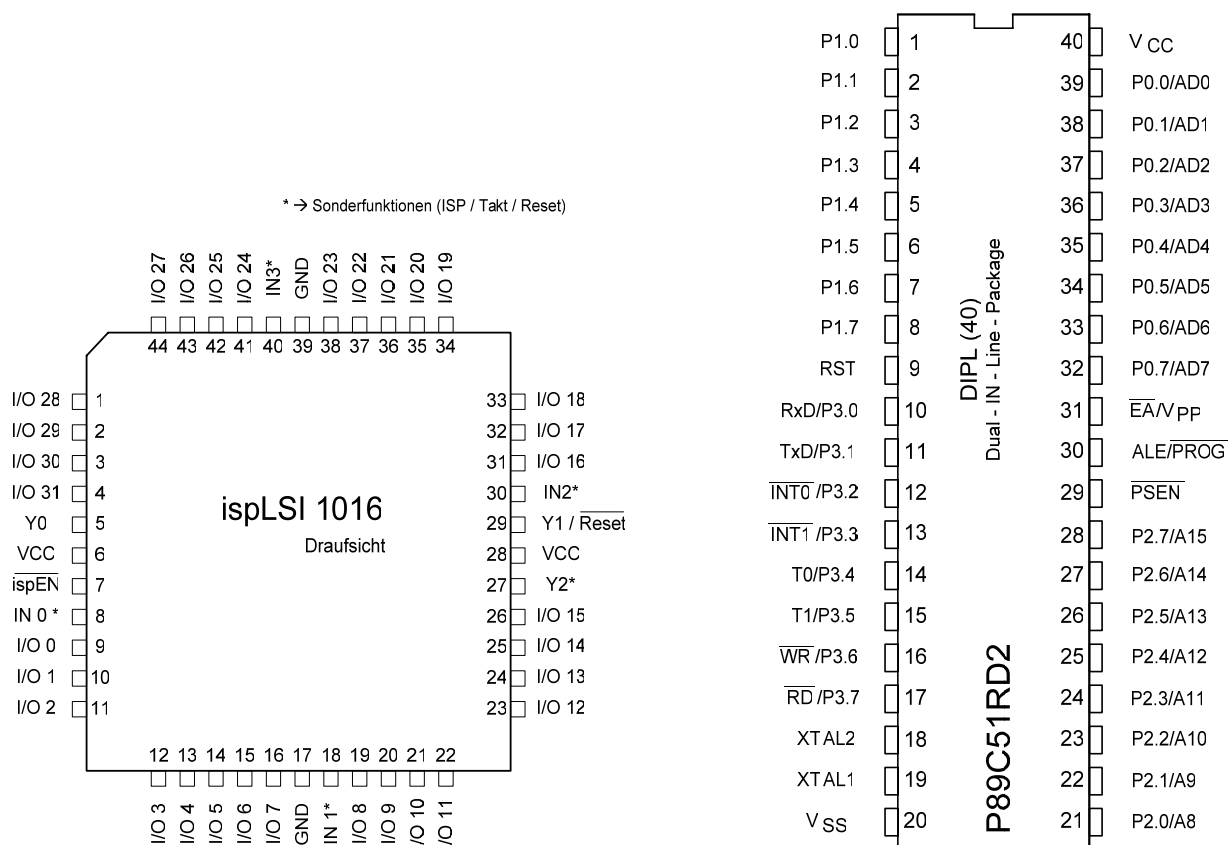


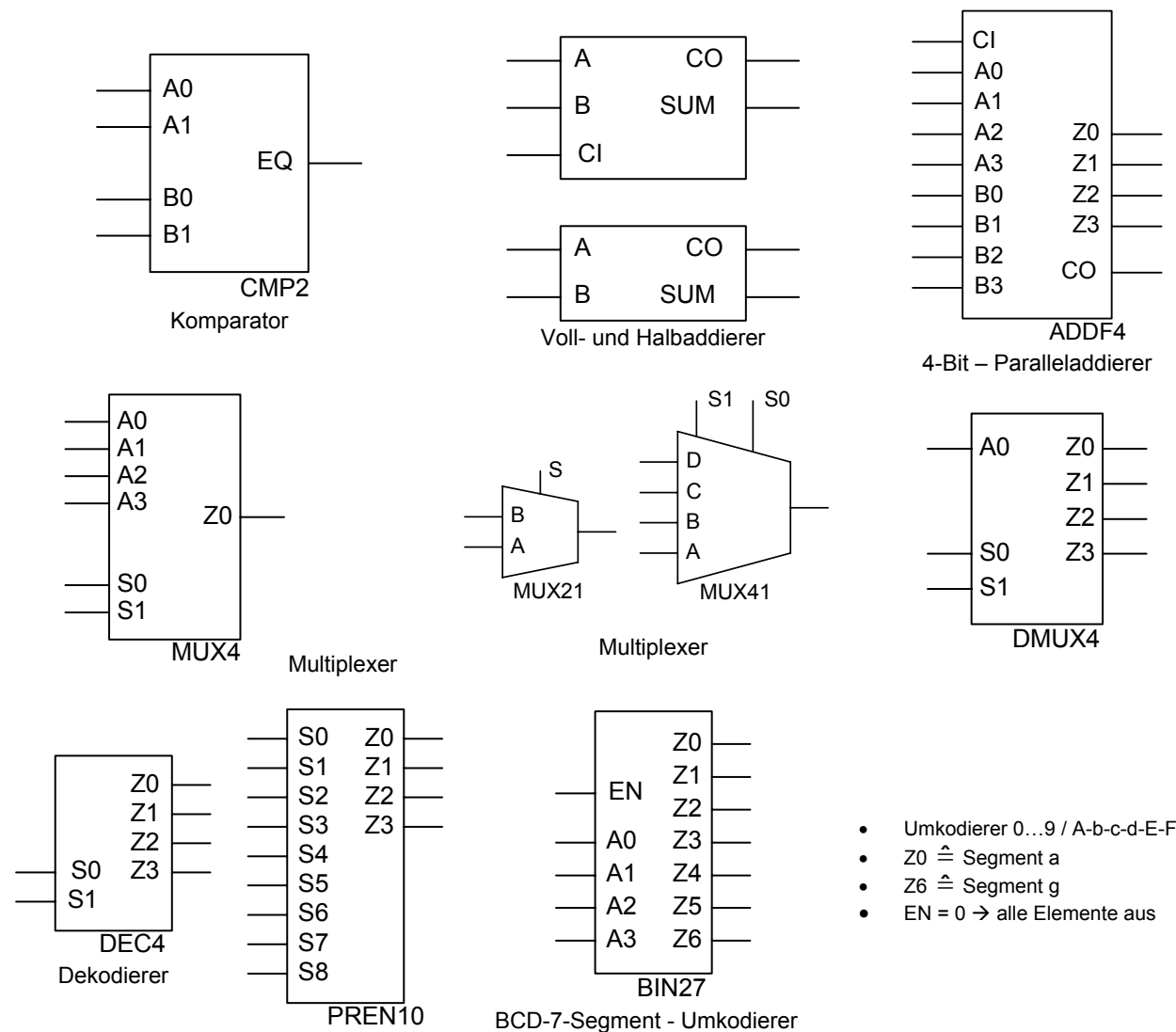
ISP (In System Programmable)	1
DesignExpert V8.1® - Schematic – Blockschaltbilder.....	2
Schaltnetze	2
Schaltwerke	2
Ausgangs-Makrozelle	5
Pinbelegung des 1016-Exp.Board	6
8051 - Controller	7
Befehlssatz	7
Datentransport	8
Arithmetische Operationen	9
Logische Operationen	10
Sprungbefehle.....	11
Speicheraufteilung	12
Internes RAM	13
Interrupt & Timer	14
Interrupt Enable (IE) Register	14
Interruptstruktur des 8051	16
Register zur Zählerkontrolle: TMOD und TCON:	17
Blockschaltbild 8051	18
Beschaltung MC-Board (BubMini)	19
Besonderheiten des C-Compilers von Keil	20

ISP (In System Programmable)



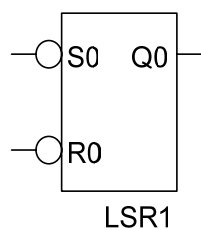
DesignExpert V8.1® - Schematic – Blockschaltbilder

Schaltnetze



Schaltwerke

SR-FF (statisch)



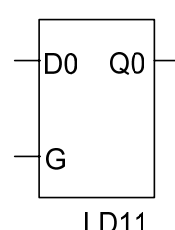
"lowaktiv"

$$\begin{aligned} Q.S0 &= !S \\ Q.R0 &= !R \end{aligned}$$

!S	!R	Q^{n+1}
0	0	Q^n
1	0	1
0	1	0
1	1	.*

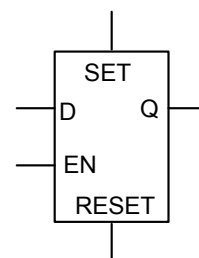
* nicht definiert

D-FF (statisch)



$$\begin{aligned} Q.D &= \text{Dat} \\ Q.G &= \text{EN} \end{aligned}$$

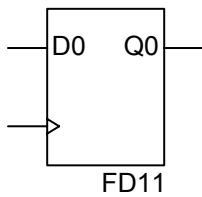
EN	Dat	Q^{n+1}
0	X	Q^n
1	0	0
1	1	1



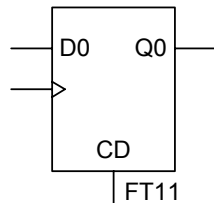
D-Latch mit asynchronen Setz- und Rücksetzeingängen

equations

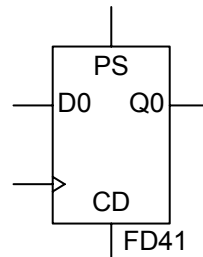
$$\begin{aligned} Q.D &= D_0 \\ Q.LH &= G \\ Q.ASET &= S \\ Q.ACLR &= R \end{aligned}$$



