Flowcode

Zagrożenie dla BASCOMa

Narzędzie dla zabieganych

Nie tak dawno na łamach „Elektroniki Praktycznej" rozgorzała dyskusja o tym, co jest lepsze: pisanie w asemblerze czy w języku wysokiego rzędu, na przykład w C. Wiele podobnych dyskusji można spotkać na forach dyskusyjnych dla elektroników. Każda ze stron przedstawia swoje racje: asemblerowcy o totalnej kontroli nad programem, programujący w języku C o szybkim tworzeniu kodu i możliwości przenoszenia na różne platformy. W swojej praktyce przeszedłem oba te etapy, a teraz zacząłem zauważać również i inne możliwości.

Z doświadczeń programisty

Początkowo pisałem w asemblerze, ponieważ nie miałem do dyspozycji żadnego innego kompilatora, a później dlatego, że nauczyłem się metod i każde zadanie wydawało mi się proste do realizacji właśnie w asemblerze. Jednak w pewnym momencie zrozumiałem, że asembler, mimo pewnych oczywistych zalet, nie ma i nie będzie miał przyszłości. Wówczas zacząłem pisać programy w języku C. Kompilatory tego języka są coraz bardziej doskonałe, dostępne i tanie. Nawet więcej – producenci mikrokontrolerów oferują darmowe wersje z licencją do celów edukacyjnych, a w Internecie można bez trudu znaleźć darmowe kompilatory bez żadnych ograniczeń, udostępniane na zasadach licencji GNU. Przykładem takich programów są kompilatory dla mikrokontrolerów AVR (np. AVR GCC) i ARM (np. GNU ARM).

W dzisiejszym świecie wszystko trzeba robić szybko. Również pisać programy. Dlatego programista piszący na przykład w C ma przewagę nad piszącym w asemblerze. Ktoś może powiedzieć, że doświadczony asemblerowiec, korzystając z makroasemblera, może tworzyć swoje programy równie szybko. Zgoda, ale tylko do momentu zmiany platformy sprzętowej. Przejście na inny typ mikrokontrolera praktycznie wiąże się z nauką nowego języka, nowej architektury i napisaniem programu od nowa. Wiem, bo doświadczyłem tego wiele razy.

Takie dylematy dotyczą głównie ludzi, którzy zawodowo się zajmują techniką mikroprocesorową. Jest jednak spora grupa tych, którzy mikrokontrolery wykorzystują okazjonalnie lub są one im potrzebne jako dodatek do głównego zajęcia. Między innymi dla takich użytkowników zaczęły powstawać środowiska pozwalające szybko tworzyć programy. Takim środowiskiem jest bardzo popularny u nas Bascom. Zintegrowane z programatorem środowisko, pozwalające pisać programy w dość łatwym do nauczenia się języku Basic, szybko znalazło swoich wiernych fanów. Bascom ma wbudowanych wiele funkcji bibliotecznych obsługujących najbardziej popularne układy peryferyjne, takie jak na przykład alfanumeryczny wyświetlacz LCD. Napisanie prostego, ale efektownego programu współpracującego z wyświetlaczem zajmuje chwilę. Nie powoduje to zniechęcenia często doświadczanego przez tych, którzy zaczynają od asemblera lub nawet od C. Zresztą, zjawisko dołączania bibliotek obsługi układów peryferyjnych zaczęło dotyczyć też komercyjnych kompilatorów C.

W walce o klienta producenci oprogramowania narzędziowego dla mikrokontrolerów poszukują takiego narzędzia, które pozwalałoby na szybkie tworzenie programu bez konieczności poznawania (przynajmniej na początku) jakiegoś języka programowania. Zamiast pisać program, można go narysować. Takim narzędziem jest Flowcode firmy Matrixmultimedia przeznaczone do graficznego tworzenia programów dla mikrokontrolerów rodzin PIC16 i PIC18 firmy Microchip, ale też w innych wersjach dla mikrokontrolerów AVR i z rdzeniem ARM.

