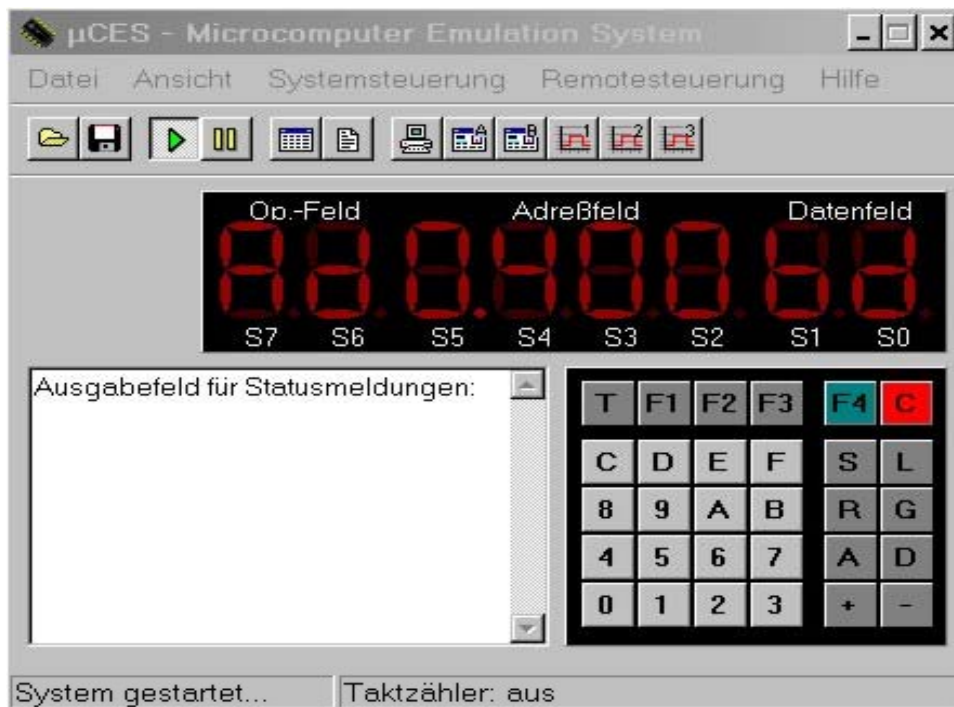


Praktikum zur Mikrorechner-Technik

Referenz-Handbuch



Autor:

Dr. Helmut Bähring

Inhaltsübersicht

0.	Systemarchitektur	1
-----------	--------------------------	----------

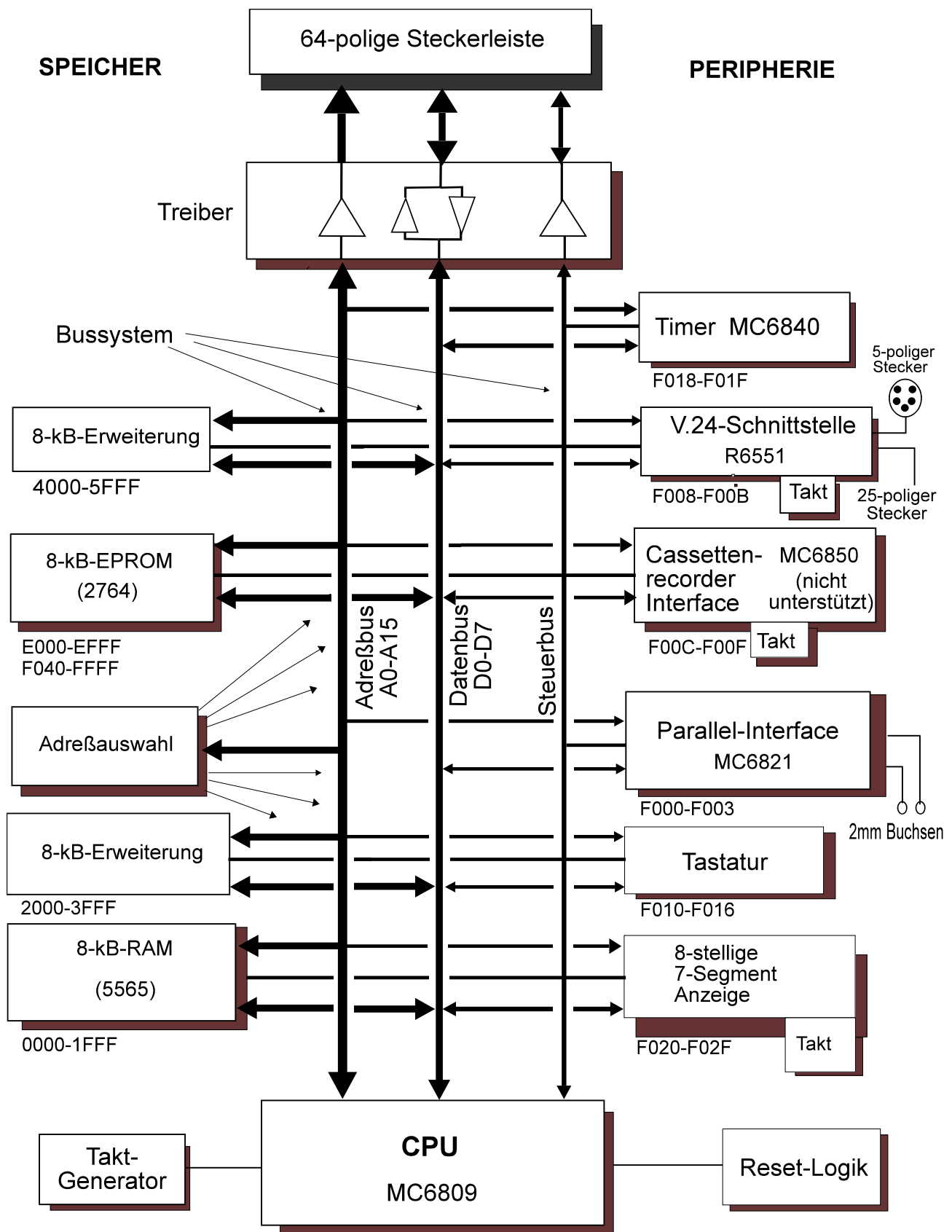
Teil I:

1.	Die CPU	2
1.1	Der MC6809	2
1.2	Aufbau eines Maschinenbefehls	5
1.3	Adressierungsarten	6
1.4	Befehlssatz	12
1.5	Übersicht der implementierten Monitorroutinen	20

Teil II:

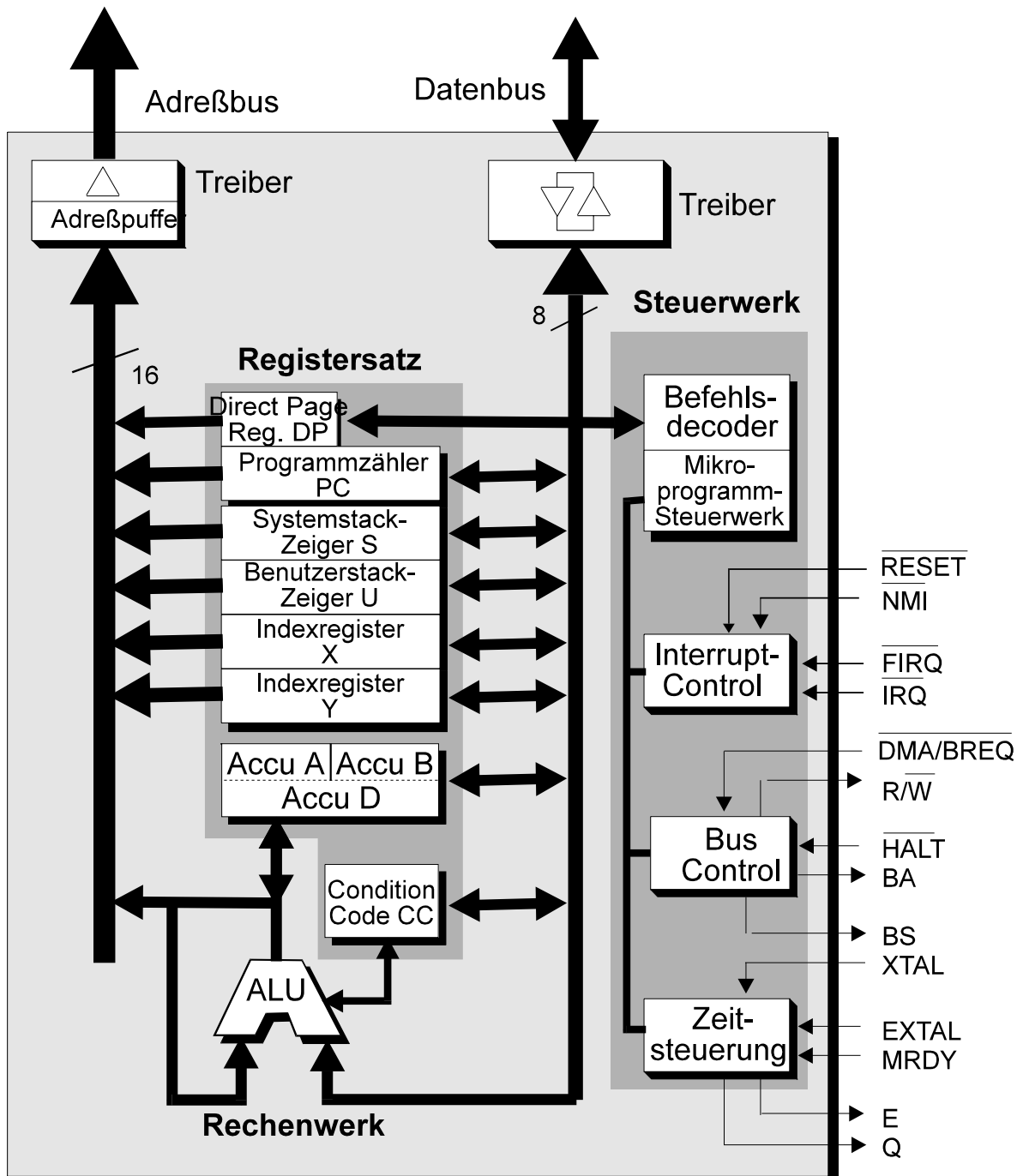
2.	Die Peripheriekomponenten	26
2.1	Interruptvektor-Tabelle	26
2.2	Der Parallelport-Baustein MC6821	27
2.3	Der Timer-Baustein MC6840	33
2.4	Der ACIA-Baustein R6551	39

0. Systemarchitektur

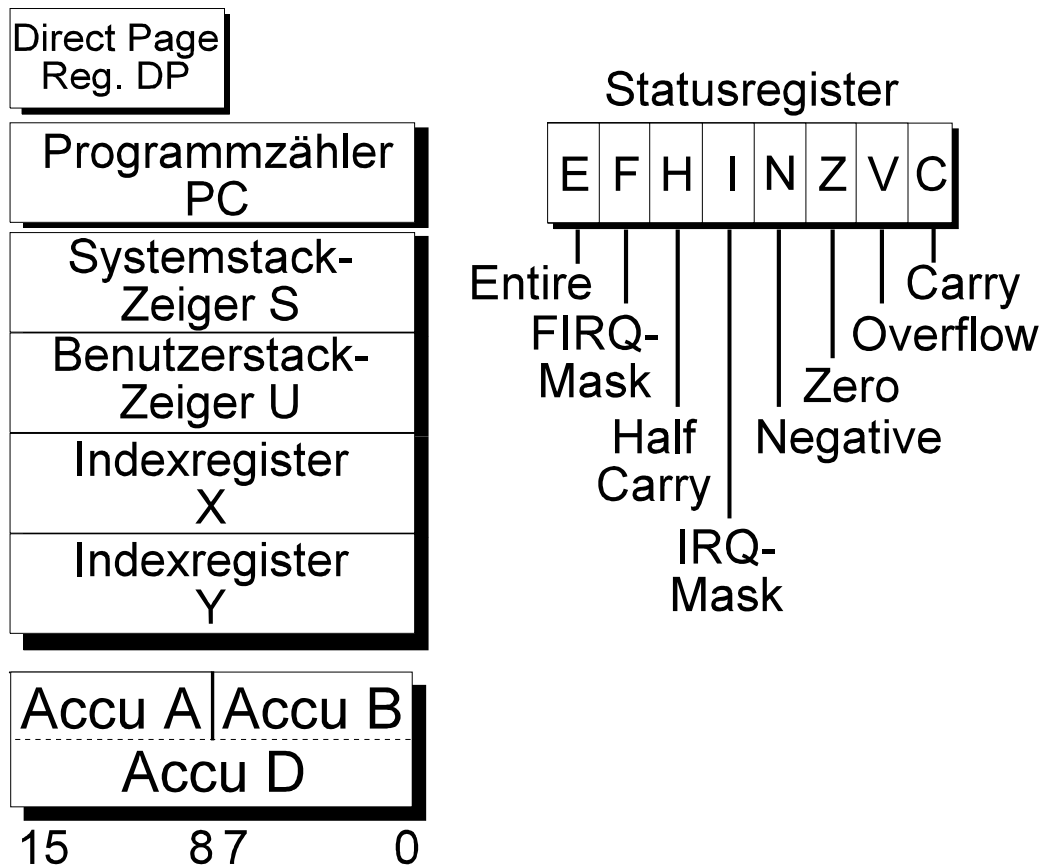


1.1 Der MC6809

- Architektur



- Registersatz des MC6809



- Aufbau des Statusregisters CC

Bit	7	6	5	4	3	2	1	0
Bez.	E	F	H	I	N	Z	V	C
Flag	Entire	FIRQ-Mask	Half-Carry	IRQ-Mask	Negative	Zero	Overflow	Carry

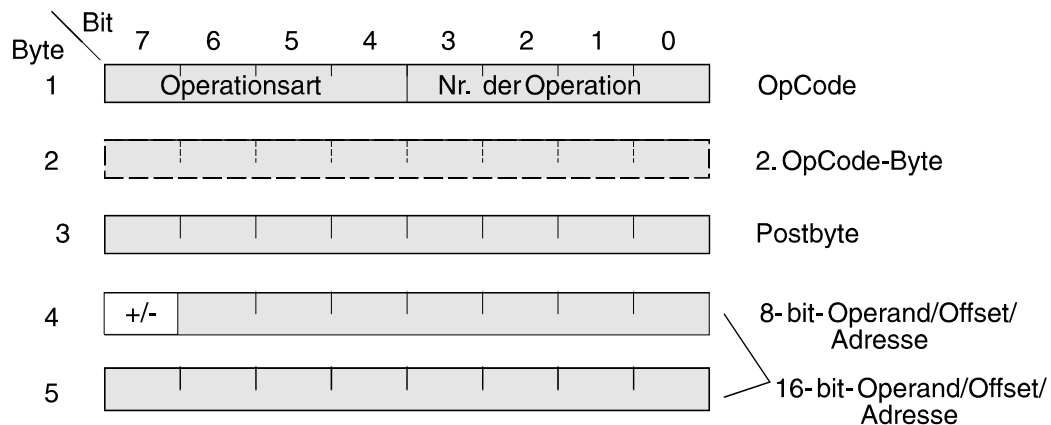
■ Interruptvektoren beim MC6809

Adresse H-Byte L-Byte	Interrupt- Vektor	Ausnahme- situation	Bemerkung
FFFE FFFF		RESET	Rücksetzen
FFFC FFFD		NMI	nicht maskierbarer Interrupt
FFF6 FFF7		FIRQ	schneller maskierbarer Interrupt
FFF8 FFF9		IRQ	maskierbarer Interrupt
FFFA FFFB		SWI1	Software-Interrupt 1
FFF4 FFF5		SWI2	Software-Interrupt 2
FFF2 FFF3		SWI3	Software-Interrupt 3+
FFF0 FFF1			(reserviert)

