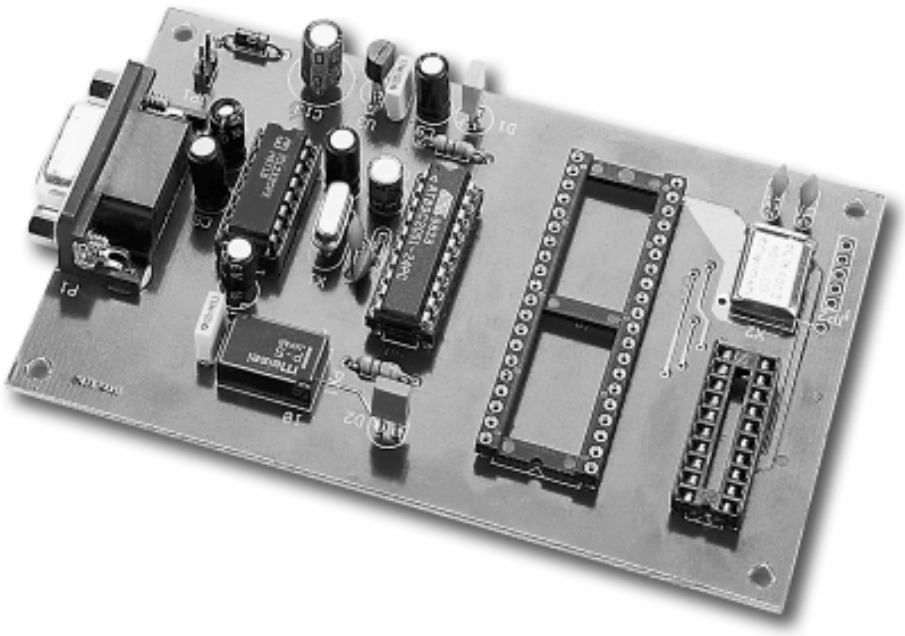
Programator procesorów AVR, część 1

kit AVT-812



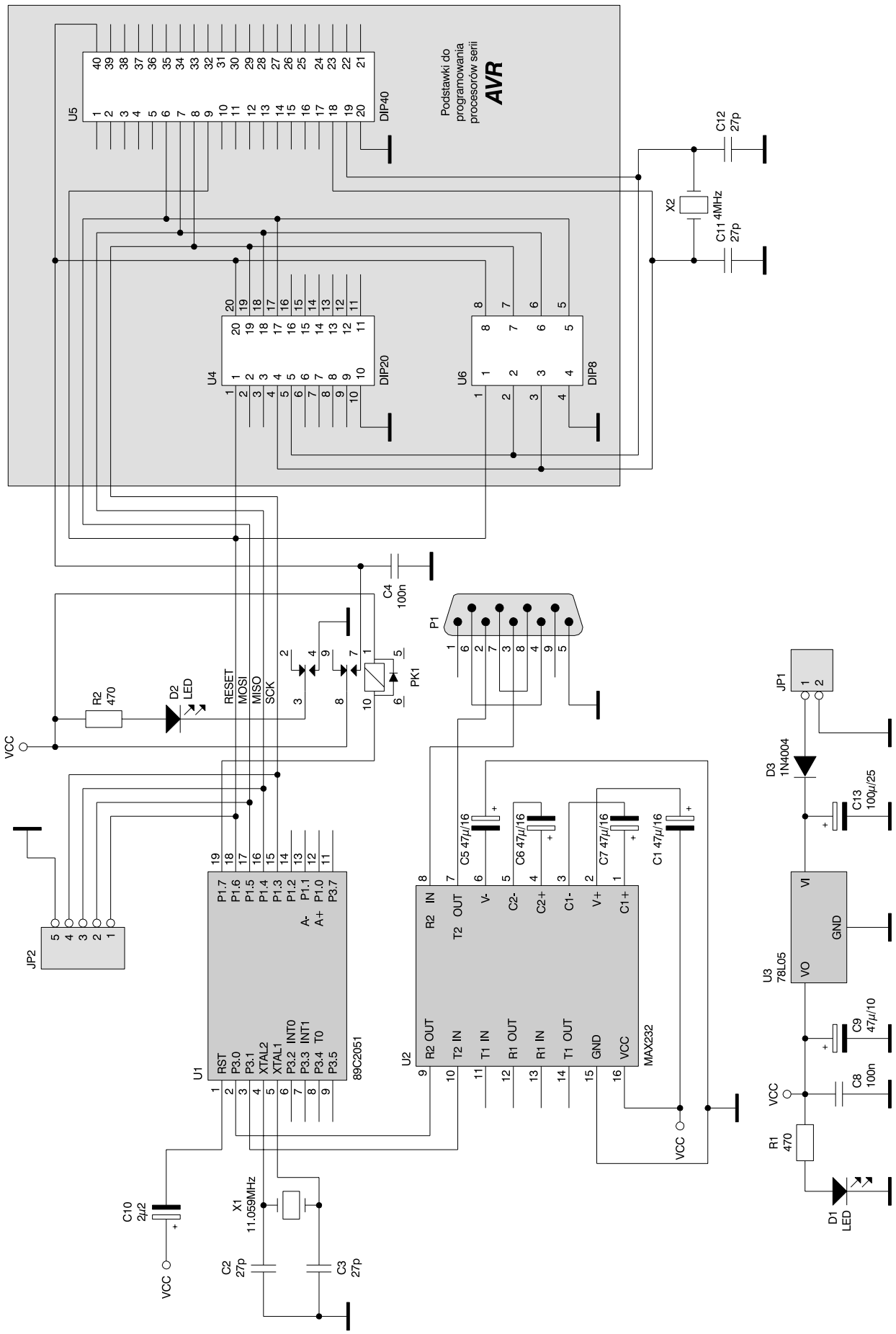
Procesory jednoukładowe zrobiły prawdziwą karierę w świecie elektroniki. Sukces ten wiąże się z rozwojem elektronicznego sprzętu powszechnego użytku. Im urządzenia stawały się łatwiejsze w użyciu, bardziej sprawne i niezawodne, tym bardziej rosło zapotrzebowanie na elementy sterujące ich pracą, czyli mikrokontrolery. Dotyczy to także układów automatyki przemysłowej. Dziś trudno spotkać urządzenie elektroniczne bez inteligentnego sterownika, będącego dalekim krewnym dużych komputerów.

Rozwój elekreonicznego sprzętu powszechnego użytku i możliwość ulokowania w nim swoich wyrobów stanowiła silny bodziec dla wielu firm zajmujących się produkcją układów wielkiej skali integracji.

Początkowo na rynku dominowały układy z grupy 8049, następnie słynny 8051 produkowany najpierw przez Intela oraz mikrosterowniki firmy Motorola. W latach 90. pojawiły się nowe rodzaje procesorów, nierzadko bardzo wyspecjalizowanych i zminiaturyzowanych.

