Hohentwiel-
Gewerbeschule Singen

Formelsammlung zur Technischen Informatik

Mikrocontrollertechnik

System C8051F340

Karl Laber
Hohentwiel-Gewerbeschule Singen
Uhlandstr. 27
78224 Singen

Version 2.0
10.03.2020

Inhaltsverzeichnis

1. Mikrocontroller C8051F340 von Silicon Labs (48 Pin Version)

Taster:

K3-RST	→ Programm Reset
K1-P20	→ Taster am Port 2.0 (Low Aktiv)
K2-P21	→ Taster am Port 2.1 (Low Aktiv)

LEDs:

D1	→Power ON
D2	→ Port 2.2
D3	→ Port 2.3

SILICON LABS

C8051F340/1/2/3/4/5/6/7/8/9/A/B/C/D

Full Speed USB Flash MCU Family

Analog Peripherals
- **10-Bit ADC (C8051F340/1/2/3/4/5/6/7/A/B only)**
  - Up to 200 ksps
  - Built-in analog multiplexer with single-ended and differential mode
  - VREF from external pin, internal reference, or V_{DD}
  - Built-in temperature sensor
  - External conversion start input option
- **Two comparators**
- **Internal voltage reference (C8051F340/1/2/3/4/5/6/7/A/B only)**
- **Brown-out detector and POR Circuitry**

USB Function Controller
- USB specification 2.0 compliant
- Full speed (12 Mbps) or low speed (1.5 Mbps) operation
- Integrated clock recovery; no external crystal required for full speed or low speed
- Supports eight flexible endpoints
- 1 kB USB buffer memory
- Integrated transceiver; no external resistors required

On-Chip Debug
- On-chip debug circuitry facilitates full speed, non-intrusive in-system debug (No emulator required)
- Provides breakpoints, single stepping, inspect/modify memory and registers
- Superior performance to emulation systems using ICE-chips, target pods, and sockets

Voltage Supply Input: 2.7 to 5.25 V
- Voltages from 3.6 to 5.25 V supported using On-Chip Voltage Regulator

High Speed 8051 µC Core
- Pipelined instruction architecture; executes 70% of Instructions in 1 or 2 system clocks
- 48 MIPS and 25 MIPS versions available.
- Expanded interrupt handler

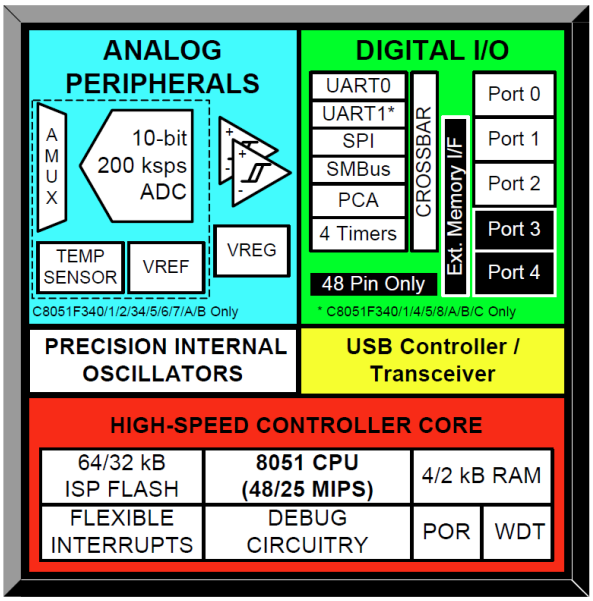
Memory
- 4352 or 2304 Bytes RAM
- 64 or 32 kB Flash; In-system programmable in 512-byte sectors

Digital Peripherals
- 40/25 Port I/O; All 5 V tolerant with high sink current
- Hardware enhanced SPI™, SMBus™, and one or two enhanced UART serial ports
- Four general purpose 16-bit counter/timers
- 16-bit programmable counter array (PCA) with five capture/compare modules
- External Memory Interface (EMIF)

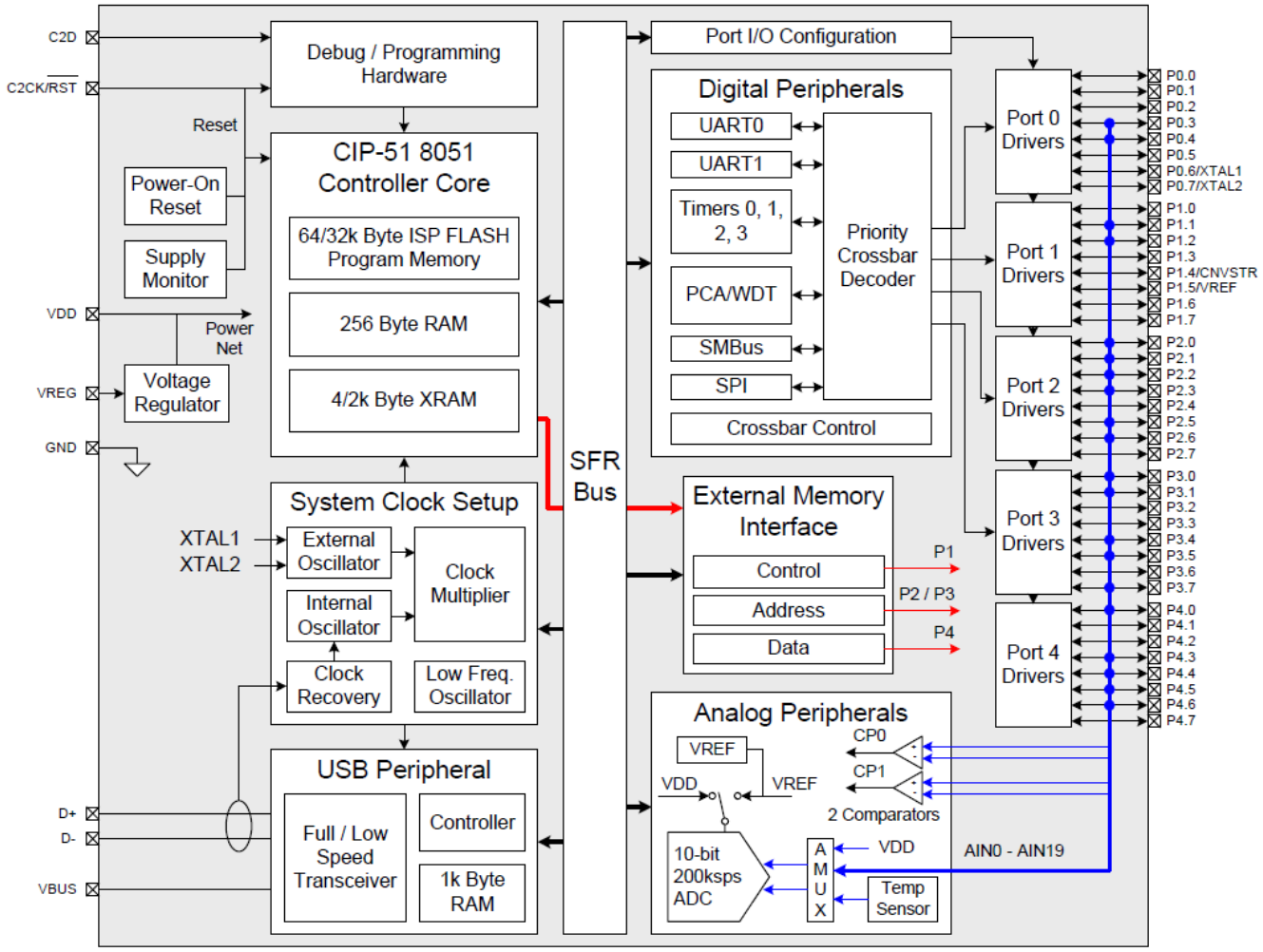
Clock Sources
- Internal Oscillator: ±0.25% accuracy with clock recovery enabled. Supports all USB and UART modes
- External Oscillator: Crystal, RC, C, or clock (1 or 2 Pin modes)
- Low Frequency (80 kHz) Internal Oscillator
- Can switch between clock sources on-the-fly

