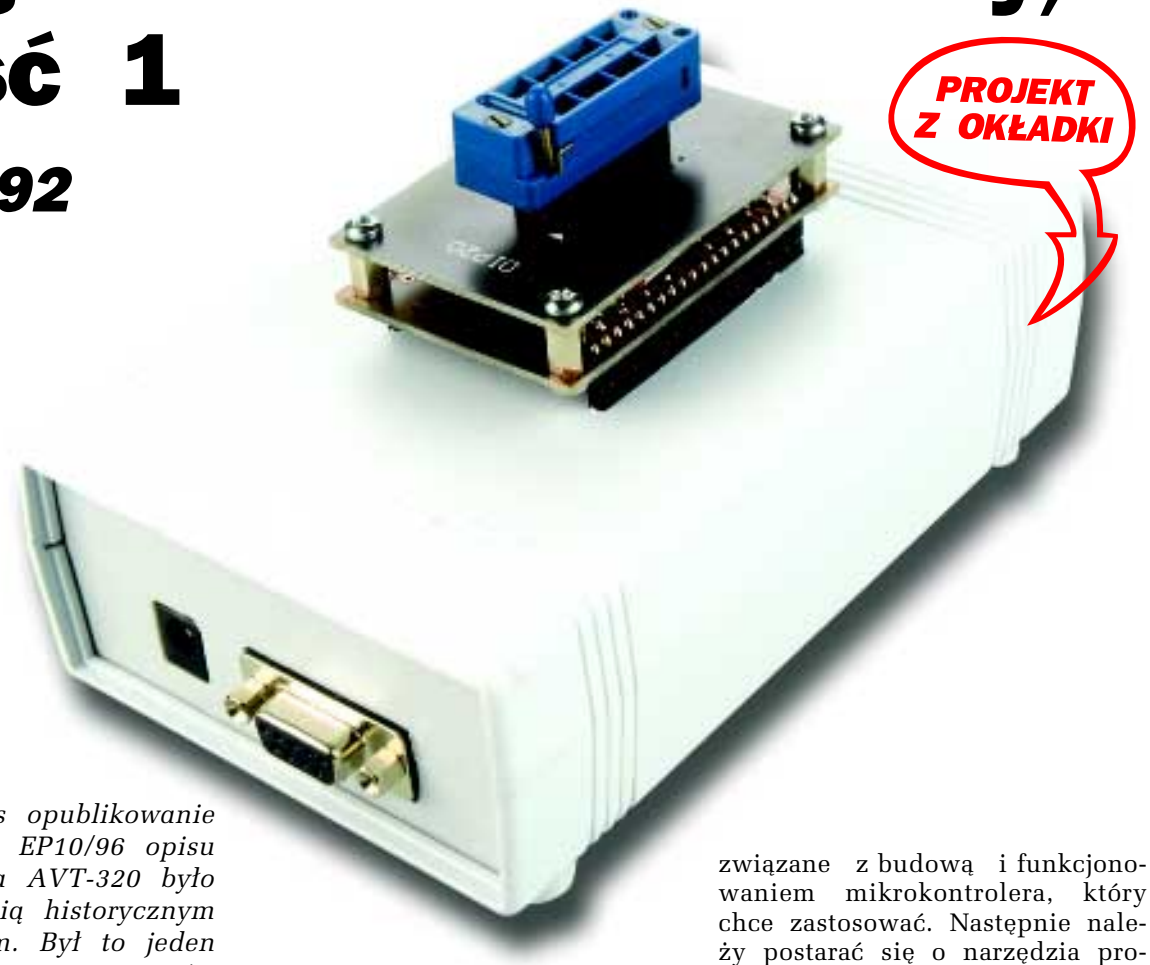
Punch Programator uniwersalny, część 1

AVT-5092



Dla nas opublikowanie w EP10/96 opisu programatora AVT-320 było z pewnością historycznym wydarzeniem. Był to jeden z pierwszych programatorów umożliwiających programowanie mikrokontrolerów '51 z pamięcią Flash. Teraz przedstawiamy jego godnego następcę - programator, który nazwaliśmy Punch, co w języku angielskim oznacza perforator do kart papierowych - niegdyś nośników programów i danych dla komputerów. ***Rekomendacje****: programator jest podstawowym przyrządem w pracowni elektronika, a więc ten opis może zainteresować większość naszych Czytelników.*

W artykule przedstawiono projekt uniwersalnego programatora elementów półprzewodnikowych działającego w oparciu o skryptowy język poleceń FEMTO. Programator umożliwia programowanie elementów, które zawiera jego biblioteka, a także wprowadzenia własnych procedur. Obecnie dostępne są skrypty dla takich elementów jak mikrokontrolery Atmela: AT89C1051/2051/4051, AT89C51/52/55, AVR-ów AT90S2313, AT90S8515, AT90S8535 i pokrewnych, a także szeregowych pamięci EEPROM z interfejsem I^2C.

Każdy, kto poważnie myśli o konstruowaniu własnych urządzeń wykorzystujących mikroprocesory oraz chciałby samodzielnie pisać dla nich oprogramowanie, musi przygotować się do tego przedsięwzięcia.

Musi przede wszystkim „zdobyć" dane techniczne i informacje związane z budową i funkcjonowaniem mikrokontrolera, który chce zastosować. Następnie należy postarać się o narzędzia programistyczne (kompilatory) oraz programator.

Karty katalogowe mikrokontrolerów oraz oprogramowanie narzędziowe można znaleźć w Internecie - chociażby na stronach producentów danego podzespołu.

Programator trzeba niestety kupić, ale można go także wykonać samemu. Wiele mikrokontrolerów można programować po zamontowaniu w systemie za pośrednictwem interfejsu ISP. Programatory, w których wykorzystuje się ten sposób programowania, są bardzo proste w wykonaniu. Kilka z nich opisaliśmy na łamach EP.

Jeżeli jednak trzeba będzie zaprogramować element, który nie ma takiej magistrali, albo w urządzeniu znajdzie się inny programowalny element, np. pamięć EPROM, konieczne jest posiadanie standardowego programatora.

To właśnie zainspirowało mnie do opracowania własnego programatora.

Dobry programator, czyli jaki?

Moim zdaniem dobry programator powinien charakteryzować się następującymi cechami:
- niezawodnością,
- dużą liczbą programowanych typów elementów,
- niską ceną,

a także poręcznością, łatwością obsługi, dostępnością serwisu i pomocy ze strony producenta oraz estetycznym wyglądem. Niestety, niektóre z tych wymagań trudno ze sobą pogodzić. Rozbudowane programatory, bogate w zaawansowane opcje, są bardzo drogie, a ze wsparciem producentów - szczególnie zagranicznych - różnie bywa. Mając to na uwadze, może warto pokusić się o skonstruowanie własnego programatora, który nie będzie konkurencyjny z produktami renomowanych firm, będzie jednak prosty, uniwersalny i w miarę niedrogi?

Opis układu

Efektem dwóch lat prób i nieustającego dopingu ze strony zespołu Elektroniki Praktycznej jest programator, którego schemat pokazano na **rys. 1**. Jego najważniejsze parametry są następujące:
- napięcie programujące ustawiane programowo w przedziale od 3 do 13 V,
- napięcie zasilania programowanego elementu ustawiane programowo w przedziale od 2 do 7 V,
- najkrótszy możliwy czas trwania impulsu potrzebnego do programowania elementu wynosi 100 ns,
- zewnętrzne napięcie zasilania programatora powinno mieć wartości: +16 VDC lub 12 VAC,
- możliwość samodzielnego wykonania prostych adapterów przeznaczonych dla różnego typu obudów programowanych elementów,
- możliwość samodzielnego pisania skryptów w języku poleceń FEMTO dla nowych elementów (skrypty są „czystymi" plikami tekstowymi) - ograniczenia w przystosowaniu programatora do obsługi nowych elementów wiążą się jedynie z jego możliwościami technicznymi (patrz wyżej),
- współpraca programatora z komputerem PC sterującym jego pracą.

