Mikrokontrolery? To takie proste...

Część 4
Mikrokontroler 8051 – opis układu

Kontynuujemy opis mikrokontrolera 8051. W tym odcinku powiemy w jaki sposób można dołączyć zewnętrzną pamięć pogramu oraz danych. Podane konkretne rozwiązania sprzętowe poparte schematami elektrycznymi, z pewnością dadzą Ci pojęcie na temat kontruowania podstawowych bloków funkcjonalnych małego systemu mikroprocesorowego. Opis takiego układu będącego jednocześnie bazą do naszej przyszłej nauki programowania 8051 prezentujemy w niniejszym numerze EdW, warto jednak abyś najpierw zapoznał się z niniejszym artykułem.

Zewnętrzna pamięć programu

Jak się dowiedziałeś z wcześniejszych części naszego cyklu, procesor 8051 i mu podobne mają możliwość dołączenia dodatkowej zewnętrznej pamięci programu, np. typu EPROM. W pamięci tej podobnie jak w wewnętrznej pamięci programu zaszytej w kostce 8751 (8752) programista umieszcza poszczególne rozkazy programu , używając do tego celu np. programatora pamięci EPROM lub programatora do programowania mikroprocesorów z rodziny MCS-51. Wiesz już że procesor może pracować w następujących konfiguracjach:

a) tylko z wewnętrzną pamięcią programu (dla kostek 8751, 8752)
b) z wewnętrzna i zewnętrzną pamięcią programu jednocześnie, w tym przypadku zewnętrzna pamięć programu stanowi jak gdyby „przedłużenie" pamięci wewnętrznej.
c) oraz tylko z zewnętrzną pamięcią programu np. typu EPROM.

Pierwszy tryb jest bardzo wygodny, pozwala na „pełne" wykorzystanie wszystkich zalet mikrokontrolera jednoukładowego w całym tego słowa znaczeniu. Programista wykorzystując procesor w wersji z wbudowaną pamięcią programu (8751,8951, xx52) cały kod swego programu umieszcza wewnątrz kości, dzięki czemu ma do dyspozycji wszystkie porty mikrokontrolera – w tym także P0 i P2. (patrz poprzednie odcinki naszego cyklu).

Kostka 87C51 (89C51) ma „tylko" 4kB (8x52 8kB pamięci programu), najczęściej te „tylko" w zupełności wystarcza, lecz zdarza się że jest to za mało, wtedy konieczne jest dołączenie dodatkowej zewnętrznej pamięci programu w postaci kostki EPROM. W tym przypadku (jak wspomniałem w poprzednim odcinku) maksymalna długość programu równa wielkości obu pamięci zewnętrznej jak i wewnętrznej nie może przekroczyć 64kB – czyli 65536 8-bitowych słów.

Czyli że np. do kostki 87C52 (zawierającej 8 kB wewnętrznej EPROM) można dołączyć maksymalnie 56kB EPROM z zewnątrz.

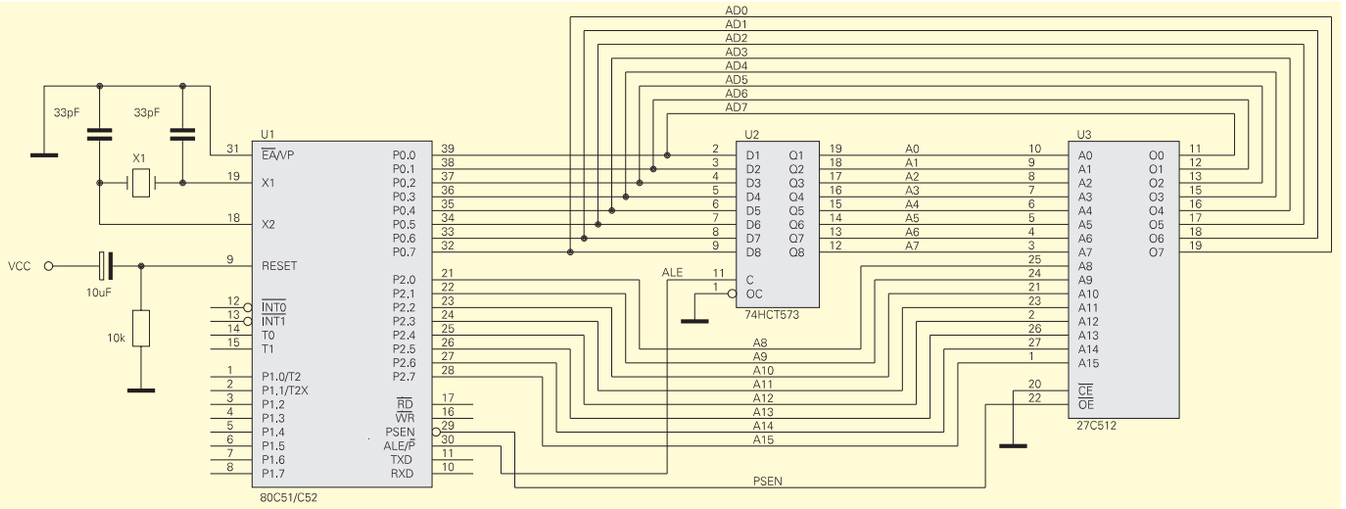
Trzeci tryb pracy tylko z zewnętrzną pamięcią programu jak pamiętasz z pierwszej części cyklu wymaga zwarcia wyprowadzenia /EA (pin 31 procesora 8051/52) do masy. Parę lat temu, kiedy procesory '51 z wbudowaną wewnętrzną pamięcią programu kosztowały niemało, natomiast kostki bez niej można było nabyć za kilkadziesiąt tysięcy (starych złotych) nazywałem ten tryb pracy „oszczędnościowym" (chyba tylko dla własnej kieszeni). Jak się wkrótce przekonasz drogi Czytelniku pierwsze praktyczne kroki z wykorzystaniem twego pierwszego systemu mikroprocesorowego warto będzie poczynić korzystając właśnie z tego trybu.

