6. PRZYKŁADY ZASTOSOWAŃ MIKROKONTROLERA

6.1. Zastosowanie mikrokontrolera SAB 80C535 do sterowania silnikiem prądu stałego

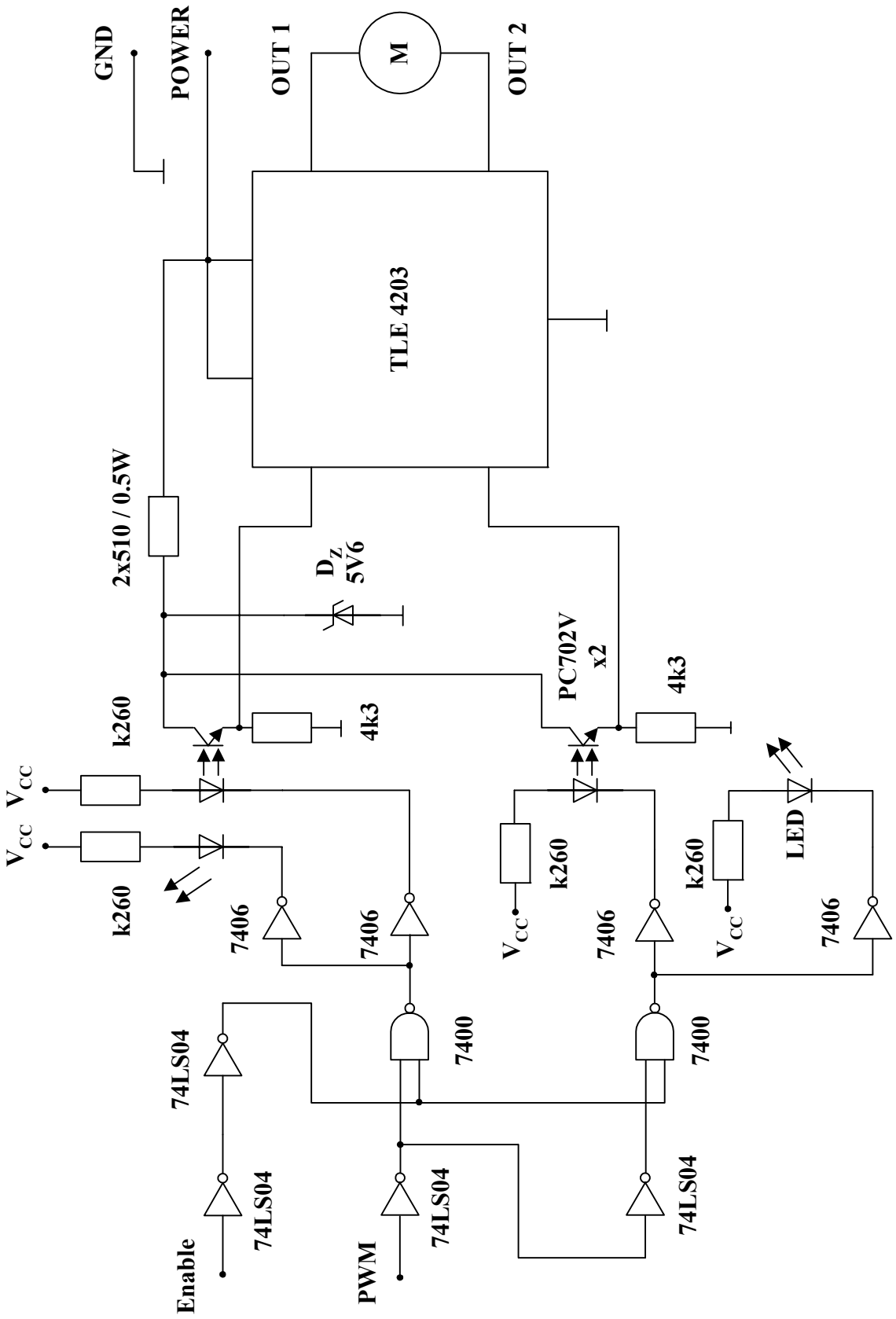
Sterowanie silnikiem prądu stałego oparte jest na wykorzystaniu licznika **T2** do generowania sygnału prostokątnego o regulowanej szerokości *MSI* (PWM). Licznik **T2** umożliwia generowanie czterech niezależnych sygnałów *MSI*, które dostępne są na liniach portu **P1.0**...**P1.3**. Jednakże do sterowania jednego silnika prądu stałego można wykorzystać tylko jeden z nich. Sygnał ten po wzmocnieniu w układzie pośredniczącym (sterowniku) zasila uzwojenia silnika.

Jako sterownik silnika prądu stałego wykorzystano układ scalony **TLE 4203**, umożliwiający regulację prędkości obrotowej silnika w dwóch kierunkach. Do sterowania silnika wykorzystuje się dwa sygnały:

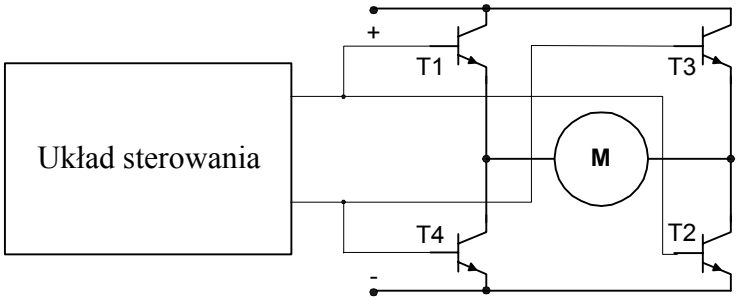
- **ENABLE** – sygnał zezwolenia, bramkujący sygnał **PWM**, przy czym poziomem aktywnym jest jedynka logiczna.
- **PWM** – sygnał o modulowanej szerokości impulsu z jednej linii portu **P1**.

Zasada działania sterownika jest następująca: jeżeli wypełnienie impulsu wynosi 50%, to wartość średnia napięcia na zaciskach silnika równa jest zero i silnik nie pracuje. Jeżeli wypełnienie impulsu jest większe od 50%, to na zacisku dodatnim silnika pojawia się dodatnie napięcie zasilające o pewnej wartość i silnik wiruje w prawo. Im większe jest wypełnienie impulsu, tym większa jest wartość napięcia zasilającego silnik, a więc tym samym większa jest prędkość obrotowa. W przypadku, gdy wypełnienie impulsu jest mniejsze od 50%, wówczas na zacisku dodatnim silnika pojawia się ujemne napięcie zasilające, a tym samym zmienia się kierunek wirowania silnika. Im mniejsze jest wypełnienie impulsu, tym większa jest wartość ujemnego napięcia zasilającego silnik i większa jest prędkość obrotowa silnika.

Schemat ideowy opisanego sterownika silnika prądu stałego przedstawiono na rys 6.1. Na rysunku 6.2 przedstawiono schemat stopni wyjściowych układu **TLE 4203** z podłączonym silnikiem.



Rys. 6.1. Schemat sterownika silnika prądu stałego



Rys. 6.2. Schemat stopni wyjściowych układu TLE 4203

Przedstawiono program sterujący pracą silnika prądu stałego współpracujący z opisanym sterownikiem. Część główna programu obliczająca współczynnik wypełnienia i częstotliwość generowanych impulsów napisana jest w języku *BASIC*. Takie rozwiązanie umożliwia wprowadzanie zadanych wartości częstotliwości i wypełnienia z klawiatury komputera. Podprogram realizujący zmianę współczynnika wypełnienia impulsów przez zmianę zawartości rejestrów komparatora licznika **T2** napisano w asemblerze.

```
5 INPUT "fr [Hz] = ",F : FX=F : F=65535-INT(1000000/F)
8 INPUT "WSP. WYP.[%]=",W
10 W=100-W : W=65535-INT(W*10000/FX)
20 F1=INT(F/256) : F2=INT(F-256*F1) : PRINT F1,F2
30 W1=INT(W/256) : W2=INT(W-256*W1) : PRINT W1,W2
40 XBY(02H)=F2 : XBY(05H)=F1
50 XBY(08H)=W2 : XBY(0BH)=W1
55 CALL 0
60 GOTO 5
```

```
;STALE PROGRAMU

RELOAD_L    EQU  $00          ;Wartość licznika dla f=3,92 kHz
RELOAD_H    EQU  $FF
COMP_2_H    EQU  $7F
COMP_2_l    EQU  $00

      ORG $00
      MOV    CRCL,#RELOAD_L   ;Wpis młodszej wartości do
                              ;rejestru CRC
      MOV    CRCH,#RELOAD_H   ;Wpis starszej wartości do
```

