Kapitel 14

Timer

Timer

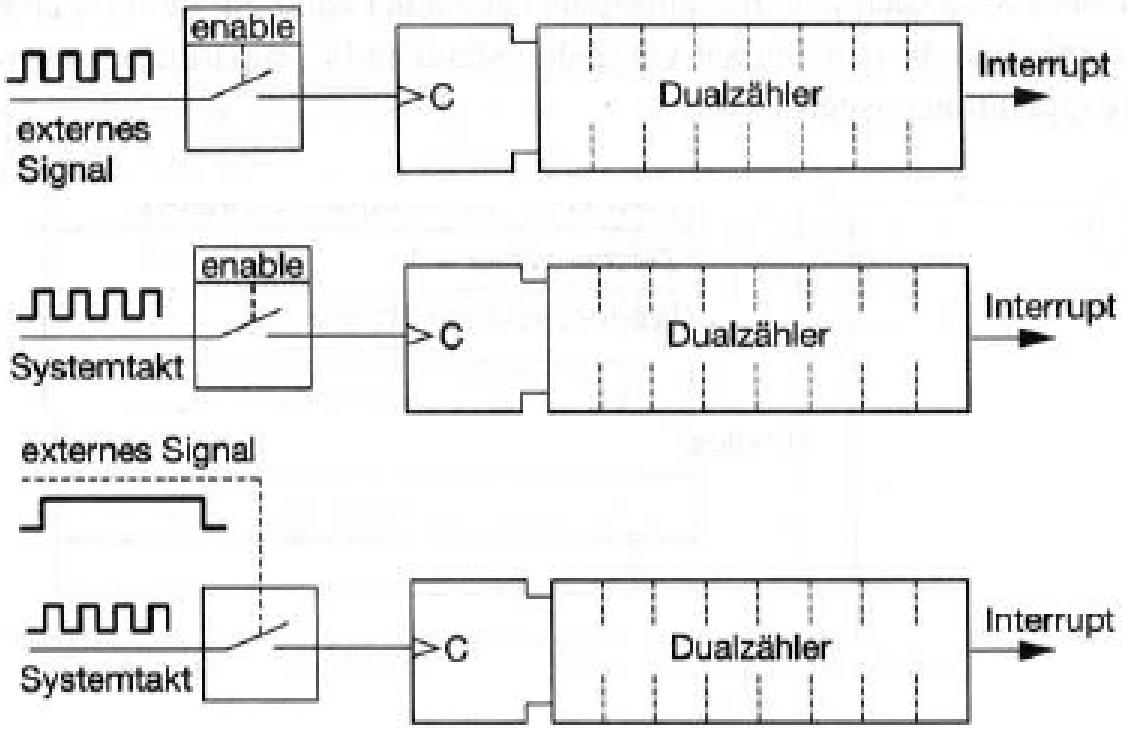
- Ein Timer ist eine Hardware-Einrichtung, die nach einer vorgegebenen Zeitspanne einen Interrupt erzeugt, (wie ein Wecker).
- Ein Timer ist im wesentlichen ein Zähler, der durch Pulse aus einer sehr regelmäßigen Signalquelle hochgezählt wird.
  - Das Hochzählen erledigt der Timer autonom (ohne Programmunterstützung).
  - Während der Timer läuft, wird das Programm normal weiter ausgeführt.
  - Das Aufsetzen und Starten des Timers sowie die Aktionen, die durchgeführt werden nachdem der Timer abgelaufen ist, sind Aufgabe des Programms.
- Der 80C515C Prozessor hat drei integrierte Timer.
  - Timer 0 und Timer 1 sind relativ einfache baugleiche Timer.
  - Timer 2 ist deutlich komplizierter und speziell für die Steuerung einer Pulsweitenmodulation ausgelegt.

Funktionen der 80c515c Timer 0 und 1

Zähler

Timer

Impulslängenmessung



Quelle: Müller/Walz, Mikroprozessortechnik

- Das externe Signal wird an Port3.4 eingespeist.
- Bei einem Oszillatortakt von 10 Mhz und 6 Oszillatorzyklen pro Maschinenzyklus wird im Timermodus der Dualzähler alle 0,6 µs inkrementiert.
- Mit seinem 16 Bit großen Dualzähler kann der Timer mit Zeiten bis zu 65536 * 0,6 µs = 39,321 ms umgehen.

Timer 0 und 1 (8051 Handbuch Seite 6-21 ff.)