Programator służy do programowania elementów z równoległą, 8-bitową magistralą danych lub z magistralą szeregową, np. I^2C. Pojemność pamięci programowanych elementów może mieć 64 kB lub więcej. Sygnały wyjściowe programatora przypisane są na stałe do wyprowadzeń jego zewnętrznych gniazd JP1 i JP2, toteż do programowania konkretnego elementu należy przygotować adapter służący do połączenia wyprowadzeń odpowiednich gniazd z wyprowadzeniami poszczególnych nóżek elementu. Samodzielne przygotowanie adapterów nie jest trudne ani specjalnie kosztowne.

Dla omówienia budowy urządzenia i istoty języka FEMTO należy przypomnieć schemat w czasie programowania typowego elementu. Najlepiej zrobić to na stosunkowo prostym przykładzie, np. równoległej pamięci EPROM 2764.

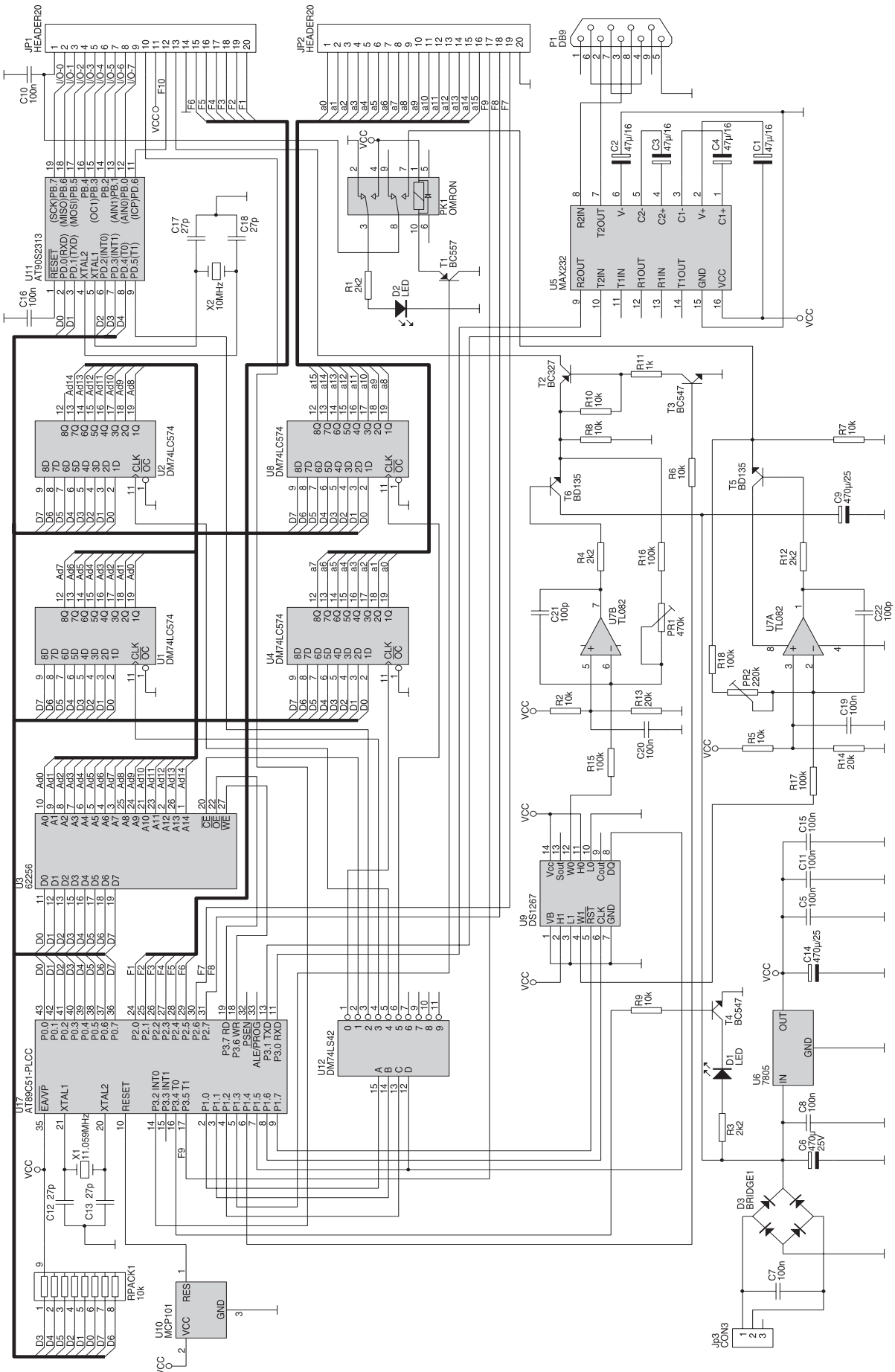
W procesie programowania jest szereg pętli składających się z kilku prostych czynności. Najpierw należy podać napięcie zasilające (np. +5V) na odpowiednie wyprowadzenia EPROM-u. Następnie na wyprowadzenia adresowe podawany jest adres komórek w matrycy pamięci, które będą programowane, a na magistralę danych jest przesłany bajt, który ma być wpisany do komórek. Następnie na wyprowadzeniu napięcia programującego powinno pojawić się napięcie o wartości np. 12,75 V. Impuls na odpowiednim wejściu sterującym wymusi otwarcie buforów magistrali danych EPROM-u i przyjęcie zapisywanej danej. Wreszcie finał, czyli wygenerowanie impulsu zapisującego, np. na wejściu CS EPROM-u, o czasie trwania 50 µs i polaryzacji ujemnej, przepisze bajt danych z magistrali EPROM-u do jego komórek pamiętających. Powtarzając te elementarne czynności odpowiednią liczbę razy, można zapisać całą wewnętrzną matrycę pamiętającą elementu. Oczywiście, w rzeczywistości wszystko jest trochę bardziej skomplikowane. Dane techniczne elementu określają parametry poszczególnych przebiegów na wyprowadzeniach elementu w czasie programowania i zależności czasowe między nimi. Należy także pamiętać o procedurze sprawdzenia, czy wszystkie dane zostały prawidłowo zapisane w matrycy elementu. Jednak ogólna zasada pozostaje taka jak opisana wyżej.

Jeżeli programator może wytworzyć odpowiednie przebiegi elektryczne i podać je na właściwe nóżki programowanego elementu, cała operacja zakończy się sukcesem. Zadaniem części sprzętowej programatora, o schemacie z rys. 1, jest właśnie generowanie odpowiednich impulsów i przesyłanie ich na odpowiednie wyprowadzenia programowanego elementu. Jednak do sterowania programatorem niezbędny jest także program, który nadzoruje działanie części sprzętowej programatora. Oprogramowanie sterujące zainstalowane w komputerze zajmuje się także magazynowaniem i przesyłaniem danych, które mają być zapisane w programowanym elemencie oraz komunikuje się z użytkownikiem.

Oprócz części sprzętowej i programu sterującego jest niezbędny jeszcze trzeci element tego systemu, czyli skrypt zapisany w języku FEMTO. W pliku skryptu zawarte są zarówno polecenia dla programu sterującego, np.: prześlij dane do programatora, zainicjuj programowanie, sprawdź poprawność programowania, jak i szczegółowe rozkazy dla części sprzętowej, np. włącz zasilanie, wygeneruj na odpowiednim wyprowadzeniu impuls, ustaw kolejny adres, a także informacje dla użytkownika w postaci komunikatów. Relacje pomiędzy tymi trzema częściami składającymi się na programator można przedstawić następująco:

skrypt
↓
program sterujący
↑↓
część sprzętowa

Współdziałanie tych trzech części systemu pozwala zaprogramować układ, a także przystosować programator do obsługi nowych typów układów.



Rys. 1. Schemat elektryczny programatora

Schemat przedstawiony na rys. 1 sprawia być może wrażenie skomplikowanego, lecz w istocie - za pomocą prostych układów, jakimi są rejestry zatrzaskowe czy wzmacniacze operacyjne - realizuje podstawowe zadania części sprzętowej: podawanie na programowany element właściwych kombinacji stanów logicznych, generację odpowiednich napięć (programującego i zasilania), a także komunikację z programem sterującym, który rezyduje w PC-cie.

