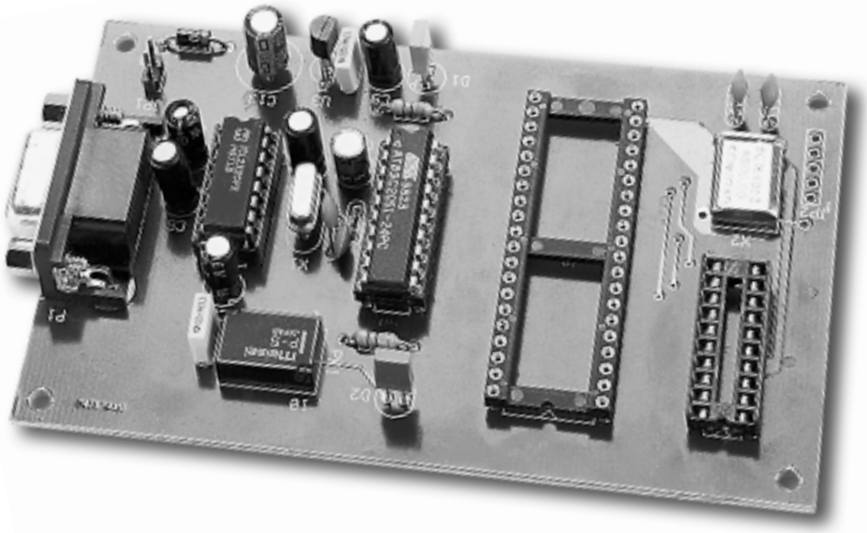
Programator procesorów AVR, część 2

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.

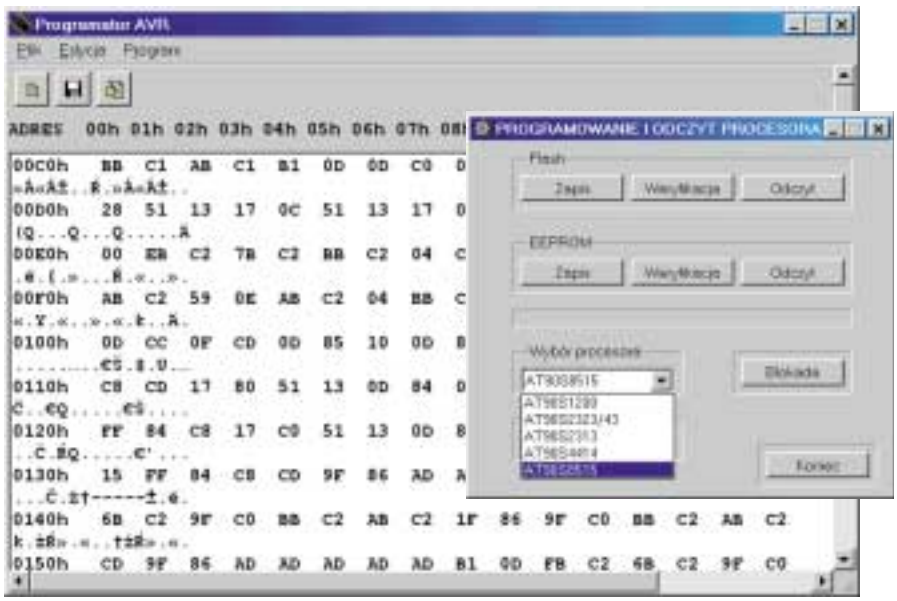
Na początku wrócimy jeszcze do samego programatora. Jak zostało to powiedziane w pierwszej części artykułu, programator współpracuje z komputerem PC, który jest sterowany przez program nadzorujący proces zapisu danych do pamięci układu AVR.

