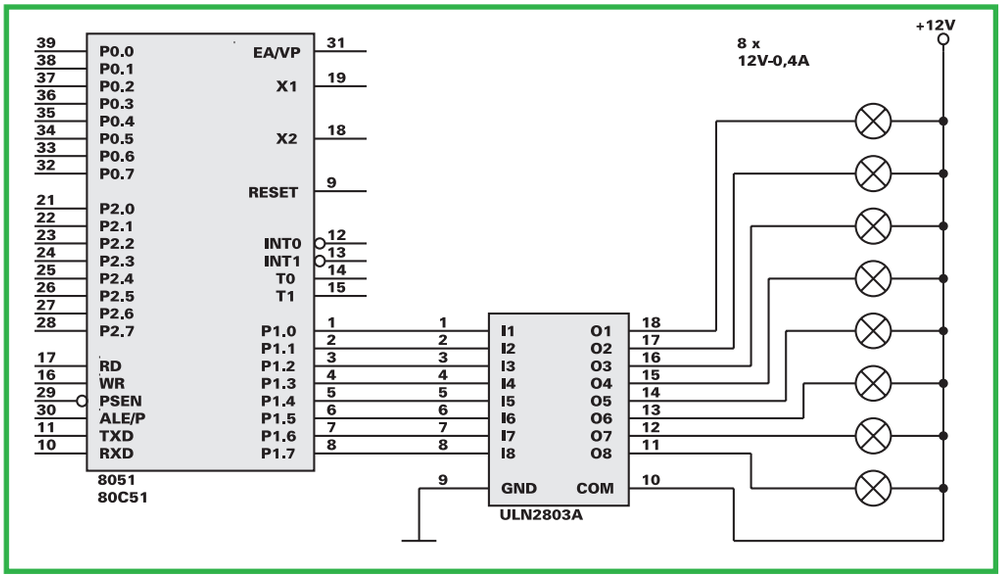
Mikrokontrolery? To takie proste...

Część 17
Peryferyjne układy wejścia-wyjścia

Dzisiejszy odcinek cyklu o mikrokontrolerach rodziny 8051 poświęcimy układom peryferyjnym I/O, czyli tzw. wejścia-wyjścia (ang. "Input/Output). Omówione zostaną typy układów, najciekawsze rozwiązania konstrukcyjne, od tych najprostszych do nieco bardziej skomplikowanych. Aby jednak zachować związek z cyklem o mikroprocesorach , wszyskie przedstawione w artykule układy będą w 100% kompatybilne z komputerkiem edukacyjnym AVT-2250, z którym wielu z Was, drodzy Czytelnicy pracuje już od kilku miesięcy. Zapoznamy się z kilkoma rozwiązaniami praktycznymi, a analiza przykładowych procedur obsługi tych układów pozwoli na zrozumienie zasady ich działania oraz nauczy każdego elastycznie dołączać dodatkowe urządzenia wejścia-wyjścia do dowolnego układu mikroprocesorowego, opartego o procesory serii '51, a także podobne.

Co to są właściwie te układy wejścia/wyjścia i do czego służą?. Najprostszą odpowiedzią na to pytanie niech będzie przykład, kiedy to za pomocą mikroprocesora (pracującego np. w układzie komputerka edukacyjnego - z zewnętrzną magistralą danych) chcemy wysterować wiele (np. 20) żarówek, zapalając je na przemian, tak jak to się dzieje w przypadku popularnych węży świetlnych. Każdy w tym miejscu powie, że do tego potrzebne będą układy pośredniczące (tzw. mocy), bowiem jak wiemy obciążalność prądowa wyjść portów procesora jest niewielka, zresztą żarówki pracują w większym zakresie napięć, toteż dołączenie ich bezpośrednio do mikrokontrolera z pewnością przyczyniło by się do jego uszkodzenia. Jest to prawdą, ale czy do końca? Odpowiedź z pewnością znajdziecie w niniejszym artykule.

Rys. 1 Dołączenie sekcji żarówek do procesora 8051



PROSTE UKŁADY WYJŚCIOWE

Na rys.1 pokazano przykładowe dołączenie 8-miu niskonapięciowych żarówek do jedynego, wolnego portu procesora 8051, pracującego z zewnętrzną pamięcią programu - tak jak nasz komputerek edukacyjny. Jak widać w naszym przykładzie zastosowano, jako bufor mocy popularny układ ULN2803, który jest ośmiokrotnym buforem mocy, dodatkowo zabezpieczony diodami zwrotnymi przed przepięciami.

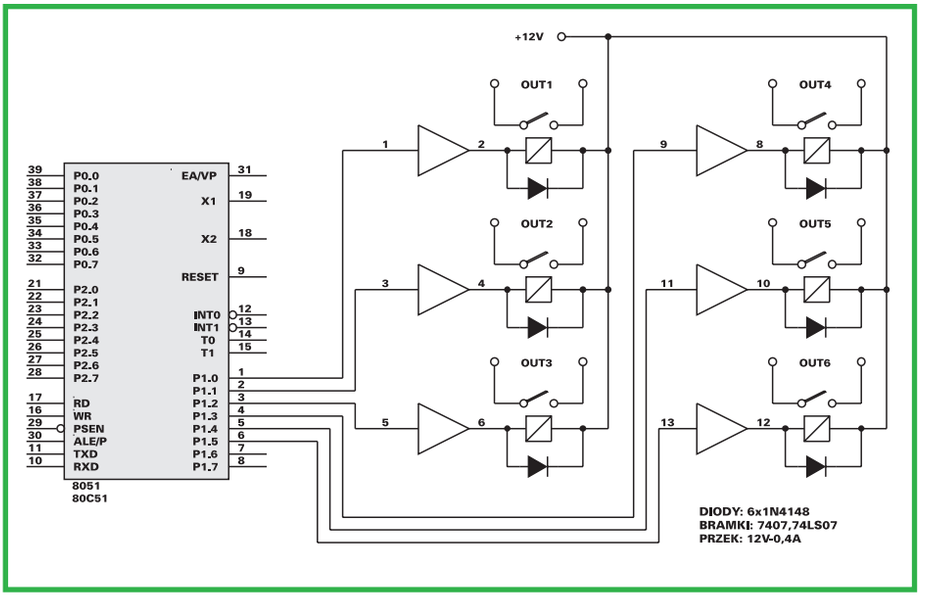
Zamiast takiego układu, można np. posłużyć się bramkami TTL z otwartym kolektorem i wyjściami wysokonapięciowymi. Do takich układów należą np.:

7406 - 6-krotny inwerter z wysokonapięciowym wyjściem (do +30V) typu otwarty kolektor. Ociążalność każdej bramki wynosi dodatkowo 10 wejść TTL.

7407 - 6-krotny wzmacniacz, bufor, z wysokonapięciowym wyjściem (do +30V) typu otwarty kolektor. Obciążalność każdej bramki wynosi 25 wejśc TTL.

7438 - 4 dwuwejściowe braki NAND typu otwarty kolektor ze zwiększoną obciążalnością do 30 wejść TTL.

