

## 2.4. Porturile interne ale microcontrolerelor

### 2.4.1 Generalități

Microcontrolerele în general sunt prevăzute cu intrări – ieșiri digitale organizate în porturi bidirecționale ( $P_0$ ,  $P_1$ ,  $P_2$  și  $P_3$ ) pe numărul de biți specific tipului de microcontroler.

Cu instrucțiuni corespunzătoare se poate scrie sau citi în, respectiv din, aceste porturi. Operația se face prin intermediul registrelor cu funcții speciale asociate porturilor notate cu  $P_0$ ,  $P_1$ ,  $P_2$  și  $P_3$  și care se află la adrese fixe din zona memoriei de date rezervată registrelor cu funcții speciale (*SFR*). Figura 2.4.1 sunt precizate ca exemplu adresele acestor registre pentru microcontrolerul 80(C)51/31.

Portul	Adresa din zona <i>SFR</i>
$P_0$	80H
$P_1$	90H
$P_2$	A0H
$P_3$	B0H

Figura 4. 1. Adresele porturilor microcontrolerelor 80(C)51/31

Fiecare linie (legătură) asociată porturilor poate fi definită ca fiind de intrare sau de ieșire și adresată în mod individual. În consecință, registrele speciale asociate porturilor vor putea fi adresate pe bit sau pe *byte* așa cum este exemplificat în Figura 2.4.2.

87H	86H	85H	84H	83H	82H	81H	80H	
P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	P0:80H

Figura 2.4. 2. Structura registrului asociat portului 0

**Observație:** O linie a unui port va fi notată cu  $P_{i,j}$ , unde:

indicele  $i$  – indică numărul portului;

indicele  $j$  – bitul din cadrul portului.

### 2.4.2 . Structura de principiu a porturilor

Schema electronică a unei linii asociate porturilor este formată din: celulă de memorie realizată cu un bistabil de tip  $D(D\text{-latch})$ , etaj de ieșire și două circuite temporare (*Buffer*).

**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

Dacă se dorește transmiterea unei date către terminalul portului, atunci această dată, generată pe magistrala internă de microcontroler, este înscrisă în bistabilul  $D$ , iar ieșirea  $\bar{Q}$  a acestuia va comanda, prin intermediul etajului de ieșire, starea logică a terminalului.

În cazul în care se dorește a se citi starea logică a semnalului aplicat pe terminalul extern, atunci în prealabil va trebui inactivat etajul de ieșire, iar apoi, cu comanda *Read Pin*,

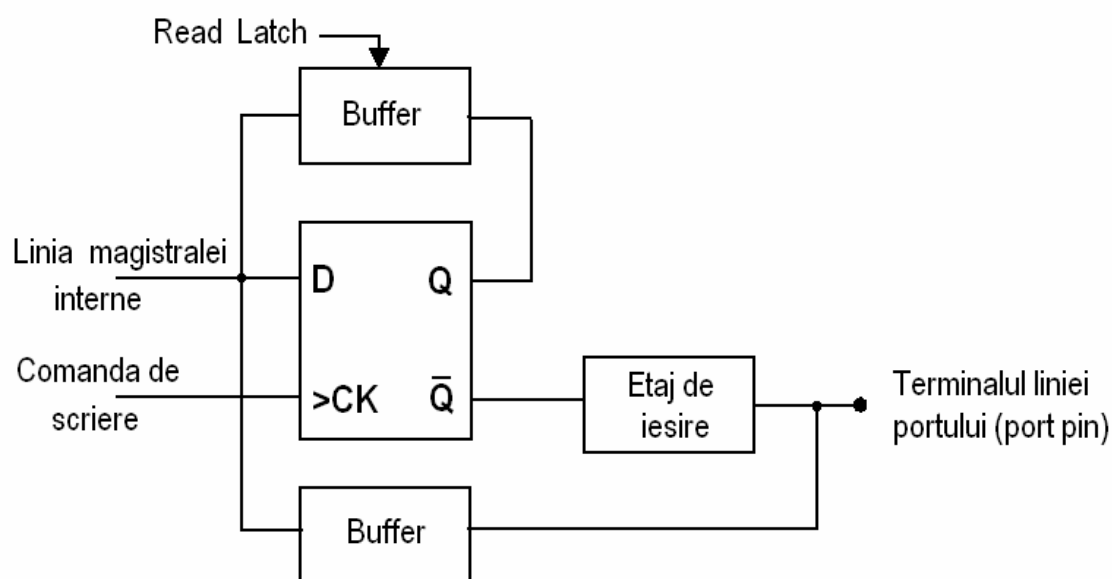


Figura 2.4. 3. Schema electronică asociată unei linii de port.

buffer-ul *tri-state* va cupla terminalul portului la linia corespunzătoare a magistralei interne a microcontrolerului.

Schema mai este prevăzută cu un *buffer* cu ieșire *tri-state* prin intermediul căruia, cu comanda *Read Latch*, se poate citi pe magistrala internă de date starea bistabilului asociat portului.

### 2.4.3. Structura porturilor $P_1$ și $P_3$

Schema electronică generală asociată unei linii de port a fost prezentată în Figura 2.4.3. În funcție de utilizarea concretă a portului apar în general modificări în structura etajului de ieșire. Porturile  $P_1$  și  $P_3$  sunt gândite a fi utilizate ca porturi generale de intrare-ieșire și ca urmare etajul de ieșire este realizat în principiu cu un tranzistor n-MOSFET cuplat printr-o rezistență la tensiunea de alimentare.

Când se dorește transmiterea stării de unu logic la ieșire, atunci această informație se înregistrează în bistabilul  $D$ , pe ieșirea  $\bar{Q}$  a acestuia apare semnalul corespunzător lui zero logic (*Low*), semnal ce nu poate deschide tranzistorul  $T$  astfel că pe pinul portului se va citi un nivel de tensiune ridicat (*High*) corespunzător informației de unu logic. În cazul în care informația înscrisă în bistabilul  $D$  este de zero logic, atunci semnalul de pe  $\bar{Q}$  va comanda intrarea în

**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

conducție a tranzistorului  $T$  iar pe pinul portului se va putea citi un nivel de tensiune coborât corespunzător lui zero logic.

În cazul în care dorim să citim nivelul logic al semnalului aplicat din exterior pe pinul portului, în prealabil va trebui să inactivăm etajul de ieșire prin înscrierea semnalului unu logic în bistabilul  $D$  ceea ce va determina blocarea tranzistorului  $T$ .

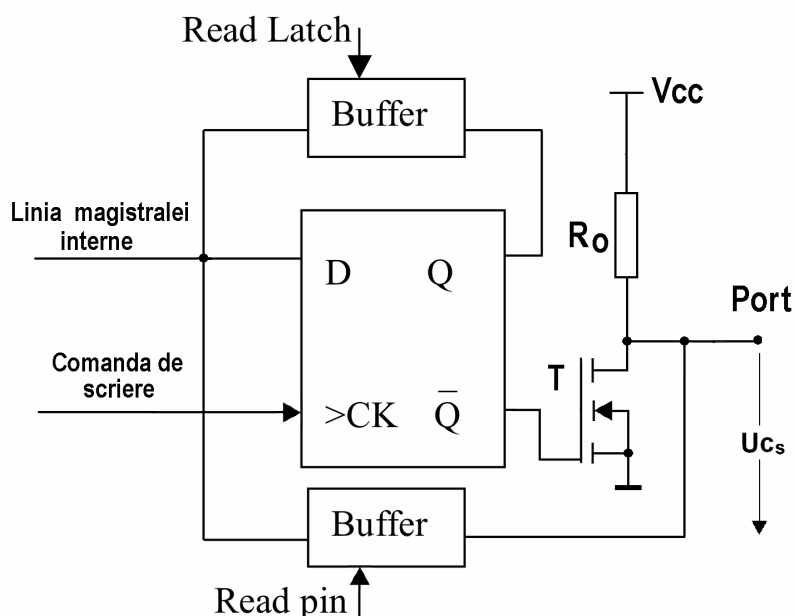


Figura 2.4. 4 Schema electronică asociată unei linii de port ( $P_1$  sau  $P_3$ )

Cu tranzistorul  $T$  blocat, nivelul logic al pinului portului va fi dictat de sursa din exterior, nivel ce va putea fi citit prin cuplarea terminalului la magistrala internă prin comanda *Read Pin*. (În cazul în care tranzistorul  $T$  nu ar fi blocat, atunci el ar scurtcircuita la masă pinul portului iar semnalul citit ar fi de zero logic indiferent de starea sursei externe conectată la pinul portului.)

La terminalele porturilor paralele sunt cuplate sarcini externe pe care acestea le comandă. Prezența unei componente capacitive a sarcinii ( $C_S$ ) este practic inevitabilă prin existența capacităților parazite care se adaugă la capacitatea propriu zisă a sarcinii (de ex. etaje de intrare cu tranzistoare MOS). În cazul în care la momentul  $t_i$  se dă o comandă de trecere a pinului portului din zero logic în unu logic, atunci tensiunea la ieșire va avea o creștere după o lege exponențială dictată de încărcarea condensatorului  $C_S$  (vezi Figura 2.4.4.b). Viteza de creștere depinde de valoarea condensatorului  $C_S$  precum și de curentul de încărcare limitat de rezistența  $R_D$ . Fenomenul mai sus menționat limitează viteza de lucru a portului iar constructorii au prevăzut soluții pentru a mări această viteză de lucru prin realizarea unor etaje de ieșire în contratimp cu tranzistoare NMOS sau cu tranzistoare CMOS, ce vor fi prezentate în continuare ca exemplu pentru microcontrolerul 8051/31

#### Structura etajului de ieșire cu tranzistoare NMOS

În Figura 2.4.5 este prezentată schema electronică asociată unei linii a portului realizată cu tranzistoare NMOS.

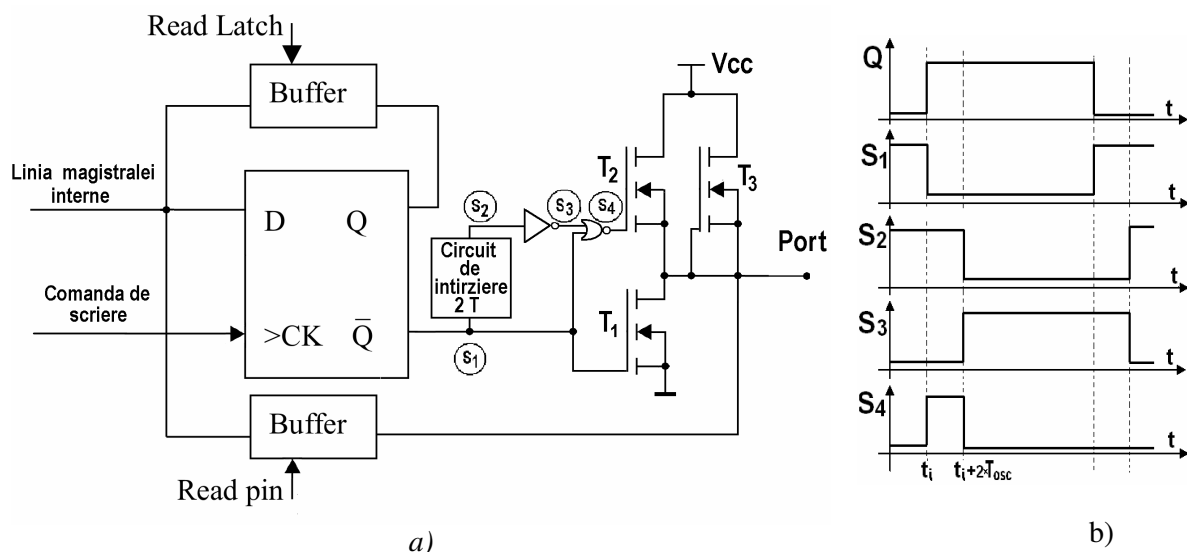


Figura 2.4. 5.. a) Schema etajului de ieșire cu tranzistoare NMOS; b) Forme de undă care ilustrează comanda tranzistorului  $T_2$

Ideea de realizare a etajului de ieșire este următoarea: tranzistorul  $T_3$  este folosit ca o rezistență prin care pinul portului este conectat în interior la  $V_{CC}$  iar schema logică din interior asigură deschiderea tranzistorului  $T_2$  pe durata a două perioade de tact ( $2T$ ) după apariția comenzii de trecere  $0 \rightarrow 1$  din momentul  $t_i$  (vezi Figura 2.4.5 b). În felul acesta, imediat după comanda de comutare a etajului de ieșire, pe durata a două perioade a tactului intern tranzistorul  $T_2$  va asigura un curent suplimentar pentru încărcarea condensatorului de sarcină ( $C_S$ ) mărind în felul acesta viteza de lucru a portului.

#### Structura etajului de ieșire cu tranzistoare CMOS (80C51/31)

Schema electronică asociată unei linii a portului realizată cu tranzistoare  $n$ -MOS și  $p$ -MOS este prezentată în Figura 2.4.6.

