4. UKLADY WEWNETRZNE MIKROKONTROLERA SAB 80C535

4.1. Porty wejscia-wyjscia

4.1.1. Opis portów wejscia-wyjscia

Porty (bramy) sluza do komunikacji procesora z otoczeniem. Umozliwiaja dolaczenie do niego klawiatury, wyswietlacza, lub innych urzadzen sterujacych i wykonawczych (np. przekazników).

Mikroprocesor **SAB 80C535** wyposazony jest w szesc 8–bitowych portów wejscia–wyjscia (**P0**...**P5**), 8–bitowy port wejsc cyfrowych i analogowych **P6**, oraz port szeregowy do dwustronnej komunikacji.

∗ **Porty P0...P5** sa portami wejscia–wyjscia ogólnego przeznaczenia, umozliwiajacymi wykorzystanie 48 linii wejscia–wyjscia, przy czym porty **P0, P1, P2** i **P3** realizuja tez pewne funkcje alternatywne w zaleznosci od konfiguracji pracy mikrokontrolera. Wykaz funkcji zamieszczono w tabeli 4.1.

Szczególna role w pracy mikrokontrolera **SAB 80C535** pelnia porty **P0** i **P2**. Poniewaz mikrokontroler nie jest wyposazony w wewnetrzna pamiec programu *ROM*, do portów **P0** i **P2** dolaczona jest zewnetrzna pamiec *RAM* o maksymalnej pojemnosci 64 kB, sluzaca jako pamiec programu i pamiec danych (konfiguracja wg von Neumanna). W przypadku adresowania pamieci 8–bitowym rejestrem wskaznikowym **R0** lub **R1** port **P0** sluzy do przesylania 8–bitowego adresu komórki pamieci i bitów danych. W przypadku adresowania 16–bitowym rejestrem wskaznikowym **DPTR** 8 mniej znaczacych bitów adresu i bity danych przesylane sa przez port **P0**, natomiast port **P2** sluzy do przeslania 8 bardziej znaczacych bitów adresu. Taka konfiguracja portów **P0** i **P2** powoduje, ze nie moga byc one wykorzystane jako standardowe porty wejscia–wyjscia.

∗ **Port P6**

Port **P6** jest portem jednokierunkowym. Linie portu moga sluzyc jedynie jako wejscia cyfrowe, a w przypadku pomiaru przetwornikiem a/c wejscia te sa równowazne

kanalom analogowym **AN0**...**AN7**.

Tabela 4.1. Wykaz funkcji alternatywnych portów wejscia−wyjscia

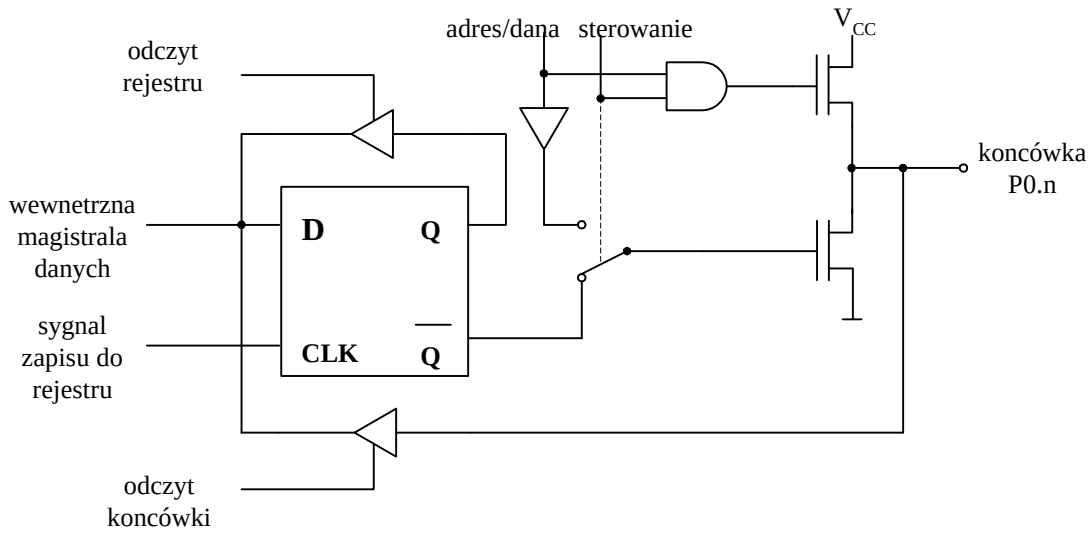
Port	Symbol	Funkcja
P1.0	INT3#/CC0	Wejscie 3 zewnetrznego przerwania, wyjscie porównania 0, wejscie zapamietania 0
P1.1	INT4/CC1	Wejscie 4 zewnetrznego przerwania, wyjscie porównania 1, wejscie zapamietania 1
P1.2	INT5/CC2	Wejscie 5 zewnetrznego przerwania, wyjscie porównania 2, wejscie zapamietania 2
P1.3	INT6/CC3	Wejscie 6 zewnetrznego przerwania, wyjscie porównania 3, wejscie zapamietania 3
P1.4	INT2#	Wejscie 2 zewnetrznego przerwania
P1.5	T2EX	Wejscie zewnetrznego wyzwolenia dla autoladowania licznika T2
P1.6	CLKOUT	Wyjscie zegara systemowego
P1.7	T2	Wejscie zewnetrznego wyzwolenia dla autoladowania licznika T2
P3.0	RXD	Sterowanie wejsciem odbiornika portu szeregowego (praca synchroniczna), lub wejsciem/wyjsciem (praca asynchroniczna)
P3.1	TXD	Sterowanie przekaznikiem portu szeregowego: wyjscie danych (asynchroniczne) lub wyjscie zegara (synchroniczne)
P3.2	INT0#	Wejscie 0 zewnetrznego przerwania, bramka kontrolna zegara 0
P3.3	INT1#	Wejscie 1 zewnetrznego przerwania, bramka kontrolna zegara 1
P3.4	T0	Wejscie zewnetrzne licznika/zegara 0
P3.5	T1	Wejscie zewnetrzne licznika/zegara 1
P3.6	WR#	Zapis danych pamieci zewnetrznej
P3.7	RD#	Odczyt danych pamieci zewnetrznej
P0		W przypadku wspólpracy z zewnetrzna pamiecia programu i danych 8 mniej znaczacych bitów adresu i danych
P2		W przypadku wspólpracy z zewnetrzna pamiecia programu i danych 8 bardziej znaczacych bitów adresu

Uwaga: w tabeli 4.1 znak # oznacza negacje, np. RD# $\equiv \overline{\text{RD}}$.

Ze wzgledu na rózne przeznaczenie, porty mikrokontrolera **SAB 80C535** maja rózna budowe. Schematy portów przedstawiono na rysunkach 4.1...4.4.

Porty **P0** i **P2** posiadaja w swojej strukturze przelacznik, który w przypadku,

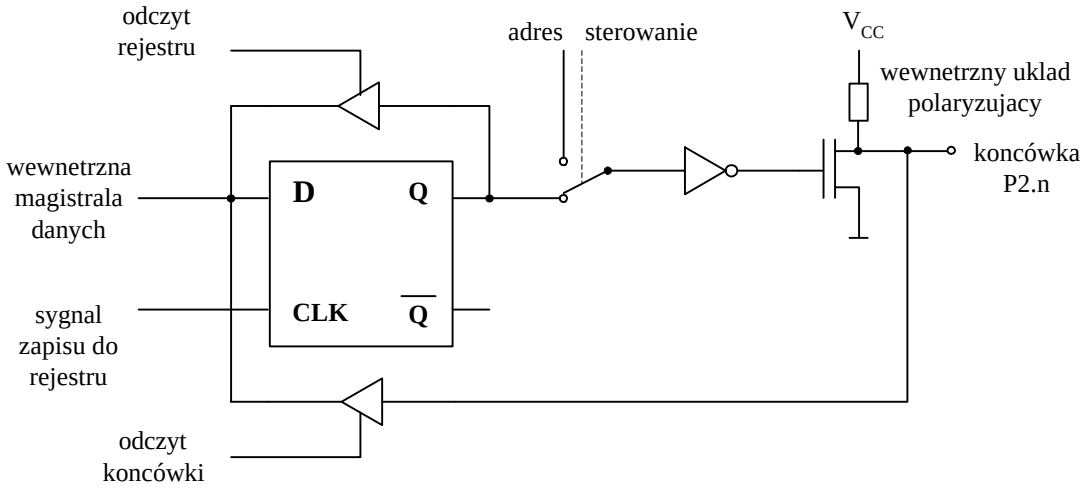
gdy mikrokontroler komunikuje sie z pamiecia zewnetrzna odlacza tranzystor wyjsciowy od rejestru portu. Wówczas zawartosc rejestru nie ma wplywu na sygnal wyjsciowy portu. W porcie **P0** tranzystor wyjsciowy podlaczony do zródla zasilania pozwala na uzyskanie wiekszego pradu wyjsciowego do sterowania wejsc zewnetrznych pamieci. W przypadku gdy port **P0** pracuje jako normalny port wejscia–wyjscia, tranzystor ten pracuje jako zródlo pradowe. Poniewaz port **P0** nie ma ukladu polaryzujacego, w przypadku gdy linie portu wykorzystywane sa jako wyjscia, konieczne jest dolaczenie do tych linii zewnetrznego rezystora polaryzujacego o wartosci ok. 10 kΩ. Gdy do rejestru portu wpisana jest jedynka, wówczas tranzystory wyjsciowe sa zatkane i w tych warunkach linie portu pracuja jako wejscia o duzej impedancji. Schemat portu **P0** przedstawiono na rys. 4.1.



Rys. 4.1. Schemat portu P0

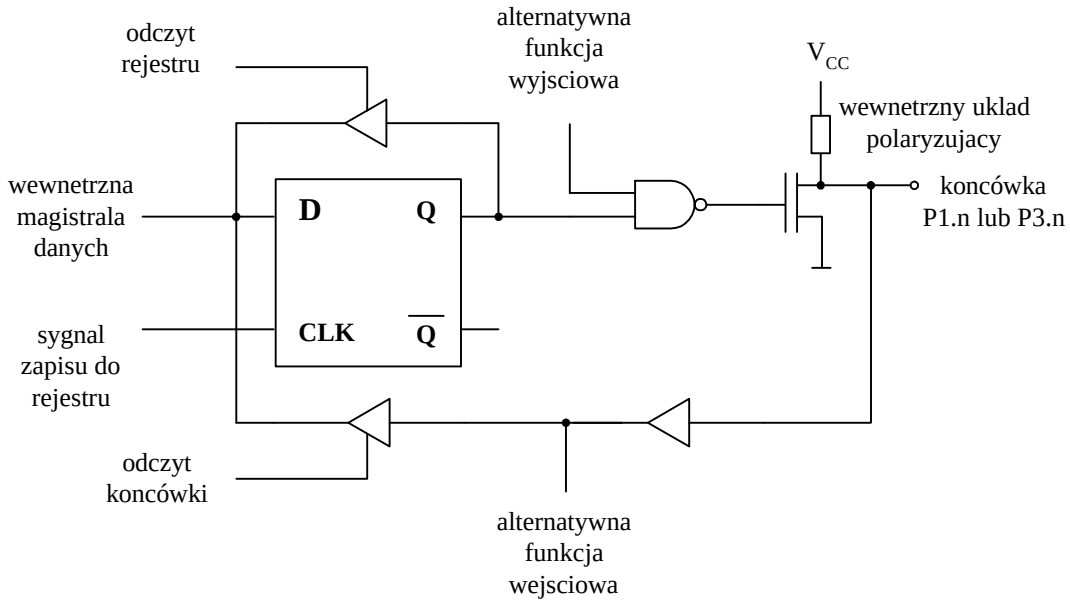
Budowa portu **P2** jest nieco prostsza niz portu **P0**, gdyz przez port **P2** moze byc dodatkowo wysylany tylko bardziej znaczacy bajt adresu. Obciazeniem tranzystora wyjsciowego, podobnie jak w pozostalych portach jest zródlo pradowe. Nalezy przy tym zaznaczyc, ze wewnetrzny uklad polaryzujacy, w który wyposazone sa wszystkie porty mikrokontrolera za wyjatkiem portu **P0** nie jest liniowym rezystorem, lecz specjalnym ukladem zbudowanym z tranzystorów polowych. Schemat portu **P2** przedsta-

wiono na rys. 4.2.



Rys. 4.2. Schemat portu P2

Schemat budowy portów **P1** i **P3**, które wykonuja tez pewne alternatywne funkcje (patrz tabela 4.1) przedstawiono na rys. 4.3.

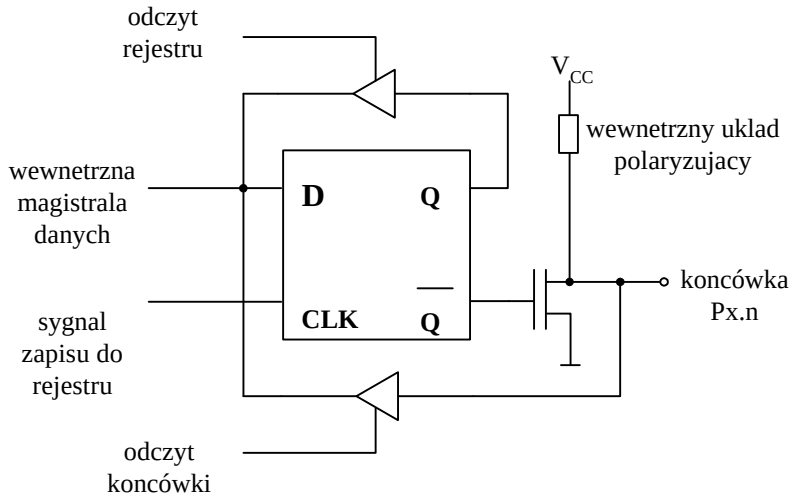


Rys. 4.3. Schemat portów P1 i P3

Wpisanie jedynki do portu powoduje zatkanie tranzystora wyjsciowego i na koncówce ukladu (dzieki wewnetrznemu ukladowi polaryzujacemu) pojawia sie stan jedynki logicznej. Wpisanie zera logicznego do rejestru portu powoduje otwarcie tran-

zystora wyjsciowego, który zwierajac linie portu do masy wymusza stan niski na koncówce ukladu. Jezeli linia portu ma pracowac jako wejscie, wówczas do rejestru portu nalezy wpisac jedynke logiczna. Wtedy stan na koncówce ukladu moze zostac wymuszony zewnetrznie. Aby funkcje alternatywne portu mogly byc aktywne, do rejestru danego portu nalezy wpisac jedynke logiczna. Sygnal alternatywnej funkcji wyjsciowej wyprowadzony jest wspólnie z sygnalem rejestru portu przez dwuwejsciowa bramke *NAND*. Natomiast sygnal alternatywnej funkcji wejsciowej jest doprowadzony do odpowiednich ukladów wewnetrznych (np. ukladu przerwan zewnetrznych) poprzez dodatkowe bufory.

Ostatnia grupe stanowia porty, które oprócz standardowych funkcji wejscia–wyjscia nie realizuja zadnych dodatkowych funkcji. W takim ukladzie sygnal wyjsciowy rejestru portu steruje bezposrednio tranzystorem wyjsciowym, wymuszajac odpowiedni stan na koncówce ukladu. Schemat standardowych portów wejscia–wyjscia przedstawiono na rys. 4.4.



Rys. 4.4. Schemat standardowych portów wejscia–wyjscia

Elementem wspólnym w strukturze kazdego z portów jest przerzutnik typu D, bedacy elementem rejestru danego portu. Wszystkie rejestry portów mikrokontrolera **SAB 80C535** umieszczone sa w przestrzeni adresowej wewnetrznej pamieci danych w obszarze rejestrów specjalnych **SFR** (patrz rozdzial 2, tabela 2.2). Sterowanie por-

tami odbywa sie wiec w podobny sposób, jak jest w przypadku pozostalych rejestrów mikrokontrolera. Rejestry portów naleza do grupy rejestrów adresowanych bitowo i bajtowo. Dzieki temu mozliwe jest ustawianie kazdego bitu rejestru danego portu (lub odczytanie stanu dowolnego bitu rejestru portu) niezaleznie, wykorzystujac rozkazy operacji na bitach. Nalezy przy tym zwrócic uwage, ze adres najmlodszego bitu rejestru portu jest zarazem adresem calego rejestru. Oczywiscie mozliwe jest modyfikowanie zawartosci calego rejestru portu (lub odczytywanie jego stanu) rozkazami operujacymi na bajtach, np. rozkazem *MOV*.

Dane przesylane do portu zapisywane sa w buforowym rejestrze wyjsciowym. Stan tego rejestru nie zmienia sie az do ponownego wpisania nowych wartosci.

Odczytywanie danej z portu odbywa sie przez bufory, przy czym dana moze zostac odczytana albo z rejestru portu, albo bezposrednio z koncówki portu: zalezy to od uzytego rozkazu.

Pobieranie danych z koncówek ukladu odbywa sie poprzez rozkazy sluzace do odczytania danych z wejsc mikrokontrolera i testowania ich oraz przesylania do pamieci lub do innego rejestru mikrokontrolera, np. MOV R0,P1; ADD A,P1; ANL A,P1; JB P1.0,d; CJNE A,P5.

Do odczytywania danych z rejestru wyjsciowego portu sluza rozkazy, których wykonanie powoduje odczytanie, modyfikacje i ponowne zapisanie danych do portu. Nalezy przy tym pamietac, ze rozkazy te dotycza stanu wyjsc mikrokontrolera wymuszanego przez zawartosc rejestru wyjsciowego portu.

W niektórych przypadkach stan rejestru wyjsciowego portu nie jest zgodny ze stanem logicznym, okreslonym przez poziom napiecia na koncówkach ukladu. Dzieje sie tak na przyklad, wtedy gdy bezposrednio do wyjscia portu przylaczona jest baza tranzystora. Aby wprowadzic tranzystor w stan przewodzenia nalezy do komórki rejestru portu wpisac stan jedynki logicznej. Poniewaz spadek napiecia na zlaczu baza–emiter przewodzacego tranzystora wynosi 0,6 ÷ 0,7V, wiec odczytujac stan koncówki portu otrzymamy wartosc zera logicznego.

Rozkazy modyfikujace zawartosc rejestru wyjsciowego portu, ale nie zmieniajace stanu logicznego na koncówkach ukladu przedstawiono ponizej, przy czym w opisie

rozkazów uzyto oznaczen:

Pi – adres portu jako rejestr rejestr specjalny **SFR**, np. P1,

Pi.x – adres bitu x portu i np. P1.1,

r – oznacza A lub argument bezposredni #n (patrz tez: zal. 1 – opis listy rozkazów),

d – przesuniecie

MOV Pi.x,C	ANL Pi,r	INC Pi	JBC Pi.x
SETB Pi.x	ORL Pi,r	DEC Pi	DJNZ Pi,d
CLR Pi.x	XRL Pi,r		CPL Pi.x

4.1.2. Przyklady programowania portów

Przyklad 1

```
;*********************************************************************
;Program generowania przebiegu o czestotliwosci zegara systemowego i wypelnieniu
;50% na wyjsciu P1.0
;*********************************************************************

        ORG $50                 ;Adres programu

LOOP1:  CPL  P1.0               ;Zmiana stanu bitu P1.0
          SJMP      LOOP1       ;Skok do poczatku  programu
```

Przyklad 2

```
;*********************************************************************
;Program generowania przebiegu o czestotliwosci 1kHz i wypelnieniu 50% na wyjsciu
;P1.1
;*********************************************************************
DEL_1       EQU   $0FA          ;Deklaracja opóznienia czasowego
                                 ;decydujacego o czestotliwosci

        ORG $100                ;Adres  programu

LOOP2:  MOV ACC,#DEL_1          ;Przeslanie do akumulatora  wartosci
LOOP3:  DJNZ ACC,LOOP3          ;opóznienia zmiany stanu bitu P1.1
          CPL P1.1              ;Zmiana stanu bitu P1.1
          SJMP LOOP2            ;Skok do poczatku  programu
```

Przyklad 3

```
;*********************************************************************
;Program generowania przebiegu o czestotliwosci 1kHz i wypelnieniu 25% na wyjsciu
;P1.2
;*********************************************************************
DEL_2       EQU  $5A          ;Deklaracja opóznienia czasowego 2
DEL_3       EQU  $FA          ;Deklaracja opóznienia czasowego 3


        ORG $150              ;Adres programu
LOOP4:  MOV  ACC,#DEL_2       ;Wartosc opóznienia zmiany stanu
LOOP5:  DJNZ ACC,LOOP5        ;bitu P1.2
        CPL  P1.2             ;Zmiana stanu bitu P1.2
        MOV  ACC,#DEL_3       ;Wartosc opóznienia zmiany stanu
LOOP6:  DJNZ ACC,LOOP6        ;bitu P1.2
        CPL  P1.2             ;Zmiana stanu bitu P1.2
        SJMP LOOP4            ;Skok do poczatku programu
```

Przyklad 4

```
;*********************************************************************
;Generowanie przebiegu PWM na wyjsciu P1.3 przy uzyciu licznika T2.
;Program wykorzystuje komparator CC3
;*********************************************************************

;************************
;Deklaracja stalych programu
;************************

TIMER       EQU  $11          ;Autoladowanie po przepelnieniu licznika
                              ;taktowanie sygnalem wewnetrznym bez
                              ;dzielnika
COMP_EN     EQU $80           ;Odblokowanie trybu porównania i wpisu
RELOAD_L    EQU $00           ;Wartosc poczatkowa licznika T2 po
                              ;przepelnieniu
RELOAD_H    EQU $FF           ;Wartosc poczatkowa licznika T2 po
                              ; przepelnieniu
COMP_L      EQU $37           ;Wartosc porównania rejestru CCL3
COMP_H      EQU $FF           ;Wartosc porównania rejestru CCH3


;***************
;Program glówny.
;***************

        ORG $200                    ;Adres poczatku programu
```

```
ORL  T2CON,#TIMER        ;Konfiguracja licznika T2
MOV  CCEN,#COMP_EN       ;Wybór komparatora CC3
MOV  CRCL,#RELOAD_L      ;Wpis wartosci poczatkowej L , czestotli-
                         ;wosc
MOV  CRCH,#RELOAD_H      ;Wpis wartosci poczatkowej H , czestotli-
                         ;wosc
MOV  CCL3,#COMP_L        ;Wpis wartosci porównania L , wypelnienie
MOV CCL3,#COMP_H         ;Wpis wartosci porównania H, wypelnienie
```

4.2. Port szeregowy

4.2.1. Opis portu szeregowego

Mikrokontroler **SAB 80C535** jest wyposazony w port szeregowy, umozliwiajacy transmisje danych przez linie portu **P3.** Port szeregowy jest portem typu „full–duplex" co oznacza, ze dane moga byc wysylane i przyjmowane równoczesnie. Dane przyjmowane sa buforowane w 9–bitowym rejestrze przesuwnym. Dzieki temu nastepna dana moze byc juz przyjmowana, w czasie kiedy poprzednia jest przepisywana z rejestru wejsciowego **SBUF** do akumulatora. Przepisywanie danych z rejestru **SBUF** do akumulatora odbywa sie w sposób równolegly. Przepisywanie musi zakonczyc sie przed przyjeciem nowej danej, w przeciwnym razie dana przyjmowana jest tracona. Podczas wysylania danych rejestr **SBUF** traktowany jest jako rejestr wyjsciowy. Wpisanie danych do tego rejestru powoduje wyslanie ich przez port szeregowy.

