8-bitowa kontrofensywa (1)

Nowe peryferie mikrokontrolerów PIC

Microchip, jeden z największych na świecie graczy w obszarze projektowania i produkcji mikrokontrolerów, wymyślił strategię, dzięki której użytkownicy poszukujący nieskomplikowanych mikrokontrolerów chętnie sięgną po 8-bitowe, sprawdzone mikrokontrolery PIC. Zastosowano pomysłowe połączenie starego, ale sprawdzonego i bardzo popularnego rdzenia PIC16 z nietypowymi, nowatorskimi peryferiami, pracującymi niezależnie od rdzenia (core independent). Ta „niezależność" polega na tym, że działanie układów peryferyjnych nie zależy od częstotliwości taktowania mikrokontrolera a ich praca nie obciąża CPU. Mogą zatem wykonywać pewne specyficzne czynności dużo szybciej niż rdzeń, a jedyne obszary wspólne to rejestry konfiguracyjne i rejestry danych.

Mikrokontroler w roli komponentu elektronicznego stał się tak pospolity, jak niegdyś tranzystor, a wcześniej lampa elektronowa. Ta popularność to wynik splotu wielu czynników: małej ceny, elastyczności zastosowań, szerokiej oferty, ale też umiejętności elektroników – programistów potrafiących go poprawnie zastosować i oprogramować w różnych aplikacjach. Firmy projektujące i produkujące mikrokontrolery stosują różne strategie, by pozyskać klientów w różnych obszarach zastosowania. Jedną z bardziej nośnych jest stosowanie szybkich 32-bitowych jednostek zamiast starszych 8-bitowych. Sprzyjać temu ma kompatybilność obudów *pin-to-pin* i niska cena, porównywalna z ceną prostych mikrokontrolerów. Ma to wiele zalet, ale też i wady. Programowanie 32-bitowców wymaga większej wiedzy programistów, zazwyczaj peryferie brane wprost z dużych jednostek są skomplikowane i trudniejsze w konfiguracji. Nie każdy projekt wykorzystujący mikrokontrolery musi być programowany przez programistę o wyższych kwalifikacjach. Z tych i podobnych powodów są nadal bardzo chętnie stosowane proste (z dzisiejszego punktu widzenia) jednostki 8-bitowe, mimo że ich cena jest porównywalna z „przeskalowanymi" 32-bitowcami.

Środowisko projektowe, kompilator i moduł ewaluacyjny

Do mikrokontrolerów są potrzebne odpowiednie narzędzia programowe. Microchip jest znany z tego, że zapewnia swoim klientom pełne wsparcie w postaci środowiska projektowego MPALB IDE, kompilatorów języka C oraz modułów ewaluacyjnych. Pakiet MPALB IDE jest bezpłatny, podobnie jak kompilatory C. Trzeba jednak pamiętać, że kompilatory C mogą być używane bez ograniczeń, w tym również do zastosowań komercyjnych, ale w wersjach bezpłatnych mają wyłączoną optymalizację wielkości lub szybkości kodu. Jeżeli zależy nam na optymalizowaniu kodu, to trzeba wykupić odpowiednią licencję.

Środowisko MPLAB X IDE można pobrać bezpłatnie ze strony producenta. Od jakiegoś czasu jest dostępna też wersja MPAB Xpress pracująca w chmurze, niewymagająca instalowania IDE i kompilatora C. Oba rozwiązania mają wady i zalety. Jeżeli pracujemy nad projektem w jednym miejscu i na jednym komputerze, to MPLAB X IDE jest wygodniejsze. Kiedy jednak pracujemy jednocześnie w domu, zakładzie, na uczelni, w delegacji itp., to MPLAB XPRESS daje więcej swobody i pozwala w prosty sposób korzystać z różnych komputerów bez konieczności przenoszenia aktualnej wersji projektu na każdy z nich.

Idea aplikacji pracujących w chmurze opiera się na dostępie do Internetu. Użytkownik nie zapisuje w pamięci swojego komputera wersji instalacyjnej programu i go nie instaluje. W czasie pracy nie zapisuje na dysku swojego komputera efektów pracy. Aplikacja jest otwierana poprzez okno przeglądarki internetowej na dowolnym komputerze dołączonym do Internetu. Wszystkie pliki są umieszczane gdzieś w pamięci jakichś serwerów. Użytkownik tworzy sobie konto z nazwą i hasłem i to konto identyfikuje projekty aplikacji.