- Vier Betriebsarten : Mode 0, 1, 2 oder 3
  - Mode 0 und 1 sind bis auf die Dauer der messbaren Zeit gleich.
  - Mode 2 wird eingesetzt, wenn sehr exakt gemessen werden muss.
  - Mode 3 ist sehr speziell (Bei Interesse sind Details im User's Manual zu finden).
- Steuerbits

Special Function Register TCON (Address 88_H) **Reset Value : 00_H**

Bit No.	MSB							LSB
	7	6	5	4	3	2	1	0
	$8F_H$	$8E_H$	$8D_H$	$8C_H$	$8B_H$	$8A_H$	89_H	88_H
88_H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TCON

Special Function Register TMOD (Address 89_H) **Reset Value : 00_H**

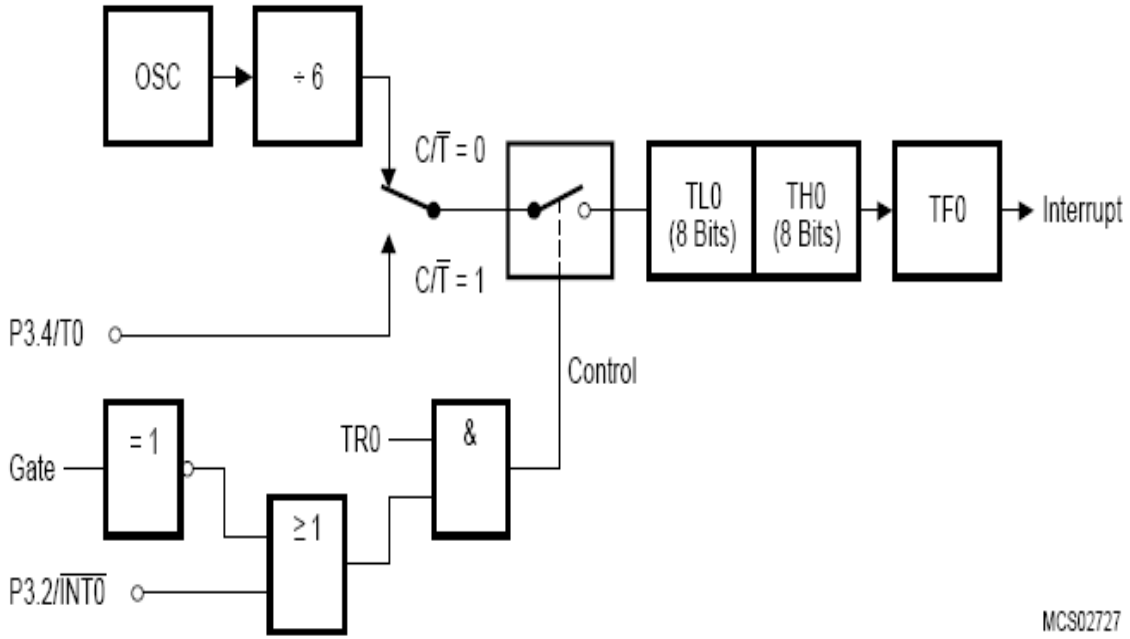
Bit No.	MSB							LSB
	7	6	5	4	3	2	1	0
89_H	Gate	C/$\overline{T}$	M1	M0	Gate	C/$\overline{T}$	M1	M0