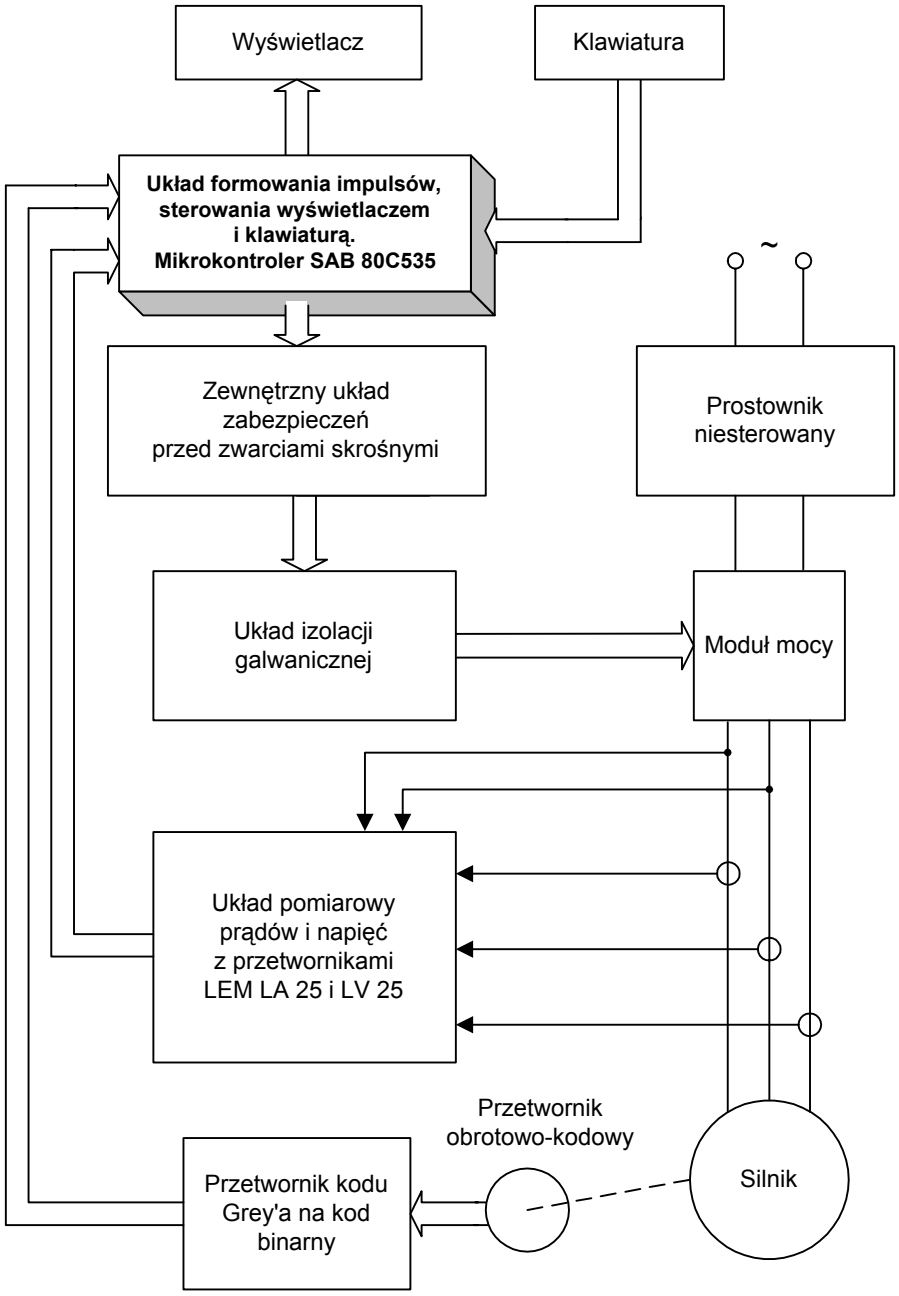
```
                                    ;rejestru CRC
MOV     CCL2,#COMP_2_L      ;Wpis młodszej wartości do
                            ;rejestru komparatora
MOV     CCH2,#COMP_H        ;Wpis starszej wartości do
                            ;rejestru komparatora
MOV     CCEN,#$A8           ;Wybór komparatora
MOV     T2CON,#$11          ;Wybór trybu pracy licznika,
                            ;autoładowanie po przepełnieniu
                            ;licznika T2,taktowanie
                            ;sygnałem wewnętrznym
        RET
```

6.2. Zastosowanie mikrokontrolera SAB 80C535 do sterowania silnikiem indukcyjnym klatkowym

Do sterowania silnikiem indukcyjnym klatkowym powszechnie stosowane są układy przekształtnikowe, którymi najczęściej są falowniki napięcia *MSI*. Ze względu na realizowanie skomplikowanych algorytmów sterowania urządzenia te wyposażone są w cyfrowe układy sterujące budowane przy użyciu mikroprocesorów.

Schemat układu sterowania silnikiem indukcyjnym klatkowym przedstawiono na rys.6.3. Regulacja prędkości silnika indukcyjnego odbywa się w układzie otwartym, tzn. bez kontroli prądu i momentu silnika. W sterowniku tym zastosowano modulację szerokości impulsów z wykorzystaniem wektora przestrzennego napięcia.

Działanie układu jest następujące: przebiegi *MSI* generowane są w mikrokontrolerze **SAB 80C535** przez wewnętrzne układy licznika **T2**. W trakcie realizacji programu formowania impulsów pobierane są z tablicy wartości liczbowe i umieszczane w trzech rejestrach komparatorów. W rezultacie na wyjściach portu **P1.1**, **P1.2** i **P1.3** pojawiają się przebiegi zmodulowane przesunięte względem siebie o 120° o zadanej częstotliwości i wypełnieniu. Układ sterowania wyposażony jest w zabezpieczenie przed zwarciami skrośnymi, oraz układy pomiarowe napięć, prądów i prędkości obrotowej silnika. Wartości mierzonych parametrów wyświetlane są na wyświetlaczu. Sygnały prądowe służą również do zabezpieczenia układu przed przeciążeniem. Za pomocą klawiatury można wprowadzić wartości graniczne prądu, dokonać rozruchu lub zatrzymania silnika. Program sterujący umieszczono w pamięci stałej *EPROM*.



Rys. 6.3. Schemat układu sterowania silnika indukcyjnego klatkowego

Przedstawiono fragment programu z procedurą generowania trzech przesuniętych w fazie o 120° przebiegów sinusoidalnych potrzebnych do sterowania kluczami falownika.

