

W bascomowym „kąciku” będziemy się starać w miarę przystępnie przedstawiać rozwiązania problemów napotykanym przez naszych Czytelników podczas pisania programów w Bascomie. Rubryka ta powstała z myślą o rozwiązywaniu problemów, na jakie najczęściej napotykają programiści, zatem zachęcamy wszystkich Czytelników do zgłaszania problemów na jakie się natknęli podczas tworzenia własnych programów.

Konfiguracja portów mikrokontrolera AVR Stosowanie aliasów

W pierwszym odcinku cyklu przedstawimy proste i zasadnicze dla posługiwania się mikrokontrolerami zagadnienie, jakim jest sposób konfiguracji portów mikrokontrolerów AVR. Ze względu na odmienną od '51 budowę mikrokontrolerów AVR, konfigurowanie portów w Bascom AVR różni się od konfiguracji portów w Bascom 51, w którym linię portu - aby pracowała jako wejście - wystarczyło ustawić w stan logicznej „1”. Problemy z konfiguracją portów mikrokontrolerów AVR przez początkujących wynikają zazwyczaj z braku wiedzy o działaniu, budowie i konfiguracji mikrokontrolerów AVR, które funkcjonalnie różnią się znacznie od popularnych '51.

Mikrokontrolery AVR, ze względu na swoją budowę wymagają aby linie portów, które będą wykorzystywane w urządzeniu były odpowiednio skonfigurowane - musi to zrobić programista, najlepiej na początku programu. Na rys. 1 przedstawiono jako przykład budowę jednej linii portu mikrokontrolera AVR. Na rysunku widać, że do konfiguracji portu przeznaczono dwa rejestry: DDRx oraz PORTx, które umożliwiają konfigurację oraz obsługę linii I/O. Należy mieć na uwadze, że linie portów mikrokontrolera mogą pełnić także alternatywne funkcje - np. mogą być wejściem przetwornika A/C, analogowego komparatora, timera itd.

Na początku programu (choć także w jego dowolnej części można zmienić konfigurację linii portu) należy zadeklarować czy używane wyprowadzenie (port) będzie pełnić funkcję wejścia czy też wyjścia mikrokontrolera. Służy do tego specjalny rejestr DDRx, który możemy oprogramować „wprost” (wpisując bezpośrednio do niego wartość) lub w sposób „bascomowy”, który jest bardziej czytelny i łatwiejszy do

opanowania dla początkującego. W Bascom AVR używamy do tego celu następujących instrukcji:

```
CONFIG PORTx = [INPUT/OUTPUT]
```

```
CONFIG PINx.y = [INPUT/OUTPUT]
```

gdzie: x - numer portu, y - numer końcówki

Jeśli podamy parametr INPUT, wtedy końcówka będzie wejściem, analogicznie OUTPUT - skonfiguruje port lub daną linię jako wyjście. Proszę o zwrócenie uwagi na różnice w składni: polecenie CONFIG PORTx konfiguruje cały 8-bitowy port jako wejście lub wyj-

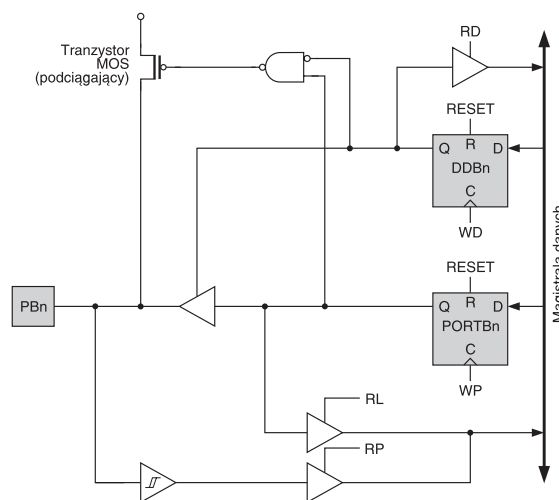
rzystane dwie instrukcje: SET oraz RESET. Jeśli np. nóżka portu mikrokontrolera skonfigurowana jest jako wyjście, wtedy stan wysoki ustawiamy instrukcją SET PortX.Y. Stan niski wymuszamy instrukcją RESET PortX.Y. Na przykład, aby ustawić linię B.2 mikrokontrolera należy użyć instrukcja: SET PortB.2, a jeśli chcemy zmienić stan linii B.0 na niski, to należy użyć instrukcji RESET PortB.0. Bascom udostępnia także instrukcję TOGGLE, która zmienia stan danej końcówki portu na przeciwny. Na przykład instrukcja TOGGLE PortB.0 zmieni na przeciwny stan linii B.0 mikrokontrolera.

