-Pegel. Der Ausgangsbuffer eines Bausteins muss als Open-Drain oder als Open-Kollektor ausgeführt werden.



Connection of Standard devices to the I²C-bus.

Es muss mindestens ein Master und ein Slave vorhanden sein. Der Slave hat eine feste Adresse und kann so eindeutig vom Master adressiert werden. Es können sowohl mehrere Slave wie auch Master gleichzeitig am Bus betrieben werden. Es findet jedoch zur gleichen Zeit immer nur ein Datentransfer zwischen einem Master und einem Slave statt. Die verbleibenden Master müssen warten bis der Bus wieder frei ist. Probleme bei der Buszuteilung (Arbitration), wenn zwei Master gleichzeitig den Bus benutzen wollen, müssen im Softwaretreiber behandelt werden.
Der 80552 Prozessor unterstützt den I2C-Bus hardwaremäßig, so dass gegenüber einer reinen Software Lösung eine wesentlich höhere Übertragungsrate erzielt wird. Im Praktikum wird eine Übertragungsrate von 100kbit/s benutzt, in der Spezifikation 2.1 des I2C-Busses gibt es über den normalen Modus hinaus noch den Fast- und den Highspeed Modus mit einer Übertragungsrate bis zu max. 3.4 Mbit/s (das Protokoll im Standard- und im Fast-Mode unterscheidet sich nicht, ältere Bausteine können aber nur bis max. 100kbit/s betrieben werden). Im Highspeed-Modus wird ein modifiziertes Busprotokoll benutzt.



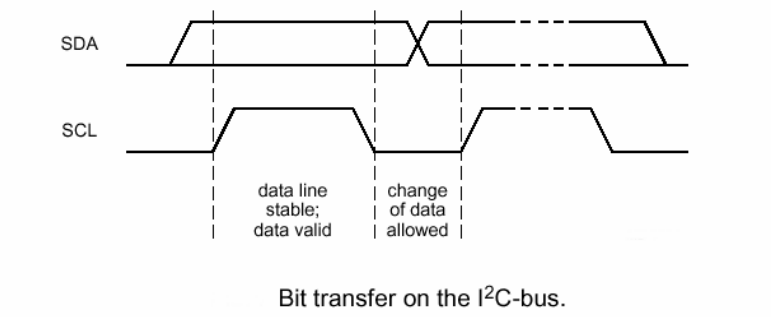
Data transfer on the I²C-bus.

Master Transfer Mode:
Serielle Daten werden über SDA mit Bezug zum Clock SLC ausgegeben. Das erste Byte enthält die Adresse des Slavebaustein der adressiert werden soll. (7bit Adresse und das Datenrichtungsbit). Beim Master Transfer Mode ist das Bit low (SLA+W). Nach jeden Byte wird ein Acknowledge Bit vom Slavebaustein empfangen. Die Zustände START und STOP markieren die Datenübertragungsphase.

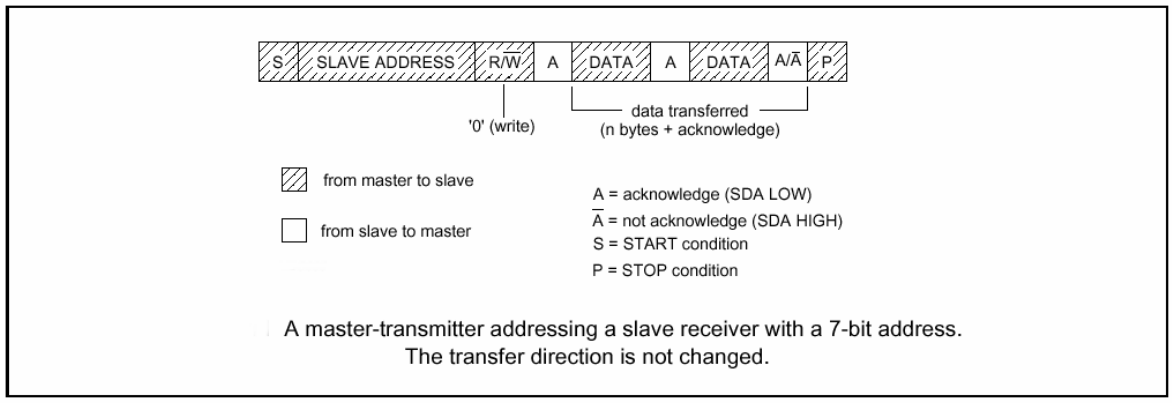
Master Receiver Mode:
Das erste Byte enthält die Adresse des Slavebausteins und das Datenrichtungsbit ist in diesem Fall high (SLA+R). Serielle Daten werden über SDA empfangen während der Takt vom Master mit SCL vorgegeben wird. Es wird jeweils ein Byte mit einen Acknowledge-Bit quittiert. Die Zustände START und STOP markieren den Anfang und das Ende der Datenübertragung.