Zamiana postaci danych z równoleglej na szeregowa i odwrotnie oraz sterowanie wysylaniem slowa odbywa sie automatycznie. Rejestr **SBUF** umieszczony jest w przestrzeni adresowej rejestrów specjalnych pod adresem 99H**.** Port szeregowy moze pracowac w jednym z czterech trybów pracy, które przedstawiono w tabeli 4.2. Sterowanie portem szeregowym odbywa sie za pomoca adresowanego bitowo rejestru **SCON.**

Rejestr **SCON** adres **98H**

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Znaczenia poszczególnych bitów sa nastepujace:

- **SM1, SM0** - ustawienie trybu pracy (patrz tab. 4.2).

- **SM2** - znacznik maskowania odbioru transmisji:

  w trybie 0: **SM2**=0,

  w trybie 1: jesli **SM2**=1 i bit stopu=0, to przyjmowane slowo jest ignorowane,

  jesli **SM2**=0 i bit stopu=0, to slowo jest przyjete,

  w trybie 2 i 3: jesli **SM2**=1 i dziewiaty bit odebranego slowa=0, to przyjmowane slowo jest ignorowane,

  jesli **SM2**=0 i dziewiaty bit odebranego slowa=0, to slowo jest przyjete.
- **REN** - uaktywnienie odbiornika transmisji szeregowej (**REN**=1),
- **TB8 –** w trybie 0 i 1 nie uzywany, w trybie 2 i 3 przyjmuje wartosc dziewiatego bitu slowa wysylanego,
- **RB8 –** w trybie 0 nie uzywany, w trybie 1 bit przyjmuje wartosc bitu stopu odbieranego slowa (0 lub 1) jesli **SM2**=0. Jesli **SM2**=1, przyjmuje wartosc bitu stopu (wylacznie, gdy bit stopu ma wartosc 1). W trybie 2 i 3 przyjmuje wartosc dziewiatego bit odebranego slowa.
- **TI** - znacznik wyslania slowa i zgloszenie przerwania. Zerowany wylacznie programowo.
- **RI** - znacznik odebrania slowa i zgloszenie przerwania. Zerowany wylacznie programowo.

Tabela 4.2. Tryby pracy portu szeregowego

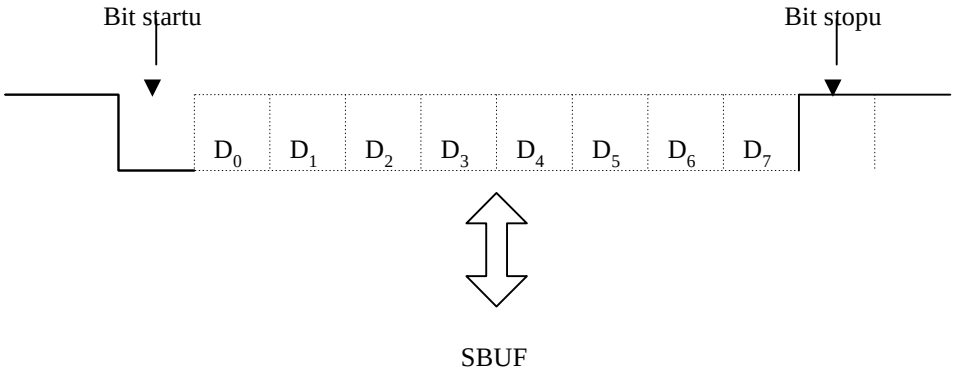
Tryb	SM0	SM1	Opis trybu
0	0	0	Transmisja szeregowa synchroniczna slów 8-bitowych taktowanych sygnalem zegarowym
1	0	1	Transmisja szeregowa asynchroniczna slów 8-bitowych, przy czym szybkosc ustalana jest programowo
2	1	0	Transmisja szeregowa asynchroniczna slów 9-bitowych o szybkosci 1/32 lub 1/64 czestotliwosci zegara
3	1	1	Transmisja szeregowa asynchroniczna slów 9-bitowych, szybkosc okreslana programowo

⇒ Praca portu szeregowego w trybie 0

W trybie 0 nadawanie i odbiór znaków odbywa sie przez linie **P3.0** (**RXD**). Przez linie **P3.1** (**TXD**) wysylany jest natomiast sygnal taktujacy o stalej czestotliwosci równej f_{osc}/12. Dlugosc wysylanego slowa wynosi 8 bitów, przy czym jako pierwszy wysylany jest bit najmniej znaczacy. Wysylanie rozpoczyna sie automatycznie po wpisaniu do **SBUF** wysylanego slowa. Po wyslaniu 8 bitów danych nastepuje ustawienie znacznika **TI** w rejestrze **SCON**, co dla procesora jest sygnalem konca wysylania znaku. Znacznik **TI** moze byc kasowany programowo w trakcie wysylania danych. Odbiór danych rozpoczyna sie w momencie programowego wyzerowania znacznika **RI** w rejestrze **SCON** pod warunkiem, ze bit **REN** ma wartosc jedynki logicznej. Po odebraniu 8 bitów slowo z rejestru przesuwnego przepisywane jest do **SBUF**, a nastepnie ustawiany jest znacznik **RI**, co oznacza koniec odbioru danych.

⇒ Praca portu szeregowego w trybie 1

W trybie 1 wysylanie slowa odbywa sie przez linie **P3.1**, odbiór natomiast dokonywany jest przez linie **P3.0**. Nadawane i odbierane slowo ma dlugosc 10 bitów: bit startu, 8 bitów danych i bit stopu. Format slowa przedstawiono na rys. 4.5.



Rys. 4.5. Format slowa w trybie 1 pracy portu szeregowego

Nadawanie rozpoczyna sie automatycznie po wpisaniu do **SBUF** wysylanej danej, przy czym bity danych wysylane sa w kolejnosci od najmniej znaczacego. Po wyslaniu wszystkich bitów danych zostaje wytworzony i wyslany bit stopu, oraz nastepuje ustawienie znacznika **TI** w stan jedynki logicznej, co jest sygnalem zakonczenia nada-

wania. Odbiór danych rozpoczyna sie po wykryciu na wyprowadzeniu **P3.0** zmiany stanu z 1 na 0 pod warunkiem, ze znacznik **REN** ma wartosc jedynki logicznej i wyzerowany jest znacznik **RI**. Po odebraniu wszystkich bitów przyjete slowo przepisywane jest do rejestru **SBUF** oraz ustawiany jest w stan jedynki logicznej znacznik **RI**, co jest sygnalem zakonczenia odbioru. Bit stopu odebranego slowa wpisywany jest na pozycje znacznika **RB8**.

Predkosc transmisji danych ustalana jest programowo z wykorzystaniem licznika **T1**, która zakladajac prace licznika w trybie 2, okreslona jest zaleznoscia:

$$BD = \frac{2^{SMOD}}{32} \frac{f_{osc}}{12\,[256-(TH1)]},$$

gdzie (TH1) – liczba wpisana do rejestru TH1.

⇒ Praca portu szeregowego w trybie 2

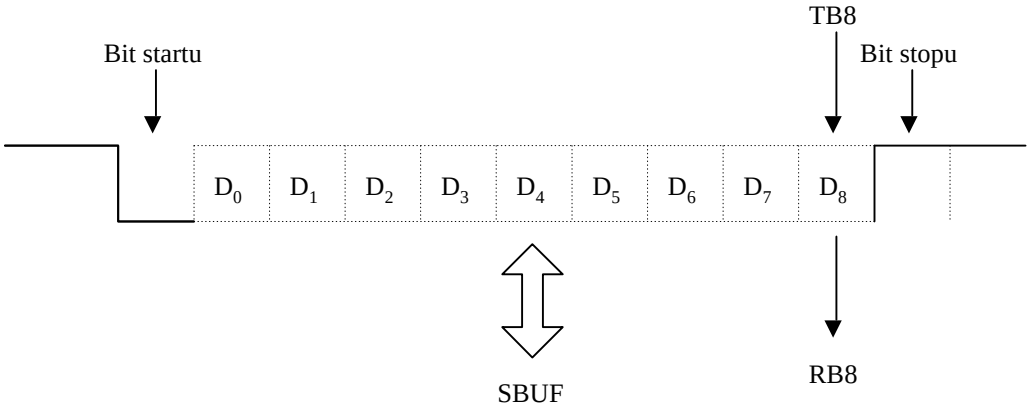
W trybie 2 proces wysylania i odbierania danych przebiega tak samo jak w trybie 1, z ta róznica, ze inna jest dlugosc slowa. W tym trybie slowo jedenastobitowe (bit startu, 9 bitów danych i bit stopu) wysylane jest przez linie **P3.1**, odbiór odbywa sie przez linie **P3.0**. W trakcie nadawania bit **TB8** rejestru **SCON** traktowany jest jako dziewiaty bit danych, w trakcie odbioru dziewiaty bit danych przepisywany jest na pozycje bitu **RB8**. Format slowa przedstawiono na rys. 4.6.

Predkosc transmisji w tym trybie pracy zalezy od ustawienia bitu **SMOD** w rejestrze **PCON** i moze wynosic:

- 1/32 czestotliwosci oscylatora f_{osc} dla SMOD=1,
- 1/64 czestotliwosci oscylatora f_{osc} dla SMOD=0.

⇒ Praca portu szeregowego w trybie 3

Nadawanie i odbiór danych w trybie 3 odbywa sie tak samo jak w trybie 2 pracy portu szeregowego. Róznica wystepuje w ustalaniu predkosci transmisji. W odróznieniu od trybu 2, w trybie 3 predkosc transmisji ustalana jest programowo tak samo jak w trybie 1.Format slowa (taki sam jak w trybie 2) przedstawiono na rys. 4.6.



Rys. 4.6. Format slowa w trybie 2 i 3 pracy portu szeregowego

4.2.2. Przyklady programowania portu

Przyklad 1

```
;*****************************************************************************
;Program demonstrujacy dzialanie portu szeregowego w SAB 80C535. Program
;wysyla co pewien czas dane z pamieci (od $30 do $3F) portem szeregowym
;pracujacym w trybie 0 i jednoczesnie wysyla dane na port P1
;*****************************************************************************
;*************************
;Deklaracja symboli i adresów
;*************************

MEM_START   EQU   $30      ;Adres poczatku pamieci z danymi
MEM_END     EQU   $40      ;Koniec zakresu pamieci z danymi

;*****************************************************************************
      ORG   $00
MAIN:
      LCALL  INIT_MEM          ;Inicjalizacja danych wysylanych
      MOV  R0,#MEM_START  ;Rejestr R0 sluzy do adresowania pamieci
      MOV  SCON,#$00           ;Tryb 0 portu szeregowego oraz wylaczenie
                               ;odbiornika (REN = 0)

LOOP:                          ;Petla glówna
      MOV  P1,@R0              ;Wyslanie danej do portu (w celu wizualizacji
                               ;sposobu dzialania programu)
      MOV  SBUF,@R0            ;Wyslanie danej do portu szeregowego
      INC   R0                 ;R0 wskazuje na nastepna komórke pamieci
      CJNE  R0,#MEM_END,LOOP_END      ;Jesli nie przekroczyl zakresu to
```

```
                                        ;kontynuuje wysylanie danych
    MOV    R0,#MEM_START   ;Jesli przekroczyl zakres, to zaczyna
                           ;wysylanie danych od poczatku
LOOP_END:
    LCALL  DELAY1          ;Opóznienie umozliwiajace obserwacje
SJMP   LOOP                ;dzialania programu


;****************************
;Procedura  inicjowania  pamieci
;****************************
INIT_MEM:                  ;Procedura inicjuje dane do wyslania
    MOV   R0,#MEM_START    ;Dane umieszczane w pamieci danych mozna
                           ; zmienic w zaleznosci od potrzeby
    MOV   R7,#1            ;Wpis pierwszej danej do rejestru pomocnicze-
                           ;go.
FILL:
    MOV  A,R7              ;Wysylanie danej przez akumulator,  poniewaz  nie
                           ;ma rozkazu  MOV @R0,R7
    MOV  @R0,A             ;Ta konkretna  dana wypelnia pamiec kolejnymi
                           ;potegami liczby 2, co realizowane  jest
                           ;przez przesuwanie w lewo zawartosci rejestru
    RL   A                 ;R7 (za posrednictwem akumulatora)
    MOV   R7,A
    INC    R0
    CJNE  R0,#MEM_END,FILL
    RET

;****************************
;Procedura  opóznienia   czasowego
;****************************

DELAY1:
            MOV    R5,#20
HOP7:       MOV    R6,#100
HOP6:       MOV    R7,#100
HOP5:       DJNZ   R7,HOP5
            DJNZ   R6,HOP6
            DJNZ   R5,HOP7
            RET
```

Przyklad 2

```
;*********************************************************************
;Program demonstrujacy dzialanie portu szeregowego w SAB 80C535. Program
;odbiera dane przychodzace przez port szeregowy pracujacy w trybie 0 i umieszcza je
;w pamieci wewnetrznej o adresach od $30 do $3F oraz wysyla dane na port P1
;*********************************************************************

;**************************
;Deklaracja symboli i adresów
;**************************
MEM_START   EQU   $30      ;Adres poczatku pamieci na przychodzace dane
MEM_END     EQU   $40      ;Koniec zakresu pamieci na dane

ORG   $00
      LJMP   MAIN          ;Skok do programu glównego, w celu ominiecia
                           ;obszaru obslugi przerwan
      ORG    $23
      LJMP   INT_RS        ;Skok do procedury obslugi przerwania od portu
                           ;szeregowego

MAIN:
      MOV    R0,#MEM_START ;Rejestr R0 sluzy do adresowania pamieci danych
      SETB   EAL           ;Odblokowanie wszystkich przerwan
      SETB   ES            ;Odblokowanie przerwania od portu szeregowego
      MOV    SCON,#$10     ;Tryb 0 portu szeregowego oraz uaktywnienie
                           ;odbiornika (REN = 1)
LOOP: SJMP   LOOP          ;Nieskonczona petla, gdyz odbieranie i tak
                           ;odbywa sie w procedurze obslugi przerwania

;*********************************
;Procedura odbioru portu szeregowego
;*********************************
INT_RS:
      MOV   @R0,SBUF       ;Przepisanie odebranej danej do pamieci danych
      MOV   P1,@R0         ;a nastepnie wyslanie jej do portu P1
      INC   R0             ;R0 wskazuje na kolejna komórke pamieci
      CJNE  R0,#MEM_END,INT_RS_END  ;Jesli komórka miesci sie jeszcze
                                    ;w zakresie, to skok na koniec.
      MOV   R0,#MEM_START  ;Jesli nie, to pamiec bedzie zapelniana od poczatku
INT_RS_END:
      CLR   RI             ;Znacznik RI wymaga kasowania programowego
      RETI
```

Przyklad 3

```
;*****************************************************************
;Program demonstrujacy dzialanie portu szeregowego w SAB80C535. Program
;odbiera i wysyla dane przez port szeregowy pracujacy w trybie 1. Dane przychodzace
;umieszczane sa w pamieci od adresu $30 do $3F i wysylane do portu P1. Procedura
;odbierajaca wywolywana jest przez przerwanie.
;Dane wysylane znajduja sie w pamieci o adresie $40 do $4F i sa wysylane co pewien
;czas w petli  glównej programu
;*****************************************************************
**************************
;Deklaracja symboli i adresów
;*************************

MEM_R_START   EQU   $30      ;Adres poczatku pamieci na przychodzace dane
MEM_R_END     EQU   $40      ;Koniec zakresu pamieci na dane przychodzace
MEM_T_START   EQU   $40      ;Adres poczatku pamieci na wysylane dane
MEM_T_END     EQU   $50      ;Koniec zakresu pamieci na dane wysylane

;*****************************************************************
      ORG   $00            ;Adres poczatku programu

      LJMP  MAIN           ;Skok do programu glównego, w celu ominiecia obszaru
                           ;obslugi przerwan
      ORG   $23

      LJMP  INT_RS         ;Skok do procedury obslugi przerwania od portu
                           ;szeregowego

MAIN:
      LCALL  INIT_MEM              ;Inicjalizacja danych do wysylania
      MOV    R0,#MEM_R_START       ;Rejestr R0 sluzy do adresowania
                                   ;obszaru pamieci na dane odbierane
      MOV    R1,#MEM_T_START       ;Rejestr R1 sluzy do adresowania pamieci
                                   ;z danymi wysylanymi

      SETB   EAL           ;Odblokowanie wszystkich przerwan
      SETB   ES            ;Odblokowanie  przerwania od portu szeregowego
      SETB   BD            ;Wlaczenie taktowania z dzielnika :39
      ORL    PCON,#80      ;Ustawienie bitu SMOD w celu uzyskania taktowania
                           ;czestotliwoscia 9600 Hz (przy czestotliwosci oscylatora
                           ;12 MHz)
      MOV    SCON,#$70     ;Tryb 1 portu szeregowego oraz uaktywnienie odbiornika
                           ;(REN = 1) i kontroli przychodzacych danych (SM2 = 1)

LOOP:                      ;Petla glówna
      MOV    SBUF,@R1      ;Wyslanie danej do portu szeregowego
```

```
    INC   R1                    ;R1 wskazuje na nastepna komórke pamieci
    CJNE  R1,#MEM_T_END,LOOP_END ;Jesli  obszar nie przekroczyl zakresu, to
                                ;kontynuacja wysylania
    MOV  R1,#MEM_T_START        ;Jesli przekroczyl, to zaczyna od
                                ;poczatku

LOOP_END:
    LCALL  DELAY1       ;Opóznienie umozliwiajace obserwacje
                        ;dzialania programu
    SJMP   LOOP

;***************************
;Procedura  inicjowania  pamieci
;***************************
INIT_MEM:                       ;Procedura inicjuje dane do wyslania
    MOV   R1,#MEM_T_START       ;Dane mozna  zmieniac w zaleznosci od
                                ;potrzeby
    MOV   R7,#1
FILL:
    MOV  A,R7           ;Przesylanie przez akumulator poniewaz nie ma rozkazu
                        ;MOV @R0,R7
    MOV   @R1,A         ;Ta konkretna  dana wypelnia pamiec kolejnymi potegami
                        ;liczby 2, co realizowane jest przez przesuwanie w lewo
    RL     A            ;zawartosci rejestru R7 (za posrednictwem akumulatora)
    MOV  R7,A
    INC    R1
    CJNE  R1,#MEM_T_END,FILL
    RET

;***************************
;Procedura  opóznienia  czasowego
;***************************
DELAY1:
        MOV     R5,#100
HOP7:   MOV     R6,#100
HOP6:   MOV     R7,#100
HOP5:   DJNZ    R7,HOP5
        DJNZ    R6,HOP6
        DJNZ    R5,HOP7
        RET

;*********************************
;Procedura  odbioru  z  portu  szeregowego
INT_RS:
    JNB   RI,CLR_TI     ;Jesli przerwanie wywolal nadajnik, to skok na
```

```
                        ;koniec  procedury, gdyz jest to obsluga jedynie odbiornika
    MOV  @R0,SBUF       ;Przepisanie odebranej danej do pamieci
    MOV   P1,@R0        ;a nastepnie do portu P1
    INC    R0           ;R0 wskazuje na kolejna komórke pamieci
    CJNE  R0,#MEM_R_END,INT_RS_END  ;Jesli komórka miesci sie jeszcze
                                    ;w zakresie to skok na koniec
    MOV    R0,#MEM_R_START          ;Jesli nie, to pamiec bedzie zapelniana
                                    ;od poczatku

INT_RS_END:
    CLR    RI           ;Znacznik RI wymaga kasowania programowego
CLR_TI:
    CLR    TI           ;Zeruje znacznik TI, gdyz i  tak nie jest on wykorzystywany
                        ;w procedurze wysylajacej, gdyz czas trwania petli
                        ;DELAY1 jest duzo dluzszy niz czas wysylania danej
    RETI
```

4.3. Liczniki T0 i T1

4.3.1. Opis liczników T0 i T1

Liczniki **T0** i **T1**, wraz z licznikiem **T2** stanowia uklad czasowo-licznikowy mikrokontrolera **SAB 80C535**. Moga pracowac one zarówno *jako liczniki zliczajace impulsy zewnetrzne*, doprowadzone do wejsc ukladu (**P3.4** dla licznika **T0** i **P3.5** dla licznika **T1**), jak i *czasomierze zliczajace wewnetrzne impulsy zegarowe*. Liczniki te moga pracowac w jednym z czterech, indywidualnie ustawianych, trybach pracy.

Do programowego sterowania praca liczników **T0** i **T1** sluza dwa rejestry sterujace: **TMOD** (adresowany bajtowo) i **TCON** (adresowany bajtowo i bitowo). Rejestr **TMOD** sluzy do ustawiania trybu pracy i funkcji realizowanej przez wybrany licznik. Znaczenie bitów sterujacych przedstawiono ponizej:

Rejestr **TMOD** adres **089H**

GATE	$C/\overline{T}$	M1	M0	GATE	$C/\overline{T}$	M1	M0
Licznik T1				Licznik T0			

- **M0, M1** – ustawienie jednego z czterech trybów pracy, przy czym:

  M0=0, M1=0 – tryb 0

M0=1, M1=0 – tryb 1

M0=0, M1=1 – tryb2

M0=1, M1=1 – tryb 3

- **C/$\overline{\text{T}}$ -** ustawienie realizowanej funkcji:

  C/$\overline{\text{T}}$=0 – licznik pracuje jako czasomierz,

  C/$\overline{\text{T}}$=1 – licznik zlicza impulsy zewnetrzne.

