2. CHARAKTERYSTYKA MIKROKONTROLERA SAB 80C535

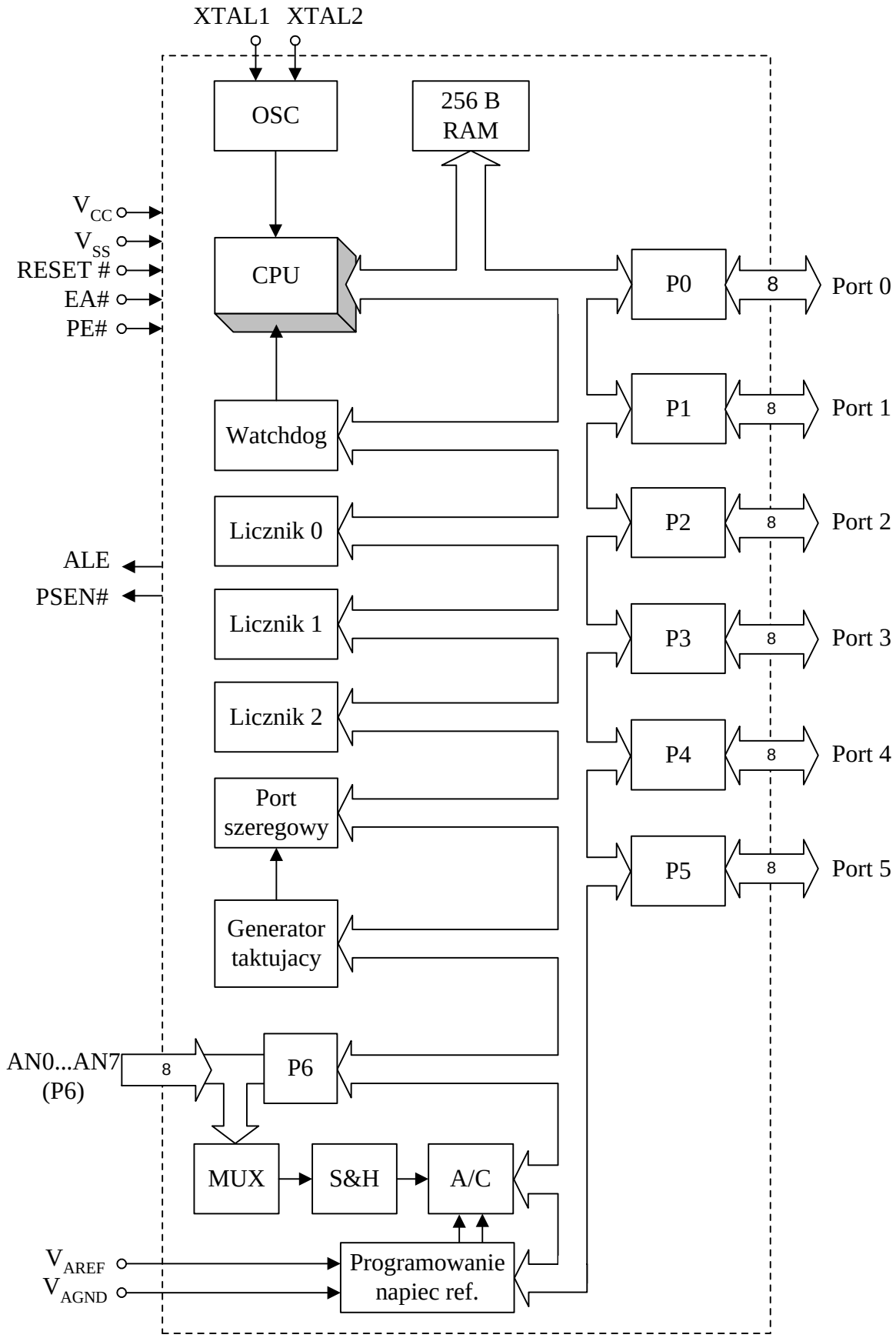
2.1. Budowa i wlasciwosci

Mikrokontroler **SAB 80C535** jest zbudowany na podstawie jednoukladowego 8–bitowego mikroprocesora, wzorowanego na ukladzie **8051** firmy INTEL. Litera *C* w oznaczeniu wskazuje na wykonanie mikroprocesora w technologii *ACMOS*. Mikroprocesory wykonane w tej technologii róznia sie nieznacznie od produkowanych w technologii *MYMOS* (zmiany dotycza glównie zakresu pracy rezonatora kwarcowego i wprowadzeniu w technologii *ACMOS* portu **P6**). Podstawowe wlasciwosci procesora to:

– 256 bajtów wewnetrznej pamieci danych,
– 128 bajtów rejestrów specjalnych SFR,
– mozliwosc adresowania do 64 kB zewnetrznej pamieci programu,
– mozliwosc adresowania do 64 kB zewnetrznej pamieci danych,
– szesc 8–bitowych portów wejscia–wyjscia (**P0**...**P5**),
– 8–bitowy port wejsciowy dla wejsc cyfrowych i analogowych (**P6**),
– trzy 16–bitowe zegary-liczniki,
– 8–bitowy programowalny przetwornik analogowo–cyfrowy,
– 16–bitowy uklad typu watchdog,
– wbudowany czteropoziomowy system przerwan obslugujacy 12 zródel: 7 zewnetrznych i 5 wewnetrznych,
– 256–bajtowy stos,
– port szeregowy umozliwiajacy dwustronna komunikacje.

Schemat blokowy struktury mikrokontrolera **SAB 80C535** przedstawiono na rysunku 2.1.

Mikrokontroler wymaga napiecia zasilajacego V_{cc}=5 V ± 10% i V_{ss}=0 V i jest umieszczony w obudowie PLCC 68. Producent oferuje szeroki zakres czestotliwosci rezonatorów kwarcowych taktujacych procesor (0,5...12 MHz) i szeroki zakres temperatury pracy (–40...+70 °C lub w niektórych wersjach +110°C).



Rys.2.1. Schemat blokowy mikrokontrolera SAB 80C535

Mikrokontrolery **SAB 80C535** nie posiadaja wewnetrznej pamieci *ROM* i w porównaniu z procesorem **INTEL 8051/52** maja rozszerzone funkcje wielu układów wewnetrznych oraz dodane nowe wewnetrzne uklady:

– programowanie szybkosci transmisji szeregowej moze odbywac sie bez wykorzystania licznika **T1**,
– licznikowi **T2** zostaly dodane funkcje zwiazane z modulacja szerokosci impulsów MSI (ang. *PWM – Pulse Width Modulation*) i synchroniczna zmiana wartosci czterech najmniej znaczacych linii portu **P1** (**P1.0**...**P1.3**),
– podwyzszono czestotliwosc rezonatora kwarcowego,
– dodano dwa uniwersalne porty wejscia–wyjscia **P4** i **P5**,
– dodano 8-bitowy port wejsciowy **P6** dla wejsc cyfrowych i analogowych,
– wprowadzono 8–bitowy przetwornik analogowo–cyfrowy (A/C) z wewnetrznym ukladem próbkujaco–pamietajacym (ang. *sample & hold S&H*) i programowalnymi podzakresami pomiarowymi,
– wprowadzono uklad nadzorujacy wykonywane programy tzw. „watchdog”,
– rozszerzono tryby redukcji mocy pobieranej przez mikrokontroler.

Pamiec wewnetrzna *RAM* jest zorganizowana w taki sam sposób jak w procesorach *8051/52*. Znacznej rozbudowie ze wzgledu na nowe funkcje i nowe wewnetrzne uklady uległy rejestry specjalne **SFR**.

W mikrokontrolerach **SAB 80C535** struktura dodatkowych portów **P4** i **P5** jest taka sama jak w procesorze *8051/52*. Linie dodatkowego portu **P6** sa standardowymi wejsciami portu i wejsciami multipleksera analogowego współpracujacego z przetwornikiem analogowo–cyfrowym.

Liczniki **T0** i **T1** oraz zasada ich programowania nie uległy zmianie natomiast licznik **T2** w mikrokontrolerach **SAB 80C535** jest zupełnie innym rozwiazaniem niz w procesorach *8051/52*. Mozliwe jest wprowadzanie licznika **T2** w tryb porównania wykorzystywany do generowania impulsów o programowanym wypełnieniu (MSI) oraz automatyczny wpis wartosci poczatkowej i zapamietanie wartosci chwilowej.

