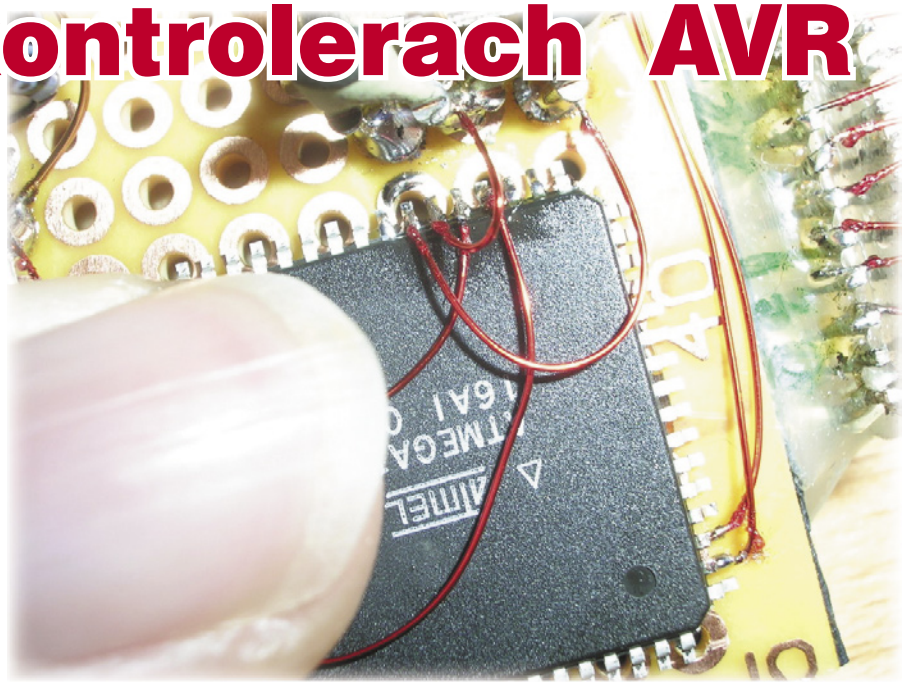
Bootloader w mikrokontrolerach AVR



Przywykliśmy do tego, że aby zaprogramować pamięć Flash mikrokontrolera potrzebny jest specjalny programator szeregowy lub równoległy. Niektóre mikrokontrolery potrafią jednak modyfikować swoją własną pamięć programu, co przy zastosowaniu specjalnego (wpisanego wcześniej do Flash'a) bootloadera umożliwi również programowanie układu bez dodatkowych narzędzi – przez port szeregowy.

Rekomendacje:

opisana metoda programowania mikrokontrolerów AVR będzie przydatna dla każdego praktykującego elektronika, w większości przypadków umożliwi programowanie układów bez dodatkowego wyposażenia sprzętowego.

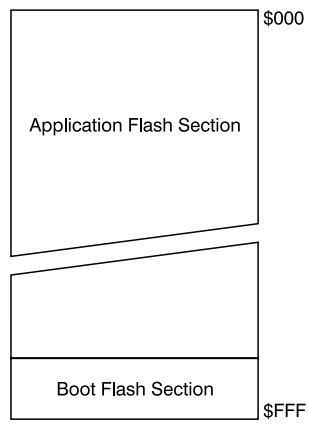
PODSTAWOWE PARAMETRY

- Możliwość programowania pamięci Flash i EEPROM mikrokontrolera przez port szeregowy lub USB (po zastosowaniu konwertera FT232R)
- Prędkość transmisji max. 115200 bodów
- Zdalne zerowanie mikrokontrolera linią DTR interfejsu RS232
- Wielkość obszaru pamięci bootloadera: 128, 256, 512 lub 1024 słowa
- Możliwość bezpośredniego korzystania z bootloadera w pakiecie Bascom AVR

