TOMASZ FRANCUZ

JĘZYK C

DLA MIKROKONTROLERÓW AVR

OD PODSTAW DO ZAAWANSOWANYCH APLIKACJI

Przedstawiamy przebojowy duet — język C i mikrokontroler AVR!

Poznaj budowę i podstawy programowania mikrokontrolerów

Dowiedz się, jak do swoich celów wykorzystać język C

Naucz się rozwiązywać rzeczywiste problemy i tworzyć praktyczne rozwiązania

Helion

Język C dla mikrokontrolerów AVR. Od podstaw do zaawansowanych aplikacji

Autor: Tomasz Francuz
ISBN: 978-83-246-3064-6
Format: 158×235, stron: 568

Przedstawiamy przebojowy duet – język C i mikrokontroler AVR!

- Poznaj budowę i podstawy programowania mikrokontrolerów
- Dowiedz się, jak do swoich celów wykorzystać język C
- Naucz się rozwiązywać rzeczywiste problemy i tworzyć praktyczne rozwiązania

Mikrokontrolery AVR firmy Atmel stanowią dynamicznie rozwijającą się rodzinę układów. Dzięki niskiej cenie, dużym możliwościom i dostępności darmowych narzędzi od lat niezmiennie cieszą się dużą popularnością wśród hobbystów i osób profesjonalnie zajmujących się programowaniem mikrokontrolerów.

Pewnym utrudnieniem dla polskich użytkowników AVR jest brak literatury na temat wykorzystania do ich programowania języków wysokiego poziomu, takich jak C. Niniejsza książka jest próbą wypełnienia tej luki. W sposób syntetyczny pokazuje różnice pomiędzy programowaniem w języku C komputerów klasy PC i mikrokontrolerów. Omawia programowanie peryferii dostępnych w mikrokontrolerach AVR w języku C, bibliotekę standardową oraz jej rozszerzenia znane jako AVR-libc. Dzięki temu nawet osoby w niewielkim stopniu znające podstawy języka C będą mogły bez problemów „przesiąść się" na programowanie mikrokontrolerów AVR. Z drugiej strony książka opisuje zaawansowane techniki programowania, związane z obsługą bootloadera, zabezpieczaniem i szyfrowaniem kodu aplikacji oraz realizacją najpowszechniej stosowanych protokołów wymiany danych pomiędzy urządzeniami opartymi na mikrokontrolerach i komputerami PC. Porusza także tematy związane ze specyfiką pisania aplikacji na mikrokontrolery oraz wyszukiwaniem i usuwaniem błędów.

Podstawy programowania mikrokontrolerów AVR

- Warsztat pracy programisty AVR
- Wprowadzenie do języka C na AVR
- Budowa programu i jego części składowe
- Korzystanie z zasobów sprzętowych mikrokontrolera
- Używanie rejestrów i różnych rodzajów pamięci
- Zastosowania przetwornika ADC
- Obsługa wyświetlaczy LCD
- Korzystanie z interfejsów
- Zapewnianie bezpieczeństwa kodu

Programowanie mikrokontrolerów jeszcze nigdy nie było tak proste!

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 32 230 98 63
e-mail: helion@helion.pl

Spis treści

Rozdział 2.

Programowanie mikrokontrolera

Po wygenerowaniu plików wynikowych należy ich zawartość umieścić w pamięci mikrokontrolera. Dzięki temu po restarcie procesor będzie mógł rozpocząć wykonywanie programu. Procesory AVR dysponują możliwością programowania „w układzie" przy pomocy interfejsu ISP, część może być programowana poprzez interfejsy JTAG, debugWire, PDI, TPI, a w przypadku procesorów posiadających interfejs USB można także programować procesor poprzez wbudowany *bootloader*. W tym ostatnim przypadku nie da się jednak zmieniać konfiguracji *fusebitów*. Każda z metod programowania ma swoje zalety i wady.

Podłączenie — uwagi ogólne

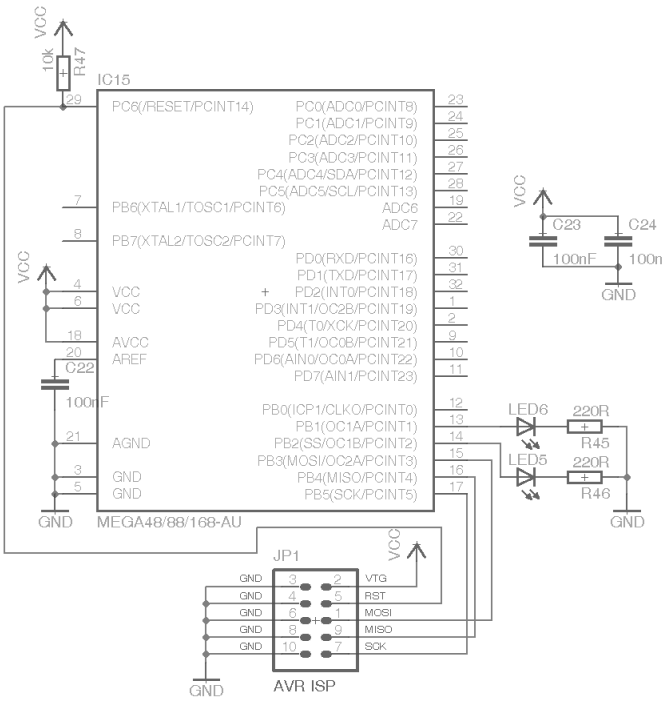
Każdy programator łączy się z układem docelowym przy pomocy dedykowanych wyprowadzeń. Dla programatorów szeregowych jest to zwykle 4 – 5 wyprowadzeń, dla równoległych znacznie więcej. Wykorzystanie wyprowadzeń mikrokontrolera do programowania ogranicza możliwość ich wykorzystania do innych celów. Najlepiej, jeśli takie wyprowadzenia nie będą wykorzystywane do niczego innego — w układzie będą one podłączone wyłącznie do gniazda łączącego z programatorem. Jednak w układach posiadających niewielką liczbę wyprowadzeń nie zawsze jest to możliwe. Stąd też powinniśmy pamiętać, aby podłączone do tych wyprowadzeń urządzenia nie obciążały ich zbytnio (w trybie programowania będą one obciążały wyjście programatora). Z tego powodu nie zaleca się podłączać do nich np. diod LED, nie należy na tych liniach dodawać także kondensatorów, szczególnie o większych pojemnościach (>1 nF). Dodatkowo jeśli jakieś wyprowadzenie jest wejściem (czyli wyjściem układu programującego), nie należy łączyć do niego innych wyjść — w takiej sytuacji w trakcie programowania powstanie konflikt pomiędzy wyjściem programatora a wyjściem układu korzystającego z tego pinu.

Pamiętajmy, że dla programatora nie ma znaczenia, jak w programie skonfigurowane są piny wykorzystywane do programowania.

Programator na czas programowania wprowadza procesor w stan *RESET*, co wiąże się z wprowadzeniem wszystkich pinów *IO* w stan wysokiej impedancji.

Szczególną uwagę należy zwrócić na podłączenie sygnału *RESET*. Aby wejść w tryb programowania, programator musi mieć możliwość wymuszenia na tej linii stanu niskiego (lub +12 V w przypadku programatorów wysokonapięciowych). W efekcie problem może wystąpić, jeśli w układzie używane są zewnętrzne układy generujące *RESET* lub monitorujące zasilanie. Przykładowy schemat podłączenia procesora do programatora ISP pokazano na rysunku 2.1.

Rysunek 2.1. *Przykład podłączenia programatora ISP do mikrokontrolera ATMega88*



W dalszych rozdziałach książki pokazane schematy, dla uproszczenia i większej przejrzystości, nie będą już zawierały elementów pokazanych na schemacie z rysunku 2.1, czyli gniazda ISP, kondensatorów odsprzęgających oraz połączeń z zasilaniem (Vcc i GND).

Aby pokazane w dalszej części układy działały poprawnie, należy zapewnić poprawne połączenie elementów pokazanych na rysunku 2.1. Szczególnie istotne jest podłączenie w każdym procesorze wszystkich występujących w nim wyprowadzeń zasilania (Vcc i GND). Przy braku podłączenia niektórych wyprowadzeń układ może działać

niestabilnie i stwarzać problemy. Drugim istotnym elementem są kondensatory odsprzęgające C23 i C24. Są to elementy, których zadaniem jest odsprzęganie zasilania, a ich znaczenie rośnie wraz ze wzrostem stopnia skomplikowania układu.

Problemy

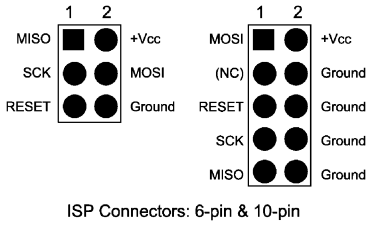
Najczęstsze problemy z zaprogramowaniem procesora:

1. W przypadku programowania w trybie ISP „zablokowanie" procesora, poprzez niewłaściwą konfigurację *fusebitów*.
2. Nieprawidłowa częstotliwość sygnału *SCK* (w przypadku programatorów ISP). Jeśli podejrzewamy taki problem, należy zmniejszyć szybkość programowania. W żadnym przypadku nie może ona przekroczyć ¼ częstotliwości taktowania procesora.
3. Zbyt długi kabel łączący programator z układem. Im dłuższy kabel, tym większe ryzyko niepoprawnej pracy układu. Zwykle problem ten objawia się niestabilną pracą programatora.
4. Błędne podłączenie sygnałów. Zawsze warto się upewnić, że wszystkie sygnały zostały prawidłowo połączone z odpowiednimi wyprowadzeniami procesora.
5. Pomyłkowe podłączenie programatora nie do wyprowadzeń związanych z ISP, lecz do wyprowadzeń związanych z interfejsem SPI (oznaczenia linii sygnałowych są podobne). Problem ten dotyczy głównie procesorów ATMega128.
6. Wybór niewłaściwego programatora lub niewłaściwego trybu programowania.
7. Zablokowanie wykorzystywanego interfejsu (dotyczy głównie próby programowania przy wyłączonym interfejsie JTAG lub próby programowania przy pomocy ISP, z włączonym interfejsem debugWire).

