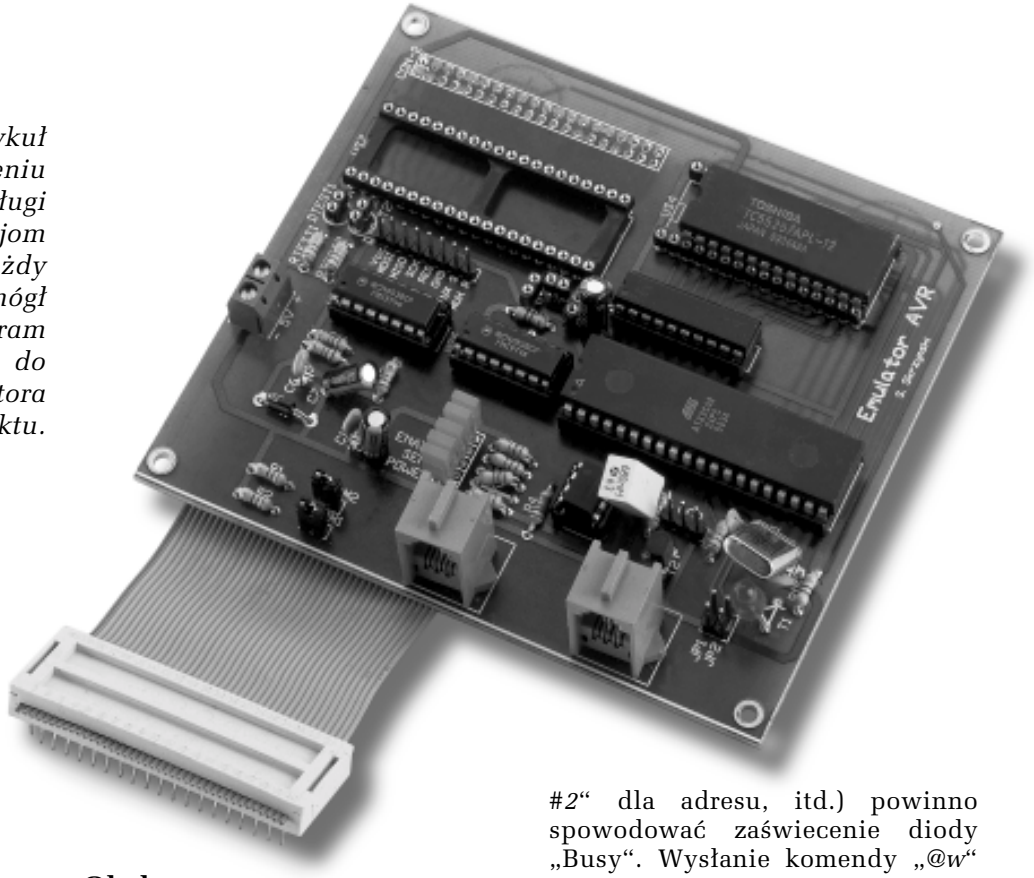
Emulator-programator mikrokontrolerów AVR i ’51 do każdego typu komputera, część 2

AVT-5037

W drugiej części artykuł skupiamy się na przybliżeniu sposobu i języka obsługi emulatora. Dzięki informacjom zawartym w artykule każdy programista będzie mógł samodzielnie stworzyć program obsługi alternatywny do przygotowanego przez autora projektu.



Obsługa

Opis będzie dotyczył zarówno nowej, jak i starej (AVT-995) wersji emulatora. Różnice pomiędzy modelami będą wyszczególnione w tekście. Ze względu na zmianę idei obsługi przyrządu użytkownicy AVT-995 będą musieli używać nowych wersji skryptów (plików wsadowych) przeznaczonych dla AVT-995.

Po dołączeniu emulatora do komputera, można go przetestować. Podłączamy emulator do portu RS komputera i uruchamiamy program terminalowy. Ustawiamy parametry transmisji: 4800bd/8N1. Wysłanie przez RS komendy „*@emu avr&51 #0*“ („*@emu avr&51 #1*“ dla adresu 1, „*@emu avr&51 #2*“ dla adresu, itd.) powinno spowodować zaświecenie diody „Busy“. Wysłanie komendy „*@w*“ powinno spowodować miganie diody „Busy“. Po 10 sekundach powinna zaświecić się dioda „Error“ informująca o przekroczeniu czasu oczekiwania.

