12. Die CCU - unit (= Capture - compare - unit)

12.1. Der Timer 2 im Reload-Betrieb

Die **Capture - Compare - Unit** ermöglicht die Erzeugung von pulsbreiten- oder frequenzmodulierten Signalen und bietet viele andere Möglichkeiten. Ihr Kern ist der **Timer 2**, aber im 80C535 hat er überhaupt nichts mehr mit dem Timer 2 vom früheren 8032 / 8052 zu tun. Es handelt sich jetzt um eine der kompliziertesten Baugruppen auf dem Chip und deshalb gibt es auch keine “Kompatibilität” mit älteren Programmen.
Timer 2 besteht aus **2 je 8 Bit langen** Registern (TL2 und TH2), der mit 4 weiteren „**CC“-Registern** (= Capture – Compare – Registern) zusammenarbeiten kann.

Beim „**Compare**“- Betrieb wird der Inhalt des ausgewählten CC-Registers mit dem Zählerstand von Timer 2 verglichen und bei Gleichheit ein Interrupt ausgelöst.

Beim „**Capture**“-Betrieb wird durch ein “Auslösesignal” der momentane Zählerstand von Timer 2 in einem dieser Zusatzregister gespeichert, wobei der Zähler ungestört weiterläuft.

Zusätzlich ist das erste Register als „**CRC“-Register** ausgeführt und zusätzlich für den „**Reload**“-Betrieb eingerichtet. Damit kann nach dem Überlauf von Timer 2 erneut ein bestimmter „Startwert“ nachgeladen und so die Wiederholfrequenz programmiert werden.