Packages
- 48-pin TQFP (C8051F340/1/4/5/8/C)
- 32-pin LQFP (C8051F342/3/6/7/9/A/B/D)
- 5x5 mm 32-pin QFN (C8051F342/3/6/7/9/A/B)

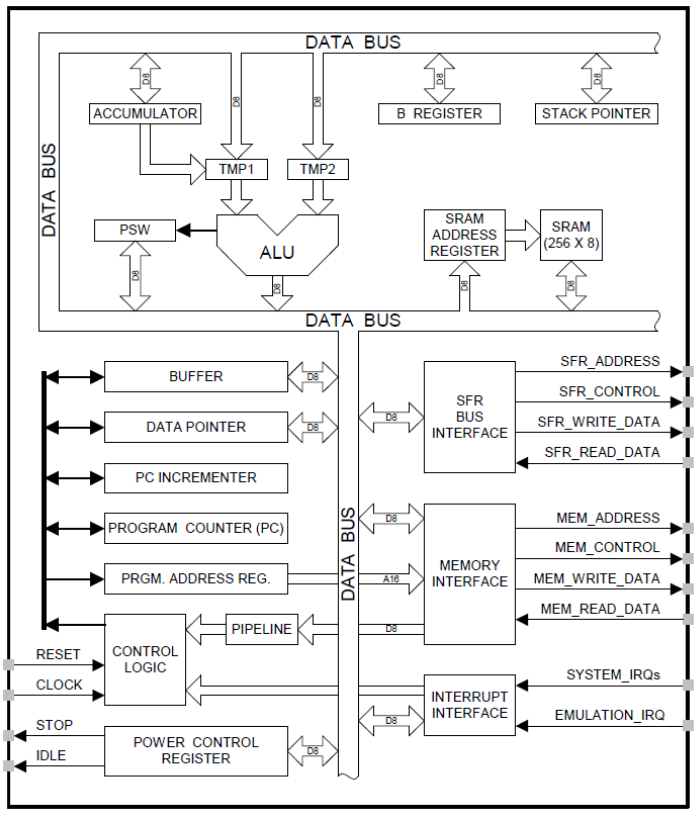
Temperature Range: –40 to +85 °C



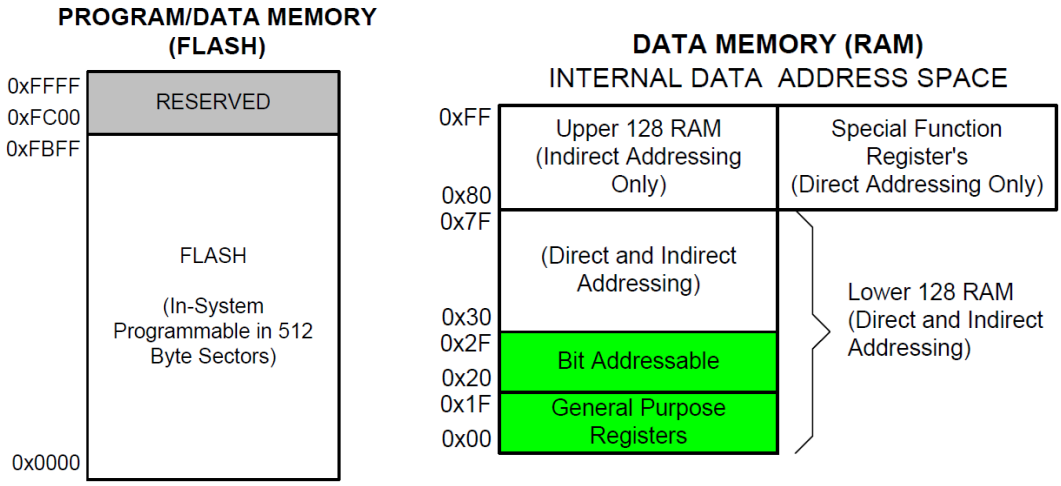
2. Blockschaltbild des C8051F340



Chip-51 8051 Controller Core (Kompatibel zum MCS-51 Befehlssatz)



3. Speicherorganisation



Special Function Register (SFR-Bereich)

