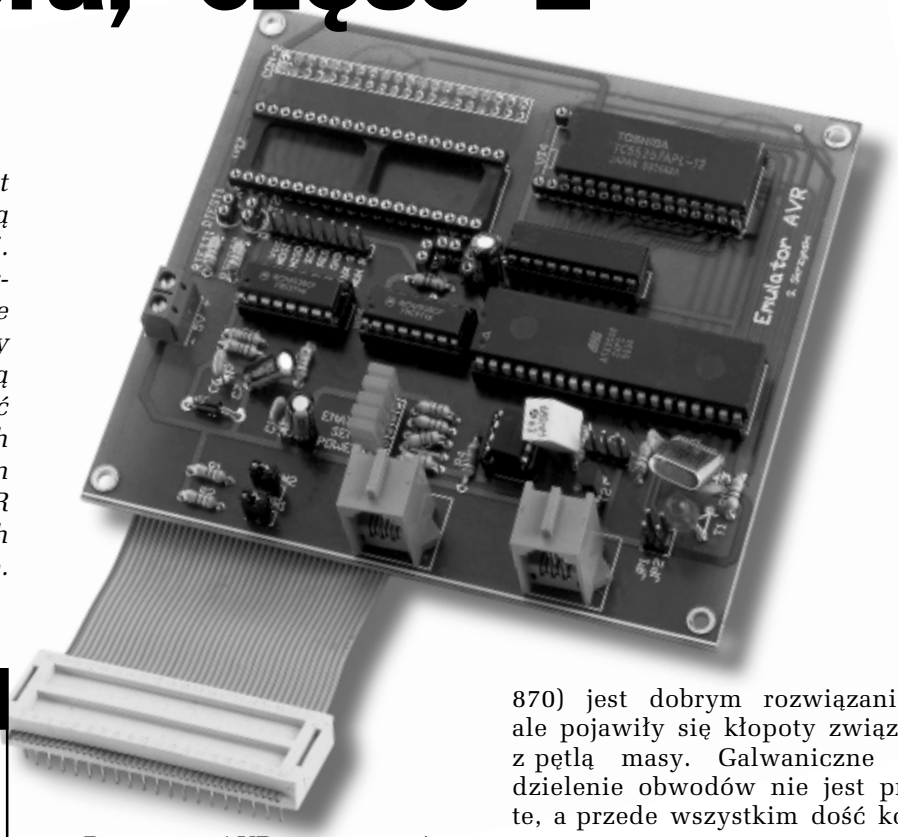
Emulator-programator mikrokontrolerów AVR i ’51 do każdego typu komputera, część 1

AVT-5037

Kolejny projekt przygotowany z myślą o użytkownikach Amigi. Prezentowany emulator-programator doskonale spisuje się także we współpracy z innymi komputerami. Jedną z jego zalet jest możliwość emulacji wybranych mikrokontrolerów z rodzin AVR (w tym także AVR Mega) i ’51, a także ich programowanie.



Najważniejsze cechy emulatora-programatora:

- ✓ zasilanie z uruchamianego systemu,
- ✓ współpraca z każdym komputerem wyposażonym w port RS232C od 2400...57600bd,
- ✓ interfejs RS485 lub RS232C z izolacją galwaniczną,
- ✓ obsługa danych w formacie IntelHex standard (adres 16-bit) oraz IntelHex 20-bit: koniec linii CR (MAC, C-64), LF (Amiga) lub CR+LF (PC), długość rekordu do 255 bajtów,
- ✓ emulacja: 8051, 8052, 89S8252, 89S53 (AT89Cx051 z ograniczeniem) oraz AVR w obudowach z 40 wyprowadzeniami (8/20-pinowe z ograniczeniami),
- ✓ programowanie: 89S8252, 89S53, AVR (także w pracującym urządzeniu przez SPI),
- ✓ programowanie bitów blokady 1, 2 i 3,
- ✓ programowanie pamięci danych i programu,
- ✓ sygnalizacja trybu pracy i błędów,
- ✓ możliwość podłączenia czterech symulatorów do jednego portu RS.

Przykładowe czasy programowania układów:

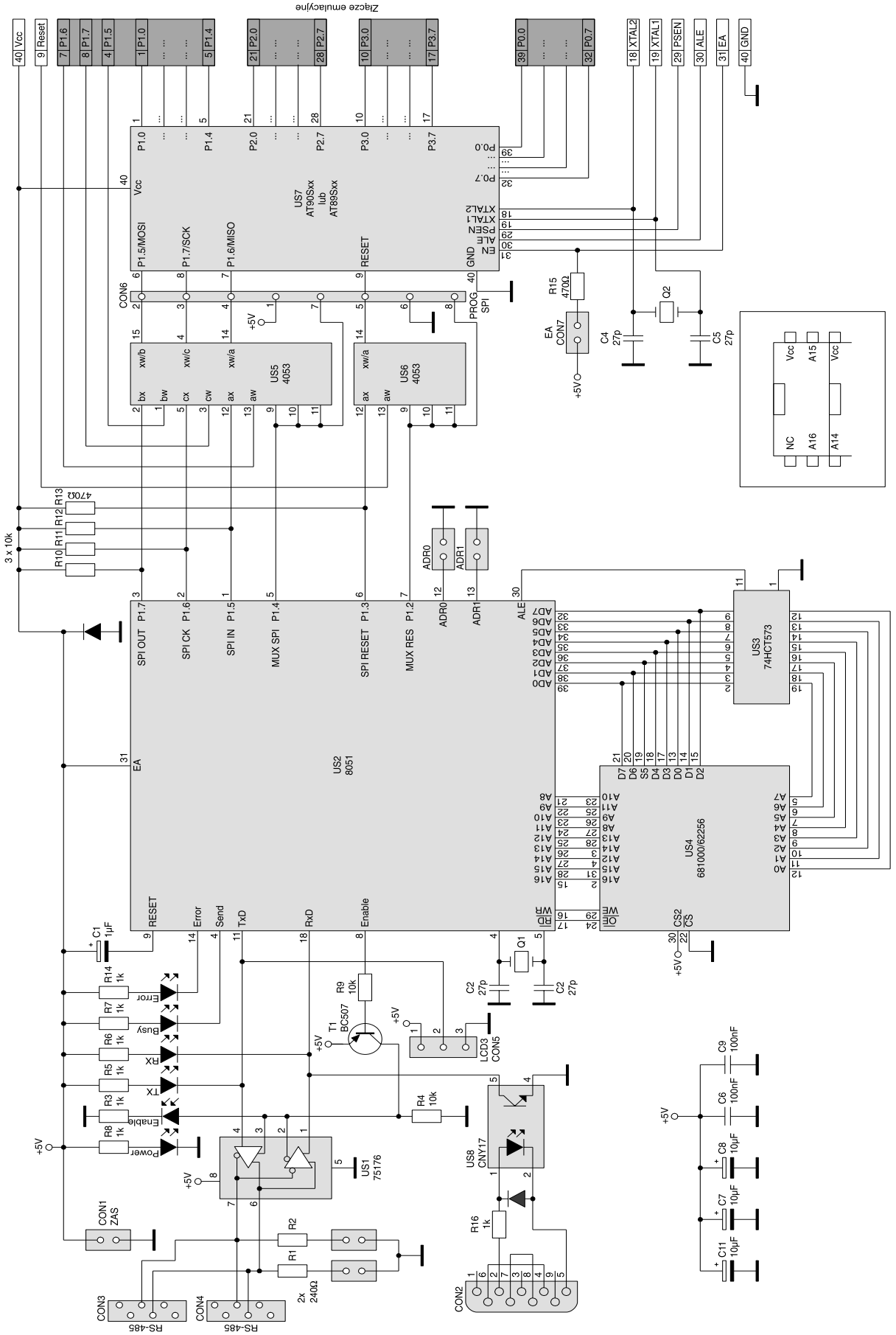
- ✗ AT89S8252 ..... 42s (250B/s),
- ✗ AT89S53 ..... 63s (250B/s),
- ✗ AT90S8515 ..... 30s (250B/s),
- ✗ ATMega161 ..... 4s (4kB/s),
- ✗ ATMega106 ..... 8s (8kB/s),
- ✗ ATMega103 ..... 16s (8kB/s),

Procesory AVR są wyposażane w pamięć programu o pojemności do 128kB (64k słów). Standardowy plik w formacie IntelHex umożliwia obsługę pamięci o pojemności do 64kB. Po przekroczeniu tej granicy konieczne jest zastosowanie pliku o zmodyfikowanym sposobie adresowania, co wymaga wczytania całego rekordu do pamięci programatora-emulatora i jego analizę. Poza tym procesory serii AVR Mega posiadają wewnętrzny bufor na programowane dane o wielkości 128 lub 256B, dzięki któremu znacznie skraca się czas programowania procesora.

Podczas konstruowania emulatora miałem pewne kłopoty z wyborem interfejsu łączącego go z komputerem. Zastosowanie przelotowego portu RS232 (jak w AVT-870) jest dobrym rozwiązaniem, ale pojawiły się kłopoty związane z pętlą masy. Galwaniczne oddzielenie obwodów nie jest proste, a przede wszystkim dość kosztowne.

Przyjąłem też, że interfejs nie powinien być zbyt skomplikowany. Znalazłem więc rozwiązanie likwidujące większość problemów: interfejs RS485. Jest to interfejs podobny do RS232, w którym dane są przesyłane z wykorzystaniem pętli prądowej. Oznacza to, że połączone przyrządy nie muszą mieć wspólnej masy. Ponadto standard RS485 uwzględnia możliwość pracy „sieciowej“ z kilkoma urządzeniami nadawczymi, chociaż w danej chwili nadawać może tylko jedno.

W pierwszej fazie projektowania wybrałem tryb *fullduplex*, ale po analizie protokołu transmisji pomiędzy urządzeniami a komputerem doszedłem do wniosku, że *semiduplex* wystarczy. Dzięki temu interfejs znacznie się uprościł.



Rys. 1. Schemat elektryczny emulatora-programatora.