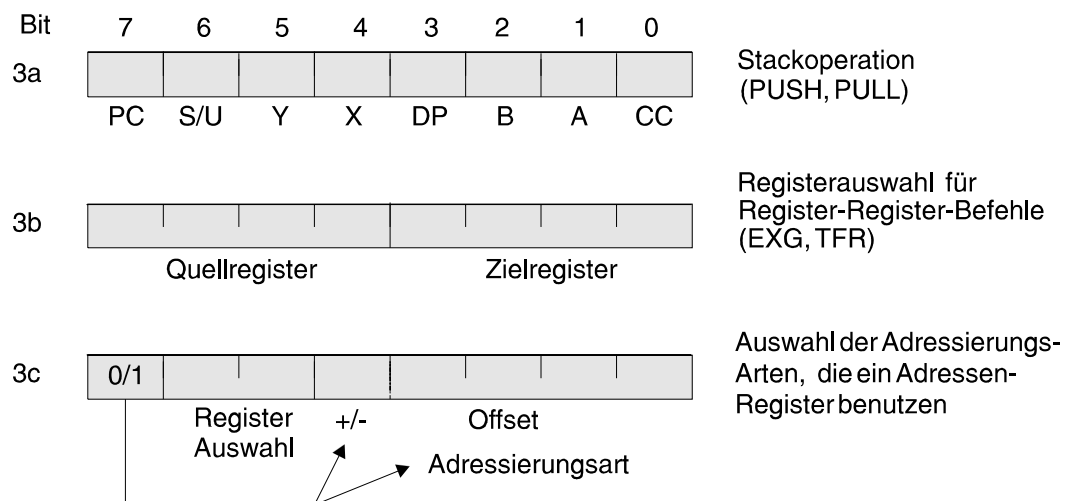
■ Prioritäten der Interrupts beim MC6809

Priorität	Interrupt	gesetztes Flag im CC-Register	gesperrte Interrupts
1	NMI	E,F,I	IRQ , FIRQ
2	FIRQ	-,F,I	IRQ , FIRQ
3	IRQ	E,-,I	IRQ
4	SWI1	E,F,I	IRQ , FIRQ
5	SWI2	E,-,-	----
6	SWI3	E,-,-	----

1.2 Aufbau eines Maschinenbefehls



Aufbau des Postbytes

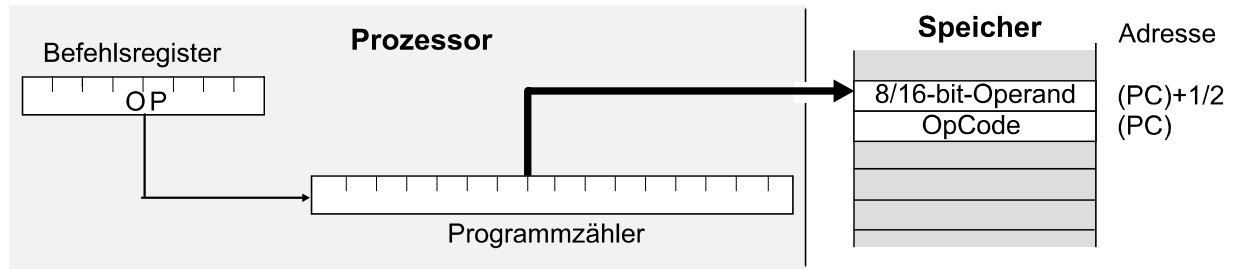


Codierung der Register für die Befehle TFR und EXG

Register	Binär-code	Hex-Code
D	0000	0
X	0001	1
Y	0010	2
U	0011	3
S	0100	4
PC	0101	5
A	1000	8
B	1001	9
CC	1010	A
DP	1011	B

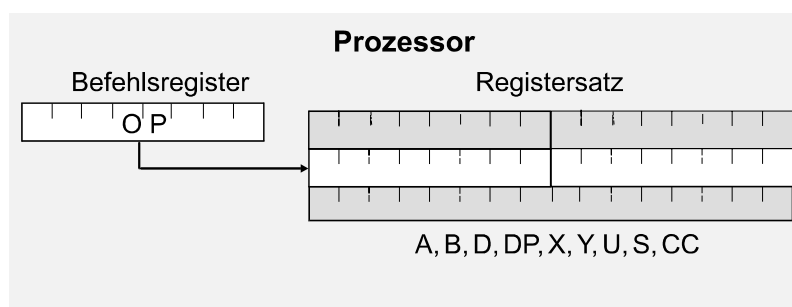
1.3 Adressierungsarten

- unmittelbare Adressierung (*immediate*)



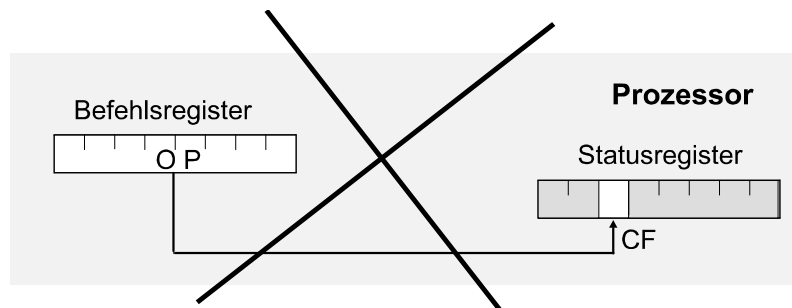
Beispiel: LDA #\$7F (86 7F)

- implizite Adressierung (*implied, inherent*)



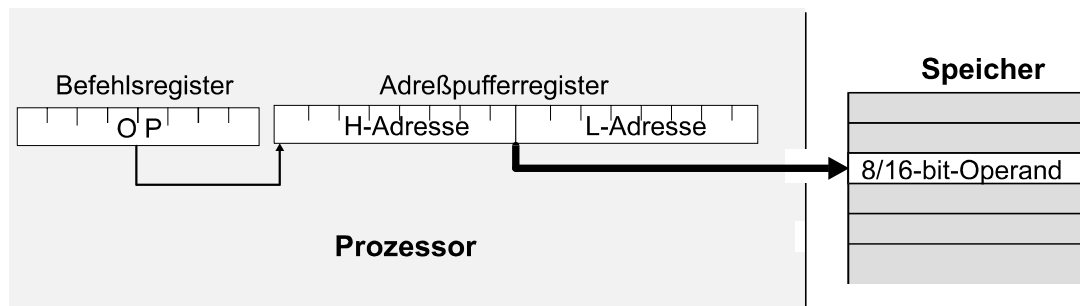
Beispiele: INCA, CMPX, LDX

- Achtung: keine Flag-Adressierung



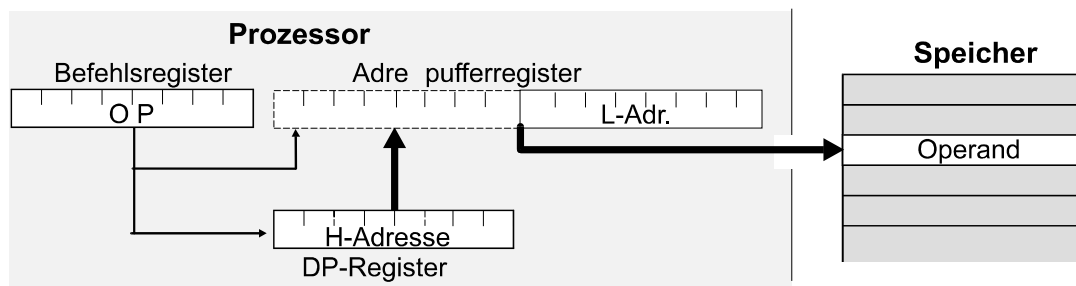
Ersatz: ANDCC, ORCC

- absolute (erweiterte) Adressierung (*extended*)



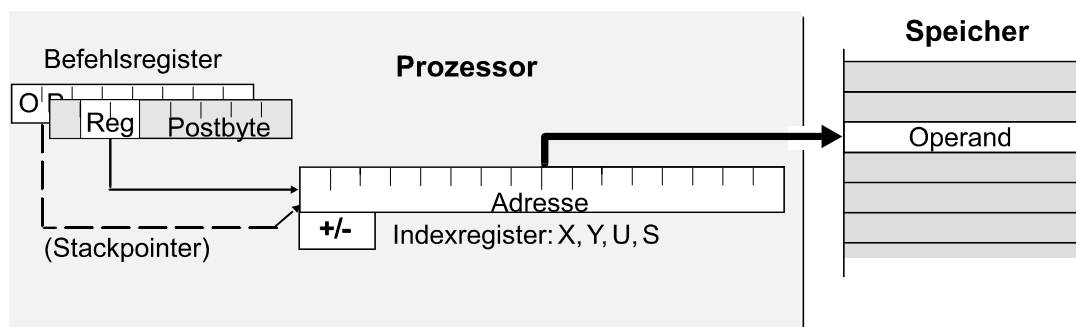
Beispiel: LDA \$A730

- Seiten-Adressierung (*direct*)



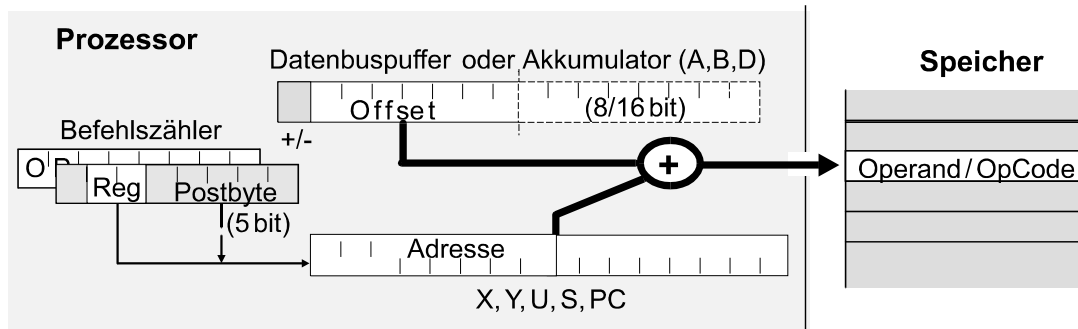
Beispiele: LDB <\$A1, LDB \$A1: D6 A1, für (DP)=\$XY: \$XYA1
LDB >\$A1: F6 00 A1

- indizierte Adressierung ohne Offset (*indexed*)



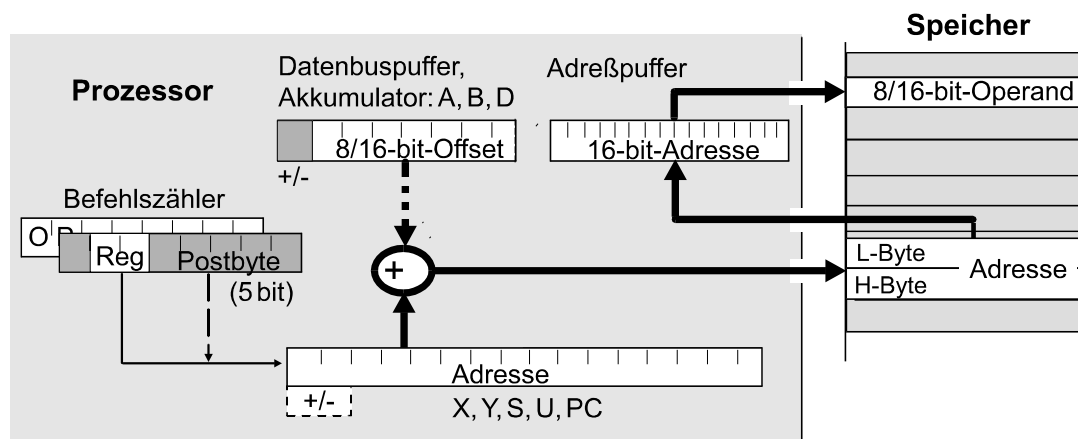
Beispiele: LDA ,X LDB ,Y++

- indizierte Adressierung mit Offset



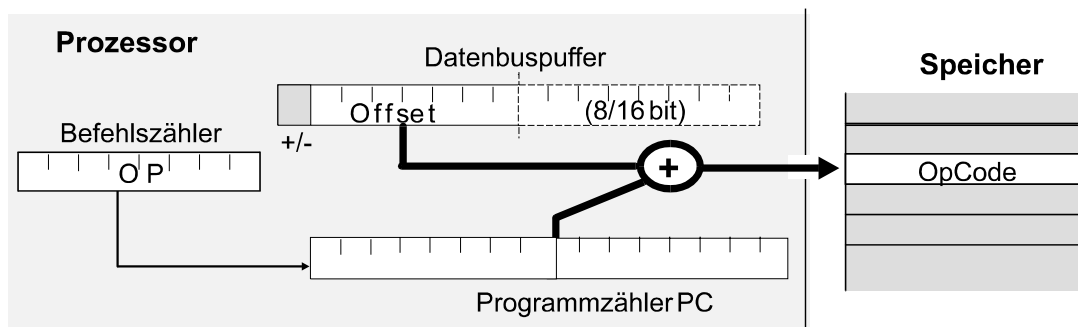
Beispiele: LDA -2,X LDB A,U LDD -50,PCR

- indirekte Adressierung (*indirect*)



Beispiele: LDB [,U] LDD [A,X] LDA [-50,PCR]

- Programmzähler-relative Adressierung (*relative*)



Beispiele: BRA -100 BEQ \$79 LBPL \$FFFC

- Indizierte Adressierung

Lfd. Nr.	Offset	Assembler	Postbyte (Hex)	~	#	Assembler	Postbyte (Hex)	~	#
1	kein Offset	,R	1rr00100	0	0	[,R]	1rr10100	3	0
2	5-Bit*	n,R	0rrnnnnn	1	0	-	8-Bit Offset benutzen	/	/
3	8-Bit	n,R	1rr01000	1	1	[n,R]	1rr11000	4	1
4	16-Bit	n,R	1rr01001	4	2	[n,R]	1rr11001	7	2
5	A-Register	A,R	1rr00110	1	0	[A,R]	1rr10110	4	0
6	B-Register	B,R	1rr00101	1	0	[B,R]	1rr10101	4	0
7	D-Register	D,R	1rr01011	4	0	[D,R]	1rr11011	7	0
8	Inc. um1	,R+	1rr00000	2	0	nicht	erlaubt	/	/
9	Inc. um 2	,R++	1rr00001	3	0	[,R++]	1rr10001	6	0
10	Dec. um 1	,-R	1rr00010	2	0	nicht	erlaubt	/	/
11	Dec. um 2	,--R	1rr00011	3	0	[,--R]	1rr10011	6	0
12	PCR 8-Bit	n,PCR	1rr01100 ₁₎	1	1	[n,PCR]	1rr11100 ₂₎	4	1
13	PCR 16-Bit	n,PCR	1rr01101 ₁₎	5	2	[n,PCR]	1rr11101 ₂₎	8	2
14	Extended Indirekt	-	-	-	-	[n]	1rr11111 ₂₎	5	2

* Der Offset ist im Postbyte enthalten, deshalb ist das Postbyte binär dargestellt. An der Stelle nnnn wird der Offset im Binär-code gesetzt und dann in den Hex-Code umgeformt.

