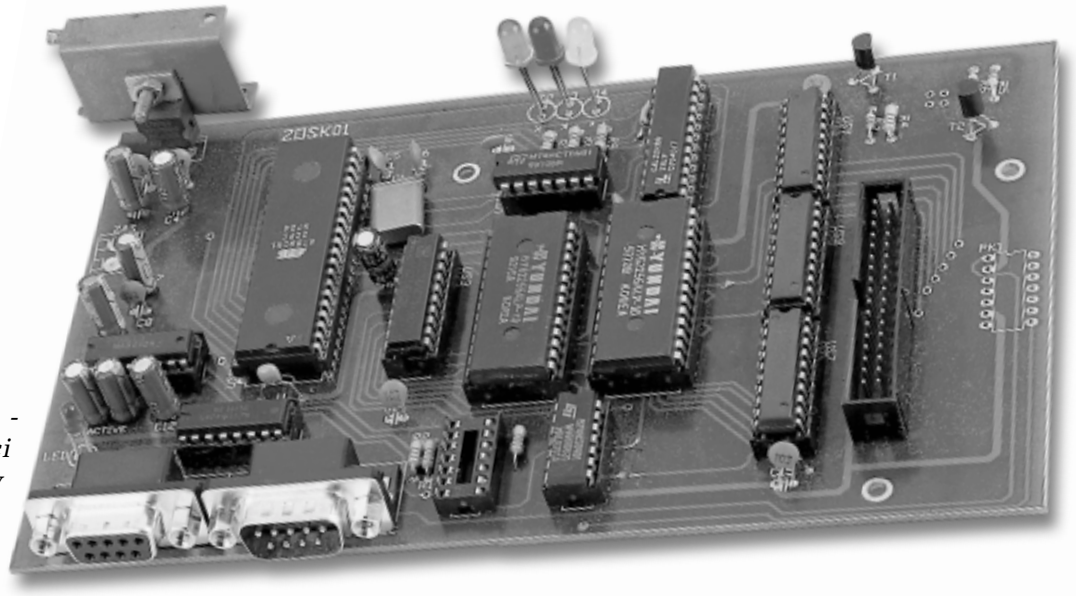
Symulator EPROM/EEPROM do wszystkich typów komputerów, część 2

AVT-870



W drugiej - ostatniej- części artykułu kończymy opis symulatora pamięci EPROM/ EEPROM dla Amigi (ale nie tylko!). Znajdziecie w niej opis montażu i uruchomienia, opis „języka" programowania symulatora oraz opis możliwości rozbudowy urządzenia.

Montaż i uruchomienie

Przebrnęliśmy przez długi opis, czas zająć się montażem. Schemat montażowy płytki drukowanej znajduje się na **rys. 2**. W pierwszej kolejności montujemy elementy najmniejsze (rezystory, diody, kondensatory), podstawki, złącza i stabilizator scalony. Przyłączamy zasilanie i sprawdzamy obecność napięć zasilających w podstawkach układów. Jeśi napięcie ma 5V ±10% możemy umieścić układy w podstawkach (pamiętajmy o wyłączeniu zasilania). Szczególną uwagę należy zwrócić na układy US2 (procesor) i US6 (GAL), które są zamontowane odwrotnie niż pozostałe. Wykonujemy kabelek (bez skrzyżowań lini TxD i RxD - **rys. 3**).

Aby nie „zaciemniać" rysunku narysowano tylko połączenia linii TxD, RxD i GND. Jak widać kable łączące komputer z symulatorem i połączenia pomiędzy innymi urządzeniami z portem przelotowym to zwykłe przedłużacze (jak do modemu). Kabel łączący urządzenie z przelotowym portem (jak symulator) a komputerkiem AVT, to zwykły kabel jakim łączymy komputer z AVT (zakładając, że w komputerze są zamontowane złącza 9 pin).

Uruchamiamy program terminala, ustawiamy prędkość transmisji na 4800, jeden bit stopu, brak parzystości. Zmontowany

Tab. 1.