Tab. 1. Zestawienie stanów pracy sygnalizowanych przez diody LED

LED BUSY	LED ERROR	Stan urządzenia
zgaszona	zgaszona	Tryb emulacji
świeci	zgaszona	Podłączenie do magistrali RS485
miga	zgaszona	Zajętość emulatora (programowanie, itp.)
zgaszona	miga	Przekroczony adres
zgaszona	świeci	Przekroczono czas oczekiwania na transmisję RS
świeci	miga	Błąd pliku IntelHex
miga	miga	Błąd zapisu bajtu do procesora
świeci	świeci	Przepełniony bufor odbiorczy RS
migają naprzemiennie	migają naprzemiennie	Przepełnienie stosu

Aby umożliwić współpracę emulatora z komputerem wyposażonym w interfejs RS232, konieczny jest dodatkowy konwerter RS232/RS485, którego opis także zamieszczono w artykule.

Opis układu

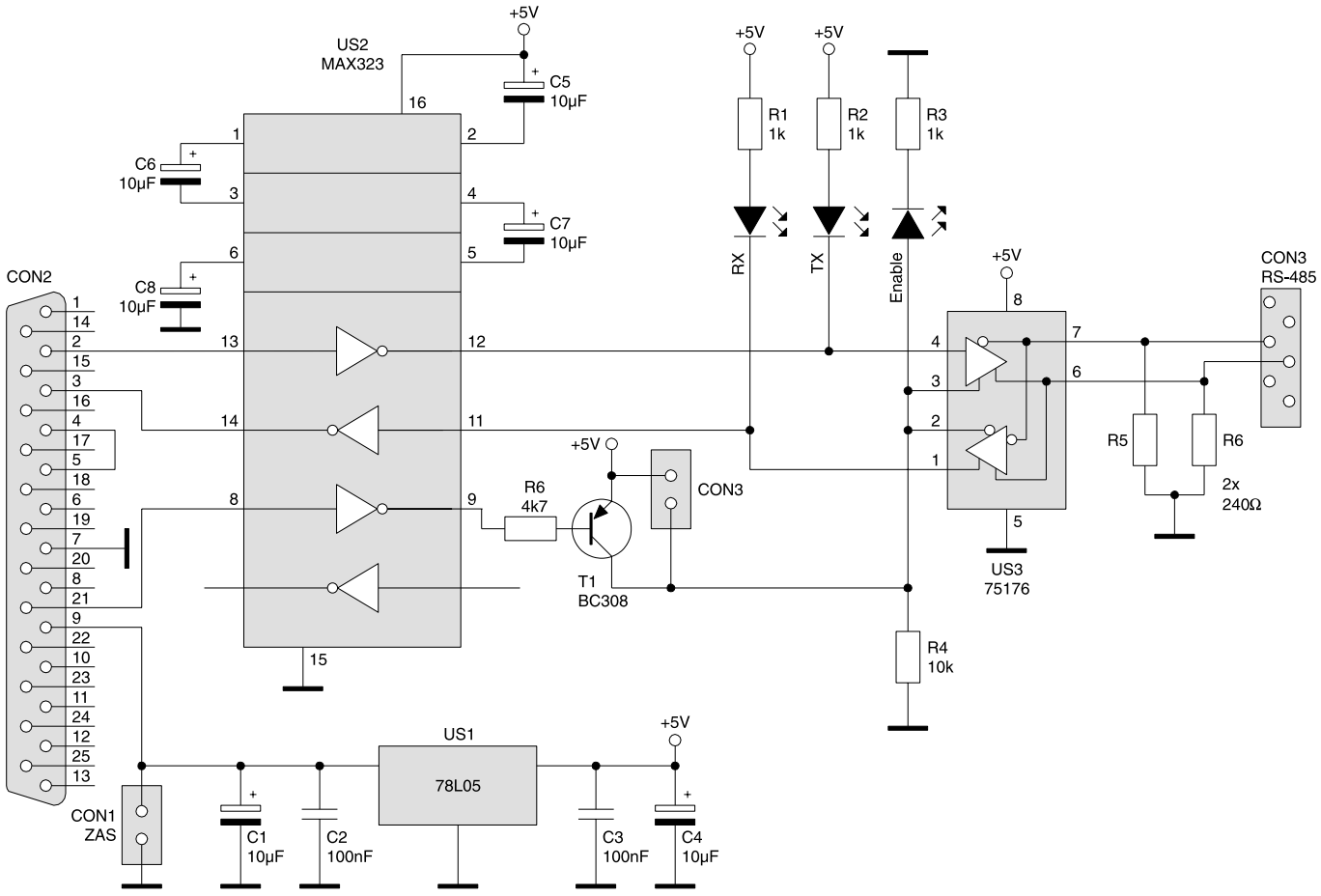
Schemat elektryczny emulatora-programatora pokazano na **rys. 1**. Jak widać jego budowa jest podobna do budowy poprzednika - AVT-995.