Nie będę tu przytaczał schematów wewnętrznych tych układów, wszystkie one mniej lub bardziej dokładnie zostały omówione w cyklu artykułów "Pierwsze kroki w cyfrówce" w EdW. Na rys.2 przedstawiono sposób na zwiększenie obciążalności wyjść procesora za pomocą jednego z w/w układów do sterowania sekcją 6-ciu przekaźników. Do tego celu zaprzęgnięto 1 układ 7407. Dodatkowo każdą cewkę przekaźnika zbocznikowano diodą małosygnałową w celu tłumienia przepięć tworzących się w cewce podczas zdejmowania napięcia z jej zacisków.



Rys.2 Sterowanie przekaźnikami za pomocą portu P1 procesora.

Aby zilustrować sterowanie taką sekcją przekaźników posłużmy się przykładem. Otóż aby załączyć np. tylko wyjścia OUT1, OUT2 i OUT5, należy wykonać instrukcję:

```
MOV     P1, #101100b      ;wyzerowanie bitów - końcówek
                          0, 1 i 4 portu P1
```

Wyzerowanie odpowiednich końcówek portu P1 spowoduje "otwarcie" wyjścia odpowiedniej bramki i załączenie przekaźnika. Te przekaźniki, które mają pozostać wyłączone, powinny mieć ustawione odpowiednie bity rejestru portu P1. W naszym przykładzie są to bity 2, 3 i 5 (bity numeruje się od zera), co odpowiada przekaźnikom OUT3, OUT4 i OUT6.

Można także włączyć lub wyłączyć dowolny przekaźnik oddzielnie, jaką instrukcją, to zapewne już wiecie, ano chociażby włączenie przekaźnika OUT4 realizowane jest za pomocą polecenia:

```
CLR  P1.3          ;włączenie przekaźnika OUT4
```

a wyłączenie

```
SETB  P1.3         ;wyłączenie przekaźnika OUT4
```

Zauważmy że takie sterowanie przekaźnikami "od plusa" zasilania (kiedy cewka znajduje się po stronie dodatniego napięcia a nie masy) w układach procesorowych dodatkowo zabezpiecza przekaźniki, przed przypadkowym, krótkotrwałym włączeniem w przypadku, kiedy uruchamiamy cały układ. Jak bowiem wiecie z poprzednich odcinków klasy mikroprocesorowej, procesor po włączeniu zasilania "zeruje się" i na wszystkich jego portach wstępnie ustawiają się stany wysokiej impedancji, co dla bramek TTL jest po prostu jedynką. Dlatego w takiej sytuacji przekaźniki pozostaną wyłączone.

Dlatego w tak prostych układach sterowania, np. przekaźnikami nie zaleca się stosowania bramek np. 7406. Wspominałem o tym już w pierwszych dwóch numerach klasy mikroprocesorowej ponad rok temu.

Przedstawione powyżej przypadki to najprostsze z możliwych przykłady urządzeń wyjścia (ang. "Output") w układach mikroprocesorowych. I nie są to bynajmniej wspominane żarówki, czy przekaźniki oraz towarzyszące im elementy, ale "to co nimi steruje". W tym przypadku jest to port P1 procesora - to on był właśnie układem wyjścia, wyposażone dodatkowo w bufory zwiększające jego obciążalność.

PROSTE UKŁADY WEJŚCIOWE

Często zachodzi potrzeba monitorowania stanów logicznych na kilku liniach jednocześnie i podejmowania w związku z tym określonych działań. Najprostszym przykładem jest "obserwacja" przez procesor linii dozorowych w systemie alarmowym. Na rys.3 pokazano uzbrojone wejścia portu P1, jako najprostszy układ wejściowy procesora. Rolę buforów wejściowych pełnią bramki zabezpieczone dodatkowo na wejściach przez przepięciami diodami małosygnałowymi lub transilami. Te ostatnie to nowoczesne elementy potrafiące "gasić" przepięcia sięgające kilkuset woltów i trwające mikrosekundy - a więc bardzo krótko, wystarczająco jednak długo aby zniszczyć bramkę, a co za tym idzie cały układ scalony. Oczywiście nie muszę w tym miejscu przestrzegać przez bezpośrednim dołączaniem wysokonapięciowych linii zewnętrznych do wejść procesora, może mieć to bowiem złe skutki i spowodować awarię całego systemu.

W układzie w celu wyeliminowania stanów przejściowych zastosowano bramki Schmitta zawarte w popularnym układzie 74132. Taki system pozwala na monitorowanie sygnałów cyfrowych nadchodzących np. ze znacznej (jak na układy cyfrowe) odległości, czyli już od 50 cm w górę nawet do kilku metrów. Układy wejściowe dostosowane do większych napięć przedstawimy przy innej okazji, już teraz można jednak stosować chociażby np. proste dzielniki napięcia dodatkowo transilami lub diodami.

Odczyyt stanu na danym wejściu możliwy jest podobnie jak w przypadku wyjść. Można więc odczytać cały port na raz np. za pomocą instrukcji:

```
MOV    A, P1
```

a potem analizować poszczególne bity akumulatora, lub monitorować każdy pin portu oddzielnie. Nie należy jednak zapomnieć, aby przedtem zainicjować końcówkę portu jako wejście wpisując do rejestru tego portu logiczną jedynkę, za pomocą instrukcji:

```
SETB     P1.x
```

gdzie x - oznacza numer końcówki portu P1. Sytuacja taka jest konieczna tylko wtedy, kiedy od momentu uruchomienia (zasilenia) procesora, port P1 (lub niektóre jego piny) były wykorzystywane jako wyjścia pracujące w stanie logicznego "0".

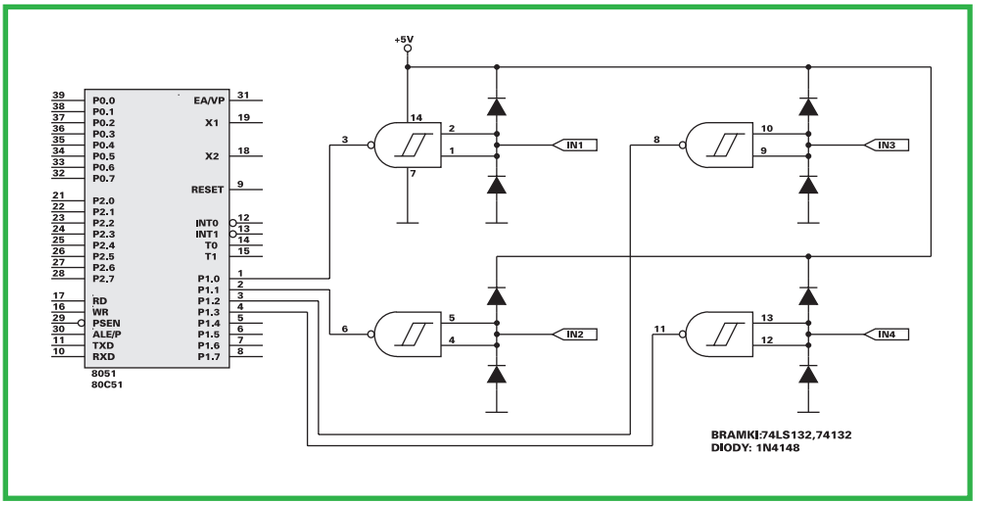
Toteż jeżeli przedtem użyłeś polecenia np.