Symulator rozpoznaje następujące komendy:

@emu avr&51 #0 - przyłączenie emulatora do magistrali RS485 (LED „Busy“ świeci);

@A - ustawienie procesora rodziny AVR - po tej komendzie emulator określi typ procesora, wielkość pamięci FLASH i EEPROM;

@5 - ustawienie procesora rodziny 8051 - po tej komendzie emulator ustawi wielkość pamięci FLASH na $2FFF (max. dla AT89S53) i EEPROM na $07FF (max dla AT89S8252);

@C - czyszczenie pamięci procesora;
@e - odłączenie emulatora od magistrali RS485 (LED „Busy" gaśnie);
@r - wysłanie sygnału zerującego do uruchamianego systemu;
@w - uruchomienie trybu programowania procesora - po tej komendzie emulator czeka na plik IntelHex; przed programowaniem AVR automatycznie jest czyszczona pamięć FLASH (opcja automatycznego czyszczenia nie działa dla AVT-995!);
@L1 - ustawienie bitu zabezpieczającego 1;
@L2 - ustawienie bitu zabezpieczającego 1 i 2;
@L3 - ustawienie bitu zabezpieczającego 1, 2 i 3;
@b24 - zmiana prędkości transmisji na 2400bd (opcja niedostępna dla AVT-995);
@b48 - zmiana prędkości transmisji na 4800bd (opcja niedostępna dla AVT-995);
@b96 - zmiana prędkości transmisji na 9600bd (opcja niedostępna dla AVT-995);
@b19 - zmiana prędkości transmisji na 19200bd (opcja niedostępna dla AVT-995);
@b28 - zmiana prędkości transmisji na 28800bd (opcja niedostępna dla AVT-995);
@b57 - zmiana prędkości transmisji na 57600bd (opcja niedostępna dla AVT-995).

Komenda *@5* lub *@A* musi być wysłana przed komendami operującymi na procesorze (np. *@C*, *@r @L1*, *@w*).

Zmiana prędkości transmisji obowiązuje do czasu odłączenia emulatora komendą *@e* lub automatycznie po 10 sekundach nieaktywności na magistrali. Wtedy jest ustawiana standardowa prędkość transmisji 4800bd (dla AVT-995 można wybrać zworką SLOW_RS prędkość 2400bd).

Na dyskietce dołączonej do zestawu znajdują się skrypty i pliki dla Amigi i PC. Pakiety są dość rozbudowane. Zawierają kompilatory wielu procesorów (między innymi 65xx, 68xx, 680xx, 80xx, AVR, Z80) oraz skrypty obsługujące kity AVT-2250 (komputerek edukacyjny z 8051), AVT-498 (programator-emulator AT89Cx051), AVT-870 (symulator EPROM). O sposobie wykorzystania skryptów można przeczytać w instrukcji pakietu. Zaleca się użycie instalera do instalacji lub upgrade pakietu. Niżej przedstawiono ich zawartość dla Amigi i PC.

Amiga:
katalog *AVR* - kompilator, dokumentacja i skrypty dla AVR;
skrypt *AVR/AVR_EmuAVR.rexx* - kompilacja kodu dla AVR i wysłanie do emulatora;
skrypt *AVR/AVR_AVT995+.rexx* - kompilacja kodu dla AVR i wysłanie do emulatora AVT995+;
katalog *8051* - kompilator, dokumentacja i skrypty dla 8051;
skrypt *8051/AVR_EmuAVR.rexx* - kompilacja kodu dla 8051 i wysłanie do emulatora;
skrypt *8051/AVR_AVT995+.rexx* - kompilacja kodu dla 8051 i wysłanie do emulatora AVT995+;
katalog *INCLUDE* - definicje rejestrów itp.;
katalog *AVT_EmuAVR* - pliki tekstowe - sterowanie emulatorem;
skrypt *AVT_EmuAVR/Clear* - czyszczenie pamięci procesora;
skrypt *AVT_EmuAVR/Lock1* - ustawienie bitu blokady 1;
skrypt *AVT_EmuAVR/Lock2* - ustawienie bitów blokady 1 i 2;
skrypt *AVT_EmuAVR/Lock3* - ustawienie bitów blokady 1, 2 i 3;
skrypt *AVT_EmuAVR/Reset* - wysyła sygnał zerujący do emulowanego CPU.