Nowym elementem w strukturze mikrokontrolera jest przetwornik analogowo-cyfrowy oraz watchdog. Rozdzielczosc przetwornika wynosi 8 bitów, jednak mozna ja

powiekszyc do 10 bitów dzieki mozliwosci zaprogramowania jednego z szesnastu podzakresów pomiarowych. Wymaga to dwukrotnego pomiaru napiecia: najpierw w pelnym zakresie pomiarowym, a nastepnie w zmniejszonym. Na podstawie obu pomiarów oblicza sie wartosc zmierzonego napiecia. Analogowy 8–wejsciowy multiplekser (MUX), poprzedzajacy przetwornik A/C, umozliwia pomiar napiecia w 8 róznych kanalach pomiarowych. Czas przetwarzania przetwornika wynosi 13 μs, co stanowi 13 cykli maszynowych.

Watchdog jest sprzetowym zabezpieczeniem programu przed zaklóceniami, które moga spowodowac nie przewidziana zmiane realizowanego programu. Zabezpieczenie jest realizowane za pomoca 16–bitowego licznika, który raz uruchomiony nie moze zostac zatrzymany. Przepelnienie licznika powoduje wewnetrzne zerowanie mikrokontrolera. Aby nie doszlo do takiej sytuacji, licznik musi byc cyklicznie, programowo zerowany.

W mikrokontrolerze **SAB 80C535** zostaly znacznie rozbudowane mozliwosci redukcji pradu przez niego pobieranego. W najbardziej oszczednym trybie redukcji mocy prad pobierany z zasilacza V_{cc} jest zredukowany do wartosci mniejszej niz I_{cc}=50 μA.

Szczególowy opis budowy i programowania poszczególnych ukladów wewnetrznych mikrokontrolera **SAB 80C535** przedstawiono w rozdziale 4.

2.2. Organizacja pamieci

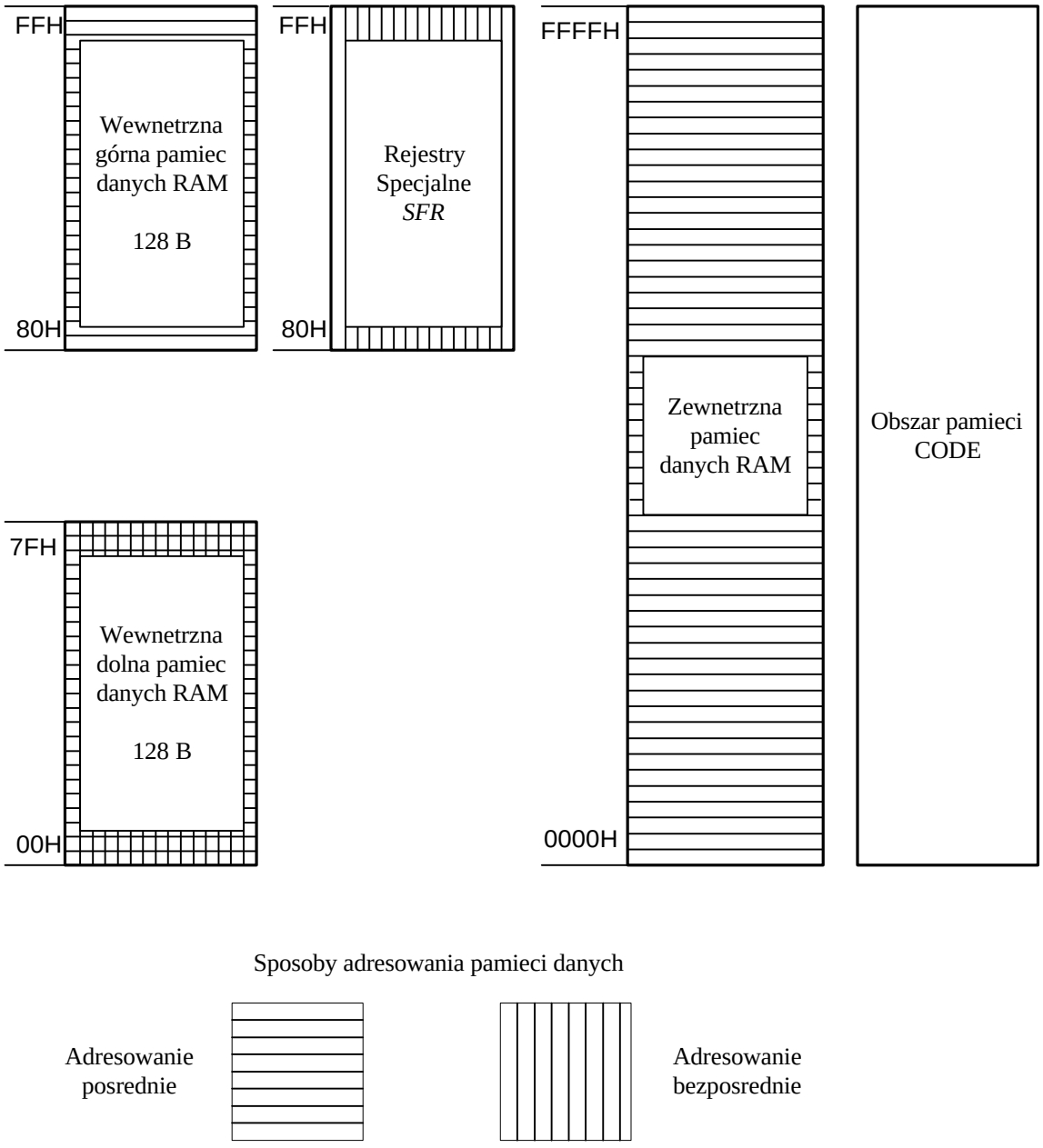
Na rysunku 2.3 przedstawiono mape pamieci mikrokontrolera **SAB 80C535.** Przez sposób zakreskowania (linie poziome lub pionowe) oznaczono w poszczególnych blokach pamieci sposób adresowania.

Pamiec mikrokontrolera podzielona jest funkcjonalnie na nastepujace bloki:

⇒ Pamiec programu

Mikrokontroler **SAB80C535** nie jest wyposazony w wewnetrzna pamiec *ROM*. Nalezy zatem korzystac z zewnetrznej pamieci programu, przy czym nalezy uaktywnic

obsluge zewnetrznej pamieci programu przez podanie na linie procesora ***EA#*** potencjalu niskiego (logiczne 0).



Rys.2.3. Mapa pamieci danych i rejestrów specjalnych *SFR*

Pamiec programu moze byc zapisana jedynie, gdy pokrywa sie z obszarem zewnetrznej pamieci danych *RAM*, przy czym obszar 03H do 93H w pamieci programu jest uzywany dla programów obslugi przerwan (w obszarze tym umieszczone sa adresy

wektorów przerwan). W obszarze pamieci programu mozliwe jest umieszczanie danych, do których dostep jest mozliwy jedynie poprzez wykorzystanie specjalnego rozkazu *MOVC.*

Maksymalna objetosc pamieci programu moze wynosic 64 kB, przy czym organizacyjnie podzielona jest na strony o rozmiarze 2 kB kazda. Pamiec programu adresowana jest **posrednio** poprzez rejestr bazowy, którym moze byc 16–bitowy wskaznik danych **DPTR**, lub 16–bitowy licznik rozkazów **PC**, przy czym bity 0–10 okreslaja adres na stronie, natomiast bity 11–15 numer strony pamieci. Stronicowanie pamieci jest zwiazane z ewentualnym uzyciem rozkazów *ACALL* i *AJMP*, poniewaz umozliwiaja one absolutny skok jedynie w obrebie 2 kB pamieci. Nalezy pamietac, ze w adresowaniu posrednim pamieci programu (z wykorzystaniem rozkazu *MOVC*) adres zródlowy jest suma zawartosci akumulatora (traktowanej jako liczba dwójkowa bez znaku z zakresu 0–255) i 16–bitowego rejestru bazowego.

⇒ Pamiec danych

Pamiec danych obejmuje obszar pamieci **wewnetrznej** *RAM* i obszar pamieci **zewnetrznej** *RAM* (rys. 2.4). Wewnetrzna pamiec danych ma pojemnosc 256 bajtów. Organizacyjnie pamiec ta podzielona jest na dwie partycje po 128 bajtów: dolna, tzw. dolny obszar pamieci uzytkowej (adresy 0H...7FH) oraz górna (adresy 80H...FFH), w której umieszczono zarówno górny obszar pamieci uzytkowej, jak i rejestry specjalne **SFR**. Ze wzgledu na przyjety podzial pamieci, wyróznia sie nastepujace tryby adresowania **pamieci danych**:

Wewnetrzna pamiec danych

- **Adresowanie przez nazwe rejestru**

Adresowanie przez nazwe rejestru oznacza, ze w rozkazie przeslania mozna uzyc bezposrednio nazwy rejestru, np. *MOV R0,#20H*.