TMOD

Timer 1 Control | Timer 0 Control

User's Manual p.6-23 ... 6-26

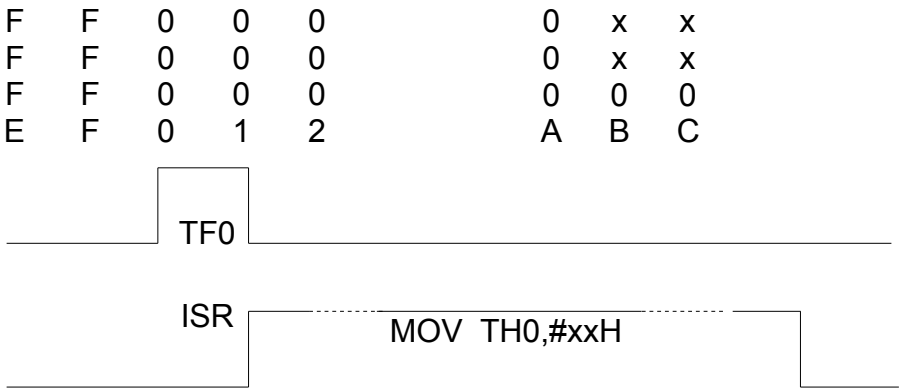
Timer 0 in Mod



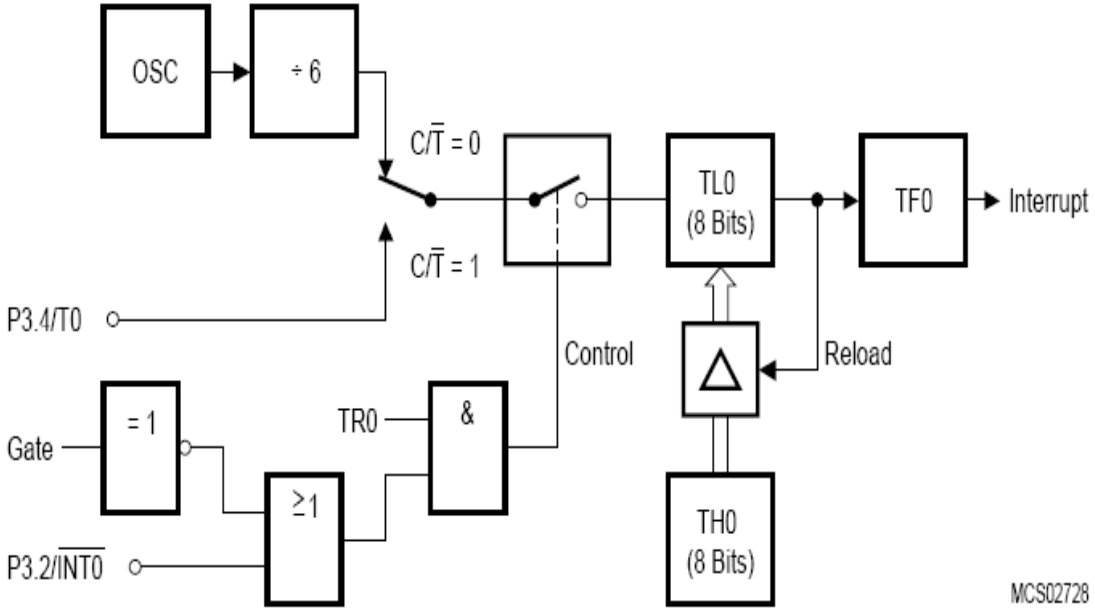
- **TH0** (SFR Adresse 8CH) und **TL0** (SFR Adresse 8AH) enthalten den 16-Bit Zähler für Timer 0.
  - Sie können per Programm gesetzt und auch ausgelesen werden.
- Steuerung mit Gate = 0
  - Aktivierung (TR0 =1) der Timer Run Leitung startet das Hochzählen, Deaktivierung (TR0 = 0) stoppt das Hochzählen.
  - C/$\overline{T}$ selektiert Timer- bzw. Zählerbetrieb.
- Steuerung mit Gate = 1
  - Kann zur Messung der Pulsbreite des an Port 3 Bit 2 (INT0) anliegenden Signals verwendet werden.
  - TR0 = 1 aktiviert die Messeinrichtung.
  - Das Hochzählen beginnt mit der Aktivierung von P3.2 und endet mit der Deaktivierung von P3.2.

Timer 0 in Mode 1 (Forts.)

- Wenn der Timer eingeschaltet ist (Schalter Control aktiviert), wird der Zähler hochgezählt.
  - Wenn der Zähler den Wert 0xFFFF erreicht, setzt der Timer das Timer Overflow Bit **TF0** (SFR Adresse 88H.5) und startet den Zähler wieder bei 0x0000.
  - Die Interrupt Steuerung kann so aufgesetzt werden (Interrupt enabled), dass das Timer Overflow Signal einen Timer 0 Interrupt erzeugt, wodurch dann die Timer 0 Interrupt Service Routine (**ISR**) ausgeführt wird.
  - In dieser Interrupt Routine könnte dann der Anfangswert im Zähler so verändert werden, dass sich eine gewünschte Zeitdauer bis zum nächsten Timer Interrupt ergibt.
    - Auf diese Weise lässt sich die Zeitdauer bis zum nächsten Interrupt beliebig einstellen.
  - Bei der Aktivierung der Timer 0 Interrupt Service Routine setzt das Steuerwerk auch das Timer Overflow Bit TF0 selbständig zurück.



Timer 0 in Mode 2 : Auto-Reload Mode



- Nur noch der 8-Bit Zähler in TL0 wird hochgezählt.
  - Maximal messbare Zeitdauer ist 256 * 0,6 = 153,6 us.
- Beim Übergang des Zählers in TL0 von 0xFF nach 0x00 lädt der Timer automatisch den Wert aus TH0 als neuen Startwert in den Zähler in TL0.
- Die restliche Steuerung wie in Mode 1.
- Dadurch dass das Aufsetzen des neuen Startwertes komplett von der Hardware gesteuert wird, arbeitet der Timer in Mode 2 zyklusgenau.
  - Es muss nur sicher gestellt sein, dass die Interrupt Service Routine irgendwann vor dem nächsten Timerüberlauf ausgeführt wird.

Beispiel : **Digitaluhr**

Auf einem Display soll die Uhrzeit angezeigt werden. Jede Sekunde ist also eine neue Anzeige erforderlich.

- Timer 0 soll alle 100 us einen Timer 0 Interrupt auslösen.
  - 100 us entspricht 166,667 Maschinenzyklen.
  - Reload Wert in TH0 = 256 -167 = 89 (Initialisierung)
  - Fehler ist 0,333 Maschinenzyklen = 0,2 us = 0,2 %
- Interrupt Service Routine zählt alle 100 us einen 2-Byte Zähler von 10.000 bis auf 0 herunter und setzt dann einen Indikator.
  - Der Indikator zeigt an, dass 1 Sekunde vergangen ist.
  - Der 2-Byte Zähler wird beim Programmstart und jeweils bei Erreichen des Wertes 0 auf 10000 zurückgesetzt.
  - Die Interrupt Service Routine muss innerhalb von 100 us nach dem Interrupt ausgeführt werden.
- Die Hauptroutine überprüft regelmäßig den Indikator. Wenn die Hauptroutine feststellt, dass der Indikator gesetzt ist,
  - inkrementiert sie die anzuzeigende Uhrzeit, und
  - setzt den Indikator zurück.
- Die eigentliche Anzeige der Uhrzeit und das Stellen der Uhr werden ignoriert.