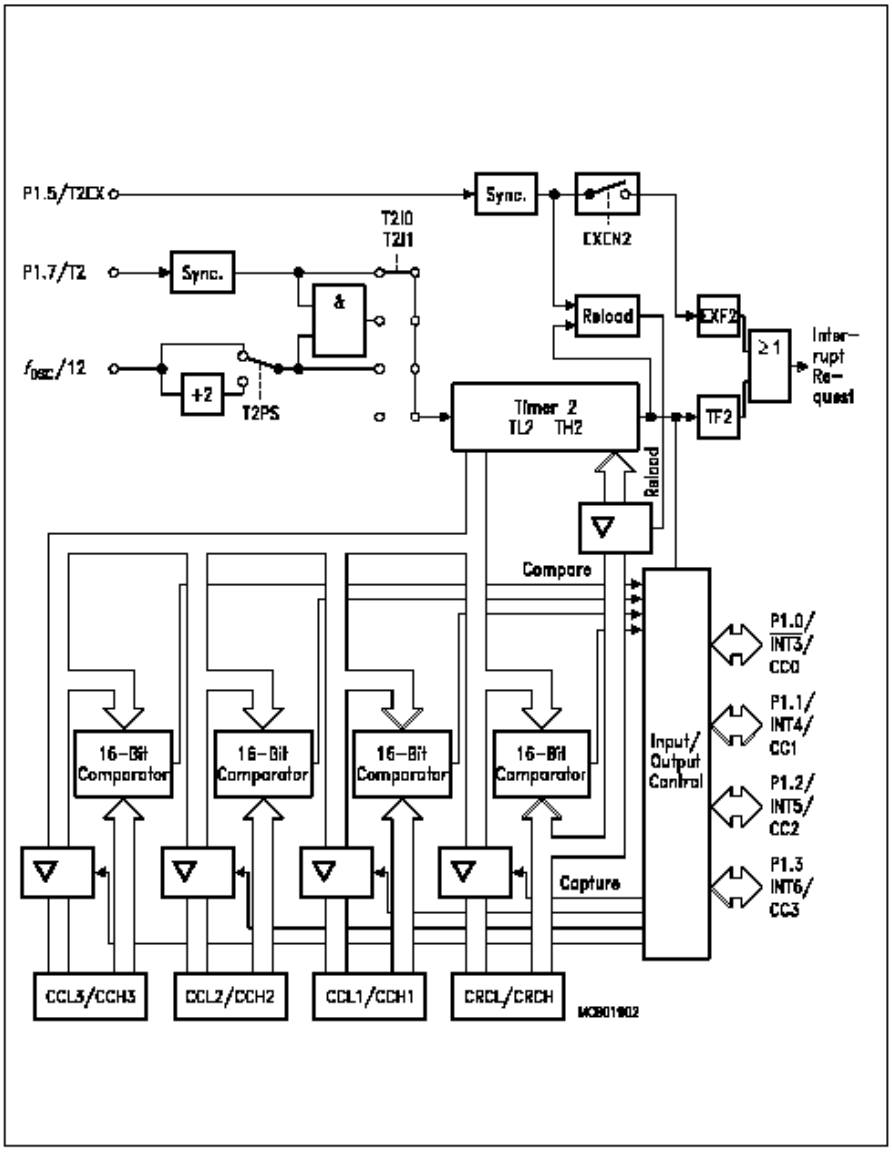

Figure 7-33 a)
Timer 2 Block Diagram

Wir wollen als Anwendungsbeispiel mal den **16 - Bit – Reload - Timerbetrieb** ansehen und haben hier ein 16-Bit-Register vor uns, das sowohl als Timer wie auch als Zähler betrieben werden kann (Timer-Register TL2 und TH2). Die verschiedenen Optionen können durch das **T2CON**-Register eingestellt werden:

T2CON -Register (Adresse 0C8H)

Bit:	T2CON.7	T2CON.6	T2CON.5	T2CON.4	T2CON.3	T2CON.2	T2CON.1	T2CON.0
	T2PS	I3FR	I2FR	T2R1	T2R0	T2CM	T2I1	T2I0

Der Eingang lässt sich durch die **beiden untersten Bits** des "Timer 2 - Control - Registers" (also durch T2CON.0 und T2CON.1) zwischen **vier Optionen** umschalten. Diese beiden Steuerbits tragen die offizielle Bezeichnung "**Timer 2 Input Selection Bits T2I0 und T2I1**, und sie haben folgende Wirkungen:

Modus	T2I1	T2I0	Einstellung
0	0	0	Timer 2 ist **abgeschaltet** und steht still
1	0	1	**Timer - Betrieb**. Eingangssignal ist entweder die durch 12 oder die durch 24 geteilte Oszillatorfrequenz. Zwischen diesen beiden Möglichkeiten kann mit dem Bit T2CON.7 (Name: "**Timer 2 - Prescale" = T2PS**) ausgewählt werden.
2	1	0	**Zähler - Betrieb**. Die zu zählenden Eingangsimpulse werden an den Controllerpin P1.7 angelegt.
3	1	1	**"Gated Timer Function**". Solange am Portpin P1.7 High - Pegel anliegt, laufen die internen Taktimpulse mit der ausgewählten Prescale - Frequenz in das Timer 2 - Register. So kann man z. B. sehr leicht Impulslängen bestimmen

Erläuterung der weiteren Bits im T2CON-Register:

T2CON.7 = T2PS	schaltet den **Vorteiler (Prescaler)** ein.
T2CON.6 und T2CON.5 = I3FR und I2FR	gehören nicht hierher, denn sie sind ein Teil der **Interruptsteuerung**. Mit ihrer Hilfe kann bei den beiden externen Interrupts INT3 und INT2 die steigende oder fallende Flanke ausgewählt werden.
T2CON.4 und T2CON.3 = T2R1 und T2R0	steuern den **Reload - Betrieb des Timers 2** (dazu kommen wir gleich!).
T2CON.2 = T2CM	ist beim **"Compare-Betrieb**" des Timers 2 wichtig und schaltet zwischen **Compare-mode 1 bzw.Compare - mode 2** um.