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

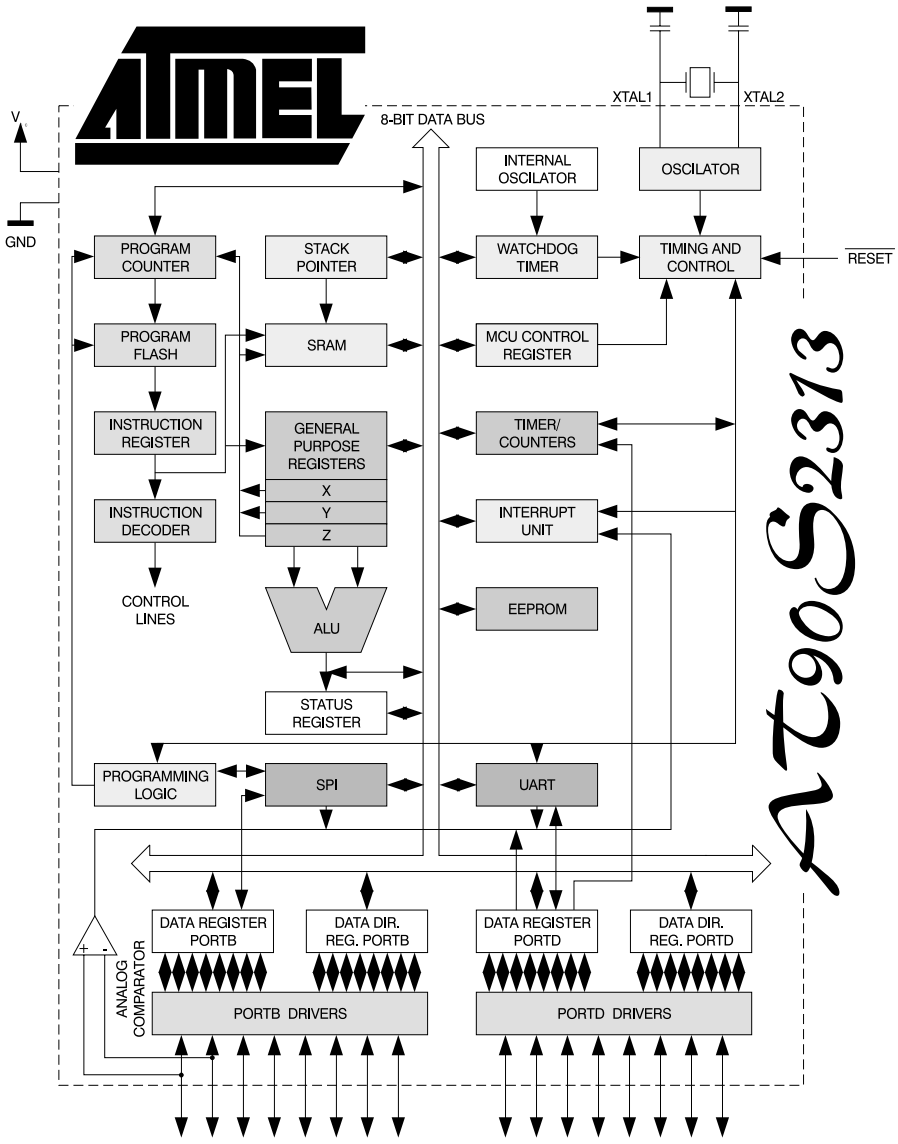


Rys. 4. Okno programu sterującego pracą programatora.

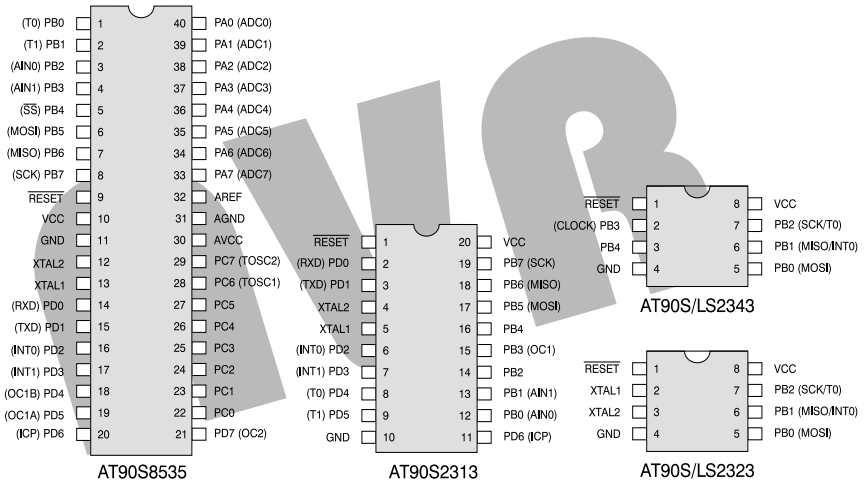
interesujących nas plików. Można tam znaleźć zarówno asemblery jak i symulatory pozwalające za pomocą komputera śledzić zachowanie napisanego przez nas oprogramowania oraz podglądać zawartość rejestrów i pamięci procesora, co umożliwia wykrycie błędów. Dostępne są także przykładowe programy i procedury 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:

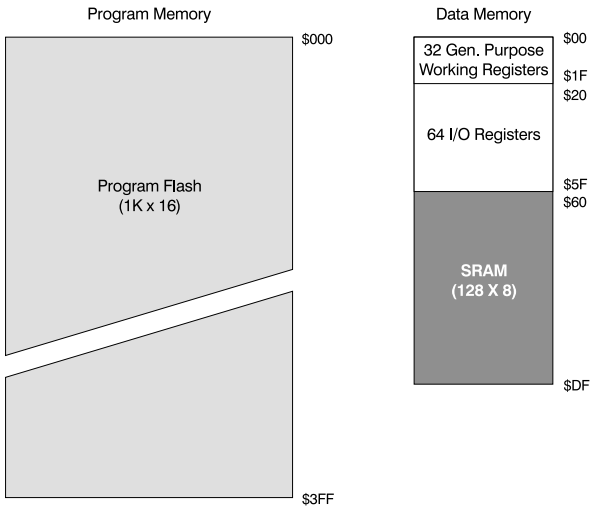
1. Każda linia programu asemblerowego może składać się z pewnych elementów, których położenie w jej obrębie nie jest obojętne. Na pierwszej pozycji w nowej linii mogą znaleźć się etykiety, czyli nazwy symboliczne. Program może odwoływać się do etykiet, tak jak np. do konkretnych adresów w pamięci programu.

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