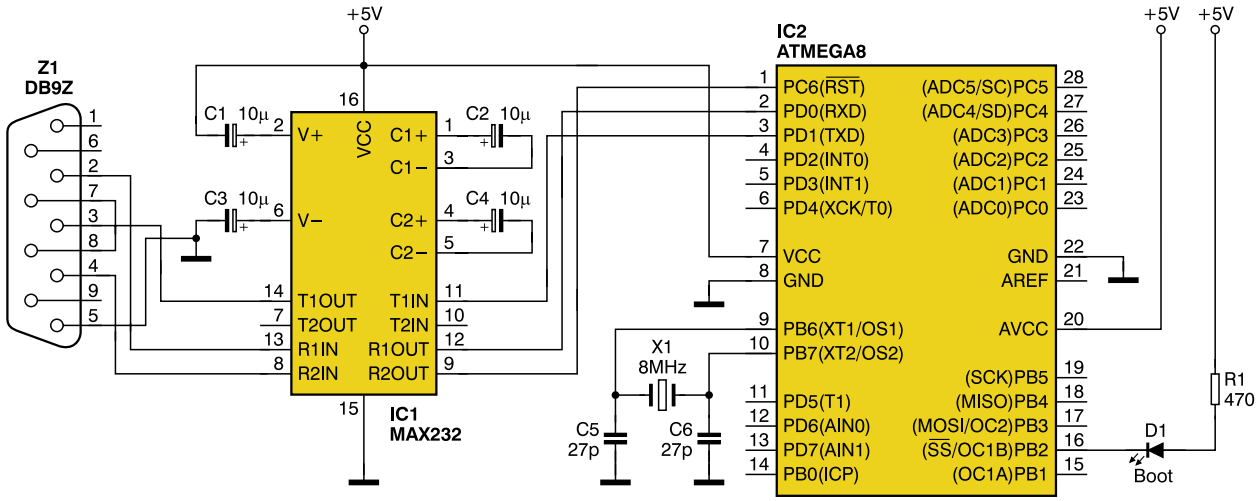
Wykorzystując specjalny program – *bootloader* – rezydujący w pamięci Flash mikrokontrolera można dokonywać zmiany jego programu sterującego poprzez interfejs RS232 lub bardziej popularny USB. Nadają się do tego oczywiście również i inne interfejsy. W artykule zostanie przedstawiony przykład uniwersalnego *bootloadera* napisanego w języku Bascom AVR, który można umieścić w większości mikrokontrolerów ATmega. Środowisko Bascom AVR posiada również wbudowaną opcję aktualizacji programu za pomocą *bootloadera*. Zapisanie do mikrokontrolera AVR programu *bootloadera* umożliwi nie tylko jego programowanie bez użycia zewnętrznego programatora, ale i nieskomplikowaną aktualizację oprogramowania. Choć bootloader ma zalety, to do jego wad można zaliczyć brak możliwości zmiany bezpieczników mikrokontrolera AVR (*Fuse Bits*). Producent mikrokontrolerów AVR nie udostępnia ich z zaprogramowanym *bootloaderem*, należy więc go we własnym zakresie jednokrotnie zapisać do wydzielonego obszaru pamięci Flash mikrokontrolera AVR. Po zapisaniu programu *bootloadera* do pamięci mikrokontrolera AVR, nie będzie już potrzebny zewnętrzny programator. Przykład wykorzystania programu *bootloadera* zostanie pokazany z wykorzystaniem mikrokontrolera ATmega8 i pakietu Bascom AVR.