****

Deklaracja stałych i zmiennych programu

```
**********************************
FAZA_A    BYTE     ($50)  ;Adres wpisu pozycji próbki fazy A
FAZA_B    BYTE     ($51)  ;Adres wpisu pozycji próbki fazy B
FAZA_C    BYTE     ($52)  ;Adres wpisu pozycji próbki fazy C
LICZNIK   BYTE     ($20)  ;Adres wpisu znacznika faz
ZNAK_A    BIT      (LICZNIK.0)
ZNAK_B    BIT      (LICZNIK.1)
ZNAK_C    BIT      (LICZNIK.2)
ADDR_TB   EQU      $0200

**********************************
        ORG  $100

        MOV  FAZA_A,#$40      ;Wpis pozycji próbki w tablicy dla fazy A
        MOV  FAZA_B,#$20      ;Wpis pozycji próbki w tablicy dla fazy B
        MOV  FAZA_C,#$5F      ;Wpis pozycji próbki w tablicy dla fazy C
        MOV  DPTR,ADDR_TB     ;Wpis do rejestru wskaźnikowego adresu tablicy
                              ;z próbkami
        CLR  ZNAK_A           ;Zerowanie znacznika fazy A
        SETB ZNAK_B           ;Ustawienie znacznika fazy B
        SETB ZNAK_C           ;Ustawienie znacznika fazy C

        MOV  CRCL,#$00        ;Wpis wartości młodszego bajtu częstotliwości
        MOV  CRCH,#$0FF       ;Wpis wartości starszego bajtu częstotliwości
        MOV  CCH1,#$0FF       ;Wpis wartości starszego bajtu fazy A
        MOV  CCH2,#$0FF       ;Wpis wartości starszego bajtu fazy B
        MOV  CCH3,#$0FF       ;Wpis wartości starszego bajtu fazy C
        MOV  CCEN,#$0A8       ;Konfiguracja komparatorów licznika T2,
                              ;odblokowanie komparatorów CC1,CC2,CC3
        MOV  T2CON,#$11       ;Ustawienie trybu pracy licznika T2, start licznika

STAR_1: MOV CCL1,R2           ;Wpis wartości początkowych młodszych bajtów
          MOV CCL2,R3         ;komparatorów CC1,CC2,CC3
          MOV CCL3,R4
          JB  ZNAK_A,MIN      ;Sprawdzanie znacznika fazy A, jeśli 1 to faza A
PLUS: INC  FAZA_A             ;wypełniana jest próbkami malejącymi, jeśli 0 to
                              ;faza A wypełniana jest próbkami narastającymi
        MOV  A,FAZA_A
        MOVC A,@A+DPTR
        MOV  R2,A
        SJMP STAR_2
MIN:  DEC  FAZA_A
        MOV  A,FAZA_A
        MOVC A,@A+DPTR
        MOV  R2,A
```

```
STAR_2: JB ZNAK_B,MIN_1   ;Sprawdzanie znacznika fazy B, jeśli 1, to faza B
                          ;wypełniana jest próbkami malejącymi, jeśli 0, to
                          ;faza B wypełniana jest próbkami narastającymi
PLUS_1: INC FAZA_B
        MOV A,FAZA_B
        MOV A,@A+DPTR
        MOV R3,A
        SJMP STAR_3
MIN_1: DEC FAZA_B
        MOV A,FAZA_B
        MOV A,@A+DPTR
        MOV R3,A

STAR_3: JB ZNAK_C,MIN_2   ;Sprawdzanie znacznika fazy C, jeśli 1, to faza C
                          ;jest wypełniana próbkami malejącymi, jeśli 0, to
                          ;faza C jest wypełniana próbkami narastającymi
PLUS_2: INC FAZA_C
          MOV A,FAZA_C
          MOVC A,@A+DPTR
          MOV R4,A
          SJMP SPR
MIN_2:  DEC FAZA_C
          MOV A,FAZA_C
          MOVC A,@A+DPTR
          MOV R4,A

SPR:  MOV  A,FAZA_A
        CJNE A,#$80,SPR_1   ;Sprawdzanie pozycji próbki fazy A w tablicy w celu
        SETB ZNAK_A         ;ustawienia znacznika fazy A
SPR_1: CJNE A,#$00,SPR_2    ;Sprawdzanie pozycji próbki fazy A w tablicy w celu
         CLR ZNAK_A         ;wyzerowania znacznika fazy A
SPR_2: MOV A,FAZA_B
        CJNE A,#$80,SPR_3   ;Sprawdzanie pozycji próbki fazy B w tablicy
        SETB ZNAK_B         ;w celu ustawienia znacznika fazy B
SPR_3: CJNE A,#$00,SPR_4    ;Sprawdzanie pozycji próbki fazy B w tablicy
         CLR ZNAK_B         ;w celu wyzerowania znacznika fazy B
SPR_4: MOV A,FAZA_C
         CJNE A,#$80,SPR_5  ;Sprawdzanie pozycji próbki fazy C w tablicy
         SETB ZNAK_C        ;w celu ustawienia znacznika fazy C
SPR_5: CJNE A,#$00,WYJ      ;Sprawdzenie pozycji próbki fazy C w tablicy
         CLR ZNAK_C         ;w celu  wyzerowania znacznika fazy C
WYJ:    LJMP STAR_1
        ORG  $0200

        db      5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26
```

```
db      27,28,29,30,31,32,33,34,36,38,40,42,44,46,48,50,52,54,56,58
db      60,62,64,66,68,71,74,77,80,83,86,89,92,96,100,104,108,112
db      116,120,124,128,132,136,140,144,148,152,156,160,164,168
db      171,174,177,180,183,186,188,190,192,194,196,198,200,202
db      204,206,208,210,212,214,216,218,219,220,221,222,223,224
db      225,226,227,228,229,230,231,232,233,234,235,236,237,238
db      239,240,241,242,243,244,245,246,247,248,249,250,250,250
```

ZAŁĄCZNIKI

ZAŁĄCZNIK 1

LISTA ROZKAZÓW MIKROKONTROLERA

Lista rozkazów mikrokontrolera SAB 80C515/535 zawiera 111 rozkazów, wśród których jest 49 jednobajtowych, 45 dwubajtowych i 17 trzybajtowych. Wykonywane są one w ciągu jednego lub dwóch cykli maszynowych. Wyjątkiem są rozkazy mnożenia i dzielenia, które wykonywane są w ciągu czterech cykli maszynowych. W opisie rozkazów przyjęto następujące oznaczenia:

A	– akumulator,
B	– rejestr B,
Rr	– rejestr roboczy (R0...R7),
Ri	– rejestr roboczy - wskaźnik danych: i=0,1. (R0 i R1),
DPTR	– wskaźnik danych,
PC	– licznik rozkazów,
SP	– wskaźnik stosu,
C (CY)	– wskaźnik przeniesienia,
AC	– znacznik (flaga) przeniesienia pomocniczego,
OV	– znacznik (flaga) przepełnienia,
ad	– 8–bitowy adres bezpośredni,
n	– 8–bitowy argument bezpośredni,
nn	– 16–bitowy argument bezpośredni,
bit	– 8–bitowy adres bitu w pamięci RAM lub SFR,
adr11	– adres 11–bitowy,
adr16	– adres 16–bitowy,
d	– 8–bitowe przesunięcie o wartości z przedziału <–128, 127> (w praktyce d oznacza etykietę lub adres),
@	– w mnemoniku rozkazu poprzedza adres pośredni,
#	– w mnemoniku rozkazu poprzedza argument bezpośredni,
x	– w zapisie operacji oznacza zawartość rejestru x,
(x)	– w zapisie operacji oznacza zawartość pamięci o adresie x,
<r>	– miejsce pobrania argumentu i zapisania wyniku operacji,
<s>	– miejsce pobrania drugiego argumentu operacji,
 	– rejestr bazowy,
<dp>	– wskaźnik danych,

LISTA ROZKAZÓW POGRUPOWANYCH FUNKCJONALNIE

Rozkazy przesyłania danych

Kod rozkazu (hex)	Mnemonik	Operacja	Bajty	Cykle
E8-EF	MOV A,Rr	A ← Rr	1	1
E5	MOV A,ad	A ← (ad)	2	1
E6-E7	MOV A,@Ri	A ← (Ri)	1	1
74	MOV A,#n	A ← n	2	1
F8-FF	MOV Rr,A	Rr ← A	1	1
A8-AF	MOV Rr,ad	Rr ← (ad)	2	2
78-7F	MOV Rr,#n	Rr ← n	2	1
F5	MOV ad,A	(ad) ← A	2	1
88-8F	MOV ad,Rr	(ad) ← Rr	2	2
85	MOV ad1,ad2	(ad1) ← (ad2)	3	2
86-87	MOV ad,@Ri	(ad) ← (Ri)	2	2
75	MOV ad,#n	(ad) ← n	3	2
F6-F7	MOV @Ri,A	(Ri) ← A	1	1
A6-A7	MOV @Ri,ad	(Ri) ← (ad)	2	2
76-77	MOV @Ri,#n	(Ri) ← n	2	1
90	MOV DPTR,#nn	DPTR ← nn	3	2
93	MOVC A,@A + DPTR	A ← (A+DPTR)	1	2
83	MOVC A,@A + PC	A ← (A+PC)	1	2
E2-E3	MOVX A,@Ri	A ← (Ri)	1	2
F2-F3	MOVX @Ri,A	(Ri) ← A	1	2
E0	MOVX A,@DPTR	A ← (DPTR)	1	2
F0	MOVX @DPTR,A	(DPTR) ← A	1	2
C8-CF	XCH A,Rr	A ↔ Rr	1	1
C5	XCH A,ad	A ↔ (ad)	2	1
C6-C7	XCH A,@Ri	A ↔ (Ri)	1	1
D6-D7	XCHD A,@Ri	$A_{3..0} \leftrightarrow (Ri)_{3..0}$	1	1

Rozkazy operacji arytmetyczno–logicznych

Kod rozkazu (hex)	Mnemonik	Operacja	Bajty	Cykle
28-2F	ADD A,Rr	A ← A + Rr	1	1
25	ADD A,ad	A ← A + (ad)	2	1

26-27	ADD A,@Ri	A ← A + (Ri)	1	1
24	ADD A,#n	A ← A + n	2	1
38-3F	ADDC A,Rr	A ← A + Rr + C	1	1
35	ADDC A,ad	A ← A + (ad) + C	2	1
36-37	ADDC A,@Ri	A ← A + (Ri) + C	1	1
34	ADDC A,#n	A ← A + n + C	2	1
98-9F	SUBB A,Rr	A ← A – Rr – C	1	1
95	SUBB A,ad	A ← A – (ad) – C	2	1
96-97	SUBB A,@Ri	A ← A – (Ri) – C	1	1
94	SUBB A,#n	A ← A – n – C	2	1
04	INC A	A ← A + 1	1	1
08-0F	INC Rr	Rr ← Rr + 1	1	1
05	INC ad	(ad) ← (ad) + 1	2	1
06-07	INC @Ri	(Ri) ← (Ri) + 1	1	1
A3	INC DPTR	DPTR ← DPTR + 1	1	2
14	DEC A	A ← A – 1	1	1
18-1F	DEC Rr	Rr ← Rr – 1	1	1
15	DEC ad	(ad) ← (ad) – 1	2	1
16-17	DEC @Ri	(Ri) ← (Ri) – 1	1	1
A4	MUL AB	B.A ← A x B	1	4
84	DIV AB	A ← A div B	1	4
D4	DA A	Korekcja dziesiętna A (nie dotyczy odejmowania)	1	1
58-5F	ANL A,Rr	A ← A ∧ Rr	1	1
55	ANL A,ad	A ← A ∧ (ad)	2	1
56-57	ANL A,@Ri	A ← A ∧ (Ri)	1	1
54	ANL A,#n	A ← A ∧ n	2	1
52	ANL ad,A	(ad) ← (ad) ∧ A	2	1
53	ANL ad,#n	(ad) ← (ad) ∧ n	3	2
48-4F	ORL A,Rr	A ← A ∨ Rr	1	1
45	ORL A,ad	A ←A ∨ (ad)	2	1
46-47	ORL A,@Ri	A ← A ∨ (Ri)	1	1
44	ORL A,#n	A ← A ∨ n	2	1
42	ORL ad,A	(ad) ← (ad) ∧ A	2	1
43	ORL ad,#n	(ad) ← (ad) ∨ n	3	2
68-6F	XRL A,Rr	A ← A ⊕ Rr	1	1
65	XRL A,ad	A ← A ⊕ (ad)	2	1
66-67	XRL A,@Ri	A ← A ⊕ (Ri)	1	1
64	XRL A,#n	A ← A ⊕ n	2	1
62	XRL ad,A	(ad) ← (ad) ⊕ A	2	1
63	XRL ad,#n	(ad) ← (ad) ⊕ n	3	2