Kiedy pierwszy raz otworzyłem pudełko z Flowcode i przeczytałem zawartość krótkiej książeczki dołączonej do płyty, pomyślałem sobie: to jest właśnie narzędzie dla tych, którym mikrokontrolery są tylko trochę potrzebne. Muszą szybko napisać prosty program, nie wnikając w to, jak to wszystko działa. W miarę poznawania możliwości Flowcode to pierwsze wrażenie stawało się coraz bardziej mylne. Ale po kolei.

Narysuj sobie program

Flowcode jest dystrybuowany na standardowej płycie CD-ROM. W pudełku wersji instalacyjnej umieszczona jest naklejka z wydrukowanym kluczem, który należy wpisać na początku, podczas instalacji programu. Po zainstalowaniu programu niezbędne jest przeprowadzenie dodatkowej procedury rejestracji. Jeżeli przebiegnie ona prawidłowo, to producent wyśle poprzez e-mail drugi klucz, którym trzeba zakończyć instalowanie pełnoprawnej wersji programu (która nie ma żadnych ograniczeń). Na rejestrację mamy 30 dni od daty instalacji. W tym czasie zainstalowana wersja nie ma żadnych ograniczeń, jednak jeśli w tym czasie nie zarejestrujemy programu, to po upływie tego terminu zostaną wprowadzone ograniczenia funkcjonalne. Producent tłumaczy taką procedurę koniecznością walki z piractwem komputerowym i trudno nie przyznać mu racji.

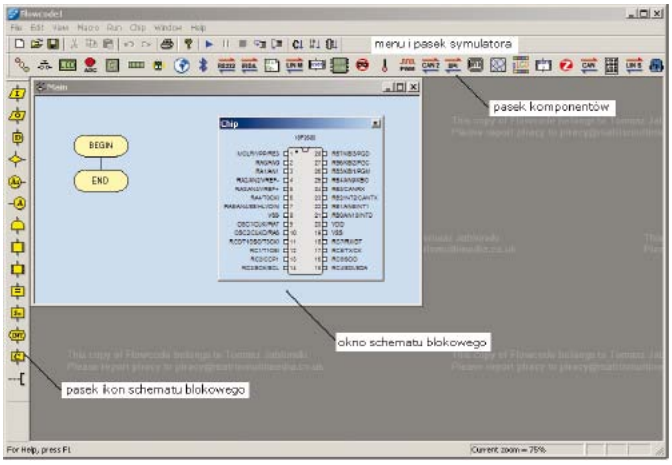
Instalacja przebiega standardowo i raczej nie sprawia żadnego problemu. Dodatkową pomocą jest drukowana instrukcja dołączana do płyty. Na początku instalacji można opcjonalnie zainstalować program PPP3, który obsługuje firmowy programator USB mikrokontrolerów PIC będący składowym płytki ewaluacyjnej E-Block.

Program sterujący pracą mikrokontrolera rysowany jest w formie schematu blokowego techniką „chwyć i puść" ikonę. Jeżeli ktoś nie chce, to nie musi używać żadnego języka programowania, ale producent Flowcode przewidział możliwość „wstawek" pisanych w języku C. Bardzo atrakcyjnym elementem tego programu narzędziowego jest rozbudowana biblioteka obsługi urządzeń zewnętrznych i układów peryferyjnych samego mikrokontrolera. Do dyspozycji są funkcje sterujące wyświetlaczem numerycznym LED, alfanumerycznym LCD, graficznym LCD, obsługujące odczytywaniem styków dołączanych do linii mikrokontrolera i klawiatury matrycowej. Można wybierać z bardzo bogatej oferty funkcji wspierających transmisję danych od standardowych interfejsów RS232, SPI i I^2C poprzez CAN i LIN do Bluetooth i TCPIP. Znajdziemy tu też wsparcie dla ZigBee, RFID oraz wbudowany webserver. Jako ciekawostkę można potraktować dodanie kompletnych funkcji dla urządzeń typu programowany robot (pojazd) i centralka alarmowa oferowanych przez producenta Flowcode.

