10. Die Timer / Counter - Baugruppen

10.1. Überblick

Der Mikrocontroller 80C535 enthält **3 verschiedene Timer**, nämlich: **Timer 0** **Timer 1** **Timer 2**

Von den "Vorfahren" (Typen 8031, 8032, 8051, usw.) wurden die beiden **Timer 0 und 1** direkt übernommen. Jeder enthält ein 16 Bit - Register, das aus der Reihenschaltung von 2 Registern zu je 8 Bit zusammengesetzt ist. Je nach Wunsch können diese Register als Frequenzzähler, als Ereigniszähler oder als Zeitgeber betrieben werden.

Vorsicht: Der Timer 2 dagegen ist eine sehr komplexe und aufwendige Schaltung, die insgesamt 12 Register enthält! Sie hat mit dem "TIMER 2" der Vorfahren nichts mehr zu tun -- deshalb sind auch die alten Programme nicht mehr kompatibel und es ist ein eigenes Kapitel dafür nötig.

10.2. Arbeitsweise von Timer 0 und 1

Zur Einstellung der verschiedenen Betriebsarten dient das Register

"Timer modus" **TMOD** (Adresse 088H im internen RAM):

Bits:	Gate	C/#T	M1	M0	Gate	C/#T	M1	M0
	(Kontrolle für Timer 1)				(Kontrolle für Timer 0)			

Bedeutung:

a) **Gate**
Wird hier eine **Null** hineingeschrieben, so lässt sich der entsprechende Zähler / Timer **nur** durch das passende **Timer-Run-Bit (TR0 oder TR1) im TCON - Register starten oder stoppen**.
Steht das Gate - Bit auf **Eins**, so läuft der Zähler nur los, wenn **TR-Bit und der zugehörige "externe Interrupt - Pin"** -- z. B #INT0 -- gleichzeitig auf HIGH liegen. (Damit lasse sich sehr einfach Impulsbreiten bestimmen).

b) **C / #T** (Umschaltung zwischen Timer- und Zählerbetrieb)
Setzt man dieses Bit, so wird die Schaltung zum **Zähler.**
Löscht man dieses Bit, so arbeitet die Baugruppe als **Timer**. (Dem Eingang wird hierbei intern eine Taktfrequenz von 1 MHz zugeführt. Beim Überlauf wird das zugehörige Timer-Flag gesetzt).

c) **M0 und M1**
Mit diesen beiden Bits wird der "Modus" des Zählers gemäß folgender Tabelle eingestellt:

Mode	M1	M0	Wirkung
Null	0	0	13 Bit –Timer: das Low - Byte wird als 5-Bit-Vorteiler für das High - Byte benützt (Überrest aus uralten MCS-48-Zeiten.....)
Eins	0	1	**Normalbetrieb als 16 - Bit - Zähler/ Timer!!**
Zwei	1	0	**Reload - Betrieb** (= -8-Bit- Timer oder Zähler mit Selbstnachladung. Beim Überlauf wird der Inhalt des High - Bytes wieder als neuer "Anfangswert" ins Low - Byte geschrieben und von diesem Wert aus weitergezählt).
Drei	1	1	Recht komplizierte Wirkung: a) Schaltet man Timer 1 in diesen Modus, so bleibt er einfach stehen und speichert seinen alten Inhalt. b) Schaltet man Timer 0 in diesen Modus, so werden aus seinen beiden Bytes zwei von einander unabhängige 8-Bit-Timer.

Diese verschiedenen Betriebsarten sind auf den beiden folgenden Blättern ausführlich dargestellt.