Tak jak przy nauce jazdy samochodem istnieje większe ryzyko stłuczki, tak w przypadku mikroprocesora nie warto na początku ryzykować uszkodzeniem kostki z zapisanym w niej programem, lepiej jest kod programu umieścić w zewnętrznej pamięci EPROM.

A propos „pierwszego systemu z 8051" – w bieżącym numerze EdW znajdziesz opis kitu zawierającego wszystkie niezbędne elementy do zbudowania małego lecz w pełni funkcjonalnego systemiku do nauki programowania i obsługi procesorów rodziny MCS-51. Publikacja tego systemu jest celowa i wyprzedza nieco rozpoczęcie nauki programowania. Wszystko po to abyś drogi Czytelniku miał trochę czasu na nabycie tego zestawu, zapoznanie się z nim, zmontowanie oraz wstępne sprawdzenie poprawnego działania. Wtedy wraz z pierwszym artykułem dotyczącym programowania 8051 będziesz mógł rozpocząć praktyczne lekcje z mikrokontrolerem. Początek kursu planowany jest w październiku.

Wracajmy jednak do naszego artykułu. Pozostając wiernym zasadzie przekazywania Tobie, tylko niezbędnych, praktycznych informacji na temat '51-ki nie będę się rozwodził nad teorią obsługi zewnętrznej pamięci programu przez sam procesor, a jedynie przedstawię Ci niezawodny sposób dołączenia jej do układu mikrokontrolera.

Rysunek 5 przedstawia schemat elektryczny takiego połączenia. W układzie tym



Rys. 5. Sposób dołączenia zewnętrznej pamięci programu do procesora

zrezygnowano z użycia wewnętrznej pamięci programu – wykorzystano więc kostkę w wersji bez ROM (8051/52), pin /EA zwarto do masy, w jako zewnętrzną pamięć programu wykorzystano kostkę EPROM o pojemności 64kB typu 27C512. Oczywiście w praktyce nie jest konieczne używanie tak pojemnej pamięci. Do naszych celów i eksperymentów wystarczy pamięć 27C64 o pojemności 8kB. W takim przypadku linie adresowe A15, A14, A13 (piny 28, 27 i 26 portu P2) pozostaną niedołączone, lecz niestety i tak będą nie do wykorzystania, bowiem w trybie adresowania zewnętrznej pamięci programu (jak i danych) cały port P2 przekazuje starszy bajt 16-bitowego adresu.

Zgodnie z opisem z I części artykułu, w układzie wykorzystano dodatkowy rejestr w postaci układu 74HCT573, którego zadaniem jest zatrzaśnięcie młodszej części 16-bitowego adresu, czyli sygnałów A0...A7. W czasie trwania dodatniego poziomu sygnału na wyjściu ALE, mikroprocesor wystawia: na piny portu P2 starsza część 16-bitowego adresu A8...A15, natomiast na port P0 wystawiona zostaje młodsza część adresu A0...A7, która zostaje od razu przekazana poprzez 74HCT573 na wejścia adresowe A0...A7 pamięci EPROM. Dzieje się tak dlatego że układ '573 aktywowany jest poziomem, toteż w przypadku gdy na jego wejściu C (połączonym z ALE) panuje stan wysoki, rejestr ten jest „przezroczysty" tzn. że dane pojawiające się na jego wejściach D1...D8 natychmiast pojawiają się na wyjściach Q1...Q8. Dołączenie wejścia /OC układu U2 do masy powoduje odblokowanie na stałe wyjść Q1...Q8

(gdy /OC=1 to wyjścia te przechodzą w stan wysokiej impedancji – lecz w naszym przypadku nie ma to praktycznego zastosowania).