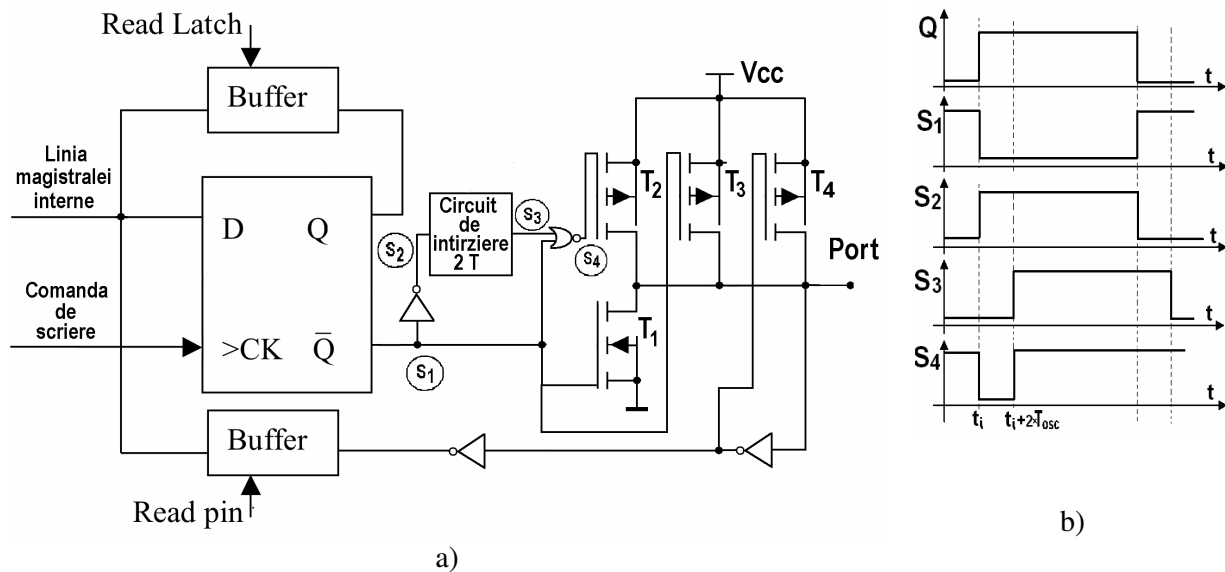
**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

Figura 2.4. 6 .a) schema etajului de ieșire; b) forme de undă ce ilustrează comanda tranzistorului  $T_2$

Ideea care se urmărește este similară celei expuse anterior (creșterea vitezei de lucru prin mărirea curentului de încărcare a sarcinilor capacitive la apariția unei tranziții logice din 0 în 1). În cazul transmiterii către ieșire a unui nivel de tensiune ridicat ( $Q=1$ ), tranzistorul  $T_1$  este blocat iar tranzistorul  $T_3$  conduce, pe când în cazul transmiterii unui nivel de tensiune coborât ( $Q=0$ ), tranzistorul  $T_1$  conduce și tranzistorul  $T_3$  este blocat. La apariția unei tranziții de la nivelul de tensiune coborât la nivel de tensiune ridicat ( $Q=0 \rightarrow 1$ ) logica de comandă asociată tranzistorului de ieșire va asigura intrarea în conducție pe durata a două perioade de tact ( $2T$ ) a tranzistorului  $T_2$ , mărindu-se în felul acesta curentul injectat în sarcină și implicit viteza de comutare (vezi Figura 2.4.6.b).

O particularitate a schemei o reprezintă prezența tranzistorului  $T_4$ , care se va deschide ori de câte ori tensiunea pe pinul portului depășește nivelul de  $1 \div 1,5$  V asigurându-se în felul acesta un nou curent suplimentar pentru accelerarea tranziției.

#### 2.4.4.Funcțiile îndeplinite de porturi

Porturile microcontrolerului sunt gândite a avea utilizări multiple. În Figura 2.4.7.a) și Figura 2.4.7.b) sunt prezentate porturile  $P_0$ ,  $P_1$ ,  $P_2$  și  $P_3$  cu descrierea și precizarea utilizărilor alternative ale terminalelor.

Portul 0	P0.0	$A_0/D_0$
	P0.1	$A_1/D_1$
	P0.2	$A_2/D_2$
	P0.3	$A_3/D_3$
	P0.4	$A_4/D_4$
	P0.5	$A_5/D_5$
	P0.6	$A_6/D_6$
	P0.7	$A_7/D_7$

Generează octetul inferior al adresei, dacă  $ALE = High$ , sau este utilizat ca magistrală bidirecțională de date, dacă  $ALE = Low$

**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

Portul 1	P1.0	I/O	$\overline{INT3/CC0}$	}	intrări – pentru întreruperi externe și <i>Capture Function</i> , respectiv ieșiri – pentru <i>Compare Function</i>
	P1.1	I/O	$INT4/CC1$		
	P1.2	I/O	$INT5/CC2$		
	P1.3	I/O	$INT6/CC3$		
	P1.4	I	$\overline{INT2}$		intrare întreruperi externe <i>External Reload Function</i> ieșire tact intrare <i>Timer 2</i>
	P1.5	I	<i>TREX</i>		
	P1.6	I	<i>CLK OUT</i>		
	P1.7	I	<i>T2</i>		
Portul 2	P2.0	A <sub>8</sub>	Portul 2 este destinat generării octetului superior al adresei		
	P2.1	A <sub>9</sub>			
	P2.2	A <sub>10</sub>			
	P2.3	A <sub>11</sub>			
	P2.4	A <sub>12</sub>			
	P2.5	A <sub>13</sub>			
	P2.6	A <sub>14</sub>			
	P2.7	A <sub>15</sub>			

Figura 2.4. 7 a) Utilizarea porturilor 0;1;2

Portul 3	P3.0	I	R×D		intrare de date pentru comunicația serie ieșire de date pentru comunicația serie întreruperi externe 0, respectiv Timer 0 Gate întreruperi externe 1, respectiv Timer 1 Gate intrare numărător 0 intrare numărător 1
	P3.1	O	T×D		
	P3.2	I	$\overline{INT0}$		
	P3.3	I	$\overline{INT1}$		
	P3.4	I	T0		
	P3.5	I	T1		
	P3.6	O	$\overline{WR}$		
	P3.7	O	$\overline{RD}$		

Figura 2.4.7. b) Utilizarea portului 3

Din prezentarea făcută în Figura 2.4.7.a). observăm că: portul  $P_2$  este utilizat ca port de adresare pentru generarea octetului superior al adresei. Portul  $P_0$  are o funcție dublă: pe de o parte este utilizat pentru generarea octetului inferior al adresei (magistrală de adresă) iar pe de altă parte (după ce octetul de adresă a fost înscris cu semnalul *ALE* în registrul de memorie extern) este utilizat pentru transferul bidirecțional de date între procesor și mediul exterior (magistrala de date). Porturile  $P_1$  și  $P_3$  sunt utilizate în aplicații ca porturi generale de intrare și ieșire însă ele mai pot îndeplini și funcții alternative menționate în descrierea din Figura 2.4.7.a) și Figura 2.4.7.b).

**Observație importantă.** Pentru a putea fi utilizate funcțiile alternative ale portului este important ca în prealabil în bistabilul portului să se înscrie informația  $Q=1$  pentru a inactiva etajul de ieșire și pentru a putea interpreta în mod corect semnalul din exterior aplicat terminalului (vezi Figura 2.4.3.).

Pentru a ilustra modificările de structură pe care le impun funcțiile alternative ale porturilor vom prezenta în Figura. 2.4.8. aceste modificări doar în cazul portului  $P_0$ .

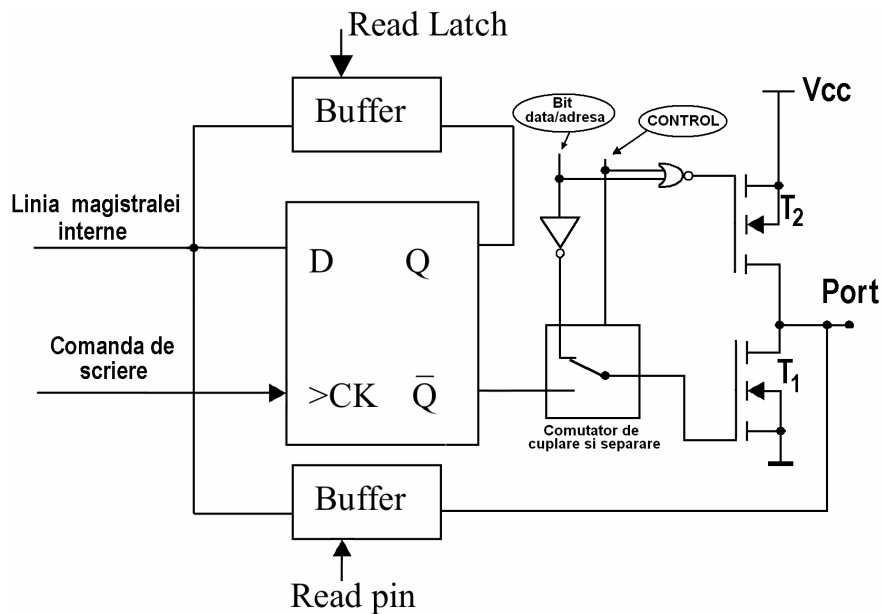
**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

Figura 2.4. 8. Structura de principiu a portului  $P_0$  cu funcții alternative

Așa cum a fost menționat anterior, microcontrolerul utilizează acest port ca port de adresă și de date. Din acest motiv etajul de ieșire este într-o măsură modificat față de varianta standard în sensul că pe lângă modul clasic de funcționare în care ori  $T_1$  ori  $T_2$  conduce este prevăzută și posibilitatea blocării ambelor tranzistoare (ieșire *tri-state*).

#### Lucrul în regim de magistrală de adresă.

La orice adresare a memoriei portul  $P_0$  este utilizat ca magistrală de adresă. Pentru aceasta, cu semnalul  $CONTROL=1$  generat de microcontroler, ieșirea  $\bar{Q}$  a bistabilului de tip  $D$  este izolată de etajul de ieșire de către comutatorul de cuplare/separare. Comanda etajului de ieșire este preluată de către bitul de adresă generat pe intrarea alternativă astfel: dacă bitul de adresă este  $A_i=1$  atunci  $T_2$  conduce și  $T_1$  este blocat iar dacă bitul de adresă este  $A_i=0$  atunci  $T_1$  conduce iar  $T_2$  este blocat.

**Observație.** Ori de câte ori se generează o adresă pentru accesarea memoriei, microcontrolerul înscrie în prealabil în bistabilele de tip  $D$  asociate portului starea  $I$ .

#### Lucrul în regim de magistrală de date.

Magistrala de date este bidirecțională și în consecință și funcționarea etajului va depinde de sensul de propagare a datelor.

**Pentru transferul de date spre exterior,** semnalul  $CONTROL=1$  va determina decuplarea celulei de tip  $D$  de etajul de ieșire și comanda acestuia din urmă va fi preluată de către bitul de date generat pe intrarea alternativă astfel: dacă bitul de date  $D_2=1$  atunci  $T_1$  blocat  $T_2$  conduce iar pentru  $D_2=0$ ,  $T_1$  conduce iar  $T_2$  este blocat.

**Pentru transferul datelor dinspre exterior spre interior** microcontrolerul generează semnalul  $CONTROL=0$ . Acest semnal de comandă pe de o parte va bloca tranzistorul  $T_2$  și prin comutatorul de cuplare/ separare va conecta ieșirea  $\bar{Q}$  a celulei  $D$  la poarta tranzistorului  $T_1$ . Deoarece înaintea oricărui transfer de date are loc o generare de adresă, și așa cum a fost



**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

precizat, în cazul generării unei adrese microcontrolerul înscrie în celulele  $D$  starea  $I$ . În felul acesta, prin cuplarea ieșirii  $\overline{Q}$  la poarta tranzistorului  $T_1$  acesta va fi blocat. Cum ambele tranzistoare sunt blocate (starea *tri-state*), nivelul de tensiune al pinului portului (și deci starea lui logică) va fi dictată de sursa externă de semnal cuplată la acest terminal, ce poate fi citită cu comanda *Read Pin*.

**Observație.** Semnalul de comandă *ALE* generat de microcontroler indică regimul de utilizare al portului  $P_0$  (magistrală de adresă când  $ALE=1$  și magistrală de date când  $ALE=0$ ).

**2.4.5. Particularitățile citirii porturilor**

Unele instrucțiuni de citire ale porturilor citesc semnalul aplicat pe pinul portului, altele, în schimb, citesc semnalul din bistabilul asociat pinului portului.

Instrucțiunile cu care se citește conținutul bistabilului asociat pinului sunt cunoscute sub denumirea de instrucțiuni **READ-MODIFY-WRITE**, deoarece cu ele este citit conținutul registrului de memorie asociat portului, asupra acestuia se efectuează o anumită operație (în funcție de instrucțiune) și rezultatul operației este rescris în registrul de memorie.

De exemplu:

-Instrucțiunea **ANL  $P_3, A$**  citește conținutul registrului de memorie asociat portului  $P_3$ , realizează un produs logic cu conținutul acumulatorului și scrie rezultatul înapoi în registrul portului.

-Instrucțiunea **ANL  $A, P_3$**  va citi nivelele logice aplicate pe pinii portului, va realiza un produs logic cu conținutul acumulatorului și va scrie rezultatul în acumulator.

**Observație:** Interpretarea rezultatelor citite de la pinii unui port trebuie făcută cu atenție pentru a evita interpretări greșite. De exemplu, dacă pe pinul unui port este conectat un tranzistor ce comandă o diodă electroluminiscentă pe care dorim s-o aprindem, atunci vom înscrie în bistabilul asociat pinului valoarea de  $1$  logic. Acest semnal va comanda tranzistorul care va intra în conducție și va determina coborârea tensiunii de pe pinul de comandă la  $0,65\text{ V}$  (cazul unui tranzistor bipolar). Dacă acum vom citi tensiunea de pe pin o vom interpreta ca fiind de  $0$  logic, ceea ce este de fapt greșit și se datorează sarcinii cuplate pe pin. Informația corectă legată de starea logică a pinului se află în bistabilul asociat acestuia și ea va trebui citită din acesta!