F8	SPI0CN	PCA0L	PCA0H	PCA0CPL0	PCA0CPH0	PCA0CPL4	PCA0CPH4	VDM0CN
F0	B	P0MDIN	P1MDIN	P2MDIN	P3MDIN	P4MDIN	EIP1	EIP2
E8	ADC0CN	PCA0CPL1	PCA0CPH1	PCA0CPL2	PCA0CPH2	PCA0CPL3	PCA0CPH3	RSTSRC
E0	ACC	XBR0	XBR1	XBR2	IT01CF	SMOD1	EIE1	EIE2
D8	PCA0CN	PCA0MD	PCA0CPM0	PCA0CPM1	PCA0CPM2	PCA0CPM3	PCA0CPM4	P3SKIP
D0	PSW	REF0CN	SCON1	SBUF1	P0SKIP	P1SKIP	P2SKIP	USB0XCN
C8	TMR2CN	REG0CN	TMR2RLL	TMR2RLH	TMR2L	TMR2H	-	-
C0	SMB0CN	SMB0CF	SMB0DAT	ADC0GTL	ADC0GTH	ADC0LTL	ADC0LTH	P4
B8	IP	CLKMUL	AMX0N	AMX0P	ADC0CF	ADC0L	ADC0H	-
B0	P3	OSCXCN	OSCICN	OSCICL	SBRLL1	SBRLH1	FLSCL	FLKEY
A8	IE	CLKSEL	EMI0CN	-	SBCON1	-	P4MDOUT	PFE0CN
A0	P2	SPI0CFG	SPI0CKR	SPI0DAT	P0MDOUT	P1MDOUT	P2MDOUT	P3MDOUT
98	SCON0	SBUF0	CPT1CN	CPT0CN	CPT1MD	CPT0MD	CPT1MX	CPT0MX
90	P1	TMR3CN	TMR3RLL	TMR3RLH	TMR3L	TMR3H	USB0ADR	USB0DAT
88	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON	PSCTL
80	P0	SP	DPL	DPH	EMI0TC	EMI0CF	OSCLCN	PCON
	0(8) (bit addressable)	1(9)	2(A)	3(B)	4(C)	5(D)	6(E)	7(F)

Stackpointer (SP-Register)

Die Default-Einstellung ist 07h, d.h. der Stackbereich beginnt dann bei 08h im RAM, was auch gleich ist mit der Registerbank 1 (siehe unten). Man kann den Stackbereich aber auch mit z.B. MOV SP,#2Fh verschieben.

Program Status Word (PSW-Register)

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	Reset Value
CY	AC	F0	RS1	RS0	OV	F1	PARITY	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0xD0

RS1	RS0	Register Bank	Address
0	0	0	0x00 - 0x07
0	1	1	0x08 - 0x0F
1	0	2	0x10 - 0x17
1	1	3	0x18 - 0x1F

4. Timer/Counter

Der µC C8051F340 verfügt über insgesamt 4 Timer/Counter. Prüfungsrelevant sind jedoch nur Timer/Counter 0 und Timer/Counter 1!

Timer 0 and Timer 1 Modes:	Timer 2 Modes:	Timer 3 Modes:
13-bit counter/timer	16-bit timer with auto-reload	16-bit timer with auto-reload
16-bit counter/timer		
8-bit counter/timer with auto-reload	Two 8-bit timers with auto-reload	Two 8-bit timers with auto-reload
Two 8-bit counter/timers (Timer 0 only)		

Die Aktivierung des Timers/Counters kann unterschiedlich erfolgen (siehe Tabelle, Bsp. für Timer/Counter 0)

TR0	GATE0	$\overline{\text{INT0}}$	Counter/Timer
0	X	X	Disabled
1	0	X	Enabled
1	1	0	Disabled
1	1	1	Enabled

X = Don't Care

Einstellungen im TMOD-Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
GATE1	C/T1	T1M1	T1M0	GATE0	C/T0	T0M1	T0M0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x89

Bit7: GATE1: Timer 1 Gate Control.
0: Timer 1 enabled when TR1 = 1 irrespective of $\overline{\text{INT1}}$ logic level.
1: Timer 1 enabled only when TR1 = 1 AND $\overline{\text{INT1}}$ is active as defined by bit IN1PL in register INT01CF (see SFR Definition 9.13).

Bit6: C/T1: Counter/Timer 1 Select.
0: Timer Function: Timer 1 incremented by clock defined by T1M bit (CKCON.3).
1: Counter Function: Timer 1 incremented by high-to-low transitions on external input pin (T1).

Bits5–4: T1M1–T1M0: Timer 1 Mode Select.
These bits select the Timer 1 operation mode.

T1M1	T1M0	Mode
0	0	Mode 0: 13-bit counter/timer
0	1	Mode 1: 16-bit counter/timer
1	0	Mode 2: 8-bit counter/timer with auto-reload
1	1	Mode 3: Timer 1 inactive

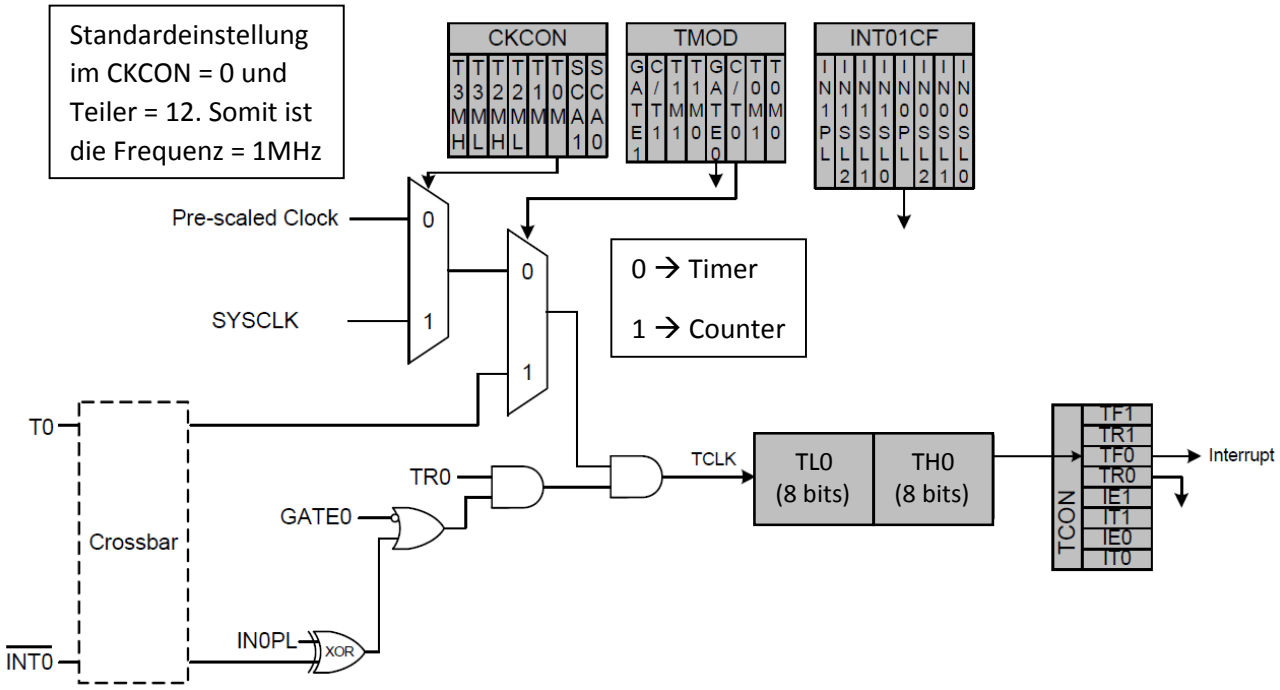
Bit3: GATE0: Timer 0 Gate Control.
0: Timer 0 enabled when TR0 = 1 irrespective of $\overline{\text{INT0}}$ logic level.
1: Timer 0 enabled only when TR0 = 1 AND $\overline{\text{INT0}}$ is active as defined by bit IN0PL in register INT01CF (see SFR Definition 9.13).

