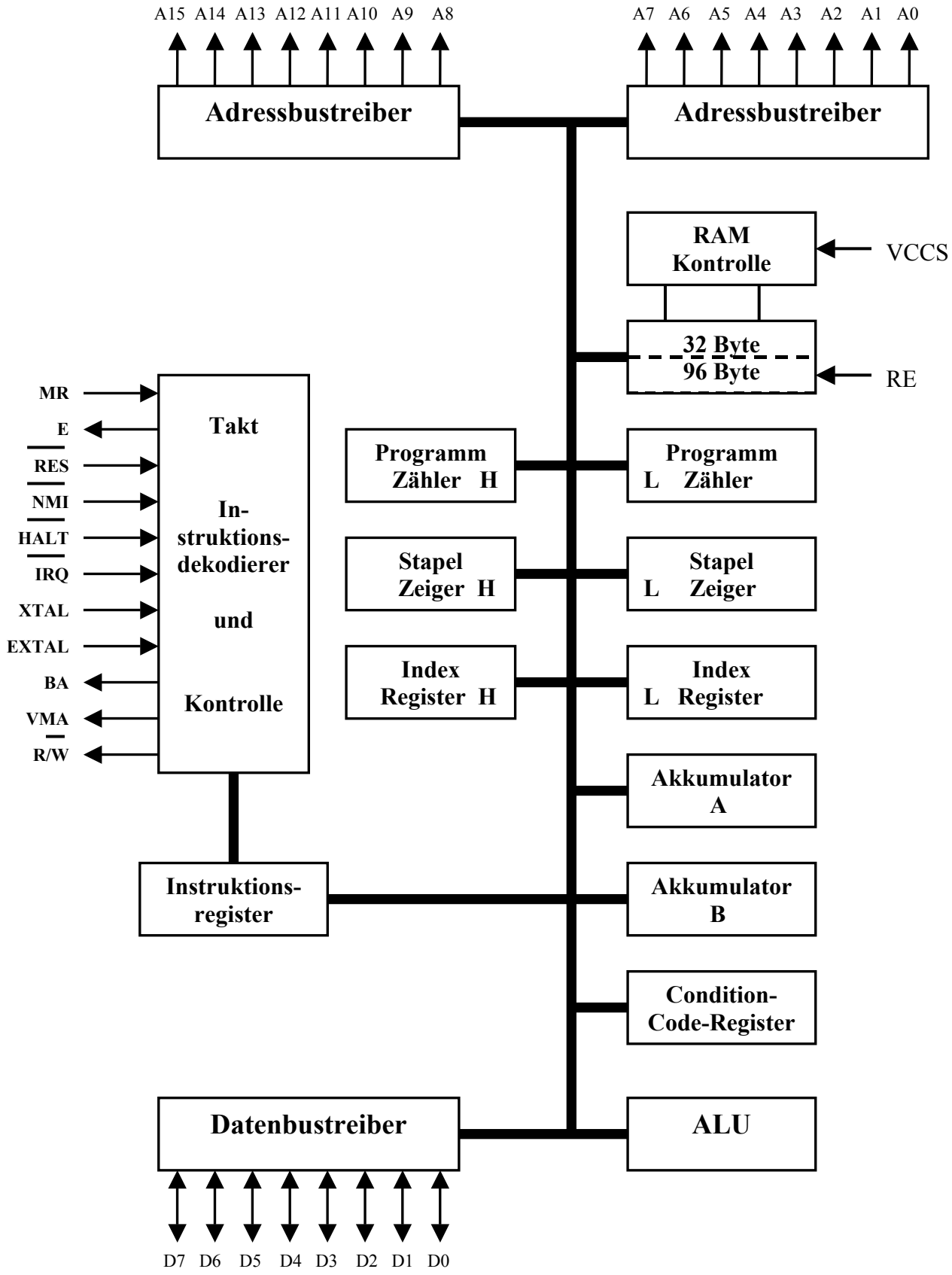
Praktikum

Mikroprozessortechnik
Motorola MC 68xx

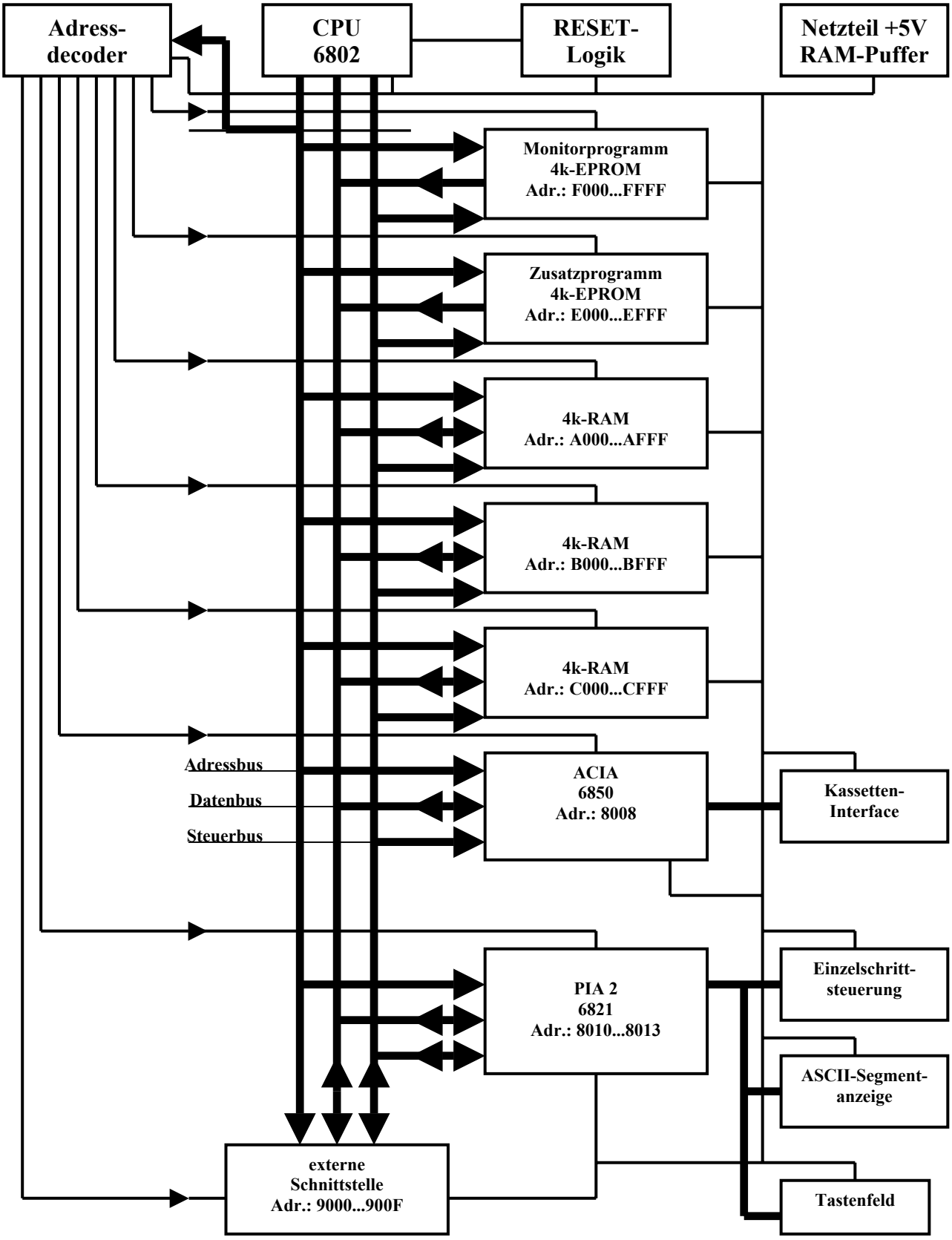
UNIKASSEL VERSITÄT

Dipl.Ing.F.Bollensen und C.Christmann

Blockschaltbild des Mikroprozessors 6802

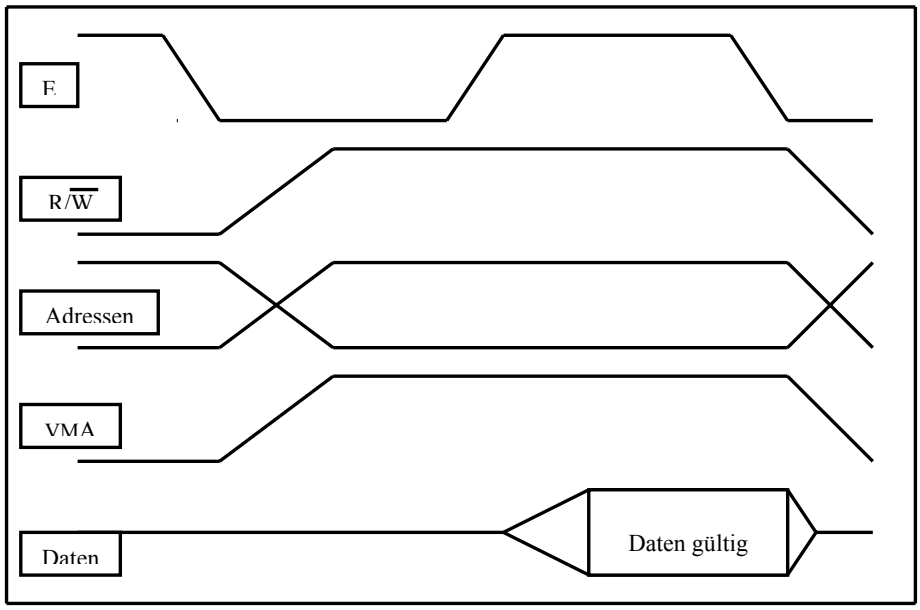


Blockschaltbild des Minicomputers 6802

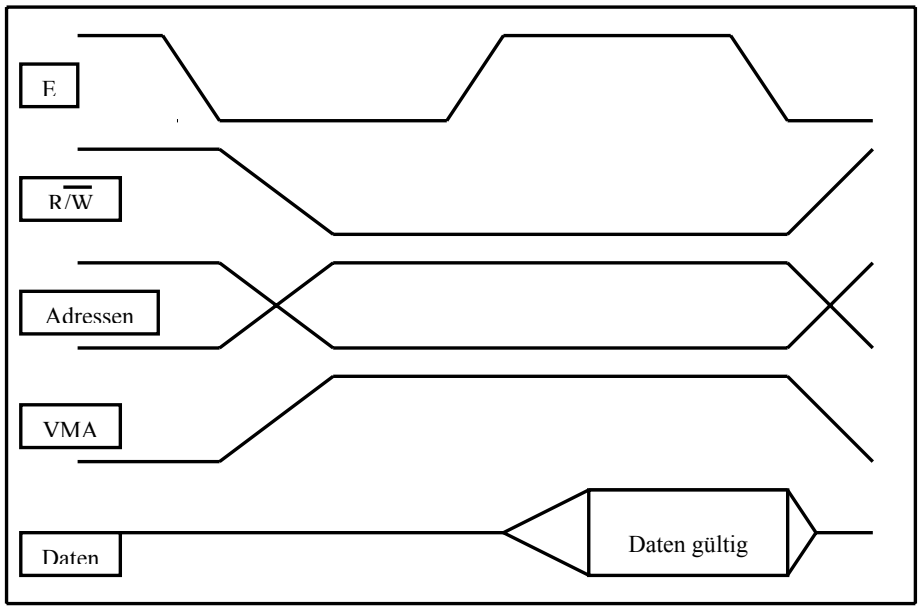


Timingdiagramme des Mikroprozessors MC6802

Lesezyklus



Schreibzyklus



Quelle: Motorola Semiconductor Products

Befehlsliste des Minicomputers 6802

Accumulator and Memory Operations	MNEMONIC	Adressing Modes: IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	INHER OP	INHER ~	INHER #	Boolean and Arithmetic Operation	Cond. Code 5 H	4 I	3 N	2 Z	1 V	0 C
Add	ADDA	88	2	2	9B	3	2	AB	5	2	BB	4	3				A+M → A	↕	•	↕	↕	↕	↕
	ADDB	C8	2	2	DB	3	2	EB	5	2	FB	4	3				B+M → B	↕	•	↕	↕	↕	↕
Add Accumulators	ABA													1B	2	1	A+B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A+C+M → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B+C+M → B	↕	•	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A•M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B•M → B	•	•	↕	↕	R	•
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A•M	•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B•M	•	•	↕	↕	R	•
Clear	CLR							6F	7	2	7F	6	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A-M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B-M	•	•	↕	↕	↕	↕
Compare Accumulators	CBA													11	2	1	A-B	•	•	↕	↕	↕	↕
Complement 1´s	COM							63	7	2	73	6	3				M\ → M	•	•	↕	↕	R	S
	COMA													43	2	1	A\ → A	•	•	↕	↕	R	S
	COMB													53	2	1	B\ → B	•	•	↕	↕	R	S
Complement 2´s (Negate)	NEG							60	7	2	70	6	3				00-M → M	•	•	↕	↕	a	b
	NEGA													40	2	1	00-A → A	•	•	↕	↕	a	b
	NEGB													50	2	1	00-B → B	•	•	↕	↕	a	b
Decimal Adjust, A	DAA													19	2	1	Binary Add for BCD	•	•	↕	↕	↕	c
Decrement	DEC							6A	7	2	7A	6	3				M-1 → M	•	•	↕	↕	d	•
	DECA													4A	2	1	A-1 → A	•	•	↕	↕	d	•
	DECB													5A	2	1	B-1 → B	•	•	↕	↕	d	•
Exclusiv Or	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A (+) M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B (+) M → B	•	•	↕	↕	R	•

Befehlsliste Teil 2/5

Accumulator and Memory Operations	MNEMONIC	Adressing Modes: IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	INHER OP	INHER ~	INHER #	Boolean and Arithmetic Operation	Cond. Code 5 H	4 I	3 N	2 Z	1 V	0 C
Increment	INC							6C	7	2	7C	6	3				M+1 → M	•	•	↕	↕	e	•
	INCA													4C	2	1	A+1 → A	•	•	↕	↕	e	•
	INCB													5C	2	1	B+1 → B	•	•	↕	↕	e	•
Load Accumulator	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A+M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B+M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	A→Msp, SP-1→SP	•	•	•	•	•	•