1) 1. Tetrade auch: 8, A, C, E

2) 1. Tetrade auch: 9, B, D, F

Register	rr-Code
X	00
Y	01
U	10
S	11

Code für die indizierte Adressierung mit dem X-Register

Lfd. Nr.	Offset	Direkt		X ~	+ #	Indirekt		X ~	+ #
		Assembler	Postbyte (Hex)			Assembler	Postbyte (Hex)		
1	kein Offset	,X	84	0	0	[,X]	94	3	0
2	5-Bit*	n,X	000nnnnn	1	0	-	8-Bit Offset benutzen	-	-
3	8-Bit	n,X	88	1	1	[n,X]	98	4	1
4	16-Bit	n,X	89	4	2	[n,X]	99	7	2
5	A-Register	A,X	86	1	0	[A,X]	96	4	0
6	B-Register	B,X	85	1	0	[B,X]	95	4	0
7	D-Register	D,X	8B	4	0	[D,X]	9B	7	0
8	Inc. um1	,X+	80	2	0	nicht erlaubt		-	-
9	Inc. um 2	,X++	81	3	0	[,X++]	91	6	0
10	Dec. um 1	,-X	82	2	0	nicht erlaubt		-	-
11	Dec. um 2	,--X	83	3	0	[,--X]	93	6	0
12	PCR 8-Bit	n,PCR	8C ¹⁾	1	1	[n,PCR]	9C ²⁾	4	1
13	PCR 16-Bit	n,PCR	8D ¹⁾	5	2	[n,PCR]	9D ²⁾	8	2
14	Extended Indirekt	-	-	-	-	[n]	9F ²⁾	5	2

Code für die indizierte Adressierung mit dem Y-Register

Lfd. Nr.	Offset	Direkt		X ~	+ #	Indirekt		X ~	+ #
		Assembler	Postbyte (Hex)			Assembler	Postbyte (Hex)		
1	kein Offset	,Y	A4	0	0	[,Y]	B4	3	0
2	5-Bit*	n,Y	001nnnnn	1	0	-	8-Bit Offset benutzen	-	-
3	8-Bit	n,Y	A8	1	1	[n,Y]	B8	4	1
4	16-Bit	n,Y	A9	4	2	[n,Y]	B9	7	2
5	A-Register	A,Y	A6	1	0	[A,Y]	B6	4	0
6	B-Register	B,Y	A5	1	0	[B,Y]	B5	4	0
7	D-Register	D,Y	AB	4	0	[D,Y]	BB	7	0
8	Inc. um1	,Y+	A0	2	0	nicht erlaubt		-	-
9	Inc. um 2	,Y++	A1	3	0	[,Y++]	B1	6	0
10	Dec. um 1	,-Y	A2	2	0	nicht erlaubt		-	-
11	Dec. um 2	,--Y	A3	3	0	[,--Y]	B3	6	0
12	PCR 8-Bit	n,PCR	AC ¹⁾	1	1	[n,PCR]	BC ²⁾	4	1
13	PCR 16-Bit	n,PCR	AD ¹⁾	5	2	[n,PCR]	BD ²⁾	8	2
14	Extended Indirekt	-	-	-	-	[n]	BF ²⁾	5	2

* Der Offset ist im Postbyte enthalten, deshalb ist das Postbyte binär dargestellt. An die Stelle nnnnn wird der Offset im Binär-Code gesetzt und dann in den Hex-Code umgeformt.

1) 1. Tetrade auch: 8, A, C, E

2) 1. Tetrade auch: 9, B, D, F

Code für die indizierte Adressierung mit dem S-Register

Lfd. Nr.	Offset	Direkt		X ~	+ #	Indirekt		X ~	+ #
		Assembler	Postbyte (Hex)			Assembler	Postbyte (Hex)		
1	kein Offset	,S	E4	0	0	[,S]	F4	3	0
2	5-Bit*	n,S	011nnnnn	1	0	-	8-Bit Offset benutzen	-	-
3	8-Bit	n,S	E8	1	1	[n,S]	F8	4	1
4	16-Bit	n,S	E9	4	2	[n,S]	F9	7	2
5	A-Register	A,S	E6	1	0	[A,S]	F6	4	0
6	B-Register	B,S	E5	1	0	[B,S]	F5	4	0
7	D-Register	D,S	EB	4	0	[D,S]	FB	7	0
8	Inc. um1	,S+	E0	2	0	nicht erlaubt		-	-
9	Inc. um 2	,S++	E1	3	0	[,S++]	F1	6	0
10	Dec. um 1	,-S	E2	2	0	nicht erlaubt		-	-
11	Dec. um 2	,--S	E3	3	0	[,--S]	F3	6	0
12	PCR 8-Bit	n,PCR	EC ¹⁾	1	1	[n,PCR]	FC ²⁾	4	1
13	PCR 16-Bit	n,PCR	ED ¹⁾	5	2	[n,PCR]	FD ²⁾	8	2
14	Extended Indirekt	-	-	-	-	[n]	FF ²⁾	5	2

Code für die indizierte Adressierung mit dem U-Register

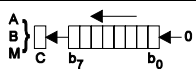
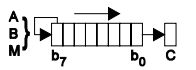
Lfd. Nr.	Offset	Direkt		X ~	+ #	Indirekt		X ~	+ #
		Assembler	Postbyte (Hex)			Assembler	Postbyte (Hex)		
1	kein Offset	,U	C4	0	0	[,U]	D4	3	0
2	5-Bit*	n,U	010nnnnn	1	0	-	8-Bit Offset benutzen	-	-
3	8-Bit	n,U	C8	1	1	[n,U]	D8	4	1
4	16-Bit	n,U	C9	4	2	[n,U]	D9	7	2
5	A-Register	A,U	C6	1	0	[A,U]	D6	4	0
6	B-Register	B,U	C5	1	0	[B,U]	D5	4	0
7	D-Register	D,U	CB	4	0	[D,U]	DB	7	0
8	Inc. um1	,U+	C0	2	0	nicht erlaubt		-	-
9	Inc. um 2	,U++	C1	3	0	[,U++]	D1	6	0
10	Dec. um 1	,-U	C2	2	0	nicht erlaubt		-	-
11	Dec. um 2	,--U	C3	3	0	[,--U]	D3	6	0
12	PCR 8-Bit	n,PCR	CC ¹⁾	1	1	[n,PCR]	DC ²⁾	4	1
13	PCR 16-Bit	n,PCR	CD ¹⁾	5	2	[n,PCR]	DD ²⁾	8	2
14	Extended Indirekt	-	-	-	-	[n]	DF ²⁾	5	2

* Der Offset ist im Postbyte enthalten, deshalb ist das Postbyte binär dargestellt. An die Stelle nnnnn wird der Offset im Binärcode gesetzt und dann in den Hex-Code umgeformt.

1) 1. Tetrade auch: 8, A, C, E

2) 1. Tetrade auch: 9, B, D, F

1.4 Befehlssatz

Instruction	Forms	Adressing Modes												Description	5	3	2	1	0				
		Immediate			Direct			Indexed ¹			Extended				Inherent			H	N	Z	V	C	
		Op	~	#	Op	~	#	Op	~	#	Op	~	#		Op	~	#						
ABX														3A	3	1	B + X → X (Unsigned)	•	•	•	•	•	
ADC	ADCA	89	2	2	99	4	2	A9	4+	2+	B9	5	3				A + M + C → A						
	ADCB	C9	2	2	D9	4	2	E9	4+	2+	F9	5	3				B + M + C → B						
ADD	ADDA	8B	2	2	9B	4	2	AB	4+	2+	BB	5	3				A + M → A						
	ADDB	CB	2	2	DB	4	2	EB	4+	2+	FB	5	3				B + M → B						
	ADDD	C3	4	3	D3	6	2	E3	6+	2+	F3	7	3				D + M : M + 1 → D	•					
AND	ANDA	84	2	2	94	4	2	A4	4+	2+	B4	5	3				A ∧ M → A	•			0	•	
	ANDB	C4	2	2	D4	4	2	E4	4+	2+	F4	5	3				B ∧ M → B	•			0		
	ANDCC	1C	3	2													CC ∧ IMM → CC					7	
ASL	ASLA													48	2	1		8					
	ASLB													58	2	1		8					
	ASL				08	6	2	68	6+	2+	78	7	3					8					
ASR	ASRA													47	2	1		8			•		
	ASRB													57	2	1		8			•		
	ASR				07	6	2	67	6+	2+	77	7	3					8			•		
BIT	BITA	85	2	2	95	4	2	A5	4+	2+	B5	5	3				Bit Test A (M ∧ A)	•			0	•	
	BITB	C5	2	2	D5	4	2	E5	4+	2+	F5	5	3				Bit Test B (M ∧ B)	•			0	•	
CLR	CLRA													4F	2	1	0 → A	•	0	1	0	0	
	CLRB													5F	2	1	0 → B	•	0	1	0	0	
	CLR				0F	6	2	6F	6+	2+	7F	7	3				0 → M	•	0	1	0	0	
CMP	CMPA	81	2	2	91	4	2	A1	4+	2+	B1	5	3				Compare M from A	8					
	CMPB	C1	2	2	D1	4	2	E1	4+	2+	F1	5	3				Compare M from B	8					
	CMPD	10	5	4	10	7	3	10	7+	3+	10	8	4				Compare M : M + 1 from D	•					
		83			93			A3			B3												
	CMPS	11	5	4	11	7	3	11	7+	3+	11	8	4				Compare M : M + 1 from S	•					
		8C			9C			AC			BC												
	CMPU	11	5	4	11	7	3	11	7+	3+	11	8	4				Compare M : M + 1 from U	•					
		83			93			A3			B3												
	CMPX	8C	4	3	9C	6	2	AC	6+	2+	BC	7	3				Compare M : M + 1 from X	•					
		10	5	4	10	7	3	10	7+	3+	10	8	4				Compare M : M + 1 from Y	•					
	8C			9C			AC			BC													
COM	COMA													43	2	1	¬ A → A	•			0		
	COMB													53	2	1	¬ B → B	•			0		
	COM				03	6	2	63	6+	2+	73	7	3				¬ M → M	•			0		
CWAI		3C	≥20	2													CC ∧ IMM → CC					7	
																	Wait for Interrupt						
DAA														19	2	1	Decimal Adjust A	•			0		
DEC	DECA													4A	2	1	A - 1 → A	•				•	
	DECB													5A	2	1	B - 1 → B	•				•	
	DEC				0A	6	2	6A	6+	2+	7A	7	3				M - 1 → M	•				•	
EOR	EORA	88	2	2	98	4	2	A8	4+	2+	B8	5	3				A ∨ M → A	•			0	•	
	EORB	C8	2	2	D8	4	2	E8	4+	2+	F8	5	3				B ∨ M → B	•			0	•	
EXG	R1, R2													1E	8	2	R1 ↔ R2 ²	•	•	•	•	•	
INC	INCA													4C	2	1	A + 1 → A	•				•	
	INCB													5C	2	1	B + 1 → B	•				•	
	INC				0C	6	2	6C	6+	2+	7C	7	3				M + 1 → M	•				•	
JMP					0E	3	2	6E	3+	2+	7E	4	3				EA ³ → PC	•	•	•	•	•	
JSR					9D	7	2	AD	7+	2+	BD	8	3				Jump to Subroutine	•	•	•	•	•	
LD	LDA	86	2	2	96	4	2	A6	4+	2+	B6	5	3				M → A	•			0	•	
	LDB	C6	2	2	D6	4	2	E6	4+	2+	F6	5	3				M → B	•			0	•	
	LDD	CC	3	3	DC	5	2	EC	5+	2+	FC	6	3				M : M + 1 → D	•			0	•	
	LDS	10	4	4	10	6	3	10	6+	3+	10	7	4				M : M + 1 → S	•			0	•	
		CE			DE			EE			FE												
	LDU	CE	3	3	DE	5	2	EE	5+	2+	FE	6	3				M : M + 1 → U	•			0	•	
	LDX	8E	3	3	9E	5	2	AE	5+	2+	BE	6	3				M : M + 1 → X	•			0	•	
	LDY	10	4	4	10	6	3	10	6+	3+	10	7	4				M : M + 1 → Y	•			0	•	
		8E			9E			AE			BE												

Instruction	Forms	Adressing Modes												Description	5	3	2	1	0				
		Immediate			Direct			Indexed ¹			Extended				Inherent			H	N	Z	V	C	
		Op	~	#	Op	~	#	Op	~	#	Op	~	#		Op	~	#						
LEA	LEAS LEAU LEAX LEAY							32 33 30 31	4+ 4+ 4+ 4+	2+ 2+ 2+ 2+							EA ³ → S EA ³ → U EA ³ → X EA ³ → Y	•	•	•	•	•	
LSL	LSLA LSLB LSL					08	6	2	68	6+	2+	78	7	3	48 58	2 2	1 1		•	•	•	•	•
LSR	LSRA LSRB LSR					04	6	2	64	6+	2+	74	7	3	44 54	2 2	1 1		•	0	•	•	•
MUL															3D	11	1	A × B → D (Unsigned)	•	•	•	•	9
NEG	NEGA NEGB NEG					00	6	2	60	6+	2+	70	7	3	40 50	2 2	1 1	¬ A + 1 → A ¬ B + 1 → B ¬ M + 1 → M	8	•	•	•	•
NOP															12	2	1	No Operation	•	•	•	•	•
OR	ORA ORB ORCC	8A CA 1A	2 2 3	2 2 2	9A DA	4 4	2 2	AA EA	4+ 4+	2+ 2+	BA FA	5 5	3 3				A ∨ M → A B ∨ M → B CC ∨ IMM → CC	•	•	•	0	•	
PSH	PSHS ¹⁰ PSHU ¹⁰	34 36	5+ ⁴ 5+ ⁴	2 2														Push Registers on S Stack Push Registers on U Stack	•	•	•	•	•
PUL	PULS ¹⁰ PULU ¹⁰	35 37	5+ ⁴ 5+ ⁴	2 2														Pull Registers from S Stack Pull Registers from U Stack	•	•	•	•	•
ROL	ROLA ROLB ROL					09	6	2	69	6+	2+	79	7	3	49 59	2 2	1 1		•	•	•	•	•
ROR	RORA RORB ROR					06	6	2	66	6+	2+	76	7	3	46 56	2 2	1 1		•	•	•	•	•
RTI															3B	6 - 15	1	Return from Interrupt					7
RTS															39	5	1	Return from Subroutine	•	•	•	•	•
SBC	SBCA SBCB	82 C2	2 2	2 2	92 D2	4 4	2 2	A2 E2	4+ 4+	2+ 2+	B2 F2	5 5	3 3				A - M - C → A B - M - C → B	8	•	•	•	•	
SEX															1D	2	1	Sign Extend B into A	•	•	•	0	•
ST	STA STB STD STS					97 D7 DD 10	4 4 5 6	2 2 2 3	A7 E7 ED 10	4+ 4+ 5+ 6+	2+ 2+ 2+ 3+	B7 F7 FD 10	5 5 6 7	3 3 3 4				A → M B → M D → M : M + 1 S → M : M + 1	•	•	•	0	•
	STU STX STY					DF 9F 10 9F	5 5 6	2 2 3	EF AF 10 AF	5+ 5+ 6+	2+ 2+ 3+	FF BF 10 BF	6 6 7	3 3 4				U → M : M + 1 X → M : M + 1 Y → M : M + 1	•	•	•	0	•
SUB	SUBA SUBB SUBD	80 C0 83	2 2 4	2 2 3	90 D0 93	4 4 6	2 2 2	A0 E0 A3	4+ 4+ 6+	2+ 2+ 2+	B0 F0 B3	5 5 7	3 3 3				A - M → A B - M → B D - M : M + 1 → D	8	•	•	•	•	
SWI	SWI ⁶ SWI2 ⁶ SWI3 ⁶														3F 10 3F 11 3F	19 20 20 20 20	1 2 1 1 1	Software Interrupt 1 Software Interrupt 2 Software Interrupt 3	•	•	•	•	•
SYNC															13	≥4	1	Synchronize to Interrupt	•	•	•	•	•
TFR	R1, R2														1F	6	2	R1 → R2 ²	•	•	•	•	•
TST	TSTA TSTB TST					0D	6	2	6D	6+	2+	7D	7	3	4D 5D	2 2	1 1	Test A Test B Test M	•	•	•	0	•

