MPLAB X IDE

Nowe środowisko projektowe od Microchipa

Darmowy pakiet MPLAB IDE jest znany wszystkim użytkownikom wszystkich rodzin mikrokontrolerów firmy Microchip. Przez lata, wraz z naturalnym rozwojem tego typu narzędzi i poszerzaniem oferty o nowe wyroby, zmieniał się też MPLAB IDE. Te zmiany miały najczęściej charakter ewolucyjny, ale co jakiś czas trafiała się mała rewolucja. Tak było na przykład przy zmianie z wersji 5 na wersję 6. Zmieniono wtedy całkowicie między innymi interfejs graficzny. A jak jest w MPLAB X?

Rysunek 1. Okno pobierania

Ostatnią, najbardziej znaną jest wersja MPLAB IDE V8.x. Wspiera ona projektowanie wszystkich rodzin mikrokontrolerów oraz ma drivery dla wszystkich, obecnie produkowanych, firmowych programatorów i emulatorów sprzętowych, włączając w to moduły ewaluacyjne. W dotychczasowej wersji MPLAB IDE zapewnia kompleksową obsługę w zakresie projektowania układów z mikrokontrolerami PIC. Jednak jego koncepcja i interfejs użytkownika z czasem zaczęły odstawać od podobnych, konkurencyjnych narzędzi. Zapewne z tego powodu producent postanowił odświeżyć IDE i zaproponował MPLAB X IDE. Już sama zmiana nazwy sugeruje, że jest to całkowicie nowy produkt, zrywający z poprzednimi rozwiązaniami.

Oprogramowanie MPLAB X IDE pojawiło się już jakiś czas temu w wersji beta. To dość powszechny wybieg stosowany przez firmy, pozwalający wykorzystać aktywnych użytkowników do testowania jeszcze niegotowego wyrobu. Ponieważ IDE jest zaawansowanym pakietem dającym duże możliwość projektantom, a jednocześnie dystrybuowanym darmowo, to takie podejście jest zupełnie zrozumiałe. Etap końcowego testowania jest bardzo pracochłonny, a przez to kosztowny. „Zatrudnienie" chętnych do współpracy beta-testerów pozwala na znaczące zmniejszenie kosztów. Obecnie jest dostępna oficjalna, stabilna wersja V1.10 i należy spodziewać się, że usunięto większość zauważonych błędów.

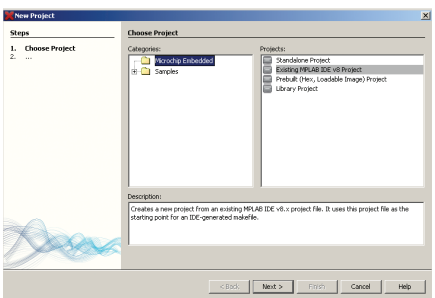
MPLAB X IDE wykonano w oparciu o platformę open source – NetBeans IDE. Pozwala to firmom trzecim na powiększanie funkcjonalności pakietu poprzez dostarczanie dodatkowych modułów, tzw. wtyczek lub plug-inów. Należy spodziewać się, że podniesie to atrakcyjność środowiska. Poza tym, w końcu X IDE może być również używane w systemach Linux i Mac OS! Wszystkie wcześniejsze wersje pakietu można było uruchomić tylko w systemie Windows, a współpraca z produktami firm trzecich była bardzo utrudniona.

Praca nad projektem w MPALB IDE V8.x jest organizowana przez przestrzeń roboczą (*Workspace*) i projekt (*Project*). W przestrzeni roboczej można otworzyć więcej niż jeden projekt i z poziomu pakietu nawigować pomiędzy nimi. Głównym elementem projektu jest wybór mikrokontrolera. Zależnie od niego trzeba wybrać jeden aktywny kompilator i programator/debuger. Każda zmiana mikrokontrolera wymaga zmiany konfiguracji projektu.

