Mikrocontroller-Grundlagen

Teil 10

Im zehnten Teil dieser Artikelserie zeigen wir eine übersichtliche Darstellung des MCS-51-Befehlssatzes, gefolgt von der Beschreibung der Timerfunktionen des Mikrocontrollers.

3.6 Befehlsübersicht

Wie aus den vorangegangenen Folgen der Mikrocontroller-Grundlagen-Serie ersichtlich, ist der MCS-51-Befehlssatz in 5 logisch trennbare Befehlsgruppen wie Transport-, Logik-, Arithmetik-, Bitmanipulations- und Programm-Steuerungsbefehle unterteilt. Tabelle 16 zeigt eine übersichtliche Darstellung des MCS-51-Befehlssatzes. Neben der eigentlichen Mnemonik sind weiterhin der eigentliche Befehlscode in hexadezimaler und binärer Schreibweise, die Anzahl der benötigten Speicher (ROM), die benötigte Zykluszeit sowie eine Kurzbeschreibung aufgeführt.

Durch die hexadezimale und binäre Aufführung des Befehlscodes ist es möglich, auch ohne einen Assembler, der die Mnemonik in den zugehörigen Maschinencode übersetzt, ein kleines Mikrocontroller-Programm zu erstellen.

Für die Register R 0 bis R 7 ist die Nummer des verwendeten Registers einzusetzen. Beispielsweise ist für das Register R4 r durch 4H bzw. 100B zu ersetzen.

Weiterhin zeigt Tabelle 16 die Anzahl der benötigten Bytes im Programmspeicher. Einfache Transfer- und Verknüpfungsbefehle, die kein Datenbyte benötigen, kommen mit einem Byte im Programmspeicher aus.

Die meisten Verknüpfungs- und Transferbefehle, die mit konstanten oder bitadressierbaren Speicherzellen arbeiten, benötigen neben dem eigentlichen Befehl noch ein zweites Byte für den Parameter. Nur wenige Befehle, wie beispielsweise der „MOV DPTR #KONST16"-Befehl, benötigen 3 Byte im Programmspeicher.

Ein Befehlszyklus wird grundsätzlich in 12 Taktzyklen des Prozessors abgearbeitet. Einfache Lade- und Verknüpfungsbe-

Tabelle 16: Übersichtliche Darstellung des MCS-51-Befehlssatzes, getrennt nach den Befehlsgruppen wie Transport-, Logik-, Arithmetik-, Bitmanipulations- und Programm-Steuerungsbefehle