- **GATE** – bramkowanie zliczania sygnalem zewnetrznym z wejscia $\overline{\text{INT0}}$ lub $\overline{\text{INT1}}$ (odpowiednio dla kazdego z liczników).

W rejestrze **TCON** znaczenie bitów sterujacych i kontrolnych jest nastepujace:

Rejestr **TCON** adres **088H**

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

- **TF1, TF0** – znaczniki przepelnienia liczników,
- **TR1, TR0** – bity sterujace:

  TR1=0 – licznik T1 zatrzymany,

  TR0=0 – licznik T0 zatrzymany,

  TR1=1 – licznik T1 pracuje,

  TR0=1 – licznik T0 pracuje.

- **IE1 –** znacznik zgloszenia przerwania zewnetrznego $\overline{\text{INT1}}$,
- **IT1 –** ustawienie sposobu zgloszenia przerwania $\overline{\text{INT1}}$:

  IT1=0 – zgloszenie przerwania poziomem niskim sygnalu,

  IT1=1 – zgloszenie przerwania poziomem wysokim sygnalu.

- **IE0** – znacznik zgloszenia przerwania zewnetrznego $\overline{\text{INT0}}$:

  IT0=0 – zgloszenie przerwania poziomem niskim sygnalu,

  IT0=1 – zgloszenie przerwania poziomem wysokim sygnalu.

Uruchomienie licznika **Ti** nastepuje po wpisaniu jedynki logicznej do bitu **TRi** w rejestrze **TCON**. Raz uruchomiony moze byc zatrzymany w wyniku wpisania zera

logicznego do bitu **Tri.** Jesli w rejestrze **TMOD** bit **GATE** ma wartosc jedynki logicznej, to mozliwe jest zewnetrzne sterowanie zliczaniem – licznik pracuje tylko wtedy, gdy sygnal $\overline{\text{INTi}}$ =1.

Jesli liczniki pracuja jako czasomierze (zliczaja impulsy wewnetrzne), to zawartosc licznika w kazdym cyklu maszynowym zwiekszana jest o 1. Jesli natomiast liczniki realizuja funkcje zliczania impulsów zewnetrznych, to odpowiednie wejscie **T0** lub **T1** próbkowane jest w trakcie kazdego cyklu maszynowego. Jesli przy dwóch kolejno po sobie nastepujacych próbkach wykryta zostanie zmiana poziomu z 1 na 0, to w czasie nastepnego cyklu maszynowego zawartosc licznika zwiekszona zostanie o 1. Ponadto w tym trybie pracy spelnione musza byc nastepujace warunki:

– kazdy stan logiczny zliczanych impulsów musi trwac przez co najmniej jeden cykl maszynowy,
– maksymalna czestotliwosc zliczanych impulsów nie moze byc wieksza niz $f_{osc}/24$ (poniewaz do wykrycia zmiany na wejsciu potrzebne sa dwa cykle maszynowe).

Szesnastobitowe liczniki **T0** i **T1** podzielone sa na dwa osmiobitowe bajty (mniej i bardziej znaczacy) dostepne programowo jako rejestry specjalne:

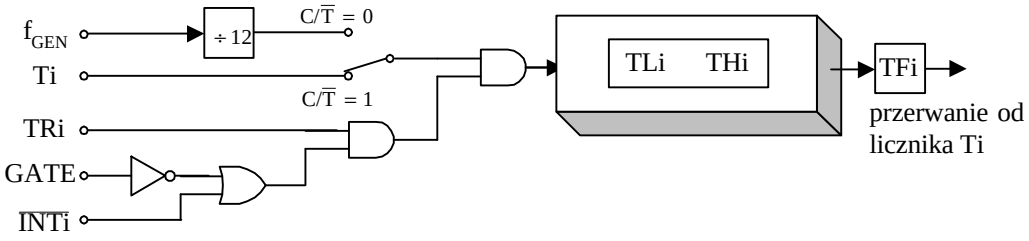
- **TH0** (adres 8CH) – bardziej znaczacy bajt licznika T0,
- **TL0** (adres 8AH) – mniej znaczacy bajt licznika T0,
- **TH1** (adres 8DH) – bardziej znaczacy bajt licznika T1,
- **TL1** (adres 8BH) – mniej znaczacy bajt licznika T1.

Kazdy z liczników moze pracowac w jednym z czterech trybów pracy, niezaleznie od realizowanej funkcji. Tryby 0, 1 i 2 moga pracowac oba liczniki **T0** i **T1**, natomiast w trybie 3 licznik **T1** nie pracuje.

⇒ Tryb 0 i tryb 1

W trybie 0 modyfikowanych jest tylko 13 bitów licznika: osiem bitów rejestru **THi** (i=0,1) i bity 3...7 rejestru **TLi** (piec bardziej znaczacych). Stan bitów 0...2 rejestru **TLi** jest nieokreslony i powinien byc ignorowany. Po uruchomieniu licznika impulsy zliczane sa od wartosci poczatkowej. Po przepelnieniu licznika (osiagnieciu war-

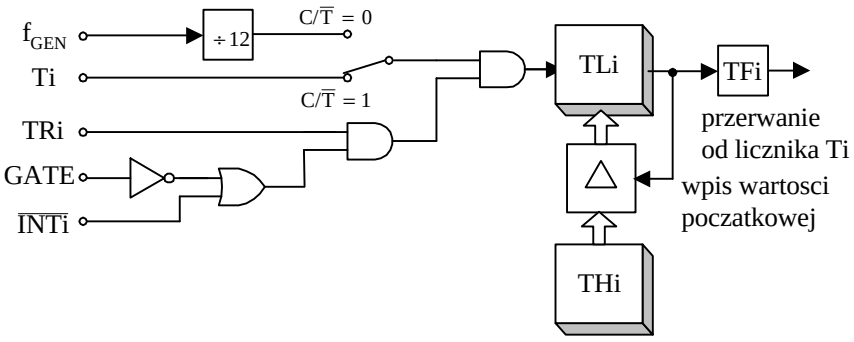
tosci 1FFFH+1) zawartosc rejestru jest kasowana i zliczanie rozpoczyna sie od nowa. W trybie 1 pracuje caly 16-bitowy licznik. W tym trybie pracy licznik moze byc wykorzystywany jako czasomierz, a takze jako licznik impulsów zewnetrznych. Schemat blokowy liczników **T0** i **T1** pracujacych w trybie 0 i 1 przedstawiono na rys. 4.7.



Rys. 4.7. Schemat blokowy liczników T0 i T1 pracujacych w trybie 0 i 1

⇒ Tryb 2

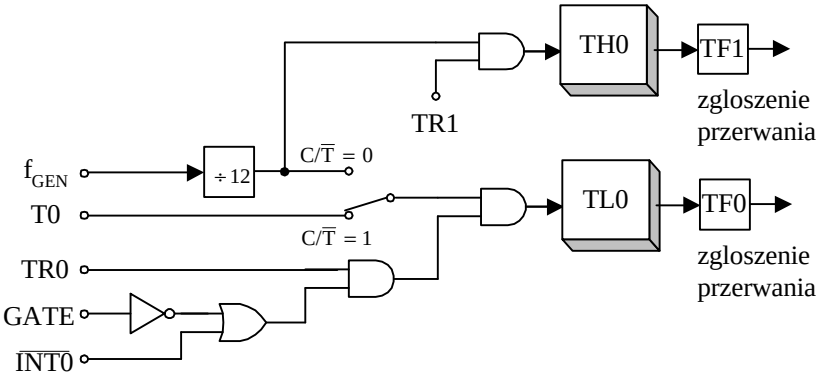
W tym trybie pracy licznik pracuje jako licznik 8-bitowy z autoladowaniem, przy czym rejestr **TLi** jest rejestrem zliczajacym, natomiast w rejestrze **THi** zapisana jest wartosc poczatkowa, która moze byc zmieniana programowo. W momencie przepelnienia licznika (osiagniecia przez rejestr **TLi** wartosci FF+1) nastepuje autoladowanie wartosci poczatkowej i ustawienie znacznika **TFi**. Schemat blokowy liczników **T0** i **T1** pracujacych w tym trybie przedstawiono na rys. 4.8.



Rys. 4.8. Schemat blokowy liczników T0 i T1 pracujacych w trybie 2

⇒ Tryb 3

W trybie 3 licznik **T1** nie pracuje. Poszczególne bajty licznika **T0** (**TH0** i **TL0**) pracuja jako dwa niezalezne liczniki 8-bitowe. Mniej znaczacy bajt licznika **TL0** moze pracowac zarówno jako czasomierz, jak i licznik impulsów zewnetrznych z wejscia **T0**. W trybie 3 licznik **TL0** jest sterowany tak jak licznik **T0** w trybach 0 i 1 z wykorzystaniem bitów **TR0**, **GATE**, **C/$\overline{T}$,** oraz tak jak licznik **T0** w momencie przepelnienia ustawia znacznik przerwania **TF0**. Natomiast bardziej znaczacy bajt licznika **TH0** moze byc wykorzystany jedynie jako czasomierz zliczajacy impulsy wewnetrzne. Nalezy przy tym zaznaczyc, iz w tym trybie licznik **TH0** jest sterowany bitem **TR1**, oraz w momencie przepelnienia ustawia znacznik przerwania **TF1**. Z tego wzgledu, jesli licznik **T0** pracuje w trybie 3, to licznik **T1** moze pracowac w pozostalych trybach pracy bez mozliwosci bramkowania jego wejscia i testowania znacznika przerwania. Tryb 3 pracy licznika **T0** jest w praktyce rzadko stosowany, gdyz wtedy licznik **T1** moze byc uzyty prawie wylacznie do ustalania predkosci transmisji portu szeregowego. Schemat blokowy licznika **T0** pracujacego w 3 trybie przedstawiono na rys.4.9.



Rys. 4.9. Schemat blokowy licznika T0 pracujacego w trybie 3

4.3.2. Przyklad programowania liczników T0 i T1

```
;*****************************************************************
;Program zliczania impulsów podawanych na wejscie licznika T1 w czasie ustalonym
;przez przerwania generowane w liczniku T2
;*****************************************************************
;*********************************
; Deklaracja adresów wyniku pomiarów
```

```
;**********************************
POM1        BYTE ($20)
POM2        BYTE ($21)
;******************
; Start programu
;******************
        ORG $00                 ;Adres startu programu
        LJMP MAIN               ;Skok do programu  głównego
        ORG $002B               ;Adres  przerwania  od  przepelnienia licznika  T2
        LJMP T2INT              ;Skok do programu  obslugi  przerwania

;****************************
;Konfiguracja  licznika  T1 i T2
;****************************
ORG $100
MAIN:  SETB  EAL                ;Odblokowanie  wszystkich  przerwan
        SETB ET2                ; Odblokowanie  przerwania  od  licznika  T2
        MOV  TMOD,#$50          ;Ustawienie  licznika  T1  jako  16  bitowy  do
        ANL   TCON,#$4F         ;zliczania  impulsów  zewnetrznych
        ORL   TCON,#$40         ;Start licznika T1
        ANL   T2CON,#$20        ;Ustawienie  licznika  T2  w  trybie  0
        ORL   T2CON,#$11        ;do generowania  przerwan  co  10 ms
        MOV  CRCL,#$0F0         ;Wartosc  poczatkowa  licznika  T2  po
        MOV  CRCH,#$0D8         ;przeladowaniu
        MOV  TL2,#$0F0          ;Wartosc  poczatkowa  licznika  przy
        MOV  TH2,#$0D8          ;starcie
LOOP: SJMP   LOOP               ;Adres programu oczekiwania na przerwanie

;*****************************
;Program  obslugi  przerwania
;*****************************
        ORG  $200
T2INT:  CLR TF2         ;Zerowanie  znacznika  przerwania  od  licznika  T2
          CLR  TR1      ;Wylaczenie  licznika  T1
        MOV  R7,TH1     ;Przeslanie  wartosci  licznika  T1  do  rejestru  R7
        MOV  POM1,R7    ;Przeslanie  wartosci  licznika  do  pamieci wewn. RAM
        MOV  R6,TL1     ;Przeslanie  wartosci  licznika  T1  do  rejestru  R6
        MOV  POM2,R6    ;Przeslanie  wartosci  licznika  do  pamieci  wewn. RAM
        MOV  P5,R6
        MOV  TL1,#$00   ;Zerowanie  licznika  T1 , bajtu  mniej  znaczacego
        MOV  TH1,#$00   ;Zerowanie  licznika  T1 , bajtu  bardziej  znaczacego
        SETB TR1        ;Start  licznika  T1
        RETI            ;Wyjscie  z  przerwania
```

4.4. Licznik T2

4.4.1. Opis licznika T2

Licznik **T2** jest jednym z najbardziej rozbudowanych ukladów wewnetrznych mikrokontrolera **SAB 80C535**. Z tego tez wzgledu ma bardzo duzo mozliwosci funkcjonalnych, zawartych w trzech trybach pracy, okreslanych skrótem **CCR:**

CCR – Compare (porównanie),

C**C**R – Capture (zapamietanie wartosci chwilowej),

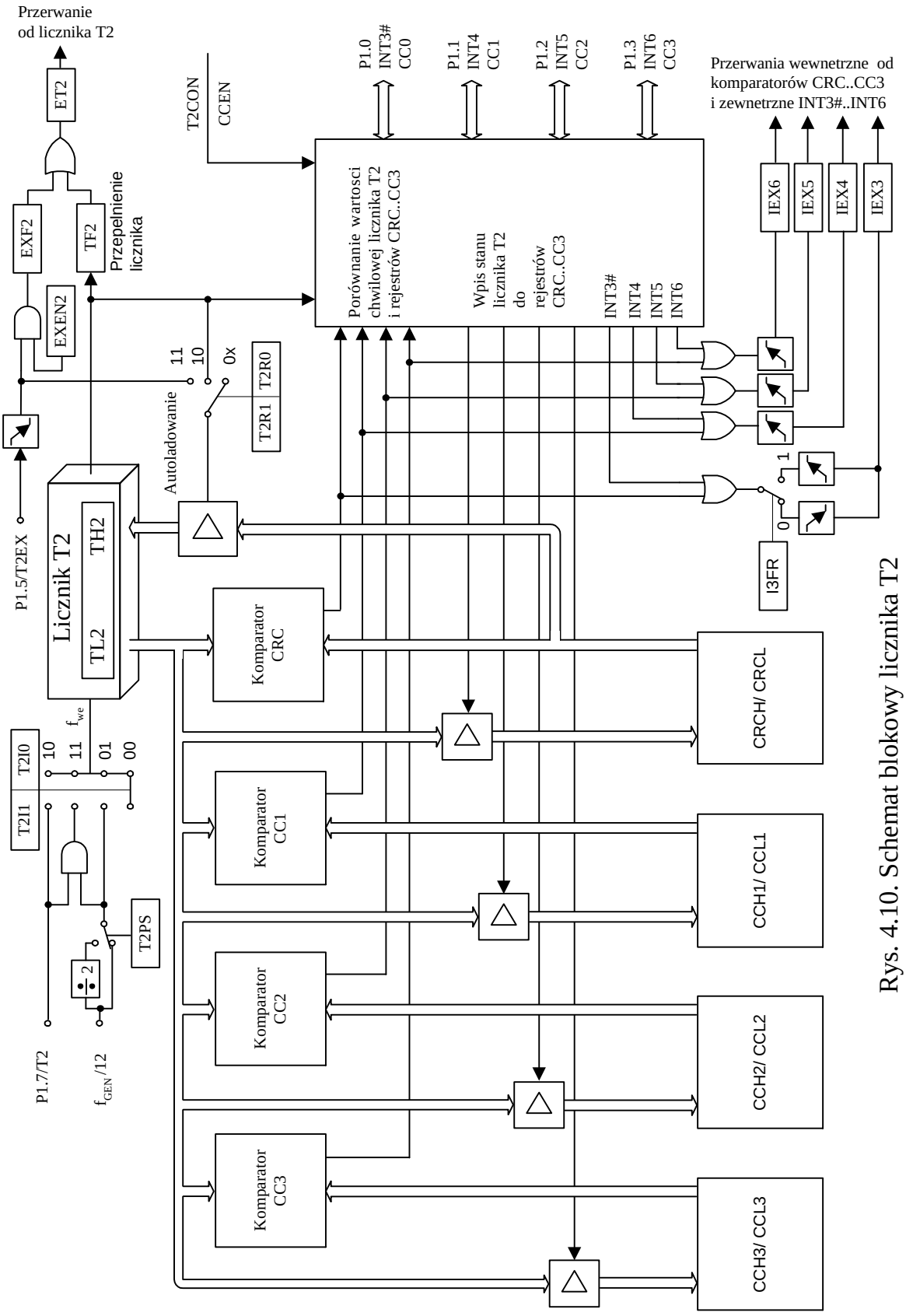
CC**R** - Reload (autoladowanie wartosci poczatkowej).

Schemat blokowy licznika **T2** przedstawiono na rys. 4.10. Szesnastobitowy licznik **T2** tworza dwa 8–bitowe rejestry: bardziej znaczacy **TH2** i mniej znaczacy **TL2.** Z obsluga licznika **T2** zwiazane sa linie portu **P1**, które w zaleznosci od funkcji licznika moga byc wykorzystane jako:

- Linie **P1.0**...**P1.3**:
  - wyjscia impulsów o modulowanej szerokosci sygnalu (MSI),
  - wejscia przerwan INT3#..INT6,
- Linia **P1.5/T2EX**:
  - sprzetowe ustalenie momentu wpisu wartosci poczatkowej do licznika **T2**.
- Linia **P1.7/T2**:
  - wejscie impulsów zewnetrznych,
  - wejscie bramkujace zliczanie impulsów wewnetrznych.

Jesli licznik **T2** nie jest wykorzystywany w trybie modulacji szerokosci impulsów, to linie portu **P1** moga byc wykorzystywane jako standardowe wejscia/wyjscia cyfrowe, dostepne programowo.

Z licznikiem **T2** zwiazane sa takze rejestry komparatorów **CRC**,**CC1**..**CC3**, w których przechowywane sa wartosci chwilowe (w trybie autoladowania w rejestrze **CRC** przechowywana jest wartosc poczatkowa). Ponadto, po kazdym przepelnieniu licznika **T2,** a takze w momencie zrównania sie wartosci chwilowej licznika z wartoscia wpisana do komparatora istnieje mozliwosc wygenerowania przerwania.



Rys. 4.10. Schemat blokowy licznika T2

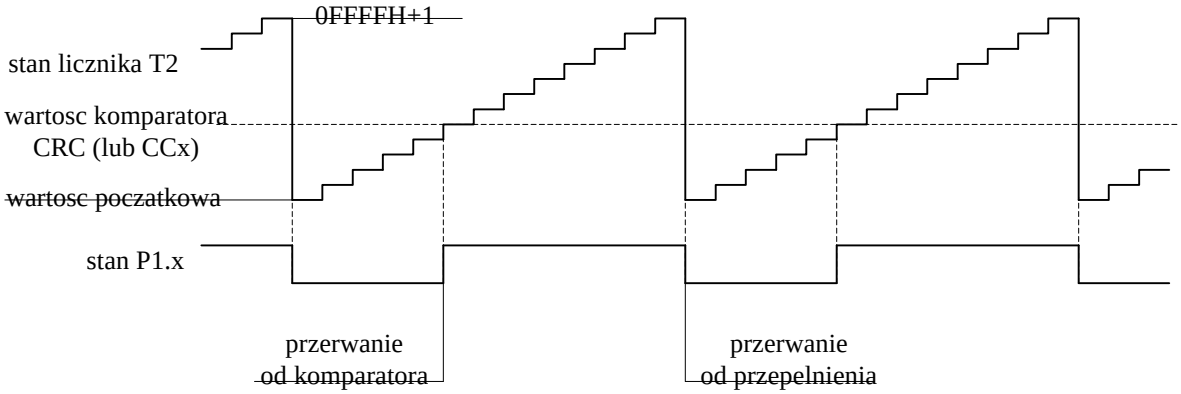
Ponizej opisano poszczególne tryby pracy licznika **T2**: porównania, zapamietania

wartosci chwilowej i autoladowania wartosci poczatkowej.

⇒ Tryby porównania (compare)

W czterech niezaleznych komparatorach porównywana jest wartosc rejestru licznika **T2** z wartosciami rejestrów **CRC**, **CC1**, **CC2** i **CC3**. Wynik porównania przedstawiony jest w postaci stanów linii portu **P1**, oraz ustawienia znaczników przerwan **IEX3**..**IEX6**. Linie **P1.0**, **P1.1**, **P1.2**, **P1.3** przyjmuja stan jedynki logicznej w momencie wystapienia równosci zawartosci rejestru licznika **T2** oraz zawartosci rejestrów **CRC**, **CC1**, **CC2**, **CC3** (w tym samym cyklu maszynowym). Zerowanie bitu nastepuje po przepelnieniu rejestru licznika **T2**.

Jednoczesnie w momencie ustawienia linii portu **P1** generowane sa przerwania i ustawiane odpowiednie znaczniki przerwan **IEX3**..**IEX6**. W momencie zerowania linii portu **P1** po przepelnieniu licznika **T2** ustawiany jest znacznik przerwania **TF2**. Znaczniki te znajduja sie w rejestrze **IRCON**, opisanym w rozdziale 4.6. Zaleznosci czasowe na wyjsciach portu **P1.0**..**P1.4** przedstawiono na rys. 4.11.