Emulator jest zasilany z uruchamianego urządzenia. Dioda D1 zabezpiecza przed skutkami odwrotnego włożenia złącza emulacyjnego w podstawkę. Do zerowania procesora wykorzystano obwód z kondensatorem C1 i rezystorem umieszczonym w strukturze US2. Należy wspomnieć, że choć długość kodu programu nie przekracza 4kB, to nie można użyć procesora AT89C51, ponieważ w aplikacji wykorzystano 256B wewnętrznej pamięci RAM (niestety stosu nie da się przenieść do zewnętrznej pamięci RAM). Jako US2 można zastosować procesor: AT89C52, AT89S8252, AT89S53 itp.

Sygnał prądowy z komputera o standardzie RS-485 jest przekształcany do postaci napięciowej o poziomach TTL za pośrednictwem transceivera US1. Jeśli korzystamy z RS-232, konwersja napięć następuje w transoptorze US8. Należy pamiętać, że o maksymalnej szybkości decydują wówczas parametry zastosowanego transoptora. Diody LED sygnalizują:

- *Power* - zasilanie emulatora,
- *Busy* i *Error* - tryb pracy i błędy (szczegóły w **tab. 1**),
- *Rx* - odbiór danych z komputera,
- *Tx* - transmisję do komputera lub wyświetlacza LCD,
- *Enable* - transmisję z emulatora do komputera przez RS-485.

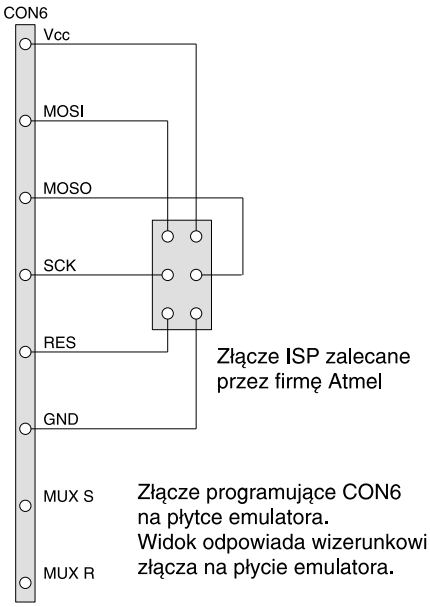
Zewnętrzny bufor danych przychodzących z RS232 i na dekodowanie rekordów IntelHex stanowi pamięć o pojemności 32 lub 128kB. Sterowanie pamięcią odbywa się w sposób standardowy dla rodziny 8051, tj. za pośrednictwem zatrzasku US3. Jak widać, linie adresowe i danych pamięci RAM nie są połączone



Rys. 2. Schemat elektryczny konwertera RS232/RS485.

Tab. 2. Połączenia konieczne do wykonania w przejściówce dla procesorów z ośmioma wyprowadzeniami.

złącze IDC40		złącze IDC20		złącze IDC10	
pin	nazwa	pin	nazwa	pin	nazwa
1	PB0 (T0)	12	PB0 (AIN0)	7	PB0 (AIN0/MOSI/AREF)
2	PB1 (T1)	13	PB0 (AIN1)	6	PB1 (INT0/MISO/AIN1)
3	PB2 (AIN0)	14	PB0	5	PB2 (T0/SCK/ADC1)
4	PB3 (AIN1)	15	PB3 (OC1)		
5	PB4 (SS)	16	PB4		
6	PB5 (MOSI)	17	PB5 (MOSI)		
7	PB6 (MISO)	18	PB6 (MISO)		
8	PB7 (SCK)	19	PB7 (SCK)		
9	RESET	1	RESET	1	RESET (PB5)
10	PD0 (RXD)	2	PD0		
11	PD1 (TXD)	3	PD1 (TXD)		
12	PD2 (INT0)	6	PD2 (INT0)	6	(Przez JP1)
13	PD3 (INT1)	7	PD3 (INT1)		
14	PD4 (OC1A)	8	PD4 (T0)		
15	PD5 (WR)	9	PD5 (T1)		
16	PD6 (RD)	11	PD6 (ICP)		
18	XTAL2	4	XTAL2	3	XTAL2 (PB4/ADC2)
19	XTAL1	5	XTAL1	2	XTAL1 (PB3/ADC3/CLOCK)
20	GND	10	GND	4	GND
40	VCC	20	VCC	5	VCC



Rys. 3. Schemat elektryczny kabla do programatora.

z odpowiadającymi im liniami procesora. Nie ma to wpływu na pracę układu. W pamięciach RAM numer linii adresowej czy danych można traktować jako umowny. To samo dotyczy innych układów pamięci z tym, że przy pamięciach stałych (ROM/EPROM) należy odpowiednio zmienić plik zapisujący/symulujący pamięć. Przeważnie łączy się linie adresowe i danych z odpowiadającymi im liniami procesora.

Przełączanie linii interfejsu SPI zapewniają klucze analogowo-cyfrowe US5 i US6. Jeśli będziemy stosowali duże wartości częstotliwości zegarowej dla procesorów AVR, to konieczna może być ich wymiana na 74HCT4053.