Program *bootloadera*

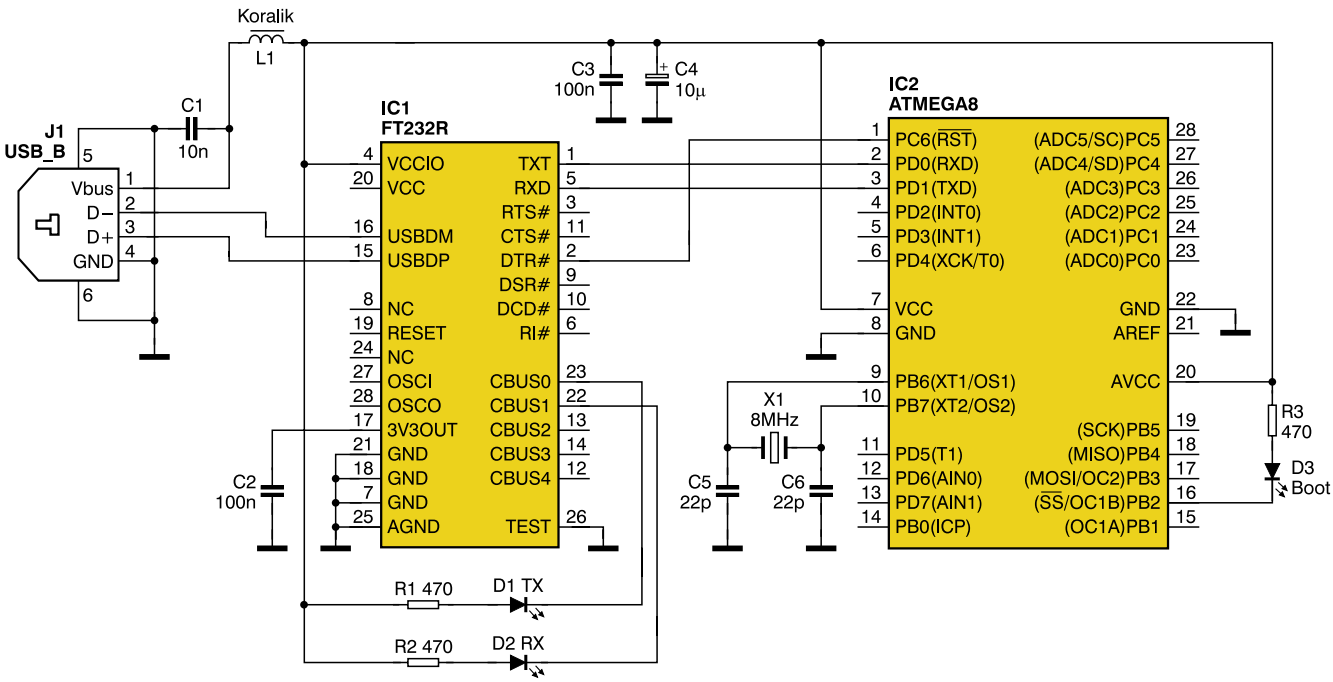
Na **rys. 1** pokazano obszar pamięci Flash mikrokontrolera ATmega8. Program *bootloadera* należy umieścić w obszarze *Boot Flash Section*. W przypadku mikrokontrolera ATmega8, w zależności od ustawień *Fuse Bits*, obszar na *bootloader* może posiadać pojemności 128, 256, 512 lub 1024 słowa. Program *bootloadera* został napisany przez firmę MCS Electronics. Przez niewielkie modyfikacje kodu programu, można go umieścić w różnych typach mikrokontrolerów AVR.



Rys. 1. Obszar pamięci Flash mikrokontrolera ATmega8



Rys. 2. Sposób połączenia mikrokontrolera do portu COM komputera podczas programowania z wykorzystaniem *bootloadera*

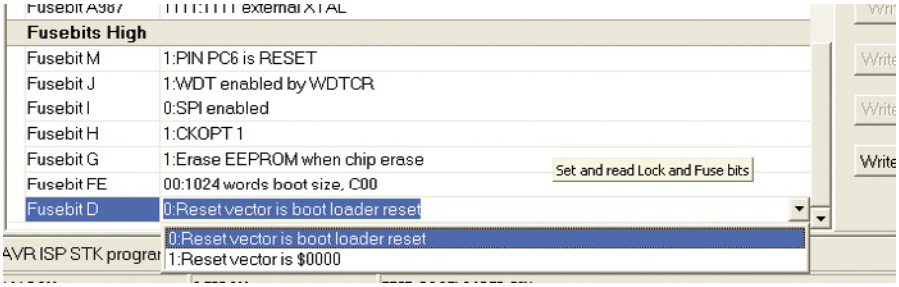


Rys. 3. Sposób połączenia mikrokontrolera do portu USB komputera podczas programowania z wykorzystaniem *bootloadera*