Rys. 4.11. Zaleznosci czasowe na wyjsciach portu P1

Jezeli do pracy licznika **T2** wykorzystywany jest tryb autoladowania, to do porównania wartosci chwilowej licznika **T2** zastosowane moga byc tylko trzy rejestry: **CC1**, **CC2** i **CC3,** poniewaz w rejestrze **CRC** przechowywana jest wartosc poczatkowa. W trybach porównania linie portu **P1** moga byc sterowane sprzetowo lub progra-

mowo. Ze wzgledu na sposoby sterowania liniami portu **P1** wyróznia sie dwa tryby porównania:

- **tryb 0 porównania**

W trybie 0 porównania linie portu **P1.0**...**P1.3** nadzorowane sa tylko przez uklady licznika **T2** i zmiana stanu tych linii nie jest mozliwa przez programowy wpis dowolnych wartosci, poniewaz wewnetrzna szyna danych jest zablokowana. Wlasciwosc nadzorowania linii portu **P1.0**...**P1.3** przez licznik **T2** wykorzystywana jest do generowania na tych liniach impulsów o modulowanym wspólczynniku wypelnienia. Oznacza to, ze programowo mozna ustalic okres i czas trwania stanu zera i jedynki logicznej na odpowiednich liniach portu **P1**. Zatem do dyspozycji sa cztery linie portu **P1**, którymi mozna sterowac np. silniki pradu stalego i przemiennego.

Z opisu trybu 0 porównania wynika, ze od wpisu do rejestru komparatora **CRC** zalezy czestotliwosc generowanych impulsów, natomiast wypelnienie regulowane jest wartoscia wpisana do komparatora **CCx**. Nalezy pamietac, ze minimalnej (0000H) i maksymalnej (0FFFFH) wartosci wpisywanej do rejestrów **CRC**, **CC1**..**CC3** towarzysza na wyjsciach portu **P1** impulsy szpilkowe o czasie trwania ½ cyklu maszynowego. Zjawisko to powoduje, ze niemozliwe jest uzyskanie zmiany wypelnienia impulsu w granicach 0...100%. Dla *n*–bitowego rejestru porównania **CCx** maksymalna wartosc wypelnienia wynosi $(1–½^n)*100\%$.

Zaprogramowany i uruchomiony licznik **T2** pracuje autonomicznie, tzn. generuje impulsy na wyjsciach portu **P1** niezaleznie od jednostki arytmetyczno–logicznej. Zatrzymanie pracy licznika jest mozliwe tylko sprzetowo poprzez reset mikrokontrolera linia *RESET#* lub odlaczenie zasilania.

- **tryb 1 porównania**

W trybie 1 porównania zmiana stanu linii portu **P1.0**...**P1.3** dokonywana jest programowo, przy czym dane wpisywane sa do 4–bitowego pomocniczego bufora portu. Dane te sa automatycznie przepisywane do glównego bufora portu **P1** w momencie

wystapienia równosci zawartosci licznika i rejestru **CCx**, przy czym przepisywanie z bufora pomocniczego do glównego dokonywane jest sprzetowo. Informacja z bufora pomocniczego pojawia sie na wyjsciu bufora glównego tak dlugo, jak dlugo wartosc chwilowa licznika równa jest zawartosci rejestru **CCx**. Nowa wartosc moze zostac wpisana do portu **P1** po ustapieniu tej równosci, w przeciwnym wypadku zmiany te nie sa wzajemnie synchronizowane, poniewaz zmianie ulega stan glównego bufora wyjsciowego portu. Czas trwania równosci wartosci licznika i rejestru **CCx** zalezy od sposobu taktowania licznika **T2**. Jezeli licznik taktowany jest sygnalem wewnetrznym, to stan równosci trwa 1 lub 2 cykle maszynowe, jesli natomiast licznik **T2** sterowany jest sygnalem zewnetrznym, to stan ten moze trwac przez wiele cykli maszynowych.

⇒ Tryby zapamietania wartosci chwilowej (capture)

Do zapamietania wartosci chwilowej licznika **T2** moze byc uzyty kazdy z rejestrów **CRC**, **CC1**..**CC3**. Operacja ta jest realizowana sprzetowo, dzieki czemu trakcie jej wykonywania nie nastepuje zatrzymanie zliczania. Licznik **T2** moze pracowac w dwóch trybach zapamietania wartosci chwilowej:

- **tryb 0 zapamietania**

W tym trybie moment zapamietania wartosci chwilowej licznika wyznaczony jest sygnalem zewnetrznym. Wpisanie wartosci chwilowej licznika **T2** do odpowiedniego rejestru **CRC**,**CC1**...**CC3** oraz wpisanie jedynki na pozycje odpowiedniego znacznika przerwania **IEX3**...**IEX6** spowodowane jest aktywnym zboczem sygnalów **INT3#**...**INT6**. Ze wzgledu na sposób sterowania sygnalami zewnetrznymi aktywnym zboczem moze byc:

– zbocze narastajace dla wejsc przerwan INT4...INT6 zwiazanych z rejestrami **CC1**...**CC3,**

– zbocze narastajace lub opadajace dla wejscia INT3# zwiazanego z rejestrem **CRC**.

Nalezy pamietac, aby do bufora wyjsciowego portu **P1** wpisana byla wartosc jedynki logicznej, w przeciwnym razie niemozliwe byloby okreslenie momentu pojawie-

nia sie aktywnego zbocza sygnalu.

- **tryb 1 zapamietania**

W trybie 1 zapamietanie wartosci chwilowej licznika **T2** w rejestrach **CRC**, **CC1**...**CC3** spowodowane jest wpisaniem dowolnej wartosci do mniej znaczacych rejestrów **CRCL**, **CCL1**...**CCL3**. W tym trybie zapamietania przerwania nie sa wykorzystywane.

⇒ **Tryby autoladowania licznika T2 (reload)**

Autoladowanie licznika **T2** oznacza przepisanie wartosci poczatkowej, zawartej w 16–bitowym rejestrze **CRC** do rejestru licznika **T2**. Rejestr **CRC** zlozony jest z dwóch 8–bitowych rejestrów **CRCL** i **CRCH**, podobnie 16-bitowy rejestr licznika **T2** tworza dwa 8–bitowe rejestry **TH2** i **TL2**. Wyróznia sie dwa tryby autoladowania:

- **tryb 0 autoladowania**

W trybie **0** autoladowania wpis wartosci poczatkowej z rejestru **CRC** do licznika **T2** dokonywany jest wskutek przepelnienia licznika **T2**, czyli osiagniecia przez licznik wartosci 0FFFFH+1. Przepelnienie licznika powoduje ustawienie flagi przerwania **TF2** (znacznik ten przyjmuje wartosc jedynki logicznej).

- **tryb 1 autoladowania**

W trybie **1** autoladowania przepisanie wartosci poczatkowej z rejestru **CRC** do rejestru licznika **T2** dokonywane jest wskutek wystapienia opadajacego zbocza sygnalu **T2EX** na wejsciu **P1.5**. Jesli znacznik **EXEN2** w rejestrze **IEN1** ma wartosc jedynki logicznej, to równiez w tym trybie zostanie wygenerowane przerwanie.

4.4.2. Sterowanie praca licznika T2

Sygnal taktujacy licznik **T2** moze pochodzic ze zródla zewnetrznego lub we-

wnetrznego. Programowo ustala sie czestotliwosc i sposób doprowadzenia sygnalu wewnetrznego lub sygnalu zewnetrznego przez ustawienie 8 bit portu **P1** (**P1.7**).

Niezaleznie od wybranego trybu pracy licznika **T2**, po kazdym jego przepelnieniu, tzn. po osiagnieciu stanu 0FFFFH+1, jest generowany sygnal przerwania wewnetrznego. Jest to sygnalizowane wpisaniem jedynki logicznej na pozycji znacznika **TF2** (wektor przerwania 02BH) w rejestrze **IRCON** (patrz rozdzial 4.6). Znacznik ten jest kasowany programowo, poniewaz to samo przerwanie moze byc równiez wywolane zewnetrznym sygnalem doprowadzonym do wejscia **P1.5/T2EX**, przy czym ustawiany jest wówczas znacznik **EXF2** (wektor przerwania 02BH).

Z konfiguracja licznika **T2** zwiazanych jest 13 rejestrów specjalnych **SFR**, przedstawionych w tabeli 4.3.

Tabela 4.3. Rejestry SFR zwiazane z licznikiem T2

Rejestr	**Adres**	**Funkcja**
TH2	0CDH	8 bardziej znaczacych bitów licznika T2
TL2	0CCH	8 mniej znaczacych bitów licznika T2
T2CON	0C8H	Programowanie trybów pracy, zródla i czestotliwosci taktowania licznika oraz startu licznika
CCEN	0C1H	Wybór trybów pracy rejestrów CRC, CC1, CC2, CC3
CRCH	0CBH	8 bardziej znaczacych bitów rejestru CRC
CRCL	0CAH	8 mniej znaczacych bitów rejestru CRC
CCH1	0C3H	8 bardziej znaczacych bitów rejestru CC1
CCL1	0C2H	8 mniej znaczacych bitów rejestru CC1
CCH2	0C5H	8 bardziej znaczacych bitów rejestru CC2
CCL2	0C4H	8 mniej znaczacych bitów rejestru CC2
CCH3	0C7H	8 bardziej znaczacych bitów rejestru CC3
CCL3	0C6H	8 mniej znaczacych bitów rejestru CC3
IRCON	0C0H	Rejestr znaczników obslugujacych przerwania zewnetrzne lub od komparatorów

Programowanie pracy licznika **T2** dokonywane jest w rejestrze **T2CON** przez odpowiednie ustawienie poszczególnych bitów, przy czym wszystkie znaczniki po wyzerowaniu mikrokontrolera równiez przyjmuja wartosci zerowe.

Rejestr **T2CON** adres **0C8H**

T2PS	I3FR	**I2FR**	T2R1	T2R0	T2CM	T2I1	T2I0

- **T2PS** – wlaczenie dodatkowego dzielnika wstepnego dla wewnetrznego zródla taktowania:

  T2PS = 0 – taktowanie licznika sygnalem f_{osc} /12,

  T2PS = 1 – taktowanie licznika sygnalem f_{osc} /24.

- **I3FR** – wybór aktywnego zbocza sygnalu przerwania zewnetrznego INT#3, wyjscia komparatora CRC, wpisu wartosci poczatkowej:

  I3FR = 0 – zbocze opadajace,

  I3FR = 1 – zbocze narastajace.

- **I2FR** – w programowaniu licznika **T2** bit nie uzywany.

- **T2R1**, **T2R0** – wybór trybu autoladowania licznika:

  T2R1 = 0, T2R1 = x – zablokowane funkcje autoladowania,

  T2R1 = 1, T2R0 = 0 – tryb 0 autoladowania,

  T2R1 = 1, T2R0 = 1 – tryb 1 autoladowania.

- **T2CM** – wybór trybu porównania:

  T2CM = 0 – tryb 0,

  T2CM = 1 – tryb 1.

- **T2I1**, **T2I0** – wybór zródla sygnalu taktujacego:

  T2I1 = 0, T2I0 = 0 – zatrzymanie licznika,

  T2I1 = 0, T2I0 = 1 – taktowanie sygnalem wewnetrznym f_{osc}/12 lub f_{osc}/24,

  T2I1 = 1, T2I0 = 0 – taktowanie sygnalem zewnetrznym,

  T2I1 = 1, T2I0 = 1 – bramkowanie wewnetrznego sygnalu taktujacego (blokowanie licznika niskim poziomem sygnalu doprowadzonego do wejscia P1.7).

Programowanie trybów pracy rejestrów **CRC,CC1...CC3** jest dokonywane w rejestrze **CCEN**. Nalezy zwrócic uwage, ze w odróznieniu od trybu porównania mozliwe jest jednoczesne wybranie trybu 0 wpisu wartosci poczatkowej licznika **T2**

dla jednych rejestrów **CCR**, **CC1**...**CC3** i trybu 1 dla pozostalych. Ponadto istotne jest, ze przy zerowaniu mikrokontrolera linia ***RESET#*** zawartosc rejestru **CCEN** równiez jest zerowana. Przedstawiono zawartosc rejestru **CCEN** , a przeznaczenie poszczególnych znaczników rejestru opisano w tabeli 4.4.

Rejestr **CCEN** adres **0C1H**

COCAH3	COCAL3	COCAH2	COCAL2	COCAH1	COCAL1	COCAH0	COCAL0

Tabela 4.4. Sterowanie komparatorami

Rejestr	Znacznik		Przeznaczenie
CRC	**COCAH0**	**COCAL0**	
	0	0	zablokowany tryb porównania / wpisu wartosci poczatkowej
	0	1	tryb 0 zapamietania
	1	0	odblokowany tryb porównania / wpisu wartosci poczatkowej
	1	1	tryb 1 zapamietania
CC1	**COCAH1**	**COCAL1**	
	0	0	zablokowany tryb porównania / wpisu wartosci poczatkowej
	0	1	tryb 0 zapamietania
	1	0	odblokowany tryb porównania / wpisu wartosci poczatkowej
	1	1	tryb 1 zapamietania
CC2	**COCAH2**	**COCAL2**	
	0	0	zablokowany tryb porównania / wpisu wartosci poczatkowej
	0	1	tryb 0 zapamietania
	1	0	odblokowany tryb porównania / wpisu wartosci poczatkowej
	1	1	tryb 1 zapamietania

	COCAH3	COCAL3	
CC3	0	0	zablokowany tryb porównania / wpisu wartosci poczatkowej
	0	1	tryb 0 zapamietania
	1	0	odblokowany tryb porównania / wpisu wartosci poczatkowej
	1	1	tryb 1 zapamietania

4.4.3. Przyklady programowania licznika T2

Przyklad 1

```
;*********************************************************************
;Przyklad wykorzystania licznika T2 do generowania sygnalu PWM. Przetwornik a/d
;obsluguje kanal pomiarowy AN1. W przerwaniach od T2 jest modyfikowana wartosc
;CCL2 i CCH2. Na wyjsciu P1.2 generowany jest sygnal PWM
;*********************************************************************

RAM_1     BYTE ($20)      ;Deklaracja adresu w pamieci wewnetrznej
EPR_1     EQU  $0100      ;Deklaracja adresu w pamieci zewnetrznej

;**********************
;Start programu
;**********************
      ORG  $00            ;Adres poczatku programu
      LJMP MAIN           ;Skok do programu glównego
;***************************
;Podprogram obslugi przerwania
;***************************
      ORG  $002B          ;Adres obslugi przerwania od T2

      LJMP MODUL          ;Skok do programu wykonania przerwania

;*******************************
;Podprogram wykonania przerwania
;*******************************
      ORG  $0050                ;Adres programu wykonania przerwania
MODUL: CLR TF2                  ;Kasowanie znacznika przerwania od T2
      MOV  DPTR,#$100           ;Adresowanie pamieci zewnetrznej
      MOVX A,@DPTR              ;Przeslanie wyniku do akumulatora
      ADD  A,#$17               ;Sumowanie z wartoscia poczatkowa
      MOV  CCL2,A               ;Modyfikacja rejestru CCL2
```

```
        INC   DPTR              ;Adresowanie kolejnego rejestru pamieci
        MOVX A,@DPTR            ;Przeslanie wyniku do akumulatora
        ADDC A,#$FC             ;Sumowanie z wartoscia poczatkowa
        MOV  CCH2,A             ;Modyfikacja rejestru CCH2
        SETB C                  ;Ustawienie flagi C oznaczajacej koniec
        RETI                    ; modyfikacji

;************************************
;Podprogram konfiguracji licznika T2
;************************************
        ORG  $0200              ;Adres programu  konfiguracji licznika T2

MAIN: MOV  CCEN,#%00100000      ;Uaktywnienie trybu porównania dla CC2
        ANL  T2CON,#%00100000   ;Zerowanie rejestru T2CON
        ORL  T2CON,#%00010001   ;Autoladowanie po przepelnieniu licznika T2
        MOV  CRCH,#$FC          ;Wpis wartosci poczatkowej do rejestru
                                ;CRCH
        MOV  CRCL,#$17          ;Wpis wartosci poczatkowej do rejestru
                                ;CRCL
        MOV  CCH2,#$FE          ;Wpis wartosci poczatkowej do rejestru
                                ;CCH2
        MOV  CCL2,#$17          ;Wpis wartosci poczatkowej do rejestru
                                ;CCL2
        SETB EAL                ;Uaktywnienie wszystkich przerwan
        SETB ET2                ;Uaktywnienie przerwania od licznika T2

LOOP: LCALL  PAD                ;Wywolanie programu obslugi
                                ;przetworników AD
        CLR  C                  ;Zerowanie flagi C
ALA:  JNC   ALA                 ;Oczekiwanie na zakonczenie programu
                                ;obslugi przerwania
        SJMP LOOP

;*********************************************************
; Program obslugi przetworników AD w SAB80C535
; Wynik (10 bitów) w $100 i $101 pamieci zewnetrznej
;*********************************************************
        ;ADCON    $D8        rejestr konfiguracji przetwornika
        ;ADDAT    $D9        rejestr wyniku przetwarzania anologowo-cyfrowego
        ;DAPR     $DA        rejestr programowania napiec wzorcowych
        ORG  $1000           ;Adres programu obslugi przetworników

PAD:  MOV  A,ADCON
        ANL  A,#$F0          ;Zerowanie mniej znaczacych bitów
        ORL  A,#$01          ;Wybór trybu przetwarzania i kanalu
```

```
        MOV  ADCON,A          ; pomiarowego
        MOV  DAPR,#$00        ;Wybór zakresu pomiarowego , start
                              ;przetwarzania

LOOP1: JB   ADCON.4,LOOP1  ;Oczekiwanie na zakonczenie przetwarzania
        MOV A,ADDAT           ;Wpis wyniku do akumulatora
        ANL  A,#$C0           ;Wyzerowanie 6 bitów mniej znaczacych
        SWAP A                ;Konwersja czterech bitów
        RR     A
        RR     A
        MOV  RAM_1,A          ;Przeslanie 2 bitów do pamieci wewnetrznej
        MOV  DPTR,#CONV       ;Adresowanie tablicy zakresów
        MOVC A,@A+DPTR        ;Pobranie elementu tablicy
        MOV  DAPR,A           ;Wybór nowego podzakresu, start przetwarzania
LOOP2: JB ADCON.4,LOOP2    ;Oczekiwanie na zakonczenie przetwarzania
        MOV  A,ADDAT          ;Wpis wyniku do akumulatora
        MOV  DPTR,#EPR_1      ;Adresowanie pamieci zewnetrznej
        MOVX @DPTR,A          ;Wyslanie wyniku do pamieci zewnetrznej
        INC    DPTR           ;Adresowanie kolejnego rejestru
        MOV  A,RAM_1          ;Wyslanie wartosci 2 bitowej do pamieci
        MOVX @DPTR,A          ; zewnetrznej
        RET
CONV:       DB     #$40,#$84,#$C8,#$FC ;Tablica zakresów pomiarowych
```

Przyklad 2

```
;*********************************************************************
;Program zmiany współczynnika wypelnienia impulsu PWM przez modyfikacje
;rejestrów komparatora wartosciami z tablicy.
;*********************************************************************
        ORG  $00
        LJMP MAIN             ;Skok do programu konfiguracji licznika T2

;***************************************
;Modyfikacja rejestru komparatora CCL2
;***************************************
        ORG  $100
LOOP_1:     CJNE R6,#$00,LOOP_2   ;Licznik powtórzen  modyfikacji , skok
                                  ;do procedury  opóznienia  czasowego
        MOV  DPTR,#TAB            ;Adresowanie tabeli
        MOV  R6,#$18              ;Wpis liczby elementów tabeli do
                                  ;licznika powtórzen
LOOP_2:     MOV  R0,#$03          ;Procedura opóznienia czasowego,
```

```
 LOOP_5:   MOV  R1,#$FF          ;wartosc opóznienia do rejestrów R0,
LOOP_4:    MOV  R2,#$FF         ;R1 i R2
LOOP_3:    DJNZ R2,LOOP_3       ;Dekrementowanie rejestru R2
           DJNZ R1,LOOP_4       ;Dekrementowanie rejestru R1
           DJNZ R0,LOOP_5       ;Dekrementowanie rejestru R0
           ANL  A,#$00          ;Zerowanie akumulatora
           MOVC A ,@A+DPTR      ;Pobranie z tabeli wartosci  wspólczynnika
           MOV  CCL2,A          ;wypelnienia,  modyfikacja rejestru CCL2
           INC  DPTR            ;Adresowanie kolejnego elementu tabeli
           DEC  R6              ;Dekrementowanie licznika powtórzen
           LJMP LOOP_1          ;Skok do poczatku programu modyfikacji
                                ;rejestru  komparatora

;*************************************
;Konfiguracja licznika T2
;*************************************
           ORG  $200

MAIN:  MOV  CCEN,#$20      ;Odblokowanie komparatora CC2
           ANL  T2CON,#$20      ;Zerowanie rejestru sterujacego licznika T2
           ORL  T2CON,#$11      ;Wybór sygnalu taktujacego, trybu autoladowania
                                ;i trybu porównania , start licznika T2
           MOV  CRCH,#$FF       ;Wartosc  poczatkowa licznika T2 po
           MOV  CRCL,#$00       ;przepelnieniu , czestotliwosc  sygnalu  PWM
           MOV  CCH2,#$FF       ;Wartosc  porównania  rejestru  komparatora,
           MOV  CCL2,#$01       ;zmiana  wspólczynnika  wypelnienia  sygnalu PWM
           MOV  R6,#$18         ;Liczba  powtórzen  modyfikacji
           LJMP LOOP_1          ;Skok  do programu  modyfikacji  wypelnienia

TAB:  DB   #05,#10,#20,#30,#40,#50,#60,#70,#80,#90,#100,#110
           DB   #120,#130,#140,#150,#160,#170,#180,#190,#200,#210
           DB   #220,#230,$240,
```

4.5. Przetwornik analogowo - cyfrowy

4.5.1. Opis przetwornika a/c

Przetwornik a/c jest wewnetrznym ukladem mikrokontrolera **SAB 80C535** przeznaczonym do przetwarzania sygnalów z postaci analogowej na cyfrowa 8–bitowa. Istnieje mozliwosc zwiekszenia rozdzielczosci przetwornika do 10 bitów dzieki programowalnemu zawezeniu zakresu pomiarowego przetwornika.