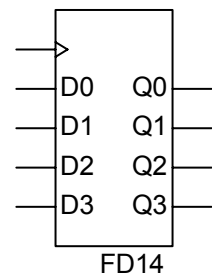
D-FF mit dynamischem Takteingang (pos. Flanke)



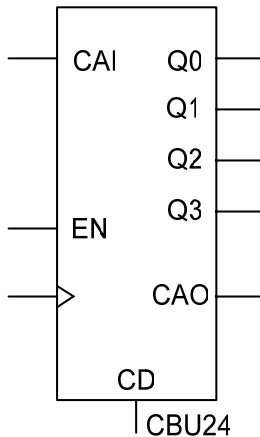
... zusätzlichem asynchronem Reset (ClearDevice)



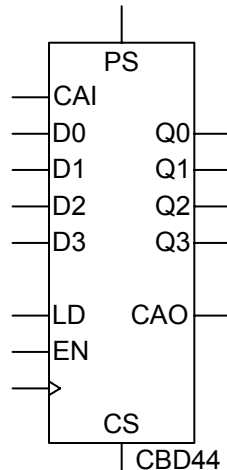
... zusätzlichem synchronem Setzen (PreSet)



4 - D-FF (4-Bit-Register)



Vorwärtzähler mit CarryIn und CarryOut



CBD44

- Rückwärtszähler (CounterBinaryDown)
- Ein- und Ausgangsübertrag **CAI** / **CAO** (kaskadierbar)
- Enable (**EN**)
- Load (**LD**) - beliebiger Startwert möglich
- **CS** (synchrones Rücksetzen / clear)
- **PS** (synchrones Setzen)

Input							Output	
PS	CS	LD	D	EN	CAI	CLK	Q	CAO
1	x	x	x	x	x	↑	1	0
0	1	x	x	x	x	↑	0	²⁾
0	0	1	d	x	x	↑	d	³⁾
0	0	0	x	0	x	x	Q	0
0	0	0	x	x	0	x	Q	0
0	0	0	x	1	1	↑	1	⁴⁾

- 1) count down
 2) CAO = CAI & EN
 3) CAO = CAI & EN & !D0 & !D1 & !D2 & !D3
 4) CAO = 1 nach dem letzten Zählschritt

ABEL-Syntax

The Header Section

- **Module** (required)
- Interface (lower level, optional)
- Title

The Declarations Section

- Declarations Keyword
- Device Declaration
- Hierarchy Declarations
- **Signal** Declarations
- **Constant** Declarations
- Symbolic State Declarations
- Macro Declarations
- Library Declarations

The Logic Description Section

- Dot Extensions
- **Equations**
- **Truth Tables**
- **State Descriptions**
- Fuses Declarations
- XOR Factors

The Test Vectors Section

- Test Vectors
- Trace Statement

The End Statement

- Keyword: **end**

Beispiel

```

module szd07vr;

Takt      pin;
u, QC, QB, QA  pin istype 'com,dc';

equations
  QC.Clk = Takt;
  QB.Clk = Takt;
  QA.Clk = Takt;

truth table
  ([u, QC, QB, QA] => [QC, QB, QA])

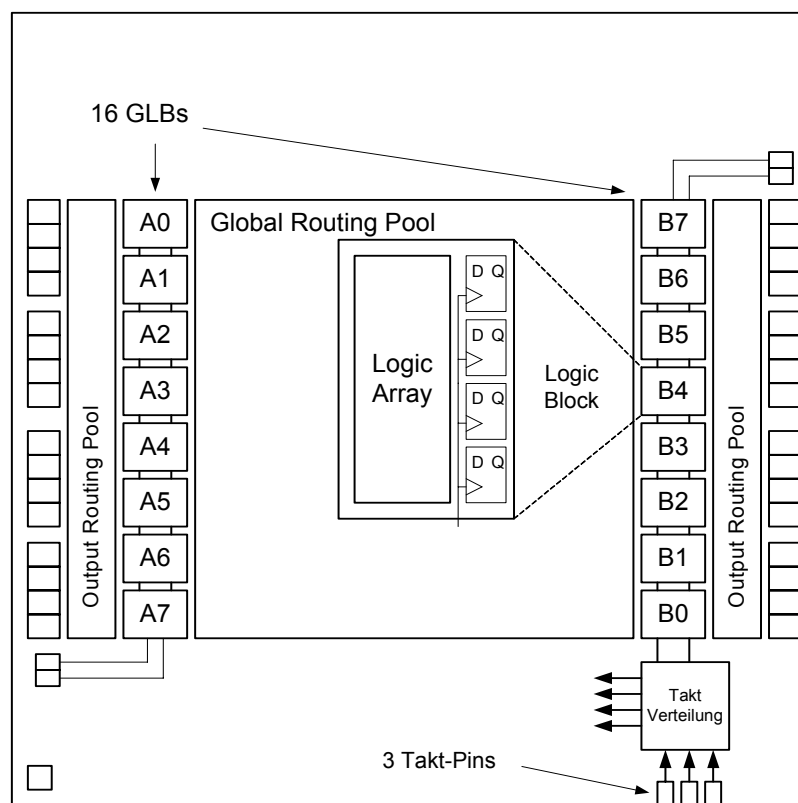
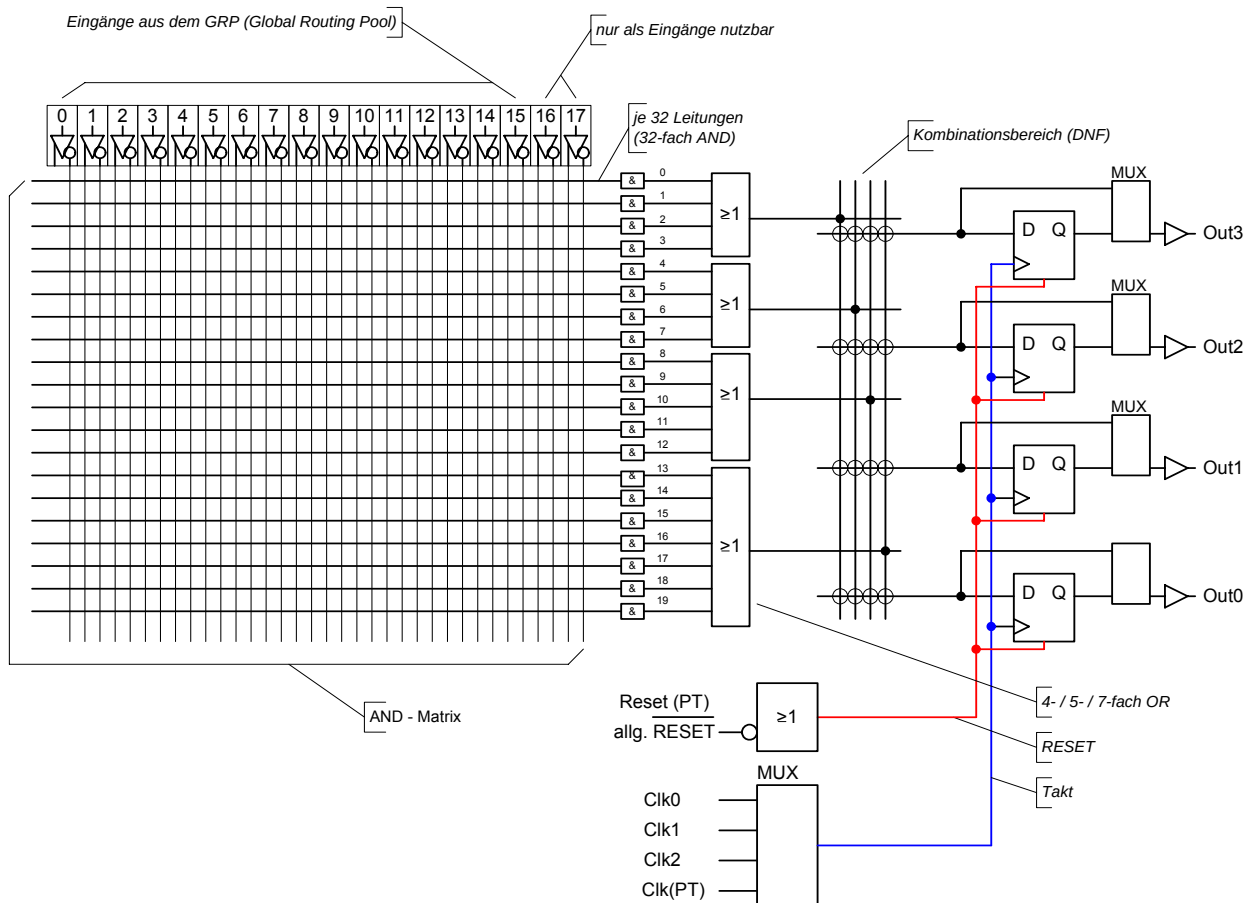
  [0, 0, 0, 0] => [0, 0, 1];
  [0, 0, 0, 1] => [0, 1, 0];
  [0, 0, 1, 0] => [0, 1, 1];
  [0, 0, 1, 1] => [1, 0, 0];
  [0, 1, 0, 0] => [1, 0, 1];
  [0, 1, 0, 1] => [1, 1, 0];
  [0, 1, 1, 0] => [1, 1, 1];
  [0, 1, 1, 1] => [0, 0, 0];

  [1, 0, 0, 0] => [1, 1, 1];
  [1, 0, 0, 1] => [0, 0, 0];
  [1, 0, 1, 0] => [0, 0, 1];
  [1, 0, 1, 1] => [0, 1, 0];
  [1, 1, 0, 0] => [0, 1, 1];
  [1, 1, 0, 1] => [1, 0, 0];
  [1, 1, 1, 0] => [1, 0, 1];
  [1, 1, 1, 1] => [1, 1, 0];

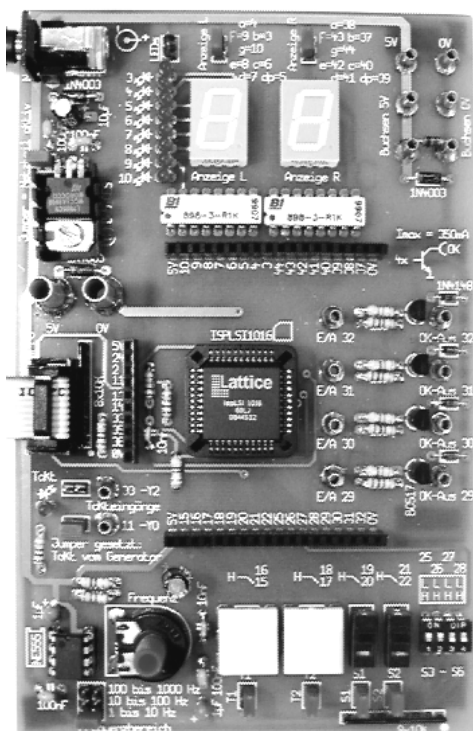
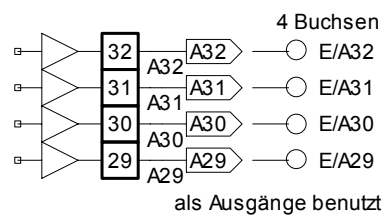
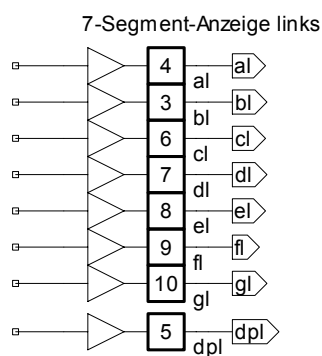
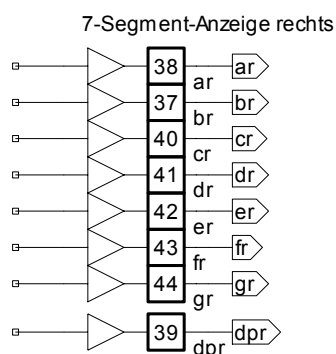
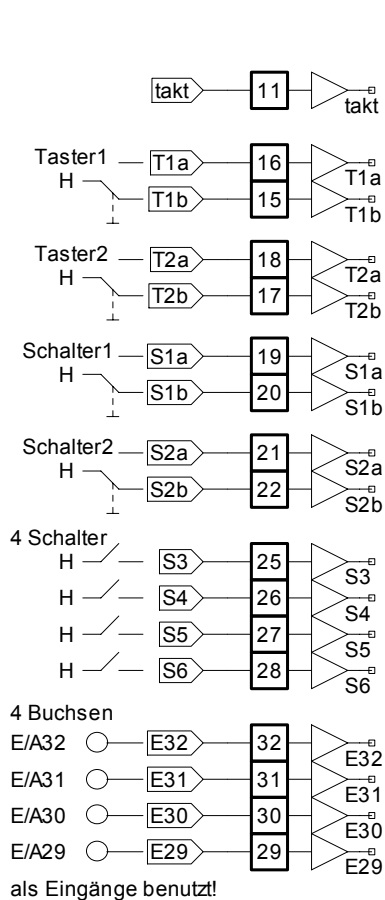
end