	PSHB													37	4	1	B→Msp, SP-1→SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP+1→SP, Msp→A	•	•	•	•	•	•
	PULB													33	4	1	SP+1→SP, Msp→B	•	•	•	•	•	•
Rotate Left	ROL							69	7	2	79	6	3				M	•	•	↕	↕	f	↕
	ROLA													49	2	1	A C ← b7..←..b0	•	•	↕	↕	f	↕
	ROLB													59	2	1	B	•	•	↕	↕	f	↕
Rotate Right	ROR							66	7	2	76	6	3				M	•	•	↕	↕	f	↕
	RORA													46	2	1	A C → b7..→..b0	•	•	↕	↕	f	↕
	RORB													56	2	1	B	•	•	↕	↕	f	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	•	•	↕	↕	f	↕
	ASLA													48	2	1	A C ← b7..←..b0 ← 0	•	•	↕	↕	f	↕
	ASLB													58	2	1	B	•	•	↕	↕	f	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	•	•	↕	↕	f	↕
	ASRA													47	2	1	A b7.. →..b0 → C	•	•	↕	↕	f	↕
	ASRB													57	2	1	B	•	•	↕	↕	f	↕
Shift Right, Logic	LSR							64	7	2	74	6	3				M	•	•	R	↕	f	↕
	LSRA													44	2	1	A 0 → b7.. →..b0 → C	•	•	R	↕	f	↕
	LSRB													54	2	1	B	•	•	R	↕	f	↕

Befehlsliste Teil3/5

		Adressing Modes																Cond. Code					
Accumulator and Memory		**IMMED**			**DIRECT**			**INDEX**			**EXTND**			**INHER**			**Boolean and Arithmetic**	5	4	3	2	1	0
Operations	**MNEMONIC**	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	**Operation**	H	I	N	Z	V	C
Store Accumulator	STAA				97	4	2	A7	6	2	B7	5	3				A → M	•	•	↕	↕	R	•
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A-M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B-M → B	•	•	↕	↕	↕	↕
Subtract Accumulators	SBA													10	2	1	A-B → A	•	•	↕	↕	↕	↕
Subtract with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A-M-C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B-M-C → B	•	•	↕	↕	↕	↕
Transfer Accumulators	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M - 00	•	•	↕	↕	R	R
	TSTA													4D	2	1	A - 00	•	•	↕	↕	R	R
	TSTB													5D	2	1	B - 00	•	•	↕	↕	R	R
Compare Index-Register	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				XH → M , XL → M+1	•	•	g	↕	h	•
Decrement Index-Register	DEX													09	4	1	X-1 → X	•	•	•	↕	•	•
Decrement Stack-Pointer	DES													34	4	1	SP-1 → SP	•	•	•	•	•	•
Increment Index-Register	INX													08	4	1	X+1 → X	•	•	•	↕	•	•
Increment Stack-Pointer	INS													31	4	1	SP+1 → SP	•	•	•	•	•	•
Load Index-Register	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				M → XH , M+1 → XL	•	•	i	↕	R	•
Load Stack-Pointer	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				M → SPH , M+1 → SPL	•	•	i	↕	R	•
Store Index-Register	STX				DF	5	2	EF	7	2	FF	6	3				XH → M , XL → M+1	•	•	i	↕	R	•
Store Stack-Pointer	STS				9F	5	2	AF	7	2	BF	6	3				SPH → M , SPL → M+1	•	•	i	↕	R	•
Index-Reg. → Stack-Pntr.	TXS													35	4	1	X-1 → SP	•	•	•	•	•	•
Stack-Pntr. → Index-Reg.	TSX													30	4	1	SP+1 → X	•	•	•	•	•	•