Jetzt geht es wieder mit dem Timer 2 weiter:
Sobald beim Zählvorgang die Register "überlaufen", wird das Flag **TF2 im IRCON - Register** gesetzt.
Anschließend gibt es mehrere Möglichkeiten zur Auswertung. Die Auswahl wird durch die beiden Bits

T2R0 und T2 R1 im T2CON - Register

getroffen:

Bit		Funktion
T2R1	T2R0	Timer 2 reload selection
0	X	**Reload disabled** (kein Reload - Betrieb möglich)
1	0	***Mode 0***: **Reload durch Überlauf von Timer 2** ausgelöst
1	1	***Mode 1***: **Reload durch fallende Flanke am Portpin P1.5 (= T2EX)** ausgelöst

Übersichtsschaltplan für den Reload - Betrieb:

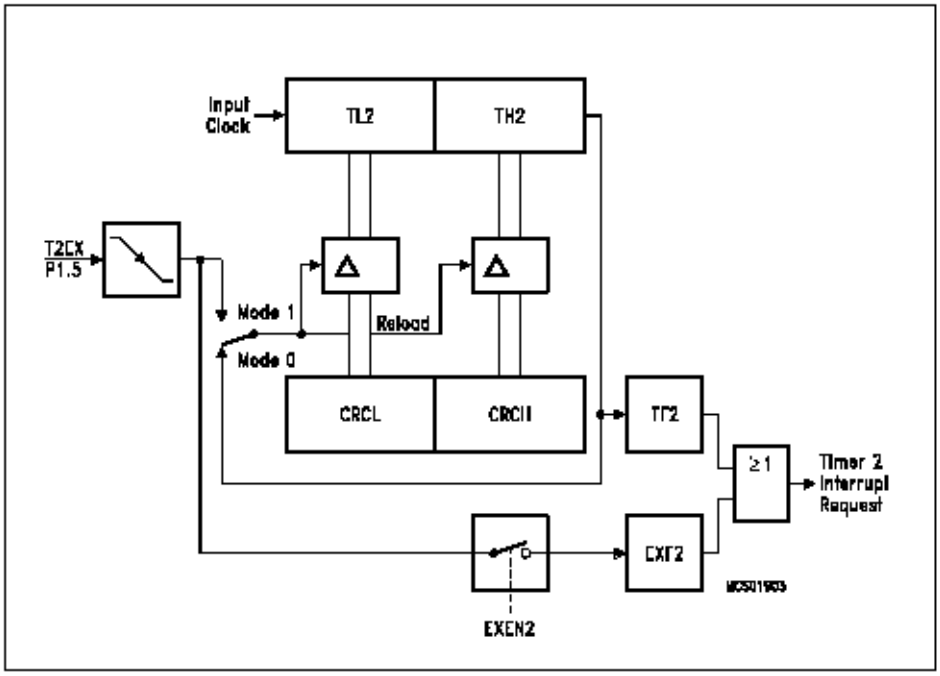

Figure 7-33 b)
Timer 2 in Reload Mode

---

1. Aufgabe: Schreiben Sie ein Programm in Assembler und C, mit dem Sie ein 50 Hz-Rechtecksignal an Portpin P1.0 erzeugen. Benützen Sie dazu den Timer 2 im 16 Bit-Reloadbetrieb und verwenden Sie den Timer 2 - Interrupt beim Zähler-Überlauf dazu.

Lösungsbeispiel in Assembler

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

OFFSET      EQU 4000h
?Stack      SEGMENT IDATA
            RSEG ?STACK
            DS 5
;=====================================================================
CSEG AT 0+OFFSET
            jmp START

TON2        SEGMENT CODE
            RSEG TON2

START:      mov SP,#?STACK-1
            mov IEN0,#10100000b     ;EAL und Timer2 - Interrupt einschalten
            mov T2CON,#00010001b    ;Reloadbetrieb durch Überlauf mit voller Taktfrequenz wählen
            mov CRCL,#              ;CRC-Register mit der Zahl 55535 =          vorladen
            mov CRCH,#              ;
            jmp $                   ;Endlosschleife
;=====================================================================
CSEG AT     02BH + OFFSET           ;ISR für Timer 2 nach 402B verschieben
            ljmp T2INT              ;Verbindung zum neuen Vektor herstellen

