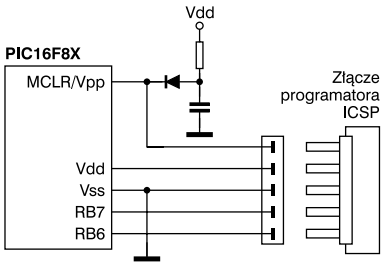
Bootloader dla mikrokontrolerów PIC

Wraz z zastępowaniem w mikrokontrolerach jednoukładowych pamięci typu ROM czy EPROM nowszymi pamięciami typu Flash. proces ich programowania stawał się bardziej przyjazny dla użytkownika. Nieocenioną zaletą okazała się możliwość wielokrotnego programowania i kasowania tych pamięci w sposób elektryczny. Postępująca miniaturyzacja wymusiła także zastąpienie programowania równoległego przez programowanie szeregowe.

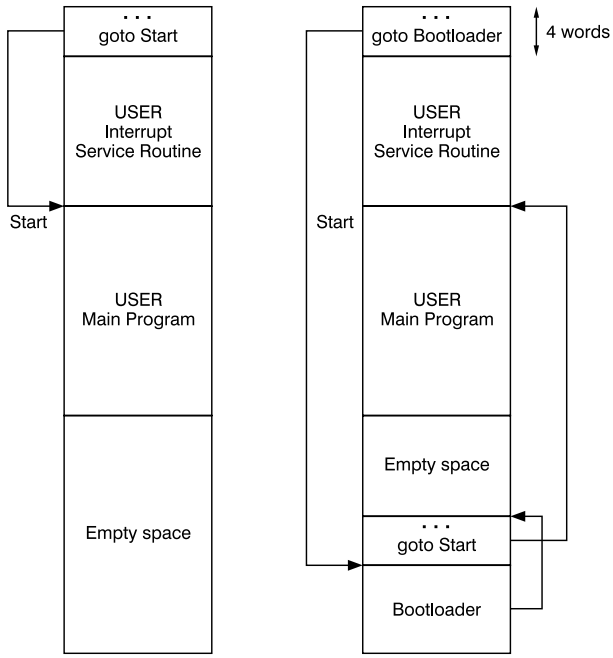
W chwili obecnej niemalże każdy mikrokontroler z wewnętrzną pamięcią programu przystosowany jest do programowania przez interfejs szeregowy, co daje możliwość programowania w systemie. W przypadku układów umieszczonych w obudowach SMD jest to praktycznie jedyna metoda ich wielokrotnego programowania, którego nie możemy uniknąć podczas tworzenia programu.

W mikrokontrolerach firmy Microchip stosowany jest interfejs składający się z linii sygnału zegarowego oraz dwukierunkowej linii danych. Oprócz tego w procesie programowania jest wymagany sygnał zerowania, który służy jednocześnie do podania napięcia programującego VPP) o wartości około 13 V. Pomimo tego, że do programowania są potrzebne tylko trzy linie, to konieczność podania wysokiego napięcia może być kłopotliwa. Ponieważ typowe wartości napięć na wejściu zerowania !MCLR mieszczą się w zakresie 0...VCC (+5 V), to chcąc programować układ w systemie należy zmodyfikować obwód zerowania. Przykładowy schemat dołączenia sygnałów programatora jest przedstawiony na **rys. 1**. Jak widać, w przypadku stosowania zewnętrznych specjalizowanych układów zerowania systemu (ich wyjścia najczęściej dostosowane są do pracy przy napięciu maksymalnym 5 V), konieczna jest także rozbudowa bloku zerowania.

Niektóre rodziny układów umożliwiają programowanie bez użycia napięcia programującego VPP, jednak w procesie programowania konieczne jest wykorzystanie dodatkowego wyprowadzenia PGM. Układy mogą być wtedy programowane przy napięciu równym napięciu zasilania, jednak ograniczona jest wtedy funkcjonalność wyprowadzenia PGM. Przypomnijmy, że jest to standardowa linia wejścia/wyjścia uzupełniona o dodatkową funkcję wejścia sygnału PGM.

