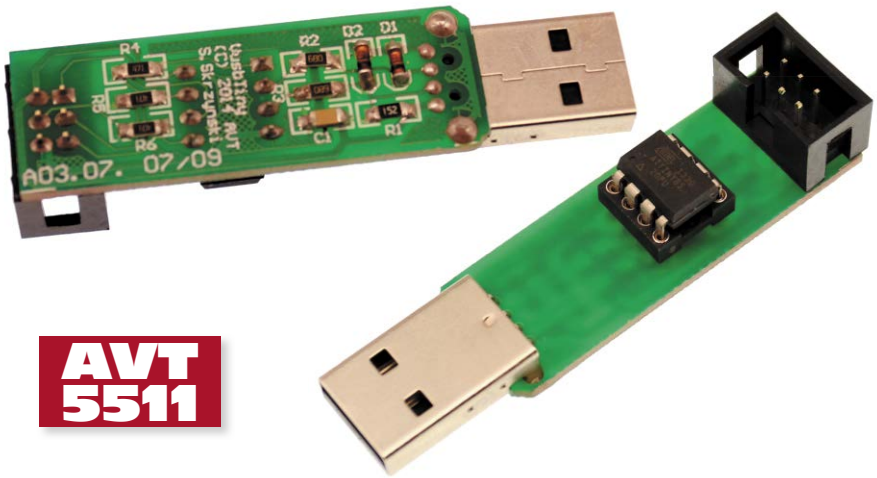
VUSBtiny

Miniaturowy programator mikrokontrolerów AVR

W Elektronice Praktycznej nr 12/2011 opisano tani programator mikrokontrolerów AVR – USB ASP. W tym numerze prezentujemy jeszcze mniejszy (wielkość porównywalna z pendrive) i jeszcze tańszy programatora, który ze względu na swoje wymiary i niewielki koszt wykonania przyda się w warsztacie niejednego elektronika lub do celów serwisowych.

***Rekomendacje:** programator polecamy każdemu, kto zajmuje się programowanie mikrokontrolerów AVR lub myśli o pierwszych aplikacjach wykonanych z ich użyciem.*



Schemat ideowy programatora pokazano na **rysunku 1**. Widać na nim niewiele komponentów – mikrokontroler ATtiny45, dwie diody Zenera, kondensator i kilka rezystorów. Oczywiście, są jeszcze złącza – wtyk USB oraz wtyk IDC-6. Rezystor R1 informuje host o obecności urządzenia USB transmitującego dane w trybie *Slow*. Rezystory R2 i R3 w połączeniu z D1 i D2 zapewniają maksymalne napięcie na magistrali USB 3,6 V, a nie 5 V, ponieważ napięcie +5 V na magistrali powoduje występowanie błędów SYNC. Rezystory R4…R6 zabezpieczają programowany układ przed uszkodzeniem w wypadku, gdy jest on zasilany napięciem niższym niż 5 V.

Zwora JP1 powinna być otwarta. Zakłada się ją tylko w sytuacji, gdy chcemy zasilić uruchamiany układ z USB. Trzeba oczywiście pamiętać o ograniczeniach wynikających z maksymalnego dopuszczalnego prądu obciążenia oraz zakresu napięcia dostarczanego przez USB.

Montaż i uruchomienie

Schemat montażowy programatora zamieszczono na **rysunku 2**. Montaż jest typowy i nie wymaga omawiania. Aby zmniejszyć wysokość programatora nie należy używać podstawki pod mikrokontroler. Ponadto, trzeba zastosować kątowe złącze SPI (J2) i zrezygnować z zworki JP2. Cały programator można umieścić w koszulce termokurczliwej, ale przedtem należy zaprogramować mikrokontroler. Mikrokontrolery dostarczane w zestawach AVT są już zaprogramowane, jeśli jednak programator budujemy samodzielnie, trzeba to zrobić własnoręcznie.

Ustawienie konfiguracyjnych mikrokontrolera:
Low: 0xE1
High: 0x5D
Ext: 0xFF

Mikrokontroler należy programować w trybie wysokonapięciowym, ponieważ linia zerowania pracuje jak typowe I/O. Taką opcję mają programatory tzw. równoległe. Można jednak postąpić inaczej. Większość programów nie pozwala na wyłączenie linii RST w trybie SPI, ale np. AvrDude daje taką możliwość. Korzystając z nakładki Burn-O-Mat można przełączyć linię RST w tryb I/O, tyle, że należy włączyć tryb *Expert*.