Befehlsliste Teil 4/5

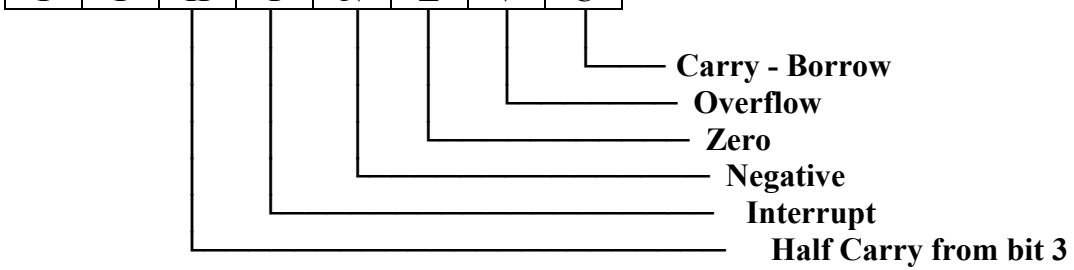
Jump and Branch Operations	MNEMONIC	Adressing Modes: RELATIV OP	RELATIV ~	RELATIV #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	INHER OP	INHER ~	INHER #	Branch Test	Cond. Code 5 H	4 I	3 N	2 Z	1 V	0 C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	4	2										N (+) V = 0	•	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										Z + (N (+) V) = 0	•	•	•	•	•	•
Branch If Higher	BHI	22	4	2										C + Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										Z + (N (+) V) = 1	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										N (+) V = 1	•	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2										See Special Operations	•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3					•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										01	2	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1	See Special Operations	j					
Return From Subroutine	RTS										39	5	1		•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1		•	S	•	•	•	•
Wait for Interrupt	WAI										3E	9	1		•	k	•	•	•	•

Befehlsliste Teil 5/5

Conditions Code Reg. Operations	MNEMONIC	INHER OP	~	#	Boolean Operation	5 H	4 I	3 N	2 Z	1 V	0 C
Clear Carry	CLC	0C	2	1	0 → C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	0 → I	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	0 → V	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	1 → C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	1 → I	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	1 → V	•	•	•	•	S	•
Accu A → CCR	TAP	06	2	1	A → CCR	l					
CCR → Accu A	TPA	07	2	1	CCR → A	•	•	•	•	•	•

Condition Code Register

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
1	1	H	I	N	Z	V	C



Condition Code Register Notes

a) (Bit V) Test: Result = 1000000 ?
b) (Bit C) Test: Result = 00000000 ?
c) (Bit C) Test: Decimal value of most significant BCD Character for greater then nine ? (Not cleared if previously set)
d) (Bit V) Test: Operant = 10000000 prior to execution ?
e) (Bit V) Test: Operant = 01111111 prior to execution ?
f) (Bit V) Test: Set equal to result of N(+)C after shift has occurred.
g) (Bit N) Test: Sign bit of most significant byte of result = 1 ?
h) (Bit V) Test: 2´s complement overflow from subtraction of last significant bytes?
i) (Bit N) Test: Result less then zero
j) (All) Load Condition-Code-Register from Stack.
k) (Bit I) Set when interrupt occurs if previously set a Non-Maskable-Interrupt is required to the wait state.
l) (All) Set according to the contents of Accumulator A.

SP	Stack
SP	Condition Code
SP+1	Accumulator B
SP+2	Accumulator A
SP+3	Index-Register X-High
SP+4	Index-Register X-Low
SP+5	Program-Counter-High
SP+6	Program-Counter-Low
→ SP+7	

Benötigte Unterprogramme für den Minicomputer 6802

1) <Disp4> Einsprungadresse: $F831
Stellt den Code des ASCII-Puffers auf der Anzeige dar.
Tastatureingaben werden nicht ausgewertet.

2) <Wait> Einsprungadresse: $F876
Bringt den in ASCII-Zeichen gewandelten Inhalt des ASCII-Puffers auf die Anzeige.
Ferner wird auf das betätigen einer beliebigen Taste gewartet.

3) <Binäranzeige> Einsprungadresse: $F884
Zeigt den im Speicherplatz $A818 stehenden Wert binär auf dem Display an und wartet auf einen Tastendruck.

4) <Hexadezimalanzeige> Einsprungadresse: $F89F
Zeigt den im Speicherplatz $A818 stehenden Wert in hexadezimaler Form auf dem Display an und wartet auf einen Tastendruck.

5) <Shif88> Einsprungadresse: $F902
Löscht die Anzeige.

6) <Inbyte> Einsprungadresse: $F97C
Teilt ein Byte in zwei hexadezimale Ziffern. Anschließend werden diese in den ASCII-Code umgewandelt und von rechts in den ASCII-Puffer geschoben. Die Adresse des zu konvertierenden Bytes ist zuvor in das x-Register zu schreiben.

7) <LText> Einsprungadresse: $FA1B
Acht beliebige ASCII-Zeichen werden in den ASCII-Puffer (Adressen: $A7B0 ... $A7B7) abgespeichert.
Die Adresse des ersten ASCII-Zeichens muss zuvor im X-Register abgelegt worden sein.

Hauptkontrollschleife („HKS“) Startadresse: $FA54
Sofern nicht anders angegeben, müssen die Programme mit einem Sprung in die HKS ordnungsgemäß beendet werden.

Arbeitsblatt zur Einführung in die Programmierung des 6802-Minicomputers

Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	
		A000				

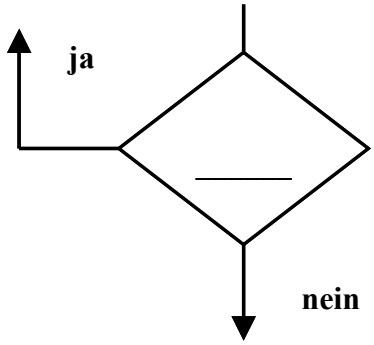
Einführungsprogramm 1: Bedingte Sprungbefehle (Branch If)

Übersetzen Sie das nachfolgende Programm.
Beschreiben Sie die benutzen Befehle im Feld Bemerkung.

Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	
Start	LDAA #$3C	A000				
Neu	INC A					
	CMPA #$40					
	BNE „Neu“					
	JMP „HKS“					

Welches Flag des Condition-Code-Registers wird durch den Befehl „Branch Not Equal“ (BNE) abgefragt?

- Tragen Sie das abgefragte Flag und dessen Status in das nebenstehende Verzweigungsfeld ein.
- Beobachten Sie Akku A, das durch „BNE“ abgefragte Flag und die Verzweigungsrichtung.



AKKU A	___-Flag	Richtung

Welche Rechenoperation führt der Befehl CMP A aus?
- Führen Sie die Rechenoperation binär aus.

Rechenoperation beim 1.Programmdurchlauf

	b7	b6	b5	b4	b3	b2	b1	b0	___-Flag
AKKU A									
40hex	0	1	0	0	0	0	0	0	
Ergebnis									

Rechenoperation beim 2.Programmdurchlauf

	b7	b6	b5	b4	b3	b2	b1	b0	___-Flag
AKKU A									
40hex	0	1	0	0	0	0	0	0	
Ergebnis									

Rechenoperation beim 3.Programmdurchlauf

	b7	b6	b5	b4	b3	b2	b1	b0	___-Flag
AKKU A									
40hex	0	1	0	0	0	0	0	0	
Ergebnis									

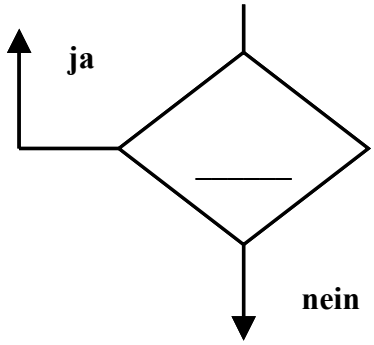
Einführungsprogramm 2: Bedingte Sprungbefehle (Branch If)

Übersetzen Sie das nachfolgende Programm.
Beschreiben Sie die benutzen Befehle im Feld Bemerkung.

Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	
Start	LDAA #$20	A000				
Neu	ASL A					
	BIT A #$80					
	BEQ „Neu“					
	JMP „HKS“					

Welches Flag des Condition-Code-Registers wird durch den Befehl „Branch Equal“ (BEQ) abgefragt?

- Tragen Sie das abgefragte Flag und dessen Status in das nebenstehende Verzweigungsfeld ein.
- Beobachten Sie Akku A, das durch „BEQ“ abgefragte Flag und die Verzweigungsrichtung.



AKKU A	___-Flag	Richtung
20hex		

Welche Rechenoperation führt der Bit-Test aus?
- Führen Sie die Rechenoperation binär aus.

Rechenoperation beim 1.Programmdurchlauf

	b7	b6	b5	b4	b3	b2	b1	b0	Z Flag
AKKU A									
80hex	1	0	0	0	0	0	0	0	
Ergebnis									

Rechenoperation beim 2.Programmdurchlauf

	b7	b6	b5	b4	b3	b2	b1	b0	Z Flag
AKKU A									
80hex	1	0	0	0	0	0	0	0	
Ergebnis									

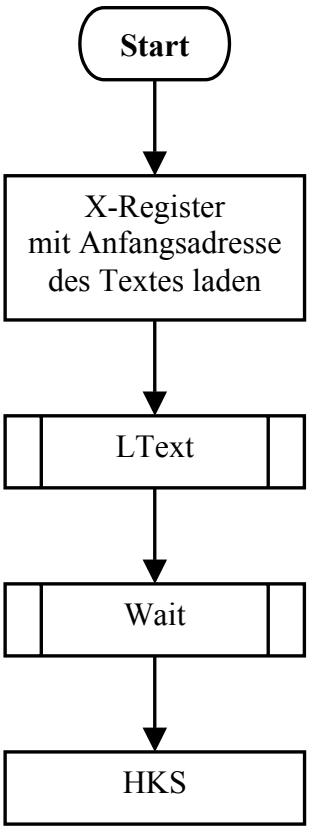
Rechenoperation beim 3.Programmdurchlauf

	b7	b6	b5	b4	b3	b2	b1	b0	Z Flag
AKKU A									
80hex	1	0	0	0	0	0	0	0	
Ergebnis									

Programm 1: Anzeige von ASCII-Text

Das Wort Beispiel soll auf der Anzeige des Minicomputers dargestellt werden. Hierfür werden die Unterprogramme <LText> und <Wait> verwendet. Das Wort ist in seine Buchstaben zu zerlegen und deren ASCII-Codes ab Adresse $A100 abzuspeichern.
(Hinweis: „B: $42“, „E: $45“, „I: $49“, „L: $4C“, „P: $50“, „S: $53“)

Org.: $A000



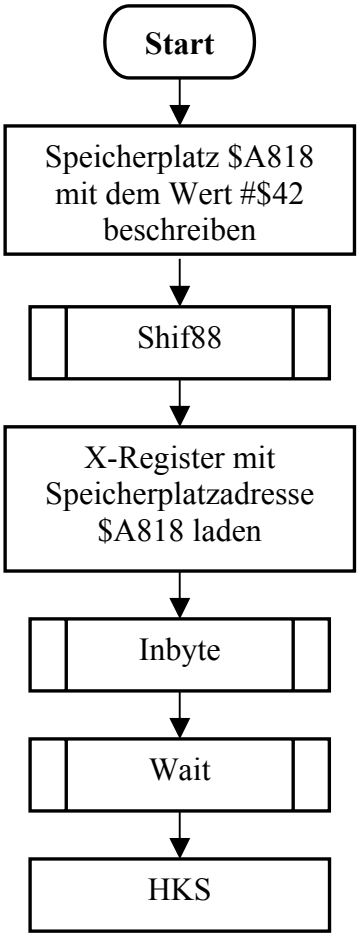
Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	
		A000				

Adresse	ASCII	Zeichen
A100	42	B

Adresse	ASCII	Zeichen

Programm 2: Umwandlung von acht Dual- in zwei Hexadezimalziffern

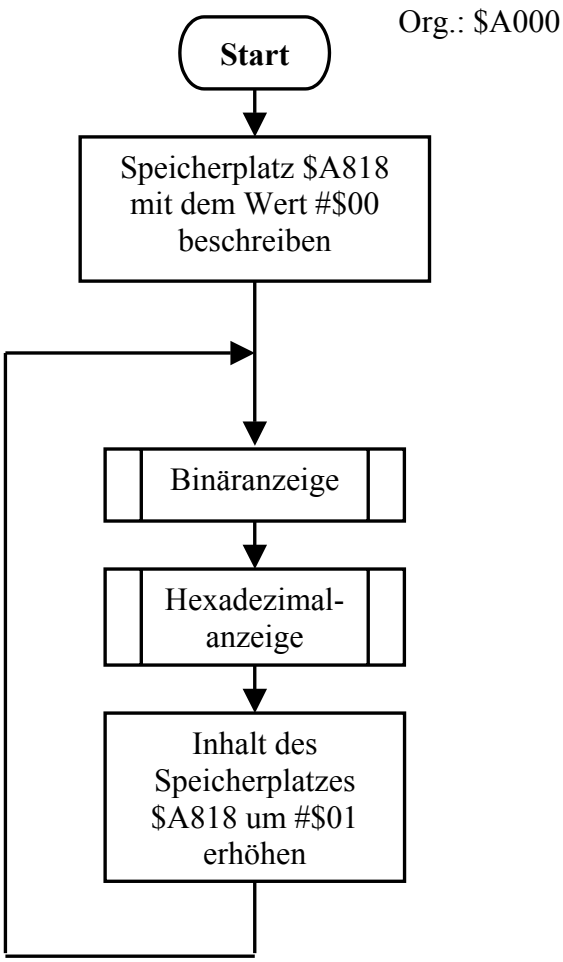
Der Speicherplatz $A818 soll den Wert $42 erhalten. Dieser Wert ist auf dem Display sichtbar zu machen. Folgende Unterprogramme sollen dafür verwendet werden: <Shif88>, <Inbyte>, <Wait>.



Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	

Programm 3: Zahlendarstellung im Dual- und Hexadezimalcode

Der Speicherplatz $A818 (Platzhalter für Binär- und Hexadezimalanzeige) soll durch das Programm beschrieben werden. Durch die Eingabe von „Go A000“ bzw. durch das Betätigen der Taste „Z“ (Adresse für den Sprungbefehl: $A812) soll das Programm gestartet werden können. Der Inhalt von $A818 ist binär und hexadezimal anzuzeigen. Zu benutzen sind die Unterprogramme <Binäranzeige> und <Hexadezimalanzeige>.

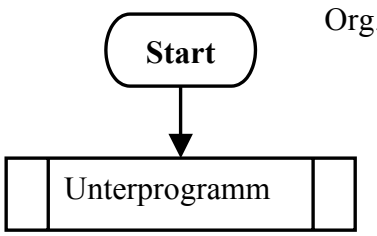


Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	

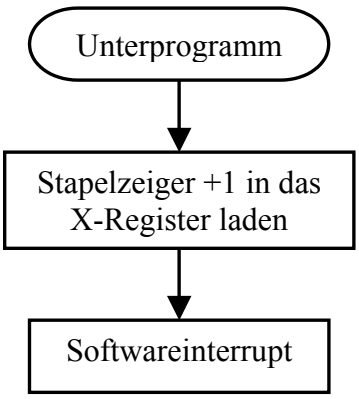
Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	
Z-Taste		A812				

Programm 4: Untersuchung des Stapels

4.1. Anzeige des Stapelzeigers mit Hilfe des Softwareinterrupts sowie des X-Registers.



Org.: $A000



Org.: $A100

Hauptprogramm:

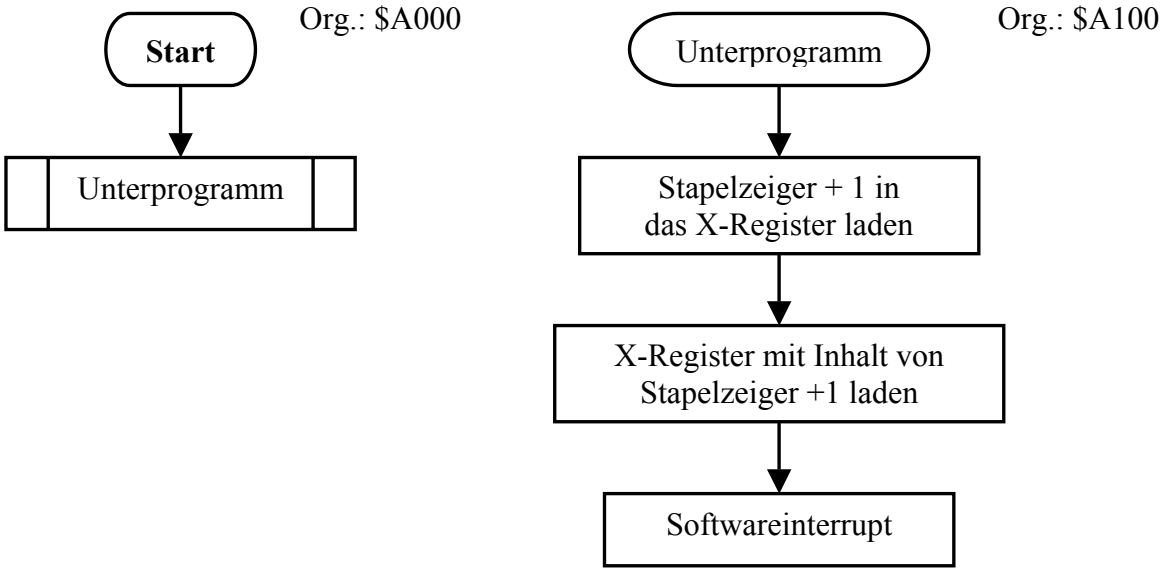
Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	
		A000				

Unterprogramm:

Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	
		A100				

Register	Inhalt	Bemerkung
X-Register		
Stapelzeiger		

4.2. Anzeige der Rücksprungadresse (Inhalt des Stapelzeigers) mit Hilfe des Softwareinterrupts und des X-Registers.



Hauptprogramm:

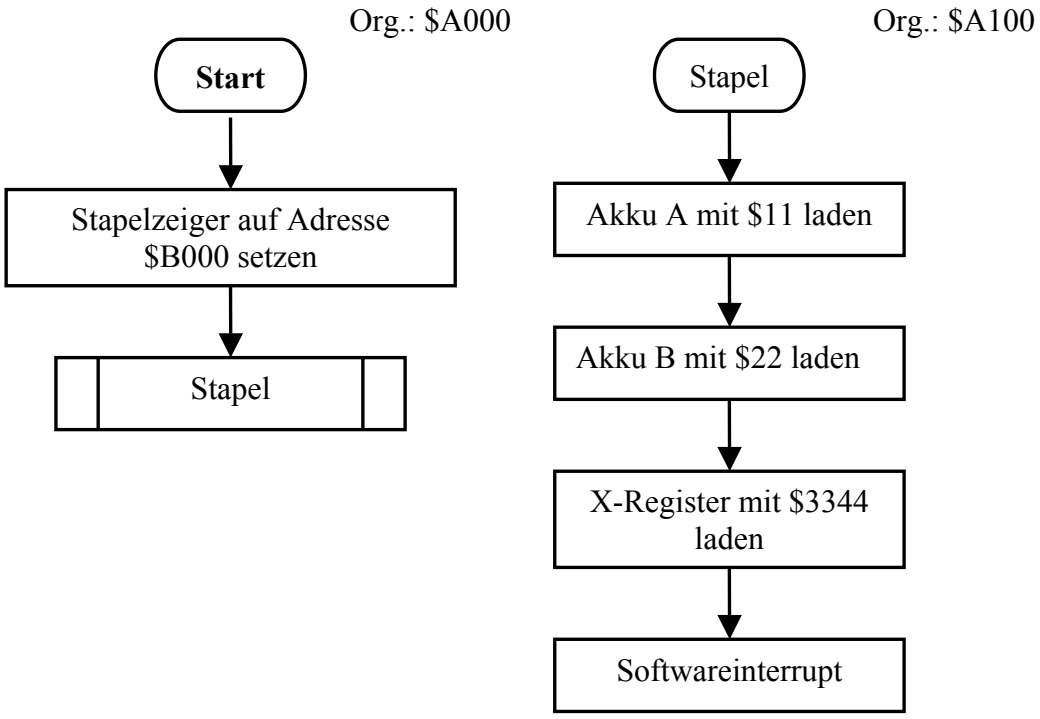
Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	
		A000				

Unterprogramm:

Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	
		A100				

Register	Inhalt	Bemerkung
X-Register		
Stapelzeiger		

4.3. Untersuchung des Stapels und Neudefinition des Stapelzeigers



Hauptprogramm:

Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	
Start		A000				

Unterprogramm:

Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	
		A100				

Stapel:

Adresse	Byte	Bemerkung
B000		

Adresse	Byte	Bemerkung

Programm 5: Ein Programm kopiert sich selbst

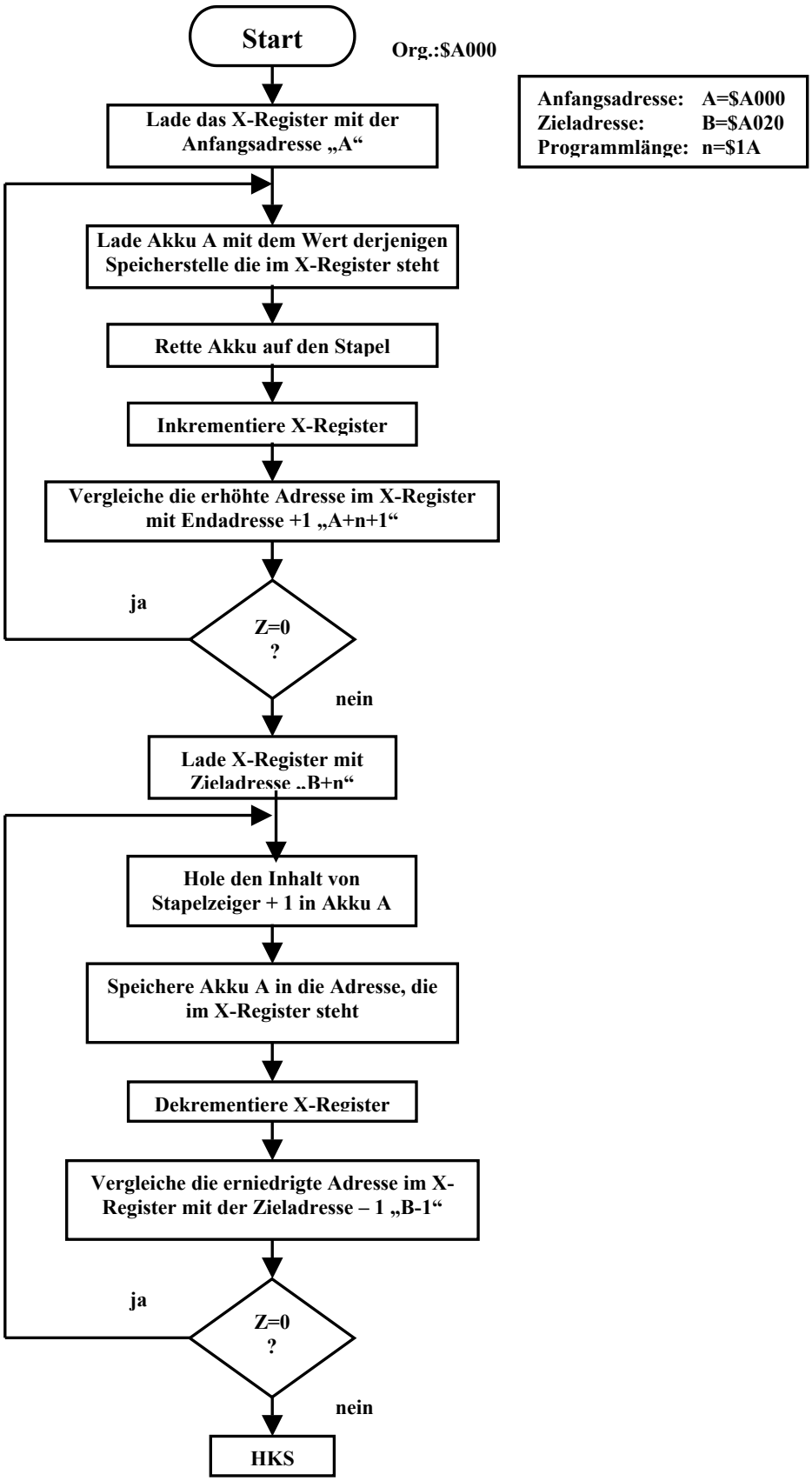
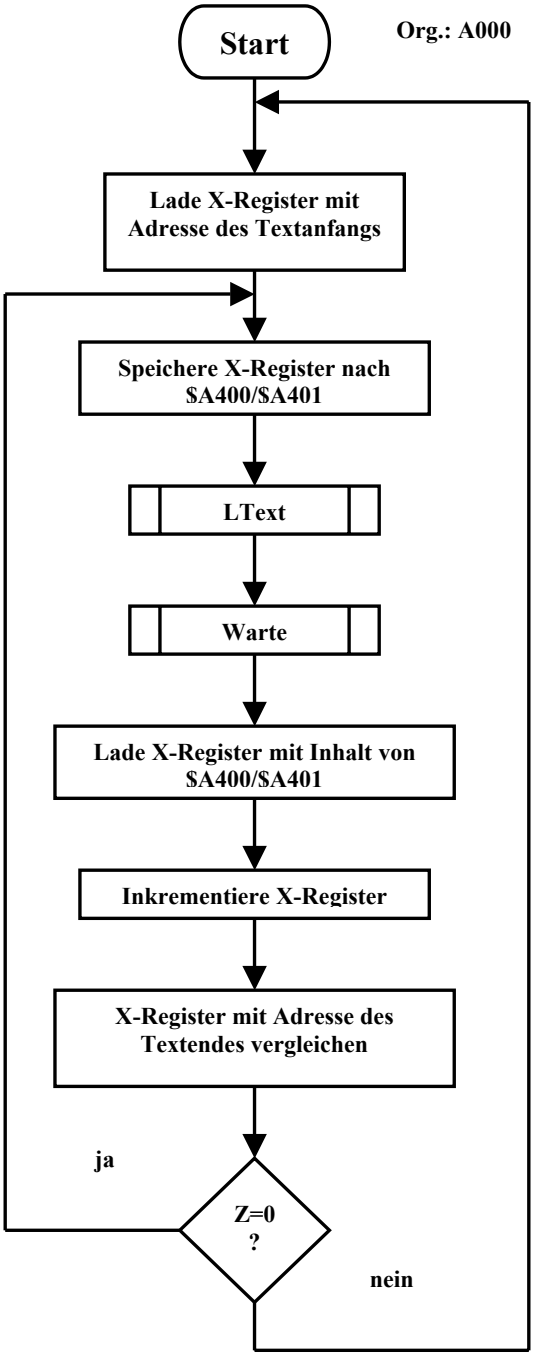