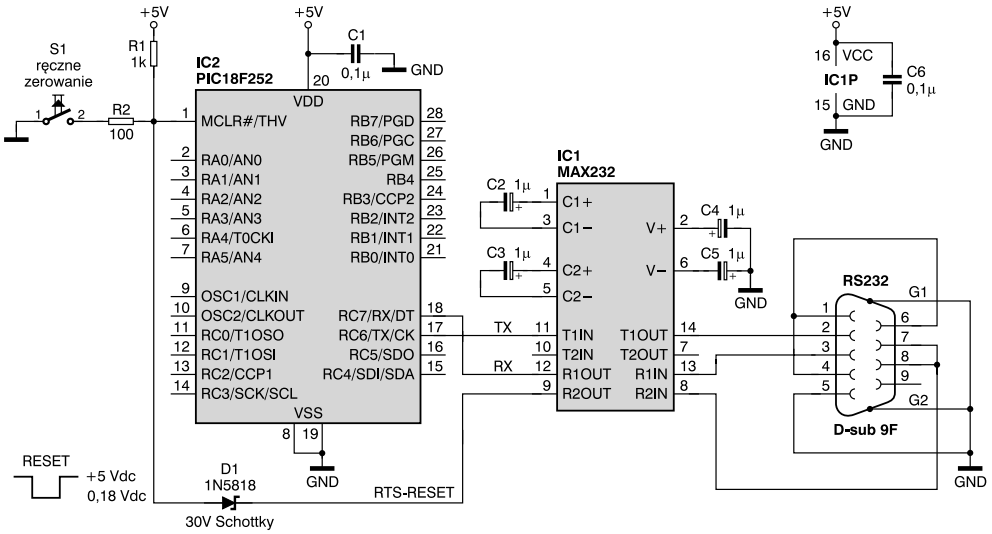


Rys. 1. Sposób dołączenia programatora do układu pracującego w systemie

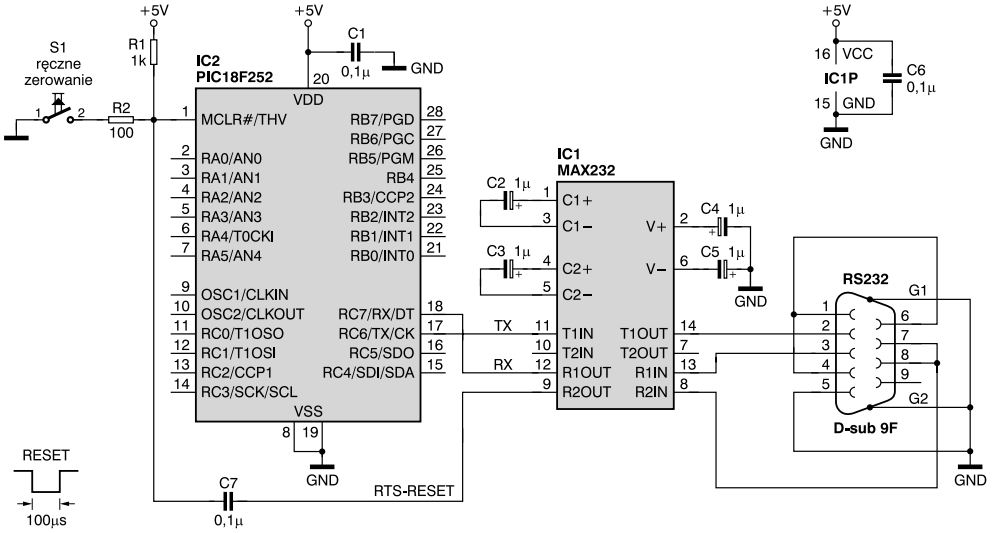


Rys. 2. Mapa pamięci programu

W pewnych przypadkach proces programowania można znacznie uprościć stosując tak zwany *bootloader*. Jest on szczególnie przydatny, gdy urządzenie z założenia jest wyposażone w port szeregowy umożliwiający komunikację na przykład z komputerem. Wtedy do programowania mikrokontrolera nie będzie konieczny programator, dodatkowo zbędne będą jakiekolwiek dodatkowe połączenia, gdyż proces programowania będzie wykonywany przez port szeregowy. Czym jest ów *bootloader*? Niczym innym jak oprogramowaniem umieszczonym w pamięci mikrokontrolera. Jest to swoisty system operacyjny umożliwiający start mikrokontrolera, podobnie jak to ma miejsce w komputerach. System ten umożliwia obsługę portu szeregowego oraz potrafi zapisać dane do pamięci programu typu Flash. Aby *bootloader* mógł być zastosowany, to dany mikrokontroler musi posiadać funkcję zapisu danych do pamięci programu. Ponadto, jak wspomniano wcześniej do wgrywania programu nie jest potrzebny programator. Będzie on jednak niezbędny do wgrania pliku *bootloadera*. Czynność ta jest wykonywana jednorazowo, a późniejsze aktualizacje programu użytkownika mogą być wykonywane przez port szeregowy. Oprócz zalet takiego programowania w czasie tworzenia programu, bardzo istotna jest możliwość wykonania zdalnej aktualizacji programu mikrokontrolera, gdy powstanie nowa jego wersja. Wtedy do aktualizacji nie jest konieczny programator, a tylko aplikacja umożliwiająca wgrywanie oprogramowania. Dodatkowo sam proces programowania jest prostszy i nie wymaga wiedzy na temat sposobu programowania mikrokontrolerów. Podob-