Układy U4, U8, U11 sterowane przez mikrokontroler służą do podawania sygnałów na element programowany. Układy U1, U2 związane są z pamięcią RAM U3, w której m.in. przechowywane są dane zapisywane i odczytywane. Do generowania napięć: programującego i zasilania służą układy U7 i U9, także sterowane przez procesor. Komunikacja z komputerem sterującym jest możliwa dzięki interfejsowi RS232 zbudowanemu na U5.

Wszystkie sygnały i napięcia wyprowadzone są na złącza JP1 i JP2. Są to m.in.: 8 linii portu danych I/O0...I/O7, 16 linii adresowych A0...A15, 10 linii portów oznaczonych symbolami F1...F10. Zsumowanie liczby potrzebnych linii wskazuje, że zastosowany procesor AT89C52 nie jest w stanie ich obsłużyć własnymi portami i musi „podeprzeć się" dodatkowymi układami. Wszystkie są podłączone do wspólnej magistrali danych obsługiwanej przez port P0 procesora U17. Każdy z tych układów ma swoje wejście aktywujące połączone z wyprowadzeniami multipleksera U12. Dzięki temu procesor, wybierając po kolei każdy z układów, może do niego zapisać dane lub je odczytać, wykorzystując w tym celu jeden port P0 i cztery linie portu P1 sterujące multiplekserem. W ten sposób wystawianych jest 16 bitów adresu potrzebnych np. w czasie programowania EPROM-u. Tak samo realizowany jest dostęp do wewnętrznej pamięci RAM programatora (U3), w której przechowywane są dane.

Niektórych Czytelników może dziwić obecność w układzie drugiego procesora oznaczonego jako U11. Element ten odpowiada za realizację kilku zadań. Pełni m.in. rolę bramy wejścia-wyjścia magistrali danych, obsługuje port F10, na który można wysłać precyzyjnie odmierzone impulsy o czasie od 100 ns do 6,5 ms, przechowuje także w swojej wewnętrznej pamięci EEPROM dane konfiguracyjne programatora. Każdą z tych funkcji można powierzyć osobnemu układowi scalonemu, jednak jest korzystniej, gdy wykonuje je tylko jedna kostka zajmująca dużo mniej miejsca.

Zazwyczaj do zaprogramowania wielu typów elementów potrzebne jest napięcie programujące o wartości znacznie przekraczającej +5 V. Są także elementy wymagające w czasie programowania podwyższenia napięcia zasilania np. do +6,5 V. Z tego powodu wartość obu tych napięć może być zmieniana i ustawiana programowo.

Do realizacji tego zadania wykorzystano w programatorze podwójny potencjometr elektroniczny typu DS1267 oznaczony na schemacie symbolem U9. Procesor, wysyłając dane do potencjometru liniami portów P1.5...P1.7, może wymusić na wyprowadzeniu potencjometru pełniącego rolę suwaka napięcie z przedziału 0...5 V. To jednak nie wystarcza, ponieważ do programowania potrzebne są znacznie wyższe napięcia. W celu obejścia tego ograniczenia zastosowano wzmacniacze napięcia stałego. Spójrzmy na schemat: napięcie z wyjścia suwaka W0 U9...12 jest podawane poprzez opornik R15 na wejście odwracające wzmacniacza U7B. Po wzmocnieniu napięcie trafia do wtórnika emiterowego T6, natomiast poziom wzmocnienia układu wzmacniacz - tranzystor określa pętla sprzężenia zwrotnego R16, PR1 i rezystor R15. Z emitera tranzystora napięcie programujące jest podawane poprzez tranzystor odcinający T2 na złącze JP1. Tranzystor odcinający jest potrzebny, aby w czasie wkładania lub wyjmowania z podstawki programowanego elementu na jego wyprowadzeniach nie pojawiały się potencjały mogące doprowadzić do uszkodzenia. W podobny sposób wytwarzane jest napięcie zasilające. Tym razem elementem odcinającym nie jest tranzystor, a przekaźnik PK1, do którego drugiej pary styków dołączona jest dioda LED D2 sygnalizująca stan programowania.

Układ U5 jest zwykłym konwerterem poziomów sygnałów RS232 na poziom TTL. Program sterujący komunikuje się z częścią sprzętową, korzystając z portu COM komputera, którym są przesyłane rozkazy sterujące i dane. Po konwersji poziomów sygnały są przekazywane do wyprowadzeń RxD i TxD procesora.

Oprogramowanie procesora sterującego częścią sprzętową programatora

Układ stanowiący część sprzętową programatora nie jest jedynie biernym wykonawcą poleceń programu sterującego, ale posiada pewną autonomię. Przede wszystkim jest wyposażony w programowy interpreter języka FEMTO. Kody poleceń dotyczą elementarnych działań na wyprowadzeniach, np. zmiany poziomu z niskiego na wysoki na wyprowadzeniu F1, operacji logicznych, np. porównań stanów na wyprowadzeniach I/O0... I/O7 z ostatnio programowanym bajtem danych oraz dostępem do wewnętrznych rejestrów niezbędnych do pracy części programującej, do których ma także dostęp program sterujący w PC-cie. Takim rejestrem jest licznik adresu, którego 16 bitów wyprowadzonych jest na złącze JP2 i oznaczonych symbolami A0...A15. W rzeczywistości licznik ten składa się z 4 bajtów, a wartość bitów pozostałych bajtów można za pomocą poleceń przepisywać np. do wyprowadzeń F1...F10.

Kolejną grupą dostępnych z zewnątrz rejestrów są rejestry wskaźników dostępu do buforów pamięci RAM części sprzętowej. Wskaźnik określa po prostu adres, do którego można się odwołać.

Jakie to są rejestry (bufory)? Programator dzieli wewnętrzną pamięć RAM (U3) na trzy obszary. W pierwszym lokowane są kody rozkazów języka FEMTO interpretowane w czasie pracy programatora. Jak to zostało opisane w części dotyczącej opisu działania, kody określają elementarne czynności, jakie część sprzętowa musi wykonać np. w czasie zapisu danych do pamięci EPROM lub

w czasie weryfikacji itd. Drugi obszar pamięci jest przeznaczony na dane wejściowe. Jest to po prostu wydzielona część pamięci RAM, w której gromadzone są dane przesłane z komputera do części sprzętowej i przeznaczone do zapisu w programowanym elemencie. W trzecim obszarze zorganizowano bufor danych wyjściowych. Jest to obszar gromadzenia odczytanych danych z programowanego elementu przed przesłaniem ich do komputera PC.

Każdy z tych obszarów ma własne wskaźniki określające jego położenie i bieżący adres. Obszar kodu posiada jeden wskaźnik. Najpierw będzie z niego korzystał program sterujący, który prześle z PC-ta kody rozkazów ze skryptu. Następnie wskaźnik zostanie wyzerowany. Gdy rozpocznie się programowanie, kontrolę nad nim przejmie procesor części sprzętowej, realizując kolejne rozkazy pętli programowania. Bufory danych mają po dwie pary wskaźników. Jednym zarządza wyłącznie program sterujący, natomiast drugim procesor części sprzętowej. Dzięki temu możliwe jest przesyłanie danych, gdy toczą się jeszcze operacje programowania. Czytelnik tego opisu może zapytać: co się stanie, gdy wskaźniki dotrą do końca obszaru buforów? Wówczas przestawiane są od nowa na początek swojego bufora. Dzięki temu możliwe jest programowanie elementów o pojemności pamięci przekraczającej pojemność zastosowanej pamięci RAM, czyli 32 k. Po prostu, gdy zapisana zostanie do programowanego elementu część danych z bufora, na zwolnione miejsce wpisywane są nowe dane przesyłane z komputera.

Osobom, które zechcą skonstruować własny programator, opierając się na tym projekcie, należy opisać współpracę procesora głównego części sprzętowej z procesorem U11. Ze względu na brak odpowiedniej liczby wyprowadzeń U11, komunikacja z nim odbywa się z wykorzystaniem jedynie 5 linii magistrali danych D0...D4. Czterema młodszymi przesyłane są połówki bajtów danych, natomiast linia D4 (zależnie od sytuacji) służy do sygnalizacji gotowości do transmisji lub określa, która część bajtu jest aktualnie transmitowana. Tak jak w przypadku innych układów, procesor sygnalizuje chęć nawiązania kontaktu poprzez ustawienie poziomu niskiego na wyprowadzeniu multipleksera U12, dołączonego do linii portu PD5 układu U11.