```

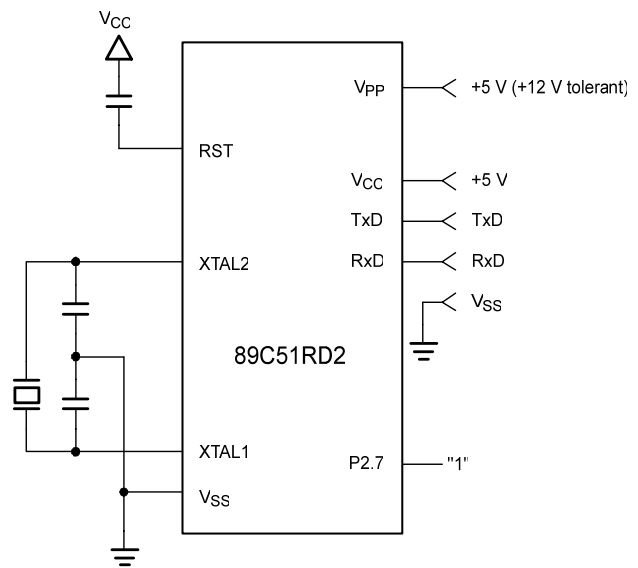
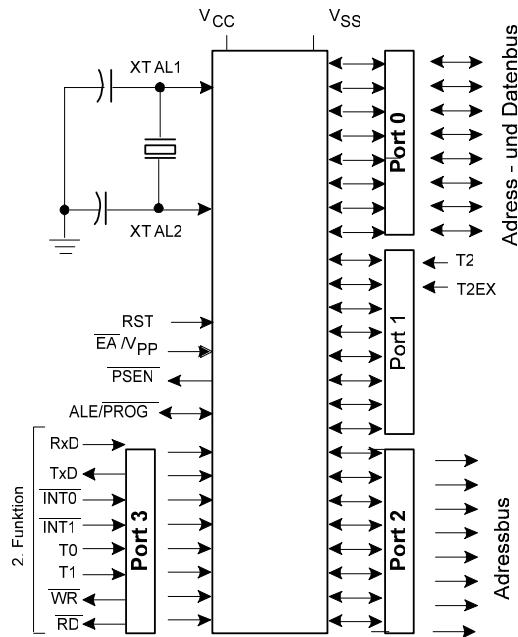

Ausgangs-Makrozelle



Pinbelegung des 1016-Exp.Board



8051 - Controller



In-System Programming with a Minimum of Pins

Befehlssatz

Operand	Bedeutung
A	Akkumulator (E0)
dadr	8 Bit – Adresse im internen RAM oder im SFR-Ber.
adr11	11 Bit - Adresse
adr16	16 Bit - Adresse
AC	Auxiliary Carry (Hilfsübertrag-Flag) (D6)
B	Register B
badr	Bitadresse im internen RAM (20-2F) oder im SFR-Bereich(80-FF)
/badr	Invertierter Inhalt der Bitadresse (Komplement)
CY	Carry-Flag (D7)
#c8	8 Bit - Konstante
#c16	16 Bit - Konstante
D	Kennzeichnung für ein 4 Bit - Digit (Nibble)
DPTR	Datenpointerregister
HB	Highbyte eines Datenwortes
I	Interrupt
LB	Lowbyte eines Datenwortes
LSB	Bit 0 eines Bytes
MSB	Bit 7 eines Bytes
MZ	Maschinenzyklen
OV	Overflow (Überlauf)- Flag (D2)
P	Port
PC	Programmzähler
PSW	Programmstatuswort
rel	Signiertes 8 Bit - Offset für Sprungbefehle
@Ri	Adressregister für internes und externes RAM
Rn	Register 0 bis 7 der aktuellen Registerbank
SFR	Spezialfunktionsregister
SP	Stackpointer
T	Timer
TF	Timer - Flag
■	Flag wird beeinflusst
--	Flag wird nicht beeinflusst
WB	Wortbreite des Befehls in Bytes