Linia z wymuszonym poziomem wysokim	Efekt na ekranie
wszystkie=L	$ca %00000000, $00xx %00000000xxxxxxxx, RD
A0=H	$00 %00000000, $00xx %00000000xxxxxxxx, RD
A1=H	$01 %00000001, $00xx %00000000xxxxxxxx, RD
A2=H	$02 %00000010, $00xx %00000000xxxxxxxx, RD
A3=H	$03 %00000011, $00xx %00000000xxxxxxxx, RD
A4=H	$04 %00001000, $00xx %00000000xxxxxxxx, RD
A5=H	$05 %0000101, $00xx %00000000xxxxxxxx, RD
A6=H	$06 %0000110, $00xx %00000000xxxxxxxx, RD
A7=H	$07 %0000111, $00xx %00000000xxxxxxxx, RD
A8=H	$08 %0001000, $01xx %00000001xxxxxxxx, RD
A9=H	$08 %0001001, $02xx %00000010xxxxxxxx, RD
A10=H	$08 %0001010, $04xx %00000100xxxxxxxx, RD
A11=H	$08 %0001011, $08xx %00001000xxxxxxxx, RD
A12=H	$08 %0001100, $10xx %00010000xxxxxxxx, RD
A13=H	$08 %0001101, $20xx %00100000xxxxxxxx, RD
A14=H	$08 %0001110, $40xx %01000000xxxxxxxx, RD
A15=H	$08 %0001111, $80xx %10000000xxxxxxxx, RD
inne kombinacje	$ff %11111111, $??xx %????????xxxxxxxx,??

układ powinien zacząć działać od razu. W oknie terminala piszemy: *@se30*, na co uzyskamy odpowiedź:

```
Symulator Eprom V3.0-64KB
(C) 1999 by AVT-Korporacja
    Autor: S.Skrzynski
     Prog&Emul: Amiga
```

Dioda D1 powinna zaświecić. Wpisujemy i zatwierdzamy klawiszem [Enter]: *@27512* w oknie terminala powinien pojawić się znak „+“ (plus).

Jeśli wpiszemy np. *@ala ma kota* [Enter] ujrzymy:

Error: syntax

Gdy wpiszemy tekst dłuższy niż 16 znaków, w którym nie będzie znaku @ symulator odpowie:

Error: Buffer too short

Naciśnięcie znaku „:“ (dwukropka) spowoduje zaświecenie diody D4 (żółta). Po kilkukrotnym naciśnięciu klawisza 1 (jeden) dioda D4 zgaśnie, a symulator zgłosi komunikat błędu sumy kontrolnej.

Rozkaz: *@end* lub dziesięciosekundowa nieaktywność spowoduje odłączenie symulatora od magistrali RS (LED D1 gaśnie). Symulator można uznać za sprawny.

Sondę emulacyjną można wykonać zaciskając złącze 34pin na taśmę i złącze ISV28. Nie należy przesadzać z długością taśmy - maksymalna długość nie powinna przekraczać 25cm. Przy zaciskaniu należy zwrócić uwagę, że pin 1 złącza 34P jest wolny. Szczegóły można zobaczyć na **rys. 4**.

Przy ewentualnych błędach pomocna może być instrukcja symulatora: *@mon*.

Po jej wysłaniu do czasu nadania dowolnego znaku w oknie cyklicznie będzie pojawiać się informacja:

Dana $DDDD %DDDDDDDD, Adres $AAAAXXXX %AAAAAAAAXXXX-XXXX,

gdzie *DDDD* - dana odczytana z układu US7, *AAAA* - adres odczytany z układu US8, *XXXX* - znaki x ponieważ nie można wyświetlić stanu linii adresowych A0..A7. Dzięki instrukcji *@mon* możemy niejako skanować magistralę danych i adresową. Teraz krótko scharakteryzuję wszystkie rozkazy (niewykluczone, że będzie ich więcej, dlatego uważnie przeczytajcie plik READ.ME na dyskietce):

@se30 - przyłączenie symulatora do magistrali RS

@2716 - wybór typu pamięci EPROM (tylko odczyt)

@2732 - wybór typu pamięci EPROM (tylko odczyt)

@2764 - wybór typu pamięci EPROM (tylko odczyt)

@27128 - wybór typu pamięci EPROM (tylko odczyt)