Umożliwia on modyfikowanie zarówno pamięci Flash, jak i pamięci EEPROM, która jest programowana za pomocą bascomowej instrukcji *Writeeeprom*. Program *bootloadera* został zamieszczony na **list. 1**. Maksymalna prędkość transmisji interfejsu szeregowego RS232 podczas współpracy z *bootloaderem* jest równa 115200 bodów. Prędkość tę można ustawić w programie *bootloadera*, ale będzie ona zależała od częstotliwości oscylatora mikrokontrolera. W celu uzyskania poprawnej transmisji przez interfejs RS232, mikrokontroler należy taktować z wykorzystaniem rezonatora kwarcowego, gdyż wewnętrzny oscylator RC nie posiada zbyt dużej stabilności. W programie *bootloadera* dla danego typu mikrokontrolera należy odblokować dwie instrukcje. W przypadku mikrokontrolera ATmega8 będą to:

```
$regfile = „m8def.dat”
Const Loaderchip = 8
```

Pierwsza z nich określa typ mikrokontrolera, a druga jest stałą, od której zależy liczba stron obliczanych w dalszej części programu. W przykładzie mikrokontroler jest taktowany częstotliwością 8 MHz, a prędkość transmisji wynosi 38400 bod. W ramach przykładu wykorzystania *bootloadera*, mikrokontroler został podłączony do komputerowego portu COM1 w sposób pokazany na **rys. 2**. Konwerter MAX 232 jest wymagany ze względu na rożne poziomy napięć w interfejsie RS232 komputera i mikrokontrolera. Dioda D1 sygnalizuje pracę *bootloadera*. *Bootloader* może być wywoływany na dwa sposoby: poprzez instrukcję skoku pod adres *bootloadera* lub poprzez zerowanie mikrokontrolera. Aby zautomatyzować zerowanie, oprogramowanie sterujące *bootloaderem* wykorzystuje do tego celu linię DTR portu COM. Bardzo ułatwia to wymianę oprogramowania mikrokontrolera. Linia DTR, po zmianie poziomu przez układ MAX 232, steruje wejściem zerowania mikrokontrolera (rys. 2). Zamiast in-



Rys. 4. Konfiguracja programu pozwalająca skorzystać z automatycznego uaktywniania *bootloadera* przy każdym włączeniu mikrokontrolera

Rys. 5. Okno programu *bootloader*

terfejsu RS232 można również zastosować konwerter USB–RS232, na przykład w postaci układu FT232R, jak to pokazano na **rys. 3**. Diody D1, D2 sygnalizują przepływ informacji przez port szeregowy, a dioda D3 sygnalizuje pracę *bootloadera*. Oczywiście dla układu FT232R należy zainstalować sterowniki wirtualnego portu COM. Po skoku do programu *bootloadera*, oczekuje on na otrzymanie wartości 123 lub 124. Po otrzymaniu wartości 123 będzie programowana pamięć Flash, a po otrzymaniu wartości 124 pamięć EEPROM. Pamięć Flash jest zapisywana z wykorzystaniem asemblerowej instrukcji SPM. Skonfigurowany i skompilowany program *bootloadera* należy umieścić w mikrokontrolerze za pomocą dowolnego programatora mikrokontrolerów AVR. Dodatkowo w przypadku wykorzystania rezonatora kwarcowego należy skonfigurować jego bezpieczniki (*Fuse Bits*).