PC:
katalog *EXE* - kompilatory i dokumentacje;
katalog *TENT* - pliki tekstowe - sterowanie emulatorem;
katalog *INCL* - definicje rejestrów itp.;
skrypt *Clear.bat* - czyszczenie pamięci procesora;
skrypt *Lock1.bat* - ustawienie bitu blokady 1;
skrypt *Lock2.bat* - ustawienie bitów blokady 1 i 2;
skrypt *Lock3.bat* - ustawienie bitów blokady 1, 2 i 3;
skrypt *Reset.bat* - wysyła sygnał zerujący do emulowanego CPU;
skrypt *51.bat* - kompilacja kodu dla 8051 i wysłanie do emulatora;
skrypt *51_995.bat* - kompilacja kodu dla 8051 i wysłanie do emulatora AVT995+;
skrypt *AVR.bat* - kompilacja kodu dla AVR i wysłanie do emulatora;
skrypt *AVR_995.bat* - kompilacja kodu dla AVR i wysłanie do emulatora AVT995+.

Aby skompilować kod programu źródłowego, należy wydać komendę „51.bat {nazwa pliku do kompilacji} [numer portu szeregowego]" lub „AVR.bat [nazwa pliku do kompilacji] [numer portu szeregowego]". O szczegółach można dowiedzieć się, czytając komentarze w skryptach.

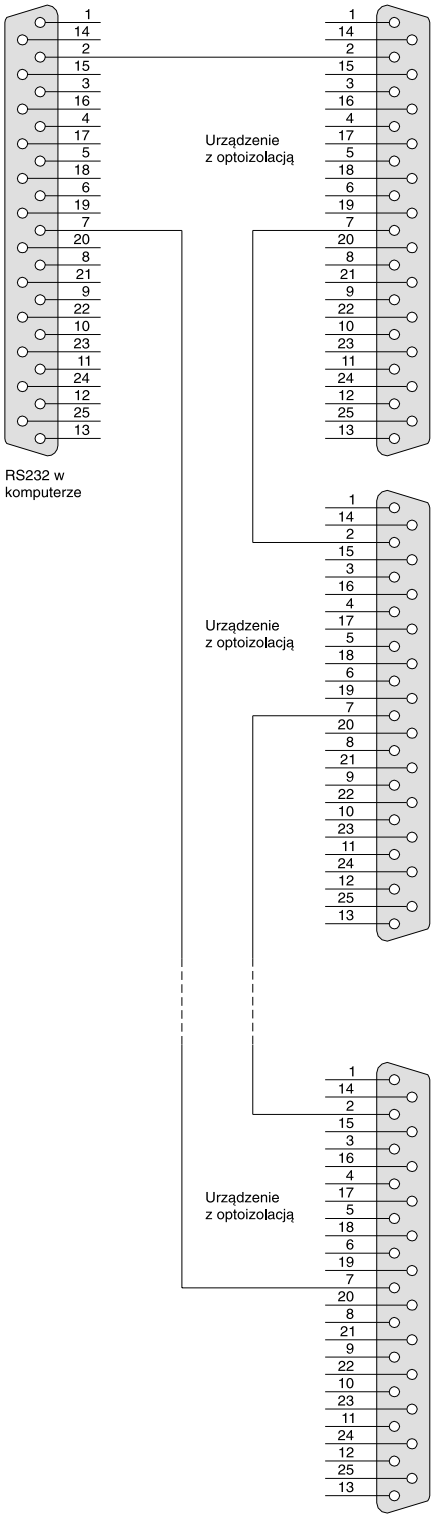
Ważne! Skrypty dla emulatora AVT-995 różnią się nieznacznie od skryptów dla nowej wersji. Spowodowane jest to tym, że AVT-995 nie czyści pamięci AVR przed programowaniem i jest konieczne wykonanie tej operacji przez wydanie odpowiedniego rozkazu w skrypcie oraz tym, że w AVT-995 nie można użyć większej prędkości transmisji.

