Kurs programowania mikrokontrolerów PIC18

Informacje wstępne

Dodatkowe materiały na CD i FTP:
ftp://ep.com.pl, user: 10460, pass: 0646g3n0

Kurs programowania mikrokontrolerów PIC rozpoczynamy od opisu narzędzi programowych, sposobu integracji środowiska MPLAB z kompilatorem Hi-Tech oraz prostych programów przykładowych. Podajemy też istotne informacje na temat konfigurowania mikrokontrolera, w tym roli poszczególnych bitów, i wyboru źródła sygnału zegarowego.

MPLAB i kompilator Hi-Tech

MPLAB IDE jest zintegrowanym środowiskiem projektowym umożliwiającym pisanie, kompilowanie i uruchamianie programów dla mikrokontrolerów PIC. Jest to też aplikacja sterująca wszystkimi firmowymi programatorami debuggerami i emulatorami sprzętowymi. Jeżeli chcemy wykorzystywać te narzędzia, to poza nielicznymi wyjątkami jesteśmy niejako „skazani" na MPLAB. Ponieważ jest ono rozwijane przez wiele lat, to ma opinię dopracowanego i wygodnego. Co ważne, jest dostępne bezpłatnie. Można je pobrać ze strony www.microchip.com.

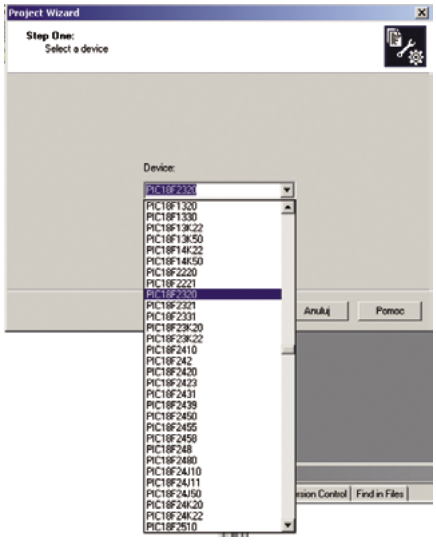
MPLAB ma spore możliwości, a opisanie wszystkich znacznie wykracza poza ramy kursu. Skupimy się tylko na tych niezbędnych do utworzenia programu, jak tworzenie projektu i środowiska pracy, dodawanie do niego plików, wybór kompilatora, ustawienie opcji kompilacji i wybór programatora.

Wszystkie pliki źródłowe używane w czasie pracy oraz ścieżki dostępu do nich, rodzaj kompilatora i jego ustawienia, rodzaj programatora itp. są zapisywane w pliku projektu. Po każdym uruchomieniu MPLAB-a można sobie otworzyć wcześniej zapisany projekt i rozpocząć pracę od momentu, w którym ostatnio nad nim pracowaliśmy.

Nowy projekt można utworzyć z menu *Project* na dwa sposoby: używając kreatora *Project Wizard* lub polecenia *New*. Przed tworzeniem projektu musimy znać typ mikrokontrolera, mieć zainstalowany kompilator i utworzyć sobie katalog, w którym będzie zapisany projekt i pliki źródłowe. Pliki źródłowe nie muszą być w katalogu projektu – można je umieszczać w dowolnych lokalizacjach.