Datentransport

Mnemonischer Befehl		Hex-Code	W B	M Z	Beeinflussung Zustandsbits			Befehlsbeschreibung
OpCode	Operanden				CY	OV	AC	
MOV	A,#c8	74	2	1	--	--	--	Akku direkt mit Konstante laden
MOV	Rn,#c8	78-7F	2	1	--	--	--	Direktes Laden des Registers mit einer Konstanten
MOV	dadr,#c8	75	3	2	--	--	--	Internen Speicher mit Konstante laden
MOV	A,Rn	E8-EF	1	1	--	--	--	Kopieren Registerinhalt in den Akku
MOV	Rn,A	F8-FF	1	1	--	--	--	Kopieren des Akkuinhaltes in ein Register
MOV	A,dadr	E5	2	1	--	--	--	Inhalt interner Speicherplätze in den Akku kopieren
MOV	dadr,A	F5	2	1	--	--	--	Inhalt Akku in einen internen Speicherplatz kopieren
MOV	Rn,dadr	A8-AF	2	2	--	--	--	internen Speicherplatz in ein Register kopieren
MOV	dadr,Rn	88-8F	2	2	--	--	--	Registerinhalt in internen Speicherplatz kopieren
MOV	dadr,dadr	85	3	2	--	--	--	Inhalt interner Speicherplatz in einen anderen kopieren
MOV	A,@R0	E6	1	1	--	--	--	Speicherinhalt des internen RAM in den Akku kopieren (R0 bzw. R1 enthält die Quellenadresse)
MOV	A,@R1	E7	1	1	--	--	--	
MOV	@R0,A	F6	1	1	--	--	--	Akkuinhalt in Speicherplatz des internen RAM kopieren (R0 bzw. R1 enthält die Zieladresse)
MOV	@R1,A	F7	1	1	--	--	--	
MOV	dadr,@R0	86	2	2	--	--	--	Inhalt eines internen Speicherplatz in einen anderen kopieren (R0 bzw. R1 enthält die Quellenadresse)
MOV	dadr,@R1	87	2	2	--	--	--	
MOV	@R0,#c8	76	2	1	--	--	--	Konstante in internen RAM laden R0 bzw. R1 bestimmen die Ziel Adresse
MOV	@R1,#c8	77	2	1	--	--	--	
MOV	@R0,dadr	A6	2	2	--	--	--	Inhalt interner Speicherplatz in einen anderen kopieren (R0 bzw. R1 enthält die Zieladresse)
MOV	@R1,dadr	A7	2	2	--	--	--	
POP	dadr	D0	2	2	--	--	--	Speicherinhalt vom Stack holen
PUSH	dadr	C0	2	2	--	--	--	Speicherinhalt auf den Stack schreiben
NOP		00	1	1	--	--	--	Keine Aktivität
MOV	badr,C	92	2	2	--	--	--	Carry-Inhalt in angegebene Bitadresse kopieren
MOV	C,badr	A2	2	1	■	--	--	Der Inhalt Bitadresse in das Carry kopieren
MOV	DPTR,#c16	90	3	2	--	--	--	16Bit-Konstante in Datenpointer laden
XCH	A,Rn	C8-CF	1	1	--	--	--	Akku- und Registerinhalt austauschen
XCH	A,dadr	C5	2	1	--	--	--	Internen Speicher mit dem Akkuinhalt tauschen
XCH	A,@R0	C6	1	1	--	--	--	Inhalt interner Speicherplatzes Akku austauschen (R0 bzw. R1 enthält die Zieladresse)
XCH	A,@R1	C7	1	1	--	--	--	
XCHD	A,@R0	D6	1	1	--	--	--	Das LOW-Nibble eines Speicherplatzes im internen RAM gegen das LOW-Nibble des Akkus austauschen. Die HIGH-Nibble beider Speicher werden nicht verändert. (R0 bzw. R1 enthält die Zieladresse)
XCHD	A,@R1	D7	1	1	--	--	--	
MOVX	A,@R0	E2	1	2	--	--	--	Inhalt eines externen Speicherplatzes in den Akku kopieren
MOVX	A,@R1	E3	1	2	--	--	--	
MOVX	@R0,A	F2	1	2	--	--	--	Inhalt des Akkus in einen externen Speicherplatz kopieren.
MOVX	@R1,A	F3	1	2	--	--	--	
MOVX	A,@DPTR	E0	1	2	--	--	--	Inhalt eines externen Speicherplatzes in den Akku kopieren
MOVX	@DPTR,A	F0	1	2	--	--	--	Inhalt des Akkus in einen externen Speicherplatz kopieren
MOVC	A,@A+DPTR	93	1	2	--	--	--	Hole Konstante aus einer Tabelle im EEPROM.
MOVC	A,@A+PC	83	1	2	--	--	--	Hole Konstante aus einer Tabelle im EEPROM.

Arithmetische Operationen

Mnemonicischer Befehl		Hex-Code	W B	M Z	Beeinflussung Zustandsbits			Befehlsbeschreibung
OpCode	Operanden				CY	OV	AC	
CLR	A	E4	1	1	--	--	--	Löschen des Akku-Inhaltes
CPL	A	F4	1	1	--	--	--	Komplementieren des Akku-Inhaltes
INC	A	04	1	1	--	--	--	Inhalt des Akku um „1“ erhöhen
INC	Rn	08-0F	1	1	--	--	--	Inhalt des Registers um „1“ erhöhen
INC	dadr	05	2	1	--	--	--	Inhalt intern. Speicherstelle um „1“ erhöhen
INC	DPTR	A3	1	2	--	--	--	Inhalt des Datenpointers um „1“ erhöhen
INC	@R0	06	1	1	--	--	--	Inhalt einer Speicherstelle im internen
INC	@R1	07	1	1	--	--	--	RAM um „1“ erhöhen
DEC	A	14	1	1	--	--	--	Inhalt des Akku um „1“ vermindern
DEC	Rn	18-1F	1	1	--	--	--	Inhalt des Registers um „1“ vermindern
DEC	dadr	15	2	1	--	--	--	Inhalt interne Speicherst. um „1“ vermindern
DEC	@R0	16	1	1	--	--	--	Inhalt int. Speicherstelle um „1“ vermindern
DEC	@R1	17	1	1	--	--	--	Inhalt int. Speicherstelle um „1“ vermindern
ADD	A,#c8	24	2	1	■	■	■	Addition einer Konstante zum Akkuinhalt
ADDC	A,#c8	34	2	1	■	■	■	Addition einer Konstante plus Carry
ADD	A,Rn	28-2F	1	1	■	■	■	Addition eines Registerinhaltes zum Akkuinhalt
ADDC	A,Rn	38-3F	1	1	■	■	■	Add. eines Registerinh. plus Übertrag zum Akkuinhalt
ADD	A,dadr	25	2	1	■	■	■	Inhalt int. Speicherstelle zum Akkus addieren
ADDC	A,dadr	35	2	1	■	■	■	Inhalt int. Speicherstelle plus CY zum Akku addieren
ADD	A,@R0	26	1	1	■	■	■	Inhalt einer Speicherstelle im internen
ADD	A,@R1	27	1	1	■	■	■	RAM zum Inhalt des Akkus addieren
ADDC	A,@R0	36	1	1	■	■	■	Inhalt einer Speicherstelle im internen RAM
ADDC	A,@R1	37	1	1	■	■	■	plus CY zum Akku addieren
DA	A	D4	1	1	■	--	--	Dezimalkorrektur des Akku <i>nur</i> nach einer BCD-Addition
SUBB	A,#c8	94	2	1	■	■	■	Subtraktion Konstante plus Carry vom Akku
SUBB	A,dadr	95	2	1	■	■	■	Subtrakt. Int. Speicherinhalt plus Carry vom Akku
SUBB	A,Rn	98-9F	1	1	■	■	■	Subtrakt. eines Registers plus Carry vom Akku
SUBB	A,@R0	96	1	1	■	■	■	Subtraktion eines Speicherinhaltes des
SUBB	A,@R1	97	1	1	■	■	■	internen RAM plus Carry vom Akkuinhalt
SWAP	A	C4	1	1	--	--	--	Vertausche die Nibbles des Akkus
MUL	AB	A4	1	4	■	■	--	Multipliziere den Akku B- Register
DIV	AB	84	1	4	■	■	--	Teile Akkuinhalt durch den B-Registerinhalt
RL	A	23	1	1	--	--	--	Rotiere Akku-Inhalt eine Stelle nach links
RLC	A	33	1	1	■	--	--	Rotiere Akku-Inhalt durch Carry nach links
RR	A	03	1	1	--	--	--	Rotiere Akku-Inhalt eine Stelle nach rechts
RRC	A	13	1	1	■	--	--	Rotiere Akku-Inhalt durch Carry nach rechts
SETB	C	D3	1	1	■	--	--	Setze das CY-Bit auf „1“
CLR	C	C3	1	1	■	--	--	Setze das CY-Bit auf „0“
CPL	C	B3	1	1	■	--	--	Komplementiere das CY-Bit
SETB	badr	D2	2	1	--	--	--	Setze das adressierte Bit auf „1“
CLR	badr	C2	2	1	--	--	--	Setze das adressierte Bit auf „0“
CPL	badr	B2	2	1	--	--	--	Komplementiere das adressierte Bit