Bit2: C/T0: Counter/Timer Select.
0: Timer Function: Timer 0 incremented by clock defined by T0M bit (CKCON.2).
1: Counter Function: Timer 0 incremented by high-to-low transitions on external input pin (T0).

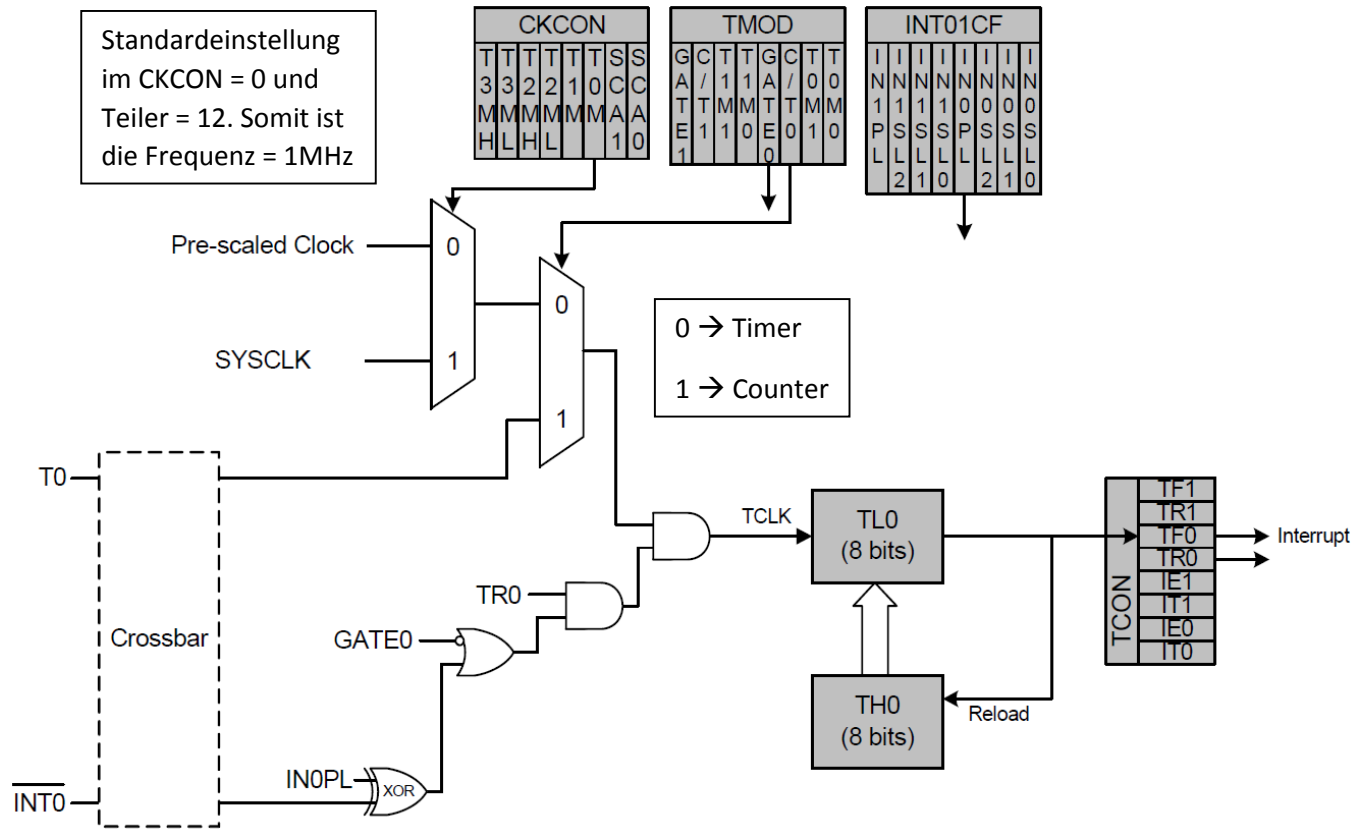
Bits1–0: T0M1–T0M0: Timer 0 Mode Select.
These bits select the Timer 0 operation mode.

T0M1	T0M0	Mode
0	0	Mode 0: 13-bit counter/timer
0	1	Mode 1: 16-bit counter/timer
1	0	Mode 2: 8-bit counter/timer with auto-reload
1	1	Mode 3: Two 8-bit counter/timers

Blockschaltbild für Mode 1 (16 Bit)



Blockschaltbild für Mode 2 (8-Bit Auto Reload)



Hinweis: Für den Timer1 gelten entsprechend die gleichen Blockschaltbilder. Anstatt TR0 gilt dann TR1, bzw. für TH0 gilt TH1, TL1, GATE1, TF1, T1 usw. Die Blockschaltbilder für Mode 0 und Mode 3 findet man im Datenblatt des Herstellers auf Seite 236 bzw. Seite 238.

5. Crossbar Einstellung (Countereingänge T0/T1)

	P0								P1								P2								P3							
SF Signals (32-pin Package)					XTAL1	XTAL2		CNVSTR	VREF																P3.1-P3.7 unavailable on the 32-pin packages							
SF Signals (48-pin Package)							XTAL1	XTAL2				ALE	CNVSTR	VREF	$\overline{RD}$	$\overline{WR}$																
PIN I/O	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
TX0					■																											
RX0						■																										
SCK	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■							
MISO		■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■						
MOSI			■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■					
NSS*				■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■				
SDA	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■			
SCL		■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■		
CP0	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	
CP0A	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
CP1	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
CP1A	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
SYSCLK	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
CEX0	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
CEX1		■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
CEX2			■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
CEX3				■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
CEX4					■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
ECI	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
T0	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
T1	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
TX1**	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	
RX1**		■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■

■ Port pin potentially available to peripheral

SF Signals Special Function Signals are not assigned by the Crossbar. When these signals are enabled, the Crossbar must be manually configured to skip their corresponding port pins.