Rys. 3. Schemat połączenia mikrokontrolera z komputerem (zerowanie poziomem)



Rys. 4. Schemat połączenia mikrokontrolera z komputerem (zerowanie impulsem)

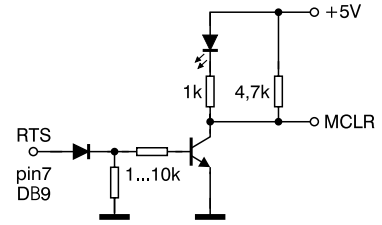
nie jak to ma miejsce, na przykład w urządzeniach sieciowych (*Router*) użytkownik końcowy ma możliwość samodzielnej aktualizacji oprogramowania sterującego urządzeniem.

Działanie *bootloadera* przedstawiono na **rys. 2**. Po lewej stronie widać typową mapę pamięci mikrokontrolerów firmy Microchip. Na samym początku znajdują się cztery słowa pamięci programu, które są wykorzystywane do umieszczenia instrukcji skoku do właściwego programu. Następny obszar (*Interrupt Service Routine*) jest wykorzystywany do obsługi przerwań. W kolejnym obszarze znajduje się właściwy program użytkownika. Po kompilacji, w pliku wynikowym (hex) na jego początku jest umieszczana instrukcja skoku do etykiety „Start" skąd następuje wykonanie programu głównego. Programując procesor *bootloaderem* adres skoku do programu użytkownika zostaje zmodyfikowany i po uruchomieniu procesora wykonywany jest program *bootloadera* (prawa strona rys. 2). Zadaniem programu *bootloadera* jest wykrycie po uruchomieniu procesora, czy ma być wykonywany zapisany program użytkownika, czy będzie wgrywany nowy. W zależności od rodzaju *bootloadera* najczęściej stosuje się zmianę stanu na dodatkowym wejściu procesora lub oczekiwanie na znacznik z portu szeregowego. Może to być pojedynczy znak zachęty lub ciąg znaków, za wysłanie których odpowiada aplikacja w komputerze obsługująca dany port szeregowy. Jeśli w ustalonym czasie (zazwyczaj do kilkuset milisekund) połączenie nie zostanie nawiązane, to następuje skok do adresu oznaczonego „goto Start". Pod tym adresem znajduje się informacja, od którego miejsca rozpoczyna się program użytkownika – jeżeli wcześniej był wgrywany, w przeciwnym przypadku instrukcja pętli nieskończonej. Jeśli program użytkownika jest wgrany, to po włączeniu zasilania i bezskutecznej próbie nawiązania połączenia z aplikacją mikrokontroler przejdzie do wykonywania właściwego programu. Taka sytuacja będzie występowała przy każdym uruchomieniu procesora, co będzie powodowało start programu użytkownika z opóźnieniem. Opóźnienie to uzależnione jest od konkretnego *bootloadera*. Jeśli ważne jest, aby procesor startował szybko, to należy zastosować *bootloader* ładujący program po ustawieniu stanu na odpowiednim wyprowadzeniu procesora. Wtedy uruchamianie programu użytkownika będzie następowało z opóźnieniem równym zaledwie kilku cyklom procesora (potrzebnym na sprawdzenie stanu wyprowadzenia portu).