@27256 - wybór typu pamięci EPROM (tylko odczyt)

@27512 - wybór typu pamięci EPROM (tylko odczyt)

@2816 - wybór typu pamięci EEPROM (zapis/odczyt)

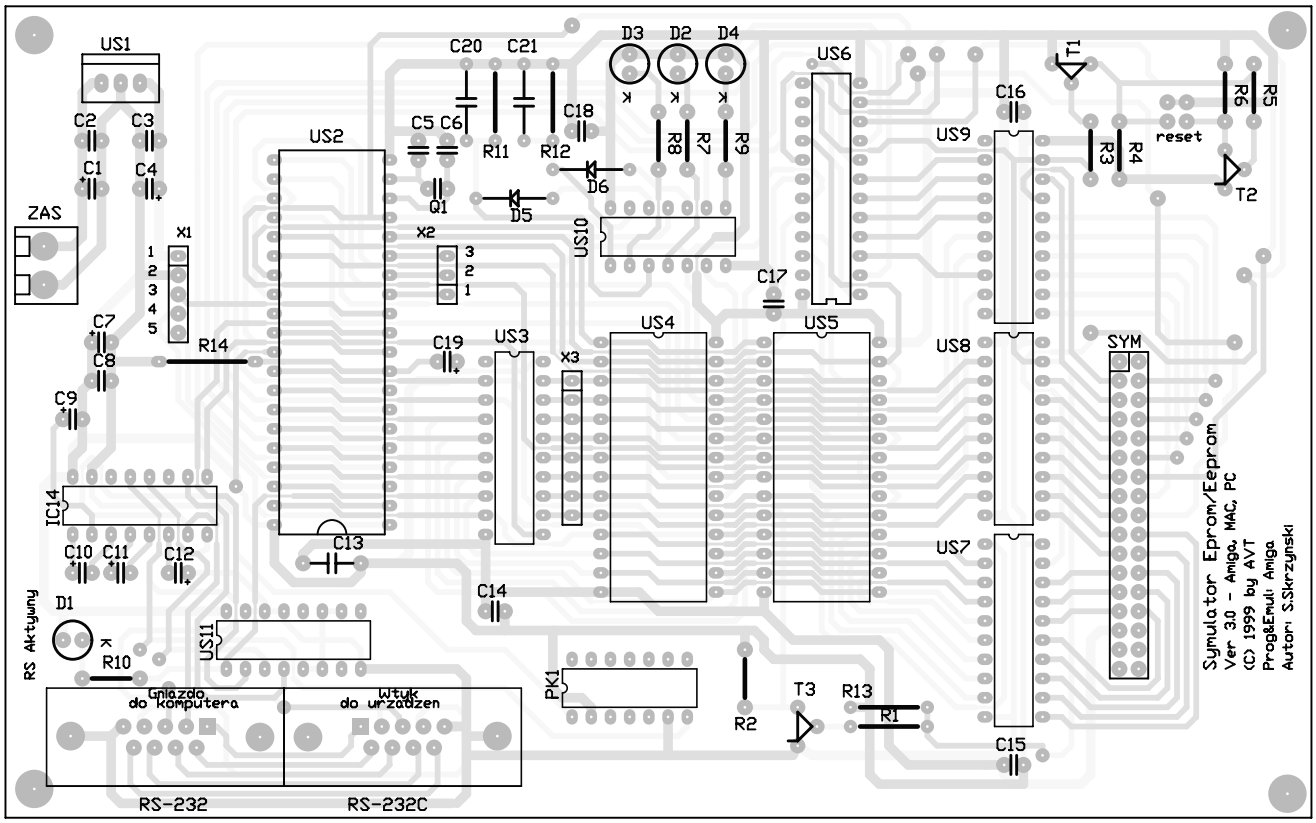
@2864 - wybór typu pamięci EEPROM (zapis/odczyt)

@28256 - wybór typu pamięci EEPROM (zapis/odczyt)