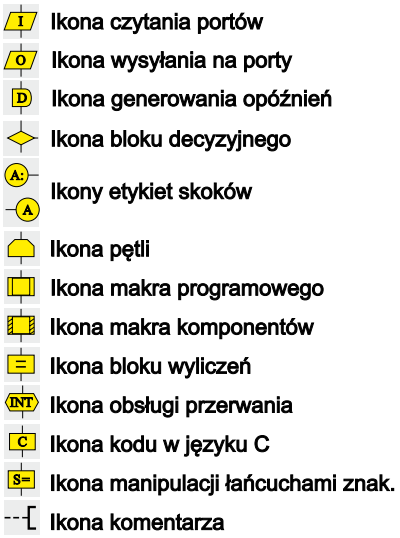
Zaraz po narysowaniu schematu blokowego, a jeszcze przed skompilowaniem do kodu wykonywalnego przez mikrokontroler, program można przetestować na wbudowanym, szybkim programowym symulatorze.

Nowy projekt

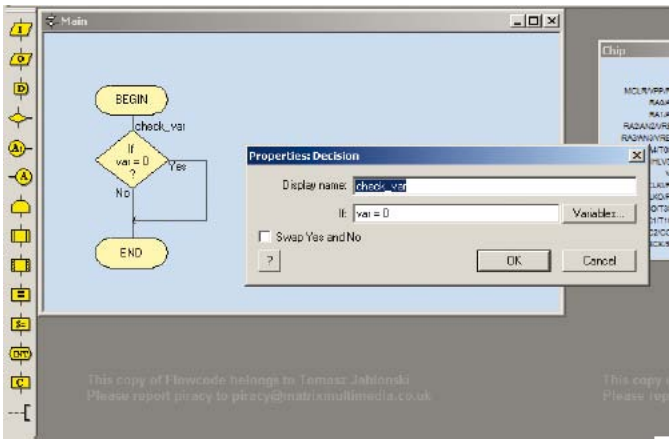
Tworzenie nowego projektu rozpoczyna się standardowo przyciskiem *New* z menu *File*. Potem wybierany jest typ mikrokontrolera. Na **rys. 1** pokazano fragment okna projektu dla mikrokontrolera



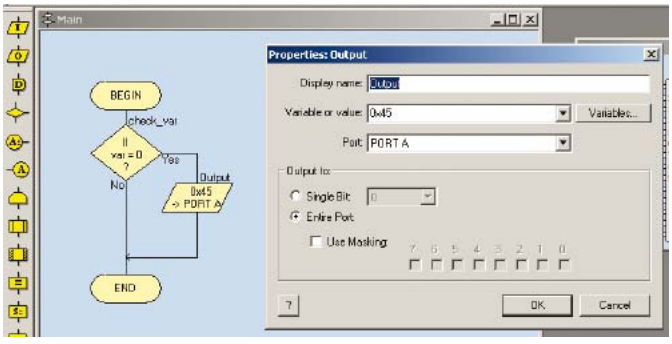
Rys. 1. Okno początku projektu



Rys. 2. Ikony Flowcode



Rys. 3. Dodanie bloku decyzyjnego



Rys. 4. Dodanie bloku manipulacji liniami wyjściowymi portu

PIC18F2680. Okno schematu blokowego zawiera tylko dwa elementy *Begin* i *End*, a w oknie *Chip* wyświetlana jest obudowa z wyprowadzeniami.

Schemat blokowy programu rysuje się, używając 14 ikon umieszczonych w pionowym pasku narzędziowym z lewej strony okna programu (**rys. 2**). Ikonę wybiera się po najechaniu na nią kursorem myszki, a potem naciska się i przytrzymuje lewy klawisz. Po przeniesieniu w żądane miejsce na schemacie pomiędzy istniejącymi już blokami puszcza się lewy klawisz i ikona zostaje automatycznie wstawiona pomiędzy istniejące bloki.