Firma Atmel zaistniała na rynku najpierw ze swoją odmianą sterownika '51, w którym zastąpiono niewygodną, kasowaną ultrafioletem pamięć EPROM, elektrycznie programowaną pamięcią FLASH EEPROM. Taki sposób zapisu kodu programu do pamięci sterownika jest bardzo wygodny, zwłaszcza na etapie pracy nad programem i w produkcji małoseryjnej. W przypadku polskiego rynku można stwierdzić, że procesory te znalazły na nim swoje miejsce. Od ponad dwóch lat Atmel promuje nową rodzinę procesorów ochrzczonych wspólnym mianem AVR.

Procesory wchodzące w skład rodziny różnią się wielkością i możliwościami, jednak kilka cech pozostaje wspólnych. Najważniejszą z nich jest oparcie wewnętrznej budowy sterowników na architekturze RISC, bazującej na uproszczonej liście rozkazów wykonywanych najczęściej podczas jednego cyklu zegarowego. Powoduje to znaczne przyśpieszenie pracy układu, w którym jeden cykl zegara taktującego odpowiada jednemu cyklowi rozkazowemu. Określenie „uproszczona lista rozkazów“ wcale nie oznacza, że jest ona krótka, ponieważ wzbogacono ją o cały zestaw skoków warunkowych i trybów adresowania. W związku z tym wszystkie procesory wyposażone są w rozbudowany zestaw rejestrów uniwersalnych bezpośrednio współpracujących z akumulatorem, co dodatko-



Rys. 1. Schemat elektryczny urządzenia.

wo zwiększa szybkość działania układów.

Producent określa moc obliczeniową sterowników na równą 1MIPS przy częstotliwości zegara 1MHz. Maksymalna częstotliwość zegara dla większości typów procesorów AVR zawiera się w przedziale 10..24MHz. Kolejną cechą wspólną jest wyposażenie prawie wszystkich typów sterowników w wewnętrzne pamięci RAM i EEPROM oraz sprzętowy zegar watchdoga. Wszystkie procesory posiadają szeregowy interfejs SPI umożliwiający w prosty sposób ich programowanie oraz zapisywanie i odczytywanie danych do i z wewnętrznej pamięci EEPROM. Linie portów wyjściowych procesorów pozwalają na bezpośrednie sterowanie różnych układów zewnętrznych, np. diod LED, ponieważ w stanie niskim potrafią przyjąć prąd o wartości nawet 20mA. Konstruktorzy dużo uwagi poświęcili redukcji mocy pobieranej przez sterowniki, co umożliwia ich stosowanie w sprzęcie zasilanym bateryjnie.

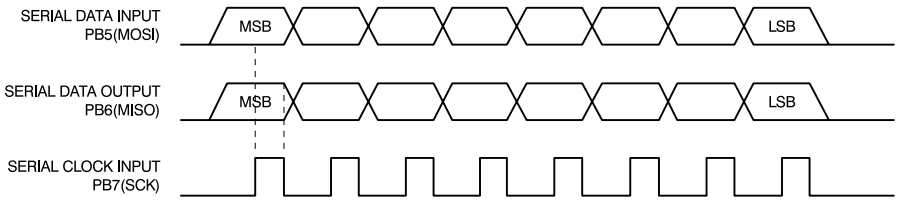
Pobór prądu w czasie normalnej pracy wynosi przeciętnie kilka miliamperów, natomiast w czasie uśpienia, gdy podtrzymywana jest zawartość wewnętrznych rejestrów i aktywny jest tylko wewnętrzny zegar watchdoga, pobór prądu wynosi jedynie 50µA.

Kolejną interesującą cechą wszystkich procesorów jest możliwość zakończenia trybu uśpienia poprzez podanie odpowiedniego poziomu napięcia na linii portu P3 i wygenerowanie przerwania.

Mamy nadzieję, że ten krótki wstęp zachęci Was do bliższego poznania procesorów AVR. Aby ułatwić Wam nieco to zadanie, w drugiej części artykułu omówimy nieco bardziej szczegółowo możliwości poszczególnych układów tej rodziny, a teraz przejdziemy do prezentacji konstrukcji programatora, opisywanego w artykule.

Opis układu

Schemat elektryczny programatora pokazano na **rys. 1**. Jego budowa jest bardzo prosta, ale można za jego pomocą zaprogramować praktycznie każdy rodzaj procesora AVR. Wynika to z faktu zastosowania w nim szeregowego interfejsu SPI.



Rys. 2. Przebieg sygnałów w czasie transmisji.

Wszystkie sterowniki z rodziny AVR mogą być programowane na dwa sposoby. Pierwszy z nich, który można określić jako tradycyjny, wykorzystuje do wymiany danych pomiędzy programatorem, a programowanym procesorem jeden z jego portów, a kilka dodatkowych sygnałów podawanych na linie pozostałych portów steruje całym procesem. Sposób ten wymaga zaangażowania wielu linii danych. Jeżeli programator ma obsłużyć procesory w różnych obudowach i o różnej liczbie wyprowadzeń, trzeba się liczyć z koniecznością stosowania adapterów lub multipleksowaniem wyprowadzeń programatora, w zależności od typu aktualnie programowanego procesora.

Drugi sposób wiąże się z wykorzystaniem w każdym typie procesora specjalnego szeregowego interfejsu o zredukowanej liczbie wyprowadzeń, w tym przypadku trzech. Zastosowanie do programowania jedynie trzech „drutów" - SPI (tak naprawdę dochodzi jeszcze zasilanie, linia RESET oraz doprowadzenia zegara) bardzo upraszcza procedurę programowania, a w pewnych warunkach umożliwia przeprogramowanie procesora nawet wtedy, gdy znajduje się w systemie, w którym pracuje.

Interfejs SPI składa się z następujących linii sygnałowych: linii zegara synchronizującego transfer informacji SCK, linii danych wejściowych MOSI i linii danych wyjściowych MISO. Przebieg sygnałów na tych trzech liniach w czasie transmisji pokazano na **rys. 2**.

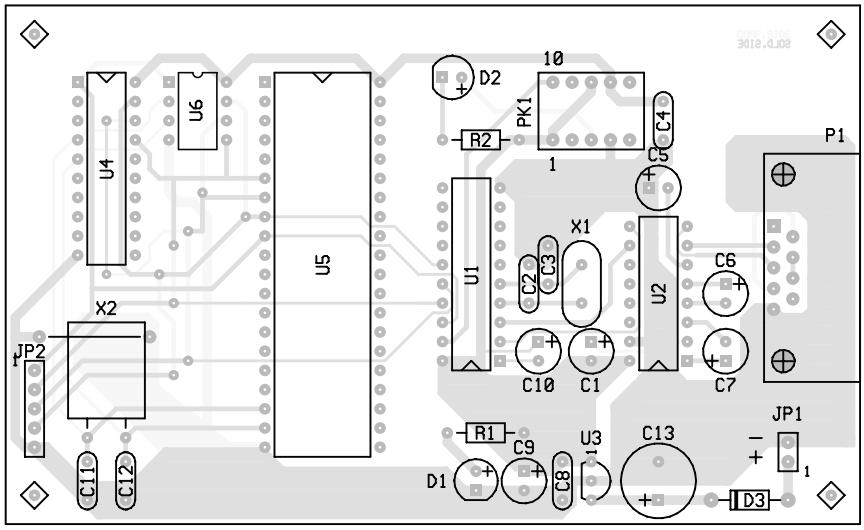
Programowanie polega na wysłaniu linią MOSI kodów sterujących i ewentualnie danych, które określą sposób w jaki ma się zachować programowany układ. Kody są to 3 lub 4 bajty wysyłane bit po bicie, z najstarszym bitem jako pierwszym. Odczytane dane procesor wysyła w ten sam sposób linią MISO, z najstarszym bitem jako pierwszym.