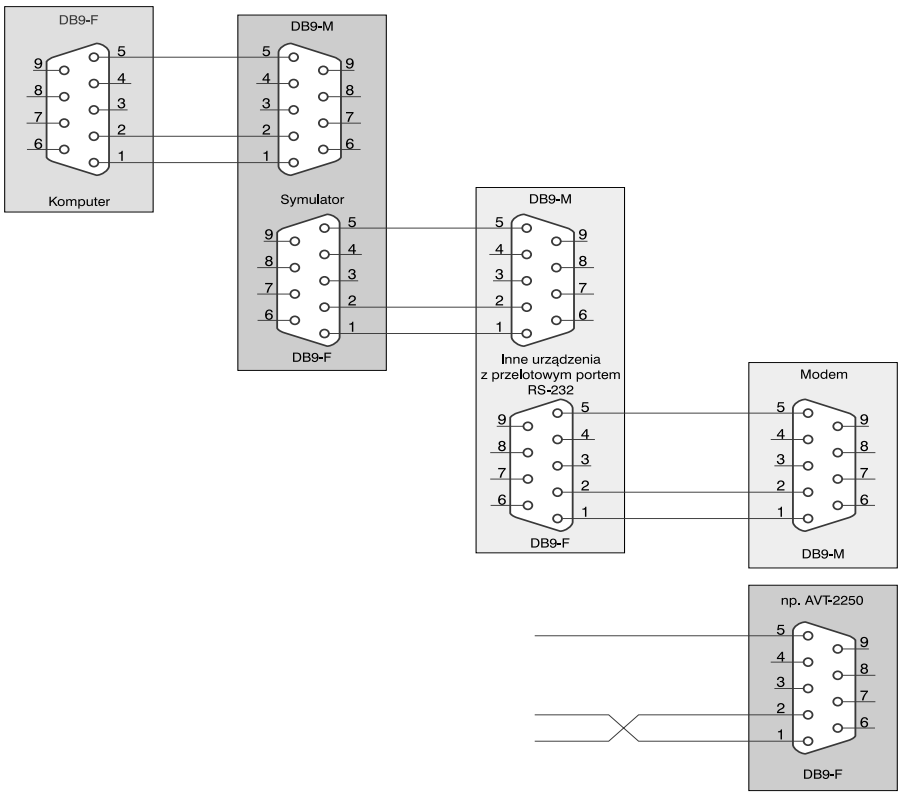
@reset - wysłanie sygnału zerującego do uruchamianego systemu o czasie trwania 0,5s.

@read $xxxx $yyyy - odczytanie obszaru od xxxx do yyyy. Dane w formacie IntelHex. Transmisję można przerwać wysyłając do emulatora znak kropki.

@offset $xxxx - ustawienie offsetu dla ładowanych plików. Obowiązuje ono do chwili odłączenia symulatora od szyny RS rozkazem *@end* lub automatycz-



Rys. 2. Rozmieszczenie elementów na płytce drukowanej.



Rys. 3. Sposób połączenia emulatora i urządzeń zewnętrznych.

nie po czasie 10 sekund. Po odłączeniu od RS offset ustawia się na $0000.

@baud 1200 - ustawienie nowej szybkości transmisji. Ustawienie obowiązuje do odłączenia symulatora od szyny RS rozkazem *@end* lub automatycznie po czasie 10 s.

@baud 2400 - ustawienie nowej szybkości transmisji. Ustawienie obowiązuje do odłączenia symulatora od szyny RS rozkazem *@end* lub automatycznie po czasie 10 s.

@baud 4800 - ustawienie nowej szybkości transmisji. Ustawienie obowiązuje do odłączenia symulatora od szyny RS rozkazem *@end* lub automatycznie po czasie 10 s.

@baud 9600 - ustawienie nowej szybkości transmisji. Ustawienie obowiązuje do odłączenia symulatora od szyny RS rozkazem *@end* lub automatycznie po czasie 10 s.

@baud 19200 - ustawienie nowej szybkości transmisji. Ustawienie obowiązuje do odłączenia symulatora od szyny RS rozkazem *@end* lub automatycznie po czasie 10 s.

@baud 28800 - ustawienie nowej szybkości transmisji. Ustawienie obowiązuje do odłączenia symulatora od szyny RS rozkazem *@end* lub automatycznie po czasie 10 s.

