1. SISTEM DE DEZVOLTARE CU MICROCONTROLLER 80C552

Acest sistem de dezvoltare este realizat pe baza microcontroller-ului 80C552, realizat în tehnologie **CMOS** de firma **PHILIPS**, compatibil ***software*** cu familia de procesoare 8051, fiind astfel conceput încât permite dezvoltarea rapidă a aplicațiilor în domenii diverse - automatizări industriale, aparate de măsură, industrie uşoară, medicină, industria automobilelor, domeniul casnic, etc. Utilizarea tehnologiei **CMOS** îl recomandă pentru aplicațiile care necesită un consum redus de energie şi care necesită imunitate ridicată la perturbații. Prețul scăzut de cost îl recomandă atât pentru produsele de serie, cât şi pentru prototipuri şi unicate.

În figura 1.1 este prezentat sistemul de dezvoltare, iar în figura 1.2 sunt prezentate conectoarele de interconexiune ale sistemului de dezvoltare şi cablul de interconexiune cu un sistem de calcul de tip **PC**.

Sistemul de dezvoltare este destinat în principal dezvoltării de programe. ***Hardware***-ul suplimentar utilizat - de exemplu tastatura, afişajul cu cristale lichide, etc. - permite unificarea din punct de vedere constructiv a diferitelor produse. Acest proces de unificare ***hardware*** direcționează efortul de proiectare spre programe de aplicație.

Acest sistem de dezvoltare bazat pe microcontroller-ul 80C552 acoperă din punct de vedere ***hardware*** şi ***software*** aplicațiile dezvoltate cu microprocesoarele 80C31, 80C32 şi alte procesoare din familia 8051, putând fi folosit la dezvoltarea de aplicații cu aceste procesoare.

1.1 RESURSELE SISTEMULUI DE DEZVOLTARE

Sistemul de dezvoltare realizat dispune de următoarele resurse ***hardware*** şi caracteristici tehnice:

- microcontroller **PCB80C552** (fără memorie internă de program), lucrând la o frecvență maximă a ceasului de 16MHz;
- frecvența ceasului sistemului de dezvoltare 11,059200 MHz;
- memoria de date statică externă (**DATA MEMORY**), implementată cu un circuit **KM62256**, realizat în tehnologie **CMOS**, cu capacitatea de 32 kocteți şi caracterizat de un timp de acces de 35ns. Spațiul de adresare ocupat de memoria de date este cuprins între 8000H şi FFFFH;
- memoria de program externă (**PROGRAM MEMORY**), implementată cu un circuit **EPROM** de tip **27C256**, realizat în tehnologie **CMOS**, cu capacitatea de 32 kocteți şi caracterizat de un timp de acces de 70ns. Spațiul de adrese ocupat de memoria de program externă este cuprins între 0000H şi 7FFFH. Memoria externă de program conține un program monitor, destinat dezvoltării de programe de aplicație;
- interfață serială compatibilă **RS-232** de mare viteză, full duplex;
- bus serial **I^2C** (bus multimaster cu arbitrare de priorități şi viteză mare de transmisie, frecvența maximă a ceasului serial este 100kHz. Destinația principală este comunicația cu circuite integrate sau controller-e, prevăzute cu interfața **I^2C**, aflate în aceeaşi carcasă;
- memorie **EEPROM** serială, implemetată cu un circuit de tip **ST24C04**, realizat în tehnologie **CMOS**, cu capacitatea de 512 octeți şi conectată la interfața **I^2C**;
- **2** porturi paralele de ieşire de **8** biți;
- **1** port paralel de intrare de **8** biți;
- **8** intrări multiplexate la un convertor analog-digital cu rezoluția de 10 biți,

implementat în structura microcontroller-ului 80C552 şi caracterizat de un timp de conversie de 50 cicluri maşină;

- **8** ieşiri decodificate de selecție porturi;

Tabelul 1.1 - Spațiul de adrese pentru selecțiile de porturi.

Liniile de adrese $A_0 \div A_{15}$																Ieşire	Adresa
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	DCD	(HEXA)
0	X	X	X	X	X	X	1	0	0	0	X	X	X	X	X	S_0/	100H..11FH
0	X	X	X	X	X	X	1	0	0	1	X	X	X	X	X	S_1/	120H..13FH
0	X	X	X	X	X	X	1	0	1	0	X	X	X	X	X	S_2/	140H..15FH
0	X	X	X	X	X	X	1	0	1	1	X	X	X	X	X	S_3/	160H..17FH
0	X	X	X	X	X	X	1	1	0	0	X	X	X	X	X	S_4/	180H..19FH
0	X	X	X	X	X	X	1	1	0	1	X	X	X	X	X	S_5/	1A0H..1BFH
0	X	X	X	X	X	X	1	1	1	0	X	X	X	X	X	S_6/	1C0H..1DFH
0	X	X	X	X	X	X	1	1	1	1	X	X	X	X	X	S_7/	1E0H..1FFH
MOV P_2,#1								MOV R_0,#$A_7A_6A_5A_4A_3A_2A_1A_0$B MOV @R_0,A									

Dintre cele 8 semnale de decodificare porturi ocupă un spațiu de adrese cu dimensiunea de FFH, aşa după cum reiese din tabelul prezentat anterior. Dintre cele 8 semnale de selecție sintetizate, în structura sistemului de dezvoltare sunt utilizate doar 4, şi anume:

- S_0 - semnal de selecție pentru afişajul cu cristale lichide LCD;
- S_1 - semnal de selecție pentru portul de ieşire mai puțin semnificativ;
- S_2 - semnal de selecție pentru portul de ieşire mai semnificativ;
- S_3 - semnal de selecție pentru portul de intrare;
- $S_4 \div S_7$ - neutilizate (disponibile pentru extensii ***hardware***).

Extinderea numărului de porturi de intrare-ieşire poate fi făcută fie prin utilizarea semnalelor de selecție disponibile, ceea ce conduce la încărcarea magistralei interne a sistemului de dezvoltare, fie prin subdecodificarea liniilor inferioare de adrese neutilizate $A_4 \div A_0$ şi multiplexarea, respectiv demultiplexarea, intrărilor, respectiv a ieşirilor, portului de intrare, respectiv a portului de ieşire mai puțin semnificativ. Procesul de multiplexare se realizează bazat pe circuite cu ieşiri de tip *three-state,* minimizând deci numărul de resurse ***hardware*** suplimentare necesare.

- **2** ieşiri analogice de **8** biți modulate în durată. Prin integrarea lor se pot obține două convertoare digital-analogice de **8** biți;
- **3** numărătoare de tip timer/counter;
- **1** ***watchdog*** programabil (mijloc de autodeblocare în cazul execuției eronate a programelor, datorită perturbațiilor sau interferențelor;
- **15** linii de întreruperi, dintre care **6** linii externe;
- reset la punerea sub tensiune;
- conectarea directă a unui afişaj cu cristale lichide.

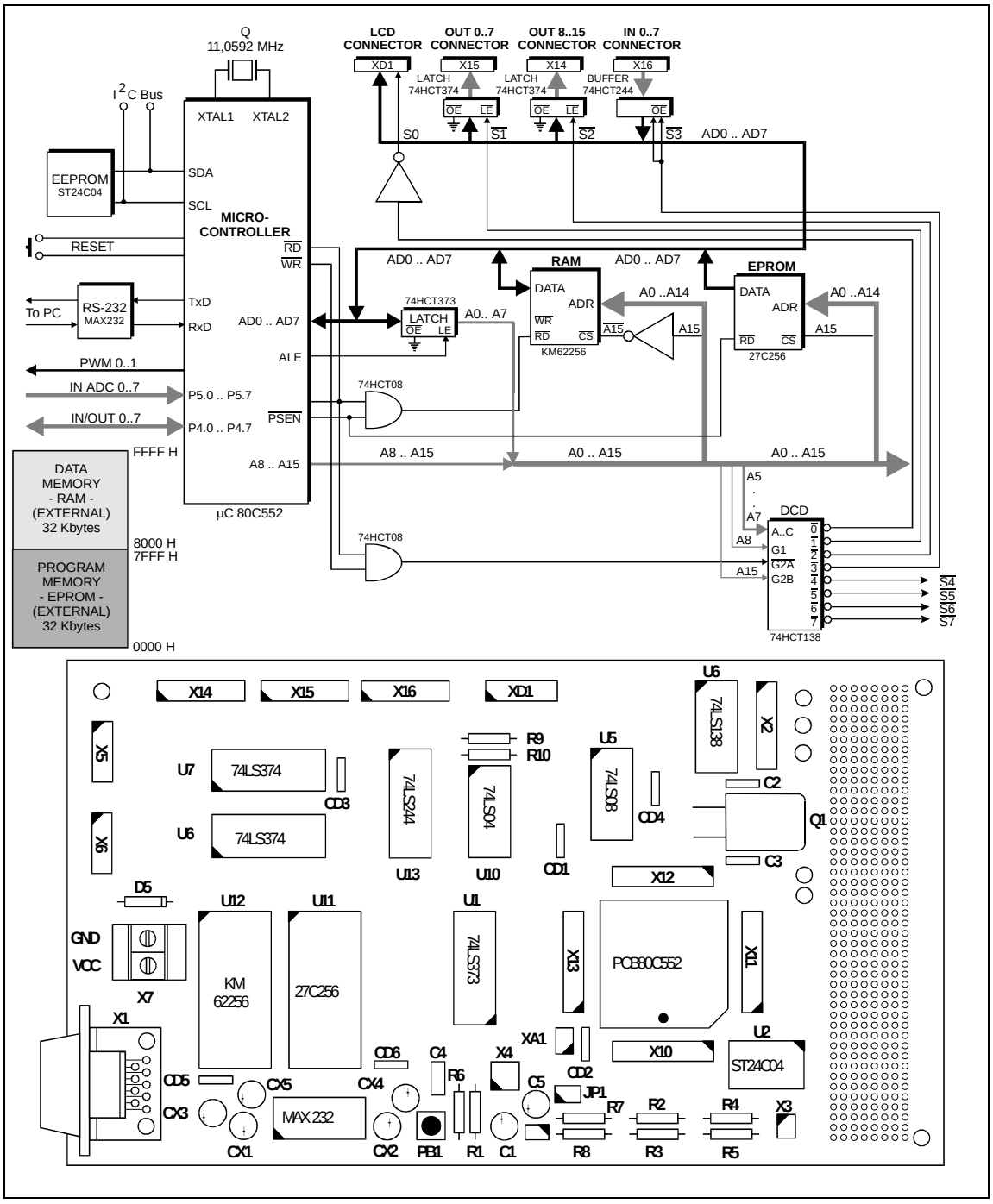

Figura 1.1 - Structura sistemului de dezvoltare cu microcontroller **80C552**.

În continuare, este detaliată configurația pinilor conectoarelor sistemului de dezvoltare cu microcontroller 80C552.

CONECTOR X1 (SERIAL_LINK)

1	NC	2	TXD
3	RXD	4	NC
5	GND	6	NC
7	NC	8	NC
9	NC		

CONECTOR X2 (DECODE)

1	VCC	2	S0/
3	VCC	4	S1/
5	VCC	6	S2/
7	NC	8	S3/
9	NC	10	S4/
11	GND	12	S5/
13	GND	14	S6/
15	GND	16	S7/

CONECTOR X3 (I2C_LINK)

1	SDA	2	SCL

CONECTOR X4 (INTERNAL_MEM)

1	VCC	2	EA/
3	GND	4	EA/

CONECTOR X5 (ROM_SEL)

1	PSEN/	2	CE_P
3	RD/	4	CE_P

CONECTOR X6(ROM_EN)

1	VCC_P	2	VCC
3	GND	4	NC

CONECTOR X7 (MAIN_SUPPLY)

1	GND	2	VCC

CONECTOR X8 (JP1)

1	EW/	2	GND

CONECTOR X9 (SW1_RESET)

1	R6_VCC	2	R1_GND

CONECTOR X10 (μC)

1	CMSR2	2	CMSR4
3	CMSR0	4	CMSR1
5	PWM1	6	EW/
7	STAD	8	PWM1
9	ADC0	10	VCC
11	ADC2	12	ADC1
13	ADC4	14	ADC3
15	ADC6	16	ADC5
17	AVDD	18	ADC7

CONECTOR X11 (μC)

1	CMSR4	2	CMSR3
3	CMT2	4	CMSR5
5	RST	6	CMT1
7	CT1I	8	CT0I
9	CT3I	10	CT2I
11	RT2	12	T2
13	PSDA	14	PSCL
15	PTXD	16	PRXD
17	T0	18	INT0

CONECTOR X12 (μC)

1	T0	2	INT1
3	WR/	4	T1
5	NC	6	RD/
7	XT2	8	NC
9	GND	10	XT1
11	NC	12	GND
13	A9	14	A8
15	A11	16	A10
17	A14	18	A12

CONECTOR X13 (μC)

1	AVSS	2	ADC7
3	AVREF-	4	AVREF+
5	AD1	6	AD0
7	AD3	8	AD2
9	AD5	10	AD4
11	AD7	12	AD6
13	ALE	14	EA/
15	A15	16	PSEN/
17	A13	18	A14

CONECTOR X14 (OUT_HIGH)

1	AX8	2	VCC
3	AX9	4	VCC
5	AX10	6	NC
7	AX11	8	NC
9	AX12	10	NC
11	AX13	12	NC
13	AX14	14	GND
15	AX15	16	GND

CONECTOR X15 (OUT_LOW)

1	AX0	2	VCC
3	AX1	4	VCC
5	AX2	6	NC
7	AX3	8	NC
9	AX4	10	NC
11	AX5	12	NC
13	AX6	14	GND
15	AX7	16	GND

CONECTOR X16 (INPUT)

1	IX0	2	VCC
3	IX1	4	VCC
5	IX2	6	NC
7	IX3	8	NC
9	IX4	10	NC
11	IX5	12	NC
13	IX6	14	GND
15	IX7	16	GND

CONECTOR XA1 (SUPPLY)

1	VCC	2	GND

CONECTOR XD1 (LCD)

1	VSS	2	VDD
3	V0	4	RS
5	R/W	6	EN
7	D0	8	D1
9	D2	10	D3
11	D4	12	D5
13	D6	14	D7

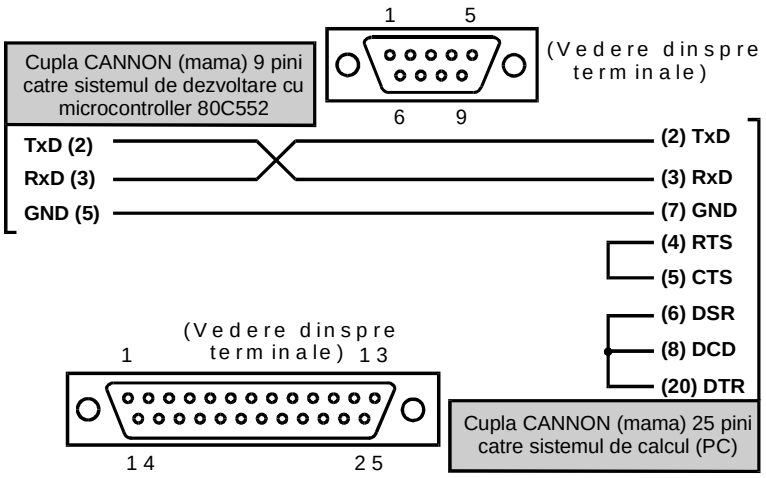

Figura 1.2 - Interconexiunile sistemului de dezvoltare cu exteriorul.

CONECTOR X1 (SERIAL_LINK)

1	NC	2	TXD
3	RXD	4	NC
5	GND	6	NC
7	NC	8	NC
9	NC		

CONECTOR X2 (DECODE)

1	VCC	2	S0/
3	VCC	4	S1/
5	VCC	6	S2/
7	NC	8	S3/
9	NC	10	S4/
11	GND	12	S5/
13	GND	14	S6/
15	GND	16	S7/

CONECTOR X3 (I2C_LINK)

1	SDA	2	SCL

CONECTOR X4 (INTERNAL_MEM)

1	VCC	2	EA/
3	GND	4	EA/

CONECTOR X5 (ROM_SEL)

1	PSEN/	2	CE_P
3	RD/	4	CE_P

CONECTOR X6(ROM_EN)

1	VCC_P	2	VCC
3	GND	4	NC

CONECTOR X7 (MAIN_SUPPLY)

1	GND	2	VCC

CONECTOR X8 (JP1)

1	EW/	2	GND

CONECTOR X9 (SW1_RESET)

1	R6_VCC	2	R1_GND

CONECTOR X10 (μC)

1	CMSR2	2	CMSR4
3	CMSR0	4	CMSR1
5	PWM1	6	EW/
7	STAD	8	PWM1
9	ADC0	10	VCC
11	ADC2	12	ADC1
13	ADC4	14	ADC3
15	ADC6	16	ADC5
17	AVDD	18	ADC7

CONECTOR X11 (μC)

1	CMSR4	2	CMSR3
3	CMT2	4	CMSR5
5	RST	6	CMT1
7	CT1I	8	CT0I
9	CT3I	10	CT2I
11	RT2	12	T2
13	PSDA	14	PSCL
15	PTXD	16	PRXD
17	T0	18	INT0

CONECTOR X12 (μC)

1	T0	2	INT1
3	WR/	4	T1
5	NC	6	RD/
7	XT2	8	NC
9	GND	10	XT1
11	NC	12	GND
13	A9	14	A8
15	A11	16	A10
17	A14	18	A12

CONECTOR X13 (μC)

1	AVSS	2	ADC7
3	AVREF-	4	AVREF+
5	AD1	6	AD0
7	AD3	8	AD2
9	AD5	10	AD4
11	AD7	12	AD6
13	ALE	14	EA/
15	A15	16	PSEN/
17	A13	18	A14

CONECTOR X14 (OUT_HIGH)

1	AX8	2	VCC
3	AX9	4	VCC
5	AX10	6	NC
7	AX11	8	NC
9	AX12	10	NC
11	AX13	12	NC
13	AX14	14	GND
15	AX15	16	GND

CONECTOR X15 (OUT_LOW)

1	AX0	2	VCC
3	AX1	4	VCC
5	AX2	6	NC
7	AX3	8	NC
9	AX4	10	NC
11	AX5	12	NC
13	AX6	14	GND
15	AX7	16	GND

CONECTOR X16 (INPUT)

1	IX0	2	VCC
3	IX1	4	VCC
5	IX2	6	NC
7	IX3	8	NC
9	IX4	10	NC
11	IX5	12	NC
13	IX6	14	GND
15	IX7	16	GND

CONECTOR XA1 (SUPPLY)

1	VCC	2	GND

CONECTOR XD1 (LCD)

1	VSS	2	VDD
3	V0	4	RS
5	R/W	6	EN
7	D0	8	D1
9	D2	10	D3
11	D4	12	D5
13	D6	14	D7

2. PREZENTARE GENERALĂ A MICROCONTROLLER-ULUI 80C552

2.1 MEMORIA INTERNĂ A MICROCONTROLLER-ULUI 80C552

2.1.1 MEMORIA DE PROGRAM (PROGRAM MEMORY)

Familia 8XC552 conține 8 kocteți de memorie de program implementată intern, memorie ce poate fi extinsă la 64 kocteți prin utilizarea unei memorii EPROM externă (figura 2.1). Atunci când pinul EA este forțat la nivel ridicat, microcontroller-ul citește instrucțiunile (ciclu de ***fetch***) din memoria internă de program dacă adresa acestora este inferioară valorii 1FFFH. Locațiile cu adrese cuprinse între 2000H şi FFFFH corespund memoriei externe de program. Atunci când pinul EA este forțat la nivel coborât, toate ciclurile de ***fetch*** (ciclurile de citire a instrucțiunii) sunt executate din memoria externă de program. Locațiile de memorie de program cu adresele 0003H şi 0007H sunt utilizate de rutinele de tratare a întreruperilor.