Na **rys. 3** pokazano przykładowy schemat po dodaniu ikony bloku decyzyjnego. W otwartym oknie właściwości *Properties Decision* wpisuje się nazwę bloku (*check_var*) i warunek *if(var=0)*. Po zaznaczeniu okienka *Swap Yes and No*, zamieniane są miejscami wyniki porównania. Tam, gdzie jest *Yes*, pojawi się *No*, a tam gdzie *No*, pojawi się *Yes*. Oczywiście sprawdzanie warunku bez żadnej akcji w rozgałęzieniu nie ma sensu. Możemy tam dodać dowolną ikonę, lub grupę ikon, na przykład funkcję *OUT* – wystawienia kombinacji stanów na liniach portów (**rys. 4**). W zakładce właściwości ustawia się port i wartość do niego wpisywaną. Można również zaznaczyć opcję *Single bit* wraz z numerem bitu, a w oknie *Value* wpisać 0 lub 1.

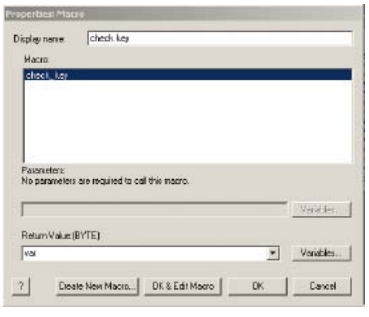
Nasz króciutki program teraz wykonuje sekwencję: sprawdza wartość zmiennej *var* i jeżeli jest ona równa zero, to na linii portu PORTA wystawiana jest wartość 45H. Podobnie działa blok odczytujący stany linii portów reprezentowany przez ikonę *IN*. Tu również w zakładce właściwości ustala się czytany port, a przeczytana wartość jest wpisywana do wcześniej zdefiniowanej zmiennej.

Tworzenie makr

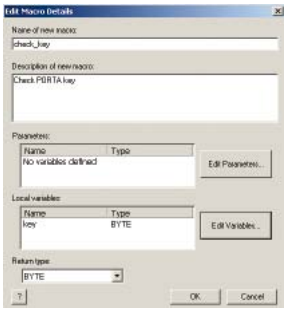
Ikony umieszczone na pasku umożliwiają narysowanie każdego algorytmu, jednak większe programy na pewno staną się nieczytelne. Jeżeli programy mają nie stać się wielką plątaniną bloków, to musi być możliwość dzielenia ich na mniejsze funkcjonalne fragmenty. Programista piszący w języku C ma możliwość wywoływania funkcji z argumentami wejściowymi. Funkcje te mogą zwracać zmienną lub wskaźnik do listy zmiennych. Poza tym, w dobrych kompilatorach można zadanie dzielić na pliki, co również bardzo ułatwia poruszanie się po dużych programach.

W pakiecie Flowcode wykonywane zadania programowe można dzielić na części nazywane makrami programowymi. Takie makro to nic innego, jak podprogram z argumentami i możliwością zwracania zmiennych. Po umieszczeniu ikony wywołania makra na schemacie, trzeba w zakładce właściwości z listy *Macro* wybrać nazwę podprogramu (**rys. 5**).

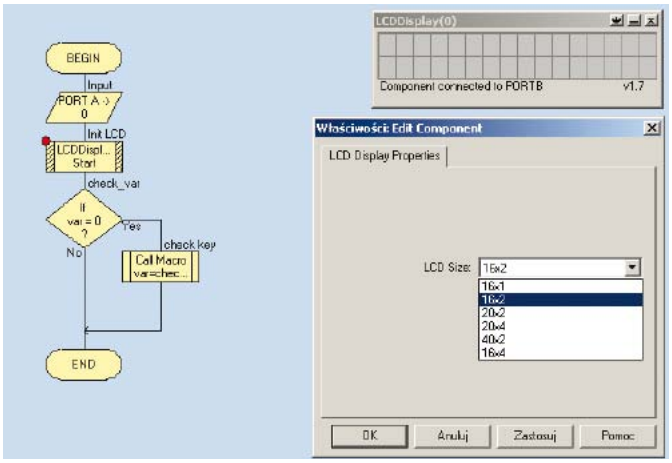
Jeżeli lista jest pusta, lub potrzebny jest nowy podprogram, to po naciśnięciu przycisku *Create New Macro* należy utworzyć nowe makro. W pierwszej linijce wpisujemy nazwę, która będzie wyświetlana w miejscu wywołania tego makra. Potem można opisać funkcje wykonywane przez podprogram makra – bardzo użyteczna właściwość przy pisaniu większych programów. Podprogram może jak każda funkcja mieć zdefiniowane argumenty. Definiuje się też zmienne lokalne i typ