@baud 57600 - ustawienie nowej szybkości transmisji. Ustawienie obowiązuje do odłączenia symulatora od szyny RS rozkazem *@end* lub automatycznie po czasie 10 s.

#SSSSEEEEDDDD...DD - plik binarny, gdzie:

SSSS - adres początku obszaru do zapisu,

EEEE - adres końca obszaru do zapisu,

DD - dane w liczbie EEEE-SSSS, w SSSS i EEEE starszy bajt jako pierwszy

:LLAAAATTDDDD...DDSS - ładowanie pliku w formacie IntelHex, gdzie:

LL - liczba bajtów danych,

AAAA - adres zapisu danych,

TT - typ danych (tu zawsze 00 lub 01),

DD - dane w liczbie LL,

SS - suma kontrona (w AAAA starszy bajt jako pierwszy).

@end - odłączenie symulatora od magistrali RS.

Warto zaznaczyć, że wielkość znaków ma znaczenie w wypadku rozkazów, natomiast w plikach IntelHex wielkość znaków jest ignorowana.

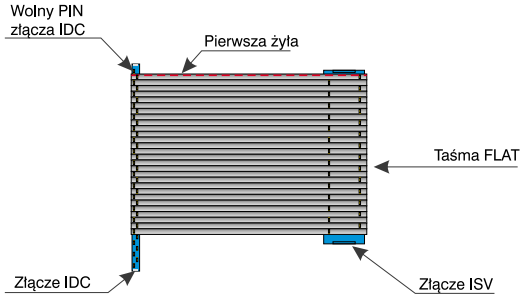
Rozkaz *@offset* jest przydatny podczas emulowania ROM-u dla procesorów, w których przestrzeń adresowa dla pamięci programu zaczyna się od adresu różnego od $0000. Gdy np. emulujemy pamięć 27128 procesora, dla którego pamięć programu zaczyna się od $C000 offset należy ustawić na $4000 (suma $C000 i $4000 = $10000). Dla np emulacji 27256 dla procesora, którego pamięć programu zaczyna się od $8000 offset ustawiamy na $8000 (suma $8000 i $8000 = $10000), dla innych wartości posrępujemy analogicznie).

Rozkaz *@mon* może być przydatny podczas uruchamiania systemów mikroprocesorowych. Umożliwia on „podglądanie" szyny adresowej i danych, co może być przydatne zwłaszcza podczas pracy krokowej. Jeśli będzie zapotrzebowanie na oglądanie całej linii adresowej proszę o listy. Procesor ma kilka wolnych linii portów, co umożliwi dobudowanie układu odczytującego młodszą część adresu.

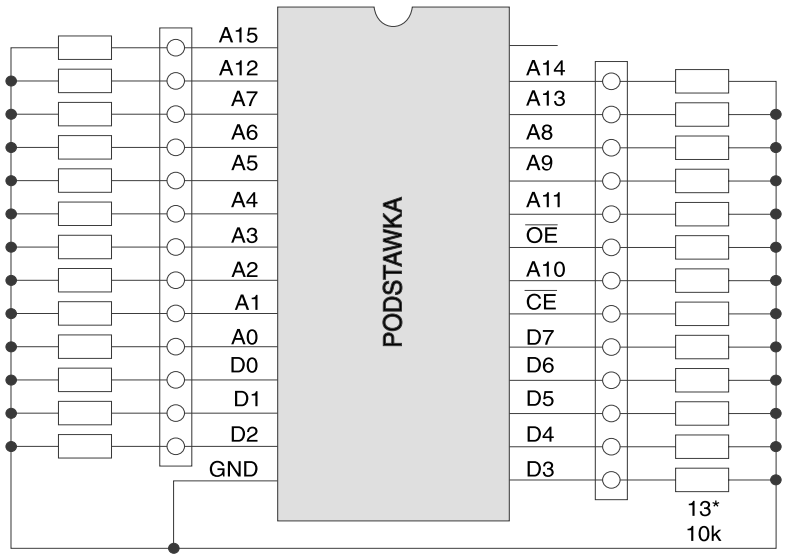
Co zrobić gdy nie działa?