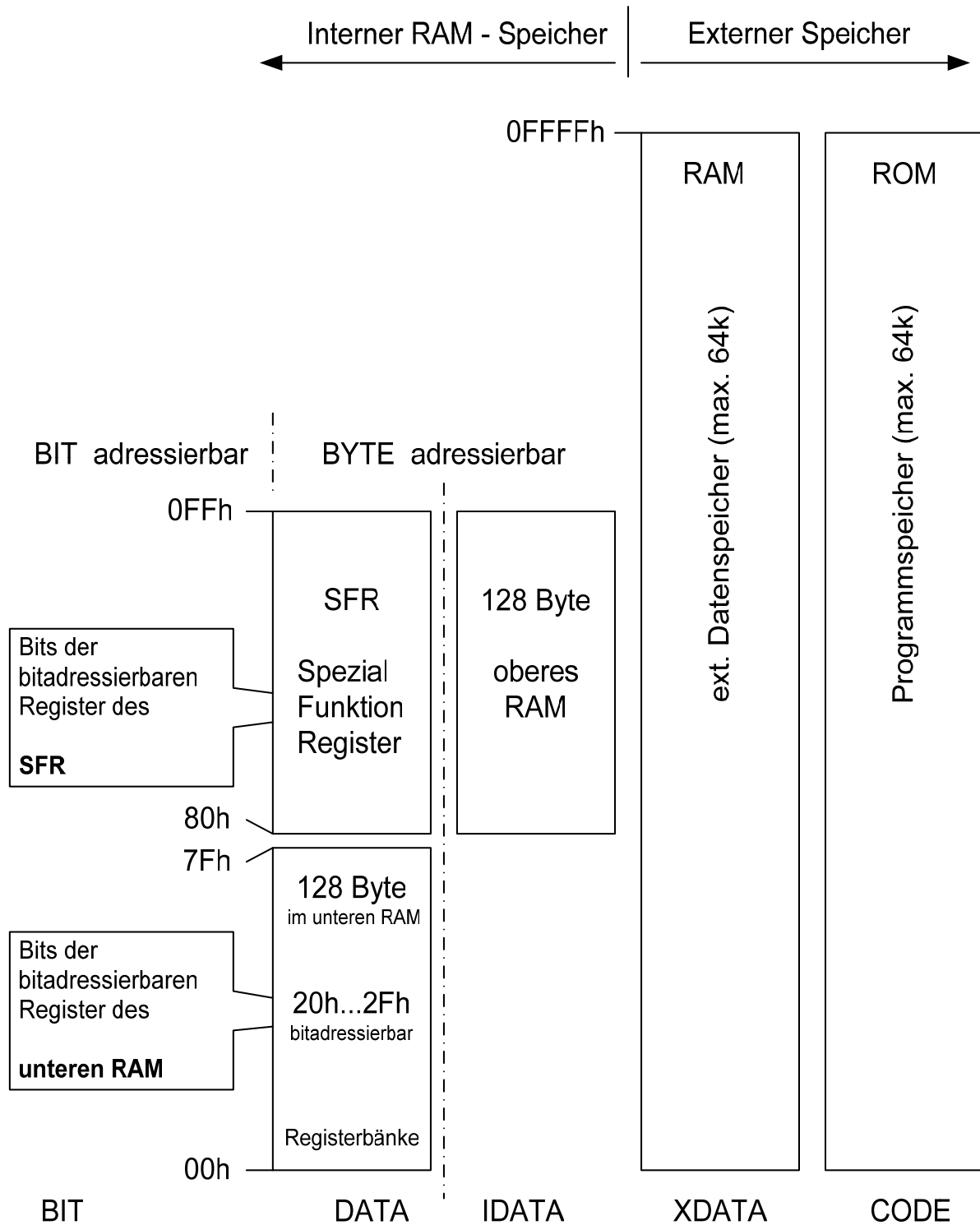
Logische Operationen

ANL	A,#c8	54	2	1	--	--	--	Bitweise UND-Verknüpfung Konstante und Akku
ANL	A,Rn	58-5F	1	1	--	--	--	Bitweise UND-Verknüpfung zwischen Akku und Register. <i>Ergebnis im Akku.</i>
ANL	A,dadr	55	2	1	--	--	--	Bitweise UND-Verknüpfung zwischen Akku inter. Speicherst., <i>Ergebnis im Akku</i>
ANL	dadr,#c8	53	3	2	--	--	--	Bitweise UND-Verkn. Konstante und int. Speicherst., das Ergebnis steht in der Speicherst.
ANL	dadr,A	52	2	1	--	--	--	Bitweise UND-Verknüpfung zwischen Akku und RAM-internem Speicher, <i>Ergebnis im Speicher</i>
ANL	C,badr	82	2	2	■	--	--	UND-Verknüpfung zwischen Carry und Bit
ANL	C,/badr	B0	2	2	■	--	--	UND-Verknüpfung zwischen Carry und invert. Bit. <i>Ergebnis jeweils im Carry-Bit</i>
ANL	A,@R0	56	1	1	--	--	--	Bitweise UND-Verknüpfung
ANL	A,@R1	57	1	1	--	--	--	Bitweise UND-Verknüpfung
ORL	A,#c8	44	2	1	--	--	--	Bitweise ODER-Verknüpfung Akku und Konstante
ORL	dadr,#c8	43	3	2	--	--	--	Bitweise ODER-Verknüpfung Konstante und int. Speicherst. <i>Das Ergebnis steht in der Speicherstelle</i>
ORL	A,Rn	48-4F	1	1	--	--	--	Bitweise ODER-Verknüpfung zwischen Akku und Register. <i>Ergebnis im Akku.</i>
ORL	A,dadr	45	2	1	--	--	--	Bitweise ODER-Verknüpfung zwischen Akku und RAM-internem Speicher, <i>Ergebnis im Akku</i>
ORL	dadr,A	42	2	1	--	--	--	Bitweise ODER-Verknüpfung Akku und int. Speicherst., <i>Ergebnis im Speicher</i>
ORL	C,badr	72	2	2	■	--	--	ODER-Verknüpfung zwischen Carry Bit
ORL	C,/badr	A0	2	2	■	--	--	ODER-Verknüpfung zwischen Carry invertiertem Bit. <i>Ergebnis jeweils im Carry-Bit.</i>
ORL	A,@R0	46	1	1	--	--	--	Bitweise ODER-Verknüpfung
ORL	A,@R1	47	1	1	--	--	--	Bitweise ODER-Verknüpfung
XRL	A,#c8	64	2	1	--	--	--	Bitweise EXKLUSIV-ODER-Verknüpfung
XRL	dadr,#c8	63	3	2	--	--	--	Bitweise EXKLUSIV-ODER-Verknüpfung Konst. und int. Speicherst. <i>Ergebnis in Speicherstelle.</i>
XRL	A,Rn	68-6F	1	1	--	--	--	Bitweise EXKLUSIV-ODER-Verknüpfung Akku und angeg. Register. <i>Ergebnis im Akku.</i>
XRL	A,dadr	65	2	1	--	--	--	Bitweise EXKLUSIV-ODER-Verknüpfung zw. Akku und RAM-internem Speicher, <i>Ergebnis im Akku</i>
XRL	dadr,A	62	2	1	--	--	--	Bitweise EXKLUSIV-ODER-Verknüpfung zw. Akku und RAM-internem Speicher, <i>Ergebnis im Speicher</i>
XRL	A,@R0	66	1	1	--	--	--	Bitweise EXKLUSIV-ODER-Verknüpfung
XRL	A,@R1	67	1	1	--	--	--	Bitweise EXKLUSIV-ODER-Verknüpfung

Sprungbefehle

Mnemonicischer Befehl		Hex-Code	W B	M Z	Beeinflussung Zustandsbits			Befehlsbeschreibung
OpCode	Operanden				CY	OV	AC	
LJMP	adr16	02	3	2	--	--	--	Programmsprung im 64K-Block
SJMP	rel	80	2	2	--	--	--	relativer Programmsprung im Bereich -128 bis +127 zur nachfolgenden Befehlsadresse
AJMP	adr11	01-E1	2	2	--	--	--	Sprung im 2k-Block
JMP	@A+DPTR	73	1	2	--	--	--	Springe zur Adresse, die aus Akku- und Datenpointerinhalt gebildet wird.
JBC	badr,rel	10	3	2	--	--	--	Springe bei <i>gesetztem</i> Bit und lösche es
JB	badr,rel	20	3	2	--	--	--	Springe bei <i>gesetztem</i> Bit
JNB	badr,rel	30	3	2	--	--	--	Springe bei <i>gelöschtem</i> Bit
JC	rel	40	2	2	--	--	--	Springe bei <i>gesetztem</i> Carry-Bit
JNC	rel	50	2	2	--	--	--	Springe bei <i>gelöschtem</i> Carry-Bit
JZ	rel	60	2	2	--	--	--	Springe, wenn Akkuinhalt <i>gleich Null</i>
JNZ	rel	70	2	2	--	--	--	Springe, wenn Akkuinhalt <i>ungleich Null</i>
DJNZ	Rn,rel	D8-DF	2	2	--	--	--	Vermindere Register um Eins und springe, wenn der Rest ungleich Null
DJNZ	dadr,rel	D5	3	2	--	--	--	Vermindere den Speicherinhalt im internen RAM um Eins und springe, wenn der Rest ungleich Null.
CJNE	A,#c8,rel	B4	3	2	■	--	--	Vergleiche Akku mit Konstante und verzweige bei Ungleichheit. Andernfalls fahre im Programm fort.
CJNE	Rn,#c8,rel	B8-BF	3	2	■	--	--	Vergleiche Register mit Konstante und verzweige bei Ungleichheit, andernfalls fahre fort.
CJNE	A,dadr,rel	B5	3	2	■	--	--	Vergleiche Akku- und Speicherinhalt und verzweige bei Ungleichheit.
CJNE	@R0,#c8,rel	B6	3	2	■	--	--	Vergleiche den Inhalt des RAM-internen Speichers mit der Konstante und verzweige bei Ungleichheit. (R0 bzw. R1 enthält die Quellenadresse)
CJNE	@R1,#c8,rel	B7	3	2	■	--	--	
LCALL	Adr16	12	3	2	--	--	--	Unterprogrammaufruf im 64k-Block
ACALL	adr11	11-F1	2	2	--	--	--	Unterprogrammaufruf im 2k-Block
RET		22	1	2	--	--	--	Ende Unterprogramm
RETI		32	1	2	--	--	--	Ende UP plus löschen des INT-Flags

Speicheraufteilung



Internes RAM

[Byte - Adresse]		[Byte - Name]		[Bit - Adresse]		[Bit - Name]			
F0	B	F7	F6	F5	F4	F3	F2	F1	F0
		B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0

E0	ACC	E7	E6	E5	E4	E3	E2	E1	E0
		ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0

D0	PSW	D7	D6	D5	D4	D3	D2	D1	D0
		CY	AC		RS1	RS0	OV		P

B0	P3	B7	B6	B5	B4	B3	B2	B1	B0
		P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0
				T1	T0	IT1	IT0		

A0	P2	A7	A6	A5	A4	A3	A2	A1	A0
		P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0

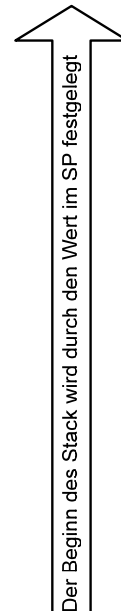
90	P1	97	96	95	94	93	92	91	90
		P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0

8D	TH1	nicht bitadressierbar							
8C	TL1								
8B	TH0								
8A	TL0								
89	TMOD								
88	TCON	8F	8E	8D	8C	8B	8A	89	88
		TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
87	PCON								

83	DPH	DPTR (nicht bitadressierbar)							
82	DPL								
81	SP								
80	P0	87	86	85	84	83	82	81	80
		P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0

		[Byte -Adresse							
		[Bit -Adresse							
2F	7F	7E	7D	7C	7B	7A	79	78	
2E	77	76	75	74	73	72	71	70	
2D	6F								
2C	67								
2B	5F								
2A	57								
29	4F								
28	47								
27	3F								
26	37								
25	2F								
24	27								
23	1F								
22	17	16	15	14	13	12	11	10	
21	0F	0E	0D	0C	0B	0A	09	08	
20	07	06	05	04	03	02	01	00	

1F	R7	Registerbank 3 (RS1 = 1 / RS0 = 1)
:	:	
18	R0	
17	R7	Registerbank 2 (RS1 = 1 / RS0 = 0)
:	:	
10	R0	
0F	R7	Registerbank 1 (RS1 = 0 / RS0 = 1)
0E	R6	
0D	R5	
0C	R4	
0B	R3	
0A	R2	
09	R1	
08	R0	
07	R7	Registerbank 0 (nach dem RESET eingeschaltet)
06	R6	
05	R5	
04	R4	
03	R3	
02	R2	
01	R1	
00	R0	



81 SP [nach RESET mit 07h geladen]

[Name]

[Adresse]

[es kann nur eine Registerbank aktiv sein RS1 und RS0 bestimmen]

Interrupt & Timer

Abfallende Flanken oder Low-Signale können externe Interruptereignisse auslösen, überlaufende Timer lösen Timer-Interrupts aus, über die serielle Schnittstelle gesendete oder empfangene Zeichen können Interrupts auslösen.