Instrucțiunile care realizează funcția **READ-MODIFY-WRITE** sunt următoarele:

Instr.	Exemplu
<b>ANL</b>	<b>ANL <math>P_1, A</math></b>
<b>ORL</b>	<b>ORL <math>P_3, A</math></b>
<b>XRL</b>	<b>XRL <math>P_1, A</math></b>
<b>JBC</b>	<b>JBC <math>P_{3.1}, \text{Etichetă}</math></b>
<b>CPL</b>	<b>CPL <math>P_{1.5}</math></b>
<b>INC</b>	<b>INC <math>P_1</math></b>
<b>DEC</b>	<b>DEC <math>P_1</math></b>
<b>DJNZ</b>	<b>DJNZ <math>P_2, \text{Etichetă}</math></b>
<b>MOV <math>P_{i,j}, C</math></b>	<b>MOV <math>P_{3,4}, C</math></b>
<b>CLR <math>P_{i,j}</math></b>	<b>CLR <math>P_{3.5}</math></b>
<b>SETB <math>P_{i,j}</math></b>	<b>SETB <math>P_{3.5}</math></b>

**Observație:** Ultimele trei instrucțiuni citesc de fapt întregul octet asociat portului, modifică biții menționați și rescriu octetul.





**2.4.6. Circuite de numărare – temporizare (TIMER)**

În numeroase aplicații practice apare necesitatea generării unor impulsuri cu perioadă și factor de umplere variabile (funcția *TIMER*), sau de numărare a unor impulsuri externe furnizate de un traductor (funcția *COUNTER*).

Aceste funcții pot fi realizate: prin soft (cu ajutorul unor bucle de întârziere); cu circuite digitale specializate sau utilizând circuitele de tipul *TIMER COUNTER* existente în structura *HW (HardWare)* a microcontrolerelor.

Pentru realizarea acestor funcții, necesare în aplicații de tipul comandă și control, microcontrolerelor familiei Intel sunt echipate cu circuite de tipul *TIMER-COUNTER*. De exemplu, microcontrolerelor *80(C)51-31* sunt echipate cu două circuite *Timer* notate cu  $T_0$  respectiv  $T_1$ , iar microcontrolerul *80552* este echipat cu trei circuite *Timer* notate cu  $T_0$ ,  $T_1$ ,  $T_2$ . *Timer*-ele  $T_0$  și  $T_1$  implementează funcțiile clasice de numărare și de generare de impulsuri iar *Timer*-ul  $T_2$  este prevăzut cu posibilitatea realizării funcțiilor *RELOAD COMPARE* și *CAPTURE* ceea ce mărește domeniile de utilizare a acestuia. În esență aceste circuite de tip *TIMER-COUNTER* sunt echipate cu un registru numărător de 16 biți, încărcabil în paralel cu o valoare inițială și care contorizează impulsurile ceasului intern (câte unul la fiecare ciclu mașină) în cazul funcției de generare de impulsuri (*Timer*) sau contorizează impulsuri aplicate din exterior pe un pin al portului paralel (funcția *COUNTER*). Constructorul a prevăzut posibilitatea programării (configurării) acestor circuite de către utilizator, ceea ce creează o flexibilitate extrem de utilă în aplicații.

Indiferent de funcția îndeplinită (*Timer* sau *Counter*), aceste circuite contorizează impulsuri (registru numărător își crește conținutul cu câte o unitate la fiecare impuls aplicat) începând de la valoarea inițială până în momentul atingerii valorii maxime. La următorul impuls contorizat apare fenomenul de depășire (trecerea numărătorului de exemplu din starea *FFFFH* în *0000H*), însoțit de generarea unei cereri de întrerupere.

Pentru a putea utiliza aceste circuite la rezolvarea unor probleme concrete cerute în aplicații, este necesar să înțelegem structura și modul lor de funcționare iar pentru aceasta trebuie să cunoaștem mecanismul lucrului cu întreruperi precum și modul de alegere a funcțiilor dorite prin setarea sau resetarea unor biți din cadrul registrelor interne prevăzute a controla aceste circuite.

**2.4.6.1. Mecanismul de lucru în întreruperi**

În timpul rulării normale a unui program (pe un microcalculator) pot apărea situații numite de excepție, a căror rezolvare să necesite întreruperea temporară a execuției programului principal și identificarea, încărcarea și rularea unei subrutine (aflată în memorie) pentru a trata respectiva situație de excepție. Sursele care generează situațiile de excepție pot fi interne sau externe procesorului. Cele externe sunt cunoscute sub denumirea de cereri de întrerupere. În aplicațiile practice ale circuitelor *Timer-Counter* este folosită pe scară largă capacitatea acestora de a genera cereri de întrerupere la depășirea capacității de numărare). Cererile de întrerupere ale circuitelor  $T_0$ ,  $T_1$ ,  $T_2$  sunt memorate în poziții predefinite ale unor registre de comandă speciale, numite fanioane și sunt notate cu  $TF_0$ ,  $TF_1$ , și  $TF_2$ . Pentru *Timer*-ele  $T_1$  respectiv  $T_0$  acești biți se află în registrul *TCON* din cadrul registrului cu funcții speciale (*SFR*)



**Structuri hardware si algoritmi specifici microsystemelor EMBEDDED**

<i>8FH</i>	<i>8EH</i>	<i>8DH</i>	<i>8CH</i>	<i>8BH</i>	<i>8AH</i>	<i>89H</i>	<i>88H</i>	
<i>TF<sub>1</sub></i>	<i>TR<sub>1</sub></i>	<i>TF<sub>0</sub></i>	<i>TR<sub>0</sub></i>	<i>IE<sub>1</sub></i>	<i>IT<sub>1</sub></i>	<i>IE<sub>0</sub></i>	<i>IT<sub>0</sub></i>	<i>TCON(88H)</i>
				↑	↑			

*TF<sub>1</sub>*, *TF<sub>0</sub>* –fanionul (*flag*) *Timer*-ului *T<sub>1</sub>* respectiv *T<sub>0</sub>* este setat ori de câte ori apare o depășire a conținutului registrului de numărare asociat circuitului *T<sub>1</sub>* respectiv *T<sub>0</sub>*. Acest bit va fi în mod automat *resetat* dacă microprocesorul ia în considerare cererea de întrerupere și derulează subrutina aferentă. În cazul în care microprocesorul controlerului nu ia în considerare cererea de întrerupere, acest bit rămâne setat și va trebui să fie *resetat* printr-o instrucțiune specială din program. (Registrul *TCON* este adresabil pe bit).

**Observație:** Fanionul *Timer*-ului *T<sub>2</sub>* se află într-un alt registru de comandă *TM<sub>2</sub>CON* (*IRCON* aflat la adresa *COH* din *SFR*).

Ca răspuns la o cerere de întrerupere, microprocesorul din structura microcontrolerului, salvează contextul programului principal (prin transferul conținutului registrelor interne în memoria stivă cu instrucțiunea *PUSH*), identifică adresa de start a rutinei de servire, o încarcă și o derulează. Rutina de servire a întreruperii se termină cu instrucțiunea *RETI* (**RETURN FORMS INTERRUPT**).

Microprocesorul recunoaște instrucțiunea *RETI* și ca urmare va restabili contextul programului principal (prin citirea memoriei stivă și reîncărcarea registrelor interne cu valorile corespunzătoare programului principal din momentul întreruperii, instrucțiunea *POP*) pe care îl va continua cu instrucțiunea următoare.

În cazul microcontrolerelor familiei Intel, rutinele de servire a cererilor de întrerupere generate de circuitele de tip *Timer-Counter* se află în memoria externă de date la adrese fixate de constructor.

Sursa cererii de întrerupere	Adresa de start a rutinei de servire
Depășire <i>Timer T<sub>0</sub></i>	<i>000BH</i>
Depășire <i>Timer T<sub>1</sub></i>	<i>001BH</i>
Depășire <i>Timer T<sub>2</sub></i>	<i>002BH</i>

**Observație:** Constructorul a prevăzut pentru fiecare rutină de servire câte 8 locații de memorie. În cazul în care rutina de servire (concepută în funcție de aplicație) presupune un program de dimensiuni mai mari atunci se procedează astfel: se încarcă programul rutinei de servire în memoria de date la o adresă arbitrară aleasă de utilizator (de exemplu *801BH*) iar la adresa *001BH* se mai scrie doar o instrucțiune de salt necondiționat la adresa de început a subrutinei (*LJMP 801BH*).

Din considerente de flexibilitate constructorul a prevăzut posibilitatea validării sau nu a unei cereri de întrerupere generată de un circuit *Timer-Counter*. Fiecare circuit *T<sub>0</sub>*, *T<sub>1</sub>*, *T<sub>2</sub>* are asociat câte un bit de validare individuală notat cu *ET<sub>0</sub>*, *ET<sub>1</sub>*, *ET<sub>2</sub>* (*Enable Timer*), care dacă are valoarea *zero logic* atunci cererea de întrerupere generată de acel *Timer* nu va fi luată în considerare de microprocesorul controlerului, iar dacă valoarea acestui bit este de *unu logic* atunci cererea de întrerupere va fi luată în considerare. Pe lângă acești biți de validare individuali,

**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

microcontrolerul mai este prevăzut și cu un bit de validare general *EA* (*Enable All*). Dacă *EA*=0 sunt ignorate de către microprocesor toate cererile de întrerupere, indiferent de sursa acestora, iar dacă *EA*=1 se admite lucrul în întreruperi.

Biții de validare menționați anterior sunt prevăzuți în anumite părți ale registrului special *IENO* (exemplu  $T_0$ ,  $T_1$ ,  $ENI$ ,  $T_2$ ). Acest registru face parte din registrele cu funcții speciale (*SFR*), este adresabil pe Bit ceea ce permite setarea sau *resetarea* individuală a acestor biți de validare (vezi Figura 2.4.6.1.).

Din prezentarea făcută până în prezent reiese faptul că fiecare din cele trei circuite *Timer-Counter* poate genera o cerere de întrerupere. Este posibil ca într-o anumită aplicație să fie folosite toate trei și atunci se pune firesc întrebarea ce se întâmplă dacă la un moment dat sunt generate simultan mai multe cereri de întrerupere. Constructorii microcontrolerului au prevăzut o tratare ierarhizată a acestora prin introducerea anumitor priorități. De fapt partea de *Hardware* a microcontrolerului examinează cererile tuturor surselor de întrerupere într-o anumită ordine conform unor priorități prestabilite. În cazul apariției simultane a mai multor cereri de

<i>AFH</i>	<i>AEH</i>	<i>ADH</i>	<i>ACH</i>	<i>ABH</i>	<i>AAH</i>	<i>A9H</i>	<i>A8H</i>	
<i>EA</i>	<i>WDT</i>	<i>ET<sub>2</sub></i>	<i>ES</i>	<i>ET<sub>1</sub></i>	<i>EX<sub>1</sub></i>	<i>ET<sub>0</sub></i>	<i>EX<sub>0</sub></i>	<i>IENO</i> ( <i>A8H</i> )
↑		↑		↑		↑		
<i>EA</i> =0      invalidează toate întreruperile								
<i>EA</i> =1      validează lucrul în întreruperi a microcontrolerului								
<i>ET<sub>2</sub></i> =0      invalidează cererile de întrerupere ale <i>Timer</i> -ului $T_2$								
<i>ET<sub>2</sub></i> =1      validează cererile de întrerupere ale <i>Timer</i> -ului $T_2$								
<i>ET<sub>1</sub></i> =0      invalidează cererile de întrerupere ale <i>Timer</i> -ului $T_1$								
<i>ET<sub>1</sub></i> =1      validează cererile de întrerupere ale <i>Timer</i> -ului $T_1$								
<i>ET<sub>0</sub></i> =0      invalidează cererile de întrerupere ale <i>Timer</i> -ului $T_0$								
<i>ET<sub>0</sub></i> =1      validează cererile de întrerupere ale <i>Timer</i> -ului $T_0$								

Figura 6.1. Funcțiile speciale ale registrului *IENO* (*A8H*)

întrerupere, acestea vor fi servite în ordinea priorității lor. Pentru a acorda o anumită flexibilitate și în această privință, constructorul oferă utilizatorului un anumit control asupra priorităților. Utilizatorul va putea schimba prioritatea cererii de întrerupere a unui circuit *Timer-Counter* prin setarea respectiv *reset*-area biților de prioritate atașați acestora. Biții de control al priorității  $PT_2$ ,  $PT_1$  și  $PT_0$  sunt atașați *Timer*-elor  $T_2$ ,  $T_1$ ,  $T_0$  și se află în registrul de control al priorităților *IP* din cadrul setului de registre speciale (*SFR*). Prin setarea respectiv *reset*-area acestor biți se va putea schimba prioritatea unei cereri de întrerupere.