T2INT_code_seg      SEGMENT CODE
                    RSEG T2INT_code_seg

T2INT:      clr TF2                 ;Timer2-Overflow-Flag löschen
            cpl P1.0                ;Ausgang P5.7 invertieren
            reti                    ;Zurück aus der Interrupt - Service-Routine
;=====================================================================
end
```

Aufgabe: Betreiben Sie den Timer 2 im Reload - Betrieb und erzeugen Sie auf diese Weise ein Rechtecksignal mit der Frequenz von 1 Hz an Portpin P1.0. Benützen Sie dazu den Interrupt-Betrieb und schreiben Sie das Programm in C.

Lösungsweg:

a) Richtige Eingangstaktfrequenz sowie Timer-Betrieb einstellen.

b) Auto – Reload - Betrieb mit Nachladung beim Timer-2-Überlauf für eine Zählzeit von 50 Millisekunden wählen.

c) Korrekten Reloadwert für die vorgesehene Ausgangsfrequenz bestimmen und in die beiden Register CRCL bzw. CRCH laden

d) Hauptprogramm und Interrupt-Service-Routine schreiben, wobei der Sekundentakt durch eine passende Routine in der Interrupt-Service-Routine erzeugt wird.

Lösung:

12.2. Erzeugung von PWM-Signalen (Compare-Betrieb)

Sehr oft werden in der Elektronik pulsweitenmodulierte Rechtecksignale benötigt (Beispiel: Drehzahlregelung von Gleichstrom-Motoren, Regelung der Ausgangsspannung bei DC-DC-Convertern usw.). Bei unserem Mikrocontroller können wir sogar an 4 verschiedenen Portpins solche (unabhängig voneinander steuerbaren) PWM-Signale ausgeben lassen.

Arbeitsweise:

In ein CC-Register (= Capture – Compare – Register) wird eine 16 Bit - Hex-Zahl geladen.

Timer 2 wird vom Oszillatortakt hochgezählt bis zum Überlauf und zählt einfach immer weiter. Das gibt im Normalfall eine Wiederholfrequenz von f = 1 / 65535 Mikrosekunden = 15,259 Hz , sie kann jedoch durch Einschalten eines „Prescalers“ noch halbiert werden. Außerdem gibt es ja die Möglichkeit des „Auto-Reloadbetriebs“ beim Timer 2. Damit lassen sich Periodendauern von wenigen Mikrosekunden bis zu den eben erwähnten 65535 Mikrosekunden erreichen.

Sobald Timer 2 und das CC – Register den gleichen Inhalt aufweisen, wird der vorgesehene Ausgabe-Portpin auf „HIGH“ gesetzt.

Timer 2 läuft weiter bis zu seinem **Überlauf** und setzt dann den Portpin wieder auf „**LOW**“.

usw.

Zur Änderung der Pulsweite muss nur ein anderer Wert in das CC-Register geladen werden.

Bei der Programmierung müssen 2 Register mit den richtigen Inhalten versehen werden:

Im **CCEN-Register** (= Compare – Capture – Enable – Register mit der Adresse 0xC1 im SFR-Bereich) wird mit jeweils 2 Bit festgelegt, welches der 4 CC-Register zum Einsatz kommen soll und welche Betriebsart gilt:

Bit:	7	6	5	4	3	2	1	0
Register	CC3		CC2		CC1		CRC	

Dabei haben diese Bitkombinationen folgende Wirkung:

10 bewirkt: Compare oder Capture AUS

11 bewirkt Compare AUS, Capture EIN

10 bewirkt Compare EIN, Capture AUS

11 bewirkt Compare AUS, Capture Modus 1 EIN (Start durch Beschreiben des Low-Byte-Registers).

Vorsicht:
Dieses Register ist nicht bitadressierbar und muss deshalb mit einem kompletten Byte beschrieben werden...

Hinweis:

a) beim Register „CRC“ ist der Reload-Betrieb dabei.....

b) die pulsweitenmodulierten Signale erscheinen an den Portpins P1.0 bis P1.3

Beispiel:
Soll das CC1-Register im Compare-Betrieb verwendet werden, dann wird CCEN mit der Bitkombination „00001000“ gefüllt. Also schreibt man im C-Code **CCEN=0x08;**

An Portpin P1.1 kann dann das PWM-Signal abgenommen werden.

Zusätzlich brauchen wir (wie schon beim Reload-Betrieb!) noch das **T2CON-Register:**

T2CON -Register (Adresse 0C8H), bitadressierbar

Bit:	T2CON.7	T2CON.6	T2CON.5	T2CON.4	T2CON.3	T2CON.2	T2CON.1	T2CON.0
	T2PS	I3FR	I2FR	T2R1	T2R0	T2CM	T2I1	T2I0

Die Programmierung erkennt man am besten am folgenden Programm-Ausschnitt, durch den CC1 zur PWM-Erzeugung eingesetzt wird:

```
/*Nun folgen die einzelnen Bits im T2CON-Register!*/

T2I0=1;        //Modus "01" bedeutet "Timer 2 ein" und
T2I1=0;        //"Taktsignal = Oszillatortakt"

T2CM=0;        /*gibt Modus "0", also High an P1.1 bei Gleichstand von CC1 und Timer 2.
               Low kommt wieder bei Timer 2 – Überlauf */

T2R0=0;        //Timer 2 - Reload ausschalten
T2R1=0;

I2FR=0;        //Rising oder Falling Edge-Wahl für Interrupt INT3
I3FR=0;        //bzw INT2 (ist hier unwichtig)

T2PS=0;        //1 MHz - Takt für Timer 2 (kein Prescaler)

/*Achtung: diese obigen 8 Bits im T2CON-Register können natürlich
auch auf einmal durch Beschreiben des Registers mit einem
entsprechenden Byte auf die gewünschten Werte gesetzt werden.
Diese Schreibweise dient nur zum besseren Verständnis.....)*/
```

Aufgabe:

Verwenden Sie die Drucktasten-Zusatzplatine bzw. den zu den Drucktasten parallelgeschalteten DIP-Schalter an Port P5 zur **Einstellung verschiedener Pulsbreiten** bei einer Grundfrequenz von f = 1 / 65535 µs = 15, 26 Hz. Verwenden Sie Register CC1 und beobachten Sie das entstehende PWM-Signal mit dem Oszilloskop an Portpin P1.1.
Schreiben Sie das Programm in C.

Lösungsbeispiel:

```
/*-------------------------------------------------------------------
Programmbeschreibung
-------------------------------------------------------------------
Name:       PWM_01.c
Funktion:   Am Portpin P1.1 (Bitadresse 0x91, hier mit "ausgang"
            bezeichnet) wird ein pulsweitenmoduliertes
            Rechtecksignal mit der Frequenz f = 1 / 65535us
            = 15,26 Hz ausgegeben. Die Pulsbreite lässt sich über den
            parallel zu den Drucktasten liegenden  DIP-Schalter
            (auf der entsprechenden Zusatzplatine) oder direkt über
            den Druck auf die Tasten einstellen.
            Der DIP-Schalter wird über Port P5 abgefragt.