LEGEND:

LEGEND:		¬	Complement of M		Test and set if true, cleared otherwise
		M			
O	Operation Code	→	Transfer Into	•	Not Affected
P (Hexadecimal)					
~	Number of CPU Cycles	H	Half-carry (from bit 3)	CC	Condition Code Register
#	Number of Program Bytes	N	Negative (sign bit)	:	Concatenation
+	Arithmetic Plus	Z	Zero result	∨	Logical or
-	Arithmetic Minus	V	Overflow, 2's complement	^	Logical and
×	Multiply	C	Carry from ALU	⊕	Logical Exclusive or

NOTES:

- 1 This column gives a base cycle and byte count. To obtain total count, add the values obtained from the INDEXED ADDRESSING MODE table (s. Tabellen A-1 bis A-4).
- 2 R1 and R2 may be any pair of 8 bit or any pair of 16 bit registers.
The 8 bit registers are A, B, CC, DP.
The 16 bit registers are X, Y, U, S, D, PC.
- 3 EA is the effective address.
- 4 The PSH and PUL instructions require 5 cycles plus 1 cycle for each **byte** pushed or pulled.
- 5 5(6) means 5 cycles if branch not taken, 6 cycles if taken (Branch instructions).
- 6 SWI1 sets I and F bits. SWI2 and SWI3 do not affect I and F.
- 7 Conditions Codes set as a direct result of the instruction.
- 8 Value of half-carry flag is undefined.
- 9 Special case - Carry set if b_7 is SET.
- 10 (Bestimmung des Postbytes nach Bild A-1)

NOTES:

- 1 This column gives a base cycle and byte count. To obtain total count, add the values obtained from the INDEXED ADDRESSING MODE table
- 2 R1 and R2 may be any pair of 8 bit or any pair of 16 bit registers.
The 8 bit registers are A, B, CC, DP.
The 16 bit registers are X, Y, U, S, D, PC.
- 3 EA is the effective address.
- 4 The PSH and PUL instructions require 5 cycles plus 1 cycle for each byte pushed or pulled.
- 5 5(6) means 5 cycles if branch not taken, 6 cycles if taken (Branch instructions).
- 6 SWI1 sets I and F bits. SWI2 and SWI3 do not affect I and F.
- 7 Conditions Codes set as a direct result of the instruction.
- 8 Value of half-carry flag is undefined.
- 9 Special case - Carry set if b_7 is SET.
- 10 (Bestimmung des Postbytes nach Bild A-1)

- Verzweigungsbefehle

Befehl	Form	Addr.-Art: Relative			Beschreibung						CMPR ^{*)}	Vorzeichen
		OP	~	#		H	N	Z	V	C		
BCC	BCC LBCC	24 10 24	3 5(6)	2 4	Branch C = 0 Long Branch C = 0	•	•	•	•	•		
BCS	BCS LBCCS	25 10 25	3 5(6)	2 4	Branch C = 1 Long Branch C = 1	•	•	•	•	•		
BEQ	BEQ LBEQ	27 10 27	3 5(6)	2 4	Branch Z = 1 Long Branch Z = 1	•	•	•	•	•		
BGE	BGE LBGE	2C 10 2C	3 5(6)	2 4	Branch ≥ Zero Long Branch ≥ Zero	•	•	•	•	•	R ≥ M	vorzeichen- behaftet
BGT	BGT LBGT	2E 10 2E	3 5(6)	2 4	Branch > Zero Long Branch > Zero	•	•	•	•	•	R > M	vorzeichen- behaftet
BHI	BHI LBHI	22 10 22	3 5(6)	2 4	Branch Higher Long Branch Higher	•	•	•	•	•	R > M	vorzeichen- los
BHS	BHS LBHS	24 10 24	3 5(6)	2 4	Branch Higher or Same Long Branch Higher or Same	•	•	•	•	•	R ≥ M	vorzeichen- los
BLE	BLE LBLE	2F 10 2F	3 5(6)	2 4	Branch ≤ Zero Long Branch ≤ Zero	•	•	•	•	•	R ≤ M	vorzeichen- behaftet
BLO	BLO LBLO	25 10 25	3 5(6)	2 4	Branch Lower Long Branch Lower	•	•	•	•	•	R < M	vorzeichen- los
BLS	BLS LBLS	23 10 23	3 5(6)	2 4	Branch Lower or Same Long Branch Lower or Same	•	•	•	•	•	R ≤ M	vorzeichen- los
BLT	BLT LBLT	2D 10 2D	3 5(6)	2 4	Branch < Zero Long Branch < Zero	•	•	•	•	•	R < M	Vorzeichen- behaftet
BMI	BMI LBMI	2B 10 2B	3 5(6)	2 4	Branch Minus (N=1) Long Branch Minus	•	•	•	•	•		
BNE	BNE LBNE	26 10 26	3 5(6)	2 4	Branch on Z=0 Long Branch on Z=0	•	•	•	•	•		
BPL	BPL LBPL	2A 10 2A	2 5(6)	2 4	Branch Plus (N=0) Long Branch Plus	•	•	•	•	•		
BRA	BRA LBRA	20 16	3 5	2 3	Branch Always Long Branch Always	•	•	•	•	•		
BRN	BRN LBRN	21 10 21	3 5	2 4	Branch Never Long Branch Never	•	•	•	•	•		
BSR	BSR LBSR	8D 17	7 9	2 3	Branch to Subroutine Long Branch to Subr.	•	•	•	•	•		
BVC	BVC LBVC	28 10 28	3 5(6)	2 4	Branch V = 0 Long Branch V = 0	•	•	•	•	•		
BVS	BVS LBVS	29 10 29	3 5(6)	2 4	Branch V = 1 Long Branch V = 1	•	•	•	•	•		

- **Verzweigungsbefehle**

Test	wahr	OpCode	falsch	OpCode
unbedingte Verzweigungen				
---	BRA / LBRA	20 / 16	---	---
---	BRN / LBRN	21 / 10 21	---	---
---	BSR / LBSR	BD / 17	---	---
einfache bedingte Verzweigung				
N = 1	BMI / LBMI	2B / 10 2B	BPL / LBPL	2A / 10 2A
Z = 1	BEQ / LBEQ	27 / 10 27	BNE / LBNE	26 / 10 26
V = 1	BVS / LBVS	29 / 10 29	BVC / LBVC	28 / 10 28
C = 1	BCS / LBCS	25 / 10 25	BCC / LBCC	24 / 10 24
vorzeichenbehaftete bedingte Verzweigung				
$r > m$	BGT / LBGT	2E / 10 2E	BLE / LBLE	2F / 10 2F
$r \geq m$	BGE / LBGE	2C / 10 2C	BLT / LBLT	2D / 10 2D
$r = m$	BEQ / LBEQ	27 / 10 27	BNE / LBNE	26 / 10 26
$r \leq m$	BLE / LBLE	2F / 10 2F	BGT / LBGT	2E / 10 2E
$r < m$	BLT / LBLT	2D / 10 2D	BGT / LBGT	2C / 10 2C
vorzeichenlose bedingte Verzweigung				
$r > m$	BHI / LBHI	22 / 10 22	BLS	23 / 10 23
$r \geq m$	BHS / LBHS	24 / 10 24	BLO / LBLO	25 / 10 25
$r = m$	BEQ / LBEQ	27 / 10 27	BHE / LBHE	26 / 10 26
$r \leq m$	BLS / LBLS	23 / 10 23	BHI / LBHI	22 / 10 22
$r < m$	BLO / LBLO	25 / 10 25	BHS / LBHS	24 / 10 24

Tabelle: Befehlscode in hexadezimaler Reihenfolge - Teil I

OP	Mnemonic	Mode	~	#	OP	Mnemonic	Mode	~	#	OP	Mnemonic	Mode	~	#
00	NEG	Direct	6	2	30	LEAX	Indexed	4+	2+	60	NEG	Indexed	6+	2+
01	•	-	-	-	31	LEAY	Indexed	4+	2+	61	•	-	-	-
02	•	-	-	-	32	LEAS	Indexed	4+	2+	62	•	-	-	-
03	COM	Direct	6	2	33	LEAU	Indexed	4+	2+	63	COM	Indexed	6+	2+
04	LSR	Direct	6	2	34	PSHS	Immed.	5+	2	64	LSR	Indexed	6+	2+
05	•	-	-	-	35	PULS	Immed.	5+	2	65	•	-	-	-
06	ROR	Direct	6	2	36	PSHU	Immed.	5+	2	66	ROR	Indexed	6+	2+
07	ASR	Direct	6	2	37	PULU	Immed.	5+	2	67	ASR	Indexed	6+	2+
08	ASL, LSL	Direct	6	2	38	•	-	-	-	68	ASL, LSL	Indexed	6+	2+
09	ROL	Direct	6	2	39	RTS	Inherent	5	1	69	ROL	Indexed	6+	2+
0A	DEC	Direct	6	2	3A	ABX	Inherent	3	1	6A	DEC	Indexed	6+	2+
0B	•	-	-	-	3B	RTI	Inherent	6/15	1	6B	•	-	-	-
0C	INC	Direct	6	2	3C	CWAI	Inherent	≥20	2	6C	INC	Indexed	6+	2+
0D	TST	Direct	6	2	3D	MUL	Inherent	11	1	6D	TST	Indexed	6+	2+
0E	JMP	Direct	3	2	3E	•	-	-	-	6E	JMP	Indexed	3+	2+
0F	CLR	Direct	6	2	3F	SWI	Inherent	19	1	6F	CLR	Indexed	6+	2+
10	Page 2	-	-	-	40	NEGA	Inherent	2	1	70	NEG	Extended	7	3
11	Page 3	-	-	-	41	•	-	-	-	71	•	-	-	-
12	NOP	Inherent	2	1	42	•	-	-	-	72	•	-	-	-
13	SYNC	Inherent	≥4	1	43	COMA	Inherent	2	1	73	COM	Extended	7	3
14	•	-	-	-	44	LSRA	Inherent	2	1	74	LSR	Extended	7	3
15	•	-	-	-	45	•	-	-	-	75	•	-	-	-
16	LBRA	Relative	5	3	46	RORA	Inherent	2	1	76	ROR	Extended	7	3
17	LBSR	Relative	9	3	47	ASRA	Inherent	2	1	77	ASR	Extended	7	3
18	•	-	-	-	48	ASLA, LSLA	Inherent	2	1	78	ASL, LSL	Extended	7	3
19	DAA	Inherent	2	1	49	ROLA	Inherent	2	1	79	ROL	Extended	7	3
1A	ORCC	Immed.	3	2	4A	DECA	Inherent	2	1	7A	DEC	Extended	7	3
1B	•	-	-	-	4B	•	-	-	-	7B	•	-	-	-
1C	ANDCC	Immed.	3	2	4C	INCA	Inherent	2	1	7C	INC	Extended	7	3
1D	SEX	Inherent	2	1	4D	TSTA	Inherent	2	1	7D	TST	Extended	7	3
1E	EXG	Immed.	8	2	4E	•	-	-	-	7E	JMP	Extended	4	3
1F	TFR	Immed.	6	2	4F	CLRA	Inherent	2	1	7F	CLR	Extended	7	3
20	BRA	Relative	3	2	50	NEGB	Inherent	2	1	80	SUBA	Immed.	2	2
21	BRN	Relative	3	2	51	•	Inherent	-	-	81	CMPA	Immed.	2	2
22	BHI	Relative	3	2	52	•	Inherent	-	-	82	SBCA	Immed.	2	2
23	BLS	Relative	3	2	53	COMB	Inherent	2	1	83	SUBD	Immed.	4	3
24	BHS, BCC	Relative	3	2	54	LSRB	Inherent	2	1	84	ANDA	Immed.	2	2
25	BLO, BCS	Relative	3	2	55	•	Inherent	-	-	85	BITA	Immed.	2	2
26	BNE	Relative	3	2	56	RORB	Inherent	2	1	86	LDA	Immed.	2	2
27	BEQ	Relative	3	2	57	ASRB	Inherent	2	1	87	•	-	-	-
28	BVC	Relative	3	2	58	ASLB, LSLB	Inherent	2	1	88	EORA	Immed.	2	2
29	BVS	Relative	3	2	59	ROLB	Inherent	2	1	89	ADCA	Immed.	2	2
2A	BPL	Relative	3	2	5A	DECB	Inherent	2	1	8A	ORA	Immed.	2	2
2B	BMI	Relative	3	2	5B	•	Inherent	-	-	8B	ADDA	Immed.	2	2
2C	BGE	Relative	3	2	5C	INCB	Inherent	2	1	8C	CMPX	Immed.	4	3
2D	BLT	Relative	3	2	5D	TSTB	Inherent	2	1	8D	BSR	Relative	7	2
2E	BGT	Relative	3	2	5E	•	Inherent	-	-	8E	LDX	Immed.	3	3
2F	BLE	Relative	3	2	5F	CLRB	Inherent	2	1	8F	•	-	-	-

LEGEND

- ~ Number of CPU cycles (less possible push pull or indexed-mode cycles)
- # Number of program bytes
- Denotes unused opcode

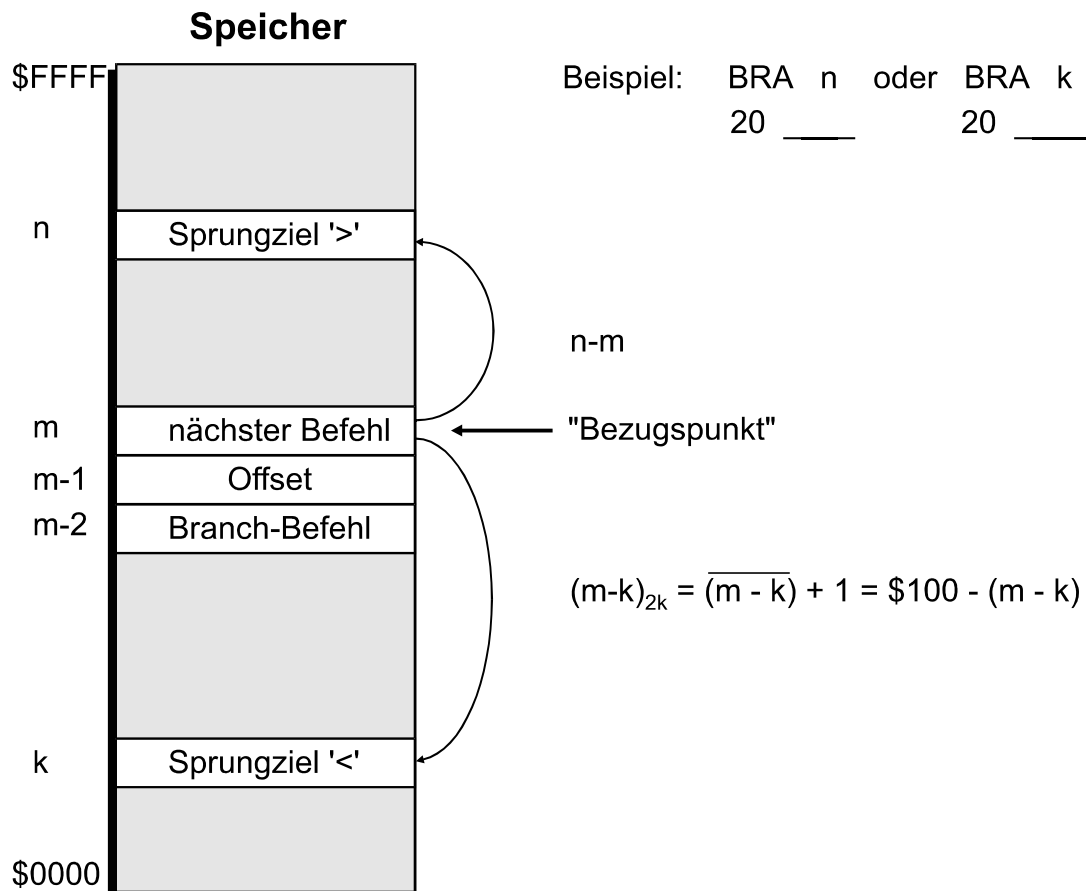
Tabelle: Befehlscode in hexadezimaler Reihenfolge - Teil II