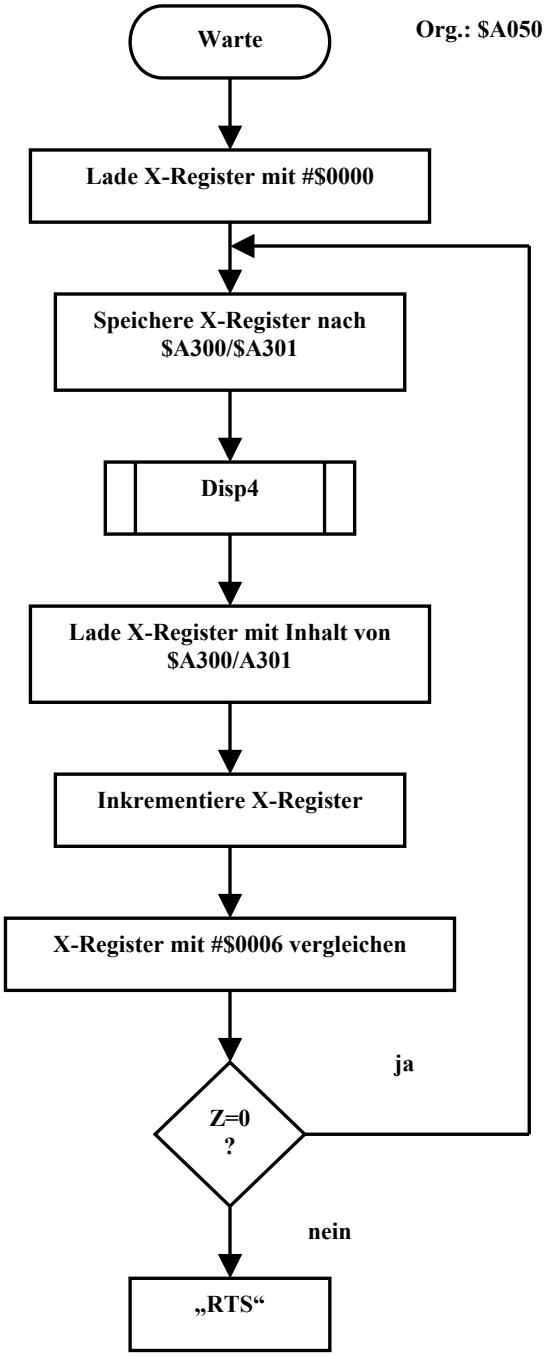
Tabelle für Programm 5

Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	

Programm 6: Laufschrift

Auf dem Display soll kontinuierlich eine Laufschrift erscheinen. Der darzustellende Text (ASCII-Code) ist in einer Tabelle ab Adresse $A100 abzulegen. Die Adresse des Textanfangs (**$A100**), sowie die Adresse des Textendes (**$A121**)müssen im Programm enthalten sein.
Hinweis: Am Anfang des Textes sowie hinter dem Textende sollten 8 Leerzeichen stehen, um die Anzeige zu löschen.





Funktionsweise des Laufschriftprogramms

In einem Beispiel wird die Laufschrift „Uni Kassel“ auf dem Display erzeugt.
Der Text liegt im ASCII-Code ab Adresse $B000 im RAM, das X-Register übergibt dem Unterprogramm „LTEXT“ die Adresse, von der an 8 ASCII-Zeichen auf dem Display dargestellt werden sollen. Die sichtbaren Zeichen auf dem Display sind hier grau hinterlegt gezeichnet.

X-Reg. / Adresse	$B000	$B001	$B002	$B003	$B004	$B005	$B006	$B007	$B008	$B009	$B00A	$B00B	$B00C	$B00D	$B00E	$B00F	$B010	$B011	$B012	$B013	$B014	$B015	$B016	$B017	$B018
$B000									U	N	I		K	A	S	S	E	L							
$B001									U	N	I		K	A	S	S	E	L							
$B002									U	N	I		K	A	S	S	E	L							
$B003									U	N	I		K	A	S	S	E	L							
$B004									U	N	I		K	A	S	S	E	L							
$B005									U	N	I		K	A	S	S	E	L							
$B006									U	N	I		K	A	S	S	E	L							
$B007									U	N	I		K	A	S	S	E	L							
$B008									U	N	I		K	A	S	S	E	L							
$B009									U	N	I		K	A	S	S	E	L							
$B00A									U	N	I		K	A	S	S	E	L							
$B00B									U	N	I		K	A	S	S	E	L							
$B00C									U	N	I		K	A	S	S	E	L							
$B00D									U	N	I		K	A	S	S	E	L							
$B00E									U	N	I		K	A	S	S	E	L							
$B00F									U	N	I		K	A	S	S	E	L							
$B010									U	N	I		K	A	S	S	E	L							
$B011									U	N	I		K	A	S	S	E	L							
$B000									U	N	I		K	A	S	S	E	L							
$B001									U	N	I		K	A	S	S	E	L							
$B002									U	N	I		K	A	S	S	E	L							

Die Laufschrift wandert wie ein Band von rechts nach links über das Display.
Das Display, dass 8-Zeichen aus dem RAM sichtbar macht bewegt sich wie ein Fenster von links nach rechts über das RAM.
Es wird als mit Hilfe des Programm eine Schiebe-Funktion (shift) ausgeführt.