Programatory ISP

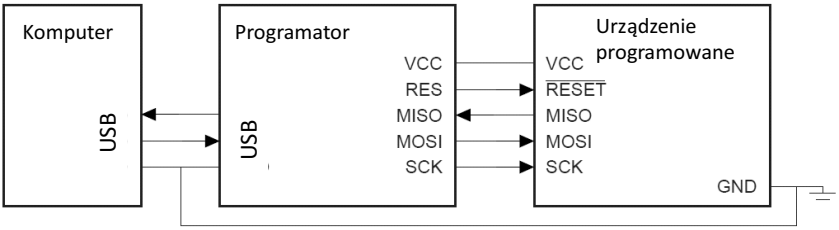
Prawie każdy procesor AVR dysponuje możliwością programowania przy pomocy interfejsu ISP (ang. *In-system Programming Interface*). Interfejs ten wykorzystuje do programowania piny *RESET*, *MISO*, *MOSI* oraz *SCK*. Zwykle piny te pokrywają się z analogicznymi wyprowadzeniami interfejsu SPI, lecz nie zawsze tak jest. Jednym z takich wyjątków jest procesor ATMega128. Stąd też zawsze należy dokładnie sprawdzić, jakie wyprowadzenia procesora wykorzystywane są do programowania przy pomocy interfejsu ISP. Informacje o wykorzystanych wyprowadzeniach znajdują się w nocie katalogowej procesora, w sekcji *Memory Programming/Serial Downloading*. Programatory ISP mają znormalizowany układ sygnałów wyprowadzonych na złącze programujące, pokazany na rysunku 2.2.

Istnieją dwa typy złącza ISP — jedno mniejsze, 6-pinowe, oraz większe 10-pinowe. Odstęp pomiędzy pinami wynosi 2,54 mm, chociaż w nowszych konstrukcjach Atmela



Rysunek 2.2. *Rozkład sygnałów standardowych programatorów. Kwadratem oznaczono wyprowadzenie o numerze 1*

spotyka się rozstaw pinów 1,27 mm. Umożliwia to zmniejszenie rozmiarów złącza programującego na płytce. Pewnego omówienia wymaga przeznaczenie pinu oznaczonego jako *+Vcc*. Do tego pinu należy podłączyć napięcie zasilające układ. Napięcie z tego pinu wykorzystywane jest przez programator do zasilania buforów wyjściowych, dzięki czemu programator dostosowuje poziomy napięć na pozostałych pinach programatora do napięć panujących w programowanym układzie. Część programatorów posiada także specjalną zworkę przełączającą napięcie. W takiej sytuacji jedna z pozycji powoduje zasilenie programatora z programowanego układu, w drugiej pozycji to programator zasila programowany układ. Programator łączy się z programowanym układem tak, jak pokazano na rysunku 2.3.



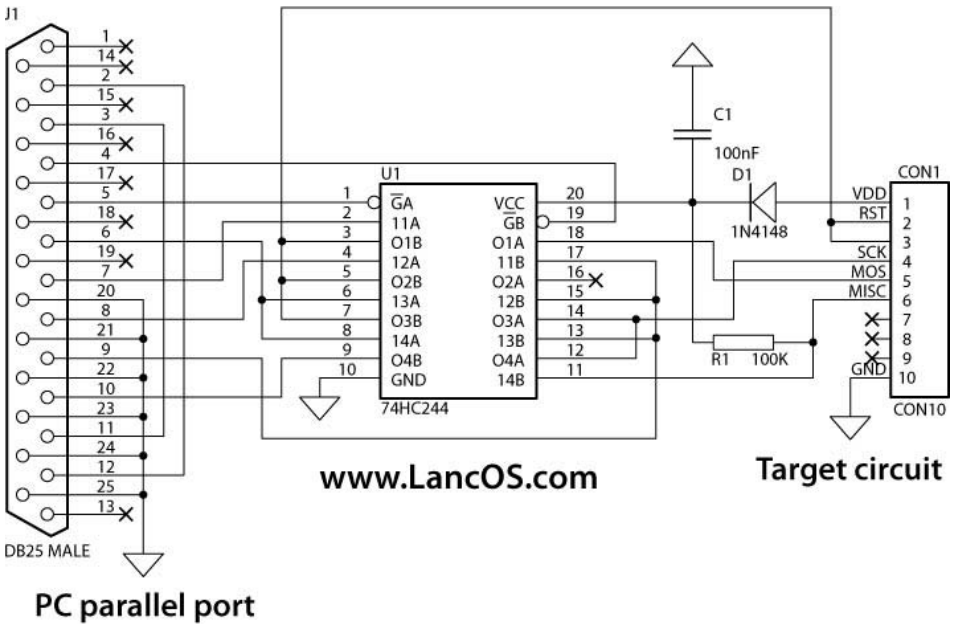
Rysunek 2.3. *Połączenie programowanego układu z programatorem i komputerem PC. Masy wszystkich urządzeń muszą zostać połączone razem*

Łącząc programator z komputerem i programowanym układem, należy zwracać uwagę na potencjał masy. W komputerach klasy PC, ze względu na budowę zasilacza, masa (obudowa komputera) przy braku zerowania ma potencjał ok. 115 V względem ziemi (wynika to z istnienia w zasilaczu układu filtrującego). W efekcie przy braku zerowania komputera lub niepoprawnym zerowaniu może dojść do uszkodzenia programatora lub programowanego układu. Aby uniknąć takich przykrych niespodzianek, można zaopatrzyć się w programator z optoizolowanymi wyjściami, lecz jest to dodatkowy, spory wydatek.

Budowa programatora

Programatory ISP są jednymi z najprostszych w budowie, w związku z tym każdy może poskładać sobie taki programator, dosłownie z niczego. W szczególnie dobrej sytuacji są tu użytkownicy posiadający komputery z wyprowadzonym portem równoległym. W takiej sytuacji programator może być zwykłą przejściówką pomiędzy portem komputera a gniazdem ISP. Układ taki jest niezwykle prosty, lecz niezalecany. Jakikolwiek błąd w połączeniach może bardzo łatwo doprowadzić do uszkodzenia portu równoległego,

dodatkowo jego niewielka wydajność prądowa powoduje znaczne ograniczenie maksymalnej długości przewodu łączącego komputer z programowanym układem (w praktyce do kilkunastu cm). Stąd znacznie lepszym rozwiązaniem jest to pokazane na rysunku 2.4.



Rysunek 2.4. *Schemat prostego programatora ISP podłączanego do portu równoległego komputera*

Jak widać, programator taki składa się z bufora 74XX244 (nie musi to być układ serii HC). Jego wprowadzenie umożliwia znaczne wydłużenie przewodu łączącego komputer z programatorem, nawet do 1 m i więcej (należy mieć na uwadze, że zbyt długi przewód łączący nie jest zalecany i w pewnych okolicznościach może prowadzić do problemów z programowaniem). Wprowadzenie tego układu chroni także port równoległy. W przypadku błędnego podłączenia zasilania np. do pinów programatora uszkodzeniu ulegnie tylko tani układ buforujący, ochraniając port równoległy. Wykonanie takiego programatora to koszt rzędu kilku złotych, lecz już za kilkanaście złotych można kupić programatory bardziej rozbudowane, których zaletami są:

- bezpośrednia współpraca z AVR Studio;
- możliwość programowania układów zasilanych innym napięciem niż 5 V;
- współpraca z portem USB mikrokontrolera.

Szczególnie ta ostania cecha jest pożądana. Porty USB, w przeciwieństwie do równoległych, występują praktycznie w każdym urządzeniu, lecz ich największą zaletą jest możliwość czerpania energii z takiego portu. Stwarza to możliwość nie tylko zasilania samego programatora, ale także zasilania programowanego układu (tu jednak trzeba mieć na uwadze ograniczoną do ok. 0,5 A wydajność prądową portu USB).

Programator AVRISP

Jest to prosty programator z możliwością podłączenia poprzez port szeregowy RS232 lub USB. Na rynku dostępne są liczne klony tego układu, w efekcie można go kupić już za kilkanaście złotych, co czyni go szczególnie interesującym dla amatora. W stosunku do prostych programatorów, posiadających tylko bufor, jego zaletą jest możliwość programowania układów zasilanych napięciem w granicach 2,7 – 5,5 V. Klasyczny programator AVRISP zasilany jest z urządzenia programowanego, lecz jego wersje na USB często posiadają zworki, umożliwiające wybór źródła zasilania. Przy jego pomocy można programować szeroką gamę modeli procesorów AVR. Wyjątkiem jest tu tylko rodzina AVR XMega oraz AVR32, wymagające programatora AVRISP mkII. Programator ten posiada wbudowany procesor, którego firmware kontroluje proces programowania. Wraz z uaktualnianiem AVR Studio firmware ten też może zostać uaktualniony, w efekcie poszerza się lista obsługiwanych procesorów.