W takiej sytuacji w trybie SPI należy zaprogramować pamięć Flash (plik: vusbtiny.hex dostępny w materiałach dodatkowych), a następnie odpowiednio ustawić bity konfiguracyjne. Oczywiście po tym stracimy możliwość programowania w trybie SPI. Jeśli więc zostanie popełniony jakiś błąd, to trzeba użyć nowego mikrokontrolera lub użyć programatora w trybie HVPROG. Dlatego zdecydowałem się na mikrokontroler w obudowie DIP, który można zamontować w podstawce, co ułatwia jego wymianę. Nawet, gdy zdecydujemy się na wlutowanie mikrokontrolera, to łatwo go wylutować ze względu na użycie płytki jednowarstwowej.

Przed przyłączeniem VUSBtiny do komputera należy pobrać sterowniki dostępne w materiałach dodatkowych lub pod adresem *https://goo.gl/cPUKLC*. Po dołączeniu programatora do interfejsu USB komputera pojawi się kreator. Instalacja nie przebiegnie automatycznie, należy wskazać sterowniki. Gdy instalacja sterowników przebiegła poprawnie, co można sprawdzić w Menadżerze Urządzeń, uruchamiamy *avrdude* z konsoli. W konsoli przechodzimy do katalogu

W ofercie AVT*

AVT-5511 A	AVT-5511 B
AVT-5511 C	AVT-5511 UK

Podstawowe informacje:

- Miniaturowe wymiary – wielkość pendrive.
- Bazuje na mikrokontrolerze ATtiny45.
- Obsługiwany prze popularne programy np. avrdude, Bascom AVR itp.
- Obsługiwane mikrokontrolery: ATmega128, ATmega1280, ATmega1281, ATmega16, ATmega162, ATmega164, ATmega168, ATmega169, ATmega2560, ATmega2561, ATmega32, ATmega324, ATmega328, ATmega329, ATmega3290, ATmega48, ATmega64, ATmega640, ATmega644, ATmega649, ATmega6490, ATmega8, ATmega8515, ATmega8535, ATmega88, ATtiny12, ATtiny13, ATtiny15, ATtiny2313, ATtiny25, ATtiny26, ATtiny45, ATtiny85.
- Na podstawie programatora opisanego na stronie *http://www.simpleavr.com/avr/vusbtiny*.

Dodatkowe materiały na FTP:

ftp://ep.com.pl, user: 87550, pass: rxoaagj8

- wzory płytek PCB

Projekty pokrewne na FTP:
(wymienione artykuły są w całości dostępne na FTP)

AVT-5388	Programator AVR-ISP MKII (EP 3/2013)
AVT-1683	Przystawka do programowania mikrokontrolerów AVR firmy Atmel (EP 7/2012
AVT-5325	UsbAsp – Programator mikrokontrolerów AVR (EP 11/2011)
AVT-5322	AVR JTAG-ICE – interfejs debugera dla mikrokontrolerów AVR (EP 11/2011)

* Uwaga:
Zestawy AVT mogą występować w następujących wersjach:

AVT xxxx UK	to zaprogramowany układ. Tylko i wyłącznie. Bez elementów dodatkowych.
AVT xxxx A	płytka drukowana PCB (lub płytki drukowane, jeśli w opisie wyraźnie zaznaczono), bez elementów dodatkowych.
AVT xxxx A+	płytka drukowana i zaprogramowany układ (czyli połączenie wersji A i wersji UK) bez elementów dodatkowych.
AVT xxxx B	płytka drukowana (lub płytki) oraz komplet elementów wymienionych w załączniku pdf
AVT xxxx C	to nic innego jak zmontowany zestaw B, czyli elementy wlutowane w PCB. Należy mieć na uwadze, że o ile nie zaznaczono wyraźnie w opisie, zestaw ten nie ma obudowy ani elementów dodatkowych, które nie zostały wymienione w załączniku pdf
AVT xxxx CD	oprogramowanie (nieczęsto spotykana wersja, lecz jeśli występuje, to niezbędne oprogramowanie można ściągnąć, klikając w link umieszczony w opisie kitu)

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! (UK, A, A+, B lub C). *http://sklep.avt.pl*

Wykaz elementów

Rezystory: (SMD 1206)
R1: 1,5 kΩ
R4…R6: 470 Ω
R2, R3: 68 Ω
Kondensatory:
C1: 100 nF (SMD 1206)
Półprzewodniki:
D1, D2: dioda Zenera SMD na 3,6 V
U1: ATtiny45/85 (zaprogramowany)
Inne:
J1: USB-A (wtyk USB do druku)
J2: IDC6MLP (gniazdo 6 pin)
JP1: goldpin 1×2+zworka