Pozornie bardziej skomplikowana wydawać się może obsługa wejścia, ale tak w praktyce nie jest. Jeśli linia jest skonfigurowana jako wejściowa, wtedy (wynika to z wewnętrznej budowy mikrokontrolera) powinniśmy zadeklarować: czy linia jest wejściem „podciągniętym” do VCC czy też „pływającym”. Jeśli stan linii portu ustawimy poleceniem SET PortX.Y, wtedy na linii X.Y w stanie spoczynku pojawi się napięcie VCC

AVR-y od środka
Szczegółowe informacje o budowie rdzenia mikrokontrolerów AVR oraz opisy ich wybranych bloków funkcjonalnych można znaleźć w książce „Mikrokontrolery AVR w praktyce”, dostępnej w ofercie handlowej AVT pod symbolem KS-230929.

ście czyli CONFIG PORTB=INPUT ustawi wszystkie 8 końcówek portu B jako wejścia. Jeśli natomiast chcemy ustawić pojedyncze nóżki jako wejścia/wyjścia należy użyć drugiej składni: CONFIG PINx.y. Czyli aby ustawić linię B.2 jako wyjście, polecenie konfiguracyjne jest następujące: CONFIG PortB.2 = OUTPUT. Jeśli linia B.1 ma pracować jako wejście, to polecenie konfiguracyjne będzie następujące: CONFIG PortB.1 = INPUT. Jak widać, możliwe są dwie opcje: INPUT - gdy linia ma być wejściem lub OUTPUT, gdy linia ma być wyjściem. Można także użyć wartości bitowej (0 lub 1) aby szybko skonfigurować poszczególne linie portu.

W Bascomie do ustawiania oraz zerowania danych linii mikrokontrolera ustawionych jako wyjście mogą być wyko-



Rys. 1. Przykład budowy jednej z linii portu mikrokontrolera AVR

List. 1. Program sterujący diodą LED w zależności od stanu na wejściu D.5

```

CONFIG PinB.2= OUTPUT      'wyjście do którego podłączona jest dioda
CONFIG PinD.5 = INPUT      'wejście do którego podłączony jest mikroswitch
Set Portb.2                 'teraz ustawiamy na obydwu linach stan wysoki
                             'tu żeby dioda po włączeniu nie świeciła
Set Portd.5                 'a tu podciągamy port

Do                           'pętla główna programu
  If Pind.5 = 0 Then         'porównujemy czy na końcówkę D.5 podano masę
                             '(naciśnięty przycisk)
    Reset Portb.2           'ustaw stan niski na Portb.2 - co spowoduje zapalenie
                             'diody
  Else                       'w przeciwnym wypadku
    set PortB.2             'ustaw na portB.2 z powrotem na stan wysoki
                             'czyli zgaś diodę
  End if
LOOP                         'koniec nieskończonej pętli
    
```

(wewnętrzny rezystor podciągający zostanie dołączony do danej linii). Natomiast, jeśli nie skonfigurujemy wejścia, dołączając do niego wewnętrzny rezystor podciągający (np. poprzez ustawienie danego bitu w rejestrze *portx*) lub wydamy polecenie RESET PortX.Y, wtedy na linii wejściowej nie będzie (tak jak w wypadku wyjścia) „pewnej“ masy, lecz wejście będzie „wisieć“ w powietrzu. Na takiej końcówce wejściowej nie będzie stabilnego, jednoznacznie określonego stanu logicznego. Taką konfigurację linii można wykorzystać np. gdy z jakichś przyczyn wejście musi być „ściągnięte“ przez zewnętrzny rezystor do masy. Rezystory „podciągające“ ustalają stan spoczynkowy linii wejściowych i uodparniają je także - w pewnym stopniu - na zakłócenia. Dlatego też zaleca się, aby podczas korzystania z wejść konfigurować je jako linie z aktywnymi rezystorami „podciągającymi“ lub zastosować zewnętrzne rezystory „podciągające“ (lub „ściągające“, zależnie od aplikacji). Oczywiście rezystory podciągające należy używać, gdy na wejściach portów będzie zewnętrznie podawany tylko jeden z dwóch możliwych stanów logicznych, co będzie mieć miejsce np. w przypadku dołączenia przycisków do linii I/O mikrokontrolera. W przypadku sterowania

wejścia z układu cyfrowego, na którego wyjściu pojawiać się będzie na zmianę stan wysoki i niski, rezystory podciągające nie są potrzebne.