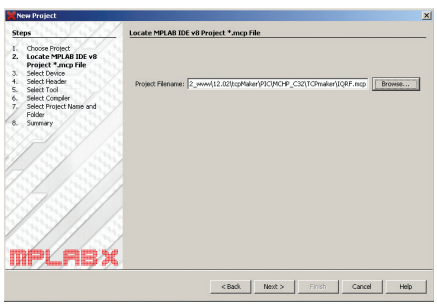
W MPLAB X IDE użytkownik również musi utworzyć projekt, w którym wybiera się mikrokontroler, kompilator i programator/debuger. Nie ma pojęcia przestrzeni roboczej. Jeżeli to konieczne, to projekty można grupować poleceniem *Project Grouping*.

Jedną z bardziej zauważalnych różnic pomiędzy poprzednimi wersjami a wersją X jest możliwość jednoczesnej pracy z kilkoma kompilatorami i narzędziami typu programator/debuger w ramach jednego projektu. Na przykład, kiedy dla mikrokontrolera PIC18 mamy zainstalowanych kilka rozwojowych wersji kompilatorów MPLAB C-18 lub HI-Tech, to bez zmiany konfiguracji MPLAB X IDE, możemy wybrać dowolną wersję. Podobnie jest z programatorem lub debugerem.

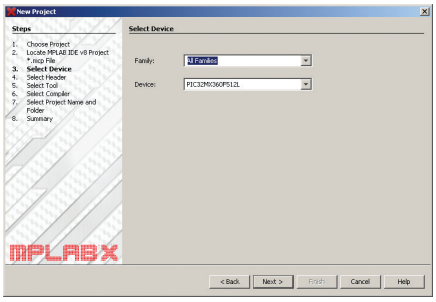
Program można pobrać ze strony producenta spod adresu *http://ww1.microchip.com/downloads/mplab/X/index.html* (**rysunek 1**). W oknie *Select Platform* wybieramy system operacyjny. Razem z pakietem IDE – zaznaczając odpowiednie okienka – mo-



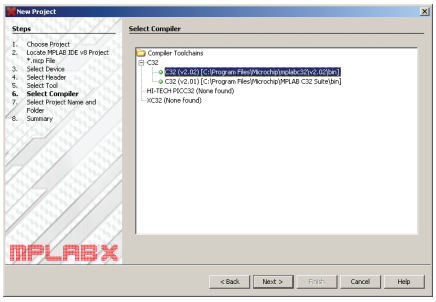
Rysunek 2. Konwersja projektu MPLAB IDE V8 na MPLAB X IDE



Rysunek 3. Otworzenie istniejącego projektu dla MPLAB IDE V8.x



Rysunek 4. Wybór mikrokontrolera



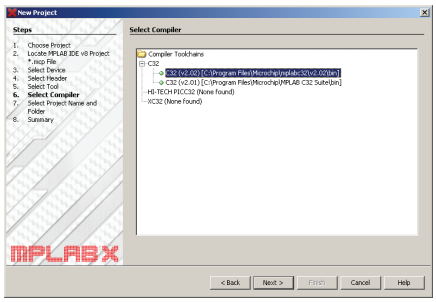
Rysunek 5. Wybór programatora/ debugera

żemy zaopatrzyć się w bezpłatne wersje Lite kompilatorów języka C do wszystkich rodzin mikrokontrolerów PIC.

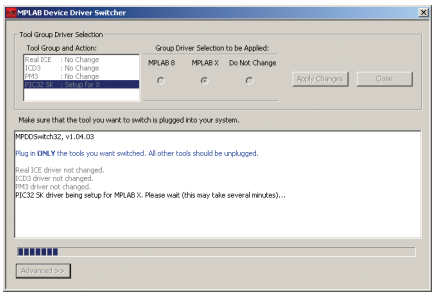
Instalacja programu jest nieskomplikowana i nie odbiega od podobnych (dla systemu Windows). Do prawidłowego działania jest wymagane środowisko Java Runtime Enviroment. Razem z MPLAB X IDE instaluje się aplikacja *MPLAB Driver Switcher* używana przy jednoczesnej pracy z MPLAB IDE V8.x i MPLAB X IDE.