E4	CLR A	$A \leftarrow 0$	1	1
F4	CPL A	$A \leftarrow \overline{A}$	1	1
C4	SWAP A	$A_{3...0} \leftrightarrow A_{7...4}$	1	1
23	RL A	Przesuń w lewo	1	1
33	RLC A	Przesuń w lewo z przeniesieniem	1	1
03	RR A	Przesuń w prawo	1	1
13	RRC A	Przesuń w prawo z przeniesieniem	1	1

Rozkazy operacji na bitach

Kod rozkazu (hex)	Mnemonik	Operacja	Bajty	Cykle
C3	CLR C	$CY \leftarrow 0$	1	1
C2	CLR bit	$(bit) \leftarrow 0$	2	1
D3	SETB C	$CY \leftarrow 1$	1	1
D2	SETB bit	$(bit) \leftarrow 1$	2	1
B3	CPL C	$CY \leftarrow \overline{C}$	1	1
B2	CPL bit	$(bit) \leftarrow (\overline{bit})$	2	1
82	ANL C,bit	$CY \leftarrow CY \wedge (bit)$	2	2
B0	ANL C,/bit	$CY \leftarrow CY \wedge (\overline{bit})$	2	2
72	ORL C,bit	$CY \leftarrow CY \vee (bit)$	2	2
A0	ORL C,/bit	$CY \leftarrow CY \vee (\overline{bit})$	2	2
A2	MOV C,bit	$CY \leftarrow (bit)$	2	1
92	MOV bit,C	$(bit) \leftarrow CY$	2	2

Rozkazy skoków i rozkazy sterujące

Kod rozkazu (hex)	Mnemonik	Operacja	Bajty	Cykle
01,21,41,61,81,A1,C1,E1	AJMP adr11	Skocz bezwarunkowo na stronie	2	2
02	LJMP adr16	Skocz bezwarunkowo	3	2
80	SJMP d	Skocz względem PC	2	2
73	JMP @A + DPTR	Skocz pośrednio	1	2
40	JC d	Skocz, jeśli CY=1	2	2
50	JNC d	Skocz, jeśli CY=0	2	2
60	JZ d	Skocz, jeśli A=0	2	2

70	JNZ d	Skocz, jeśli A≠0	2	2
20	JB bit,d	Skocz, jeśli bit=1	3	2
30	JNB bit,d	Skocz, jeśli bit=0	3	2
10	JBC bit,d	Jeśli bit=1, to zeruj go i skocz	3	2
B5	CJNE A,ad,d	Skocz, jeśli A≠(ad)	3	2
B4	CJNE A,#n,d	Skocz, jeśli A≠n	3	2
B8-BF	CJNE Rr,#n,d	Skocz, jeśli Rr≠n	3	2
B6-B7	CJNE @Ri,#n,d	Skocz, jeśli (Ri)≠n	3	2
D8-DF	DJNZ Rr,d	Zmniejsz o 1 i skocz, jeśli Rr≠0	2	2
D5	DJNZ ad,d	Zmniejsz o 1 i skocz, jeśli (ad)≠0	3	2
00	NOP	Nie rób nic	1	1

Rozkazy obsługi podprogramów, przerwań i operacji na stosie

Kod rozkazu (hex)	Mnemonik	Operacja	Bajty	Cykle
11,31,51,71,91,B1,D1,F1	ACALL adr11	Skocz do podprogramu na stronie	2	2
12	LCALL adr16	Skocz do podprogramu	3	2
22	RET	Powróć z podprogramu	1	2
32	RETI	Powróć z przerwania	1	2
C0	PUSH ad	Zdejmij ze stosu	2	2
D0	POP ad	Wyślij na stos	2	2

DOKŁADNY OPIS ROZKAZÓW

Rozkazy przesyłania danych

MOV bit,C Prześlij flagę CY do bitu	Zawartość znacznika przeniesienia przesyłana jest do bitu o podanym adresie bezpośrednim. (bit) ← CY
MOV C,bit Prześlij wartość bitu do flagi przeniesienia	Zawartość bitu o podanym adresie bezpośrednim przesyłana jest do znacznika przeniesienia. Cy ← (bit)
MOV DPTR,#nn	Do 16–bitowego rejestru DPTR wpisywany jest argu-

Wpisz 16-bitową wartość do wskaźnika danych	ment bezpośredni, podany jako drugi i trzeci bajt rozkazu. Pierwszy bajt argumentu wpisywany jest do rejestru DPH, a drugi do rejestru DPL. DPTR ← nn
MOV <r>,<s> Prześlij dane	Ośmiobitowy argument <s> jest przesyłany do miejsca wskazanego przez argument <r>. Możliwe jest piętnaście kombinacji adresowania argumentów: MOV A,Rr A ← Rr MOV A,@Ri A ← (Ri) MOV A,ad A ← (ad) MOV A,#n A ← n MOV Rr,A Rr ← A MOV Rr,ad Rr ← (ad) MOV Rr,#n Rr ← n MOV @Ri,A (Ri) ← A MOV @Ri,ad (Ri) ← (ad) MOV @Ri,#n (Ri) ← n MOV ad,A (ad) ← A MOV ad,Rr (ad) ← Rr MOV ad,@Ri (ad) ← (Ri) MOV ad1,ad2 (ad1) ← (ad2) MOV ad,#n (ad) ← n
MOVC A, @A + Prześlij bajt z pamięci programu do akumulatora	Do akumulatora przesyłana jest zawartość komórki pamięci programu o adresie będącym sumą zawartości akumulatora i 16-bitowego rejestru bazowego . Jako rejestry bazowe mogą być użyte wskaźnik danych DPTR i licznik rozkazów PC. Jeżeli rejestrem bazowym jest PC, to adresem bazowym jest adres pierwszego bajtu rozkazu następnego po MOVC. MOVC A,@A + DPTR A ← (A + DPTR) MOVC A,@A + PC A ← (A + PC)
MOVX A,@<dp> Prześlij bajt z zewnętrznej pamięci danych do akumulatora	Do akumulatora wpisywane są dane z komórki zewnętrznej pamięci danych o adresie pośrednim zawartym we wskaźniku danych <dp>. Wskaźnikiem danych może być rejestr roboczy R0 lub R1 albo rejestr DPTR.

	Jeżeli wskaźnikiem danych jest rejestr R0 lub R1, to adres 8-bitowy wysłany jest tylko przez port P0, natomiast jeżeli wskaźnikiem danych jest DPTR, to 16-bitowy adres wysyłany jest przez porty P0 i P2 . MOVX A,@DPTR A ← (DPTR) MOVX A,@Ri A ← (Ri)
MOVX @<dp>,A Prześlij bajt z akumulatora do zewnętrznej pamięci danych	Do komórki zewnętrznej pamięci danych o adresie pośrednim zawartym we wskaźniku danych <dp> wpisywane są dane z akumulatora. Wskaźnikiem danych może być rejestr roboczy R0 lub R1 albo rejestr DPTR. Jeżeli wskaźnikiem danych jest rejestr R0 lub R1, to adres 8–bitowy wysłany jest tylko przez port P0, natomiast jeżeli wskaźnikiem danych jest DPTR, to 16–bitowy adres wysyłany jest przez porty P0 i P2. MOVX @DPTR,A (DPTR) ← A MOVX @Ri,A (Ri) ← A
XCH A,<s> Wymień akumulator z zawartością komórki pamięci	Zawartość akumulatora wymieniana jest z zawartością wskazanego argumentu. Możliwe są trzy tryby adresowania argumentu: XCH A,Rr A ↔ Rr XCH A,@Ri A ↔ (Ri) XCH A,ad A ↔ (ad)
XCHD A,@Ri Wymień półbajty z akumulatora i pamięci	Zawartość mniej znaczących bitów akumulatora (bity 3...0) zostaje wymieniona z zawartością mniej znaczących bitów komórki wewnętrznej pamięci danych o adresie zawartym w rejestrze Ri (R1 lub R0). Bardziej znaczące bity akumulatora (bity 7...4) i komórki pamięci pozostają bez zmian. $A_{3...0} \leftrightarrow (Ri)_{3...0}$