Tabelle für Programm 6

Hauptprogramm

Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	

Unterprogramm

Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	

Anzeigetext

Adresse	ASCII	Zeichen
A100…7	20	blank
A108	4D	M
A109	49	I
A10A	4B	K
A10B	52	R
A10C	4F	O
A10D	50	P
A10E	52	R
A10F	4F	O
A110	5A	Z
A111	45	E
A112	53	S
A113	53	S
A114	4F	O

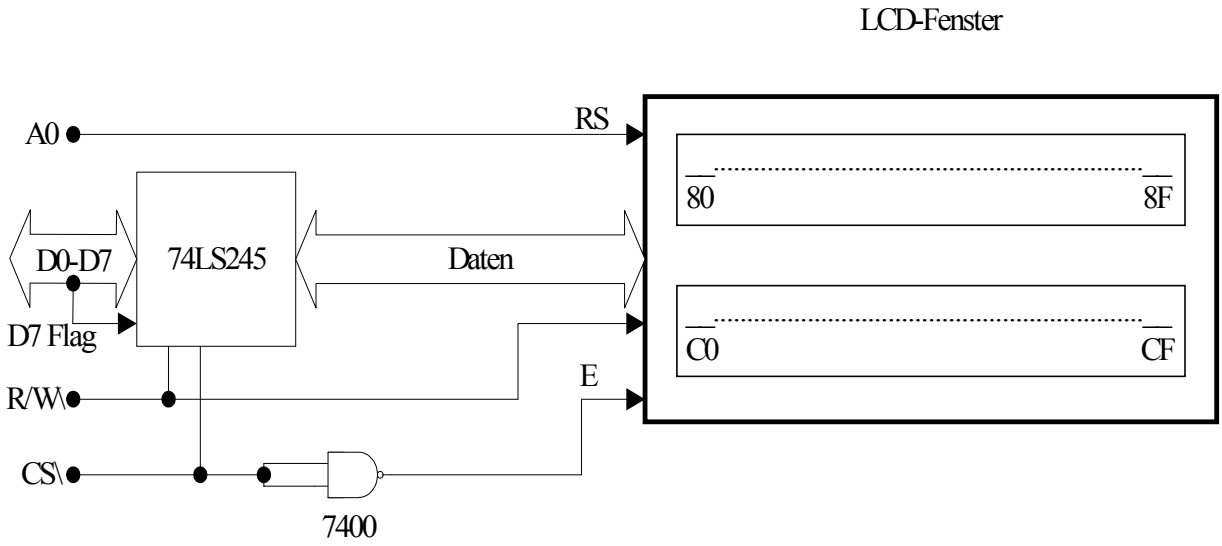
Adresse	ASCII	Zeichen
A115	52	R
A116	54	T
A117	45	E
A118	43	C
A119	48	H
A11A	4E	N
A11B	49	I
A11C	4B	K
A11D	4C	L
A11E	41	A
A11F	42	B
A120	4F	O
A121	52	R
A122…9	20	blank

Programm 7: Ansteuerung einer LCD-Anzeige

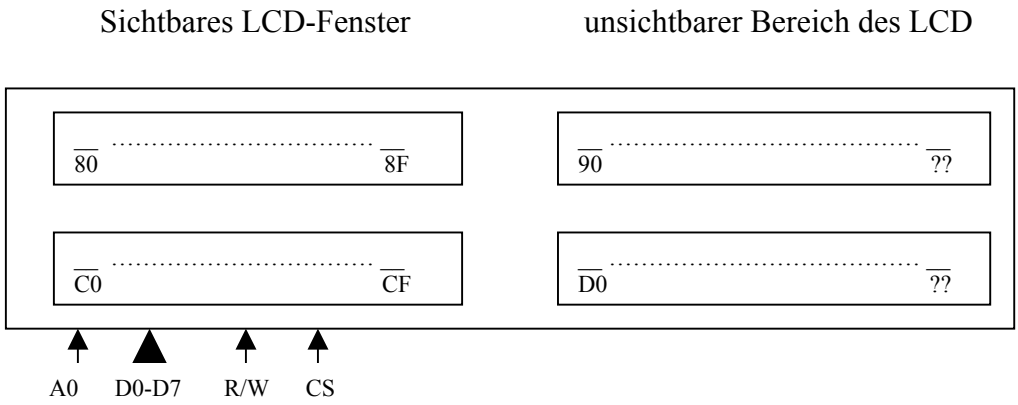
Der Text „**Steuerwerke u. Rechenwerke**“ soll im Display angezeigt werden

Vorgehensweise: Display unter Verwendung der grau hinterlegten Intialisierungsbefehle initialisieren.
Text in Form der einzelnen Buchstaben zum Display senden.
Überprüfung des Busyflag (D7=0 ?) nach jedem Senden zum Display.

- Testen Sie das Programm im Step- Betrieb.
- Überprüfen Sie die Speichertiefe des Displays.
- Testen Sie die Funktionen des Cursors.
- Testen Sie die Shift Funktionen des Displays.
- Laden Sie Daten vom Display zum Minicomputer zurück.



CS\ Basisadresse = $9000



Initialisierung

Die Initialisierung legt die Betriebsart des LC-Displays fest.
Initialisierungsbefehle werden grundsätzlich zur Adresse $9000 geschickt (A0=RS=0).

Initialisierungsbefehle:

Clear Display:

Beschreibt alle DD-RAM Speicherstellen mit Leerzeichen ($20 hex).
Der Cursor wird auf die Position $80 hex gesetzt und das Anzeigefenster beginnt bei Adresse $80 hex.

$01 HEX	Clear Display

Return Home

Setzt den Cursor die Position $80 hex. Der Inhalt des DD-RAM bleibt unverändert.

$02,$03 HEX	Return Home

Entry Mode Set

Festlegung ,ob die interne DD- RAM Adresse beim Schreiben oder Lesen eines Zeichens in das DD-RAM automatisch erhöht (Inkrement) oder erniedrigt (Dekrement) wird und ob beim Schreiben in das DD- RAM automatisch ein Display- Shift erfolgen soll. Der Cursor behält dann innerhalb des Displayfensters die gleiche Position bei.

$04 HEX	Dekrement
$05 HEX	Dekrement & Shift
$06 HEX	Inkrement
$07 HEX	Inkrement & Shift

Display ON – OFF Control

Ein- bzw. aus schalten der Anzeige und des Cursor, ohne den Inhalt des DD- RAM zu verändern.

$08 bis $0B HEX	Display AUS		
$0C HEX	Display EIN	Cursor AUS	Blinken AUS
$0D HEX	Display EIN	Cursor AUS	Blinken EIN
$0E HEX	Display EIN	Cursor EIN	Blinken AUS
$0F HEX	Display EIN	Cursor EIN	Blinken EIN

Cursor oder Display Shift

Bewegen des Cursor oder verschieben des Displays.

$01C bis $01F HEX	Display Shift	Nach rechts
$018 bis $01B HEX	Display Shift	Nach links
$014 bis $017 HEX	Cursor Move	Nach rechts
$010 bis $013 HEX	Cursor Move	Nach links

Funktion Set

Festlegung der Funktionsart des Displays nach dem Einschaltreset.

$03C bis $03F HEX	8 Bit	Zwei Zeilen	5*10 Punkte
$038 bis $03B HE	8 Bit	Zwei Zeilen	5*7 Punkte
$034 bis $037 HEX	8 Bit	Eine Zeile	5*10 Punkte
$030 bis $033 HEX	8 Bit	Eine Zeile	5*7 Punkte
$020 bis $02C HEX	4 Bit		

Zeichensatz des LC-Displays

	obere 4-Bit (D4 bis D7) des Zeichencodes (hexadezimal)						
		2	3	4	5	6	7
untere 4-Bit (D0 bis D3) des Zeichencodes (hexadezimal)	0		0	@	P	\	p
	1	!	1	A	Q	a	q
	2	“	2	B	R	b	r
	3	#	3	C	S	c	s
	4	$	4	D	T	d	t
	5	%	5	E	U	e	u
	6	&	6	F	V	f	v
	7	´	7	G	W	g	w
	8	(	8	H	X	h	x
	9	)	9	I	Y	i	y
	A	*	:	J	Z	j	z
	B	+	;	K	[	k	{
	C	,	<	L	¥	l	|
	D	-	=	M	]	m	}
	E	.	>	N	^	n	→
	F	/	?	O	_	o	←

Kommunikation zwischen Minicomputer und Display
Prinzipiell kann das Display wie ein RAM-Baustein betrachtet werden.

Empfangsbereitschaft
Da das Display jedoch unterschiedliche Befehlsverarbeitungszeiten ist vor jeder Datenübertragung eine Überprüfung der Empfangsbereitschaft erforderlich. Dies geschieht durch die Abfrage des im Datenbit 7 enthaltenen BUSY-Flags.