W skryptach prędkość transmisji jest zwiększana do 19200bd, co gwarantuje poprawną współpracę ze wszystkimi transoptorami. Jeśli korzystamy z RS485 lub mamy szybki transoptor, prędkość można zwiększyć. Aby tego dokonać, należy zmienić we wszystkich miejscach tekst „19200" na np. „57500". Gdyby w czasie transmisji pojawiał się często błąd „Przepełniony bufor RS", prędkość transmisji należy zmniejszyć.

W skryptach dla PC, w celu uzyskania opóźnienia, użyto rozkazu PAUSE. Powoduje on zawieszenie wykonywania programu do czasu naciśnięcia dowolnego klawisza. Musi to nastąpić w ciągu 10 sekund, w przeciwnym przypadku emulator zakomunikuje błąd. W kolejnych wersjach skryptów dodam rozkaz zawieszający działanie komputera na 1 sekundę.

Jeśli podłączymy kilka urządzeń do portu RS485, należy pamiętać, aby zworki TERM były zamontowane na tym, które jest ostatnie (do którego dochodzi jeden kabel).

Jeśli podłączymy kilka urządzeń do portu RS232C z optoizolacją, zalecane jest łączenie LED transoptorów szeregowo (**rys. 8**). Oczywiście nie można przesadzać



Rys. 8. Sposób dołączenia do interfejsu RS232 kilku urządzeń z optoizolowanym wejściem.

z liczbą portów, ponieważ spadek napięcia na diodach transoptorów może przekroczyć napięcie zasilania lub zbytnio zmniejszy się prąd transoptorów (to zależy od konstrukcji urządzenia). W praktyce można podłączyć 3..4 urządzeń z optoizolacją. Jeśli potrzebujemy ich więcej, można spróbować połączenia mieszanego (szeregowo-równoległego) lub należy zbudować wzmacniacz prądowy na tranzystorze i wszystkie urządzenia połączyć równolegle.

Opis stanów urządzenia

Tryb emulacji - oznacza stan spoczynkowy programatora - procesor IC4 jest podłączony do złącza emulacyjnego.

Połączenie z RS485 - oznacza, że emulator został prawidłowo zaadresowany i oczekuje na rozkazy.

Zajętość emulatora - Może oznaczać transmisję danych (w tym czasie miga dioda „Data"), czyszczenie pamięci procesora lub wysyłanie sygnału zerującego do uruchamianego systemu.

Przekroczony adres - Przekroczono pojemność pamięci programu lub danych dla danego typu procesora.

Przekroczono czas oczekiwania - nie otrzymano wymaganych danych w ciągu 10 sekund.

Błąd pliku IntelHex - może oznaczać:
- błędną sumę kontrolną rekordu pliku IntelHex,
- niewykrycie początku rekordu pliku IntelHex (znak „:"),
- zły typ nagłówka (powinien być $00 - dane lub $01 koniec pliku),
- błąd sumy kontrolnej - najbardziej prawdopodobna przyczyna wystąpienia błędu to wysłanie pliku w formacie innym niż IntelHex.

Błąd zapisu bajtu do procesora - nie można zapisać bajtu do procesora. Weryfikacja przez 10ms nie jest poprawna. Przyczyną pojawienia się błędu może być:
- ustawione bity zabezpieczające,
- nieodpowiedni typ procesora dla użytego skryptu (kompilowano dla AVR, a US7 jest z rodziny 8051 lub na odwrót),
- brak procesora w podstawce lub uruchamianym systemie,
- brak połączenia emulatora z procesorem w uruchamianym systemie,
- brak oscylacji generatora zegarowego w programowanym procesorze lub za niska częstotliwość generatora,
- zakłócenia na liniach SPI.