			Mnemonik		Befehlscode Hex	Befehlscode Binär	Register	Bytes	Zykl.	Beschreibung
Transport-Befehle	RAM	interne RAM	MOV	A,Rr	E8+r	11101rrr	r=0..7	1	1	Move Register to Accumulator
			MOV	A,direct	E5	11100101		2	1	Move direct (byte) to Accumulator
			MOV	A,@Ri	E6+i	1110011i	i=0..1	1	1	Move indirect RAM to Accumulator
			MOV	A,#data	74	01110100		2	1	Move immediate data to Accumulator
			MOV	Rr,A	F8+r	11111rrr	r=0..7	1	1	Move Accumulator to Register
			MOV	Rr,direct	A8+r	10101rrr	r=0..7	2	2	Move direct (byte) to Register
			MOV	Rr,#data	78+r	01111rrr	r=0..7	2	1	Move immediate data to Register
			MOV	direct,A	F5	11110101		2	1	Move Accumulator to direct byte
			MOV	direct,Rr	88+r	10001rrr	r=0..7	2	2	Move Register to direct Byte
			MOV	direct,direct	85	10000101		3	2	Move direct (byte) to direct (byte)
			MOV	direct,@Ri	86+i	1000011i	i=0..1	2	2	Move indirect RAM to direct byte
			MOV	direct,#data	75	01110101		3	2	Move immediate data to direct byte
			MOV	@Ri,A	F6+i	1111011i	i=0..1	1	1	Move Accumulator to indirect RAM
			MOV	@Ri,direct	A6+i	1010011i	i=0..1	2	2	Move direct (byte) to indirect RAM
			MOV	@Ri,#data	76+i	0111011i	i=0..1	2	1	Move immediate data to indirect RAM
			MOV	DPTR,#data16	90	10010000		3	2	Load Data Pointer with a 16-bit constant
			XCH	A,Rr	C8+r	11001rrr	r=0..7	1	1	Exchange Register with Accumulator
			XCH	A,direct	C5	11000101		2	1	Exchange direct (byte) with Accumulator
			XCH	A,@Ri	C6+i	1100011i	i=0..1	1	1	Exchange indirect RAM with Accumulator
			XCHD	A,@Ri	D6+i	1101011i	i=0..1	1	1	Exchange low-order Digit from ind.RAM with A
			PUSH	direct	C0	11000000		2	2	Push direct (byte) onto stack
			POP	direct	D0	11010000		2	2	Pop direct (byte) from stack
		externes RAM	MOVX	A,@Ri	E2+i	1110001i	i=0..1	1	2	Move external RAM (8-Bit addr) to Accumulator
			MOVX	A,@DPTR	E0	11100000		1	2	Move external RAM (16-Bit addr) to Accumulator
			MOVX	@Ri,A	F2+i	1111001i	i=0..1	1	2	Move Accumulator to external RAM (8-Bit addr)
			MOVX	@DPTR,A	F0	11110000		1	2	Move Accumulator to external RAM (16-Bit addr)
	ROM		MOVC	A,@A+DPTR	93	10010011		1	2	Move Code byte relative to DPTR to Accumulator
			MOVC	A,@A+PC	83	10000011		1	2	Move Code byte relative to PC to Accumulator
Logik-Befehle	UND		ANL	A,Rr	58+r	01011rrr	r=0..7	1	1	AND Accumulator with Register to Accumulator
			ANL	A,direct	55	01010101		2	1	AND Accumulator with direct (byte) to Accumulator
			ANL	A,@Ri	56+i	0101011i	i=0..1	1	1	AND Accumulator with indirect RAM to Accumulator
			ANL	A,#data	54	01010100		2	1	AND Accumulator with immediate data to Accumulator
			ANL	direct,A	52	01010010		2	1	AND direct with Accumulator to direct (byte)
			ANL	direct,#data	53	01010011		3	2	AND direct with immediate data to direct (byte)
	ODER		ORL	A,Rr	48+r	01001rrr	r=0..7	1	1	OR Accumulator with Register to Accumulator
			ORL	A,direct	45	01000101		2	1	OR Accumulator with direct (byte) to Accumulator