Następnie podczas opadającego zbocza sygnału ALE młodsza cześć adresu zostaje „zapamiętana" w U2, a procesor może wtedy z portu P0 odczytać kolejny bajt z pamięci EPROM podając stan niski na wyprowadzenie /PSEN, które jak widać ze schematu jest dołączone do wejścia /OE pamięci.

Wejście wyboru pamięci U3 /CE jest na stałe zwarte do masy, co oznacza że pamięć EPROM jest ciągle aktywna a do jej odczytu wystarczy niski poziom podany na wejście /OE.

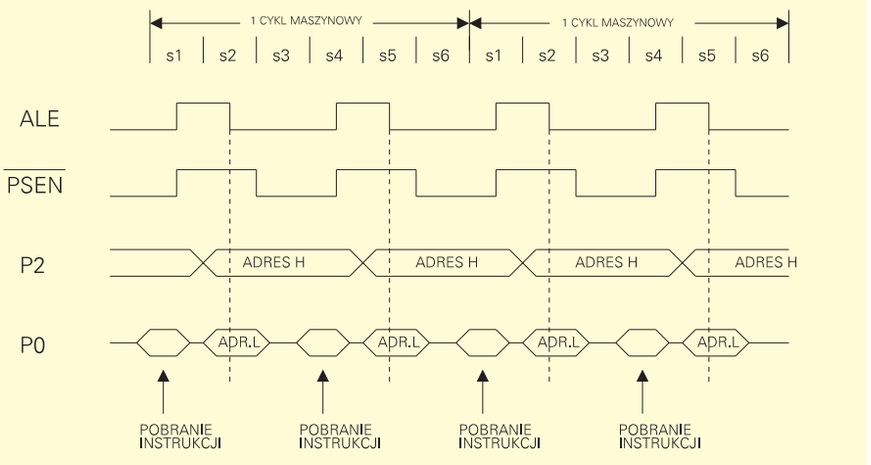
Czasem w różnych aplikacjach szczególnie w urządzeniach zasilanych bateryjnie wejście /OC układu U2 oraz wejście wyboru pamięci EPROM /CE są razem dołączone do wyjścia sygnału ALE. Nie jest to błędem, bowiem podczas cyklu odczytu z zewnętrznej pamięci tak programu jak i danych wszystkie operacje tak odczytu jak i zapisu do tych pamięci odbywają się podczas stanu niskiego na wyjściu ALE procesora.

Cykl odczytu z zewnętrznej pamięci programu możesz dodatkowo prześledzić na **rysunku 6**.

Podsumowując te akapit powinieneś zapamiętać, że

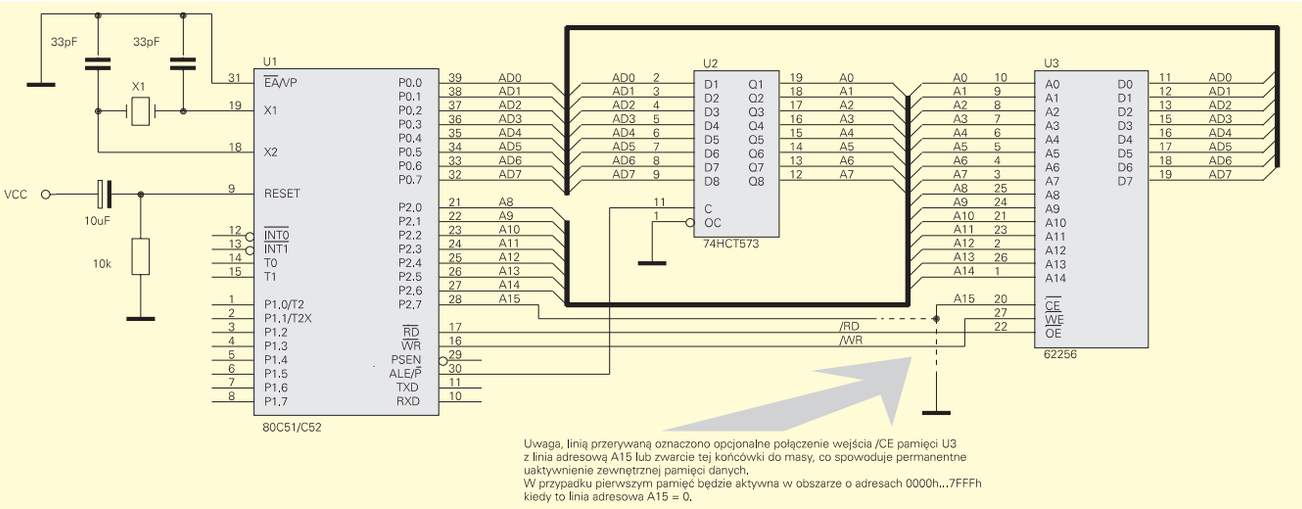
– przestawiony sposób dołączenia zewnętrznej pamięci programu jest typowym dla tej konfiguracji pracy procesora
– odczyt kolejnych bajtów (rozkazów) z zewnętrznej EPROM odbywa się za pośrednictwem sygnału /PSEN, o którym pisałem w I części naszego cyklu przy okazji omawiania wyprowadzeń procesora
– dodatkowo w układzie z zewnętrzną pamięcią programu (jak się zaraz okaże także i danych) niezbędny jest dodatkowy układ scalony w postaci 8-krotnego rejestru równoległego 74HCT573. Można użyć wersji LS. Natomiast kostki 74HCT373 (LS373), które maja taka samą strukturę wewnętrzną, różnią się od układu 573 rozmieszczeniem wyprowadzeń D1...D8 i Q1...Q8.