W przypadku wystąpienia pierwszego błędu wystarczy skasować

WYKAZ ELEMENTÓW

Płyta główna

Rezystory
R1, R2: 240Ω
R3, R5..R8, R14, R16: 1kΩ
R4, R9..R12: 10kΩ
R13, R15: 470Ω
RTEST1, RTEST2: 470Ω (opcja)

Kondensatory
C1: 1µF
C2, C3: 27pF
C4, C5: 27pF (opcja)
C6, C9: 100nF
C7, C8, C11: 10µF

Półprzewodniki
D1: 1N4007
D2: 1N4148
DTEST1, DTEST2: LED (opcja)
US1: 74176 (MAX485)
US2: 89C52
US3: 74HCT573
US4: 62256 lub RAM-128KB
US5, US6: 4053
US7: emulowany procesor
US8: CNY-17 lub po modyfikacji 6N137

Różne
Q1: 11.0592Mhz
Q2: opcja (zależny od US7)
CON1: ARK-2
CON2: DB9pin-F
CON3, CON4: gniazdo telefoniczne 6p4c do druku
CON5: goldpin 3 piny
CON7: goldpin 2 piny
CON-2: IDC40

Płyta interfejsu RS-485

Rezystory
R1..R3: 1kΩ
R4: 10kΩ
R5, R7: 240Ω
R6: 4,7kΩ

Kondensatory
C1, C4..C8: 10µF
C2, C3: 100nF

Półprzewodniki
T1: BC308
US1: 78L05
US2: MAX232 (ICL232)
US3: 75176 (MAX485)

Różne
CON1: ARK-2
CON2: DB25pin-M
CON3: gniazdo telefoniczne 6p4c do druku

pamięć procesora. W przypadku ostatniego błędu najczęściej pomaga ponowne wysłanie pliku do emulatora.

Przepełniony bufor odbiorczy RS - błąd może się pojawić, gdy bufor jest prawie pełny i są problemy z zapisem bajtu do procesora. Nastąpi wtedy szybkie zapełnienie bufora. Jeśli wykorzystujemy zwiększoną prędkość transmisji, należy ją zmniejszyć. Czasem pomaga ponowienie transmisji.

Przepełnienie stosu - ten błąd nie powinien się pojawić. Funkcja użyteczna podczas pisania oprogramowania emulatora. W razie wystąpienia błędu proszę o kontakt z autorem.

Uwaga! Każdy błąd jest wskazywany przez 5 sekund po zakończeniu transmisji z komputera. Przez ten czas emulator nie reaguje na wysyłane do niego rozkazy i pliki.

Pisanie programów

Programator umożliwia zapisywanie pamięci danych i pamięci programu. Pierwsze wystąpienie adresu $0000 (rozkaz kompilatora *org $0000*) spowoduje, że dane za nim zawarte zostaną zapisane do pamięci programu. Kolejne wystąpienie adresu $0000 spowoduje, że dane te zostaną zapisane do pamięci danych. Najłatwiej to zrozumieć analizując poniższy przykład:

```
org$0000
;jeśli nie użyjemy dyrektywy
;to kompilator domyślnie
;przyjmie adres $0000
start:nop
;treść programu głównego
 mov dptr,#2  ;adres w EEPROM
 mov a,#$af   ;dana do zapisu
 acall write_eeprom
 mov a,#$e3   ;kolejny bajt do
              ;zapisu
 acall write_eeprom

org$0000
 STRING"ala ma kota"
 ;te dane zostaną
 ;zaprogramowane w pamięci
 ;EEPROM
```

Sławomir Skrzyński, AVT
slawomir.skrzynski@ep.com.pl

Przy uruchamianiu emulatora wykorzystano zestawy AVT-995 i AVT-498 współpracujące z Amigą. Do zaprogramowania procesorów w prototypie wykorzystano programator AVT-996.

Dyskietka dostarczana wraz zestawem zawiera programy dla PC i Amigi. Najnowsze wersje oprogramowania dla Amigi PC będą dostępne na stronie internetowej EP.

Płytka „przejściówki" dla procesorów 8/20pin nie wchodzi w skład kitu.

Wzory płytek drukowanych w formacie PDF są dostępne w Internecie pod adresem: ***http://www.ep.com.pl/?pdf/listopad01.htm*** *oraz na płycie CD-EP11/2001B w katalogu* ***PCB****.*