Istnieje kilka metod przetwarzania analogowo-cyfrowego. W mikrokontrolerach **SAB 80C535** zamiana wielkosci analogowej mierzonego napiecia na odpowiadajaca jej wielkosc cyfrowa odbywa sie metoda kompensacji wagowej, przy czym jako element kompensujacy zastosowano matryce kondensatorów. W metodzie tej napiecie mierzone U_x porównywane jest z napieciem referencyjnym $U_{REF.}$ W pierwszym cyklu porównania (i=1) napiecie referencyjne U_{REF} przyjmuje wartosc $\Delta U_1 = \frac{U_{REF\,max}}{2}$, a nastepnie porównywane jest z napieciem U_x. Dopóki jest spelniona nierównosc $U_x < U_{REF}$, w kazdym nastepnym cyklu nastepuje zwiekszenie napiecia referencyjnego U_{REF} o wartosc $\Delta U_i = \frac{U_{REF\,max}}{2^i}$. Charakterystyczny dla tej metody jest fakt, ze czas przetwarzania jest staly i nie zalezy od wartosci mierzonego napiecia, a liczba cykli porównania równa jest liczbie bitów przetwarzania.

Schemat blokowy przetwornika a/c przedstawiono na rys. 4.12.

Wejscie przetwornika a/c stanowi 8–wejsciowy analogowy multiplekser umozliwiajacy pomiar napieciowego sygnalu analogowego w 8 róznych kanalach w zakresie od 0 do 5V.

Przetwornik a/c wyróznia sie nastepujacymi parametrami:

- 8 kanalów analogowych AN0...AN7,
- 8–bitowy port wejsc cyfrowych P6,
- 8–bitowa rozdzielczosc pomiaru (10-bitowa przy programowym zawezeniu zakresu pomiarowego),
- 16 programowalnych podzakresów pomiarowych,
- programowe wyzwalanie pojedynczego pomiaru lub serii pomiarów,
- mozliwosc generowania przerwania po kazdym pomiarze,
- czas przetwarzania przetwornika wynosi 13 cykli maszynowych.

Przetwornik analogowo−cyfrowy korzysta z 3 rejestrów specjalnych **SFR**:

⇒ **ADCON** (adres 0D8H) – wybór kanalu pomiarowego i rodzaju przetwarzania.

Rejestr **ADCON** adres **0D8H**

BD	**CLK**	**-**	BSY	ADM	MX2	MX1	MX0

Znaczenia poszczególnych bitów sa nastepujace:

- **BD** – bit uzywany do okreslenia szybkosci transmisji lacza szeregowego. Zmiana stanu tego bitu moze powodowac zawieszenie systemu, dlatego podczas pracy przetwornika nie nalezy modyfikowac jego wartosci.
- **CLK** - umozliwia uzyskanie na wyprowadzeniu procesora *CLKOUT* (P1.6) sygnalu o czestotliwosci $f_{CLKOUT} = f_{osc}/12$ i wypelnieniu 1/6, przy czym f_{osc} jest czestotliwoscia zewnetrznego rezonatora kwarcowego dolaczonego do mikrokontrolera. Podobnie jak dla bitu **BD** nie modyfikuje sie jego wartosci podczas pracy przetwornika.
- **BSY** – wskazuje aktualny stan przetwornika. Jest znacznikiem o atrybucie tylko do odczytu, ustawianym i kasowanym sprzetowo.

  BSY=0 – przetwornik po zakonczeniu przetwarzania,

  BSY=1 – przetwornik w trakcie przetwarzania.
- **ADM** – okresla rodzaj przetwarzania.

  ADM=1 – przetwornik w trybie pracy wielokrotnego pomiaru. Przetwornik wykonuje serie pomiarów, az do chwili gdy ADM=0.

  ADM=0 – pomiar pojedynczy, przy czym kazdy pomiar musi byc uruchamiany programowo.
- **MX2...MX0** – wybór numeru kanalu pomiarowego (patrz tabela 4.5).

Numer kanalu pomiarowego okreslany jest przez wpisanie odpowiedniej kombinacji bitów na pozycje znaczników MX2...MX0. Sposób okreslania kanalu pomiarowego przedstawiono w tabeli 4.5.

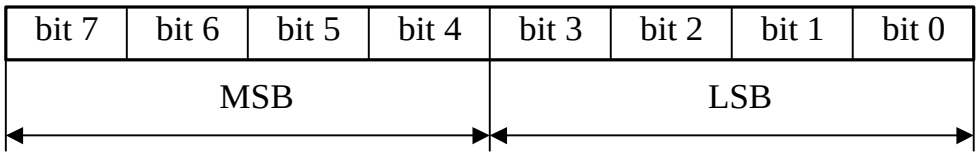
Tabela 4.5. Wybór numeru kanalu pomiarowego

MX2	MX1	MX0	Kanal	Wejscie
0	0	0	0	AN0/P6.0
0	0	1	1	AN1/P6.1
0	1	0	2	AN2/P6.2
0	1	1	3	AN3/P6.3
1	0	0	4	AN4/P6.4
1	0	1	5	AN5/P6.5
1	1	0	6	AN6/P6.6
1	1	1	7	AN7/P6.7

Nalezy pamietac, ze wejscia AN0...AN7 sa jednoczesnie wejsciami portu cyfrowego P6.

⇒ **ADDAT** (adres 0D9H) - modyfikowany wynik przetwarzania,

Rejestr **ADDAT** adres **0D9H**



MSB – najbardziej znaczacy bit,

LSB - najmniej znaczacy bit.

W rejestrze **ADDAT** przechowywana jest calkowita wielokrotnosc poziomów kwantowania :

$$\text{ADDAT} = \frac{U_x}{1\text{LSB}},$$

przy czym 1 LSB oznacza najmniejszy poziom kwantowania:

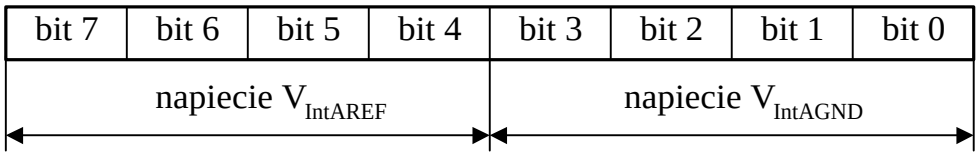
$$1\ \text{LSB} = \frac{V_{\text{IntAREF}} - V_{\text{IntAGND}}}{2^8}$$

Jezeli podczas pracy mikrokontrolera nie wykorzystujemy przetwornika a/c, to rejestr **ADDAT** moze sluzyc jako rejestr ogólnego przeznaczenia.

⇒ **DAPR** (adres 0DAH) – wybór zakresu przetwarzania. Wybór zakresu rozpoczyna

pomiar i przetwarzanie a/c na wybranym kanale.

Rejestr **DAPR** adres **0DAH**



przy czym:

napiecie $\boldsymbol{V_{IntAREF}}$ − górne wewnetrzne napiecie odniesienia,

napiecie $\boldsymbol{V_{IntAGND}}$ − dolne wewnetrzne napiecie odniesienia.

Napiecia te tworzone sa przez podzial róznicy napiec $\boldsymbol{V_{AREF}}$ i $\boldsymbol{V_{GND}}$ zgodnie ze wzorami:

$$V_{IntAREF} = V_{AGND} + \frac{DAPR_{7...4}}{16} (V_{AREF} - V_{AGND})$$

gdy $DAPR_{7...4} > 3H$

$$V_{IntAGND} = V_{AGND} + \frac{DAPR_{3...0}}{16} (V_{AREF} - V_{AGND})$$

gdy $DAPR_{3...0} < 0CH$

Wyboru podzakresu pomiarowego dokonuje sie wg tabeli 4.6 przy zalozeniu, ze wartosci napiec odniesienia sa nastepujace:

V_{AREF} = +5 V,

V_{AGND} = 0 V.

Od wlasciwego doboru podzakresu pomiarowego zalezy dokladnosc przetwarzania, uzyskiwana przez uklad przetwornika. Przy wyborze podzakresu pomiarowego nalezy zwrócic uwage, aby graniczne wartosci podzakresu (tzn. ani napiecie $\boldsymbol{V_{IntAREF}}$ ani $\boldsymbol{V_{IntAGND}}$) nie pokrywaly sie z wartoscia napiecia mierzonego, gdyz prowadzi to do bledów pomiarowych.

Tabela 4.6. Wybór podzakresu pomiarowego

Podzakres pomiarowy	**$DAPR_{7...4}$**	**$V_{IntAREF}$ [V]**	**$DAPR_{3...0}$**	**$V_{IntAGND}$ [V]**

0	0000	5,0	0000	0,0
1	0001	–	0001	0,3125
2	0010	–	0010	0,625
3	0011	–	0011	0,9375
4	0100	1,25	0100	1,25
5	0101	1,5625	0101	1,5625
6	0110	1,875	0110	1,875
7	0111	2,1875	0111	2,1875
8	1000	2,5	1000	2,5
9	1001	2,8125	1001	2,8125
10	1010	3,125	1010	3,125
11	1011	3,4375	1011	3,4375
12	1100	3,75	1100	3,75
13	1101	4,0625	1101	–
14	1110	4,375	1110	–
15	1111	4,6875	1111	–

Przy pomiarze na pelnym zakresie pomiarowym 0V...5V dokladnosc przetwarzania przetwornika wynosi:

$$1\ \text{LSB} = \frac{V_{\text{IntAREF}} - V_{\text{IntAGND}}}{2^8} = \frac{5-0}{256} = 0{,}019531\ \text{V},$$

natomiast przy pomiarze na zakresie zawezonym do 1,25V

$$1\ \text{LSB} = \frac{V_{\text{IntAREF}} - V_{\text{IntAGND}}}{2^8} = \frac{1{,}25-0}{256} = 0{,}004883\ \text{V}.$$

Z obliczen wynika prosty sposób na zwiekszenie dokladnosci przetwornika o dodatkowe dwa bity. Nalezy tylko dokonac dwóch pomiarów: pierwszego, na pelnym zakresie pomiarowym dla okreslenia dwóch najbardziej znaczacych bitów wyniku oraz okreslenia nowego podzakresu pomiarowego, oraz drugiego pomiaru na zawezonym podzakresie pomiarowym dla uzyskania 8 mniej znaczacych bitów wyniku przetwarzania.

Poniewaz rejestr **ADDAT** jest modyfikowany po kazdym cyklu pomiarowym, nalezy zwrócic uwage na to, aby podczas pomiaru z dokladnoscia 10-bitowa przed do-

konaniem drugiego pomiaru na zawezonym zakresie zapamietac aktualna zawartosc rejestru w innym, nie modyfikowanym podczas pracy przetwornika rejestrze ogólnego przeznaczenia (lub w komórce pamieci).

Do prawidlowej pracy przetwornika a/c wymagane jest spelnienie nastepujacych warunków:

- napiecia $V_{IntAREF}$ i $V_{IntAGND}$ musza byc dolaczone do wyprowadzen mikrokontrolera. Obecnosc tych napiec nie jest konieczna, gdy przetwornik nie pracuje,
- jezeli napiecia $V_{IntAREF}$ i $V_{IntAGND}$ sa dolaczone do napiec zasilajacych mikrokontroler $\boldsymbol{V_{CC}}$ i $\boldsymbol{V_{SS}}$, to musza spelniac warunek:

  $V_{IntAREF} = V_{CC} \pm 5\%$,

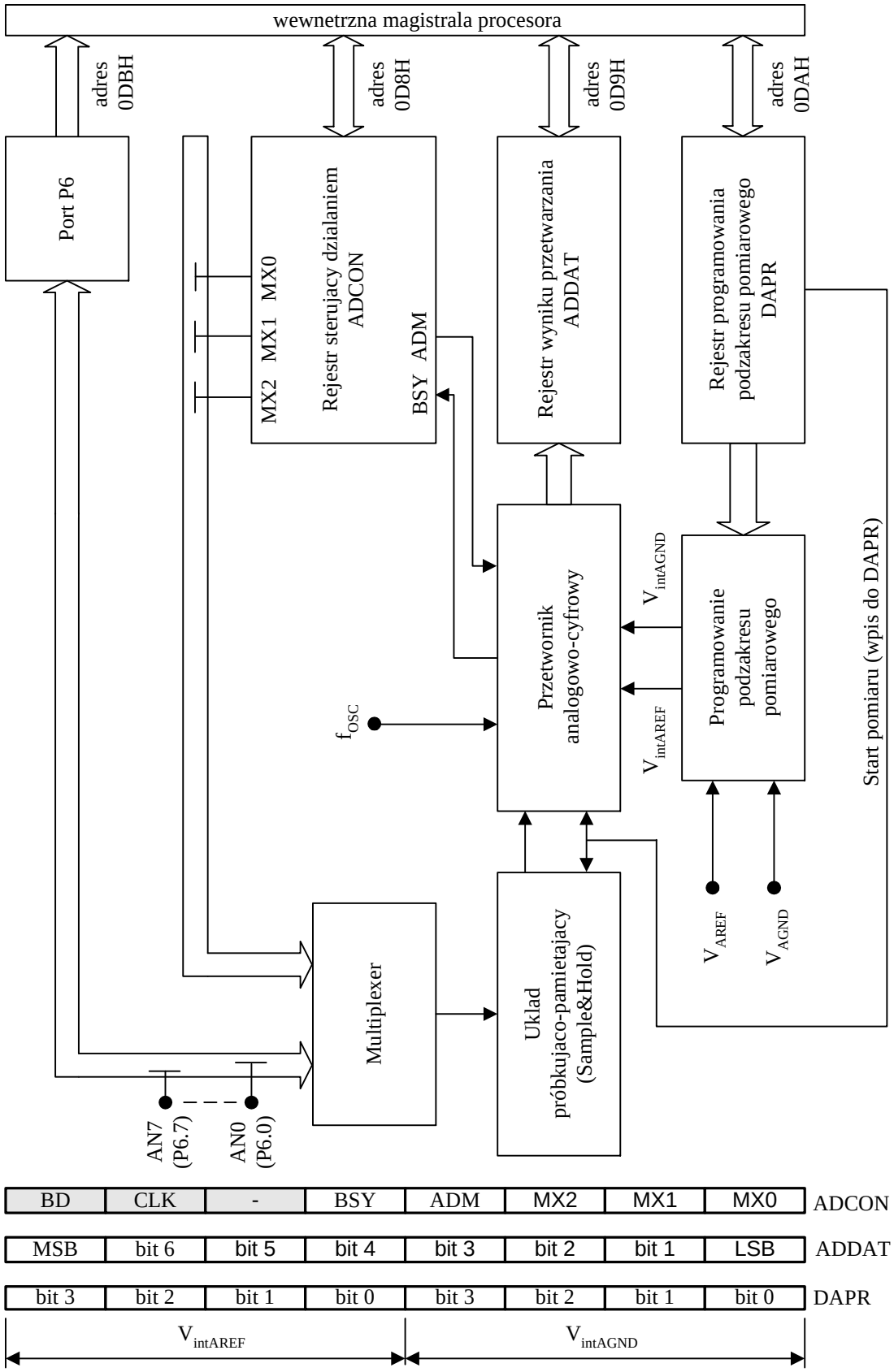
  $V_{IntAGND} = V_{SS} \pm 0{,}2$ V,
- minimalna róznica napiec $(V_{IntAREF} - V_{IntAGND}) \geq 1$V,
- rezystancja wewnetrzna zródla mierzonego napiecia i napiecia wzorcowego nie moze byc wieksza niz 5 kΩ.

Po kazdym wykonanym pomiarze przetwornik ma mozliwosc generowania przerwania. Ustawiany jest wówczas bit **IADC** w rejestrze **IRCON**, przy czym wektor przerwania ma adres 43H.

Rejestr **IRCON** adres **0C0H**

EXF2	TF2	IEX6	IEX5	IEX4	IEX3	IEX2	**IADC**

Znaczenie pozostalych bitów rejestru **IRCON** opisano w rozdziale 4.6.1



Rys. 4.6. Schemat blokowy przetwornika a/c

4.5.2. Pomiar z dokladnoscia 10-bitowa

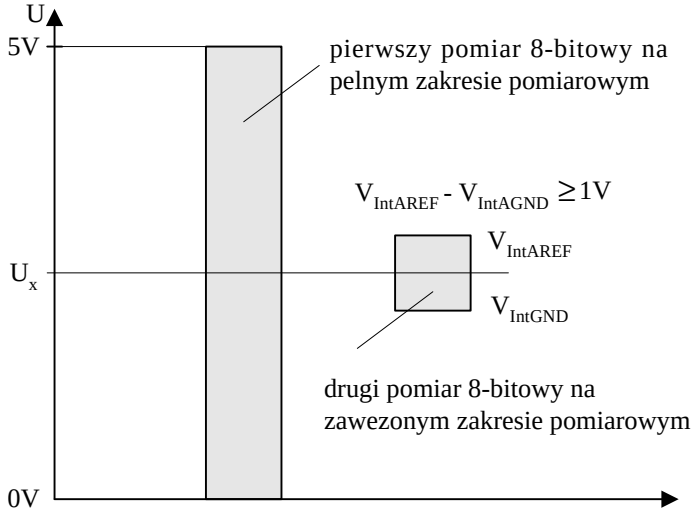
Jezeli istnieje potrzeba pomiaru z dokladnoscia 10 bitów, to pomiaru napiecia U_x zawartego w przedziale 0...5V nalezy dokonac dwukrotnie. Pierwszy pomiar odbywa sie przy pelnym zakresie pomiarowym 0...5V. Na podstawie wyniku pomiaru dokonywany jest dobór wlasciwego podzakresu pomiarowego do wykonania drugiej operacji przetwarzania a/c. Przy doborze podzakresu nalezy spelnic warunki:

– $V_{IntAREF} - V_{IntAGND} \geq 1V$,
– mierzone napiecie nie moze znajdowac sie na granicy podzakresu, poniewaz wynik drugiego pomiaru moze zostac zafalszowany.

Sposób pomiaru z rozdzielczoscia 10–bitowa przedstawiono na rys. 4.13.

Dla lepszego zrozumienia algorytm doboru podzakresu pomiarowego ograniczono do 4 podzakresów pomiarowych: 0–1,25 V; 1,25–2,5 V; 2,5V–3,75 V; 3,75–5 V.

Dobór podzakresu pomiarowego dokonywany jest na podstawie analizy dwóch najbardziej znaczacych bitów wyniku uzyskanego z pierwszego pomiaru (poniewaz na dwóch bitach mozna zapisac cztery rózne cyfry, co odpowiada czterem róznym podzakresom pomiarowym).



Rys. 4.13. Dobór zakresu pomiarowego

Jezeli dwa najbardziej znaczace bity maja wartosc 00, to wynik pomiaru miesci sie w przedziale 6 mniej znaczacych bitów, czyli moze przyjmowac wartosci $0...64_d$

(0...3FH), stad mierzone napiecie znajduje sie w zakresie:

$$(0...64)*\frac{5}{256} = 0...1{,}25\ \text{V}$$

Nalezy zatem do drugiego pomiaru wybrac zakres 0..1.25 V, czyli do rejestru **DAPR** wpisac wartosc 01000000_b = 40H.

Jezeli dwa najbardziej znaczace bity maja wartosc 01, to wynik pomiaru miesci sie w przedziale 7 mniej znaczacych bitów, czyli moze przyjac wartosci $64...128_d$ (40H...80H), a mierzone napiecie zmienia sie w zakresie:

$$(64...128)*\frac{5}{256} = 1{,}25...2{,}5\ \text{V},$$

do drugiego pomiaru nalezy zatem wybrac zakres 1,25...2,5 V, czyli do rejestru **DAPR** wpisac wartosc 10000100_b = 84H.

Jezeli dwa najbardziej znaczace bity maja wartosc 10, to wynik pomiaru miesci sie w przedziale 8 bitów, czyli przyjmuje wartosci $128...192_d$ (80H...0C0H). Mierzone napiecie przyjmuje zatem wartosci w zakresie:

$$(128...192)*\frac{5}{256} = 2{,}5...3{,}75\ \text{V},$$

czyli nalezy wybrac zakres 2,5...3,75 V wpisujac do rejestru **DAPR** wartosc 11001000_b = 0C8H.

W ostatnim przypadku dwa najstarsze bity moga przyjac wartosc 11. Wówczas wynik pomiaru miesci sie w przedziale $192...255_d$ (0C0H...0FFH), a zmierzone napiecie miesci sie w zakresie

$$(192...255)*\frac{5}{256} = 3{,}75...5{,}0\ \text{V},$$

czyli nalezy wybrac zakres 3.75..5V wpisujac do rejestru **DAPR** wartosc 00001100_b = 0CH.

Zatem deklaracja tablicy, w której umieszczone sa wartosci podzakresów pomiarowych wyglada nastepujaco:

db #$40, #$84,#$0C8,#$0C

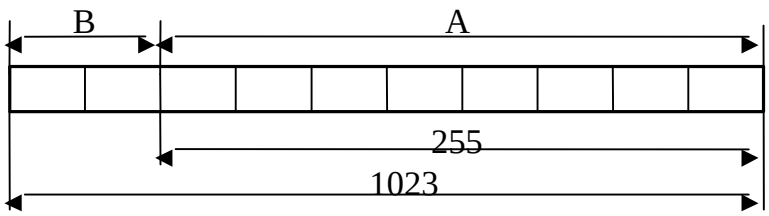
Wartosc mierzonego napiecia obliczana jest na podstawie obu pomiarów wg wzoru:

$$U_x = \frac{A + B * 255}{1024} * 5,$$

przy czym:

A – osiem bitów z drugiego pomiaru,

B – dwa najbardziej znaczace bity z pierwszego pomiaru.