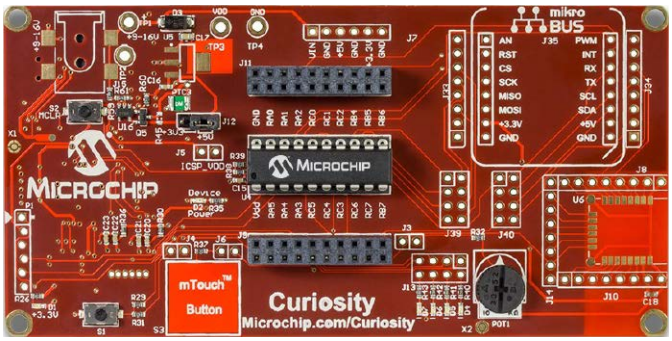
MPLAB X IDE pobiera się ze strony ***https://goo.gl/OhNjFe***, a kompilator MPLAB XC-8 ze strony ***https://goo.gl/x4rlyP***. Przy instalowaniu MPLAB X IDE trzeba pamiętać o odinstalowaniu wcześniejszych wersji za pomocą ich własnego deinstalatora. Wtedy po zainstalowaniu nowszej wersji będą pamiętane i otworzone konfiguracje projektów i ustawienia już zainstalowanych kompilatorów.

W czasie instalowania kompilatora będziemy pytani o licencję. Jeżeli jej nie kupiliśmy, to mamy dwa wyjścia: zainstalować wersję próbną **PRO Evaluation**, działającą przez 60 dni w pełnej wersji lub od razu zainstalować wersję bezpłatną, bez ograniczenia czasowego. W obu sytuacjach należy w czasie instalacji kierować się komunikatami instalatora.

Do celu tego artykułu korzystałem z pakietu MPLAB Xpress. Aby z niego korzystać, trzeba najpierw się zarejestrować (utworzyć konto) na stronie firmy Microchip. Jest to jedno wspólne konto, potrzebne na przykład do dostępu do próbek układów, dostępu do kompilatorów, dostępu do pomocy technicznej itp.

Do uruchomienia IDE wchodzimy na stronę ***https://goo.gl/U62SUF*** i logujemy się do swojego konta klikając na *My Account (login/register)*. Po prawidłowym wpisaniu użytkownika (adresu e-mail) i hasła otwiera się okno MPLAB Xpress.

Interfejs Xpress jest prawie identyczny z interfejsem MPLAB X IDE. Niewielkie uproszczenia są głównie wynikiem



Fotografia 1. Moduł Curiosity

ograniczenia funkcjonalności do mikrokontrolerów 8-bitowych i możliwości stosowania trzech programatorów/debuggerów. Do dyspozycji jest jak na razie tylko jeden plug-in MCC.

Przykłady konfigurowania modułów peryferyjnych będą testowane na module ewaluacyjnym Curiosity – **fotografia 1**. Ma on wbudowany programator/debugger współpracujący z driverami USB przeznaczonymi dla pakietu MPLAB Xpress. Zgodnie z założeniem producenta, jest to tani moduł przeznaczony do testowania mikrokontrolerów 8-bitowych w obudowach 8-, 14- i 20-wyprowadzeniowych DIP. Na płytce oprócz podstawki dla mikrokontrolera umieszczono „przycisk" do testowania funkcji interfejsu dotykowego mTouch, potencjometr do testowania przetwornika A/C i przycisk S1. Wyprowadzenia mikrokontrolera są doprowadzone do gniazd IDC20. Żeby moduł był bardziej uniwersalny, można do niego dołączać moduły zewnętrzne, produkowane przez firmę MikroElektronika (mikro BUS) oraz moduł Bluetooth RN4020. Płytka jest zasilana poprzez złącze USB, ale opcjonalnie można ją zasilić z zewnętrznego zasilacza o napięciu 9…16 V. Wymaga to wlutowania współosiowego gniazda zasilającego i stabilizatora +5 V.

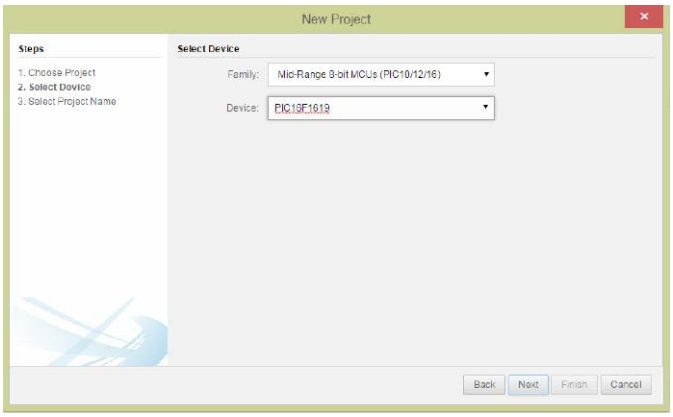
Moduł, który otrzymałem dzięki uprzejmości polskiego oddziału Microchipa, wyposażono w mikrokontroler PIC16F1619.