**UART1 available only on C8051F340/1/4/5/8/A/B devices

*NSS is only pinned out in 4-wire SPI mode

Abb. 15.3. (C8051F340 Datasheet, S.144)

T0/T1 kann somit einem beliebigen Pin von Port P0 . . P3 zugewiesen werden!

SFR Definition 15.2. XBR1: Port I/O Crossbar Register 1

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
WEAKPUD	XBARE	T1E	T0E	ECIE	PCA0ME			00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xE2

Bit7: WEAKPUD: Port I/O Weak Pull-up Disable.
0: Weak Pull-ups enabled (except for Ports whose I/O are configured as analog input or push-pull output).
1: Weak Pull-ups disabled.

Bit6: XBARE: Crossbar Enable.
0: Crossbar disabled; all Port drivers disabled.
1: Crossbar enabled.

Bit5: T1E: T1 Enable
0: T1 unavailable at Port pin.
1: T1 routed to Port pin.

Bit4: T0E: T0 Enable
0: T0 unavailable at Port pin.
1: T0 routed to Port pin.

Um den Countereingang T0 entsprechend zu aktivieren (z.B. T0 → P3.4) sind dabei folgende Anweisungen erforderlich (Assembler-Code):

```
MOV XBR1,#01010000b     ;XBARE und T0E setzen
MOV P0SKIP,#0FFh        ;schiebt den T0 Pin auf P1.0
MOV P1SKIP,#0FFh        ;schiebt den T0 Pin auf P2.0
MOV P2SKIP,#0FFh        ;schiebt den T0 Pin auf P3.0
MOV P3SKIP,#0Fh         ;schiebt den T0 Pin auf P3.4
```

Um den Countereingang T1 entsprechend zu aktivieren (z.B. T1 → P3.5) sind dabei folgende Anweisungen erforderlich (C-Code):

```
XBR1 = 0b01100000;      //XBARE und T1E setzen
P0SKIP = 0xFF;          //schiebt den T1 Pin auf P1.0
P1SKIP = 0xFF;          //schiebt den T1 Pin auf P2.0
P2SKIP = 0xFF;          //schiebt den T1 Pin auf P3.0
P3SKIP = 0x1F;          //schiebt den T1 Pin auf P3.5
```

6. Interrupts

Interrupt-Quellen mit den Einsprungadressen (Interrupt Vector) und den Interrupt-Flags (Pending Flag)

Table 9.4. Interrupt Summary

Interrupt Source	Interrupt Vector	Priority Order	Pending Flag	Bit addressable?	Cleared by HW?	Enable Flag	Priority Control
Reset	0x0000	Top	None	N/A	N/A	Always Enabled	Always Highest
External Interrupt 0 ($\overline{INT0}$)	0x0003	0	IE0 (TCON.1)	Y	Y	EX0 (IE.0)	PX0 (IP.0)
Timer 0 Overflow	0x000B	1	TF0 (TCON.5)	Y	Y	ET0 (IE.1)	PT0 (IP.1)
External Interrupt 1 ($\overline{INT1}$)	0x0013	2	IE1 (TCON.3)	Y	Y	EX1 (IE.2)	PX1 (IP.2)
Timer 1 Overflow	0x001B	3	TF1 (TCON.7)	Y	Y	ET1 (IE.3)	PT1 (IP.3)

Interrupt-Freigabe Register (IE-Register)

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
EA	ESPI0	ET2	ES0	ET1	EX1	ET0	EX0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0xA8

Externe Interrupts (am Port 0)

Pegeleinstellung der externen Interrupts (Default Einstellung → IT0/1=0, IN0/1PL=0 → negativer Pegel!)

IT0	IN0PL	$\overline{INT0}$ Interrupt
1	0	Active low, edge sensitive
1	1	Active high, edge sensitive
0	0	Active low, level sensitive
0	1	Active high, level sensitive

IT1	IN1PL	$\overline{INT1}$ Interrupt
1	0	Active low, edge sensitive
1	1	Active high, edge sensitive
0	0	Active low, level sensitive
0	1	Active high, level sensitive

Hinweis: Bei der Einstellung auf Flanke, wird das Interrupt-Flag IE0 bzw. IE1 durch die Hardware gelöscht!

Einstellung der externen Interruptquelle am Port 0 im IT01CF Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
IN1PL	IN1SL2	IN1SL1	IN1SL0	IN0PL	IN0SL2	IN0SL1	IN0SL0	00000001
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
								0xE4

Externer Interrupt 1

IN1SL2–0	$\overline{INT1}$ Port Pin
000	P0.0
001	P0.1
010	P0.2
011	P0.3
100	P0.4
101	P0.5
110	P0.6
111	P0.7

Externer Interrupt 0

IN0SL2–0	$\overline{INT0}$ Port Pin
000	P0.0
001	P0.1
010	P0.2
011	P0.3
100	P0.4
101	P0.5
110	P0.6
111	P0.7

Default-Einstellung in IT01CF → Externer Interrupt 0 am Port P0.1 , Externer Interrupt 1 am Port P0.0

7. Befehlsliste des C8051F340

Notes on Registers, Operands and Addressing Modes:

Rn - Register R0-R7 of the currently selected register bank.

@Ri - Data RAM location addressed indirectly through R0 or R1.

rel - 8-bit, signed (two's complement) offset relative to the first byte of the following instruction. Used by SJMP and all conditional jumps.

direct - 8-bit internal data location's address. This could be a direct-access Data RAM location (0x00-0x7F) or an SFR (0x80-0xFF).

#data - 8-bit constant

#data16 - 16-bit constant

bit - Direct-accessed bit in Data RAM or SFR

addr11 - 11-bit destination address used by ACALL and AJMP. The destination must be within the same 2K-byte page of program memory as the first byte of the following instruction.

addr16 - 16-bit destination address used by LCALL and LJMP. The destination may be anywhere within the 8K-byte program memory space.

There is one unused opcode (0xA5) that performs the same function as NOP.