Programator ten jest rozwiązaniem tanim, lecz warto mieć na uwadze, że obecnie nie jest on już praktycznie rozwijany przez firmę Atmel. Stąd potencjalnie mogą być problemy z jego wykorzystaniem z najnowszymi modelami procesorów.

Programator AVRISP mkII

Rozwiązaniem dla bardziej zaawansowanych amatorów i osób, które półprofesjonalnie chcą się zajmować mikrokontrolerami AVR, jest programator AVRISP mkII. Również ten programator dostępny jest w postaci klonów, w efekcie można go kupić za cenę ok. 100 – 150 zł. Jest to programator aktywnie wspierany przez firmę Atmel, wspierający wszystkie rodziny procesorów AVR (łącznie z procesorami XMega oraz AVR32). Wspiera także procesory ATTiny, nieposiadające interfejsu ISP, dzięki możliwości wykorzystania interfejsu TPI.

Programator ten może programować układy zasilane napięciem od 1,8 do 5,5 V, dodatkowo można regulować częstotliwość sygnału zegarowego taktującego transmisję w zakresie 50 Hz – 8 MHz. Ma to istotną zaletę w przypadku programowania układów niskonapięciowych, taktowanych z wolnych zegarów, np. kwarców zegarkowych o częstotliwości 32 768 Hz.

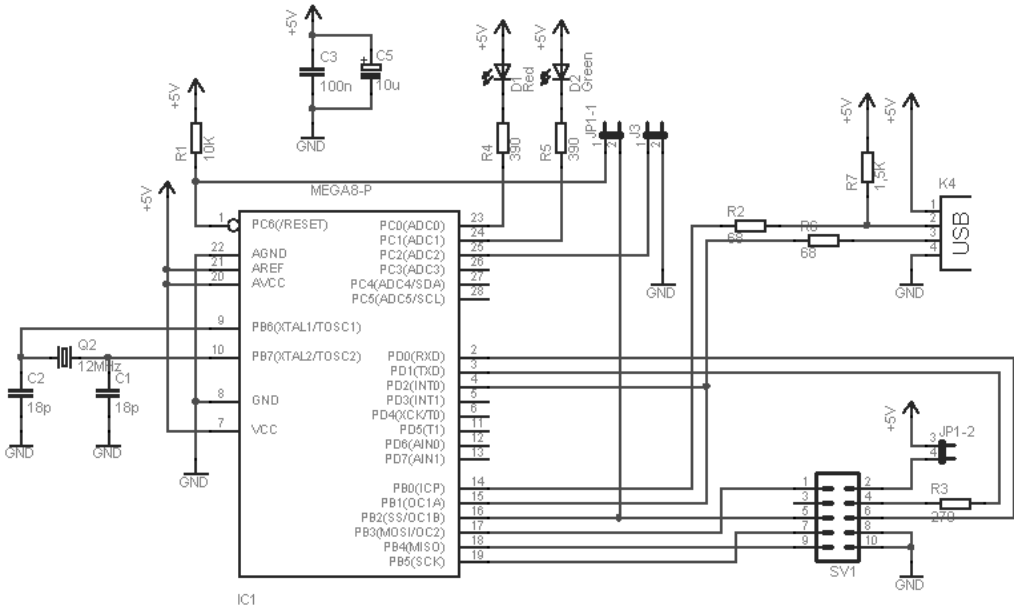
> Maksymalna prędkość programowania wynika z ograniczeń interfejsów szeregowych — częstotliwość linii *SCK* nie może być większa niż czterokrotność częstotliwości taktującej rdzeń procesora.

Tak więc wykorzystanie programatora AVRISP mkII umożliwia programowanie procesorów taktowanych zegarem od 200 Hz wzwyż.

Programator ten współpracuje z interfejsem USB, posiada także wyjścia zabezpieczone przez zwarciem.

Programator USBASP

Programator ten jest niezwykle popularny ze względu na jego prostotę oraz niską cenę. Dodatkową zaletą tego programatora jest wsparcie ze strony WinAVR oraz bardzo popularnego programu AVRDude. Schemat tego programatora pokazano na rysunku 2.5.



Rysunek 2.5. *Schemat programatora USBASP. Programator ten zawiera mikrokontroler sterujący procesem programowania układu docelowego. Dzięki temu programator ten jest niezależny od przebiegów czasowych generowanych przez komputer. Łączność z komputerem następuje poprzez złącze USB, stąd też programator czerpie zasilanie. Po zwarciu zworki JP1-2 możliwe jest także zasilanie z portu USB układu programowanego*

Programator ten umożliwia także programowanie procesorów taktowanych zegarami o niskiej częstotliwości. Przy pomocy zworki JP3 można przełączać częstotliwość linii *SCK* z 375 kHz na 8 kHz, co umożliwia programowanie układów taktowanych kwarcem zegarkowym o częstotliwości 32 768 Hz. Zwarcie zworki JP1-1 umożliwia zaprogramowanie lub uaktualnienie oprogramowania programatora poprzez jego złącze ISP.

Kilka procesorów w jednym układzie

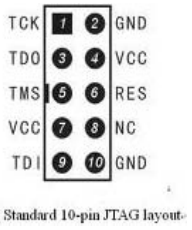
Sporadycznie zdarza się, że na jednej płytce znajduje się więcej niż jeden procesor AVR i każdy powinien mieć zapewnioną możliwość programowania. Najchętniej w takiej sytuacji chcielibyśmy móc korzystać tylko z jednego gniazda programującego. Konfiguracja taka jest możliwa, musimy tylko pamiętać o spełnieniu pewnych dodatkowych założeń. W takiej sytuacji sygnały z programatora (*RESET*, *MISO*, *MOSI*) powinny być rozprowadzone do wszystkich mikrokontrolerów. Natomiast sygnał *SCK* musi pozostać

rozdzielony. Przy takiej konfiguracji wyboru programowanego mikrokontrolera dokonuje się poprzez wybranie mikrokontrolera, do którego doprowadzony zostanie sygnał *SCK*. Układ taki może działać, ponieważ aby procesor wszedł w tryb programowania (a co za tym idzie, linie *MOSI* i *MISO* stały się aktywne), musi być spełnionych kilka założeń. Po pierwsze, programator musi zapewnić aktywność sygnału *RESET*. Dzięki utrzymywaniu go w stanie aktywnym wyprowadzenia wszystkich mikrokontrolerów przechodzą w stan wysokiej impedancji. Dzięki temu nie zakłócają one transmisji. Uaktywnienie trybu programowania wymaga w takiej sytuacji doprowadzenia do wejścia *SCK* odpowiedniego przebiegu. Ponieważ przebieg taki zostanie doprowadzony wyłącznie do wybranego procesora, inne procesory pozostaną nieaktywne, z wyprowadzeniami w stanie wysokiej impedancji.

W przypadku gdy na płytce znajduje się jeden większy procesor i jeden lub więcej procesorów ze stosunkowo niewielką ilością pamięci FLASH, można rozważyć jeszcze jedną możliwość. Funkcję programatora może przejąć procesor „większy", odpowiednio sterując wyprowadzeniami odpowiedzialnymi za programowanie innych procesorów. W takiej sytuacji ich przeprogramowanie wymaga wczytania do procesora kontrolującego pozostałe odpowiedniego programu oraz zawartości pamięci FLASH pozostałych procesorów. Rozwiązanie takie jest stosunkowo proste, lecz wymaga takiego podłączenia wszystkich mikrokontrolerów, aby ich wyprowadzenia programujące były dostępne dla procesora nadrzędnego.

Programatory JTAG

Programatory wykorzystujące interfejs JTAG są o wiele droższe, ale oprócz możliwości programowania przy ich pomocy procesora oferują także możliwość debugowania. Obecnie na rynku występują dwie wersje programatora JTAG dla mikrokontrolerów AVR — JTAGICE oraz JTAGICE II. Ten drugi cechuje się bardzo wysoką ceną (ok. 700 – 1200 zł), ale oferuje możliwość programowania wszystkich mikrokontrolerów AVR wyposażonych w interfejs JTAG. Za jego pomocą można także programować mikrokontrolery AVR32. Możliwości programatora JTAGICE są skromniejsze, ale za to jego cena jest niewiele wyższa niż programatora ISP. Programator ten łączy się z programowanym układem przy pomocy gniazda o innym rozkładzie sygnałów niż w przypadku programatora ISP — rysunek 2.6.



Rysunek 2.6. *Rozkład sygnałów na złączu JTAG. Pin 1 oznaczono kwadratem*

Interfejs JTAG wykorzystuje pięć sygnałów: *RESET*, *TCK*, *TMS*, *TDI* oraz *TDO*. Do pinu 7 (VCC) należy doprowadzić napięcie zasilające tylko w sytuacji, w której programator ma być zasilany z układu. Jeśli programator ma własne zasilanie, pin 7 można pozostawić

niepodłączony. Z kolei pin 4 dostarcza napięcia umożliwiającego programatorowi dostosowanie poziomu napięć na liniach *RESET*, *TCK*, *TMS*, *TDI* i *TDO* do napięć panujących w układzie. Na podstawie napięcia na tej linii programator wykrywa także podłączenie do układu programowanego. Wyprowadzenia oznaczone jako *NC* należy pozostawić niepodłączone.

Aby móc korzystać z tego trybu, procesor musi obsługiwać interfejs JTAG, a *fusebit* JTAGEN musi być zaprogramowany (mieć wartość 0).