Rozkazy operacji arytmetyczno–logicznych

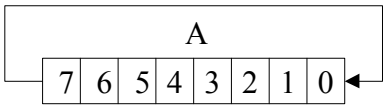
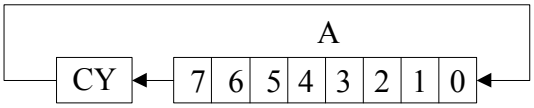
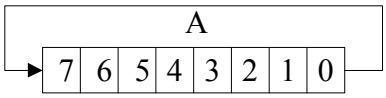
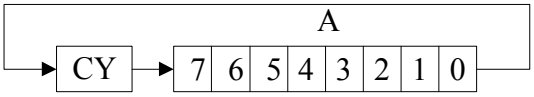
ADD A,<s>	Wskazany argument dodawany jest do akumulatora, do

Dodaj do akumulatora	którego wpisywany jest wynik. Znaczniki CY, AC i OV ustawiane są zgodnie z wynikiem operacji. Możliwe są cztery tryby adresowania argumentu <s>: ADD A,Rr A ← A + Rr ADD A,@Ri A ← A + (Ri) ADD A,ad A ← A + (ad) ADD A,#n A ← A + n
ADDC A,<s> Dodaj do akumulatora z przeniesieniem	Do akumulatora dodawany jest wskazany argument i zawartość znacznika przeniesienia, przy czym wynik wpisywany jest do akumulatora. Znaczniki CY, AC i OV ustawiane są zgodnie z wynikiem operacji. Możliwe są cztery tryby adresowania argumentu <s>: ADDC A,Rr A ← A + Rr + CY ADDC A,@Ri A← A + (Ri) + CY ADDC A,ad A← A + (ad) + CY ADDC A,#n A← A + n + CY
SUBB A,<s> Odejmij od akumulatora z pożyczką	Zawartość wskazanego argumentu oraz zawartość znacznika przeniesienia CY odejmowana jest od akumulatora, przy czym wynik operacji wpisywany jest do akumulatora i ustawiane są znaczniki CY, AC i OV. Możliwe są cztery tryby adresowania argumentu: SUBB A,Rr A ← A – Rr – CY SUBB A,@Ri A ← A – (Ri) – CY SUBB A,ad A ← A – (ad) – CY SUBB A,#n A ← A – n – CY
INC <r> Zwiększ o 1	Do wskazanego argumentu dodawana jest jedynka, przy czym stan znaczników nie ulega zmianie. Jeżeli rozkaz użyty jest do zmiany stanu wyjścia, to jest odczytywana i modyfikowana zawartość rejestru wyjściowego portu, a nie stan logiczny końcówek układu. Możliwe są cztery tryby adresowania argumentu: INC A A ← A + 1 INC Rr Rr ← Rr + 1 INC @Ri (Ri) ← (Ri) + 1 INC ad (ad)← (ad) + 1

INC DPTR Zwiększ o 1 wskaźnik danych	Do 16-bitowego rejestru DPTR dodawana jest 1, przy czym stan znaczników nie ulega zmianie. DPTR ← DPTR + 1
DEC <r> Zmniejsz o 1	Od wskazanego argumentu odejmowana jest jedynka, przy czym stan znaczników nie ulega zmianie. Jeżeli rozkaz użyty jest do zmiany stanu wyjścia, to jest odczytywana i modyfikowana zawartość rejestru wyjściowego portu, a nie stan logiczny końcówek układu. Możliwe są cztery tryby adresowania argumentu: DEC A A ← A – 1 DEC Rr Rr ← Rr – 1 DEC @Ri (Ri) ← (Ri) – 1 DEC ad (ad) ← (ad) – 1
MUL AB Pomnóż	Ośmiobitowa liczba dwójkowa bez znaku z akumulatora mnożona jest przez 8–bitową liczbę bez znaku z rejestru B. Osiem bardziej znaczących bitów 16–bitowego wyniku wpisywanych jest do rejestru B, natomiast 8 mniej znaczących do akumulatora. Jeśli wynik mnożenia jest większy od 255, to jest ustawiany znacznik OV, w przeciwnym razie znacznik ten jest zerowany. Znacznik CY jest zerowany. B.A ← [A x B]
DIV AB Podziel	Operacja dokonuje dzielenia ośmiobitowej liczby bez znaku zawartej w akumulatorze przez ośmiobitową liczbę bez znaku zawartą w rejestrze B. Część całkowita wyniku dzielenia wpisywana jest do akumulatora, reszta zaś do rejestru B. Znaczniki CY i OV są zerowane. Jeśli w rejestrze B jest 0, to po wykonaniu rozkazu zawartość akumulatora i rejestru B jest nieokreślona, a znacznik OV = 1. A ← [A/B] B ← reszta (A/B)

DA A Korekcja dziesiętna	Wykonywana jest korekcja dziesiętna wyniku dodawania (nie jest dokonywana korekcja wyniku odejmowania!). Jeżeli argumenty podane są w formacie BCD, to wynik przedstawiany jest w postaci dwóch cyfr dziesiętnych w tym kodzie. Jeśli liczba dziesiętna jest większa od 99, ustawiany jest znacznik przeniesienia CY. Nie zmienia się stan znaczników AC i OV.
ANL <r>,<s> Wymnóż logicznie	Wykonywane jest mnożenie logiczne wskazanych argumentów, przy czym wynik operacji wpisywany jest do miejsca, z którego został pobrany argument <r>. Należy pamiętać, że jeżeli rozkaz jest użyty do zmiany stanu wyjścia, to jest odczytywana i zapisywana zawartość rejestru wyjściowego portu, a nie stan logiczny z końcówek układu. Możliwych jest sześć różnych trybów adresowania argumentów: ANL A,Rr $A \leftarrow A \wedge Rr$ ANL A,@Ri $A \leftarrow A \wedge (Ri)$ ANL A,ad $A \leftarrow A \wedge (ad)$ ANL A,#n $A \leftarrow A \wedge n$ ANL ad,A $ad \leftarrow (ad) \wedge A$ ANL ad,#n $ad \leftarrow (ad) \wedge n$
ANL C,bit Wymnóż logicznie przez bit	Znacznik przeniesienia C jest zerowany, jeżeli wartość logiczna bitu o podanym adresie jest równa 0, w przeciwnym wypadku nie zmienia się. $CY \leftarrow CY \wedge (bit)$
ANL C,/bit Wymnóż logicznie przez negację zawartości bitu	Znacznik przeniesienia C jest zerowany, jeżeli wartość logiczna bitu o podanym adresie jest równa 1, w przeciwnym wypadku nie zmienia się. $CY \leftarrow CY \wedge (\overline{bit})$
ORL <r>,<s> Sumuj logicznie	Wykonywana jest suma logiczna wskazanych argumentów, przy czym wynik wpisywany jest do miejsca, z którego pobrany został argument <r>. Jeżeli operacja

	została użyta do zmiany stanu wyjścia, to zmianie ulega zawartość rejestru wyjściowego portu a nie stan logiczny końcówek układu. Możliwych jest sześć kombinacji adresowania argumentów: ORL A,Rr $A \leftarrow A \vee Rr$ ORL A,@Ri $A \leftarrow A \vee Ri$ ORL A,ad $A \leftarrow A \vee (ad)$ ORL A,#n $A \leftarrow A \vee n$ ORL ad,A $(ad) \leftarrow (ad) \vee A$ ORL ad,#n $(ad) \leftarrow (ad) \vee n$
ORL CY,bit Sumuj logicznie CY z bitem	Wykonana zostaje operacja sumowania logicznego flagi przeniesienia CY z zawartością bitu o podanym adresie bezpośrednim: jeśli wartość logiczna bitu równa jest 1, to flaga CY przyjmuje wartość 1. W przeciwnym razie zawartość CY nie zmienia się. $CY \leftarrow CY \vee (bit)$
ORL C,/bit Sumuj logicznie CY z negacją bitu	Wykonana zostaje operacja sumowania logicznego flagi przeniesienia CY z negacją zawartości bitu o podanym adresie bezpośrednim: jeśli wartość logiczna bitu równa jest 0, to flaga CY przyjmuje wartość 1. W przeciwnym razie zawartość CY nie zmienia się. $CY \leftarrow CY \vee (\overline{bit})$
XRL <r>,<s> Sumuj mod 2	Wykonywana jest suma mod 2 wskazanych argumentów, przy czym wynik wpisywany jest do miejsca, z którego pobrany został argument <r>. Możliwych jest sześć kombinacji trybów adresowania argumentów: XRL A,Rr $A \leftarrow A \oplus Rr$ XRL A,@Ri $A \leftarrow A \oplus (Ri)$ XRL A,ad $A \leftarrow A \oplus (ad)$ XRL A,#n $A \leftarrow A \oplus n$ XRL ad,A $(ad) \leftarrow (ad) \oplus A$ XRL ad,#n $(ad) \leftarrow (ad) \oplus n$
CLR A	Zawartość akumulatora zostaje wyzerowana.