Ten tryb adresowania dotyczy nastepujacych rejestrów:

– akumulatora **A** (**ACC**),

– rejestrów roboczych **R0**...**R7** z aktywnego banku rejestrów,
– wskaznika danych **DPTR**.

- **Adresowanie bezposrednie**

Adresowanie bezposrednie, w którym wystepuje 8-bitowy adres rozkazu obejmuje nastepujace obszary pamieci:
– obszar pamieci uzytkowej *RAM* o adresach 0H...7FH,
– obszar rejestrów specjalnych *SFR* (adresy 80H...FFH).

Przykladem adresowania bezposredniego moze byc rozkaz *MOV A,20#*.

- **Adresowanie posrednie zawartoscia rejestru**

Adresowanie posrednie dotyczy calego obszaru pamieci uzytkowej *RAM* (adresy 0H...FFH), przy czym adres zawarty jest w rejestrach **R0** lub **R1** z aktywnego banku rejestrów.

Jako przyklad adresowania posredniego moze posluzyc rozkaz *MOV @R0,A*
Rejestr **R0** zawiera adres docelowy.

- **Adresowanie bitowe (bezposrednie bitów)**

Ten tryb adresowania dotyczy obszaru pamieci uzytkowej *RAM* zajmujacego przestrzen adresowa 20H...2FH i czesc rejestrów specjalnych *SFR*. Rejestry te oznaczone sa gwiazdka w tabeli 2.2.

Przyklad adresowania bitowego: *SETB P1.1*

Zewnetrzna pamiec danych

Zewnetrzna pamiec danych *RAM* moze miec maksymalna pojemnosc 64 kB i adresowana jest **posrednio** zawartoscia rejestru adresowego z wykorzystaniem rozkazu *MOVX*. Jako rejestr adresowy moze byc wykorzystany:
– wskaznik danych **DPTR**: 16–bitowy adres wysylany jest przez porty **P0** i **P2**, co pozwala zaadresowac do 64 kB zewnetrznej pamieci danych,

– rejestr roboczy **R0** lub **R1**: 8–bitowy adres pozwala zaadresowac 256 bajtów danych. Adres wysylany jest przez port **P0**, co pozwala wykorzystac port **P2** jako zwykly port wejscia–wyjscia. Przyklad adresowania posredniego zewnetrznej pamieci danych: *MOV DPTR,#20H*

MOVX @DPTR,A

⇒ **Obszar pamieci CODE**

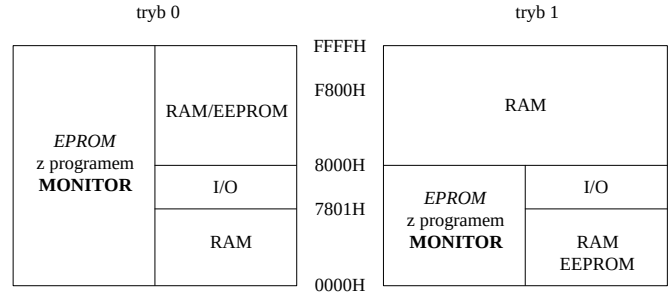
Zewnetrzna pamiec *EPROM* z programem **MONITOR** (patrz rozdzial 3.3) oraz z interpreterem jezyka *BASIC* jest ulokowana w obszarze pamieci *CODE***.** W zaleznosci od wybranej konfiguracji tej pamieci inny jest adres poczatkowy segmentu zewnetrznej pamieci danych *RAM*, w którym lokowany jest program uzytkownika:

– adres 0000H dla konfiguracji EP007,

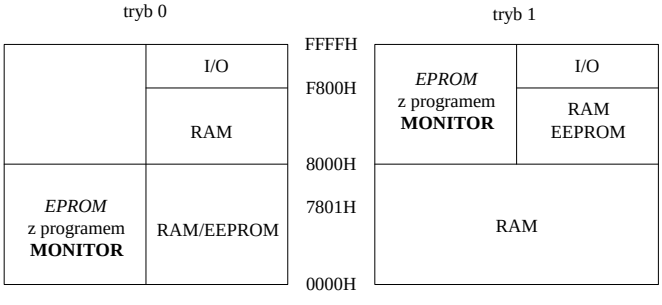
– adres 8000H dla konfiguracji EP 009.

Dodatkowo, w kazdej z wymienionych konfiguracji, obszar pamieci *CODE* moze byc podzielony wedlug jednego z dwóch dostepnych trybów: trybu 0 i trybu 1.

Na rysunku 2.2 przedstawiono rodzaje konfiguracji pamieci *CODE* wraz z podzialem na tryby. W zestawach dydaktycznych wykorzystywanych w laboratorium zastosowano konfiguracje typu EP007 i EP009 pracujace w trybie 1.



Konfiguracja EP007

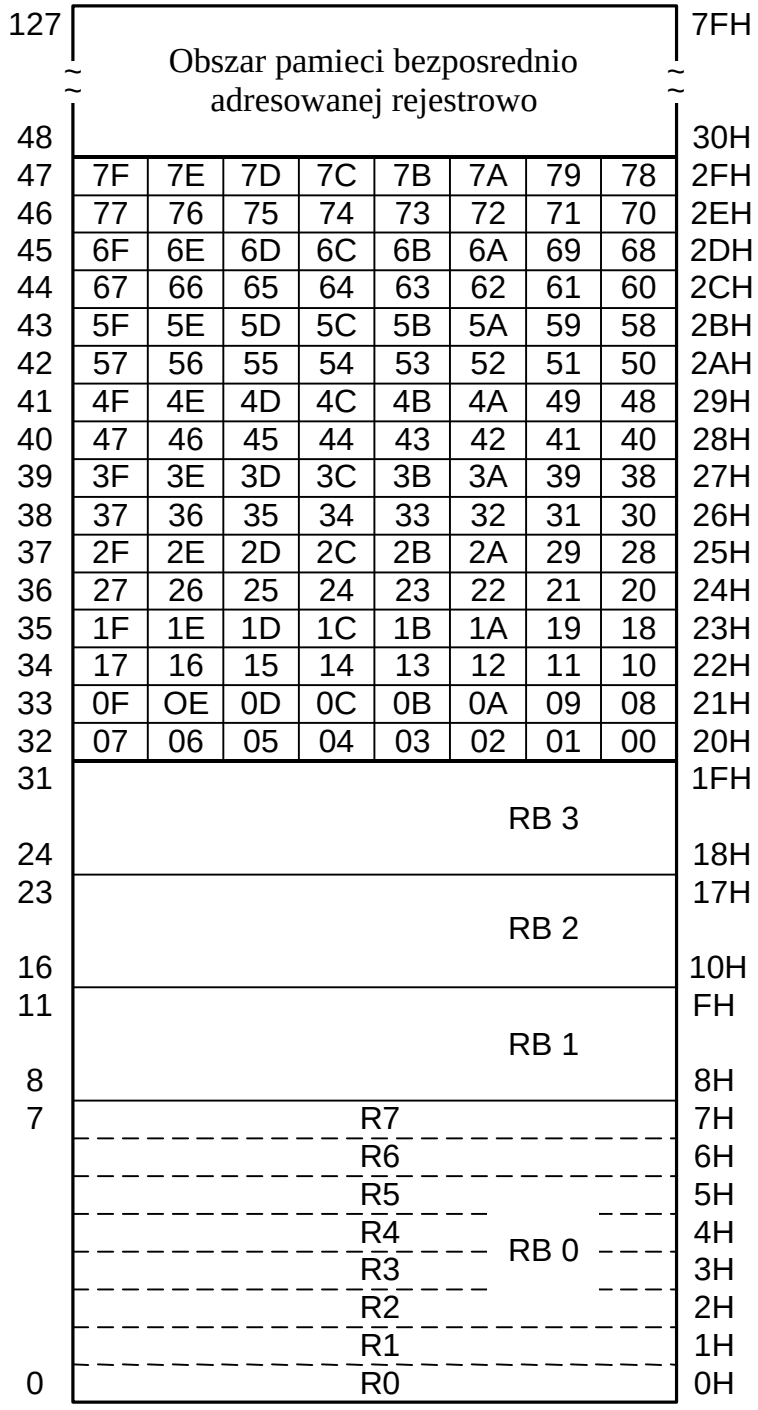


Konfiguracja EP009

Rys.2.2. Konfiguracja obszaru pamieci *CODE*

2.3. Rejestry robocze i rejestry specjalne *SFR*

Rejestry robocze zajmuja przestrzen adresowa od 0H do 1FH w wewnetrznej pamieci *RAM* (rys. 2.4).