Kody sterujące wpisywane są do procesora podczas narastającego zbocza zegara SCK, natomiast dane pojawiające się na linii MISO mogą być odczytane podczas opadającego zbocza impulsu zegarowego. Sygnały na liniach MOSI i MISO mogą się zmieniać jedynie podczas stanu niskiego na linii zegarowej SCK.

Procesory AVR wyposażone w interfejs SPI reagują na kilka kodów sterujących. Ich format w przypadku procesora 90S2313 pokazano w **tab. 1**.

W celu rozpoczęcia korzystania z interfejsu SPI należy spełnić kilka prostych warunków. Przed podaniem napięcia zasilającego trzeba podać na wyprowadzenia procesora RESET i SCK stan niski oraz dołączyć do wyprowadzeń XTAL rezonator kwarcowy o nominalnej częstotliwości lub podać sygnał taktujący na wyprowadzenie XTAL1. Po włączeniu zasilania należy odczekać 20ms, a następnie wysłać cztery bajty rozkazu *Programing enable*. Od tego momentu procesor znajduje się w trybie programowania. Zapisanie do pamięci procesora nowego kodu programu wymaga wcześniejszego wykasowania zawartości pamięci FLASH, co nastąpi po wysłaniu rozkazu *Chip erase*, a następnie odczekaniu 10ms na zakończenie operacji kasowania pamięci.

Jednocześnie z pamięcią FLASH wykasowane zostaną wszystkie dane zapisane w pamięci EEPROM procesora. Zapis danych do pamięci programu wykonuje się za pomocą rozkazu Write Program Memory. Litery *a, b* oznaczają wyrażony binarnie adres komórki pamięci, do której zostanie dokonany zapis. Liczba znaczących bitów w przypadku procesorów o większej pojemności



Rys. 3. Rozmieszczenie elementów na płytce drukowanej.

pamięci FLASH będzie oczywiście większa niż w przykładzie odnoszącym się do procesora 90S2313. Ponieważ format rozkazów procesorów AVR jest 16-bitowy, a przesyłane bajty danych są 8-bitowe (4 bajt rozkazu oznaczony literami „i"), identyfikator „H" określa, która połówka kodu jest aktualnie transmitowana. Starsza i młodsza połówka kodu rozkazu zapisywane są pod jednakowym adresem *a, b*.

W czasie odczytu pamięci procesora adresowanie z wykorzystywaniem bitu „H" jest identyczne jak podczas zapisu. Różnica polega na tym, że po wysłaniu 3 bajtów linią MOSI, odczytywany bajt danych pojawi się na linii MISO.

Do zapisu i odczytu danych do i z pamięci EEPROM procesora służą rozkazy *Write EEPROM Memory* i *Read EEPROM Memory*, a cała wymiana danych przebiega podobnie jak w przypadku pamięci programu. Różnica polega na tym, że przed nowym zapisem nie ma konieczności czyszczenia pamięci rozkazem *Chip erase*.

Zaadresowana komórka EEPROM jest automatycznie czyszczona przed zapisem nowych danych. Następnie trzeba odczekać 4ms przed wysłaniem kolejnego kodu interfejsem SPI.

Rozkaz *Write Lock Bits* pozwala zaprogramować bity zabezpieczeń chroniące obie pamięci przed możliwością doprogramowania nowych danych, a także przed ich odczytaniem. Bity są aktywne, gdy przyjmują wartość 0. Bity mogą być skasowane jedynie w wyniku działania rozkazu *Chip erase*. Rozkaz *Read Device Code* odczytuje kod typu procesora. Zakończenie sesji programowania wymaga wyłączenia zasilania oraz pozostawienia wyprowadzenia RESET na poziomie wysokim, gdy zasilanie zostanie włączone ponownie.

Dzięki temu, że programator wykorzystuje w swoim działaniu interfejs SPI, jego budowa może być stosunkowo prosta. Zasadniczym elementem urządzenia jest procesor 89C2051, który kontroluje przepływ danych liniami interfejsu oraz włącza i wyłącza przekaźnik PK1 dołączający napięcie zasilania do programowanego procesora. Pracą programatora steruje komputer poprzez standardową linię RS dołączaną do gniazda P1. Parametry transmisji portem RS to: szybkość 9600 bodów, 8 bitów danych, brak bitu parzystości oraz 1 bit stopu. Układ scalony U2 dokonuje konwersji poziomów logicznych sygnałów do standardu TTL.

W założeniu programator miał być jak najprostszym urządzeniem współpracującym z zewnętrznym komputerem klasy PC. Z tego powodu oprogramowanie procesora U1 potrafi jedynie obsługiwać interfejs SPI oraz rozróżnia 3 rozkazy sterujące, co zupełnie wystarczy, aby prawidłowo zapisać i odczytać dane z wszystkich procesorów AVR.

Każdy z rozkazów składa się z kilku bajtów danych. Najpierw wysyłany jest znak litery (w formacie ASCII) określający rodzaj rozkazu, następnie jego parametry, dane, a na końcu bajt sumy kontrolnej. Bajt ten powstaje poprzez wykonanie operacji XOR na wszystkich kolejnych bajtach rozkazu z wyłączeniem oczywiście samego bajtu sumy kontrolnej. Po prawidłowym wykonaniu każdej operacji programator potwierdza ten fakt wysyłając w odpowiedzi literę „A". Wysłanie jakiegokolwiek innego znaku lub brak odpowiedzi powinien być interpretowany przez komputer sterujący jako błąd.