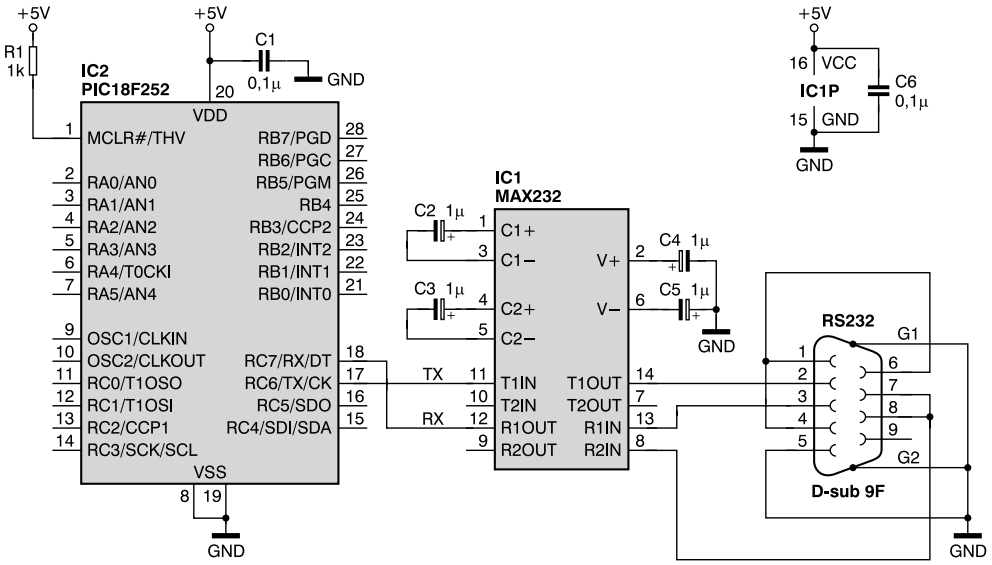
Który wybrać?

Przeszukując strony Internetowe można natknąć się na różnego rodzaju *bootloadery* różniące się od siebie sposobem komunikacji z komputerem oraz pełnionymi funkcjami dodatkowymi. Ponieważ *bootloader* jest programem, który zajmuje pamięć programu procesora istotne jest także, aby on sam nie pochłaniał jej zbyt dużo. Nieekonomiczna byłaby konieczność stosowania układu o większej pamięci tylko ze względu na użycie *bootloadera*.

Drugim ważnym parametrem jest sposób ładowania programu i co jest z tym powiązane koszt aplikacji sterującej. W najprostszym przypadku można zastosować program terminalowy HyperTerminal, w który jest wyposażony system Windows.



Rys. 5. Układ automatycznego zerowania procesora z sygnalizacją



Rys. 6. Schemat połączenia procesora z komputerem bez sygnału automatycznego zerowania

Jednak nie jest on zbyt przyjazny w obsłudze, a dodatkowo program *bootloadera* musi być rozbudowany, gdyż pełni funkcję sterowania przepływem danych, a także zajmuje się kontrolą poprawności zapisanych danych. Dedykowana aplikacja może natomiast przejąć niemalże wszystkie funkcje kontrolne, a program *bootloadera* będzie służył jedynie do ich zapisania oraz odczytania. Zestawienie wraz z podstawowymi właściwościami znalezionych w Internecie *bootloaderów* przedstawiono w **tab. 1** (tabela zaczerpnięta ze strony *http://www.etc.ugal.ro/cchiculita/software/picbootloader.htm*).

Tab. 1. Porównanie różnych bootloader'ów

Nazwa bootloadera/ autor	Obsługiwane układy	Rozmiar (słowo)
Microchip	16F, 18F	1000
MicrochipC	16F, 16F*A, 18F	256/2000
Wloader Wouter van Ooijen	16f877	1000
ZPL Wouter van Ooijen	18F	384
KarlLunt	16f87x	512
PICLOADER Rick Farmer	PIC16F87x	2000
bootload	PIC16F877	800
theByteFactory	16F877	1000
Jolt Martin Dubuc	18F	256
HI–TECH Software	16F87x	256
PIC downloader Petr Kolomaznik	16F876	256
Ivar Johnsrud	18Fxx2/ 18Fxx8	360
B Bootloader	PIC16F87x, PIC16F87xA	340
SGupta	16f876	256
Tiny	16F, 18F, dsPIC	100