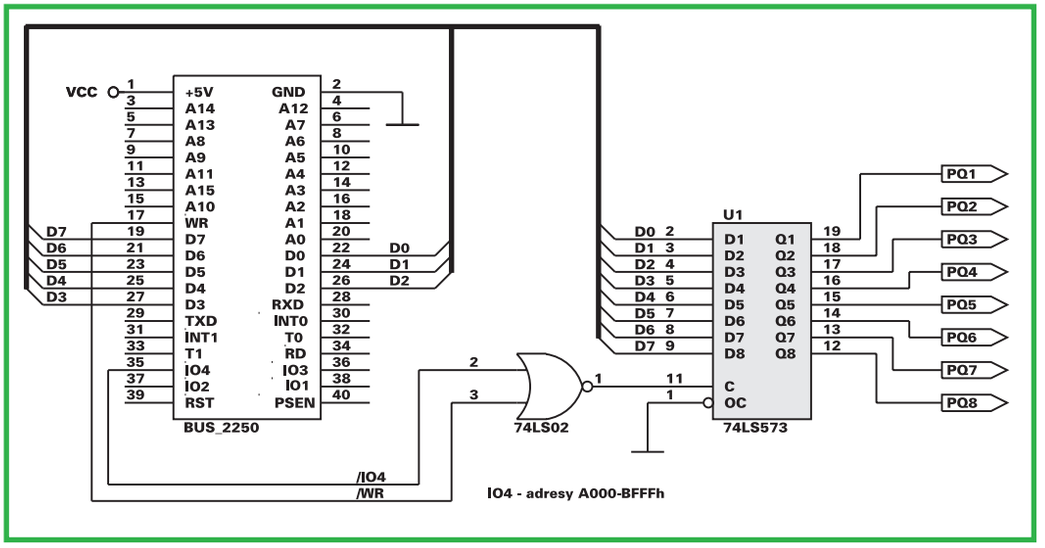
```
CLR    P1.5        ;ustawienie stanu
                   logicznego "0" na
                   końcówce 6
                   procesora
```

zerowałeś jakieś piny portu P1 poleceniem

```
MOV    P1, xx
```

Rys.3 Najprostszy sposób na bezpieczny monitorowanie sygnałów niskonapięciowych z zakresu -0,5... +5,5V.





Rys.4 Dołączenie zewnętrznego portu wyjściowego

gdzie xx to albo stała albo rejestr zawierający liczbę z wyzerowanymi bitami

powinieneś, przed użyciem tego pinu jako wejścia, ustawić w rejestrze portu P1 jego bit, np. poleceniem:

```
SETB    P1.5
```

tylko raz, a potem odczytywać stan końcówki, gdy zajdzie potrzeba przy pomocy instrukcji np.

```
MOV     C, P1.5
```

lub testować końcówkę portu wprost za pomocą instrukcji np. skoków warunkowych

```
JB      P1.5, ustawiony

.....   ; instrukcje, gdy stan na pinie 5 portu P1 jest niski
```

ustawiony:

```
.....   ; instrukcje, gdy stan na tej końcówce jest wysoki
```

... ZA MAŁO..., ODWIECZNY PROBLEM

W praktyce, kiedy mamy do czynienia z procesorami 8051 pracującymi z zewnętrzną pamięcią programu, do wykorzystania pozostaje jedynie port P1 oraz niektóre końcówki portu P3. Porty P0 i P2 są zajęte generowaniem sygnałów na szynie adresowej w celu odczytu kolejnych instrukcji z pamięci programu - EPROM, lub odczytem / zapisem danych do zewnętrznej pamięci danych (SRAM). Wtedy w najlepszej sytuacji mamy do wykorzystania wolne piny:

- P1.0... P1.7 - 8 końcówek

- P3.2 (INT0), P3.3 (INT1), P3.4 (T0), P3.5 (T1), P3.0 (RXD), P3.1 (TXD) - 4 końcówki

- P3.6 (WR) i P3.7 (RD) - dodatkowo 2 końcówki, jeżeli nie korzystamy z zewnętrznej pamięci danych

W sumie więc mamy do dyspozycji 16 uniwersalnych końcówek wejścia/wyjścia (I/O). Niestety często okazuje się że to stanowczo za mało. Wtedy z pomocą może nam przyjść rozbudowanie układów wejścia/wyjścia za pomocą dodatkowych układów cyfrowych (np. serii TTL), dołączonych do magistrali procesora.

Dzięki takim układom oraz za pomocą instrukcji odwołujących się do zewnętrznej pamięci danych (MOVX.....) możliwe jest "ustawianie" lub odczytywanie większej - praktycznie dowolnej ilości końcówek cyfrowych.

Na rys.4 przedstawiony jest dobudowany do magistrali komputerka edukacyjnego prosty układy wyjściowy. BUS_2250 to złącze - magistrala komputerka znajdująca się na płytce bazowej w postaci 2-rzędowego 40-pinowego złącza, na jej krawędzi.

Zastosowano układ scalony, 74LS573, który jest 8-bitowym latch'em, znanym z konstrukcji komputerka AVT-2250, gdzie pełnił rolę zatrzasku młodszej części 16-bitowego adresu. Dla przypomnienia powiem, że podanie wysokiego poziomu logicznego na wejście C (pin 11) układu 74LS573 powoduje przeniesienie stanów logicznych panujących na wejściach D1...D8 tego układu odpowiednio na wyjścia Q1...Q8. Wejście OC steruje trójstanowymi wyjściami Q1...Q8. Podanie stanu niskiego na tę końcówkę powoduje odblokowanie wyjść Q1...Q8, dzięki czemu zatrzaśnięte z wejść D1...D8 stany logiczne pojawią się na wyjściach układu U1.

Utrzymywanie tej końcówki w stanie logicznej "1" powoduje ustawienie wszystkich wyjść w stan wysokiej impedancji.

W tabeli 1 przypominam zasadę działania układu 74573. Znaczenie symboli jest następujące:

Z – stan wysokiej impedancji
H – stan wysoki (logiczny)
L – stan niski (logiczny)
x – stan nieistotny (dowolny)

Tabela 1

wejścia			wyjścia
OC	C	D	Q
H	x	x	Z
L	L	x	bez zmian
L	H	L	L
L	H	H	H

Ponieważ w naszym przykładzie układ U1 pracuje jako wyjściowy, końcówkę OC dołączyliśmy do masy, tak aby na wyjściach panował stan - zatrzaśnięty z wejść D1...D8 podczas opadającego zbocza sygnału na wejściu "C" (pin 11).