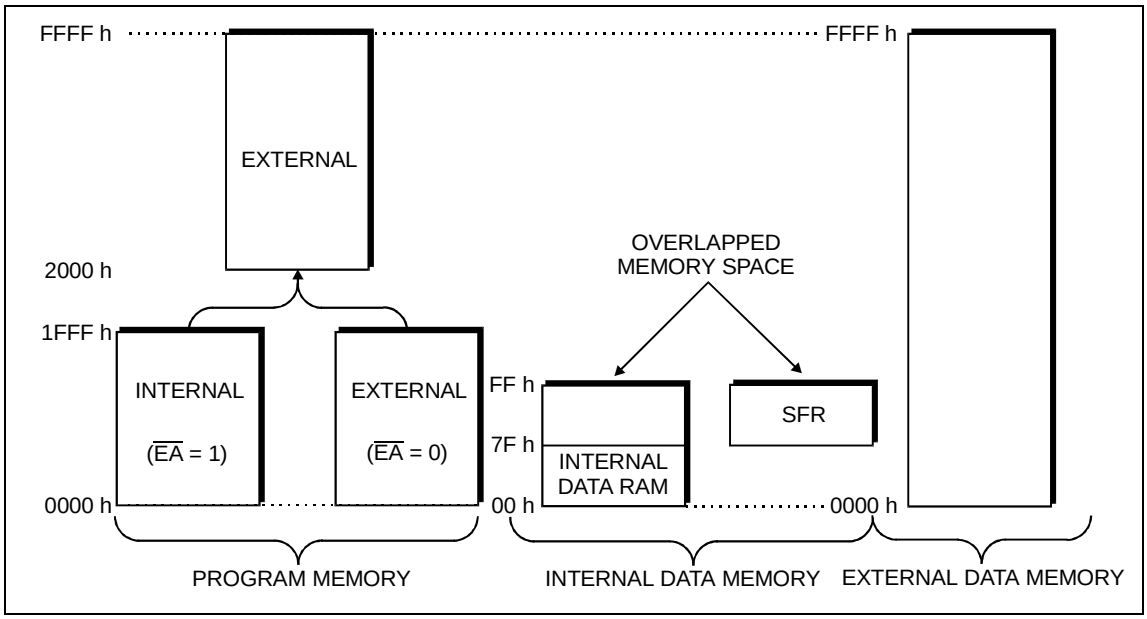

Figura 2.1 - Memoria microcontroller-ului 80C552.

2.1.2 MEMORIA DE DATE (DATA MEMORY)

Microprocesorul conține o memorie internă cu citire/scriere cu adrese cuprinse între 00H şi FFH. Aceasta este organizată pe trei secțiuni:

- zona inferioară a memoriei de date, cu capacitatea de 128 octeți, având adrese cuprinse între 00H şi 7FH. Adresarea acestei zone zone de memorie se poate face direct sau indirect;
- zona superioară a memoriei de date, cu capacitatea de 128 octeți. Această zonă de memorie poate fi adresată numai indirect;
- zona registrelor destinate funcțiilor speciale, cu capacitatea de 128 octeți.

Cei 128 octeți ai zonei inferioare a memoriei de date sunt organizați astfel:

- primii 32 de octeți (cu adrese cuprinse între 00H şi 1FH) sunt împărțiți în 4 bancuri de căte 8 octeți numite şi registre generale şi care pot fi apelate în instrucțiuni ca R_0-R_7. Selectarea unuia dintre bancuri la un moment dat se realizează cu ajutorul a doi biți din cuvântul de stare al progamului (PSW).

Aceste registre pot fi accesate şi la nivel de bit;

- 16 octeţi cu adrese cuprinse între 20H şi 2FH, care pot fi adresaţi şi ca un spaţiu de 128 de biţi cu adrese cuprinse între 00H şi 7FH.

Zona superioară de memorie şi zona registrelor destinate funcţiilor speciale împart acelaşi spaţiu al adreselor de memorie cuprinse între 80H şi FFH, deşi ele sunt entităţi fizice distincte. Nu toţi cei 128 octeţi ai zonei registrelor cu funcţii speciale sunt implementaţi fizic. Registrele destinate funcţiilor speciale ale căror adrese se termină în 0H sau 8H pot fi adresate şi la nivel de bit.

Pe lângă cei 256 de octeţi de date ai memoiei interne, microprocesorul poate adresa şi până la 64 Kocteţi de memorie externă de date. Adresele memoriei de date externă pornesc tot de la 0000H. Adresa pentru memoria externă poate fi pe un octet sau pe doi octeţi. Liniile portului de intrare-ieşire P_0 se folosesc multiplexat pentru a obţine octetul inferior al adresei de memorie, respectiv octetul de date citit/scris în memorie. În cazul în care adresa de memorie este de un octet (octet conţinut într-unul din registrele generale ale bancului de registre activ), acesta se poate folosi în conjuncţie cu un număr de linii ale portului de intrare-ieşire P_2 care paginează memoria. Dacă adresa este pe doi octeţi, aceasta este conţinută în registrul destinat funcţiilor speciale DPTR, iar adresarea memoriei se relizează folosind cele 16 linii ale porturilor de intrare-ieşire P_0 şi P_2.

2.1.3 REGISTRELE CU FUNCŢII SPECIALE

Cele mai multe dintre cele 56 de registre speciale se folosesc pentru controlul ***hardware***-ului de periferice aflat pe cip. Altele sunt registre aritmetice (ACC, B, PSW), indicator al stivei (SP), indicatoare de date (DPH, DPL). 16 dintre aceste registre pot fi adresate la nivel de bit. În continuare vor fi prezentate registrele cu funcţii speciale cele mai des folosite.

Acumulatorul - este registrul implicit pentru multe instrucţiuni. În cadrul unei instrucţiuni este apelet cu numele **A**. Adresa sa directă este E0H. Poate fi adresat şi la nivel de bit. Conţinutul său devine 00H la resetarea microprocesorului.

Registrul B - este utilizat obligatoriu în instrucţiunile de înmulţire şi împărţire. Poate fi folosit şi în alte instrucţiuni. Adresa directă este F0H. Poate fi adresat şi la nivel de bit. Conţinutul său devine 00H la resetarea microprocesorului.

Cuvântul de stare al programului (PSW) - conţine informaţia de stare. Denumirile şi semnificaţiile celor 8 biţi ai PSW sunt date mai jos:

(MSB)							(LSB)
CY	AC	F0	RS_1	RS_0	OV	-	P

- **CY** (**PSW.7**) - indicatorul de transport din bitul cel mai semnificativ al acumulatorului, în cazul instrucţiunilor aritmetice;
- **AC** (**PSW.6**) - indicatorul de transport auxiliar. Se foloseşte pentru operaţii în BCD;
- **F_0** (**PSW.5**) - indicatorul 0 folosit de utilizator;
- **RS_1** (**PSW.4**) şi **RS_2** (**PSW.3**) - folosiţi pentru selecţia bancului activ al registrelor generale. Sunt setaţi sau resetaţi prin program pentru a selecta bancul dorit;
- **OV** (**PSW.2**) - indicatorul de depăşire;
- **P** (**PSW.0**) - indicatorul de paritate. Este setat sau resetat prin ***hardware*** la fiecare ciclu de instrucţiune pentru a indica numărul de 1 (par sau impar) din acumulator.

Adresa directă a PSW este D0H.

Indicatorul de stivă (SP) - este un registru de 8 biţi. Este incrementat înainte

de execuția unei instrucțiuni PUSH sau CALL. Este decrementat după execuția unei instrucțiuni POP sau RET. La resetarea microprocesorului este inițializat la valoarea 07H, deci stiva începe de la adresa 08H. Poate fi încărcat prin program, deci stiva poate începe de la orice adresă. Adresa directă este 81H.

Indicatorul pentru date (DPTR) - este format din doi octeți, cel de adresă mai mare fiind apelat ca DPH, iar cel de adresă mai mică fiind apelat ca DPL. El păstrează o adresă de 16 biți pentru apelarea memoriei RAM externe (dacă aceasta există). Poate fi manevrat ca un singur registru de 16 biți sau ca două registre de 8 biți. Adresa directă este 82H.

Registrele porturi $\mathbf{P_0 \div P_5}$ - sunt registrele ***latch*** pentru porturile de intrare-ieşire $P_0 \div P5$. Când se scrie o informație într-un bit al unui registru, aceasta apare la pinul de intrare-ieşire corespunzător. Când se citeşte informația de la un port de intrare-ieşire, aceasta este memorată în registrul port corespunzător. La resetarea microprocesorului registele P_0 la P_4 sunt inițializate cu valoarea 00H. Adresele directe ale acestor registre sunt: P_0 - 80H, P_1 - 90H, P_2 - A0H, P_3 - B0H, P_4 - C0H, P_5 - C4H. Registrele $P_0 \div P_3$ sunt adresabile şi la nivel de bit.

Buffer serial de date ($\boldsymbol{S_0}$***BUF)*** - este compus de fapt din două registre separate de 8 biți având aceeaşi adresă directă, unul pentru transmisie şi altul pentru recepție, folosite pentru interfața serială SIO_0. Când un octet de date este scris în S_0BUF, acesta este introdus în ***buffer***-ul pentru transmisie şi delanşează începutul transmisiei seriale a acestuia. Când se realizează o citire din S_0BUF, se citeşte conținutul ***buffer***-ului de recepție. Adresa directă este 99H.

Registrele de control - există mai multe astfel de registre care conțin biți de control şi de stare pentru întreruperi, temporizatoare/numărătoare şi porturi seriale. Ele vor fi descrise la funcțiunile corespunzătoare.

Registrele de temporizare - există trei perechi de registre de câte 8 biți T_0 (TH_0, TL_0), T_1 (TH_1, TL_1), T_2 (TMH_2, TML_2) numărătoare pe 16 biți, şi un registru numărător pe 8 biți T_3. T_3 este folosit ca ***watchdog*** pentru reinițializarea sistemului după un anumit interval de timp convenabil ales, în cazul când acesta s-a blocat şi nu a putut fi reîncărcat temporizatorul T_3.

Tabelul 2.1 - Registrele microcontroller-ului 80C552.

Simbol	Descriere	Adresa directa	Adresa bit, simbol sau functie alternativa port								Valoare RESET
ACC	Accumulator	E0 H	E7	E6	E5	E4	E3	E2	E1	E0	00 H
ADCH	A/D converter High	C6 H	A/D.9	A/D.8	A/D.7	A/D.6	A/D.5	A/D.4	A/D.3	A/D.2	xxxxxxxx B
ADCON	A/D control	C5 H	A/D.1	A/D.0	ADEX	ADCI	ADCS	ADR2	ADR1	ADR0	xx000000 B
B	B register	F0 H	F7	F6	F5	F4	F3	F2	F1	F0	00 H
CTCON	Capture control	EB H	CTN3	CTP3	CTN2	CTP2	CNT1	CNP1	CNT0	CNP0	00 H
CTH3	Capture 3 High	CF H									xxxxxxxx B
CTH2	Capture 2 High	CE H									xxxxxxxx B
CTH1	Capture 1 High	CD H									xxxxxxxx B
CTH0	Capture 0 High	CC H									xxxxxxxx B
CMH2	Compare 2 High	CB H									00 H
CMH1	Compare 1 High	CA H									00 H
CMH0	Compare 0 High	C9 H									00 H
CTL3	Capture 3 Low	AF H									xxxxxxxx B
CTL2	Capture 2 Low	AE H									xxxxxxxx B
CTL1	Capture 1 Low	AD H									xxxxxxxx B
CTL0	Capture 0 Low	AC H									xxxxxxxx B
CML2	Compare 2 Low	AB H									00 H
CML1	Compare 1 Low	AA H									00 H
CML0	Compare 0 Low	A9 H									00 H
DPTR: DPH	Data pointer Data pointer High	83 H									00 H

DPL	Data pointer Low	82 H									00 H
IEN0	Interrupt enable 0	A8 H	AF EA	AE EAD	AD ES1	AC ES0	AB ET1	AA EX1	A9 ET0	A8 ET1	00 H
IEN1	Interrupt enable 1	E8 H	EF ET2	EE EMC2	ED EMC1	EC EMC0	EB ECT3	EA ECT2	E9 ECT1	E8 ECT0	00 H
IP0	Interrupt priority 0	B8 H	BF --	BE PAD	BD PS1	BC PS0	BB PT1	BA PX1	B9 PT0	B8 PX0	x0000000 B
IP1	Interrupt priority 1	F8 H	FF PT2	FE PCM2	FD PCM1	FC PCM0	FB PCT3	FA PCT2	F9 PCT1	F8 PCT0	00 H
P5	Port 5	C4 H	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	xxxxxxxx B
P4	Port 4	C0 H	C7 CMT1	C6 CMT0	C5 MSR5	C4 MSR4	C3 MSR3	C2 MSR2	C1 MSR1	C0 MSR0	FF H
P3	Port 3	B0 H	B7 RD	B6 WR	B5 T1	B4 T0	B3 INT1	B2 INT0	B1 TXD	B0 RXD	FF H
P2	Port 2	A0 H	A7 A15	A6 A14	A5 A13	A4 A12	A3 A11	A2 A10	A1 A9	A0 A8	FF H
P1	Port 1	90 H	97 SDA	96 SCL	95 RT2	94 T2	93 CT3I	92 CT2I	91 CT1I	90 CT0I	FF H
P0	Port 0	80 H	87 AD7	86 AD6	85 AD5	84 AD4	83 AD3	82 AD2	81 AD1	80 AD0	FF H
PCON	Power control	87 H	SMOD	--	--	WLE	GF1	GF0	PD	IDL	00xx0000 B
PSW	Program status word	D0 H	D7 CY	D6 AC	D5 F0	D4 RS1	D3 RS0	D2 OV	D1 F1	D0 P	00 H
PWMP	PWM prescaler	FE H									00 H
PWM1	PWM 1 register	FD H									00 H
PWM0	PWM 0 register	FC H									00 H
RTE	Reset/toggle enable	EF H	TP47	TP46	RP45	RP44	RP43	RP42	RP41	RP40	00 H
SP	Stack pointer	81 H									07 H
S0BUF	Serial 0 data buffer	99 H									
S0CON	Serial 0 control	98 H	9F SM0	9E SM1	9D SM2	9C REN	9B TB8	9A RB8	99 TI	98 RI	00 H
S1ADR	Serial 1 address	DB H	SLAVE ADDRESS							GC	00 H
SIDAT	Serial 1 data	DA H									00 H
S1STA	Serial 1 status	D9 H	SC4	SC3	SC3	SC1	SC0	0	0	0	F8 H
SICON	Serial 1 control	D8 H	DF CR2	DE ENS1	DD STA	DC STO	DB SI	DA AA	D9 CR1	D8 CR0	00 H
STE	Set enable	EE H	TG47	TG46	SP45	SP44	SP43	SP42	SP41	SP40	C0 H
TH1	Timer 1 High	8D H									00 H
TH0	Timer 0 High	8C H									00 H
TL1	Timer 1 Low	8B H									00 H
TL0	Timer 0 Low	8A H									00 H
TMH2	Timer 2 High	ED H									00 H
TML2	Timer 2 Low	EC H									00 H
TMOD	Timer mode	89 H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	00 H
TCON	Timer control	88 H	8F TF1	8E TR1	8D TF0	8C TR0	8B IE1	8A IT1	89 IE0	88 IT0	00 H
TM2CON	Timer 2 control	EA H	T2IS1	T2IS0	T2ER	T2B0	T2P1	T2P0	T2MS 1	T2MS 0	00 H
TM2IR	T2 interrupt flag register	C8 H	CF T20V	CE CMI2	CD CMI0	CC CMI0	CB CTI3	CA CTI2	C9 CTI1	C8 CTI0	00 H
T3	Timer 3	FF H									00 H

2.2 STRUCTURA ŞI LUCRUL CU PORTURILE DE INTRARE-IEŞIRE

În figura 2.2 este prezentată structura microcontroller-ului 80C552. Semnificațiile detaliate ale porturilor sunt prezentate în cele ce urmează.

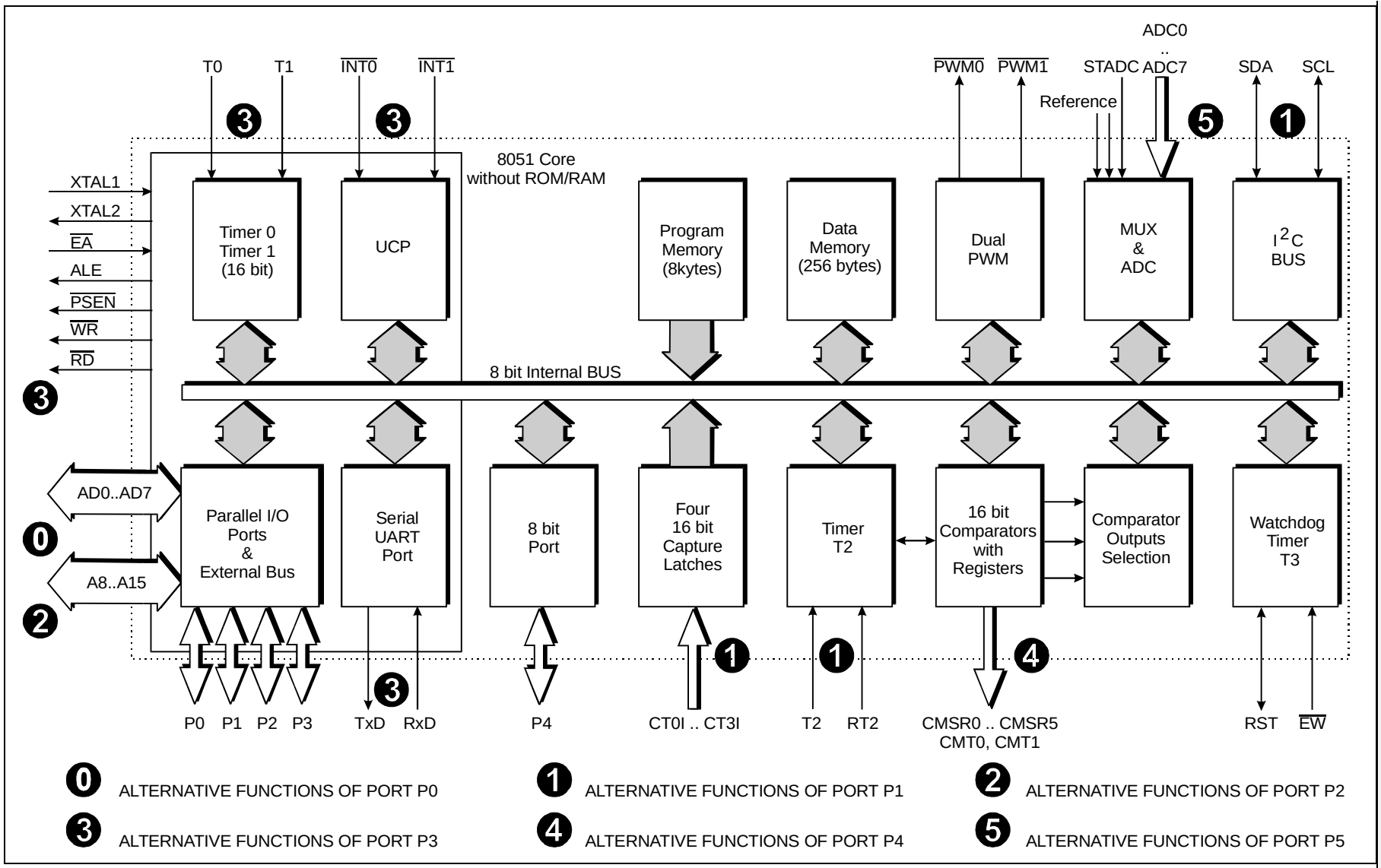

Figura 2.2 - Structura microcontroller-ului 80C552.