Rozkazy sterujące i opis poszczególnych bajtów:

1.Rozkaz ustawienia parametrów programatora:

„S"*abk*0*c*

„S" - kod ASCII (53h) identyfikatora rozkazu

ab - dwa bajty określające adres początkowy, od którego programator rozpocznie odczytywanie lub zapisywanie danych do procesora

k - parametr określający sposób pracy programatora. Bajt ten może przyjmować następujące wartości:

0 - następne rozkazy odczytu lub zapisu będą dotyczyły pamięci programu procesora

1 - następne rozkazy będą dotyczyły pamięci EEPROM procesora

2 - programator powinien uaktywnić bity zabezpieczające programowanego procesora

FFh - programator powinien się zresetować

0 - bajt o stałej wartości równy zero

c - bajt sumy kontrolnej

2.Rozkaz odczytu danych:

„R"*xc*

„R" - kod ASCII (52h) identyfikatora rozkazu

x - liczba bajtów danych, które mają być odczytane z pamięci procesora

c - bajt sumy kontrolnej

3.Rozkaz zapisu danych do pamięci procesora:

„W"*xd...dc*

„W" - kod ASCII (57h) identyfikatora rozkazu

x - liczba bajtów danych *d*, które mają być zapisane do pamięci procesora (FLASH lub EEPROM, co zależy od paramet-

ru *k* wysłanego we wcześniejszym rozkazie ustawienia parametrów)

d - bajty danych, których liczba została określona parametrem *x*. Liczba danych może być dowolna lecz nie większa niż 32

c - bajt sumy kontrolnej

Programator po odebraniu rozkazu ustawienia parametrów, przede wszystkim ustawia swój wewnętrzny licznik danych zgodnie z wartością przekazaną parametrami *ab*. Licznik ten po każdym zapisie lub odczycie pamięci procesora AVR jest zwiększany o jeden. Z tego powodu ustawienie początkowego adresu, np. podczas odczytu całej dostępnej pamięci FLASH procesora, można przeprowadzić tylko raz, wysyłając na początku rozkaz S*abk*0*c*. Programator zapamiętuje także parametr *k* i do czasu jego zmiany wszelkie odczyty lub zapisy będą dotyczyć wybranego typu pamięci.

Potwierdzenie wykonania rozkazu przez programator w przypadku rozkazu odczytu jest nieco zmodyfikowane. Po wysłaniu kodu litery „A" (41h) programator wysyła także odczytane dane w liczbie określonej w rozkazie odczytu parametrem *x*, dołączając na końcu bajt sumy kontrolnej *c*. Po wykonaniu każdego rozkazu odczytu danych lub zapisu, programator przez ok. 0,6s podtrzymuje w stanie załączenia przekaźnik PK1. Jeżeli w tym czasie nie zostanie odebrany kolejny rozkaz, styki przekaźnika zostaną rozłączone i programowany procesor AVR bez obaw można wyjąć z podstawki. Wysłanie polecenia resetu programatora powoduje natychmiastowe rozłączenie styków i przejście programatora w stan oczekiwania na kolejny rozkaz.

Przykładowa sekwencja rozkazów, dotycząca zapisania do pamięci FLASH nowego programu dla sterownika AVR, a potem sprawdzenia czy zapis został dokonany poprawnie, może wyglądać następująco:

S, 0, 0, 0, 0, *c* (ustawienie początkowego adresu zapisu na 00h i typu pamięci na FLASH)

W,20h,*d,...,d,c* (zapis bloku 32 bajtów)... (zapis kolejnych bloków danych)

W,32,*d,...,d,c*

S,0,0,FFh,*c* (reset programatora)

S,0,0,0,0,*c*

R,20h,*c* (odczyt danych z pamięci programowanego procesora w celu weryfikacji)

R,20h,*c*

S,0,0,FFh,*c* (reset programatora)

Podany opis powinien okazać się wystarczający do napisania własnego programu sterującego pracą programatora. Wszystkim, którzy nie mają ochoty tworzyć go samodzielnie proponujemy jego funkcjonalną, prostą wersję pracującą w środowisku Windows95 (wchodzi w skład kitu).

Montaż i uruchomienie

Konstrukcja mechaniczna programatora jest bardzo prosta. Na płytce drukowanej (rozmieszczenie elementów na **rys. 3**) oprócz innych części znajdują się także podstawki do osadzania procesorów AVR na czas programowania. Zastosowanie 3 typów podstawek pozwala programować niemal wszystkie typy układów. Dodatkowo sygnały interfejsu SPI wyprowadzone są na gniazdo JP2. Procesor przed rozpoczęciem programowania lub czytania należy umieścić w podstawce z odpowiadającą mu liczbą styków (na fotografii pokazana jest starsza wersja programatora, jedynie z dwoma typami podstawek). Wcześniej płytkę programatora należy połączyć z odpowiednim gniazdem portu RS komputera standardowym kablem oraz zasilić napięciem stałym o wartości 8..12V dołączanym do gniazda JP1.

Po uruchomieniu na komputerze programu sterującego i rozpoczęciu czytania lub zapisu procesora AVR, przekaźnik na płytce zostanie załączony, co sygnalizuje zapalenie się diody D2. Po zakończeniu programowania dioda zgaśnie i procesor może być wyjęty z podstawki. Jednocześnie może być programowany tylko jeden procesor. W programatorze najlepiej użyć podstawek ze złączami precyzyjnymi, które nie ulegną zniszczeniu na skutek częstego wkładania i wyjmowania programowanych procesorów.

Ryszard Szymaniak, AVT

Tab. 1.

Kod sterujący	Bajt1	Bajt2	Bajt3	Bajt4
Programing enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx
Chip erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx
Read Program Memory	0010 H000	xxxx xxaa	bbbb bbbb	oooo oooo
Write Program Memory	0100 H000	xxxx xxaa	bbbb bbbb	iiii iiii
Read EEPROM Memory	1010 0000	xxxx xxxx	xbbb bbbb	oooo oooo
Write EEPROM Memory	1100 0000	xxxx xxxx	xbbb bbbb	iiii iiii
Write Lock Bits	1010 1100	111x x21x	xxxx xxxx	xxxx xxxx
Read Device Code	0011 0000	xxxx xxxx	xxxx xxbb	oooo oooo

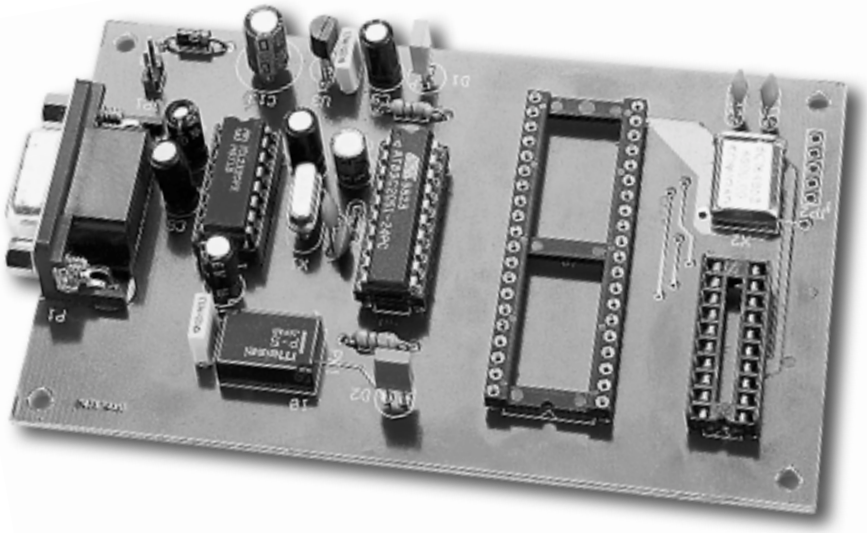
gdzie:
a, b binarnie określony adres pamięci z 'a' oznaczającymi starsze bity
H bit określający czy chodzi o młodszy [0] czy starszy [1] bajt kodu programu (dane zapisywane są do pamięci programu jako bajty, natomiast w ALU przetwarzane jako 16-bitowe słowa),
o dana odczytywana z wyjścia MISO
i dana zapisywana do pamięci
1, 2 bity zabezpieczenia przed odczytem
x ustawienie tak oznaczonego bitu nie jest istotne

