Programator USBasp

instrukcja obsługi

Tomasz Ostrowski
Aktualizacja: 2007.03.21

Spis treści

1. Informacje ogólne

Autorem projektu USBasp jest Thomas Fischl. Kod źródłowy programatora wykorzystuje dostępną na zasadach GPL lub komercyjnych bibliotekę software'owego interfejsu USB firmy Objective Development. Licencja projektu znajduje się na stronie wyżej wymienionej firmy.

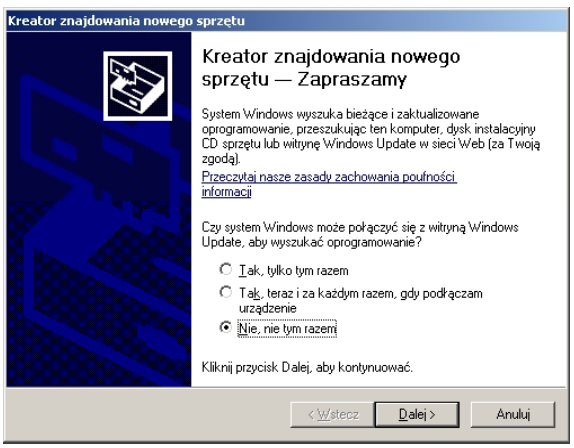
2. Instalacja.

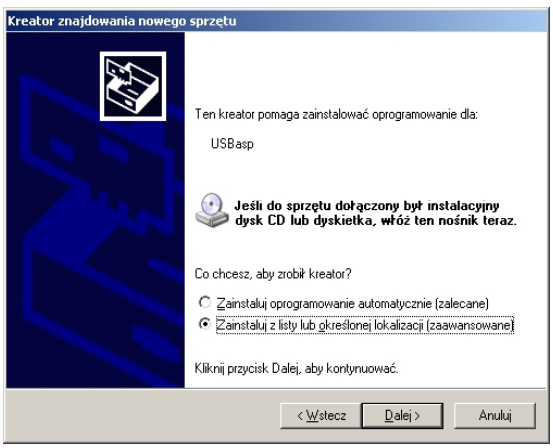
Do podłączenia programatora z PC wykorzystywany jest przewód USB A-B (często spotykany przy drukarkach). Zalecane jest wykorzystanie przewodu o długości 1.8m lub mniejszej i nie korzystanie z gniazd na przednim panelu komputera połączonych z płytą główną dodatkowym przewodem.

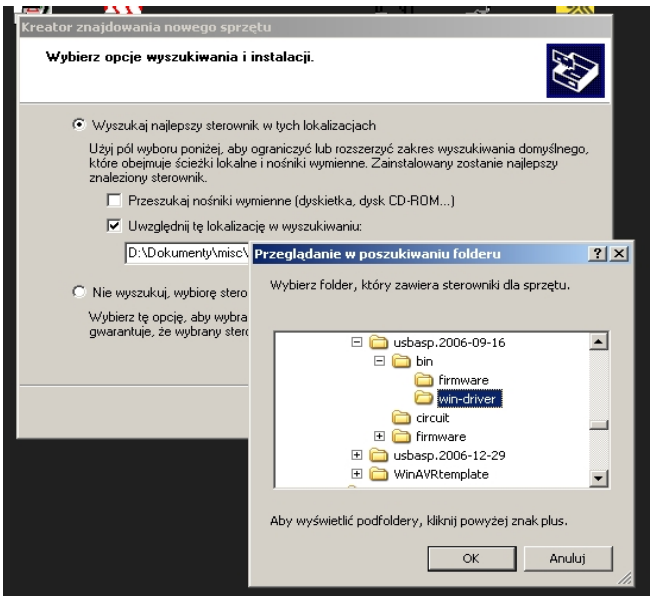
Połączenie z układem programowanym zapewnia taśma zakończona z obu stron wtykami IDC-10. Przy podłączaniu programatora do układu programowanego istotny jest sposób dołączenia wtyków: skrajny przewód taśmy o wyróżniającym się kolorze powinien zostać dołączony z obu stron do pinu nr 1 gniazda. Strona po której znajduje się pin 1 gniazda wyróżniona jest czarną kropką oraz cyfrą 1 po stronie druku.

