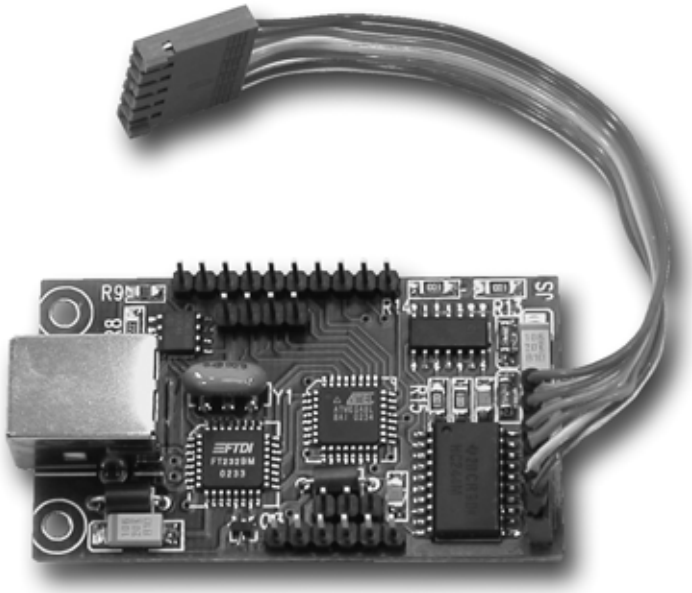
Programator USB mikrokontrolerów ATmega - ISP, część 1

AVT-524



Programatory ISP dla mikrokontrolerów AVR to temat od dawna znany i obfitujący w rozmaite rozwiązania - zarówno komercyjne jak i amatorskie (jedno z możliwych rozwiązań przedstawiamy w Miniprojektach). Pojawia się więc pytanie, czy warto od podstaw opracowywać coś nowego? Na pewno nie ma sensu powielanie typowego projektu opartego np. na porcie równoległym LPT. Co w zamian? Oczywiście USB! ***Rekomendacje****: programator dla wymagających konstruktorów, przede wszystkim tych, którzy korzystają z mikrokontrolerów ATmega.*

Podczas konstruowania programatora prezentowanego w artykule nie koncentrowałem się na zapewnieniu jego maksymalnej prostoty i obniżeniu ceny, ważniejsze były jego cechy użytkowe. Dlatego też:

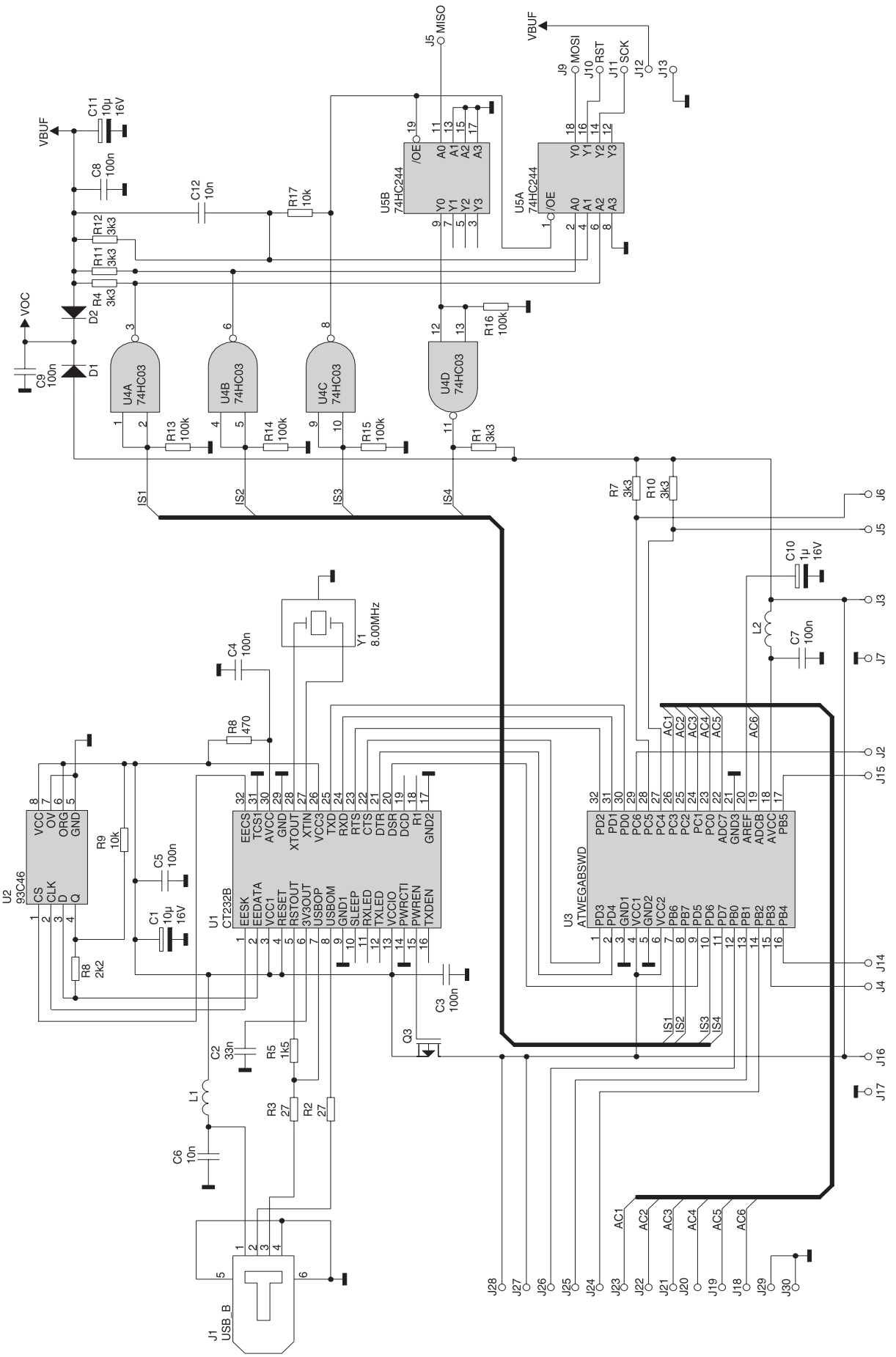
- komunikacja z komputerem sterującym jest zrealizowana za pomocą łącza USB, co często jest jedynym dostępnym sposobem dla posiadaczy laptopów,
- własne oprogramowanie sterujące pozwala na dowolne modyfikacje, poprawki i rozszerzenia a także na stosunkowo łatwą integrację z rozmaitymi środowiskami,
- płytka programatora posiada oprócz buforowanego interfejsu ISP wyprowadzone także inne interfejsy (I2C, SPI) oraz linie przetwornika A/C, PWM itd. co pozwala na jej wykorzystanie na rozmaite sposoby,
- mikrokontroler programatora można wyposażyć w bootloader, co pozwoli na szybkie i sprawne wymiany wersji oprogramowania za pośrednictwem tego samego łącza USB.

Konstrukcja programatora

Schemat elektryczny urządzenia przedstawiono na **rys. 1**. Można na nim wyróżnić trzy podstawowe bloki funkcyjne: interfejs USB, mikrokontroler sterujący oraz buforowany interfejs ISP.

Połączenie z magistralą USB jest zrealizowane z pomocą układu U1 - FT8U232BM - w typowej aplikacji proponowanej przez FTDI. Pamięć U2 pozwala na wprowadzenie własnych identyfikatorów programatora. Tranzystor Q3 zapewnia - zgodnie z wymaganym standardem USB - wyłączanie zasilania pro-

Programator prezentowany w artykule współpracuje ze środowiskiem projektowym dla AVR-GCC - AVRSide (dostępne bezpłatnie na stronie *http://www.avrside.fr.pl* oraz na CD-EP8/2003B). Możliwości jego wczesnej wersji opisaliśmy w EP1/2003).



Rys. 1. Schemat elektryczny programatora

gramatora podczas enumeracji oraz w stanie uśpienia komputera - hosta.

Mikrokontroler U3 steruje interfejsem ISP zgodnie z komendami i danymi otrzymywanymi z działającego na komputerze - hoście oprogramowania. Zwraca też informacje o prawidłowym zakończeniu kolejnych operacji lub o zaistniałych błędach. Dzięki zastosowaniu mikrokontrolera ATmega 8 nie ma konieczności dodawania praktycznie żadnych dodatkowych peryferiów. Mikrokontroler pracuje z wewnętrznym oscylatorem 8 MHz i ma wbudowany układ zerujący oraz watchdog. Jednocześnie dysponuje całą gamą dodatkowych sprzętowych interfejsów: I2C, SPI, wielokanałowym przetwornikiem analogowo-cyfrowym, wyjściami PWM. Odpowiednie styki wyprowadzono na płytce, co pozwoli na ewentualne wykorzystanie programatora do różnych innych celów (konwertery USB<->I2C, USB<->SPI, programatory pamięci szeregowych, przetwornik A/C do komputera itd.). Obwód filtrujący L2, C7 zasila obwody A/C, natomiast kondensator C10 dodatkowo filtruje wewnętrzne napięcie odniesienia 2,56 V, które zostało przewidziane do wykorzystania w układzie.