Kompatybilność z MPLAB IDE V8.x

Pracę z pakietem rozpoczynamy od utworzenia lub nowego projektu, lub otwarcia już istniejącego. Pierwszą rzeczą, którą starałem



Rysunek 6. Wybór kompilatora



Rysunek 7. MPLAB Device Driver Switcher

się sprawdzić, była możliwość otwierania projektów utworzonych za pomocą starszej wersji V8.x. Trudno sobie bez tego wyobrazić bezbolesną zmianę na nowe środowisko. Na szczęście jest taka możliwość i nie ma potrzeby ponownego wyważania otwartych drzwi. Ale nie można otworzyć projektu MPLAB IDE V8 bezpośrednio w MPLAB X IDE, ponieważ format plików projektów w obu IDE jest inny. Dlatego otwieranie starego projektu jest w rzeczywistości tworzeniem go na nowo przez kreatora projektu, ale z wykorzystaniem danych z pliku *.mcs projektu MPLAB IDE V8. Czynność ta jest wykonywana z menu *File -> New Project* (**rysunek 2**). W zakładce *Projects* możemy wybrać:

- *Standalone Project* – nowy projekt dla MPLAB X IDE.
- *Existing MPLAB IDE V8 Project* – konwersja MPLAB IDE V8 do MPLAB X IDE.
- *Prebuilt (HEX , Loadable Image) Project* – otworzenie gotowego pliku wynikowego.
- *Library Project* – projekt biblioteki.

Jeżeli chcemy przekonwertować stary projekt, wybieramy opcję *Existing MPLAB IDE V8 Project*. Otwiera się wtedy kolejne okno pokazane na **rysunku 3**. Z lewej strony okna są pokazane kroki, które należy wykonać, aby dokonać konwersji:

- Wyszukanie pliku *.mcp projektu MPLAB IDE V8 (rysunek 3).
- Wybór mikrokontrolera (**rysunek 4**).
- Wybór pliku nagłówkowego.
- Wybór programatora – starter kit PIC32 (**rysunek 5**).
- Wybór kompilatora (**rysunek 6**).
- Zapisanie nowego projektu.

Takie same kroki – z pominięciem kroku 2 (rys. 3) – są wykonywane przy tworzeniu nowego projektu (*Stand Alone Project*). W wypadku otwierania istniejącego projektu MPLAB IDE V8 dodawane są wszystkie pliki źródłowe, a w nowotworzonym – pliki źródłowe trzeba dodać samodzielnie.

Konwersja projektów nie jest trudna, a jej wykonanie jest łatwe i intuicyjne. Ja w ten sposób bez problemu przekonwertowałem projekt serwera WWW wygenerowany przez program TCPMaker.

MPLAB IDE V8.x i MPLAB X IDE mogą być zainstalowane na jednym komputerze i w trakcie działania nie przeszkadzają sobie. Współpracę MPLAB IDE V8.x z firmowymi narzędziami poprzez interfejs USB zapewniały firmowe drivery specjalnie napisane do tego celu. W MPLAB X do obsługi magistrali USB zastosowano standardowe drivery WinUSB/LibUSB, które mają działać szybciej i nie przysparzać problemów. Potwierdziły to już pierwsze moje próby z modułem *PIC32 Starter Kit*.

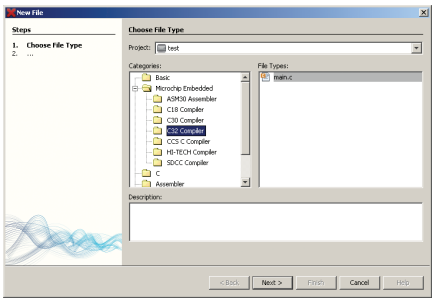
Dwa różne drivery dla jednego narzędzia wymagają przełączania. Do tego celu jest przeznaczona specjalna aplikacja *MPLAB Driver Switcher* instalująca się z MPLAB X IDE. Na **rysunku 7** pokazano okno *MPLAB Device Driver Switcher* w trakcie przełączania drivera modułu *PIC32 Starter Kit* do MPLAB X IDE. Trakcie przełączania muszą być zamknięte MPLAB IDE V8 i MPLAB X IDE i moduł musi by podłączony do USB. Tu również, po próbach z *PIC32 Starter Kit* okazało się, że wszystko działa jak powinno. Przełączanie jest konieczne dla *Real ICE*, *ICD3*, *PM3* i wspomnianego *PIC32 Starter Kit*. Większość nowych modułów ewaluacyjnych oraz *PICkit-2* i *PICkit-3* nie wymagają przełączania driverów.