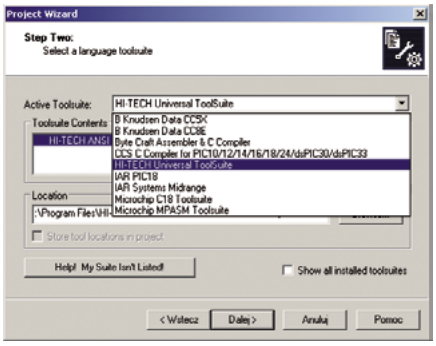
Na potrzeby kursu wybrałem mikrokontroler PIC18F2320 i bezpłatny kompilator języka C firmy Hi-Tech dla rodziny PIC 18F w wersji Lite. Przykładowy projekt utworzymy, korzystając z kreatora *Project Wizard*. Po uruchomieniu kreator wymaga wybrania z listy typu mikrokontrolera (**rysunek 1**). Następnie wybieramy język programowania i kompilator. MPLAB-IDE pozwala na pracę z wieloma kompilatorami asemblera lub języków wyższego poziomu. Kompilatory kompatybilne z MPLAB mają swoje programy narzędziowe (kompilator, linker, asembler itp.) pogrupowane w tak zwany *Toolsuite*. W oknie *Active toolsuite* wybieramy Hi-Tech, a w oknie *Location* ścieżkę dostępu do pliku kompilatora (**rysunek 2**). W kolejnym kroku tworzymy plik projektu, wybierając lokalizację i wpisując nazwę projektu (**rysunek 3**). Jeżeli już mamy pliki źródłowe, to można je dodać w następnym kroku działania kreatora. Ja wcześniej na potrzeby tego przykładu utworzyłem pusty plik *main.c* w katalogu projektu i mogłem go dodać, jednak dodawanie plików można wykonać w dowolnym momencie pracy nad projektem.

Po dodaniu plików źródłowych kreator kończy działanie, zapisuje plik projektu o wybranej nazwie (tutaj PicTest) i rozszerzeniu .mcp. Pojawia się okno, w którym jest umieszczane drzewo projektu z katalogami plików źródłowych (*Source Files*), nagłówkowych (*Header Files*), wynikowych (*Object Files*) i bibliotecznych (*Library Files*).

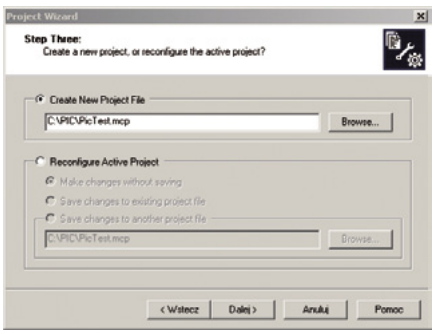


Rysunek 1. Wybór mikrokontrolera

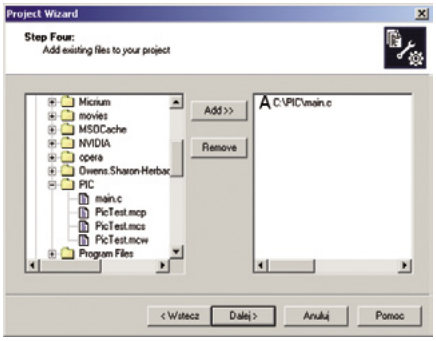
Warto wspomnieć o jeszcze jednym elemencie organizacji pracy w MPLAB-ie, a mianowicie o przestrzeni roboczej (*Workspace*). Kreator *Project Wizard* przy tworzeniu projektu automatycznie tworzy plik *Workspace* o takiej samej nazwie,