Wyprowadzenia interfejsu SPI tworzą zarazem wejście magistrali ISP pozwalającej na szeregowe wpisanie do mikrokontrolera programu wykonawczego oraz ustawień konfiguracyjnych.

Interfejs ISP łączy linie I/O mikrokontrolera sterującego z wejściami ISP docelowego układu. Musi on spełnić następujące wymagania:
- całkowicie uniezależnić zespół urządzeń od kolejności włączania zasilania,
- umożliwić dołączanie układu docelowego zasilanego innym napięciem,
- uaktywniać linie sterujące ISP tylko i wyłącznie w trakcie programowania - normalnie powinny pozostawać w stanie wysokiej impedancji nie wpływając w żaden sposób na działanie docelowego układu bez konieczności odłączania.

Zadania te zostały zrealizowane przy pomocy inwertera z wyjściem OC (U4) oraz trójstanowego bufora HC244 (U5). Bufor U5 oraz *pull-up y* jego linii wejściowych są zasilane napięciem docelowego systemu - zawsze więc uzyskamy potrzebną zgodność poziomów logicznych. Inwerter U4 jest zasilany dwustronnie poprzez diody Schottky ego D1 i D2. Kostka pozostaje więc „pod napięciem" niezależnie od chwilowego podłączenia zasilania - eliminuje to możliwość wysterowania wejść przy braku zasilania U4 (co układy cyfrowe niezbyt lubią). Wyjścia typu otwarty kolektor pozwalają na odpowiednie dopasowanie poziomów logicznych (dla poziomu niskiego 0 V, a dla poziomu wysokiego Vcc - niezależnie od jego wartości - oczywiście w ramach możliwości układu, czyli do 6 V). Nie ma to większego znaczenia przy zasilaniu obu układów jednakowym napięciem (czyli +5 V - tyle programator otrzymuje z magistrali USB). Natomiast przy np. 3,3 V systemu docelowego mamy następującą sytuację:
- zasilanie U5 = 3,3 V,
- wysokie poziomy logiczne na wejściach i wyjściach U5 = ok. 3,3 V,
- zasilanie U4 = ok. 4,7...4,8 V (ok. +5 V z USB pomniejszone o spadek na diodzie Schottky ego),
- wysokie poziomy logiczne na wejściach U4 od strony programatora - ok. 5 V (co jest dopuszczalne - nie przekraczamy napięcia zadziałania wejściowych wewnętrznych diod ochronnych kostki),
- wysoki poziom logiczny na wejściu U4 od strony bufora (wyprowadzenia 12, 13) = ok. 3,3 V. Jest to jedyna linia wymagająca dokładniejszego sprawdzenia: z danych serii HC wynika, że przy takiej wartości napięcia zasilającego minimalne napięcie rozpoznawane jako poziom wysoki wynosi nieco mniej niż 3,3 V, mieścimy się więc w wymaganym zakresie.

Wynika z tego, że układy z zasilaniem poniżej 3,3 V mogą sprawiać problemy. Jak nisko można zejść - pozostaje do praktycznego sprawdzenia (zazwyczaj rzeczywiste właściwości układów są lepsze niż gwarantowane wartości podawane w katalogach).

Zwróćmy też uwagę na sposób włączania linii RST interfejsu. Sekwencja przebiega następująco:
- ustawiamy programowo niski poziom SCK - bufor jest jeszcze w stanie wysokiej impedancji i na wyjściu nic się nie dzieje,
- ustawiamy programowo niski poziom RST - bufor 244 zostaje włączony podając na wyjście SCK poziom niski, ale linia RST zostaje przestawiona z poziomu wysokiego na niski dopiero po krótkiej chwili wynikającej ze stałej czasowej obwodu R17, C12. W ten sposób gwarantujemy stabilny stan niski SCK w chwili zakończenia zerowania, zgodnie z zaleceniami Atmela dotyczącymi wchodzenia w tryb programowania szeregowego.

Montaż układu

Całość układu została zmontowana na dwuwarstwowej płytce drukowanej, której schemat montażowy pokazano na **rys. 2**. Użyto głównie elementów SMD, co pozwoliło na zachowanie niewielkich wymiarów. Wyprowadzenia interfejsów możemy wyposażyć w listwy *goldpin* albo pozostawić do podłączeń przewodami - w zależności od potrzeb. Do polutowania najlepiej użyć grota *mini wave*, ale ponieważ rastry kostek nie należą do najmniejszych - tradycyjne techniki też będą zupełnie wystarczające.