Projekt

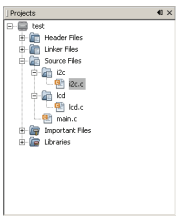
Głównym elementem nowotworzonego projektu są pliki źródłowe. Nowy plik jest tworzony w kreatorze wywoływanym z menu *File -> New File* (skrót klawiszowy *Ctrl+N*). W oknie *Categories* wybiera się typ pliku: dla kompilatora Basic, dla kompilatorów C (zakładka *Microchip Embeded* lub C), dla asemblera, skryptu linkera itp. Po wybraniu typu pliku, w kolejnym oknie nadaje mu się nazwę i wybiera lokalizację.

Utworzony plik dodaje się do projektu klikając prawym przyciskiem myszy na zakładce *Source Files* w oknie *Projects* i wybierając z menu rozwijanego *Add Existing Item*. Z tego samego menu można też utworzyć nowy plik lub folder logiczny. Foldery logiczne są pomocne przy grupowaniu plików w rozbudowanym projekcie. Na **rysunku 9** pokazano okno Projects z przykładowymi plikami źródłowymi pogrupowanymi w folderach *i2c* (plik *i2c.c*), lcd (plik *lcd.c*). Plik main.c jest umieszczony w folderze głównym.

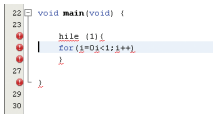
Wspieranie tworzenia plików źródłowych przez kreatora i foldery logiczne to przydatne ułatwienia, ale podobne mechanizmy są znane z poprzedniej wersji IDE. Dużo



Rysunek 8. tworzenie nowego pliku źródłowego



Rysunek 9. Okno Projects z przykładowymi plikami źródłowymi



Rysunek 10. Błędy edycji

większe zmiany są w samym edytorze plików źródłowych. Każdy, kto pisze programy – obojętnie w jakim języku – wie, że kompilator wymaga 100% poprawności syntaktycznej. Każda literówka, brak średnika czy przecinka, nie mówiąc już o kompletności nawiasów, powoduje wygenerowanie komunikatu o błędach składni przez kompilator. Edytor MPALB X IDE ma wbudowana kontrolę poprawności „on-line". Już w trakcie pisania programu są wychwytywane i zaznaczane błędy syntaktyczne i jeżeli takie są, to na zostaną wykryte przez edytor. Nie ma potrzeby czasochłonnego sprawdzani składni poprzez kompilowanie pliku lub całego projektu.

Na **rysunku 10** pokazano fragment programu z celowo wygenerowanymi błędami. Numery linii, w których znaleziono błędy, są zastępowane symbolami (czerwone kółko z wykrzyknikiem). Dodatkowo, nieprawidłowe wyrażenia w liniach są podkreślane na czerwono. Przyznam, że ta funkcjonalność edytora bardzo mi się spodobała.