Programowanie przy użyciu interfejsu JTAG ma liczne zalety:

- Jest 3 – 4 razy szybsze w stosunku do programowania przy użyciu interfejsu ISP.
- Podobnie, znacznie szybsze jest także programowanie pamięci EEPROM.
- Umożliwia zmianę *fusebitów* określających źródło sygnału zegarowego, niezależnie od ich poprzednich wartości. Interfejs JTAG sam generuje zegar dla układu docelowego, stąd wybranie nawet błędnych wartości nie blokuje możliwości dalszego programowania (odmiennie niż w przypadku interfejsu ISP).
- Istnieje możliwość łączenia urządzeń w konfigurację *daisy-chain*, umożliwiającą programowanie wielu urządzeń przy pomocy jednego złącza JTAG.
- Istnieje możliwość programowania nie tylko mikrokontrolerów AVR, ale także innych układów kompatybilnych ze standardem JTAG (np. FPGA).

Niezwykle istotną zaletą interfejsu JTAG jest możliwość debugowania przy jego pomocy programu w trakcie jego działania w docelowym układzie elektronicznym.

Możliwość taka jest wprost trudna do przecenienia, szerzej zostanie opisana w rozdziale 29.

Programator JTAGICE

Cena tego układu porównywalna jest z ceną dobrego programatora ISP. Jest to więc propozycja dla hobbystów zdecydowanie poważniej myślących o zajęciu się budowaniem układów w oparciu o mikrokontrolery AVR. Zastosowania tego programatora ogranicza stosunkowo niewielka liczba wspieranych układów [ATmega16(L), ATmega162(L), ATmega169(L or V), ATmega32(L), ATmega323(L), ATmega64(L), ATmega128(L)]. Lecz nawet pomimo tej wady warto rozważyć jego zakup, szczególnie jeśli jesteśmy w posiadaniu płytki rozwojowej zawierającej jeden z wyżej wymienionych procesorów. Programowanie przy jego pomocy jest nie tylko szybsze, lecz przede wszystkim udostępnia szerokie możliwości debugowania układu w systemie. Dzięki temu nawet jeśli pisany program będzie docelowo działał na innym typie procesora, łatwiej jest napisać aplikację na jednym ze wspieranych przez JTAGICE procesorów, a następnie ją tylko zmodyfikować dla potrzeb procesora docelowego. Użycie interfejsu JTAG umożliwia nie tylko debugowanie samego programu, ale także sprawdzenie stanu wszystkich bloków

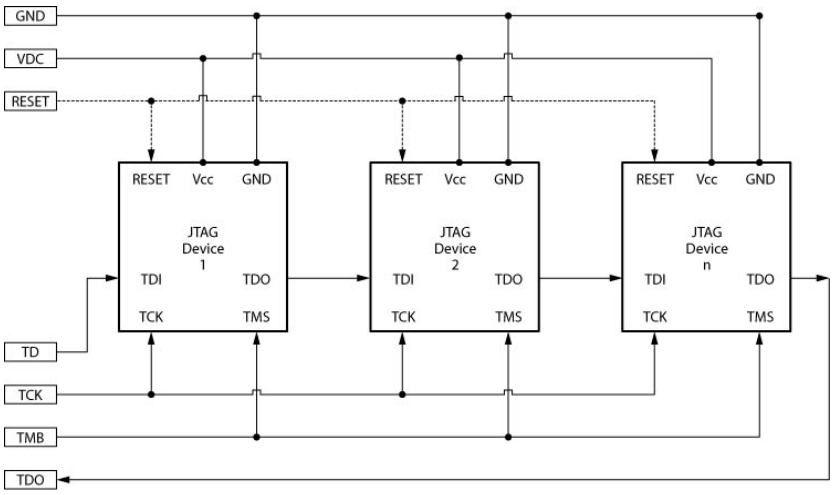
procesora, a także jego portów *IO*. Oprócz możliwości sprawdzenia stanu można ich stan także modyfikować „w locie". Ułatwia to testowanie poprawności połączeń elektrycznych na płytce i poprawności montażu.

Programator JTAGICE mkII

Programator JTAGICE mkII jest rozwinięciem układu JTAGICE. Umożliwia on programowanie wszystkich procesorów AVR wyposażonych w interfejs JTAG, w tym także procesorów z rodziny AVR32. Ze względu na cenę tego programatora (przekraczająca 1000 zł) jest to raczej propozycja dla osób chcących bardziej profesjonalnie zająć się programowaniem i budowaniem układów w oparciu o mikrokontrolery. Funkcjonalnie programator ten nie różni się od swojego poprzednika, udostępnia podobne możliwości. Oprócz programowania przez interfejs JTAG udostępnia także możliwość programowania z wykorzystaniem interfejsów PDI, debugWire, SPI oraz aWire. W efekcie za jego pomocą można zaprogramować praktycznie wszystkie procesory AVR.

Kilka procesorów w jednym układzie

Podobnie jak w przypadku ISP, także JTAG umożliwia wykorzystanie jednego złącza do programowania kilku układów. Funkcja taka jest wpisana w specyfikację protokołu JTAG, więc teoretycznie taka konfiguracja powinna być nawet łatwiejsza w realizacji. Tu, niestety, jak to zwykle bywa, napotykamy na problemy natury programowej. Większość dostępnego oprogramowania nie wspiera możliwości wybierania procesora w konfiguracji łańcuchowej JTAG (ang. *Daisy-chain JTAG mode*). Sytuacja ta stopniowo się zmienia i część oprogramowania dostarczonego przez firmę Atmel wspiera taką konfigurację dla programatora AVRICE mkII. Schemat podłączenia interfejsów JTAG w konfiguracji *daisy-chain* pokazano na rysunku 2.7. Linie *TDI* i *TDO* kolejnych procesorów są połączone szeregowo.



Rysunek 2.7. *Połączenie kilku układów AVR, wykorzystujących jedno złącze JTAG*

Inną możliwością jest rozwiązanie analogiczne do pokazanego przy okazji programowania ISP — połączenie równoległe odpowiednich linii JTAG, z wyjątkiem linii *SCK*. Wybór aktywnej linii *SCK* umożliwia wybór programowanego/debugowanego układu.

> Korzystając z możliwości konfiguracji *daisy-chain*, należy mieć na uwadze jeszcze jeden problem — niektóre mikrokontrolery AVR mają błędną implementację obsługi JTAG, uniemożliwiającą zastosowanie konfiguracji *daisy-chain*. Stąd przed jej użyciem należy zawsze sprawdzić erratę do noty katalogowej procesora, zgodną z jego modelem oraz wersją układu.

AVR Dragon

Alternatywą dla wcześniej wymienionych programatorów, w tym dla drogiego JTAGICE mkII, jest układ AVR Dragon. W przeciwieństwie do wcześniejszych układów jest on sprzedawany bez obudowy, złącz i kabli. Potrzebne złącza należy wlutować samemu. Dzięki temu jego cena jest niezwykle atrakcyjna — można go kupić w cenie ok. 200 – 240 zł. **Niestety, brak wielu wbudowanych zabezpieczeń czyni go niezwykle podatnym na uszkodzenie.** Aby taką możliwość znacznie zmniejszyć, należy samemu dodać odpowiednie układy zabezpieczające — np. układy buforujące wyjścia programatora. Układ AVR Dragon umożliwia programowanie wszystkich mikrokontrolerów AVR dzięki wyposażeniu go w interfejsy HVPP, HVSP, ISP, JTAG, PDI. Umożliwia także debugowanie układu docelowego dzięki interfejsom JTAG i debugWire. Programowane układy mogą być zasilane napięciem z zakresu 1,8 – 5,5 V. Układ AVR Dragon może także dostarczać dla nich napięcia zasilającego o natężeniu maksymalnie 300 mA.

Programatory HW i równoległe

Programatory wysokonapięciowe (HW, ang. *High Voltage*) oraz równoległe są niezwykle rzadko wykorzystywane. Programator wysokonapięciowy wykorzystuje podobne sygnały co programator ISP, lecz podczas programowania na wejściu *RESET* procesora zamiast stanu niskiego doprowadzane jest napięcie +12V. Dzięki temu można programować procesory, w których przy pomocy *fusebitu* RSTDSBL wejście *RESET* zostało zablokowane. Nie wszystkie procesory dysponują możliwością programowania wysokonapięciowego. W tym trybie programowania procesor wymaga doprowadzenia sygnałów pokazanych w tabeli 2.1.

Tabela 2.1. *Sygnały wykorzystywane do programowania w trybie wysokonapięciowym*

Sygnał	Kierunek	Opis
SDI	Wejście	Wejście danych
SII	Wejście	Wejście instrukcji
SDO	Wyjście	Wyjście danych
SCI	Wejście	Wejście zegarowe

Programatory równoległe wykorzystywane są jeszcze rzadziej. Ich potencjalną zaletą jest większa szybkość działania, lecz do poprawnej pracy wymagają podłączenia kilkunastu różnych sygnałów. Zaletą tego typu programatorów jest możliwość programowania procesora zablokowanego w wyniku przeprogramowania *fusebitów* odpowiedzialnych za wybór zegara. Jest to możliwe, ponieważ w tym trybie programator generuje przebieg zegarowy taktujący procesor, który jest doprowadzony do wejścia *XTAL1*.