Datum:      02. 01. 2005
Autor:      G. Kraus
-------------------------------------------------------------------
Deklarationen und Konstanten
-------------------------------------------------------------------*/
#include <reg515.h>
#include <stdio.h>
sbit ausgang=0x91;          //PWM-Signal an Portpin P1.1
sfr DIP=0xF8;               //DIP-Schalter an Port P5
/*-------------------------------------------------------------------
Hauptprogramm
-------------------------------------------------------------------*/
void main(void)
{   CCEN=0x08;              //Ergibt "Compare CC1"

    /*Nun folgen die einzelnen Bits im T2CON-Register!*/

    T2I0=1;                 //Modus "01" bedeutet "Timer 2 ein" und
    T2I1=0;                 //"Taktsignal = Oszillatortakt"
    T2CM=0;                 //gibt Modus "0", also High an P1.1 bei Gleichstand
                            //von CC1 und Timer 2.
                            //Low kommt wieder bei Timer 2 - Überlauf
    T2R0=0;                 //Timer 2 - Reload ausschalten
    T2R1=0;
    I2FR=0;                 //Rising oder Falling Edge-Wahl für Interrupt INT3
    I3FR=0;                 //bzw INT2 (ist hier unwichtig)
    T2PS=0;                 //1 MHz - Takt für Timer 2 (kein Prescaler)

    /*Achtung: diese obigen 8 Bits im T2CON-Register können natürlich
    auch auf einmal durch Beschreiben des Registers mit einem
    entsprechenden Byte (Kommando: „T2CON=0x01;“)
    auf die gewünschten Werte gesetzt werden.
    Diese Schreibweise dient nur zum besseren Verständnis.....*/

    CCL1=0xFF;              //Steuerung nur durch High-Byte, deshalb Low-Byte auffüllen
    while(1)
    {   CCH1=DIP;           //DIP-Schalter-Einstellung ins High-Byte von CC1 kopieren
    }
}
```

Zusatzaufgabe:
Stellen Sie im obigen Beispiel die Grundfrequenz des PWM-Signals auf f = 1 kHz ein und betreiben Sie dazu den Timer 2 im Autoreload-Modus.

Aufgabe:

Verbinden Sie die 8 Einstellpotis (zur Vorgabe von Gleichspannungen zwischen 0 und +5V) auf der DA-Zusatzplatine über ein Flachbandkabel mit dem AD-Eingang des Microcontrollers. Die **einstellbare Gleichspannung am Eingang AN0** soll nun mit dem integrierten AD-Konverter gemessen und das Messergebnis als **Vorgabe für die Pulsweite** verwendet werden. Die Grundfrequenz sei weiterhin f = 1 / 65535 µs = 15,26 Hz.
Schreiben Sie das Programm in C.
(Damit lässt sich die Pulsbreite des Ausgangssignals über ein Poti verändern und z. B. die Drehzahl eines Gleichstrom-Motors steuern).

Lösungsvorschlag:

```
/*-------------------------------------------------------------------
Programmbeschreibung
-------------------------------------------------------------------
Name:       PWM_02.c
Funktion:   Am Portpin P1.1 (Bitadresse 0x91, hier mit "ausgang" bezeichnet) wird ein pulsweiten-moduliertes
            Rechtecksignal mit der Frequenz f = 1/65535us ausgegeben. Zusätzlich wird an den Eingang AN0 eine
            Gleichspannung zwischen 0 und +5V angelegt, die mit dem integrierten AD-Wandler des Controllers
            gemessen wird. Das Messergebnis dient als Vorgabe für die
            einzustellende Pulsbreite.
Datum:      01. 01. 2005
Autor:      G. Kraus
-------------------------------------------------------------------
Deklarationen und Konstanten
-------------------------------------------------------------------*/
#include <reg515.h>
#include <stdio.h>
sbit ausgang=0x91;      //PWM-Signal an Portpin P1.1
/*-------------------------------------------------------------------
Hauptprogramm
-------------------------------------------------------------------*/
void main(void)
{   CCEN=0x08;          //CC-Register auf "10" setzen. Ergibt "Compare CC1"