<i>BFH</i>	<i>DEH</i>	<i>BDH</i>	<i>BCH</i>	<i>BBH</i>	<i>BAH</i>	<i>B9H</i>	<i>B8H</i>	
		<i>PT<sub>2</sub></i>	<i>PS</i>	<i>PT<sub>1</sub></i>	<i>PX<sub>1</sub></i>	<i>PT<sub>0</sub></i>	<i>PX<sub>0</sub></i>	<i>IP</i> ( <i>B8H</i> )
↑		↑		↑		↑		

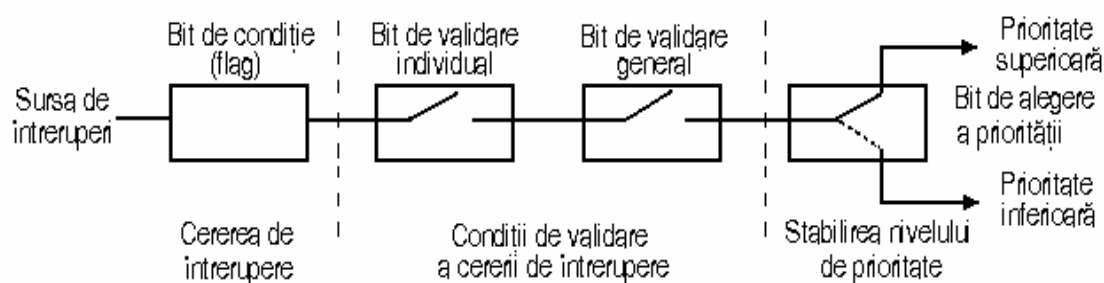
**Exemplu:** În cazul a două cereri de întrerupere simultane de la  $T_1$  și  $T_0$  cu  $PT_1=1$   $PT_0=0$ , va fi servită prima dată cererea de întrerupere pentru *Timer*-ul  $T_1$  și apoi pentru *Timer*-ul  $T_0$ . Dacă cele două cereri de întrerupere simultană apar atunci când prin program avem setat  $PT_1=1$  și  $PT_0=1$  atunci va fi servită la început cererea de întrerupere a *Timer*-lui  $T_0$  și apoi cea a *Timer*-ului  $T_1$ , pentru că în acest caz va conta prioritatea stabilită prin *Hardware*.

**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

Am putea rezuma mecanismul de generare a unui asemenea întrerupere cu desenul din Figura 2.4.6.3a cu precizarea biților ce intervin (Figura 2.4.6.3b).

$PT_2 = 0$  prioritate inferioară  
 $PT_2 = 1$  prioritate superioară  
 $PT_1 = 0$  prioritate inferioară  
 $PT_1 = 1$  prioritate superioară  
 $PT_0 = 0$  prioritate inferioară  
 $PT_0 = 1$  prioritate superioară

Figura 2.4.6.2. Funcțiile speciale ale registrului IP (B8H)



a)

Sursa cererii de întrerupere	Bitul de condiție (flag)	Bitul de validare individual	Bitul de validare general	Bitul pentru stabilirea priorității	Adresa mecanismului de întrerupere
Depășire Timer $T_0$	$TF_0$	$ET_0$	$EA$	$PT_0$	$000BH$
Depășire Timer $T_1$	$TF_1$	$ET_1$	$EA$	$PT_1$	$001BH$
Depășire Timer $T_2$	$TF_2$	$ET_2$	$EA$	$PT_2$	$002BH$

b)

Figura 2.4.6.3. a) Condiții pentru generarea unei cereri de întrerupere. b) Tabelul cu biții de condiționare a întreruperilor la circuitele Timer-Counter

### 2.4.6.2. Funcții de bază realizate cu circuitele de numărare-temporizare $T_0$ și $T_1$

Circuitele de numărare-temporizare, contorizează impulsurile ceasului intern (câte unul la fiecare ciclu mașină) atunci când realizează funcția de temporizare (Timer) sau contorizează impulsurile aplicate din exterior atunci când realizează funcția de numărare (Counter). În ambele cazuri, circuitul poate lucra în regim comandat sau necomandat din exterior și, în final rezultă patru posibilități de funcționare:

- generator de impulsuri, programabil;

**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

- generator de impulsuri, programabil comandat din exterior;
- numărător de impulsuri, programabil;
- numărător de impulsuri, programabil comandat din exterior.

Alegerea funcției (numărare sau temporizare), a modului de comandă (intern sau extern) precum și pornirea sau oprirea circuitelor se face prin setarea sau resetarea unor biți de comandă asociați *Timer*-ului. Astfel în registrul de comandă *TCON*, în porțile 5 și 7 (*TCON*<sub>4</sub> respectiv *TCON*<sub>6</sub>) se află biții de control a funcționării *TR*<sub>0</sub> respectiv *TR*<sub>1</sub>

<i>TF</i> <sub>1</sub>	<i>TR</i> <sub>1</sub>	<i>TF</i> <sub>0</sub>	<i>TR</i> <sub>0</sub>	<i>IE</i> <sub>1</sub>	<i>IT</i> <sub>1</sub>	<i>IE</i> <sub>0</sub>	<i>IT</i> <sub>0</sub>	<i>TCON</i> (89H)
------------------------	------------------------	------------------------	------------------------	------------------------	------------------------	------------------------	------------------------	-------------------

↑  
*TR*<sub>1</sub>, *TR*<sub>0</sub> = 0 circuitul *Timer* corespunzător este oprit;

↑  
*TR*<sub>1</sub>, *TR*<sub>0</sub> = 1 circuitul *Timer* corespunzător este pornit;

Pentru alegerea funcției îndeplinite, a modului de comandă precum și a modului de lucru sunt utilizați câte patru biți pentru fiecare circuit *Timer* (*T*<sub>1</sub> și *T*<sub>0</sub>) grupați împreună în registrul special de control al modului de lucru *TMOD*

<i>TMOD</i> <sub>7</sub>	<i>TMOD</i> <sub>6</sub>	<i>TMOD</i> <sub>5</sub>	<i>TMOD</i> <sub>4</sub>	<i>TMOD</i> <sub>3</sub>	<i>TMOD</i> <sub>2</sub>	<i>TMOD</i> <sub>1</sub>	<i>TMOD</i> <sub>0</sub>	<i>TMOD</i> (89H)
<i>Gate</i> <sub>1</sub>	<i>C / TH</i>	<i>M</i> <sub>1</sub>	<i>M</i> <sub>0</sub>	<i>Gate</i> <sub>0</sub>	<i>C / TH</i>	<i>M</i> <sub>1</sub>	<i>M</i> <sub>0</sub>	

*Timer 1*

*Timer 0*

Funcția îndeplinită se alege cu biții notați cu C/T după cum urmează:

*C/T*=0 funcția *Timer* (numără impulsurile oscilatorului  $f_{osc}/12$ ;

*C/T*=1 funcția *Counter*, numără impulsurile aplicate din exterior. Aceste impulsuri se aplică la terminalul *P*<sub>3.4</sub> al portului paralel *P*<sub>3</sub> dacă sunt destinate circuitului *T*<sub>0</sub> și pe terminalul *P*<sub>3.5</sub> al portului *P*<sub>3</sub> dacă sunt destinate circuitului *T*<sub>1</sub>.

Cu bitul notat *Gate* se va putea autoriza sau nu comanda din exterior a circuitului. Semnalul extern de comandă destinat circuitului *T*<sub>0</sub> se aplică pe terminalul *P*<sub>3.2</sub> al portului *P*<sub>3</sub> iar cel destinat circuitului *T*<sub>1</sub> pe terminalul *P*<sub>3.3</sub> al portului *P*<sub>3</sub>.

Pentru:

- *Gate*=0, semnalul extern de comandă nu are nici o influență asupra funcționării circuitului. Controlul funcționării este asigurat doar de bitul *TR*<sub>1</sub> respectiv *TR*<sub>0</sub>. *Timer*-ul corespunzător va fi declanșat dacă *TR*<sub>0</sub> (respectiv *TR*<sub>1</sub>) este 1 și va fi oprit dacă *TR*<sub>0</sub> (respectiv *TR*<sub>1</sub>) este 0;
- *Gate*=1, în acest caz dacă circuitul este pornit (*TR*<sub>0</sub> respectiv *TR*<sub>1</sub> sunt în starea 1) atunci comanda funcționării este preluată de semnalul aplicat din exterior ( pe terminalele *P*<sub>3.2</sub> respectiv *P*<sub>3.3</sub>) în sensul că atâta timp cât acest semnalul este pe nivelul logic unu circuitul funcționează în regim de lucru ales prin bitul *C/T* (*Timer* sau *Counter*)

**Structuri hardware si algoritmi specifici microsystemelor EMBEDDED**

Realizarea acestor funcții de comandă a impus existența în fața registrului numărător a unei scheme cu porți logice și comutatoare. În Figura 6.4. este prezentată această schemă pentru circuitul  $T_0$ . Prin urmărirea atentă a schemei se vor înțelege mai ușor comenzile descrise anterior.

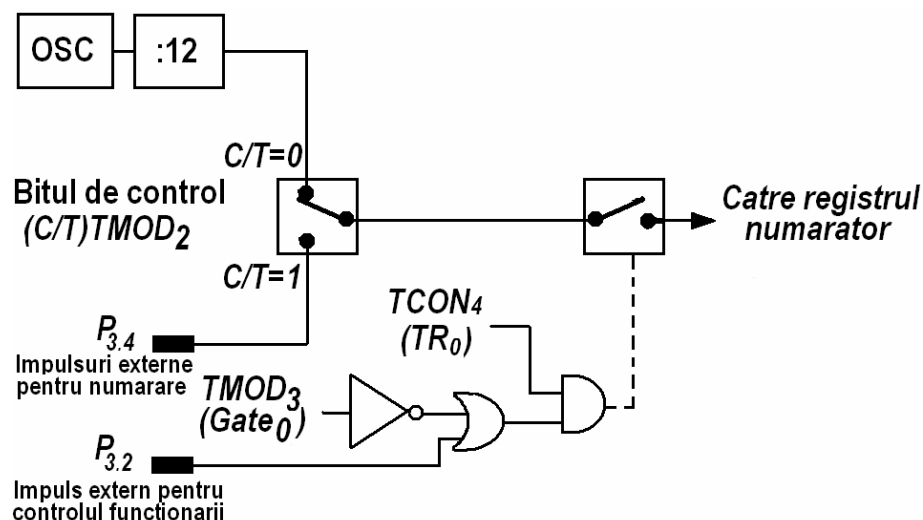


Figura 2.4.6.4 Modul 1 de lucru (16-bit Timer/Counter)

În tabelul 2.4.6.1 sunt prezentate comenzile necesare pentru cele patru moduri de funcționare (ilustrată numai pentru circuitul  $T_0$ )

Exemplu: În regimul generator programabil de impulsuri comandat din exterior, circuitul Timer poate fi folosit pentru măsurarea duratei unor impulsuri cu o precizie de  $1\mu\text{S}$ . (vezi Figura 6.5.). Registrul numărător va contoriza impulsurile de  $f_{\text{osc}}/12=1\mu\text{S}$  ale ceasului intern pe durata  $t_2-t_1$  cât semnalul de comandă este pe nivelul *HIGH*.



**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**Tabelul 2.4.6.1 Comenzile necesare celor patru moduri de funcționare a circuitului  $T_0$ 

Funcția îndeplinită	Starea biților de comandă			Impulsuri care se numără	Comanda externă	Observații
	TMOD <sub>2</sub> (C/T)	TMOD <sub>3</sub> (Gate 0)	TMOD <sub>4</sub> (TR <sub>0</sub> )			
Generator programabil de impulsuri	0	0	1	$f_{osc}/12$	Nu	Contorizează impulsurile ceasului intern
Generator programabil de impulsuri comandat din exterior	0	1	1	$f_{osc}/12$	Semnalul de comandă extern aplicat pe $P_{3.2}$	Contorizează impulsurile ceasului intern pe durata cât semnalul de pe $P_{3.2}$ este 1
Numărător programabil de impulsuri	1	0	1	Extern de pe $P_{3.4}$	Nu	Contorizează impulsuri din afară aplicate pe $P_{3.4}$ $(f_{max} = \frac{f_{osc}}{24})$
Numărător programabil de impulsuri comandat din exterior	1	1	1	Extern de pe $P_{3.4}$	Semnalul de comandă extern aplicat pe $P_{3.2}$	Contorizează impulsuri din afară aplicate pe $P_{3.4}$ pe durata cât semnalul de comandă de pe $P_{3.2}$ este 1
	x	x	0	x	x	Nu contorizează nimic (nu funcționează)

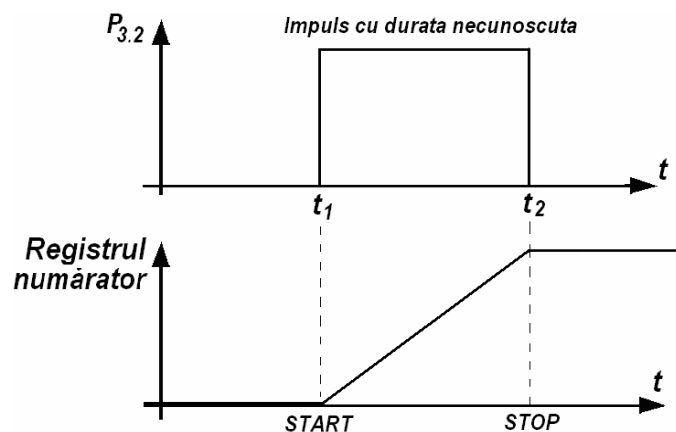


Figura 6.5. Măsurarea duratei unui impuls dreptunghiular



**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

Dacă pe frontul negativ al semnalului de comandă (momentul  $t_2$ ) se declanșează o cerere de întrerupere, atunci în rutina de servire va fi posibilă citirea conținutului registrului numărator din momentul  $t_2$  ceea ce va permite determinarea duratei impulsului.

- Cu biții  $M_1$  și  $M_0$  din registrul de comandă a modului de lucru TMOD, fiecare circuit *Timer* va putea fi programat să lucreze într-unul din următoarele patru moduri posibile:

$M_1$	$M_0$	
0	0	13 bit <i>Timer Counter</i> $THx$ este un numărator de 8 biți $TLx$ este un factor de prescalare de 5 biți
0	1	16 bit <i>Timer Counter</i> $THx$ este un numărator de 8 biți $TLx$ este un factor de prescalare de 8 biți
1	0	8 bit <i>Timer Counter with Auto-Reload</i> $THx$ numărator de 8 biți $TLx$ folosit pentru păstrarea valorii de reîncărcare
1	1	8 bit <i>Timer dublu</i> (mod în care nu poate fi folosit decât Timerul 0) $TH_0$ lucrează ca un <i>Timer</i> de 8 biți $TL_0$ lucrează în regim de <i>Timer Counter</i> de 8 biți

Schemele corespunzătoare celor patru moduri de lucru sunt prezentate în Figura 6.6, Figura 2.4.6.7, Figura 2.4.6.8 respectiv Figura 2.4.6.9.