Po zlutowaniu sprawdźmy dokładnie, czy nie ma zwarć ani przerw. Wskazane jest też przemycie płytki jednym z dostępnych preparatów chemicznych (np. PCC). Podłączenie do ISP prototypowego docelowego układu wykonamy według własnych potrzeb niezbyt długim (najwyżej kilkanaście centymetrów) przewodem taśmowym.

Uruchamianie programatora

Sam sprzętowy układ bez skonfigurowania i zaprogramowania nie przyda się nam do niczego. „Ożywianie" płytki przebiega w kilku etapach. Wszystkie przedstawione opisy i programy dotyczą systemu Windows. Adaptację układu do innych systemów operacyjnych pozostawiam inwencji Czytelników.

Niezbędne programy (łącznie ze środowiskiem AVRSide) oraz pliki pomocnicze publikujemy na płycie CD-EP8/2003B.

Etap 1 - konfiguracja układu interfejsu USB

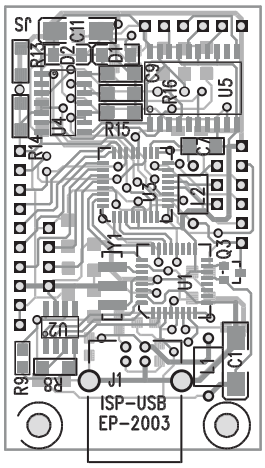
Najpierw musimy wyposażyć się w narzędzia ze strony producenta (*www.ftdichip.com*) - będzie nam potrzebny sterownik tzw. bezpośredni (*direct driver*) D2xx (najnowsza wersja z obsługą układów w wersji BM) oraz program *Ftd2xxst.exe* do obsługi szeregowej pamięci EEPROM 93C46. Rozpakowane pliki umieszczamy na dysku i zapamiętujemy lokalizację.

Instalacja sterowników

Teraz możemy po raz pierwszy podłączyć nasz układ do magistrali USB (bezpośrednio do gniazda A komputera albo poprzez huba z własnym zasilaniem, nie ograniczającego poboru prądu do 100 mA).

Po włączeniu Windows wyświetli informację o obecności nowego urządzenia USB i poprosi o podanie lokalizacji sterownika - odnajdujemy i zatwierdzamy lokalizację folderu z plikami (jak wyżej) - po chwili instalacja jest zakończona. Menedżer urządzeń w kluczu *Kontrolery uniwersalnej magistrali szeregowej* pokaże teraz obecność urządzenia „FTDI FT8U2XX DEVICE".

Uwaga! Sprawa może się poważnie skomplikować jeśli używamy już jakiegoś układu z kostką FTDI obsługiwaną jako wirtualny port szeregowy. System ma wtedy dla firmowych VID oraz PID zainstalowany inny zestaw sterowników (VCP), co wyklucza jednoczesne użycie D2xx. Jedynym wyjściem będzie wtedy skonfigurowanie kostki FT8U232 na innym komputerze i zmiana np. PID. Taki sam zmieniony PID musimy ręcznie wpisać do pliku **.inf* sterownika D2xx, aby system umiał przypisać płytce z nowym PID właśnie ten sterownik. Operacje ze zmianą PID (VID) muszą być przeprowadzone z należytą starannością - jeśli się pomylimy system w ogóle nie będzie w stanie obsłużyć płytki (zgłaszając tylko obecność nieznanego urządzenia).



Rys. 2. Rozmieszczenie elementów na płytce drukowanej

Wpisy do deskryptorów FT8U232

Służy do tego wspomniany powyżej program narzędziowy *Ftd2xxst.exe*. Sposób jego użycia jest dokładnie opisany w podręczniku, który w postaci elektronicznej publikujemy na CD-EP8/2003B.

Po wybraniu nowego pliku edytujemy wszystkie potrzebne pola:
Manufacturer = FTDI
Manufacturer ID = FT
Vendor ID = 0403
Product ID = 6001 (chyba, że wymagana jest zmiana zgodnie z uwagą powyżej)
Description = avr isp loader (tej pozycji używa program nadrzędny PC do lokalizacji urządzenia, nie może być więc ona zmieniana, chyba że równolegle z kodem programu).