Slave Receiver Mode:
Serielle Daten werden vom Master über SDA mit taktbezug auf SCL gesendet und vom Slavebaustein empfangen. Ist die für den Slavebaustein gültige Adresse gesendet worden, wird nach dem Empfang eines Bytes ein Ackowledge Bit gesendet. Die START- und STOP- Zustände markieren die Übertragungsphase.

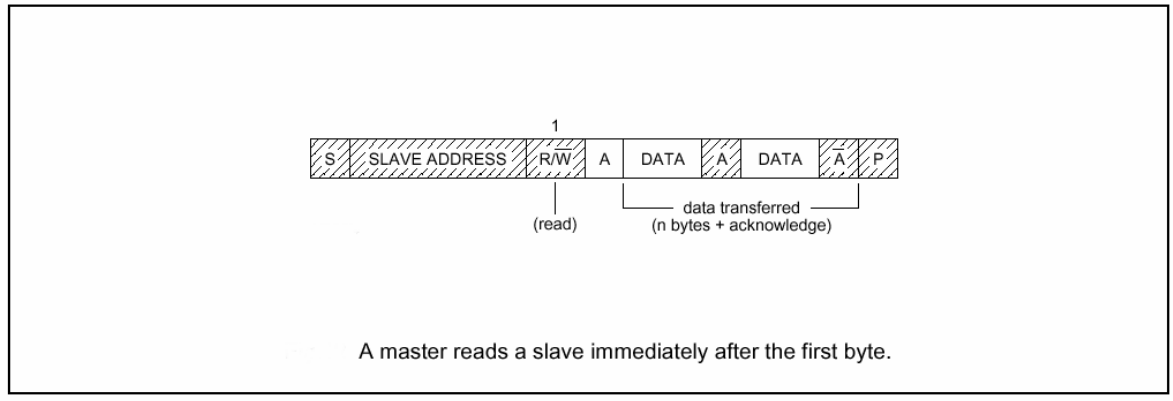
Slave Transmitter Mode:
Nachdem eine für den Slavebaustein gültige Adresse gesendet wurde und das Datenrichtungsbit 1 war (SLA+R) wird die Senderichtung umgekehrt und der Slavebaustein sendet synchron zum vorgegebenen Takt Daten. START- und STOP- Zustand markieren dir Übertragungsphase.



Bit transfer on the I²C-bus.



A master-transmitter addressing a slave receiver with a 7-bit address.
The transfer direction is not changed.



A master reads a slave immediately after the first byte.

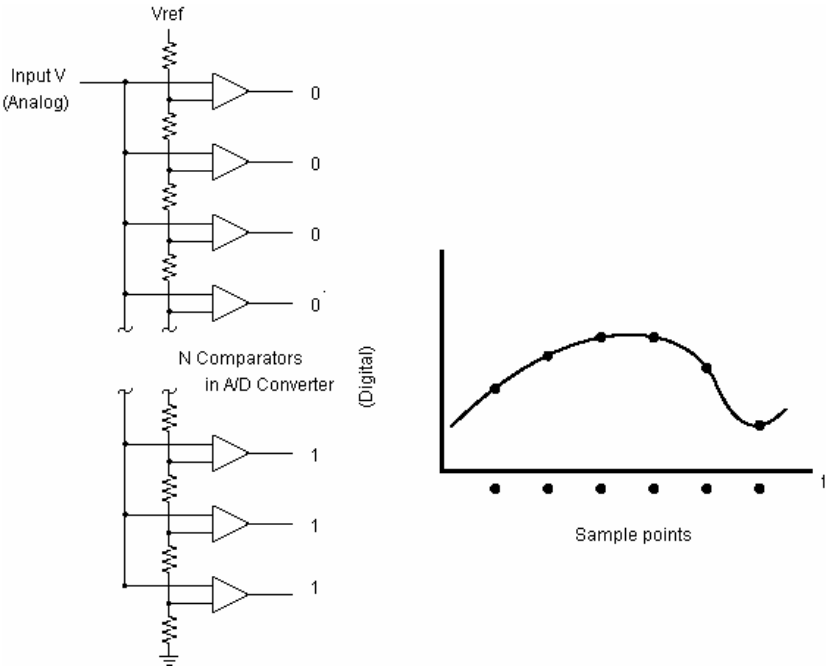
Beschreibung der PWM Ausgänge

Mit Hilfe der Pulsweiten–Modulation Ausgänge PWM0 und PWM1 werden Pulse von programmierbarer Länge und Frequenz erzeugt. Im Praktikum wird über die PWM-Ausgänge ein Servo angesteuert