***Portul* $\mathbf{P_0}$** - este un port bidirecțional cu drena în gol de 8 biți. Dacă se scrie 1 în el tranzistorul de ieşire este blocat şi ieşirea este în starea de impedanță ridicată. În cazul folosirii unei memorii externe (de program - ROM sau de date - RAM), pinii portului P_0 sunt multiplexați între octetul inferior de adresă ($A_0 \div A_7$) şi octetul de date citit sau scris din sau în memorie. Poate fi accesat şi la nivel de pin ca $P_{0.0} \div P_{0.7}$.

***Portul* $\mathbf{P_1}$**- este un port bidirecțional de 8 biți. Pinii $P_{1.0} \div P_{1.5}$ sunt prevăzuți cu rezistență internă de pull-up, iar biții $P_{1.6}$ şi $P_{1.7}$ sunt cu drena în gol. Pinii acestui port pot îndeplini şi următoarele funcții alternative:

- $\mathbf{P_{1.0} \div P_{1.3}}$ pot fi şi CT0I÷CT3I, adică semnale de intrare pentru temporizatorul T_2 în modul captură (vezi funcționarea timer-ului T_2);
- $\mathbf{P_{1.4}}$ poate fi şi intrare externă de numărare a temporizatorului T_2, numită chiar T_2;
- $\mathbf{P_{1.5}}$ poate fi şi intrare de reset a temporizatorului T_2, adică RT_2;
- $\mathbf{P_{1.6}}$ poate fi SCL, adică semnal de ceas pentru interfața serială SIO_1;
- $\mathbf{P_{1.7}}$ poate fi SDA, adică linia de date a interfeței seriale SIO_1.

***Portul* $\mathbf{P_2}$** - este un port de 8 biți, bidirecțional, cu rezistențe interne de pull-up. În cazul adresării unei memorii externe, conține octetul superior al adresei $A_8 \div A_{15}$.

***Portul* $\mathbf{P_3}$** - este un port bidirecțional de 8 biți, cu rezistențe interne de pull-up. Alternativ pinii săi pot îndeplini următoarele funcții:

- $\mathbf{P_{3.0}}$ poate fi RxD, adică intrare de date pentru interfața serială, full duplex, SIO_0;
- $\mathbf{P_{3.1}}$ poate fi TxD, adică ieşire de date pentru SIO_0;
- $\mathbf{P_{3.2}}$ poate fi INT0\, adică prima întrerupere externă;
- $\mathbf{P_{3.3}}$ poate fi INT_1\, adică a doua întrerupere externă;
- $\mathbf{P_{3.4}}$ poate fi T_0, adică intrarea de numărare externă pentru temporizatorul T_0;
- $\mathbf{P_{3.5}}$ poate fi T_1, adică intrarea de numărare externă pentru temporizatorul T_1;
- $\mathbf{P_{3.6}}$ poate fi WR\, adică semnalul de comandă a scrierii în memoria de date (RAM) externă;
- $\mathbf{P_{3.7}}$ poate fi RD\, adică semnalul de comandă a citirii din memoria externă de date (RAM).

***Portul* $\mathbf{P_4}$** - este un port bidirecțional de 8 biți, cu rezistențe interne de pull-up.

***Portul* $\mathbf{P_5}$** - este un port de 8 biți numai de intrare. Pinii acestui port pot fi şi $ADC_0 \div ADC_7$, adică 8 canale analogice de intrare ale convertorului analogic-numeric incorporat.

Operațiunea de citire-modificare-scrire la un port. Unele instrucțiuni care citesc un port, citesc regisrul pentru funcții speciale corespunzător, iar altele citesc direct starea la momentul respectiv a pinului. Insrucțiunile care citesc registrul sunt acelea care citesc o valoare pe care eventual o modifică şi apoi o scriu din nou în registru. Acestea se numesc instrucțiuni "citeşte-modifică-scrie". Următoarele instrucțiuni acționează asupra registrului pentru funcții speciale corespunzător, atunci când în ele apare numele unui port de intrare-ieşire: ANL, ORL, XRL, JBC, CPL, INC, DEC, DJNZ, MOV PX.Y,C (scrie În bitul Y al portului X transportul (***CARRY***), CLR PX.Y (scrie 0 în bitul Y al portului X), SET PX.Y (scrie 1 în bitul Y al portului X). Nu este foarte evident că ultimele 3 instrucțiuni de mai sus sunt de tipul "citeşte-modifică-scrie", dar ele sunt deoarece execuția lor constă în citirea tuturor celor 8 biți ai registrului, modificarea corespunzătoare a bitului dorit şi rescrierea rezultatului în registru. Acest lucru se realizează astfel, pentru că, dacă s-ar citi valoarea semnalului la pin (care de fapt reprezintă valoarea bitului corespunzător al registrului pentru funcții speciale asociat portului), din cauza circuitelor comandate nivelul de tensiune ar putea să nu fie cel corespunzător (de exemplu dacă pinul comandă în bază un tranzitor care are emitorul la masă, iar bitul corespunzător din registru ar fi "1", tensiunea pe pin ar fi egală cu tensiunea unei joncțiuni *pn* polarizată direct, adică 0,6-0,7 volți, care evident nu este un nivel corespunzător lui "1".

2.2.1 PROGRAMAREA ŞI UTILIZAREA TEMPORIZATOARELOR

Aşa cum s-a spus în paragraful relativ la registrele pentru funcţii speciale, microprocesorul 80552 are 4 registre temporizatoare / numărătoare. Ele sunt descrise în continuare.

***Temporizatorul* 0** şi ***temporizatorul* 1** Ambele pot fi configurate să funcţioneze ca temporizatoare sau ca numărătoare de evenimente. Configurarea este realizată cu ajutorul registrului de control a modului TMOD (el are adresa directă 89H), aşa cum este indicat mai jos:

(MSB)							(LSB)
gate	C/T	M_1	M_0	gate	C/T	M_1	M_0
Temporizator 0				Temporizator 1			

unde:

- **gate** - temporizatorul/numărătorul "x" este activat numai dacă bitul TR_x din registrul de control TCON are valoarea "1" şi acest bit are valoarea "0" sau semnalul de pe pinul INT_x/ are valoarea "1" (vezi portul de intrare-ieşire P_3);
- **C/T** - când acest bit are valoarea "0" registrul funcţionează ca temporizator, iar semnalul de numărare este dat de oscilatorul microprocesorului divizat cu 12, care astfel incrementează registrul corespunzător la fiecare ciclu maşină. Dacă acest bit are valoarea "1" registrul funcţionează ca numărător, fiind incrementat la fiecare tranziţie din "0" în "1" a semnalului de pe pinul T_0 sau T_1 (vezi portul de intrare ieşire P_3). Semnalul de pe acest pin este testat o dată pentru fiecare ciclu maşină în starea S_5 faza P_2 a ciclului, dar numărătorul este efectiv incrementat abia în starea S_3 faza P_1 a ciclului următor celului care a detectat tranziţia 1-0 a semnalului T_1. De aceea frecvenţa maximă de numărare este 1/24 din frecvenţa oscilatorului microprocesorului. Pentru a putea fi detectată tranziţia, semnalul trebuie să stea în "0" o durată cel puţin egală cu a ciclului maşină (adică 12 perioade ale oscilatorului);
- **M_1, M_0** - selectează unul unul din cele patru moduri posibile de operare descrise în continuare.

  Modul 0. Este atunci când M_1=0 şi M_0=0. El este identic pentru ambele temporizatoare. În acest mod cele două numărătoare pe 8 biţi TH_x şi TL_x sunt conectate în cascadă, dar TL_x funcţionează ca un numărător pe cinci biţi (deci un divizor cu 32), folosindu-se numai primii 5 biţi ai acestuia. Când numărătorul trece din starea în care toţi biţii sunt pe "1" în starea când toţi biţii sunt pe zero, se setează indicatorul de întrerupere TF_x din registrul de control TCON (TCON.7 pentru TF_1 şi TCON.5 pentru TF_0);

  Modul 1. Este atunci când M_1=0 şi M_0=1. Este identic cu modul 0, doar că TL_x se foloseşte ca numărător pe 8 biţi (se obţine deci un numărător pe 16 biţi). El este identic pentru temporizatoarele 0 şi 1;

  ***Modul 2*.** Este atunci când M_1=1 şi M_0=0. Este identic pentru ambele temporizatoare. În acest mod se foloseşte numai registrul TL_x ca un numărător pe 8 biţi cu reîncărcare automată. Conţinutul acestuia trece din FFH în 00H, se setează TF_x şi se încarcă în TL_x conţinutul lui TH_x, fără a fi afectat conţinutul lui TH_x. TH_x poate fi încarcat prin program;

  Modul 3. Este atunci când M_1=1 şi M_0=1. În acest mod temporizatorul 1 îşi opreşte numărarea, conservându-şi conţinutul. Este ca şi când TR_1 ar fi egal cu "0". Temporizatorul 0 fucţionează ca două numărătoare de 8 biţi independente. TH_0 este incrementat de semnalul oscilatorului microprocesorului divizat cu 12, dacă TR_1=1. La trecerea conţinutului lui de la valoare FFH la 00H este setat TF_1. TL_0 poate fi incrementat fie de semnalul de la oscilator

dacă C/T=0, fie de semnalul aplicat la pinul T_1 dacă C/T=1. Incrementarea are loc dacă TR_1=1 şi gate=0 sau INT_0/=1. La schimbarea conţinutului de la valoarea FFH la 00H este setat TF_0.

Funcţionarea celor două temporizatoare este controlată de registrul TCON. Configuraţia acestuia este dată în continuare:

(MSB)							(LSB)
TF_1	TR_1	TF_0	TR_0	IE_1	IT_1	IE_0	IT_0

- **TF_1** şi **TF_0** sunt setate aşa cum s-a arătat mai sus, şi sunt resetate prin ***hardware*** atunci procesorul intră în rutina de întrerupere;
- **TR_1** şi **TR_0** inhibă (când sunt egale cu 0) sau permit numărarea (când sunt egale cu 1) a temporizatorului corespunzător. Sunt setate sau resetate prin program.

2.2.2 INTERFAŢA SERIALĂ SIO_0

Aceasta este o interfaţă full-duplex, adică poate transmite şi recepţiona date simultan. Are ***buffer*** la recepţie, deci poate începe recepionarea unui nou octet de date înaite ca cel deja recepţionat să fie preluat. Aşa cum s-a menţionat în paragraful referitor la registrele cu funcţii speciale (vezi registrul SBUF), o scriere în SBUF încarcă registrul pentru transmisia datelor, iar citirea din SBUF se face din registrul de recepţie a datelor, separat fizic de cel transmisie. Interfaţa serială SIO_0 poate lucra în 4 moduri posibile. Acestea sunt următoarele:

Modul 0. Este atunci când SM_0=0 şi SM_1=0 (biţii 7 respectiv 6 din registrul de control al interfeţei SCON - unul din registrele pentru funcţii speciale). În acest mod interfaţa serială funcţionează semiduplex. Se transmit câte 8 biţi. Pe linia RxD ($P_{3.0}$) se emit şi se recepţionează date. Semnalul de ceas al transmisiei/recepţiei se transmite/recepţionează pe linia TxD ($P_{3.1}$). Transmisia este iniţiată prin scrierea unui octet de date în registrul SBUF. Ceasul pentru transmisie are o frecvenţă egală cu 1/12 din frecvenţa oscilatorului. La sfârşitul transmisiei celor 8 biţi de date se setează bitul de întrerupere TI (bitul 1 din SCON), care apoi trebuie şters prin program. Recepţia este validată atunci când bitul REN (bitul 4 din SCON) este 1 (el este setat şi resetat prin program) şi bitul de întrerupere pentru recepţie RI (bitul 0 din SCON) este 0 (el este trecut în 1 la sfârşitul recepţionării celor 8 biţi de date şi trebuie trecut în zero prin program). Datele sunt emise începând cu bitul cel mai puţin semnificativ.

Modul 1. Este atunci când SM_0=0 şi SM_1=1. În acest mod transmisia este asincronă pe 10 biţi: un bit de start având valoarea 0, 8 biţi de date (primul este cel mai puţin semnificativ) şi un bit de stop având valoarea 1. Datele sunt emise pe linia TxD şi recepţionate pe linia RxD. Recepţia este iniţiată la detectarea unei tranziţii din 1 în 0 a liniei RxD, dacă bitul REN=1. Octetul de date este încărcat în SBUF, iar bitul de stop este introdus în RB8 (adică bitul 2 din SCON) şi poziţionează în 1 bitul de întrerupere pentru recepţie RI (discutat la modul zero) dacă acesta era 0, şi dacă SM_2=0 (bitul 5 din SCON, setat sau resetat prin program) sau bitul de stop recepţionat este 1. Rata transmisiei este dată de semnalul de depăşire (TF1) al temporizatorului 1. Transmisia este iniţiată prin scrierea unui octet de date în SBUF. În momentul transmisiei bitului de stop se poziţionează în 1 bitul de întrerupere la transmisie SI. Recepţia este iniţiată în urma detecţiei unei tranziţii 1-0 pe linia RxD.

Modul 2. Este atunci când SM_0=1 şi SM_1=0. În acest mod tramsmisia este full duplex, asincronă, cu 11 biţi transmişi (pe linia TxD) şi recepţionaţi pe linia RxD), astfel: un bit de start egal cu 0, 8 biţi de date (primul este cel mai puţin semnificativ), un al noulea bit de date programabil şi un bit de stop egal cu 1. Cel de al noulea bit de date în cazul transmisiei reprezintă conţinutul lui TB8 (bitul 3 din SCON), iar în cazul recepţiei este introdus în RB_8. Transmisia este iniţiată prin scrierea unui octet de date în SBUF. În timpul transmisiei bitului de stop este setat TI. Recepţia este identică cu

cea din modul 1, numai că în RB_8 se introduce al nouălea bit de date. Încărcarea datelor recepționate în SBUF şi RB_8 şi setarea lui RI este validată numai dacă RI=0 şi, SM_2=0 sau al nouălea bit de date este 1. Altfel octetul de date recepționat se pierde. Rata de transmisie poate 1/32 din frecvența oscilatorului dacă SMOD=1 (bitul 7 din registrul pentru funcții speciale PCON) sau 1/64 din frecvența acestuia dacă SMOD=0.

Modul 3. Este atunci când SM_0=1 şi SM_0=1. El este identic cu modul 2, cu singura deosebire că rata transmisiei de semnalul de depăşire TF_1 al timerului T_1.

OBSERVAȚIE: În cazul modurilor 1 şi 3 când este folosit temporizatorul 1 pentru fixarea ratei de transmisie, întreruperea acestuia trebuie dezactivată. El poate fi folosit ca "temporizator" sau "numărător" în oricare din cele 3 moduri posibile de funcționare. De obicei este folosit ca "temporizator" în modul 2 cu autoîncărcare. În acest caz rata de transmisie se calculează cu formula:

$$\frac{2^{SMOD}}{32} \times \frac{f_{OSC}}{12 \times [256 - (TH1)]} \tag{2.1}$$

2.2.3 PRINCIPIUL DE FUNCȚIONARE AL INTERFEȚEI I^2C

Pentru a exploata similaritățile care există în proiectele şi echipamentele proiectate de diverşi utilizatori, ca şi pentru maximizarea eficienței ***hardware***-ului şi pentru simplificarea proiectării circuitelor, a fost dezvoltată o magistrală bidirecțională pe două fire, cu scopul eficientizării controlului interconectării circuitelor integrate. Acestă magistrală se numeşte **INTER IC** sau **I^2C**. În prezent, această magistrală permite cuplarea a mai mult de 150 de tipuri de circuite integrate, realizate în tehnologie CMOS sau bipolară, realizând funcții în domeniul controlului inteligent, a circuitelor integrate de uz dedicat (***driver***-e pentru afişaje cu cristale lichide, porturi de intrare-ieşire, memorii RAM şi EEPROM, convertoare) şi a circuitelor orientate pe aplicații (procesare de semnale pentru sisteme radio şi video, generatoare DTFM pentru telefoane, etc.). Toate circuitele compatibile **I^2C** încorporează o interfață care permite intercomunicația rapidă prin intermediul acestui tip de magistrală.

Dintre caracteristicile generale ale magistralei **I^2C** putem menționa:

- magistrala conține doar două linii: o linie serială de date (**SDA**) şi o linie de clock serial (**SCL**);
- fiecare dispozitiv conectat la magistrală este sdresabil prin ***software***, având o adresă unică; pe magistrala **I^2C** se manifestă, la orice moment de timp, o relație de tip master-slave;
- magistrala **I^2C** este o magistrală multi-master, incluzând detecția conflictelor şi arbitrarea acesteia, pentru a preveni alterarea informației dacă două sau mai multe dispozitive master inițiază transferuri simultane;
- transferurile bidirecționale de date, cu lungimi de 8 biți, pot fi efectuate cu rate de transfer de 100 kbiți pe secundă, în modul standard, sau cu maxim 400 kbiți pe secundă, în modul rapid;
- rejectarea impulsurilor scurte, parazite, de pe magistrală, este asigurată de circuitele de filtrare implementate în fiecare dispozitiv cuplat la magistrală. Rejecția acestor impulsuri asigură păstrarea integrității datelor;
- numărul de dispozitive cuplabile pe aceeaşi magistrală **I^2C** este limitat doar de capacitatea maximă suportată de aceasta şi care este de 400 pF.

Circuitele integrate compatibile cu magistrala **I^2C** permit dezvoltarea rapidă a proiectării de la o schemă bloc funcțională la prototip, asigurând proiectanților o serie întregă de avantaje:

- structura extrem de simplă a magistralei (2 fire) minimizează interconexiunile cu exteriorul;

- protocolul complet integrat al magistralei **I²C** elimină folosirii decodificatoarelor de adrese şi a unei logici externe, suplimentare;
- capabilitățile de multimaster ale magistralei **I²C** permite testarea rapidă şi alinierea utilizatorilor, prin utilizarea unor conexiuni externe, la un sistem de calcul;
- disponibilitatea circuitelor integrate **I²C** sub amprente de tip **SO** (**S**mall **O**utline), **VSO** (**V**ery **S**mall **O**utline) şi **DIL** (**D**ual **I**n **L**ine) reduce necesitățile de spațiu.

2.2.3.1 SPECIFICAȚIILE INTERFEȚEI I²C

Pentru aplicații de control industrial pe 8 biți, care necesită utilizarea unor microcontroller-e, pot fi stabilite a priori anumite criterii de proiectare:

- un astfel de sistem este compus din cel puțin un microcontroller şi din alte dispozitive periferice, ca de pildă memorii şi circuite de intrare-ieşire;
- criteriul principal de proiectare constă în minimizarea costului de interconectare a diferitelor dispozitive din componența sistemului;
- un sistem care asigură o funcție de reglare (control) într-un proces nu necesită rate mari ale transferurilor de date;
- eficiența globală a sistemului depinde de natura circuitelor utilizate şi de structura magistralei de interconectare a acestora.

Pentru a satisface aceste criterii, este necesară utilizarea unei magistrale seriale, care deşi nu permite rate de transfer a informațiilor atât de mari ca o magistrală de interconectare de tip paralel, asigură minimizarea numărului firelor şi pinilor de interconectare între diversele circuite utilizate în proiect.