Zastosowanie jako układu U11 szybkiego procesora AT90S2313 z rodziny AVR pozwoliło na programowe generowanie krótkich, ale precyzyjnie odmierzonych impulsów, które mogą pojawiać się na wyjściu F10. Wytworzenie takich impulsów przez procesor rodziny '51 jest niemożliwe, ponieważ jak wynika z zasady jego pracy, impulsy taktujące wytwarzane są na podstawie sygnału generatora kwarcowego przez podział jego częstotliwości przez 12. Nawet wykonując następujące bezpośrednio po sobie rozkazy, ustawienia któregoś z portów naprzemian: na poziomie wysokim i następnie niskim, jesteśmy w stanie wygenerować impuls o czasie trwania nie krótszym niż 1,09 µs, co wynika z częstotliwości własnej zastosowanego kwarcu X1. Takie impulsy lub ich wielokrotność można uzyskać na wyprowadzeniach F1...F9. Najczęściej są one wystarczające.

Tam jednak, gdzie potrzebne są krótkie impulsy o małym błędzie czasu trwania, należy użyć wyprowadzenia F10 obsługiwanego przez AT90S2313. Zaletą procesorów AVR jest ich szybkość działania wynikająca z tego, że cykl rozkazowy w większości przypadków jest równy jednemu okresowi sygnału generatora kwarcowego. Procesory te są co najmniej 12 razy szybsze od ich odpowiedników z rodziny '51. Dodatkowo, dzięki wykorzystaniu skoku pośredniego, adresowanego rejestrem Z, można wytworzyć dokładnie impuls o czasie trwania 100 ns lub jego wielokrotności.

Realizacja programowa generatora impulsów dodatnich o krótkim czasie trwania jest następująca: w pamięci programu procesora AVR wpisana jest tablica zawierająca np. 256 razy powtórzony rozkaz ustawienia portu PD6 (w programatorze obsługuje wyjście F10) na poziom wysoki. Bezpośrednio za tablicą powinien być rozkaz ustawiający PD6 na poziom niski. Ponieważ oba roz-

WYKAZ ELEMENTÓW

Rezystory
R1, R3, R4, R12: 2,2kΩ
R2, R5...R10: 10kΩ
R11: 1kΩ
R13, R14: 20kΩ
R15...R18: 100kΩ
RPACK1: drabinka rezystorów 10kΩ
PR1: potencjometr wieloobrotowy 470kΩ
PR2: potencjometr wieloobrotowy 220kΩ

Kondensatory
C1...C4: 47µF/16V
C5, C7, C8, C10, C11, C15, C16, C19, C20: 100nF
C6, C9: 470µF/25V
C12, C13, C17, C18: 27pF
C14: 470µF/16V
C21, C22: 100pF

Półprzewodniki
D1, D2: diody LED: czerwona i zielona ϕ3 lub 5mm z oprawkami
D3: mostek prostowniczy 1A/50V
T1: BC557
T2: BC327
T3, T4: BC547
T5, T6: BD135
U1, U2, U4, U8: 74LS574SMD
U3: pamięć RAM 62256 SMD
U5: MAX232
U6: 7805
U7: TL082
U9: DS1267
U10: MCP101 (lub podobny)
U11: AT90S2313 SMD zaprogramowany
U12: 74LS42SMD
U17: AT89C52 zaprogramowany PLCC

Różne
JP1, JP2: szpilki do złącz zaciskanych na taśmie HEADER20
JP3: CON3 gniazdo zasilania wlutowywane do płytki
P1: DB9 gniazdo kątowe żeńskie wlutowywane do płytki
PK1: przekaźnik 5V miniaturowy
X2: 10MHz
X1: 11,059MHz
Dwustronna płytka drukowana programatora
Jednostronna płytka drukowana „connect board" złącza dla wymiennych adapterów
Obudowa typu Z50
Podstawka PLCC44
Podstawka DIP16
Styki precyzyjne ϕ0,8mm
Taśma 20-żyłowa 20cm
Wkręty stożkowe M3
Złącza zaciskane na taśmę 20-żyłową

Uwaga! *Wszystkie oporniki i kondensatory nie-elektrolityczne typu SMD 1206*

kazy potrzebują do realizacji tylko jednego okresu oscylatora, organizując skok do tablicy bliżej lub dalej jej końca, można wytworzyć impuls będący wielokrotnością 100 ns. Adres skoku należy wyliczyć przed jego realizacją i zapisać w rejestrze Z. Sam skok wykonywany jest poleceniem IJMP (*indirect jump*). W ten sposób stosując kwarc 10 MHz można wygenerować impulsy o czasie trwania dokładnie 0,1 µs...25,6 µs. Oczywiście, sposób ten zajmuje sporo miejsca w pamięci programu (256 razy powtórzony ten sam rozkaz), ale jest prosty i skuteczny. Do generacji dłuższych impulsów wykorzystywane są programowe pętle opóźniające.

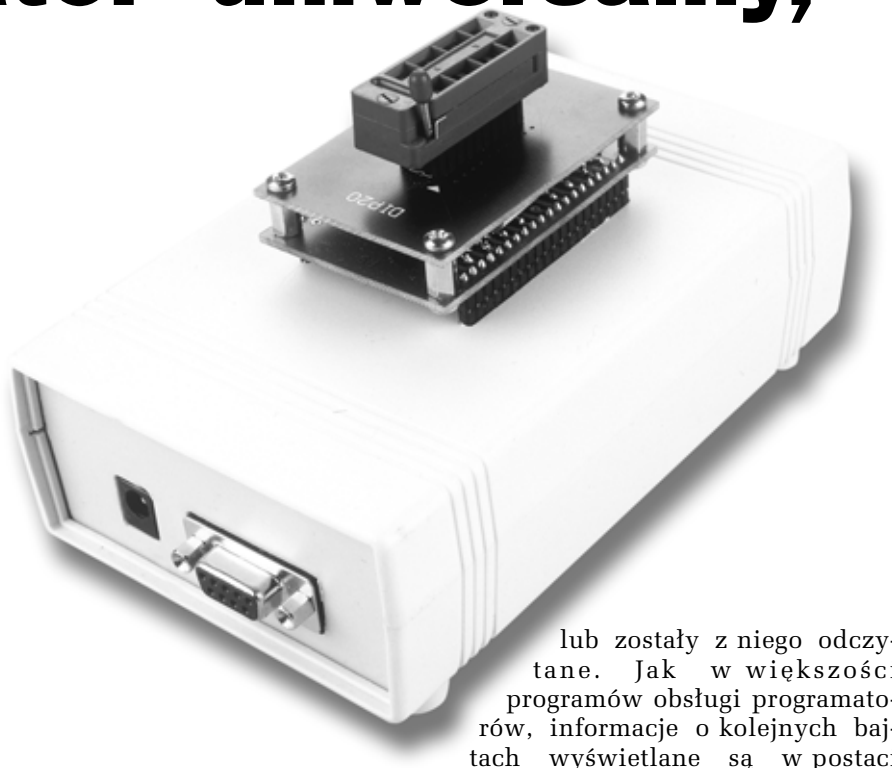
Ryszard Szymaniak, AVT
ryszard.szymaniak@ep.com.pl

Opis języka i ewentualnych zmian jest dostępny na stronie http://www.aries-rs.com.pl/femto.