Im Timer – Control - Register **TCON** finden sich in der **oberen Hälfte** diejenigen Bits, die

a) den **entsprechenden Timer starten bzw. stoppen**
b) beim **Überlauf gesetzt** werden und damit zum Auslösen eines entsprechenden **Interrupts** für die Weiterverarbeitung des Zählerstandes verwendet werden können.
c) Die **unteren 4 Bits** dienen zur **Interrupt - Programmierung von #INT 0 und #INT1.**

TCON - Register, Adresse 088H im internen RAM, bitadressierbar

| **Bits:** | TF1 | TR1 | TF0 | TR0 || IE1 | IT1 | IE0 | IT0 |

Erklärung:
a) TF0 und TF1 sind die erwähnten **Überlauf-Flags.**
b) TR0 und TR1 sind die "**Timer - Run - Bits**" zum **Starten / Stoppen** der Zähler/Timer-Baugruppen**.**
c) IE0 und IE1 heißen **"Interrupt - Edge-Flags**". Sie werden gesetzt, sobald die CPU eine **auslösende Impulsflanke** bei #INT0 bzw. #INT1 **erkannt** hat. Damit erzwingt man bei der nächsten Abfrage der Flags durch die CPU den Sprung in die Interrupt - Routine. Bei diesem Sprung werden sie dann wieder gelöscht.
d) IT0 und IT1 sind die "**Interrupt - Type - Flags**". Soll der Controller auf eine **negative** Impuls***flanke*** am Interrupt-Eingang reagieren, dann muss es gesetzt werden. Steht das Flag auf LOW, dann löst das Erreichendes **LOW-Pegels** am #INT0- bzw. #INT1-Pin den Interrupt aus.

Anhang: Übersicht für die Abläufe bei Modus 0 .... Modus 3

a) Modus 0: 13 Bit – Timer – Counter

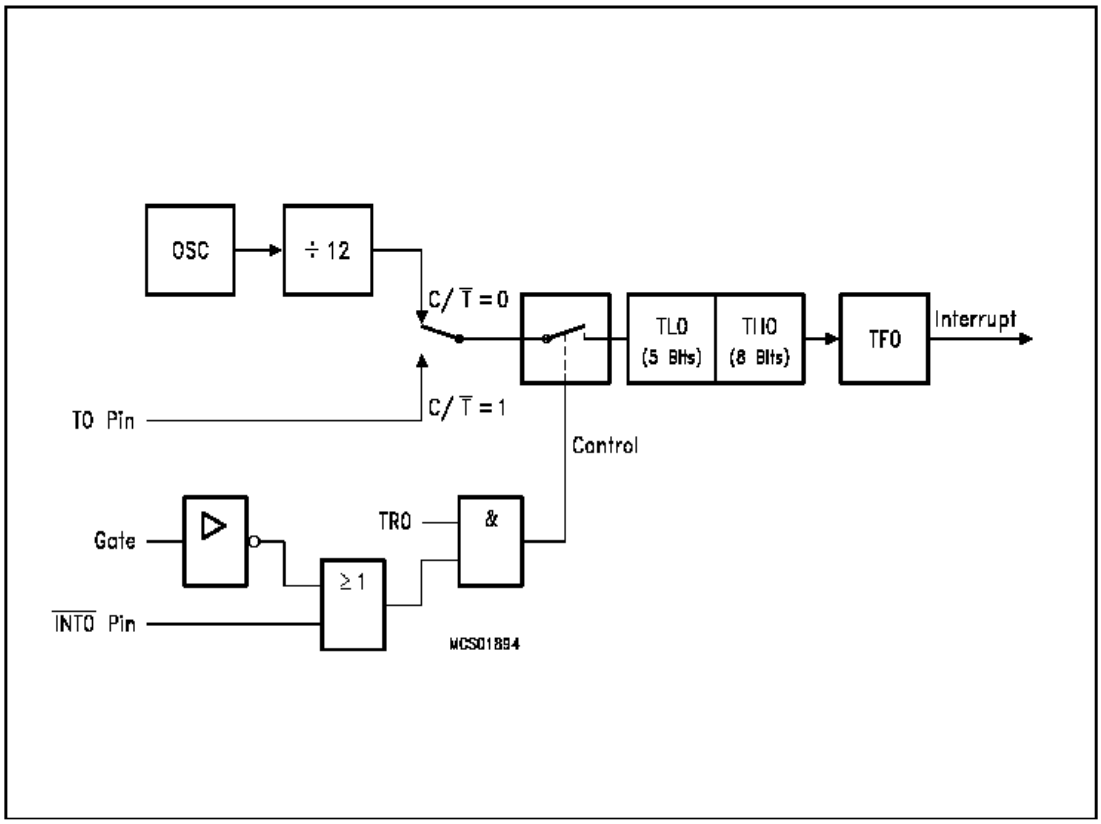

Figure 7-21
Timer/Counter 0, Mode 0: 13 Bit-Timer/Counter
The same applies to timer/counter 1

b) Modus 1: 16 Bit – Timer oder Counter

7.3.2 Mode 1

Mode 1 is the same as mode 0, except that the timer register is run with all 16 bits. Mode 1 is shown in figure 7-22.