Zainteresowanych w/w układami odsyłam do katalogów serii TTL. My w naszych rozwiązaniach będziemy nagminnie korzystać z układów '573 ze względu na bardziej korzystny rozkład jego wyprowadzeń wprost dostosowany do użycia



ADRES H - starsza część adresu A8...A15
ADR.L - młodsza część adresu A0...A7

Rys. 6. Cykl odczytu z zewnętrznej pamięci programu



Rys. 7. Sposób dołączenia zewnętrznej pamięci danych 32kB do procesora 8051

wraz z poczciwym 8051 oraz pochodnymi w obudowach DIL-40.

Zewnętrzna pamięć danych

W podobny sposób jak w przypadku zewnętrznej pamięci programu, do większości procesorów serii MCS-51 (w tym 8051/52) dołącza się zewnętrzną pamięć danych. Do tego celu używa się zazwyczaj (a prawie zawsze) kostek pamięci statycznych nazywanych często SRAM od skrótu: „Static RAM", czyli pamięć statyczna. Pamięć taka powinna charakteryzować się 8-bitową organizacją danych oraz równoległym adresowaniem, tak jak w przypadku zwykłych EPROM ów. Najpopularniejsze i spotykane w handlu, a przy tym najlepiej nadające się do naszych potrzeb kości pamięci SRAM to:

a) 6116 – pamięć SRAM o pojemności 2kB (6116 – 16kbitów, czyli 16384 bitów / 8 = 2kB = 2048 B).
b) 6264 – j/w lecz o pojemności 8kB (6264 – 64kbity, czyli 65536 / 8 = 8kB = 8192 B)
c) 62256 – j/w lecz o pojemności 32kB (62256 – 256 kbitów, czyli 262144 / 8 = 32kB...)

Cena układów wymienionych pamięci jest obecnie znikoma, nie znaczy to jednak że warto stosować pamięci z zapasem (np. największe 32kB), zawsze warto przewidzieć wielkość praktycznie wykorzystywanej ilości komórek pamięci i odpowiednio dobrać wymagany typ pamięci.

Pierwszy wymienionych układ SRAM 6116 spotykany jest w handlu w typowej obudowie 24-końcówkowej o rozstawie 600 mils, dwa ostatnie natomiast w obudowach 28-pinowych (600 mils, spotykane tez są wersje w „wąskich" obudowach o rozstawie 300 mils).

Do kontrolerów serii '51 można dołączyć maksymalnie 64kB zewnętrznej pamięci danych. Można więc w takim (skrajnym) przypadku zastosować:

– 2 kostki 62256 (2x 32kB = 64kB) lub
– 8 kostek 6264 (8x 8kB = 64kB) lub w ostateczności
– 32 kostki 6116 (32x 2kB = 64kB).

Oczywiście dwie ostatnie sytuacje w przypadku wymaganej maksymalnej pojemności zewnętrznej SRAM są niepraktyczne, po pierwsze ze względu na ilość układów scalonych co prowadzi do zwiększenia płytki drukowanej, po drugie pojawia się konieczność stosowania dodatkowego dekodera adresu w postaci np. układu serii TTL-LS typ 74LS138 – dekoder 1 z 8.