    /*Nun folgen die einzelnen Bits im T2CON-Register!*/

    T2I0=1;             //Modus "01" bedeutet "Timer 2 ein" und
    T2I1=0;             //"Taktsignal = Oszillatortakt"
    T2CM=0;             //gibt Modus "0", also High an P1.1 bei Gleichstand
                        //von CC1 und Timer 2.
                        //Low kommt wieder bei Timer 2 - Überlauf

    T2R0=0;             //Timer 2 - Reload ausschalten
    T2R1=0;
    I2FR=0;             //Rising oder Falling Edge-Wahl für Interrupt INT3
    I3FR=0;             //bzw INT2 (ist hier unwichtig)
    T2PS=0;             //1 MHz - Takt für Timer 2 (kein Prescaler)

    /*Achtung: diese obigen 8 Bits im T2CON-Register können natürlich auch auf einmal durch Beschreiben des
    Registers mit einem entsprechenden Byte auf die gewünschten Werte gesetzt werden. Diese Schreibweise dient nur
    zum besseren Verständnis.....*/

    CCL1=0xFF;          //Steuerung nur durch High-Byte

    ADCON=0x00;         //Eingang AN0 und Einzelmessung beim Wandler wählen
    DAPR=0x00;          //Wandlerstart mit Messbereich von 0.....+5V

    while(1)
    {       while(BSY==1);
            CCH1=255-ADDAT;     //Wandelergebnis in CCH1 kopieren
            DAPR=0x00;          //Neuer Start des Wandles