WYKAZ ELEMENTÓW

Rezystory
R1, R2: 470Ω
Kondensatory
C1, C5, C6, C7, C10: 47µF/16V
C2, C3, C11, C12: 27pF
C8, C4: 100nF
C10: 2,2µF
C13: 100µF/25V
Półprzewodniki
D1, D2: LED czerwona i zielona
D3: 1N4004
U1: 89C2051 zaprogramowany
U2: MAX232
U3: 78L05
Różne
PK1: przekaźnik miniaturowy 5V typu OMRON
P1: złącze DB9 żeńskie do druku
X1: 11,059MHz
X2: 4MHz
U4: precyzyjna podstawka DIP20
U5: precyzyjna podstawka DIP40
U6: precyzyjna podstawka DIP8

Programator procesorów AVR, część 2

kit AVT-812



W drugiej części artykułu o programatorze AVR postaramy się dostarczyć nieco wiedzy o samych procesorach, podzielimy się także kilkoma praktycznymi uwagami dotyczącymi ich właściwości oraz sposobów pisania programów asemblerowych.

Wydaje nam się to potrzebne, zwłaszcza w przypadku nowych układów, a takimi są na rynku procesory AVR. Każdy zainteresowany i tak samodzielnie będzie musiał się nauczyć nowych procesorów, warto jednak już na początku wiedzieć, czy wysiłek może się opłacać i jakich korzyści można się spodziewać stosując nowe układy.

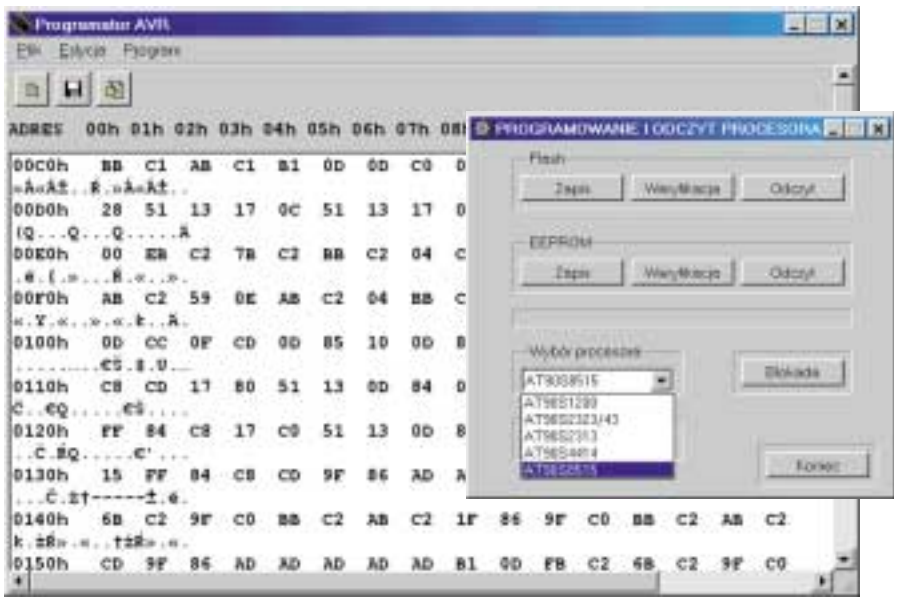
Na początku wrócimy jeszcze do samego programatora. Jak zostało to powiedziane w pierwszej części artykułu, programator współpracuje z komputerem PC, który jest sterowany przez program nadzorujący proces zapisu danych do pamięci układu AVR.

Program ten steruje programatorem za pomocą opisanych wcześniej 3 rozkazów, spełnia także rolę interfejsu, za pomocą którego użytkownik może decydować co i jak zapisać lub odczytać z pamięci programowanego procesora. Korzystając z informacji dostępnych w pierwszej części artykułu, każdy może samodzielnie stworzyć taki program. Dla pozostałych, którzy nie chcą lub nie mogą napisać programu dla PC-ta, przygotowaliśmy jego wersję działającą w środowisku Windows95. Na **rys. 4** przedstawiono ekran pracującego programu w trybie zapisu lub odczytu danych z procesora. Programik jest bardzo prosty, ale w zupełności wystarcza do sterowania płytki programatora oraz pozwala na podstawowe manipulacje danymi.

Wybierając odpowiednią opcję w menu „Plik“ albo wybierając kursorem ikonę zapisanej kartki można otworzyć zbiór zawierający dane do zapisu do pamięci procesora. Zbiór może mieć postać INTEL HEX (format pliku generowany przez program asemblera) lub danych w postaci binarnej.

Zawartość odczytanego pliku można wyświetlić na ekranie wybierając opcję „Edycja“ lub klikając na ikonę piszącej dłoni. Otwarty zbiór można także zapisać na dysku lub dyskietce (tylko w formacie binarnym) klikając na ikonę dyskietki lub wybierając opcję „Zapisz“ menu „Plik“.

Polecenie „Program“ uaktywnia opcje związane bezpośrednio z programatorem. Pojawiające się nowe okienko udostępnia szereg klawiszy, których naciśnięcie rozpoczyna zapis, weryfikację lub odczyt danych z pamięci programu (Flash) procesora lub z pamięci EEPROM. Klawisz „Blokada“ służy do wydania polecenia zaprogramowania bitów zabezpieczających przed odczytem danych z pamięci, a klawisz „Koniec“ zamyka sesję programowania i pozwala powrócić