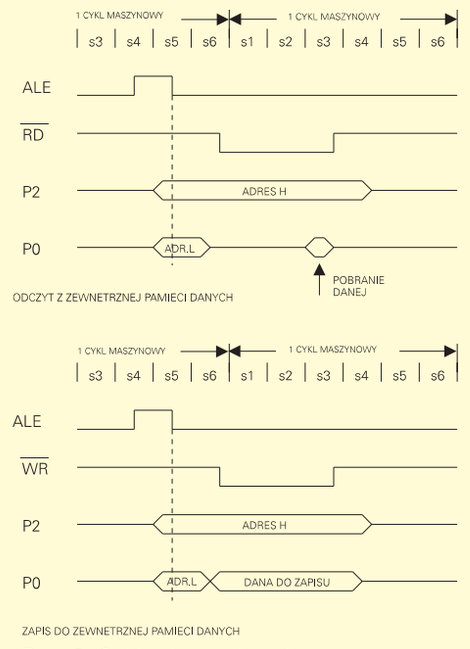
Rysunek 7 pokazuje podstawowy układ dołączenia zewnętrznej kostki SRAM o pojemności 32kB do mikrokontrolera 8051. Jak widać układ połączeń jest bardzo zbliżony do schematu z rysunku 5, gdzie omawialiśmy praktyczny sposób dołączenia zewnętrznej pamięci programu typu EPROM. Dla zwiększenia czytelności rysunku rezygnujemy z rysowania każdego z połączeń magistrali adresowej (A0...A15) oraz danych (AD0...AD7) oddzielnie, w zamian zastąpimy je wspólnym „fachowym" symbolem „szyny" – w postaci pogrubionej kreski. Każde połączenie „odchodzące" od takiej magistrali ma swoją nazwę (etykietę), czyli że końcówki układów scalonych (w tym przypadku: procesora U1, pamięci U3 oraz zatrzasku U2) przy których znajduje się taka sama etykieta literowa, są ze sobą elektrycznie połączone. Prawda, że prostsze?!

Przyzwyczajmy się zatem do takiego sposobu czytania i rysowania schematów, szczególnie jeżeli przedstawiony schemat zawiera wielokońcówkowe układy scalone takie jak mikroprocesor. Wracajmy jednak do tematu.

Do adresowania pamięci wykorzystuje się, podobnie jak do pamięci programu, te same linie adresowe: A0...A15, oraz danych: AD0...AD7. Jednak sterowanie odczytem zewnętrznej pamięci danych zajmuje się sygnał /RD (końcówka 17 – U1) od skrótu „Read" – przypomnij sobie część I naszego cyklu). Podanie odpowiednich stanów logicznych na końcówki adresowe pamięci (U3) odpowiadające adresowi komórki pamięci wewnątrz układu, a następnie podanie stanu niskiego na końcówkę /RD spowoduje pojawienie się danej (z pamięci) na wyprowadzeniach magistrali AD0...AD7 i w konsekwencji odczyt jej przez mikroprocesor, za pośrednictwem portu P0.

Sytuację te w postaci przebiegów przedstawia **rysunek 8**. Nie podaję tu czasów poszczególnych cykli, po pierwsze z czysto prakt]ycznego powodu, po drugie ze względu że zależą one od częstotliwości zegara (rezonatora kwarcowego dołączonego do procesora).

Skoro wiemy jak praktycznie (sprzętowo) realizowany jest odczyt z zewnętrznej



Rys. 8. Cykl odczytu i zapisu z zewnętrznej pamięci danych

pamięci danych, to warto powiedzieć jak dokonywany jest zapis. Otóż do tego celu służy kolejny sygnał procesora nazywany /WR (od skrótu „write"- zapis), a wystawiany przez procesor na końcówkę 16 (U1) w czasie tej operacji. Podobnie ja w przypadku odczytu, najpierw procesor ustala adres na końcówkach A0...A15 (port P2 oraz wyjścia zatrzasku U2), następnie podaje daną na port P0, po czym wystawia stan niski na pin /WR, co powoduje zapis danej w pamięci SRAM.

Jak już wcześniej wspomniałem, zastosowanie zewnętrznej pamięci tak programu jak i danych wiąże się ze zubożeniem typowej „jednoukładowości" mikrokontrolera, bowiem zajęte zostają portu P0 i P2 procesora. O ile w przypadku pracy z zewnętrzną pamięcią programu dzieje się tak zawsze: port P0 pracuje jako multipleksowana (na zmianę) szyna danych lub młodszej części adresu, a port P2 wystawia starsze 8 bitów 16-bitowego adresu, o tyle w przypadku pracy z zewnętrzną pamięcią danych (bez zewnętrznej pamięci programu) istnieje możliwość oszczędniejszego gospodarowania portami procesora.

W przykładzie opisanym powyżej mówiliśmy , że procesor przy odczycie lub zapisie „....wystawia 16-bitowy adres (A0...A15) ... . Do tego potrzebne są 2 porty, które w takim przypadku nie nadają się do dodatkowego wykorzystania, jako np. wyjścia sterowania przekaźnikami, lub czymkolwiek innym.