Tabela 1. Obsługiwane mikrokontrolery i ich kody:

Typ mikrokontrolera	Kod w poleceniu
ATmega128	m128
ATmega1280	m1280
ATmega1281	m1281
ATmega16	m16
ATmega162	m162
ATmega164	m164
ATmega168	m168
ATmega169	m169
ATmega2560	m2560
ATmega2561	m2561
ATmega32	m32
ATmega324	m324
ATmega328	m328
ATmega329	m329
ATmega3290	m3290
ATmega48	m48
ATmega64	m64
ATmega640	m640
ATmega644	m644
ATmega649	m649
ATmega6490	m6490
ATmega8	m8
ATmega8515	m8515
ATmega8535	m8535
ATmega88	m88
ATtiny12	t12
ATtiny13	t13
ATtiny15	t15
ATtiny2313	t2313
ATtiny25	t25
ATtiny26	t26
ATtiny45	t45
ATtiny85	t85

z programem. Dla wygody program można zainstalować w głównym katalogu jakiegoś dysku, załóżmy, że jest to *d:\avrdude*. W takiej sytuacji musimy wydać dwie komendy:

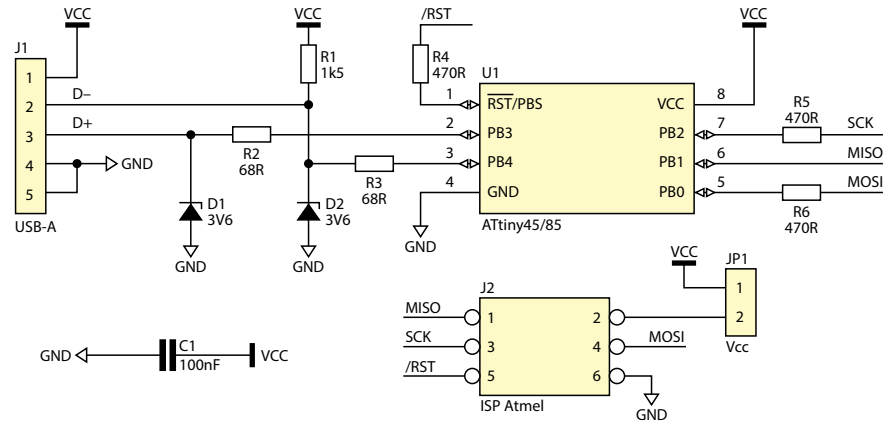
```
d:
cd\avrdude
```

Wykonujemy test. Ważne, aby programator był połączony z programowanym mikrokontrolerem. Wydajemy komendę:

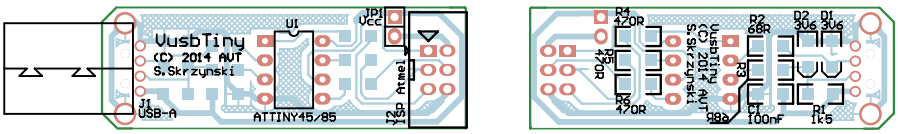
```
avrdude -c usbtiny -p m8
```

Jeśli połączenia były poprawne zobaczymy:

```
avrdude: AVR device initialized
and ready to accept instructions
```



Rysunek 1. Schemat ideowy programatora VUSBtiny



Rysunek 2. Schemat montażowy programatora VUSBtiny

```
Reading | ####################
############################## |
100% 0.00s
avrdude: Device signature =
0x1e910a
avrdude: safemode: Fuses OK
avrdude done.  Thank you.
```

Jeśli będą problemy z komunikacją, to zostanie wyświetlony następujący komunikat:

```
avrdude: error: programm enable:
target doesn't answer. 1
avrdude: AVR device initialized
and ready to accept instructions
Reading | ####################
############################## |
100% 0.02s
avrdude: Device signature =
0x000000
avrdude: Yikes!  Invalid device
signature.
          Double check connec-
tions and try again, or use -F
to override
          this check.
```

Za pomocą komendy avrduce –c usbtiny –p m8 odczytuje się sygnaturę układu, jej składnia to avrdude -c {programator} -p {typ procesora}. Nie ma pomyłki w rozkazie, jeśli chodzi o typ programatora. Trzeba wybrać „usbtiny”, a nie „vusbtiny”. Vusbtiny zachowuje się jak usbtiny, a „V” to nowsza wersję procedur obsługi USB. Najważniejsza zmiana to możliwość wykorzystania wewnętrznego oscylatora RC. Co do typu procesora. Przy odczycie sygnatury nie ma to znaczenia, ale podczas programowaniu już tak. Jeśli chodzi o typ procesora to, gdy jest to na przykład ATmega8, wpisujemy „m8”. Dla ATtiny2313 wpisujemy „t2313”. Pełna listę kodów umieszczono w **tabeli 1**.