Dispozitivele care intercomunică prin intermediul unei magistrale seriale necesită utilizarea unor protocoale care au rolul de a elimina erorile, pierderile de informații şi conflictele pe magistrală şi de asemenea, posibilitatea ca unele dispozitive rapide să poată comunica cu dispozitive lente. Este necesar ca sistemul să poată funcționa independent de numărul de dispozitive înglobate în structura sa, sau cu alte cuvinte, adăugarea de dipozitive în structura sistemului să nu afecteze funcționarea acestuia.

2.2.3.2 CONCEPTUL DE MAGISTRALĂ I²C

Magistrala **I²C** permite cuplarea unor circuite compatibile în structura sistemului, indiferent de tehnologia de fabricație a acestora: NMOS, CMOS sau bipolară. Magistrala constă în două linii: o linie serială de date (**SDA**) şi o linie de ceas serial (**SCL**), ce manipulează informațiile între oricare două dispozitive cuplate la magistrală. Orice dispozitiv este recunoscut prin intermediul unei adrese unice asociate, indiferent dacă este vorba de un microprocesor, display cu cristale lichide, interfață de tastatură, etc., şi poate funcționa ca emițător sau receptor, depinzând de funcția realizată de acesta. O clasificare suplimentară a dispozitivelor cuplate la magistrala **I²C** constă în dispozitive ***master***, respectiv ***slave***. Un dispozitiv ***master*** este acela care poate iniția un transfer de date pe magistrală şi care generează semnalul de ceas ce coordonează transferul. În tot acest timp, orice alt dispozitiv adresat este privit ca ***slave***.

Magistrala **I²C** este o magistrală de tip ***multi-master***. Aceasta înseamnă că mai multe dispozitice care pot controla magistrala pot fi cuplate la aceasta. Posibilitatea de a cupla mai mult de un dispozitiv ***master*** la magistrală înseamnă că mai mult de un singur dispozitiv poate încerca să inițieze un transfer pe magistrală, la acelaşi moment de timp. Pentru a se evita această situație de incertitudine, a fost elaborată o procedură de arbitrare a priorităților, bazată pe conectarea de tip ŞI-cablat a tuturor dispozitivelor la magistrală. Semnalele de ceas pe durata arbitrării de priorități reprezintă rezultatul sincronizării semnalelor de ceas generate de

cele două dispozitive ***master*** prin utilizarea funcției de tip ŞI-cablat a liniilor **SCL**. Generarea semnalelor de ceas pe magistrală revine întotdeauna în sarcina dispozitivelor ***master***; fiecare dispozitiv ***master*** generează propriul său semnal de ceas pe durata transferului de date pe magistrala sistemului. Semnalele de ceas de pe magistrală pot fi doar alterate doar dacă un dispozitiv ***slave*** lent forțează linia de ceas la nivel logic LOW sau de un alt dispozitiv ***master***, pe durata arbitrării priorităților.

Ambele linii, **SDA** şi **SCL**, sunt bidirecționale şi conectate printr-o rezistență de pull-up la tensiunea de alimentare. Atunci când magistrala este liberă, ambele linii sunt în starea HIGH. Etajul de ieşire al dispozitivului conectat la magistală trebuie să fie de tip ***open-drain*** sau ***open-collector*** pentru a se realiza funcția ŞI-cablat. Pe magistrala **I²C**, transferurile de date pot fi efectuate cu rate de maxim 100 kbiți/s în modul standard sau maxim 400 kbiți/s în modul rapid. Numărul de dispozitive cuplabile la magistrala **I²C** este limitat doar de încărcarea capacitivă (maxim 400 pF) a liniilor magistralei.

2.2.3.3 TRANSFERURILE PE MAGISTRALA I²C

Datorită diversității tehnologiilor de implementare a circuitelor cuplabile la liniile interfeței **I²C**, nivelele logice nu sunt fixate şi depind de valoarea tensiunii de alimentare V_{DD}. Pentru transferul fiecărui bit este generat câte un impuls de ceas.

2.2.3.3.1 VALIDITATEA DATELOR

Datele vehiculate pe linia **SDA** trebuie să fie stabile pe durata HIGH a impulsului de ceas. Modificările stării liniei **SDA** trebuie să se producă doar atunci când semnalul de ceas este LOW.

2.2.3.3.2 CONDIȚIILE START ŞI STOP

Printre procedurile implementate pe magistrala **I²C**, situații de excepție sunt considerate condițiile de START şi STOP.

O tranziție din starea HIGH în starea LOW a liniei **SDA**, pe durata căreia linia **SCL** este HIGH, este interpretată ca o condiție de START.

O tranziție din starea LOW în starea HIGH a liniei **SDA**, pe durata căreia linia **SCL** este HIGH, este interpretată ca o condiție de STOP.

Condițiile de START şi de STOP sunt generate întotdeauna de un dispozitiv ***master***. După generarea unei condiții de START se consideră că magistrala este ocupată. Magistrala este considerată din nou ca fiind neutilizată după apariția unei condiții de STOP.

Detectarea condițiilor de START ş di de STOP de către dispozitivele ***slave*** cuplate la magistrală este foarte facilă dacă acestea înglobează ***hardware***-ul specializat de interfațare. Pentru dispozitivele care nu dispun de acest ***hardware*** specializat, se impune ca linia **SDA** să fie eşantionată de două ori pe durata unei perioade de ceas, pentru ca această tranziție să poată fi detectată.

2.2.3.3.3 TRANSFERURILE DE DATE PE MAGISTRALĂ

a) Transferurile de date sub formă de cuvânt

Orice cuvânt de date transferat pe magistrală trebuie să aibă lungimea de 8 biți. În schimb, numărul de octeți ce pot fi transferați pe linia **SDA** este practic nelimitat. Fiecare octet transferat trebuie să fie urmat de un bit de confirmare (***acknowledge***). Transferurile de date încep întotdeauna cu bitul cel mai semnificativ al octetului respectiv. Dacă un dispozitiv receptor nu poate accepta un alt octet de date înainte de a realiza o funcție specială cum ar fi de pildă tratarea unei întreruperi interne, acesta poate forța linia de ceas, **SCL**, la nivel LOW

pentru a forța ca emițătorul să intre în stare de WAIT. Transferul de date poate continua atunci când receptorul eliberează linia **SCL**.

În anumite cazuri, este posibilă utilizarea unui alt format pentru transferul pe magistrală. Un mesaj care începe cu o astfel de adresă poate fi terminat prin utilizarea unei condiții de STOP, chiar în timpul transmiterii unui octet de informație. În această situație nu se generează bitul de confirmare.

b) Bitul de confirmare

Transferurile de date cu confirmare sunt obligatorii pentru a se asigura integritatea datelor pe magistrală. Semnalul de ceas asociat bitului de confirmare este generat de dispozitivul ***master***. Pe durata acestui impuls de ceas, disozitivul emitent eliberează linia **SDA** (nivelul acesteia este HIGH).

Dispozitivul de recepție trebuie să forțeze linia **SDA** la nivel coborât pe durata impulsului de ceas de confirmare, astfel acest nivel coborât să rămână stabil pe durata HIGH a impulsului de ceas de confirmare.

În mod obişnuit, un dispozitiv ce realizează funcția de recepție mesaj trebuie să emită câte un semnal de confirmare după fiecare octet recepționat. Atunci când un dispozitiv ***slav***e cu funcție de recepție nu confirmă adresa asociată (de exemplu, ecest dispozitiv nu este capabil să răspundă deoarece efectuează un set de operații în timp real), linia de date trebuie lăsată neutilizată (HIGH) de către dispozitivul ***slave***. În această situție, dispozitivul ***master*** poate genera o condiție de STOP pentru a termina transferul. Dacă dispozitivul ***slave*** ce realizează funcția de recepție confirmă adresa asociată dar în procesul de transfer ulterior nu mai poate recepționa octeți, este, de asemenea, necesar ca dispozitivul ***master*** să termine transferul. Acest fapt este indicat prin faptul că receptorul nu confirmă recepționarea următorului octet, lasă linia **SDA** pe nivel HIGH, iar dispozitivul master generează condiția de STOP.

Dacă în procesul de transfer este implicat un dispozitiv ***master*** ce realizează funcția de recepție, acest dispozitiv trabuie să semnalizeze sfârşitul transferului prin neconfirmarea ultimului octet recepționat de la ***slave***. Dispozitivul ***slave*** trebuie să elibereze linia **SDA** pentru ca dispozitivul ***master*** să poată transmite o condiție de STOP.

2.2.3.4 ARBITRAREA PRIORITĂȚILOR ŞI GENERAREA CEASULUI

a) Sincronizarea pe magistrala **I²C**

Toate dispozitivele ***master*** generează propriul semnal de ceas pe linia **SCL** pentru a transmite mesaje pe magistrala **I²C**. Datele sunt valide doar pe durata HIGH a impulsurilor de ceas. Prezența unui semnal de ceas pe magistrală este necesară pentru procedura de arbitrare bit cu bit.

Sincronizarea ceasului este asigurată prin utilizarea conexiunii de tip ŞI-cablat a interfețelor de magistrală la linia **SCL**. Aceasta înseamnă că o tranziție din HIGH în LOW pe linia **SCL** va determina dispozitivele cuplate la magistrală să îşi înceapă procesul de contorizare a perioadelor LOW odată ce semnalul de ceas al unui dispozitiv a devenit LOW, se va menține linia **SCL** în această stare până semnalul de ceas devine din nou HIGH. Totuşi, tranziția din starea LOW în starea HIGH nu va determina schimbarea stării liniei de ceas dacă un alt semnal de ceas cuplat la linia de ceas a magistralei se află în stare LOW. Durata cât timp linia **SCL** va fi menținută în stare LOW va fi determinată de dispozitivul care este caracterizat de cea mai mare durată a nivelului coborât al ceasului. Celelalte dispozitive, caracterizate de o durată mai mică a palierului stării LOW a semnalului de ceas, trec în stare de WAIT cu semnalul de ceas la nivel ridicat.

Atunci când, toate dispozitivele, implicate în procesul de comunicare pe magistrală, şi-au încheiat contorizarea perioadei LOW a semnalului de ceas, linia respectivă va fi eliberată şi va trece în stare HIGH. În acest mod, nu vor mai exista diferențe între semnalele de ceas ale

dispozitivelor şi starea linie de ceas a magistralei, toate dispozitivele începându-şi contorizarea duratelor HIGH ale semnalelor de ceas. Primul dispozitiv care îşi încheie perioada HIGH a semnalului de ceas va forța linia **SCL** din nou la nivel LOW.

Semnalul de ceas de pe linia SCL este astfel sincronizat, având durata de nivel coborât determinată de dispozitivul caracterizat de cea mai lungă perioadă LOW a semnalului de ceas şi durata de nivel ridicat determinată de dispozitivul caracterizat de cea mai scurtă perioadă HIGH a semnalului de ceas.

b) Arbitrarea priorităților

Un dispozitiv ***master*** poate inița un transfer de date doar dacă magistrala este liberă. Două sau mai multe dispozitive ***master*** de magistrală pot genera o condiție de START pe durata timpului de HOLD din condiția de START. Arbitrarea are loc prin intermediul liniei de date, **SDA**, pe durata cât linia de ceas, **SCL**, este pe nivel HIGH. Astfel, unul dintre dispozitivele ***master*** transmite un nivel HIGH pe magistrală, în timp ce celălalt, care transmite un nivel LOW, îşi va dezactiva etajul de ieşire deoarece nivelul logic de pe magistrală nu corespunde cu nivelul logic transmis de către acesta. Arbitrarea poate continua pentru mai mulți biți. Prima etapă constă în compararea biților de adresă. Dacă două dispozitive ***master*** încearcă să adreseze acelaşi dispozitiv ***slave***, arbitrarea continuă cu compararea datelor. Deoarece adresele şi datele sunt utilizate pentru arbitrarea magistralei, se constată că nu există pierderi de informație pe liniile magistralei pe durata acestui proces.

Un dispozitiv ***master*** care pierde arbitrarea poate genera impulsuri de ceas până la încheierea procesului de transmitere a octetului în cursul căruia a pierdut arbitrarea.

2.2.4 IEŞIRILE MODULATE ÎN DURATĂ

Microcontroller-ul 80C552 posedă două canale de ieşire modulate în durată. Aceste ieşiri generează impulsuri ale căror durate şi factori de umplere pot fi programate. Frecvența de repetiție a impulsurilor este controlabilă prin intermediul registrului PWMP,având lungimea de 8 biți. Expresia frecvenței impulsurilor de ieşire este dată de următoarea ecuație:

$$f_{PWM} = \frac{f_{OSC}}{2 \times (1 + PWMP) \times 255} \tag{2.2}$$

Registrul PWMP asigură semnalul de ceas pentru un numărător; atât registrul PWMP, cât şi numărătorul fiind comune pentru ambele canale de ieşire. Numărătorul funcționează modulo 255. Valoarea de 8 biți a numărătorului este comparată cu două registre, PWM0 şi PWM1, asociate fiecare câte unui canal de ieşire. Dacă conținutul acestor registre este mai mare decât valoarea curentă a numărătorului, atunci semnalele de ieşire sunt forțate la nivel coborât. În schimb, dacă conținutul acestor registre este mai mic, sau cel puțin egal, cu valoarea curentă a numărătorului, atunci semnalele de ieşire sunt forțate la nivel ridicat - HIGH. Ca urmare, duratele impulsurilor de ieşire sunt determinate de conținutul registrelor PWM0 şi PWM1, denumite şi registre de prescalare.

Prin integrarea semnalelor de la ieşirile modulate în durată se poate obține un convertor digital-analog dual.

Încărcarea registrelor de prescalare cu valorile 00H sau FFH determină ca ieşirile canalelor să rămână constante la nivel ridicat sau coborât.

Reprogramarea registrelor de prescalare determină modificarea imediată a ieşirilor modulate, nefiind nevoie să se aştepte până la terminarea perioadei curente.

Factorul de umplere al impulsurilor de la ieşire rezultă ca fiind:

$$\gamma = \frac{PWM}{255 - PWM} \tag{2.3}$$

În figura 2.3 esteprezentată diagrama funcțională a celor două canale de ieşire modulate în durată.

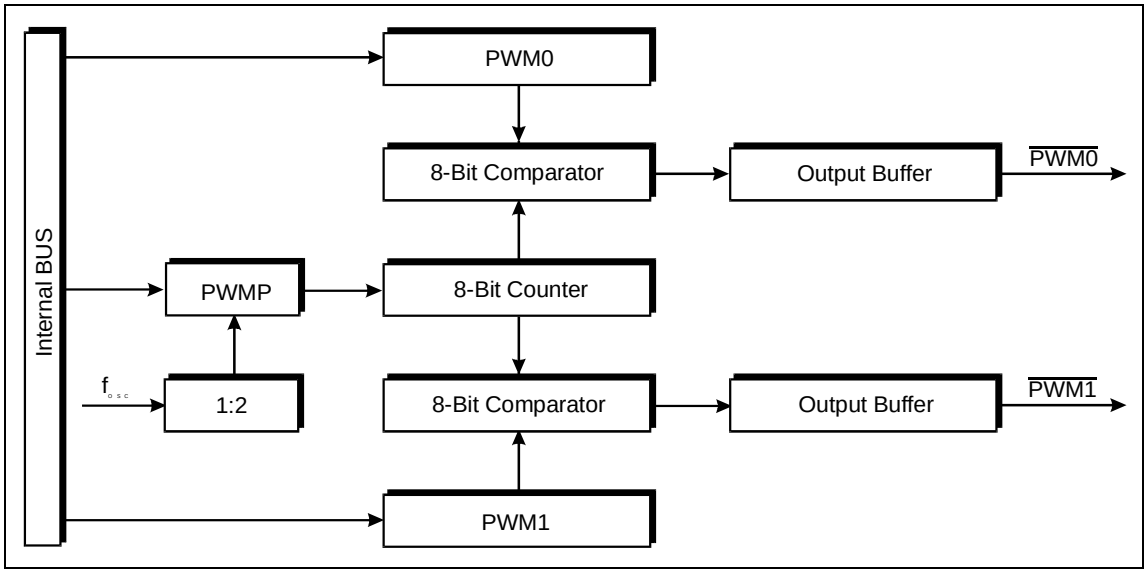

Figura 2.3 - Structura ie¿irilor modulate în durată.

2.2.5 SECȚIUNEA ANALOGICĂ A MICROCONTROLLER-ULUI

Secțiunea analogică este reprezentată de un multiplexor analogic şi de un convertor analog-digital cu aproximații succesive, cu rezoluția de 10 biți, ce furnizează rezultatul în cod binar direct (figura 2.4).

Tensiunea de referință a convertorului analog-digital se aplică structurii prin intermediul unui pin dedicat acestei manipulări.

Un ciclu de conversie durează 50 de ciclii maşină, ceea ce înseamnă aproximativ 50 µs pentru frecvența ceasului de 11,0592 MHz. Codul binar de ieşire nu prezintă discontinuități în funcția de transfer a convertorului analog-digital.

Atât intrările multiplexorului analogic, cât şi intrarea convertorului analog-digital acceptă tensiuni de intrare în gama (0 ÷ +5)V.

Conversia analog-digitală este efectuată prin metoda aproximațiilor succesive; în figura 2.5 sunt reprezentate elementele componente ale convertorului analog-digital cu aproximații succesive. Structura conține un convertor digital-analogic (DAC) care converteşte conținutul registrului de aproximații succesive într-o tensiune V_{DAC}. Această tensiune este comparată cu tensiunea de intrare, V_{in}; ieşirea comparatorului este furnizată circuitului de comandă a registrului de aproximații succesive, care controlează funcționarea acestuia.

O conversie este inițiată prin setarea bitului ADCS (ADC Start) din registrul ADCON (A/D Control). Acest bit poate fi modificat atât prin ***software***, cât şi prin ***hardware*** sau combinat.

Declanşarea unei conversii analog-digitale exclusiv prin mijloace software se poate face doar atunci când bitul ADCON.5 (ADEX) este setat la "0"; începerea unui ciclu de conversie se efectuează prin setarea bitului de control ADCS. La inițierea prin software a unei conversii, aceasta este demarată la începutul ciclului maşină care urmează celui în care a fost setat bitul ADCS.

Declanşarea unei conversii analog-digitale prin mijloace combinate software-hardware se poate face doar atunci când bitul ADCON.5 = "1"; inițierea unui ciclu de conversie poate fi efectuată fie la fel ca în cazul precedent, prin setarea bitului de start (ADCS), fie prin aplicarea unui front crescător pinului extern STADC. În această ultimă situație, frontul aplicat trebuie precetat de un nivel logic coborât, care să dureze minimum un ciclu maşină, respectiv urmat de un nivel logic ridicat, cu durata de minimum un ciclu maşină. Tranziția aplicată pinului STADC este recunoscută la sfârşitul ciclului maşină curent, iar conversia este inițiată la începutul următorului ciclu maşină.

Următoarele două cicluri maşină sunt folosite pentru iniţializarea convertorului analog-digital. La sfârşitul primului ciclu, se citeşte indicatorul de stare ADCS; dacă indicatorul este citit pe durata efectuării unei conversii, rezultatul citirii acestui indicator constă în poziţionarea acestuia în "1". La sfârşitul celui de-al doilea ciclu maşină începe eşantionarea semnalelor de intrare.