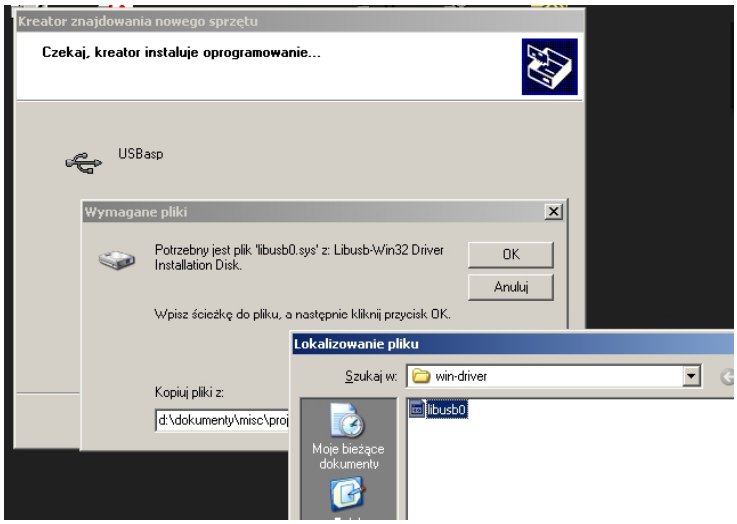
Rozkład pinów złącza goldpinowego na płytce programatora odpowiada standardowi STK-200. Możliwe jest wykonanie dodatkowego przewodu o innym połączeniu pinów lub innym standardzie gniazda ISP który potraktować można jako przejściówkę.

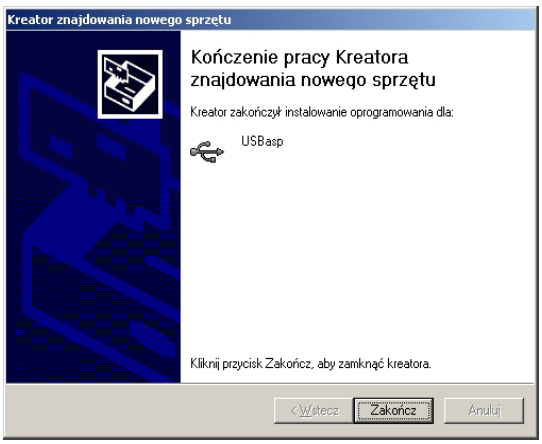
Przy pierwszym podłączeniu programatora do komputera wykryty zostanie on jako nowe urządzenie. Konieczne jest wskazanie ścieżki do sterownika (katalog `win-driver`).











Po zainstalowaniu sterownika programator powinien być widoczny w menadżerze urządzeń jako

urządzenie o nazwie USBasp. Podłączenie go do innego portu USB niż poprzednio wywoła ponownie okno instalacji sterownika (jak zresztą przy każdym innym urządzeniu USB).

Aplikacją współpracującą ze starszą wersją firmware programatora jest patchowany przez Thomasa Fischla program `avrdude`. Nowsza wersja (*usbasp.2006-12-29.tar.gz*) obsługiwana jest przez „zwykły" avrdude w wersji **5.3.1**. Aby korzystać z tego programu należy w systemach operacyjnych Win2000/XP zainstalować sterownik `giveio.sys` (wystarczy uruchomić plik `install_giveio.bat` z katalogu `WinAVR/bin`). Najprostszym sposobem korzystania z niego jest wykorzystanie środowiska WinAVR wraz z Programmer's Notepadem. Instalacja polega na skopiowaniu plików z katalogu `bin` do katalogu `WinAVR/bin` i nadpisaniu plików poprzednio tam istniejących.

! Nowa wersja avrdude (5.3.1) jest niekompatybilna ze starszym firmware programatora i odwrotnie, użyj wersji zgodnych ze sobą. Niezgodność wersji objawia się komunikatem:
`avrdude: error: could not find USB device vendor=0x3eb product=0xc7b4`
lub podobnym, pomimo tego, że urządzenie jest widoczne w menedżerze urządzeń i zainstalowany został sterownik.

3. Użytkowanie.

Przeznaczenie zworek programatora:

- zworka 1 – aktualizacja firmware programatora,
- zworka 2 – aktualizacja firmware programatora,
- zworka 3 – zmniejszenie częstotliwości pracy ISP, przydatne gdy programowany mikrokontroler taktowany jest z częstotliwością poniżej 1MHz.