			ORL	A,@Ri	46+i	0100011i	i=0..1	1	1	OR Accumulator with indirect RAM to Accumulator
			ORL	A,#data	44	01000100		2	1	OR Accumulator with immediate data to Accumulator
			ORL	direct,A	42	01000010		2	1	OR direct with Accumulator to direct (byte)
			ORL	direct,#data	43	01000011		3	2	OR direct with immediate data to direct (byte)
	EXOR		XRL	A,Rr	68+r	01101rrr	r=0..7	1	1	Exclusive-OR Accumulator with Register to Accumulator
			XRL	A,direct	65	01100101		2	1	Exclusive-OR Accumulator with direct (byte) to Accumulator
			XRL	A,@Ri	66+i	0110011i	i=0..1	1	1	Exclusive-OR Accumulator with indirect RAM to Accumulator
			XRL	A,#data	64	01100100		2	1	Exclusive-OR Accumulator with immediate data to Accumulator
			XRL	direct,A	62	01100010		2	1	Exclusive-OR direct with Accumulator to direct (byte)
			XRL	direct,#data	63	01100011		3	2	Exclusive-OR direct with immediate data to direct (byte)
	Rotation		RL	A	23	00100011		1	1	Rotate Accumulator Left
			RLC	A	33	00110011		1	1	Rotate Accumulator Left trough Carry flag
			RR	A	03	00000011		1	1	Rotate Accumulator Right
			RRC	A	13	00010011		1	1	Rotate Accumulator Right trough Carry flag
	Spezial		SWAP	A	C4	11000100		1	1	Swap nibbles within the Accumulator
			CLR	A	E4	11100100		1	1	Clear Accumulator
			CPL	A	F4	11110100		1	1	Complement Accumulator
Arithmetik-Befehle	Addition		INC	A	04	00000100		1	1	Increment Accumulator
			INC	Rr	08+r	00001rrr	r=0..7	1	1	Increment Register
			INC	direct	05	00000101		2	1	Increment direct (byte)
			INC	@Ri	06+i	0000011i	i=0..1	1	1	Increment indirect RAM
			INC	DPTR	A3	10100011		1	2	Increment Data Pointer
			ADD	A,Rr	28+r	00101rrr	r=0..7	1	1	Add Register to Accumulator
			ADD	A,direct	25	00100101		2	1	Add direct (byte) to Accumulator
			ADD	A,@Ri	26+i	0010011i	i=0..1	1	1	Add indirect RAM to Accumulator
			ADD	A,#data	24	00100100		2	1	Add immediate data to Accumulator
			ADDC	A,Rr	38+r	00111rrr	r=0..7	1	1	Add Register to Accumulator with Carry
			ADDC	A,direct	35	00110101		2	1	Add direct (byte) to Accumulator with Carry
			ADDC	A,@Ri	36+i	0011011i	i=0..1	1	1	Add indirect RAM to Accumulator with Carry
			ADDC	A,#data	34	00110100		2	1	Add immediate data to Accumulator with Carry
	Spezial		DA	A	D4	11010100		1	1	Decimal Adjust Accumulator
	Subtraktion		DEC	A	14	00010100		1	1	Decrement Accumulator
			DEC	Rr	18+r	00011rrr	r=0..7	1	1	Decrement Register
			DEC	direct	15	00010101		2	1	Decrement direct (byte)
			DEC	@Ri	16+i	0001011i	i=0..1	2	1	Decrement indirect RAM
			SUBB	A,Rr	98+r	10011rrr	r=0..7	1	1	Subtract Register from Accumulator with Borrow
			SUBB	A,direct	95	10010101		2	1	Subtract direct (byte) from Accumulator with Borrow
			SUBB	A,@Ri	96+i	1001011i	i=0..1	1	1	Subtract indirect RAM from Accumulator with Borrow
			SUBB	A,#data	94	10010100		2	1	Subtract immediate data from Accumulator with Borrow
	Multiplikation		MUL	AB	A4	10100100		1	4	Multiply A with B
	Division		DIV	AB	84	10000100		1	4	Divide A by B