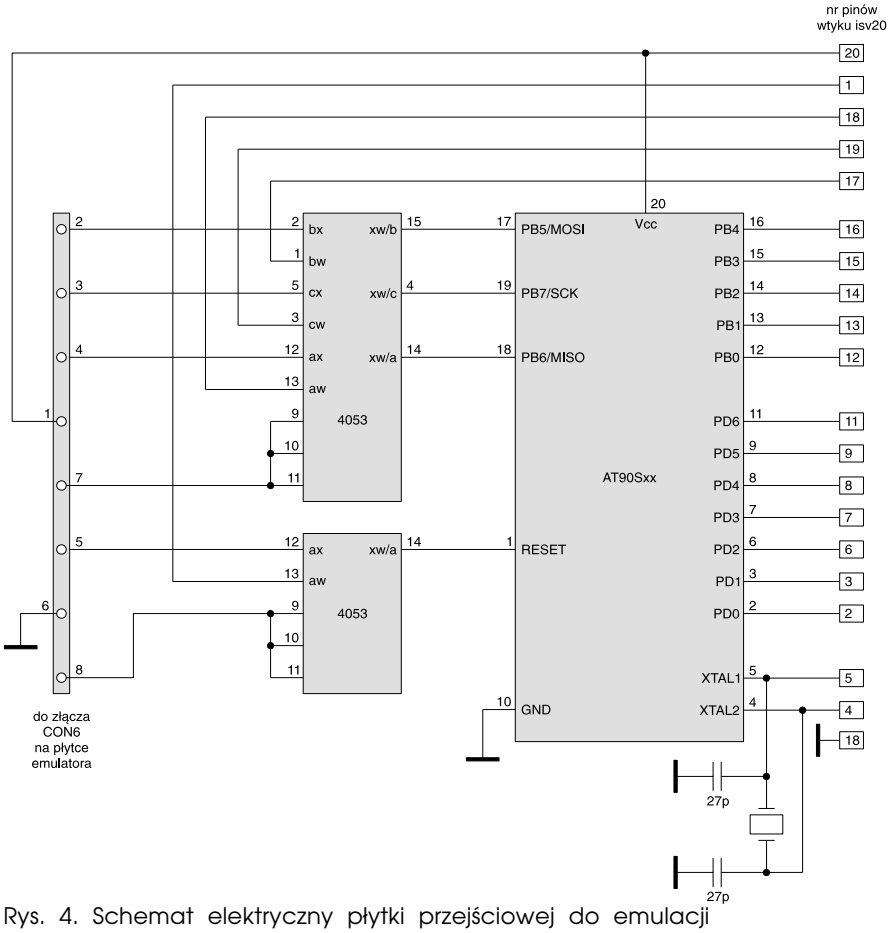
Jumpery ADR_0 i ADR_1 ustalają adres emulatora. Możliwe jest ustawienie czterech adresów:

Adres	ADR_0	ADR_1
0	rozwarty	rozwarty
1	zwarty	rozwarty
2	rozwarty	zwarty
3	zwarty	zwarty

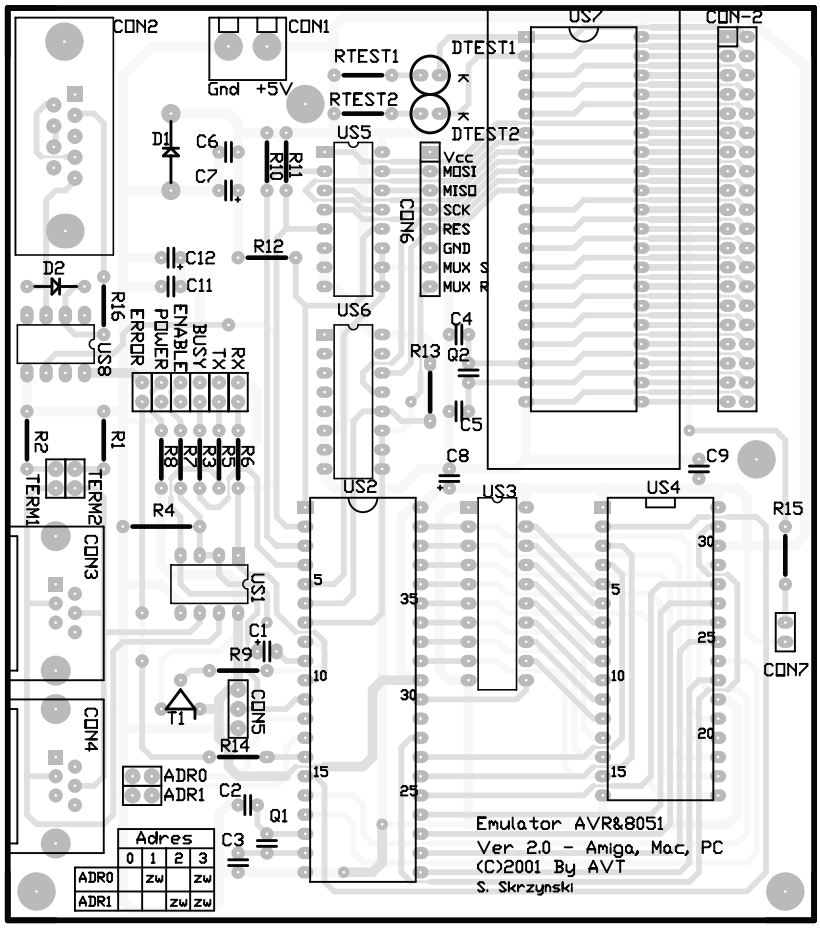
Złącze CON-5 można wykorzystać do podłączenia wyświetlacza LCD. Będą na nim wyświetlane komunikaty o błędach, wielkość pamięci podłączonego procesora itp. W aktualnej wersji oprogramowania opcja ta nie jest aktywna!