Beispielprogramm : **Digitaluhr**

Datenspezifikationen :

```
MyData    SEGMENT    DATA
          RSEG       MyData
Uhrzeit:  DS         6          ;HH:MM:SS
Ctr10000: DS         2          ;Zähler für 10000 * 100 us
Stack:    DS         10

MyBits    SEGMENT    BIT
          RSEG       MyBits
SekTick:  DBIT       1          ;Indikator für eine vergangene Sekunde
```

Codespezifikationen :

```
                                ;Feste Einsprungadressen ab Adresse 0
          CSEG       AT   0     ;Einsprungadresse nach Reset
          LJMP       Init       ; zum Initialisierungscode
          ORG        000BH      ;Timer 0 ISR Einsprungadresse
          LJMP       T0ISR      ;zur Timer 0 ISR

MyCode    SEGMENT    CODE
          RSEG       MyCode
```

```
Init:                   ;Initialisierungscode
        MOV     SP,#Stack-1
        CLR     SekTick
        MOV     Ctr10000,#16        ;Little Endian Format
        MOV     Ctr10000+1,#39      ;39 * 256 = 9984
        MOV     TL0,#89             ;Zähler auf 89
        MOV     TH0,#89             ;Reloadwert auf 89
        MOV     TMOD,#0x02          ;Gate = 0, C/T = 0, Mode = 2
        SETB    TR0                 ;Starte den Timer
        SETB    ET0                 ;Enable Timer 0 Interrupts
        SETB    EA                  ;Enable All Interrupts

Main:                   ;Hauptroutine
        :
        JNB     SekTick,EndeTick
        :                           ;HH:MM:SS + 1
        CLR     SekTick
        :                           ;Uhrzeit anzeigen
EndeTick:
        :
        JMP     Main
```

```
T0ISR:                ;Timer 0 Interrupt Service Routine
        PUSH    ACC
        PUSH    PSW
        CLR     C               ;Lösche Carry Flag
        MOV     A,Ctr10000
        SUBB    A,#1            ;Dekrementiere Zähler Lo Byte
        MOV     Ctr10000,A
        MOV     A,Ctr10000+1
        SUBB    A,#0            ;Dekrementiere Hi By, wenn Carry Flag = 1
        MOV     Ctr10000+1,A
        JNZ     EndeISR         ;Sprung , wenn Zähler noch nicht 0
        MOV     A,Ctr10000
        JNZ     EndeISR         ;Sprung , wenn Zähler noch nicht 0
        SETB    SekTick         ;Zähler = 0 → Setze Indikator
        MOV     Ctr10000,#16    ;Initialisiere Zähler
        MOV     Ctr10000+1,#39
EndeISR: POP    PSW
        POP     ACC
        RETI                    ;Ende der ISR
        END
```

Timer 2

Viele unterschiedliche Betriebsarten. (Fokus in der Vorlesung auf Pulsweitenmodulation.)