4.5.3. Przyklady programowania przetwornika a/c

Przyklad 1

```
;*****************************************************************
;Program obslugi przetwornika a/c 8-bitowego.
;Wynik pomiaru wyswietlany jest na segmencie LED
;*****************************************************************

;*****************
;Rejestry sterujace
;*****************

        ;ADCON   $0D8       rejestr sterujacy przetwornikiem
        ;ADDAT   $0D9       rejestr wyniku przetwarzania
        ;DAPR    $0DA       rejestr programowania napiec wzorcowych

;***************
;Program glówny
;***************
        ORG $00

        MOV  A,ADCON        ;Wybór kanalu pomiarowego i trybu
        ANL  A,#$F0         ;przetwarzania(ADM)
        MOV  ADCON,A
POM1:MOV DAPR,#$00          ;Ustawienie zakresu pomiarowego,
```

```
                              ;start przetwarzania.
POM: JB  BSY,POM              ;Oczekiwanie na zakonczenie przetwarzania.

     MOV R2,ADDAT             ;Zmiana kierunku wyswietlania na segmencie LED.
     MOV R1,#00               ;Wynik pomiaru z R2 jest przesuwany w prawo
     MOV R3,#08               ;z flaga C i umieszczany w R1 przez przesuwanie
                              ;zawartosci w lewo z flaga C
PET: MOV A,R2                 ;Liczba przesuniec zawarta jest w R3
     RRC  A
     MOV R2,A
     MOV A,R1
     RLC  A
     MOV R1,A
     DJNZ R3,PET
     MOV P5,R1                ;Przeslanie wyniku na segment LED
     SJMP POM1
```

Przyklad 2

```
;*****************************************************************
;Program wykonania serii pomiarów napiecia na wejsciu AN3 z rozdzielczoscia
;8–bitowa. Wartosci zmierzonych napiec nalezy umiescic w pamieci zewnetrznej od
adresu $100
;*****************************************************************

;******************
; Deklaracja symboli
;******************
NR_pom          EQU  $05   ;Liczba pomiarów
ADR_pom         EQU  $20   ;Adres poczatkowy wyników w pamieci
                           ;wewnetrznej
STR_pom         EQU  $0B   ;Przetwarzanie ciagle , kanal  pomiarowy  AN3
ZAK_pom         EQU  $00   ;Zakres pomiarowy 5 V
ADR_wyn         EQU  $100  ;Adres poczatkowy wyników w pamieci
                           ;zewnetrznej

;*****************************************************************

     ORG  $00              ;Adres poczatku programu

     MOV  A,ADCON
     ANL  A,#$0F0          ;Zerowanie bitów sterujacych w ADCON
     ORL  A,#STR_pom       ;Konfiguracja przetwornika AD do pracy
     ORL  ADCON,A          ;ciaglej z pomiarem na 3 kanale
```

```
        MOV  R0,#NR_pom       ;Rejestr R0 jako licznik pomiarów
        MOV  R1,#ADR_pom      ;Rejestr R1 adresuje pamiec wewnetrzna RAM
        MOV  DAPR,#ZAK_pom

LOOP1: JB   BSY,LOOP1         ;Oczekiwanie na zakonczenie przetwarzania

        MOV  A,ADDAT          ;Wynik do akumulatora
        MOV  @R1,A            ;Przeslanie wyniku do pamieci wewnetrznej
        INC  R1               ;Zmiana adresu pamieci wewnetrznej
        DJNZ R0,LOOP1         ;Jesli R0>0 to skok do etykiety LOOP1
        ANL  ADCON,#$0F3      ;Koniec serii pomiarów
        MOV  R0,#NR_pom       ;Rejestr R0 jako licznik pomiarów
        MOV  R1,#ADR_pom      ;Adresowanie pamieci wewnetrznej
        MOV  A,@R1            ;Wynik pierwszego pomiaru do akumulatora
        MOV  DPTR,#ADR_wyn    ;Adresowanie pamieci zewnetrznej
        MOVX @DPTR,A          ;Wynik pierwszego pomiaru do pamieci zewnetrznej
LOOP2: INC  R1                ;Adresowanie kolejnego rejestru pamieci wewn.
        INC  DPTR             ;Adresowanie kolejnego rejestru pamieci zewn.
        MOV  A,@R1            ;Wynik drugiego pomiaru do akumulatora
        MOVX @DPTR,A          ;Wynik drugiego pomiaru do pamieci zewnetrznej
        DJNZ R0,LOOP2         ;Jesli R0>0, to skok do etykiety LOOP2
        RET
```

Przyklad 3

```
;*****************************************************************
;Program obslugi przetwornika a/c w SAB 80C535.
;Wynik (10 bitów) w pamieci zewnetrznej $100 I $101
;*****************************************************************

;************************
;Deklaracja symboli i adresów
;************************
        ;ADCON     $D8

        ;ADDAT     $D9
        ;DAPR      $DA

RAM_1       BYTE(($20)        ;Adres w pamieci wewnetrznej
EPR_1       EQU  $100         ;Wartosc pomocnicza do adresowania pamieci
                              ;zewnetrznej

        ORG  $00
        MOV  A,ADCON   ;Konfiguracja przetwornika , kanal  pomiarowy AN0
        ANL  A,#$0F0   ;pojedynczy pomiar
        MOV  ADCON,A
```

```
        MOV  DAPR,#$00  ;Zakres pomiaru 5V, start przetwarzania

Conv1:  JB   BSY,Conv1  ;Oczekiwanie na zakonczenie przetwarzania

        MOV  A,ADDAT    ;Wartosc wyniku do akumulatora
        ANL  A,#$C0     ;Wydzielenie 2 bitów najbardziej znaczacych
        SWAP A          ;Wymiana bitów 0...3 z bitami 4...7
        RR A            ;Dwukrotne przesuniecie wartosci akumulatora w prawo
        RR A
        MOV  RAM_1,A    ;Wyslanie wartosci 2 bitowej do pamieci wewnetrznej
        MOV  DPTR, #Tab         ;Ustawienie adresu tablicy
        MOVC A,@A+DPTR          ;Pobranie wartosci z tablicy jako nowego
        MOV  DAPR,A             ;podzakresu pomiarowego, start przetwarzania

Conv2:  JB   BSY,Conv2          ;Oczekiwanie na zakonczenie przetwarzania

        MOV  A,ADDAT
        MOV  DPTR,#EPR_1        ;Przeslanie mniej znaczacej wartosci pomiaru
        MOVX @DPTR,A            ;10–bitowego do pamieci zewnetrznej
        INC  DPTR
        MOV A,RAM_1             ;Pobranie 2–bitowej wartosci z pamieci
                                ;wewnetrznej
        MOVX @DPTR,A            ;i wyslanie jej  jako  bardziej znaczacej wartosci
        RET                     ;pomiaru 10–bitowego do pamieci zewnetrznej

Tab:    DB   #$40,#$84,#$0C8,#$0FC   ;Tablica zakresów pomiarowych
```

4.6. System przerwan

4.6.1. Opis systemu przerwan

Przerwanie jest to taki stan pracy, w którym mikroprocesor przerywa wykonywanie programu głównego i wykonuje podprogram zwiazany ze zródlem przerwania (sygnalem wywolujacym przerwanie). Sygnal powodujacy przerwanie moze pochodzic zarówno z ukladów wewnetrznych mikroprocesora, jak i z ukladów zewnetrznych otoczenia.

System przerwan mikrokontrolera **SAB 80C535** identyfikuje 12 zródel przerwan – 7 zewnetrznych i 5 pochodzacych z ukladów wewnetrznych. Schemat systemu przerwan przedstawiono na rys.4.6.

Kazdy sygnal mogacy wywolac przerwanie zwiazany jest z odpowiednim znacznikiem, przy czym znacznik ustawiany jest w stan 1 w momencie, gdy sygnal przerwania jest aktywny. Skasowanie zgloszenia przerwania spowodowane jest sprzetowym wyzerowaniem znacznika, które realizowane jest po wykonaniu programu obslugi przerwania. Wyjatkiem sa znaczniki **TI**, **RI**, **TF2** i **EXF2**, które musza zostac wyzerowane programowo. Wszystkie pozostale znaczniki moga byc takze ustawiane i kasowane programowo. Znaczniki te znajduja sie w rejestrach **TCON** i **IRCON**.

Rejestr **TCON** adres **088H**

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Rejestr **IRCON** adres **0C0H**

EXF2	TF2	IEX6	IEX5	IEX4	IEX3	IEX2	IADC

Znaczenie bitów w rejestrach jest nastepujace:

- **TF1** – znacznik od przepelnienia licznika **T1**.
- **TF0** – znacznik od przepelnienia licznika **T0**.
- **IE1** – znacznik od przerwania na wejsciu *INT1*.
- **IT1** – znacznik wyboru sygnalu przerywajacego na wejsciu *INT1*:
  IT1=0 – poziom niski,
  IT1=1 – zbocze opadajace.
- **IE0** – znacznik od przerwania na wejsciu *INT0*.
- **IT0** – znacznik wyboru sygnalu przerywajacego na wejsciu *INT0*:
  IT0=0 – poziom niski,
  IT0=1 – zbocze opadajace.
- **EXF2** – **T2**.znacznik ustawiany zewnetrznym sygnalem od przeladowania licznika **T2**.
- **TF2** – znacznik od przepelnienia licznika **T2**
- **IEX6** – znacznik od zewnetrznego sygnalu *INT6* lub od sygnalu wyjsciowego z komparatora 3.

- **IEX5** – znacznik od zewnetrznego sygnalu *INT5* lub od sygnalu wyjsciowego z komparatora 2.
- **IEX4** – znacznik od zewnetrznego sygnalu *INT4* lub od sygnalu wyjsciowego z komparatora 1.
- **IEX3** – znacznik od zewnetrznego sygnalu *INT3* lub od sygnalu wyjsciowego z komparatora 0.
- **IEX2** – znacznik od zewnetrznego sygnalu *INT2*.
- **IADC** – znacznik od przetwornika a/c ustawiany na 4 cykle przed koncem przetwarzania.

Znaczniki **TR1** i **TR0** nie sa zwiazane z systemem przerwan. Ich znaczenie omówiono w rozdziale 4.3.1.

Sygnal z kazdego zródla przerwania moze byc indywidualnie zablokowany – wówczas mikroprocesor nie przyjmuje zgloszenia przerwania z danego zródla i nie przerywa wykonywania programu glównego. Znaczniki blokujace przerwania znajduja sie w rejestrach **IEN0** i **IEN1**. Ustawienie znacznika na 1 oznacza odblokowanie przerwania, natomiast ustawienie znacznika na 0 powoduje zablokowanie przerwania.

Rejestr **IEN0** adres **0A8H**

EAL	WDT	ET2	ES	ET1	EX1	ET0	EX0

Rejestr **IEN1** adres **0B8H**

EXEN2	SWDT	EX6	EX5	EX4	EX3	EX2	EADC

Znaczenie poszczególnych bitów rejestrów **IEN0** i **IEN1** jest nastepujace:

- **EAL** – blokowanie wszystkich przerwan,
  EAL=0 – zadne przerwanie nie zostanie przyjete,
  EAL=1 – kazde przerwanie moze zostac przyjete pod warunkiem, ze odpowiadajacy mu znacznik ma wartosc 1.
- **ET2** – blokowanie lub odblokowanie przerwania od licznika **T2**.
- **ES** – blokowanie lub odblokowanie przerwanie od portu szeregowego.

- **ET1** – blokowanie lub odblokowanie przerwanie od licznika **T1**.
- **EX1** – blokowanie lub odblokowanie przerwania zewnetrznego z wejscia *INT1*.
- **ET0** – blokowanie lub odblokowanie przerwania od licznika **T0**.
- **EX0** – blokowanie lub odblokowanie przerwania zewnetrznego z wejscia *INT0*.
- **EXEN2**. – blokowanie lub odblokowanie przerwania od licznika **T2** wywolanego zewnetrznym sygnalem przeladowania licznika.
- **EX6** – blokowanie lub odblokowanie przerwania zewnetrznego z wejscia *INT6* lub od komparatora 3.
- **EX5** – blokowanie lub odblokowanie przerwania zewnetrznego z wejscia *INT5* lub od komparatora 2.
- **EX4** – blokowanie lub odblokowanie przerwania zewnetrznego z wejscia *INT4* lub od komparatora 1.
- **EX3** – blokowanie lub odblokowanie przerwania zewnetrznego z wejscia *INT3* lub od komparatora 0.
- **EX2** – blokowanie lub odblokowanie przerwania zewnetrznego z wejscia *INT2*.
- **EADC** – blokowanie lub odblokowanie przerwania od przetwornika a/c.

Uklady przerwan zewnetrznych *INT0* i *INT1* moga reagowac albo na poziom, albo na zbocze opadajace sygnalu wejsciowego – zalezne jest to od ustawienia bitów **IT0** i **IT1** w rejestrze **TCON**.

Uklady przerwan zewnetrznych *INT2* i *INT3* moga reagowac albo na zbocze opadajace, albo na zbocze narastajace sygnalu wejsciowego. Zalezy to od ustawienia bitów **I2FR** i **I3FR** znajdujacych sie w rejestrze **T2CON**.

Pozostale uklady przerwan *INT4*, *INT5* i *INT6* reaguja tylko na zbocza

narastajace sygnalu wejsciowego.

Wejscia sygnalów przerwan zewnetrznych testowane sa przez procesor w kazdym cyklu maszynowym, dlatego zarówno poziom niski, jak i wysoki powinien byc utrzymywany na wejsciu przez czas co najmniej jednego cyklu maszynowego – jest to warunek konieczny do prawidlowego odczytania sygnalu przerwania.

Obsluga przerwania rozpoczyna sie wywolaniem instrukcji *LCALL*. W tym samym czasie zerowany jest znacznik wywolujacy przerwanie, z wyjatkiem znaczników: **TI**, **RI**, **TF2** i **EXF2**, które musza zostac wyzerowane programowo. Nastepnie nastepuje skok pod odpowiedni adres wektora przerwania w pamieci programu i wykonanie podprogramu przerwania umieszczonego pod tym adresem (tabela 4.7).

Tabela 4.7. Adresy wektorów przerwan

Lp.	Zródlo przerwania	Znacznik przerwania	Adres wektora przerwania
1	Przerwanie zewn. *INT0*	IE0	03H
2	Licznik **T0**	TF0	0BH
3	Przerwanie zewn. *INT1*	IE1	13H
4	Licznik **T1**	TF1	1BH
5	Port szeregowy	RI+TI	23H
6	Licznik **T2**	TF2+EXF2	2BH
7	Przetwornik a/c	IADC	43H
8	Przerwanie zewn. *INT2*	IEX2	4BH
9	Przerwanie zewn. *INT3*	IEX3	53H
10	Przerwanie zewn. *INT4*	IEX4	5BH
11	Przerwanie zewn. *INT5*	IEX5	63H
12	Przerwanie zewn. *INT6*	IEX6	6BH

Podprogram obslugi przerwania musi byc zakonczony rozkazem *RETI*, który informuje procesor o zakonczeniu obslugi przerwania. Nastepuje wówczas ustawienie systemu przerwan w stan wyjsciowy.

Procedura obslugi przerwania moze rozpoczac sie w dowolnym momencie wykonywania programu glównego w nastepnym cyklu maszynowym po wykryciu zgloszenia przerwania, jesli spelnione sa nastepujace warunki:

– cykl maszynowy, w którym nastapilo zgloszenie przerwania jest ostatnim cyklem

aktualnie wykonywanej instrukcji rozkazu,

– nie jest wykonywana instrukcja *RETI* lub inna zwiazana z dzialaniem na zawartosci rejestrów **IE** lub **IPx**,
– aktualnie wykonywane przerwanie znajduje sie na nizszym poziomie.

W trakcie pracy mikrokontrolera moze sie zdarzyc, ze jednoczesnie wystapi zgloszenie kilku przerwan lub nastapi zgloszenie nowego przerwania w trakcie wykonywania podprogramu obslugi innego przerwania. Brak kontroli w takiej sytuacji móglby doprowadzic do zawieszenia sie systemu, gdyz procesor nie bylby w stanie zdecydowac o kolejnosci obslugi przerwan. Aby tego uniknac, wszystkie przerwania podzielone sa na poziomy, które decyduja o kolejnosci wykonywania przerwan. Dzieki temu mozliwe jest przerwanie wykonywania podprogramu obslugi przerwania znajdujacego sie na nizszym poziomie przez podprogram obslugi przerwania z wyzszego poziomu. Dodatkowo przerwania maja ustalone priorytety , które okreslaja kolejnosc obslugi przerwan, jesli jednoczesnie pojawia sie sygnaly przerwan znajdujacych sie na tym samym poziomie. Oznacza to, ze jako pierwsze zostanie wykonane przerwanie o wyzszym priorytecie. W tabeli 4.8 przedstawiono uszeregowanie zródel przerwan wg priorytetu.

Tabela 4.8. Priorytety zródel przerwan

Lp.	**Zródlo przerwania**	**Znacznik przerwania**	**Priorytet przerwa-nia**
1	Przerwanie zewn. *INT0*	IE0	Najwyzszy
2	Przetwornik a/c	IADC	
3	Licznik **T0**	TF0	
4	Przerwanie zewn. *INT2*	IEX2	
5	Przerwanie zewn. *INT1*	IE1	
6	Przerwanie zewn. *INT3*	IEX3	
7	Licznik **T1**	TF1	
8	Przerwanie zewn. *INT4*	IEX4	
9	Port szeregowy	RI+TI	
10	Przerwanie zewn. *INT5*	IEX5	
11	Licznik **T2**	TF2 + EXF2	
12	Przerwanie zewn. *INT6*	IEX6	Najnizszy

Mikrokontroler **SAB 80C535** wyposazony jest w czteropoziomowy system przerwan. Poziom przerwania ustalany jest przez ustawienie odpowiednich bitów

w rejestrach **IP0** i **IP1**. Sposób ustalania poziomów przerwan przedstawiony jest w tabeli 4.9.

Rejestr **IP0** adres **0A9H**

–	WDTS	IP0.5	IP0.4	IP0.3	IP0.2	IP0.1	IP0.0

Rejestr **IP1** adres **0B9H**

–	–	IP1.5	IP1.4	IP1.3	IP1.2	IP1.1	IP1.0

Tabela 4.9. Poziomy przerwan

Znaczniki		**Poziom przerwania**
IP1.x	**IP0.x**	
0	0	0
0	1	1
1	0	2
1	1	3

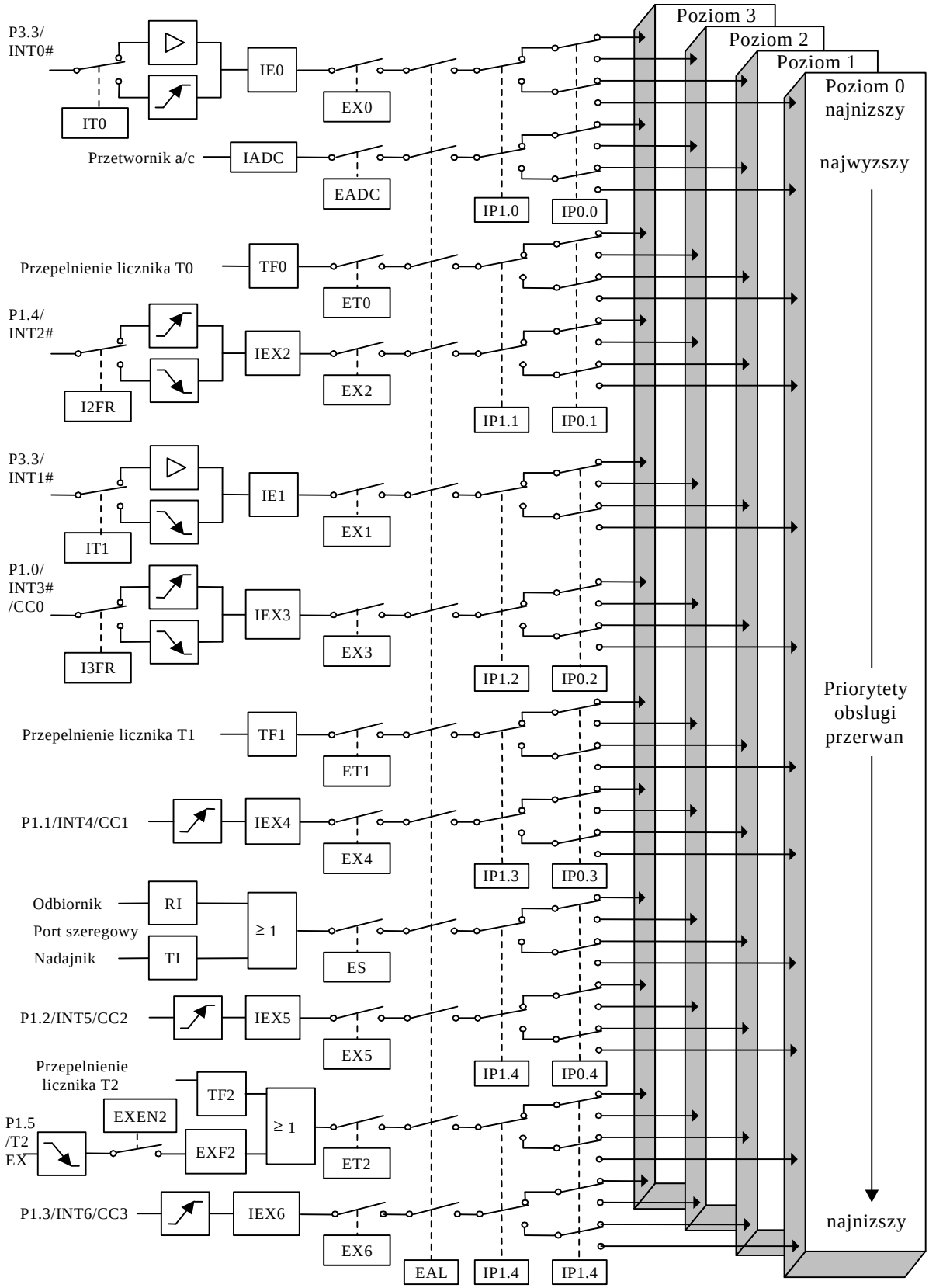
Uwaga: W tabeli 4.9 x moze przyjmowac wartosci 0, 1, 2, 3, 4, 5 zgodnie z oznaczeniem bitów w rejestrach **IP0** i **IP1**.

Liczba znaczników dostepnych w rejestrach **IP0** i **IP1** jest za mala, aby okreslic poziomy przerwan dla wszystkich zródel. Dlatego poszczególne zródla polaczono w pary i przypisano do odpowiednich znaczników. Przedstawiono to w tabeli 4.10.

Tabela 4.10. Powiazanie zródel przerwan ze znacznikami poziomów

Lp.	**Znaczniki przerwan**		**Zródla przerwan**	
1	IP1.0	IP0.0	IE0	IADC
2	IP1.1	IP0.1	TF0	IEX2
3	IP1.2	IP0.2	IE1	IEX3
4	IP1.3	IP0.3	TF1	IEX4
5	IP1.4	IP0.4	RI+TI	IEX5
6	IP1.5	IP0.5	TF2+EXF2	IEX6

Schemat systemu przerwan mikrokontrolera **SAB 80C535** przedstawiono na rys. 4.14.