Istnieje jednak możliwość innego odczytu zewnętrznej pamięci danych, nazywana „stronicowaniem". Ma ona zastosowanie szczególnie wtedy, kiedy zewn. pamięć danych mam mniejszy rozmiar od maksymalnej przestrzeni adresowej procesora np. 2kB. Jak widać z **rysunku 9** pamięć taka (U3) ma tylko 11 linii adresowych (A0...A10), co pozwala na zaadresowanie 2048 komórek pamięci (bajtów). Tak więc pozostałe linie adresowe procesora A11...A15 pozostałyby niewykorzystane gdyby zastosować odczyt jak w poprzednim przypadku z pełnym adresem A0...A15.

Przy „stonicowanym „ sposobie obsługi zewn. pamięci danych, procesor wystawia tylko młodszą część 16-bitowego adresu (linie AD0...AD7), zaś port P2 pozostaje „nietknięty". Wnikliwy czytelnik zauważy że w konsekwencji takiego sposobu obsługi możliwe będzie zaadresowanie tylko 256 bajtów (2 do potęgi 8 = 256) tej pamięci, a nie jak w naszym przykładzie aż 2kB. No tak, chyba że przed odczytem przez procesor, sami, za pomocą sygnałów A8...A10 (wystawianych poprzez 3 piny portu P0) ustawimy niejako „fizyczny" adres 256 bajtowej strony adresowanej pamięci.

Zauważmy przecież że za pomocą tych trzech końcówek można „zaadresować" 8 stron po 256 bajtów każda co w sumie da nam do dyspozycji pełne 2048 bajtów, czyli 2kB. Zauważmy też że, co najważniejsze, pozostałe końcówki portu P2 pozostają wolne i możemy je dowolnie wykorzystać jako wejścia lub wyjścia cyfrowe... Prawda, że genialne?!

No tak, ale kiedy ten procesor adresuje pamięć za pomocą pełnego adresu, a kiedy za pomocą stronicowania .....!. Otóż w celu rozróżnienia przedstawionych dwóch typów adresowania wprowadzone są dwie różne instrukcje procesora, których używa programista (w przyszłości Ty ! drogi Czytelniku) podczas projektowania układu i pisania programu, w zależności od potrzeb, ale o tym powiemy dokładnie przy okazji nauki programowania 8051.

„Miksowanie" przestrzeni adresowych programu i danych

Często w praktyce podczas konstruowania i oprogramowywania układów wykorzystujących zewnętrzne pamięci tak programu jak i danych zdarzają się następujące dwie sytuacje:

a) w zewnętrznej pamięci programu (EPROM – stałej) znajdują się, wprowadzone przez programistę na etapie tworzenia programu, stałe np. w postaci tablic (sinusów, logarytmów niezbędnych do obliczeń) lub cokolwiek innego, nie będącego typową „instrukcją" programu lub jej argumentem. Aż się prosi żeby te dane umieszczane były w zewnętrznej pamięci danych (w wewnętrznej RAM prawdopodobnie zabrakłoby na nie miejsca), czyli w układach SRAM. Nie ma sprawy, można przecież je przenieść (programowo) z EPROM do SRAM w czasie inicjacji mikroprocesora, tylko po co !. Czy nie lepiej „jakoś" połączyć obszar pamięci programu i danych, zachowując oczywiście odrębne obszary adresowe, pozostawiając tylko jednolity sposób odczytu tych danych?!

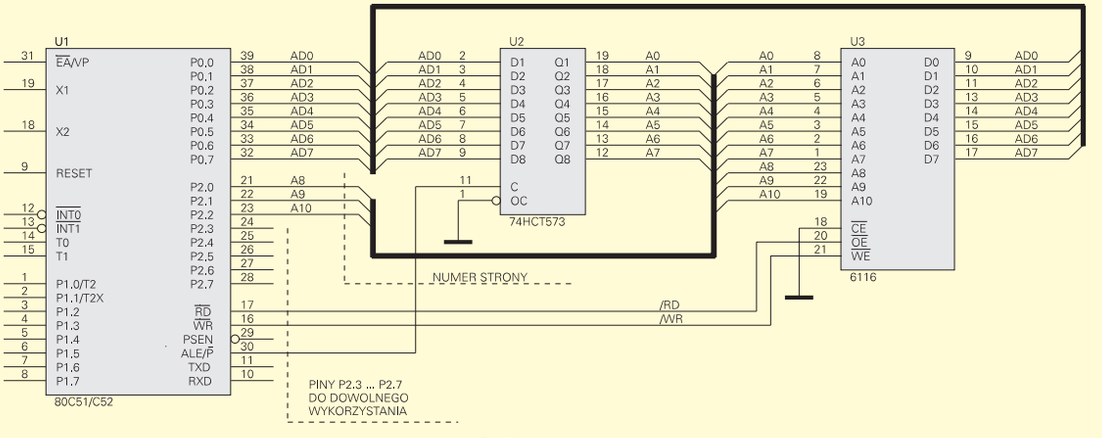
I druga sytuacja, bardziej praktyczna w fazie nauki sytuacja.