Fortsetzung Tabelle 16

		Mnemonik		Befehlscode Hex	Befehlscode Binär	Register	Bytes	Zykl.	Beschreibung
Bitmanipulations-Befehle	Transport	MOV	C,bit	A2	10100010		2	1	Move direct bit to Carry flag
		MOV	bit,C	92	10010010		2	1	Move Carry flag to direct bit
	UND	ANL	C,bit	82	10000010		2	2	AND direct bit to Carry flag
		ANL	C,/bit	B0	10110000		2	2	AND complement of direct bit to Carry flag
	ODER	ORL	C,bit	72	01110010		2	2	OR direct bit to Carry flag
		ORL	C,/bit	A0	10100000		2	2	OR complement of direct bit to Carry flag
	Invertierung	CPL	C	B3	10110011		1	1	Complement Carry flag
		CPL	bit	B2	10110010		2	1	Complement direct bit
	Setzen	SETB	C	D3	11010011		1	1	Set Carry flag
		SETB	bit	D2	11010010		2	1	Set direct bit
	Löschen	CLR	C	C3	11000011		1	1	Clear Carry flag
		CLR	bit	C2	11000010		2	1	Clear direct bit
Programmablauf-Steuerungs-Befehle	Unterprogramm-Befehle	ACALL	addr11(page)	p1	ppp00001	p=0..7	2	2	Absolute Subroutine Call
		LCALL	addr16	12	00010010		3	2	Long Subroutine Call
		RET		22	00100010		1	2	Return from Subroutine
		RETI		32	00110010		1	2	Return from Interrupt
	Verzweigungs-Befehle: unbedingte Sprünge	AJMP	addr11	p1	ppp10001	p=0..7	2	2	Absolute Jump
		LJMP	addr16	02	00000010		3	2	Long Jump
		SJMP	rel	80	10000000		2	2	Short Jump (relative addr)
		JMP	@A+DPTR	73	01110011		1	2	Jump indirect Accumulator relative to DPTR
		NOP		00	00000000		1	1	No Operation
	Verzweigungs-Befehle: bedingte Sprünge	JZ	rel	60	01100000		2	2	Jump if Accumulator is Zero
		JNZ	rel	70	01110000		2	2	Jump if Accumulator is Not Zero
		JC	rel	40	01000000		2	1	Jump if Carry flag is set
		JNC	rel	50	01010000		2	2	Jump if Carry flag is Not set
		JB	bit,rel	20	00100000		3	2	Jump if direct bit is set
		JNB	bit,rel	30	00110000		3	2	Jump if direct bit is Not set
		JBC	bit,rel	10	00010000		3	2	Jump if direct bit is set and Clear bit
		CJNE	A,direct,rel	B5	10110101		3	2	Compare direct to Accumulator and Jump if Not Equal
		CJNE	A,#data,rel	B4	10110100		3	2	Compare immediate data to A and Jump if Not Equal
		CJNE	Rr,#data,rel	B8+r	10111rrr	r=0..7	3	2	Compare immediate data to Reg. and Jump if Not Equal
		CJNE	@Ri,#data,rel	B6+i	1011011i	i=0..1	3	2	Compare immediate data to ind.RAM and Jump if Not
		DJNZ	Rr,rel	D8+r	11011rrr	r=0..7	2	2	Decrement register and Jump if Not Zero
		DJNZ	direct,rel	D5	11010101		3	2	Decrement direct and Jump if Not Zero

fehle benötigen lediglich einen Befehlszyklus, während Befehle, die eine direkt adressierbare Speicherzelle ansprechen oder die Berechnung einer Adresse vornehmen müssen, benötigen 2 Befehlszyklen. Die Ausnahme stellen die Multiplikations- und Divisionsbefehle dar, die für die Berechnung einer 8 x 8-Bit-Multiplikation bzw. Division 4 Befehlszyklen, d. h. 48 Taktzyklen benötigen.

Tabelle 17 zeigt die Befehlsliste der MCS-51-Familie in numerischer Reihenfolge. Damit lassen sich bereits vorhandene 8031/51-Programme auf einfache Weise disassemblieren, um eine einfache Überprüfung des vorliegenden Programmes vornehmen zu können.

Die relativen und absoluten Sprungadressen, deren Parameter mit „rel" und „page" gekennzeichnet sind, werden direkt in Abhängigkeit des aktuellen Programmcounterstands (Adresse des Befehls) berechnet.

Bei der relativen Adressierungsart ist zu dem Stand des Programmcounters nach Ausführung des Befehls der vorzeichenbehaftete Parameter (-128 bis +127) zu addieren. Damit ist eine Sprungweite von maximal 128 Bytes, bezogen auf den ge-