W opcjach zaawansowanych ustawiamy:
Plug and Play - nie
Fixed Serial Number - dowolnie, w prototypach ustawiłem na „tak", wprowadzając ręcznie numery 20000001, 20000002 itd.)
Self Powered - nie
Remote Wakeup - nie
Max Power (mA) - 250 mA (wprawdzie sam programator mieści się w podstawowych 100 mA, ale być może będziemy stosować płytkę w innym celu i przyda się możliwość zasilenia dodatkowych zewnętrznych podzespołów).

WYKAZ ELEMENTÓW

Rezystory
R1, R4, R7, R10...R12: 3,3kΩ 1206
R2, R3: 27Ω 0805
R5: 1,5kΩ 0805
R6: 470Ω 1206
R8: 2,2kΩ 1206
R9: 10kΩ 0805
R13...16: 100kΩ 1206

Kondensatory
C1, C11: 10µF/16V 6032 tantal
C2: 33nF 0805
C3...C5, C13: 100nF 1206
C6, C12: 10nF 0805
C7...C9: 100nF 1206
C10: 1µF/16V 3216 tantal

Półprzewodniki
U1: FT8U232BM TQFP32
U2: 93C46 SO8
U3: Atmega 8 TQFP32
U4: 74HC03 SO14
U5: 74HC244 SO20
Q3: tranzystor P-MOS MMBF 2202 PT1 SOT23
D1, D2: dioda uniwersalna *Schottky* Minimelf

Różne
Y1: rezonator ceramiczny 6,00 MHz CSTCC6.00MG - TC (Murata) SMD (można też powierzchniowo przylutować rezonator przewlekany)
L1, L2: koralik ferrytowy przewlekany
J1: gniazdo USB typ B
listwy *goldpin*, przewód taśmowy

Po zakończeniu edycji zapamiętujemy plik i programujemy EEPROM. Nowe ustawienia deskryptorów będą użyte po następnej enumeracji (czyli odłączeniu i ponownym podłączeniu płytki).

Etap 2 - konfiguracja mikrokontrolera ATmega 8

Używamy do tego dowolnego posiadanego programatora ISP. Prototypy były konfigurowane z poziomu AVRSide za pośrednictwem optoizolowanego programatorka RS. Zwróćmy tylko uwagę na prawidłowe podłączenie linii - za pomocą schematu i rysunku płytki zlokalizujemy bez trudu wyprowadzenia interfejsu ISP. Mikrokontroler zastosowany w projekcie pracuje z wewnętrznym oscylatorem 8 MHz. Prze-

stawiamy więc fabryczne 0001 (1 MHz) na 0100. Pozostałych bitów możemy w naszym zastosowaniu nie zmieniać. Jednocześnie należy odczytać i zanotować wartość bajtu kalibracyjnego dla 8 MHz. ATmega po zerowaniu zawsze automatycznie ładuje do korekcyjnego rejestru OSCCAL wartość dla 1 MHz. Musimy więc w części inicjalizacyjnej programu przeładować samodzielnie OSCCAL odczytaną powyżej wartością. Wymaga to wprawdzie przekompilowania programu oddzielnie dla każdego egzemplarza urządzenia, co jednak na etapie testów i uruchamiania (czy też przy składaniu pojedynczych płytek na własne potrzeby) nie jest żadnym utrudnieniem.

Uwaga! Konfiguracja *fuseíów* wymaga uwagi i staranności. Sprawdźmy dwa razy zanim coś ostatecznie wpiszemy do mikrokontrolera. Niektóre bity mogą nam bowiem całkiem zablokować interfejs ISP, ale nawet zmiana źródła taktowania (inne niż potrzebne ustawienie CSEL) sprawi w zmontowanym już urządzeniu mnóstwo kłopotów.

Etap 3 - zaprogramowanie mikrokontrolera

Jak zaznaczono wyżej, należy wpisać do programu właściwą dla danego egzemplarza ATmega wartość OSCCAL (na początku funkcji *main()* w module *u_main.c*). Kod dla programatora jest wolnym oprogramowaniem, całkowicie dostępnym - można więc przy okazji go poprawiać czy rozszerzać o nowe funkcje (na razie uruchomione są tylko podstawowe operacje dla serii ATmega). Otrzymany po kompilacji plik wynikowy *avrisp.hex* ładujemy do mikrokontrolera dowolnym programatorem ISP. Teraz możemy przystąpić do praktycznych prób ale o tym za miesiąc.

Jerzy Szczesiul, AVT
jerzy.szczesiul@ep.com.pl

Wzory płytek drukowanych w formacie PDF są dostępne w Internecie pod adresem: ***http://www.ep.com.pl/?pdf/sierpien03.htm*** *oraz na płycie CD-EP8/2003B w katalogu* ***PCB***.