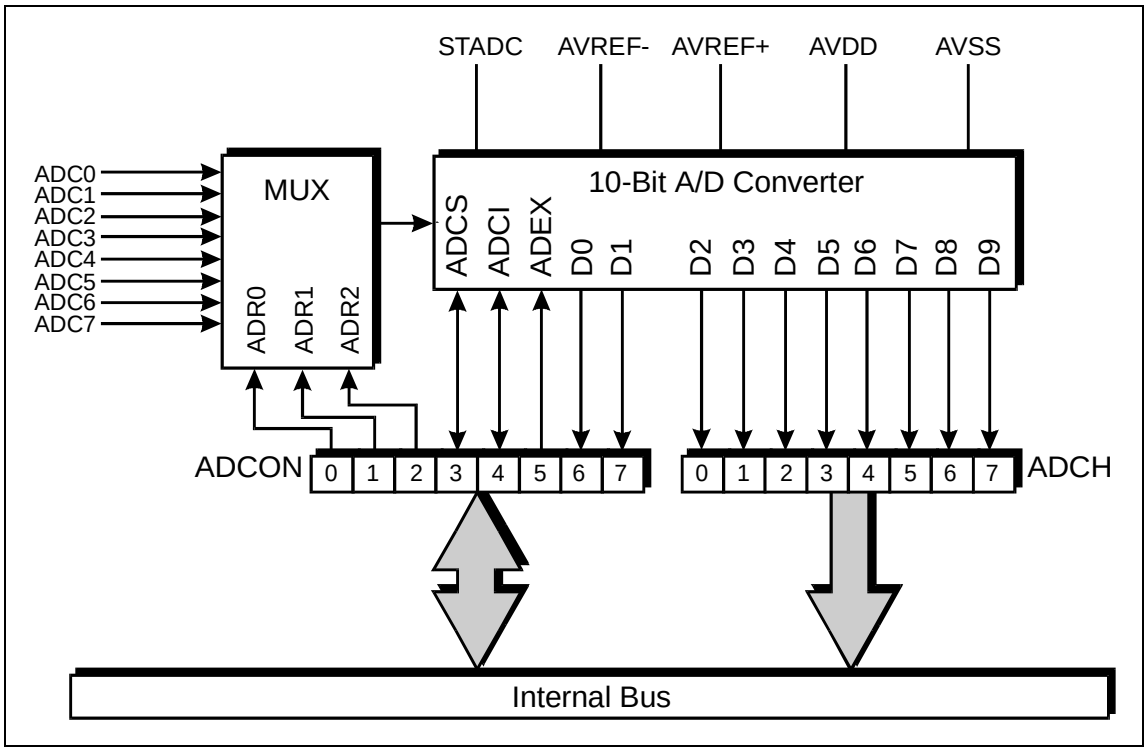

Figura 2.4 - Sec¡iunea analogicå a microcontroller-ului 80C552.

Pe durata următoarelor 8 cicluri maşină este eşantionată tensiunea de intrare aplicată pinului, anterior selectat, al portului P_5. Această tensiune trebuie să rămână stabilă pe această durată pentru a se obţine un rezultat corect. Se admite o viteză de variaţie a tensiunii eşantionate de cel mult 10V/ms.

În continuare, se efectuează conversia analog-digitală propriu-zisă. Aceasta se desfăşoară prin setarea bitului cel mai semnificativ la "1", de către circuitul de control al registrului de aproximaţii succesive; ieşirea registrului SAR este convertită într-o tensiune proporţională şi comparată cu tensiunea de intrare. Dacă aceasta este mai mare, bitul MSB rămâne setat la "1"; în caz contrar, acest bit este resetat. Procesul continuă în ordine inversă a importanţei biţilor, până când toţi cei zece biţi au fost testaţi. În acest moment, rezultatul conversiei este conţinut în registru de aproximaţii succesive.

Sfârşitul conversiei analog-digitale pe 10 biţi este semnalizată prin setarea bitului ADCI (ADCON.4) în cadrul registrului de comenzi şi stare. Cei mai semnificativi 8 biţi ai rezultatului sunt memoraţi în registrul ADCH. Cei mai puţin semnificativi doi biţi sunt memoraţi în biţii ADCON.7 şi ADCON.6 ai registrului de comenzi şi stare. Utilizatorul poate ignora cei mai puţin semnificativi doi biţi ai rezultatului şi poate utiliza doar cei mai semnificativi 8 biţi, memoraţi în registrul ADCH.

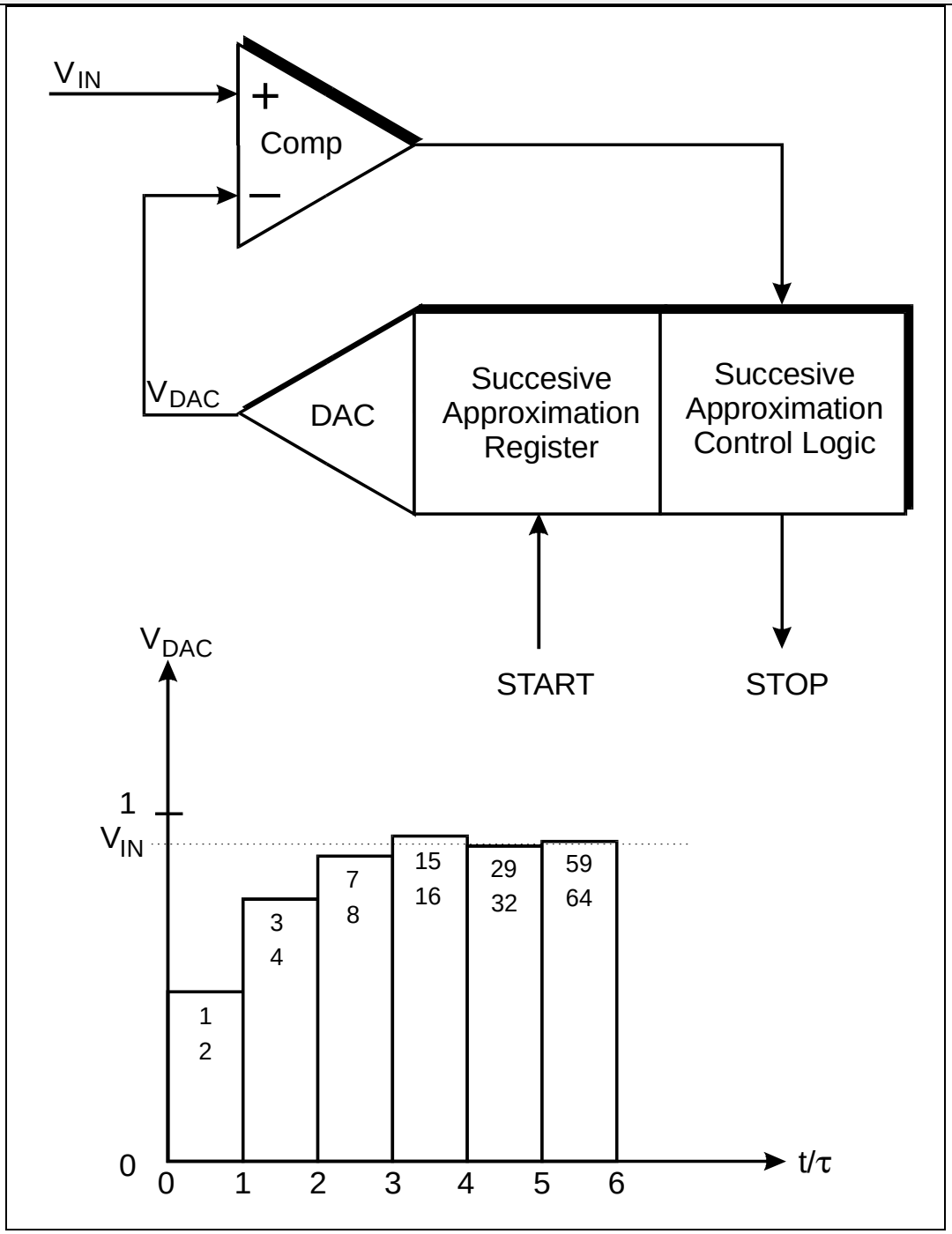

Figura 2.5 - Convertorul analog-digital.

Figura 2.6 ilustrează desfăşurarea procesului de conversie analog-digitală cu aproximații succesive.

În orice situație, durata unei conversii este de 50 ciclii maşină. Bitul de sfârşit de conversie, ADCI, este setat, iar bitul de start conversie, ADCS, este resetat după 50 de cicluri maşină de la poziționarea pe nivel ridicat al acestuia din urmă.

Biții de control ADCON.0, ADCON.1 şi ADCON.2 sunt utilizați pentru comanda intrărilor de selecție ale multiplexorului analogic, de tip 8:1.

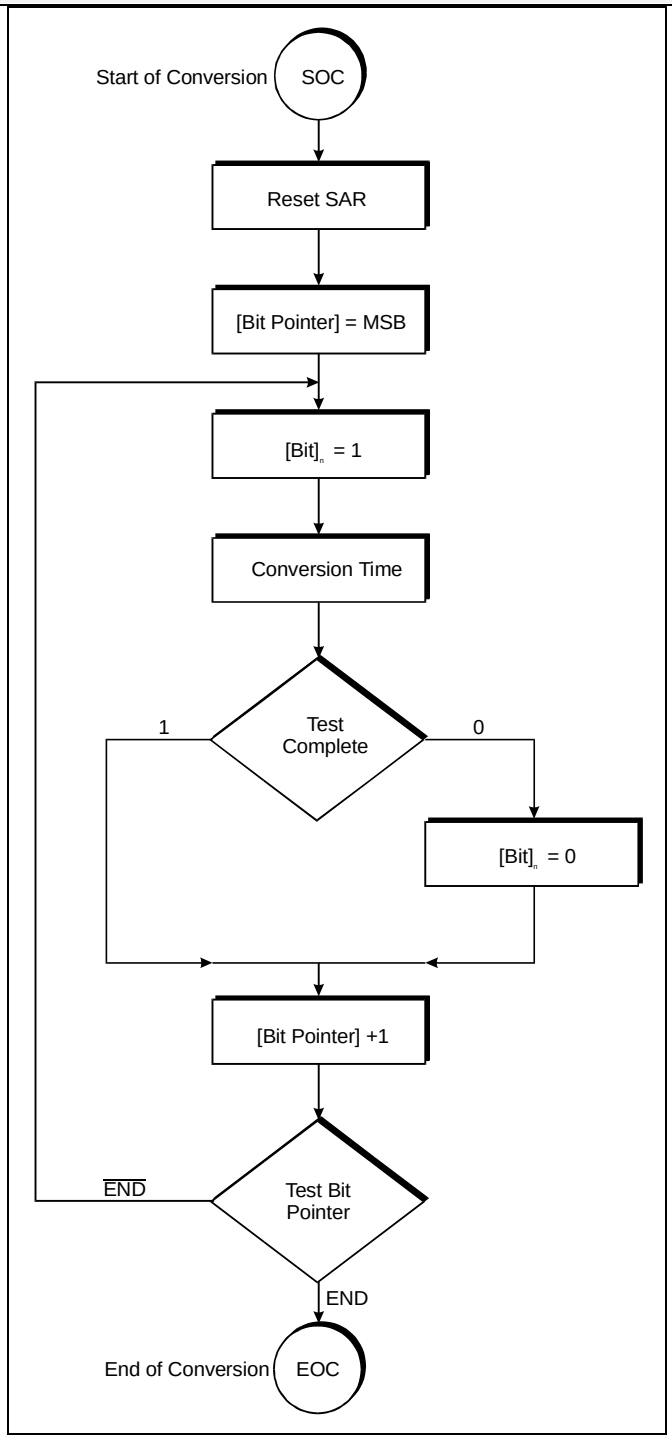

Figura 2.6 - Organigrama de desfå¿urare a unei conversii.

O rutină de conversie aflată în desfăşurare nu este afectată de o comandă de start, indiferent de natura sa (***hardware*** sau ***software***).

Rezultatul obținut la încheierea unui proces de conversie rămâne neafectat, cu bitul ADCI poziționat pe "1", chiar dacă se comandă intrarea în modul "inactiv" (***idle***).

La intrarea în modul de lucru "***idle***" sau "***power-down***" al microcontroller-ului, o conversie analog-digitală aflată în plin proces de desfăşurare este întreruptă. Rezultatele parțiale obținute până acum sunt iremediabil pierdute, nefiind transferate în registrele rezultat şi/sau în registrul de comenzi şi stare.

Convertorul analog-digital dispune de pini proprii de alimentare (AV_{DD} şi AV_{SS}) şi de doi pini pentru tensiunea de referință externă (V_{ref+} şi V_{ref-}). Rezultatul conversiei poate fi

calculat prin utilizarea următoarei caracteristici de transfer:

$$N = 2^{10} \times \frac{V_{IN} - V_{REF-}}{V_{REF+} - V_{REF-}} = 1024 \times \frac{V_{IN} - V_{REF-}}{V_{REF+} - V_{REF-}} \quad (2.4)$$

Prin utilizarea numai a celor mai semnificativi 8 biți ai rezultatului, conținuți în registrul ADCH, caracteristica de transfer devine:

$$N = 2^{8} \times \frac{V_{IN} - V_{REF-}}{V_{REF+} - V_{REF-}} = 256 \times \frac{V_{IN} - V_{REF-}}{V_{REF+} - V_{REF-}} \quad (2.5)$$

2.2.6 MĂSURAREA INTERVALELOR DE TIMP PRIN UTILIZAREA REGISTRELOR DE CAPTARE A EVENIMENTELOR

Atunci când un eveniment extern recursiv este reprezentat sub forma unui front crescător sau descrecător care este aplicat unuia dintre cei patru pini de captare a evenimentelor, intervalul de timp dintre două evenimente poate fi măsurat folosind temporizatorul T_2 şi unul dintre cele patru registrele de captare a evenimentelor, CT0, CT1, CT2 sau CT3. La apariția unui eveniment, conținutul timer-ului T_2 este transferat în registrul de captare a evenimentelor selectat şi este generat un semnal de întrerupere specific acestui registru, CT0I, CT1I, CT2I sau CT3I. Indicatorii de stare anterior menționați reprezintă patru dintre cei 8 biți ai registrului de funcții speciale, denumit TM2IR.

Această rutină de întrerupere poate fi folosită pentru calcularea intervalului de timp corespunzător, dacă se cunoaşte valoarea anterioară a timer-ului T2. Pentru o frecvență a ceasului de 12 MHz programarea timer-ului poate fi făcută astfel încât acesta să furnizeze indicații de depăşire la fiecare 524 ms. Dacă intervalul de timp dintre două evenimente succesive este mai scurt de 524 ms, atunci calculul intervalului de timp este foarte simpu, rutina de întrerupere fiind mult mai scurtă. Pentru intervale de timp mai mari de 524ms, trebuie utilizată o extensie a timer-ului T_2.

Timer-ul T_2 este conectat la patru registre de 16 biți, menționate anterior şi, de asemenea la trei registre de comparare, cu lungimea de 16 biți (figura 2.7). Registrele de comparare potfi folosite pentru a seta, reseta sau comuta pinii de ieşire ai portului P_4 la anumite intervale preprogramate de timp.

În continuare sunt prezentate registrele utilizate în secțiunea de captare a evenimentelor şi de comparare, adresele lor, precum şi semnificația fiecăruia dintre biții aparținând acestor registre.

Registrul TM2CON (EAH) - valoare la reset 00 H

(MSB)							(LSB)
T2IS1	T2IS0	T2ER	T2B0	T2P1	T2P0	T2MS1	T2MS0

T2MS1: 0 0 - Timer T2 oprit; 0 1 - ceas = f_{OSC}:12; 1 0 - mod test; 1 1 - ceas extern (T2).

T2P1: 0 0 - ceas : 1 (intern sau extern); 0 1 - ceas : 2 (intern sau extern); 1 0 - ceas : 4 (intern sau extern); 1 1 - ceas : 8 (intern sau extern).

T2B0: Indicator intrerupere depasire pe 8 biti Timer T2

T2ER: activare reset extern Timer T2; daca este 1, T2 poate fi resetat cu un front crescator pe pinul RT2 (P1.5).

T2IS0: selectare intrerupere de depasire pe 8 biti Timer T2

T2IS1: selectare intrerupere de depasire pe 16 biti Timer T2

Registrul IEN1 (E8H) - valoare la reset 00 H

(MSB) (LSB)

ET2	ECM2	ECM1	ECM1	ECT3	ECT2	ECT1	ECT0

ECT0 – activare intrerupere registru captare 0
ECT1 – activare intrerupere registru captare 1
ECT2 – activare intrerupere registru captare 2
ECT3 – activare intrerupere registru captare 3
ECM1 – activare intrerupere comparator 0 timer T2
ECM1 – activare intrerupere comparator 1 timer T2
ECM2 – activare intrerupere comparator 2 timer T2
ET2 – activare intrerupere depasire timer T2

Registrul CTCON (EBH) - valoare la reset 00 H

(MSB) (LSB)

CTN3	CTP3	CTN2	CTP2	CTN1	CTP1	CTN0	CTP0

CTP0 – captare pe front ↑ al CT0I
CTN0 – captare pe front ↓ al CT0I
CTP1 – captare pe front ↑ al CT1I
CTN1 – captare pe front ↓ al CT1I
CTP2 – captare pe front ↑ al CT2I
CTN2 – captare pe front ↓ al CT2I
CTP3 – captare pe front ↑ al CT3I
CTN3 – captare pe front ↓ al CT3I

Registrul RTE (EFH) - valoare la reset 00 H

(MSB) (LSB)

TP4.7	TP4.6	RP4.5	RP4.4	RP4.3	RP4.2	RP4.1	RP4.0

RP4.0 – daca e 1, P4.0 e resetat la egalitatea CM1 si T2
RP4.1 – daca e 1, P4.1 e resetat la egalitatea CM1 si T2
RP4.2 – daca e 1, P4.2 e resetat la egalitatea CM1 si T2
RP4.3 – daca e 1, P4.3 e resetat la egalitatea CM1 si T2
RP4.4 – daca e 1, P4.4 e resetat la egalitatea CM1 si T2
RP4.5 – daca e 1, P4.5 e resetat la egalitatea CM1 si T2
TP4.6 – daca e 1, P4.6 basculeaza la egalitatea CM2 si T2
TP4.7 – daca e 1, P4.7 basculeaza la egalitatea CM2 si T2

Registrul STE (EEH) - valoare la reset 00 H

(MSB) (LSB)

TG4.7	TG4.6	SP4.5	SP4.4	SP4.3	SP4.2	SP4.1	SP4.0

SP4.0 – daca e 1, P4.0 e setat la egalitatea CM0 si T2
SP4.1 – daca e 1, P4.1 e setat la egalitatea CM0 si T2
SP4.2 – daca e 1, P4.2 e setat la egalitatea CM0 si T2
SP4.3 – daca e 1, P4.3 e setat la egalitatea CM0 si T2
SP4.4 – daca e 1, P4.4 e setat la egalitatea CM0 si T2
SP4.5 – daca e 1, P4.5 e setat la egalitatea CM0 si T2
TG4.6 – basculare bistabil
TG4.7 – basculare bistabil

Registrul TM2IR (C8H) - valoare la reset 00 H

(MSB) (LSB)

T2OV	CM2	CM1	CMO	CTI3	CTI2	CTI1	CTI0

CTI0 – indicator

intrerupere CTI0
indicator intrerupere CTI1
indicator intrerupere CTI2
indicator intrerupere CTI3
indicator intrerupere CM0
indicator intrerupere CM1
indicator intrerupere CM2
indicator depasire pe 16 biti a timer-ului T2

Registrul IP1 (F8H) - valoare la reset 00 H

(MSB) (LSB)

PT2	PCM2	PCM1	PCM0	PCT3	PCT2	PCT1	PCT0

nivel de prioritate al intreruperii asociate registrului de captare 0
nivel de prioritate al intreruperii asociate registrului de captare 1
nivel de prioritate al intreruperii asociate registrului de captare 2
nivel de prioritate al intreruperii asociate registrului de captare 3
nivel de prioritate al intreruperii asociate comparatorului 1
nivel de prioritate al intreruperii asociate comparatorului 1
nivel de prioritate al intreruperii asociate comparatorului 2
nivel de prioritate al intreruperii asociate depasirii timer-ului T2