Rys. 4.14. System przerwan mikrokontrolera SAB 80C535

4.6.2. Przyklady programowania systemu przerwan

Przyklad 1

```
;*********************************************************************
;Program sterowania wyjsciem P1.3 za pomoca przerwan zewnetrznych. Na wyjsciu
;P1.3 generowany jest przebieg PWM z uzyciem licznika T2. Program wykorzystuje
;komparator CC3, przy czym zbocze opadajace na wejsciu INT0, blokuje generowanie
;przebiegu PWM, a zbocze opadajace na wejsciu INT1 zalacza generowanie przebiegu
;*********************************************************************

;**************************
;Deklaracja stalych programu
;**************************
TIMER      EQU  $11   ;Autoladowanie po przepelnieniu licznika taktowanie
                      ;sygnalem wewnetrznym bez dzielnika
COMP_EN    EQU  $80   ;Odblokowanie trybu porównania i wpisu komparatora CC3
RELOAD_L   EQU  $00   ;Wartosc poczatkowa licznika T2 po przepelnieniu
RELOAD_H   EQU  $FF   ;Wartosc poczatkowa licznika T2 po przepelnieniu
COMP_L     EQU  $37   ;Wartosc porównania rejestru CCL3
COMP_H     EQU  $FF   ;Wartosc porównania rejestru CCH3
;*********************************************************************

      ORG    $0000

      LJMP   MAIN       ;Skok do programu glównego aby ominac obszar przerwan
      ORG    $0003      ;Adres przerwania zewnetrznego INT0
      LJMP   INT_0      ;Skok do programu obslugi przerwania INT0

      ORG    $0013      ;Adres przerwania zewnetrznego INT1
      LJMP   INT_1      ;Skok do programu obslugi przerwania INT1

     ORG     $0100
MAIN:
      SETB   EAL        ;Wlaczenie wszystkich przerwan
      SETB   IT0        ;Przerwania przyjmowane beda po wykryciu
      SETB   IT1        ;zmiany stanu z wysokiego na niski (zbocze opadajace)
      SETB   EX0        ;Odblokowanie przerwania zewnetrznego INT0
      SETB   EX1        ;Odblokowanie przerwania zewnetrznego INT1
      ORL    T2CON,#TIMER          ;Konfiguracja licznika T2
      MOV    CCEN,#COMP_EN         ;Wybór komparatora CC3
      MOV    CRCL,#RELOAD_L        ;Wpis wartosci poczatkowej L, czestotliwosc
      MOV    CRCH,#RELOAD_H        ;Wpis wartosci poczatkowej H, czestotliwosc
      MOV    CCL3,#COMP_L          ;Wpis wartosci porównania L, wypelnienie
      MOV    CCH3,#COMP_H          ;Wpis wartosci porównania H, wypelnienie
LOOP: SJMP   LOOP                 ;Petla nieskonczona, umozliwiajaca reali-
                                  ;zacje programu jedynie na przerwaniach
```

```
;************************************
;Procedura obslugi przerwania od  INT0
;************************************
INT_0:
      MOV   CCEN,#0            ;Wylaczenie trybu porównania
      CLR    P1.3              ;Zerowanie wyjscia generowanego sygnalu
      RETI

;************************************
;Procedura obslugi przerwania od INT1
;************************************
INT_1:
     MOV   CCEN,#COMP_EN        ;Wlaczenie trybu porównania
     RETI
```

Przyklad 2

```
;*********************************************************************
;Program wykorzystania przerwan INT2 i INT3 na wejsciach P1.4 i P1.0 do
;wygenerowania na wyjsciach P4.0 i P4.1 przebiegów prostokatnych przesunietych
;w fazie
;*********************************************************************
        ORG   $00
        LJMP  MAIN              ;Skok do programu glównego aby ominac obszar
                                ;przerwan

        ORG   $004B             ;Adres przerwania zewnetrznego  INT2
        LCALL INT_2             ;Wywolanie programu obslugi  przerwania
        RETI

        ORG   $0053             ;Adres przerwania zewnetrznego INT3
        LCALL INT_3             ;Wywolanie programu obslugi przerwania
        RETI

        ORG   $100              ;Adres programu glównego
MAIN: SETB  EAL                 ;Odblokowanie wszystkich przerwan
        SETB  EX2               ;Odblokowanie przerwania  zewnetrznego INT2
        SETB  EX3               ;Odblokowanie przerwania  zewnetrznego INT3
        ANL  T2CON,#$9F         ;Zerowanie bitów I2FR i I3FR
         ORL  T2CON,#$60        ;Wybór zboczy aktywnych sygnalu  przerwania
LOOP: CLR   P1.4                ;Zerowanie bitu P1.4
        CLR   P1.0              ;Zerowanie bitu P1.0
        LCALL DELAY2            ;Czas zerowego stanu bitów  P1.4 , P1.0
        SETB  P1.4              ;Wywolanie przerwania INT2 na  wejsciu P1.4
        LCALL DELAY1            ;Czas ustawionego stanu bitu P1.4
```

```
        SETB  P1.0              ;Wywolanie przerwania INT3 na  wejsciu P1.0
        LCALL DELAY1            ;Czas ustawionego stanu bitu P1.0
        SJMP  LOOP              ;Skok do poczatku programu wywolywania
                                ;przerwan

;*************************************
;Realizacja  programu  w  przerwaniu  od INT2
;*************************************
INT_2:  SETB  P4.0              ;Ustawienie bitu P4.0
       CLR   P4.1               ;Zerowanie bitu P4.1
       RET

;*************************************
;Realizacja programu w przerwaniu od INT3
;*************************************
INT_3:  CLR   P4.0              ;Zerowanie bitu P4.0
        SETB  P4.1              ;Ustawienie bitu P4.1
        RET

;*******************************
;Podprogram opóznienia czasowego 1
;*******************************
DELAY1:
        MOV   R5,#20            ;Wpisanie wartosci do rejestrów  opóznienia 1
HOP7:   MOV   R6,#100
HOP6:   MOV   R7,#100
HOP5:   DJNZ  R7,HOP5
        DJNZ  R6,HOP6
        DJNZ  R5,HOP7
        RET

;*******************************
;Podprogram opóznienia czasowego 2
;*******************************
DELAY2:
        MOV   R1,#10            ;Wpisanie wartosci do rejestrów  opóznienia 2
HOP3:   MOV   R2,#100
HOP2:   MOV   R3,#100
HOP1:   DJNZ  R3,HOP1
        DJNZ  R2,HOP2
        DJNZ  R1,HOP3
        RET
```

4.7. Uklad watchdog

4.7.1. Opis ukladu

W trakcie dzialania systemu mikroprocesorowego moze dojsc do sytuacji, w której na skutek wystapienia zaklócen nastapi zmiana zawartosci przesylanych na szynie danych procesora bitów lub zmiana przesylanego na szynie adresowej adresu. Zaklócenia te moga pochodzic zarówno od ukladów wewnetrznych (np. indukowanie impulsów elektromagnetycznych w sciezkach przewodzacych sygnaly), jak i od ukladów zewnetrznych (np. zle odfiltrowane napiecie zasilajace). Zmiana zawartosci bitów danych lub bitów adresu powoduje niewlasciwe dzialanie programu, co w niektórych sytuacjach moze doprowadzic do nieobliczalnych, czesto tragicznych dla uzytkownika systemu nastepstw. Aby temu zapobiec, stosuje sie pewne elementy kontroli pracy programu. Pierwszy z nich polega na takim pisaniu programów, aby byly one wyposazone w procedury samokorekcyjne – zmniejsza to prawdopodobienstwo wystapienia niewlasciwej pracy programu. Nie jest to jednak sposób niezawodny, poniewaz jesli zaklóceniom moze ulec program glówny, tak samo wiec mozna zaklócic programowe procedury zabezpieczajace program.

Znacznie efektywniejszym rozwiazaniem jest zastosowanie dodatkowego ukladu nadzorujacego prace programu, który w razie wystapienia nieprawidlowosci w jego dzialaniu lub nawet zawieszenia sie jego wykonywania spowoduje sprzetowy reset systemu i rozpoczecie wykonywania programu z parametrami poczatkowymi. W zwiazku z tym wymagania stawiane takiemu ukladowi brzmia nastepujaco:

– raz uruchomiony nie móglby byc zatrzymany w sposób programowy; zatrzymanie ukladu powinno byc mozliwe jedynie przez sprzetowy reset linia zewnetrzna,
– nadzorowany program nie moze dopuscic do zadzialania ukladu powodujacego reset systemu. Najczesciej realizuje sie to przez okresowa zmiane stanu tego ukladu w trakcie wykonywania programu.

Mikrokontroler **SAB 80C535** wyposazony jest w uklad watchdog, który jest ukladem nadzorujacym dzialanie programu spelniajacym powyzsze warunki. Uklad ten wyposazony jest w 16–bitowy licznik zwiekszajacy swoja zawartosc o 1 w kazdym cyklu maszynowym procesora. W momencie przepelnienia licznika nastepuje wygenerowanie wewnetrznego sygnalu resetujacego procesor. Sygnal ten jest traktowany jako przerwanie o najwyzszym priorytecie. Zadaniem poprawnie pracujacego nadzorowa-

nego przez uklad watchdog programu jest niedopuszczenie do przepelnienia licznika ukladu watchdoga przez cykliczne wpisywanie do niego wartosci poczatkowej równej 0. Zerowanie licznika powinno nastapic najpózniej na 4 takty przed jego przepelnieniem, poniewaz wtedy wlasnie generowany jest wewnetrzny sygnal resetujacy procesor. Dla procesora taktowanego zegarem 12MHz odstep czasu pomiedzy kolejnym wyzerowaniem licznika nie moze byc wiec wiekszy niz 65532 μs.

Programowe uruchomienie ukladu watchdog dokonywane jest przez ustawienie bitu **SWDT** znajdujacego sie w rejestrze **IEN1**.

Rejestr **IEN1** adres **0B8H**

EXEN2	**SWDT**	EX6	EX5	EX4	EX3	EX2	EADC

Po ustawieniu bitu **SWDT** sprzetowo wpisywana jest jedynka na pozycje bitu **WDTS** znajdujacego sie w rejestrze **IP0** i licznik rozpoczyna zliczanie od wartosci poczatkowej równej zeru.

Rejestr **IP0** adres **0A9H**

–	**WDTS**	IP0.5	IP0.4	IP0.3	IP0.2	IP0.1	IP0.0

Od tego momentu zatrzymanie i trwale zerowanie licznika ukladu watchdog mozliwe jest jedynie przez wyzwolenie zewnetrznego sygnalu *RESET#*, przy czym równoczesnie z wyzerowaniem rejestrów specjalnych **SFR** kasowany jest równiez bit **SWDT**. Programowe, okresowe kasowanie licznika ukladu watchdog polega na wpisaniu jedynki logicznej na pozycje bitów **WDT** i **SWDT** koniecznie w podanej kolejnosci. Znacznik **WDT** znajduje sie w rejestrze **IEN0** i kasowany jest przez procesor automatycznie 3 cykle po jego programowym ustawieniu.

Rejestr **IEN0** adres **0A8H**

EAL	**WDT**	ET2	ES	ET1	EX1	ET0	EX0

Znacznik **WDT** umozliwia programowa interpretacje zródla zerowania mikro-

kontrolera, gdyz zerowanie zewnetrzna linia *RESET#* wywoluje inne skutki niz zerowanie przez uklad watchdog. Dzieki takiemu rozwiazaniu, po restarcie systemu, mozna stwierdzic, jakie bylo zródlo jego wystapienia. W tabeli 4.11 przedstawiono porównanie skutków obu sposobów zerowania mikrokontrolera.

Tabela 4.11. Sposoby zerowania mikrokontrolera **SAB 80C535**

Zerowanie sprzetowe linia *RESET#*	**Zerowanie ukladem watchdog**
– zatrzymuje licznik ukladu watchdog – zeruje rejestry specjalne **SFR** i bit **WDT**	– nie zatrzymuje licznika ukladu watchdog – zeruje rejestry specjalne **SFR**, ale nie zeruje bitu **WDT**

4.7.2. Przyklady programowania ukladu watchdog

Przyklad 1

```
;*********************************************************************
;Program demonstrujacy dzialanie „watchdoga” w SAB 80c535. Program testuje stan
;przycisku dolaczonego do P5.0. Jesli przycisk jest przycisniety, to uklad watchdog jest
;zerowany programowo, jesli przycisk jest zwolniony, to uklad watchdog zeruje proce;sor, co powoduje miganie diody podlaczonej do P1.0
;*********************************************************************

      ORG    $00

        SETB SWDT           ;Wlaczenie ukladu watchdog

        MOV  P1,#0          ;Wygaszenie diody
        LCALL   DELAY1      ;Opóznienie czasowe, aby zauwazalny byl
                            ;efekt migania diody
        SETB    P1.0        ;Zapalenie diody
        LCALL   DELAY1      ;Opóznienie czasowe po zapaleniu diody
LOOP:  JB  P5.0,LOOP        ;Jesli wcisniety jest przycisk (stan niski na
                            ;wejsciu), to nastapi skok do procedury zerujacej
                            ;uklad watchdog
                            ;Jesli nie, to petla wykonywana bedzie tak dlugo, az
                            ;uklad watchdog nie zresetuje procesora, efektem
                            ;czego bedzie miganie diody
        SETB    WDT         ;Zerowanie ukladu watchdog
        SETB    SWDT
        SJMP    LOOP
```

```
;****************************
;Procedura opóznienia czasowego
;****************************
DELAY1:
        MOV    R5,#10
HOP7:   MOV    R6,#100
HOP6:   MOV    R7,#100
HOP5:   SETB   WDT          ;uklad watchdog musi byc zerowany w procedurze
        SETB   SWDT         ;opózniajacej, gdyz czas jej trwania jest
        DJNZ   R7,HOP5      ;dluzszy niz czas potrzebny na przepelnienie
        DJNZ   R6,HOP6      ;licznika ukladu watchdog
        DJNZ   R5,HOP7
        RET
```

Przyklad 2

```
;*************************************************************************
;Program demonstrujacy dzialanie ukladu watchdog w SAB 80C535. Uklad watchdog
;zerowany jest w procedurze obslugi przerwania od licznika T2, który poczatkowo
;autoladowany jest duza wartoscia tak, aby licznik ukladu watchdog nie zdazyl sie
;przepelnic. Wartosc do autoladowania licznika T2 jest stopniowo zmniejszana az do
;zera, co spowoduje przepelnienie licznika ukladu watchdoga i wygenerowanie przez
;niego sygnalu reset 4 takty przed osiagnieciem wartosci maksymalnej
;*************************************************************************

        ORG    $00
        LJMP MAIN           ;Skok na poczatek programu, w celu ominiecia
                            ;procedur obslugi przerwan

        ORG  $2B            ;Adres przerwania od licznika T2
        LJMP INT_T2         ;Skok do procedury obslugi

MAIN:
        SETB EAL            ;Odblokowanie wszystkich  przerwan
        SETB ET2            ;Odblokowanie  przerwania od licznika T2
        MOV  CRCH,#$0F      ;Wpis wartosci automatycznie ladowanej
        MOV  CRCL,#0        ;do T2 po jego przepelnieniu
        MOV  TH2,#$0F       ;Wpis wartosci poczatkowej licznika T2
        MOV  TL2,#0
        MOV  T2CON,#$11     ;Tryb 0 autoladowania  licznika T2, taktowanie
                            ;sygnalem wewnetrznym, uruchomienie licznika
        SETB SWDT           ;Uruchomienie ukladu watchdog
        LCALL INTRO         ;Uruchomienie procedury demonstracyjnej
```

```
LOOP:
        MOV  P1,CRCH        ;W petli glównej nastepuje zmniejszanie
        LCALL  DELAY1       ;wartosci poczatkowej do przeladowywania
        DEC   CRCH          ;licznika T2 i jednoczesne wysylanie 8
        SJMP LOOP           ;starszych bitów tej wartosci do portu P1
                            ;w celu wizualizacji

;*****************************
;Procedura opóznienia czasowego
;*****************************
DELAY1:
          MOV     R5,#20
HOP7:   MOV     R6,#100
HOP6:   MOV     R7,#100
HOP5:   DJNZ    R7,HOP5
          DJNZ    R6,HOP6
          DJNZ    R5,HOP7
          RET
;*********************************
;Procedura obslugi przerwania od  T2
;*********************************
INT_T2:
        SETB WDT        ;Zerowanie ukladu watchdog
        SETB SWDT
        CLR   TF2       ;Znacznik T2 wymaga programowego zerowania
        RETI

INTRO:                  ;Procedura wyswietlajaca na diodach
        MOV  R0,#1      ;podlaczonych do portu P1 pewnej kombinacji
        MOV  R1,#8      ;w celu pokazania momentu resetu przez uklad watchdog
INTRO_LOOP:
        MOV  P1,R0
        MOV  A,R0
        RL     A
        MOV  R0,A
        LCALL  DELAY1
        DJNZ R1,INTRO_LOOP
        RET
```

4.8. Praca mikrokontrolera w trybach oszczedzania energii

W niektórych przypadkach, gdy mikrokontroler zasilany jest z baterii i nie wykonuje zadnych obliczen, a jedynie oczekuje na sygnal ze sterowanego urzadzenia, praktyczne byloby wprowadzenie go w stan pracy z obnizonym poborem mocy. Musi

sie to odbyc jednak w taki sposób, aby mikrokontroler zachowal zawartosc najwazniejszych rejestrów i komórek pamieci, oraz zeby byl mozliwy szybki powrót do stanu normalnej pracy.

Mikrokontroler **SAB 80C535** wyposazony jest w mechanizm umozliwiajacy wprowadzenie ukladu w specjalny tryb pracy, w którym nie jest wykonywany program i znacznie zmniejszony jest pobór pradu. Mikrokontroler zachowuje jednak zawartosc calej wewnetrznej pamieci *RAM* i rejestrów specjalnych **SFR**. Nie jest przy tym wymagane dolaczenie dodatkowego napiecia podtrzymujacego zawartosc wewnetrznej pamieci *RAM*, gdyz jest ona zasilana napieciem z wejscia U_{cc}, przy czym w trybie **Power Down** napiecie zasilajace moze byc obnizone do 2 V. Obnizenie napiecia zasilajacego umozliwia zmniejszenie poboru pradu nawet o ok. 500 razy w stosunku do pradu pobieranego w trakcie normalnej pracy mikrokontrolera.

W mikrokontrolerze **SAB 80C535** dostepne sa dwa tryby pracy o obnizonym poborze mocy: tryb **Idle** i tryb **Power Down**. Tryby te moga byc sprzetowo kontrolowane przez zmiane stanu na wyprowadzeniu **PE#** mikrokontrolera, przy czym:

- **PE# = 1** – zablokowanie programowego wyboru trybu pracy z obnizonym poborem mocy,
- **PE# = 0** – odblokowanie programowego wyboru trybu pracy z obnizonym poborem mocy.

Tryb **Power Down** ma wyzszy priorytet od trybu **Idle**, dlatego przy jednoczesnym wybraniu obu trybów mikrokontroler wprowadzany jest w stan pracy **Power Down**.

Wybór jednego z dwóch przedstawionych trybów odbywa sie przez ustawienie odpowiednich bitów sterujacych w rejestrze **PCON** przy zalozeniu, ze wyprowadzenie **PE#** sterowane jest sygnalem logicznego zera. Porównanie trybów oszczedzania mocy mikrokontrolera przedstawiono w tabeli 4.12.

Tabela 4.12. Porównanie trybów pracy jalowej

	Tryb Idle	**Tryb Power Down**

Wybór trybu	– PE# = 0 – IDLE=1 i IDLS=0 IDLE=0 i IDLS=1	– PE#=0 – PDE=1 i PDS=0 PDE=0 i PDS=1
Zakonczenie trybu	– Wywolanie dowolnego przerwania – Zerowanie linia *RESET#*	– Zerowanie linia *RESET#*

Znaczniki **IDLE**, **IDLS**, **PDE** i **PDS** znajduja sie w rejestrze **PCON**. Pozostale znaczniki **GF0** i **GF1** sa znacznikami ogólnego przeznaczenia.

Rejestr **PCON** adres **87H**

SMOD	PDS	IDLS	–	GF1	GF0	PDE	IDLE

Po wprowadzeniu mikrokontrolera w tryb pracy **Idle** jednostka arytmetyczno–logiczna i uklad watchdog nie sa taktowane z wewnetrznego zegara ukladu, natomiast pozostale uklady wewnetrzne moga dzialac normalnie. Ograniczenie poboru mocy zalezne jest od liczby dzialajacych ukladów wewnetrznych.

Wprowadzenie mikrokontrolera w stan pracy **Idle** musi nastapic wg okreslonej procedury:

w pierwszym kroku nalezy ustawic znaczniki nastepujaco:

IDLE=1 i IDLS=0,

w drugim kroku znacznikom nalezy przypisac wartosci:

IDLE=0 i IDLS=1.

Dzieki takiej procedurze unika sie przypadkowego wprowadzenia mikrokontrolera w stan obnizonego poboru mocy. Poniewaz znaczniki rejestru **PCON** nie moga byc adresowane bitowo, dlatego ustawienia poszczególnych bitów rejestru nalezy dokonac instrukcja *ORL*. Znaczniki te sa kasowane automatycznie przez mikrokontroler zaraz po ich ustawieniu, wiec nie jest konieczne ich programowe kasowanie.

Wprowadzenie mikrokontrolera w tryb pracy **Power Down** powoduje zatrzymanie generatora mikrokontrolera, licznika ukladu watchdog oraz wszystkich pozostalych ukladów wewnetrznych. Zablokowane sa równiez przerwania. Podobnie jak w

przypadku trybu **Idle**, przy wyborze trybu **Power Down** bity sterujace **PDE** i **PDS** musza byc ustawiane wg kolejnosci:

– w kroku pierwszym:

  PDE=1 i PDS=0,

– w kroku drugim:

  PDE=0 i PDS=1.

Wyjscie z trybu **Power Down** mozliwe jest jedynie przez sprzetowe wyzerowanie mikrokontrolera zewnetrzna linia *RESET#*. Nalezy przy tym pamietac, ze jesli podczas pracy w tym trybie mikrokontroler zasilany byl obnizonym napieciem, to impuls zerujacy musi byc podany dopiero po osiagnieciu przez napiecie zasilajace normalnego poziomu. Ponadto, czas trwania impulsu musi byc na tyle dlugi, aby wewnetrzny generator mikrokontrolera osiagnal stabilne warunki pracy (10...20 ms).