Wyzeruj akumulator	$A \leftarrow 0$
CPL A Negacja akumulatora	Zawartość akumulatora jest negowana bit po bicie. $A \leftarrow \overline{A}$
SWAP A Wymień półbajty w akumulatorze	Zawartość bardziej znaczących bajtów (MSB) wymieniona zostaje z zawartością mniej znaczących bajtów, co jest równoważne z czterokrotnym cyklicznym przesunięciem akumulatora. $A_{7...4} \leftrightarrow A_{3...0}$
RL A Przesuń akumulator cyklicznie w lewo	Zawartość akumulatora przesuwana jest cyklicznie w lewo. 
RLC A Przesuń akumulator cyklicznie w lewo z CY	Zawartość akumulatora przesuwana jest cyklicznie w lewo z uwzględnieniem flagi CY. Wykonanie tego rozkazu przy zerowej wartości flagi CY odpowiada pomnożeniu przez 2 liczby zawartej w akumulatorze. 
RR A Przesuń akumulator cyklicznie w prawo	Zawartość akumulatora przesuwana jest cyklicznie w prawo. 
RRC A Przesuń akumulator cyklicznie w prawo z CY	Zawartość akumulatora przesuwana jest cyklicznie w prawo z uwzględnieniem flagi CY. Wykonanie tego rozkazu przy zerowej wartości flagi CY odpowiada podzieleniu przez 2 liczby zawartej w akumulatorze. 

Rozkazy operacji na bitach

CLR bit Wyzeruj bit	Zerowany jest bit o podanym adresie bezpośrednim. (bit) ← 0
CLR C Wyzeruj znacznik (flagę) przeniesienia	Zerowany jest znacznik (flaga) przeniesienia. C ← 0
CPL bit Negacja bitu	Negowany jest bit o podanym adresie bezpośrednim. (bit) ← ($\overline{\text{bit}}$)
CPL C Negacja CY	Negowany jest znacznik (flaga) przeniesienia. CY ← $\overline{\text{CY}}$
ANL C,bit Pomnóż logicznie przez bit.	Znacznik przeniesienia C jest zerowany, jeżeli wartość logiczna bitu o podanym adresie jest równa 0, w przeciwnym wypadku nie zmienia się. CY ← CY ∧ (bit)
ANL C,/bit Pomnóż logicznie przez negację zawartości bitu	Znacznik przeniesienia C jest zerowany, jeżeli wartość logiczna bitu o podanym adresie jest równa 1, w przeciwnym wypadku nie zmienia się. CY ← CY∧ ($\overline{\text{bit}}$)
ORL <r>,<s> Sumuj logicznie	Wykonywana jest suma logiczna wskazanych argumentów, przy czym wynik wpisywany jest do miejsca, z którego pobrany został argument <r>. Jeżeli operacja została użyta do zmiany stanu wyjścia, to zmianie ulega zawartość rejestru wyjściowego portu, a nie stan logiczny końcówek układu. Możliwych jest sześć kombinacji adresowania argumentów: ORL A,Rr A ← A ∨ Rr ORL A,@Ri A ← A ∨ Ri ORL A,ad A ← A ∨ (ad) ORL A,#n A ← A ∨ n ORL ad,A (ad) ← (ad) ∨ A ORL ad,#n (ad) ← (ad) ∨ n

ORL CY,bit Sumuj logicznie CY z bitem	Wykonana zostaje operacja sumowania logicznego flagi przeniesienia CY z zawartością bitu o podanym adresie bezpośrednim: jeśli wartość logiczna bitu równa jest 1, to flaga CY przyjmuje wartość 1. W przeciwnym razie zawartość CY nie zmienia się. $CY \leftarrow CY \vee (bit)$
ORL C,/bit Sumuj logicznie CY z negacją bitu	Wykonana zostaje operacja sumowania logicznego flagi przeniesienia CY z negacją zawartości bitu o podanym adresie bezpośrednim: jeśli wartość logiczna bitu równa jest 0, to flaga CY przyjmuje wartość 1. W przeciwnym razie zawartość CY nie zmienia się. $CY \leftarrow CY \vee (\overline{bit})$
MOV bit,C Prześlij flagę CY do bitu	Zawartość flagi przeniesienia przesyłana jest do bitu o podanym adresie bezpośrednim. $(bit) \leftarrow CY$
MOV C,bit Prześlij wartość bitu do flagi przeniesienia	Zawartość bitu o podanym adresie bezpośrednim przesyłana jest do flagi przeniesienia. $Cy \leftarrow (bit)$

Rozkazy skoków i rozkazy sterujące

AJMP adr11 Skocz bezwarunkowo na stronę	Zawartość licznika rozkazów zwiększa się o 2. Do bitów 0...10 licznika rozkazów jest wpisywany 11–bitowy adres bezpośredni, przy czym pięć bardziej znaczących bitów licznika rozkazów nie zmienia się. Skok wykonywany jest pod adres na stronie, na której jest umieszczony pierwszy bajt rozkazu następnego po AJMP. $PC_{10...0} \leftarrow adr11$
CJNE <r>,<s>,d Porównaj argumenty	Wskazane argumenty <r> i <s> są porównywane. Jeśli nie są równe, to następuje skok względem adresu

i skocz, jeśli nie są równe	pierwszego bajtu rozkazu następnego po CJNE. Jeżeli wartość <r> jest mniejsza niż <s>, to znacznik C przyjmuje wartość 1, w przeciwnym razie C jest zerowany. Możliwe są cztery tryby adresowania argumentów: CJNE A,ad,d PC ← PC + 3, jeśli A = (ad) PC ← PC + d, jeśli A ≠ (ad) CJNE A,#n,d PC ← PC + 3, jeśli A = n PC ← PC + d, jeśli A ≠ n CJNE Rr,#n,d PC ← PC + 3, jeśli Rr = n PC ← PC + d, jeśli Rr ≠ n CJNE @Ri,#n,d PC ← PC + 3, jeśli (Ri) = n PC ← PC + d, jeśli (Ri) ≠ n
DJNZ <r>,d Zmniejsz o 1 i skocz, jeśli nie zero	Od wskazanego argumentu odejmowana jest jedynka. Jeśli po tej operacji argument nie jest równy zero, to wykonywany jest skok względem adresu pierwszego bajtu rozkazu następnego po DJNZ, przy czym stan znaczników nie ulega zmianie. Możliwe są dwa rodzaje trybów adresowania: DJNZ Rr,d Rr ← Rr – 1, jeśli Rr ≠ 0, to PC ← PC + d jeśli Rr = 0, to PC ← PC + 2 DJNZ ad,d (ad) ← (ad) – 1, jeśli (ad) ≠ 0, to PC ← PC + d jeśli (ad) = 0, to PC ← PC + 2
JB bit,d Skocz, jeśli bit=1	Jeżeli wartość bitu o podanym adresie wynosi 1, to następuje skok względem adresu pierwszego bajtu rozkazu następnego po JB (do wskaźnika licznika rozkazów dodaje się przesunięcie *d*). Testowany bit nie ulega zmianie. PC ← PC + d, jeśli (bit) = 1 PC ← PC + 3, jeśli (bit) = 0
JBC bit,d Zeruj bit jeśli jest ustawio-	Jeśli wartość bitu o wskazanym adresie bezpośrednim wynosi 1, to jest on zerowany i wykonywany jest skok

ny i skocz	względem pierwszego bajtu rozkazu następnego po JBC (do zawartości licznika rozkazów dodawane jest przesunięcie *d*). PC ← PC + 3, jeśli (bit) = 0 PC ← PC + d, jeśli (bit) = 1
JC d Skocz, jeśli wystąpiło przeniesienie (jeśli CY= 1)	Jeżeli wartość flagi przeniesienia CY wynosi 1, to wykonywany jest skok względem pierwszego bajtu rozkazu następnego po JC, czyli do licznika rozkazów dodawane jest przesunięcie *d*. PC ← PC + 2, jeśli CY = 0 PC ← PC + d, jeśli CY = 1
JMP @A + DPTR Skocz pośrednio	Wykonywany jest skok pod adres będący sumą wartości DPTR i wartości akumulatora, przy czym zawartość akumulatora traktowana jest jako liczba dwójkowa bez znaku (z zakresu 0...255). Do licznika rozkazów wpisywana jest suma zawartości rejestru DPTR i akumulatora. PC ← A + DPTR
JNB bit,d Skocz, jeśli bit o podanym adresie jest zerowy	Jeżeli wartość bitu o podanym adresie bezpośrednim jest zerowa, to wykonywany jest skok względem adresu pierwszego bajtu rozkazu następnego po JNB (do zawartości licznika rozkazów dodawane jest przesunięcie *d*), przy czym wartość testowanego bitu nie ulega zmianie. PC ← PC +3, jeśli (bit) = 1 PC ← PC + d, jeśli (bit) = 0
JNC d Skocz, jeśli nie wystąpiło przeniesienie (jeśli CY=0)	Jeżeli wartość flagi przeniesienia CY wynosi 0, to wykonywany jest skok względem adresu pierwszego bajtu rozkazu następnego po JNC, czyli do zawartości licznika rozkazów dodawane jest przesunięcie *d*. PC ← PC + 2, jeśli CY = 1 PC ← PC + d, jeśli CY = 0
JNZ d	Jeżeli zawartość akumulatora nie jest równa zeru, to