MPLAB Xpress – pierwszy projekt

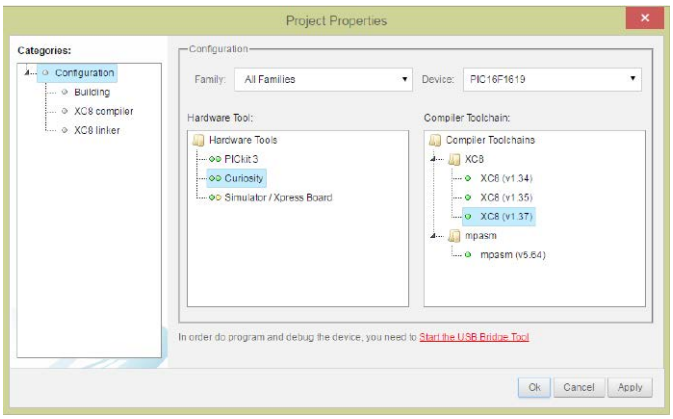
Jak w wielu podobnych środowiskach projektowych IDE, efekty pracy nad programową aplikacją są zapisywane w postaci projektu. Ma on swoją unikalna nazwę i zawiera – oprócz wszystkich niezbędnych plików źródłowych – pliki *makefile*, skrypty linkera oraz pliki zawierające ustawienia na przykład kompilatora, ale też ustawienia i konfiguracje plug-inów – u nas będą to ustawienia wtyczki MCC.

Kreator nowego projektu jest wywoływany z menu *File* → *New Project*. W pierwszym oknie jest wybierana kategoria Microchip Embeded i projekt Standalone Project (tylko ta opcja jest dostępna dla Xpress). W kolejnym kroku trzeba wybrać typ mikrokontrolera (**rysunek 2**). W ostatnim, trzecim oknie nadajemy nazwę projektu i kreator kończy działanie. Nazwa projektu i logiczne katalogi *Header Files* oraz *Source Files* są wyświetlane w oknie *Project*. Z projektem jest też związane okno *DashBoard* wyświetlające:

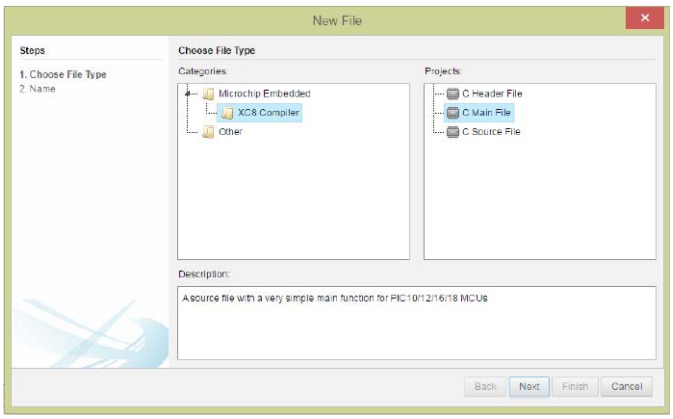
- Typ wybranego mikrokontrolera.
- Wybrany kompilator.
- Narzędzie do programowania/debugowania.
- Ustawienia debuggera: punkty zatrzymań (breakpoint) w kodzie programu.



Rysunek 2. Kreator projektu – wybór mikrokontrolera



Rysunek 3. okno właściwości projektu



Rysunek 4. Kreator dodawania pliku źródłowego main.c do projektu

W tym momencie projekt jest skonfigurowany domyślnie: mikrokontroler wybrany przez kreatora projektu i ostatnio używana lub najnowsza wersja kompilatora.

Ustawienia projektu są wykonywane w oknie *Project Properties*. To okno można wywołać na dwa sposoby. Pierwszy to kliknięcie prawym przyciskiem myszy na nazwę projektu w oknie *Project*, a drugi to kliknięcie na ikonkę w oknie Dashboard. Okno właściwości projektu pokazano na **rysunku 3**.