Na **rys. 2** pokazano schemat elektryczny konwertera RS232/RS485, który zapewnia dwukierunkową konwersję sygnałów pomiędzy interfejsem RS485 emulatora i standardowym interfejsem szeregowym Amigi lub PC.

Złącze CON-6 można wykorzystać do programowania proce-



Rys. 4. Schemat elektryczny płytki przejściowej do emulacji mikrokontrolerów 8- i 20-nóżkowych.



Rys. 5. Rozmieszczenie elementów na bazowej płytce drukowanej.

sorów w systemie lub do emulowania procesorów AVR z 8/20 wyprowadzeniami korzystając z dodatkowych układów. Złącze to jest bliźniaczo podobne do złącza umieszczonego w AVT-995. Dodano tylko dwie linie MUX_S i MUX_R, a pozostałe wyprowadzenia mają identyczne rozmieszczenie.

Jeśli zdecydujemy się na programowanie procesorów w systemie, to należy wykonać odpowiedni kabelek. Jego wykonanie zależy od tego, jakie złącze zastosowano w programowanym urządzeniu. Najczęściej spotyka się złącza zalecane przez firmę Atmel. Kabel będzie miał wygląd jak na **rys. 3**. Można też z jednej strony zakończyć go chwytakami.

Na **rys. 4** przedstawiono schemat montażowy płytki przejściowej umożliwiającej prostą emulację procesorów w obudowach 8 i 20 pin. Przejściówkę łączymy z emulatorem kablem FLAT40, z zaciśniętymi na obu końcach złączami IDC40. Do emulacji procesorów z 20 wyprowadzeniami należy wykonać kabel taki sam jak dla emulacji procesorów z 40 wyprowadzeniami, ale taśmą FLAT20 ze złączami IDC20 i ISV20.

Dla złącza emulacyjnego procesora 8-pinowego, z powodu braku złącz IDC 8-stykowych, przewidziano złącze IDC10. Dwa ostatnie styki złącza nie są wykorzystane. Brak wtyku ISV8 zmusza do wykorzystania wtyku ISV20 po uprzednim usunięciu niepotrzebnych pinów. W **tab. 2** przedstawiono połączenia przejściówki.

Jak widać, przy emulacji procesora 20-nóżkowego nie są wykorzystywane wejścia analogowe i bramkowania timerów (dla AVR dodatkowo interfejs SPI).

Przy emulacji procesora 8-nóżkowego nie działa bramkowanie timera T0, a dla procesorów serii ATiny porty PB3, PB4, PB5 oraz wejścia analogowe. Zworką JP1 można symulować stan na wejściu przerwania INT0 dla procesorów w obudowie z ośmioma wyprowadzeniami.

Montaż i uruchomienie

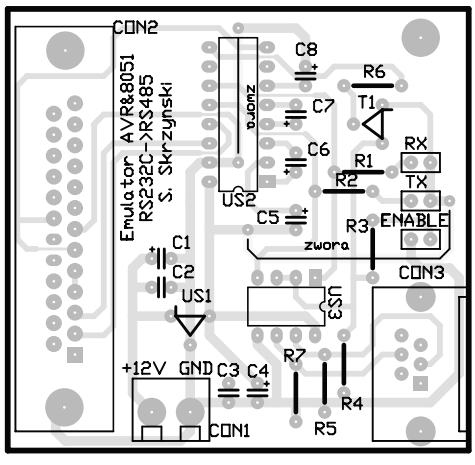
Schemat montażowy płytki emulatora pokazano na **rys. 5**, a schemat montażowy płytki konwertera RS232/485 na **rys. 6**.

Montaż rozpoczynamy od elementów najmniejszych, a kończymy na największych. Pod emulowany procesor można zastosować podstawkę zatrzaskową lub precyzyjną. Pod elementy testowe DTEST1, DTEST2, C4, C5, Q2 najlepiej zastosować odcinki listwy „tulipanowej".

Ze względu na problemy ze zdobyciem podstawki 32-stykowej, pod pamięć US4 także można zastosować listwę tulipanową. Jeśli nie przewidujemy zastosowania pamięci 128kB, można wlutować podstawkę 28-stykową.

W aktualnej wersji oprogramowania pamięć o pojemności większej niż 32kB nie jest obsługiwana! W podstawce układu US7 będziemy umieszczać emulowany lub programowany procesor. Najlepiej jest zastosować tam podstawkę zatrzaskową 40-stykową lub ostatecznie podstawkę precyzyjną (tulipanową).