b) Zbudowałem uniwersalny mikrosterownik z 8051...., chciałbym mieć możliwość wpisywania na bieżąco (np. za pomocą komputera) moich programów i testowania ich w tym układzie bez potrzeby każdorazowego programowania pamięci EPROM za pomocą drogiego programatora, na który mnie zresztą nie stać ..... . Czy nie dało by się wykorzystać dołączoną pamięć danych SRAM jako część obszaru adresowego pamięci programu (i odwrotnie) i wpisywać do niej nowe programy bez wyjmowania układu scalonego z podstawki ?... .

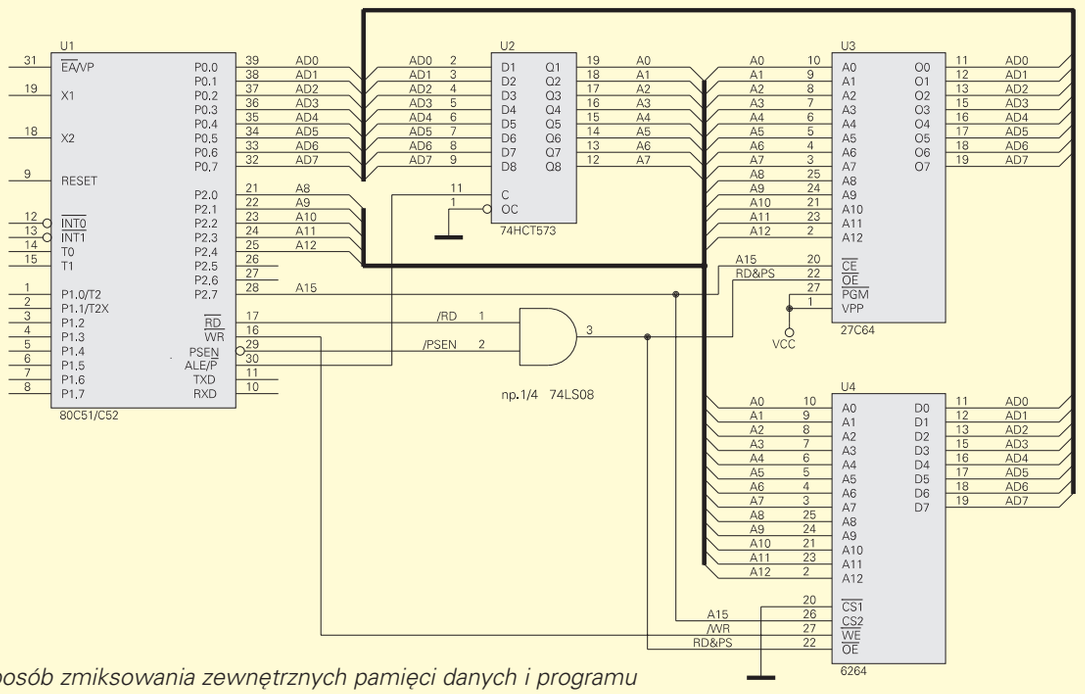
W przedstawionych dwóch przypadkach, rozwiązaniem problemu jest wykorzystanie do odczytu danych tak z pamięci programu jak i danych, iloczynu sygnałów:

– odczytu z zewnętrznej pamięci danych /RD oraz
– odczytu z zewnętrznej pamięci programu /PSEN

Iloczyn taki w praktyce najłatwiej jest wykonać za pomocą pojedynczej 2-wejściowej bramki AND, jak pokazano na **rysunku 10**. Przedstawiono nieco rozbudowaną konfigurację procesora pracującego



Rys. 9. Przykład adresowania zewnętrznej pamięci danych 2kB poprzez stronicowanie



Rys. 10. Sposób zmiksowania zewnętrznych pamięci danych i programu

z zewnętrzną pamięcią programu U3 (EPROM 8kB) oraz pamięci danych U4 (SRAM 8kB). Zauważmy jednak że odczyt z obu pamięci może odbywać się „tak samo". Dzięki wymnożeniu sygnałów /RD i /PSEN, procesor ma możliwość odczytu kolejnej instrukcji programu (za pomocą /PSEN) tak z pamięci U3 jak i U4, bowiem sygnał RD&PS (będący iloczynem /RD i /PSEN) steruje wejściami /OE odczytu obu pamięci (U3 i U4). Oczywiście sprawa zapisu nadal dotyczy tylko pamięci SRAM U4 (dołączony sygnał /WR procesora U1 do końcówki /WE pamięci U4).