W zakładce *Configuration* można zmienić typ mikrokontrolera, wybrać z listy dostępny programator/debugger oraz jedną z dostępnych wersji kompilatora MPLAB XC8. Ustawienia dostępne w oknach Building, XC Compiler i XC8 linker można pozostawić domyślne.

Listing 1 Szkielet funkcji *main()*

```
/* File:   main.c
 * Author: tomasz.jablonski@ep.com.pl
 *
 * Created on 8/12/2016 1:17:29 PM UTC
 * „Created in MPLAB Xpress”
 */

#include <xc.h>

void main(void) {
    return;
}
```

Listing 2. Funkcje *System_Initialize()* i *OSCILLATOR_Initialize()*

```
#include „mcc.h”

void SYSTEM_Initialize(void)
{

    PIN_MANAGER_Initialize();
    OSCILLATOR_Initialize();
}

void OSCILLATOR_Initialize(void)
{
    // SCS INTOSC; SPLLEN disabled; IRCF 8MHz_HF;
    OSCCON = 0x72;
    // TUN 0;
    OSCTUNE = 0x00;
    // Set the secondary oscillator

}
```

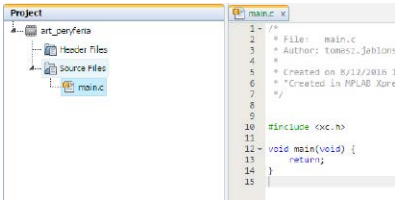
Kreator projektu nie generuje żadnych plików źródłowych, nawet szkieletu pliku z funkcją *main.c*. Ten plik można dodać na dwa sposoby. Pierwszy z nich to ręczne dodanie pliku źródłowego przez kliknięcie prawym przyciskiem myszy na folderze logicznym *Source Files* i wybranie kreatora *New File* (**rysunek 4**). W oknie *Categories* wybieramy *XC8 Compiler* i z listy *Projects* wybieramy *C Main File*. Kreator również umożliwia dodanie szkieletu dowolnego pliku źródłowego (*C Source File*) lub pliku nagłówkowego (*C Header File*). Po wybraniu *C Main File* w kolejnym oknie możemy nadać nazwę tworzonemu plikowi. Ja standardowo nazwałem swój plik *main.c*. Wygenerowany szkielet funkcji *main.c* pokazano na **listingu 1**.

Drugi sposób to wygenerowanie pliku z funkcja *main()* przez wtyczkę MCC. Jeżeli MCC nie znajdzie pliku z funkcją *main()* w trakcie generowania plików konfiguracyjnych, to automatycznie go wygeneruje. Ta druga opcja jest wygodniejsza, bo MCC umieszcza w funkcji *main()* wywołanie funkcji *SYSTEM_Initialize()* z odpowiednim plikiem nagłówkowym (**listing 2**).

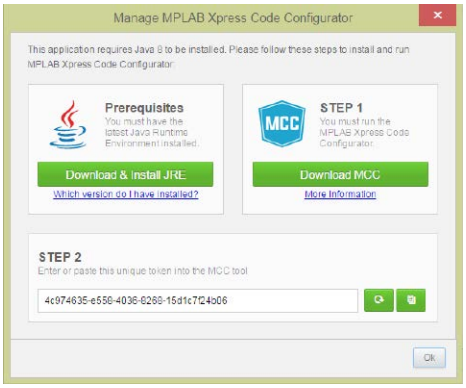
Plik został dodany do logicznego folderu *Source Files*. Te foldery nie mają bezpośredniego powiązania z fizycznymi folderami na dysku komputera. Należy je traktować jako listę plików mogących się fizycznie znajdować w dowolnym miejscu (na dysku lub w chmurze). Folder logiczny tylko je porządkuje, by łatwo je było otworzyć z poziomu projektu.

Wtyczka MCC – MPLAB Code Configurator

Każdy program napisany dla mikrokontrolera musi zawierać funkcje konfigurujące taktowanie, układ przerwań, jeżeli jest wykorzystywany i układy peryferyjne. Konfiguracja taktowania, włączenie/wyłączenie licznika watchdoga, włączenie/wyłączenie protekcji zapisu, układu kontroli napięcia zasilania (brown – out) itp. odbywa się przez programowanie bitów bezpieczników (fuse). Wszystkie czynności konfiguracyjne nawet dla prostych układów są czasochłonne i łatwo się przy tym pomylić. Dlatego coraz częściej do pakietów projektowych IDE są dołączane funkcje ułatwiające skonfigurowanie mikrokontrolera. W MPLAB IDE do tego celu jest przeznaczona wtyczka MCC (*MPLAB Code Configurator*). W MPLAB X IDE wtyczka