Tabelle 17: Befehlsliste der MSC-51-Familie in numerischer Reihenfolge

Hex	Mnemonik		Hex	Mnemonik		Hex	Mnemonik		Hex	Mnemonik		Hex	Mnemonik		Hex	Mnemonik		Hex	Mnemonik	
00	NOP		20	JB	bitadr, rel	40	JC	rel	60	JZ	rel	80	SJMP	rel	A0	ORL	C,/bitadr	C0	PUSH	dadr	E0	MOVX	A, @DPTR
01	AJMP	page 0	21	AJMP	page 1	41	AJMP	page 2	61	AJMP	page 3	81	AJMP	page 4	A1	AJMP	page 5	C1	AJMP	page 6	E1	AJMP	page 7
02	LJMP	addr16	22	RET		42	ORL	dadr, A	62	XRL	dadr, A	82	ANL	C, bitadr	A2	MOV	C, bitadr	C2	CLR	bitadr	E2	MOVX	A, @R0
03	RR	A	23	RL	A	43	ORL	dadr, #data	63	XRL	dadr, #data	83	MOVC	A, @A+PC	A3	INC	DPTR	C3	CLR	C	E3	MOVX	A, @R1
04	INC	A	24	ADD	A, #data	44	ORL	A, #data	64	XRL	A, #data	84	DIV	AB	A4	MUL	AB	C4	SWAP	A	E4	CLR	A
05	INC	dadr	25	ADD	A, dadr	45	ORL	A, dadr	65	XRL	A, dadr	85	MOV	dadr, dadr	A5	reserviert		C5	XCH	A, dadr	E5	MOV	A, dadr
06	INC	@R0	26	ADD	A, @R0	46	ORL	A @R0	66	XRL	A, @R0	86	MOV	dadr, @R0	A6	MOV	@R0, dadr	C6	XCH	A, @R0	E6	MOV	A, @R0
07	INC	@R1	27	ADD	A, @R1	47	ORL	A,@R1	67	XRL	A, @R1	87	MOV	dadr, @R1	A7	MOV	@R1, dadr	C7	XCH	A, @R1	E7	MOV	A, @R1
08	INC	R0	28	ADD	A, R0	48	ORL	A, R0	68	XRL	A, R0	88	MOV	dadr, R0	A8	MOV	R0, dadr	C8	XCH	A, R0	E8	MOV	A, R0
09	INC	R1	29	ADD	A, R1	49	ORL	A, R1	69	XRL	A, R1	89	MOV	dadr, R1	A9	MOV	R1, dadr	C9	XCH	A, R1	E9	MOV	A, R1
0A	INC	R2	2A	ADD	A, R2	4A	ORL	A, R2	6A	XRL	A, R2	8A	MOV	dadr, R2	AA	MOV	R2, dadr	CA	XCH	A, R2	EA	MOV	A, R2
0B	INC	R3	2B	ADD	A, R3	4B	ORL	A, R3	6B	XRL	A, R3	8B	MOV	dadr, R3	AB	MOV	R3, dadr	CB	XCH	A, R3	EB	MOV	A, R3
0C	INC	R4	2C	ADD	A, R4	4C	ORL	A, R4	6C	XRL	A, R4	8C	MOV	dadr, R4	AC	MOV	R4, dadr	CC	XCH	A, R4	EC	MOV	A, R4
0D	INC	R5	2D	ADD	A, R5	4D	ORL	A, R5	6D	XRL	A, R5	8D	MOV	dadr, R5	AD	MOV	R5, dadr	CD	XCH	A, R5	ED	MOV	A, R5
0E	INC	R6	2E	ADD	A, R6	4E	ORL	A, R6	6E	XRL	A, R6	8E	MOV	dadr, R6	AE	MOV	R6, dadr	CE	XCH	A, R6	EE	MOV	A, R6
0F	INC	R7	2F	ADD	A, R7	4F	ORL	A, R7	6F	XRL	A, R7	8F	MOV	dadr, R7	AF	MOV	R7, dadr	CF	XCH	A, R7	EF	MOV	A, R7