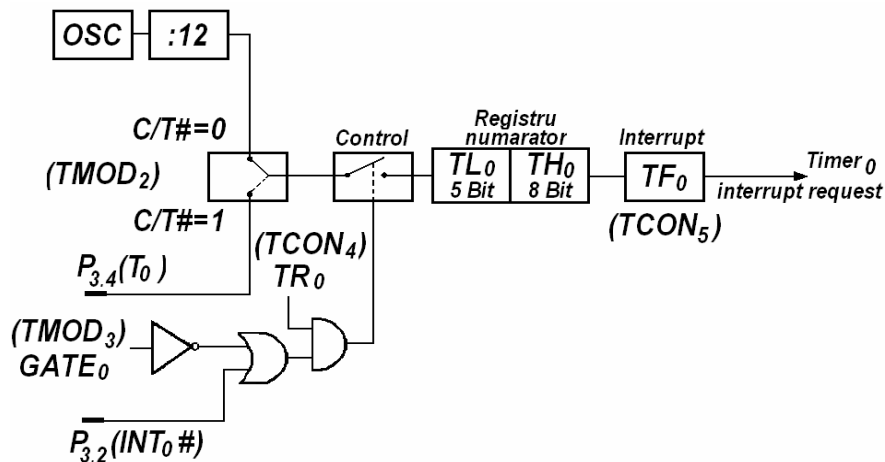


Figura 2.4.6.6. Modul 0 de lucru (13-bit Timer/Counter)

În Figura 2.4.6.6. este ilustrat modul de lucru 0 pentru *Timer*-ul  $T_0$ . Registrul numărator este configurat ca un divizor cu 13 format dintr-un numărator de 8 biți ( $TH_0$ ) și un divizor cu 5 ( $TL_0$ ). La fiecare depășire (trecerea din  $1FFFH$  în  $0000H$  se setează bitul  $TF_0$ . Generarea unei cereri de întrerupere, la sfârșitul derulării rutinei de servire a unei cereri de întrerupere Hardware, microcontrolerul resetează în mod automat bitul  $TF_0$ . Dacă cererea de întrerupere nu

**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

este generată (din considerente de validare sau de prioritate) bitul  $TF_0$  trebuie resetat prin soft. După o depășire, registrul numărator începe din nou contorizarea impulsurilor de la  $0000H$  până la o nouă depășire.

În cazul modului de lucru 1 ambele registre lucrează ca numărătoare de 8 biți. Dacă se dorește realizarea unui factor de divizare oarecare, acest lucru se poate obține prin încărcarea număratorului cu o anumită valoare inițială, astfel încât numărul ce rămâne până la realizarea depășirii să corespundă factorului de divizare dorit. De exemplu dacă dorim o divizare cu  $418=01A2H$  atunci valoarea de inițializare va fi:  $FFFFH-01A2H=FE5DH$ . Pentru inițializarea în registrul  $TH_0$  se va încărca octetul mai puțin semnificativ al valorii inițiale ( $5DH$  în cazul nostru) în registrul  $TL_0$  valoarea mai semnificativă a valorii inițiale ( $FEH$  în cazul nostru). Observațiile referitoare la întreruperi, prezentate pentru modul 0 rămân valabile și în acest caz.

În Figura 2.4.6.7. este prezentat un mod de lucru al numărătoarelor pe 8 biți cu posibilitatea de autoinițializare (modul 2).

Registrul  $TL_0$  este folosit ca numărator iar registrul  $TH_0$ , este utilizat pentru a memora valoarea de inițializare. La fiecare depășire a conținutului registrului  $TL_0$ , va fi setat *flag*-ul  $TF_0$  (și va fi generată o cerere de întrerupere în funcție de starea biților de validare și de prioritate) și va fi reîncărcat registrul  $TL_0$  cu conținutul registrului  $TH_0$ .

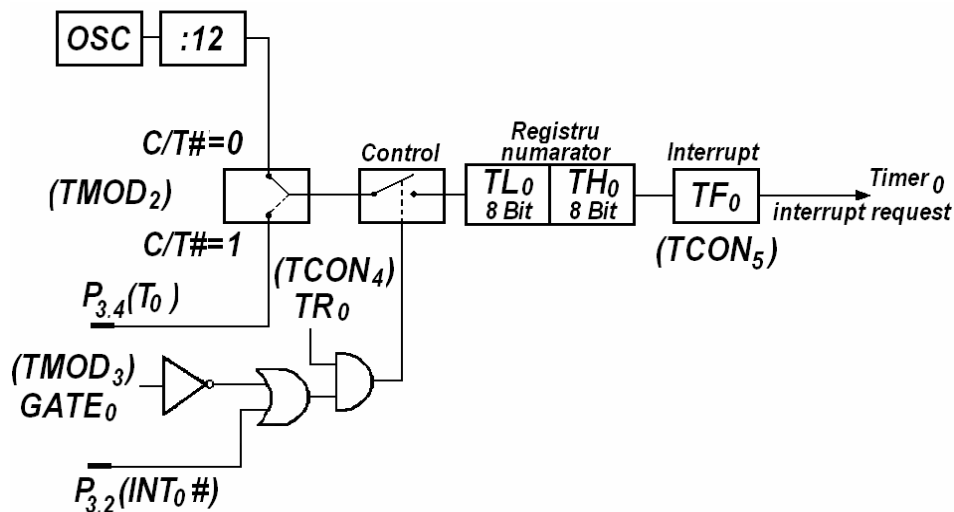


Figura 2.4.6.7. Modul 1 de lucru (16-bit Timer/Counter)

Modul 3 de lucru al numărătoarelor este prezentat în Figura 2.4.6.8. Cele două registre de numărare sunt conectate ca numărătoare independente pe 8 biți. Registrul numărator  $TH_0$  se poate folosi numai în regim de *Timer*, funcționarea sa fiind controlată de bitul  $TR_1$  (TCON<sub>6</sub>) iar la depășire (trecerea din starea  $FFH$  în  $00H$ ) setează bitul  $TF_1$  (TCON<sub>7</sub>).

Registrul  $TL_0$  este utilizat atât ca generator de impulsuri (*Timer*) cât și ca numărator. În ambele cazuri are și posibilitatea de comandă din exterior (după modelul cunoscut, prezentat anterior).

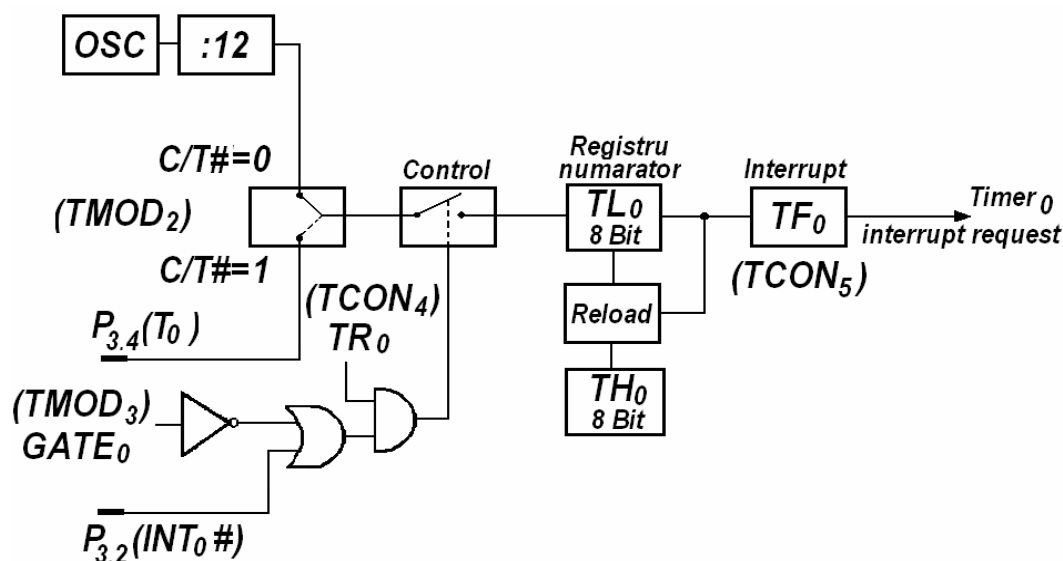
**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

Figura 2.4.6.8. Modul 2 de lucru (8-bit Timer/Counter with Auto-Reload)

La depășire el va seta bitul  $TF_0$  (TCON<sub>5</sub>), generarea unei cereri de întrerupere, în cazul registrului  $TL_0$ , este însă condiționată de starea biților de validare ( $EA$  și  $ET_0$ ) iar în cazul registrului  $TH_0$  de starea biților ( $EA$  și  $ET_1$ ), precum și de priorități. În cazul în care nu este servită cererea de întrerupere se va avea în vedere setarea prin soft a *flag*-urilor.

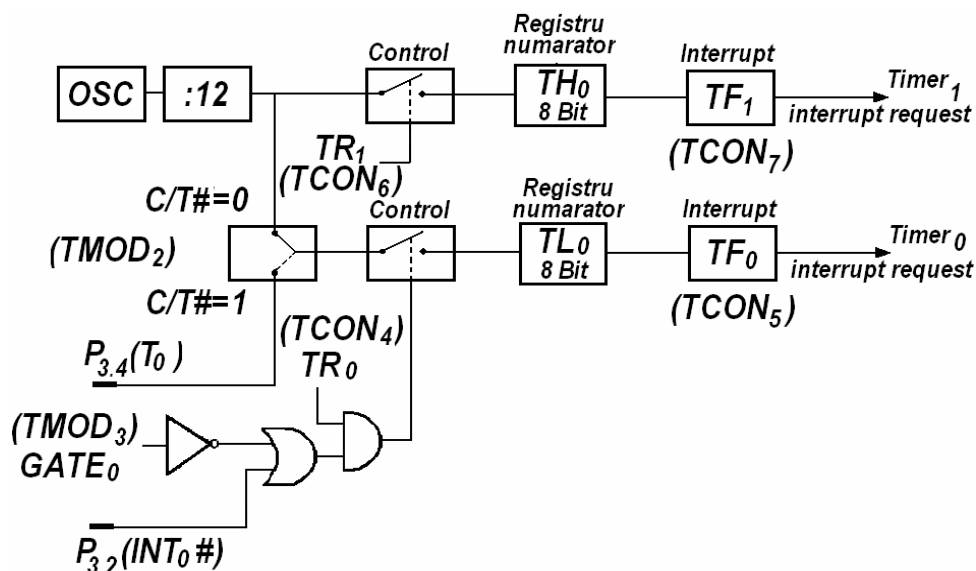
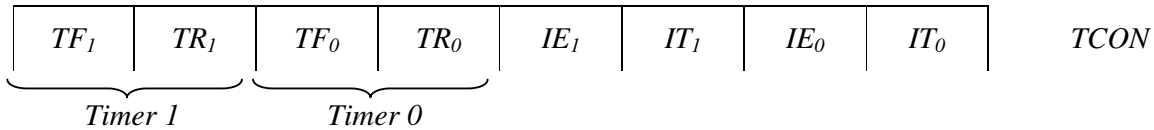


Figura 2.4.6.9. Modul 3 de lucru (Two-8 bit Timer/Counter)

**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

Pe lângă registrul *TMOD* ai cărui biți permit alegerea modului de lucru a *Timer*-ului  $T_0$  și  $T_1$  cu primii patru biți ai registrului de comandă *TCON* putem avea un control suplimentar asupra *Timer*-ului  $T_0$  și  $T_1$ .

Structura registrului de comandă *TCON* este următoarea:



În primele patru poziții ale registrului *TCON* se află biții de comandă *START-STOP* ( $TR_i$ ), respectiv biții indicatori de depășire ( $TF_i$ ), cu următoarea semnificație:

- $TR_0$  (respectiv  $TR_1$ ) = 1 – pornește funcția de *Timer-Counter*;
- $TR_0$  (respectiv  $TR_1$ ) = 0 – oprește funcția de *Timer-Counter*;
- $TF_0$  (respectiv  $TF_1$ ) – bitul indicator de depășire (*Overflow Flag*).

**Observație:** Ultimii patru biți din acest registru nu se referă la circuitele *Timer* ci la întreruperile externe!

**2.4.6.2.1. Semnificația comenzilor**

Pentru a se înțelege mai bine semnificația comenzilor să considerăm următorul exemplu: Dorim să măsurăm durata unui impuls. În acest scop vom folosi *Timer*-ul 0 în regim de lucru ca generator de impulsuri comandat. Semnalul de comandă va fi chiar impulsul a cărui durată urmează a fi măsurată și îl vom aplica pe pinul 2 al portului 3 ( $P_{3.2}$ ) așa cum este indicat Figura 2.4.6.10.