Rys.2.4. Mapa dolnej partycji wewnetrznej pamieci danych *RAM*

Rejestry podzielone sa na cztery banki (**RB0**...**RB3**) po 8 rejestrów **R0**...**R7**, przy czym jednoczesnie moze byc uzyty tylko jeden bank rejestrów (patrz tabela 2.1). Aktywny bank rejestrów wybierany jest w rejestrze **PSW** przez odpowiednie ustawienie bitów **RS1** i **RS0**.

Rejestr **PSW** adres **0D0H**

CY	AC	F0	**RS1**	**RS0**	OV	F1	P

- **CY** – znacznik przeniesienia, ustawiany lub zerowany sprzetowo,
- **AC** – znacznik przeniesienia pomocniczego, ustawiany lub zerowany sprzetowo,
- **F0** – znacznik ogólnego przeznaczenia, zmieniany wylacznie programowo,
- **RS1**, **RS0** – Wybór aktywnego banku rejestrów:

Tabela 2.1. Sterowanie bankiem rejestrów

RS1	RS0	Aktywny bank rejestrów
0	0	Bank 0, obszar adresowy 00H-07H
0	1	Bank 1, obszar adresowy 08H-0FH
1	0	Bank 2, obszar adresowy 10H-17H
1	1	Bank 3, obszar adresowy 18H-1FH

- **OV** – znacznik przepelnienia, ustawiany lub zerowany sprzetowo,
- **F1** – znacznik ogólnego przeznaczenia, zmieniany wylacznie programowo,
- **P** – znacznik parzystosci, ustawiany lub zerowany sprzetowo w kazdym cyklu maszynowym. Wskazuje na liczbe jedynek w akumulatorze:

  P=0 – parzysta liczba jedynek,

  P=1 – nieparzysta liczba jedynek.

Rejestry specjalne **SFR** zajmuja obszar pamieci o adresach w zakresie 80H...FFH w wewnetrznej pamieci *RAM*. Generalnie rejestry te adresowane sa bajtowo i maja okreslone nazwy symboliczne i okreslone adresy, jednak niektóre bity rejestrów moga byc adresowane bitowo. Wyjatkiem jest rejestr akumulatora oznaczany **A** lub **ACC**, oraz bit (znacznik) przeniesienia oznaczany **C** lub **CY**. W tabeli 2.2 przedstawiono wykaz rejestrów specjalnych mikrokontrolera.

Tabela 2.2. Wykaz rejestrów specjalnych *SFR*

Rejestr	Nazwa	Adres	Stan po *RESET#*
* P0	Port 0	80H	0FFH
SP	Wskaznik stosu	81H	07H
DPL	Mniej znaczace bity wskaznika danych	82H	00H
DPH	Bardziej znaczace bity wskaznika danych	83H	00H
PCON	Rejestr kontroli mocy	87H	0XXX XXXXB
* TCON	Rejestr sterujacy timerem	88H	00H
TMOD	Rejestr wyboru trybu pracy timera	89H	00H
TL0	Mniej znaczace bity timera 0	8AH	00H
TL1	Mniej znaczace bity timera 1	8BH	00H
TH0	Bardziej znaczace bity timera 0	8CH	00H
TH1	Bardziej znaczace bity timera 1	8DH	00H
* P1	Port 1	90H	0FFH
* SCON	Rejestr sterujacy praca lacza szeregowego	98H	00H
SBUF	Rejestr bufora lacza szeregowego	99H	niezdefiniowany
* P2	Port 2	0A0H	0FFH
* IENO	Rejestr 0 odblokowania przerwan	0A8H	00H
IP0	Rejestr 0 priorytetu przerwan	0A9H	00H
* P3	Port 3	0B0H	0FFH
* IEN1	Rejestr 1 odblokowania przerwan	0B8H	00H
IP1	Rejestr 0 priorytetu przerwan	0B9H	00H
* IRCON	Rejestr kontroli przerwan	0C0H	00H
CCEN	Rejestr sterujacy komparatorami	0C1H	00H
CCL1	Mniej znaczace bity komparatora 1	0C2H	00H
CCH1	Bardziej znaczace bity komparatora 1	0C3H	00H
CCL2	Mniej znaczace bity komparatora 2	0C4H	00H
CCH2	Bardziej znaczace bity komparatora 2	0C5H	00H
CCL3	Mniej znaczace bity komparatora 3	0C6H	00H
CCH3	Bardziej znaczace bity komparatora 3	0C7H	00H
* T2CON	Rejestr sterujacy praca licznika T2	0C8H	00H
CRCL	Mniej znaczace bity rejestru CRC	0CAH	00H
CRCH	Bardziej znaczace bity rejestru CRC	0CBH	00H
TL2	Mniej znaczace bity licznika T2	0CCH	00H
TH2	Bardziej znaczace bity licznika T2	0CDH	00H
* PSW	Slowo statusu	0D0H	00H
* ADCON	Rejestr sterowania przetwornikiem a/c	0D8H	00H
ADDAT	Rejestr wyniku pomiaru przetwornika a/c	0D9H	00H
DAPR	Rejestr wyboru podzakresów pomiarowych przetwornika a/c	0DAH	00H
P6	Port 6	0DBH	niezdefiniowany
* ACC	Akumulator	0E0H	00H
* P4	Port 4	0E8H	0FFH
* B	Rejestr B	0F0H	00H
* P5	Port 5	0F8H	0FFH

* – rejestry adresowane bajtowo i bitowo

3. OPIS SRODOWISKA PROGRAMOWEGO

3.1. Opis jezyka programowania asm535

Kazdy mikroprocesor, a wiec i mikrokontroler dziala wedlug odpowiedniego programu, umieszczonego w pamieci stalej (*ROM*, *EPROM*), znajdujacej sie na zewnatrz lub wewnatrz ukladu.

Do napisania odpowiedniego programu, jak przedstawiono to w rozdziale 1, niezbedna jest znajomosc jezyka niskiego poziomu – *asemblera*.

W zaleznosci od zastosowanego rozwiazania sprzetowego dysponuje sie róznymi srodowiskami programowymi (pakietami oprogramowania). W przypadku mikrokontrolera **SAB 80C535** zostanie omówione srodowisko programowe **asm535** firmy *Keil* wraz z modulem dydaktycznym **MINICON** firmy *Micromax*. Przyjecie takiego srodowiska programowego jest zwiazane z wyposazeniem sprzetowym laboratorium.

Srodowisko programowe ***asm535*** nie rózni sie istotnie od innych, umozliwiajacych zaprogramowanie mikrokontrolera. Istotna zaleta sa male wymagania sprzetowe: do poprawnej pracy wystarczy komputer *PC 286* i system operacyjny *MS-DOS* w wersji 2.11 lub nowszej.

Przy pisaniu programu zródlowego nalezy przestrzegac kilku podanych nizej zasad:

1. Linia programu zródlowego ma postac:

Nazwa etykiety: mnemonik rozkazu operand,operand ; komentarz

np.:

```
PETLA:  DJNZ  R0,PETLA  ;  opóznienie wykonania programu
```

2. Mozna korzystac z trzech formatów przedstawiania liczb, przy czym liczby w formacie szesnastkowym i binarnym musza byc poprzedzone identyfikatorem:

   $ – liczba w szesnastkowa (heksadecymalna), np. $0D3F,

   % – liczba binarna, np. %00001111,

   – liczby dziesietne bez znaku poprzedzajacego, np. 50.

3. Liczby pisane w kodzie szesnastkowym i zawierajace znaki A...F musza zaczynac

sie cyfra 0, np. 0F3H.