Jeśli montaż przeprowadzony zostanie prawidłowo nie powinno być z tym żadnych kłopotów. Ale jeśli już mamy błąd to pomocny będzie układ z **rys. 5** i rozkaz *@mon*.

W podstawce umieśćmy sondę emulacyjną, rezystory wymuszają poziom niski na wszystkich wyprowadzeniach. Wczytujemy do symulatora program testujący magistralę adresową (dostępna wersja źródłowa i IntelHex) uruchamiając program „T_MagAdr.BAT"



Rys. 4. Sposób wykonania kabla emulującego.



Rys. 5. Pomocniczy układ testowy.

(dla Amigi „T_MagAdr.skrypt“). Uruchamiamy program terminala, wydajemy rozkazy:

```
@se30
@mon
```

W oknie programu pojawi się stan szyny danych i adresowej. Wymuszamy poziom wysoki na kolejnych liniach adresowych. Powinniśmy uzyskać wyniki zgodnie z **tab. 1**. Po naciśnięciu dowolnego klawisza, procedura zostanie przerwana.

Drugi test sprawdzający magistralę adresową („T_MagDat.BAT“ dla Amigi „T_MagDat.skrypt“) wywołuje efekty pokazane w **tab. 2**.

Dzięki temu testowi możemy w łatwy sposób sprawdzić poprawność sygnałów na złączu emulacyjnym. Możemy też sprawdzić, przełączając symulator w tryb @2716, jak są ignowane linie adresowe A12-A15, czy po przełączeniu w tryb @2864, @2816, że linia A14, czy A11 staje się linią sterującą zapisem.

Jak wspomiałem na wstępie, możliwe jest podłączenie symulatora do AVT-2250. Wystarczy połączyć urządzenia kabelkiem. Trzeba jednak napisać program, który wyśle do symulatora tekst: `@se30@2716` (lub inna pamięć) i kod `return`. Aby zapewnić maksymalną uniwersalność jako kod return symulator akceptuje następujące kody: `$0D` (kod CR), `$0A` (kod LF) i `$21` (kod znaku wykrzyknika). Zapytacie po co wykrzyknik? Ułatwił on pisanie skryptów. Co nam potrzebne już wiemy, a jak tego używać? To proste. Najpierw uruchamiamy program wysyłający tekst: `@se30@2716`, następnie naciskamy klawisz 8 (SEND) na komputerku AVT2051, wpisujemy adresy, zatwierdzamy przez OK i już. Nie ma konieczności pisania programu, który odłączy symulator od magistrli RS, ponieważ nastąpi to automatycznie po 10 sekundach. I tu uwaga. Na wpisanie adresów mamy dziesięć sekund (ale to dużo czasu).

Jeśli wystarczy nam 32KB pamięci, możemy nie montować układu US5. Symulator będzie się wtedy zgłaszał:

```
Symulator Eprom V3.0-32KB
(C) 1999 By AVT-Korporacja
    Autor: S.Skrzynski
  Prog&Emul: Amiga
```

Komputery klasy PC przy wykonywaniu rozkazu COPY na urządzenie COMx wymagają sprzętowego potwierdzenia transmisji (wystarczy, aby linie RTS i CTS, oraz DSR i DTR były ze sobą połączone). Dlatego w ostatnim urządzeniu z przelotowym portem RS do wyjścia należy podłączyć wtyczkę z połączeniami zgodnie z **rys. 6**.

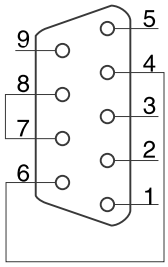
Z tego powodu mogą być problemy przy współpracy z np modemami. Aby rozkaz *Copy* został prawidłowo wykonany, modem musi być włączony.