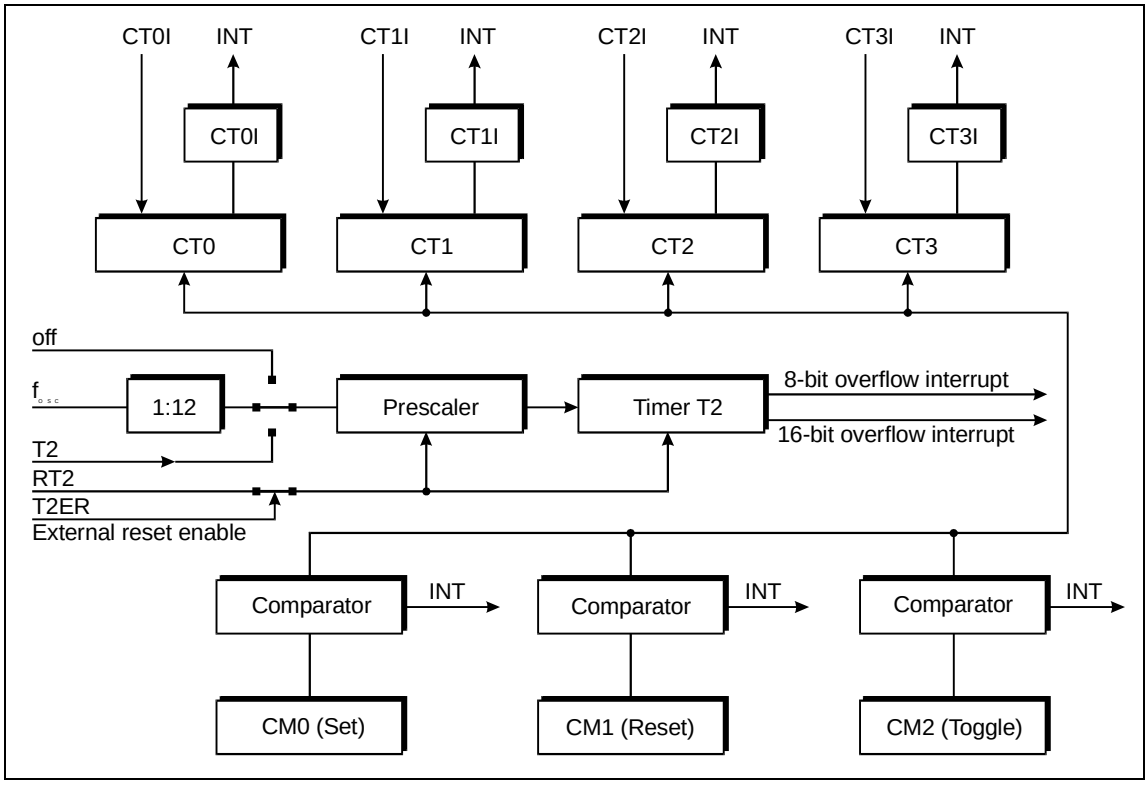

Figura 2.7 - Sec¡iunea de captare a evenimentelor ¿i comparare a microcontroller-ului 80C552.

3. PREZENTAREA SETULUI DE INSTRUCȚIUNI AL MICROCONTROLLER-ULUI 80C51

3.1 ACALL addr11

Funcția: Absolute Call

Descriere: Apelează necondiționat o subrutină localizată la adresa indicată. Instrucțiunea incrementează contorul programului de două ori, pentru a obține adresa următoarei instrucțiuni, apoi încarcă în stivă rezultatul pe 16 biți (primul octet este cel mai puțin semnificativ) şi incrementează pointerul stivei de două ori. Adresa destinație e obținută prin concatenarea succesivă a celor 5 biți mai semnificativi ai contorului programului, după incrementare, biții 7÷5 ai codului operației şi octetul al doilea al instrucțiunii. Subrutina apelată trebuie să înceapă în acelaşi bloc (de dimensiune 2 kocteți) al memoriei de program ca şi primul octet al instrucțiunii care urmează instrucțiunii ACALL. Nu se afecteaza indicatorii de condiție.

Exemplu: Inițial registrul pointer de stivă, SP, conținea valoarea 07H. Eticheta SUBRTN e localizată în memorie la adresa 0345H. După execuția instrucțiunii:

```
ACALL    SUBRTN
```

la locatia 0123H, SP va conține valoarea 09H, locațiile interne RAM 08H şi 09H vor conține 25H şi, respectiv 01H, iar PC va contine 0345H.

Octeți: 2

Ciclii maşină: 2

Codare:

$a_{10}\ a_9\ a_8\ 1$	0 0 0 1	$a_7\ a_6\ a_5\ a_4$	$a_3\ a_2\ a_1\ a_0$

Operații:

ACALL

$(PC) \leftarrow (PC)+2$

$(SP) \leftarrow (SP)+1$

$(SP) \leftarrow (PC_{7\div0})$

$(SP) \leftarrow (SP)+1$

$(SP) \leftarrow (PC_{15\div8})$

$(PC_{10\div0}) \leftarrow$ adresa blocului

3.2 ADD A, <scr-byte>

Funcția: Adunare

Descriere: Instrucțiunea ADD adună acumulatorul cu octetul indicat, lăsând rezultatul în acumulator. Indicatorii de transport şi de transport auxiliar sunt setați, respectiv dacă există un transport de la bitul 7 sau de la bitul 3, şi sunt resetați dacă nu există transport. Când se adună întregi fără semn, indicatorul de transport indică depăşire. Indicatorul de depăşire, OV, este setat dacă este un transport la bitul 6, dar nu mai departe de bitul 7, sau un transport de la bitul 7, dar nu provenit de la bitul 6. În celelalte situații, OV este resetat. Când se adună întregi cu semn, indicatorul OV semnalizează un număr negativ ca sumă a doi operanzi pozitivi, sau o sumă pozitivă ca rezultat a adunării a doi operanzi negativi. Sunt permise 4 moduri de adresare a operandului sursă: adresare prin registru, adresare directă, adresare indirectă prin registru sau adresare imediată.

Exemplu: Acumulatorul conține valoarea 0C3H (11000011B) iar registrul 0

conține 0AAH (10101010B). Instrucțiunea:

ADD A, R_0

va determina ca valoarea conținută de acumulator să fie 6DH (01101101B), cu indicatorul AC resetat şi atât indicatorul de transport, cât şi OV, setate la 1.

ADD A, Rn

Octeți: 1
Cicluri maşină: 1
Operații: ADD
(A) ← (A) + (Rn)

ADD A, direct

Octeți: 2
Cicluri maşină: 1
Operații: ADD
(A) ← (A) + (direct)

ADD A, @Ri

Octeți: 1
Cicluri maşină: 1
Operații: ADD
(A) ← (A) + ((Ri))

ADD A, #data

Octeți: 2
Cicluri maşină: 1
Operații: ADD
(A) ← (A) + #data

3.3 ADDC A, <scr-byte>

Funcția: Adunare cu indicatorul de transport

Descriere: Instrucțiunea ADDC adună acumulatorul cu octetul indicat şi cu indicatorul de transport, lăsând rezultatul în acumulator. Indicatorii de transport şi de transport auxiliar sunt setați, respectiv dacă există un transport de la bitul 7 sau de la bitul 3, şi sunt resetați dacă nu există transport. Când se adună întregi fără semn, indicatorul de transport indică depăşire. indicatorul de depăşire, OV, este setat dacă este un transport la bitul 6, dar nu mai departe de bitul 7, sau un transport de la bitul 7, dar nu provenit de la bitul 6. În celelalte situații, OV este resetat. Când se adună întregi cu semn, indicatorul OV semnalizează un număr negativ ca sumă a doi operanzi pozitivi, sau o sumă pozitivă ca rezultat a adunării a doi operanzi negativi. Sunt permise 4 moduri de adresare a operandului sursă: adresare prin registru, adresare directă, adresare indirectă prin registru sau adresare imediată.

Exemplu: Acumulatorul conține 0C3H (11000011B) şi registrul 0 conține 0AAH (10101010B) cu setarea indicatorului de transport. Instrucțiunea:

ADDC A, R0

va determina ca acumulatorul să conțină valoarea 6EH (01101110B), cu indicatorul AC resetat, iar indicatorul de transport şi OV setate la 1.

ADDC A, Rn

Octeți: 1
Cicluri maşină: 1
Operații: ADDC
(A) ← (A) + (C) + (Rn)

ADDC A, direct

Octeți: 2
Cicluri maşină: 1
Operații: ADDC
(A) ← (A) + (C) + (direct)

ADDC A, @Ri

Octeți: 1
Cicluri maşină: 1
Operații: ADDC
(A) ← (A) + (C) + ((Ri))

ADDC A, #data

Octeți: 2
Cicluri maşină: 1
Operații: ADDC
(A) ← (A) + (C) + #data

3.4 AJMP addr11

Funcția: Absolute Jump
Descrierea: AJMP transferă execuția programului la adresa indicată, care este formată la momentul rulării prin concatenarea celor 5 biți mai semnificativi ai contorului programului, PC, (după incrementarea de două ori a acestuia), biții 7÷5 de cod ai operației şi al doilea octet al instrucțiunii. Destinația trebuie să fie în acelaşi bloc de 2 kocteți din memoria de program ca primul octet al instrucțiunii următoare instrucțiunii AJMP.
Exemplu: Eticheta JMPADR e situată la locația de memorie 0123H. Instrucțiunea

```
AJMP JMPADR
```

este situată la locația 0345H şi va încarca contorul programului, PC, cu 0123H.
Octeți: 2
Cicluri maşină: 2
Operații: AJMP
(PC) ← (PC) + 2
($PC_{10\div0}$) ← adresa de pagină

3.5 ANL <dest-byte>, <scr-byte>

Funcția: ŞI LOGIC între variabile de tip octet
Descriere: ANL efectuează operația de ŞI LOGIC între variabilele de tip octet indicate şi încarcă rezultatul în variabila destinație. Nu afectează nici un indicator de condiție. Cei doi operanzi permit 6 combinări de moduri de adresare. Când destinația este acumulatorul, sursa poate fi adresată prin registru, direct, indirect prin registru sau imediat; când destinația este o adresă directă, sursa poate fi acumulatorul sau o dată imediată.
Notă: Când această instrucțiune este utilizată pentru a modifica un port de ieşire, valoarea utilizată ca dată inițială va fi citită de la ieşire, nu de la pinii de intrare.
Exemplu: Dacă acumulatorul conține valoarea 0C3H (11000011B) şi registrul 0 conține valoarea 55H (01010101B) atunci instrucțiunea

```
ANL A, R0
```

va determina ca acumulatorul să conțină valoarea 41H. Când destinația este un octet direct adresabil, această instrucțiune va reseta combinațiile

de biți din orice locație RAM sau registru hardware. Octetul mascat care determină resetarea biților va fi o constantă conținută în instrucțiune sau o valoare creată în acumulator la rulare. Instrucțiunea

```
ANL  P1, #0111001B
```

va reseta biții 7, 3 şi 2 ai portului de ieşire 1.

ANL A, Rn

Octeți: 1
Cicluri maşină: 1
Operații: ANL
(A) ← (A) ∧ (Rn)

ANL A, direct

Octeți: 2
Cicluri maşină: 1
Operații: ANL
(A) ← (A) ∧ (direct)

ANL A, @Ri

Octeți: 1
Cicluri maşină: 1
Operații: ANL
(A) ← (A) ∧ ((Ri))

ANL A, #data

Octeți: 2
Cicluri maşină: 1
Operații: ANL
(A) ← (A) ∧ #data

ANL direct, A

Octeți: 2
Cicluri maşină: 1
Operații: ANL
(direct) ← (direct) ∧ (A)

ANL direct, #data

Octeți: 3
Cicluri maşină: 2
Operații: ANL
(direct) ← (direct) ∧ #data

3.6 ANL C, <scr-bit>

Funcția: ŞI LOGIC între variabile de tip bit
Descriere: Dacă valoarea booleană a bitului sursă este un 0 logic, atunci resetează indicatorul de transport; altfel acest indicator rămâne în starea curentă. Simbolul ('/') precedind operandul, în limbaj de asamblare, indică faptul că complementul logic al bitului adresat e utilizat ca valoare sursă, fără însă ca bitul sursă să fie afectat. Pentru operandul sursă este permisă numai adresarea directă.
Exemplu: Seteaza indicatorul de transport dacă şi numai dacă P1.0=1, ACC.7=1 şi OV=0:

```
MOV  C, P1.0        ; incarca carry cu bitul 0 al portului 1
ANL  C, ACC.7       ; SI cu bitul 7 al acumulatorului
ANL  C, /OV; SI cu negatul indicatorului de depasire.
```

ANL C, bit

Octeți: 2

Cicluri maşină: 2
Operații: ANL
(C) ← (C) ∧ (bit)

ANL C, /bit
Octeți: 2
Cicluri maşină: 2
Operații: ANL
(C) ← (C) ∧ ⌝(bit)

3.7 CJNE <dest-byte>, <scr-byte>,rel

Funcția: Comparație şi salt dacă nu este egalitate
Descriere: Instrucțiunea CJNE compară cei doi operanzi şi determină un salt în execuția programului dacă valorile lor nu sunt egale. Destinatia saltului este compusă prin adunarea deplasarii relative cu semn, în ultimul octet al instrucțiunii, cu conținutul contorului programului, PC, după incrementarea acestuia la adresa de start a următoarei instrucțiuni. Indicatorul de transport e setat dacă valoarea întregului fără semn a octetului destinație e mai mică decât valoarea întregului fără semn a <scr-byte>; altfel, indicatorul de transport e resetat. instrucțiunea nu afectează nici un operand. Cei doi operanzi permit 4 combinații de moduri de adresare: acumulatorul poate fi comparat cu orice octet direct sau imediat adresabil, cu orice locație indirectă de RAM sau registrii de lucru pot fi comparați cu o constantă imediată.
Exemplu: Acumulatorul conține valoarea 34H. Registrul 7 conține valoarea 56H. Prima instrucțiune din secvența:

```
                CJNE  R7, #60H, NOT_EQ
                R7=60H
NOT_EQ:         JC    REQ_LOW
```

setează indicatorul de transport şi sare la instrucțiunea cu eticheta NOT_EQ. Prin testarea indicatorului de transport, această instrucțiune determină dacă registrul R7 e mai mare sau mai mic decât 60H.
Dacă data care a fost prezentă la portul 1 este 34 H, atunci instrucțiunea

```
WAIT:CJNE  A, P1, WAIT
```

resetează indicatorul de transport şi continuă cu următoarea instrucțiune din secvență, până când acumulatorul se egalează cu data citită din portul P1. Dacă în portul P1 au fost introduse alte valori, programul va bucla în acest punct până când data din portul P1 va deveni 34H.

CJNE A, direct, rel
Octeți: 3
Cicluri maşină: 2
Operații: (PC) ← (PC) + 3
IF (A) < > (direct)
THEN
(PC) ← (PC) + offset relativ
IF (A) < (direct)
THEN
(C) ← 1
ELSE
(C) ← 0

CJNE A, #data, rel
Octeți: 3
Cicluri maşină: 2

Operații: (PC) ← (PC) + 3
IF (A) < > #data
THEN
(PC) ← (PC) + offset relativ
IF (A) < #data
THEN
(C) ← 1
ELSE
(C) ← 0

CJNE Rn, #data, rel

Octeți: 3
Cicluri maşină: 2
Operații: (PC) ← (PC) + 3
IF (Rn) < > #data
THEN
(PC) ← (PC) + offset relativ
IF (Rn) < #data
THEN
(C) ← 1
ELSE
(C) ← 0

CJNE @Ri, #data, rel

Octeți: 3
Cicluri maşină: 2
Operații: (PC) ← (PC) + 3
IF ((Ri)) < > #data
THEN
(PC) ← (PC) + offset relativ
IF ((Ri)) < #data
THEN
(C) ← 1
ELSE
(C) ← 0

3.8 CLR A

Funcția: Resetează (inițializează) acumulatorul
Descriere: Acumulatorul este resetat (toti biții sunt 0). Nu este afectat nici un indicator de condiție.
Exemplu: Acumulatorul conține valoarea 5CH (01011100B). Instrucțiunea:

```
CLR  A
```

va determina conținutul acumulatorului să fie 00H (00000000B).

Octeți: 1
Cicluri maşină: 1
Operații: CLR
(A) ← 0

3.9 CLR bit

Funcția: Resetează bitul
Descriere: Bitul specificat este resetat (poziționat la 0). Nici un alt indicator de condiție nu este afectat. Instrucțiunea **CLR bit** poate opera asupra indicatorului de transport sau asupra oricărui bit direct adresabil.

Exemplu: Portul 1 a fost înscris cu valoarea 5DH (01011101B). Instrucțiunea

```
CLR  P1,2
```

va lăsa portul P1 setat la valoarea 59H (01011001B).

CLR C

Octeți: 1
Cicluri maşină: 1
Operații: CLR
(C) ← 0

CLR bit

Octeți: 2
Cicluri maşină: 1
Operații: CLR
(bit) ← 0

3.10 CPL A

Funcția: Complementarea acumulatorului
Descriere: Fiecare bit al acumulatorului este complementat logic (complement față de 1). Biții care anterior erau 1 devin 0, şi invers. Nici un indicator de condiții nu este afectat.
Exemplu: Acumulatorul conține valoarea 5CH (01011100B). Instrucțiunea:

```
CPL  A
```

va determina ca acumulatorul să conțină valoarea 0A3H (10100011B).
Octeți: 1
Cicluri maşină: 1
Operații: CPL
(A) ← ⌉(A)

3.11 CPL bit

Funcția: Complementează o variabilă de tip bit
Descriere: Instrucțiunea efectuează complementul variabilei de tip bit specificate. Un bit care era 1 devine 0, şi invers. Nu sunt afectați indicatorii de condiții.CLR poate opera asupra bitului de transport sau asupra oricărui alt bit direct adresabil.

Notă: Când instrucțiunea este utilizată pentru a modifica un bit de ieşire, valoarea utilizată ca dată originală va fi citită ca dată de ieşire şi nu de la intrare.

Exemplu: Portul 1 a fost înscris anterior cu valoarea 5DH (01011101B).Secvența de instrucțiuni:

```
CPL  P1,1
CPL  P1,2
```

va determina ca portul să fie setat la valoarea 5BH (01011011B).

CPL C

Octeți: 1
Cicluri maşină: 1
Operații: CPL
(C) ← ⌉(C)

CPL bit

Octeți: 2
Cicluri maşină: 1
Operații: CPL
(bit) ← ⌉(bit)

3.12 **DA A**

Functia: Ajustarea zecimală a acumulatorului

Descriere: Instrucțiunea DA A ajustează valoarea pe 8 biți din acumulator, ca rezultat al ultimei instrucțiuni de adunare a două variabile (reprezentate în format BCD-împachetat), producând două cifre de câte 4 biți. Pentru a se efectua instrucțiunea de adunare au fost folosite instrucțiunile ADD sau ADDC. Dacă biții 3÷0 ai acumulatorului reprezintă un număr mai mare decât 9 (xxx1010÷xxx1111), sau dacă indicatorul AC este setat, se adună valoarea 6 la acumulator, producând astfel un digit optimal în format BCD. Această adunare internă va seta indicatorul de transport daca s-a propagat o depăşire de la cei mai puțin semnificativi 4 biți spre biții mai semnificativi, dar altfel nu va reseta indicatorul de transport. Dacă indicatorul de transport e setat în urma acestei operații, sau dacă cei 4 biți mai semnificativi reprezintă un număr mai mare decât 9 (1010xxx÷1111xxxx), aceşti biți sunt incrementați cu 6, producând digitul optimal în format BCD. Indicatorul de transport va indica dacă suma celor două variabile BCD originale este mai mare decât 100, permițând sumarea zecimală în precizie multiplă. Indicatorul OV este afectat. Execuția acestei instrucțiuni durează un singur ciclu. Elemetul de noutate este acela că această instrucțiune permite conversia zecimală adăugând 00H, 06H, 60H sau 66H la conținutul acumulatorului.