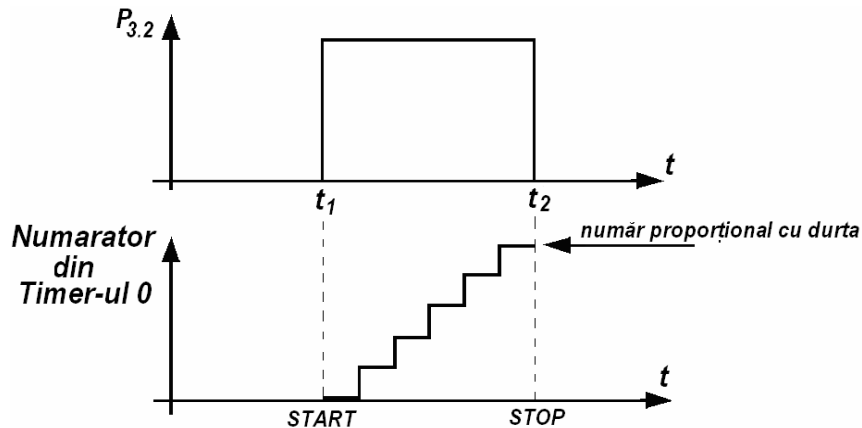


Figura 0.10. Măsurarea duratei unui impuls dreptunghiular

Comenzile necesare pentru programarea *Timer*-ului  $T_0$ , pentru exemplul de mai sus, vor fi specificate prin biții registrului *TMOD* astfel:

- $TMOD_3 = GATE = 1$  *Timer*-ul va funcționa doar dacă pe borna  $P_{3.2}$  avem nivel logic unu și dacă  $TR_0 = 1$ ;
- $TMOD_2 = C/T \# = 0$  Mod de lucru *Timer* cu numărarea ceasului intern  $f_{osc}/12$ ;

**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

- $TMOD_1 = 0$
  - $TMOD_0 = 1$
- } Alegem modul 1 de lucru 16-bit Timer-Counter

Durata impulsului va fi egală cu numărul de impulsuri înregistrate, înmulțite cu perioada unui impuls ( $1\mu s$  dacă  $f_{osc} = 12MHz$ ).

Pentru a putea citi conținutul numărătorului la sfârșitul impulsului a cărui durată a fost măsurată, va trebui să folosim frontul negativ (comanda *STOP*) a impulsului aplicat din exterior pentru a genera o cerere de întrerupere.

**2.4.6.2.2. Mecanismul de generare a unei cereri de întrerupere**

Pentru validarea întreruperilor, respectiv pentru asigurarea unui anumit nivel de prioritate, sunt utilizate registrele speciale: *IE* (*Interrupt Enable*) și *IP* (*Interrupt Priority*). Structura registrului *IE* este următoarea:

<i>AFH</i>	<i>AEH</i>	<i>ADH</i>	<i>ACH</i>	<i>ABH</i>	<i>AAH</i>	<i>A9H</i>	<i>A8H</i>	
<i>EA</i>	<i>WDT</i>	<i>ET<sub>2</sub></i>	<i>ES</i>	<i>ET<sub>1</sub></i>	<i>EX<sub>1</sub></i>	<i>ET<sub>0</sub></i>	<i>EX<sub>0</sub></i>	<i>IE (A8H)</i>

Semnificațiile biților din registru sunt:

- $EA = 0$  – invalidare generală a întreruperilor;
- $EA = 1$  – validare generală a întreruperilor;
- $ET_1 = 0$  – invalidează întreruperile la *Timer 1*;
- $ET_1 = 1$  – validează întreruperile la *Timer 1*;
- $ET_0 = 0$  – invalidează întreruperile la *Timer 0*;
- $ET_0 = 1$  – validează întreruperile la *Timer 0*.

**Observație:** Semnificația celorlalți biți din registrul *IE* va fi explicată ulterior.

Structura registrului *IP* este următoarea:

<i>BFH</i>	<i>BEH</i>	<i>BDH</i>	<i>BCH</i>	<i>BBH</i>	<i>BAH</i>	<i>B9H</i>	<i>B8H</i>	
–	–	<i>PT<sub>2</sub></i>	<i>PS</i>	<i>PT<sub>1</sub></i>	<i>PX<sub>1</sub></i>	<i>PT<sub>0</sub></i>	<i>PX<sub>0</sub></i>	<i>IP (B8H)</i>

Semnificațiile biților din registru sunt:

- $PT_1 = 0$  – prioritate inferioară la *Timer-ul 1*;
- $PT_1 = 1$  – prioritate superioară la *Timer-ul 1*;
- $PT_0 = 0$  – prioritate inferioară la *Timer-ul 0*;
- $PT_0 = 1$  – prioritate superioară la *Timer-ul 0*.

**Observații:**

- *Timer-ul 1* este mai prioritar decât *Timer-ul 0*;
- Schema de generare a unei întreruperi interne datorită depășirii la *Timer-ul 0* este prezentată în Figura 6.11.

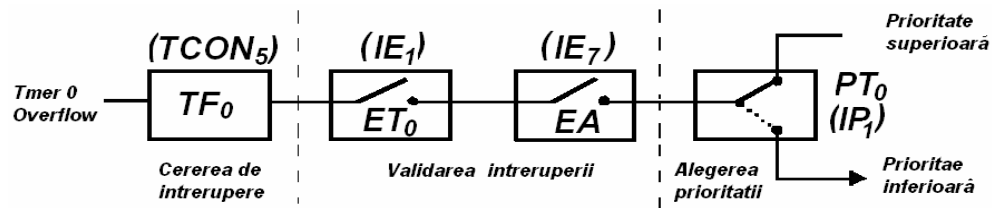


Figura 2.4.6.11. Schema de generare a unei cereri de întrerupere

### 2.4.6.3. Exemple de programe

#### 6.3.1. Exemplul 1. Generator de impulsuri

Să se programeze *Timer-ul 0*, astfel încât după fiecare  $1,5ms$ , ieșirea cu numărul 5 a portului 1 ( $P_{1,5}$ ) să fie inversată. Frecvența oscilatorului este de  $12MHz$ . Rezolvarea se face în mai multe etape:

1. Vom preciza modul de lucru ales pentru *Timer 0* prin precizarea biților din registrul *TMOD*:

Gate	C / TH	M <sub>1</sub>	M <sub>0</sub>	Gate	C / TH	M <sub>1</sub>	M <sub>0</sub>	
X	X	X	X	0	0	0	1	<i>TMOD</i>
Timer 1				Timer 0				

S-a ales *Timer-ul 0*, în regimul 1 necomandat (16-bit *Timer*).

2. Calculul valorilor de inițializare:

- Timpul dorit pentru inversarea ieșirii este de  $t=1,5ms$ . Numărul ciclurilor mașină necesare (numărătorul *Timer*-ului se incrementează la fiecare ciclu mașină):

$$z = \frac{t}{t_{ciclu}} = \frac{1,5ms}{1\mu s} = 1500 = 05DCH$$

- Valoarea de inițializare a numărătorului *Timer*-lui 0 este:

$$n = 10000 - 05DCH = 0FA24H$$

- Valorile de inițializare ale registrelor numărătorului sunt:

$$TH0 = 24H \text{ și } TL0 = 0FAH$$

3. Organigramele programelor sunt prezentate în Figura 6.12.

4. Programul principal pentru Exemplul 1 este următorul:

Inițializări:

TH0 EQU 8CH

## Structuri hardware si algoritmi specifici microsistemelor EMBEDDED

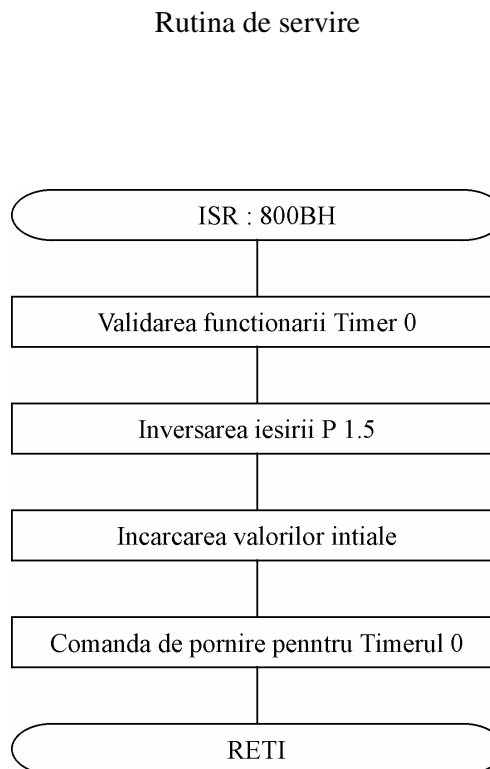
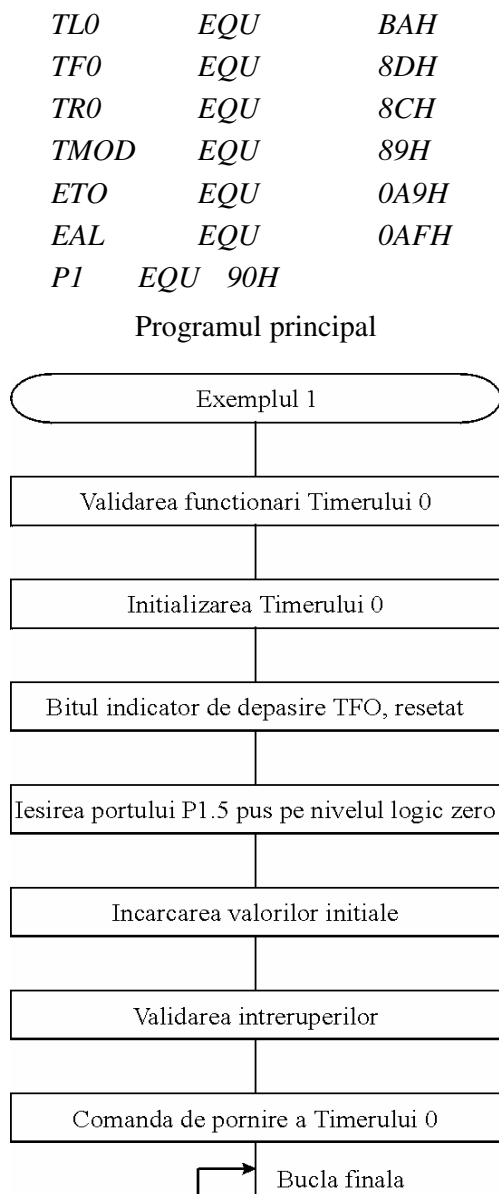


Figura 0.12. Organigramele programului principal și a rutinei de servire

Programul principal:

<i>ORG</i>	<i>8100H;</i>	adresa de start a programului
<i>CLR</i>	<i>TR0;</i>	invalidarea funcționării <i>Timer</i> -ului 0
<i>ANL</i>	<i>TMOD, #11110000B;</i>	resetarea biților de control ai <i>Timer</i> -ului 0
<i>ORL</i>	<i>TMOD, #00000001B;</i>	setarea biților de control <i>Timer 0</i> pentru modul de funcționare ales
<i>CLR</i>	<i>TFO;</i>	resetarea bitului indicator de depășire



UNIUNEA EUROPEANĂ


 MINISTERUL MUNCII, FAMILIEI ȘI  
 PROTECȚIEI SOCIALE  
 AMFOSDRU

 FONDUL SOCIAL EUROPEAN  
 POS DRU  
 2007-2013

 INSTRUMENTE STRUCTURALE  
 2007-2013



**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

<i>MOV</i>	<i>TH0,</i>	<i>#24H;</i>	
<i>MOV</i>	<i>TL0,</i>	<i>#0FAH</i>	
<i>CLR</i>	<i>P1.5;</i>		ieșirea portului pus pe zero logic
<i>SETB</i>	<i>ET0;</i>		validarea întreruperilor pentru <i>Timer 0</i>
<i>SETB</i>	<i>EAL;</i>		validarea întreruperilor generale
<i>SETB</i>	<i>TR0;</i>		pornirea <i>Timer 0</i>
<i>LOOP:</i>	<i>LJMP</i>		<i>LOOP;</i> buclă infinită pentru simularea programului principal

Subrutina de servire:

<i>ORG</i>	<i>800BH;</i>		adresa de start a rutinei de servire = adresa vectorului de întreruperi <i>000BH + 8000H</i>
<i>CLR</i>	<i>TR0;</i>		invalidarea funcționării <i>Timer 0</i>
<i>CPL</i>	<i>P1.5;</i>		inversarea ieșirii portului
<i>MOV</i>	<i>TH0,</i>	<i>#24H;</i>	încarcă valorile inițiale pentru <i>Timer-ul 0</i>
<i>MOV</i>	<i>TL0,</i>	<i>#0FAH;</i>	
<i>SETB</i>	<i>TR0;</i>		pornirea <i>Timer-ului 0</i>
<i>RETI</i>			
<i>END.</i>			

**2.4.6.3.2. Exemplul 2. Numărător de impulsuri**

Cu ajutorul numărătorului  $TL_0$  al *Timer-ului 0* se vor număra impulsuri aplicate din exterior la  $P_{3.4}$ . La fiecare o sută de impulsuri numărate va fi inversat nivelul logic pe  $P_{1.3}$ .

Pentru rezolvare vom indica doar programul, celelalte etape ilustrate în exemplul precedent urmând să fie făcute de cititorii interesați.