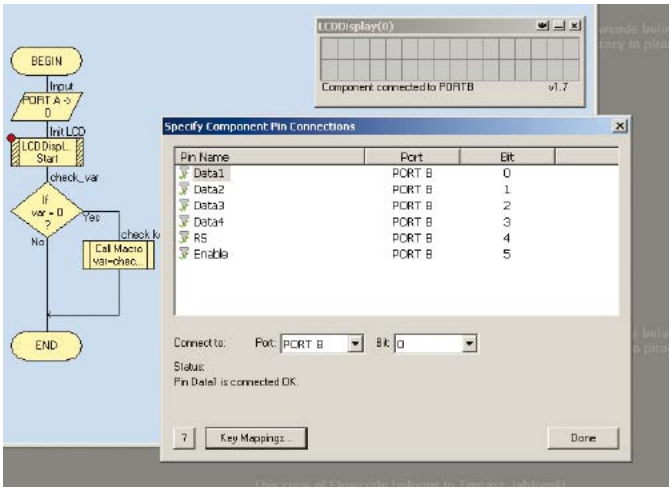
Rys. 5. Okno właściwości bloku makro



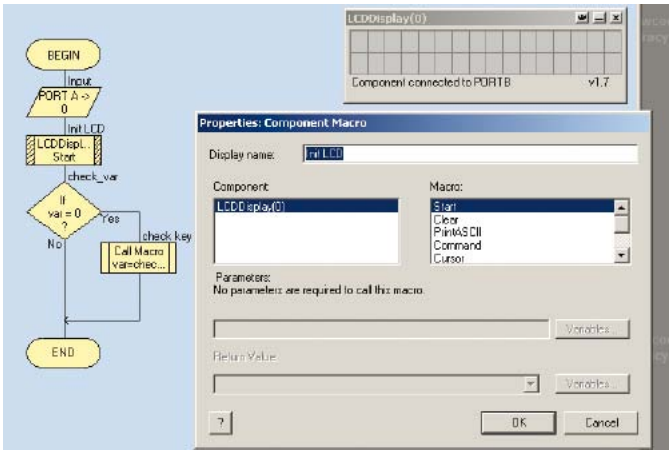
Rys. 6. Okno definiowania nowego makra programowego



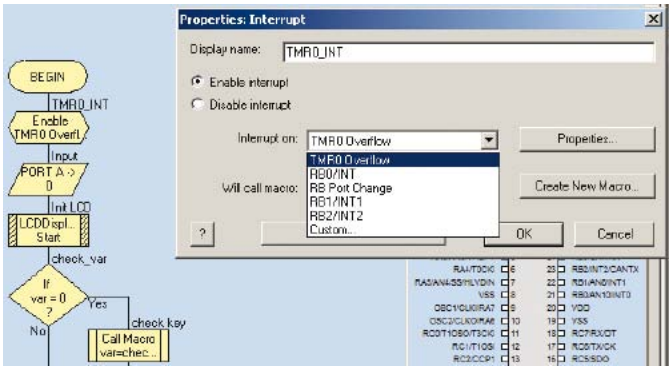
Rys. 7. Dodanie wyświetlacza alfanumerycznego



Rys. 8. Definicja połączeń wyświetlacza



Rys. 9. Wybranie funkcji inicjalizacji wyświetlacza



Rys. 10. Wybór źródła przerwania

zwracanej zmiennej. Wszelkie dostępne manipulacje związane z makrami są dostępne z zakładki Macro, umieszczonej w pasku Menu.