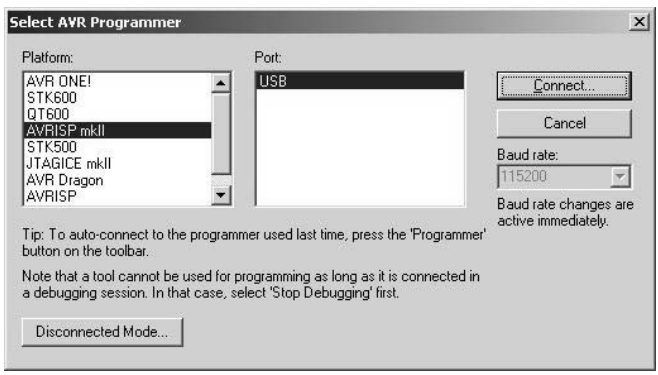
Tryb TPI

Jest to uproszczony interfejs umożliwiający programowanie najmniejszych procesorów Atmel z serii ATTiny. Używa on linii *RESET* oraz linii danych *TPIDATA* i zegara *TPICLK*. W przypadku kiedy pin *RESET* jest wykorzystywany jako zwykły pin *IO*, wejście w tryb TPI jest wymuszane poprzez podanie na ten pin napięcia +12 V. Protokół ten wspierany jest przez najnowsze programatory, m.in. AVRISP mkII, AVR Dragon.

Programowanie procesora w AVR Studio

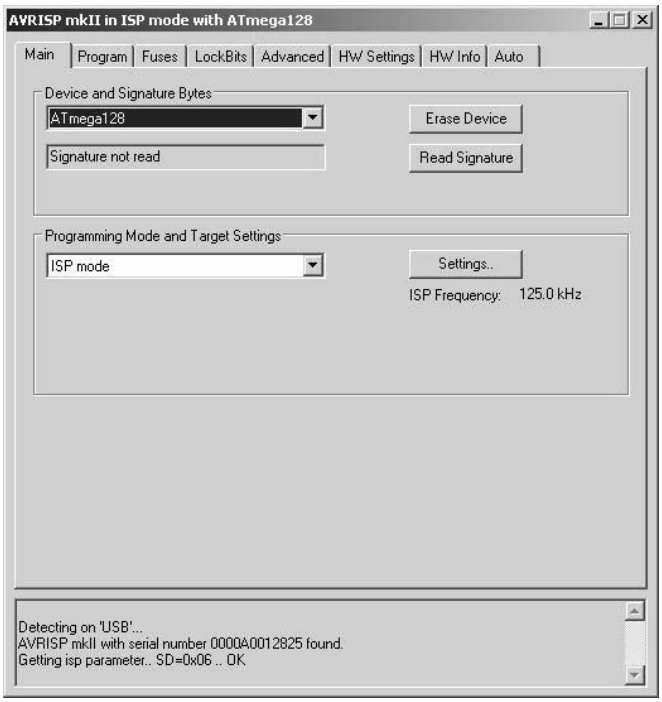
Zdecydowanie najłatwiejszą opcją jest wykorzystanie do programowania zintegrowanego środowiska, jakim jest AVR Studio. Dzięki temu mamy możliwość, przy pomocy jednego programu, pisać program, kompilować go, debugować (za pomocą interfejsów sprzętowych lub wbudowanego w AVR Studio symulatora), a efekt finalny przy pomocy jednego przycisku wgrywać do pamięci procesora. Bezpośrednio AVR Studio wspiera narzędzia dostarczane przez firmę Atmel — programatory AVRISP, JTAGICE, Dragon. Po pewnych zabiegach można także korzystać z innych programatorów.

Rozpoczęcie procesu programowania w AVR Studio wymaga najpierw skonfigurowania interfejsu programatora — czyli wybrania z listy programatora, który posiadamy, oraz podania sposobu komunikacji z nim (rysunek 2.8). Opcję konfiguracji wybiera się z menu *Tools/Program AVR/Connect*.



Rysunek 2.8. *Konfiguracja programatora w AVR Studio — w powyższym przykładzie został wybrany programator AVRISP mkII, podłączony przez port USB*

Po udanej próbie nawiązania połączenia z programatorem wyświetlone zostanie kolejne okno, z opcjami, jakie możemy wybrać. Opcje niewspierane przez dany typ programatora nie będą dostępne (rysunek 2.9).



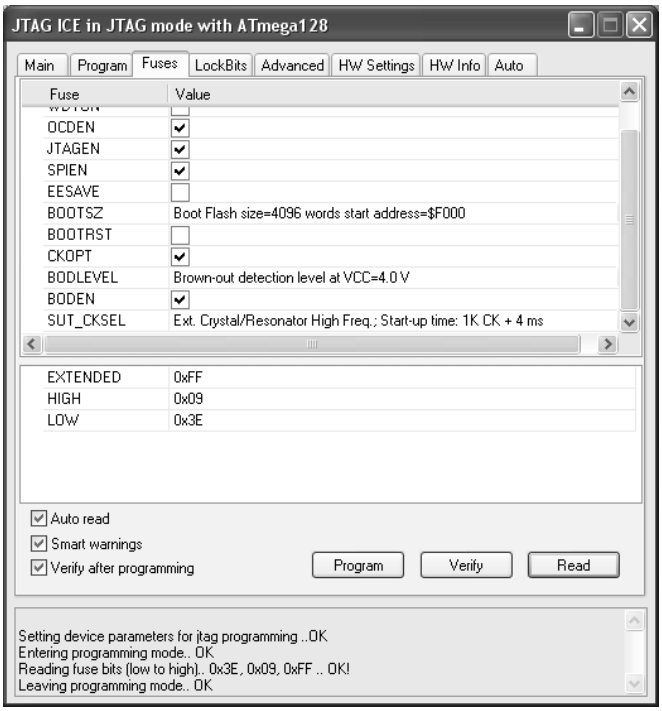
Rysunek 2.9. *Opcje dostępne dla programatora AVRISP MkII. Ponieważ programator ten można łączyć z układem programowanym przy pomocy różnych interfejsów, właściwy tryb połączenia należy wybrać w menu Programming Mode and Target Settings. Dla tego programatora należy także określić częstotliwość zegara taktującego transmisję, pamiętając, że nie może ona być wyższa niż ¼ częstotliwości zegara taktującego rdzeń procesora*

W oknie konfiguracji można także ustawić konfigurację *fusebitów* oraz *lockbitów* (zakładki *Fuse* oraz *LockBits*) — rysunek 2.10.

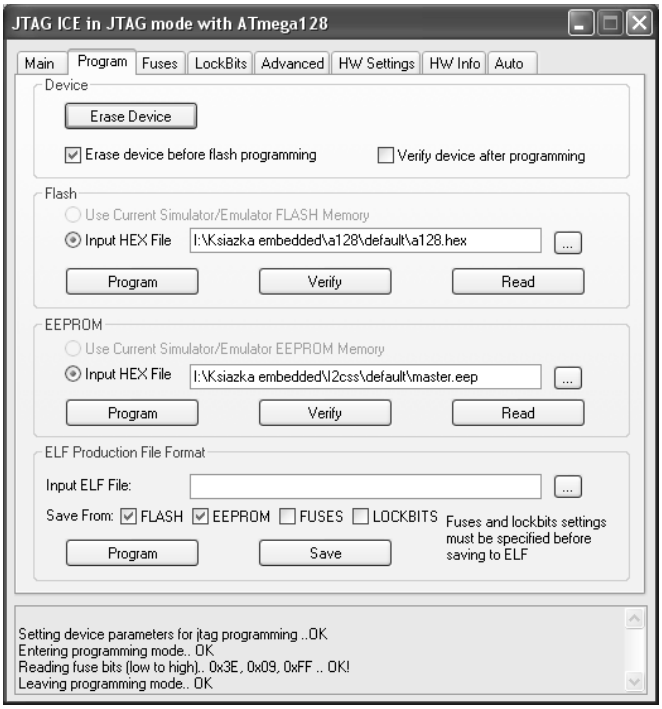
W zakładce *Program* określa się ścieżki do plików zawierających program, który chcemy wczytać do mikrokontrolera (pliki muszą być w formacie *elf* lub *Intel HEX*) — rysunek 2.11.

Programowanie przy pomocy narzędzi dostarczonych przez firmę Atmel

Firma Atmel wraz z AVR Studio dostarcza wielu różnych programów umożliwiających programowanie z linii poleceń przy pomocy programatorów kompatybilnych z protokołami firmy Atmel. Służą one generalnie do automatyzacji procesu programowania w przypadku programowania dłuższych serii procesorów. Wśród licznych programów na szczególną uwagę zasługuje program FLIP. Nie jest on dostarczany razem z AVR Studio, lecz wymaga osobnego pobrania ze strony *www.atmel.com* i instalacji. Wśród licznych



Rysunek 2.10. *Konfiguracja fusebitów w AVR Studio. W przypadku złożonych operacji (np. wyboru zegara) zamiast wybierać konfigurację poszczególnych fusebitów, możemy posłużyć się rozwijalnymi listami z możliwymi do wybrania opcjami. W znacznym stopniu ogranicza to możliwość pomyłki. Wybraną konfigurację fusebitów wprowadza się do procesora po naciśnięciu przycisku Program*



Rysunek 2.11. *W zakładce Program określa się ścieżki dostępu do plików wykorzystywanych w trakcie programowania. Najwygodniej jest użyć pliku w formacie elf, gdyż zawiera on wszystkie niezbędne do zaprogramowania procesora dane. Alternatywnie można podać ścieżki do plików hex i eep, zawierających wsad do zaprogramowania pamięci FLASH i EEPROM*

jego możliwości jest także możliwość programowania urządzeń wyposażonych w *bootloader* kompatybilny ze specyfikacją Atmela dla urządzeń klasy DFU (ang. *Device Firmware Update*). Do tej klasy urządzeń zalicza się m.in. procesory AVR wyposażone w sprzętowy interfejs USB. Są one sprzedawane z firmowo wgranym *bootloaderem*, umożliwiającym wczytanie oprogramowania do pamięci FLASH i EEPROM mikrokontrolera.