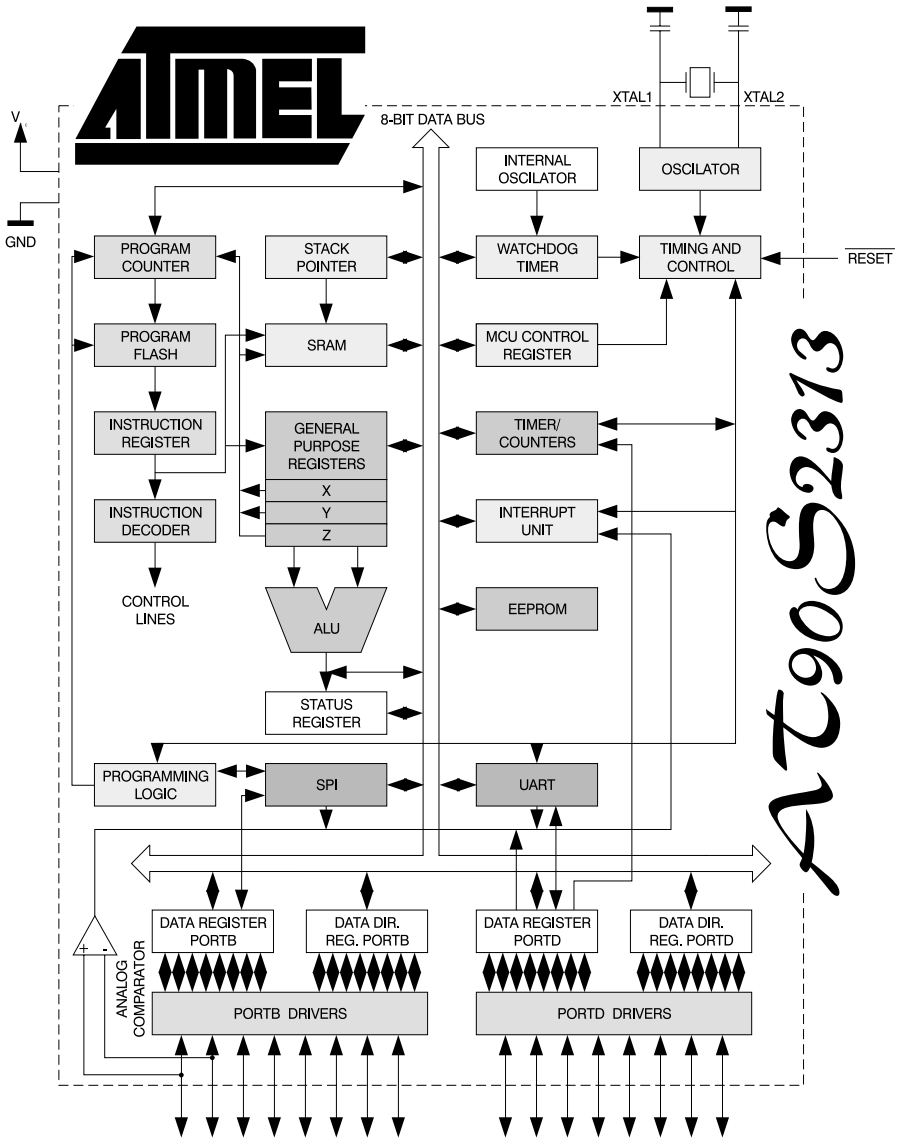


Rys. 4. Okno programu sterującego pracą programatora.

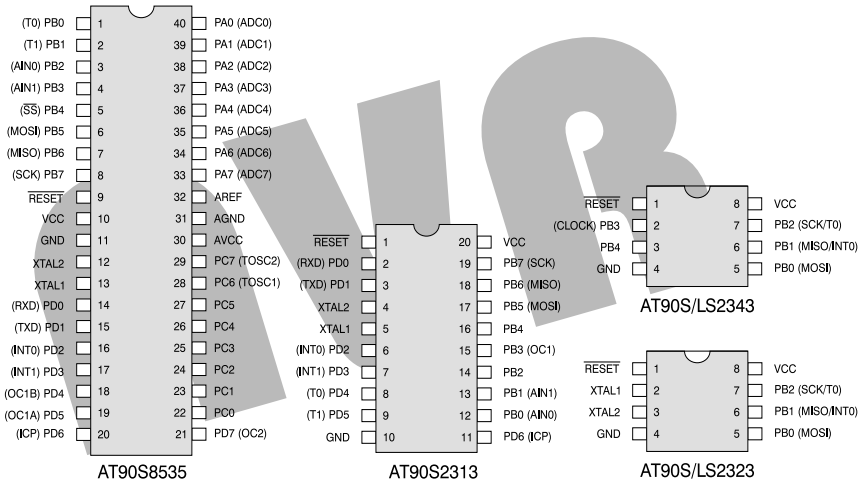
interesujących nas plików. Można tam znaleźć zarówno asemblery jak i symulatory pozwalające za pomocą komputera śledzić zachowanie napisanego przez nas oprogramowania oraz podglądać zawartość rejestrów i pamięci procesora, co umożliwia wykrycie błędów. Dostępne są także przykładowe programy i procedury napisane w języku asemblera. Najważniejsze pliki to: ASMPACK.EXE, ASM.ZIP, ASTUDIO.EXE. Na stronie internetowej można także znaleźć dokładne informacje techniczne związane z konkretnym typem procesora. Brak tam niestety najprostszego chociażby kompilatora języka C, którego użycie stanowi duże ułatwienie podczas pisania profesjonalnego oprogramowania. Z do-

do edycji danych. Dwie rozwijane listy służą do wyboru typu procesora, który będzie programowany oraz portu COM1 lub COM2, do którego dołączona zostanie płytka programatora.

Zmuszenie procesora do wykonania jakiejkolwiek sensownej pracy polega na stworzeniu dla niego programu, którego kody zostaną zapisane w pamięci Flash procesora za pomocą naszego programatora. W czasie pisania programu asemblerowego używa się nazw symbolicznych. Potem specjalny program przekształca polecenia i nazwy symboliczne na kody bezpośrednio przetwarzane przez procesor. Program taki nazywany jest potocznie asemblerem (wiedzą o tym doskonale Czytelnicy, którzy kiedykolwiek mieli do czynienia z programowaniem, a powyższe uwagi skierowane są do nowicjuszy, którzy dopiero od niedawna interesują się procesorami i sterownikami jednoukładowymi). W przypadku procesorów AVR program asemblera rozpowszechniany jest nieodpłatnie przez firmę ATMEL. Asembler oraz kilka innych programów narzędziowych i ciekawych przykładów oprogramowania dla procesorów AVR można znaleźć na stronie internetowej firmy pod adresem *www.atmel.com*. Na tej stronie kierując się następującym kluczem: PRODUCTS/AVR 8-bit RISC/SOFTWARE, dotrzemy do



Rys. 5. Schemat blokowy procesora AT90S2313.



Rys. 6. Wyprowadzenia niektórych procesorów serii AVR.

stępnych informacji wynika, że kompilator dla sterowników AVR oferuje firma IAR, co jest wiadomością dobrą i złą. Dobrą ponieważ narzędzia tej firmy są profesjonalnie przygotowane i cieszą się uznaniem, a złą, ponieważ zazwyczaj są bardzo drogie i praktycznie niedostępne dla zwykłego śmiertelnika. Należy mieć tylko nadzieję, że producenci układów w swoim własnym interesie będą wspierali powstawanie taniego oprogramowania narzędziowego, zachęcając w ten sposób do wykorzystywania w konstrukcjach elektronicznych właśnie ich procesorów.