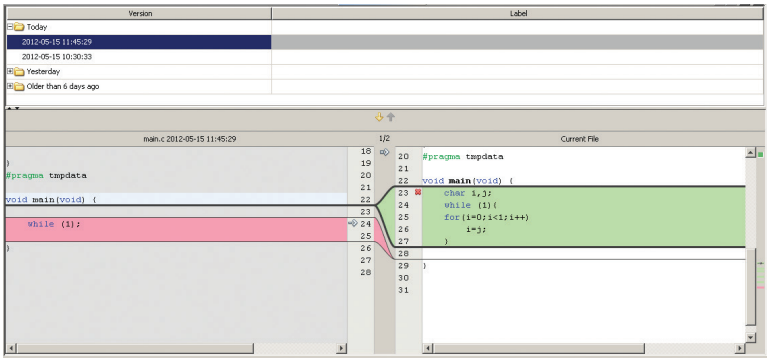
Przydatną funkcją jest zaznaczanie przez edytor wszystkich wystąpień zmiennej w pliku źródłowym po kliknięciu na nią w dowolnym miejscu programu. Występowanie zaznaczonych zmiennych jest wyświetlane w linijce umieszczonej na prawej krawędzi okna edytora. Wystarczy kliknąć na kolorowym markerze na tej linijce, a edytor przeniesie kursor do odpowiedniego miejsca. Jest to bardzo wygodne przy bardziej rozbudowanych programach, które z natury nie mieszczą się na jednym ekranie.

Istnieje też możliwość wyszukiwania ciągu znaków w komentarzach. Ciągi można dowolnie definiować w zakładce *Tools -> Options -> Miscellaneous -> Tascs*. Jest to również pomocne w identyfikacji fragmentów bardzo rozbudowanych programów składających się z wielu plików.

Jeżeli w projekcie jest plik źródłowy, do którego musimy się odwoływać częściej niż zwykle (na przykład *main.c*), to można go sklonować poprzez kliknięcie prawym klawiszem myszy na belkę z jego nazwą w edytorze i wybraniu z rozwijanego menu *Clone Document*. Po utworzeniu kopii można go przesunąć (za pomocą lewego przycisku myszy) w dolny obszar okna edytora.

Nowa funkcja *Local history* pozwala na obejrzenie i ewentualne cofnięcie/ponowienie zmian w pliku źródłowym (**rysunek 11**).

Gotowy projekt jest kompilowany po wykonaniu polecenia *Run -> Build Project* lub kliknięciu na ikonę z młotkiem. Są wtedy kompilowane tylko pliki, które zostały zmodyfikowane po ostatniej kompilacji. Kompilowanie wszystkich plików, które jest połączone z usunięciem wszystkich plików przejściowych z rozszerzeniem **.o* jest wykonywana po wybraniu *Run -> Clean and Build Project* lub po kliknięciu na ikonę z młotkiem i miotłą. Przebieg i wynik kompilacji są wyświetlane w oknie *Output*.



Rysunek 11. Okno funkcji Local History

Właściwości projektu są ustawiane poleceniem *Run -> Set Project Configuration -> Customize*. Na **rysunku 12** pokazano okno projektu dla mikrokontrolera z rodziny PIC32 z kompilatorem MPLAB C-32 i programatorem/debugerem *PIC32 Starter Kit*. Można tu między innymi zmieniać wersję kompilatora i jego ustawienia, na przykład model pamięci i poziom optymalizacji. Ważne dane związane z projektem są wyświetlane w oknie Dashboard (**rysunek 13**). Są tu umieszczone między innymi: typ mikrokontrolera, używany kompilator, zajętość pamięci RAM i Flash, aktywny programator/debuger.

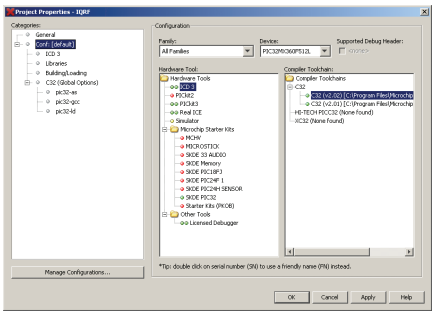
Zazwyczaj na którymś z etapów projektu trzeba przejść przez etap uruchamiania programu, czyli inaczej mówiąc – wykonać debugowanie. Dziś uruchamianie programu wykonuje się niemal zawsze w aplikacji docelowej. Ma to bardzo wiele zalet, ale wymaga odpowiednich narzędzi – debugerów. Microchip oferuje wiele ich typów, ale obecnie najbardziej znane to: *MPLAB ICD2*, *MPLAB ICD3*, *PICkit-2*, *PICKit-3*, *REAL ICE*. Można też zauważyć tendencję (nie tylko w firmie Microchip) do wyposażania modułów ewaluacyjnych we własny, wbudowany programator/debuger. Każde z tych rozwiązań pozwala na programowanie pamięci Flash mikrokontrolera i uruchamianie programu w układzie docelowym. MPLAB X IDE nie wspiera wszystkich narzędzi firmowych. Dotyczy to na przykład kultowego, ale już dość leciwego, MPLAB ICD2.