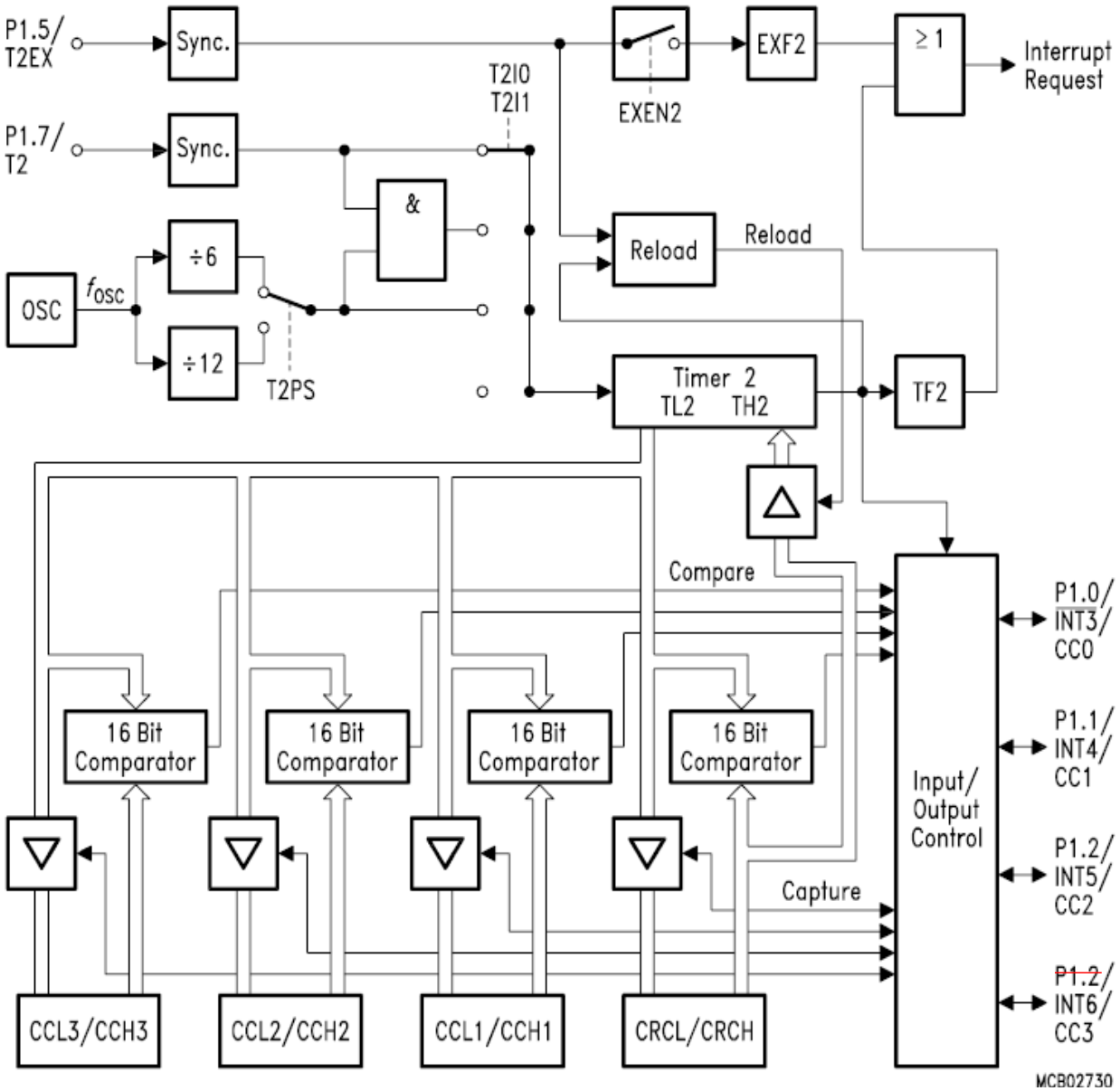


Figure 12 :

Timer 2 Block Diagramm

Fehler im Dokument; Korrektur : P1.3

Timer 2 (Forts.)

- 16 Bit Zähler in SFRs **TH2** (Adresse 0xCD) und **TL2** (Adresse 0xCC).
  - Bei Überlauf von 0xFFFF nach 0x0000 wird das Timer Overflow Flag **TF2** gesetzt.
  - Bei entsprechender Einstellung der Interrupt-Logik löst dies einen Timer 2 Interrupt aus.
- Timer 2 kann als Timer oder als Zähler verwendet werden
  - Im Timerbetrieb (T2I1 = 0, T2I0 = 1) kann das Hochzählen bei jedem Maschinenzyklus (T2PS = 0) oder bei jedem zweiten Maschinenzyklus (T2PS = 1) erfolgen.
  - im Zählerbetrieb (T2I1 = 1, T2I0 = 0) sorgen die Pulse am Port Pin P1.7 für das Hochzählen.
  - im Betrieb zum Ausmessen der Pulsbreite (T2I1 = 1, T2I0 = 1) werden die vom Oszillator stammenden Pulse gezählt, solange das Signal am Port Pin P1.7 aktiv ist.
- Vier Einheiten CC3, CC2, CC1 und CC0 können dazu verwendet werden, eine **Capture** Operation oder eine **Compare** Operation durchzuführen.
  - Im **Capture Mode** wird der Wert des Timers in TL2 und TH2 in die Register CCLx und CCHx kopiert.
    - Das Kopieren wird durch einen Impuls an P1.0, P1.1, P1.2 bzw. P1.3 getriggert.
  - Im **Compare Mode** wird der Wert des Timers in TL2 und TH2 mit dem Wert in den Registern CCLx und CCHx verglichen.
    - Sobald die beiden Werte gleich sind wird die Leitung P1.0, P1.1, P1.2 bzw. P1.3 aktiviert.