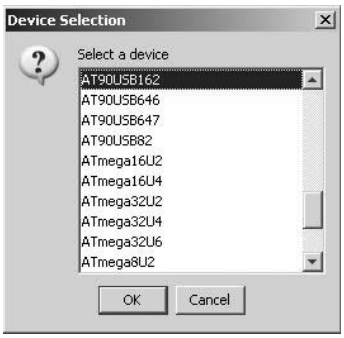
Tryb DFU nie umożliwia zmiany *fusebitów*. W tym celu należy posłużyć się innym programatorem.

Aby uruchomić wbudowany w urządzenie *bootloader*, podczas wyprowadzenia urządzenia ze stanu *RESET* należy zewrzeć do masy pin HWB. Dzięki temu zamiast programu zostanie uruchomiony *bootloader* umożliwiający wczytanie nowego oprogramowania.

Po podłączeniu programowanego układu do komputera przy pomocy USB i uruchomieniu *bootloadera* przy pomocy pinu HWB urządzenie jest gotowe do programowania.

Aby klasa DFU była rozpoznawana przez komputer, należy zainstalować sterowniki DFU dostarczone przez firmę Atmel.

Po uruchomieniu programu FLIP wybieramy z menu *Device/Select*; w efekcie ukazuje się okno wyboru procesora (rysunek 2.12).

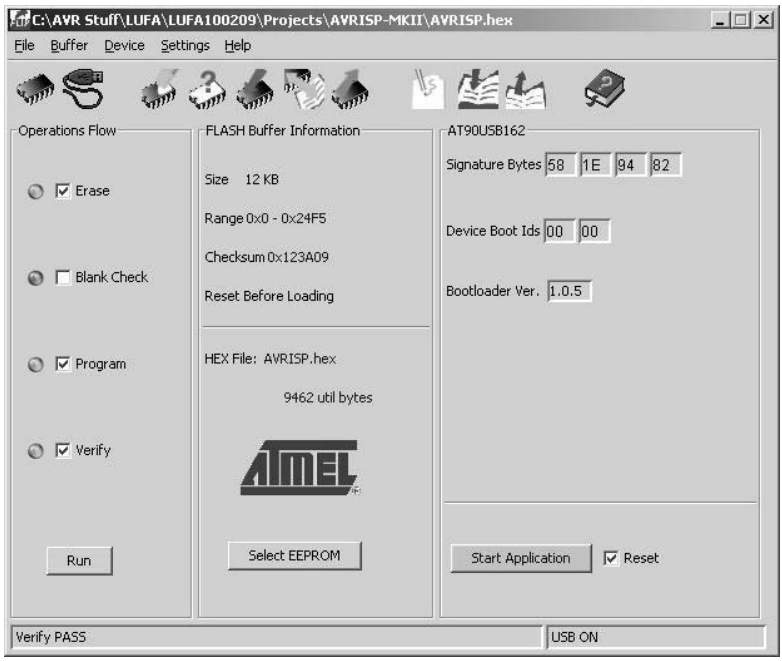


Rysunek 2.12. *Okno wyboru procesora. Będzie on programowany przy pomocy bootloadera w trybie DFU*

Po wyborze procesora klikamy na *Open*, co powoduje nawiązanie połączenia z programowanym układem. Następnie wczytujemy pliki do zaprogramowania (*File/Load HEX*), wybieramy opcje programowania i weryfikacji układu i klikamy na przycisk *Run*, co inicjuje proces uaktualniania oprogramowania (rysunek 2.13).

Program AVRDUDE

Jest to jeden z najpopularniejszych programów używanych do programowania mikrokontrolerów AVR. Jest on dostarczany wraz z pakietem WinAVR. Sam program AVRDUDE jest aplikacją uruchamianą z wiersza pleceń, parametry podaje się jako opcje wywołania.



Rysunek 2.13. *Proces uaktualniania oprogramowania przy pomocy programu FLIP*

Aby uczynić go nieco bardziej przyjaznym, w Internecie dostępnych jest wiele graficznych nakładek, umożliwiających uzyskanie tych samych efektów przy pomocy prostego interfejsu graficznego. Program AVRDUDE obsługuje następujące programatory:

- STK500, STK600,
- AVRISP i AVRISP mkII,
- AVRICE i AVRICE mkII,
- proste programatory podłączane do wyjścia równoległego i szeregowego komputera.

Program ten wspiera wszystkie protokoły transmisji używane przez firmę Atmel. Niestety, do programowania można posłużyć się wyłącznie plikami *IntelHEX*, gdyż nie wspiera on formatu *elf*. W efekcie musimy dysponować oddzielnymi plikami zawierającymi obrazy pamięci FLASH, EEPROM, a także wartościami numerycznymi *fuse-* i *lockbitów*. Stwarza to pewne dodatkowe możliwości pomyłki.

Program ten może pracować w dwóch trybach — terminalowym oraz wywoływany z wiersza poleceń. Poniżej krótko pokazane zostaną podstawowe opcje wywołania, umożliwiające zaprogramowanie przy jego pomocy mikrokontrolera.

Parametry wywołania:

- `-p` *procesor* — jest to obowiązkowy parametr wywołania programu. Określa on typ procesora podłączonego do programatora. Listę dostępnych typów można wyświetlić, wydając polecenie `avrdude -p ?`. W efekcie powinna wyświetlić się lista wspieranych typów procesorów.

- `-B` *okres* — parametr ten jest używany przy programowaniu za pomocą interfejsu JTAG w trybie ISP. Umożliwia on określenie prędkości programowania poprzez podanie okresu (w mikrosekundach) sygnału *SCK*. Np. `avrdude -B 1` powoduje, że linia *SCK* będzie taktowana sygnałem o częstotliwości 1 MHz.
- `-c` *programator* — określa typ programatora, który ma zostać użyty do programowania mikrokontrolera. Listę dostępnych typów można wyświetlić poleceniem `avrdude -c ?`. Na liście tej należy odnaleźć używany programator.


Wskazówka

Czasami dany programator wspiera różne protokoły programowania. W takiej sytuacji będzie występował na liście wiele razy z sufiksami określającymi wybrany tryb programowania.

- `-F` — powoduje, że program nie weryfikuje sygnatury układu z typem podanym jako parametr `-p`. W nielicznych sytuacjach umożliwia to obejście pewnych problemów związanych z uszkodzeniem sygnatury procesora, lecz normalnie opcja ta nie powinna być używana.
- `-n` — wykonuje wszystkie operacje, ale bez fizycznego zapisu do układu. Jest to przydatne do testowania różnych skryptów automatyzujących proces programowania.
- `-O` — przeprowadza kalibrację wewnętrznego generatora RC zgodnie z opisem z noty AVR053. Uzyskany w wyniku kalibracji bajt kalibracyjny jest zapisywany do komórki pamięci EEPROM o adresie 0, skąd może zostać odczytany przez program i użyty do kalibracji rejestru `OSCCAL` mikrokontrolera. Co prawda operacja ta nie poprawia stabilności wewnętrznego generatora RC, ale określa dokładniej jego częstotliwość.
- `-U` *obszar*`:`*typ*`:`*plik*`[:`*format*`]` — opcja ta przeprowadza operację na wskazanym obszarze (może to być operacja odczytu lub zapisu). Parametr *obszar* może być jednym z symboli: `eeprom`, `flash`, `fuse`, `hfuse`, `lfuse`, `efuse`, `lock`. Określa on obszar podlegający danej operacji, zgodnie z nazwą podanych symboli. Parametr *typ* określa typ operacji: `r` — odczyt, `w` — zapis, `v` — weryfikacja, Parametr *plik* określa nazwę pliku, z którego będą odczytywane dane w przypadku operacji zapisu lub do którego będą zapisywane dane w przypadku operacji odczytu. Ostatni parametr, *format*, określa format pliku. Z licznych formatów istotne są `i` — określający, że plik jest w formacie *IntelHEX*, i `m` — określający, że parametr będzie wartością bezpośrednią, podaną w linii wywołania (najczęściej używane do programowania *fuse-* i *lockbitów*).

Wywołanie:

```
avrdude -p m88 -u -U flash:w:test.hex -U eeprom:w:test.eep -U efuse:w:0xff:m -U
↪hfuse:w:0x89:m -U lfuse:w:0x2e:m
```

spowoduje zaprogramowanie procesora ATMega88 plikami *test.hex* i *test.eep*, których zawartość zostanie umieszczona odpowiednio w pamięci FLASH i EEPROM mikrokontrolera. Dodatkowo wartość *fusebitów* zostanie ustawiona na 0x2E89FF. Z kolei wywołanie:

```
avrdude -c avrisp2 -P usb -p t26 -U flash:w:main.hex:i
```

powoduje zaprogramowanie procesora typu ATTiny26 plikiem *main.hex*, przy pomocy programatora AVRISP mkII, podłączonego przez port USB, w trybie ISP.

