Mikrokontrolery? To takie proste...

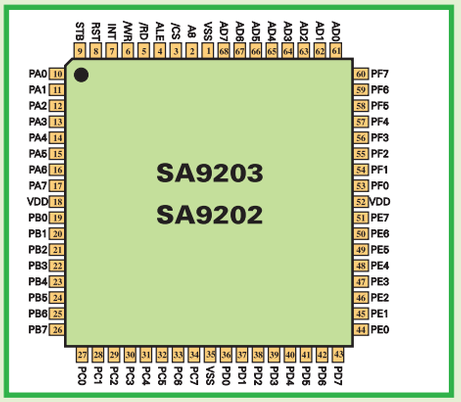
Część 19
Programowane układy wejścia-wyjścia cz III

W tym odcinku kończymy omawianie układów I/O ciekawym przykładem wykorzystania dostępnego ostatnio na rynku polskim układu scalonego SA9203. Kostka może stać się otyle atrakcyjna gdyż jest ostatnio dostępna w sieci handlowej AVT, toteż każdy z Was kto zainteresuje się niniejszym układem , będzie mógł spróbować wykorzystać go dla swoich potrzeb – a jak się za chwilę przekonacie zastosowań może być wiele. Dziś także rozwiązanie zadania z poprzedniej lekcji – czyli drukowanie z komputerka AVT-2250 na zwykłej drukarce komputerowej wyposażonej w równoległe złącze standardu Centronics.

Dziś chciałbym zapoznać Cię drogi czytelniku z ciekawym układem I/O, którego możliwości odpowiadają mniej więcej dwóm, omawianym wcześniej układom 8255. Chodzi o układ dość egzotycznej bo południowoafrykańskiej firmy produkującej szeroka gamę układów dla potrzeb nowoczesnej telekomunikacji - SAMES. Mowa będzie o układzie SA9203. Ponieważ od niedawna układ ten jest dostępny w ofercie handlowej AVT, a jego cena jest przystępna jak na możliwości kostki, postanowiłem zakończyć cykl o układach I/O omówieniem właśnie jego możliwości. Drugim powodem dla którego chcę zwrócić szczególną uwagę na kostkę SA9203 to idealna wprost kompatybilność z rodziną mikroprocesorów 8051 jeżeli chodzi o połączenie obu tych układów. Dzięki zastosowaniu SA9203 system mikroprocesorowy może zostać wzbogacony o 6 dodatkowych uniwersalnych portów wejścia-wyjścia, każdy o szerokości 8-miu bitów. W sumie daje to aż 48! dodatkowych końcówek do dowolnego wykorzystania.

UKŁAD SA9203 – TROCHĘ TEORII

Kostka została wykonana w technologii CMOS. Jak wcześniej powiedziałem, układ świetnie nadaje się do podłączenia z mikrokontrolerem 8051 a to ze względu na fakt posiadania multipleksowanej szyny danych i adresu. Układ dostarczany jest przez producenta w typowej obudowie PLCC68. Nie powinno to jednak odstraszyć od zastosowania nawet amatora elektroniki, bowiem do układu można zastosować odpowiednią podstawkę która posiada typowy calowy rozstaw (2,54 mm). Na rys.1 przedstawiono opis wyprowadzeń układu a na rys.2 schemat wewnętrzny układu SA9203.



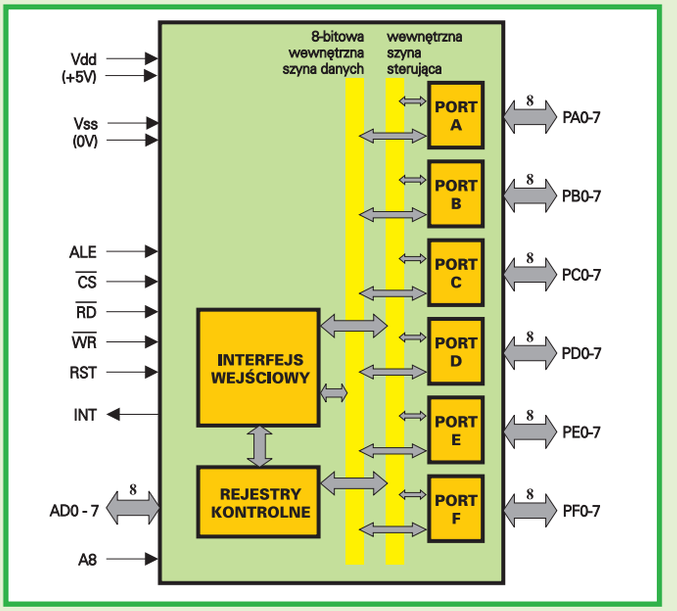
Rys.1 Rozkład wyprowadzeń układu SA9203.

Układ składa się z: interfejsu wejściowego, łączącego multipleksowaną szynę danych / adresu z zewnętrznym układem sterującym, rejestrów konfiguracyjnych oraz rejestrów 6-ciu portów A...F. Znaczenie zewnętrznych sygnałów sterujących podaję w tabeli 1.

W tabeli 2 przedstawiam parametry elektryczne układu SA9203.

Jak widać parametry odpowiadają warunkom zasilania TTL, czyli 5V. Układ SA9203 może z powodzeniem pracować z układami wykonanymi w technologii CMOS, a także TTL-LS, lub nawet TTL przy zachowaniu odpowiednich dopuszczalnych obciążeń.

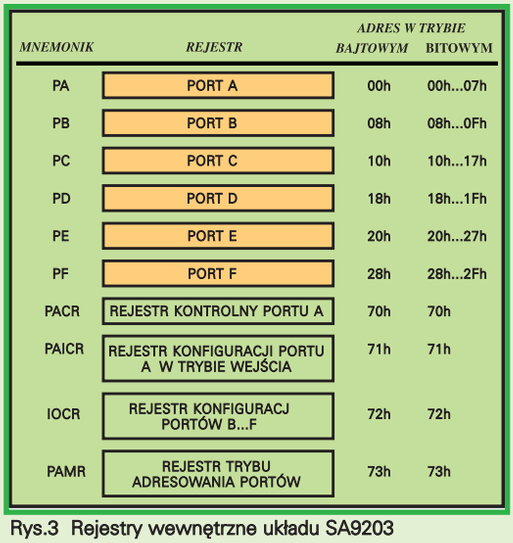
Na rys.3 przedstawiono układ i adresy wewnętrznych rejestrów układu. Układ podobnie jak mikroprocesor 8051 posiada 8-bitową multipleksowaną szynę adresową – danych AD0...AD7, co dla zewnętrznego układu sterującego kostką SA9203 zajmuje obszar 256 bajtów w jego (8051) przestrzeni adresowej. Wybrane obszary układu zajęte są przez



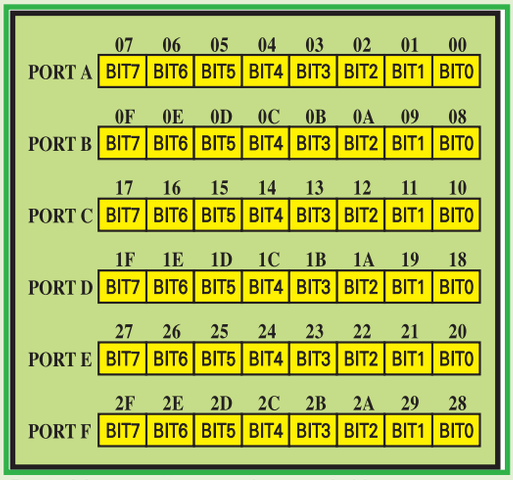
Rys.2 Schemat wewnętrzny układu SA9203.