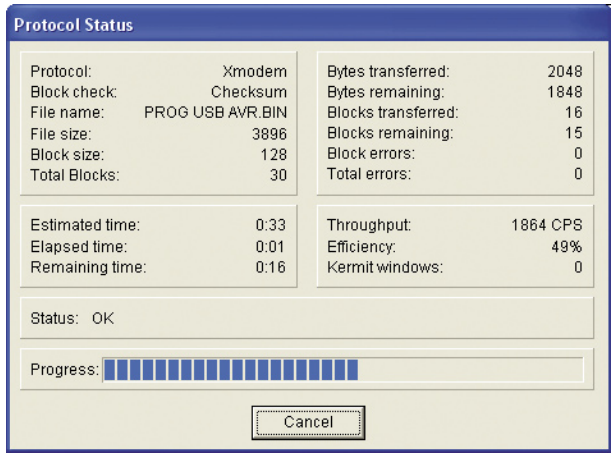
Sposoby uaktywniania *bootloadera*

Aby skorzystać z automatycznego uaktywniania *bootloadera* przy każdym włączeniu mikrokontrolera, należy ustawić *fusebit* BOOTRST (Fusebit D), jak to pokazano na **rys. 4**. Po ustawieniu tego bitu, najpierw jest wykonywany skok do programu *bootloadera*. Gdy nie zostanie przesłana wartość 123 lub 124 w określonym czasie, *bootloader* kończy działanie i zaczyna być wykonywany program główny mikrokontrolera. *Fusebity* BOOTSZ0 i BOOTSZ1 (Fusebit FE) umożliwiają wybór wielkości obszaru pamięci zajmowanego przez *bootloader*. W tym przypadku ustawiono maksymalny obszar 1024 słów. Z tego sposobu wywoływania korzysta program obsługujący *bootloader* (*Bootloader.exe*) oraz pakiet Bascom AVR. Mikrokontroler jest zerowany automatycznie za pomocą linii DTR. Gdy nie będzie wykorzystywane automatyczne zerowanie w celu uaktywnienia *bootloadera*, można je przeprowadzić ręcznie za pomocą przycisku dołączonego do wejścia zerującego mikrokontroler. Innym sposobem wywołania *bootloadera* jest wydanie rozkazu skoku pod adres, pod którym został zapisany *bootloader*. Taki skok może być zainicjowany przez dowolnie wybrane przez programistę zdarzenie (np. wciśnięcie przycisku). Przykład programowego wywołania *bootloadera* przedstawiono na **list. 2.**

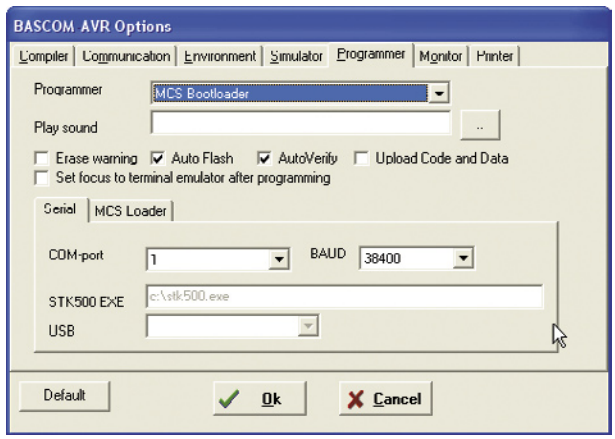
W programie tym skok do *bootloadera* nastąpi, gdy z portu COM zostanie odebrany znak o kodzie ASCII równym 27 (ESC). Za pomocą instrukcji *Goto*, nastąpi wówczas skok do początku programu *bootloadera*. Aktywacja *bootloadera* jest sygnalizowana miganiem diody LED dołączonej do portu PB.2 mikrokontrolera. Linia portu, do której została dołączona dioda LED może być skonfigurowana w programie *bootloadera*. Sposób aktywacji *bootloadera* w dużej mierze będzie zależał od sprzętowej, jak i programowej budowy projektowanego urządzenia. W przykładzie jest wykorzystywany pierwszy sposób aktywacji *bootloadera*, czyli za pomocą wejścia zerującego.

Programowanie mikrokontrolera z wykorzystaniem *bootloadera*

Mikrokontroler z zapisanym programem *bootloadera* można zaprogramować własnym programem za pomocą programu *Bootloader.exe* lub bezpośrednio w pakiecie Bascom AVR. Na **rys. 5** pokazano okno programu *bootloader*. Za jego pomocą można wysłać plik o rozszerzeniu .BIN do mikrokontrolera. W menu *Options* można wybrać numer portu COM oraz prędkość transmisji, która powinna być zgodna z prędkością ustawioną w *bootloaderze* mikrokontrolera. W przypadku programowania pamięci EEPROM mikrokontrolera,



Rys. 6. Okno wyświetlające informację o postępie programowania



Rys. 7. Konfiguracja środowiska Bascom AVR do programowania mikrokontrolerów z wykorzystaniem *bootloadera*

plik posiada rozszerzenie .EEP. Programowanie mikrokontrolera rozpoczyna się od naciśnięcia przycisku *Upload*. Wcześniej mikrokontroler jest zerowany za pomocą sygnału DTR. Gdy nie jest wykorzystywany sygnał DTR, należy ręcznie wyzerować mikrokontroler. Do mikrokontrolera wysłano prosty program testowy, który pokazano na **list. 3.**