Program PonyProg

Program PonyProg jest bardzo prostym programem umożliwiającym programowanie różnych układów przy pomocy prostych interfejsów podłączanych do portu równoległego lub szeregowego komputera. Obsługuje on pliki *Intel Hex*. Program można pobrać ze strony *http://www.lancos.com/prog.html*. Oprócz programowania procesorów przy jego pomocy można także programować różnego typu pamięci szeregowe. Po skonfigurowaniu typu posiadanego programatora (opcja *Setup/Interface Setup*) należy wybrać typ programowanego układu. Następnie z menu *File* wybieramy *Open Program File* oraz *Open Data File* i wczytujemy uzyskane w trakcie kompilacji pliki z rozszerzeniem *hex* i *eep*. Ostatnią czynnością jest zaprogramowanie procesora poleceniem *Command/Write All*. PonyProg umożliwia także konfigurację *fuse-* i *lockbitów*. W tym względzie jego prostota prowadzi często do błędów. Do dyspozycji mamy tylko pojedyncze *fusebity*, których odpowiednią wartość należy ustalić po przejrzeniu sekcji noty katalogowej procesora poświęconej konfiguracji *fusebitów*.

Należy pamiętać, że podobnie jak w przypadku innych programów, *fusebit* zaprogramowany oznacza *fusebit* o wartości 0.

Po skonfigurowaniu *fusebitów* wybieramy opcję *Write*, co powoduje ich zapisanie do procesora.

Fusebity i lockbity w AVR-libc

Biblioteka AVR-libc udostępnia wygodny sposób modyfikacji bitów konfiguracyjnych procesora. Ponieważ bity te nie mogą być zmieniane programowo, aby taka konfiguracja była możliwa, potrzebne jest specjalne oprogramowanie wspierające funkcje biblioteki AVR-libc. Zwykle wspierają taką możliwość programy, które jako źródło danych do programowania procesora wykorzystują pliki w formacie *elf*. W plikach w formacie *Intel HEX* nie ma możliwości umieszczenia informacji o konfiguracji *fuse-* i *lockbitów*; w efekcie programatory wykorzystujące ten format nie wspierają funkcji AVR-libc. W takiej sytuacji pozostaje ręczna konfiguracja tych bitów poprzez wybranie odpowiednich opcji programatora.

Programując *lock-* i *fusebity*, należy pamiętać, że wartości jeden odpowiada *fusebit* niezaprogramowany, natomiast wartości 0 — zaprogramowany.

Aby *fusebity* i *lockbity* zostały poprawnie skonfigurowane i umieszczone w wynikowym pliku *elf*, należy wybrać właściwy typ procesora. Błędne ustawienie typu procesora może spowodować jego zablokowanie na skutek próby wpisania nieprawidłowej konfiguracji *fuse-* i *lockbitów*.

Lockbity

Lockbity zostały dokładnie omówione w rozdziale 25. Ich funkcją jest ochrona pamięci mikrokontrolera przed możliwością jej odczytania przy pomocy programatora. Dzięki temu umieszczony w pamięci FLASH program po zaprogramowaniu *lockbitów* nie daje się odczytać. Ich zaprogramowanie nie blokuje możliwości komunikacji z procesorem, przy próbie odczytu zwracane są wartości będące kolejnymi adresami komórek pamięci FLASH. Natomiast w żaden sposób nie da się odczytać ich zawartości, mimo że programator nie zasygnalizuje żadnego błędu. Raz zaprogramowane *lockbity* można skasować wyłącznie razem z kasowaniem pamięci FLASH i EEPROM poleceniem *Chip Erase*. W ten sposób odzyskujemy możliwość programowania i odczytywania zawartości pamięci FLASH procesora, lecz jednocześnie tracimy zawarty w niej poprzednio program. Zwykle właściwa konfiguracja *lockbitów* określana jest na końcowym etapie tworzenia urządzenia. Nie ma sensu ich używać w trakcie pisania aplikacji.

Fusebity

Wszystkie procesory AVR posiadają tzw. *fusebity*, umożliwiające określenie konfiguracji początkowej procesora po włączeniu zasilania. W zależności od modelu procesora dostępne *fusebity* mogą się nieznacznie różnić, oferując więcej lub mniej opcji konfiguracyjnych. Poniżej przedstawiona zostanie krótko charakterystyka poszczególnych *fusebitów*. Należy pamiętać, że nowa konfiguracja *fusebitów* zaczyna obowiązywać dopiero po wyjściu z trybu programowania procesora. Dzięki temu jeśli przypadkowo wprowadzono nieprawidłową konfigurację *fusebitów*, to można ją poprawić, o ile procesor nadal znajduje się w trybie programowania.

Fusebity BODLEVEL

Są one odpowiedzialne za konfigurację układu odpowiedzialnego za detekcję awarii zasilania. Jeśli napięcie zasilające procesor będzie poniżej progu wyznaczonego wartością *fusebitów* `BODLEVEL`, procesor utrzymywany będzie w stanie resetu do czasu, aż napięcie wróci do wartości prawidłowych. Domyślnie ich konfiguracja odpowiada zablokowanemu układowi detekcji awarii zasilania. **W gotowym układzie właściwie w każdej sytuacji należy włączyć ten układ ochronny.** Zapobiega to pracy procesora przy napięciach spoza specyfikacji, co może doprowadzić do nieprawidłowej pracy rdzenia procesora i układów peryferyjnych. Często spotykanym problemem przy wyłączonym układzie BOD jest uszkodzenie komórek pamięci EEPROM. Włączenie układu BOD praktycznie eliminuje ten problem. Układ BOD można wyłączać okresowo w systemach, w których pobór mocy jest szczególnie istotny. Jego wyłączenie nieznacznie zmniejsza pobór energii przez procesor. Szczegółowo zostało to omówione w rozdziale poświęconym trybom oszczędzania energii.

Fusebit WDTON

Jego zaprogramowanie powoduje włączenie układu *watchdoga*. W takiej sytuacji układ ten nie może zostać wyłączony. O konsekwencjach włączenia układu *watchdoga* szerzej napisano w rozdziale 5.

Fusebit EESAVE

Ma on znaczenie tylko podczas kasowania pamięci procesora przed programowaniem. Jego zaprogramowanie powoduje zachowanie zawartości pamięci EEPROM, zawartość pamięci FLASH nadal będzie mogła być normalnie kasowana. *Fusebit* ten ma zastosowanie w sytuacjach, w których wgrywana jest przy pomocy programatora nowa zawartość pamięci FLASH, a jednocześnie nowy program ma mieć dostęp do danych umieszczonych w pamięci EEPROM przez program poprzedni. Poza sytuacją programowania procesora przez programator, *fusebit* `EESAVE` nie ma znaczenia.

Fusebity BOOTSZ i BOOTRST

Ich znaczenie zostało szerzej omówione w rozdziale 25.

Fusebit JTAGEN

Fusebit ten umożliwia wyłączenie interfejsu JTAG. Domyślnie procesory sprzedawane są z zaprogramowanym *fusebitem* `JTAGEN` (o ile posiadają interfejs JTAG). **Odblokowanie układu JTAG powoduje przejęcie kontroli nad pinami** `IO` **wspólnymi z tym interfejsem, co jest częstą przyczyną pomyłek.** Nad takimi pinami nie ma żadnej kontroli ze strony programu. Jeśli interfejs JTAG nie jest używany, można go wyłączyć, dzięki czemu wykorzystywane przez niego piny `IO` zostaną zwolnione i będą mogły zostać wykorzystane w programie.

Fusebit SPIEN

Fusebit `SPIEN` odblokowuje interfejs ISP procesora. Domyślnie procesory są sprzedawane z zaprogramowanym *fusebitem* `SPIEN`, dzięki czemu można je programować przy pomocy programatorów szeregowych. Tego *fusebitu* nie można skasować w trybie programowania szeregowego — można to uczynić np. w trybie programowania poprzez interfejs JTAG lub w trybie wysokonapięciowym.

Fusebit CKDIV8

Określa on częstotliwość taktowania procesora. Domyślnie procesory sprzedawane są z zaprogramowanym *fusebitem* `CKDIV8`, w efekcie zegar taktujący jest dzielony przez 8, co prowadzi do częstych pomyłek — np. wyliczone pętle opóźniające są 8-krotnie dłuższe. Jego zaprogramowanie powoduje wpisanie po *resecie* do rejestru preskalera zegara (`CLKPR`) wartości odpowiadającej podziałowi przez 8. Zamiast kasować ten *fusebit*, można programowo zmienić wartość preskalera.

Przykładowe programy przedstawione w dalszej części książki zakładają, że *fusebit* CKDIV8 ma wartość 1 — ponieważ nie jest to domyślna wartość tego *fusebitu*, należy przed uruchomieniem aplikacji go przeprogramować.

Fusebity SUT

Określają one liczbę cykli zegara po włączeniu zasilania, po których procesor będzie wyprowadzony ze stanu reset. W większości przypadków ich konfiguracja nie ma znaczenia, tym bardziej że ich wartością domyślną jest najdłuższy możliwy czas wyjścia z resetu. W przypadku kiedy zasilanie procesora szybko ulega po włączeniu stabilizacji, czas ten można skrócić.

Fusebit CKOUT

Powoduje on wyprowadzenie na wyjście procesora *CKOUT* zbuforowanego zegara taktującego rdzeń. Dzięki temu inne układy mogą korzystać z zegara procesora, umożliwia to także synchroniczną pracę innych układów z procesorem. Domyślnie ten *fusebit* nie jest zaprogramowany, w efekcie wyjście *CKOUT* zachowuje się jak normalny pin portu *IO*.