Wzory płytek drukowanych w formacie PDF są dostępne w Internecie pod adresem: ***http://www.ep.com.pl/?pdf/grudzien02.htm*** *oraz na płycie CD-EP12/2002B w katalogu* ***PCB****.*

Punch Programator uniwersalny, część 2

AVT-5092



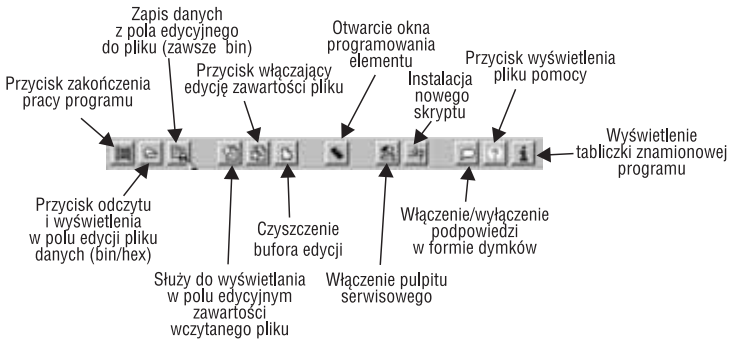
Drugą część artykułu poświęcamy przedstawieniu obsługi oprogramowania sterującego pracą programatora oraz zaleceniom, które warto uwzględnić podczas montażu urządzenia. ***Rekomendacje****: programator jest podstawowym przyrządem w pracowni elektronika, a więc ten opis może zainteresować większość naszych Czytelników.*

Obsługa programu

Po opisie działania programatora nadszedł czas, aby opisać jego program sterujący. Program ten jest odpowiedzialny za komfort obsługi przyrządu. O tym, jak trudno napisać program przyjazny, poprawnie działający z estetycznie wyglądającym interfejsem użytkownika miałem okazję przekonać się wielokrotnie. Program składa się z trzech wywoływanych kolejno pulpitów:

Pulpit edycji

Służy on do weryfikacji i edycji danych, które mają być zapisane w programowanym układzie lub zostały z niego odczytane. Jak w większości programów obsługi programatorów, informacje o kolejnych bajtach wyświetlane są w postaci heksadecymalnej oraz jednocześnie jako znaki ASCII. Ponad polem edycyjnym usytuowane są zwykle przyciski wyboru poszczególnych funkcji programatora i menu w trybie tekstowym, powielające funkcje przycisków. Na **rys. 2** opisałem ich funkcje.



Rys. 2. Opis belki narzędziowej programu sterującego

Oprócz tego, na pulpicie edycyjnym znajduje się kilka dodatkowych elementów. Na wysokości przycisków wyboru, po wczytaniu z dysku pliku danych wyświetli się jego rozmiar w bajtach. Nazwa i ścieżka dostępu do ostatnio wczytanego pliku wyświetli się na pasku statusu na dole pulpitu.

Poruszanie się po polu edycyjnym mają ułatwić elementy umieszczone po prawej stronie pulpitu (**rys. 3**). Są to: pasek przewijania i dwa duże klawisze. Wyświetlane standardowo pole edycyjne odpowiada rozmiarom pamięci programowanego elementu. Rozmiar zależy od typu elementu (np. w procesorze AT89C2051 można maksymalnie zaprogramować 2 kB pamięci programu) oraz od wybranej funkcji (sygnatura tego procesora to od-

Rys. 3. Główne okno programu z widocznymi paskami przewijania

czyt 2 bajtów). W przypadku bufora o rozmiarach do 1000h poruszanie się po nim i odnajdywanie interesujących fragmentów danych ułatwia suwak. Za pomocą dwóch dużych klawiszy można wybrać do wyświetlania kolejne segmenty danych o rozmiarze 1000h.

Pulpit wyboru funkcji programujących

Naciśnięcie przycisku otwierającego okno programowania (**rys. 4**) spowoduje przejście do pulpitu programowania. Za pośrednictwem tej części programu sterującego można zaprogramować element danymi znajdującymi się aktualnie w buforze edycji, odczytać do tego bufora dane z układu lub wywołać inną z dostępnych funkcji związanych z programowanym elementem np. jego kasowanie, jeżeli wyposażony jest w pamięć typu EEPROM lub Flash. Na pulpicie znajduje się sporo okienek wyświetlających informacje mające ułatwić pracę, jednak na początku mogą one trochę dezorientować operatora z powodu ich liczby. Są to głównie pola edycyjne informujące o podstawowych parametrach programowanego elementu i dostępnych opcjach. Tak jak w przypadku poprzedniego pulpitu podaję po kolei (poczynając od góry) opis i przeznaczenie każdego z jego elementów:

1. Okienko z wykazem dostępnych skryptów. Wybór skryptu poprzez dwukrotne kliknięcie nazwy spowoduje jego wczytanie. Od tego momentu programator jest gotów do programowania układów obsługiwanych przez wybrany skrypt. Wczytanie nowego skryptu spowoduje zazwyczaj zmianę zawartości pozostałych okienek znajdujących się na pulpicie.

2. W okienku z prawej strony wyświetlane są informacje, które autor skryptu przygotowanego zgodnie ze standardem języka FEMTO może w nim umieścić. Uważam, że w okienku powinna znaleźć się lista obsługiwanych przez skrypt typów elementów, informacja o producencie, typie adaptera. Ja takie informacje zamieściłem w swoich skryptach.

3. Poniżej znajduje się lista rozwijana z wykazem obsługiwanych przez skrypt elementów. W obecnej wersji oprogramowania każdy skrypt może opisywać procedury programowania do 8 elementów. Po rozwinięciu listy i kliknięciu nazwy wybranego elementu program sterujący dostosowuje swoje ustawienia do dokonanego wyboru. Ponieważ w obecnej wersji każdy skrypt opisuje elementy, których procedury programowania są identyczne, wybór oznacza w praktyce jedynie zmianę rozmiarów dostępnej do programowania pamięci. Na przykład dla procesora AT89C2051 będzie to 2 kB, natomiast w przypadku AT89C4051 rozmiar pamięci wynosi 4 kB.

4. Po wyborze elementu z listy, w sąsiednim okienku pojawią się zwięzłe informacje dotyczące wybranego elementu.

5. Poniżej linii rozdzielającej górę i dół pulpitu, z lewej strony znajduje się okienko z listą funkcji dostępnych dla wybranego układu. Dostępne funkcje to np. programowanie, weryfikacja, kasowanie.

6. Nazwa wybranej funkcji zostanie wyświetlona w okienku z prawej strony. Sposób, w jaki dokonany zostanie wybór, określają dwa przyciski znajdujące się pomiędzy okienkami.

7. Jeżeli aktywny jest przycisk z czerwoną strzałką, możliwy jest wybór pojedynczych opcji.

8. Jeżeli aktywny jest przycisk z plusem, można wybrać więcej niż jedną z dostępnych funkcji np. zapis + weryfikacja + zabezpieczenie przed odczytem. Oczywiście nie wszystkie kombinacje są sensowne, np. zapis + kasowanie + weryfikacja doprowadzi do wyświetlenia informacji o błędzie.

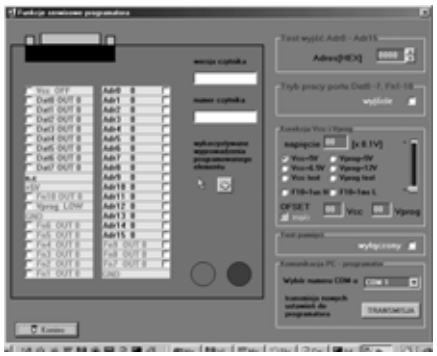
9. Jeszcze niżej, z lewej strony pulpitu znajdują się dwa okienka, w których wyświetlany jest adres początku i końca programowanego obszaru. Jeżeli mamy do czynienia z elementem o dużej pamięci, w której zapisywanych będzie na początku niewiele danych, można dla przyśpieszenia procesu programowania ograniczyć od góry zakres programowania do rozmiaru zapisywanego bloku. Znacznie rzadziej będzie stosowane ograniczenie dolnego zakresu programowania, ponieważ np. procesory wymagają, aby pewne elementy programu rozpoczynały się od zerowego adresu komórki pamięci.

10. Obok znajdują się dwa pola edycyjne. Alternatywne pole edycji (APE, alternatywne w stosunku do pola edycji na pierwszym pulpicie) przeznaczone jest do wyświetlania małej ilości danych w formacie heksadecymalnym lub tekstowym. Może mieć zastosowanie chociażby przy wyświetlaniu odczytanej sygnatury elementu. O tym, w którym oknie informacja ta zostanie wyświetlona decyduje skrypt elementu.