Tabela 1

Pin	Typ	Symbol	Opis
18,52		VDD	zasilanie układu +5V;
1,35		VSS	masa zasilania 0V
61...68	we/wy	AD0...AD7	3-stanowa multipleksowana szyna danych/adresu. 8-bitowy adres zatrzaskiwany jest we wnętrzu układu SA9203 podczas opadającego zbocza sygnału ALE. Dane zapisywane są lub odczytywane po podaniu odpowiednio sygnałów /WR (zapisu) lub /RD (odczytu);
2	we	A8	nie używany w układzie SA9203, powinien być połączony z masą (Vss), znaczenie tego pinu opiszę w dalszej części artykułu;
3	we	/CS	aktywny niski sygnał na tym wejściu powoduje wybór układu;
4	we	ALE	pin kontrolujący zatrzaśnięcie adresu na szynie AD0...AD7 podczas zapisu przez urządzenie zewnętrzne, następuje to podczas opadającego zbocza tego sygnału;
5	we	/RD	poziom niski na tym wejściu pozwala na odczyt wewnętrznych rejestru układu
6	we	/WR	poziom niski na tym wejściu powoduje zapis danej do wewnętrznego rejestru układu
7	wy	INT	programowane wyjście zgłoszenia przerwania do układu zewnętrznego, możliwość ustalenia polaryzacji oraz uaktywnienia tego pinu
8	we	RST	wysoki poziom podany na to wejście powoduje zresetowanie układu
9	we	STB	wejście zatrzaskiwania danej w porcie A, gdy port ten pracuje jako wejście,
10...17	we/wy	PA0...PA7	uniwersalny 8-bitowy port I/O. Możliwość indywidualnego zdefiniowania każdego pinu portu jako zatrzaskiwanego wyjścia lub wejścia. W trybie pracy jako wejście port może pracować w trybie zatrzaskiwania (sygnałem STB) lub jako "przezroczysty" ("transparent")
19...26	we/wy	PB0...PB7	8-bitowy uniwersalny port I/O. Wszystkie piny mogą być ustawione jako zatrzaskiwane wyjścia lub jako wejścia typu "transparent".
27...34	we/wy	PC0...PC7	identyczny jak port B
36...43	we/wy	PD0...PD7	identyczny jak port B
44...51	we/wy	PE0...PE7	identyczny jak port B
53...60	we/wy	PF0...PF7	identyczny jak port B



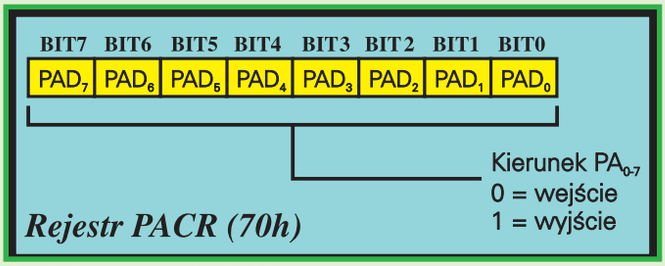
Rys.3 Rejestry wewnętrzne układu SA9203



Rys.4 Mapa portów przy adresowaniu bitowym.

Poniżej zapoznam cię ze znaczeniem poszczególnych portów kontrolnych układu SA9203.

PACR - rejestr kontrolny portu A. Na rys.5 pokazano znaczenie poszczególnych bitów rejestru. Odpowiednie ustawienie lub wyzerowanie pozycji pozwala na indywidualne ustalenie pinu portu A jako wejścia lub wyjścia cyfrowego.



Rys.5 Rejestr specjalny PACR

PAICR - rejestr konfiguracyjny portu A w trybie wejścia. Rys.6 przedstawia znaczenie bitów rejestru. W przypadku ustawienia pinu jako zatrzaskiwane wejście ("latched input") dane zostają zapamiętane po nadejściu sygnału na wejściu STB. W przypadku ustawienia dowolnego pinu portu A jako wyjście, odpowiadający mu bit w rejestrze PAICR nie ma wpływu na działanie tej linii portu.

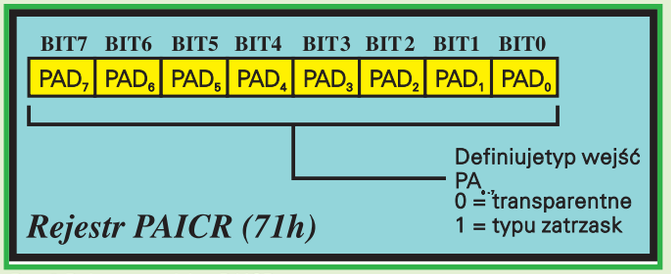
IOCR - rejestr konfiguracyjny wejścia-wyjścia. Rejestr pozwala na konfigurację portów konfiguracyjne oraz porty PA...PF jak pokazano na rysunku. Układ pozwala na dwojakie adresowanie każdego z rejestrów portów PA...PF. Pierwszy polega na jednoczesnym adresowaniu całego portu, drugi pozwala na zaadresowanie pojedynczego pinu każdego z portów. W tym drugim przypadku przy odczycie danej w postaci bajtu, siedem najstarszych bitów nie ma znaczenia, jedynie najmłodszy D0 wskazuje na stan pinu lub wymusza go w przypadku pracy portu jako cyfrowego wyjścia.

Dodatkowe rejestry sterujące pracą całego układu zawsze adresowane są bajtowo. Dość użyteczną funkcją w przypadku tych rejestrów jest możliwość odczytu ich zawartości. Dzięki temu programista nie musi zapamiętywać ich stanu po zapisie w dodatkowych zmiennych wykorzystywanych w programie.

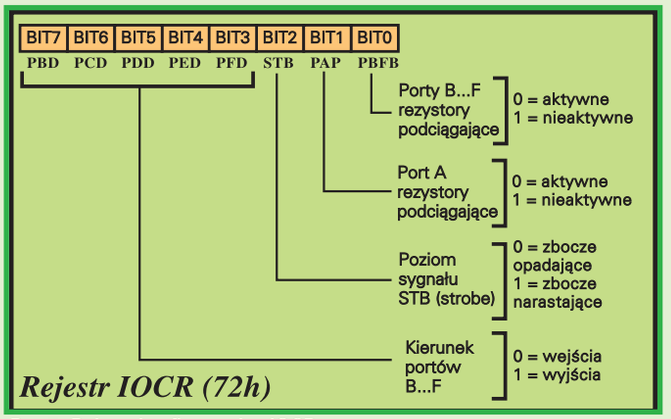
Na rys.4 przedstawiona jest mapa adresowa poszczególnych portów w trybie adresowania bitowego.

Tabela 2

Parametr	Symbol	Min	Typ	Max	Jednostka	Warunek pomiaru
zasilanie	Vdd	4,75	5,0	5,25	V	
pobór prądu (statyczny)	Idds		15	50	uA	Vdd = 5,0V
pobór prądu (dynamiczny)	Iddd			20	mA	Vdd = 5,0V
napięcie wej. w stanie wysokim	Vih	2,0			V	Vdd = 5V
napięcie wej. w stanie niskim	Vil			1,0	V	Vdd = 5V
napięcie wyj. w stanie wysokim	Voh	4,5	4,7		V	Vdd = 5V Ioh = 5mA
napięcie wyj. w stanie niskim	Vol		0,25	0,5	V	Vdd = 5V Ioh = 7mA



Rys.6 Rejestr specjalny PAICR



Rys.7 Rejestr konfiguracyjny IOCR.

portów PB...PF, załączanie wewnętrznych rezystorów podciągających w trybie wejścia, oraz określa polaryzację sygnału STB (patrz rys.7).