4. Jezeli w rozkazie wpisywana jest wartosc liczbowa, to musi byc poprzedzona znakiem #. W przeciwnym razie kompilator potraktuje ta wartosc jako adres komórki pamieci wewnetrznej *RAM* lub *SFR*, np.

```
MOV A,$32   ;wpisanie do A zawartosci komórki pamieci wewnetrznej
            ;RAM o adresie 32H
MOV A,#$32  ;wpisanie do A liczby 32H
```

5. Symbole predefiniowane, którymi sa nazwy rejestrów, slowa kluczowe rozkazów itp. sa slowami zastrzezonymi i nie moga byc uzyte jako nazwa etykiety.
6. Pozostale symbole (nazwy okreslajace adresy lub wartosci liczbowe) nie moga miec wiecej niz 31 znaków, przy czym pierwszym znakiem musi byc litera, np.

```
C_DA          BYTE      ($20)  ;przypisanie symbolowi C_DA adresu 20H
                                 ;w pamieci RAM
CONV_E_100    EQU       $100   ;przypisanie symbolowi CONV_E_100
                                 ;wartosci 100H
```

7. Etykiety musza zaczynac sie od lewego marginesu i musza byc zakonczone dwukropkiem.
8. Komentarze musza byc poprzedzone srednikiem. Podczas kompilacji programu sa ignorowane, widoczne sa jedynie w zbiorze zawierajacym listing programu, np.:

```
; to jest komentarz
```

9. Program zródlowy powinien posiadac rozszerzenie .asm.

Po napisaniu programu w jezyku asemblera, nalezy skompilowac go wykorzystujac kompilator **asm535**. Wywolanie kompilacji ma postac:

asm535 nazwa_programu_zródlowego.asm/H/L/S

przy czym parametry **H**, **L**, **S** nie musza wystepowac lacznie i oznaczaja:

H – tworzenie programu w kodzie maszynowym .hex, bedacego zbiorem roboczym programu **MONITOR** do rzeczywistej pracy z modulem **MINICON**,

L – tworzenie listingu programu, tzn. zbioru z rzeczywistymi adresami skompilowanego programu zródlowego,

S – tworzenie zbioru z tablicami symboli uzywanych w programie zródlowym.

Ostatecznym produktem kompilacji, jezeli zbiór .asm nie zawieral bledów, jest zbiór o rozszerzeniu .obj.

⇒ Podstawowe komendy asemblera

ORG

<**ORG** adres_poczatku_segmentu>

Dyrektywa powoduje ustalenie adresu poczatkowego segmentu absolutnego pamieci, od którego bedzie zaczynal sie program, przy czym ponizej adresu poczatkowego nie mozna umieszczac modulów relokowalnych.

Przyklad: ORG $100 ;adres poczatku segmentu 100H, w obszarze
;0...100H nie mozna umieszczac modulów reloko-
;walnych.

EQU

<symbol **EQU** wyrazenie>

Dyrektywa powoduje przypisanie nazwie „symbol" wartosci „wyrazenie", przy czym nie wolno w danym module drugi raz nadawac nowej wartosci nazwie „symbol".

Przyklad: WSP EQU $100 ; nazwie WSP zostala nadana wartosc 100H

SET

<symbol **SET** wyrazenie>

Dyrektywa powoduje przypisanie nazwie „symbol" wartosci „wyrazenie", przy czym w danym module mozna drugi raz nadawac nowa wartosc nazwie „symbol".

Przyklad: WSP1 SET $100 ; nazwie WSP1 zostala nadana wartosc 100H

BIT

<symbol **BIT** wyrazenie>

Dyrektywa powoduje przypisanie nazwie „symbol" wartosci „wyrazenie", która

jest adres bitu w obszarze pamieci wewnetrznej *RAM* lub rejestrów *SFR*, przy czym nie wolno w danym module drugi raz nadawac nowej wartosci nazwie „symbol". Wartosc „wyrazenie" musi byc z przedzialu (0...255)

Przyklad: DIODA BIT P1.5 ;nazwie DIODA zostal przypisany bit P1.5 ;(6 bit portu P1).

BYTE

<symbol **BYTE** (wyrazenie)>

Dyrektywa powoduje przypisanie nazwie „symbol" wartosci „wyrazenie", która jest adres bajtu w obszarze pamieci wewnetrznej RAM, przy czym nie wolno w danym module drugi raz nadawac nowej wartosci nazwie „symbol". Wartosc „wyrazenie" musi byc z przedzialu (0...255).

Przyklad: PORT BYTE ($90) ;symbolowi PORT zostal przypisany adres ;90H

DB

<etykieta: DB wyrazenie,wyrazenie,...>

Dyrektywa powoduje umieszczenie w kolejnych bajtach pamieci programu kolejnych wartosci wskazanych przez „wyrazenie", przy czym wartosc „wyrazenie" musi byc z przedzialu (0...255). Zamiast wyrazenia numerycznego mozna umiescic lancuch znaków dowolnej dlugosci. Lancuch znaków musi byc umieszczony w apostrofach.

Przyklad:
```
TABLICA:  DB  $25,$43,$55
TEKST:    DB 'Mikroprocesor'
MIKRO:    DB 'Mikrokontroler ''51'
```

DW

<etykieta: DW wyrazenie,wyrazenie,...>

Dyrektywa powoduje umieszczenie w kolejnych bajtach pamieci programu kolejnych 16-bitowych wartosci wskazanych przez „wyrazenie", przy czym wartosc „wyrazenie" musi byc z przedzialu (0...65535). Zamiast wyrazenia numerycznego mozna umiescic lancuch znaków dowolnej dlugosci. Lancuch znaków

musi byc umieszczony w apostrofach.

Przyklad: TABLICA: DW $254C,$1024

TEKST: DW ‘Mikroprocesor 16-bitowy’

3.2. Program MONITOR-51 i MONTERM

3.2.1. Opis programu MONITOR-51

⇒ Wlasciwosci, wymagania sprzetowe i konwencja wprowadzania danych

Program **MONITOR-51** przystosowany jest do pracy z mikrokomputerami, w których zastosowano procesory rodziny **MCS-51**. Komunikacja pomiedzy uzytkownikiem a programem odbywa sie poprzez zlacze szeregowe, dlatego do pracy z **MONITOR-51** oprócz zwyklego terminala jest potrzebny program komunikacyjny **MONTERM**, opisany w rozdziale 3.3.2.

MONITOR-51 wyposazony jest w szereg komend umozliwiajacych:

- wyswietlanie zawartosci pamieci w postaci heksadecymalnej i ASCII,
- zmiane zawartosci pamieci,
- wypelnianie pamieci wartoscia stala,
- wyswietlanie i zmiane zawartosci rejestrów roboczych i rejestrów specjalnych *SFR*,
- disasemblacje programu, czyli wyswietlanie zawartosci pamieci programu w postaci mnemoników rozkazów,
- liniowa asemblacje,
- prace w czasie rzeczywistym z mozliwoscia ustawienia do 10 stalych i 1 tymczasowego *punktu wstrzymania* (ang. *breakpoint*).,
- prace krokowa z mozliwoscia traktowania podprogramów jako pojedynczego kroku,
- wczytywanie i zapisywanie programów w formacie Intel-Standard-Hex.

Do poprawnej pracy programu **MONITOR-51** musza byc spelnione nastepujace warunki:

- procesor rodziny **MCS-51,**

– 8 kB pamieci *EPROM*,
– minimum 2 kB pamieci *RAM* w konfiguracji wg Von Neumanna (pamiec programu i pamiec danych umieszczone sa we wspolnej przestrzeni adresowej),
– zlacze szeregowe do przylaczenia terminala.

Ponadto, podczas pracy program **MONITOR-51** wykorzystuje nastepujace elementy systemu:

– zlacze szeregowe,
– dodatkowe 6 bajtów stosu do testowania programów uzytkowych,
– 512 bajtów zewnetrznej pamieci danych *RAM*,
– 8 kB zewnetrznej pamieci programu *EPROM* (zajmowanej przez program **MONITOR-51** i interpreter jezyka *BASIC*).

Komendy oraz dane do ich wykonania wprowadzanie sa wg odpowiedniej konwencji, przedstawionej ponizej:

DUZE LITERY	– wszystkie dane napisane duzymi literami musza zostac wprowadzone dokladnie ze wzorem, przy czym wprowadzac m.ozna zarówno duze jak i male litery
< >	– pole ograniczone znakami < > wskazuje na zmienny parametr, który nalezy wpisac zgodnie ze specyfikacja wymienionej komendy
[ ]	– dane zapisane w nawiasach prostokatnych sa opcjonalne i nie musza byc wprowadzone do prawidlowego wykonania komendy, przy czym nie wolno wprowadzac samych nawiasów
...	– trzy kropki wskazuja, ze poprzedni element moze zostac powtórzony wiele razy.
<cr>	– symbol ten oznacza nacisniecie klawisza *ENTER* i wykonanie przez **MONITOR-51** danej komendy

⇒ Praca z programem MONITOR-51

Przed uruchomieniem programu **MONITOR-51** nalezy polaczyc model dydaktyczny (uruchomieniowy) *MINICON* z terminalem lub komputerem *PC* symulujacym terminal. Przez pojecie terminal nalezy rozumiec urzadzenie umozliwiajace programowanie modelu dydaktycznego i jego komunikacje ze srodowiskiem programowym.