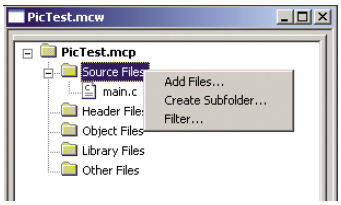
Rysunek 2. Wybór kompilatora



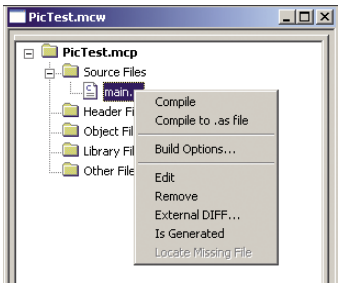
Rysunek 3. Utworzenie pliku projektu



Rysunek 4. Dodawanie plików źródłowych w kreatorze Project Wizard



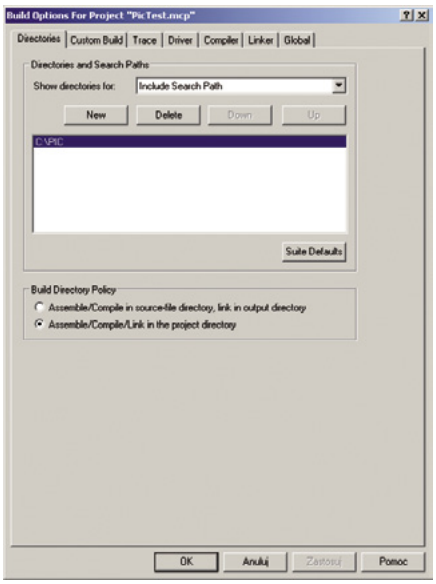
Rysunek 5. Dodawanie nowego pliku źródłowego



Rysunek 6. Menu operacji na pliku projektu

jak plik projektu i o rozszerzeniu .mcw. W przestrzeni roboczej można otworzyć więcej niż jeden projekt i w prosty sposób przełączać jeden z nich jako aktywny (*Project -> Set Active Project*). Nazwę przestrzeni roboczej można zmieniać z poziomu menu *File -> Save Workspace As*.

We wcześniej utworzonym projekcie pliki źródłowe dodaje się po kliknięciu prawym klawiszem na folderze *Source Files*. Pojawia się wtedy krótkie menu z możliwością dodania pliku, dodania podkatalogu oraz stosowania filtra (**rysunek 5**). Dla nas jest interesująca pierwsza pozycja – *Add Files*. Po jej wybraniu otwiera się okno wyboru pliku. W związku z tym, że system Windows wyświetla menu okno z opisami w języku angielskim, a przyciski standardowe z opisami w języku polskim (np. *Otwórz*, *Anuluj*), po wskazaniu pliku akceptuje się jego wybór przez naciśnięcie *Otwórz*. Inne operacje na plikach projektu, jak usunięcie, edycja i kompilacja, są dostępne z menu kontekstowego (**rysunek 6**)



Rysunek 7. Dodawanie ścieżki dostępu do plików nagłówkowych

wyświetlanego po naciśnięciu prawego klawisza myszy, gdy kursor jest w oknie projektu.

Utworzenie projektu i zapisanie przestrzeni roboczej nie kończy pracy nad projektem. W wielu przypadkach trzeba uzupełnić dodatkowe opcje projektu w oknie *Build Options* rozwijanym z menu *Project*. Zawartość tego okna zależy od wybranego *Toolsuite*, czyli od producenta kompilatora.

W przypadku kompilatora Hi-Tech na pewno będzie nas interesowała zakładka *Directories*. Można w niej ustawić ścieżkę dostępu do zapisywania pliku wynikowego *Oputput Directory* i ścieżki dostępu do plików nagłówkowych *Include Search Patch*. W przypadku ustawiania ścieżki dostępu do katalogu z plikiem wynikowym trzeba pamiętać, by zaznaczyć na dole okna opcję *Link In output Directory*. Domyślnie plik wynikowy jest zapisywany w katalogu projektu.

Dużo bardziej przydatna jest opcja ustawiania ścieżek dostępu dla plików nagłówkowych. Pliki nagłówkowe dołącza się instrukcją preprocesora *#include*. Jeżeli plik nagłówkowy jest umieszczony w domyślnej lokalizacji, to jest ujęty w ostre nawiasy np. *#include <htc.h>*. Dołączanie plików z innych lokalizacji wymaga podania pełnej ścieżki dostępu lub ścieżki w odniesieniu do katalogu projektu. Jeżeli plik nagłówkowy jest umieszczony w katalogu projektu, to jego nazwa jest ujęta w cudzysłów, np. *#include „test.h”*.