Jak wspomniałem wcześniej wszystkie wyprowadzenia portów układu SA9203 posiadają wbudowane rezystory podciągające pin portu kiedy ten pracuje jako wejście. Zwalnia to użytkownika od stosowania dodatkowych elementów rezystancyjnych w niektórych aplikacjach. Rezystory te mogą być uaktywnione lub wyłączone oddzielnie dla każdego z portów PA...PF

Bit IOCR.2 definiuje polaryzację sygnału STB ("strobe") który zatrzaskuje daną w porcie A (lub wybranych jego pinach) kiedy ten pracuje jako wejście. W przypadku ustawienia bitu na 0, dane zatrzaskiwane są podczas opadającego zbocza sygnału STB, gdy bit = 1, podczas narastającego zbocza sygnału.

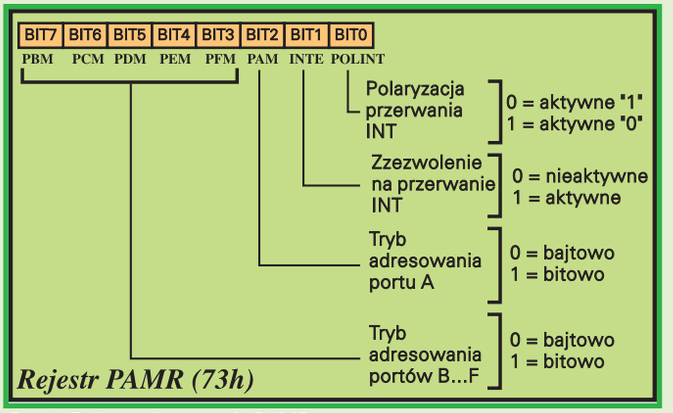
PAMR - rejestr trybu adresowania. Zgodnie z rys.8 ustawienie lub wyzerowanie odpowiedniego bitu w tym rejestrze pozwala na zmianę trybu adresowania poszczególnych portów A...F, aktywację oraz polaryzację sygnału zgłoszenia przerwania INT.

INT - wyjście zgłoszenia przerwania w przypadku zatrzaśnięcia danej w porcie A po nadejściu sygnału STB. Aktywacja pinu następuje po odpowiednim ustawieniu bitu PAMR.1. Polaryzacja zgłoszenia przerwania ustalana jest poprzez bit PAMR.0.

RST - podanie wysokiego poziomu na to wejście resetuje cały układ. Zawartość wszystkich rejestrów zostaje wyzerowana. Następstwem tego jest:

- wszystkie piny portów PA...PF ustawione zostają jako wejścia
- wejścia portu A pracują jako transparentne
- rezystory podciągające są uaktywnione we wszystkich portach
- wyjście przerwania jest nieaktywne
- adresowanie portów ustalone zostaje jako bajtowe

Aktywny (wysoki) stan sygnału RST powinien trwać minimum 100ns.



Rys.8 Rejestr adresowania PAMR.

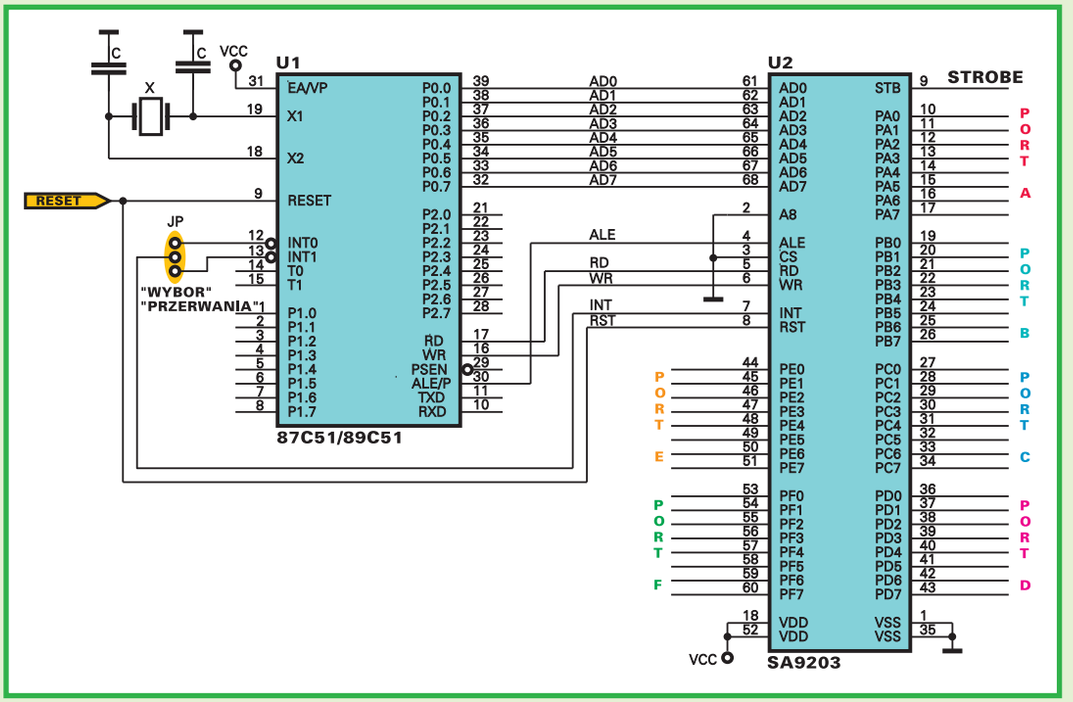
PRAKTYCZNE UKŁADY Z SA9203

Nie przedłużając teoretycznych wywodów na temat układu SA9203 na rys.9 przedstawiam najprostszy sposób na połączenie z mikroprocesorem serii '51 w wersji z wewnętrzną pamięcią programu.

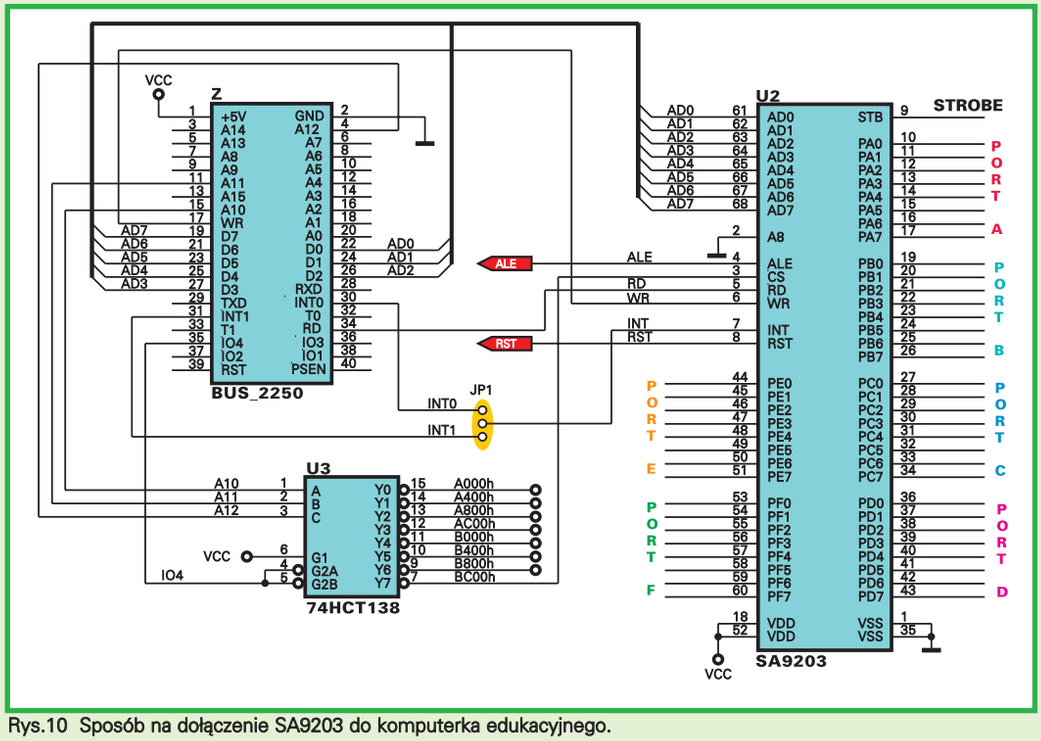
Jak widać z rysunku, do działania nie są potrzebne żadne dodatkowe układy sterujące czy dekoder adresów. Dzięki wspomnianej wcześniej kompatybilnej z 8051, magistrali adresowej-danych układ SA9203 "widziany" jest przez procesor jako 256 komórek w zewnętrznej pamięci danych, do której procesor odwołuje się za pośrednictwem rozkazów:

MOVX @DPTR, A podczas zapisu do portu
oraz
MOVX A, @DPTR przy odczycie z portu

Tak naprawdę, to możliwe jest zaadresowanie (zgodnie z rys.3) tylko do adresu 73h, pod którym znajduje się rejestr PAMR. Pozostałe komórki są niewykorzystane. Ktoś może zapytać, po co w układzie SA9203 potrzebna jest linia adresowa A8 ? Otóż istnieje wersja układu SA9202, kompatybilna z omawianym SA9203, lecz posiadająca dodatkowo w swojej strukturze 256 bajtów statycznej pamięci RAM. Tak więc zaadresowanie układu SA9202 w zakresie 100h...1FFh (linia A8=1) powoduje dostęp do tej właśnie pamięci. Jeżeli ktoś z czytelników będzie miał okazję nabyć tę wersję układu, zachęcam do zakupu i eksperymentów. Dodatkowa pamięć RAM idealnie może nadawać się do przechowywania danych podczas wykonywania jakiegoś programu. Zintegrowanie jej w układzie portu SA9202 zwalnia od stosowania dodatkowej kostki SRAM, co ma wpływ na wielkość i skomplikowanie obwodu drukowanego.



Rys.9 Współpraca układu SA9203 z kontrolerem 87C51/89C51



Rys.10 Sposób na dołączenie SA9203 do komputerka edukacyjnego.

W przypadku chęci dołączenia układu SA9203 do komputerka AVT-2250 najlepiej posłużyć się schematem z rys.10.

W układzie oprócz SA9203 zastosowano dodatkowe dekodowanie adresów dzięki wykorzystaniu dekodera 1 z 8-miu typu 74HCT138 (74LS138). Użycie tego układu wydało mi się uzasadnione, a to ze względu na poprzednio prezentowane przykłady. Jak zapewne pamiętasz wszystkie one korzystały z sygnału magistrali komputerka IO4. Ponieważ możesz zastosować ich kilka za jednym razem, należało zastosować dodatkowe dekodowanie adresów, tak aby podzielić obszar adresowy dekodowany przez sygnał IO4 na kilka mniejszych (w tym przypadku na osiem 1kbajtowych). W naszym przykładzie z rys.10 układ SA9203 dekodowany jest przez sygnał Y7 kostki U3, co odpowiada adresom BC00h...BFFFh. Ponieważ na złącze BUS_2250 komputerka nie wyprowadzono sygnałów ALE oraz RESET (polaryzacja dodatnia), należy te dwa połączenia wykonać bezpośrednio łącząc te końcówki kawałkiem drutu.

Aby teraz "dobrać" się do układu SA9203 należy w programie jego obsługi zadeklarować następująco:

```
SA9203          EQU     BC00h
```

Ze względu na to że układ zawiera kilka rejestrów wewnętrznych, które opisałem wcześniej w programie warto od razu zapisać kolejne deklaracje:

```
PORTA   EQU   BC00h   ;adres portu PA
PORTB   EQU   BC08h   ;adres portu PB
PORTC   EQU   BC10h   ;adres portu PC
PORTD   EQU   BC18h   ;adres portu PD
PORTE   EQU   BC20h   ;adres portu PE
PORTF   EQU   BC28h   ;adres portu PF
PACR    EQU   BC70h   ;adres rejestru kontrolnego portu PA
PAICR   EQU   BC71h   ;adres rej. konfiguracji portu PA
IOCR    EQU   BC72h   ;adres rej. konfiguracji wejść-wyjść
PAMR    EQU   BC73h   ;adres rejestru wyboru trybu adresowania
```

Można od razu zadeklarować adresy poszczególnych bitów portów, które przydadzą się w trybie adresowania bitowego, np. tak:

```
BITPA0 EQU    BC00h    ;adres pinu 10 układu (PA0)
BITPA1 EQU    BC01h
BITPA2 EQU    BC02h
BITPA3 EQU    BC03h
BITPA4 EQU    BC04h
BITPA5 EQU    BC05h
BITPA6 EQU    BC06h
BITPA7 EQU    BC07h    ;adres pinu 17 układu (PA7)
BITPB0 EQU    BC08h
BITPB7 EQU    BC0Fh
```

i tak dalej aż do portu PF gdzie deklaracje będą wyglądały jak następuje:

```
BITPF0 EQU    BC28h
BITPF7 EQU    BC2Fh
```

Teraz można zabrać się do programowania układu. W naszym pierwszym przykładzie zaprogramujemy układ SA9203 do pracy w trybie wyjścia – wszystkie porty PA...PF ustawimy jako wyjścia.

Najpierw zajmiemy się rejestrem PACR – konfiguracji portu PA. Ponieważ port PA ma pracować jako wyjście zgodnie z rys.5 należy ustawić wszystkie bity tego rejestru, tak więc piszemy instrukcje:

```
MOV    A, #0FFh
MOV  DPTR, #PACR
MOVX  @DPTR, A
```

Ponieważ port PA ma pracować jako wyjście programowanie rejestru PAICR jest zbędne. Pora teraz na rejestr IOCR, gdzie należy ustawić bity 7...3 tak aby porty PB...PF pracowały jako wyjścia. Pozostałe bity rejestru są w trybie wyjścia nieużywane. Tak więc zapiszemy:

```
MOV    A, #11111000b
MOV  DPTR, #IOCR
MOVX  @DPTR, A
```

Jako ostatni zostaje rejestr PAMR. Ustawmy adresowanie portów PA...PF jako bitowe, co pozwoli na oddzielne sterowanie każdy pinem układu. Dodatkowo, w przypadku, kiedy wyjście przerwania INT układu SA9203 połączone będzie z jednym z wejść procesora INT0 lub INT1 (poprzez jumper JP1) dobrze jest profilaktycznie ustawić polaryzację sygnału zgłoszenia przerwania, mimo iż w naszym przykładzie ta funkcja nie będzie wykorzystywana. Można to zrobić ustawiając bit 0 w rejestrze PAMR. Bit 1 rejestru powinien być wyzerowany – funkcja przerwania jest nieaktywna.