Mode Spezifikation

Special Function Register CCEN (Address $C1_H$) **Reset Value : 00_H**

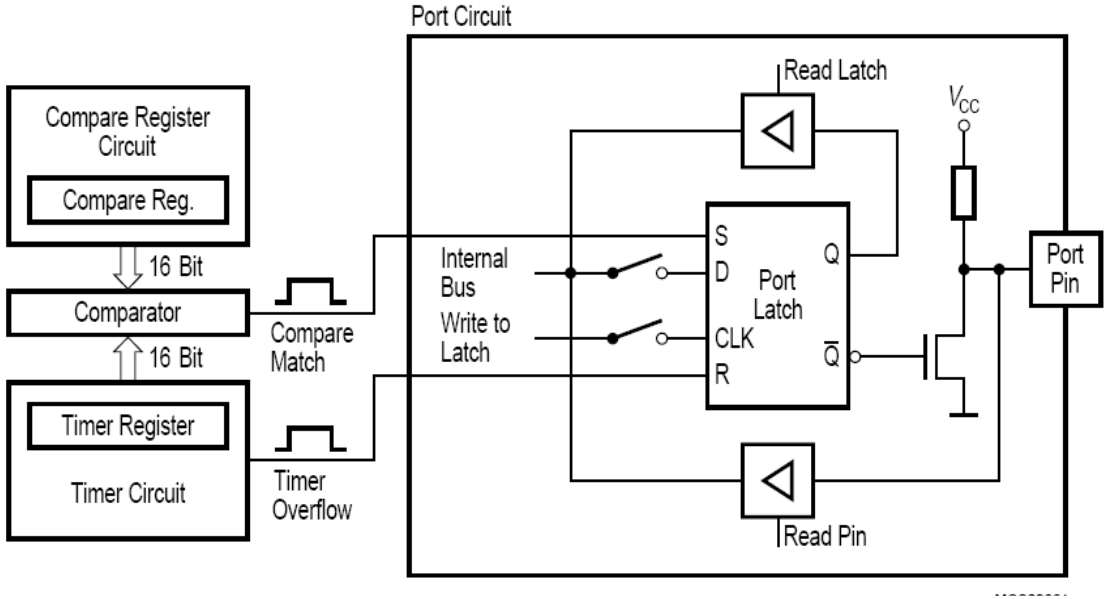
Bit No.	MSB 7	6	5	4	3	2	1	LSB 0	
$C1_H$	COCAH3	COCAL3	COCAH2	COCAL2	COCAH1	COCAL1	COCAH0	COCAL0	CCEN

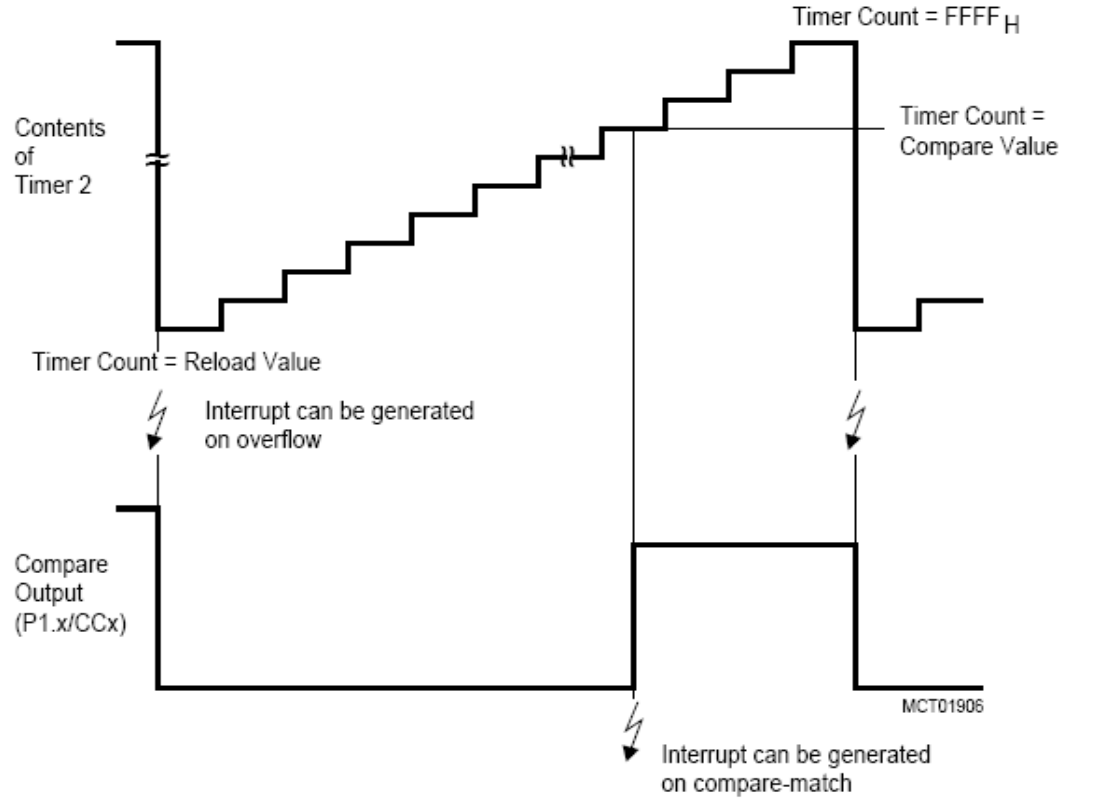
Bit	Function
COCAH3 COCAL3	Compare/capture mode for CC register 3

COCAH3	COCAL3	Function
0	0	Compare/capture disabled
0	1	Capture on rising edge at pin P1.3 / INT6 / CC3
1	0	Compare enabled
1	1	Capture on write operation into register CCL3

Timer 2 (Forts.)