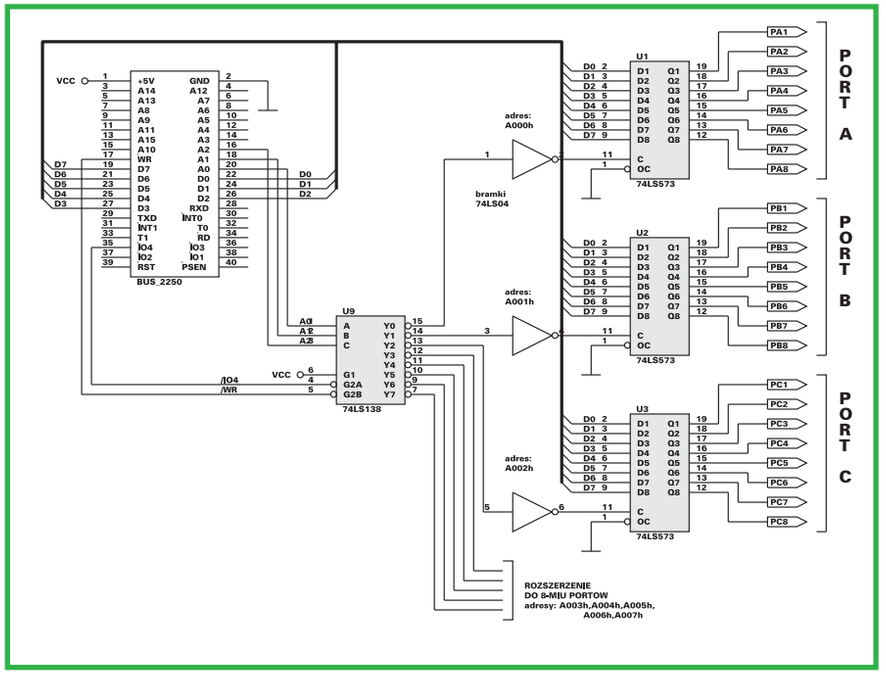
Nie wchodząc na początku w szczegóły powiem, że układ z rys.4 pozwala na zapamiętanie stanów cyfrowych na ośmiu wyjściach (Q1...Q8) układu U1 za pomocą instrukcji:

```
MOV     DPTR, #IO4

MOV     A, #10101010b

MOVX  @DPTR, A
```

W przykładzie tym wykonanie instrukcji spowoduje ustawienie na przemian stanów logicznych wysokich i niskich na końcówkach Q1...Q8 portu U1. Stanie się tak za sprawą zapisania liczby 10101010b pod adres w zewnętrznej przestrzeni adresowej od adresu IO4 (A000h).

Przeanalizujmy zapis do układu U1. Procesor wykonując instrukcję MOVX,... powoduje pojawienie się stanu niskiego na końcówce IO4 złącza komputerka w przypadku kiedy w rejestrze DPTR znajdzie się dowolny adres z zakresu A000h...BFFFh. Równocześnie w momencie zapisu do rejestru procesor podaje stan niski na linię /WR, co w efekcie powoduje powstanie stanu wysokiego na wyjściu braki 74LS02 (rys.4), która realizuje negację sumy logicznej sygnałów /WR i /IO4. Wysoki stan na wyjściu tej bramki powoduje zapisanie danych D0...D7 z szyny danych procesora 8051 do rejestru U1, a dzięki temu, że wejście /OC U1 jest zwarte do masy, dane te pojawią się natychmiast na wyjściach Q1...Q8 zatrzasku U1.

W ten sposób za pomocą jednej bramki logicznej, jednego układu scalonego (U1) oraz trzech instrukcji zrealizowaliśmy prosty układ portu wyjściowego o ośmiu wyjściach. Wyjścia tse, podobnie jak opisywanego wcześniej portu P1, można teraz wykorzystać do dowolnych celów, np.. do sterowania układami wykonawczymi dowolnych urządzeń, np. 8-kanałowego węża świetlnego.



Rys.5 Jeden ze sposobów na rozbudowanie ilości układów I/O procesora.

Zapisując dowolne, zależne od sytuacji wartości za pomocą instrukcji podanych w ostatnim przykładzie (w miejsce #10101010b) możemy wpływać na stany 8-miu wyjść portu U1, a dzięki temu sterować maksymalnie 8-moma urządzeniami zewnętrznymi.

Na rys.5 przedstawiłem sposób na powiększenie ilości układów wyjściowych. Jak widać do tego celu wykorzystano dodatkowy układ dekodera 74LS138 (U9). Dzięki takiemu rozwiązaniu, za pomocą 1 dekodera '138 możliwe jest zaadresowanie maksymalnie 8-miu rejestrów typu 74LS573. Aby zrozumieć schemat ideowy najlepiej jest znaleźć podobieństwa z rys.4. W tym przypadku dekoder U9 dekoduje 3 najmłodsze linie adresowe magistrali procesora: A2,A1 i A0. Dzięki temu każdy z rejestrów 74LS573 może być zapisany oddzielnie, za pomocą sekwencji instrukcji podanych w poprzednim przykładzie, z tym że inne będą wartości wpisywane do rejestru DPTR. I tak aby np. zaadresować (zapisać) rejestr U3 należy posłużyć się instrukcjami:

```
MOV     DPTR, #IO4_U3

MOV   A, # 11001100b

MOVX  @DPTR, A
```

wcześniej należy jednak umieścić deklarację opisującą zapis: IO4_U3, tzn.

```
IO4_U3          equ     A002h
;bo IO4=A000h +2 (ofset U3)
```

Można także zapisać adres od razu do rejestru DPTR, instrukcją:

```
MOV   DPTR, #A002h
```

a efekt będzie ten sam.

W układzie z rys.5 sygnały /IO4 i /WR zostały dołączone do dwóch wejść zezwalających dekodera U9, dzięki czemu w momencie gdy jeden i drugi przyjmuje stan "0" dekoder zostaje odblokowany i uaktywnione zostaje (stanem niskim) jedno z 8-miu wyjść układu U9, zależnie od stanu linii adresowych A2...A0.

Stan niski z wyjścia dekodera U9 zostaje zanegowany przez dołączoną do niego bramkę negacji (NOT) a następnie w postaci dodatniego impulsu (zapisu) powoduje zatrzaśnięcie danych D0...D7 z magistrali komputerka w jednym z rejestrów 74LS573. Reasumując aby "wysterować" poszczególne rejestry, i każde z ich 8-miu wyjść, należy do DPTR wpisać ich odpowiednie adresy, i tak:

```
dla U1   - MOV  DPTR, #0A000h

dla U2   - MOV  DPTR, #0A001h

dla U3   - MOV  DPTR, #0A002h

dla U8   - MOV  DPTR, #0A007h
```

a następnie wykonać zapis do akumulatora stanu 8-miu wyjść danego rejestru, np. :

```
MOV   A, 11111110b          ;tylko wyjście Q8 w stanie "0", reszta "1"
```

i wykonać zapis

```
MOVX    @DPTR, A
```

i to wszystko, prawda że proste!