Aby korzystać z programatora USBasp z poziomu Programmers Notepada lub podobnego IDE należy w pliku `makefile` projektu zadeklarować użycie tego programatora:

- znajdź i zakomentuj (dodaj znak # na początku) wiersz rozpoczynający się od
  `AVRDUDE_PROGRAMMER`,
- dopisz wiersz:
  `AVRDUDE_PROGRAMMER = usbasp`
- w zależności od potrzeby dodaj uzupełnij cykl programowania o programowanie pamięci eeprom i bajtów fusebitów/lockbitów:
  `AVRDUDE_WRITE_EEPROM = -U eeprom:w:`**`eeprom.hex`**
  `AVRDUDE_WRITE_LOCK = -U lock:w:`**`0x3C`**`:m`
  `AVRDUDE_WRITE_HFUSE = -U hfuse:w:`**`0xC9`**`:m`
  `AVRDUDE_WRITE_LFUSE = -U lfuse:w:`**`0x9F`**`:m`
  Jeżeli któraś z tych opcji nie jest potrzebna, wskazane jest zakomentowanie związanego z nią wiersza.

Przykładowy zestaw fusebitów dla mikrokontrolera Atmega8 pracującego z zewnętrznym kwarcem 12MHz:

```
# Fuse high byte:
```

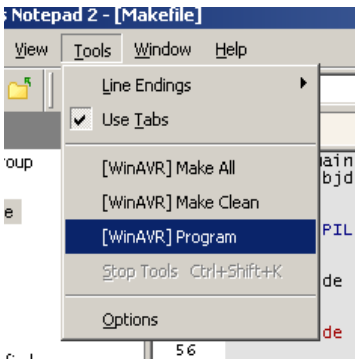
```
# 0xc9 = 1 1 0 0   1 0 0 1 <-- BOOTRST (boot reset vector at 0x0000)
#        ^ ^ ^ ^   ^ ^ ^------ BOOTSZ0
#        | | | |   | +-------- BOOTSZ1
#        | | | |   + ---------- EESAVE (don't preserve EEPROM over chip erase)
#        | | | +-------------- CKOPT (full output swing)
#        | | +---------------- SPIEN (allow serial programming)
#        | +------------------ WDTON (WDT not always on)
#        +-------------------- RSTDISBL (reset pin is enabled)
# Fuse low byte:
# 0x9f = 1 0 0 1   1 1 1 1
#        ^ ^ \ /   \--+--/
#        | |  |       +------- CKSEL 3..0 (external >8M crystal)
#        | |  +--------------- SUT 1..0 (crystal osc, BOD enabled)
#        | +------------------ BODEN (BrownOut Detector enabled)
#        +-------------------- BODLEVEL (2.7V)
```

w pliku makefile powinna znaleźć się linia:

```
program: $(TARGET).hex $(TARGET).eep
(lub np. program: $(TARGET).hex eeprom.eep)
	$(AVRDUDE) $(AVRDUDE_FLAGS) $(AVRDUDE_WRITE_FLASH) $(AVRDUDE_WRITE_EEPROM)
$(AVRDUDE_WRITE_HFUSE) $(AVRDUDE_WRITE_LFUSE) $(AVRDUDE_WRITE_LOCK)
```

! nieostrożność przy ustalaniu wartości fusebitów może zakończyć się zablokowaniem mikrokontrolera, niemożliwym do usunięcia bez wysokonapięciowego programatora równoległego; konieczne jest dokładne zapoznanie się z kartą katalogową

Po przeprowadzeniu powyższych operacji możliwe jest programowanie mikrokontrolera poprzez opcję Tools/Program Programmer's Notepada.



Alternatywne sposoby korzystania z `avrdude` jest użycie linii komend lub nakładki `avrdude-gui` nie wspierającej jednak w pełni możliwości tego programu.

UWAGA! W przypadku gdy wystąpi błąd programowania, poprzedzony odczytem przez avrdude zerowej sygnatury (*avrdude: Device signature = 0x000000*) winą może być niskie taktowanie lub duże opóźnienie startowe mikrokontrolera programowanego – należy założyć w programatorze **zworkę nr 3** aby zmniejszyć szybkość SPI przy programowaniu. Dotyczyć to może m.in. fabrycznie nowych mikrokontrolerów (dla Atmega8 ustawione fusebity SUT dają duże opóźnienie startowe a źródłem taktowania jest generator 1MHz).