11. Bitowe pole edycji (BPE) przeznaczone zostało do wyświetlania i edycji także niewielkich porcji danych w postaci bitowej (dwójkowej). Taki tryb prezentacji danych jest najwygodniejszy w przypadku odczytu lub ustawiania bitów konfiguracyjnych elementu. Jeżeli takich bitów nie ma, pole pozostaje puste.

12. Jeszcze niżej znajduje się pasek pokazujący graficznie postęp realizowanej operacji, a w okienku obok wyświetla się liczba bajtów pozostałych do zaprogramowania lub odczytania. W przypadku błędu weryfikacji, w kolorze czerwonym wyświetlony zostanie adres pierwszego bajtu różnego od bajtu w buforze edycji.

Rys. 4. Widok okna programowania



Rys. 5. Widok okna serwisowego

13. Na samym dole, na dodatkowym pasku statusu, mogą pojawiać się komunikaty, np. o bieżącej lub zakończonej operacji.

14. Naciśnięcie klawisza z prawej strony pulpitu z napisem *Start*, oznaczonego symbolem diody LED, oznacza rozpoczęcie realizacji wybranej funkcji. Spowoduje to przesłanie do części sprzętowej odpowiednich rozkazów i najczęściej dołączenie zasilania do programowanego elementu, co sygnalizowane jest świeceniem się diody LED. Przerwanie procesu programowania jest możliwe po naciśnięciu na klawiaturze komputera klawisza *escape*. Część sprzętowa przerwie programowanie i wyłączy zasilanie elementu jeszcze w jednym przypadku - jeżeli zerwana zostanie komunikacja z programem sterującym i w ciągu kilku sekund z komputera PC nie zostanie przesłany żaden rozkaz.

15. Ostatni z elementów na pulpicie - klawisz z napisem *Koniec* - zamyka pulpit funkcji programujących i otwiera pulpit z polem edycyjnym.

Pulpit funkcji serwisowych

Projektując programator, postanowiłem rozbudować możliwości jego testowania i sprawdzania poprawności działania. Ułatwi to korzystanie z programatora, szczególnie w sytuacji, gdy użytkownik przygotowuje własny adapter, a potem chciałby go sprawdzić bez używania do tego celu często kosztownego elementu programowanego. Z tego powodu funkcje serwisowe umożliwiają sterowanie i sprawdzanie stanu poszczególnych wyprowadzeń złączy JP1 i JP2.

Z lewej strony pulpitu (**rys. 5**) wyświetlono w uproszczeniu obydwa złącza wraz z przyjętymi dla nich nazwami. Jeżeli przyjmiemy, że górę części sprzętowej programatora wyznacza złącze RS232, to z lewej strony znajdzie się złącze JP1, a z prawej JP2. Oprócz nazwy, obok oznaczenia każdego styku znajdzie się informacja o jego statusie: czy jest wejściem, wyjściem oraz jaki powinien być na nim poziom. W znajdujących się obok polach, po nawiązaniu łączności pomiędzy programem sterującym a częścią sprzętową programatora, wyświetlane są informacje o wersji programatora i jego numerze. Schematycznie zaznaczona zielona dioda LED zaświeci się na chwilę po każdej poprawnej transmisji między komputerem a programatorem. Czerwona dioda zaświeci się, gdy wyprowadzenie *Vcc* zaznaczymy jako *ON*, co będzie oznaczać, że po wysłaniu kolejnych danych powinien załączyć się przekaźnik podający zasilanie na programowany element. Jeżeli uprzednio wczytany został skrypt konkretnego elementu, to po naciśnięciu małego przycisku oznaczonego symbolem otwartej książki zamiast standardowych nazw złącz pojawi się opis nóżki układu scalonego, do którego powinno zostać podłączone wyprowadzenie za pośrednictwem adaptera i podstawki programującej.

Z prawej strony pulpitu znajdują się narzędzia służące do wymuszania (w celach testowych) odpowiednich stanów logicznych na wybranych wyprowadzeniach złącz. W przypadku wyprowadzeń adresowych można to zrobić dwoma sposobami. Np. kliknięcie na oznaczenie *Adr5* będzie cyklicznie przełączało jego poziom z wysokiego na niski i odwrotnie. Dla wszystkich wyjść adresowych ten sam efekt można uzyskać wpisując w polu edycyjnym dwubajtową liczbę, która zostanie przekształcona na odpowiednią kombinację zer i jedynek na wyjściach adresowych. Po kolejnej transmisji danych testowych, ustawiona kombinacja zostanie przepisana na wyprowadzenia sprzętowej części programatora. W ten sposób porównując występujący stan logiczny z zaprogramowanym, można sprawdzić poprawność działania wyjść programatora.

Pozostałe wyprowadzenia, zarówno *Dat0-7*, jak i *F1-10* mogą pełnić rolę wyjść i wejść. Wyboru trybu dokonuje się zaznaczając określone pole na pulpicie. W trybie wyjścia test przebiega podobnie jak dla wyprowadzeń adresowych. W trybie wejścia można na badane złącze JP1 lub 2 podać wybrany stan logiczny, który zostanie odczytany i zasygnalizowany przy opisie wyprowadzenia. Warunkiem powodzenia tego testu jest uprzednie ustawienie badanego wejścia na poziomie wysokim.

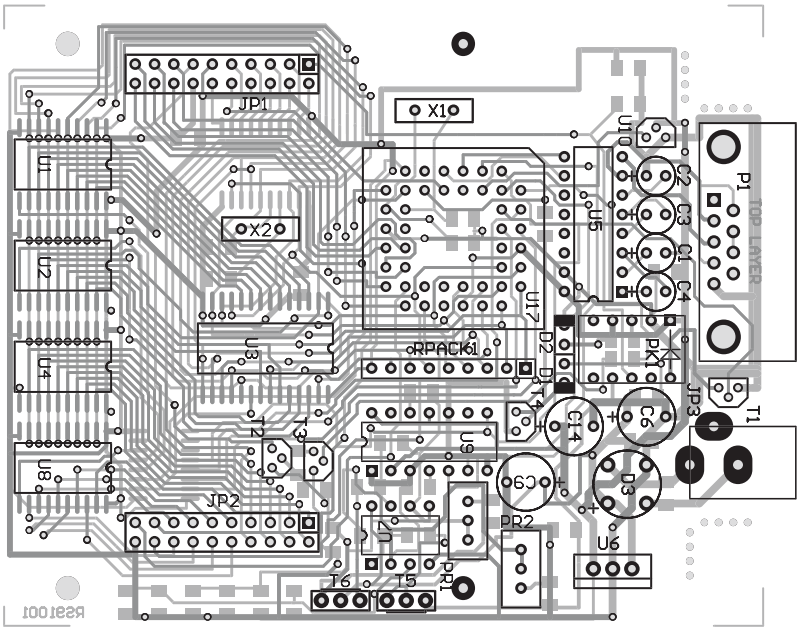
Następna sekcja na pulpicie służy do testowania wyprowadzenia napięcia zasilania *Vcc* i napięcia programowania *Vprog*. Gdy oba wyprowadzenia oznaczone są jako wyłączone, nie powinno występować na nich żadne napięcie. W czasie testu zarówno napięcia *Vcc*, jak i *Vprog* powinny być zaznaczone jako włączone, co jest sygnalizowane za pomocą symbolu świecącej się czerwonej diody LED. Użytkownik może wymusić na obu wyjściach dowolne napięcie z określonego w parametrach programatora zakresu z rozdzielczością 0,1 V. Oprócz tego, na wyprowadzeniu *F10* można wygenerować impuls o czasie trwania 1 µs lub 1 ms.

Jeszcze niżej, za pomocą listy rozwijanej użytkownik może określić, do którego portu COM będzie przyłączona część sprzętowa programatora. Ustawienie to będzie aktualne także w czasie normalnej pracy z programatorem. Naciśnięcie znajdującego się obok klawisza *Transmisja* spowoduje jednokrotne wysłanie rozkazów z ustawionymi wcześniej parametrami wyprowadzeń. Klawisz *Koniec* zamyka funkcje serwisowe i powoduje powrót do pulpitu edycji.