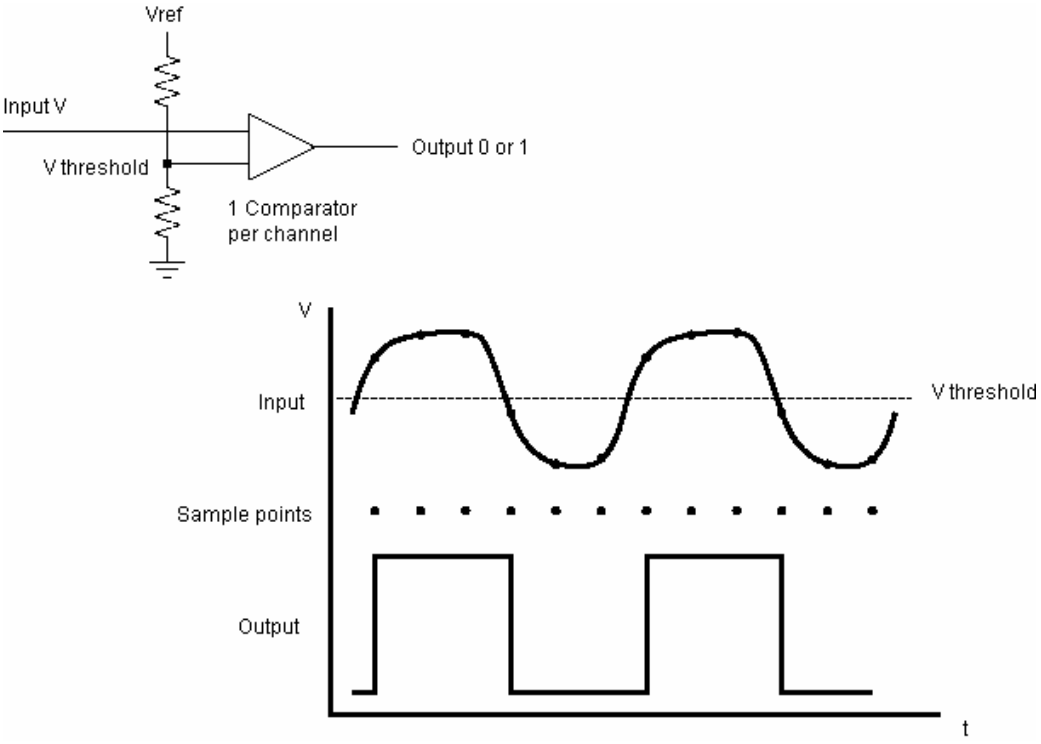
Anhang

Logic analyzers vs. oscilloscopes

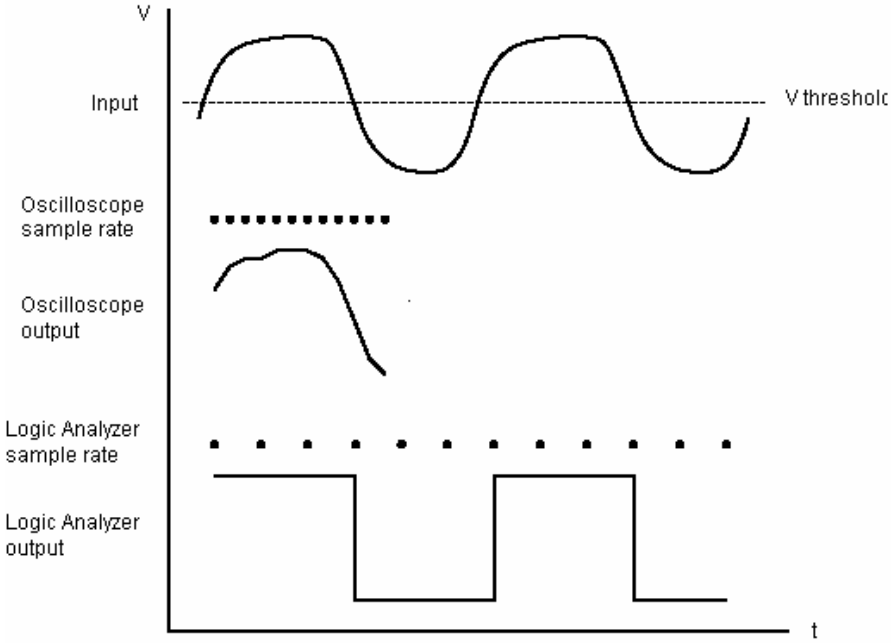
A logic analyzer is an instrument that captures digital signals. It's like a digital oscilloscope, except that it captures logic high and logic low values instead of many voltage levels.
For each channel (or probe) in a digital oscilloscope, many comparators are used to give high resolution on the voltage measured.