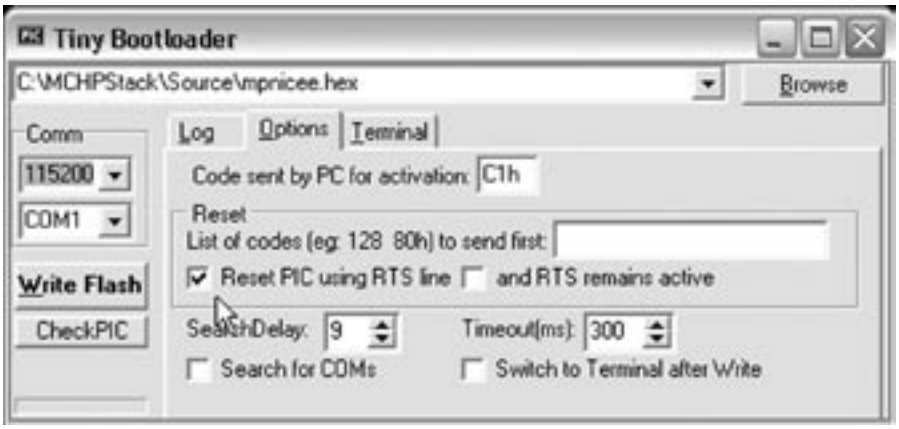
W poszukiwaniach tego „doskonałego" bootloadera na szczególną uwagę zasługuje „Tiny PIC bootloader", który jest udostępniany przez autora na stronie *http://www.etc.ugal.ro/cchiculita/software/picbootloader.htm*. Wyróżnia się on najmniejszą objętością kodu wynikowego (100 słów pamięci programu), a dodatkowo umożliwia współpracę z układami z rodzin 16F i 18F, a także dsPIC. Wykaz obsługiwanych procesorów jest przedstawiony w **tab. 2**. Ponieważ oprogramowanie jest ciągle rozwijane i wzbogacane o nowe układy, to wszystkie układy są podzielone na trzy grupy. Określają one stopień poprawności współpracy z *bootloaderem*. Stopień ten jest uzależniony od okresu testowania oraz opinii osób stosujących *bootloader*. Brak precyzji określającej, z którymi układami poprawnie współpracuje *bootloader* wynika z faktu, że program przeznaczony dla jednego typu mikrokontrolera może współpracować prawidłowo także z innymi układami. Dlatego w tab. 2 przedstawione są układy, z którymi zostały przeprowadzone próby.

Strona sprzętowa

Do wgrania nowego programu do procesora poprzez *bootloader* niezbędne jest połączenie sprzętowego sterownika portu szeregowego procesora (UART) z portem szeregowym komputera. Jeśli układ, w którym jest zainstalowany procesor posiada taki port, to można go wykorzystać do aktualizacji oprogramowania. Jeśli jednak urządzenie nie posiada połączenia przez port szeregowy, to należy wykonać zewnętrzny konwerter napięć zamieniający sygnały z portu RS232 na kompatybilne ze standardem TTL. Podstawowym układem realizującym taką funkcję jest układ MAX232. Na stronie autora można znaleźć kilka przykładowych schematów przedstawiających sposób dołączenia go do procesora. Na **rys. 3 i 4** przedstawiono schematy realizujące także funkcję automatycznego zerowania procesora przed wgrywaniem programu. W pierwszym przypadku dane są przesyłane liniami RX i TX. Sygnał zerowania pochodzi od sygnału RTS portu szeregowego i poprzez diodę D1 jest kierowany do wejścia zerującego !MCLR. Takie podłączenie pozwala także na dołączenie zewnętrznego przycisku zerującego. Jeśli nie ma potrzeby jego stosowa-

Tab. 2. Wykaz układów współpracujących z Tiny Bootloader

Rodzina PIC16F	
Układ	**Układ tego samego typu**
PIC16F876	877, 873, 874
PIC16F876A	**877A**, 873A, 874A
PIC16F88	87
Rodzina PIC18F	
Układ	**Układ tego samego typu**
PIC18F252	*452*, 242, 442, *2420*, *2520*, 4420, *4520*
PIC18F258	458, 248, 448, 2480, 2580, 4480, 4580
PIC18F2620	*4620*, *2525*, *4525*
PIC18F1320	1220, *2220*, *2320*
PIC18F8720	6520, 8520, 6620, 8620, 6720; 6621
PIC18F2550	4550, 2455, 4455
PIC18F4431	2331, 2431, 4331
PIC18F4680	*2585*, *2680*, *4585*
PIC18F4580	2480, 2580, *4480*
PIC18F4620	2525, 2620, *4525*
PIC18F4320	2220, 2320, *4220*
Rodzina dsPIC	
Układ	**Układ tego samego typu**
dsPIC20F2010	
dsPIC30F6014	6012, 6013, 6011
dsPIC30F3013	3012, 2012, 2011
dsPIC30F4012	4011
pogrubienie – testowane długoterminowo, podkreślenie – sprawdzony przez autora, kursywa – sprawdzony przez użytkowników – być może potrzebna jest korekta programu	