Kończąc swoją pracę, program sterujący w osobnym pliku konfiguracyjnym zapisuje bieżące ustawienia. Po jego ponownym uruchomieniu będą one przywrócone, co dotyczy m.in. wybranego skryptu, wczytanego ostatnio pliku danych czy numeru portu COM, który będzie używany przez programator.

Montaż programatora

Tym czytelnikom, których zainteresował przedstawiany projekt i przystąpią do budowy programa-



Rys. 6. Rozmieszczenie elementów na płytce programatora

tora, może przydać się parę moich rad wynikających z doświadczenia, jakie zdobyłem przy budowie trzech prototypów programatora.

Programator zaprojektowany został do montażu w plastikowej obudowie oznaczonej symbolem producenta Z50. Jest ona dostępna w kilku wariantach, ja wykorzystałem pudełka o zewnętrznej wysokości 43 mm lub 36 mm. W pudełku trzeba zamontować dwie płytki drukowane. Płytka główna (dwustronna) ma wymiary 100 x 81 mm. Dla zaoszczędzenia miejsca, wszystkie oporniki i kondensatory nieelektrolityczne są przeznaczone do montażu powierzchniowego, tak jak większość układów scalonych (**rys. 6**). Zastosowano elementy R i C w obudowach 1206. Tym z Czytelników, którzy nie mają dobrej lutownicy o cienkim grocie i cienkiej cyny, a przede wszystkim nie mają wystarczająco dużo cierpliwości, odradzam samodzielny montaż. Źle wlutowany element SMD trudno wylutować, a w przypadku układów scalonych oznacza to najczęściej ich zniszczenie. Po tym ostrzeżeniu przystępuję do wypunktowania czynności związanych z montażem. Moim zdaniem warto przestrzegać podanej kolejności, gdyż ułatwi to pracę:

1. Płytkę drukowaną przystosowujemy do wymiarów obudowy w ten sposób, że w obu narożnikach (od strony gniazda DB9) wycinamy kwadraty o wymiarach mniej więcej 10 mm (zaznaczono na płytce), aby zrobić miejsce na wewnętrznie prowadzone wkręty mocujące górę i dół obudowy.

2. Na dolnej stronie płytki lutujemy wszystkie oporniki i kondensatory SMD. Można najpierw zwilżyć cyną jeden z punktów lutowniczych i przylutować końcówkę elementu. Po sprawdzeniu, że przylega on do płytki i ewentualnej korekcie jego położenia, lutujemy jego drugą końcówkę. Radzę się nie śpieszyć, co zmniejszy ryzyko pomyłki. O ile bowiem oporniki mają wytrawione na powierzchni cyfry z oznaczeniem oporności, to kondensatory pozbawione są takich oznaczeń i mogą nie różnić się wyglądem przy zupełnie różnej pojemności.

3. Od dołu wlutowujemy jeszcze układ multipleksera U12 i chwilowo przestajemy się interesować dolną częścią płytki.

4. Na górze należy wlutować montowane powierzchniowo układy scalone U1, U2, U4, U8, U3. Tak jak w przypadku elementów RC, najpierw należy przylutować jedną, najlepiej skrajną nóżkę układu, sprawdzić jego położenie i dopiero potem lutować pozostałe nóżki.

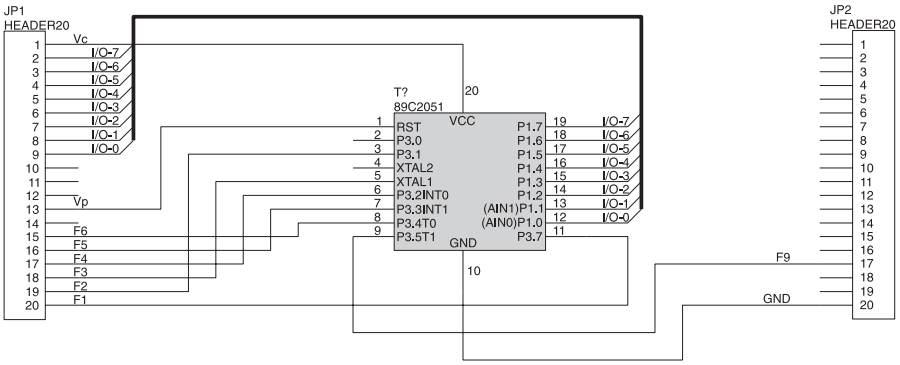
5. Wlutowujemy pozostałe elementy od najmniejszych do największych. Lutowanie małego tranzystora, gdy wokoło znajdują się wysokie elementy, może być bardzo stresujące. Proszę zwrócić uwagę na prawidłową lokalizację markera podstawki PLCC procesora. Ścięta krawędź powinna znaleźć się u góry po lewej stronie. Kwarc X2, jeżeli nie jest w niskiej obudowie, radzę położyć, co zapobiegnie jego wyłamywaniu przez kable. Przed jego wlutowaniem, obszar płytki drukowanej, do której może dotykać metalowa obudowa kwarcu, należy zabezpieczyć przez przyklejenie np. kawałka taśmy izolacyjnej. Jako ostatnie należy wlutować wysokie elektrolity i gniazdo DB9.

6. Dla poprawy warunków pracy stabilizatora U6 warto zwiększyć jego chłodzenie, stosując mały radiator.

7. Jako złącza JP1 i JP2 można zastosować dwurzędowe, kwadratowe szpilki stosowane w złączach komputerowych. Podobnych można użyć jako złącza dla diod LED. Zastosowanie złącz zamiast lutowanych na stałe przewodów ułatwi zarówno montaż, jak i demontaż programatora, co może się przydać szczególnie w fazie uruchamiania układu.

8. Następnie należy wykonać czynności mechaniczne związane z przystosowaniem obudowy do potrzeb programatora. W tym celu od strony gniazda RS-a i gniazda zasilającego w plastikowej płytce obudowy należy wyciąć prostokątne otwory na te gniazda.

9. Druga płytka programatora jest płytką jednostronną. Przewidziano na niej miejsce na dwa gniazda wyprowadzające na zewnątrz obudowy sygnały złączy JP1, JP2. W czasie normalnej pracy, do złączy będą wkładane adaptery programujące. Ponieważ gniazda te są przewidziane do intensywnej pracy (częste wkładanie i wyjmowanie adapterów), a jednocześnie muszą zapewniać pewny styk, muszą być dobrej jakości. Najodpowiedniejsze wydają się listwy tzw. gniazd precyzyjnych o średnicy ok. 0,8 mm (średnica większa niż gniazd precyzyjnych dla układów scalonych), których wewnętrzne styki pokryte są stopem z domieszką złota. Gniazda powinny być wlutowane dokładnie prostopadle do płaszczyzny płytki. Dodatkowo na płytce przewidziano miejsce na opor-



Rys. 7. Schemat elektryczny adaptera dla mikrokontrolerów 89C2051/4051

niki podciągające poziom niektórych sygnałów do napięcia +5V, co może być konieczne przy długich przewodach łączących płytkę główną programatora z płytką gniazd.

10. W górnej części obudowy należy wyciąć dwie równoległe szczeliny na opisywane wyżej gniazda. Dodatkowo trzeba wywiercić dwa otwory na diody LED o średnicy zależnej od wymiarów zastosowanych elementów i ich oprawek. Moim zdaniem najlepiej mocuje się diody o średnicy 5 mm. Otwory powinny znaleźć się na obudowie od strony operatora, aby w czasie pracy programatora diody były widoczne.