Mnemonic	Description	Bytes	Clock Cycles
	Data Transfer		
MOV A, Rn	Move Register to A	1	1
MOV A, direct	Move direct byte to A	2	2
MOV A, @Ri	Move indirect RAM to A	1	2
MOV A, #data	Move immediate to A	2	2
MOV Rn, A	Move A to Register	1	1
MOV Rn, direct	Move direct byte to Register	2	2
MOV Rn, #data	Move immediate to Register	2	2
MOV direct, A	Move A to direct byte	2	2
MOV direct, Rn	Move Register to direct byte	2	2
MOV direct, direct	Move direct byte to direct byte	3	3
MOV direct, @Ri	Move indirect RAM to direct byte	2	2
MOV direct, #data	Move immediate to direct byte	3	3
MOV @Ri, A	Move A to indirect RAM	1	2
MOV @Ri, direct	Move direct byte to indirect RAM	2	2
MOV @Ri, #data	Move immediate to indirect RAM	2	2
MOV DPTR, #data16	Load DPTR with 16-bit constant	3	3
MOVC A, @A+DPTR	Move code byte relative DPTR to A	1	3
MOVC A, @A+PC	Move code byte relative PC to A	1	3
MOVX A, @Ri	Move external data (8-bit address) to A	1	3
MOVX @Ri, A	Move A to external data (8-bit address)	1	3
MOVX A, @DPTR	Move external data (16-bit address) to A	1	3
MOVX @DPTR, A	Move A to external data (16-bit address)	1	3
PUSH direct	Push direct byte onto stack	2	2
POP direct	Pop direct byte from stack	2	2
XCH A, Rn	Exchange Register with A	1	1
XCH A, direct	Exchange direct byte with A	2	2
XCH A, @Ri	Exchange indirect RAM with A	1	2
XCHD A, @Ri	Exchange low nibble of indirect RAM with A	1	2

Mnemonic	Description	Bytes	Clock Cycles
Arithmetic Operations			
ADD A, Rn	Add register to A	1	1
ADD A, direct	Add direct byte to A	2	2
ADD A, @Ri	Add indirect RAM to A	1	2
ADD A, #data	Add immediate to A	2	2
ADDC A, Rn	Add register to A with carry	1	1
ADDC A, direct	Add direct byte to A with carry	2	2
ADDC A, @Ri	Add indirect RAM to A with carry	1	2
ADDC A, #data	Add immediate to A with carry	2	2
SUBB A, Rn	Subtract register from A with borrow	1	1
SUBB A, direct	Subtract direct byte from A with borrow	2	2
SUBB A, @Ri	Subtract indirect RAM from A with borrow	1	2
SUBB A, #data	Subtract immediate from A with borrow	2	2
INC A	Increment A	1	1
INC Rn	Increment register	1	1
INC direct	Increment direct byte	2	2
INC @Ri	Increment indirect RAM	1	2
DEC A	Decrement A	1	1
DEC Rn	Decrement register	1	1
DEC direct	Decrement direct byte	2	2
DEC @Ri	Decrement indirect RAM	1	2
INC DPTR	Increment Data Pointer	1	1
MUL AB	Multiply A and B	1	4
DIV AB	Divide A by B	1	8
DA A	Decimal adjust A	1	1

Boolean Manipulation			
CLR C	Clear Carry	1	1
CLR bit	Clear direct bit	2	2
SETB C	Set Carry	1	1
SETB bit	Set direct bit	2	2
CPL C	Complement Carry	1	1
CPL bit	Complement direct bit	2	2
ANL C, bit	AND direct bit to Carry	2	2
ANL C, /bit	AND complement of direct bit to Carry	2	2
ORL C, bit	OR direct bit to carry	2	2
ORL C, /bit	OR complement of direct bit to Carry	2	2
MOV C, bit	Move direct bit to Carry	2	2
MOV bit, C	Move Carry to direct bit	2	2
JC rel	Jump if Carry is set	2	2/4
JNC rel	Jump if Carry is not set	2	2/4
JB bit, rel	Jump if direct bit is set	3	3/5
JNB bit, rel	Jump if direct bit is not set	3	3/5
JBC bit, rel	Jump if direct bit is set and clear bit	3	3/5

Logical Operations			
ANL A, Rn	AND Register to A	1	1
ANL A, direct	AND direct byte to A	2	2
ANL A, @Ri	AND indirect RAM to A	1	2
ANL A, #data	AND immediate to A	2	2
ANL direct, A	AND A to direct byte	2	2
ANL direct, #data	AND immediate to direct byte	3	3
ORL A, Rn	OR Register to A	1	1
ORL A, direct	OR direct byte to A	2	2
ORL A, @Ri	OR indirect RAM to A	1	2
ORL A, #data	OR immediate to A	2	2
ORL direct, A	OR A to direct byte	2	2
ORL direct, #data	OR immediate to direct byte	3	3
XRL A, Rn	Exclusive-OR Register to A	1	1
XRL A, direct	Exclusive-OR direct byte to A	2	2
XRL A, @Ri	Exclusive-OR indirect RAM to A	1	2
XRL A, #data	Exclusive-OR immediate to A	2	2
XRL direct, A	Exclusive-OR A to direct byte	2	2
XRL direct, #data	Exclusive-OR immediate to direct byte	3	3
CLR A	Clear A	1	1
CPL A	Complement A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through Carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through Carry	1	1
SWAP A	Swap nibbles of A	1	1

Program Branching			
ACALL addr11	Absolute subroutine call	2	4
LCALL addr16	Long subroutine call	3	5
RET	Return from subroutine	1	6
RETI	Return from interrupt	1	6
AJMP addr11	Absolute jump	2	4
LJMP addr16	Long jump	3	5
SJMP rel	Short jump (relative address)	2	4
JMP @A+DPTR	Jump indirect relative to DPTR	1	4
JZ rel	Jump if A equals zero	2	2/4
JNZ rel	Jump if A does not equal zero	2	2/4
CJNE A, direct, rel	Compare direct byte to A and jump if not equal	3	3/5
CJNE A, #data, rel	Compare immediate to A and jump if not equal	3	3/5
CJNE Rn, #data, rel	Compare immediate to Register and jump if not equal	3	3/5
CJNE @Ri, #data, rel	Compare immediate to indirect and jump if not equal	3	4/6
DJNZ Rn, rel	Decrement Register and jump if not zero	2	2/4
DJNZ direct, rel	Decrement direct byte and jump if not zero	3	3/5
NOP	No operation	1	1

Beachte: Der Befehl **DJNZ Rn, rel** benötigt **vier** Maschinenzyklen (**MZ**)! Je nach Oszillatoreinstellung (OSCICN) ergeben sich damit unterschiedliche Zeiten für die Abarbeitung dieses Befehls. Ein Maschinenzyklus (MZ) = 1/f