Rys. 7. Okno aplikacji Tiny Bootloader

nia, to sygnał RTS można dołączyć do wejścia zerowania bezpośrednio (bez szeregowo włączonej diody D1). Stosując to rozwiązanie także do komunikacji procesora z komputerem w czasie jego normalnej pracy należy zapewnić, aby aplikacja służąca komunikacji ustawiała stan wysoki (–12 V) na wyjściu RTS. Po inwersji przez układ MAX232 na wejściu !MCLR będzie panował stan wysoki (TTL), w przeciwnym przypadku procesor będzie cały czas zerowany.

Jeśli nie ma możliwości zmiany ustawień parametrów konfiguracyjnych portu RS232 istniejącej aplikacji, to można zastosować układ przedstawiony na **rys. 4**. Dotyczy to, na przykład programu Hyperterminal, który ustawia sygnał RTS w stan niski (+12 V), co na wyjściu układu MAX232 powoduje ustawienie logicznego zera (TTL), przez co procesor będzie permanentnie zerowany. Zgodnie z rys. 4 zamiast diody został zastosowany kondensator, poprzez który generowany jest krótki impuls zerowania tylko w chwili zmiany stanu na wyjściu RTS. Powoduje to generację chwilowego sygnału zerowania procesora i umożliwia jego późniejszą pracę.

Sygnał zerowania „zajmuje” jeden stopień konwertera zawartego w układzie MAX232 i jeśli w urządzeniu procesor wykorzystuje wszystkie stopnie (na przykład obsługuje dwa porty szeregowe), to można zastosować dodatkowy obwód zerowania przedstawiony na **rys. 5**. W układzie tym do zamiany poziomów sygnałów został zastosowany tranzystor. Opcjonalnie obwód ten można wyposażyć w układ sygnalizacji zapisu poprzez dołączenie diody świecącej.

Rys. 8. Ikona programu w zasobniku systemowym

Zastosowanie sygnału zerowania ułatwia proces wgrywania programu, powoduje jednak konieczność dołączenia dodatkowej linii. Może to doprowadzić do konieczności modyfikacji istniejącej płytki urządzenia. Z sygnału zerowania można jednak zrezygnować. Wprawdzie proces wgrywania programu nie będzie przebiegał automatycznie, ale do programowania potrzebne będą tylko linie: RX i TX. Schemat połączeń dla układu pozbawionego automatycznego zerowania jest przedstawiony na **rys. 6**. W tym przypadku proces wgrywania programu należy poprzedzić ręcznym zerowaniem procesora, za pomocą przedstawionego w poprzednich schematach przycisku lub poprzez wyłączenie i włączenie zasilania. W niektórych przypadkach wyłączenie zasilania może okazać się jedynym sposobem zerowania procesora. Wynika to z faktu, że w niektórych mikrokontrolerach wejście zerujące !MCLR może pełnić rolę typowej linii wejściowej, a sygnał zerowania jest generowany przez wewnętrzne moduły procesora. W takim przypadku nie jest możliwe zerowanie procesora zmianą stanu na wejściu !MCLR.