For each channel in a logic analyzer, only one comparator is used to detect a logic high or low. With the same number of comparators as an oscilloscope, a logic analyzer can capture data on many more channels.

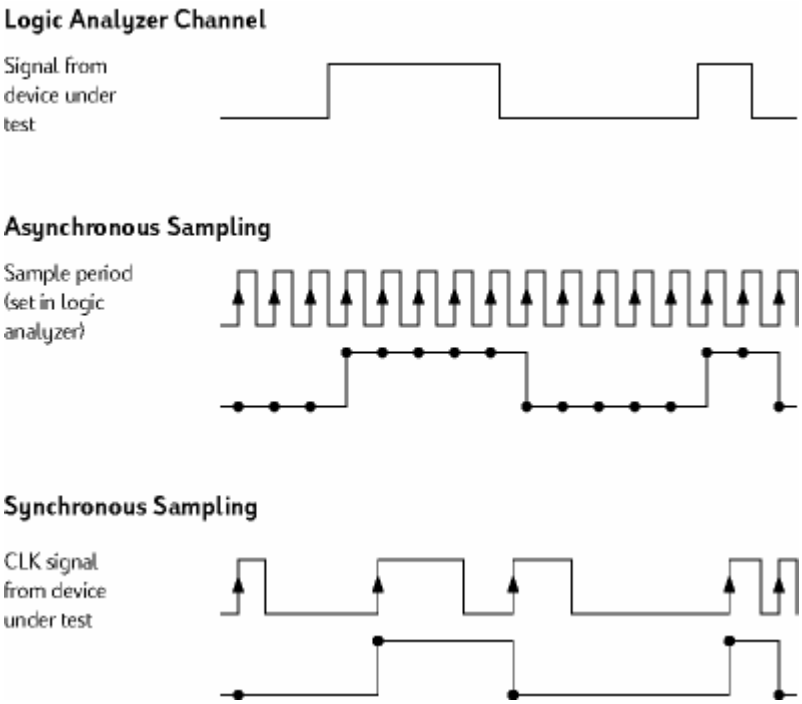


Also, in order to capture the high-frequency, analog component of signals, an oscilloscope must sample at a faster rate than a logic analyzer (although both must sample at a rate faster than the highest frequency of the signals they measure). Given the same amount of memory to store samples, a logic analyzer can capture data over a greater time period than a digital oscilloscope.



An oscilloscope gives a highly detailed view of the parametrics of a few critical signals while a logic analyzer displays the functional relationship of a larger number of signals.

State analysis vs. timing analysis



What is a Timing Analyzer?

In timing mode, samples are acquired from the device under test at regular intervals, such as every 4 nanoseconds. An internal clock is used for sampling.

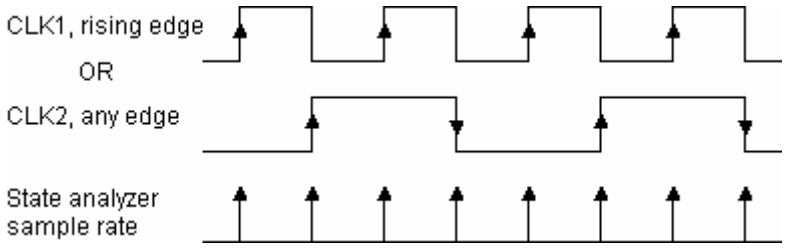
What is a State Analyzer?

In state mode, an event in the device under test, such as a rising edge of a clock signal, indicates when the logic analyzer should take a sample.

NOTE: There is a special clock channel on each pod that is labeled "CLK". These are the only channels that can be used for state mode sampling signals.

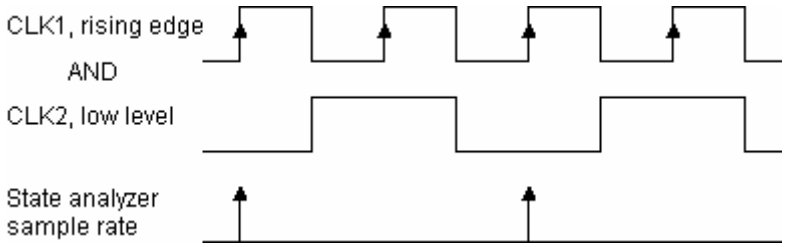
You can use one or both clock channels to provide state mode sampling signals, or you can use the signal on one clock channel to enable or disable the sampling signal on the other clock channel.

State analyzer sampling, OR'ed clocks



Two signals from the device under test are combined to tell the logic analyzer when to sample.

State analyzer sampling, qualified clock



One signal from the device under test enables/disables the sampling clock on the other signal.