W trakcie dołączenia kompilatora do projektu automatycznie są ustawiane ścieżki dostępu do domyślnych lokalizacji. Jednak w projekcie używamy swoich plików nagłówkowych, wygodnie jest ustawić ścieżkę do tych plików i dołączać je tak jak przy lokalizacji domyślnej. W oknie *Show directories for* wybieramy *Include Search Path* i klikamy na przycisk *New*. W tym oknie dodajemy ścieżkę dostępu do katalogu z plikami nagłówkowymi. Można w ten sposób dodawać więcej niż jedną ścieżkę. Wszystkie pliki nagłówkowe ze zdefiniowanymi ścieżkami można dołączać przez *#include <nawa_pliku.h>*. W przypadku innych kompilatorów może się okazać, że projekt będzie wymagał dodania ścieżek dostępu do innych elementów projektu: plików bibliotecznych, plików skryptów kompilatora itp.

Kompilator Hi-Tech dopuszcza instalowanie i używanie różnych swoich wersji w różnych lokalizacjach. Na przykład użytkownik może mieć zainstalowane pełną wersję *PRO* i jednocześnie wersję *LITE*, ale mogą to też być kompilatory dla różnych rodzin mikrokontrolerów. Wybór interesującej wersji następuje w zakładce *Driver*. Przyciskami *Move Up* i *Move Down* przemieszcza się w nim zainstalowane wersje. Zawsze jest wykonywana wersja umieszczona najwyżej, ale pod warunkiem, że obsługuje wybrany mikrokontroler. Kompatybilność wersji kompilatora z mikrokontrolerem można sprawdzić w okienku zatytułowanym *Select driver Information and supported devices*.

Rysunek 8. Instalacja wersji Lite

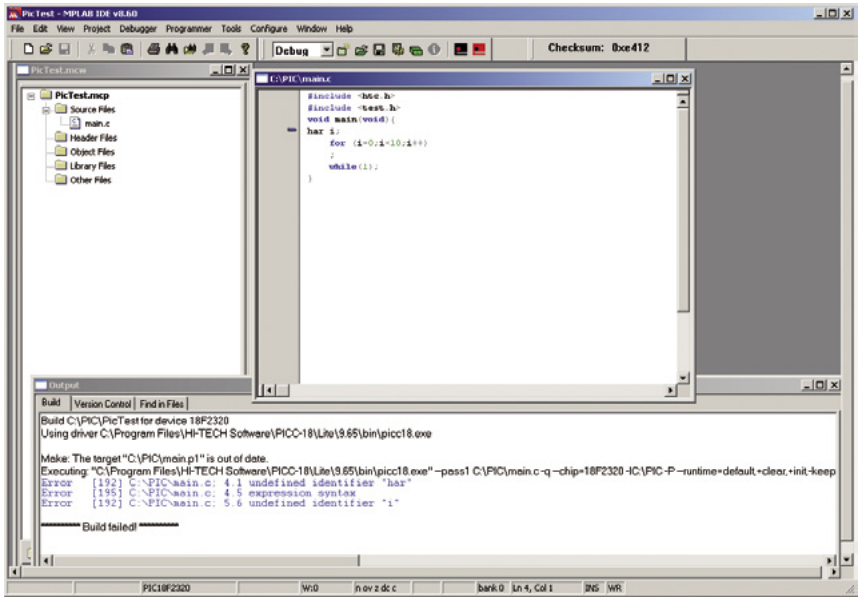
Bardziej zaawansowani użytkownicy mogą zmieniać ustawienia w zakładkach *Compiler*, *Linker* i *Global*, ale w naszych zastosowaniach wystarczają ustawienia domyślne. Na koniec możemy w okienku umieszczonym w pasku narzędziowym ustawić czy plik wynikowy ma być typu *Debug* czy *Release*. Opcję *Debug* wybiera się wtedy, gdy program będzie debugowany sprzętowo np. za pomocą debuggera ICD2, PICKIT2, PICKIT3 itp. Jeżeli plik wynikowy ma już końcową postać lub testy działania przeprowadzamy bez debuggera wgrywając go do pamięci mikrokontrolera, wybieramy opcję *Release*. Na potrzeby tego kursu należy wybierać *Release*.