Die Interrupt-Anforderungs-Bits (s.u.) werden immer gesetzt, der entsprechende Interrupt wird jedoch nur dann bearbeitet, wenn die Interrupts freigegeben sind und die globale Interrupt-Freigabe EA = 1 ist. Zur Bearbeitung der Interrupts springt der Controller zu den festgelegten Adressen (s.u.).

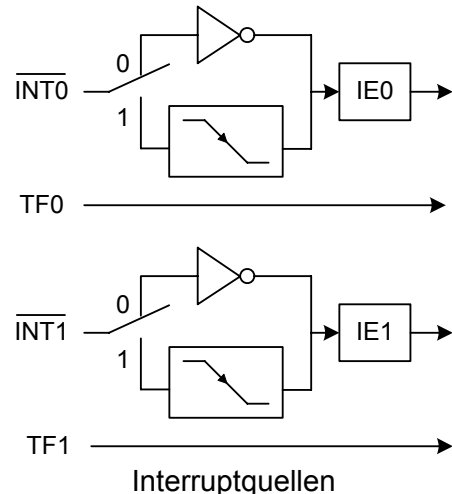
Interrupt Enable (IE) Register

EA(L)	--	--	ES	ET1	EX1	ET0	EX0
-------	----	----	----	-----	-----	-----	-----

Enable Bit = 1 (enables the interrupt)

Enable Bit = 0 (disables the interrupt)

Symbol	Position	Function
EA(L)	IE.7	EA = 1 (generelle Freigabe)
ES	IE.4	Serielle Schnittstelle
ET1	IE.3	Timer 1 (Überlauf-Interrupt Freigabe)
EX1	IE.2	Externer Interrupt 1 (Freigabe)
ET0	IE.1	Timer 0 (Überlauf-Interrupt Freigabe)
EX0	IE.0	Externer Interrupt 0 (Freigabe)



IP Interrupt-Prioritäten-Register, bitadressierbar, 0B8h								
Bit	7	6	5	4	3	2	1	0
	-	-	-	PS	PT1	PX1	PT0	PX0
Bit-Adr.	0BFh	0BEh	0BDh	0BCh	0BBh	0BAh	0B9h	0B8h
Priorität von				Serial Port	Timer1	Ext.Interrupt1	Timer0	Ext.Interrupt0

0: niedrige Priorität 1: höhere Priorität

Interrupts können nur von anderen Interrupts mit höherer Ebene unterbrochen werden. Treten 2 Interrupts gleicher Priorität gleichzeitig auf, so werden sie in folgender Reihenfolge bearbeitet:

ExtInt0 → Timer0 → ExtInt1 → Timer1

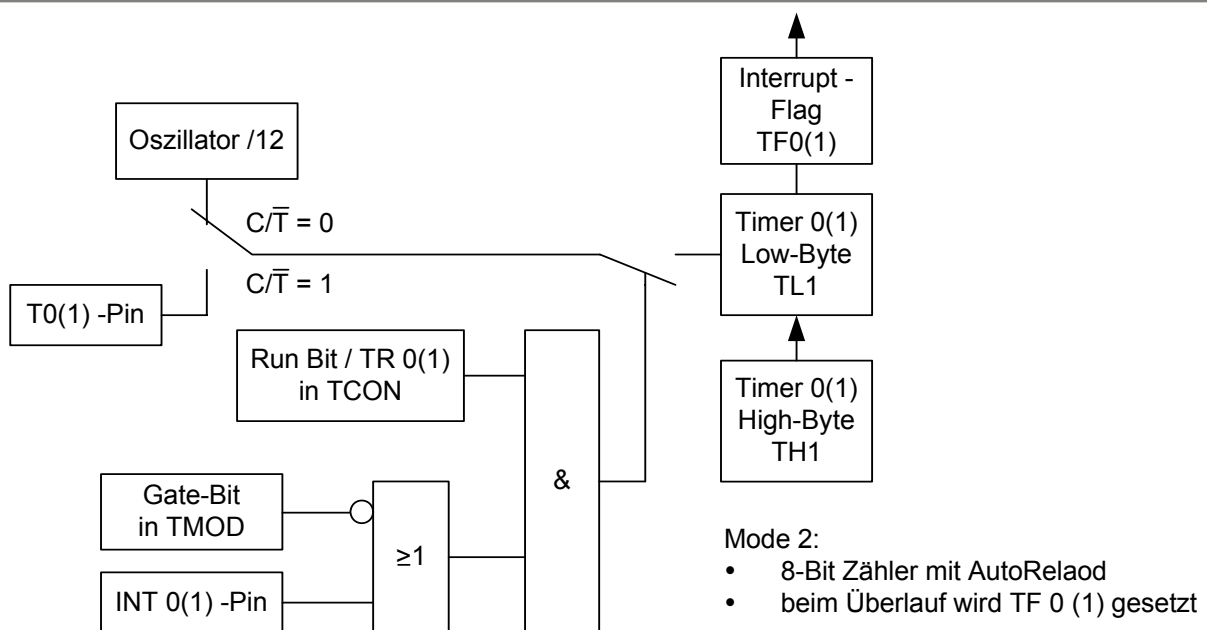
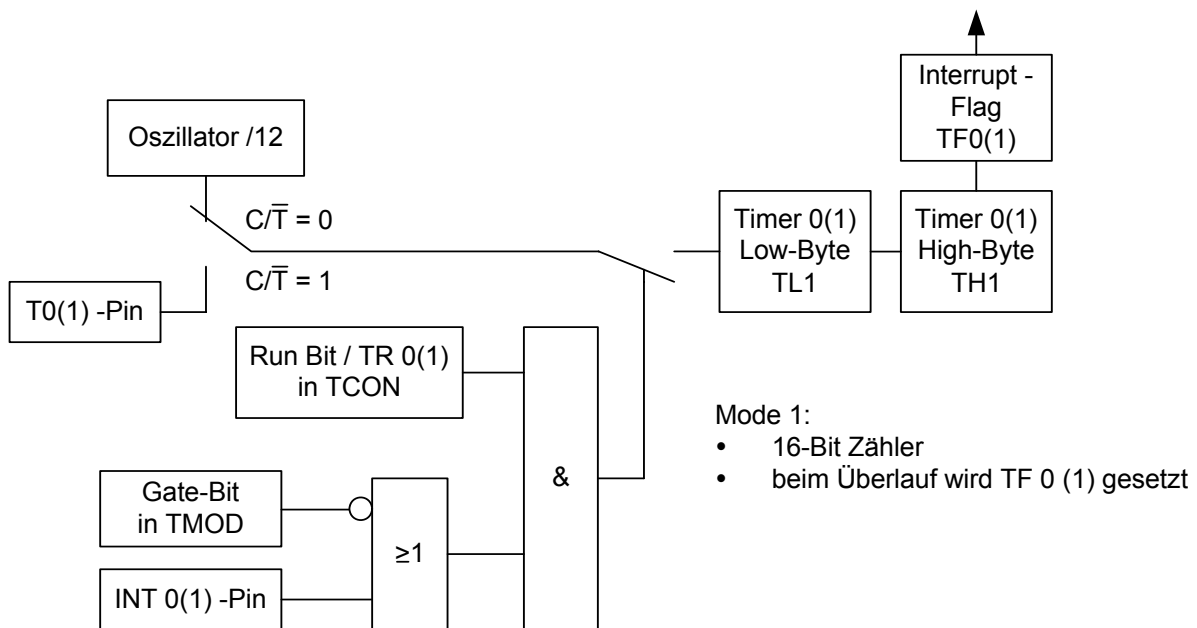
Interrupt	Einsprungsadresse	Interrupt-Anforderungs-Bit
Externer Interrupt 0	0003h	IE0
Timer0 - Überlauf	000Bh	TF0
Externer Interrupt 1	0013h	IE1
Timer1 – Überlauf	001Bh	TF1
Serieller Schnittstellen-Interrupt	0023h	RI oder TI