BUSY-Flag (D7)	Display
0	empfangsbereit
1	nicht empfangsbereit

Die Abfrage ist durch Lesen des Inhalts der Adresse $9000 und anschließender Überprüfung des Zustands von Datenbit 7 möglich.

	Daten-richtung	Adresse
Busy-Flag abfragen	lesen	$9000
Display initialisieren	schreiben	$9000
Cursorposition setzen	schreiben	$9000
Cursorposition ermitteln	lesen	$9000
ASCII-Zeichen an Cursorposition	schreiben	$9001
ASCII-Zeichen von Cursorposition	lesen	$9001

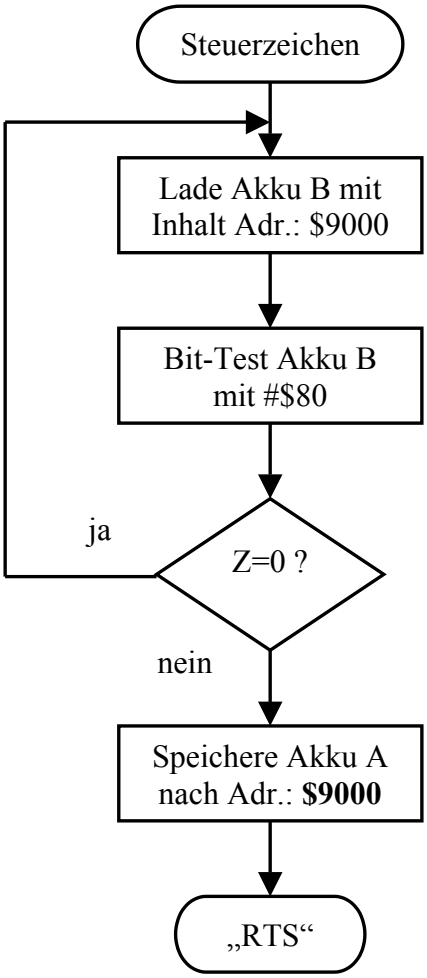
WICHTIG:
1. ASCII-Zeichen können nur zu der Anzeigestelle geschrieben werden auf die der Cursor zeigt.
2. ASCII-Zeichen können nur von der Anzeigestelle gelesen werden auf die der Cursor zeigt.

Unterprogramm: Steuerzeichen

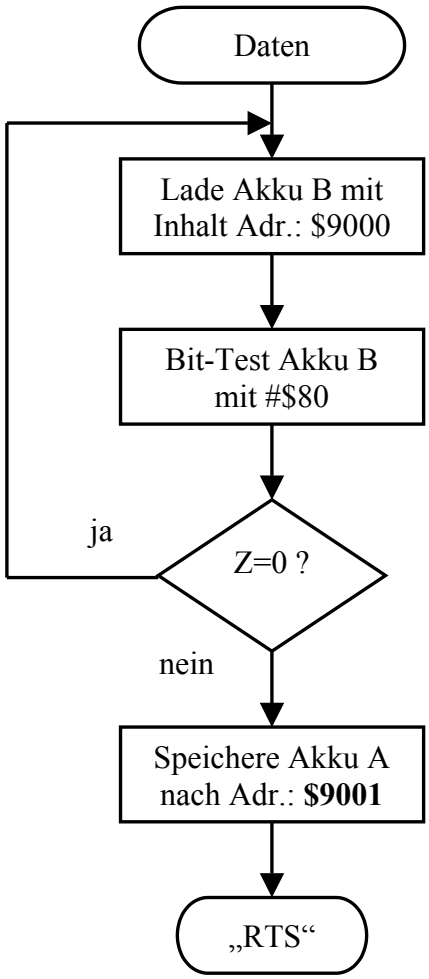
Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	

Benutzen Sie getrennte Unterprogramme für die Kommunikation mit dem Display.

Initialisierung und Cursorposition.
Die Übergabe der Werte erfolgt mit Akku A.



ASCII-Zeichen
Die Übergabe der Werte erfolgt mit Akku A.



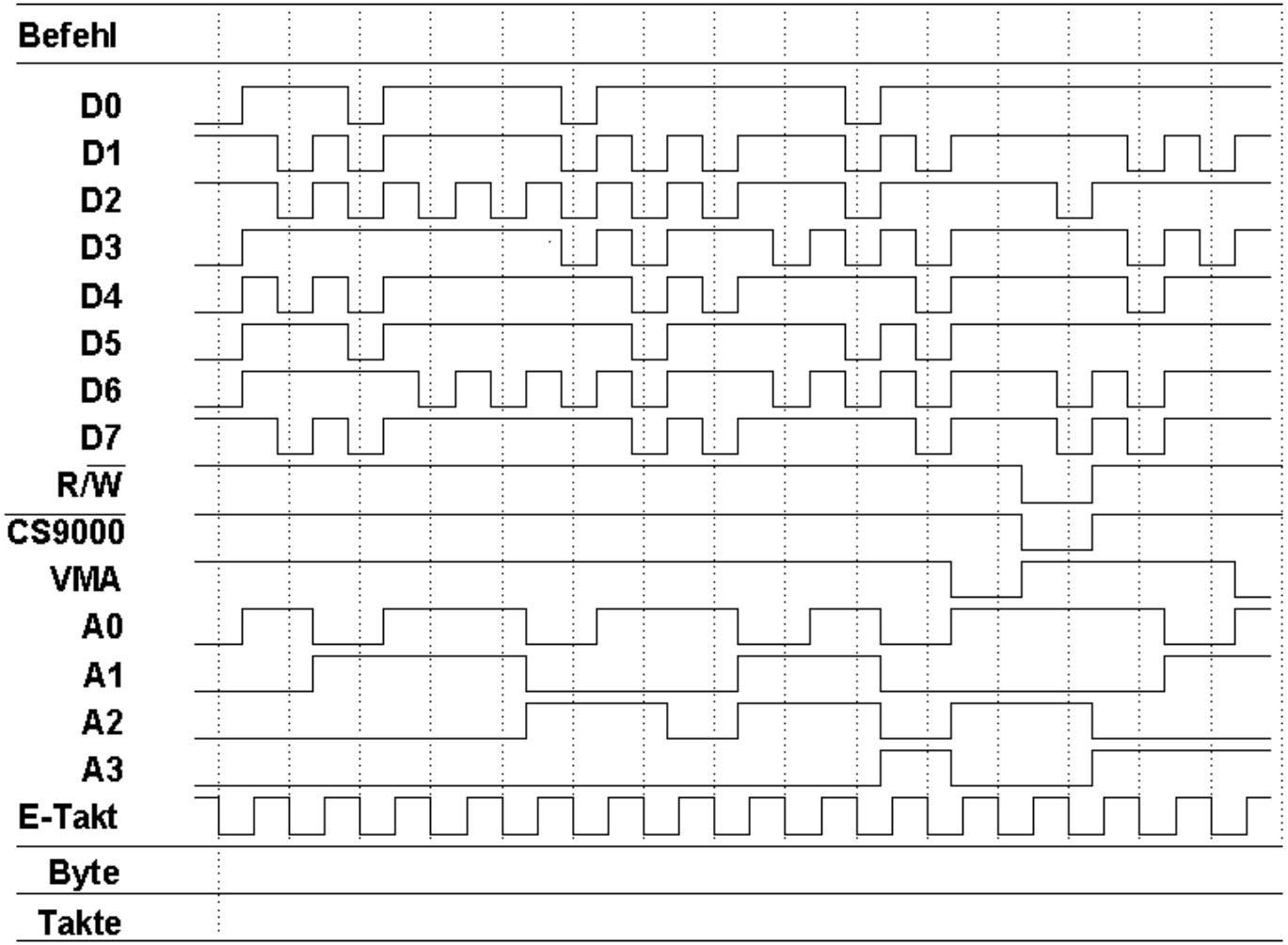
Unterprogramm: Daten

Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	

Tabelle für Programm 7

Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	

Programmaufnahme mit dem Logikanalysator



7. Entschlüsselung der Logikanalysatoraufnahme

a) Schreiben Sie das Programmlisting in ASSEMBLER-Schreibweise
ORG = $ B000

Label	MNEMONIC	Adresse	Befehl			Bemerkung
			Op-C.	1.Op.	2.Op.	
		B000				

b) Wie wird der Inhalt von Akku A durch das Programm verändert ?
Bitte diese Daten des Akkus binär in die Spalte **Bemerkung** eintragen.

c) Bitte verfollständigen Sie das letzte Bit aller Zeilen der Logikaufnahme.

c) Wie geht der Programmablauf weiter ?