W celu rozdzielenia obszarów adresowych obu pamięci (tak aby nie następował konflikt przy odczycie na magistrali AD0...AD7, co może mieć miejsce w przypadku jednoczesnego pojawienia się danej na wyjściach 11...19 układu U3 i U4), wykorzystano linię adresową A15. Dzięki niej pamięć EPROM (U3) będzie aktywna kiedy sygnał A15 będzie miał stan niski, co jest adekwatne do obszaru adresowania równemu: 0000h.....7FFFh (oczywiście wtedy pamięć U4 jest nieaktywna, bo sygnał wyboru CS2 – U4 ma stan linii A15 = niski).

Pamięć SRAM U4 będzie natomiast obsługiwana w obszarze adresowym dla A15 równego logicznemu „1", czyli w zakresie: 8000h...FFFFh.

Na zakończenie, krótki przykład w jaki sposób w takim układzie (z rys.10) realizowany jest przypadek b) omawiany wcześniej.

Otóż w pamięci EPROM U3 na stałe znajduje się program umożliwiający np. przesłanie danych (którymi mogą być także instrukcje programu) z komputera do naszego układu mikroprocesorowego, chociażby tego z rys.10. Program ten często zwany „monitorem" lub „biosem" wykonywany zostaje zaraz po włączeniu zasilania procesora, bowiem pamięć EPROM z rys.10 – układ U3 zaczyna się od adresu początkowego 0000h.

Nie jest na razie istotne w jaki sposób wspomniane dane zostają przesłane z komputera, (może być w tym celu użyty port szeregowy lub np. równoległy komputera).

Ważne jest że po przesłaniu mogą one być zapisane w zewnętrznej pamięci SRAM – U4, która przecież daje się zapisywać jak zwykłą pamięć danych dzięki sygnałowi /WR.

Kiedy już te dane, będące np. kawałkiem jakiegoś nowego, utworzonego przez Ciebie programu (powiedzmy zegarka z 256 alarmami i tyloma timerami) zostana wpisane do pamięci U4, możesz teraz rozkazać procesorowi (wydając odpowiednią instrukcję), „skoczyć" do adresu 8000h i stamtąd rozpocząć dalsze wykonywanie programu. Co będzie w efekcie ...?, w efekcie dzięki bramce AND (rys.10) jak powiedziałem wcześniej, dalej wykonywanym programem będzie ten przesłany wcześniej z komputera i umieszczony w SRAM, czyli np. twój super-zegarek (lub cokolwiek co w przyszłości stworzysz).

Zaletami takiego rozwiązania są:

– brak konieczności stosowania drogich programatorów EPROM
– bardzo szybkie ładowanie nowych programów do pamięci SRAM, co pozwoli uczącemu się programiście (takiemu jak Ty) na szybkie obserwowanie efektów swojej pracy
– niepotrzebne jest oczywiście kasowanie SRAM przed zapisem nowej wersji tworzonego programu

Nie brakuje i wad:

– po wyłączeniu zasilania program umieszczony w SRAM zostaje wymazany (ale dalej znajduje się np. w komputerze);
– konieczne jest „posiadanie" EPROM ze wspomnianym „biosem" (monitorem).

Jak się wkrótce przekonasz, drogi Czytelniku, na etapie nauki programowania i obsługi procesorów 8051 pierwsza „wada" w praktyce wcale nie jest wadą. O drugi drobiazg (zaprogramowany EPROM) nie musisz się martwić, będziesz mógł go otrzymać wraz zestawem dla początkujących programistów, który znajdziesz w tym numerze EdW.

Bardziej ambitnych muszę w tym miejscu uspokoić. Jestem pewien, że każdy kto ukończy nasz kurs projektowania będziecie w stanie sam napisać swój własny program monitora. Na razie na etapie wstępnej nauki programowania, do której niebawem przejdziemy, niezbędna będzie wersja „gotowa" biosu, wierzcie mi drodzy Czytelnicy, zaoszczędzi to wam wielu stron teorii, która na początku mogłaby niejednego z Was przerazić swoimi rozmiarami, a którą sami nabędziecie z czasem, ucząc się pisać pierwsze swoje programy.

W następnym odcinku omówię pokrótce (praktyczne rady) ważniejsze tematy dotyczące „czasowego" sposobu wykonywania przez mikrokontroler 8051 programu, po czym przejdziemy do pierwszych kroków w programowaniu.

Sławomir Surowiński