Uruchamianie w systemie wiąże się z zaprogramowaniem pamięci mikrokontrolera. MPLAB X IDE nie rozróżnia czy program ma być skompilowany w trybie *Release* (do zaprogramowania pamięci Flash bez debugowania), czy w trybie *Debug*, tak jak to było w MPLAB IDE V8.x. Jeżeli chcemy zaprogramować pamięć Flash, to używamy polecenia *Make and Program Device*. Jeżeli będziemy debugować program, trzeba wybrać polecenie *Program Device for Debugging*. Oba polecenia automatycznie uruchamiają kompilator i tworzą kod wynikowy.

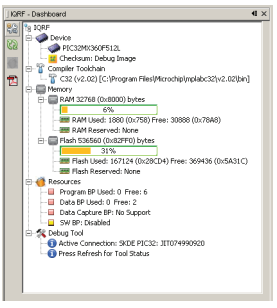
Debugowanie projektu rozpoczyna się od wykonania polecenia Debug->Debug Project. MPLAB X IDE łączy się z debugerem, programuje pamięć Flash i automatycznie wywołuje program zapisany w mikrokontrolerze. Funkcja uruchamiania w systemie ma podstawowy zestaw komend spotykany w podobnych rozwiązaniach:

- *Pause* – wstrzymanie wykonywania się programu.
- *Continue* – wznowienie wykonywania się programu.
- *Reset* – restart mikrokontrolera.
- Krokowe wykonywanie się programu: *step into*, *step over*, *run to cursor* itp.

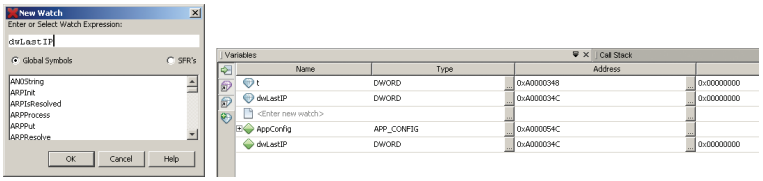
Działanie komend jest intuicyjne i nie wymaga szerszego wyjaśnienia.



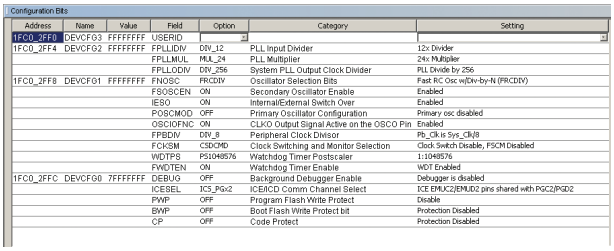
Rysunek 12. Okno właściwości projektu



Rysunek 13. Okno dashboard



Rysunek 14 Dodawanie zmiennej do okna Watch i okno Watch



Rysunek 15 Okno bitów konfiguracyjnych PIC32MX360F512L

Krokowe wykonywanie programu to oczywiście tylko cześć możliwości debugera. Jedną z podstawowych funkcji jest ustawianie pułapek – *Breakpoint*. Pułapkę ustawia się bardzo łatwo – wystarczy kliknięcie lewym przyciskiem myszy na numerze linii w pliku źródłowym. Ustawienie pułapki jest sygnalizowane zmianą numeru linii na symbol pułapki – czerwony kwadrat. Na linijce z prawej strony okna edytora pojawia się czerwony marker pozwalający na odnalezienie pułapki w dużym pliku źródłowym. Pułapka jest usuwana po kliknięciu na jej symbol. Możliwe jest też ustawianie rozbudowanych pułapek warunkowych z menu *Debug -> New BraeakPoint*. Po osiągnięciu linii z ustawioną pułapką wykonywanie programu jest zatrzymywane i kolor linii zmienia się na zielony.