OP	Mnemonic	Mode	~	#	OP	Mnemonic	Mode	~	#	OP	Mnemonic	Mode	~	#
90	SUBA	Direct	4	2	C0	SUBB	Immed	2	2	Page 2 and 3 Machine Codes				
91	CMPA	Direct	4	2	C1	CMPB	Immed	2	2					
92	SBCA	Direct	4	2	C2	SBCB	Immed	2	2					
93	SUBD	Direct	6	2	C3	ADDD	Immed	4	3					
94	ANDA	Direct	4	2	C4	ANDB	Immed	2	2	1021	LBRN	Relative	5	4
95	BITA	Direct	4	2	C5	BITB	Immed	2	2	1022	LBHI	Relative	5(6)	4
96	LDA	Direct	4	2	C6	LDB	Immed	2	2	1023	LBLS	Relative	5(6)	4
97	STA	Direct	4	2	C7	•	-	-	-	1024	LBHS, LBCC	Relative	5(6)	4
98	EORA	Direct	4	2	C8	EORB	Immed	2	2	1025	LBCS, LBLO	Relative	5(6)	4
99	ADCA	Direct	4	2	C9	ADCB	Immed	2	2	1026	LBNE	Relative	5(6)	4
9A	ORA	Direct	4	2	CA	ORB	Immed	2	2	1027	LBEQ	Relative	5(6)	4
9B	ADDA	Direct	4	2	CB	ADDB	Immed	2	2	1028	LBVC	Relative	5(6)	4
9C	CMPX	Direct	6	2	CC	LDD	Immed	3	3	1029	LBVS	Relative	5(6)	4
9D	JSR	Direct	7	2	CD	•	-	-	-	102A	LBPL	Relative	5(6)	4
9E	LDX	Direct	5	2	CE	LDU	Immed	3	3	102B	LBMI	Relative	5(6)	4
9F	STX	Direct	5	2	CF	•	-	-	-	102C	LBGE	Relative	5(6)	4
A0	SUBA	Inexed	4+	2+	D0	SUBB	Direct	4	2	102D	LBLT	Relative	5(6)	4
A1	CMPA	Inexed	4+	2+	D1	CMPB	Direct	4	2	102E	LBGT	Relative	5(6)	4
A2	SBCA	Inexed	4+	2+	D2	SBCB	Direct	4	2	102F	LBLE	Relative	5(6)	4
A3	SUBD	Inexed	6+	2+	D3	ADDD	Direct	6	2	103F	SWI2	Inherent	20	2
A4	ANDA	Inexed	4+	2+	D4	ANDB	Direct	4	2	1083	CMPD	Immed	5	4
A5	BITA	Inexed	4+	2+	D5	BITB	Direct	4	2	108C	CMPY	Immed	5	4
A6	LDA	Inexed	4+	2+	D6	LDB	Direct	4	2	108E	LDY	Immed	4	4
A7	STA	Inexed	4+	2+	D7	STB	Direct	4	2	1093	CMPD	Direct	7	3
A8	EORA	Inexed	4+	2+	D8	EORB	Direct	4	2	109C	CMPY	Direct	7	3
A9	ADCA	Inexed	4+	2+	D9	ADCB	Direct	4	2	109E	LDY	Direct	6	3
AA	ORA	Inexed	4+	2+	DA	ORB	Direct	4	2	109F	STY	Direct	6	3
AB	ADDA	Inexed	4+	2+	DB	ADDB	Direct	4	2	10A3	CMPD	Indexed	7+	3+
AC	CMPX	Inexed	6+	2+	DC	LDD	Direct	5	2	10AC	CMPY	Indexed	7+	3+
AD	JSR	Inexed	7+	2+	DD	STD	Direct	5	2	10AE	LDY	Indexed	6+	3+
AE	LDX	Inexed	5+	2+	DE	LDU	Direct	5	2	10AF	STY	Indexed	6+	3+
AF	STX	Inexed	5+	2+	DF	STU	Direct	5	2	10B3	CMPD	Extended	8	4
B0	SUBA	Extended	5	3	E0	SUBB	Inexed	4+	2+	10BC	CMPY	Extended	8	4
B1	CMPA	Extended	5	3	E1	CMPB	Inexed	4+	2+	10BE	LDY	Extended	7	4
B2	SBCA	Extended	5	3	E2	SBCB	Inexed	4+	2+	10BF	STY	Extended	7	4
B3	SUBD	Extended	7	3	E3	ADDD	Inexed	6+	2+	10CE	LDS	Immed	4	4
B4	ANDA	Extended	5	3	E4	ANDB	Inexed	4+	2+	10DE	LDS	Direct	6	3
B5	BITA	Extended	5	3	E5	BITB	Inexed	4+	2+	10DF	STS	Direct	6	3
B6	LDA	Extended	5	3	E6	LDB	Inexed	4+	2+	10EE	LDS	Indexed	6+	3+
B7	STA	Extended	5	3	E7	STB	Inexed	4+	2+	10EF	STS	Indexed	6+	3+
B8	EORA	Extended	5	3	E8	EORB	Inexed	4+	2+	10FE	LDS	Extended	7	4
B9	ADCA	Extended	5	3	E9	ADCB	Inexed	4+	2+	10FF	STS	Extended	7	4
BA	ORA	Extended	5	3	EA	ORB	Inexed	4+	2+	113F	SWI3	Inherent	20	2
BB	ADDA	Extended	5	3	EB	ADDB	Inexed	4+	2+	1183	CMPU	Immed	5	4
BC	CMPX	Extended	7	3	EC	LDD	Inexed	5+	2+	118C	CMPS	Immed	5	4
BD	JSR	Extended	8	3	ED	STD	Inexed	5+	2+	1193	CMPU	Direct	7	3
BE	LDX	Extended	6	3	EE	LDU	Inexed	5+	2+	119C	CMPS	Direct	7	3
BF	STX	Extended	6	3	EF	STU	Inexed	5+	2+	11A3	CMPU	Indexed	7+	3+
NOTE: All unused opcodes are both undefined and illegal					F0	SUBB	Extended	5	3	11AC	CMPS	Indexed	7+	3+
					F1	CMPB	Extended	5	3	11B3	CMPU	Extended	8	4
					F2	SBCB	Extended	5	3	11BC	CMPU	Extended	8	4
					F3	ADDD	Extended	7	3					
					F4	ANDB	Extended	5	3					
					F5	BITB	Extended	5	3					
					F6	LDB	Extended	5	3					
					F7	STB	Extended	5	3					
					F8	EORB	Extended	5	3					
					F9	ADCB	Extended	5	3					
					FA	ORB	Extended	5	3					
					FB	ADDB	Extended	5	3					
					FC	LDD	Extended	6	3					
					FD	STD	Extended	6	3					
					FE	LDU	Extended	6	3					
					FF	STU	Extended	6	3					

LEGEND

- ~ Number of CPU cycles (less possible push pull or indexed-mode cycles)
- # Number of program bytes
- Denotes unused opcode

- Berechnung von relativen Sprungadressen



Beispiele:

>: n = \$7A3F
 m = \$79CD
 n-m = \$72

<: m = \$79CD
 k = \$795E
 m-k = \$6F

alternativ:

$$(m-k)_{2k} = \$100 - \$6F = \$91$$

1.5 Übersicht der implementierten Monitorroutinen

Name	Funktion	Adresse
1. Umwandlungsroutinen		
T7SG	Umwandlung der unteren Tetrade von B in den 7-Segment-Code, Ergebnis in A	F100
B7SG	Umwandlung beider Tetrade von B in den 7-Segment-Code, Ergebnis in D=A,B	F103
D7SG	Umwandlung der vier Tetrade von D in den 7-Segment-Code, Ergebnis in D,Y	F106
2. Darstellungsroutinen		
CLRDISP	Löschen der Anzeige	F110
SHOWA	Bringt A in die Anzeige, Position in X	F113
SHOWD	Bringt D in die Anzeige, Position in X	F116
SHOWYD	Bringt Y,D in die Anzeige, Position in X	F119
SHOWT7SG	Umwandlung der unteren Tetrade von B in 7-Segment-Code, Darstellung in der Anzeige, Position in X	F11C
SHOWB7SG	Umwandlung beider Tetraden von B in den 7-Segment-Code, Darstellung in der Anzeige, Position in X	F120
SHOWD7SG	Umwandlung der vier Tetraden von D in den 7-Segment-Code, Darstellung in der Anzeige, Position in X	F123
3. Routinen zur Bearbeitung des Anzeige-Puffers Adresse des Puffers in X		
CLDBUF	Löschen des Anzeigepuffers	F130
SHOWDBUF	Übertragen des Puffers in die Anzeige	F133
RRDBUF	Rotieren des Puffers um eine Stelle nach rechts	F136
RLDBUF	Rotieren des Puffers um eine Stelle nach links	F139
COPYDBUF	Kopieren eines 2. Puffers in den Anzeigepuffer	F13C
4. Routinen zur Abfrage der Tastatur		
KEY	Lesen der Tastatur ohne Warten, Tastencode nach B	F140
HALTKEY	Lesen der Tastatur mit Warten, Tastencode nach B	F143
SHOWKEY	Lesen der Tastatur ohne Warten und Anzeigen, Tastencode nach B	F146
SHOWHKEY	Lesen der Tastatur mit Warten und Anzeigen, Tastencode nach B	F149
INDATA	Einlesen eines 8-bit-Datums über die Tastatur, Datum nach A, Tastencode nach B	F14C
SHOWDATA	Einlesen eines 8-bit-Datums über die Tastatur und Anzeigen, Datum nach A, Tastencode nach B	F150
INADR	Einlesen einer 16-bit-Adresse über die Tastatur, Adresse nach Y, Tastencode nach B	F153
SHOWADR	Einlesen einer 16-bit-Adresse über die Tastatur und Anzeigen, Adresse nach Y, Tastencode nach B	F156
5. weitere Routinen		
DLY1MS	Schleife zur Zeitverzögerung, Zeitdauer in Y vorgegeben	F160
RANDOM	Pseudo-Zufallszahlengenerator, alter und neuer Wert in Y	F163
COPYXD	Verschieben von Speicherbereichen, Startadressen in X,Y, Länge in D	F166

Tabelle: Tastencodes für die Hilfsroutinen

Datentasten				Funktionstasten			
Taste	Code	Taste	Code	Taste	Code	Taste	Code
0	\$00	8	\$08	+	\$80	T	\$88
1	\$01	9	\$09	-	\$81	F1	\$89
2	\$02	A	\$0A	A	\$82	F2	\$8A
3	\$03	B	\$0B	D	\$83	F3	\$8B
4	\$04	C	\$0C	R	\$84		
5	\$05	D	\$0D	G	\$85		
6	\$06	E	\$0E	S	\$86		
7	\$07	F	\$0F	L	\$87		

Tabelle: Matrix zur Dekodierung der Tasten

Adresse	Tastenbetätigung			
F010	F3	F2	F1	T
F011	0	4	8	C
F012	1	5	9	D
F013	2	6	A	E
F014	3	7	B	F
F015	+	A	R	S
F016	-	D	G	L
Datenbus:	X1	X2	X4	X8

Adressen der Anzeigestellen

1. Stelle S0 (rechts): \$F020 und \$F028
2. Stelle S1: \$F021 und \$F029
-
8. Stelle S7 (links): \$F027 und \$F02F

• Praktische Übung: Dezimalzähler

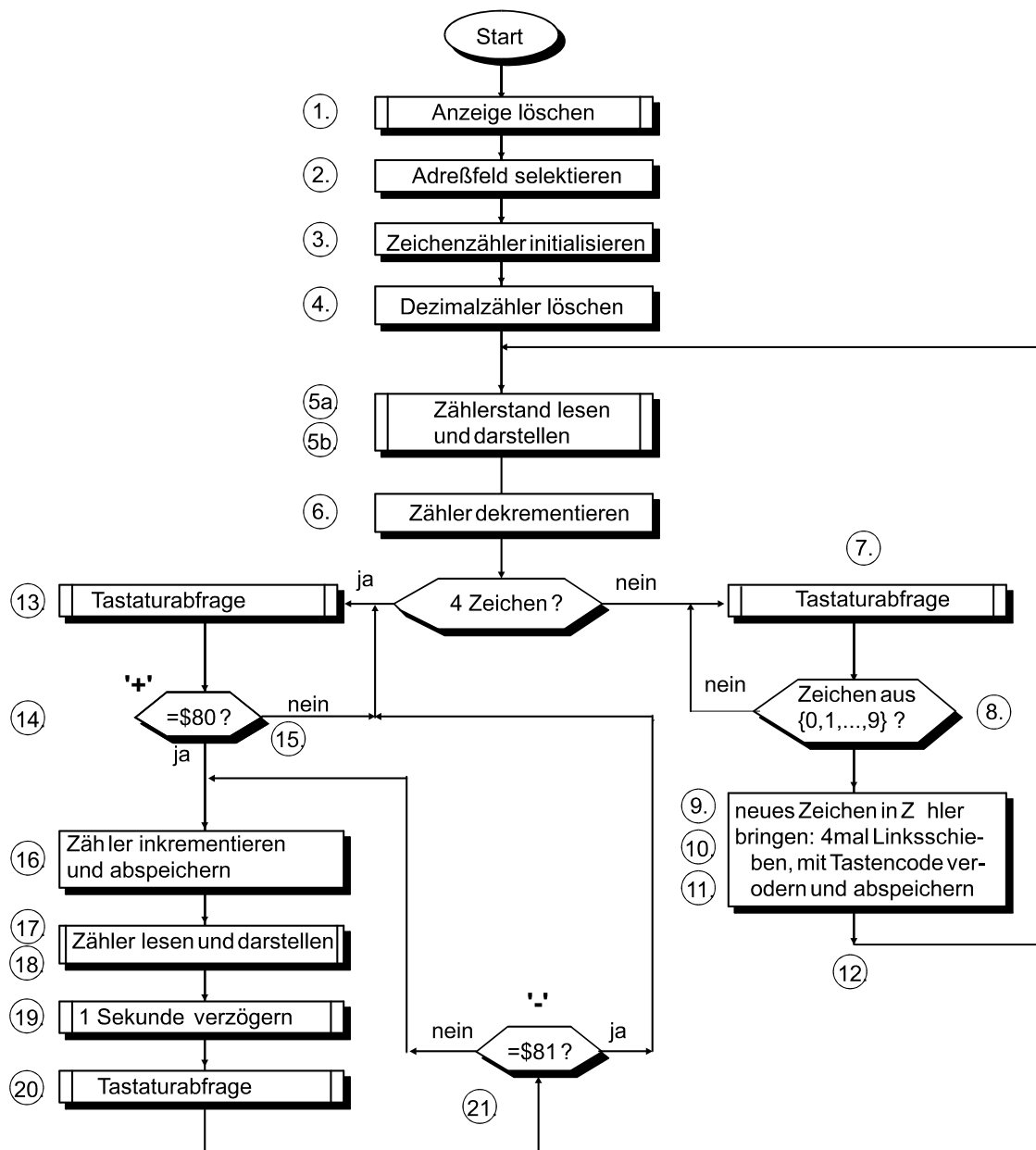
Gesucht ist ein Programm, das im Adreßfeld S5 - S2 der Anzeige einen 4-stelligen Dezimalzähler realisiert.