Instrukcja konfigurująca rejestr będzie następująca:

```
MOV   A, #11111101b
MOV   DPTR, #PAMR
MOVX  @DPTR, A
```

I to już wszystko. Teraz aby np. ustawić poziom wysoki na końcówce 3 portu PC należy wykonać instrukcje:

```
MOV   DPTR, #BITPC3      ;adres pinu PC3
MOV     A, #1                    ;wpisanie jedynki do bitu portu
MOVX  @DPTR, A
```

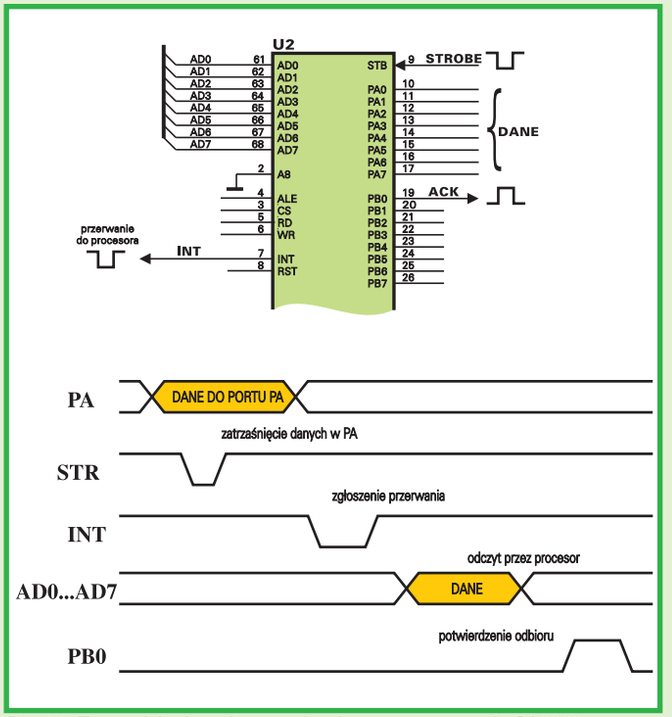
Podobnie można postąpić z każdym innym portem.

W drugim przykładzie wykorzystamy generowanie sygnału przerwania przez układ SA9203 przy odbiorze danej z portu PA, kiedy to zewnętrzne urządzenie chce przesłać daną do tego portu. Sytuację tę ilustruje rys.11

Sytuacja na rysunku przypomina przykład przedstawiony w poprzednim odcinku klasy mikroprocesorowej, kiedy to omawiałem układ 8255 i jego współpracę z drukarką. Wracajmy jednak do naszego przykładu. Do prawidłowej transmisji danych z potwierdzeniem wykorzystamy:

- port PA do transmisji danych
- linię STB do potwierdzania zapisu danych do PA przez urządzenie zewnętrzne
- linię PB0 do potwierdzenie odbioru danych przez procesor z SA9203
- linię INT układu SA9203 do zgłoszenia przerwania

Przyjrzyjmy się przebiegom z rys.11. Urządzenie zewnętrzne wystawia na linie portu PA daną. Potwierdza to przez podanie stanu niskiego na linię STROBE układu SA9203. Układ ten zapamiętuje daną w wewnętrznym rejestrze PA. Następnie SA9203 zgłasza przerwanie do pro-



Rys.11 Transmisja danych z urządzenia zewnętrznego do SA9203 z potwierdzeniem.

cesor wystawiając poziom niski na linię INT. Procesor w procedurze obsługi przerwania odczytuje daną z portu PA adresując układ SA9203 – w sposób pokazany wcześniej. Po odbiorze danej procesor powinien poinformować urządzenie zewnętrzne – nadajnik o gotowości na odbiór kolejnej danej. W tym celu korzystając z linii portu PB0 procesor wystawiając nań stan wysoki potwierdza odbiór (ACK) znaku i gotowość na następny.

Zaprogramujmy więc układ do pracy w tym trybie.

```
1. Najpierw konfigurujemy rejestr PACR:
CLR     A                   ;PA jako wejścia
MOV DPTR, #PACR
MOVX  @DPTR, A

2. Teraz rejestr PAICR:
MOV  A, #0FFh               ;dana zatrzaskiwana sygnałem STB
MOV  DPTR, #PAICR
MOVX  @DPTR, A

3. Następnie rejestr IOCR:
MOV  A, #10000010b          ;port PB – wyj., PC...PF – wej.
                            ;polaryzacja STB ujemna
                            ;rezystory podciągające PA nieaktywne
MOV  DPTR, #IOCR
MOVX  @DPTR, A

4. Pozostał jeszcze rejestr PAMR
MOV  A, #10000011b          ;port PB adresowany bitowo
                            ;port PA adresowany bajtowo
                            ;zezwolenie na INT
                            ;polaryzacja INT ujemna
MOV  DPTR, #PAMR
MOVX  @DPTR, A
```

I gotowe, układ SA9203 jest gotowy do odbioru danych z potwierdzeniem. Pozostaje jeszcze tylko ustawić odpowiednio procesor tak aby, uaktywnić wybrane jumperem JP1 (rys.10) przerwanie INT0/1. Poniżej przedstawię przykład kiedy do zgłoszenia przerwania wykorzystano linię INT1. W listingu dodatkowo będziemy wykorzystywać flagę F0 rejestru PSW procesora do detekcji nadejścia znaku w przerwaniu. Bit F0 będzie po prostu ustawiany przez procedurę obsługi przerwania jeżeli zostanie ono zgłoszone. A zerowany po odbiorze znaku. Przed sekwencją inicjująca zapiszmy więc procedurę obsługi przerwania INT1:

```
INT1_proc:
        SETB    F0          ;ustawienie znacznika o zgłoszeniu
        POP     DPL
        POP     DPH
        POP     Acc         ;odtworzenie rejestrów DPTR i Acc Bios'a
        RETI                ;patrz odcinek klasy o przerwaniach
```

Sekwencja inicjująca procesor powinna mieć postać:

```
init_proc:
        MOV             vectors, #80h  ;ustawienie offsetu tabeli
                                       przerwań w ext. RAM
        SETB    EX1     ;zezwolenie na INT1
        SETB    EA      ;odblokowanie przerwań
        CLR     F0      ;wyzerowanie znacznika zgłoszenia
        MOV     DPTR, #BITPB0
        CLR     A
        MOVX    @DPTR, A;ustawienie pinu PB0 w stan niski
        RET
```

Następnie możemy zdefiniować procedurę, która czeka na znak a po odbiorze ładuje go do akumulatora:

```
odbierz_znak:
        jnb     F0, odbierz_znak    ;czekanie na pojawienie się
znaku
        MOV     DPTR, #PORTA
        MOVX    A, @DPTR  ;odczyt znaku z portu PA SA9203
        PUSH     A        ;przechowanie znaku na stosie
        MOV     DPTR, #BITPB0
        MOV     A, #1
        MOVX    @DPTR, A  ;ACK = 1, potwierdzenie odbioru
        NOP
        NOP             ;instrukcje NOP w celu opóźnienia
                        ;sygnału ACK = 0, ilość zależna od
        NOP             ;tego jaką długość ma mieć ACK, aby być
        NOP             ;prawidłowo odczytanym przez nadajnik
        CLR     A
        MOVX    @DPTR, A ; ACK = 0, zakończenie odbioru
        POP     A       ;odtworzenie akumulatora
        CLR     F0      ;i jeszcze wyzerowanie znacznika
        RET             ;zakończenie procedury i powrót
```