Po podłączeniu napięcia +5V do złącza CON1 sprawdzamy napięcia zasilania układów scalonych. Gdy są poprawne, można umieścić układy w podstawkach. Jeśli korzystamy z interfejsu RS485, musimy wykonać konwerter zgodnie z rys. 2. Nie są wtedy potrzebne na płytce emulatora elementy CON2, R16, D2 i US8. Zworki TERM1 i TERM2 zakładamy wtedy, gdy emulator jest końcowym lub jedynym urządze-



Rys. 6. Rozmieszczenie elementów na płytce drukowanej konwertera RS232/RS485.

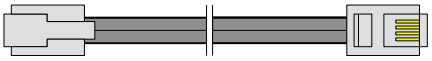
niem dołączonym do magistrali RS485.

Kabel łączący konwerter RS485 z emulatorem jest kablem telefonicznym z wtykami RJ-45. Wtyczki zaciśnięto jednak inaczej niż w typowym kablu telefonicznym - z przeplotem. Dzięki temu nie trzeba wyróżniać gniazd wejściowych i wyjściowych w interfejsie RS-485. Sposób zaciśnięcia złącz RJ-45 pokazano na **rys. 7**.

Jeśli po połączeniu emulatora z interfejsem dioda Rx w emulatorze świeci, to oznacza, że źle wykonaliśmy kabel i należy jedną z wtyczek RJ odwrócić. Jeśli ktoś będzie miał kłopoty z nabyciem złącz i kabli RJ, może zamiast nich wykorzystać złącza DB. Styk numer 1 złącza DB9 łączymy z wyprowadzeniem 6 układu US1, natomiast styk numer 2 złącza DB9 z wyprowadzeniem 7 US1.

W aktualnej wersji oprogramowania nie korzystamy z dwukierunkowej transmisji przez RS485, dlatego należy koniecznie założyć jumper na COM3 w płytce konwertera. Konwerter wymaga zasilania. Można zastosować napięcie +8 do +15V dołączane do zacisku CON-1 lub napięcie +5V (wówczas zamiast stabilizatora należy wlutować zworkę).

Amigowcy są w lepszej sytuacji. Na port RS ich komputerów są wyprowadzone napięcia +12 i -12V. Nie muszą więc stosować dodatkowego zasilania konwertera. Posiadaczom innych typów komputerów polecam wyprowadzić napięcie +12V na pin 9 portu RS, -12V na pin 10. Należy je włączyć przez szeregowe rezystory 47Ω/0,5W. Standard RS232 określa funkcję styków 9 i 10 jako napięcia testowe i nic się nie stanie jeśli je tam dołączymy.



Rys. 7. Sposób zaciśnięcia złącz RJ-45 na kablu połączeniowym.

Gdy korzystamy z interfejsu RS232C, nie musimy montować CON3, CON4, R1 i R2 na płytce emulatora, natomiast US1 musimy koniecznie usunąć! Do połączenia emulatora z komputerem wykorzystujemy typowy kabel null-modem.

Do pełni szczęścia braku tylko kabla emulacyjnego. Wykonujemy go, zaciskając złącza ISV40 i IDC40 na taśmie FLAT40.

Pierwszy test

Na płytce umieszczamy elementy: C4, C5, Q2, US7, DTEST1, DTEST2, RTEST1 i RTEST2. Zależnie od typu układu US7 wykonujemy dodatkowo:

- dla procesorów rodziny AT89Sxx: zwieramy CON7 (E-A dla AVT-995); do pinu 9 podstawki emulacyjnej dołączamy masę; kompilujemy plik „Test51.asm" wydając rozkaz dla PC „51 test51 2" lub uruchamiając skrypt „8051_AVT995+.rexx" na Amidze (plik do kompilacji „ASM:AVT-EmuAVR/Test51.asm");
- dla procesorów rodziny AT90Sxx: do wyprowadzenia 9 podstawki emulacyjnej dołączamy +5V; kompilujemy plik „Test51.asm" wydając rozkaz dla PC „avr testavr 2" lub uruchamiając skrypt „AVR_AVT995+.rexx" na Amidze (plik do kompilacji „ASM:AVT-EmuAVR/Test-AVR.asm").

Po chwili procesor powinien być zaprogramowany, diody DTEST powinny migać. Gdy tak jest, można usunąć DTEST1, DTEST2, C4, C5, Q2, rozewrzeć CON7 i odłączyć wyprowadzenie numer 9 procesora (i zworki CON7 lub EA dla AVT995).

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.