- Als weitere Alternatve kann die CC0 Einheit zum Reload des Zählers verwendet werden.
- Mit T2R1 = 1 und T2R0 = 0 wird der **Reload Mode** eingestellt.
  - Dabei wird der Wert in den Registern CRCH (Adresse 0CBH) und CRCL (Adresse 0CAH) beim Überlauf des Zählers als neuer Startwert in die Zählerregister TH2 und TL2 übernommen.
  - Damit kann die Periodendauer für die Pulsweitenmodulation eingestellt werden.
- Mit z.B. COCAH1 = 0 und CACAL1 = 0 wird Capture Mode und Compare Mode ausgeschaltet.
- Mit z.B. COCAH1 = 0 und CACAL1 = 1 wird ein Capture Mode in der Einheit **CC1** eingestellt.
  - Wenn der Externe Interrupt 4 (Port Pin P1.1) aktiviert wird, kopiert die CC1 Einheit den Zählerwert in TH2 und TL2 in die Register CCH1 (Adresse 0C3H) und CCL1 (Adresse 0C2H).
- Mit z.B. COCAH3 = 1 und CACAL3 = 0 wird der Compare Mode in der Einheit **CC3** eingestellt.
  - Die CC3 Einheit vergleicht permanent den Zählerwert in TH2 und TL2 mit dem Wert in Register CCH3 (Adresse 0C7H) und CCL3 (Adresse 0C6H). Wenn die beiden Werte gleich sind, aktiviert die CC3 Einheit den Externen Interrupt 6 (setzt das Port P1.3 Flipflop).
  - Beim Überlauf des Zählers in TH2 und TL2 wird der Externe Interrupt 6 deaktiviert (das Port P1.3 Flipflop wird zurückgesetzt).
  - Damit kann das Signal für die Pulsweitenmodulation gesteuert werden, ohne dass irgendwelche Programmbefehle ausgeführt werden müssen.
  - Durch Änderung des Werts im Compare Register wird die Ein- und Ausschaltdauer der Pulse gesteuert.
- Mit z.B. COCAH1 = 1 und CACAL1 = 1 wird ein programmgesteuerter Capture Mode eingestellt.
  - Sobald das Programm irgend einen Wert in das Register CCL1 schreibt, kopiert die CC1 Einheit den Zählerwert in TH2 und TL2 in die Register CCH1 und CCL1.


Port Circuit
Compare Register Circuit
Compare Reg.
16 Bit
Comparator
16 Bit
Timer Register
Timer Circuit
Compare Match
Timer Overflow
Internal Bus
Write to Latch
S
D
CLK
R
Port Latch
Q
Q
Read Latch
Read Pin
V_{CC}
Port Pin


Timer Count = FFFF H
Contents of Timer 2
Timer Count = Compare Value
Timer Count = Reload Value
Interrupt can be generated on overflow
Compare Output (P1.x/CCx)
MCT01906
Interrupt can be generated on compare-match

Watchdog Timer

- Der Watchdog Timer dient dazu, den Prozessor in ernsten Fehlersituationen wieder zum Leben zu erwecken.
  - Typische Fehler, in denen der Watchdog Timer eingreift :
    - Das Programm hängt (läuft z.B. in einer engen Schleife)
    - Ein Hardware- oder Programmfehler stoppt den Prozessor
    - Der Oszillator streikt
  - Wenn der Watchdog Timer den Fehler entdeckt, forciert er einen **RESET**
    - Der Prozessor wird auf den Anfangszustand zurückgesetzt, und
    - Der Prozessor startet wieder bei Adresse 0x0000
    - Da die Speicherinhalte bei einem RESET nicht verändert werden, hat das Programm die Chance, die Ausführung wieder fortzusetzen (z.B. an einem Checkpoint)
- Der Watchdog Timer enthält einen 15 Bit Zähler, Bits 14 … 8 in WDTH und Bits 7 … 0 in WDTL
  - Der Zähler wird mit einer Frequenz von f/6 : 2 oder f/6 : 2 :16 inkrementiert.
    - Bei einem 8051 mit 10 Mhz Oszillator also alle 1,2 us oder alle 19,2 us.
  - Wenn der Zähler den Wert 0x7FFFC erreicht, wird der RESET Request forciert.
  - Um einen RESET zu vermeiden, muss das Programm periodisch einen Watchdog Timer Refresh durchführen
    - Ein Refresh wird durch das Setzen der Bits WDT und SWDT eingeleitet
      - Die Bits müssen in zwei aufeinanderfolgenden Maschinenzyklen gesetzt werden, z.B. durch zwei SETB Befehle
    - Dadurch werden die 7 Bits des Registers WDTH aus dem SFR Register WDTREL übernommen