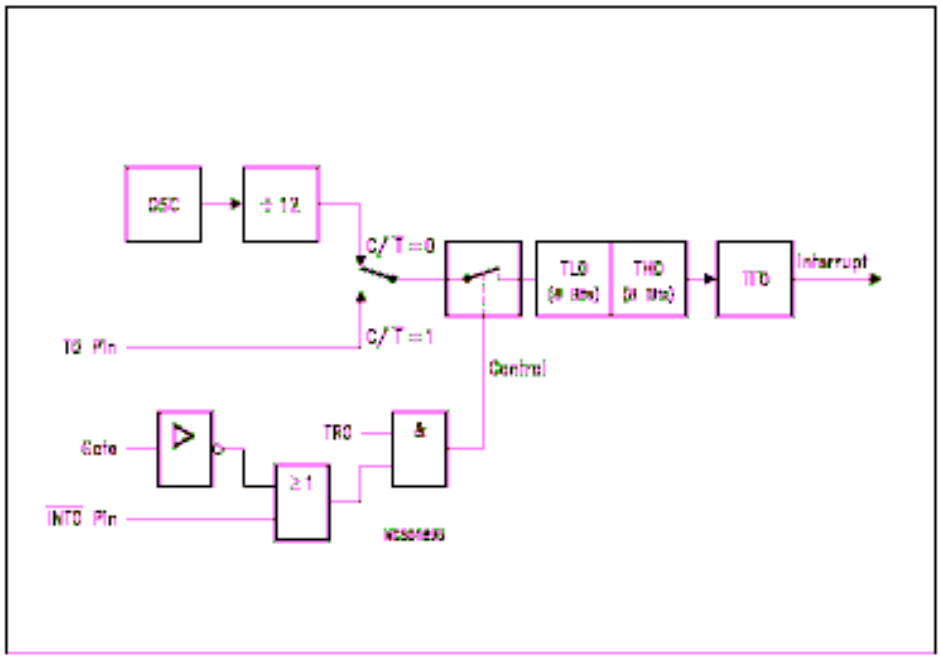

Figure 7-22
Timer/Counter 0, Mode 1: 16-Bit Timer/Counter
The same applies to timer/counter 1

c) Modus 2: 8 Bit – Autoreload – Timer oder Counter

7.3.3 Mode 2

Mode 2 configures the timer register as an 8-bit counter (TL0) with automatic reload, as shown in figure 7-23. Overflow from TL0 not only sets TF0, but also reloads TL0 with the contents of TH0, which is preset by software. The reload leaves TH0 unchanged.