Rozbudowa

Jeśli komuś bardzo zależy, może symulator wyposażyć w wyświetlacz typu emulowanej pamięci. Zasada działania jest bajecznie prosta (**rys. 7**). Demultiplekser dekoduje stan na wejściach A, B, C, D układu GAL na świecenie jednej z LED. Układ wyświetlacza można zamontować na uniwersalnej płytce drukowanej. Na dyskietce dostarczanej z kitem znajdują się dwa katalogi i plik: AMIGA, PC, READ.ME. Znajdują się tam przykładowe skrypty (dla PC pliki .BAT) przesyłające dane do symulatora, komputerka AVT-2250, kompilatory 6502, 8051, Z80, programy źródłowe, pliki w formacie IntelHex, itp. Dane dla Amigi zdecydowano zapisać w formacie MS-DOS, aby nie trzeba było osobnych dyskietek dla każdego komputera. Dla posiadaczy „małych“ Amig może być problem z odczytaniem dyskietki (1,44MB). Aby odczytać dyskietkę na Amidze należy pamiętać o uruchomieniu drivera PC0: (u mnie znajduje się w Devs/DOSDrivers i zawsze jest aktywny). Programy na dysku mają status freeware. Programy można kopiować w celach nieko-

Tab. 2.

Linia z wymuszonym poziomem wysokim	Efekt na ekranie
wszystkie=L	$ca %00000000, $00xx %00000000xxxxxxxx, RD
A0=H	$01 %00000001, $00xx %00000000xxxxxxxx, RD
A1=H	$02 %00000010, $00xx %00000000xxxxxxxx, RD
A2=H	$04 %00000100, $00xx %00000000xxxxxxxx, RD
A3=H	$08 %00001000, $00xx %00000000xxxxxxxx, RD
A4=H	$10 %00010000, $00xx %00000000xxxxxxxx, RD
A5=H	$20 %00100000, $00xx %00000000xxxxxxxx, RD
A6=H	$40 %01000000, $00xx %00000000xxxxxxxx, RD
A7=H	$80 %10000000, $00xx %00000000xxxxxxxx, RD
inne kombinacje	$ff %11111111, $??xx %????????xxxxxxxx,??



Rys. 6. Niezbędne zworki na złączu RS232.

mercyjnych. Nie można bez zgody zmieniać zawartości pakietu. Nie ma sensu marnować miejsca na opis zawartości dyskietki, rozpakowania, itp. Najważniejsze informacje można znaleźć w zbiorze READ.ME.

Symulator można podłączyć także do C-64. Aby to zrobić należy wykonać interfejs konwertujący sygnały RS z poziomów TTL na ±12V. W skład interfejs wchodzą dwa scalaczki (MC1488 i MC1489) dwa złącza (USER i DB25) oraz kilka kondensatorów i rezystorów.

Jeśli macie jakieś uwagi, propozycje, do symulatora i innych urządzeń (co byście powiedzieli na programator EPROM/EEPROM 2kB..1MB, procesorów serii 8051, seregowych EEPROM) z przelotowym portem RS proszę o listy (pocztą lub e-mail-em na adres redakcji z dopiskiem „S. Skrzyński" (nie może być S. S. bo myliłoby się z Sławomirem Surowińskim).

Różnice

Symulator widziany przez mikroprocesor różni się od prawdziwego EPROM/EEPROM kilkoma cechami:

- Większa obciążalność wyjść symulatora dzięki buforom 74HCT245 od rzeczywistej Eprom.
- Krótszy czas dostępu do pamięci symulatora (100-150ns zależnie od szybkości GAL i RAM) w porównaniu z eprom (200ns).
- Duża obciążalność dynamiczna wejść adresowych i sterujących spowodowana długimi przewodami łączącymi sondę emulacyjną z symulatorem.
- Symulowana EEPROM zachowuje się jak NVRAM (RAM podtrzymywana bateryjnie) i zapis bajtu trwa około 150ns, a nie 10ms.
- Pobór prądu przez symulowany eprom z szyny Vcc jest = 0mA.Wynika to z tego, że symulator jest zasilany z zewnętrznego zasilacza.
- Warto zauważyć, że pamięci EEPROM mają wyprowadzenia zgodnie z RAM. Pamięć 2864 w przeciwieństwie do 6264 nie posiada wejścia CS2. Dlatego dla 6264 spełniona jest zależność: „układ 6264 aktywny gdy: CE1=L, CE2=x, WR lub RD=L".