Notă: Instrucțiunea DA A nu poate converti, pur şi simplu, un număr hexazecimal, conținut în acumulator, în format BCD.

Exemplu: Acumulatorul are valoarea 56H (01010110B) reprezentind digiții în format BCD împachetat ai numărului zecimal 56. Registrul R3 conține valoarea 67H (01100111B), reprezentând digiții în format BCD împachetat ai numarului 67. Indicatorul de transport este setat. Secventa de instrucțiuni:

```
ADDC A,R3
DA   A
```

va face o adunare în complement binar față de doi, având ca rezultat valoarea 24H (00100100B) indicând în format BCD împachetat numărul zecimal 24, cele mai mici două cifre ale sumei zecimale între 56, 67 şi transportul intern. Indicatorul de transport va fi setat, indicând că a apărut o depaşire zecimală. Suma reală între 56, 67 şi 1 este 124. Variabilele BCD pot fi incrementate sau decrementate prin adăugarea lui 01H sau 99H. Dacă inițial acumulatorul conține valoarea 30H, secvența:

```
ADD  A, #99H
DA   A
```

va determina ca acumulatorul să conțină valoarea 99H (30+99=129). Octetul mai puțin semnificativ al sumei poate fi interpretat ca fiind:

30-1=29.

Octeți: 1

Cicluri maşină: 1

Operații: DA

IF $[[(A_{3\div0}) > 9] \vee [(AC)=1]]$

THEN

$(A_{3\div0}) \leftarrow (A_{3\div0}) + 6$

AND

IF $[[(A_{7\div4}) > 9] \vee [(C)=1]]$

THEN

$(A_{7\div4}) \leftarrow (A_{7\div4}) + 6$

3.13 DEC byte

Funcția: Decrementare variabilă de tip octet

Descriere: Variabilade tip octet indicată este decrementată cu 1. O valoare inițială de 00H va determina ca rezultatul să fie 0FFH. Nu afectează nici un indicator. Sunt permise 4 moduri de adresare: acumulator, adresare prin registru, adresare directă, adresare indirectă prin registru.

Notă: Când instrucțiunea este utilizată ca să modifice un port de ieşire, valoarea utilizată ca dată originală va fi citită de la ieşire, şi nu de la intrare.

Exemplu: Registrul R0 conține valoarea 7FH (01111111B). Locațiile interne RAM 7EH şi 7FH conțin valorile 00H şi 40H. Instrucțiunile:

```
DEC  @R0
DEC  R0
DEC  @R0
```

vor lăsa registrul R0 setat la valoarea 7EH şi locațiile interne RAM 7EH şi 7FH conținând 0FFH şi 3FH.

3.14 DIV AB

Funcția: Împărțire

Descrierea: Instrucțiunea DIV AB împarte întregiul fără semn, pe 8 biți, din acumulator la întregul, pe 8 biți, fără semn din registrul B. După execuția instrucțiunii, acumulatorul va conține câtul împărțirii, iar registrul B restul. Indicatorul de transport şi OV vor fi resetate.

Excepție: Dacă registrul B conținea inițial valoarea 00H, valoarea returnată în acumulator şi registrul B vor fi subdefinite şi va fi setat indicatorul de depăşire. Indicatorul de transport va fi resetat în orice caz.

Exemplu: Acumulatorul conține valoarea zecimală 251 (0FBH sau 11111011B) şi registrul B conține valoarea 18 (12H sau 00010010B). Instructiunea:

```
DIV  AB
```

va determina ca acumulatorul să conțină valoarea 13 (0DH sau 00001101B) şi valoarea 17 (11H sau 00010001B) in B. Atât indicatorul de transport, cât şi indicatorul OV vor fi ambii resetați.

3.15 DJNZ <byte>, <rel-addr>

Funcția: Decrementare şi salt dacă rezultatul nu este zero

Descriere: Instrucțiunea DJNZ decrementează octetul indicat şi determină un salt la adresa indicată de al doilea operand dacă valoarea rezultată nu e zero. O valoare inițială 00H va deveni 0FFH. Nu afectează indicatorii de condiții. Destinația saltului va fi obținută prin adăugarea valorii deplasării relative cu semn, în ultimul octet al instrucțiunii, la conținutul PC, după incrementarea acestuia la adresa primul octet al instrucțiunii următoare. Operandul decrementat poate fi un registru sau direct octetul adresat.

Notă: Atunci când această instrucțiune este utilizată ca să modifice un port de ieşire, valoarea utilizată ca dată inițială va fi citită de la ieşire, nu de la pinii de intrare.

Exemplu: Locațiile interne RAM 40H, 50H, 60H conțin respectiv valorile 01H, 70H, 15H. Secvența de instrucțiuni:

```
DJNZ  40H, LABEL_1
```

```
DJNZ  50H, LABEL_2
DJNZ  60H, LABEL_3
```

va cauza un salt la instrucțiunea cu eticheta LABEL_2 cu valorile 00H, 6FH şi 15H în cele 3 locații RAM. Primul salt n-a fost făcut, pentru că rezultatul era 0.

Această instrucțiune asigură o metodă simplă de a executa o buclă de program de un număr de ori, sau adăugarea printr-o singură instrucțiune a unei întârzieiri (între 2 şi 512 cicli maşină). Instructiunea:

```
        MOV   R2, #8
TOGGLE: CPL   P1.7
        DJNZ  R2, TOGGLE
```

va schimba bitul P1.7 de 8 ori, determinând apariția a 4 impulsuri de ieşire la bitul 7 al portului de iesire P1. Fiecare puls inseamnă trei cicluri maşină, doi pentru execuția instrucțiunii DJNZ şi unul pentru modificarea bitului.

3.16 INC <byte>

Funcția: Incrementare octet

Descriere: Instrucțiunea INC incrementează cu 1 variabila octet indicată.O valoare inițială de 0FFH va deveni în urma incrementării 00H.Nu se afectează nici un indicatorde condiții. Sunt permise trei moduri de adresare: adresare prin registru, adresare directă, adresare indirectă prin registru.

Notă: Atunci când această instrucțiune este utilizată ca să modifice un port de ieşire, valoarea utilizată ca dată inițială va fi citită de la ieşire şi nu de la intrare.

Exemplu: Registrul R0 conține valoarea 7EH (01111110B). Locațiile interne RAM cu adresele 7EH şi 7FH conțin valorile 0FFH şi respectiv 40H. Secvența de instrucțiuni:

```
INC   @R0
INC   R0
INC   @R0
```

va lăsa registrul R0 setat la 7FH şi locațiile interne RAM cu adresele 7EH şi 7FH conținând valorile 00H şi 41H.

3.17 INC DPTR

Funcția: Incrementarea pointer-ului de date

Descriere: Instrucțiunea incrementează cu 1 pointerul de date (16 biți). Este utilizată o incrementare pe 16 biți. O depăşire la octetul mai puțin semnificativ al pointerului (DPL), de la valoarea 0FFH la 00H, va incrementa octetul mai semnificativ (DPH). Nu sunt afectați indicatorii de condiții. Singurul registru care poate fi manipulat de această instrucțiune este pointerul de date, DPTR.

Exemplu: Registrele DPH şi DPL conțin valorile 12H şi respectiv 0FEH. Setul de instrucțiuni:

```
INC   DPTR
INC   DPTR
INC   DPTR
```

va schimba conținutul registrelor DPH şi DPL la valorile 13H şi respectiv 01H.

3.18 JB bit, rel

Funcția: Salt dacă bitul e setat la 1

Descriere: Dacă bitul indicat e un 1 se efectuează un salt în program la adresa indicată, altfel se trece la executarea instrucțiunii următoare. Destinația saltului este obținută prin adunarea deplasării relative cu semn, în al treilea octet al instrucțiunii, la conținutul contorului de progam, PC, după incrementarea PC la valoarea primului octet al instrucțiunii următoare. Bitul de test nu este modificat şi nu se afectează nici un indicator.

Exemplu: Data de la portul de intrare P1 e 11001010B. Acumulatorul conține valoarea 56. Secvența de instrucțiuni:

```
JB    P1.2, LABEL1
JB    ACC.2, LABEL2
```

va cauza un salt al execuției programului la eticheta LABEL2.

3.19 JBC bit, rel

Funcția: Salt dacă bitul e setat la 1 şi şterge bitul

Descriere: Dacă bitul indicat este 1, se efectuează un salt la adresa indicată. Altfel, se trece la executarea instrucțiunii următoare. Bitul nu va fi resetat dacă este deja 0. Destinația saltului este obținută prin adunarea deplasării relative cu semn, în al treilea octet al instrucțiunii, la conținutul contorului de progam, PC, după incrementarea PC la valoarea primului octet al instrucțiunii următoare. Nici un indicator de condiții nu este afectat.

Notă: Când instrucțiunea este utilizată ca sa testeze un pin de ieşire, valoarea utilizată ca dată inițială va fi citită de la ieşire, şi nu de la intrare.

Exemplu: Acumulatorul conține valoarea 56H (01010110B). Setul de instrucțiuni:

```
JBC   ACC.3, LABEL 1
JBC   ACC.2, LABEL 2
```

va face ca execuția programului să continue de la instrucțiunea identificată prin eticheta LABEL2, cu acumulatorul modificat la valoarea 52H (01010010B).

3.20 JC rel

Funcția: Salt dacă este setat indicatorul de transport

Descriere: Dacă indicatorul de transport e setat, se efectuează un salt în program, la adresa indicată. Altfel, se trece la executarea instrucțiunii următoare. Destinația saltului este obținută prin adunarea deplasării relative cu semn, în al doilea octet al instrucțiunii, la conținutul contorului de progam, PC, după incrementarea PC de două ori. Nu sunt afectați indicatorii de condiție.

Exemplu: Indicatorul de transport este resetat .Secvența de instrucțiuni:

```
JC    LABEL1
CPL   C
JC    LABEL2
```

va seta indicatorul de transport şi va face ca execuția programului să continue cu instrucțiunea de la eticheta LABEL2.

3.21 JMP @A+DPTR

Funcția: Salt indirect

Descriere: Instrucțiunea adună conținutul pe 8 biți, fără semn, al acumulatorului cu pointerul de date pe 16 biți şi încarcă suma rezultată in contorul programului, PC. Conținutul acestuia va reprezenta adresa instrucțiunii

următoare. Adunarea pe 16 biți se face astfel: transportul de la cei 8 biți mai puțin semnificativi se propagă spre biții mai semnificativi. Nu se afectează indicatorii şi nu se schimbă conținutul acumulatorului şi nici al pointerului de date.

Exemplu: În acumulator se află un număr oarecare, de la 0 la 6.Următoarea secvență de instrucțiuni va efectua un salt la una din cele 4 instrucțiuni AJMP începând de la JMP_TBL.

```
                MOV   DPTR, #JMP_TBL
                JMP   @A+DPTR
                AJMP  LABEL0
JMP_TBL:        AJMP  LABEL1
                AJMP  LABEL2
                AJMP  LABEL3
```

Dacă acumulatorul conține 04H când începe această secvență, execuția va sări la eticheta LABEL2. Amintim că AJMP e o instrucțiune pe doi octeți, deci instrucțiunea de salt va incepe la orice altă adresă.

3.22 JNB bit, rel

Funcția: Salt dacă bitul nu e setat la 1

Descriere: Dacă bitul indicat este 0, se efectuează un salt la adresa indicată. Altfel, se trece la executarea instrucțiunii următoare. Destinația saltului este obținută prin adunarea deplasării relative cu semn, în al treilea octet al instrucțiunii, la conținutul contorului de progam, PC, după incrementarea PC la valoarea primului octet al instrucțiunii următoare. Bitul testat nu este modificat. Nu este afectat nici un indicator de condiții.

Exemplu: Data prezentă la portul P1 de intrare este 11001010B. Acumulatorul conține valoarea 56H (01010110B). Secvența de instrucțiuni:

```
JNB   P1.3, LABEL1
JNB   ACC.3, LABEL2
```

va face ca execuția programului să continue de la instrucțiunea cu eticheta LABEL2.

3.23 JNC rel

Funcția: Salt dacă indicatorul de transport nu este setat

Descriere: Dacă indicatorul de transport este 0, se sare instrucțiunea de la adresa indicată. Altfel, se trece la executatea instrucțiunii următoare. Destinația saltului este obținută prin adunarea deplasării relative cu semn, în al doilea octet al instrucțiunii, la conținutul contorului de progam, PC, după incrementarea PC de două ori, pentru a se ajunge la adresa primului octet al instrucțiunii următoare. Indicatorul de transport nu e modificat.

Exemplu: Indicatorul de transport este setat. Secvența de instrucțiuni:

```
JNC   LABEL1
CPL   C
JNC   LABEL2
```

va reseta indicatorul de transport şi va face ca execuția programului să continue de la instrucțiunea specificată de eticheta LABEL2.

3.24 JNZ rel

Funcția: Salt dacă conținutul acumulatorului nu este 0

Descriere: Dacă unul dintre biții acumulatorului este 1, se efectuează un salt la adresa indicată. Altfel, se continuă cu executarea instrucțiunii urmă-

toare. Destinația saltului este obținută prin adunarea deplasării relative cu semn, în al doilea octet al instrucțiunii, la conținutul contorului de progam, PC, după incrementarea PC de două ori .Acumulatorul nu se modifică şi nici un indicator de condiții nu este afectat.

Exemplu: Acumulatorul are valoarea 00H. Instrucțiunile:

```
JNZ   LABEL1
INC   A
JNZ   LABEL2
```

vor seta acumulatorul la valoarea 01H şi vor determina continuarea programului de la instrucțiunea cu eticheta LABEL2.

3.25 JZ rel

Funcția: Salt dacă conținutul acumulatorului este zero

Descriere: Dacă toți biții din acumulator sunt zero, se execută un salt la adresa indicată. Altfel, se continuă cu execuția instrucțiunii următoare. Destinația saltului este obținută prin adunarea deplasării relative cu semn, în al doilea octet al instrucțiunii, la conținutul contorului de progam, PC, după incrementarea PC de două ori. Acumulatorul nu se modifică şi nici un indicator de condiții nu este afectat.

Exemplu: Acumulatorul conține inițial valoarea 01H. Secvența de instrucțiuni:

```
JZ    LABEL1
DEC   A
JZ    LABEL2
```

va determina ca acumulatorul să conțină valoarea 00H şi va face ca execuția programului să continue cu instrucțiunea de la LABEL2.

3.26 LCALL addr16

Funcția: Long Call

Descriere: Instrucțiunea LCALL va apela o subrutină aflată la adresa indicată. Instrucțiunea incrementează cu trei contorul programului pentru a genera adresa următoarei instrucțiuni, şi salvează rezultatul (pe 16 biti) în stivă (mai întâi octetul mai puțin semnificativ). Se incrementează pointerul stivei cu 2. Octeții, mai puțin semnificativ şi mai semnificativ, ai contorului programului, PC, sunt încărcați cu al doilea şi al treilea octet al instrucțiunii LCALL. Execuția programului va continua cu instrucțiunea de la această adresă. Subrutina poate începe oriunde în cei 64 kocteți ai spatiului de memorie de program. Nu sunt afectați indicatorii de condiție.

Exemplu: Inițial, pointerul stivei are valoarea 07H. Eticheta SUBRTN este asociată locației de memorie de program cu adresa 1234H. După execuția instrucțiunii:

```
LCALL     SUBRTN
```

la locația de memorie cu adresa 0123H, pointerul stivei va conține 09H, locațiile interne RAM cu adresele 08H şi 09H vor conține 26H şi 01H, iar PC va conține 1235H.

3.27 LJMP addr16

Funcția: Long Jump

Descriere: Instrucțiunea LJMP produce un salt necondiționat la adresa indicată, prin încărcarea octeților mai puțin semnificativ şi mai semnificativ ai contorului programului, PC, cu al doilea şi al treilea octet al instrucțiunii. Destinația poate fi oriunde în spațiul de adresare al memoriei

program de 64 kocteți. Nu se afectează indicatorii de condiții.

Exemplu: Eticheta JMPADR e asociată instrucțiunii localizate la adresa 1234H în memoria de program. Instrucțiunea:

```
LJMP  JMPADR
```

va încărca contorul programului cu valoarea 1234H.

3.28 MOV<dest-byte>, <scr-byte>

Funcția: Mută variabila sursă, de tip octet, în variabila destinație, de tip octet

Descriere: Variabila octet indicată prin al doilea operand este copiată în locația specificată de primul operand. Octetul sursă nu este afectat. Nu se afectează nici un registru sau indicator de condiții. Este de departe cea mai flexibilă operațiune. Permite 14 combinații de moduri de adresare ale sursei şi destinatiei.

Exemplu: Locația internă RAM cu adresa 30H conține valoarea 40H. Locația internă RAM cu adresa 40H conține valoarea 10H. Data de la portul de intrare P1 este 11001010B (0CAH). Instrucțiunile:

```
MOV  R0, #30H
MOV  A, @R0
MOV  R1, A
MOV  B, @R1
MOV  @R1, P1
MOV  P2, P1
```

lasă valoarea 30H în registrul R0, 40H în acumulator şi registrul P1, 10H în registrul B şi 0CAH (11001010B) în locația RAM cu adresa 40H şi în portul de ieşire P2.

3.29 MOV<dest-bit>, <scr-bit>

Funcția: Mută data de tip bit de la sursă la destinație

Descriere: Variabila booleană indicată prin al doilea operand este copiată la locația specificată de primul operand. Unul dintre operanzi trebuie să fie indicatorul de transport, celălalt poate fi orice bit adresabil direct. Nici un alt registru sau indicator nu este afectat.

Exemplu: Indicatorul de transport este inițial setat. Data prezentă la portul de intrare P3 este 11000101B. Data înscrisă anterior în portul de ieşire P1 este 35H (00110101B). Instrucțiunile:

```
MOV  P1.3, C
MOV  C, P3.3
MOV  P1.2, C
```

vor lăsa indicatorul de transport resetat şi portul P1 la valoarea 39H (00111001B).

3.30 MOV DPTR,#data16

Funcția: Încarcă pointerul de date cu o constantă pe 16 biți

Descriere: Pointerul de date este încărcat cu constanta pe 16 biți indicată. Aceasta se încarcă în al doilea şi al treilea octet al instrucțiunii. Al doilea octet (DPH) este octetul mai semnificativ, iar al treilea octet (DPL) conține octetul mai puțin semnificativ al constantei specificate. Nu se afectează indicatorii. Este singura instrucțiune de tranfer pe 16 biți.

Exemplu: Instrucțiunea:

```
MOV  DPTR, #1234H
```

va încărca valoarea 1234H în pointerul de date. DPH va conține 12H şi DPL va conține 34H.

3.31 MOVC A,@A+<base-reg>

Funcția: Mută octetul de cod

Descriere: Instrucțiunea MOVC încarcă acumulatorul cu un octet de cod sau o constantă din memoria-program. Adresa octetului reprezintă suma conținutului, pe 8 biți, fără semn, al acumulatorului şi conținutul, pe 16 biți al registrului de bază, care poate fi pointerul de date sau contorul programului. In ultimul caz, PC este incrementat la adresa urmatoarei instrucțiuni dinaintea sumării cu acumulatorul. Altfel, registrul bază nu e modificat. Adunarea pe 16 biti se face astfel încât un transport de la cei 8 biți mai puțin semnificativi să poată fi propagat la ceilalți. Nu sunt afectați indicatorii de condiții.