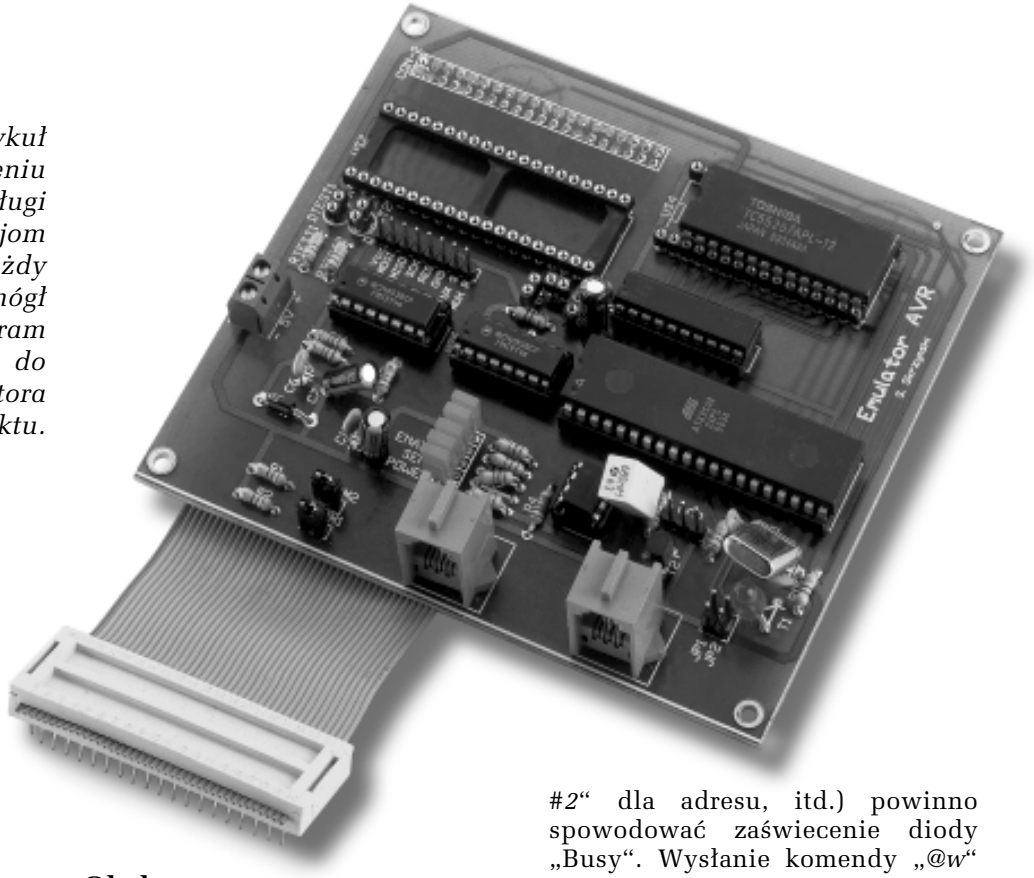
Płytka „przejściówki" dla procesorów 8/20pin nie wchodzi w skład kitu.

Wzory płytek drukowanych w formacie PDF są dostępne w Internecie pod adresem: ***http://www.ep.com.pl/?pdf/pazdziernik01.htm*** *oraz na płycie CD-EP10/2001B w katalogu* ***PCB****.*

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.

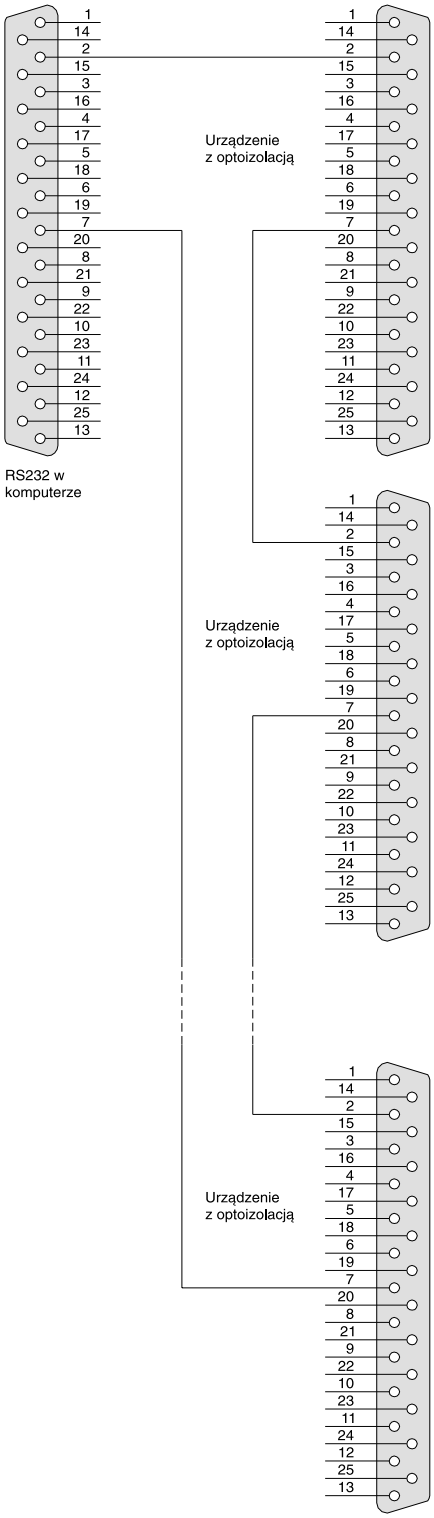
Ważne! Skrypty dla emulatora AVT-995 różnią się nieznacznie od skryptów dla nowej wersji. Spowodowane jest to tym, że AVT-995 nie czyści pamięci AVR przed programowaniem i jest konieczne wykonanie tej operacji przez wydanie odpowiedniego rozkazu w skrypcie oraz tym, że w AVT-995 nie można użyć większej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