Microchip dołącza do pakietu instalacyjnego MPLAB kompilator Hi-Tech Lite dla rodziny PIC16. Ponieważ w kursie używamy mikrokontrolera z rodziny PIC18, kompilator trzeba ściągnąć ze strony *www.microchip.com* po uprzednim zarejestrowaniu się. W czasie instalacji pobranego pliku wybieramy opcję *Operate In Lite Mode* (**rysunek 8**).

Po zapoznaniu się z nią należy zaakceptować warunki licencji. Kompilator zainstaluje się w domyślnej lokacji. Dokumentacja kompilatora Hi-Tech (jeżeli został wybrany w projekcie) jest dostępna z menu *Project -> HI-TECH C Manual* lub po naciśnięciu klawisza F11.

Wersja Lite kompilatora nie nakłada ograniczeń czasowych i rozmiaru kodu. Jest przeznaczona do zastosowań edukacyjnych i nie zawiera mechanizmów optymalizacji. Kod w porównaniu z najlepsza wersją PRO jest sporo większy, ale dla naszych celów nie ma to znaczenia.

Projekt jest kompilowany po naciśnięciu klawisza F10 lub czarnej ikony kompilacji na pasku narzędziowym. Wynik jest przedstawiany w oknie zatytułowanym *Output* w zakładce *Build*. Kompilator umieszcza tam informacje o zajętości wszystkich rodzajów pamięci przez plik wynikowy. Jeżeli kompilacja przebiegła bez błędów, pojawi się komunikat *Build Successful*. Błędy i ostrzeżenia są wyświetlane w kolorze niebieskim. Komunikat o błędzie zawiera nazwę pliku, numer wiersza w pliku, numer kolumny w wierszu oraz nazwę błędu. Wystarczy kliknąć 2 razy na linijkę z komunikatem, a edytor automatycznie ustawi wskaźnik edycji w pliku źródłowym w wierszu z błędem (**rysunek 9**).



Rysunek 9. Błędy kompilacji

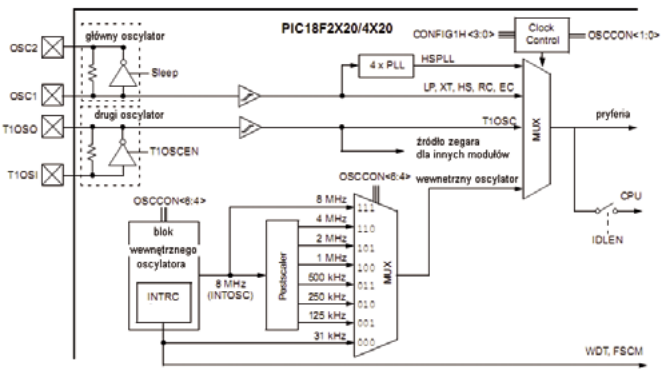
Taktowanie mikrokontrolera

Praca jednostki centralnej i wszystkich układów peryferyjnych mikrokontrolera jest synchronizowana sygnałem zegarowym, którego źródłem może być generator zewnętrzny (kwarcowy lub RC) albo wbudowany w strukturę mikrokontrolera kompletny oscylator tzn. bez konieczności stosowania elementów zewnętrznych RC (rysunek 10).