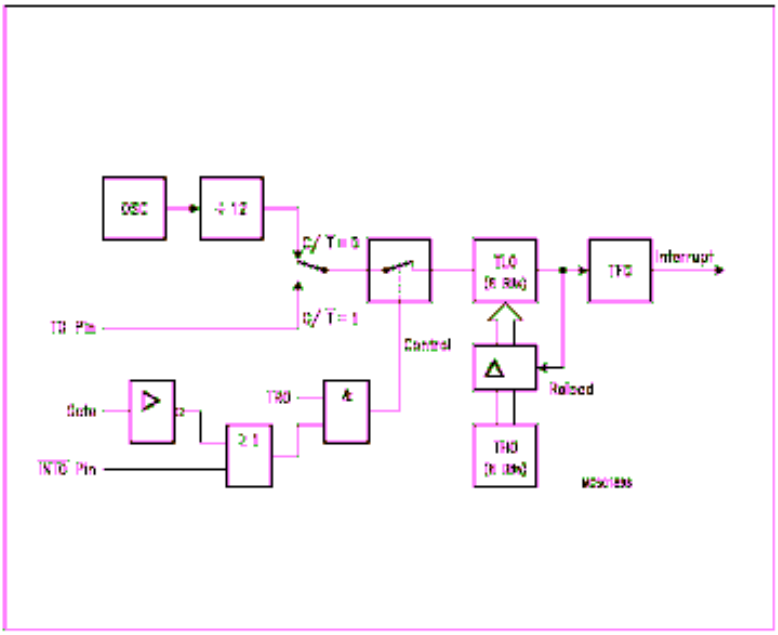

Figure 7-23
Timer/Counter 0, Mode 2: 8-Bit Timer/Counter with Auto-Reload
The same applies to timer/counter 1

d) **Modus 3: Timer 1 steht still und Timer 0 wird in zwei 8 Bit- Counter / Timer aufgeteilt**

7.3.4 Mode 3

Mode 3 has different effects on timer 0 and timer 1. Timer 1 in mode 3 simply holds its count. The effect is the same as setting TR1 = 0. Timer 0 in mode 3 establishes TL0 and TH0 as two separate counters. The logic for mode 3 on timer 0 is shown in **figure 7-24**. TL0 uses the timer 0 control bits: C/$\overline{T}$, GATE, TR0, $\overline{INT0}$, and TF0. TH0 is locked into a timer function (counting machine cycles) and takes over the use of TR1 and TF1 from timer 1. Thus, TH0 now controls the "timer 1" interrupt.

Mode 3 is provided for applications requiring an extra 8-bit timer or counter. When timer 0 is in mode 3, timer 1 can be turned on and off by switching it out of and into its own mode 3, or can still be used by the serial channel as a baud rate generator, or in fact, in any application not requiring an interrupt from timer 1 itself.

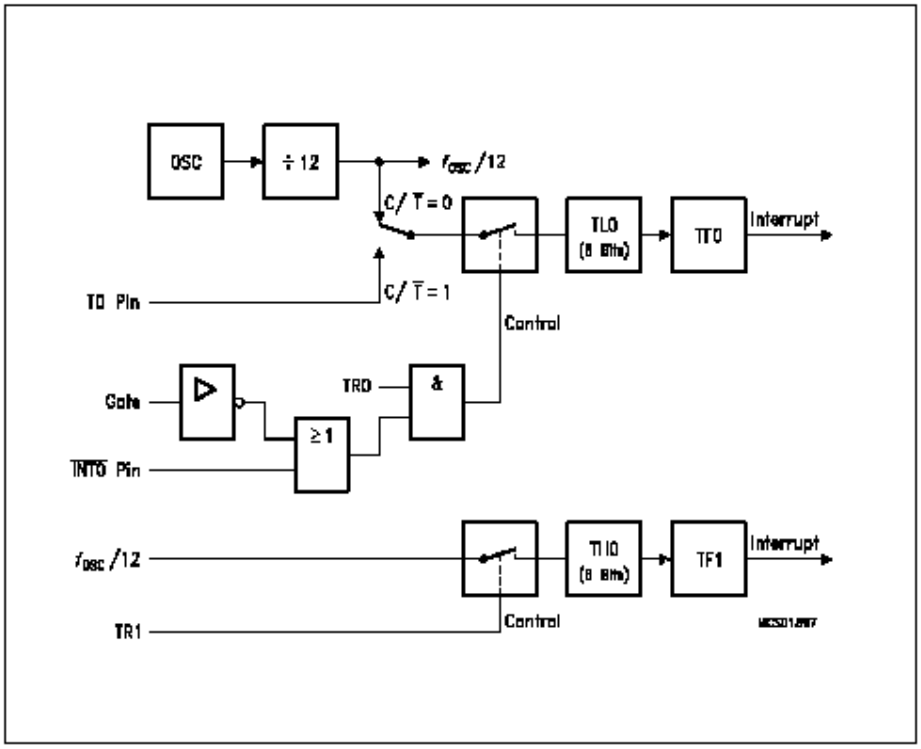

Figure 7-24
Timer/Counter 0, Mode 3: Two 8-Bit Timers/Counters

10.3. Programmierbeispiel: Erzeugung eines 2 kHz-Tones (Reload-Modus + Interrupt)

Am Portpin P1.1 soll ein 2 kHz-Rechtecksignal zur Verfügung stehen. Timer 1 soll dazu im Reload-Modus unter Verwendung des Überlauf - Interrupts benützt werden. Schreiben Sie das Programm erst in Assembler, dann in C.