Kolejnym elementem debugera jest podglądanie zawartości zmiennych – okno *Watch*. Zmienne są dodawane do okna poleceniem *New Watch* z menu rozwijanego po kliknięciu na pliku źródłowym prawym przyciskiem myszy. Na **rysunku 14** pokazano okna dodawania zmiennych do okna *Watch* (góra) i samo okno Watch (dół). Watch umożliwia wyświetlanie struktur, unii, tablic i oczywiście – rejestrów SFR.

Zawartość pamięci programu, danych, SFR i bitów konfiguracyjnych można podejrzeć z menu *Window -> PIC Memory Views*. Na **rysunku 15** pokazano okno z bitami konfiguracyjnymi mikrokontrolera PIC32MX360F512L.

Podsumowanie

Używam pakietu MPLAB IDE od wielu lat. Zaczynałem od wersji V5.x i projektów pisanych wyłącznie w asemblerze. Od wersji V.6.x program nie zmieniał się w sposób radykalny. Jednak wszystkie wersje umożliwiały bezproblemową pracę z aktualnie produkowanymi mikrokontrolerami i firmowymi narzędziami – głównie programatorami i debugerami. Microchipowe IDE miało zawsze opinię dobrego, solidnego narzędzia. Kiedy podstawowym kompilatorem stał się kompilator języka C, to MPLAB IDE również zaczął wspierać tworzenie programów w C. Czasem, głownie dla firm zewnętrznych, wymagało to zainstalowania dodatkowego drivera, ale mimo tego wszystko działo poprawnie. Użytkownicy mikrokontrolerów Micrcochip uznali MPLAB IDE jako standard i raczej rzadko korzystają z innych środowisk.

Nowy MPLAB X IDE najprawdopodobniej również będzie niekwestionowanym standardem. Właściwie znając dotychczasowe podejście Microchipa do jakości firmowych narzędzi można być tego pewnym. Mnie się ten pakiet bardzo podoba, mimo że jest to stosunkowo nowy produkt i jeszcze będzie rozwijany. Po pierwszych próbach, a potem krótkiej, właściwej pracy nad nowym projektem uważam, że w wielu aspektach przewyższa stare rozwiązania. Przede wszystkim, ma w wiele bardziej elastyczny i nowoczesny interfejs użytkownika. Bardzo przypadł mi do gustu edytor z analizą składni i wieloma funkcjami dodatkowymi. Możliwość pracy z wieloma wersjami kompilatorów i narzędzi sprzętowych jest również bardzo cenna. Liczę też na to, że platforma open source zaowocuje wieloma ciekawymi wtyczkami niezależnych producentów lub pasjonatów programowania.

Jedyna rzecz, która mnie zmartwiła, to brak wsparcia dla mojego starego *MPLAB ICD2*. To bardzo dobre narzędzie, ale jak już wspomniałem – dość leciwe. Widocznie uznano, że nie będzie wspierane, tym bardziej, że są produkowane nowe mikrokontrolery, których nie można zaprogramować za pomocą ICD2. Oczywiście, nie jest to jakiś wielki problem, bo mogę nadal korzystać z najnowszej wersji MPLAB IDE V8.x.

Przedstawiony przeze mnie opis jest z konieczności subiektywnym wyborem małej części możliwości pakietu. Dokładniej można się zapoznać z podstawowymi właściwościami MPLAB X IDE po przeczytaniu dokumentu *Getting Started with MPLAB X*, napisanego w formie prezentacji. Można go pobrać ze strony pobierania pakietu. W trakcie pracy pomocny będzie dokument *MPLAB X IDE User's Guide*. Oba są dostępne na stronie Microchipa.

Tomasz Jabłoński, EP