Im einzelnen soll das Programm:

1. die Anzeige löschen,
2. über die Tastatur einen 4-stelligen Startwert einlesen, wobei nur die Ziffern 0, 1, ..., 9 akzeptiert werden sollen, und diesen Startwert im Adreßfeld anzeigen,
3. nach Betätigen der Funktionstaste '+' den Zählvorgang mit einer Zählfrequenz von (ungefähr) 1 Hz beginnen,
4. nach Betätigen der Funktionstaste '-' den Zählvorgang solange stoppen, bis er durch die Taste '+' wieder gestartet wird,
5. bei Erreichen des Endwertes '9999' mit dem Wert '0000' zyklisch weiterzählen.

Alle nicht genannten Funktionstasten sollen keine Wirkung haben. (Natürlich außer 'F4' und 'C'.)

- Flußdiagramm



• **Musterlösung zur Praktischen Übung „Dezimalzähler“:**

```

1 F110 CLRDISP EQU $F110
2 F123 SHOWD7SG EQU $F123
3 F143 HALTKEY EQU $F143
4 F140 KEY EQU $F140
5 F160 DLY1MS EQU $F160
6 0000 ZAEHLER EQU $0000
7 0001 LZAEHLER EQU $0001
8 0000 HZAEHLER EQU $0000
9 0002 ZEIZAEH EQU $0002
10
11 0400 ORG $0400
12 0400 BD F1 10 START JSR CLRDISP ; Löschen der Anzeige
13 0403 8E 00 02 LDX #2 ; Adreßeld der Anzeige
14 0406 86 04 LDA #4 ; Zeichenzähler nach
15 0408 97 02 STA ZEIZAEH ; Speicherstelle 2
16 040A 0F 00 CLR HZAEHLER ; Dezimalzähler H-Byte löschen
17 040C 0F 01 CLR LZAEHLER ; L-Byte löschen
18 040E DC 00 EINGABE LDD ZAEHLER ; Zaehlerstand lesen
19
20 0410 BD F1 23 JSR SHOWD7SG ; und darstellen
21 0413 0A 02 DEC ZEIZAEH ; Zeichenzähler dekrementieren
22 0415 2B 1D BMI EINENDE ; falls alle Zeichen eingegeben
23
24 0417 BD F1 43 TASTE JSR HALTKEY ; Tastaturabfrage
25 041A C1 0A CMPB #$A ; Taste aus {0,...,9} oder {A,...,F}
26 041C 24 F9 BHS TASTE ; Sprung, falls nicht aus {0,...,9}
27
28 041E 08 01 LSL LZAEHLER ; neues Zeichen in den Zähler
29 0420 09 00 ROL HZAEHLER ; diesen zunächst viermal nach
30 0422 08 01 LSL LZAEHLER ; links verschieben
31 0424 09 00 ROL HZAEHLER
32 0426 08 01 LSL LZAEHLER
33 0428 09 00 ROL HZAEHLER ; und dann
34 042A 08 01 LSL LZAEHLER
35 042C 09 00 ROL HZAEHLER
36 042E DA 01 ORB LZAEHLER ; Tastendruck verodern
37 0430 D7 01 STB LZAEHLER ; und abspeichern
38 0432 20 DA BRA EINGABE ; weiter mit Eingabe Anfangszustand
39

```

40	0434	BD F1 43	EINENDE	JSR	HALTKEY	; Tastaturabfrage
41	0437	C1 80		CMPB	#\$80	; Taste '+' gedrückt ?
42	0439	26 F9		BNE	EINENDE	; Ruecksprung, falls nicht
43						
44	043B	96 01	ZAEHLEN	LDA	LZAEHLER	; Inkrementieren des Zählers
45	043D	8B 01		ADDA	#1	; zunaechst L-Byte
46	043F	19		DAA		; Dezimalkorrektur
47	0440	97 01		STA	LZAEHLER	; abspeichern
48	0442	86 00		LDA	#0	; loeschen von A
49	0444	99 00		ADCA	HZAEHLER	; Addieren des H-Bytes
50	0446	19		DAA		; Dezimalkorrektur
51	0447	97 00		STA	HZAEHLER	; abspeichern
52	0449	DC 00	LDD	ZAEHLER		; Zaehlerstand lesen
53	044B	BD F1 23		JSR	SHOWD7SG	; und darstellen
54	044E	10 8E 03 E8		LDY	#\$03E8	; Maß für 1 Sekunde Verzögerung
55	0452	BD F1 60		JSR	DLY1MS	; verzögern
56						
57	0455	BD F1 40		JSR	KEY	; Tastaturabfrage ohne Halt
58	0458	C1 81		CMPB	#\$81	; Taste '-' gedrückt ?
59	045A	26 DF	BNE	ZAEHLEN		; falls nein, weiterzählen
60	045C	20 D6	BRA	EINENDE		; falls ja, auf Taste '+' warten
61						
62	045E					

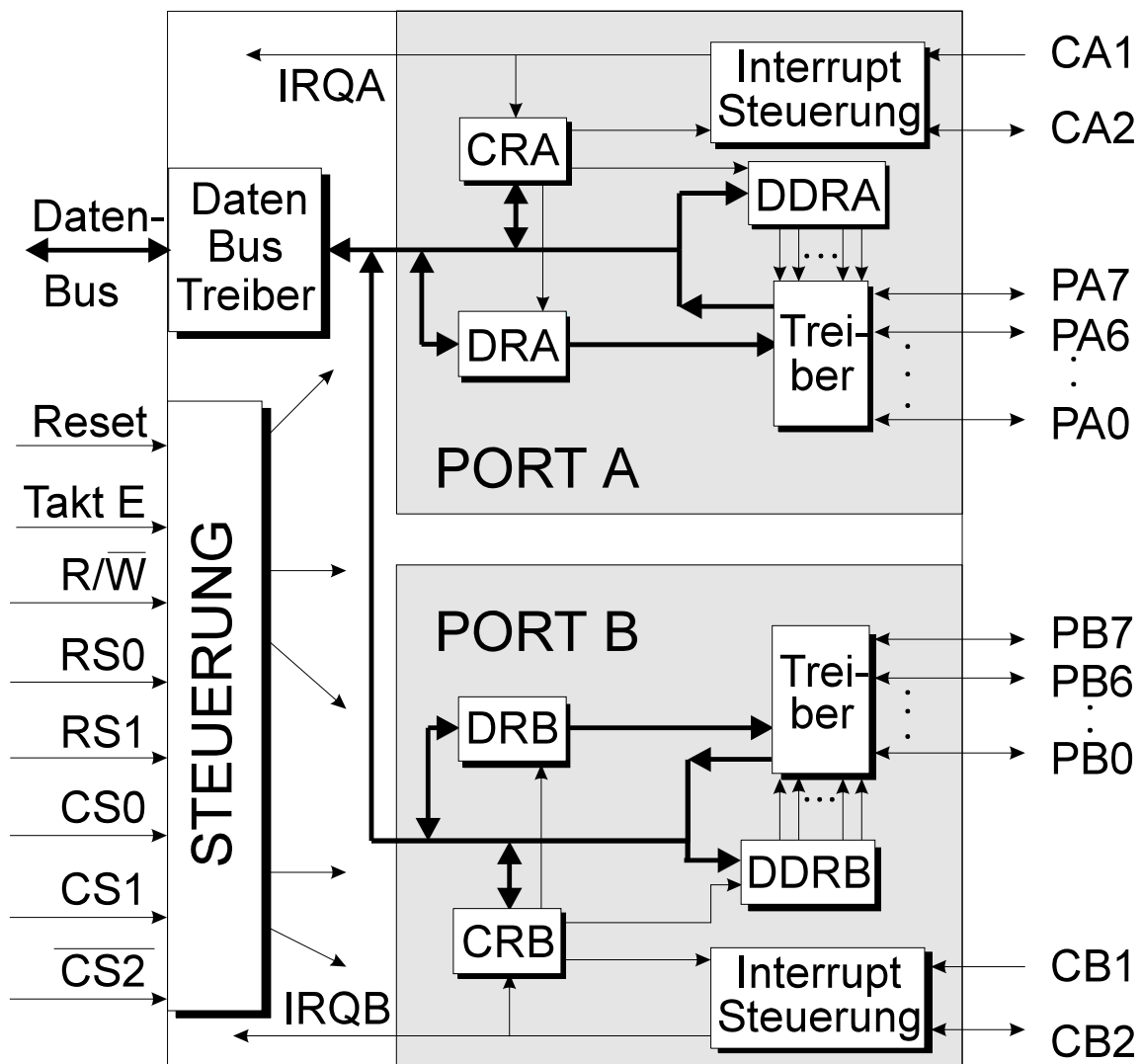
2. Die Peripheriekomponenten

2.1 Interruptvektor-Tabelle

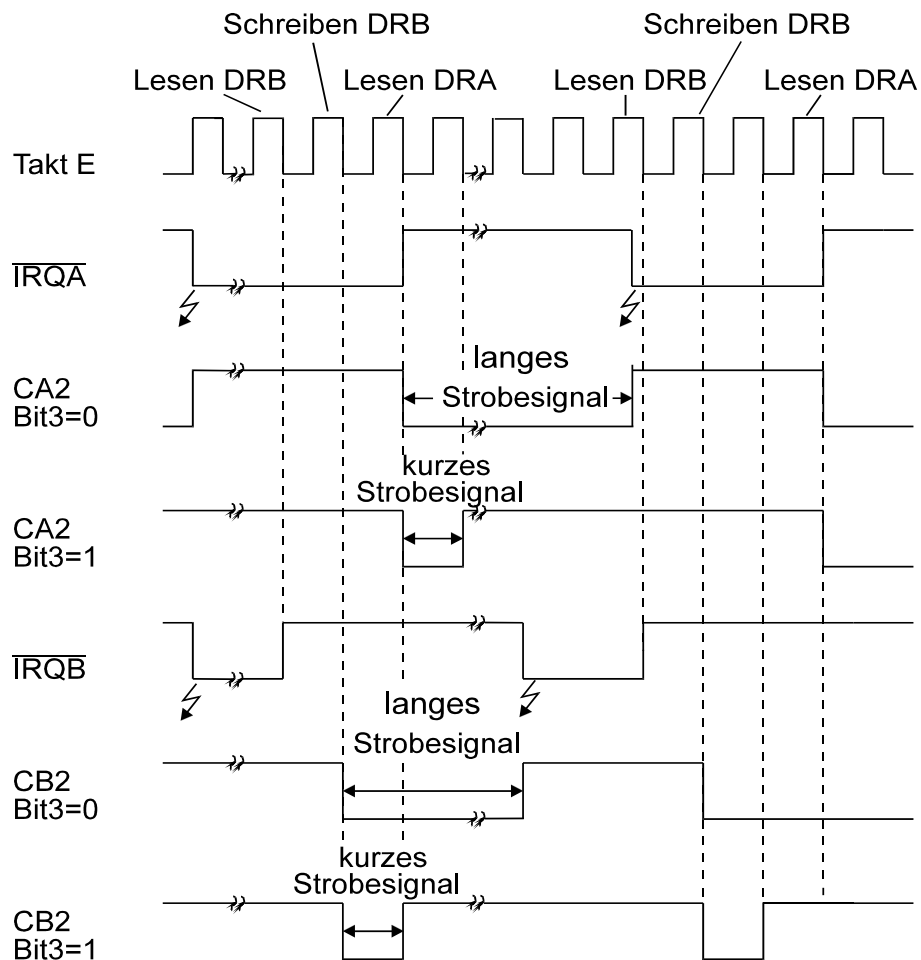
Adressen	Quelle, Baustein	Vorbelegung	Funktion
\$002C, \$002D	Timer #1, MC6840	\$E667	RTI
\$002E, \$002F	Timer #3, MC6840	\$E667	RTI
\$0030, \$0031	Timer #2, MC6840	\$E670	Trace
\$0032, \$0033	NMI Steckerleiste	\$E667	RTI
\$0034, \$0035	FIRQ Steckerleiste	\$E667	RTI
\$0036, \$0037	CA1, MC6821, Port PA	\$E520	Break
\$0038, \$0039	CB1, MC6821, Port PB	\$E667	RTI
\$003A, \$003B	CA2, MC6821, Port PA	\$E667	RTI
\$003C, \$003D	CB2, MC6821, Port PB	\$E667	RTI
\$003E, \$003F	IRQ, MC6850	\$E667	RTI
\$0040, \$0041	IRQ, R6551, V.24	\$E667	RTI
\$0042, \$0043	IRQ Steckerleiste	\$E667	RTI
\$0044, \$0045	SWI1 (SWI)	\$EC40	Breakpoint
\$0046, \$0047	SWI2 aus ROM	\$E4E0	GO-Routine
\$0048, \$0049	SWI2 aus RAM	\$E667	RTI
\$004A, \$004B	SWI3 aus ROM	\$E640	TRACE-Routine
\$004C, \$004D	SWI3 aus RAM	\$E667	RTI

2.2 Der Parallelport-Baustein MC6821

- Architektur



- Handshake-Signale zur Datenübertragung



Takt E: 1 Zyklus = 0,5 bzw. 1 µs (je nach Typ)

IRQA ausgelöst durch CA1

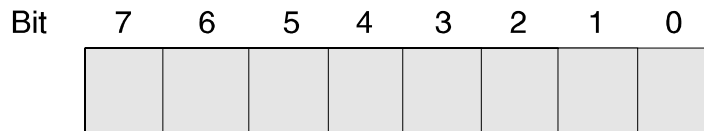
⚡ Interrupt

IRQB ausgelöst durch CB1

- Registersatz

Daten-/Datenrichtungsregister

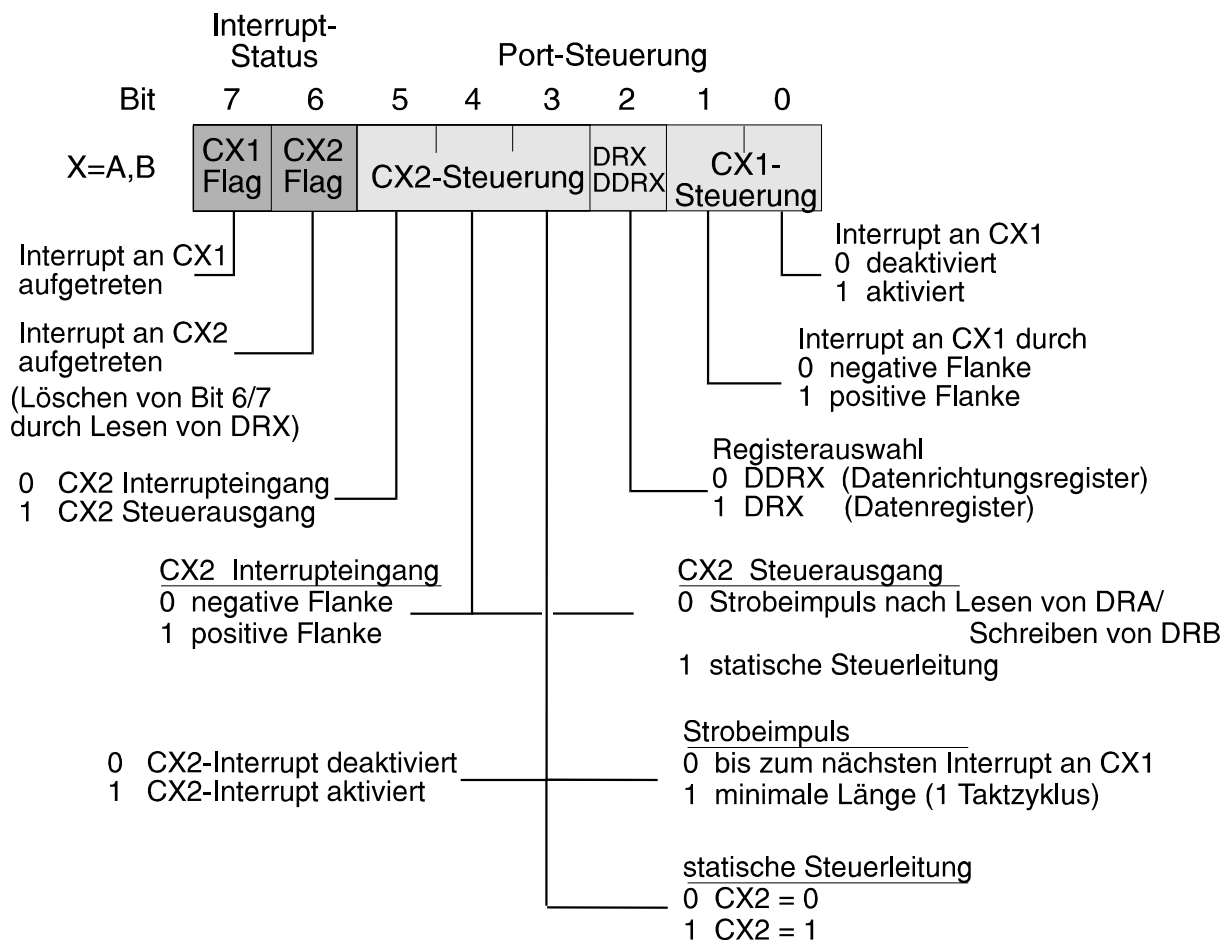
Adressen: DRA, DDRA \$F000
DRB, DDRB \$F002