Po zalaczeniu zasilania na monitorze komputera wyswietlona zostaje nazwa programu i znak firmowy:

MONITOR-51 Vx.y
(c) KEIL ELEKTRONIK GmbH 1987
#

Vx,y jest aktualnym numerem wersji programu, a znacznik **#** wskazuje na poprawne podlaczenie modelu dydaktycznego z terminalem (lub komputerem *PC*) i gotowosc programu **MONITOR-51** do przyjmowania komend.

W celu uproszczenia edycji wprowadzanych danych oraz latwego sterowania wyprowadzaniem danych na ekranie, **MONITOR-51** umozliwia poslugiwanie sie nastepujacymi skrótami klawiszowymi:

CTRL-D, CTRL-F lub **DEL**	– kasowanie znaku w miejscu kursora
CTRL-H lub **Backspace**	– kasowanie znaku przed kursorem
Ctrl-L lub ¬	– przesuniecie kursora w lewo
Ctrl-R lub ®	– przesuniecie kursora w prawo
Ctrl-X	– skasowanie calego wiersza
<cr>	– wykonanie linii komendy do aktualnej pozycji kursora, przy czym pozostale znaki zostaja zignorowane i skasowane
ESC	– edycja i wykonanie ostatnio wprowadzonej komendy, przy czym: ESC jako pierwszy znak – wywolanie poprzedniej komendy i jej edycja, ESC po pierwszym znaku – wykonanie calej linii
Ctrl-O	– przesuniecie kursora na poczatek lub koniec linii
Ctrl-S	– zatrzymanie wyprowadzania danych na ekran i zawieszenie dzialania aktualnie wykonywanej komendy
Ctrl-Q	– zniesienie dzialania komendy **Ctrl-S** (wykonywanie ostatniej komendy jest kontynuowane)
Ctrl-C	– przerwanie dzialania ostatniej komendy i powrót na poziom wprowadzania komend, przy czym MONITOR-51 zglasza komunikat: *** TERMINATED *** #

⇒ Komendy programu MONITOR-51

MONITOR-51 akceptuje parametry wejsciowe podane w postaci heksadecymalnej (cyfry 0...9 i litery A...F), przy czym jesli komenda wymaga wprowadzenia wiecej niz jednego parametru, to sa one oddzielone spacja lub przecinkiem, np.:

#FILLX 3000,4000,1 <cr> lub

#FILLX 3000 4000 1 <cr>

Ponizej przedstawiono wykaz komend programu **MONITOR-51** , przy czym w ich opisie uzyto nastepujacych pojec:

add	– adres w zakresie danego typu pamieci mikrokontrolera
range	– dwie wartosci adresu okreslajace przedzial w przestrzeni adresowej mikrokontrolera, przy czym adresy oddzielone sa przecinkiem lub spacja
val	– zawartosc komórki pamieci, przy czym dopuszczalne wartosci wynosza: 0 i 1 dla przestrzeni bitowej, 0...FFH dla pozostalej przestrzeni
num	– liczba wykonywanych kroków przy pracy krokowej, przy czym dopuszczalna wartosc miesci sie w zakresie 0..FFFFH
bp	– numer punktu wstrzymania (od 0 do 9)
add bp	– adres tymczasowego punktu wstrzymania w obszarze pamieci programu, przy czym punkt aktywny jest tylko przy wykonywaniu komendy GO
startadd	– adres poczatkowy zakresu pamieci
endadd	– adres koncowy zakresu pamieci

Komendy wyswietlania i zmiany zawartosci pamieci

MONITOR-51 umozliwia dostep do nastepujacych rodzajów pamieci mikrokontrolera (zgodnie a architektura rodziny **MCS-51**):

- pamiec PROGRAMU,
- adresowana bezposrednio wewnetrzna pamiec DANYCH,
- adresowana posrednio wewnetrzna pamiec DANYCH(I),
- zewnetrzna pamiec DANYCH(X),
- pamiec REJESTRÓW,
- obszar pamieci adresowanej bitowo.

Wyswietlanie zawartosci pamieci (Display)

DC [startadd [endadd]] – wyswietlanie pamieci PROGRAMU,

DD [startadd [endadd]] – wyswietlanie pamieci DANYCH,

DI [startadd [endadd]] – wyswietlanie pamieci DANYCH(I),

DX [startadd [endadd]] – wyswietlanie pamieci DANYCH(X),

DB [startadd [endadd]] – wyswietlanie pamieci bitowej.

Uwaga: Opuszczenie adresu poczatkowego spowoduje uzycie adresu koncowego ostatniej komendy **Display**, natomiast opuszczenie adresu koncowego powoduje wyswietlenie 4 wierszy ekranu.

Zmiana zawartosci komórki pamieci (Enter)

EC [add] – zmiana zawartosci komórki pamieci PROGRAMU,

ED [add] – zmiana zawartosci komórki pamieci DANYCH,

EI [add] – zmiana zawartosci komórki pamieci DANYCH(I),

EX [add] – zmiana zawartosci komórki pamieci DANYCH(X),

EB [add] – zmiana zawartosci komórki pamieci bitowej.

Uwaga: Opuszczenie adresu spowoduje, ze zostanie uzyty adres 0. Po wykonaniu komendy na monitorze pojawia sie adres komórki pamieci i jej aktualna wartosc, a program oczekuje na wprowadzenie nowej wartosci. Przy wprowadzaniu dostepne sa nastepujace opcje:

- **<cr>** – wcisniecie klawisza *ENTER* bez podania nowej wartosci pozostawia wartosc komórki nie zmieniona i nastepuje przeskok do nastepnego adresu,
- (znak kropki) **. <cr>** – wprowadzenie znaku **.** (kropka) powoduje zakonczenie dzialania komendy **Enter** i przejscie na poziom interpretera komend,
- **wartosc <cr>** – wprowadzenie nowej wartosci i przeskok do nastepnego adresu.

Wypelnianie pamieci stala wartoscia (Fill)

FILLC startadd, endadd, val – wypelnianie pamieci PROGRAMU,

FILLD startadd, endadd, val – wypelnianie pamieci DANYCH,

FILLI startadd, endadd, val – wypelnianie pamieci DANYCH(I),

FILLX startadd, endadd, val – wypelnianie pamieci DANYCH(X),

FILLB startadd, endadd, val – wypelnianie pamieci bitowej.

Uwaga: Adresy i wartosci **val** moga przyjmowac dowolne wartosci w dozwolonym przedziale. Niedozwolone jest wypelnienie zawartosci rejestru **SFR** zwiazanego ze zlaczem szeregowym, a takze nie wolno wypelniac pamieci zewnetrznej wykorzystywanej przez **MONITOR-51**.

Wprowadzanie rozkazów do pamieci PROGRAMU (Assembly)

A [add] – wprowadzanie rozkazów mikrokontrolera w sposób liniowy (asemblacja liniowa)

Uwaga: Komenda **Assembly** (**A**) umozliwia wprowadzenia 1 linii rozkazu w postaci mnemoników. Operandy w rozkazach moga przyjmowac postac absolutna lub symboli z zakresu rejestrów specjalnych **SFR**. W przypadku opuszczenia adresu do wykonania komendy przyjety zostanie ostatnio uzywany adres. Po wywolaniu komendy dostepne sa nastepujace opcje:

- **<cr>** – wcisniecie klawisza *ENTER* pozostawia aktualny rozkaz nie zmieniony i nastepuje przeskok do nastepnego rozkazu,
- (znak kropki) **. <cr>** – wprowadzenie znaku kropki (**.**) powoduje zakonczenie dzialania komendy **A** i przejscie na poziom interpretera komend,
- **rozkaz <cr>** – podany rozkaz zastepuje poprzedni i nastepuje przeskok do nastepnego adresu. Podanie niewlasciwego symbolu powoduje wyswietlenie komunikatu bledu i nalezy powtórzyc czynnosc wprowadzania rozkazu.

Disasemblacja pamieci PROGRAMU (Unassembly)

U [startadd [,endadd]] – disasemblacja pamieci PROGRAMU

Uwaga: Komenda **Unassemby** (**U**) powoduje wyswietlenie na ekranie zawartosci pamieci PROGRAMU w postaci mnemoników asemblera. Jesli adresy poczatkowy

i koncowy nie zostana podane, to zostanie wyswietlonych 10 rozkazów od adresu koncowego ostatnio wykonywanej komendy **U**. Komenda **U** interpretuje równiez w sposób symboliczny obszar rejestrów specjalnych **SFR**.