Externe Interrupts beim 8051

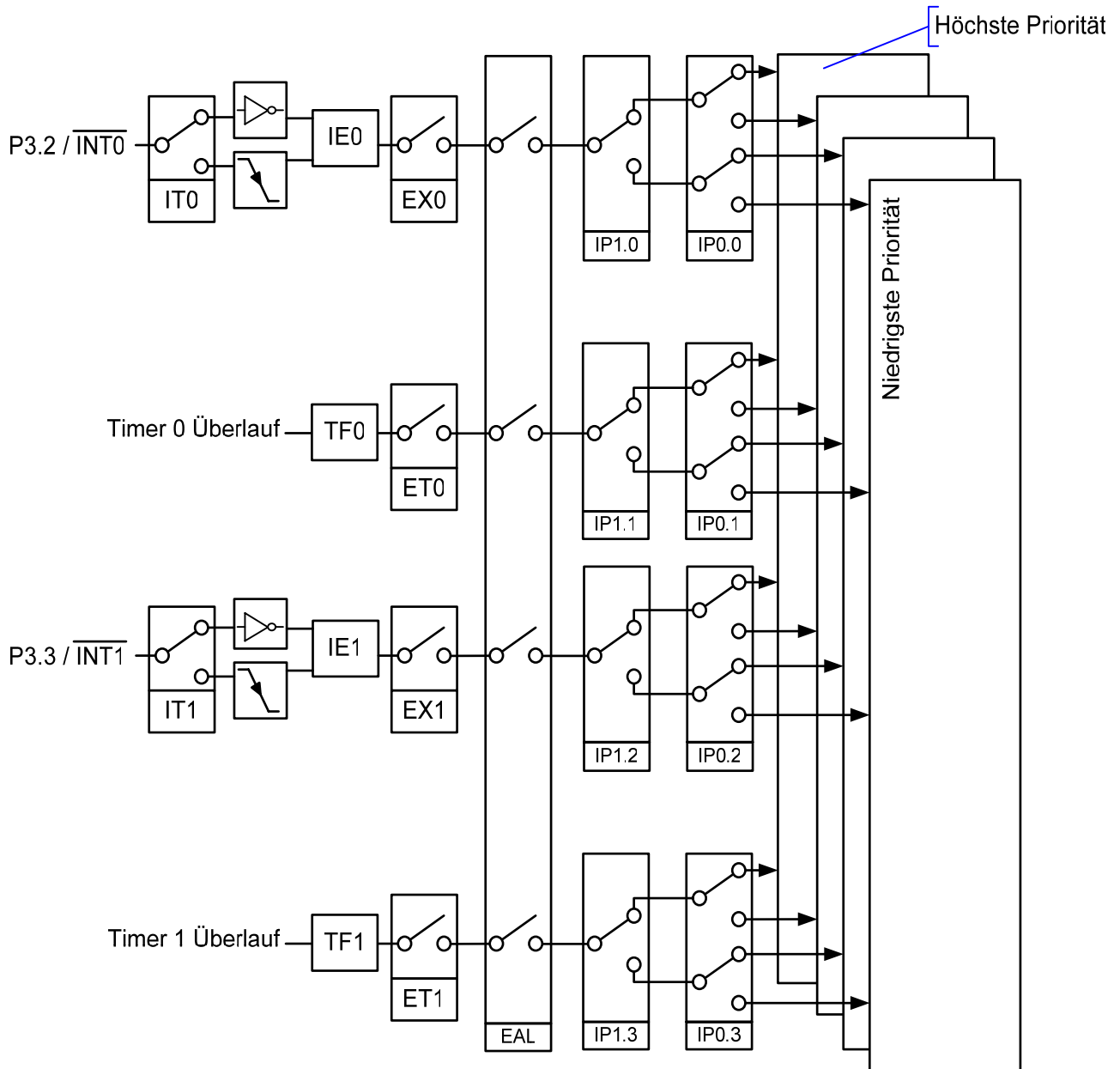
Mit den Portpins P3.2 und P3.3, können externe Interrupts ausgelöst werden.

Die Interrupts werden nur ausgelöst, wenn sie durch Setzen der Bits EX0 bzw. EX1 freigegeben sind.

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	/INT0 (external interrupt 0)
P3.3	/INT1 (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	/WR (external data memory write strobe)
P3.7	/RD (external data memory read strobe)



Interruptstruktur des 8051



IEN0 (0A8h): Interrupt Enableregister							
0AFh	0AEh	0ADh	0ACh	0ABh	0AAh	0A9h	0A8h
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EA(L)	-	-	-	ET1	EX1	ET0	EX0

Register zur Zählerkontrolle: TMOD und TCON:

TMOD (89h) : Timermodus-Kontrollregister für Timer1 & Timer0							
Kontrolle Timer 1				Kontrolle Timer 0			
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Gate	C/T	M1	M0	Gate	C/T	M1	M0

Gate	C/T	M1	M0	
		0	0	Modus 0
		0	1	Modus 1: 16 Bit-Timer ohne Nachladen
		1	0	Modus 2: 8-Bit-Timer mit Auto-Reload
		1	1	Modus 3: 2 Stück 8-Bit-Timer
	0	Timer-Betrieb		
	1	Zähler-Betrieb		
	0	Timer nur durch TR-Bit ein- und ausschalten		
1	Timer mit TR-Bit und Portpin ein- und ausschalten			

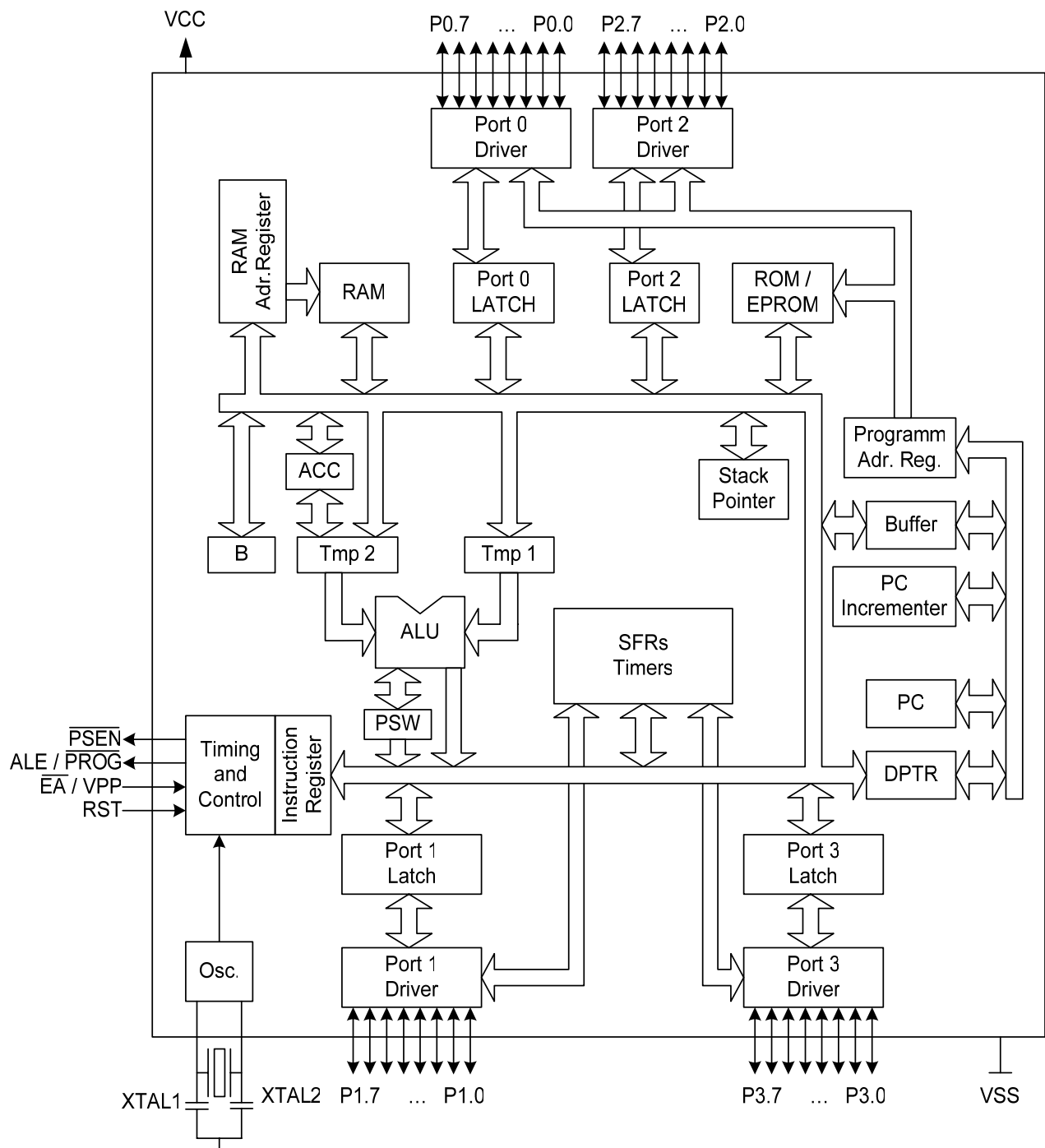
TCON (88h): Timer-Kontrollregister für Timer1 & 0 / ext Interrupt 1 & 0							
Kontrolle Timer 1 und 0				ext. Interrupt-Kontrolle			
8Fh	8Eh	8Dh	8Ch	8Bh	8Ah	89h	88h
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

IE _x	IT _x
0, 1	0 → Interrupt bei Lowpegel
	1 → Interrupt bei abfallender Flanke
wird von CPU gesetzt wenn Flanke externer Interrupt1 erkannt	