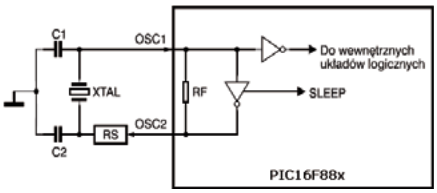
Układ generatora jest dość rozbudowany i pozwala na programowe wybieranie jednego z 10 trybów taktowania:

– LP – wbudowany generator kwarcowy z zewnętrznym kwarcem „zegarkowym" o częstotliwości 32 kHz,
– XT – wbudowany generator kwarcowy z kwarcami o średnich częstotliwościach (do 4 MHz),
– HS – wbudowany generator kwarcowy z kwarcami o najwyższych częstotliwościach (4...20 MHz),
– HSPLL – wbudowany generator kwarcowy z kwarcami o najwyższych częstotliwościach (4...10 MHz) z możliwością jej podwyższenia przez PLL,
- RC – wbudowany generator RC z zewnętrznymi elementami RC dołączanymi do wyprowadzenia OSC1/CLKIN i z wyjściem sygnału o częstotliwości $F_{OSC}/4$ na RA6,
- RCIO – wbudowany generator RC z zewnętrznymi elementami RC dołączanymi do wyprowadzenia OSC1/CLKIN; wyprowadzenie RA6 może być wykorzystywane jako linia I/O,
– INTIO1 – wbudowany generator RC bez zewnętrznych elementów RC; sygnał o częstotliwości $F_{OSC}/4$ jest dostępny na wyprowadzeniu RA6,
– INTIO2 – wbudowany generator RC bez zewnętrznych elementów RC; wyprowadzenia RA6 i RA7 mogą być wykorzystane jako linie portów IO,
– EC – taktowanie sygnałem zegarowym z zewnętrznego generatora podawanym na wejście OSC1/CLOCIN,
– ECIO – taktowanie sygnałem zegarowym z zewnętrznego generatora podawanym na wejście OSC1/CLOCIN i z linią RA6 jako port I/O.

Mikrokontrolery PIC18F mogą pracować w tak szerokim zakresie częstotliwości, że zaprojektowanie jednego oscylatora kwarcowego, który generowałby sygnał zegarowy o częstotliwości od pojedynczych kHz do dziesiątek MHz, jest praktycznie niemożliwe. W związku z tym Microchip zastosował programowany główny generator. Pozwoliło to na podzielenie zakresu częstotliwości sygnału zegarowego na podzakresy LP, XT i HS. Tryb LP (*Low Power*) jest przeznaczony do generowania niskich częstotliwości sygnału zegarowego, a co za tym idzie niskiego poboru mocy. W przypadku, kiedy potrzebna jest duża szybkość działania mikrokontrolera, wybierany jest tryb HS (*High Speed*) lub HSPLL. Dla średnich częstotliwości wybierany jest tryb XT.



Rysunek 10. Układ generatora taktującego mikrokontroler



Rysunek 11. Generator z oscylatorem kwarcowym lub ceramicznym

W trybach XT, LP lub HS kwarcowe lub ceramiczne rezonatory są dołączone do wyprowadzeń OSC1 i OSC2 (**rysunek 11**). Generatory wbudowane w mikrokontrolery PIC wymagają rezonatorów o rezonansie równoległym. Oprócz właśnie opisanego głównego oscylatora kwarcowego mikrokontroler ma wbudowany drugi oscylator dla małych częstotliwości, który również można wykorzystać do taktowania mikrokontrolera.

Zastosowanie oscylatora z zewnętrznymi elementami RC jako źródła sygnału zegarowego z jednej strony jest rozwiązaniem najtańszym, z drugiej jednak strony charakteryzuje się najmniejszą dokładnością i stabilnością. Oscylator RC jest zbudowany tylko z dwóch elementów zewnętrznych: kondensatora i rezystora, a częstotliwość sygnału zegarowego zależy od napięcia zasilania, wartości rezystancji R_{EXT}, pojemności C_{EXT} i temperatury otoczenia. Z tych powodów ta opcja nie będzie nas interesować