Rysunek 5. Okno projektu z dodanym plikiem main.c



Rysunek 6. Okno pobierania wtyczki MCC

jest ściągana z serwera Microchipa i instalowana. W przypadku MPLAB Xpress MCC jest to aplikacja w języku Java. Po kliknięciu na ikonkę MCC w pasku narzędzi jest wyświetlane okno (**rysunek 6**), w którym musimy w pierwszym kroku pobrać i zainstalować najnowszą wersję *Java Runtime Enviroment*. Potem pobieramy plik w *MCC_Xpresss.jnlp*. Wykonanie programu zawartego w tym pliku powoduje uruchomienie wtyczki MCC. Dla ułatwienia pracy można utworzyć skrót do *MCC_Xpresss.jnlp*.

Takie rozwiązanie uruchamiania MCC w MPLAB Xpress nie jest w moim odczuciu zbyt przemyślane. Po pierwsze, musimy pobrać niewielki co prawda plik i zapisać go w komputerze. Kłóci się to trochę z ideą pracy w chmurze. Po drugie, kliknięcie na ikonę MCC w MPLB Xpress oznacza coś zupełnie innego (otwarcie okna pobierania wtyczki – rys. 6) niż w „stacjonarnym” MPLAB X (otwarcie wtyczki MCC). To trochę irytujące i być może zostanie później poprawione. Okno MCC pokazano na **rysunku 7**.

Konfigurowanie taktowania mikrokontrolera

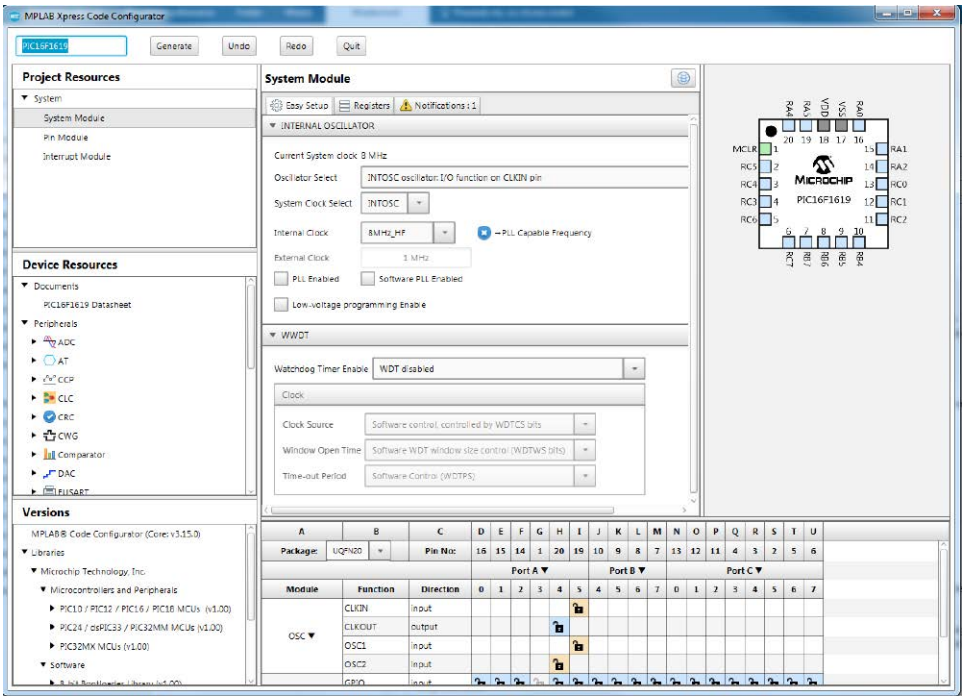
Na **rysunku 8** pokazano schemat blokowy taktowania mikrokontrolera PIC16F1619. Taki sam lub bardzo podobny układ taktowania znajdziemy we wszystkich mikrokontrolerach rodziny PIC16F1xxx. Moduł taktowania zawiera kilka różnych źródeł sygnału zegarowego. Można te źródła podzielić na dwie zasadnicze części: zewnętrzne i wewnętrzne.