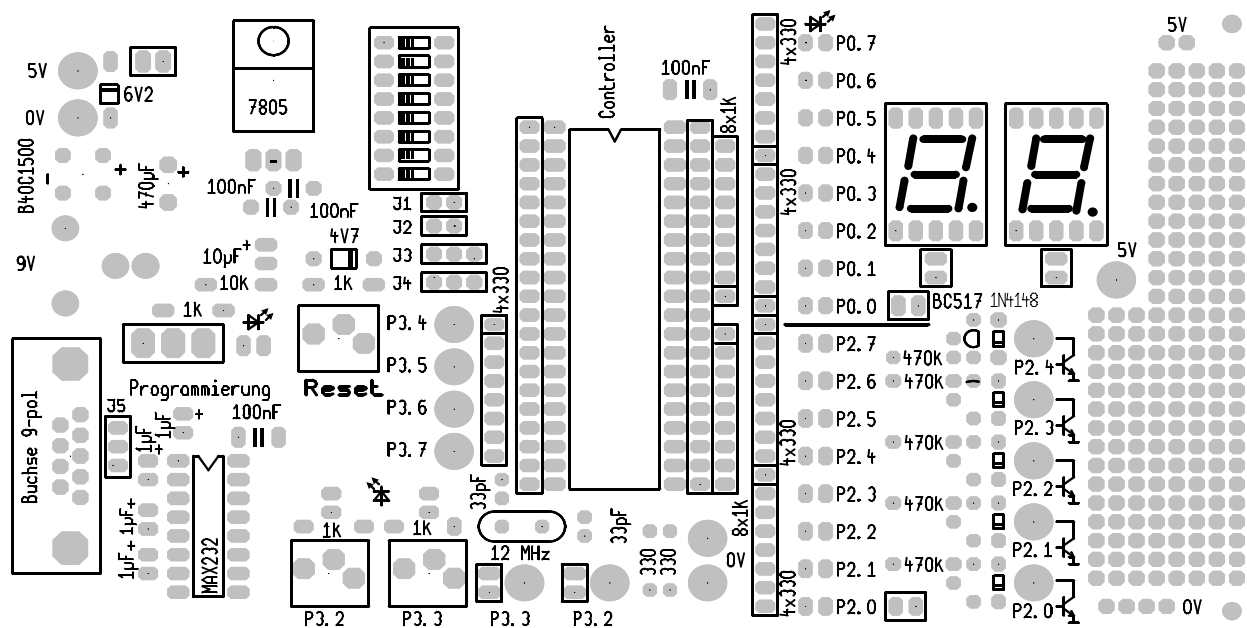
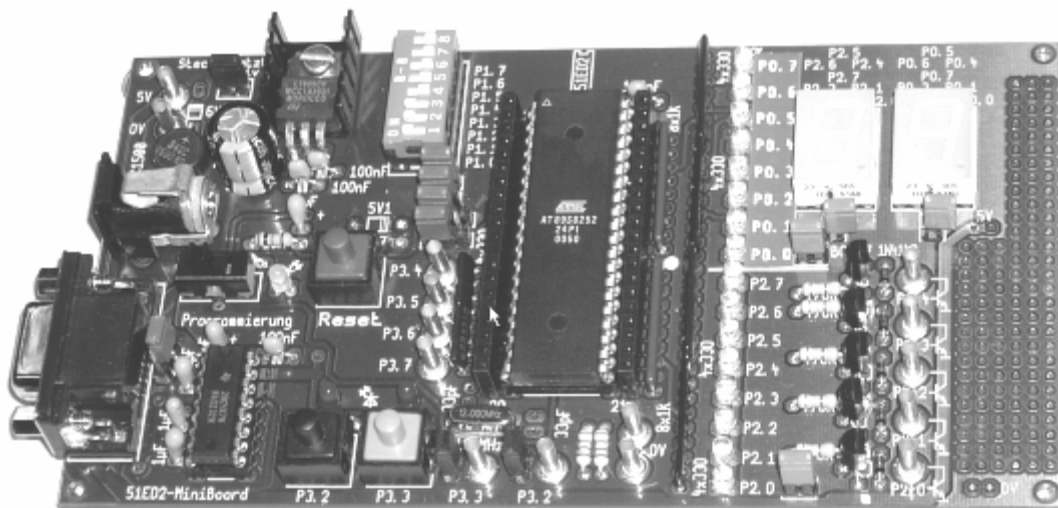
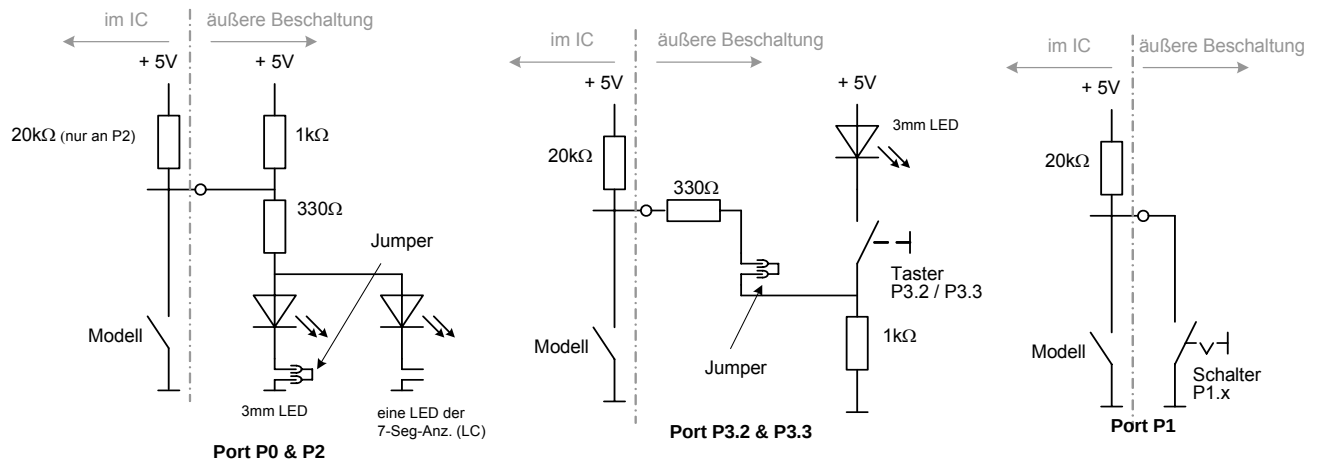
TF _x	TR _x
0, 1	0 Timer x stopp
	1 Timer x läuft
Wird beim Timer x-Überlauf gesetzt	

Interrupt	Einsprung Adresse	PortBit	Freigabe mit	Flanken-gesteuert	Auslösende Flanke	gesetztes Bit bei Interrupt-Anforderung
/INT0	0003h	P3.2	SETB EX0	SETB IT0	abfallend	IE0
/INT1	0013h	P3.3	SETB EX1	SETB IT1	abfallend	IE1

Blockschaltbild 8051



Beschaltung MC-Board (BubMini)



Besonderheiten des C-Compilers von Keil

spezielle Datentypen des C-51-Compilers für den Zugriff auf den SFR-Bereich

Datentyp	Größe	Wertebereich
sbit	1 Bit	0 oder 1
sfr	1 Byte	0 bis 255
sfr 16	2 Byte	0 bis 65535

Speichertypen bei C-51 (nach C-51-Bedienungsanleitung)

Speichertyp	Beschreibung
data	direkt adressierbarer interner Datenspeicher; ermöglicht schnellste Zugriffe auf Variablen (128 Byte) von 00h - 7Fh
bdata	bitadressierbarer, interner Datenspeicher, ermöglicht gemischten Bit und Byte-Zugriff von 20h - 2Fh.
idata	indirekt adressierbarer interner Datenspeicher; ermöglicht den Zugriff auf den vollen, internen Adressbereich bei beispielsweise 80535 von 00h - 0FFh.
pdata	'paged' (256 Byte) externer Datenspeicher, Zugriff mit dem Befehl movx @Ri.
xdata	externer Datenspeicher (64 KByte); Zugriff mit dem Befehl movx @dptr.
code	Programmspeicher (64Kbyte); Zugriff mit dem Befehl movc @A+DPTR.

Bei Variablendeklarationen ohne Angabe des Speichertyps werden in Abhängigkeit des Speichermodells die Voreingestellten Speichertypen verwendet.

Anmerkung zu Schiebe- und Bitbasierenden Befehlen:

Im Gegensatz zur PC-Programmierung ist das Schieben, sowie alle bitbasierenden Befehle nur mit Variablen erlaubt, die sich im Bitadressierbaren Speicher befinden.

Speichermodelle bei C-51 (nach C-51 Bedienungsanleitung)

Speichermodell	Beschreibung der Pragma-Anweisung
SMALL	Parameter und lokale Variablen werden im direkt adressierbaren, internen Datenspeicher platziert (max. 128 Byte; default Speichertyp: data)
COMPACT	Parameter und lokale Variablen werden im 'paged' externen Datenspeicher platziert (max. 256 Byte; default Speichertyp: pdata)
LARGE	Parameter und lokale Variablen werden im externen Datenspeicher platziert (max. 64 KByte; default Speichertyp: xdata)

Beispiel: #pragma large

Beispiele: Hexadezimale Adresse beginnen mit 0x (0x90 für P1)

Variablendeklaration	Wirkung
char data VAR1	8 Bit-Variable namens VAR1 in unteren int. Datenspeicher
char code Text[]="Eingabe !"	schreibt die Zeichenfolge Eingabe ! in den Programmspeicher
float idata x,y,z	definiert 3 4-Byte-Variablen im internen Datenspeicher
sfr P1 = 0x90	Weist dem String P1 die SFR-Adresse 90 zu (= Port 1)
sbit P10 = P1^0	Erstes Bit von Port 1 kann mit P10 angesprochen werden
sbit RD = 0xB0	Weist dem String RD die SFR-Adresse B0 zu (= P3.0)
sbit WR = 0xB1	Weist dem String WR die SFR-Adresse B1 zu (= P3.1)
char bdata FLAGS bit FLAG0 = FLAGS^0	8-Bit variable im Bitadressierbaren internen Datenspeicher bit 0 der Bitadressierten char-Variablen