Skocz, jeśli akumulator nie jest zerowy	następuje skok względem adresu pierwszego bajtu rozkazu następnego po JNZ, czyli do licznika rozkazów dodawane jest przesunięcie *d*. PC ← PC + 2, jeśli A = 0 PC ← PC + d, jeśli A ≠ 0
JZ d Skocz, jeśli akumulator jest zerowy	Jeżeli zawartość akumulatora jest równa 0, to następuje skok względem adresu pierwszego bajtu rozkazu następnego po JZ, czyli do licznika rozkazów dodawane jest przesunięcie *d*. PC ← PC + 2, jeśli A ≠ 0 PC ← PC + d, jeśli A = 0
LJMP adr16 Skocz bezwarunkowo	Do licznika rozkazów wpisywany jest 16–bitowy adres bezpośredni, pozwalając na skok pod dowolny adres w całej przestrzeni pamięci programu PC ← adr16
NOP Rozkaz pusty (nie rób nic)	Żadna operacja nie jest wykonywana.
SJMP d skocz bezwarunkowo	Do zawartości licznika rozkazów dodawane jest przesunięcie *d*. Skok jest wykonywany względem adresu pierwszego bajtu następnego rozkazu. PC ← PC + 2 PC ← PC + d
SETB bit Ustaw bit	Do bitu o podanym adresie bezpośrednim wpisywana jest 1. (bit) ← 1
SETB C Ustaw flagę przeniesienia	Do flagi przeniesienia wpisywana jest 1. CY ← 1

Rozkazy obsługi podprogramów, przerwań i operacji na stosie

ACALL adr11	Zawartość licznika rozkazów PC zwiększana jest o 2, a następnie odkładana na stos. Wskaźnik stosu jest

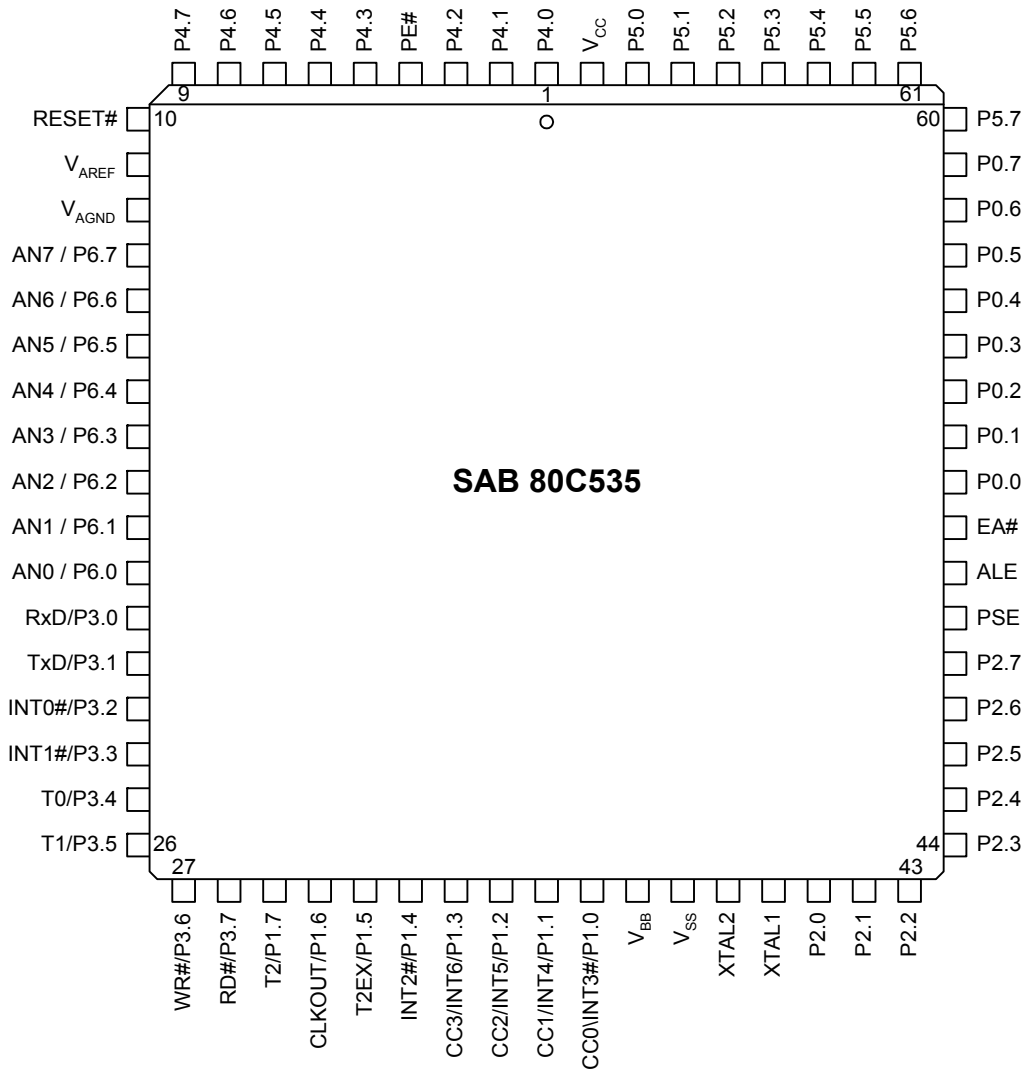
Skocz do podprogramu na stronie	zwiększany o 2, do bitów 0–10 licznika rozkazów jest wpisywany 11–bitowy adres bezpośredni, przy czym pięć bardziej znaczących bitów licznika rozkazów nie zmienia się. Skok do podprogramu wykonuje się na tej stronie, na której jest umieszczony pierwszy bajt rozkazu następnego po ACALL. PC ← PC + 2 SP ← SP + 1 (SP) ← $PC_{7...0}$ SP ← SP + 1 (SP) ← $PC_{15...8}$ $PC_{10...0}$ ← adr11
LCALL adr16 Wywołaj podprogram	Podczas pobrania rozkazu zawartość licznika rozkazów jest powiększana o 3 i ładowana na stos. Zawartość wskaźnika stosu jest zwiększana o 2, a do licznika rozkazów jest wpisywany 16-bitowy adres bezpośredni, pozwalając na skok do podprogramu pod dowolny adres w pamięci programu. PC ← PC + 3 SP ← SP + 1 (SP) ← $PC_{7...10}$ SP ← SP + 1 (SP) ← $PC_{15...8}$ PC ← adr16
POP ad Zdejmij ze stosu	Do komórki wewnętrznej pamięci danych o podanym adresie bezpośrednim wpisywane są dane z wierzchołka stosu, tzn komórki wewnętrznej pamięci danych o adresie zawartym we wskaźniku stosu SP. Wartość SP zmniejszana jest o 1. (ad) ← SP SP ← SP – 1
PUSH ad Ładuj na stos	Zawartość wskaźnika stosu zwiększana jest o 1, a następnie na wierzchołek stosu jest wpisywana zawartość komórki wewnętrznej pamięci danych lub rejestru spe-

	cjalnego SFR o podanym adresie bezpośrednim. SP ← SP + 1 (SP) ← (ad)
RET Wróć z podprogramu	Rozkaz powrotu z podprogramu. Adres powrotu jest wpisywany ze stosu do licznika rozkazów, po czym zawartość wskaźnika stosu zmniejszana jest o dwa. $PC_{15...8}$ ← (SP) SP ← SP – 1 $PC_{7...0}$ ← (SP) SP ← SP – 1
RETI Wróć z przerwania	Rozkaz powrotu z podprogramu obsługi przerwania. Adres powrotu jest wpisywany ze stosu do licznika rozkazów, po czym zawartość wskaźnika stosu zmniejszana jest o dwa. Wykonanie tego rozkazu jest dla systemu przerwań sygnałem zakończenia obsługi przerwania, czyli żadne zgłoszenie przerwania o takim samym lub niższym priorytecie nie będzie wcześniej przyjęte. $PC_{15...8}$ ← (SP) SP ← SP – 1 $PC_{7...0}$ ← (SP) SP ← SP – 1