MOV OSCICN,#80h → f=1,5MHz; MOV OSCICN,#81h → f=3MHz; MOV OSCICN,#82h → f=6MHz; MOV OSCICN,#83h → f=12MHz (Default)

8. Hochsprache C (ANSI-C)

1. Datentypen, Variable, Konstante, Operatoren und Ausdrücke

Datentypen

Datentyp	Größe	Wertebereich
bit	1 Bit	0 oder 1
signed char	1 Byte	-128 bis +127
unsigned char	1 Byte	0 bis 255
signed int	2 Byte	-32768 bis + 32767
unsigned int	2 Byte	0 bis 65535
signed long	4 Byte	-2147483648 bis +2147483647
unsigned long	4 Byte	0 bis 4294967295
float	4 Byte	±1,176E-38 bis ±3,40E+38
pointer	1-3 Byte	Adresse einer Variablen
FILE		Dateizeiger

Operatoren und ihre Priorität

Mathematische Operatoren		Priorität	Verhältnis und logische Operatoren	
++ --	Inkrement dekrement	Höchste	!	NOT
-	monadisches Minus (Vorzeichen)		> >= < <=	Größer größer gleich kleiner kleiner gleich
* / %	Mal Div mod		== !=	Gleich ungleich
+ -	Plus minus		&&	AND
		niedrigste	||	OR
Bitweise Operatoren				
& |	UND ODER			
^	EXOR			
~	Einerkomplement			
<< >>	schieben nach **links** ; …. - nach **rechts**			

Beispiele

X = 10:
Y = ++X → Y = 11 ;Y = X++ → Y = 10
Y = --X → Y = 9 ; Y = X-- → Y = 10

Y = Y >> 1 schiebe Y um 1 nach **rechts**

2. Aufbau eines C-Programms

C bietet zwei Möglichkeiten zur Kommentareingabe :

a) // Kommentar für eine Zeile
b) /* Kommentar für einen Block von einer
oder mehreren Zeilen */

```
{       Anfang eines zusammengehörigen Befehlsblocks      (begin)
}       Ende eines zusammengehörigen Befehlsblocks (end)
```

// INCLUDE Files: → Compileranweisung über zusätzliche Quellcodes mit Funktionen und Deklarationen

```
#include <reg51xx.h>      // Registerdeklaration zum 5051xx
#include <math.h>         // Einbinden mathematischer Funktionen
```

// Konstantendeklaration → Ablage im Programmspeicher → Wert im Programm nicht änderbar

```
#define ANZAHL 10         // ANZAHL entspricht 10
#define TRUE 1            // TRUE entspricht 1
```

//Deklaration globaler Variablen → Ablage im Datenspeicher → Wert im Programm änderbar

```
int i = 8, j = 3, k;          /* Zählvariablen i , j und k mit den Anfangswerten  8 für i und 3 für j */
char TASTE;                   // 1 Byte große Variable von 0-255
signed long 4BYTE;            // 4 Byte große Variable von 0 – 2^32
char code *text = "Hallo";    /* Textstring mit 5 Bytes (+ 0 als Stringende) im Programmspeicher*/
char bdata schieb;            // 1 Byte-Variable im bitadressierbaren Speicherbereich
```

//Deklaration von Funktionen

```
Typ Funktion_1(Typ Parameter1, Typ Parameter2 )
                              // Als Typ kann jeder Datentyp stehen. (void ⇒ keine Typzuweisung)
{                             // Begin von Funktion_1
                              // lokale Variablendeklaration;
                              // Befehlsfolge;
}                             // Ende von Funktion_1

Typ Funktion_2( )
{                             // Begin von Funktion_2
                              // lokale Variablendeklaration;
                              // Befehlsfolge;
}                             // Ende von Funktion_2
```

// Hauptprogramm - auch Hauptfunktion main()

```
main( )
{                             // Begin des Hauptprogramms
                              // lokale Variablendeklaration;
                              // Befehlsfolgen;
}                             // Ende des Hauptprogramms
```

Hinweis: Folgende Compileranweisungen sollten zu Beginn des Programms erscheinen.

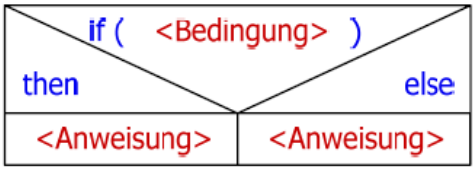
```
#include <C_8051F340.h>          //Erweiterte SFR Zuordnungen
#include <Einstellungen.h>       //beinhaltet die Funktion Grundeinstellungen
```

In der **main** wird dann gleich zu Beginn die Funktion Grundeinstellungen() aufgerufen!

3 Befehle zur Steuerung des Programmflusses

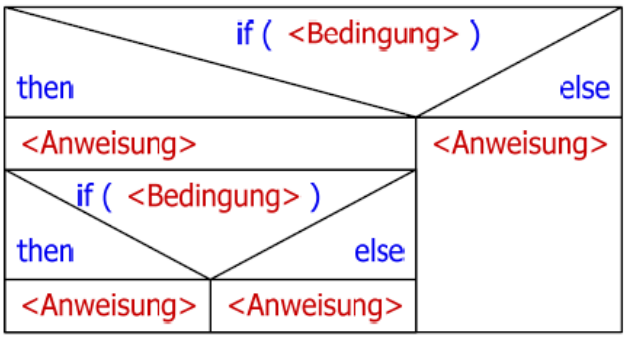
3.1 If , else

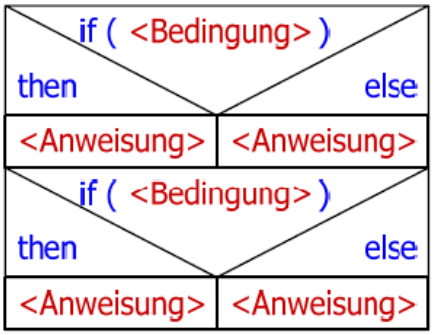
```
if (Bedingung)
  {
   Befehlsfolge für wahre Bedingung ;
  }
else
  {
   Befehlsfolge für falsche Bedingung;
  }
```



Anmerkungen:

- Die Befehlsfolgen selbst können wiederum If-Anweisungen sein oder verschachtelte if-Anweisung
- Die else- Anweisung ist nicht zwingend notwendig





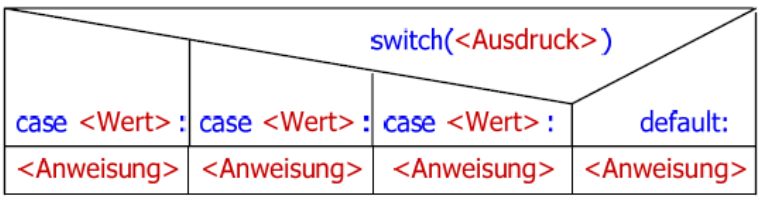
3.2 switch

```
switch (Ausdruck)
        {
        case  Konstante 1:
                Befehlsfolge 1;
                break;

        case  Konstante 2:
                Befehlsfolge 2;
                break;

        case  Konstante X:
                Befehlsfolge X;
                break;

        default:        {
                Befehlsfolge;
                }
        }
```



Anmerkungen:

- Die Befehlsfolgen selbst können wiederum If-Anweisungen oder switch-Anweisungen sein.
- Die default-Anweisung ist nicht zwingend notwendig.
- Der break-Befehl bricht die switch-Anweisung ab.