Zewnętrznym źródłem taktowania może być sygnał zegarowy z generatora przebiegu prostokątnego. Zależnie od częstotliwości tego przebiegu moduł taktowania konfiguruje się do pracy w trzech trybach:

1. ECL (External Clock Low-Power) do 500 kHz.
2. ECM (External Clock Medium-Power) 500 kHz…4 MHz.
3. ECH (External Clock HIGH-Power) 4 MHz…32 MHz.

Drugim zewnętrznym źródłem sygnału taktującego może być oscylator kwarcowy o częstotliwości 4…10 MHz – tryb pracy HS (*High Gain Crystal Resonator*). Oprócz źródeł zewnętrznych dostępne są też trzy oscylatory RC wbudowane

w strukturę mikrokontrolera. Do taktowania rdzenia i peryferii jest wykorzystywany oscylator o częstotliwości 500 kHz powielonej w układzie PLL do 16 MHz. Oscylator 500 kHz jest kalibrowany w procesie produkcyjnym, ale użytkownik może dokonać poprawek jego częstotliwości, zapisując rejestr OSCTUNE. Potem częstotliwość z wyjścia układu PLL (16 MHz) można podzielić i uzyskać częstotliwości z zakresu 31,25 kHz…16 MHz, jak pokazano na **rysunku 8**. Jeżeli taktowanie z częstotliwością 16 MHz (maksymalna częstotliwość taktowania to 32 MHz) i dokładność oscylatora RC są wystarczające, to wewnętrzny oscylator RC jest bardzo dobrym wyborem.



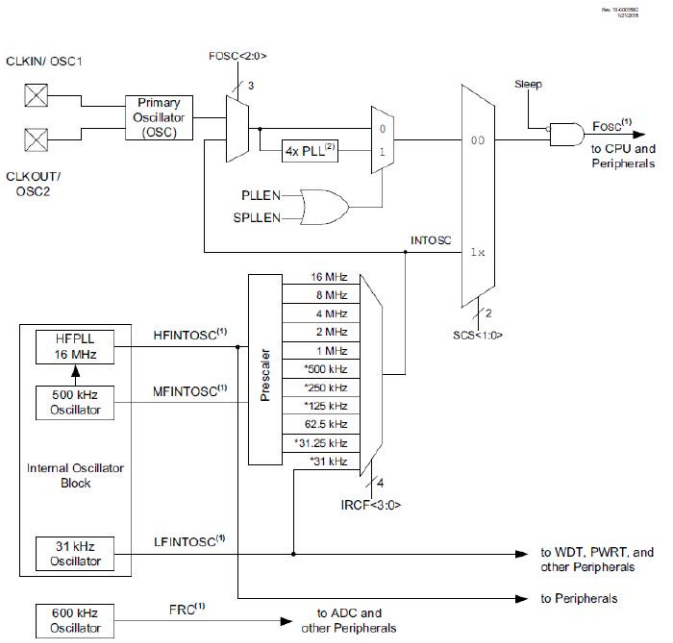
Rysunek 7. Okno wtyczki MCC

O tym, które źródło taktowania zostanie wybrane, decyduje rejestr OSCCON, a w zasadzie dwa najmłodsze bity SCS. Kiedy te bity są wyzerowane, to o tym, jak mikrokontroler jest taktowany, decydują bity FOSC<1:0> rejestru konfiguracyjnego (fuse). Dla SCS <1:0>=1x źródłem taktowania jest wewnętrzny oscylator RC. My w naszych przykładach będziemy używali wewnętrznego oscylatora RC o częstotliwości 8 MHz, skonfigurowanego za pomocą MCC.

Konfiguracja jest tak łatwa, jak to tylko możliwe. W oknie *System Module* wybieramy typ oscylatora *Oscillator Select* → *INTOSC Oscillator*, a następnie *System Clock Select* → *INTOSC* oraz *Internal Clock* → *8MHz* (**rysunek 9**). Po kliknięciu na przycisk *Generate*, MCC wygeneruje plik *mcc.c* zawierający:

- Funkcję *System_Initialize()*. Ta funkcja po każdej zmianie konfiguracji jest modyfikowana przez MCC i zawiera wywołanie funkcji konfiguracyjnych.
- Funkcję *OSCILLATOR_Initialize()*. Ta funkcja zapisuje rejestr OSCCON i zeruje rejestr OSCTUNE (**listing 3**).
- Definicję rejestrów konfiguracyjnych *fuse*.