Inițializări:

<i>TM0</i>	<i>EQU</i>	<i>8CH</i>
<i>TL0</i>	<i>EQU</i>	<i>8AH</i>
<i>TFO</i>	<i>EQU</i>	<i>8DH</i>
<i>TR0</i>	<i>EQU</i>	<i>8CH</i>
<i>TMOD</i>	<i>EQU</i>	<i>89H</i>
<i>ETO</i>	<i>EQU</i>	<i>0A9H</i>
<i>EAL</i>	<i>EQU</i>	<i>0AFH</i>
<i>P1</i>	<i>EQU</i>	<i>90H</i>

Programul principal este

<i>ORG</i>	<i>8100H;</i>	adresa de start a programului
<i>CLR</i>	<i>TR0;</i>	invalidarea funcționării <i>Timer-ului 0</i>
<i>CLR</i>	<i>TFO;</i>	resetarea bitului indicator de depășire
<i>ANL</i>	<i>TMOD,</i>	<i>#11110000B;</i> resetarea biților de control ai <i>Timer-ului 0</i>
<i>ORL</i>	<i>TMOD,</i>	<i>#00000110B;</i> setarea biților de control ai <i>Timer-ului 0</i> pentru modul de funcționare ales
<i>MOV</i>	<i>TH0,</i>	<i>#9CH;</i> valoarea de inițializare a <i>Timer-ului 0</i>
<i>MOV</i>	<i>TL0,</i>	<i>#9CH;</i> valoarea de start a <i>Timer ului 0</i>
<i>CLR</i>	<i>P1.3;</i>	punerea ieșirii portului pe zero logic
<i>SETB</i>	<i>ET0;</i>	setarea bitului individual de validare a întreruperilor pentru <i>Timer-ul 0</i>



**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

SETB      TR0;      comandă de pornire a *Timer*-ului 0  
 LOOP:    LJMP      LOOP;      buclă infinită ce ar simula programul principal

Rutina de servire a întreruperilor:

ORG      800BH;      adresa de start a rutinei  
 CPL      P1.3;      inversarea ieșirii portului  
 RETI  
 END.

**2.4.6.4. Circuitul Timer  $T_2$** **2.4.6.4.1. Structura hardware**

Schema bloc a circuitului *Timer T<sub>2</sub>* este prezentată în Figura 2.4.6.13.

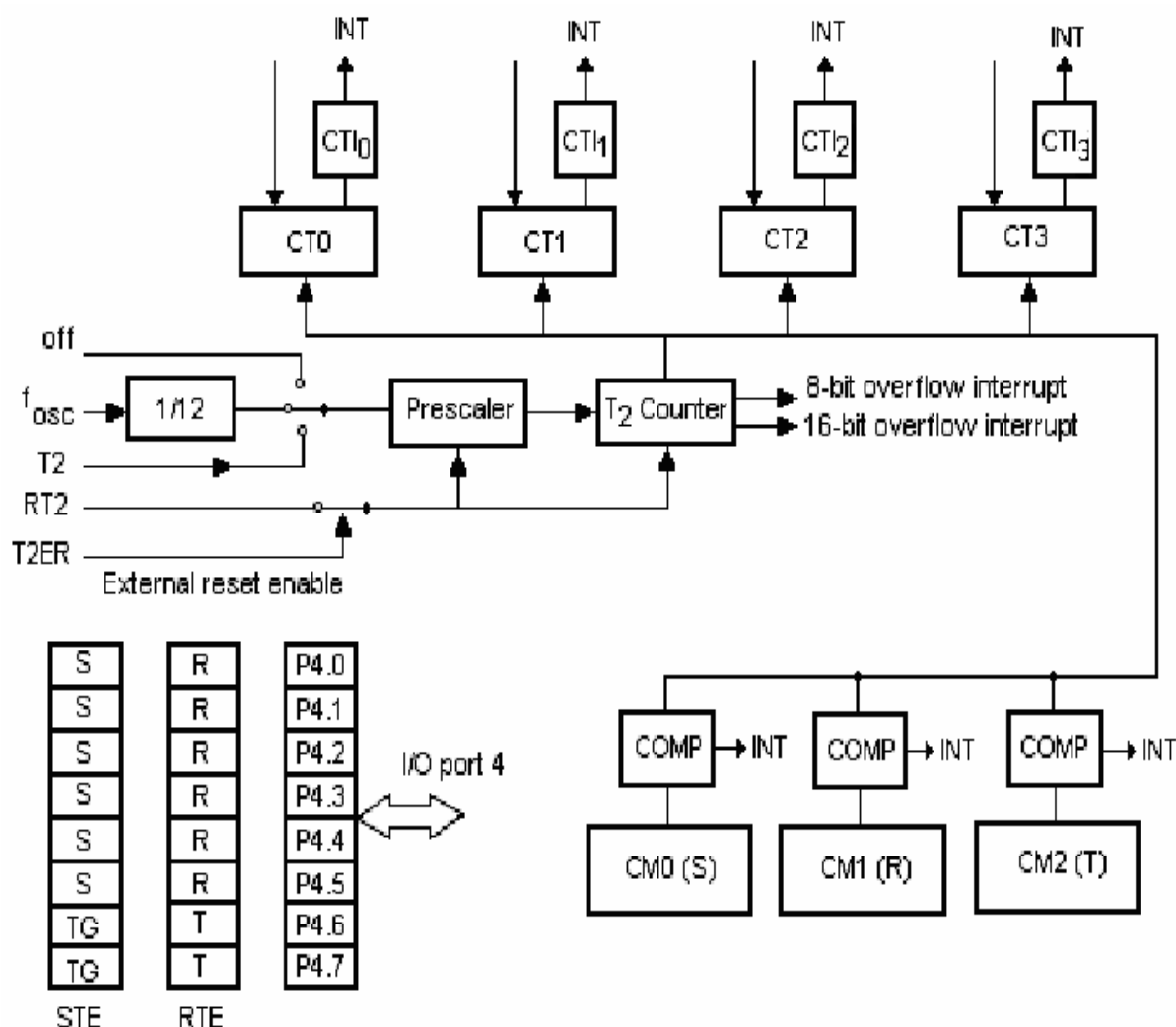


Figura 2.4.6.13. Schema bloc a circuitului *Timer T<sub>2</sub>*

Schema conține următoarele blocuri funcționale:

- un numărator de prescalare programabil pentru a divide cu 1, 2, 4 și 8;

**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

- un numărător de 16 biți (format din două registre de numărare și anume  $TMH_2$  pentru octetul superior și  $TML_2$  pentru octetul inferior);
- patru registre de 16 biți ce îndeplinesc funcția *CAPTURE* cu logica aferentă ( $CT_0$ ,  $CT_1$ ,  $CT_2$  și  $CT_3$ );
- trei registre de 16 biți ce îndeplinesc funcția *COMPARE* ( $CM_0$ ,  $CM_1$  și  $CM_2$ ).

Circuitul *Timer*  $T_2$  se poate afla într-unul din următoarele trei regimuri de lucru:

- Oprit;
- În regim generator programabil de impulsuri (*Timer*); în acest regim sunt numărate impulsurile generate de oscilatorul intern ( $f_{osc}$ ) divizate cu 12;
- În regim numărător; în acest caz sunt numărate impulsurile aplicate din exterior pe intrarea  $T_2$  ( $P_{1.4}$ ). Numărătorul de prescalare va fi incrementat la fiecare apariție a unui front crescător al semnalului aplicat pe intrarea  $T_2$ . Frecvența maximă a semnalului ce poate fi numărată este de  $1MHz$  la  $f_{osc} = 12MHz$ .

**Observație:** Frecvența maximă de numărare pentru *Timer*-ul  $T_2$  este dublă față de cea a *Timer*-elor  $T_0$  și  $T_1$ . Acest lucru se datorează faptului că, în cazul *Timer*-lui  $T_2$ , semnalul de intrare aplicat pe  $T_2$  se testează de două ori în fiecare ciclu mașină (în momentele  $S_2P_1$  și  $S_5P_1$ ). Un front pozitiv va fi recunoscut, dacă semnalul aplicat pe  $P_2$  ( $P_{1.4}$ ) este pe nivel logic coborât pe durata unei jumătăți de ciclu mașină și trece pe nivel logic ridicat pe încă cel puțin o jumătate de ciclu mașină.

Circuitul *Timer*  $T_2$  nu poate fi încărcat cu o valoare inițială, în schimb poate fi adus la zero. *Reset*-area se produce fie de către semnalul de Resetare intern *RST*, fie cu frontul crescător al unui semnal aplicat din exterior pe terminalul  $RT_2$  ( $P_{1.5}$ ). Acțiunea de Resetare externă trebuie, în prealabil, validată prin setarea prin soft a bitului de validare  $T_2ER$  din registrul de comandă  $TM_2CON_5$ .

Semnalul de *Reset* aduce la zero concomitent și circuitul pe prescalare. Acesta din urmă mai este adus la zero ori de câte ori i se schimbă factorul de divizare sau sursa generatoare de impulsuri de la intrare.

#### 2.4.6.4.2. Semnalele de depășire generate de circuitul *Timer* $T_2$

Circuitul *Timer*  $T_2$  generează două semnale de depășire:

- când octetul inferior al numărătorului  $T_2$  ( $TM_2L$ ) se umple (*least byte overflow*), acest semnal setează bitul indicator al condiției de depășire  $T_2BO$  (*Overflow Flag*) din registrul de comandă  $TM_2CON$  ( $T_2BO = TM_2CON_4$ );
- când numărătorul  $T_2$  se umple (16 bit *Overflow*), caz în care se setează bitul indicator de condiție  $T_2OV$  aflat în registrul  $TM_2IR$  ( $T_2OV = TM_2IR_7$ );

**Observație:** Setarea bitului  $T_2OV$  atrage după sine, în mod automat, și setarea bitului  $T_2BO$ .

Fenomenul de depășire reprezintă o sursă generatoare de întreruperi. Pentru ca întreruperea să se producă efectiv, este nevoie de îndeplinirea unor condiții de selecție, validare și de precizare a priorităților.

În Figura 2.4.6.14. am exemplificat acest mecanism pentru cazul unei depășiri a numărătorului  $T_2$  (16 bit *Overflow*).

Sursa care poate genera o cerere de întrerupere este depășirea capacității de numărare a circuitului de numărare pe 16 biți (*Timer*-ul  $T_2$ ).

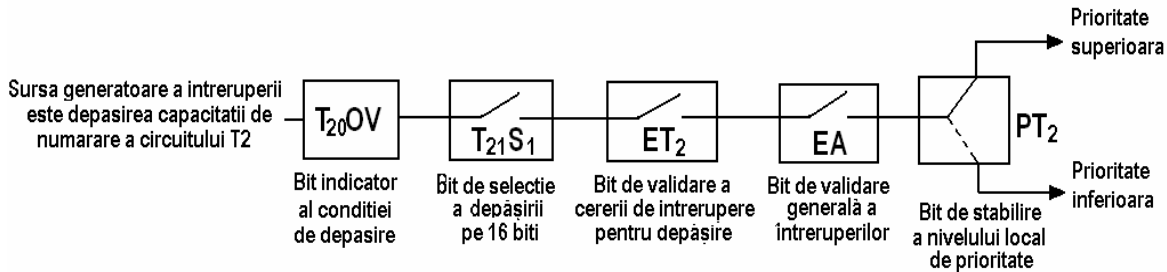
**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

Figura 2.4.6.14. Schema de generare a cererii de întrerupere prin depășire la

Apariția acestei depășiri va determina setarea bitului indicator de condiții  $T_{2OV}$ . Pentru ca cererea să fie luată în considerare, mai trebuie să fie îndeplinite și următoarele condiții simultane:

- acest tip de depășire să fie selectată prin setarea bitului  $T_{2IS_1}$  – bitul de validare a cererilor de întrerupere prin depășire;
- bitul  $ET_2$  să fie setat;
- bitul general de validare a întreruperilor –  $EA$  – să fie la rândul lui setat.

Cererilor de întrerupere generate de depășire li se mai poate atașa și un nivel de prioritate prin alegerea bitului  $PT_2$  din registrul de priorități  $IP_1$  ( $PT_2 = IP_{1,7}$ ).

Dacă alegem  $PT_2=0$  înseamnă o prioritate locală inferioară, pe când alegerea  $PT_2=1$  va însemna o prioritate locală superioară.

Dacă toate condițiile menționate sunt precizate și îndeplinite, atunci, microcontrolerul, printr-un mecanism de selecție va sesiza această cerere de întrerupere și o va achita prin derularea unei rutine de servire specifice. Un mecanism similar este valabil și în cazul unei depășiri pe 8 biți (semnal de selecție  $T_{2ISO}$ ).

**Observație:** Biții indicatori ai condiției de depășire  $T_{2OV}$  respectiv  $T_{2BO}$  trebuiesc *reset*-ați în cadrul rutinei de servire a întreruperilor! (în caz contrar, vor genera o nouă întrerupere).

#### Regimul extins al Timer-lui $T_2$

În cazul utilizării unui oscilator cu frecvența  $f_{osc}=12MHz$ , o depășire pe 16 biți la ieșirea Timer-lui  $T_2$  va apare după 65,5; 131; 262 și respectiv 524ms, în funcție de factorul de divizare programat în numărătorul de prescalare. În aplicații unde este nevoie de perioade mai mari de 0,5s, poate fi folosit regimul extins al Timer-lui  $T_2$ . Acest regim se poate obține dacă:

- folosim ca sursă de oscilație intrarea  $f_{osc}/12$  prin setarea bitului  $T_{2MSO}$  și resetarea bitului  $T_{2MSI}$ ;
- alegerea factorului de divizare a numărătorului de prescalare egal cu 8 prin setarea biților  $T_{2P_0}$  și  $T_{2P_1}$ ;
- invalidăm posibilitatea generării unei cereri de întrerupere la umplerea octetului inferior al lui  $T_2$  prin *reset*-area bitului  $T_{21S_0}$ ;
- validăm posibilitatea generării unei cereri de întrerupere la umplerea lui  $T_2$  (16 bit *Overflow*) prin *reset*-area bitului  $T_{21S_1}$ .