Powoduje on miganie diody LED dołączonej do portu PB.4 mikrokontrolera. Podczas programowania mikrokontrolera poprzez program *bootloader.exe*, wyświetlane jest okno z informacją o postępie programowania (**rys. 6**). Programowanie mikrokontrolera z poziomu Bascom AVR polega na wybraniu w menu *Programmer*, typu programatora – *MCS bootloader*, zgodnie z **rys. 7**. W zakładce *Serial* można skonfigurować numer portu oraz prędkość transmisji, a w zakładce *MCS Loader* wielkość *bootloadera* oraz sposób jego aktywacji (poprzez linię DTR). Skompilowany program można załadować do mikrokontrolera naciskając klawisz *F4*. Podczas programowania mikrokontrolera zostaje wyświetlone okno pokazane na **rys. 8**. Można również przygotować własną aplikację obsługującą *bootloader* dostosowaną do własnych potrzeb. Po zastosowaniu konwertera USB–RS232 programowanie mikrokontrolera przy użyciu *bootloadera* przebiega identycznie.

List. 2. Przykład programowego wywołania bootloadera

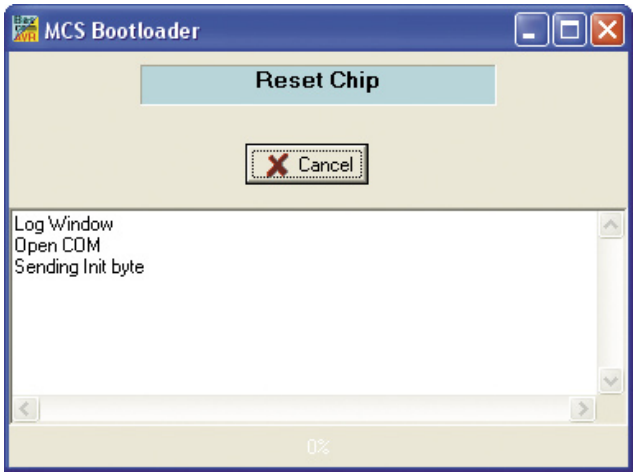
```
Do          'pętla do-loop
   Print „test"      'wysłanie przykładowego tekstu
   Waitms 1000    'opóźnienie 1000 ms
   If Inkey() = 27 Then 'jeśli odebrano znak 27, to
      Print „boot"      'wysłanie tekstu boot
      Goto &HC00     'skok do bootloadera
   End If
Loop          'koniec pętli do-loop
```

List. 3. Prosty program testowy

```
$crystal = 8000000                          ,częstotliwość oscylatora
$regfile = „m8def.dat"                      ,typ mikrokontrolera

Config Pinb.4 = Output

Do
  Toggle Portb.4
  Waitms 250
Loop
End
```



Rys. 8. Okno wyświetlane podczas programowania mikrokontrolera

Podsumowanie

Wykorzystując mikrokontroler z załadowanym *bootloaderem*, w nieskomplikowany sposób uzyskujemy możliwość aktualizacji oprogramowania, co ma duże znaczenie w przypadku użytkowników końcowych. Mogą oni sami je wymienić, bez wysyłania zakupionego urządzenia do serwisu. W przypadku pakietu Bascom AVR, wgrany wcześniej do mikrokontrolera *bootloader* można wykorzystywać zamiast dodatkowego programatora. Wystarczy nacisnąć klawisz *F4*, a skompilowane oprogramowanie zostanie przesłane do mikrokontrolera. We własnych aplikacjach sposób aktywacji *bootloadera* w dużej mierze będzie zależeć od budowanego urządzenia. Wykorzystując uniwersalny *bootloader* firmy MCS Electronics komunikacja przebiega za pomocą interfejsu RS232, Stosując konwerter USB–RS232 można uzyskać układ programowany za pomocą interfejsu USB. Oczywiście załadowanie *bootloadera* będzie możliwe tylko w mikrokontrolerach AVR, które posiadając wydzielony dla niego obszar pamięci. Większość mikrokontrolerów ATmega posiada tę możliwość.

Marcin Wiązania, EP
marcin.wiazania@ep.com.pl