Assembler- Lösung:

```
$nomod51
$include(REG515.INC)

OFFSET      EQU 4000h
ausgang     equ    P1
?Stack      SEGMENT IDATA
            RSEG ?STACK
            DS 5

CSEG AT 0+OFFSET
            ljmp START

PIEPSEN     SEGMENT CODE
            RSEG PIEPSEN

START:      mov SP,#?STACK-
            mov TMOD,#00100000b   ;Timer 1 auf Mode 2 und Timerbetrieb schalten
            mov TL1,#5            ;Startwert ist 5
            mov TH1,#5            ;Reloadwert ist 5
            setb EAL              ;Interrupts einschalten
            setb ET1              ;Timer 1-Interrupt einschalten
            clr TF1               ;vorsichtshalber Timer 1-Überlaufflag löschen
            SETB TR1              ;Timer 1 starten
            jmp $                 ;Endlosschleife
;-------------------------------------------------------------------------------------------------------------
;Interruptverschiebung und Interruptbearbeitung:
;-------------------------------------------------------------------------------------------------------------
CSEG at 01Bh + OFFSET             ;Interruptvektor nach 4000h + 1Bh verschieben
            ljmp timer1int        ;Verbindung zum neuen Vektor herstellen

T1INT_code_seg      SEGMENT CODE
                    RSEG T1int_code_seg

timer1int:  clr TF1               ;Überlaufflag löschen
            cpl ausgang           ;Portpin P1.1 invertieren
            reti                  ;raus aus der Interrupt - Service-Routine

end
```

C – Lösung für die Erzeugung eines 2 kHz – Tones mit Timer 1 und Interrupt-Steuerung

```
/*-------------------------------------------------------------------------------------------------------------------
Programmbeschreibung
-------------------------------------------------------------------------------------------------------------------
Name:         tim1int.c
Funktion:     Der Timer1 wird im Reloadbetrieb (Mode 2) zur Erzeugung eines 2kHz-Tones am Portpin
              P1.0 verwendet. Dabei wird auch der Timer1 - Interrupt eingesetzt.

Datum:        01. 01. 2005
Autor:        G. Kraus

-------------------------------------------------------------------------------------------------------------------
Deklarationen und Konstanten
-------------------------------------------------------------------------------------------------------------------*/
#include <reg515.h>
#include <stdio.h>

sbit ausgang=0x90;                      // Portpin P1.0 als Ausgang
/*-------------------------------------------------------------------------------------------------------------------
Hauptprogramm
-------------------------------------------------------------------------------------------------------------------*/
void Timer_1_ISR(void);                 // Prototyp - Anmeldung

void main(void)
{
    TMOD=0x20;                          // Timer1 im Reload-Betrieb (Mode 2)
    TL1=0x05;                           // Start-Wert ist 5
    TH1=0x05;                           // Reload-Wert ist 5
    EAL=1;                              // Interrupts freigeben
    ET1=1;                              // Timer1 - Interrupt freigeben
    TF1=0;                              // Timer1-Überlaufflag vorsichtshalber löschen
    TR1=1;                              // Timer1 einschalten

    while(1);                           // Endlosschleife
}
/*-------------------------------------------------------------------------------------------------------------------
Interrupt-Routine
-------------------------------------------------------------------------------------------------------------------*/
void Timer_1_ISR(void) interrupt 3          // Timer1 hat Interruptvektor 1B
{                                           // das ergibt Interrupt-Nummer 3
        TF1=0;                              // Überlaufflag löschen
       ausgang= ~ausgang;                   //Ausgang invertieren
}
```

Aufgabe: Schreiben Sie das Programm so um, dass a) Timer 0 zum Einsatz kommt und
b) nun ein Ton mit 10 Hz erzeugt wird!