Ciekawą alternatywą dla głównego generatora kwarcowego jest użycie wbudowanego w mikrokontroler wewnętrznego generatora RC o ustalonej częstotliwości, niewymagającego żadnych elementów zewnętrznych. Dokładność i stabilność jest na poziomie ±1...2% wystarczającym dla większości zastosowań, a dodatkowo ten generator po włączeniu zasilania zaczyna oscylować bardzo szybko w porównaniu z generatorem kwarcowym. Oscylator ma częstotliwość 8 MHz i jest kalibrowany w procesie produkcyjnym. Użytkownik może skalibrować go samodzielnie, zapisując rejestr kalibracji OSCTUNE. Maksymalna częstotliwość 8 MHz jest dzielona w bloku postskalera. Przez zapisywanie bitów ICRF <2:0> w rejestrze OSCCON (**rysunek 12**) można wybrać jedną z 8 dostępnych częstotliwości i elastycznie dobrać taktowanie do potrzeb. Po włączeniu zasilania domyślenie jest wybrany oscylator o częstotliwości 31 kHz. Jeżeli chcemy taktować z wyższą częstotliwością, to po zerowaniu trzeba przeprogramować bity ICRF2...ICRF0.

W praktyce będziemy wykorzystywać tryby XT i HS oraz tryb wewnętrznego oscylatora RC.

Bity konfiguracyjne

W mikrokontrolerach PIC18 w przestrzeni adresowej programu o adresach 30000h...3ffffh zaimplementowano nieulotną pamięć nastaw, umożliwiającą konfigurowanie i testowanie układów. Dostęp do tego obszaru jest możliwy

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IDLEN	ICRF2	ICRF1	ICFR0	OSTS	IOFS	SCS1	SCS0

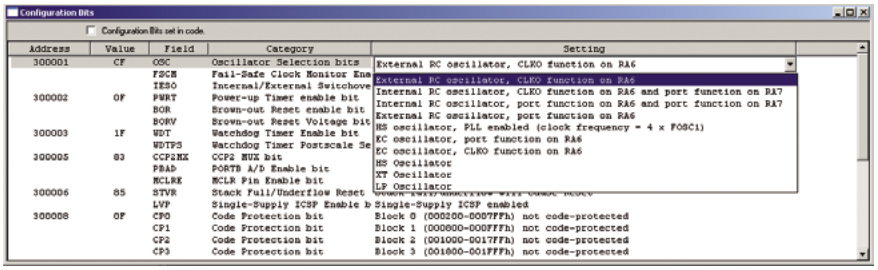
Bit 7 IDLEN
Tryb IDLE – rdzeń nie jest taktowany w trybie zarządzania energią
Tryb RUN – rdzeń jest taktowany w trybie zarządzania energią
Bity 6-4 IRCF <2:0>
111 8 MHz
110 4 MHz
101 2 MHz
100 1 MHz
011 500 kHz
010 250 kHz
001 125 kHz
000 31 kHz domyślne
Bit3 OSTS – bit statusu
Czas OST upłynął – taktowanie sygnałem zegarowym z oscylatora głównego
Czas OST upłynął – oscylator główny nie jest gotowy
Bit 2 IOFS – bit statusu
Wewnętrzny oscylator stabilny
Wewnętrzny oscylator niestabilny
Bity 1,0 SCS <1:0>
1x taktowanie wewnętrznym oscylatorem RC
01 taktowanie drugim oscylatorem kwarcowym
00 taktowanie głównym oscylatorem kwarcowym

Rysunek 12. rejestr sterujący OSCCON

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IESO	FSCM	-	-	FOSC3	FOSC2	FOSC1	FOSC0

IESO
=1 tryb Switchover włączony
=0 tryb Switchover wyłączony
FSCM
=1 tryb Fail-Safe Clock włączony
=0 tryb Fail-Safe Clock wyłączony
FOSC3:FOSC0 tryby generatora sygnału zegarowego taktującego mikrokontroler
11xx RC
1001 INTIO1
1000 INTIO2
0111 RCIO
0110 HSPLL
0101 EC
0100 ECIO
0010 HS
0001 XT
0000 LP

Rysunek 13. Rejestr konfiguracyjny CONFIG1H



Rysunek 14. Ustawienie bitów konfiguracyjnych

tylko w trakcie programowania mikrokontrolera. Znajomość bitów konfigurujących jest niezbędna, bo bez prawidłowych nastaw programowany układ może nie działać w ogóle lub będzie działał źle.