    }
}
-------------------------------------------------------------------------------------------------
```

Aufgabe: Ändern Sie die Grundfrequenz beim erzeugten PWM – Signal auf einen Wert von etwa 4 kHz.

12.3. Capture - Betrieb

Prinzip:	Beim Capture-Betrieb kann durch ein “Auslösesignal” der momentane **Zählerstand von Timer 2** in **einem von drei Zusatzregistern gespeichert** werden, wobei der **Zähler ungestört weiterläuft**!

7.5.3 Capture Function

Each of the three compare/capture registers CC1 to CC3 and the CRC register can be used to latch the current 16-bit value of the timer 2 registers TL2 and TH2. Two different modes are provided for this function. In mode 0, an external event latches the timer 2 contents to a dedicated capture register. In mode 1, a capture will occur upon writing to the low order byte of the dedicated 16-bit capture register. This mode is provided to allow the software to read the timer 2 contents "on-the-fly".

In mode 0, the external event causing a capture is

- for CC registers 1 to 3: a positive transition at pins CC1 to CC3 of port 1
- for the CRC register: a positive or negative transition at the corresponding pin, depending on the status of the bit I3FR in SFR T2CON. If the edge flag is cleared, a capture occurs in response to a negative transition; if the edge flag is set a capture occurs in response to a positive transition at pin P1.0/$\overline{\text{INT3}}$/ CC0.

In both cases the appropriate port 1 pin is used as input and the port latch must be programmed to contain a one (1). The external input is sampled in every machine cycle. When the sampled input shows a low (high) level in one cycle and a high (low) in the next cycle, a transition is recognized. The timer 2 contents is latched to the appropriate capture register in the cycle following the one in which the transition was identified.

In mode 0 a transition at the external capture inputs of registers CC0 to CC3 will also set the corresponding external interrupt request flags IEX3 to IEX6. If the interrupts are enabled, an external capture signal will cause the CPU to vector to the appropriate interrupt service routine.

In mode 1 a capture occurs in response to a write instruction to the low order byte of a capture register. The write-to-register signal (e.g. write-to-CRCL) is used to initiate a capture. The value written to the dedicated capture register is irrelevant for this function. The timer 2 contents will be latched into the appropriate capture register in the cycle following the write instruction. In this mode no interrupt request will be generated.

Figures 7-41 and **7-42** show functional diagrams of the capture function of timer 2. **Figure 7-41** illustrates the operation of the CRC register, while **figure 7-42** shows the operation of the compare/capture registers 1 to 3.

The two capture modes can be established individually for each capture register by bits in SFR CCEN (compare/capture enable register). That means, in contrast to the compare modes, it is possible to simultaneously select mode 0 for one capture register and mode 1 for another register. The bit positions and functions of CCEN are listed in **figure 7-40**.

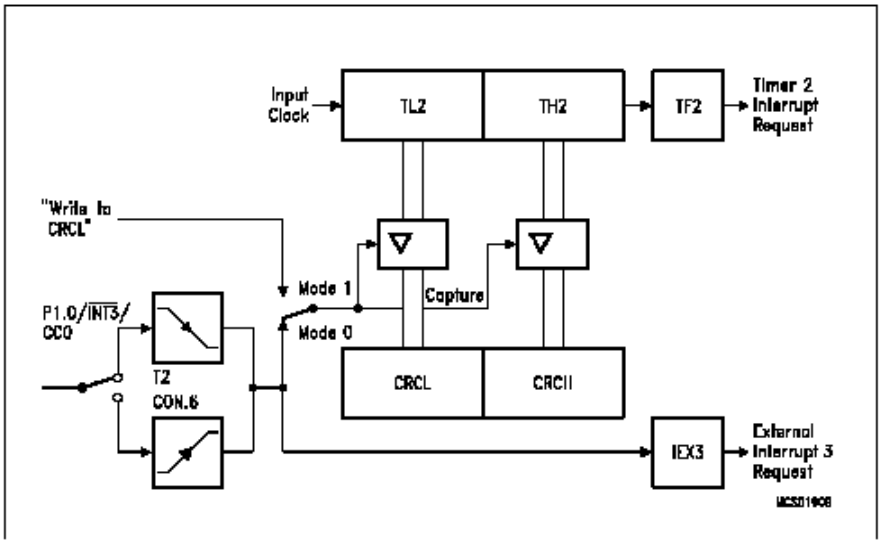

Figure 7-41
Capture with Register CRC

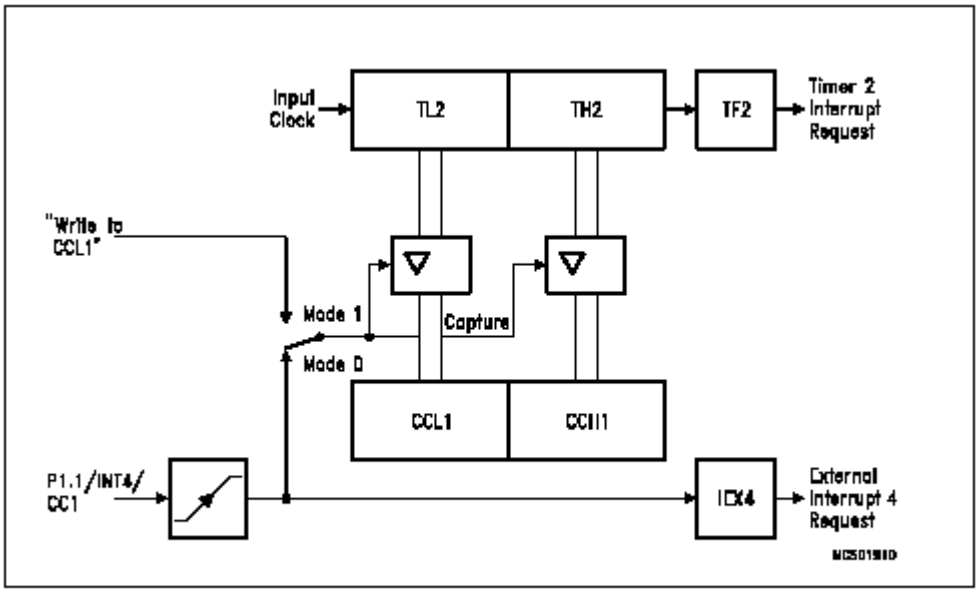

Figure 7-42
Capture with Registers CC1 to CC3