Jeśli nie było komunikatu o błędzie możemy zapisać pamięć mikrokontrolera. W tym celu wydajemy komendę:

```
avrdude -c usbtiny -p m8
-U flash:w:program.hex
```

Odczyt to komenda:

```
avrdude -c usbtiny -pm8
-U flash:r:program.hex
```

Ustawienie bitów konfiguracyjnych:

```
avrdude -c usbtiny -p m8
-U hfuse:w:0x{wartość}:m
-U lfuse:w:0x{wartość}:m
```

Zapis EEPROM:

```
avrdude -c usbtiny -p m8
-U eeprom:w:program.hex
```

Odczyt:

```
avrdude -c usbtiny -pm8
-U eeprom:r:program.hex
```

Obsługa programu z linii komend jest kłopotliwa, dlatego powstały nakładki graficzne. Najlepszą wydaje mi się Burn-O-Mat'a. Można ją pobrać spod adresu *http://goo.gl/3OOmQt*. Po zainstalowaniu i uruchomieniu program należy zainstalować. Ważne jest ustawienie ścieżki dostępu do avrdude oraz jego pliku konfiguracyjnego wywołanego z menu Setings/AVRDUDE (**rysunek 3**). Teraz, po zaakceptowaniu okna konfiguracyjnego przyciskiem *OK* należy zamknąć, po czym ponownie otworzyć program, aby mogły się wczytać ustawienia i typy programatorów z pliku konfiguracyjnego. W kolejnym kroku, ponownie w oknie Setings/AVRDUDE, wybieramy rodzaj programatora (**rysunek 4**). Listy „port” nie należy zmieniać na „USB” – należy zostawić domyślne „/dev/parport0”. Zmiany akceptujemy za pomocą *OK*. Od teraz, po wybraniu w liście *AVR type* (jak na **rysunku 5**) typu mikrokontrolera można zapisywać i odczytywać zawartość pamięci Flash i EEPROM. Aby zmienić

Więcej informacji jest dostępne w Elektronice Praktycznej nr 12/2011. Można tam znaleźć informacje (zrzuty ekranowe) na temat sposobu instalowania sterowników oraz konfigurowania programatora do współpracy z AvrStudio i Bascom AVR.

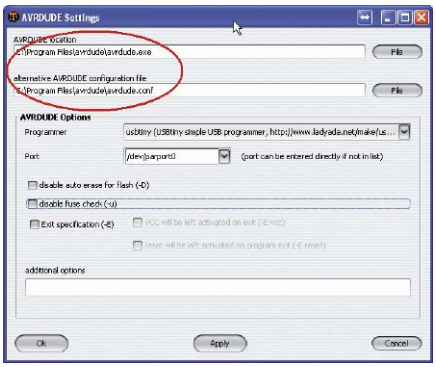
Konfiguracja AVR Studio do współpracy z USBtiny:
- Z menu „Tools” wybieramy „Customize...”.
- W nowo otwartym oknie „Command” wybieramy zakładkę „Tools”.
- Wskazujemy ikonkę „New” lub wciskamy klawisz „Insert”.
- Wpisujemy nazwę programatora, np. „USBtiny”.
- W oknie „Command” wskazujemy ścieżkę do „avrdude” (np.: C:\WinAVR-20100110\bin\avrdude.exe).
- W oknie „arguments” podajemy parametry: „-p m168 -c usbtinyp -P usb -U flash:w:”plik.hex”:a -U flash:v:”plik.hex”:a”, gdzie „m168” to typ procesora.
- W „Initial directory” wpisujemy ścieżkę dostępu do pliku (trzeba pamiętać o końcowym znaku „\”)
- Zmiany zatwierdzamy przyciskiem „Close”.

W celu zaprogramowania mikrokontrolera wybieramy w menu „Tools” nazwę naszego programatora (w przykładzie jest to „USBtiny”.