Fenomen popularności procesorów '51 wiąże się głównie z dostępnością oprogramowania narzędziowego dla tego procesora. Asembler o nazwie WAVRASM pracuje w systemie Windows i jego użycie jest stosunkowo proste. Po uruchomieniu programu należy otworzyć nowy dokument posługując się w tym celu ikoną pustej kartki albo wczytać wcześniej napisany program, który będziemy chcieli zmienić lub poprawić. Początkujący zechcą się zapewne posłużyć dostarczonymi przez firmę wzorami programów i opierając się na tych przykładach napiszą swój własny, pierwszy program dla procesora AVR. Generalnie dobrze jest pamiętać o kilku, następujących zasadach:

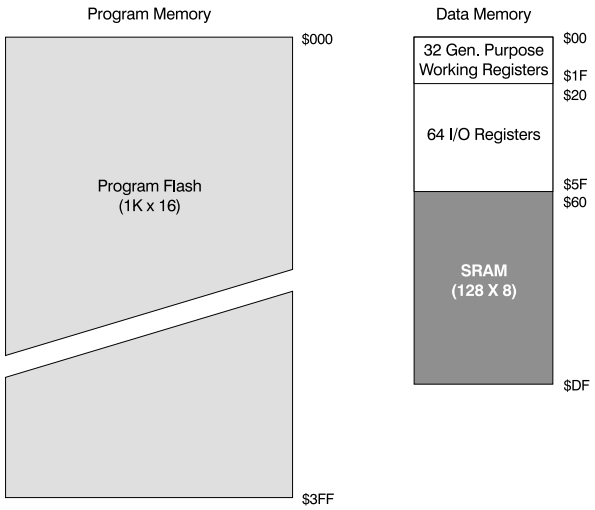
1. Każda linia programu asemblerowego może składać się z pewnych elementów, których położenie w jej obrębie nie jest obojętne. Na pierwszej pozycji w nowej linii mogą znaleźć się etykiety, czyli nazwy symboliczne. Program może odwoływać się do etykiet, tak jak np. do konkretnych adresów w pamięci programu.

Nazwa etykiety zakończona jest dwukropkiem (:). Na początku linii dopuszczalne jest także umieszczanie dyrektyw czyli specjalnych poleceń sterujących działaniem samego programu asemblującego. Nazwa dyrektywy poprzedzona jest bezpośrednio znakiem kropki (.). Dalej, po co najmniej jednej spacji za etykietą lub początkiem linii może pojawić się instrukcja, która w trakcie asemblacji zostanie przetłumaczona przez asembler na serię kodów sterujących działaniem procesora. W linii programu może pojawić się jeszcze komentarz, czyli tekst umieszczany przez programistę. Komentarz służy do przypomnienia w przyszłości, podczas przeglądania programu, jak funkcjonują jego poszczególne części, zmienne itd. Im liczniejsze i dokładniejsze są komentarze, tym mniej potem kłopotów ze zrozumieniem działania własnego programu. Komentarz poprzedzony jest znakiem średnika (;) i może się znaleźć po co najmniej jednej spacji za instrukcją lub zajmować całą linię.

2. Do programu powinien zostać dołączony, specjalną dyrektywą, plik definicji np. .INCLUDE „1200def.inc“. Plik definicji jest plikiem tekstowym, w którym za pomocą dyrektywy .EQU przypisano określone wartości liczbowe zarezerwowanym nazwom rejestrów i bitów. Np. jeden z rejestrów sterujących portem B procesora znajduje się pod fizycznym adresem 18h. Pisząc program dużo łatwiej zapamiętać i odwoływać się do jego nazwy symbolicznej (w pliku definicji określonej jako .EQU PORTB =$18) niż do konkretnego adresu. Plik definicji zawiera wszystkie takie nazwy. Plik ten powinien znaleźć się w tym samym katalogu w którym znajduje się poddawany asemblacji plik źródłowy. Można wykorzystać gotowe pliki źródłowe podawane w przykładach albo napisać taki plik samodzielnie.

3. Każdy program powinien zawierać na początku winietkę wykonaną za pomocą linii komentarza. W winietce powinna znaleźć się nazwa programu, zwięzły opis jego funkcji, oznaczenie wersji i ewentualnie inne uwagi. O przydatności takiej winietki przekonamy się bardzo szybko, gdy uzbiera nam się kilka napisanych wcześniej programów asemblerowych. Po pewnym czasie bardzo łatwo zapomnieć co właściwie dany program miał robić i jakich w nim dokonaliśmy zmian w stosunku do innych wersji. OPŁACA SIĘ TAKŻE ZAPISYWAĆ ROZBUDOWANE I DOKŁADNE KOMENTARZE!

Po napisaniu programu należy dokonać jego asemblacji używając polecenia „Assemble“. W przypadku powodzenia wyświetlone zostanie okienko komunikatów zakończonych informacją o braku



Rys. 7. Mapa pamięci procesorów AVR.

błędów. W przeciwnym razie w okienku pojawią się ostrzeżenia wskazujące linie programu, w których występują błędy.

Polecenie „Options“ pozwala ustalić format danych generowanych przez program asemblera. Dane przeznaczone dla naszego programatora powinny być utworzone w formacie Intela, a plik powinien mieć rozszerzenie HEX. W programie dostępny jest rozbudowany plik pomocy dobrze opisujący zarówno składnię poprawnie napisanego programu źródłowego jak i jego poszczególne elementy.

Ostrzeżenia wyświetlane przez program WAVRASM pozwalają wyeliminować błędy składni, przekręcone nazwy rozkazów itp., natomiast nie uchronią nas przed błędami w konstrukcji logicznej programu, które sprawiają, że zaprogramowany procesor nie dział tak, jak tego oczekujemy. To najtrudniejsze do wychwycenia błędy, bo nasze własne. Przy ich usuwaniu pomocne mogą okazać się programy AVR SIMULATOR lub AVR STUDIO, które na komputerze PC „udają“, czyli symulują sposób działania zaprogramowanego procesora. Dzięki obserwacji tego działania, wykonywaniu pojedynczych instrukcji, ustawianiu pułapek i podglądaniu zawartości symulowanych rejestrów procesora, dużo łatwiej odkryć w programie miejsca, które go prowadzą w przysłowiowe maliny niż tylko poprzez żmudne przeglądanie zapisanych linii kodu.

Każdy program napisany dla procesora AVR musi uwzględniać jego możliwości wynikające z wewnętrznej budowy. Poszczególne typy procesorów w obrębie rodziny mogą się między sobą znacznie różnić, chociażby liczbą wyprowadzeń, i nie zawsze program napisany dla jednego procesora da się uruchomić na innym. Generalnie jednak struktura wewnętrzna wszystkich sterowników jest podobna.