Teraz tak stworzone procedury można użyć w programie:

```
CPU             '8052.def'
include 'const.inc'
include 'bios.inc'

        ORG             8000h
        LJMP            START
        ORG             8013h
INT1_proc:
        SETB    F0      ;ustawienie znacznika o zgłoszeniu
        POP     DPL
        POP     DPH
        POP     Acc     ;odtworzenie rejestrów DPTR i Acc Bios'a
        RETI
init_proc:
                ;tu wstawić ciało procedury j/w
        RET
odbierz_znak:   ;tu wstawić procedurę j/w
        .......
        RET
START:
        LCALL   init_proc  ;inicjacja portów i układu przerwań
        LCALL   odbierz_znak  ;wywołanie procedury odbioru znaku
                ;dalsze instrukcje działające na odebranym znaku
        .....
        END
```

Wnikliwy Czytelnik z pewnością zauważy, że procedurę odbioru znaku z portu SA9203 można umieścić w ciele procedury obsługi przerwania, tak, aby po zgłoszeniu przerwania nowy znak był odczytany prawidłowo. Wtedy procedura ta może wyglądać następująco:

```
INT1_proc:
MOV             DPTR, #PORTA
        MOVX    A, @DPTR        ;odczyt znaku z portu
                                 PA SA9203
        MOV     BUFOR, A        ;zapisanie znaku w buforze
        MOV     DPTR, #BITPB0
        MOV     A, #1
        MOVX    @DPTR, A        ;ACK = 1, potwierdzenie
                                 odbioru
        NOP
```

```
        NOP                     ;instrukcje NOP w celu opóźnienia
        CLR     A
        MOVX    @DPTR, A        ; ACK = 0, zakończenie odbioru
        SETB    F0              ;ustawienie znacznika o odbiorze znaku
  POP           DPL
        POP             DPH
        POP             Acc             ;odtworzenie rejestrów DPTR i Acc Bios'a
        RETI
```

W tym przypadku pojawia się tajemnicza zmienna BUFOR która jest niczym innym jak zdefiniowanym wcześniej adresem w wewnętrznej pamięci RAM procesora np.

```
BUFOR EQU               7Fh     ;adres komórki przechowującej bajt z SA9203
```

Należy zwrócić jednak uwagę na pewne niedogodności przedstawionej wyżej procedury. Po pierwsze nie jest wskazana umieszczanie zbyt wielu instrukcji NOP ze względu na to że wydłuża to procedurę przerwania, i może spowodować zakłócenia w pracy innych przerwań, np. od wyświetlacza, który może zacząć migotać. Po drugie ze względu na to że potwierdzenie odbioru (ACK) jest generowane w procedurze przerwania także, może się zdarzyć, że nie zdążymy odebrać znaku z komórki BUFOR, zanim nadejdzie kolejny bajt informacji z urządzenia zewnętrznego, które po potwierdzeniu zapisze nową daną w SA9203. W tym celu warto stworzyć w programie strukturę – tablicę w RAM której zadaniem byłoby przechowywanie danych odbieranych z układu SA9203. Trzeba by jednak dodatkowo użyć także wskaźnika tej tablicy, który określałby miejsce zapisu nowego odebranego znaku i odczyt ostatnio odebranego. Proponuję powyższe zadanie jako pracę domową. Z pewnością temat jest ciekawy a rozwiązań kilka. Najciekawsze przedstawię w kolejnym odcinku klasy mikroprocesorowej.

W jednym z najbliższych odcinków naszego kursu zapoznam Was z możliwościami jakie odkrywają procesory serii 51 z wbudowaną pamięcią programu, o sposobach ich używania, programowania. Pokażę jak w dość prosty i wygodny sposób, korzystając z kilku dodatkowych urządzeń peryferyjnych, szybko tworzyć programy, testować je a następnie zapisywać w strukturze procesorów np. 87C51, czy 89C51.

Wszystko to z wykorzystaniem i do zastosowania w komputerku edukacyjnym AVT-2250. Tak więc do zobaczenia.

Sławomir Surowiński

LEKCJA 12

Ostatnim razem pozostawiłem Was z problemem wydrukowania zawartości pamięci ext. RAM z komputerka w posaci jak pokazano poniżej.

Wydruk miał przedstawiać zawartość pamięci o podanym wcześniej z klawiatury komputerka zakresie adresów. Poniżej przedstawiam listing gotowej procedury służącej realizacji tego zamierzenia. (patrz Listing 1)

```
Adres               Dana Hex                    Dana ASCII
-------------------------------------------------------------------------------
(2600h)  0E 57 31 C0 50 9A E2 36 7D 0E 9A DF 35 7D 0E 80   .W1.P..6}...5}..
(2610h)  3E CA 8A 00 74 06 C6 46 FB 01 EB 04 C6 46 FB 02   >...t..F.....F..
(2620h)  8A 46 FB 50 B0 0F 50 9A AA 10 CF 0C 8D 7E FB 16   .F.P..P......~..
(2630h)  57 B0 02 50 E8 05 F8 8D 7E FC 16 57 9A 1F 0C CF   W..P....~..W....
(2640h)  0C 80 3E 7E 89 00 74 07 C6 06 7E 89 00 EB 17 8A   ..>~..t...~.....
(2650h)  46 FB 3C 01 75 07 C6 06 CA 8A 01 EB 09 3C 02 75   F.<.u........<.u
(2660h)  05 C6 06 CA 8A 00 89 EC 5D C3 55 89 E5 31 C0 9A   ........].U..1..
(2670h)  30 05 7D 0E BF 98 89 1E 57 B8 B7 00 50 9A E3 37   0.}.....W...P..7
(2680h)  7D 0E BF 98 89 1E 57 BF 18 8A 1E 57 9A 96 38 7D   }.....W....W..8}
(2690h)  0E 83 C4 04 BF 98 89 1E 57 9A 5B 38 7D 0E 9A 47   ........W.[8}..G
(26A0h)  0C CF 0C B0 07 50 E8 8F FA 31 C0 9A 16 01 7D 0E   .....P...1....}.
(26B0h)  5D C3 00 55 89 E5 31 C0 9A 30 05 7D 0E 80 3E C8   ]..U..1..0.}..>.
(26C0h)  8A 00 B0 00 75 01 40 A2 C8 8A BF 94 89 1E 57 9A   ....u. ........W.
(26D0h)  1F 0C CF 0C C6 06 7B 89 00 C6 06 7E 89 01 C6 06   ......{....~....
(26E0h)  52 90 FF B0 00 50 BF 32 0C 0E 57 E8 7C FB 5D C3   R....P.2..W.|.].
(26F0h)  55 89 E5 31 C0 9A 30 05 7D 0E A0 CB 8A 30 E4 48   U..1..0.}....0.H
```