DDRXi = 0 Portleitung PXi ist Eingang
1 Portleitung PXi ist Ausgang X=A,B

Steuerregister

Adressen: CRA \$F001
CRB \$F003



• **Praktische Übung: Parallelport**

In dieser Aufgabe sollen die am Port PB anliegenden Binär-Informationen genau dann eingelesen und in der Anzeige hexadezimal dargestellt werden, wenn das Programm im "Einlesemodus" ist. Die Umschaltung zwischen den Modi "Einlesen/nicht Einlesen" soll durch die Tastatur geschehen.

Im einzelnen soll das Programm

1. die Anzeige löschen,
2. den Port PB geeignet initialisieren

und danach zyklisch

3. die Tastatur auf Betätigung der Funktionstasten "+" und "-" abfragen, (alle anderen Tasten sollen keine Funktion haben!)
4. sich nach Betätigen der Taste "-":
 - im Modus 0 befinden, in dem keine Eingabe über den Port PB möglich ist, und
 - dies durch die Ziffer "0" in der Anzeigestelle S0 anzeigen
(alle anderen Anzeigestellen gelöscht !)
 - sowie durch eine ausgeschaltete LED an CB2 anzeigen,
 - eine Einlese-Anforderungen über den Schalter an CB1 ignorieren (und im Statusregister des Ports löschen),
5. sich nach Betätigen der Taste "+":
 - im Modus 1 befinden, in dem eine Eingabe über den Port PB möglich ist, und
 - dies durch die Ziffer "1" in der Anzeigestelle S0
 - sowie durch eine eingeschaltete LED an CB2 anzeigen,
 - Einlese-Anforderungen über den Schalter an CB1 auswertet, danach löscht und
 - die eingelesene Binär-Information als Hexadezimal-Zahl in den Stellen S7,S6 darstellen.

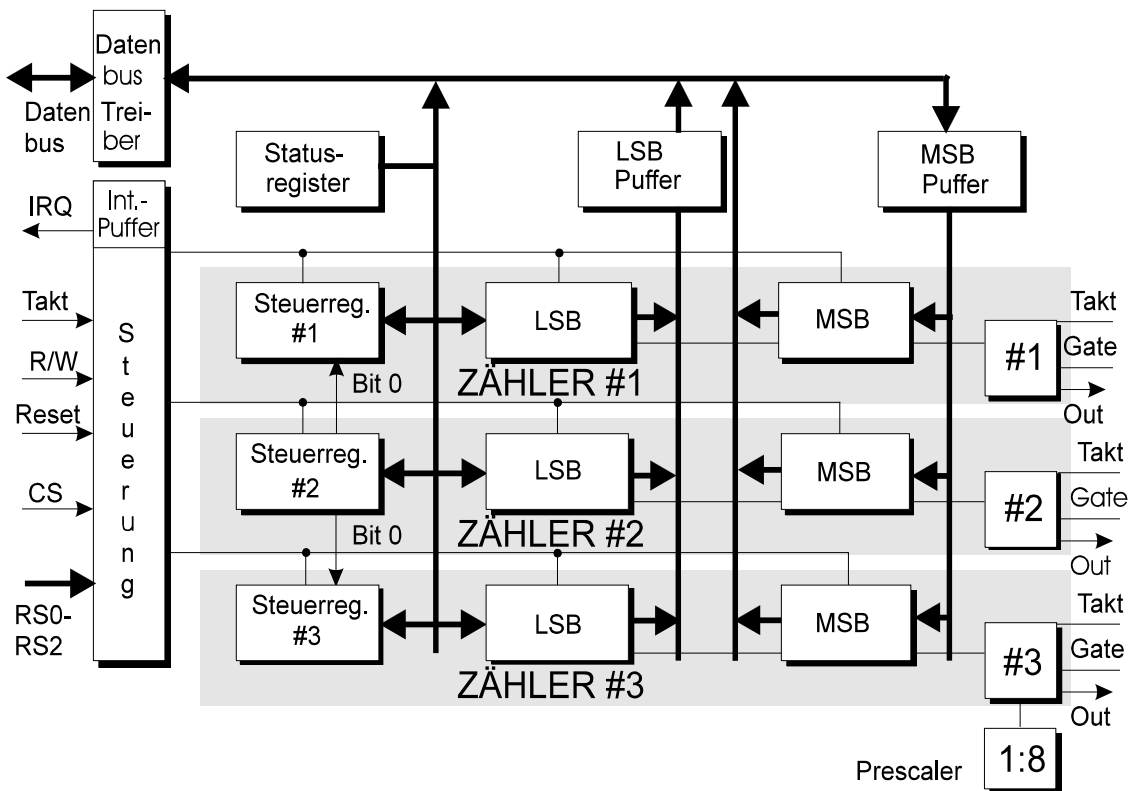
• Musterlösung

0001 f110	CLRDISP	EQU	\$F110	
0002 f140	KEY	EQU	\$F140	
0003 f143	HALTKEY	EQU	\$F143	
0004 f113	SHOWA	EQU	\$F113	
0005 f116	SHOWD	EQU	\$F116	
0006 f11c	SHOWT7SG	EQU	\$F11C	
0007 f120	SHOWB7SG	EQU	\$F120	
0008 f123	SHOWD7SG	EQU	\$F123	
0009 f156	SHOWADR	EQU	\$F156	
0010 f150	SHOWDATA	EQU	\$F150	
0011 f160	DLY1MS	EQU	\$F160	
0012 0000	E0	EQU	\$0	
0013 0001	E1	EQU	\$1	
0014 0000	S0	EQU	\$0	
0015 0006	S6	EQU	\$6	
0016 f002	DRB	EQU	\$F002	
0017 f002	DRRB	EQU	\$F002	
0018 f003	CRB	EQU	\$F003	
0019				
0020				
0021 0400		ORG	\$0400	; Programmanfang auf \$0400 setzen
0022 0400 bd f1 10	EINGABE	JSR	CLRDISP	; Anzeige loeschen
0023 0403 c6 30		LDB	#\$30	; Steuerregister PB initialisieren,
0024 0405 f7 f0 03		STB	CRB	; CB2 aus, DDRB selektieren
0025 0408 c6 00		LDB	#\$00	; alle Leitungen auf Eingang
0026 040a f7 f0 02		STB	DRRB	; schalten
0027 040d c6 34		LDB	#\$34	; DRB selektieren, CB2 aus
0028 040f f7 f0 03		STB	CRB	; " "
0029				
0030 0412 8e 00 00	LOOP	LDX	#\$0	; X auf S0 setzen
0031 0415 bd f1 40		JSR	KEY	; Tastatur ohne Halt abfragen
0032 0418 c1 80		CMPB	#\$80	; '+'-Taste gedrueckt ?
0033 041a 26 0c		BNE	MINUS	; falls nicht, minus
0034				
0035 041c c6 3c	PLUS	LDB	#\$3C	; '+' gedrueckt, CB2 an
0036 041e f7 f0 03		STB	CRB	; " "
0037 0421 c6 01		LDB	#\$1	; 1 laden
0038 0423 bd f1 1c		JSR	SHOWT7SG	; und darstellen
0039 0426 20 0e		BRA	WEITER	; nach weiter springen
0040				
0041 0428 c1 81	MINUS	CMPB	#\$81	; '-' gedrueckt ?
0042 042a 26 0a		BNE	WEITER	; falls nicht, nach weiter
0043 042c c6 34		LDB	#\$34	; CB2 aus
0044 042e f7 f0 03		STB	CRB	; " "
0045 0431 c6 00		LDB	#0	; 0 laden
0046 0433 bd f1 1c		JSR	SHOWT7SG	; und anzeigen
0047				
0048 0436 8e 00 06	WEITER	LDX	#\$6	; X auf S6 setzen
0049 0439 f6 f0 03		LDB	CRB	; Steuerregistrer CRB lesen

0050 043c c4 08		ANDB #\$08	; Abfrage, ob CB2 an
0051 043e 26 0b		BNE ENABLE	; falls nicht, nach Enable
0052			
0053 0440 f6 f0 02	DISABLE	LDB DRB	; Port PB nach B laden
0054 0443 cc 00 00		LDD #0	; Accu D auf 0 setzen
0055 0446 bd f1 16		JSR SHOWD	; und darstellen
0056 0449 20 c7		BRA LOOP	; nach LOOP springen
0057			
0058 044b f6 f0 03	ENABLE	LDB CRB	; CRB nach B laden
0059 044e 2a c2		BPL LOOP	; falls kein Interrupt an CB1, nach LOOP
0060 0450 f6 f0 02		LDB DRB	; Port PB nach B laden
0061 0453 bd f1 20		JSR SHOWB7SG	; und darstellen
0062 0456 20 ba		BRA LOOP	; nach LOOP springen
0063		END	

2.3 Der Timer-Baustein MC6840

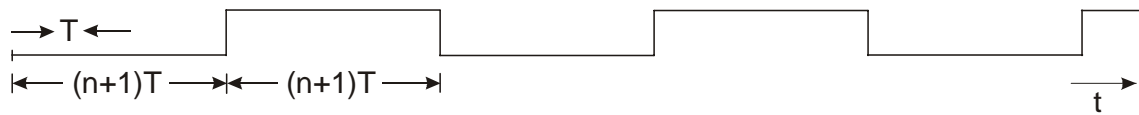
- **Architektur**



Adresse	RS2	RS1	RS0	Schreiben R/W=0	(Lesen(R/W=1)
\$F018	0	0	0	CR#3 (Bit0=0 im CR#2)	-
\$F018	0	0	0	CR#1 (Bit0=1 im CR#2)	-
\$F019	0	0	1	CR#2	Statusregister
\$F01A	0	1	0	MSB-Pufferregister	MSB-Zähler #1
\$F01B	0	1	1	Latches #1	LSB-Puffer
\$F01C	1	0	0	MSB-Pufferregister	MSB-Zähler #2
\$F01D	1	0	1	Latches #2	LSB-Puffer
\$F01E	1	1	0	MSB-Pufferregister	MSB-Zähler #3
\$F01F	1	1	1	Latches #3	LSB-Puffer

- **Zählmodi**

Zyklischer Zählbetrieb: 16-Bit-Modus (Squarewave)



2 x 8-Bit-Modus:



Monoflop-Zählbetrieb: 16-Bit-Modus

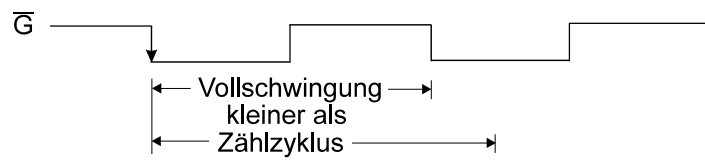
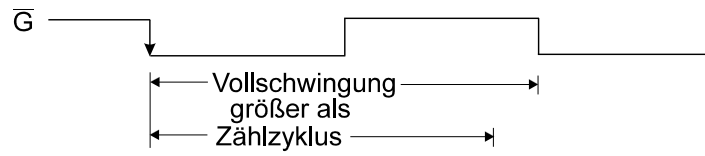


2 x 8-Bit-Modus:

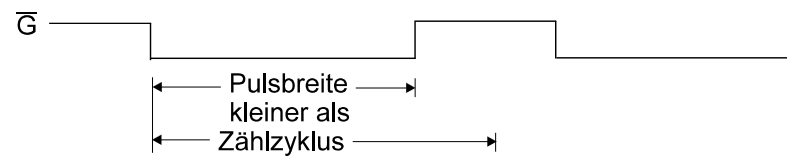
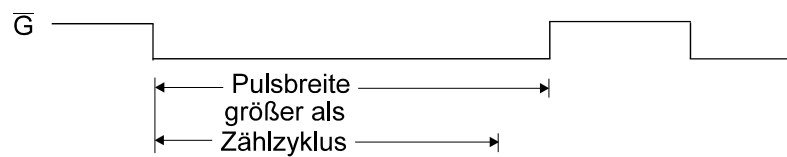