10	JBC	bitadr., rel	30	JNB	bitadr, rel	50	JNC	rel	70	JNZ	rel	90	MOV	DPTR,#data16	B0	ANL	C,/bitadr	D0	POP	dadr	F0	MOVX	@DPTR, A
11	ACALL	page 0	31	ACALL	page 1	51	ACALL	page 2	71	ACALL	page 3	91	ACALL	page 4	B1	ACALL	page 5	D1	ACALL	page 6	F1	ACALL	page 7
12	LCALL	addr16	32	RETI		52	ANL	dadr, A	72	ORL	C, bitadr	92	MOV	bitadr, C	B2	CPL	bitadr	D2	SETB	bitadr	F2	MOVX	@R0, A
13	RRC	A	33	RLC	A	53	ANL	dadr, #data	73	JMP	@A+DPTR	93	MOVC	A, @A+DPTR	B3	CPL	C	D3	SETB	C	F3	MOVX	@R1, A
14	DEC	A	34	ADDC	A, #data	54	ANL	A, #data	74	MOV	A, #data	94	SUBB	A, #data	B4	CJNE	A, #data, rel	D4	DA	A	F4	CPL	A
15	DEC	dadr	35	ADDC	A, dadr	55	ANL	A, dadr	75	MOV	dadr, #data	95	SUBB	A, dadr	B5	CJNE	A, dadr, rel	D5	DJNZ	dadr, rel	F5	MOV	dadr, A
16	DEC	@R0	36	ADDC	A, @R0	56	ANL	A, @R0	76	MOV	@R0, #data	96	SUBB	A, @R0	B6	CJNE	@R0, #data, rel	D6	XCHD	A @R0	F6	MOV	@R0, A
17	DEC	@R1	37	ADDC	A, @R1	57	ANL	A, @R1	77	MOV	@R1, #data	97	SUBB	A, @R1	B7	CJNE	@R1, #data, rel	D7	XCHD	A @R1	F7	MOV	@R1, A
18	DEC	R0	38	ADDC	A, R0	58	ANL	A, R0	78	MOV	R0, #data	98	SUBB	A, R0	B8	CJNE	R0, #data, rel	D8	DJNZ	R0, rel	F8	MOV	R0, A
19	DEC	R1	39	ADDC	A, R1	59	ANL	A, R1	79	MOV	R1, #data	99	SUBB	A, R1	B9	CJNE	R1, #data, rel	D9	DJNZ	R1, rel	F9	MOV	R1, A
1A	DEC	R2	3A	ADDC	A, R2	5A	ANL	A, R2	7A	MOV	R2, #data	9A	SUBB	A, R2	BA	CJNE	R2, #data, rele	DA	DJNZ	R2, rel	FA	MOV	R2, A
1B	DEC	R3	3B	ADDC	A, R3	5B	ANL	A, R3	7B	MOV	R3, #data	9B	SUBB	A, R3	BB	CJNE	R3, #data, rel	DB	DJNZ	R3, rel	FB	MOV	R3, A
1C	DEC	R4	3C	ADDC	A, R4	5C	ANL	A, R4	7C	MOV	R4, #data	9C	SUBB	A, R4	BC	CJNE	R4, #data, rel	DC	DJNZ	R4, rel	FC	MOV	R4, A
1D	DEC	R5	3D	ADDC	A, R5	5D	ANL	A, R5	7D	MOV	R5, #data	9D	SUUB	A, R5	BD	CJNE	R5, #data, rel	DD	DJNZ	R5, rel	FD	MOV	R5, A
1E	DEC	R6	3E	ADDC	A, R6	5E	ANL	A, R6	7E	MOV	R6, #data	9E	SUBB	A, R6	BE	CJNE	R6, #data, rel	DE	DJNZ	R6, rel	FE	MOV	R6, A
1F	DEC	R7	3F	ADDC	A, R7	5F	ANL	A, R7	7F	MOV	R7, #data	9F	SUBB	A, R7	BF	CJNE	R7, #data, rel	DF	DJNZ	R7, rel	FF	MOV	R7, A