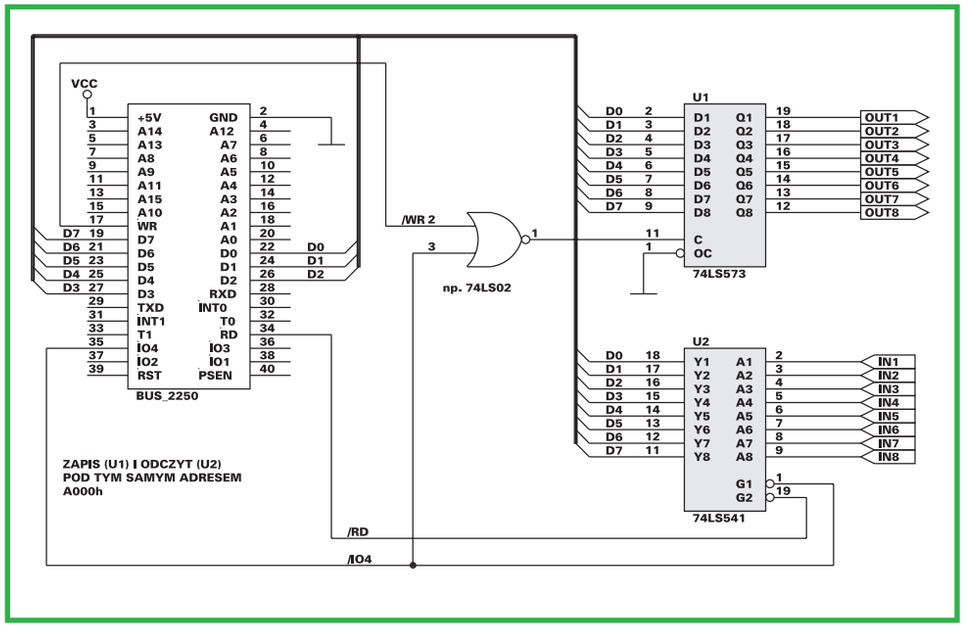
Na rys.5 nie pokazano, ze względu na oszczędność miejsca dołączenia pozostałych 5-ciu rejestrów U4...U8, ale sposób sterowania i podłączenia do dekodera jest identyczny jak w przypadku U1...U3, czyli do wyjść dekodera U9 poprzez bramki negacji.

We wszystkich pokazanych przykładach można używać układów w wersjach CMOS, np. 74HCT138, 74HCT04, 74HCT573, itp., należy jednak pamiętać aby uwzględnić mniejszą obciążalność wyjść tych ostatnich przy dołączaniu dodatkowych układów wyjściowych. Uwaga ta będzie dotyczyć także pozostałych przedstawionych w artykule przykładów.

No dobrze, układy wyjściowe, umożliwiające sterowanie zewnętrznymi urządzeniami mamy w zasadzie omówione, no przynajmniej te najprostsze rozwiązania. Pora przejść do układów, które umożliwią odczyt zewnętrznych stanów logicznych przez procesor, krótko mówiąc "badanie" cyfrowych sygnałów dochodzących z zewnątrz do układu komputerka.

Rysunek 6 przedstawia prosty sposób na sterowanie zapisem i odczytem 8-miu zewnętrznych linii cyfrowych. Spójrzmy przez chwilę jedynie na zastosowanych tu nowy układ scalony U2 – 74LS541.

Jest to 8-mio bitowa "brama", dzięki które możliwy jest transfer danych z jej



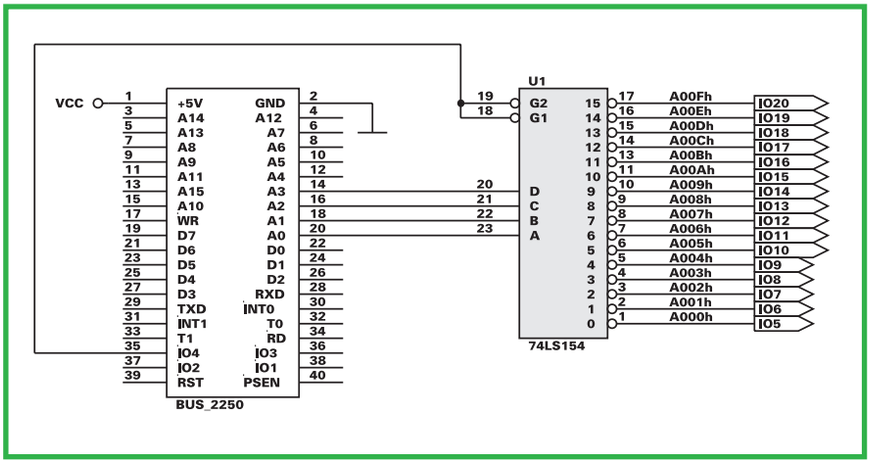
Rys.6 Sposób na wykorzystanie tego samego adresu do sterowania zapisem (U1) i odczytem (U2) sygnałów cyfrowych.

wejść (A1...A8) na wyjścia (Y1...Y8) w momencie, kiedy oba sygnały sterujące G1, G2 znajdują się w stanie niskim. Innymi słowy, kiedy podamy na wejścia G1, G2 stan niski, to "to" co pojawi się na wejściu (np. A1) pojawi się także na wyjściu Y1. W przeciwnym przypadku, gdy jedno (lub oba) wejścia znajdują się w stanie wysokim, wszystkie wyjścia Y1...Y8, znajdują się z stanie wysokiej impedancji. Dzięki temu m.in. możliwe jest zastosowanie tych układów w systemach mikroprocesorowych, takich jak komputerek edukacyjny.

Tabela 2 opisuje działanie układu.

Tabela 2

wejścia			wyjścia
G1	G2	A	Y
H	x	x	Z
x	H	x	Z
L	L	L	L
L	L	H	H



Rys.7 Zastosowanie układu demultipleksera jako dekodera adresów

Popatrzmy na tę część schematu, która odnosi się do układu U2, bowiem sposób podłączenia U1 i bramki NOR został już omówiony. Podobnie jak w przypadku układu wyjściowego (U1) podłączony został bufor wejściowy U2. W tym przypadku obyło się jednak bez korzystania z dodatkowej bramki sumy logicznej, bowiem, wykorzystano dwa wejścia zezwalające G1 i G2 układu 74LS541. Jedno z tych wejść dołączono do sygnału /RD procesora, a drugie do sygnału /IO4, który jak wiemy przyjmuje stan niski w momencie adresowania obszaru A000h...BFFFh procesora. W ten prosty sposób, kiedy np. chcemy odczytać stany wejść A1...A8 układu U2, możemy to zrobić za pomocą instrukcji:

```
MOV     DPTR, #IO4

MOVX A, @DPTR
```

i to wszystko. W akumulatorze znajdzie się liczba, której poszczególne bity odpowiadają stanom na wejściach A1...A8 w momencie odczytu (tzn. kiedy sygnał /RD przyjmuje stan niski) przez mikroprocesor portu U2.