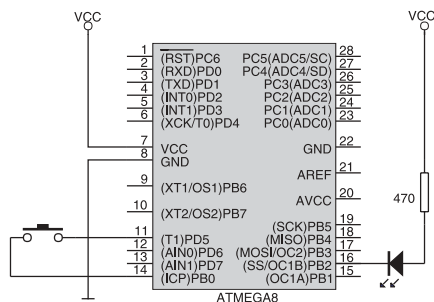
Tak więc, jeśli chcemy zapewnić zawsze stabilny stan na wejściu mikrokontrolera, należy po skonfigurowaniu portu ustawić jego stan na wysoki, najlepiej instrukcją SET PortX.Y, co powoduje dołączenie rezystora „podciągającego“.

Aby odczytać stan na wejściu mikrokontrolera posługujemy się poleceniem: *zmienna = PinX.Y* (na przykład odczyt linii D.5: *zmienna=PinD.5*). Należy pamiętać, że do odczytu stanu danej linii wejściowej nie służy rejestr *portx*, lecz rejestr *pinx*.

Pozbierajmy zatem wszystko w jakiś konkretny przykład: mikrokontroler ma sterować diodą LED włączoną między „+“ zasilania (VCC) a B.2, w zależności od stanu wejścia D.5 (linia PD5 jest więc wejściem, natomiast PB2 wyjściem). Na rys. 2 przedstawiono schemat elektryczny takiego układu z wykorzystaniem mikrokontrolera ATmega8, którym często będziemy się posługiwać w przykładach przedstawianych w tej rubryce.

Program przedstawiony na list. 1 na czas naciśnięcia przycisku będzie zapalał diodę LED.

W przykładzie, w nieskończonej pętli, cały czas jest sprawdzany przy pomocy warunku *if* stan linii wejściowej, do której dołączono przycisk. Jeśli zostanie on naciśnięty, stan linii wejściowej zmieni się na niski, co spowoduje zapalenie diody LED. Puszczanie przycisku będzie równoważne z podaniem przez wewnętrzny rezystor podciągający dodatniego napięcia zasilania, czyli stanu wyso-



Rys. 2. Schemat ideowy układu dla przedstawionego przykładu

kiego, przy którym dioda LED będzie wyłączona.

Na koniec jeszcze słowo na temat tzw. *aliasów*, czyli alternatywnych nazw dla zmiennych, stałych, itd. Ponieważ ludzka pamięć jest zawodna i wygodniej nam zapamiętywać nazwy własne niż nazwy narzucone przez producentów układów, Mark Alberts zadbał także o wygodę programistów. *Alias*y umożliwiają nadanie standardowym nazwom portów, rejestrów, zmiennym itp. nazw przyjaznych użytkownikowi. Składnia instrukcji *aliasu* jest następująca:

```

Nazwa_przyjazna ALIAS Nazwa_trudna_do_zapamietania
    
```

Co w przykładzie przedstawionym na list. 1 mogłoby mieć postać:

```

Dioda Alias PortB.2 'przypisanie 'aliasu "dioda" do nazwy portb.2
    
```

Nazwie PortB.2 zostanie przypisana alternatywna nazwa *dioda*, która jednoznacznie identyfikuje linię portu. Na przykład instrukcja RESET DIODA spowoduje zapalenie diody dołączonej do linii B.2 mikrokontrolera. Podobny *alias* można stworzyć dla dołączonego do linii wejściowej przycisku. Na przykład:

```

Przycisk alias pind.5 'przypisanie aliasu "przycisk" 'do nazwy pind.5
    
```

W tym przypadku nazwie *przycisk* przypisano bit 5 rejestru *pind*, który jest rejestrem wejściowym portu D. Jak to wcześniej wyjaśniono, nie można do odczytu stanów linii wejściowych np. portu D wykorzystać rejestru *portd*.

Mamy nadzieję, że w tym krótkim artykule przedstawiliśmy w sposób jasny i wyczerpująco zagadnienia związane z konfiguracją linii portów mikrokontrolerów AVR, a także rozwialiśmy najczęściej zgłaszane wątpliwości.

Za miesiąc przedstawimy zagadnienia związane z obsługą sprzętowego interfejsu RS232, w który wyposażony jest niemal każdy mikrokontroler AVR.

Artur Starz,
artur.starz@ep.com.pl
Marcin Wiązania,
marcin.wiazania@ep.com.pl