3.2 Schleifen

3.2.1 Die for-Schleife

```
for ( Initialisierung ; Bedingung ; Veränderung)
{
Befehlsfolge;
}
```

Anmerkungen:

- geeignet für Schleifen, bei denen die Anzahl der Durchläufe bekannt ist.
- Der Körper der Schleife kann auch leer sein (for (; ;)) jedoch die Semikolon müssen bleiben.
- Die Initialisierung legt die Startwerte der Variablen fest. Es können dabei auch mehrere Variablen mit Komma getrennt initialisiert werden.
- Ist die Bedingung erfüllt (TRUE) wird die Befehlsfolge bearbeitet
- Nach der Befehlsfolge bestimmt die Veränderung, wie die Variablen verändert werden.

Beispiel:

```
for ( i = 0, j = 8 , z = 0 ; i  < j ; i++ , j - = 2 )
  {
  z = i + j;
  }                    //   z = 6
```

3.2.2 Die while-Schleife (Kopfgesteuert)

```
while ( Bedingung )
        {
        Befehlsfolgen;
        }
```

Anmerkungen:

- Ist die Bedingung nicht erfüllt, also FALSE, dann wird die Befehlsfolge nicht bearbeitet
- Die Befehlsfolgen werden solange wiederholt, solange die Bedingung erfüllt bzw. WAHR ist
- **Endlosschleife** mit **while(1)** oder while(TRUE)
- Controller-Programme werden normalerweise mit einem Hardwarereset abgebrochen. Daher fangen die Programme für den µC meistens mit **while(1) {** an

3.2.3 Die do-while-Schleife (Fußgesteuert)

```
do {
    Befehlsfolgen;
    } while ( Bedingung );
```

Anmerkungen:

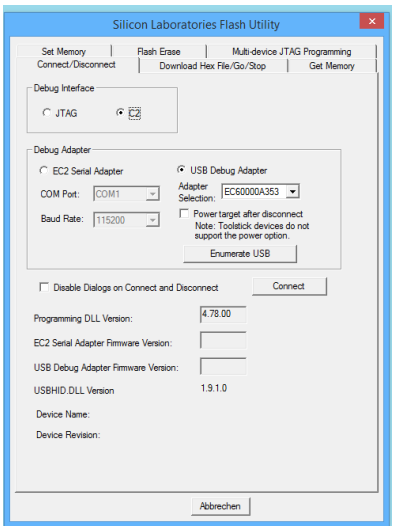
- Die Befehlsfolge wird mindestens einmal bearbeitet, auch wenn die Bedingung nicht erfüllt ist.
- Die Befehlsfolgen werden solange wiederholt, wie die Bedingung erfüllt bzw. WAHR ist
- Endlosschleife mit while(1) oder while(TRUE)

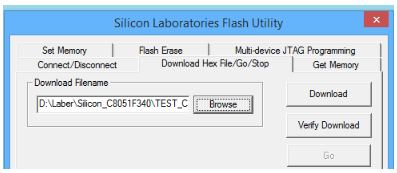
4. Der break-Befehl

In einer Schleife beendet der break-Befehl diese, die Programmsteuerung geht direkt an die auf die Schleife folgenden Befehle über.

9. Programmierung des C8051F340

Aufruf des Programms „Flash Programming Utility“ von Silicon Labs





Danach öffnet sich folgendes Fenster (eventuell muss vorher ein Update durchgeführt werden).

In der Anzeige USB Debug Adapter erscheint die Seriennummer des Programmieradapters (hier EC6000A353)

Mit **Connect**, wird der Programmer aktiviert (grüne LED bei Run/Stop des Programmers leuchtet).

Danach im Menüpunkt Download Hex File/Go/Stop im Fenster Browse das Intel-Hex File reinladen und mit **Download** den Programmiervorgang starten.

Mit der anschließenden Betätigung des Reset-Tasters **K3-RST** auf dem Board oder der Betätigung des **Go-Button** wird das Programm gestartet.

Beachte: Um den Controller erneut zu programmieren muss vorher das Programmiergerät Disconnected und anschließend wieder Connected werden!

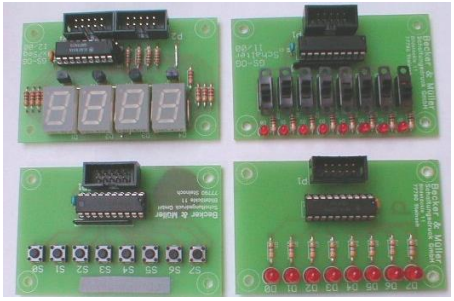
10. Software

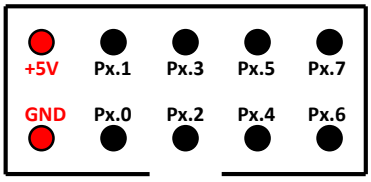
Entwicklungsumgebungen für den C8051F340 Controller:

- RIDE 7 von Raisonance (Assembler und C-Compiler bis 8KByte frei!)
- Keil µVision 5 (Assembler und C-Compiler bis 2KByte frei!)
- Simplicity Studio 4 von Silicon Labs (Assembler und C-Compiler unbegrenzt!)

11. Peripherie

Bei der Nutzung externer HGS-Peripherieplatinen ist folgende Anschlussbelegung zu beachten:





Ansicht von oben auf die Pins

12. Quellenangabe

Sämtliche Darstellungen sind dem Datenblatt C8051F340 von Silicon Labs (276 Seiten) entnommen.

Die Inhalte von Kapitel 8 sind der ehemaligen Formelsammlung 1.5.2 Informationstechnik entnommen.