Jako przykład może posłużyć schemat blokowy mikrokontrolera AT90S2313 pokazany na **rys. 5**. Centralne miejsce przypada jednostce arytmetyczno-logicznej ALU oraz zespołowi rejestrów uniwersalnych. Instrukcje programu w postaci 16-bitowej, podawane są do ALU i rejestrów uniwersalnych z pamięci programu adresowanej przez licznik Program Counter. Oprócz tych elementów, do wewnętrznej magistrali dołączone są bloki statycznej pamięci (SRAM), pamięci EEPROM, układ watchdoga, interfejs SPI oraz układy, których występowanie zależy od konkretnego typu procesora: liczniki, interfejs szeregowy UART (czyli RS232), blok przerwań itd. Od typu procesora zależy także liczba buforów portów wejścia/wyjścia. Praca wewnętrznych układów sterownika AVR przebiega w takt impulsów ze stabilizowanego kwarcem oscylatora, który w pewnych modelach może być zastąpiony przez wewnętrzny generator o stałej częstotliwości 1MHz, obywający się bez zewnętrznych elementów.

Ponieważ rodzina sterowników AVR wciąż się rozrasta, dla porównania przedstawiamy poniżej listę kilku reprezentatywnych jej członków wraz z zestawieniem ich najważniejszych z punktu widzenia użytkownika cech. Dokładniejszych informacji należy zawsze szukać w dokumentacji technicznej dostępnej chociażby na stronie internetowej producenta.

Na **rys. 6** pokazano rozkład wyprowadzeń obudów wybranych typów procesorów. Dostęp do programowalnych układów wewnętrznych procesora (np. liczników) oraz portów, za pomocą których procesor komunikuje się ze światem zewnętrznym, realizowany jest poprzez rejestry I/O. Ich adresy oraz adresy 32 rejestrów uniwersalnych znajdują się we wspólnej przestrzeni adresowej wewnętrznej pamięci RAM procesora. Mapę adresów dla układu 90S2343 pokazano na **rys. 7**. W przypadku innych procesorów zmianie ulega tylko najwyższy adres pamięci RAM, co wynika z jej rozmiarów. Wyjątkiem jest tu układ 90S1200, który oprócz bloku rejestrów uniwersalnych nie posiada wewnętrznej pamięci RAM.

Pierwsze próby pisania programów dla procesorów AVR skłaniają do podzielenia się kilkoma spostrzeżeniami z tymi czytelnikami, którzy także spróbują wykorzystać w swoich urządzeniach te sterowniki. Ze względu na różnice w wewnętrznej budowie różnych typów procesorów, nie zawsze ich listy rozkazów są identyczne. Dotyczy to zwłaszcza instrukcji skoków i wywołań podprogramów. I tak np. w procesorze 90S1200 brak jest rozkazu IJMP, czyli skoku pośredniego, adresowanego rejestrem Z. Asembler nie wykaże błędu składniowego natomiast procesor „obdarzony“ instrukcją, której nie rozumie zacznie działać w sposób trudny do przewidzenia.

Tab. 2. Zestawienie podstawowych właściwości wybranych procesorów AVR.

Oznaczenie procesora / Właściwość	90S2323	90S2343	90S1200	90S2313	90S4414	90S8515	ATmega603
pamięć programu (kB)	2	2	1	2	4	8	64
pamięć RAM (B)	128	128	-	128	256	512	4096
pamięć EEPROM (B)	128	128	64	128	256	512	2048
liczba linii wejścia/wyjścia	3	5	15	15	32	32	32+8 WY+8 WE
SPI	tak	tak	tak	tak	tak	tak	tak
UART	-	-	-	tak	tak	tak	tak
timer/licznik	1	1	1	2	2	2	3
wewnętrzny oscylator RC	-	tak	tak	-	-	-	-
PWM	-	-	-	1	2	2	2
zabezpieczenie przed odczytem	tak	tak	tak	tak	tak	tak	tak
liczba wyprowadzeń	8	8	20	20	40	40	64

Wszystkie prezentowane procesory posiadają rozbudowany zestaw rejestrów ogólnego przeznaczenia. Istnieją jednak różnice w sposobie użycia rejestrów należących do 1 i 2 połówki zestawu. Do rejestrów R0-R15 nie można w sposób bezpośredni zapisać wartości stałej. Żeby to uczynić należy posłużyć się pośrednictwem któregoś z rejestrów z drugiej części zestawu. Może to wyglądać następująco:

```
LDI  R16,156
; wpisanie do rejestru
; pośredniczącego wartości 156
MOV  R1,R16
; przepisanie wartości
; z|rejestru pośredniczącego
; do rejestru R1
```

Wszystkie procesory (z wyjątkiem AT90S1200) posiadają stos, który może być umieszczony w dowolnym miejscu pamięci RAM. W momencie włączenia zasilania wskaźnik stosu, czyli rejestr SPL, inicjowany jest wartością zero. Dopóki nie korzystamy ze stosu (nie wywoływane są podprogramy i przerwania), to takie ustawienie wskaźnika nie jest problemem. Jednak jeżeli do rejestru SPL nie wpiszemy odpowiedniego adresu, pierwszy zapis na stosie spowoduje zniszczenie zawartości rejestrów, na który SPL będzie wskazywał. Trzeba o tym pamiętać i wpisać do SPL adres pamięci RAM, w której umieszczony zostanie stos.

Procesory AVR posiadają oczywiście możliwość realizacji przerwań programowych. Po zaistnieniu sytuacji wywołującej przerwanie, licznik programu procesora ustawiony zostaje na wektor przerwania wskazujący na podprogram realizujący funkcje przerwania. Wektory te umieszczone są na początku przestrzeni adresowej procesora. Jednak różne typy procesorów z rodziny AVR cechują się różną liczbą możliwych przerwań, co wynika z ich budowy i możliwości. Jest to oczywiście zrozumiałe, bowiem procesor pozbawiony np. portu szeregowego nie może wykonywać procedury przerwania generowanej przez ten port. Jednak występuje tu pewna niekonsekwencja. Nawet jeżeli procesor będzie wyposażony w układ, który w innym typie procesora wywołuje takie samo przerwanie, to wektory przerwania w obu typach procesorów nie muszą znajdować się pod tym samym adresem. Konstruktorzy układu zrezygnowali z zasady przypisania na stałe tych samych adresów tym samym wektorom przerwań i należy o tym pamiętać.

Pojawiające się wątpliwości związane ze sposobem działania programu i procesora najłatwiej rozwiać posługując się symulatorem i obserwując efekty działania programu.

W przyszłości procesorom AVR i układom z ich użyciem zamierzamy jeszcze poświęcić trochę miejsca na łamach naszego pisma. Przygotowywane są proste urządzenia wykorzystujące ciekawe cechy procesorów, jakimi są szybkość działania i mały pobór mocy pozwalający na zasilanie układów z baterii. Mamy nadzieję, że także czytelników EP zainteresuje ten temat i spróbują sami napisać ciekawe programy dla procesorów AVR.

Ryszard Szymaniak, AVT