Exemplu: In acumulator se găseşte o valoare cuprinsă între 0 şi 3. Următoarele instrucțiuni vor translata valoarea din acumulator înspre una din cele 4 valori definite la directiva DB (define byte):

```
REL_PC:  INC   A
         MOVC  A, @A+PC
         RET
         DB    66H
         DB    77H
         DB    88H
         DB    99H
```

Dacă subrutina este apelată cu acumulatorul având valoarea 01H, va returna 77 în acumulator. Instrucțiunea INC A plasată înaintea instrucțiunii MOVC a permis "ocolirea" instrucțiunii RET din secvență. Dacă câțiva octeți de cod separă începutul de instrucțiunea MOVC din secvență, numărul corespunzator va fi adăugat la acumulator.

3.32 MOVX <dest-byte>,<scr-byte>

Funcția: Mutare externă

Descriere: Instrucțiunea MOVX transferă date între acumulator şi un octet al memoriei externe. Sunt două tipuri de instrucțiuni diferite, după cum se furnizează o adresare indirectă la RAM-ul extern, pe 8 sau pe 16 biți. In primul caz, conținutul registrelor R0 şi R1 furnizează o adresă pe 8 biți multiplexată cu data din portul P0. 8 biți sunt suficienți pentru decodarea extensiei I/O externe pe o arie de RAM mică. Pentru o arie mai mare, pinii porturilor pot fi utilizati pentru ieşirea biților mai semnificativi de adresă. Aceşti biți vor fi controlați de oinstrucțiune de ieşire care va urma instrucțiunii MOVX. In al doilea caz, pointerul de date generează o adresă pe 16 biți. Portul P2 va marca ieşirea celor 8 biți superiori de adresă (conținutul DPH) iar portul P0 va multiplexa cei 8 biți inferiori (DPL) cu cei de date. Registrul funcțiilor speciale P2 va reține conținutul anterior, iar bufferul de ieşire P2 va emite conținutul lui DPH. Această formulă e mai rapidă şi mai eficientă când se accesează zone de date foarte mari (mai mari de 64 kocteți), deoarece nu este nevoie de instrucțiuni suplimentare pentru a seta porturile de ieşire. Este posibilă şi combinația celor două tipuri de instrucțiuni MOVX. O zonă extinsă de memorie RAM şi liniile sale de adresă de ordin superior, administrate de portul P2, pot fi adresate prin pointerul de date, iar codul de ieşire al biților superiori de adresă ai portului P2 va fi urmat de o instrucțiune MOVX utilizând registrele R0 sau R1.

Exemplu: O zonă de memorie RAM externă, cu lungimea de 256 octeți, utilizând multiplexarea liniilor de adrese şi de date este conectată la portul P0 al

microcontrollerului 8051. Portul P3 asigură liniile de control pentru memoria RAM externă. Porturile P1 şi P2 sunt utilizate pentru intrările şi ieşirile normale. Registrele R0 şi R1 conţin valorile 12H şi respectiv 34H. Locaţia cu adresa 34H a RAM extern conţine valoarea 56H. Instrucţiunile:

```
MOVX    A,@R1
MOVX    @R0,A
```

copiază valoarea 56H atât în acumultor, cât şi in locaţia RAM externă cu adresa 12H.

3.33 MUL AB

Funcţia: Înmulţire

Descriere: Instrucţiunea MUL AB înmulţeşte întregii pe 8 biţi din acumulator şi din registrul B. Octetul mai puţin semnificativ al produsului pe 16 biţi este lăsat în acumulator, iar octetul mai semnificativ în registrul B. Dacă produsul e mai mare decâ 255 (0FFH) indicatorul de depăşire este setat; altfel, este zero. Indicatorul de transport este întotdeauna resetat.

Exemplu: Iniţial, acumulatorul conţine valoarea 80 (50H), iar registrul B, valoarea 160 (0A0H). Instrucţiunea

```
MUL   AB
```

va furniza produsul, 12.800 (3200H), registrul B devine 32H (00110010B) iar acumulatorul 00H. Indicatorul de depăşire este setat, iar cel de transport este zero.

3.34 NOP

Funcţia: Nici o operaţie

Descriere: Execuţia continuă cu instrucţiunea următoare. In afara contorului de program, PC, care este incrementat, nici un registru sau indicator nu este afectat.

Exemplu: Este nevoie de producerea unui puls de depăşire scurt pe bitul 7 al portului P2, durând exact 5 cicluri. O secvenţă SETB/CLR va genera un puls cu durata de un ciclu, la care trebuie adunati alţi patru, ca în secvenţa:

```
CLR   P2.7
NOP
NOP
NOP
NOP
SETB  P2.7
```

3.35 ORL <dest-byte>, <scr-byte>

Funcţia: SAU LOGIC pentru variabile de tip octet

Descriere: Instrucţiunea ORL realizează funcţia SAU LOGIC între variabilele de tip octet indicate, încărcând rezultatul în octetul destinaţie. Nu se afectează nici un indicator de condiţii. Cei doi operanzi permit 6 combinaţii ale modurilor de adresare. Când destinaţia este acumulatorul, sursa poate fi adresată ca registru, direct, registru-indirect sau imediat; când destinaţia este o adresă directă, sursa poate fi acumulatorul sau o data imediată.

Notă: Atunci când această instrucţiune este utilizată ca să modifice un port de ieşire, valoarea utilizată ca dată iniţială va fi citită de la ieşire şi nu de la intrare.

Exemplu: Dacă acumulatorul conține valaorea 0C3H (11000011B) şi registrul R0 valoarea 55H (01010101B), instrucțiunea

```
ORL  A,R0
```

va lăsa în acumulator valoarea 0D7H(11010111B).

Când destinația este un octet adresat direct, instrucțiunea poate seta combinații de biți în orice locație RAM sau registru hardware. Şablonul biților care trebuie setați este determinat de un octet-mască, care poate fi o constantă în instrucțiune sau o variabilă creată în acumulator la rulare. Instrucțiunea:

```
ORL  P1, #00110010B
```

va seta biții 5, 4 şi 1 ai portului de ieşire P1.

3.36 ORL C, <scr-bit>

Funcția: SAU LOGIC pentru variabile de tip bit

Descriere: Instrucțiunea ORL C, bit setează indicatorul de transport dacă valoarea booleană e un 1 logic. Instrucțiunea nu modifică starea indicatorului de transport. Simbolul ("/") precedând operandul, în limbaj de asamblare, indică faptul că se utilizează compementul logic al bitului adresat ca valoare sursă, fără ca bitul sursă să fie afectat. Nici un alt indicator nu e afectat.

Exemplu: Secvența prezentată setează indicatorul de transport dacă şi numai dacă P1.0=1, ACC.7=1, OV=0:

```
ORL  C, P1.0
ORL  C, ACC.7
ORL  C, /OV
```

3.37 POP direct

Funcția: Extragere din stivă

Descriere: Instrucțiunea citeşte conținutul locatiei interne RAM adresată prin pointerul stivei, iar pointerul stivei este decrementat cu 1. Valoarea citită este apoi transferată în octetul direct adresabil indicat. Nu se afectează nici un indicator de condiții.

Exemplu: Pointerul stivei conține inițial valoarea 32H şi locațiile interne RAM cu adresele 30H până la 32H conțin valorile 20H, 23H, 01H. Instrucțiunile:

```
POP  DPH
POP  DPL
```

lasă pointerul stivei setat la valoarea 30H şi pointerul de data la 0123H. In acest punct, instrucțiunea:

```
POP  SP
```

va seta pointerul stivei la valoarea 20H. In acest caz special, pointerul stivei a fost decrementat la valoarea 2FH înainte de încărcarea cu valoarea extrasă (20H).

3.38 PUSH direct

Funcția: Salvare în stivă

Descriere: Pointerul stivei, SP, este incrementat cu 1. Conținutul variabilei indicate e copiat în locația internă RAM adresată de pointerul stivei. Nici un alt indicator nu este afectat.

Exemplu: Intrând într-o rutină de întrerupere, pointerul stivei, SP, conține valoarea 09H. Pointerul de date conține valoarea 0123H. Instrucțiunile:

```
PUSH DPL
PUSH DPH
```

vor seta pointerul stivei la valoarea 0BH şi vor încărca 23H şi 01H în locaţiile interne RAM cu adresele 0AH şi 0BH.

3.39 **RET**

Funcţia: Revenire din subrutină

Descriere: Instrucţiunea RET extrage succesiv octeţii mai semnificativ şi mai puţin semnificativ ai adresei din stivă, decrementind pointerul stivei cu 2. Execuţia programului continuă de la adresa rezultată, în general instrucţiunea ce sccede instrucţiunile ACALL sau LCALL. Nici un indicator nu este afectat.

Exemplu: Pointerul stivei conţine iniţial valoarea 0BH. Locaţiile interne RAM cu adresele 0AH şi 0BH conţin valorile 23H şi, respectiv 01H. Instrucţiunea RET va seta pointerul stivei la valoarea 09H, iar execuţia programului va continua de la locaţia cu adresa 0123H.

3.40 **RETI**

Funcţia: Revenire din rutina de tratare a unei întreruperi

Descriere: Instrucţiunea RETI extrage succesiv octeţii mai semnificativ şi mai puţin semnificativ ai contorului programului, PC, din stivă şi reactivează logica de întreruperi să accepte întreruperi suplimentare, cu acelaşi nivel de prioritate ca cea tocmai procesată. Pointerul stivei e decrementat prin 2. Nici un alt registru nu este afectat. Cuvântul de stare a programului, PSW, nu este reîncărcat automat cu starea sa anterioară tratării întreruperii. Execuţia programului continuă de la adresa rezultată, care e în general, instrucţiunea imediat următoare celei după care a fost detectată apelarea întreruperii. Dacă o întrerupere cu nivel de prioritate mai mic sau egal a apărut în timp ce se executa RETI, acea instrucţiune va fi executată înaintea revenirii din întrerupere.

Exemplu: Pointerul stivei conţine valoarea initială 0BH. O întrerupere a fost detectată în timpul instrucţiunii care se termină la locaţia cu adresa 0122H. Locaţiile interne RAM cu adresele 0AH şi 0BH conţin valorile 23H şi respectiv 01H. Instrucţiunea:

```
RETI
```

va lăsa pointerul stivei setat la valoarea 09H şi va continua execuţia programului de la adresa 0123H.

3.41 **RL A**

Funcţia: Roteşte acumulatorul la stânga

Descriere: Cei 8 biţi ai acumulatorului sunt rotiţ cu un bit la stânga. Bitul 7 ajunge în poziţia bitului 0. Nici un indicator nu e afectat.

Exemplu: Acumulatorul conţine valoarea iniţială 0C5H (11000101B). Instrucţiunea:

```
RL A
```

va determina ca acumulatorul să conţină valoarea 8BH (10001011B), fără a afecta indicatorul de transport.

3.42 **RLC A**

Funcţia: Rotirea acumulatorului la stânga prin indicatorul de transport

Descriere: Cei 8 biţi ai acumulatorului şi indicatorul de transport sunt amândoi rotiţi cu un bit la stânga. Bitul 7 ajunge în indicatorul de transport, iar acesta pe poziţia bitului 0. Nu afecteaza nici un alt indicator.

Exemplu: Acumulatorul conține valoarea inițială 0C5H (11000101B), şi indicatorul de transport este 0. Instrucțiunea:

```
RLC A
```

va lăsa în acumulator valoarea 8BH (10001010B), cu setarea indicatorului de transport.

3.43 RR A

Funcție: Rotirea acumulatorului la dreapta

Descriere: Cei 8 biți ai acumulatorului sunt rotiți cu un bit la dreapta. Bitul 0 ajunge în poziția bitului 7. Nici un indicator nu e afectat.

Exemplu: Acumulatorul conține valoarea 0C5H (11000101B). Instrucțiunea:

```
RR A
```

va seta acumulatorul la valoarea 0E2H (11100010B), cu indicatorul de transport neafectat.

3.44 RRC A

Funcția: Rotirea acumulatorului la dreapta prin indicatorul de transport.

Descriere: Cei 8 biți ai acumulatorului şi indicatorul de transport sunt rotiți împreună cu un bit la dreapta. Bitul 0 ajunge în indicatorul de transport, iar acesta în poziția bitului 7. Nici un alt indicator nu este afectat.

Exemplu: Acumulatorul conține valoarea 0C5H, iar transportul este 0. Instrucțiunea:

```
RRC A
```

va determina ca acumulatorul să conțină valoarea 62H (01100010B), cu indicatorul de transport setat la 1.

3.45 SETB <bit>

Funcția: Setează bitul specificat

Descriere: Instrucțiunea SETB setează bitul indicat la 1. SETB poate opera asupra indicatorului de transport sau oricărui alt bit direct adresabil. Nu afectează alți indicatori.

Exemplu: Indicatorul de transport este resetat. Portul de ieşire P1 a fost înscris cu valoarea 34H (00110100B). Instrucțiunile:

```
SETB C
SETB P1.0
```

vor seta indicatorul de transport la 1 şi data de ieşire la portul P1 va fi 35H (00110101B).

3.46 SJMP rel

Funcția: Short Jump

Descriere: Instrucțiunea SJMP determină un salt necondiționat în program, la adresa indicată. Destinația saltului este compusă prin sumarea deplasării cu semn, în al doilea octet al instrucțiunii, cu conținutul contorului programului, PC, după incrementarea PC de două ori. Saltul permis are o valoare de la -128 de octeți (precedenți instrucțiunii SJMP) la +127 octeți (următori instrucțiunii SJMP).

Exemplu: Eticheta RELADR e asociată instrucțiunii de la locația cu adresa 0123H. Instrucțiunea:

```
SJMP RELADR
```

va duce la locația cu adresa 0100H. După ce instrucțiunea este executată, PC va conține valoarea 0123H.

3.47 SUBB A, <src-byte>

Funcția: Scădere cu împrumut.

Descriere: Instrucțiunea SUBB va scădea atât variabila octet indicată, cât şi indicatorul de transport din acumulator, lăsând rezultatul în acumulator. SUBB setează indicatorul de transport (de împrumut) dacă este nevoie de un împrumut pentru bitul 7 şi sterge indicatorul C altfel (dacă C a fost setat înaintea execuției instrucțiunii SUBB, aceasta indică că a fost necesar un împrumut la pasul anterior, într-o scădere cu precizie multiplă, astfel încât transportul este scăzut din acumulator odată cu operandul sursă). Indicatorul AC e setat dacă a fost necesar un împrumut pentru bitul 3 şi resetat în caz contrar. Indicatorul OV e setat dacă e necesar un împrumut la bitul 6, dar nu la bitul 7, sau la bitul 7, dar nu la bitul6. La scăderea întregilor cu semn, OV indică un număr negativ când o valoare negativă e scazută dintr-o valoare pozitivă, sau un rezultat pozitiv când un număr pozitiv e scăzut dintr-un număr negativ. Operandul sursă permite 4 moduri de adresare: prin registru, directă, indirectă prin registru, imediată.

Exemplu: Acumulatorul conține valoarea 0C9H (11001001B), registrul R2 conține valoarea 54H (01010100B) şi indicatorul de transport e setat. Instrucțiunea:

```
SUBB A, R2
```

va determina valoarea 74H (01110100B) în Acumulator, cu indicatorul de transport şi AC resetați, dar indicatorul OV setat. Se observă că:

```
0C9H-54H =75H
```

Diferența între acest rezultat şi cel de mai sus este datorat faptului că indicatorul de transport (împrumut) a fost setat înaintea operației. Dacă starea indicatorului de transport nu e cunoscută înainte de începerea unei scăderi în precizie simplă sau multiplă, va fi în mod explicit resetat de o instrucțiune:

```
CLR C.
```

3.48 SWAP A

Funcție: Interschimb intern în acumulator

Descriere: Instrucțiunea SWAP A interschimbă câmpurile de câte 4 biți, mai semnificativ şi mai puțin semnificativ, ale acumulatorului (biții 7÷4 şi biții 3÷0). Operația poate, de asemenea, să fie gândită ca o instrucțiune de rotație pe 4 biți. Nici un indicator nu este afectat.

Exemplu: Acumulatorul conține valoarea 0C5H (11000101B). Instrucțiunea:

```
SWAP A
```

determină în acumulator valoarea 5CH (01011100B).

3.49 XCH A, <byte>

Funcție: Schimbă conținutul acumulatorului cu o variabilă de tip octet

Descriere: Instrucțiunea XCH încarcă acumulatorul cu conținutul variabilei indicate, scriind în acelaşi timp conținutul original al acumulatorului în variabila de tip octet indicată. Operatorii sursă şi destinație pot folosi adresarea prin registru, adresarea directă sau indirect prin registru.

Exemplu: Registrul R0 conține adresa 20H. Acumulatorul conține valoarea 3FH (00111111B). Locația internă RAM cu adresa 20H conține valoarea 75H (01110101B). Instrucțiunea:

```
XCH A,@R0
```

va determina ca locația RAM cu adresa 20H să conțină valoarea 3FH (00111111B) şi 75H (01110101B) în acumulator.

3.50 XCHD A,@Ri

Funcția: Schimbă digit

Descriere: Instrucțiunea XCHD schimbă câmpul de 4 biți mai puțin semnificativ al acumulatorului (biții 3÷0), reprezentând în general un digit hexazecimal sau BCD, cu acela al locației interne RAM adresată indirect prin registrul specificat. Câmpul de 4 biți mai semnificativ (biții 7÷4) ai registrelor nu sunt afectați. Nici un indicator de condiții nu este afectat.

Exemplu: Registrul R0 conține adresa 20H. Acumulatorul conține valoarea 36H (00110110B). Locația internă RAM cu adresa 20H conține valoarea 75H (01110101B). Instrucțiunea:

```
XCHD A, @R0
```

va determina ca locația RAM cu adresa 20H să conțină valoarea 76H (01110110B) şi acumulatorul 35H (00110101B).

3.51 XRL <dest-byte>, <src-byte>

Funcție: SAU EXCLUSIV între variabile de tip octet

Descriere: Instrucțiunea XRL realizează funcția SAU EXCLUSIV la nivel de bit între variabilele octet indicate, incărcând rezultatul în octetul destinație. Nu afectează indicatorii de condiții. Cei doi operanzi permit 6 combinații de moduri de adresare. Când destinația este acumulatorul, sursa poate fi adresată ca registru, direct, registru-indirect sau imediat; când destinația este o adresă directă, sursa poate fi acumulatorul sau o dată imediată.

Notă: Atunci când această instrucțiune e utilizată ca să modifice un port de ieşire, valoarea utilizată ca dată inițială va fi citită de la ieşire, nu de la intrare.

Exemplu: Acumulatorul conține valoarea 0C3H (11000011B) şi registrul R0 conține valoarea 0AAH (10101010B). Instructiunea:

```
XRL   A, R0
```

va determina ca acumulatorul să conțină valoarea 69H (01101001B).

Când destinația e un octet direct adresabil, această instrucțiune poate încărca complementele logice ale combinațiilor de biți în orice locație RAM sau registru hardware. Şablonul biților care vor fi complementați e determinat de un octet mască, ce poate fi atât o constantă conținută în instrucțiune, cât şi o variabilă obținută în acumulator în timpul rulării programului. Instrucțiunea:

```
XRL   P1, #00110001B
```

va complementa biții 5, 4 şi 0 ai portului de ieşire P1.