Strona programowa

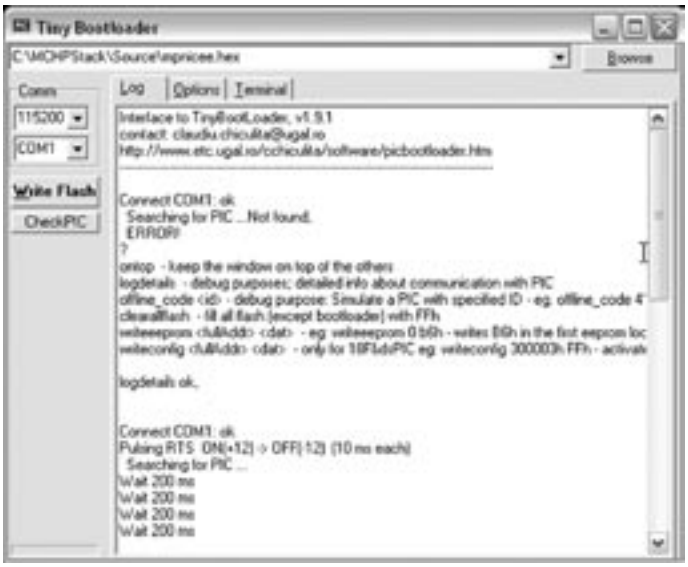
Do wgrywania programu służy aplikacja, której główne okno jest przedstawione na **rys. 7**. Przed rozpoczęciem programowania układu należy wybrać port szeregowy, do którego jest dołączony procesor i prędkość z jaką się komunikuje z aplikacją. Prędkość ta jest zależna od programu *bootloadera* umieszczonego w procesorze. Jeśli będzie wykorzystywany sygnał automatycznego zerowania procesora, to w zakładce „Options” należy zaznaczyć opcję *Reset PIC using RTS line*. Zmieniając parametr *SearchDelay* można ustawić czas oczekiwania aplikacji na zgłoszenie się procesora. Opcja ta jest szczególnie przydatna przy ręcznym zerowaniu procesora, gdyż ustawienie odpowiednio długiego czasu pozwoli na wyzerowanie procesora bez pośpiechu. Wartość przedstawiona na rys. 7 ustawia czas oczekiwania na około 10 sekund. Po ustawieniu tych opcji i podłączeniu procesora do wybranego portu szeregowego można sprawdzić komunikację naciskając przycisk „CheckPIC”. Jeśli zerowanie procesora jest wykonywane automatycznie, to należy tylko nacisnąć ten

Tab. 3. Znaczenie komend wpisywanych w oknie Log

Polecenie	Pomoc
ontop	Ustawienie atrybutu okna „zawsze na wierzchu”
logdetails	Wyświetlanie szczegółów procesu programowania
clearallflash	Kasowanie pamięci za wyjątkiem obszaru bootloadera (wpis 0xFFh)
writeeeprom <addr> <dat>	Zapisanie bajtu <dat> pod podany adres <addr.> pamięci EEPROM Np.: writeeeprom 0 B6h – zapisanie wartości B6h pod adresem 0
writeconfig <adrLow> <dat>	Tylko dla rodziny 18F; zapisanie bajtu konfiguracji <dat> pod adresem 300000h+<addrLow> Np.: writeconfig 3 FFh – Włączenie WDT

List. 1. Fragment programu bootloadera

```
/************************************************************************/
  radix DEC
  LIST      P=18F252  ; change also: Configure->SelectDevice from Mplab
xtal EQU 20000000   ; you may want to change: _XT_OSC_1H   _HS_OSC_1H  ;_HSPLL_OSC_1H
baud EQU 115200     ; standard TinyBld baud rates: 115200 or 19200
; The above 3 lines can be changed and buid a bootloader for the ;desired frequency (and PIC type)
/************************************************************************/
```



Rys. 9. Komendy wydawane z linii poleceń

przycisk. Przy ręcznym zerowaniu, po naciśnięciu przycisku należy w czasie kilku sekund wyzerować procesor. W oknie aplikacji będzie wyświetlany wykres słupkowy obrazujący upływający czas. Przechodząc do okna „Log" można odczytać komunikaty informujące o przebiegu danej operacji. Uruchomienie procesu programowania można wykonać w dwojaki sposób. W pierwszym przypadku po wczytaniu pliku *.hex zawierającego właściwy program należy nacisnąć przycisk „Write Flash" w oknie programu. Programowanie można także wykonać bez dostępu do okna programu. Wraz z uruchomieniem programu pojawia się jego ikona w zasobniku systemowym obok zegara (**rys. 8**). Kliknięcie prawym przyciskiem myszki powoduje ukrycie głównego okna aplikacji, natomiast dwukrotne kliknięcie lewym klawiszem uruchamia proces programowania z wcześniej ustawionymi parametrami oraz wskazanym plikiem programu procesora. W trakcie programowania i po jego zakończeniu ikona zmienia swoją postać informując o przebiegu procesu programowania. Funkcja ta jest szczególnie przydatna przy tworzeniu programu dla procesora, gdyż po każdej kompilacji programu jego zapisanie w pamięci procesora wymaga jedynie dwukrotnego kliknięcia na ikonę.

Oprócz poleceń wydawanych naciśnięciem przycisków, możliwe jest wpisanie komend w oknie „Log". Wykaz dostępnych komend zostanie wyświetlony po wpisaniu znaku zapytania, a ich znaczenie jest przedstawione w **tab. 3**. Na **rys. 9** przedstawiono przykład wyświetlenia opisu pomocy oraz działania funkcji *„logdetails"*. Przy pierwszej próbie połączenia z procesorem wyświetlone są tylko komunikaty informujące zakończeniu danej operacji. Natomiast po wydaniu polecenia *„logdetails"* wyświetlane są informacje o poszczególnych krokach operacji.