Fusebity CKSEL

Określają one sposób generowania sygnału zegarowego dla procesora. Prawdopodobnie są to *fusebity* sprawiające najwięcej kłopotów, gdyż ich nieprawidłowe ustawienie może doprowadzić do zablokowania procesora (niemożność dalszego programowania w trybie ISP). Domyślną ich wartością jest wartość wybierająca jako źródło zegara wewnętrzny generator RC. W wielu przypadkach to domyślne ustawienie jest wystarczające i nie ma potrzeby go zmieniać. Potrzeba taka zachodzi, jeśli chcemy taktować procesor przy pomocy zewnętrznego układu generującego zegar lub przy pomocy kwarcu. **Konfiguracja tych bitów jest zależna od procesora i przed ich zmianą należy skonsultować się z notą katalogową używanego układu.**

Jeśli zdarzy się nam wybrać niewłaściwe źródło zegara, procesor można „uratować", doprowadzając zewnętrzny przebieg zegarowy do wejścia *XTAL1*. Dzięki temu możliwe będzie nawiązanie komunikacji z programatorem i ponowne przeprogramowanie *fusebitów*.

Mikrokontrolery AVR mogą być taktowane z trzech źródeł zegara:

1. Zegara zewnętrznego doprowadzonego do wejścia *XTAL1*. Jest to rzadko wykorzystywana możliwość. W tym trybie potrzebny jest zewnętrzny generator zegara, którym może być np. inny procesor AVR. Umożliwia to synchronizację pracy obu kontrolerów.
2. Zegara wewnętrznego (układ generatora RC). W tym trybie (jest on trybem domyślnym) procesor jest taktowany z własnego generatora, w efekcie nie trzeba doprowadzać zewnętrznego przebiegu zegarowego. Wadą generatora wewnętrznego jest jego stosunkowo niewielka stabilność. W efekcie nie może on być wykorzystywany np. w programach, w których wymagana jest duża stabilność zegara.

3. Zewnętrznego rezonatora kwarcowego. W tym celu do pinów *XTAL1* i *XTAL2* należy podłączyć rezonator kwarcowy o pożądanej częstotliwości. Tryb ten umożliwia taktowanie procesora ze stabilnego źródła zegara.

Fusebit RSTDISBL

Zwykle pin RESET procesora współdzieli wyprowadzenie w pinem IO. Aby móc wykorzystać pin RESET jako normalny pin IO, należy zaprogramować *fusebit* RSTDISBL. **Po jego zaprogramowaniu dalsze programowanie procesora w trybie szeregowym jest niemożliwe.** Procesor można nadal programować programatorem wysokonapięciowym lub równoległym.

Fusebit DWEN

Jego zaprogramowanie uruchamia możliwość wykorzystania interfejsu debugWire, przy pomocy którego można programować procesor oraz w ograniczonym stopniu debugować program. Tylko nieliczne procesory dysponują tą funkcją, w dodatku wymaga ona posiadania specjalnego programatora obsługującego debugWire.

Sygnatura

Wszystkie procesory AVR dysponują unikalną sygnaturą nadawaną im w czasie procesu produkcji. Sygnatura ta określa typ procesora i ilość pamięci. Dzięki temu programator, odczytując sygnaturę, może weryfikować, czy podłączony procesor odpowiada procesorowi wybranemu przez użytkownika. **Sygnatura nie zawiera unikalnego dla konkretnego chipu numeru seryjnego.**

Lockbity w AVR-libc

Definicje wspieranych przez dany model procesora *lockbitów* zawiera plik nagłówkowy <*avr/io.h*>. Po jego włączeniu można włączyć plik <*avr/io.h*> zawierający definicję rożnych makrodefinicji związanych z konfiguracją *lockbitów* dla danego procesora. Po włączeniu do programu dostajemy do dyspozycji makrodefinicję LOCKBITS, której można przypisać pożądaną kombinację *lockbitów*. Domyślną wartością tego makra jest 0xFF; w efekcie wszystkie *lockbity* pozostają niezaprogramowane. Jeśli chcemy to zmienić, należy symbolowi LOCKBITS przypisać nową wartość, najlepiej posługując się predefiniowanymi w pliku <*avr/io.h*> symbolami. **Poszczególne symbole można łączyć ze sobą przy pomocy iloczynu bitowego.** Przykładowo:

```
LOCKBITS=(BLB1_MODE_3 & LB_MODE_3);
```

spowoduje wybranie trybu 3 dla kodu aplikacji oraz *bootloadera*. Więcej o trybach ochrony pamięci przy pomocy *lockbitów* znajdziesz w rozdziale 25.

Makrodefinicję LOCKBITS można zainicjować tylko raz, przyporządkowanie jej innej wartości w dalszych fragmentach programu nie odnosi żadnego skutku.

Wartość ustawionych *lockbitów* dla skompilowanego programu można odczytać z pliku *elf* przy pomocy polecenia:

```
avr-objdump -s -j .lock <plik ELF>
```

W efekcie dla wartości *lockbitów* określonej powyżej dla procesora ATMega88 powinniśmy uzyskać taki rezultat:

```
test.elf:     file format elf32-avr

Contents of section .lock:
 830000 cc                                      .
```

Wartość *lockbitów* wynosi więc 0xCC.

Fusebity w AVR-libc

Podobnie jak w przypadku *lockbitów*, *fusebity* określone w programie zostaną umieszczone w specjalnej sekcji pliku *elf*, skąd będą mogły zostać odczytane przez program sterujący programatorem i użyte do konfiguracji procesora. Plik <*avr/fuse.h*> zawiera definicje przydatne przy niskopoziomowym manipulowaniu *fusebitami*. Plik nagłówkowy <*avr/io.h*> zawiera definicje *fusebitów* używanych przez wybrany model procesora. W zależności od typu procesora *fusebity* mieszczą się w jednym, dwóch lub trzech bajtach. Ilość bajtów przeznaczonych na *fusebity* zwraca makro FUSE_MEMORY_SIZE. Definiowanie *fusebitów* następuje poprzez przypisanie wartości makru FUSES, np.:

```
FUSES =
 {
  .low=LFUSE_DEFAULT,
  .high=(FUSE_BOOTSZ0 & FUSE_BOOTSZ1 & FUSE_EESAVE & FUSE_SPIEN),
  .extended=EFUSE_DEFAULT,
 };
```

Jak widzimy, makro to ma trzy pola odpowiadające poszczególnym bajtom przechowującym *fusebity*. Dla procesorów, w których makro FUSE_MEMORY_SIZE zwraca wartość jeden, dostępne jest tylko pole .low, dla procesorów, w których powyższe makro zwraca 2, dostępne są pola .low i .high, w pozostałych procesorach dostępne jest także pole .extended. Niestety, wykorzystując powyższe definicje, należy pamiętać, w którym bajcie jakie *fusebity* są przechowywane. Ich pomylenie spowoduje nieprawidłowe zaprogramowanie procesora, co może się nawet skończyć jego zablokowaniem. Podobnie jak w przypadku *lockbitów*, ważne jest tylko pierwsze przypisanie wartości do makra, kolejne są ignorowane. Stąd też najlepiej taką definicję umieścić raz, na początku programu.

Podobnie jak w przypadku *lockbitów*, wyliczone wartości *fusebitów* można odczytać z pliku *elf*:

```
avr-objdump -s -j .fuse test.elf

ADCNoiseReduction.elf:     file format elf32-avr

Contents of section .fuse:
 820000 62d1f9                             b..
```

Wyświetlona wartość odpowiada poszczególnym bajtom przechowującym *fusebity*. Dla procesorów, w których mieszczą się one w mniejszej ilości bajtów, pokazanych zostanie ich mniej. **Co ważne, najmłodszy bajt wyświetlany jest z lewej, najstarszy z prawej strony.**

Skorowidz

D

E

F

G

H

I

K

R

S

T

U

V

W

Z

JĘZYK C

DLA MIKROKONTROLERÓW AVR

Mikrokontrolery AVR firmy Atmel stanowią dynamicznie rozwijającą się rodzinę układów. Dzięki niskiej cenie, dużym możliwościom i dostępności darmowych narzędzi od lat niezmiennie cieszą się dużą popularnością wśród hobbystów i osób profesjonalnie zajmujących się programowaniem mikrokontrolerów.

Pewnym utrudnieniem dla polskich użytkowników AVR jest brak literatury na temat wykorzystania do ich programowania języków wysokiego poziomu, takich jak C. Niniejsza książka jest próbą wypełnienia tej luki. W sposób syntetyczny pokazuje różnice pomiędzy programowaniem w języku C komputerów klasy PC i mikrokontrolerów. Omawia programowanie peryferii dostępnych w mikrokontrolerach AVR w języku C, bibliotekę standardową oraz jej rozszerzenia znane jako AVR-libc. Dzięki temu nawet osoby w niewielkim stopniu znające podstawy języka C będą mogły bez problemów „przesiąść się" na programowanie mikrokontrolerów AVR. Z drugiej strony książka opisuje zaawansowane techniki programowania, związane z obsługą bootloadera, zabezpieczaniem i szyfrowaniem kodu aplikacji oraz realizacją najpowszechniej stosowanych protokołów wymiany danych pomiędzy urządzeniami opartymi na mikrokontrolerach i komputerami PC. Porusza także tematy związane ze specyfiką pisania aplikacji na mikrokontrolery oraz wyszukiwaniem i usuwaniem błędów.

- Podstawy programowania mikrokontrolerów AVR
- Warsztat pracy programisty AVR
- Wprowadzenie do języka C na AVR
- Budowa programu i jego części składowe
- Zastosowania przetwornika ADC
- Korzystanie z zasobów sprzętowych mikrokontrolera
- Używanie rejestrów i różnych rodzajów pamięci
- Obsługa wyświetlaczy LCD
- Korzystanie z interfejsów
- Zapewnianie bezpieczeństwa kodu

Programowanie mikrokontrolerów jeszcze nigdy nie było tak proste!

Cena: 89,00 zł

ISBN 978-83-246-3064-6

9 788324 630646