Komponenty zewnętrzne

Drugim ważnym typem jest makro komponentów powiązane z obsługą dodawanych, zewnętrznych komponentów. Jako przykład może służyć dodanie do programu, przez kliknięcie na symbol na pasku komponentów, obsługi wyświetlacza alfanumerycznego ze sterownikiem z HD44780. W oknie właściwości wybierana jest jego wielkość. W naszym przypadku będzie to 2×16 znaków (**rys. 7**). Interfejs sterujący jest 4-bitowy, a sposób podłączenia linii ustala się w oknie *Pin Connections*, pokazanym na **rys. 8**. Biblioteka obsługi wyświetlacza składa się z kilku funkcji opisanych dokładnie w pliku pomocy. Na **rys. 9** wybierana jest funkcja Start inicjująca 4-bitowy interfejs wyświetlacza.

Napisanie funkcji obsługi wyświetlacza na przykład w języku C nie jest zadaniem trudnym dla średnio wprawnego programisty, ale sprawia sporo kłopotów początkującym. W tym środowisku projektowym sprowadza się do użycia kilku gotowych makr, a dodatkowo efekt można szybko zweryfikować na programowym symulatorze.

Obsługa przerwań

Zakłada się, że obsługa przerwań jest kolejnym krokiem wtajemniczenia programisty mikrokontrolerów. Przy stosowaniu przerwań potrzebna jest dyscyplina i dobre rozpoznanie ich mechanizmu oraz rejestrów sterujących. Użycie Flowcode powoduje, że przerwania stają się łatwiejsze do obsługi niż kiedykolwiek wcześniej. Po dodaniu ikony przerwań i otwarciu okna właściwości trzeba wybrać źródło przerwania z listy (**rys. 10**).

Żeby nie było tak różowo, to program całkowicie wspiera tylko przerwania od TMR0, przerwania zewnętrzne i zmiany na liniach portów. Dla pozostałych przerwań trzeba wybrać *Custom*, co wiąże się z napisaniem w C kodu odblokowania przerwań i ewentualnie procedury ich obsługi. Z pewną pomocą przychodzi plik pomocy, gdzie umieszczone są proste przykłady dla przerwań od układów peryferyjnych mikrokontrolera PIC16F88.

Przygodę z przerwaniami możemy rozpocząć od utworzenia programu obsługi, na przykład przerwania od licznika TMR0. Z przerwaniem trzeba skojarzyć programowe makro procedury obsługi. Tworzy się je, klikając na *Create New Macro*.

Po kliknięciu na belkę *Properties* (rys. 10) pojawia się okno definiowania parametrów pracy licznika (**rys. 11**).

Dokładne opisanie wszystkich możliwości edycji programu można znaleźć w pliku pomocy.

Symulator

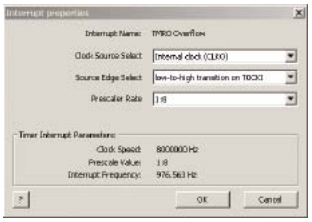
Po narysowaniu programu lub w jego trakcie można zweryfikować zamierzone efekty działania, używając wbudowanego programowego symulatora. Na pasku symulatora są umieszczone cztery standardowe przyciski symulacji: *Run*, *Pause*, *Step into* i *Step over* (**rys. 12**)

Symulator rozpoczyna działanie po przyciśnięciu przycisku *Run*, *Step into* lub *Step over*. Symulacja ciągła jest zatrzymywana przyciskami *Pause* lub *Stop* (również wyjście z symulatora). Przy ikonach można ustawiać pułapki programowe (*breakpoint*). Ikona z pułapką jest oznaczona w lewym górnym rogu czerwoną kropką. Przykład działania symulatora pokazano na **rys. 13**.

Krótki program do zmiennej *var* wpisuje wartość 2, inicjalizuje wyświetlacz LCD i wyświetla w górnej linijce napis „Symulacja Flow–" a w dolnej – „code". W trakcie symulowania w oknie komponentu wyświetlacza widać na bieżąco wynik działania programu. Jeżeli w programie są używane zmienne, to ich wartość jest wyświetlana w oknie *variables*. Nawigowanie pomiędzy wieloma makrami ułatwia okno *Call stack*.