Subrutina următoare este scrisă pentru o expresie de 3 octeți și permite obținerea unor semnale de depășire cu o perioadă de până la 2400 de ore:

**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

OVINT:	PUSH	Acc;	salvează conținutul acumulatorului
	PUSH	PSW;	salvează indicatorii de condiții
	INC	TIMEX 1;	incrementează octetul inferior al <i>Timer</i> -lui extins
	MOV A,	TIMEX 1;	
	JNZ	INTEX;	salt la adresa <i>INTEX</i> dacă nu există <i>overflow</i>
	INC	TIMEX 2;	incrementează octetul următor
	MOV A,	TIMEX 2;	
	JNZ	INTEX;	salt la <i>INTEX</i> dacă nu există <i>overflow</i>
INTEX:	INC	TIMEX 3;	
	CLR	T2OV;	resetează bitul indicator de depășire
	POP	PSW;	încarcă indicatorii de condiții
	POP	Acc;	încarcă acumulatorul
	RETI		

**Registrul de comandă a circuitului Timer T<sub>2</sub>**

Registrul de comandă *TM<sub>2</sub>CON* (vezi Figura 7.15) permite alegerea modului de lucru, a factorului de prescalare precum și a condițiilor de selecție a întreruperilor în cazul unor depășiri.

7	6	5	4	3	2	1	0	
<i>T<sub>2</sub>IS<sub>1</sub></i>	<i>T<sub>2</sub>IS<sub>0</sub></i>	<i>T<sub>2</sub>ER</i>	<i>T<sub>2</sub>B<sub>0</sub></i>	<i>T<sub>2</sub>P<sub>1</sub></i>	<i>T<sub>2</sub>P<sub>0</sub></i>	<i>T<sub>2</sub>MS<sub>1</sub></i>	<i>T<sub>2</sub>MS<sub>0</sub></i>	<i>TM<sub>2</sub>CON</i> (EAH)

Figura 2.4.6.15. Registrul de comandă *TM<sub>2</sub>CON*

Biții registrului de comandă au următoarele semnificații:

- *T<sub>2</sub>IS<sub>1</sub>* – bit de selecție pentru cerere de întrerupere la depășire pe 16 biți;
- *T<sub>2</sub>IS<sub>0</sub>* – bit de selecție pentru cerere de întrerupere la depășire pe octetul inferior;
- *T<sub>2</sub>ER*=1 *Timer*-ul *T<sub>2</sub>* poate fi *resetat* din exterior cu frontul pozitiv al unui semnal aplicat pe terminalul *RT<sub>2</sub>* (*P<sub>1.5</sub>*);
- *T<sub>2</sub>P<sub>1</sub>* și *T<sub>2</sub>P<sub>0</sub>* sunt folosiți pentru alegerea factorului de prescalare:

<i>T<sub>2</sub>P<sub>1</sub></i>	<i>T<sub>2</sub>P<sub>0</sub></i>	Factor de prescalare
0	0	divizor cu 1
0	1	divizor cu 2
1	0	divizor cu 4
1	1	divizor cu 8

- *T<sub>2</sub>MS<sub>1</sub>* și *T<sub>2</sub>MS<sub>0</sub>* permit selecția modului de lucru:

<i>T<sub>2</sub>P<sub>1</sub></i>	<i>T<sub>2</sub>P<sub>0</sub></i>	modul de lucru selectat
0	0	<i>Timer T<sub>2</sub></i> oprit (off)
0	1	<i>Timer T<sub>2</sub></i> lucrează cu $f_{osc}/12$
1	0	regim de testare (nu se va utiliza)
1	1	<i>Timer T<sub>2</sub></i> lucrează cu impulsuri aplicate din afară pe terminalul <i>T<sub>2</sub></i>

**6.4.3. Funcțiile COMPARE și CAPTURE a Timer-lui  $T_2$** 

Timer-ul  $T_2$  este conectat la patru registre *Capture* de 16 biți și la trei registre *Compare* de câte 16 biți.

Registrele *Capture* pot fi folosite pentru a prelua (capta) conținutul Timer-lui  $T_2$  la apariția unei comenzi din exterior (front pozitiv sau negativ aplicat pe un anume terminal de intrare). Un registru de comparare poate fi utilizat pentru a seta, *reseta* sau a bascula terminalele portului  $P_4$  de ieșire, la anumite intervale de timp preprogramabile. Funcțiile *Compare* și *Capture* ale Timer-lui 2 reprezintă un instrument de comandă extrem de puternic și versatil în aplicațiile în care se impune controlul turației unor motoare, comanda aprinderii la motoare cu ardere internă etc.

**Funcția Capture a Timer-ului  $T_2$** 

Cele patru registre *Capture* sunt  $CT_0$ ,  $CT_1$ ,  $CT_2$ , și  $CT_3$ . Aceste registre se vor încărca cu conținutul Timer-lui  $T_2$  dacă pe intrările lor de comandă  $CT_{0i}$ ,  $CT_{1i}$ ,  $CT_{2i}$ , sau  $CT_{3i}$  apare un front pozitiv sau negativ. Concomitent cu aceasta se poate declanșa o cerere de întrerupere.

**Observația 1:** Semnalele externe se aplică pe intrările 0–3 ale portului  $P_1$  ( $P_{1.0} = CT_{0i}$ ;  $P_{1.1} = CT_{1i}$ ;  $P_{1.2} = CT_{2i}$ ;  $P_{1.3} = CT_{3i}$ ).

**Observația 2:** Alegerea frontului de declanșare activ se face prin programarea registrului de comandă *CTCON* din *SFR* (vezi Figura 6.16.).

7	6	5	4	3	2	1	0	
$CTN_3$	$CTP_3$	$CTN_2$	$CTP_2$	$CTN_1$	$CTP_1$	$CTN_0$	$CTP_0$	<i>CTCON</i> (EBH)

Figura 2.4.6.16. Registru de comandă a transferurilor

Comenzile pentru cele patru registre sunt identice motiv pentru care vom ilustra comenzile doar pentru unul din ele (de exemplu, pentru registrul  $CT_3$ ).

Dacă bitul  $CTN_3$  (*CTCON*<sub>7</sub>) este setat, atunci transferul are loc pe frontul negativ al impulsului aplicat pe intrarea  $CT_3I$  ( $P_{1.3}$ ). Dacă bitul  $CTP_3$  (*CTCON*<sub>6</sub>) este setat, atunci transferul are loc pe frontul pozitiv al impulsului aplicat pe intrarea  $CT_3I$  ( $P_{1.3}$ ). Dacă ambii biți ( $CTN_3$  respectiv  $CTP_3$ ) sunt setați, atunci transferul din  $T_2$  în registrul  $CT_3$  are loc atât pe frontul negativ cât și pe frontul pozitiv al impulsului aplicat pe intrarea  $CT_3I$  ( $P_{1.3}$ ).

**Observația 3:** Semnalele exterioare care declanșează transferul sunt folosite și pentru setarea biților de condiție pentru declanșarea unei cereri de întrerupere (*Capture register interrupt flags*). Cei patru biți,  $CTI_3$ ,  $CTI_2$ ,  $CTI_1$  și  $CTI_0$ , corespunzători celor patru registre, se află în registrul de întreruperi a Timer-lui 2 și anume  $TM_2IR$  din *SFR* (vezi Figura 6.17.).

7	6	5	4	3	2	1	0	
T2OV	$CMI_2$	$CMI_1$	$CMI_0$	$CTI_3$	$CTI_2$	$CTI_1$	$CTI_0$	$TM_2IR$ (C8H)

Figura 2.4.6.17. Registrul de întreruperi al Timer-lui  $T_2$ 

**Observația 4:** Folosind funcția *Capture* a Timer-lui 2, putem măsura cu ușurință intervale de timp. Dacă intervalul de timp este marcat de două fronturi consecutive ale unui semnal, atunci prin preluarea conținutului numărătorului  $T_2$  cu primul front, iar apoi cu al doilea, și făcând diferența, putem deduce numărul de



**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

impulsuri care au intrat în respectivul interval. Cunoscând perioada acestor impulsuri ( $f_{osc}$  este dat) putem determina cu ușurință intervalul de timp. Pentru intervale mari de timp se poate recurge la regimul de lucru extins a *Timer*-lui  $T_2$ !

**Funcția de comparare a *Timer*-lui  $T_2$** 

După fiecare incrementare a *Timer*-lui  $T_2$ , conținutul acestuia este comparat cu conținutul registrelor de comparare  $CM_0$ ,  $CM_1$  și  $CM_2$ . În cazul unei egalități, bitul de condiție (*interrupt flag*) corespunzător din registrul  $TM_2IR$  este setat la sfârșitul ciclului mașină următor. Pot apărea următoarele situații:

- egalitate cu conținutul registrului  $CM_0$  – controlerul va seta biții 0–5 ai portului  $P_4$  cu condiția ca biții de validare corespunzători din registrul  $STE$  să fie la rândul lor pe nivel logic unu;
- egalitate cu conținutul registrului  $CM_1$  – controlerul va resetă biții 0–5 ai portului  $P_4$  dacă biții corespunzători de validare a *reset*-ării din registrul  $RTE$  sunt la rândul lor pe nivel logic unu;
- egalitate cu conținutul registrului  $CM_2$  – controlerul va complementa biții 6 și 7 ai portului  $P_4$  dacă biții de validare corespunzători din registrul  $RTE$  sunt la rândul lor pe nivel logic unu.

7	6	5	4	3	2	1	0	
TP47	TP46	RP45	RP44	RP43	RP42	RP41	RP40	RTE (EFH)

7	6	5	4	3	2	1	0	
TG47	TG46	SP45	SP44	SP43	SP42	SP41	SP40	STE (EEH)

Figura 2.4.6.1 Registrele de condiționare *STE* și *RTE*

Semnificațiile biților din regiștrii de condiționare *STE*, respectiv *RTE*, din grupul *SFR*, sunt următoarele:

- Dacă  $TP47 = 1$  atunci  $P_{4,7}$  se complementează la egalitatea  $CM_2$  cu  $T_2$ ;
- Dacă  $TP46 = 1$  atunci  $P_{4,6}$  se complementează la egalitatea  $CM_2$  cu  $T_2$ ;
- Dacă  $TP45 = 1$  atunci  $P_{4,5}$  se resetează la egalitatea  $CM_1$  cu  $T_2$ ;
- Dacă  $TP44 = 1$  atunci  $P_{4,4}$  se resetează la egalitatea  $CM_1$  cu  $T_2$ ;
- Dacă  $TP43 = 1$  atunci  $P_{4,3}$  se resetează la egalitatea  $CM_1$  cu  $T_2$ ;
- Dacă  $TP42 = 1$  atunci  $P_{4,2}$  se resetează la egalitatea  $CM_1$  cu  $T_2$ ;
- Dacă  $TP41 = 1$  atunci  $P_{4,1}$  se *reset*-ează la egalitatea  $CM_1$  cu  $T_2$ ;
- Dacă  $TP40 = 1$  atunci  $P_{4,0}$  se *reset*-ează la egalitatea  $CM_1$  cu  $T_2$ .
- $TG 47$  indică starea bistabilului;
- $TG 46$  indică starea bistabilului;
- Dacă  $SP 45 = 1$  atunci  $P_{4,5}$  se setează la egalitatea  $CM_0$  cu  $T_2$ ;
- Dacă  $SP 44 = 1$  atunci  $P_{4,4}$  se setează la egalitatea  $CM_0$  cu  $T_2$ ;
- Dacă  $SP 43 = 1$  atunci  $P_{4,3}$  se setează la egalitatea  $CM_0$  cu  $T_2$ ;
- Dacă  $SP 42 = 1$  atunci  $P_{4,2}$  se setează la egalitatea  $CM_0$  cu  $T_2$ ;
- Dacă  $SP 41 = 1$  atunci  $P_{4,1}$  se setează la egalitatea  $CM_0$  cu  $T_2$ ;
- Dacă  $SP 40 = 0$  atunci  $P_{4,0}$  se setează la egalitatea  $CM_0$  cu  $T_2$ .



**Structuri hardware si algoritmi specifici microsistemelor EMBEDDED**

**Observația 1:** În cazul egalității conținutului cu  $CM_2$ , nu conținuturile bistabililor proprii pinilor 6 și 7 ai portului vor fi complementați, ci conținuturile altor două bistabile asociate acestor pini pentru a îndeplini funcția de mai sus. Această operație nu va modifica conținuturile bistabililor proprii!

**Observația 2:** Conținuturile acestor bistabili suplimentari vor putea fi citite pe pozițiile  $STE_6$  și  $STE_7$  (corespunzând pinilor  $P_{4.6}$  și  $P_{4.7}$ ). Biții  $STE_6$  și  $STE_7$  sunt de tipul *READ ONLY*.

**Observația 3:** Rezultatul egalității este transferat pinului portului în momentul  $S_5P_1$  din ciclul mașină următor. Dacă modificarea s-a făcut prin soft, atunci ea va apare pe ieșirea pinului în  $S_1P_1$  din ciclul mașină următor.

**Observația 4:** Registrele  $CM_0$ ,  $CM_1$  și  $CM_2$  sunt *reset*-ate de semnalul *RST* generat intern.

**Bibliografie:**

1. Istvan Sztojanov, Sever Pașca, Elisabeta Buzoianu, Aplicații hardware și software cu microcontrolerul PIC12F675, Editura Cavallioti, ISBN 978-973-7622-54-9, Bucuresti 2008
2. Istvan Sztojanov, Alexandru Vasile, Elisabeta Buzoianu, Sever Pașca, *Programarea microcontrolerelor din familia Intel, Aplicații practice hardware cu 80C552*, Editura Man-Dely, ISBN 973-85681-5-3, București 2004.
3. <http://vega.unitbv.ro/~romanca/EmbSys/>
4. <http://facultate.regielive.ro/cursuri/electronica/>