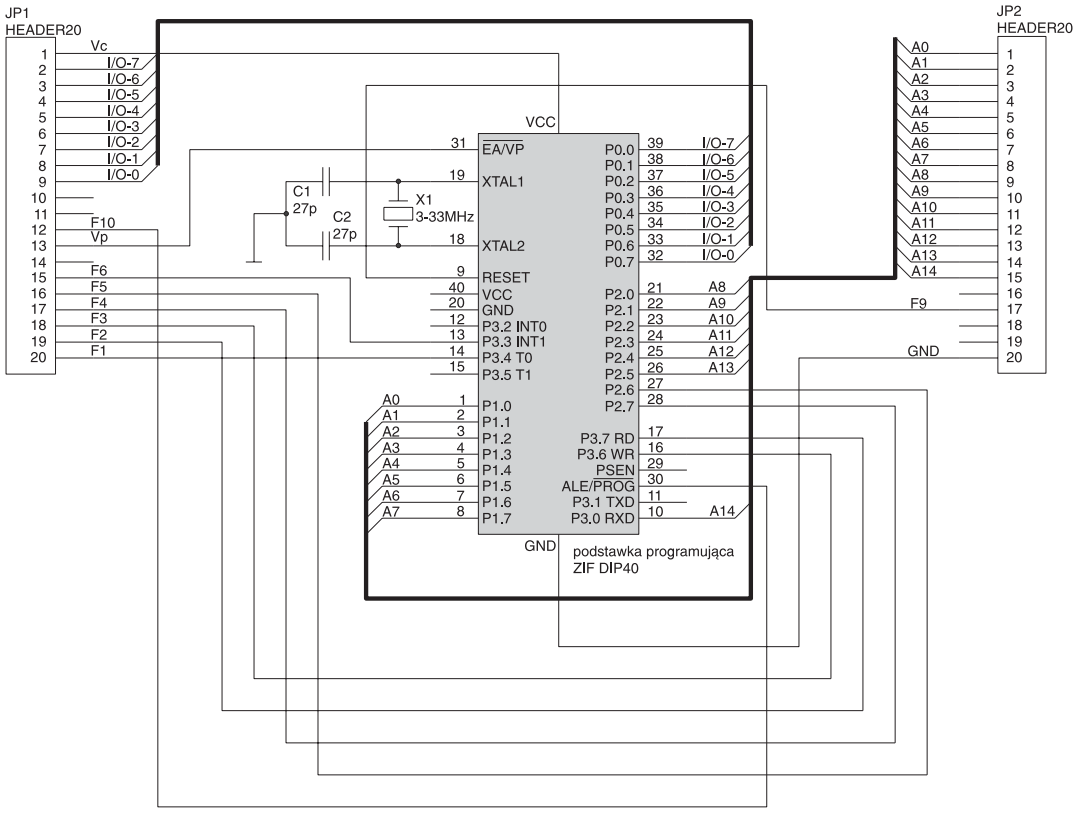
11. Do zamocowania płytki gniazd do górnej części obudowy (tej z dwiema wyciętymi wcześniej szczelinami) użyłem 4 wkrętów M3 o długości ok. 10 mm z łbami stożkowymi i kleju POXIPOL. Wkręty mocujące płytkę powinny być przyklejone do górnej części obudowy po jej wewnętrznej stronie. Miejsca klejenia należy koniecznie zmatowić albo zarysować, gdyż w przeciwnym przypadku żywica nie sklei wkrętów z plastikiem dostatecznie mocno. Klej powinien oblewać dolną część główki wkrętu, jednak nie może sięgać do części nagwintowanej. Zanim klej stężeje, korzystamy z płytki złącz i korygujemy położenie klejonych wkrętów tak, aby ich położenie pokrywało się z otworami w płytce. Oczywiście, trzeba to robić ostrożnie, żeby nie przykleić płytki do obudowy. Klej typu POXIPOL twardnieje już po 10 minutach. Przed upływem tego czasu wyjmujemy płytkę, mając pewność, że wkręty znajdują się na właściwym miejscu, natomiast płytka nie sklei się z obudową.

12. Do płytki złącz należy teraz dolutować przewody, które ją połączą ze stykami JP1 i JP2 płytki głównej. Takie połączenie można wykonać za pomocą dwóch odcinków 20-żyłowej taśmy o długości 8...10 cm zakończonych gniazdami zaciskowymi. Po rozdzieleniu z obu stron taśmy żył na długości 15...20 mm i usunięciu izolacji, końcówki należy pobielić cyną. Taka taśma nie lutuje się najlepiej i wskazane jest użycie kalafonii (sam topnik zawarty w lucie może nie wystarczyć). Taśmę należy ułożyć równolegle na płytce i dolutować do gniazd żyła po żyle, zaczynając od skrajnych i skracając w miarę konieczności środkowe. Takie ułożeniu taśm połączeniowych, które po skręceniu obudowy ułożą się w jej wnętrzu „esowato", zapobiegnie ich ruchom w miejscach lutowania ze złączem i w efekcie obrywaniu przewodów.

13. Ostatnią czynnością tej części montażu jest przykręcenie płytki złącz do górnej części obudowy za pomocą przyklejonych wkrętów oraz zamocowanie LED-ów z przewodami i wtykiem.

Budowa adapterów

Zadaniem wymiennych adapterów jest doprowadzenie sygnałów ze złącz JP1 i JP2 do podstawki programującej typu ZIF, a za jej pośrednictwem do odpowiednich wyprowadzeń programowanego elementu. Każdy adapter składa się z dwóch jednostronnych płytek drukowanych.



Rys. 8. Schemat elektryczny adaptera dla mikrokontrolerów 89C51/52

Dolna, w projekcie określona nazwą *adapter base*, jest taka sama dla wszystkich typów adapterów. Służy do wlutowania dwóch rzędów styków, które w momencie mocowania adaptera znajdą się w gniazdach JP1 i JP2. Utworzone w ten sposób złącze powinno zapewniać pewny styk, a jednocześnie pozwalać na rozłączenie bez używania nadmiernej siły. Najlepiej do tego celu nadają się listwy styków tzw. złoconych okrągłych o średnicy 0,8 mm. Gorsze są styki o przekroju kwadratowym. Zdecydowanie odradzam używanie styków pokrytych stopem w kolorze srebrnym. Tworzone połączenie jest co prawda niezawodne, ale wyjęcie adaptera z takimi stykami z gniazda wymaga dużej siły i może grozić uszkodzeniem złącz.

Druga (górna) płytka adaptera ma otwory mocujące przystosowane do użytej podstawki ZIF. Podstawki mogą się od siebie różnić tak, jak różnią się typy obudów programowanych elementów np. DIP, PLCC, FPGA itd. Różna może być także liczba wyprowadzeń, chociaż z reguły podstawki większej np. DIP40 można użyć do programowania elementów DIP o 20 nogach. Podstawki można albo na stałe wlutować do górnej płytki adaptera, albo osadzać je za pośrednictwem złączy wykonanych ze styków precyzyjnych. Tego typu rozwiązanie nie jest estetyczne, ale umożliwia wykorzystanie jednej podstawki ZIF (zwykle stosunkowo drogiej) w kilku adapterach.

Obie płytki są ze sobą łączone przewodami w sposób przypisany przez skrypt, z którym współpracują. Struktura pliku skryptu pozwala na umieszczenie w nim takich informacji przez programistę. Potem mogą być one wyświetlone przez program sterujący programatorem na pulpicie funkcji serwisowych po naciśnięciu przycisku z ikoną książki. Przykładowy schemat połączeń opisany w moich skryptach opracowanych dla programowania procesorów AT89C2051/4051 i AT89C51/52 przedstawiono na **rys. 7** i **8**.

Po wykonaniu połączeń obie płytki należy ze sobą złączyć, tworząc coś w rodzaju kanapki, z przewodami pomiędzy płytkami. Do mocowania można użyć 4 odcinków grubej srebrzanki i przylutować je od strony druku obydwu płytek. Można także użyć gwintowanych wewnętrznie tulejek dystansowych zamontowanych w czterech rogach pomiędzy obydwiema płytkami. Tulejki powinny mieć długość 6 - 8 mm. Wkręty użyte do takiego montażu powinny być oczywiście krótsze, o długości gwintu maksymalnie 5 mm.

Ryszard Szymaniak, AVT
ryszard.szymaniak@ep.com.pl

Opis języka i ewentualnych zmian jest dostępny na stronie http://www.aries-rs.com.pl/femto.

Wzory płytek drukowanych w formacie PDF są dostępne w Internecie pod adresem: ***http://www.ep.com.pl/?pdf/styczen03.htm****.*