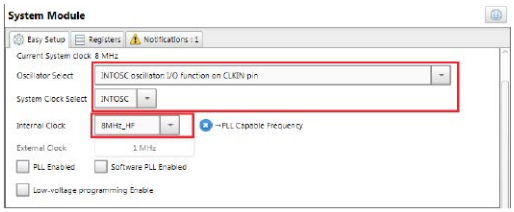
Zmiana poziomu logicznego bitów rejestrów konfiguracyjnych jest możliwa w zakładce *Registers* okna *System Module*. Dla



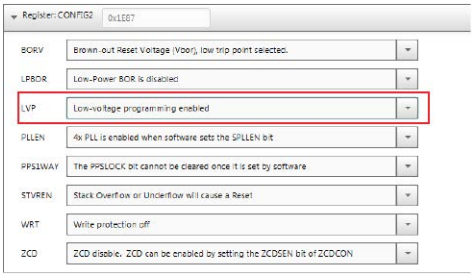
Rysunek 8. Układ taktowania mikrokontrolera PIC16F1619

Listing 3. Definicje bezpieczników

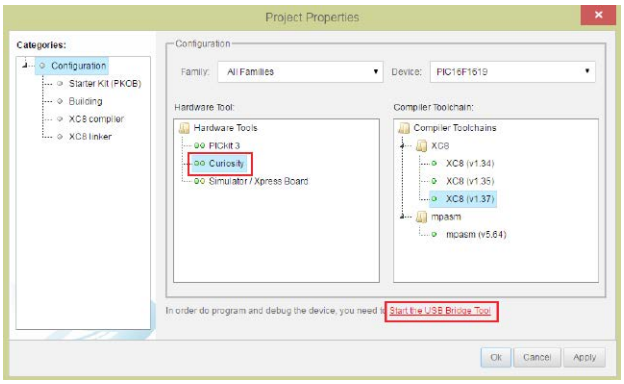
```
// CONFIG1
#pragma config FOSC = INTOSC    // Oscillator Selection Bits->INTOSC oscillator: I/O function on CLKIN pin
#pragma config PWRTE = ON    // Power-up Timer Enable->PWRT enabled
#pragma config MCLRE = ON    // MCLR Pin Function Select->MCLR/VPP pin function is MCLR
#pragma config CP = OFF    // Flash Program Memory Code Protection->Program memory code protection is disabled
#pragma config BOREN = ON    // Brown-out Reset Enable->Brown-out Reset enabled
#pragma config CLKOUTEN = OFF    // Clock Out Enable->CLKOUT function is disabled. I/O or oscillator function on the CLKOUT pin
#pragma config IESO = ON    // Internal/External Switch Over->Internal External Switch Over mode is enabled
#pragma config FCMEN = ON    // Fail-Safe Clock Monitor Enable->Fail-Safe Clock Monitor is enabled

// CONFIG2
#pragma config WRT = OFF    // Flash Memory Self-Write Protection->Write protection off
#pragma config PPS1WAY = ON    // Peripheral Pin Select one-way control->The PPSLOCK bit cannot be cleared once it is set by software
#pragma config ZCD = OFF    // Zero Cross Detect Disable Bit->ZCD disable.  ZCD can be enabled by setting the ZCDSEN bit of ZCDCON
#pragma config PLLEN = OFF    // PLL Enable Bit->4x PLL is enabled when software sets the SPLLEN bit
#pragma config STVREN = ON    // Stack Overflow/Underflow Reset Enable->Stack Overflow or Underflow will cause a Reset
#pragma config BORV = LO    // Brown-out Reset Voltage Selection->Brown-out Reset Voltage (Vbor), low trip point selected.
#pragma config LPBOR = OFF    // Low-Power Brown Out Reset->Low-Power BOR is disabled
#pragma config LVP = ON    // Low-Voltage Programming Enable->High-voltage on MCLR/VPP must be used for programming

// CONFIG3
#pragma config WDTCPS = WDTCPS1F    // WDT Period Select->Software Control (WDTPS)
#pragma config WDTE = OFF    // Watchdog Timer Enable->WDT disabled
#pragma config WDTCWS = WDTCWSSW    // WDT Window Select->Software WDT window size control (WDTWS bits)
#pragma config WDTCCS = SWC    // WDT Input Clock Selector->Software control, controlled by WDTCS bits
```