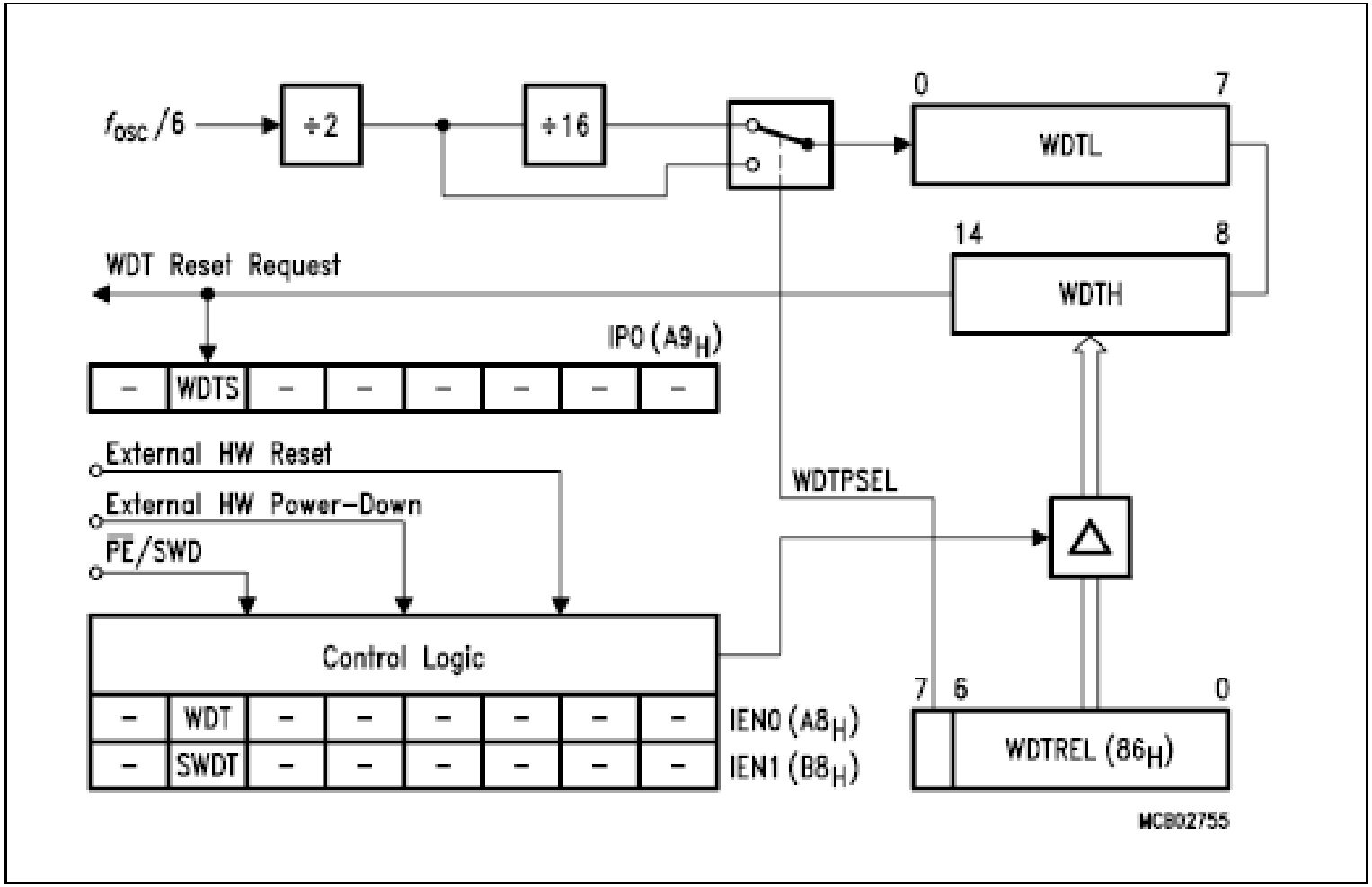


Figure 8-1
Block Diagram of the Programmable Watchdog Timer

- WDTREL Bit 7 bestimmt die Frequenz, mit der der Zähler inkrementiert wird

Table 8-1
Watchdog Timer Time-Out Periods (WDTPSEL = 0)

WDTREL	Time-Out Period		Comments
	f_{osc} = 6 MHz	f_{osc} = 10 MHz	
00_H	65.535 ms	39.322 ms	This is the default value
80_H	1.1 s	0.63 s	Maximum time period
$7F_H$	512 µs	307 µs	Minimum time period

- Der 8051 Watchdog Timer kann deaktiviert werden, in dem man den Prozessorpin $\overline{PE}$/SWD an Masse legt
  - Im Laborkoffer ist dieser Pin an Masse gelegt, d.h. der Watchdog Timer ist deaktiviert.
  - Im Keil Simulator kann man den Watchdog Timer deaktivieren über *Peripherals* → *Timer* → *Watchdog*
  - Im Laborprojekt kann man Watchdog Timer Resets verhindern, indem man in jedem Durchlauf durch die Hauptschleife des Programms einen Watchdog Timer Refresh durchführt, z.B. durch die Befehle

```
SETB    WDT
SETB    SWDT
```