Wyswietlanie i zmiana zawartosci rejestrów procesora (eXamine)

X <cr> – wyswietlenie zawartosci rejestrów,

X <nazwa_rejestru> <cr> – zmiana zawartosci podanego rejestru.

Uwaga: Jako parametr <nazwa_rejestru> mozna podac zarówno symbol dowolnego rejestru roboczego z aktywnego banku rejestrów, jak i symbol jednego z rejestrów specjalnych **SFR.** Komenda **X** bez parametru powoduje wyswietlenie aktualnej zawartosci rejestrów w nastepujacej formie:

RA	RB	R0	R1	R2	R3	R4	R5	R6	R7	PSW	DPTR	SP	PC
xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	---Rn---	xxxx	xx	xxxx

Podane symbole odpowiadaja nastepujacym rejestrom mikrokontrolera:

RA – akumulator,
RB – rejestr B,
R0..R7 – rejestry robocze z aktywnego banku rejestrów,
PSW – slowo statusu procesora,
DPTR – 16–bitowy wskaznik danych (DPH i DPL),
SP – wskaznik stosu,
PC – licznik programu,
xx, xxxx – aktualne wartosci w postaci heksadecymalnej.
Ponadto:

– znak ‘ - ‘ reprezentuje znaczniki nie ustawione,
– ustawione znaczniki przedstawione sa przez wyróznione litery w odpowiednich symbolach (**C**arry, **Ov**erflow, **F0**, **F1**, **A**uxilary, **P**arity),
– **Rn** okresla aktywny bank rejestrów (n=0,1,2,3).

Komendy sterujace punktami wstrzyman (ang. *break point*)

W trakcie testowania programu niejednokrotnie zachodzi potrzeba wstrzymania

dzialania programu np. w celu poszukiwania bledu. W zwiazku z tym **MONITOR-51** ma komendy umozliwiajace:

– definiowanie punktu wstrzymania,
– kasowanie punktu wstrzymania,
– wyswietlanie punktów wstrzyman,
– uaktywnienie punktu wstrzymania,
– czasowe wylaczenie punktu wstrzymania.

MONITOR-51 realizuje punkty wstrzyman przez automatyczne wstawienie do programu uzytkowego rozkazu *LCALL*. Zaleta tej metody jest to, ze nie wymaga zadnych dodatkowych elementów sprzetowych do wstrzymania dzialania programu. Wada tej metody jest fakt, ze punkty wstrzyman moga byc ustawione jedynie w pamieci *RAM*. Ponadto nie jest mozliwe ustawienie dwóch punktów wstrzyman w dwóch kolejnych bajtach, poniewaz rozkaz *LCALL* zajmuje 3 bajty. Dlatego tez przy definiowaniu punktów wstrzyman nalezy zwrócic uwage, aby rozkaz *LCALL* nie zastapil jakiegos istotnego rozkazu w programie uzytkownika. Sytuacji takiej mozna uniknac wstawiajac do programu uzytkowego w miejscu przewidywanego umieszczenia punktów wstrzyman 3 rozkazy puste *NOP*, które bez zadnych konsekwencji moga zostac nadpisane rozkazem *LCALL*. Innym rozwiazaniem jest wykonywanie programu w trybie pracy krokowej. Zjawisko definiowania punktu wstrzymania wyjasnia przyklad:

Zaklada sie, ze testowany program uzytkowy bez punktów wstrzyman ma postac:

```
1000   11   10      ACALL      1010
1002   NOP
1003   NOP
1004   NOP
1005   80   FC      SJMP       1000
  :     :
1010   22                      RET
```

Po zdefiniowaniu punktu wstrzymania rozkazem **BS 1002** program uzytkownika zostanie zmodyfikowany przez MONITOR-51 do nastepujacej postaci:

```
1000   11   xx      ACALL      1010      ;od adresu 1000 zapisany jest kod
1002   NOP                               ;rozkazu LCALL. Komórki o adresie
1003   XX                                ; 1001 i 1002 wypelnia adres
1004   XX                                ;wynikowy rozkazu LCALL
1005    xx   FC      SJMP       1000
  :      :
1010   22                    RET
```

Definicja punktu wstrzymania (BreakSet)

BS add – ustawienie punktu wstrzyman.

Uwaga: Rozkaz pod adresem punktu wstrzymania jest calkowicie wykonany. Nalezy tez pamietac, aby adres punktu wstrzymania byl adresem pierwszego bajtu kodu operacji danego rozkazu. Moga wystapic trzy przyczyny pojawienia sie bledu:

– ustawionych jest wiecej niz 10 punktów wstrzyman,

– punkt wstrzymania o podanym adresie jest juz zdefiniowany, lub naklada sie czesciowo z innym adresem punktu wstrzymania,

– adres punktu wstrzymania jest niewlasciwy.

Kasowanie punktu wstrzymania (BreakKill)

BK ALL – kasowanie wszystkich punktów wstrzyman,

BK bp [, bp, ...] – kasowanie wymienionych punktów wstrzyman.

Uwaga: Jesli jest jeden punkt wstrzymania do skasowania, to nalezy podac jego numer.

Wyswietlanie punktów wstrzyman (BreakList)

BL – wyswietlenie wszystkich punktów wstrzyman.

Uwaga: Wykonanie komendy powoduje wyswietlenie wszystkich punktów wstrzyman. Wyswietlane sa nastepujace informacje:

– numer punktu wstrzymania,

– status punktu wstrzymania (**ENA** (enabled) w przypadku gdy punkt wstrzymania jest aktywny lub **DIS** (disabled), gdy punkt wstrzymania jest nieaktywny),

– adres punktu wstrzymania.

Zablokowanie punktu wstrzymania (BreakDisable)

BD AL.L – zablokowanie wszystkich punktów wstrzyman,
BD bp [, bp, ...] – zablokowanie wymienionych punktów wstrzyman.

Uaktywnienie punktu wstrzymania (BreakEnable)

BE ALL – uaktywnienie wszystkich punktów wstrzyman,
BE bp [, bp, ...] – uaktywnienie wymienionych punktów wstrzyman.

Komendy testujace program

Start programu w trybie rzeczywistym (Go)

G [startadd [,endadd]] – start programu.

Uwaga: Komenda **G** powoduje start programu od adresu okreslone aktualna zawartoscia licznika programu lub od podanego adresu poczatkowego [startadd]. W przypadku gdy nie zostanie podany adres koncowy [endadd], program zatrzyma sie przy wczesniej zdefiniowanym punkcie wstrzyman. Jesli punkty wstrzyman nie zostana zdefiniowane, to zatrzymanie programu mozliwe jest jedynie przez sprzetowe wyzerowanie systemu linia *RESET#*. Bledne podanie adresu poczatkowego powoduje wyswietlenie komunikatu „BREAKPOINT NEAR CURRENT PC IGNORED”. Po osiagnieciu przez program koncowego punktu wstrzymania wyswietlony zostaje komunikat:

PROCESSING TERMINATED AT xxxx
#

Start programu w trybie pracy krokowej (Trace)

T [num] – start programu w trybie pracy krokowej.

Uwaga: Parametr [num] okresla liczbe kroków do wykonania, przy czym moze przyjmowac wartosci od 1 do FFFFH. Brak parametru powoduje wykonanie 1 kroku. Przy pracy krokowej nieaktywne sa zadne punkty wstrzyman, a przerwanie wykonywania kroku mozliwe jest przez uzycie klawiszy *Ctrl-C*. Za pomoca komendy **T** mozliwe jest testowanie programów zawartych w pamieci *EPROM*. Po wykonaniu kazdego

kroku, na ekranie wyswietlane sa aktualne wartosci poszczególnych rejestrów oraz ustawienia znaczników, przy czym zawartosc licznika **PC** oraz zdisasemblowany rozkaz wskazuje nastepny do wykonania rozkaz. Przedstawiono przykladowe wyprowadzenie dla jednego rozkazu:

```
#T <cr>
  RA  RB  R0  R1  R2  R3  R4  R5  R6  R7  PSW     DPTR  SP  PC
  0A  00  2F  00  01  11  2C  12  15  00  ---R0---  0100  05  0244
0244  SETB  2F
#
```

Start programu w trybie „procedura-praca krokowa” (Procedure)

P [num] – start programu w trybie „procedura-praca krokowa”.