data = 8-Bit-Konstante
data 16 = 16-Bit-Konstante (MSB, LSB)
dadr = direkt byteadressierbare Speicherzelle
bitadr = direkte bitadressierbare Speicherzelle
page 0.. 7 = Page im aktuellen 2k-Bereich
rel = relative Sprungadresse
addr 16 = 16-Bit-Adresse (MSB, LSB)

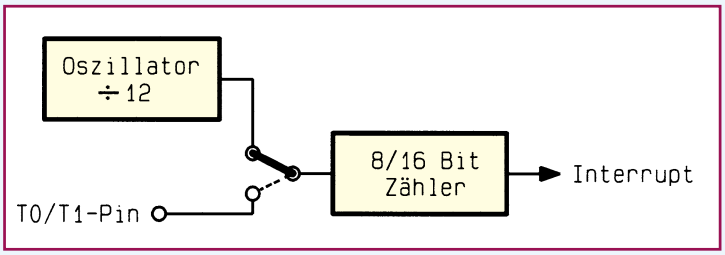

Bild 94: Wahlweise Ansteuerung der 8/16-Bit-Zähler durch die Oszillatortaktfrequenz oder einem externen Ereignis

nannten Programmcounterstand, möglich. Bei der absoluten Adressierung sind zu der Adresse des aktuellen 2k-Bereiches (nur die höchsten 5 Bit) die im Befehl genannte „page" multipliziert mit 256 und der angegebene Parameter (0 bis 255) zu addieren. Damit ergibt sich eine Sprungweite über einen Bereich von maximal 2048 Byte im aktuellen 2kByte-Segment.

Für den komfortablen Umgang mit Assembler- bzw. Binärcode für Mikrocontroller gibt es eine Reihe von Hilfsmitteln wie Assembler, Disassembler, Debugger usw., auf die wir im Laufe dieser Artikelserie noch näher eingehen.

4. Timer

Die meisten mit Mikrocontrollern realisierten Steuerungen benötigen eine Zeitsteuerung. Nun könnte ein Hauptprogramm die Anzahl der durchlaufenden Befehle zählen und so definierte Zeitabläufe bestimmen. Einfacher lassen sich prozessorinterne 8- bzw. 16-Bit-Zähler des Mikrocontrollers verwenden, insbesondere zur Realisierung von Uhrenfunktionen oder zur Steuerung von Relais. Darüber hinaus erfüllen diese wichtigen Bestandteile eines Mikroprozessors noch eine Reihe anderer wichtiger Aufgaben.

Die MCS-51-Familie besitzt 2 voneinander unabhängige 16-Bit-Timer, die sich auch 8-Bit-weise verwenden lassen. Der 8032/52 besitzt noch einen dritten 16-Bit-Timer mit weiteren Zusatzfunktionen. Die Zähler lassen sich wahlweise mit externen Ereignissen oder über den internen Oszillator, wie Abbildung 94 zeigt, takten.

Die 16-Bit-Zählregister können über jeweils 2 direkt adressierbare Adressen jederzeit beschrieben bzw. ausgelesen werden. Ein Überlauf des 16-Bit-Zählers kann die Auslösung eines Interrupts (Programmunterbrechung) auslösen.

Jeder der beiden 16-Bit-Zählregister benötigt 2 Byte, die als direkt adressierbare Speicherzellen jederzeit beschreibbar bzw. auslesbar sind. Tabelle 18 zeigt die Byteadressen und die vollständige Bezeichnung der zu jeweils 2 Byte aufgeteilten Timer. Jeder der Timer läßt sich auch als Ereigniszähler verwenden. Die Einstellung erfolgt über das direkt adressierbare Timermodus-Register TMOD.

Tabelle 19 zeigt eine Übersicht über den Inhalt des Timermodus-Registers. In dem höherwertigen Nibble sind die Steuerelemente für den Timer 1 enthalten, während die Betriebsart des Timers 0 über das untere Nibbel gesteuert wird.