W tym miejscu bardziej wnikliwi czytelnicy mogą dostrzec problem, który może pojawić się jeżeli w obszarze adresowym dekodowanym przez sygnał IO4 pojawi się inny układ, np. pamięć (lub jej część) statyczna SRAM. Wtedy odczytując port U2 (adres 0A000h) odczytana może zostać komórka w pamięci SRAM. W przypadku gdy obie dane pochodzące z portu i pamięci nie będą zgodne nastąpi efekt iloczynu montażowego, który w najbardziej niekorzystnym przypadku (przy użyciu układów serii LS) może spowodować uszkodzenie pamięci lub portu U2. Dlatego w przykładach omawianych w niniejszym artykule, należy sprawdzić, aby pamięć SRAM w komputerku miała pojemność 8kB (typ 6264). W przypadku gdy posiadasz pamięć 62256 (32kB), należy przełączyć zworę JP3 na płytce bazowej komputerka na adres C000h a programy testujące porty I/O opisane w artykule, kompilować oczywiście z dyrektywą:

```
ORG     C000h
```

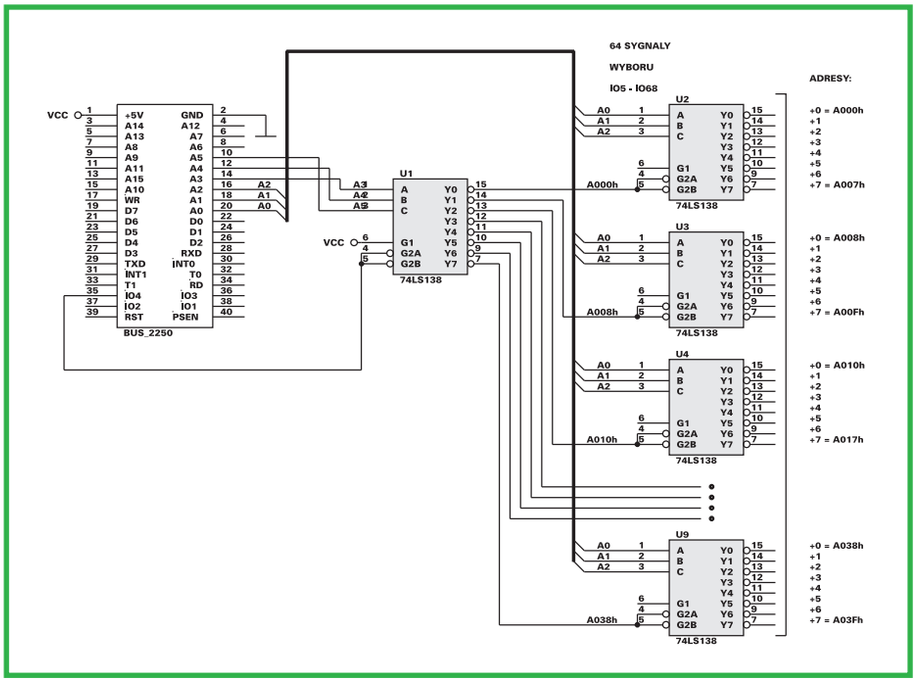
Na rys.7 przedstawiono sposób na zwiększenie liczby sygnałów wyboru I/O do 16-tu za pomocą zwykłego układu demultipleksera 74LS154. Dzięki takiemu dołączeniu linii adresowych oraz sygnału /IO4 komputerka do układu U1, w prosty sposób uzyskujemy dodatkowe linie sterujące dowolnymi urządzeniami np. I/O, takimi jakie opisałem wcześniej. W układzie z rys.7 poszczególne wyjścia IO5...IO20 odpowiadają następującym adresom:

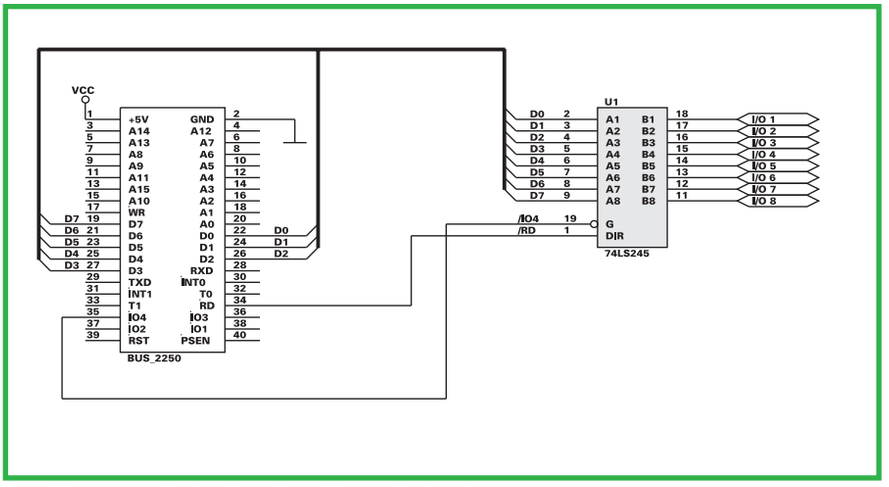
IO5, adres A000h
IO6, adres A001h
IO7, adres A002h
IO8, adres A003h
IO9, adres A004h
IO10, adres A005h
IO11, adres A006h
IO12, adres A007h
IO13, adres A008h
IO14, adres A009h
IO15, adres A00Ah
IO16, adres A00Bh
IO17, adres A00Ch
IO18, adres A00Dh
IO19, adres A00Eh
IO20, adres A00Fh

Na kolejnym rysunku 8 pokazany jest sposób na generowanie wielu sygnałów wyboru I/O za pomocą kaskadowo połączonych dekoderów 74138. Jak widać, dzięki układowi U1 możliwe jest wysterowanie kolejnych dekoderów U2...U9 i w efekcie uzyskanie 64 sygnałów wyboru. Analizując powyższy schemat należy zwrócić uwagę na sposób dołączania poszczególnych linii adresowych do wejść A,B i C dekoderów 74138.

Zrozumienie zasady łączenia wejść adresowych pozwoli na samodzielne, dowolne organizowanie przestrzeni adresowej procesora. Kolejne przykłady przybliżą Ci to, drogi Czytelniku.

Rys.8 Sposób na kaskadowe łączenie dekoderów 74138





Rys.9 Zastosowanie dwukierunkowej bramy logicznej

Rysunek 9 przedstawia zastosowanie trzeciego ciekawego układu scalonego z popularnej serii TTL, mianowicie dwukierunkowej 8-mio bitowej bramy logicznej.

Podobnie jak poprzednio wyjaśniam, że układ 74LS245 potrafi przekazywać dane z wejść A1...A8 na wyjścia B1...B8 i odwrotnie, tzn. z B1...B8 (które wtedy są wejściami) na A1...A8 (które pełnią rolę wyjść), w zależności od poziomu na końcówce DIR (pin 1). Tabela 3 przedstawia zależności logiczne układu.

Tabela 3

wejścia		funkcja
G	DIR	
H	H	A = B = Z
H	L	A = B = Z
L	H	A -> B
L	L	A <- B

Jak widać z tabeli podanie stanu wysokiego na wejście DIR układu powoduje przekazanie danych z A1...A8 odpowiednio na wyjścia B1...B8, i odwrotnie podanie stanu niskiego powoduje przekazanie stanów z B1...B8 na A1...A8. Sygnał G służy do uaktywniania układu. Podanie na ten pin stanu wysokiego powoduje ustawienie końcówek A1...A8 i B1...B8 w stan wysokiej impedancji.

Należy zwrócić uwagę, że układ 74LS245 nie jest zatrzaskiem, tzn. że nie zapamiętuje danych z wejść, lecz je tylko "przekazuje" na wyjścia.