Uwaga: Komenda **P** dziala tak samo jak komenda **T** z tym, ze podprogramy traktowane sa jako pojedyncze kroki.

Komendy odczytu i zapisu programów formacie Intel-Standard-Hex

Odczyt programu

: hex_record – odczyt rekordu w formacie *Intel-Standard-Hex*.

Uwaga: komenda sluzy do wprowadzania programu w kodzie maszynowym z poziomu **MONITOR-51**, np.:

```
#:0BBD01200345FF0133253246CFF012 <cr>
#:0000FF <cr>
```

Zapis programu (Save)

S startadd, endadd – przeksztalcenie obszaru pamieci na format *Intel-Standard-Hex* i przetransmitowanie przez zlacze szeregowe.

Uwaga: podany w komendzie zakres pamieci po przeksztalceniu na format *Intel-Standard-Hex* zostanie za pomoca programu komunikacyjnego **MONTERM** przeslany zlaczem szeregowym do komputera *PC*, np.:

```
#S 1000, 1100 <cr>
```

Komendy pomocnicze

HELP <cr> – wywolanie krótkiego opisu wszystkich komend programu **MONITOR-51**.

;komentarz – znaki rozpoczynajace sie srednikiem traktowane sa jako komentarz i sa przez **MONITOR-51** ignorowane (nie sa interpretowane).

3.2.2. Opis programu MONTERM

⇒ Wlasciwosci i wymagania sprzetowo-programowe

MONTERM jest programem komunikacyjnym umozliwiajacym podlaczenie modulu dydaktycznego *MINICON* z programem **MONITOR-51** do komputera *PC*. Umozliwia zapisywanie danych z **MONITOR-51** na komputerze *PC*, a takze odczytywanie danych w formacie *Intel-Standard-Hex* z komputera *PC* i ich transmisje do modulu dydaktycznego *MINICON*.

Dla prawidlowej pracy programu **MONTERM** musza byc spelnione nastepujace warunki: komputer *PC* z systemem *MS-DOS* w wersji 2.11 lub nowszej, 128 kB *RAM*, zlacze szeregowe.

⇒ Praca z programem MONTERM

Program **MONTERM** uruchamia sie przez wywolanie nazwy „**MT**”. Program instaluje na komputerze *PC* terminal, który wspólpracuje ze zlaczem szeregowym poprzez przerwania. Przy wywolaniu programu mozna zmienic sposób jego pracy przez podanie dodatkowych parametrów. Wywolanie programu **MONTERM** wyglada nastepujaco:

MT [par] <cr>

Jako parametr **[par]** mozna podac:

COM1: lub **1**	– do komunikacji wykorzystuje sie lacze szeregowe *COM1*
COM2: lub **2**	– do komunikacji wykorzystuje sie lacze szeregowe *COM2*
COM3: lub **3**	– do komunikacji wykorzystuje sie lacze szeregowe *COM3*
COM4: lub **4**	– do komunikacji wykorzystuje sie lacze szeregowe *COM4*

INT14 lub **I**	– zlacze szeregowe dziala poprzez przerwanie programowe 14H systemu DOS
NOINT lub **N**	– zlacze szeregowe jest sprawdzane metoda „pooling" bez korzystania z systemu przerwac *PC*
BAUDRATE (bps) lub **BR (bps)**	– umozliwia ustawienie predkosci transmisji, przy czym dopuszczalne sa wartosci: 300, 600, 1200, 2400, 4800, 9600, 19200. Brak tego parametru ustala standardowa predkosc transmisji 9600 bps
STARTUP (chars) lub **S (chars)**	– podaje litery, które przy wywolaniu programu **MONITOR** powinny zostac wyslane przez zlacze szeregowe, przy czym maksymalnie mozna podac ciag 80 znaków. Brak tego parametru powoduje wyslanie znaku pustego

W celu ulatwienia uruchamiania programu, **MONTERM** umozliwia korzystanie ze zmiennej systemowej „**MT=**", instalowanej w systemie *MS-DOS* poleceniem *SET*. Program **MONTERM** sygnalizuje wykorzystanie zmiennej systemowej przez podanie komunikatu:

ENVIROMENT STRING: MT= <parametry>

Ponizej podano przykladowa zmienna systemowa MT:

SET MT=COM1: BR (9600) S (U)

Po wlaczeniu zasilania lub sprzetowym wyzerowaniu modulu dydaktycznego linia *RESET* zglasza sie automatycznie program **MONITOR-51**.

⇒ Komendy programu MONTERM

Program **MONTERM** umozliwia wykonanie nastepujacych komend:

F1 lub **Alt-1**	– wyjscie z programu **MONTERM** i powrót do systemu *DOS*
F2 lub **Alt-2**	– transmisja programu z komputera *PC* do **MONITOR-51**. Jezeli istnieje potrzeba przerwania transmisji, to nalezy ponownie wcisnac klawisz **F2** lub **Alt-2**
F3 lub **Alt-3**	– zapisanie danych wyprowadzonych na ekran w zbiorze dyskowym. Ponowne wcisniecie klawisza **F3** lub **Alt-3** powoduje przerwanie transmisji

3.3. Symulator programowy SIM

Symulator programowy **SIM**, nalezacy do pakietu oprogramowania mikrokontrolera **SAB 80C535** jest programem umozliwiajacym symulacje programów napisanych w asemblerze procesorów nalezacych do rodziny **INTEL'51**. Umozliwia on testowanie poprawnosci programów bez koniecznosci uzywania symulatora sprzetowego lub zestawu uruchomieniowego, przy czym testowac mozna programy obliczeniowe, to znaczy takie, w których nie wykorzystuje sie przerwan czasowych wewnetrznych liczników procesora **SAB 80C535** i funkcji wzbudzania wewnetrznego ukladu nadzorujacego watchdog.

Do pracy z symulatorem **SIM** konieczne jest wytworzenie zbiorów typu .sim. Zbiory te tworzone sa programem **LOADER**. Po uruchomieniu programu **LOADER**, nalezy wczytac zbiór roboczy poleceniem

LOAD nazwa_zbioru.obj,

a nastepnie instrukcja

SIMINFO nazwa_zbioru

wytworzyc zbiór typu .sim, przy czym rozszerzenie zbioru zostanie dodane automatycznie.

Wyjscie z programu **LOADER** nastepuje po podaniu komendy *EXIT*. Program **LOADER** posiada pomoc, wywolywana instrukcja *HELP*.

Po uruchomieniu symulatora uzytkownik ma do dyspozycji szereg komend, umozliwiajacych symulacje pracy mikrokontrolera. Ponizej przedstawiono opis poszczególnych komend symulatora, przy czym typy pamieci procesora TYP uzyte w opisie komend oznaczaja:

E – pamiec zewnetrzna danych,

D – pamiec wewnetrzna danych,

P – pamiec zewnetrzna programu,

S – rejestry specjalne **SFR** procesora.

BATCH lub **@** <zbiór>	– wywolanie procedury wsadowej „zbiór”
B[REAK] <addr>	– zakladanie pulapki pod adresem addr
B[REAK] - <addr>	– kasowanie pulapki pod adresem addr
C[ONTINIUE]	– kontynuacja procedury wsadowej przerwanej komenda PAUSE w zbiorze „zbiór”
DIS <addr>	– disasemblacja programu od podanego adresu
DUMP [TYP] [addr]	– wyswietlanie pamieci typu TYP od adresu addr w kodzie ASCII-HEX
LOAD [nazwa_zbioru]	– ladowanie zbioru roboczego typu .sim
L[MEM] <TYP> <addr> <c>	– ladowanie pamieci typu TYP pod adresem addr wartoscia c
M[EM] <TYP> <addr> <c>	– podgladanie i zmiana zawartosci komórki pamieci typu TYP pod adresem addr wartoscia c
PAUSE	– przerwanie procedury wsadowej w zbiorze „zbiór”
RESET	– symulacja goracego restartu procesora
SET <nazwa> <wartosc>	– ustawienie bitu lub bajtu procesora
SIM [N]	– symulacja N rozkazów programu
STATUS	– przedstawienie aktualnego statusu procesora
S[TEP]	– symulacja jednego rozkazu
<**ENTER**>	– powtórzenie ostatniej komendy
EXIT, **QUIT**	– koniec symulacji i wyjscie z programu

W przypadku zawieszenia sie symulowanego programu przerwanie symulacji i wyjscie z programu symulatora nastepuje po wcisnieciu przycisków **Ctrl+X**.