Das Timer-on-Bit (TR im TCON-Register), auf das wir noch zu sprechen kommen, schaltet die Funktion der Timer 0 bzw. 1 ein. Das in TMOD untergebrachte Bit „Gate" legt fest, ob die Zählimpulse kontinuierlich zum 8- bzw. 16-Bit-Zähler gelangen oder diese durch den $\overline{\text{INT}}$ 0/1-Pin gesteuert werden können. Die letztgenannte Möglichkeit wird hauptsächlich zur Impulsbreitenmessung eingesetzt.

Mit dem C/$\overline{\text{T}}$-Bit erfolgt die Auswahl zwischen Zähler und Timer. Ist das Bit gesetzt, werden die nachgeschalteten Zähler über den T0- bzw. T1-Pin des Prozessors getaktet. Anderenfalls erfolgt die Ansteuerung des Zählers über die durch 12 geteilte Oszillatorfrequenz.

Die Bits M0 und M1 bestimmen den Funktionsmode der Timer 0 und 1. Abbildung 95 zeigt die Auswahl der Timerbetriebsmodi über M0 und M1 für die Timer 0 und 1.

Jeweils 2 weitere Steuerflags für die beiden Timer sind in der direkt bitadressierbaren Speicherzelle TCON, wie Tabelle 20 zeigt, untergebracht.

Bild 95: Auswahl der Timer-Betriebsmodi

M1	M0	Mode	Funktion
0	0	0	5-Bit-Verteiler (Low-Byte) und nachgeschalteter 8-Bit-Zähler
0	1	1	Low- und High-Byte zu einem 16-Bit-Zähler zusammengeschaltet
1	0	2	8-Bit selbstnachladender Timer/Zähler. Low-Byte = Zähler High-Byte = Nachladewert bei Überlauf
1	1	3	Timer 0 bildet 2 unabhängige 8-Bit-Timer Timer 1 kann in den Modi 0 bis 3 als Überlaufzähler genutzt werden

Tabelle 18: Byteadressen der Timer des 8031/51

Bezeichnung	Abkürzung	Adresse
Timer 0 Low-Byte	TL 0	08AH
Timer 0 High-Byte	TH 0	08CH
Timer 1 Low-Byte	TL 1	08BH
Timer 1 High-Byte	TH 1	08DH

Neben den 4 für die beiden Timer zuständigen Bits sind in diesem Speicher noch 4 weitere Bits für die Interrupt-Steuerung plaziert, die wir im Verlauf dieser Artikelserie noch näher beschreiben werden.

Die Run-Bits TR 0 und TR 1 übernehmen die Freigabe der beiden Zähler. Wird in dieses Bit eine 1 geschrieben, beginnt der Timer bzw. Zähler zu zählen, anderenfalls stoppt der Timer/Zähler.

Die Überlauf-Flip-Flops TF 0 und 1 kennzeichnen einen Überlauf der Zähler 0 bzw. 1. Dieses Flag wird jedesmal automatisch gesetzt, wenn der Timer oder Zähler seinen Höchstwert überschritten hat und von vorne (0) zu zählen beginnt.

Die Überlauf-Flip-Flops können zum Steuern der Interrupt-Auslösung dienen, lassen sich aber auch direkt in der Software zur Abfrage eines Zählerüberlaufes verwenden. Um eine korrekte Funktion des Programmes sicherzustellen sind diese Flags nach der Auswertung von der Software zu löschen.

Im elften Teil dieser Artikelserie fahren wir mit der Beschreibung der unterschiedlichen Timer-Betriebsarten fort. ELV

Tabelle 19: Bedeutung des Timermodus-Kontroll-Registers TMOD

Timer/Counter 1				Timer/Counter 0			
Gate	C/$\overline{\text{T}}$	M1	M0	Gate	C/$\overline{\text{T}}$	M1	M0
Byteadresse = 089H							

Tabelle 20: Bit- und Byteadressen des Timer-Kontroll-Registers TCON

Zuordnung	Timer 1		Timer 0		Interruptkontrolle			
Bedeutung	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Bitadresse	8FH	8EH	8DH	8CH	8BH	8AH	89H	88H
Byteadresse	088H							