Wiemy już, że sygnał zegara taktującego może pochodzić z wielu różnych źródeł. Rodzaj generatora taktującego mikrokontrolerem programuje się przez zapisanie rejestru konfiguracyjnego CONFIG1H umieszczonego pod adresem 30001hex – rysunek 13 .

Bit IESO jest używany do odblokowania funkcji startu układu na wewnętrznym generatorze RC do momentu ustabilizowania się drgań oscylatora kwarcowego (Two-Seed Start-Up). Ma to oczywiście znaczenie, kiedy zależy nam na szybkim „starcie" układu w konfiguracji z oscylatorem kwarcowym.

Bit FSCM steruje włączaniem funkcji Fail Safe Clock Monitor. Kiedy jest odblokowana, wtedy w przypadku problemów ze startem lub pracą oscylatora kwarcowego taktowanie jest automatycznie przełączane na wewnętrzny generator RC. Warto wtedy tak zaprogramować OSCCON, by sygnał zegarowy z tego generatora miał taką samą lub zbliżoną częstotliwość do kwarcu.

Dokładne opisywanie wszystkich bitów rejestrów konfiguracyjnych nie jest konieczne. Wszystkie potrzebne informacje dotyczące konfiguracji można znaleźć w dokumentacji technicznej mikrokontrolera w dziale *Special Features of The CPU*.

Jednak w każdym projekcie musimy dokładnie ustalić konfigurację, bo w skasowanej pamięci wszystkie bity są jedynkami. Konfigurację można zaprogramować na 2 sposoby. Pierwszy sposób wykorzystuje okno *Configuration Bits* otwierane z zakładki *Configuration* (rysunek 14). Można tu, klikając na kolumnę *Setting*, w prosty sposób ustawić bity konfiguracyjne, tak jak sobie tego życzymy.

To rozwiązanie, chociaż wygodne, ma wadę. Jeżeli będziemy chcieli plikiem wygenerowanym w projekcie zaprogramować pamięć mikrokontrolera, nie mając całego projektu, to nie ma dostępu do informacji o bitach konfiguracyjnych. Z tego powodu lepiej jest konfigurację umieścić w programie. Zaznaczamy wtedy okienko *Configuration Bits set In Code* i *MPLAB* „wie", że konfiguracje trzeba pobrać z pliku wynikowego (.hex) projektu. W kompilatorze Hi-Tech zdefiniowane jest makro __CONFIG, w którym można ustalić bity konfiguracyjne, na przykład:

__CONFIG (HS&WDTDIS&LVPDIS&PWRTEN&FCMDIS&BOREN);

Definicja argumentów makra jest umieszczona w pliku nagłówkowym mikrokontrolera *pic18F2320.h*. Umieszczenie makra w pliku programu powoduje, że po wczytaniu pliku wynikowego powstałego w wyniku kompilacji automatycznie zostaną odpowiednio ustawione bity konfiguracyjne. Jest to również bardzo wygodne rozwiązanie, bo zwalnia każdorazowego ustawiania bitów po otwarciu programu MPLAB.

W każdym projekcie kursu takie makro zostanie zdefiniowane i układ będzie skonfigurowany. Jednak zachęcam do dokładnego przejrzenia opisu bitów konfiguracyjnych, żeby móc świadomie konfigurować mikrokontroler stosownie do własnych wymagań.

Tomasz Jabłoński, EP
tomasz.jablonski@ep.com.pl