Nie można było w GAL-u zaprogramować tej zależności, ponieważ mogłoby się zdarzyć, że linia A13 (CS2 w RAM6264, w 2864 wolne) będzie połączona do poziomu niskiego i układ 2864 nie będzie aktywowany. Nie powinno być kłopotów z 6264, ponieważ w 99% przypadków CS2 jest na stałe połączone z poziomem wysokim. W GAL-u można zaprogramować zależności prawdziwe dla 6264 (ponieważ A13 dochodzi do GAL-a), ale mamy kompromis: „bardziej prawdziwy" EEPROM czy RAM?

- Warto też wspomnieć o zależności:
  zapis do RAM także gdy: CE=L, WR=L i RD=L

a dla EEPROM powinno być:
  zapis do układu gdy CE=L, WR=L i RD=H

Jest to zabezpieczenie, aby nie było fałszywych zapisów do EEPROM (np. podczas włączania zasilania). I znów kompromis. Skoro jednak symulator EPROM i EEPROM to na nieścisłości podczas emulowania RAM można przymknąć oko.

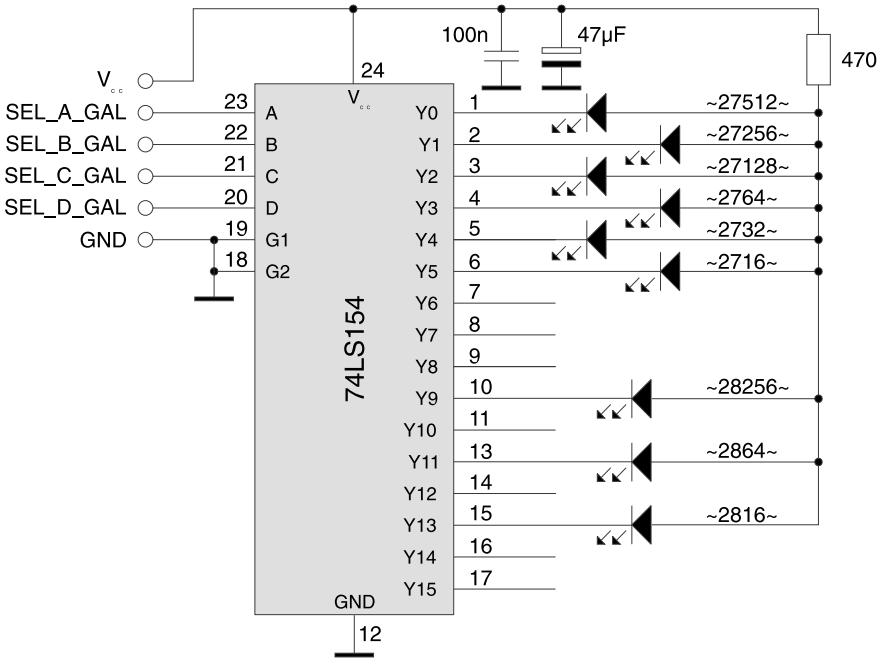
- Nie jest emulowane wyjście READY układu 2864 (ale, przeważnie stosuje się przeglądanie DATA POLLING).

UWAGA! Symulator można uszkodzić, jeśli na wyprowadzenia złącza emulacyjnego doprowadzimy napięcia większe niż +5V. Jeśli nie będziemy podłączać symulatora do programatora EPROM nic nie powinno się stać (bardzo rzadkie są przypadki, aby możliwe było programowanie EPROM w działającym urządzeniu, wyjątkiem jest kit AVT-112). Z tego powodu nie działają sygnatury EPROM i nie można odczytać bajtów ID użytkownika w EEPROM, nie działa funkcja CHIP CLEAR i stosowane w niektórych pamięciach EEPROM programowe zabezpieczenie przed zapisem.

Układy TTL powinny być z serii 74HCxx, 74HCTxx lub ostatecznie 74LSxx.

Sławomir Skrzyński
skrzynski@zt.wloclawek.tpsa.pl

Wzory płytek drukowanych w formacie PDF są dostępne w Internecie pod adresem: ***http://www.ep.com.pl/pcb.html*** *oraz na płycie CD-EP07/2000B w katalogu* ***PCB****.*



Rys. 7. Wskaźnik typu emulowanej pamięci.