- Frequenz- und Impulsmessung

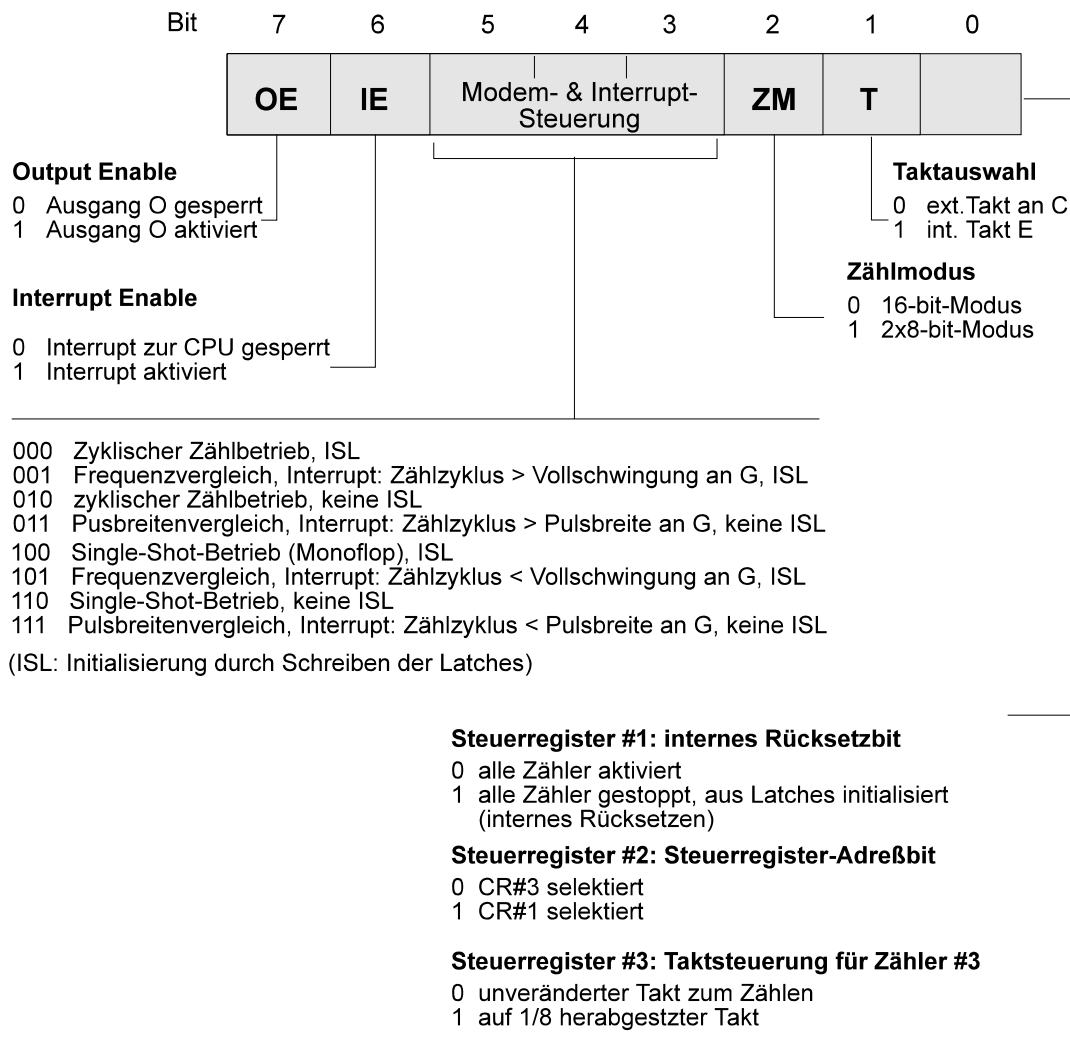
Frequenzvergleich



Pulsbreitenvergleich



- **Steuerregister CR#1, CR#2, CR#3:**



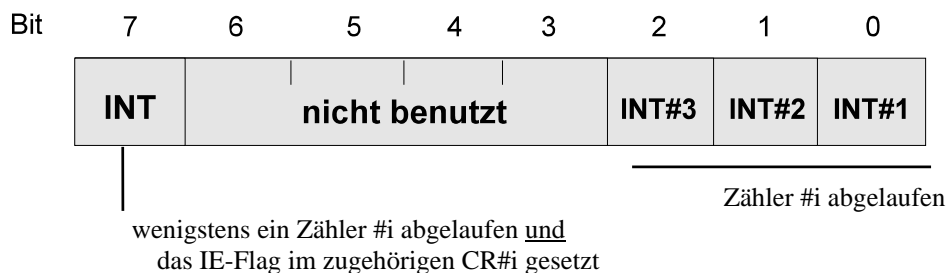
Registeradressen:

CR#1: \$F018 und Bit 0 von CR#2=1
 CR#2: \$F019
 CR#3: \$F018 und Bit 0 von CR#2=0

Latches: \$F01A, \$F01B
 Latches: \$F01C, \$F01D
 Latches: \$F01E, \$F01F

- **Statusregister:**

(Adresse: \$F019)



- **Praktische Übung: Timer-Baustein**

In dieser Aufgabe soll ein vom Zeitgeber-/Zählerbaustein ausgegebenes Rechtecksignal durch den Prozessor überwacht werden. Sobald dieses Signal durch den Steuereingang des Zählers #3 zu lange unterbrochen wird, wird die Ausgabe gestoppt, bis sie eventuell vom Benutzer erneut gestartet wird.

Im einzelnen soll das Hauptprogramm,

1. die Anzeige löschen,
2. den Zähler #3 des MC6840 so initialisieren (noch nicht starten!), daß er ein periodisches Rechtecksignal maximaler Dauer (minimaler Frequenz) ausgeben und den Prozessor am Ende jedes Zählzyklus durch einen Interrupt informieren kann,
3. den Anfangswert '0000' in den Anzeigestellen S5 - S2 und die Kennung 'S' (für 'START') in der Stelle S0 ausgeben und auf die Betätigung der Tastatur warten,
4. nach Betätigen der Funktionstaste 'S' :
 - a) den Zähler #3 des MC6840 starten,
 - b) in der Anzeigestelle S0 die Kennung 'C' (für 'COUNT') ausgeben,
5. eine Zeitverzögerung von ca. 16 Sekunden ablaufen lassen,
6. feststellen, ob während der gesamten Verzögerungszeit der Zähler #3 über seinen Steuereingang G3 gesperrt war und
 - den Zähler #3 stoppen und zu 3. zurückspringen, falls dies der Fall war,
 - mit 5. weitermachen, falls es nicht der Fall war.

In der Interruptroutine des Zählers #3 soll

1. der Hexadezimal-Zähler in den Anzeigestellen S5 - S2 nach jedem Zählzyklus des Zählers #3 um 1 erhöht werden,
2. das Hauptprogramm geeignet darüber informiert werden, daß der Zähler #3 einen Zählzyklus beendet hat und also nicht gesperrt war.

Hinweise:

- Vor Verlassen der Interruptroutine muß das Interrupt Flag des Zählers #3 zurückgesetzt werden.
- Alle jeweils nicht genannten Funktionstasten sollen keine Wirkung haben.

- Musterlösung:**

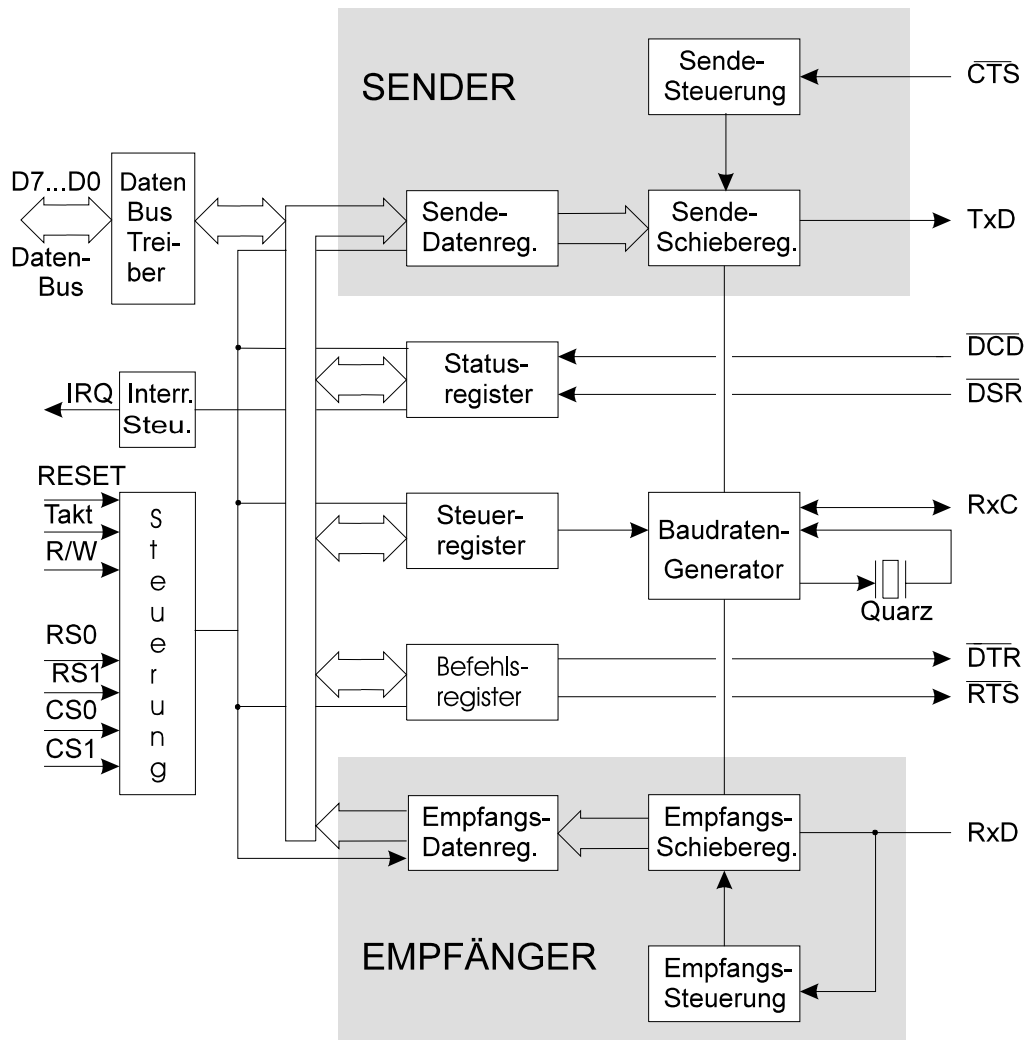
```

1  0000      S0      EQU    $0      ; Anzeigestelle S0
2  0002      S2      EQU    $2      ; Anzeigestelle S2
3  0000      ZAEHLER EQU    $0      ; Zdhler in Speicherstelle 0
4  0002      FLAG    EQU    $2      ; Benutzer-Flag
5  002E      INT_Z3   EQU    $002E ; Interrupt-Vektor für Timer #3
6
7  F110      CLRDISP  EQU    $F110
8  F11C      SHOWT7SG EQU    $F11C
9  F120      SHOWB7SG EQU    $F120
10 F123      SHOWD7SG EQU    $F123
11 F143      HALTKEY   EQU    $F143
12 F160      DLY1MS    EQU    $F160
13 002E      INT3      EQU    $002E
14 F018      SR_Z1     EQU    $F018 ; Steuerregister Timer #1
15 F018      SR_Z3     EQU    $F018 ; Steuerregister Timer #3
16 F019      SR_Z2     EQU    $F019 ; Steuerregister Timer #2
17 F019      STREG     EQU    $F019 ; SStatusregister aller Timer
18 F01E      LATCH_Z3  EQU    $F01E ; Latches für Timer #3
19
21 0400                      ORG    $0400
22 0400      CC 04 53      START  LDD    #INT
23 0403      DD 2E                      STD    INT_Z3
24 0405      7F F0 19                      CLR    SR_Z2
25 0408      86 C3                      LDA    #$C3
26 040A      B7 F0 18                      STA    SR_Z3
27 040D      CC 80 FF                      LDD    #$80FF
28 0410      FD F0 1E                      STD    LATCH_Z3
29 0413      86 01                      LDA    #$01
30 0415      B7 F0 19                      STA    SR_Z2
31 0418
32 0418      BD F1 10      ANFANG  JSR    CLRDISP
33 041B      8E 00 02                      LDX    #S2
34 041E      CC 00 00                      LDD    #$0000
35 0421      DD 00                      STD    ZAEHLER
36 0423      BD F1 23                      JSR    SHOWD7SG
37 0426      8E 00 00                      LDX    #S0
38 0429      C6 05                      LDB    #$5
39 042B      BD F1 1C                      JSR    SHOWT7SG
40
41 042E      BD F1 43      WDTS    JSR    HALTKEY
42 0431      C1 86                      CMPB   #$86
43 0433      26 F9                      BNE    WDTS
44 0435      7F F0 18                      CLR    SR_Z1
45 0438      C6 0C                      LDB    #$0C
46 043A      BD F1 1C                      JSR    SHOWT7SG
47
48 043D      C6 01      LOOP      LDB    #$01
49 043F      D7 02                      STB    FLAG
50 0441      10 8E 40 00                      LDY    #$4000
51 0445      BD F1 60                      JSR    DLY1MS
52 0448      D6 02                      LDB    FLAG
53 044A      27 F1                      BEQ    LOOP
54 044C      86 01                      LDA    #$1
55 044E      B7 F0 18                      STA    SR_Z1
56 0451      20 C5                      BRA    ANFANG
57
58 0453      0F 02      INT      CLR    FLAG
59 0455      DC 00                      LDD    ZAEHLER
60 0457      C3 00 01                      ADDD   #1
61 045A      DD 00                      STD    ZAEHLER
62 045C      8E 00 02                      LDX    #S2
63 045F      BD F1 23                      JSR    SHOWD7SG
64 0462      F6 F0 19                      LDB    STREG
65 0465      FC F0 1E                      LDD    LATCH_Z3
66 0468      3B                      RTI

```

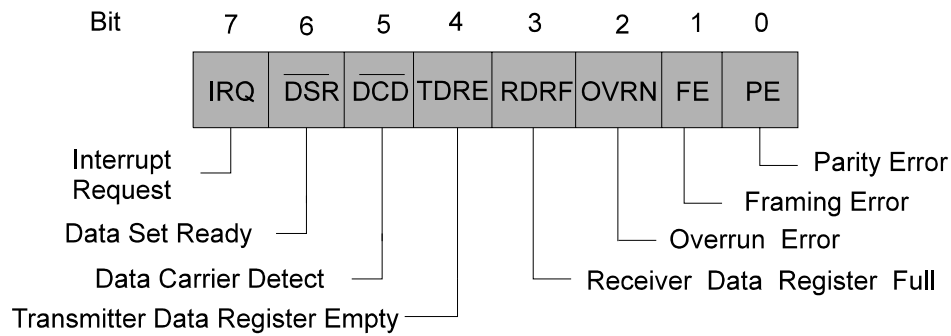

2.4 Der ACIA-Baustein R6551

- Architektur

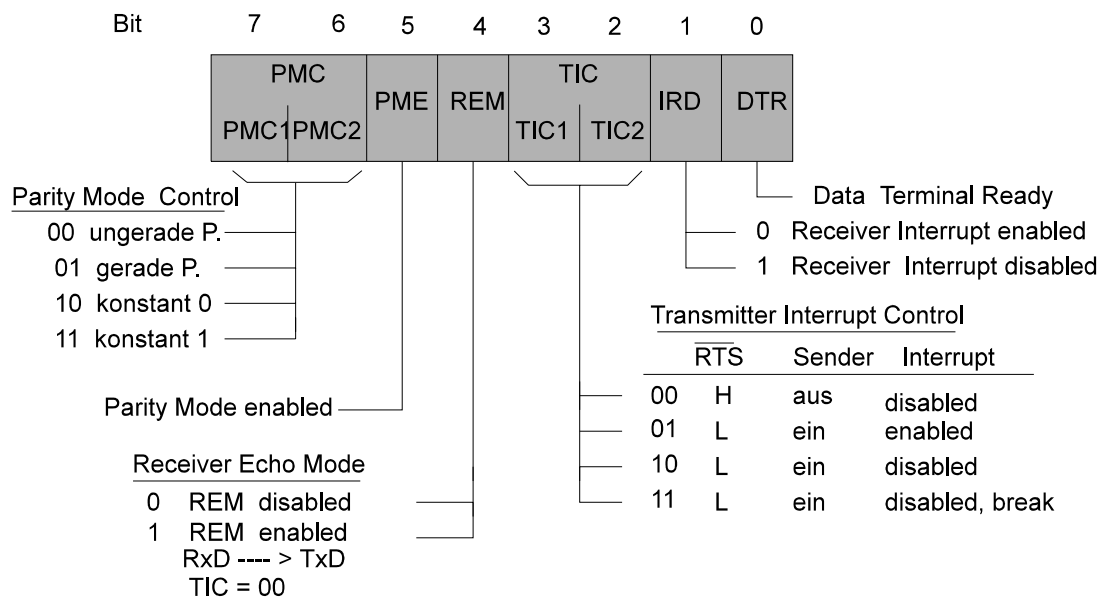


- **Registersatz**

- * **Statusregister**



- * **Befehlsregister**



* Steuerregister