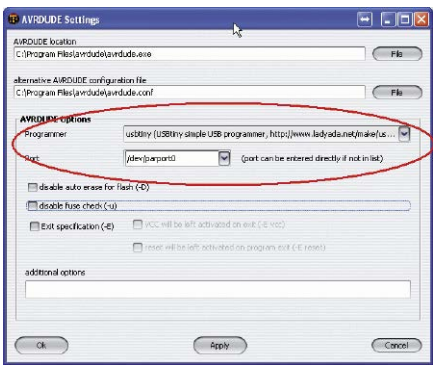
Konfiguracja Bascom do współpracy z USBtiny:
- Z menu wybieramy „Options/Programmer”.
- Z listy rozwijanej wybieramy „External programmer”.
- W zakładce „Other” (w dolnej części okna) podajemy ścieżkę do „avrdude”.
- W oknie parametry wpisujemy „„avrdude” -p m168 -c usbtiny -U flash:w:”{FILE}”:a -U flash:v:”{FILE}”:a” oraz zaznaczamy opcje „Use HEX file”, gdzie „m168” to typ procesora.

Programowanie wywołujemy klawiszem F4 lub ikonką „Program chip”.

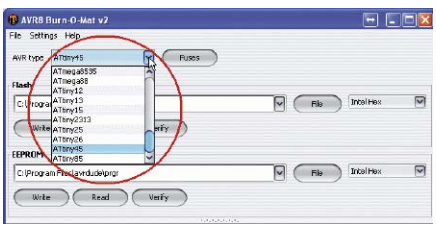
ustawienie bitów konfiguracyjnych, należy nacisnąć przycisk *Fuses* umieszczony na prawo od listy *AVR type*. Zostanie wyświetlone okno, jak na **rysunku 6**. Mamy możliwość odczytu bitów konfiguracyjnych (*read fuses*), zapisu (*write fuses*) oraz weryfikacji (*verify fuses*). Można też przywrócić ustawienia domyślne (*reset to default*). Niższe zakładki umożliwiają różne sposoby ustawiania bitów. Liczba zakładek jest zależna od typu mikrokontrolera i ich szczegółowe omówienie zajęłoby połowę EP, więc zachęcam do własnych eksperymentów. Eksperymenty



Rysunek 3. Menu konfiguracyjne avrdude



Rysunek 4. Wybór rodzaju programatora



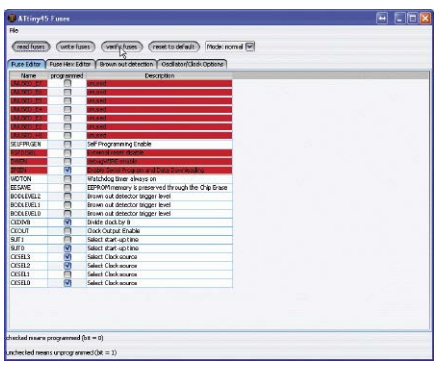
Rysunek 5. Wybór typu mikrokontrolera

z bitami są w miarę bezpieczne, ponieważ nie można niektórych zmienić (tych zaznaczonych na czerwono). W normalnym trybie pracy nie da się wyłączyć programowania przez SPI, czy też przełączyć wyprowadzenia reset w tryb I/O, co uniemożliwiłoby programowanie mikrokontrolera w trybie SPI. Wyłączona jest nawet możliwość przejścia w tryb DebugWire, który obsługiwany jest przez nieliczne programatory (np. Dragona). Można oczywiście zmienić te bity w trybie *Expert*, co opisano wcześniej w artykule. Jakkolwiek Burn-O-Mat dba o „bezpieczeństwo mikrokontrolera” to trzeba być ostrożnym przy zmianach źródła sygnału zegarowego. Na ten temat pisano już w EP, a także jak „uratować” mikrokontroler, bez uciekania się do użycia programatora w trybie HVPROG, co jest kłopotliwe zwłaszcza, gdy jest on wlutowany w płytkę. Należy pamiętać, że zaprogramowany bit to **wartości 0!** Czyli zaznaczenie „ptaszkiem” bitu nadaje mu wartość 0, co jest opisane w nocie katalogowej mikrokontrolera.

Tematu programu Burn-O-Mat z pewnością nie wyczerpałem, ale sądzę, że przekazane informacje wystarczą do używania programu. Sposób instalacji programatora pod Linux-em jest opisany pod adresem *https://goo.gl/IEy0Js*.

Sławomir Skrzyński, EP

Bibliografia:
http://goo.gl/fY4jyp
https://goo.gl/qo59aj



Rysunek 6. Konfigurowanie bitów bezpieczników