W układzie przykładowym z rys.9 sterowanie kierunkiem przepływu informacji następuje pod wpływem sygnału /RD procesora, dołączonego do wejścia DIR układu bramy U1. W takiej konfiguracji układ wykorzystywany jest działa podobnie jak 74LS541, przekazuje dane z wejść B1...B8 na wyjścia A1...A8 a dalej na szynę danych procesora (komputerka) D0...D7. Kiedy procesor adresuje bufor poprzez sygnał zapisu (/WR) uaktywniony zostaje sygnał IO4, który powoduje przekazanie stanów z szyny danych komputerka D0...D7 na końcówki B1...B8 bramy U1. Ktoś może zauważyć, że taki sposób obsługi układu w trybie zapisu może być "nieeleganckі", bowiem w procesie aktywacji układu U1 nie jest wykorzystywany sygnał zapisu /WR. I choć na pierwszy rzut oka jest to prawdą, to w praktyce takie sterowanie wystarcza. Ja jednak proponuję Ci, drogi Czytelniku, jako ćwiczenie narysowanie pełnego dekodera dla przykładu z rys.9.

Na rys.10 przedstawiłem ciekawy układ dwukierunkowego portu I/O z możliwością zapamiętania wartości w trybie zapisu. Działanie portu jest identyczne z funkcjami portów mikroprocesora (np. omawianego wcześniej P1), jednak w tym przypadku możliwe jest używanie wszystkich 8-miu bitów portu jednocześnie jako wyjścia lub wejścia. Nie ma więc możliwości ustawiania pojedynczych końcówek portu (jak to ma miejsce w przypadku procesora) jako wejście a innych jako wyjścia w tym samym momencie. Jednak często taka funkcja nie jest nam potrzebna, toteż powyższy układ często znajduje swoje zastosowanie w praktyce i przyznam spisuje się w niej znakomicie.

Można zatem porównać ten układ do portu procesora, np. P1 i powiedzieć, że instrukcja zapisu typu:

```
MOV   DPTR, #IO4

MOV   A, #11110000b

MOVX  @DPTR, A
```

działa podobnie jak instrukcja

```
MOV   P1, #11110000b
```

oczywiście w odniesieniu do dwóch różnych układów wejścia-wyjścia.

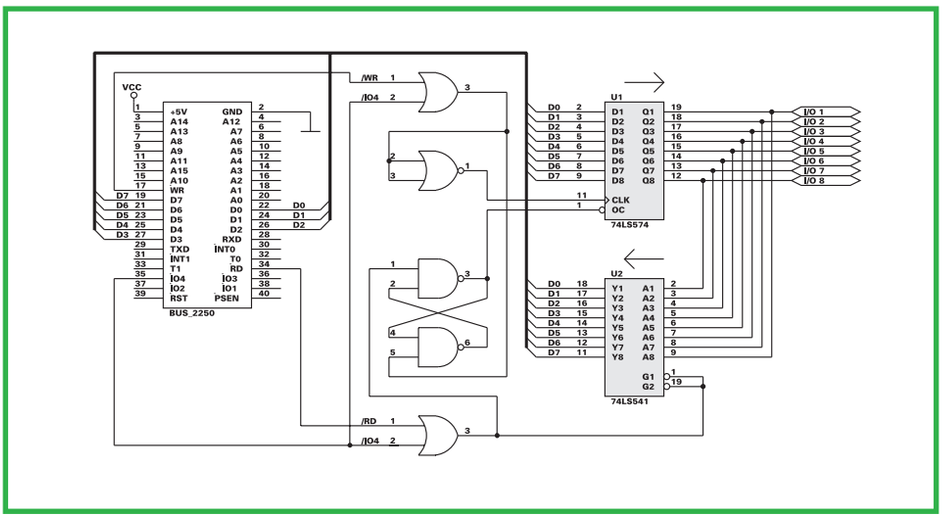
Podobnie instrukcje:

```
MOV   DPTR, #IO4

MOVX  A, @DPTR
```

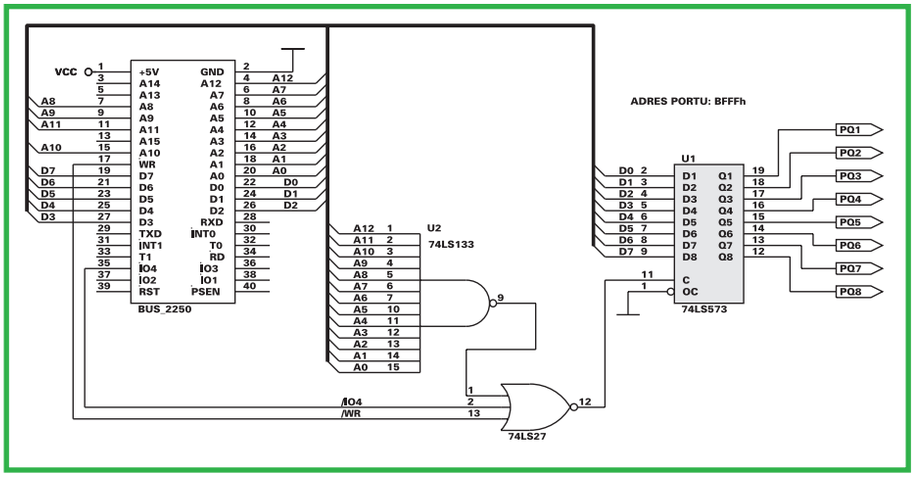
powodują odczytanie stanów końcówek portu z rys.10, co dla portu procesora można było wykonać instrukcją:

```
MOV   A, P1
```

Rys.10 Dwukierunkowy port I/O – okrojona wersja portu mikroprocesora



Zamiast znanego już układu 74LS573 zastosowano podobnie działający układ 74LS574. Różnica w działaniu obu układów polega na tym, że pierwszy układ "przepuszcza dane" przez cały czas, kiedy na wyjście C podany jest stan wysoki, a zapamiętuje je podczas opadającego zbocza sygnału na tym wejściu. Drugi układ jest typowym zatrzaskiem wyzwalanym zboczem, stąd w symbolu graficznym zamiast oznaczenia C na końcówce 1 znajduje się symbol zegara CLK. Tak więc dane są zapamiętywane w układzie w momencie narastającego zbocza sygnału na tym wejściu. Działanie układu ilustruje także tabela 4. W praktyce w większości zastosowań możliwe jest zastosowanie obu układów zamiennie – zgodna jest zresztą topologia pozostałych wyprowadzeń obu kostek.



Rys.11 Przykład pełnego dekodowania pojedynczego układu I/O

Tabela 4

wejścia			wyjścia
OC	CLK	D	Q
H	x	x	Z
L	L	x	bez zmian
L	L->H	L	L
L	L->H	H	H

Popatrzmy na rys.10. W stanie spoczynku, sygnały /RD, /WR i /IO4 są w stanie wysokim. Dzięki temu na wejściu zapisu układu U1 (pin 11) panuje stan niski, dzięki bramce NOR negującej sumę logiczną sygnałów /WR i /IO4 (która dokonywana jest za pomocą bramki OR).