5. WSPÓLPRACA MIKROKONTROLERA Z UKLADAMI ZEWNETRZNYMI

5.1. Wspólpraca mikrokontrolera z klawiatura

Klawiatura jest obecnie nieodzownym elementem kazdego nowoczesnego urzadzenia zawierajacego uklad mikroprocesorowy. Umozliwia ona programowanie urzadzenia, wybieranie trybu pracy itp. Sposób dolaczenia klawiatury do obslugujacego ja mikrokontrolera moze byc rózny, zalezy on od aktualnej konfiguracji mikrokontrolera, liczby wolnych portów, liczby klawiszy i sposobu ich dzialania.

W module dydaktycznym firmy *Micromax* zastosowano klawiature zlozona z 12 klawiszy astabilnych, podobna do typowej klawiatury telefonicznej. Stanem aktywnym po nacisnieciu klawisza jest wartosc zera logicznego. Klawisze polaczone sa poprzez bufor z magistrala danych procesora. Odczyt stanu klawiatury polega na odczytaniu stanu bufora, przeslaniu informacji z klawiatury do akumulatora i nastepnie analizie stanu klawiszy.

5.2. Wspólpraca mikrokontrolera z wyswietlaczem

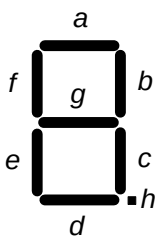
W trakcie pracy systemu mikroprocesorowego zachodzi potrzeba wyswietlania pewnych informacji dotyczacych wykonywanego przez system zadania. Rodzaj zastosowanego wskaznika zalezy od typu informacji, jaka potrzebna jest uzytkownikowi do zinterpretowania stanu, w którym znajduje sie system. W przypadku wizualizacji prostych stanów sygnalizujacych, np. zalaczenie przekaznika, wystarcza zastosowanie pojedynczej lampki sygnalizacyjnej, najczesciej diody *LED*. Jednak ilosc informacji, która nalezy wyswietlic podczas pracy nowoczesnego systemu mikroprocesorowego jest tak duza, ze obecnie stosuje sie dwa rodzaje wskazników: elektroluminescencyjne, zlozone z wielu diod *LED* oraz wskazniki cieklokrystaliczne *LCD*. Wskazniki *LCD* w odróznieniu od diod *LED* nie sa elementami pólprzewodnikowymi. Same nie wytwarzaja swiatla i dlatego, aby wyswietlane na nich znaki byly widoczne musza zostac oswietlone zewnetrznym zródlem swiatla. W normalnym stanie element cieklokrysta-

liczny jest przezroczysty. Dopiero po doprowadzeniu napiecia ciemnieje i dzieki temu staje sie widoczny.

Najczesciej uzywane sa nastepujace typy wskazników:

- **Wskazniki 7-segmentowe**

Do wyswietlania prostych informacji stosowane sa wskazniki zlozone z siedmiu krótkich, plaskich segmentów (dlatego posiadaja one nazwe wskazników 7–segmentowych) i kropki. Stosowane sa one najczesciej w sprzecie powszechnego uzytku, gdzie ilosc wyswietlanej informacji jest niewielka. Budowane sa jako wskazniki *LED*, a takze jako wskazniki *LCD*. Ze wzgledu na ich budowe mozna na nich wyswietlic jedynie cyfry 0...9, wybrane litery: A, b, c, C, d, E, F, H, I, L, o, P, r, S, U (litery B i D nie sa wyswietlane, gdyz nie mozna odróznic ich od cyfr 8 i 0) oraz znaki: minus (-), apostrof (‘), stopien (°) i grecka litere Γ. Sterowanie wskaznikiem 7–segmentowym polega na tym, ze kazdej cyfrze (a takze mozliwej do wyswietlenia literze) zazwyczaj zakodowanej w kodzie *BCD* nalezy przyporzadkowac odpowiednia kombinacje segmentów. Taki uklad sterowania nosi nazwe dekodera kodu *BCD* na kod 7–segmentowy. Dekodery te budowane sa jako uklady scalone, w których dekodowanie, czyli zamiana znaków wykonywana jest sprzetowo. Nalezy przy tym zaznaczyc, ze ukladowo zamieniane sa zwykle jedynie cyfry. Litery i pozostale znaki zamieniane sa programowo w tablicy. Na rysunku 5.1 przedstawiono rozmieszczenie segmentów typowego wskaznika 7–segmentowego.

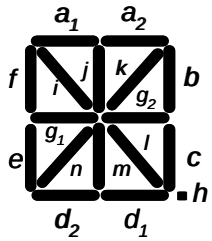


Rys. 5.1. Wskaznik 7–segmentowy

- **Wskazniki alfanumeryczne**

Ze wzgledu na mala rozdzielczosc wskazniki 7–segmentowe umozliwiaja wyswietlenie tylko niewielu znaków. Aby wyswietlic wszystkie litery alfabetu konieczne jest zwiekszenie rozdzielczosci, co osiaga sie przez zastosowanie wskaznika 16–segmentowego lub matrycy 35–znakowej.

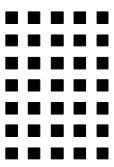
Rozmieszczenie segmentów typowego wskaznika 16–segmentowego przedstawiono na rys. 5.2. Segmenty *a*, *d* i *g* wskaznika 7–segmentowego podzielono na dwie czesci i dodano segmenty *i*...*n*.



Rys. 5.2. Wskaznik 16–segmentowy

Wskazniki 16–segmentowe sluza najczesciej do wyswietlania 64 znaków obejmujacych wielkie litery, cyfry 0...9 oraz znaki specjalne, takie jak nawiasy, znak procenta (%), apostrofy, znak plus (+) i inne.

Wskazniki matrycowe umozliwiaja uzyskanie najlepszej rozdzielczosci wyswietlania. Najczesciej sa budowane w postaci matrycy 35–punktowej (5 x 7 punktów), umozliwiajacej wyswietlenie praktycznie wszystkich mozliwych znaków, w tym 96 znaków *ASCII* i 32 znaki specjalne. Na rysunku 5.3 przedstawiono rozmieszczenie segmentów na wskazniku 35–punktowym.



Rys. 5.3. Rozmieszczenie punktów w matrycy 35–punktowej

Ze wzgledu na duza liczbe polaczen w matrycach punktowych nie wyprowadza sie zacisków kazdego elementu, lecz organizacje macierzowa wprowadza sie równiez od strony elektrycznej. Konieczne jest wtedy zasilanie impulsowe wskaznika z podzia-

lem czasowym, czyli tzw. *sterowanie multipleksowe*.

Wskazniki alfanumeryczne budowane sa jako wskazniki *LED* i jako wskazniki *LCD*. *Multipleksowe* sterowanie wskazników *LCD* jest bardziej skomplikowane, gdyz nie mozna uniknac podawania napiecia zmiennego na punkty nie wybrane. Dlatego do sterowania uzywa sie sygnalu o trzech poziomach (oprócz zera), przy czym amplituda na nie wybranych segmentach lezy ponizej progu wlaczania. Technika ta nosi nazwe sterowania *triplex*.

5.2.1. Sterowanie wskaznikami *LED*

Wskazniki elektroluminescencyjne zbudowane sa z wielu pojedynczych diod *LED* polaczonych w zaleznosci od konstrukcji w moduly 7–segmentowe, 16–segmentowe lub matryce 35–punktowe. Wspólpraca pomiedzy mikrokontrolerem a wskaznikami elektroluminescencyjnymi odbywa sie poprzez ogólnie dostepne porty wejscia–wyjscia, w zaleznosci od konfiguracji ukladu. Wskazniki elektroluminescencyjne sterowane sa najczesciej w sposób dynamiczny. Polega on na tym, ze w danej chwili aktywny jest tylko jeden wyswietlacz a pozostale pozostaja wygaszone. W nastepnej chwili wygasza sie aktywny wyswietlacz i uaktywnia nastepny. Wrazenie ciaglosci wyswietlanej informacji uzyskuje sie dzieki zjawisku bezwladnosci ludzkiego oka. Stosowane rzadziej sterowanie statyczne polega na tym, ze w danej chwili wszystkie wskazniki sa aktywne.

5.2.2. Sterowanie wskaznikami *LCD*

W wielu urzadzeniach ilosc wyswietlanych danych jest tak duza, ze zastosowanie pojedynczych wskazników nie pozwala na efektywna komunikacje z uzytkownikiem. Stosuje sie wówczas specjalne pola odczytowe, które umozliwiaja wyswietlenie informacji tekstowych. Sterowanie takim polem odczytowym jest na tyle skomplikowane, ze budowane sa one jako gotowe moduly zawierajace oprócz wyswietlacza wyspecjalizowany kontroler, który steruje przyjmowaniem i wyswietlaniem danych na wyswietlaczu. Przykladem takiego modulu jest pole *LCD* zastosowane

w module dydaktycznym firmy *Micromax*.

Wyswietlacz wyposazony jest w dwa wiersze po 16 znaków, przy czym kazdy znak zbudowany jest jako matryca 35-punktowa. Wyswietlenie wybranego znaku spowodowane jest uaktywnieniem odpowiednich punktów matrycy. Kontroler wyswietlacza zawiera generator znaków, którym jest tablica w pamieci *ROM*. Znaki te zapisane sa pod adresami zgodnymi z kodem *ASCII*. Uproszczono w ten sposób proces wyswietlania znaku, gdyz do jego wyswietlenia wystarczy wyslac do kontrolera tylko odpowiadajacy danemu znakowi kod.

Praca wyswietlacza mozna sterowac przez wyslanie do niego odpowiedniego rozkazu. Zestaw dostepnych rozkazów umozliwia m.in.:

- okreslenie szerokosci slowa sterujacego,
- wlaczenie lub wylaczenie kursora,
- wybranie rodzaju kursora,
- ustawienie kursora w wybranej pozycji,
- okreslenie, czy podczas wpisywania znaków elementem przesuwajacym sie jest kursor, czy wyswietlone juz znaki i w jakim kierunku.

Kontroler wyswietlacza zawiera dwa rejestry. Jeden z rejestrów jest rejestrem sterujacym, do którego wpisywane sa rozkazy sterujace praca wyswietlacza, drugi natomiast jest rejestrem danych, do którego wpisywane sa kody znaków wyswietlanych na ekranie. Do sterowania rejestrami (wyboru danego rejestru) wykorzystuje sie linie adresowa oznaczona w module dydaktycznym symbolem RS. Dodatkowo kontroler ma jeszcze dwie linie sterujace: R/W decydujaca o wpisie lub odczycie danej i EN, która blokuje prace kontrolera w przypadku ustawienia jej w stanie zera logicznego.

Po zalaczeniu zasilania nalezy uaktywnic kontroler przez wyslanie odpowiedniego rozkazu do rejestru sterujacego i ustawic parametry jego pracy. Przed wyslaniem do wyswietlacza rozkazów sterujacych lub znaku do wyswietlenia nalezy sprawdzic czy kontroler jest juz dostepny, poniewaz w stosunku do mikrokontrolera dziala on znacznie wolniej.

5.3. Przyklad programu sterujacego klawiatura i wyswietlaczem

```
;*********************************************************************
;Program obslugi wyswietlacza DM 1621-OS i klawiatury
;*********************************************************************

AdrBuf      equ   $60              ;Adres bufora wyswietlacza do przechowy-
                                   ;wania znaków z Demo
LineNr      byte  ($5f)            ;Kolejnosc znaku czytanego z Demo
CharNr      byte  ($5e)            ;Kolejnosc znaku wyswietlanego na LCD
BasAdr      equ   $fc00            ;Adres bazowy wyswietlacza. LCD
IR          equ   BasAdr           ;Adres bufora instrukcji
RBF         equ   BasAdr+1         ;Adres bufora  trybu pracy
RAM         equ   BasAdr+2         ;Adres bufora wyswietlania
ClrDisequ   equ   %00000001        ;Komendy wyswietlacza
FcSet equ   equ   %00111000
DispOn      equ   %00001110
DispOff     equ   %00001000
EntMod      equ   %00000110

;***************
;Program glówny
;***************

        MOV       Sp,#$0D0
        LCALL     IniLCD       ;Wywolanie podprogramu inicjalizacji wyswietlacza
        MOV       LineNr,#0    ;Kolejnosc znaku w linii do wyswietlenia z procedu-
                               ;ry Demo
Loop2:  MOV       R2,LineNr
        LCALL     IniBuf       ;Przeniesienie dwóch linii z Demo do bufora
        LCALL     DspBuf       ;Wyswietlenie dwóch linii z bufora
        MOV       A,LineNr     ;Ustawienie wskaznika LineNr na kolejne  dwie linie
                               ;w procedurze Demo
        ADD       A,#32
        MOV       LineNr,A
Loop4:  LCALL     RdKbd              ;Wywolanie procedury czytania klawiatury
        CJNE      A,#$FF,Loop5       ;Jesli jest, wczytany znak z klawiatury to skok
                                     ;do Loop5
        SJMP      Loop4              ;Brak znaku z klawiatury to skok do poczatku
                                     ;procedury
Loop5:  CJNE      A,#$3a,Loop2       ;Sprawdzanie czy znak pochodzi od lewej
                                     ;kropki
                                   ;Jesli nie, to skok do czytania kolejnej linii z Demo
Loop3:LCALL       IniLCD       ;Inicjalizacja wyswietlacza do wyswietlania znaków
                               ;z klawiatury
```

```
        MOV     CharNr,#0    ;Ustawienie pozycji znaku na 0
Loop:   LCALL   RdKbd        ;Czytanie klawiatury
        CJNE    A,#$FF,Loop1 ;Czy jest znak z klawiatury
        SJMP    Loop
Loop1:  CJNE    A,#$3a,Loop8 ;Czy znak pochodzi od lewej kropki
        SJMP    Loop2        ;Jesli tak, to skok do czytania kolejnej linii z
                             ;Demo
                             ;Jesli nie, to przechodzi do procedury wyswie-
                             ;tlenia znaku z klawiatury
Loop8:  LCALL   WrDat        ;Wywolanie procedury wyswietlania znaku z
                             ;klawiatury
        MOV     A,CharNr     ;Pozycja znaku w linii 1 wyswietlacza
        CJNE    A,#15,Loop6  ;Sprawdzenie czy linia 1 jest wypelniona
        MOV     A,#$c0       ;Jesli tak, to ustawienie adresu 2 linii wyswie-
                             ;tlacza
        LCALL   WrIR         ;Zapisanie ustawienia wyswietlacza na 2 linie
        INC     CharNr       ;Kolejna pozycja znaku w linii
        SJMP    Loop
Loop6:  CJNE    A,#31,Loop7  ;Sprawdzenie czy linia  2 jest wypelniona
        SJMP    Loop3        ;Jesli tak, to skok do procedury inicjalizacji
                             ;wyswietlacza
Loop7:  INC     CharNr       ;Jesli nie, to wczytanie kolejnego znaku
        SJMP    Loop

;************************************
; Inicjalizacja wyswietlacza LCD
;************************************

IniLCD: MOV     A,#ClrDis    ;Wyzerowanie wyswietlacza
        LCALL   WrIR
        MOV     A,#FcSet     ;Ustawienie funkcji
        LCALL   WrIR         ;Zapisanie ustawienia
        MOV     A,#DispOn    ;Ustawienie zalaczenia wyswietlacza
        LCALL   WrIR
        MOV     A,#EntMod    ;Ustawienie trybu wprowadzania danych
        LCALL   WrIR
        RET


;**************************************
; Podprogram inicjalizacji bufora wyswietlacza.
;We: R2 – pozycja znaku, R3 – nr komunikatu.
;**************************************

IniBuf: MOV     R0,#AdrBuf   ;Adresowanie pamieci przechowywania zna-
```

```
                                ;ków z Demo
        MOV     R1,#32          ;Licznik znaków wyswietlacza
IniBf1: MOV     A,R2            ;Kolejnosc znaku w linii z procedury Demo
        LCALL   RdDemo          ;Wczytanie znaku z Demo
        MOV     @R0,A           ;Umieszczenie znaku w pamieci AdrBuf
        INC     R0              ;Zmiana pozycji w pamieci
        INC     R2              ;Zmiana  pozycji znaku
        DJNZ    R1,IniBf1       ;Sprawdzanie licznika znaków wyswietlacza
        RET

;*************************************************************************
;Podprogram zapisu bufora do wyswietlacza i procedura wyswietlania zawartosci z
;bufora. AdrBuf na wyswietlaczu w dwóch liniach
;*************************************************************************

DspBuf: MOV     R0,#16          ;Liczba  znaków w linii wyswietlacza
        MOV     R1,#AdrBuf
        MOV     A,#$80          ;Ustawienie wyswietlacza LCD na wyswietlanie w
                                ;pierwszej linii
        LCALL   WrIR            ;Zapis instrukcji ustawienia linii
        LCALL   Dsp1Ln          ;Wywolanie procedury wyswietlenia znaków w 1 li-
                                ;nii
        MOV     A,#$0c0         ;Ustawienie wyswietlacza LCD na wyswietlanie
                                ;w drugiej linii
        LCALL   WrIR            ;Zapis instrukcji
        MOV     R0,#16          ;Liczba znaków w linii
        LCALL   Dsp1Ln
        RET

Dsp1Ln:MOV      A,@R1           ;Wczytanie znaku z pamieci AdrBuf
        LCALL   WrDat           ;Wyswietlenie znaku
        INC     R1              ;Zmiana pozycji znaku
        DJNZ    R0,Dsp1Ln       ;Sprawdzenie wypelnienia linii wyswietlacza
        RET

;*********************************
; Podprogram odczytu stanu klawiatury
;*********************************

RdKbd: MOV      R4,#0           ;Flaga obecnosci znaku,#0 oznacza brak nacisniecia
                                ;klawisza
RdKbd4: MOV     B,#3            ;Wartosc pomocnicza do wyznaczania kodu
        MOV     R0,#1           ;Nr kolumny klawiatury
        MOV     R1,#$ef         ;Maska do czytania stanu klawiatury
        MOV     R2,#3           ;Licznik kolumn
```

```
RdKbd3: MOV     P4,R1       ;Przeslanie maski na port P4,odczyt stanu kla-
wiszy
                            ;w kolumnie pierwszej
        MOV     A,P4        ;Stan klawiatury do A
        SWAP    A
        MOV     R3,#4       ;Licznik wierszy,
RdKbd1: RLC     A           ;Sprawdzenie czy jest ustawiony bit w wierszu
                            ;pierwszej kolumny
        JNC     RdKbd2      ;Wcisniety klawisz w wierszu to skok do procedury
                            ;wyznaczania kodu wykrytego wcisnietego klawisza
        DJNZ    R3,RdKbd1   ;Adresowanie kolejnego wiersza i sprawdzenie stanu
                            ;klawisz w nim
        INC     R0          ;Jesli klawisz w pierwszej kolumnie nie jest
                            ;wcisniety, to sprawdzanie drugiej kolumny
        MOV     A,R1        ;Przesuniecie jedynki w masce do sprawdzania dru-
                            ;giej kolumny
        RL      A
        MOV     R1,A
        DJNZ    R2,RdKbd3   ;Sprawdzanie licznika kolumn, skok do odczytu sta-
                            ;nu klawiszy w kolumnie
        CJNE    R4,#0,RdKbd5    ;Sprawdzenie flagi obecnosci znaku , R4=0 to
                                ;klawisz  nie byl nacisniety
        MOV     A,#$FF          ;Brak znaku
        RET
;
RdKbd2: MOV     R4,#1       ;Wykryty znak ,1jest przesylana do R4, procedura
                            ;obliczania kodu wykrytego znaku
        DEC     R3          ;Zmiana wskaznika wiersza
        MOV     A,R3
        MUL     AB
        ADD     A,R0
        ADD     A,#$30
        MOV     R5,A        ;Przeslanie obliczonego kodu do rejestru roboczego
        MOV     A,#1
        LCALL   Delay
        SJMP    RdKbd4
RdKbd5: MOV     A,R5
        RET

;*************************************
; Podprogram badania gotowosci wyswietlacza
;*************************************
Busy:   MOV     Dptr,#RBF
        PUSH    Acc
Bus1:   MOVX    A,@Dptr
```

```
        ANL     A,#$80
        JNZ     Bus1        ;Badanie flagi gotowosci.
        POP     Acc
        RET

;*****************************************************
; Podprogram zapisu do pamieci CG lub DD wyswietlacza LCD.
;We:  A – dana
;*****************************************************

WrDat: LCALL    Busy            ;Gotowosc wyswietlacza
        MOV     Dptr,#Ram       ;Adres bufora wyswietlacza
        MOVX    @Dptr,a         ;Wyswietlenie znaku
        MOV     A,#0            ;Ustawienie wolnego pola na wyswietlaczu
        LCALL   Busy
        MOV     Dptr,#Ram
        MOVX    @Dptr,A
        RET
;
;************************************
; Podprogram zapisu instrukcji do wyswietlacza LCD
; We:  A - instrukcja
;************************************

WrIR: LCALL     Busy
        MOV     Dptr,#IR        ;Adres bufora instrukcji
        MOVX    @Dptr,A
        RET
;
;*********************************
; Podprogram opóznienia
; We:  A - liczba powtórzen
;*********************************

Delay: MOV      R0,A
Del1:  MOV      R1,#$0FF
Del2:  MOV      R2,#$0FF
Del3:  DJNZ     R2,Del3
        DJNZ    R1,Del2
        DJNZ    R0,Del1
        RET
;
RdDemo:     INC   A
            MOVC      A,@A+PC
            RET
```

```
Demo:   db   ‘POLITECHNIKA      ‘
        db   ‘WROCLAWSKA        ‘
        db   ‘INSTYTUT          ‘
        db   ‘MASZYN I NAPEDÓW  ’
        db   ‘ELEKTRYCZNYCH     ‘
        db   ‘STUDIA DZIENNE    ‘
        db   ‘INZYNIERSKO-      ‘
        db   ‘-MAGISTERSKIE     ‘
        db   ‘STUDIA ZAOCZNE    ‘
        db   ‘INZYNIERSKIE      ‘
        db   ‘NA KIERUNKU       ‘
        db   ‘ELEKTROTECHNIKA   ‘
        db   ‘ORAZ              ‘
        db   ‘STUDIA DZIENNE    ‘
        db   ‘NA KIERUNKU       ‘
        db   ‘AUTOMATYKA        ‘
        db   ‘I ROBOTYKA        ‘
```