ZAŁĄCZNIK 2

Schemat wyprowadzeń mikrokontrolera SAB 80C535

Obudowa PL-CC-68



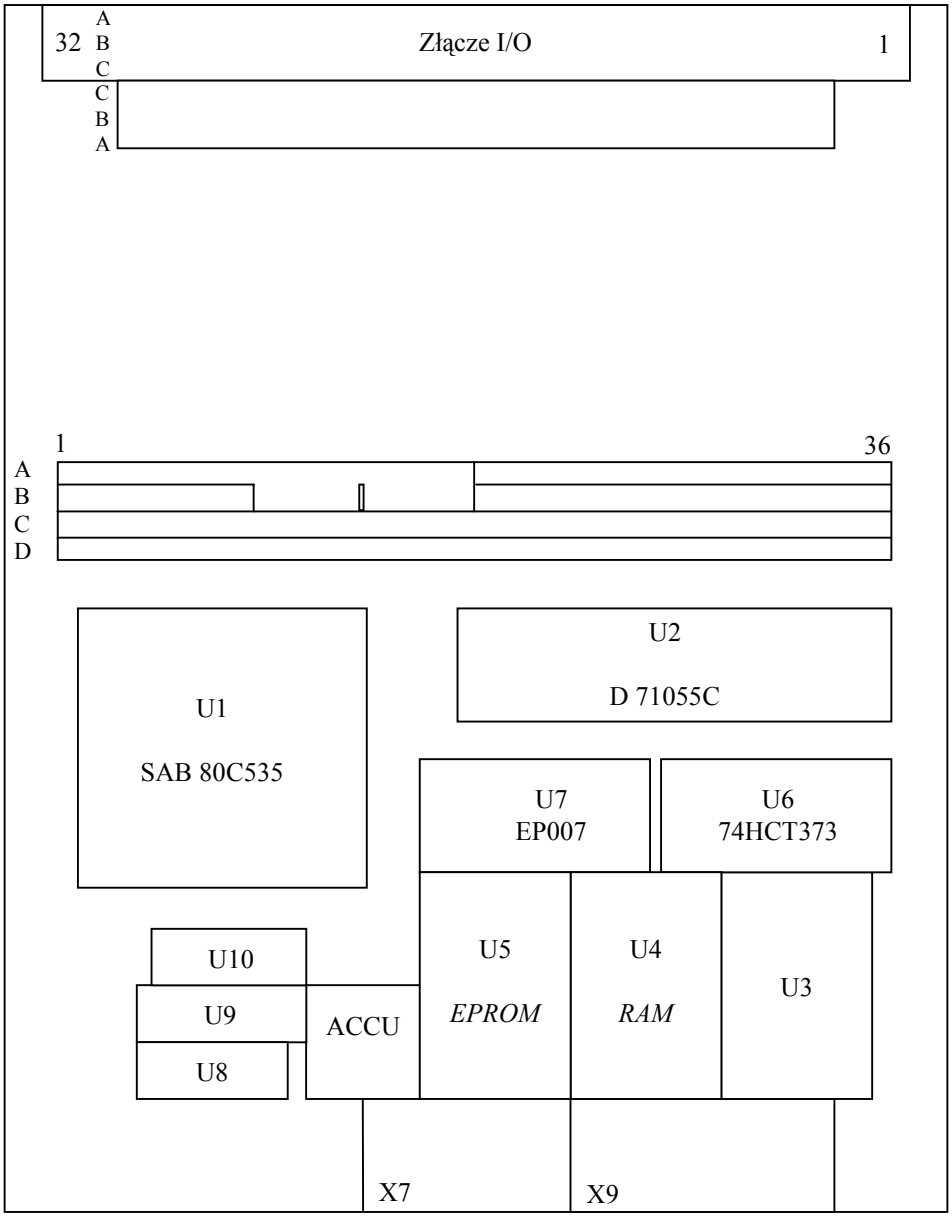
Opis wyprowadzeń mikrokontrolera SAB 80C535

Symbol	Nr wyprowa-dzenia	Typ	Funkcja

P4.0...P4.7	1...3, 5...9	we/wy	Uniwersalny port cyfrowy we/wy P4
PE#	4	we	Blokowanie trybu redukcji mocy
RESET#	10	we	Sprzętowy reset mikrokontrolera
V_{AREF}	11		Napięcie wzorcowe górne dla przetwornika a/c
V_{AGND}	12		Napięcie wzorcowe dolne dla przetwornika a/c
P6.0...P6.7	13...20	we	Wejścia analogowo-cyfrowe przetwornika a/c
P3.0...P3.7	21...28	we/wy	Uniwersalny port cyfrowy we/wy P3
XTAL1, XTAL2	39, 40		Podłączenie rezonatora kwarcowego lub zewnętrznego źródła taktującego
PSEN#	49	wy	Aktywacja zewnętrznej pamięci ROM
ALE	50	wy	Wyjście synchronizujące bufor zewnętrznej pamięci ROM i RAM
EA#	51	we	Wybór wewnętrznej/zewnętrznej pamięci programu ROM
P0.0...P0.7	52..59	we/wy	Uniwersalny port cyfrowy we/wy P0
P2.0...P2.7	41...48	we/wy	Uniwersalny port cyfrowy we/wy P2
P1.0...P1.7	29...36	we/wy	Uniwersalny port cyfrowy we/wy P1
P5.0...P5.7	60...67	we/wy	Uniwersalny port cyfrowy we/wy P5
V_{CC}	37,68		Napięcie zasilające
V_{SS}	38		Masa zasilania

ZAŁĄCZNIK 3

Rozmieszczenie elementów na płytce podstawowego modułu dydaktycznego MINICON



Oznaczenia układów scalonych są następujące:

U1 – mikrokontroler SAB 80C535

U2 – programowalny układ we/wy D71055C

U3 – wolne

U4 – pamięć RAM 62256 (32kB pamięci danych)

U5 – pamięć EPROM M27C256 (32kB pamięci kodu)

U6 – przerzutnik zatrzaskowy typu D 74HCT373

U7 – dekoder adresów (EP007 lub EP009)

U8 – sterownik łącza szeregowego TSC 232

U9 – zegar czasu rzeczywistego 72421A

U10 – zewnętrzny układ watchdog MAX 691

ACCU – bateria do zasilania rezerwowego

X7 – przycisk RESET

X9 – złącze szeregowe

ZAŁĄCZNIK 4

Schemat wyprowadzeń portów i sygnałów sterujących na płytce modułu dydaktycznego MINICON

	D	C	B	A
36	P4.1	P4.3	P4.5	P4.7
35	P4.0	P4.2	P4.4	P4.6
34	P5.0		VPD	/RESET
33	P5.1	P5.2	VAREF	VAGND
32	P5.3	P5.4	AN7	AN6
31	P5.5	P5.6	AN5	AN4
30	P5.7	D7	AN3	AN2
29	D6	D5	AN1	AN0
28	D4	D3	RXD	TXD
27	D2	D1	/INT0	/INT1
26	D0	/EA	T0	T1
25	ALE	/PSEN	/WR	/RD
24	A15	A14	P1.6	P1.7
23	A13	A12	P1.4	P1.5
22	A11	A10	P1.2	P1.3
21	A9	A8	P1.0	P1.1
20			PB2	PB3
19			PB1	PB4
18			PB0	PB5
17			PC3	PB6
16		* /LOWLINE	PC2	PB7
15			PC1	VCC
14			PC0	D7
13			PC4	D6
12			PC5	D5
11			PC6	D4
10			PC7	D3
9			A0	D2
8		A7	A1	D1
7		A6	GND	D0
6		A5	/CEPIO	RESPIO
5		A4	/RD	/WR
4		A3	PA0	PA7
3		A2	PA1	PA6
2		A1	PA2	PA5
1		A0	PA3	PA4

* sygnał /LOWLINE równoważny jest sygnałowi /RESET

LITERATURA

1. Brighouse B., Loveday G., *Microprocessors in engineering systems*, PITMAN Publishing 1987.
2. Grabowski J., Koślacz S., *Podstawy i praktyka programowania mikroprocesorów*, WNT, Warszawa 1987.
3. Janiczek J., Stępień A., *Mikrokontroler SAB 80(C)515/535,* Wydawnictwo EZN, Wrocław 1995.
4. Janiczek J., Stępień A., *Laboratorium systemów mikroprocesorowych* cz. 1 i 2*,* Wydawnictwo EZN, Wrocław 1995.
5. Majewski J., Kardach K. , *Mikrokontrolery jednoukładowe 805. Programowanie w języku C w przykładach,* Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 1995.
6. Misiurewicz P., *Podstawy techniki mikroprocesorowej*, WNT, Warszawa 1991.
7. Niederliński A., *Mikroprocesory, mikrokomputery, mikrosystemy*, WSiP, Warszawa 1987
8. Rydzewski A., *Mikrokontrolery jednoukładowe rodziny MCS-51,* WNT, Warszawa 1997.
9. Wójciak A., *Mikroprocesory w układach przekształtnikowych*, WNT, Warszawa 1992.
10. *Microcomputer Components. SAB 80515 / SAB 80C515 8 bit single chip Microcontroller Family*. User's Manual, SIEMENS.