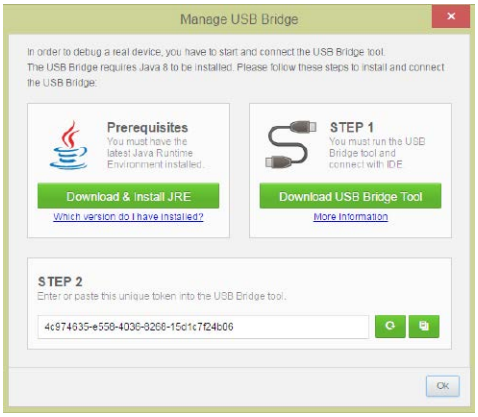
Rysunek 9. Konfiguracja oscylatora



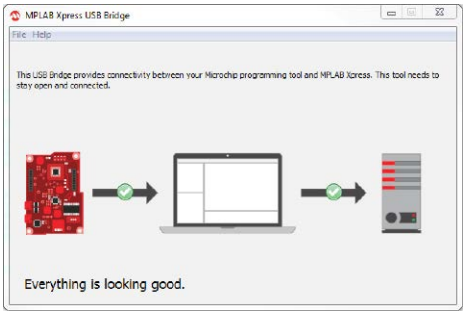
Rysunek 10. fragment okna z ustawieniami bezpieczników



Rysunek 11. Wybór modułu Curiosity i rozpoczęcie pobierania USB Bridge



Rysunek 12. Okno pobierania USB Bridge Tool



Rysunek 13. Prawidłowe połączenie Curiosity z komputerem i MPLAB Xpress

naszych celów wyłączymy *watchdog*, włączymy *PWRTE* i ustawimy *MCLRE* jako *MCLR*. Bezwzględnie należy też włączyć możliwość programowania LVP (niskim napięciem programującym). Ustawienie wysokiego napięcia programującego zablokuje możliwość programowania w Curiosity i trzeba będzie przeprogramować mikrokontroler za pomocą zewnętrznego programatora. Pozostałe ustawienia mogą zostać z wartościami domyślnymi.

Programator – zapisanie pliku wynikowego do pamięci mikrokontrolera

Żeby można było zaprogramować pamięć mikrokontrolera umieszczonego w podstawce modułu Curiosity, trzeba go połączyć z portem USB komputera, a następnie pobrać i uruchomić program USB Bridge. Proces jest podobny jak w przypadku wtyczki MCC. Otwieramy okno *Project Properties*, wybieramy *Hardware Tools* → *Curiosity* i klikamy na *Start the USB Bridge Tool* (**rysunek 11**). Klikamy na *Download Bridge Tool*, pobieramy i zapisujemy plik *USBBridge.jnlp*. Po jego uruchomieniu, gdy moduł jest przyłączony do portu USB komputera, powinno być wyświetlone okno informujące, że wszystko zostało wykonane poprawnie (**rysunek 13**). Reasumując: aby zaprogramować układ umieszczony w module Curiosity, trzeba:

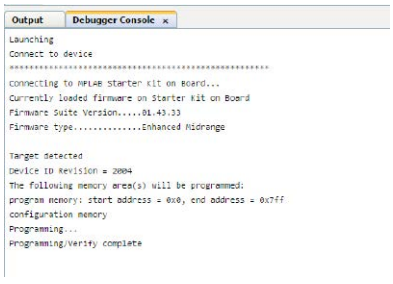
- Przyłączyć moduł do portu USB komputera.
- Przy otwartym MPLAB Xpress uruchomić program *USBBridge.jnlp*.
- Jeżeli wszystko jest prawidłowo (rys. 13), to trzeba kliknąć na ikonkę .
- Xpress wywoła kompilator XC-8, skompiluje program i jeżeli nie ma błędów, połączy się z programatorem/debuggerem z modułu Curiosity i zapisze pamięć programu i rejestry konfiguracyjne.

Oprócz programowania można użyć sprzętowego debuggera. Uruchamia się go po kliknięciu na ikonkę . Xpress kompiluje program z opcją DEBUG_RUN, łączy się z programatorem /debuggerem Curiosity i ładuje program do pamięci mikrokontrolera. Debuger dysponuje tylko jednym punktem zatrzymania *break point*. W czasie uruchamiania mamy do dyspozycji:

- Zatrzymanie debugowania (pause) .
- Restart mikrokontrolera .
- Uruchomienie wykonywania programu .
- Wykonanie jednego kroku z wejściem do funkcji (step Into) .
- Wykonanie jednego kroku bez debugowania wywoływanych funkcji (step over) .

Jak widać, sprzętowy debugger jest dość skromny, ale i tak pozwala na efektywne wykrywanie błędów w prostych programach.

TOMASZ JABŁOŃSKI, EP



Rysunek 14. Wydruk z konsoli w czasie programowania mikrokontrolera