Dodatkowo wejście kasujące przerzutnika /RS zbudowanego z 2 bramek NAND jest w stanie wysokim. Podobnie sygnał /RD jest w stanie wysokim a po zsumowaniu z sygnałem /IO4 ustawia w stanie wysokim wejścia zezwalające G1, G2 układu U2, czyli blokuje działanie bramy U2. Jeżeli teraz procesor wykona cykl zapisu do portu, to w wyniku tego nastąpi zapis do układu U1 oraz dodatkowo zostanie ustawiony przerzutnik /RS, dzięki czemu na wejściu zezwalającym U1 (OC) powstanie stan niski, co uaktywni wyjścia tego układu i w efekcie pojawienie się danych z szyny danych komputerka D0...D7 na wyjściach I/O 1....I/O 8 całego układu. Cykl zapisu nie wpłynie przy tym na działanie układu wejściowego U2, który pozostanie nadal nieaktywny. Jeżeli procesor dokona cyklu odczytu, to stan niski z wyjścia bramki OR sumującej sygnały /RD i /IO4 spowoduje skasowanie przerzutnika /RS i pojawienie się stanu wysokiego na wejściu zezwolenia /OC U1 – co zablokuje układ U1 (wyjścia w stanie wysokiej impedancji). Dzięki temu możliwy będzie odczyt stanów linii

I/O 1...I/O 8. Ze względu na pewne niedoskonałości przedstawionego układu z rys.10, polegające na nieuwzględnieniu niektórych zależności czasowych, konieczne okazać się może dwukrotne odczytanie portu IO4 w momencie, gdy ostatnio był on używany jako wyjście. Dzieje się tak dlatego, że może nastąpić sytuacja, kiedy przerzutnik /RS nie zdąży przerzucić swego stanu i zablokować wyjścia układu U1, zanim procesor odczyta stan linii wyjściowych portu. W takim przypadku odczytana zostanie ostania wartość wpisana do portu U1.

Dlatego jeżeli ostatnią operacją na układzie z rys.10 było np.

```
MOV  DPTR, #IO4

MOVX  @DPTR, A              ; zapis do portu U1
```

to przy pierwszy odczyt warto przeznaczyć "na straty"

```
MOVX  A, @DPTR
```

a dopiero drugi

```
MOVX  A, @DPTR
```

potraktować jako prawidłowy i dokonać po tym analizy akumulatora.

ACH, TE ADRESOWANIE...

Z pewnością niektórzy z Was, drodzy Czytelnicy, zauważyli, analizując powyższe przykłady, fakt tzw. "niepełnego dekodowania adresów". Chodzi o to, że adresując – zapisując układ, np. z rys.4, mamy do czynienia z pojedynczym układem wyjściowym, czyli fizycznie zajmującym tylko jeden adres w przestrzeni adresowej procesora, a tymczasem, sposób podłączenia go do komputerka pozwala na użycie aż 8192 adresów (A000h...BFFFh). Dzieje się tak dlatego, bo wykorzystany do selekcji układu U1 (rys.4) sygnał /IO4 jest aktywny dla tego właśnie zakresu adresów. Toteż zaadresowanie

```
MOV  DPTR, #A000h

MOVX, @DPTR, A
```

czy

```
MOV  DPTR, #BFFFh

MOVX, @DPTR, A
```

czy może

```
MOV  DPTR, #B260h

MOVX @DPTR, A
```

w praktyce da ten sam efekt.

Można więc z całą odpowiedzialnością stwierdzić że jest to czyste "marnotrastwo" adresów! I tak z pewnością jest. Aby temu zaradzić stosuje się pełniejsze lub całkiem pełne dekodowanie danego adresu (lub kilku) tak aby np. port U1 z rys.4 był dekodowany tylko i wyłącznie dla danego jednego adresu np. BFFFh – i żadnego więcej. W ten sposób w pozostałym obszarze A000h...BFFEh można by umieścić pozostałe 8191 układów I/O, co nie zawsze w praktyce jednak jest potrzebne. Na rys.11 pokazano zmodyfikowany układ dekodowania pojedynczego portu wyjściowego (z rys.4). Jak widać układ U1 zostanie zmodyfikowany tylko jeżeli procesor zaadresuje go pod adresem: BFFFh. Dzieje się tak dlatego, że dodatkowa bramka U2 realizuje negację iloczynu logicznego młodszych 13-tu linii adresowych: A12...A0. Dzięki temu tylko wtedy, kiedy na liniach tych pojawi się logiczna jedynka oraz dodatkowo uaktywniony zostanie sygnał IO4, port U1 zostanie zapisany – oczywiście jeżeli procesor wykona instrukcję

```
MOVX  @DPTR, A
```

kiedy to także sygnał /WR przyjmie stan niski. Wtedy na wyjściu bramki U2 pojawi się niski stan logiczny, który wraz z niskim stanem sygnału /IO4 oraz niskim stanem /WR spowoduje pojawienie się wysokiego poziomu na wyjściu trzywejściowej bramki NOR.

W przykładzie zastosowano nietypową, ale spotykaną w handlu 13-wejściową bramkę logiczną NAND typu 74LS133. W razie trudności z nabyciem, można w zastępstwie użyć połączonych kaskadowo wielu bramek NAND o mniejszej liczbie wejść. Można także spróbować zawęzić obszar dekodowania układu I/O w inny sposób, możliwości jest wiele, toteż inne rozwiązania pozostawiam wam jako pracę domową, drodzy Czytelnicy.

W dzisiejszym odcinku to tyle na temat prostych układów wejścia-wyjścia. Następnym razem zajmiemy się bardziej złożonymi – programowalnymi układami I/O, dzięki którym możliwy jest nie tylko odczyt i zapis ale także generowanie przerwań. Pokażę wam także jak w prosty sposób wyposażyć nasz komputerek edukacyjny w port drukarkowy zgodny ze standardem Centronics – w dodatku pracujący w dwóch kierunkach!

Przedstawię schemat ideowy takiego rozwiązania oraz procedury w języku 8051 realizujące pożyteczne funkcje drukowania wprost z komputerka edukacyjnego oraz dodatkowo emulujące złącze drukarki w komputerku, dzięki czemu możliwy będzie transfer danych np. z komputera PC do komputerka AVT-2250 tym razem nie przez złącze RS-232c ale przez złącze równoległe Centronics i to z o wiele większą prędkością! Na razie to tyle, zapraszam więc do kilku dodatkowych zadań, które przygotowałem na powakacyjne jesienne już wieczory.

Sławomir Surowiński

LEKCJA 10

Proponuję trzy zadania. Oto one:

1. Zmodyfikować układ z rys. 9 tak aby uwzględniał przy sterowaniu sygnał /WR zapisu, bez zmiany funkcjonalnej układu – patrz tekst.
2. Narysować zmodyfikowany układ dekodera z rys.7 który zamiast układu 74LS154 wykorzystuje układy 74LS138.
3. Do układu z rys. 10 dobudować dekoder, dzięki któremu układ U1 będzie zapisywany pod adresem A000h a odczytywany spod adresu A001h.

Rozwiązania zadań w kolejnym numerze EdW. Wesołej zabawy !