Listing 1

```
   1            CPU             '8052.def'
   2

Zbior: "const.inc"
Zbior: "bios.inc"

Zbior: "port8255.inc"
   1            ;***********************************************
   2            ;Deklaracja adresow portow ukladu 8255
   3            ;***********************************************
   4
   5  A000      IO_PA       equ     A000h
   6  A001      IO_PB       equ     A001h
   7  A002      IO_PC       equ     A002h
   8  A003      IO_CTRL     equ     A003h
   9

Zbior: "dump.s03"
   5
   6            ;Zdefiniowane znaki sterujace drukarka
   7  000D      CR          equ     13          ;Carriage Return - znak powrotu glowicy
   8  000A      LF          equ     10          ;Line Feed - znak przesuwu linii na nastepna
   9  000C      FF          equ     12          ;Form Feed - wysuniecie strony z drukarki
  10
  11  8000          org     8000h
  12  8000 028200       ljmp    START
  13
  14

Zbior: "printer.inc"
   1            ;*************************
   2            ;Procedury obslugi drukarki
   3            ;*************************
   4
   5            ;**********
   6            ;* PRNACC *  Wysyla znak z akumulatora na drukarke
   7            ;**********
```

```
 8  8003 C083      prnAcc:   push    DPH
 9  8005 C082                push    DPL
10  8007 C0E0                push    Acc
11  8009 90A000    chkprn:   mov     DPTR,#IO_PA                ;odczyta stanu drukarki
12  800C E0                  movx    A, DPTR
13  800D 540F                anl     A,#0Fh
14  800F 6409                xrl     A,#1001b          ;czy drukarka gotowa ?
15  8011 6017                jz      prnok                      ;tak to drukuj
16  8013          prnerror:
17  8013 120274              lcall   CLS
18  8016 757879              mov     DL1,#_E
19  8019 757950              mov     DL2,#_r
20  801C 757A50              mov     DL3,#_r
21  801F 1202C5              lcall   CONIN                      ;czekanie na dowolny klawisz
22  8022 D3                  setb    C                          ;ustawienie znacznika bledu (C=1)
23  8023 D0E0                pop     Acc
24  8025 D082      exit:     pop     DPL                        ;i zakonczenie drukowania
25  8027 D083                pop     DPH
26  8029 22                  ret
27  802A D0E0      prnok:    pop     Acc                        ;od tej instrukcji gdy OK !
28  802C 90A001              mov     DPTR,#IO_PB
29  802F F0                  movx     DPTR,A                    ;wyslanie znaku do drukarki
30  8030 90A000              mov     DPTR,#IO_PA                ;odczyt stanu drukarki
31  8033 E0        wait:     movx    A, DPTR
32  8034 5480                anl     A,#80h                     ;czy mozna wyslac nastepny znak ?
33  8036 60FB                jz      wait                       ;nie to czekaj
34  8038 C3                  clr     C                          ;zakoczenie drukowania z info OK !
35  8039 80EA                sjmp    exit
36
37              ;**********
38              ;* PRNTXT *  Wysyla tekst ASCIIZ na drukarke (dane: mov DPTR,#tekst)
39              ;**********
40  803B            RNTXT:
41  803B E0                  movx    A, DPTR                    ;pobranie znaku z bufora
42  803C B40001              cjne    A,#0,ok1   ;czy znak konca tekstu ?
43  803F 22                  ret                                ;tak to koniec procedury
44  8040 128003      k1:     lcall   PRNACC                     ;nie to wydrukuj znak
45  8043 5001                jnc     ok2                        ;czy blad drukowania ?
46  8045 22                  ret                                ;tak to zakoncz procedure
47  8046 A3        k2:       inc     DPTR                       ;nie to nastepny znak
48  8047 80F2                sjmp    prntxt
49  8049 22                  ret
50
51              ;************
52              ;* PRNINIT *  Inicjuje port 8255 na potrzeby drukowania
53              ;************
54  804A            PRNINIT:
55  804A 90A003              mov     DPTR,#IO_CTRL
56  804D 7495                mov     A,#95h                     ;inicjacja rejeswtru kontrolnego 8255
57  804F F0                  movx     DPTR,A
58  8050 7405                mov     A,#05h                     ;inicjacja przerzutnika INTRB 8255
59  8052 F0                  movx     DPTR,A
60  8053 90A002              mov     DPTR,#IO_PC
61  8056 7450                mov     A,#01010000b               ;SELIN=1, RESET=0 (inicjacja), AUTOFD=1
62  8058 F0                  movx     DPTR,A
63  8059 7470                mov     A,#01110000b               ;SELIN=1, RESET=1 (praca), AUTOFD=1
64  805B F0                  movx     DPTR,A
65  805C 22                  ret
```

Zbior: “dump.s03”

```
16
17              ;*********************************************************
18              ;Procedura drukowania zawartość pamięci w hexa  PRNDUMP *
19              ;*********************************************************
20  805D            PRNDUMP:
21  805D 120274              lcall   CLS                        ;wyczysz wyswietlacz
22  8060 75785E              mov     DL1, #_d   ;literka “d” na DL1
23  8063 75F005              mov     B,#5                       ;parametr procedury GETDPTR
24  8066 1203B9              lcall   GETDPTR                    ;wczytaj poczatek obszaru do drukowania
25  8069 A983                mov     R1,DPH                     ;i zapamietaj w R1.R2
26  806B AA82                mov     R2,DPL
27  806D 75F005              mov     B,#5
28  8070 1203B9              lcall   GETDPTR                    ;wczytaj koniec obszaru
29  8073 AB83                mov     R3,DPH                     ;i zapamietaj w R3.R4
30  8075 AC82                mov     R4,DPL
31
32  8077 740D                mov     A,#CR
33  8079 128003              lcall   PRNACC                     ;powrot glowicy do lewego marginesu
34  807C 908130              mov     DPTR,#nagl
35  807F 12803B              lcall   PRNTXT                     ;wydrukowanie naglowka
36
37  8082 8983                mov     DPH,R1                     ;poczatek obszaru do DPTR
38  8084 8A82                mov     DPL,R2
39  8086          astrek:                     ;kolejny rekord 16 bajtow
```

```
  40  8086 7428              mov     A,#'('
  41  8088 128003            lcall   PRNACC
  42  808B E583              mov     A,DPH
  43  808D 120235            lcall   HEXASCII
  44  8090 128003            lcall   PRNACC
  45  8093 E5F0              mov     A,B
  46  8095 128003            lcall   PRNACC
  47  8098 E582              mov     A,DPL
  48  809A 120235            lcall   HEXASCII
  49  809D 128003            lcall   PRNACC
  50  80A0 E5F0              mov     A,B
  51  80A2 128003            lcall   PRNACC
  52  80A5 7468              mov     A,#'h'
  53  80A7 128003            lcall   PRNACC
  54  80AA 7429              mov     A,#')'
  55  80AC 128003            lcall   PRNACC
  56  80AF 12811D            lcall   prnspc                  ;wydrukowanie znaku |
  57
  58  80B2 7D10              mov     R5,#16                  ;16 bajtow do wydrukowania
  59  80B4 E0      nastb:    movx    A, DPTR                 ;w postaci np.:
  60  80B5 120235            lcall   HEXASCII  ;12 45 3A 5B 74 09 BC 5A 4F 1E 12 42 54 76
;DC BA
  61  80B8 128003            lcall   PRNACC
  62  80BB E5F0              mov     A,B
  63  80BD 128003            lcall   PRNACC                  ;dana w postaci XX
  64  80C0 7420              mov     A,#20h                  ;spacja
  65  80C2 128003            lcall   PRNACC
  66  80C5 A3                inc     DPTR
  67  80C6 E583              mov     A,DPH                   ;sprawdzenie czy
  68  80C8 8BF0              mov     B,R3                    ;aby nie koniec adresu
  69  80CA B5F009            cjne    A,B,ok3                 ;do drukowania
  70  80CD E582              mov     A,DPL
  71  80CF 8CF0              mov     B,R4
  72  80D1 B5F002            cjne    A,B,ok3
  73  80D4 8002              sjmp    asciz                   ;to gdy koniec rekordu 16 bajtow
  74  80D6 DDDC      ok3:    djnz    R5,nastb   ;to gdy nie - skok na nastepny bajt
  75
  76  80D8 12811D    asciz:  lcall   prnspc
  77  80DB 8983              mov     DPH,R1                  ; ..-*...%....3.a.
  78  80DD 8A82              mov     DPL,R2
  79  80DF 7D10              mov     R5,#16                  ;16 bajtow do wydrukowania
  80  80E1 E0      nastc:    movx    A, DPTR
  81  80E2 24E0              add     A,#0E0h                 ;sprawdzenie czy znak jest drukowalny
  82  80E4 4004              jc      znakok                  ;tzn. czy jego kod jest > 13
  83  80E6 742E              mov     A,#'.'                  ;jezeli nie to drukuj kropke zamiast znaku
  84  80E8 8001              sjmp    druk
  85  80EA E0      znakok:   movx    A, DPTR
  86  80EB 128003    druk:   lcall   PRNACC                  ;jezeli tak to drukuj znak
```

```
 87  80EE A3                 inc     DPTR
 88  80EF E583               mov     A,DPH
 89  80F1 8BF0               mov     B,R3
 90  80F3 B5F009             cjne    A,B,ok4         ;sprawdzenie czy aby nie koniec adresu
 91  80F6 E582               mov     A,DPL
 92  80F8 8CF0               mov     B,R4
 93  80FA B5F002             cjne    A,B,ok4
 94  80FD 8013               sjmp    finix
 95  80FF DDE0     ok4:      djnz    R5,nastc
 96  8101 740D               mov     A,#CR
 97  8103 128003             lcall   PRNACC
 98  8106 740A               mov     A,#LF
 99  8108 128003             lcall   PRNACC          ;wydrukowanie konca linii
100  810B A983               mov     R1,DPH
101  810D AA82               mov     R2,DPL
102  810F 028086             ljmp    nastrek
103  8112 740D     finix:    mov     A,#CR
104  8114 128003             lcall   PRNACC
105  8117 740A               mov     A,#LF
106  8119 128003             lcall   PRNACC          ;koniec linii
107  811C 22                 ret
108  811D C083     prnspc: push      DPH
109  811F C082               push    DPL
110  8121 90812C             mov     DPTR,#space
111  8124 12803B             lcall   PRNTXT
112  8127 D082               pop     DPL
113  8129 D083               pop     DPH
114  812B 22                 ret
115  812C 207C2000 space     db      20h,'|',20h,0
116  8130 20416472
     8134 65732020
     8138 20202020
     813C 20202020
     8140 20202020
     8144 20202020
     8148 20204461
     814C 6E652048
     8150 6578      nagl     db      ' Adres         Dane Hex'
117  8152 20202020
     8156 20202020
     815A 20202020
     815E 20202020
     8162 20202020
     8166 20202020
     816A 20202020
     816E 2044616E
     8172 65204153
     8176 4349490D
     817A 0A                 db      '               Dane ASCII',13,10
118  817B 2D2D2D2D
     817F 2D2D2D2D
     8183 2D2D2D2D
     8187 2D2D2D2D
     818B 2D2D2D2D
     818F 2D2D2D2D
     8193 2D2D2D2D
     8197 2D2D2D2D
     819B 2D2D2D2D
     819F 2D2D               db      '-----------------------------------------'
119  81A1 2D2D2D2D
     81A5 2D2D2D2D
     81A9 2D2D2D2D
     81AD 2D2D2D2D
     81B1 2D2D2D2D
     81B5 2D2D2D2D
     81B9 2D2D2D2D
     81BD 2D2D2D2D
     81C1 2D2D2D2D
     81C5 2D2D2D0D
     81C9 0A00               db      '-----------------------------------------',13,10,0
120
121             ;*********************************************************
122  8200         org        8200h                   ;tak dla czytelnosci
123             ;*********************************************************
124  8200       START:
125  8200 12804A             lcall   PRNINIT         ;zainicjuj 8255 na potrzeby drukowania
126  8203 12805D             lcall   PRNDUMP         ;wywolanie procedury
127  8206 80FE     stop:     sjmp    stop            ;wcisnij klawisz M(onitor)
128
129  8208                    END

 Kompilacja zakonczona pomyslnie !
 Zbior: "dump.s03" , 467 bajt(ow), 0.2 sekund(y).
```