Oprócz programatora aplikacja zawiera także terminal umożliwiający komunikację z procesorem przez port szeregowy. Znajduje się on w zakładce „Terminal" i pozwala na bezpośrednią komunikację bez potrzeby uruchamiania dodatkowego programu.

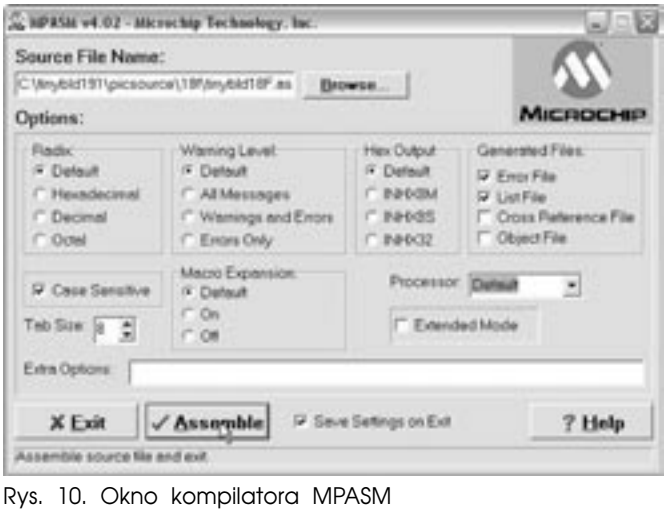
Stosowanie *bootloadera*

Jak wcześniej wspomniano użytkowanie *bootloadera* wymaga jednokrotnego zaprogramowania procesora typowym programatorem. Plik, którym należy zaprogramować procesor zależy od rodzaju procesora, częstotliwości jego pracy i prędkości transmisji szeregowej. W pakiecie, który znajduje się na stronie autora (*http://www.etc.ugal.ro/cchiculita/software/tinybld191.zip*) znajduje się kilka plików wynikowych skompilowanych dla kilku procesorów i typowych częstotliwości ich pracy. W **tab. 4** znajduje się ich wykaz. Jeśli typ procesora i częstotliwość jego pracy odpowiada zastosowanej w użytej aplikacji, to wystarczy zaprogramować procesor odpowiednim plikiem. Plik wynikowy można także dostosować do własnych potrzeb, zmieniając częstotliwość pracy lub prędkość transmisji. Wymaga to jednak kompilacji udostępnionych plików źródłowych. Ponieważ programy są napisane w asemblerze, to wystarczy do tego darmowy kompilator MPASM udostępniany przez firmę Microchip. Kompilator ten znajduje się w pakiecie MPLAB IDE, który można pobrać ze strony *www.microchip.com*. Przedstawiony na **list. 1** fragment programu wskazuje miejsce, w którym można dokonać zmian parametrów kompilacji. W celu zmiany typu procesora należy zmienić deklarację „LIST P=18F252". Częstotliwość taktowania procesora zmieniana jest parametrem „xtal EQU 20000000", a prędkość transmisji portu szeregowego „baud EQU 115200". Po zmianie tych parametrów należy przeprowadzić kompilację uruchamiając kompilator i wczytując modyfikowany plik, a następnie kompilując poleceniem „Assemble" (**rys. 10**). Tak zmodyfikowanym plikiem wynikowym (hex) należy zaprogramować mikrokontroler. Jeśli została zmieniona prędkość transmisji portu szeregowego, to należy ją także zmodyfikować w programie „Tiny Bootloader".

Tab. 4. Wykaz dostępnych plików wynikowych (hex)

Typ procesora	Częstotliwość pracy [MHz]	Prędkość transmisji [bps]
PIC16F876A	20	115200
PIC16F88	8	19200
PIC16F88	20	115200
PIC18F252	20	115200
PIC18F258	20	115200
PIC18F2550	20	115200
dsPIC30F2010	20	19200
dsPIC30F3013	7,37	115200
dsPIC30F4012	20	19200
dsPIC30F6014	7,37	115200

Krzysztof Pławsiuk, EP
krzysztof.plawsiuk@ep.com.pl



Rys. 10. Okno kompilatora MPASM