Większość komentarzy zawarłem po średniku w listingu programu. Po włączeniu do kompilacji zbioru "PORT8255.INC" zawierającego deklaracje portów układu 8255 rozpoczyna się właściwy kod programu. Deklaracja

```
ORG    8000h
```

powinna być w razie potrzeby zmieniona jeżeli przewidujesz umieszczenie kodu programu w innym obszarze RAM komputerka.

Po tej deklaracji dołączam zbiór PRINTER.INC zawierający procedury zdefiniowane i omówione w poprzednim odcinku klasy mikroprocesorowej czyli

PRNINIT – procedura inicjacji układu 8255 na potrzeby drukowania
PRNACC – procedura wysłania znaku z akumulatora na drukarkę
PRNTXT – procedura wysłania na drukarkę tekstu zakończonego znakiem "0" (zero)

Kolejne linie to już procedura drukowania danych z RAM'u w postaci przedstawionej wcześniej – czyli PRNDUMP. Rozpoczyna się ona zapaleniem na DL1 literki "d" w celu zaproszenia do wpisania z klawiatury komputerka adresu początkowego i końcowego obszaru który ma być wydrukowany. Dane te są zapamiętywane w rejestrach R1.R2 i R3.R4 odpowiednio start i koniec obszaru.

Dalej w liniach 34 i 35 drukowany jest nagłówek czyli tekst spod etykiety "nagl":

Potem to już kolejne linie zawierające adres początkowy (linie 40...56) rekordu 16 bajtów danych, następnie 16 bajtów z RAM w postaci HEX oddzielonych spacjami (linie 58...74), wreszcie te same dane wydrukowane zostają w postaci znaków ASCII (linie: 76...95) i zakończone znakiem końca linii CRLF (linie: 96...99). Dodatkowo w liniach 81 i 82 sprawdzany jest kod znaku. Jeżeli kod jest < 14 to znak należy do tzw. znaków sterujących drukarki i nie powinien być drukowany, a zamiast niego wstawiana jest w linii 83 kropka. Reszta znaków o kodach 14...255 jest drukowana normalnie. W tym miejscu jeżeli ktoś chce zawęzić bardziej zbiór drukowanych znaków może w linii 81 wstawić zamiast wartości 0E0h większą np. 20h, wtedy będą drukowane znaki ASCII od spacji w górę.

Reszta programu to po prostu inicjacja drukarki (linia 125) oraz sprawdzenie działania procedury drukowania obszaru (linia 126).

Na kolejne popołudniowe zimowe wieczory proponuję następujące zadania:

Dokonać poprawek w programie przedstawionym w dzisiejszym artykule, a dotyczącym odbioru danych z SA9203 i umieszczania ich w buforze o długości np. 128 bajtów, umieszczonym w wewn. RAM procesora 80C52 (adresy: 80h...FFh). Wykorzystać rejestr R1 jako wskaźnik początku bufora (podpowiedź - @R1)

Zapisać sekwencję instrukcji programującej układ SA9203 w postaci:
- PA jako wejście z potwierdzeniem
- PB jako wyjście – adresowany bitowo
- PC jako wyjście – adresowany bajtowo
- PD...PF jako wejścia adresowane bitowo z aktywnymi rezystorami podciągającymi

Rozwiązania zadań w kolejnym numerze EdW.

Z okazji zbliżających się świąt, wszystkim entuzjastom techniki mikroprocesorowej autor cyklu składa najserdeczniejsze życzenia dalszych sukcesów, korzyści a przede wszystkim satysfakcji z ujarzmiania tych arcyciekawych elementów elektronicznych, do których z pewnością należy przyszłość współczesnej elektroniki. Do zobaczenia w Nowym Roku 1999!

Sławomir Surowiński

```
Adres          Dana Hex          Dana ASCII
-------------------------------------------------------------------------
```