Kompilacja

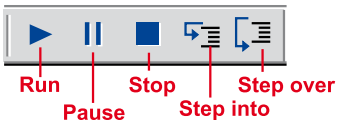
Ostatnim etapem pracy nad programem jest kompilacja schematu blokowego. Mamy do wyboru trzy możliwości: kompilacja do pliku programu napisanego w języku C, kompilacja do pliku wynikowego z rozszerzeniem hex i kompilacja do pliku wynikowego, automatycz-



Rys. 11. Definiowanie parametrów pracy licznika TMR0

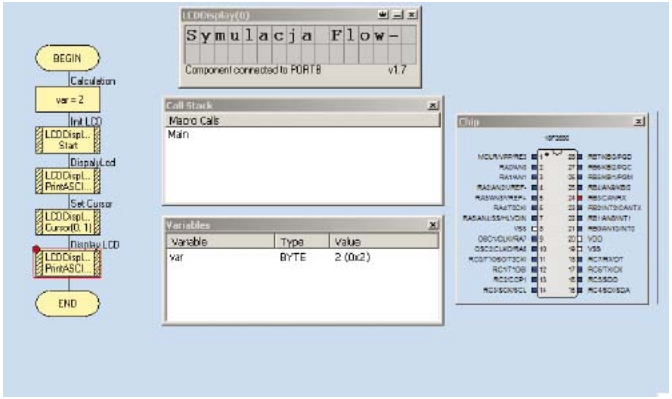
ne uruchomienie firmowego programatora z modułu ewaluacyjnego E-Block i zaprogramowanie pamięci mikrokontrolera. Rodzaj kompilacji wybierany jest z menu *Chip* lub ikonami menu.

Kompilowanie schematu blokowego do postaci kodu w języku C umożliwia wykorzystanie wyniku pracy w innych projektach. Sprawdziłem, to wklejając testowe fragmenty programów obsługujących wyświetlacz do programu kompilowanego kompilatorem PCH firmy CCS. Tak otrzymany program działał bez żadnego problemu. Producent wykorzystuje kompilator i linker Boost C, a kod wynikowy jest optymalizowany dla tego kompilatora. Kod w języku C może też służyć do dokładniejszej analizy i optymalizacji tworzonego programu.

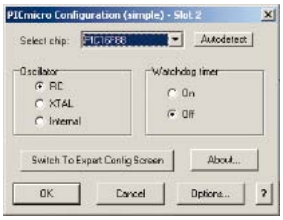


Rys. 12. Przyciski symulatora

Kompilowanie do kodu wynikowego musi być poprzedzone ustawieniem bitów konfiguracyjnych mikrokontrolera. Uproszczona wersja okna ustawiania bitów dla mikrokontrolerów PIC12 i PIC16 *PICMicro Configuration simple* jest dostępna z menu *Chip –> Configure* (**rys. 14**). Wszystkie bity, w tym również dla mikrokontrolerów z rodziny PIC18, można ustawiać w oknie *PICMicro Configuration expert* dostępnym po kliknięciu belki *Switch To Expert Config Screen*.



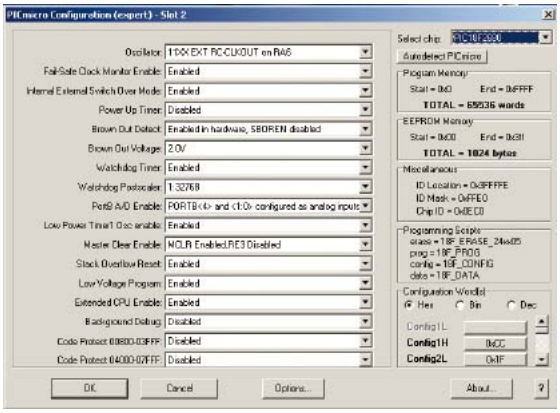
Rys. 13. Działanie symulatora programowego



Rys. 14. Okno bitów konfiguracyjnych PIC12 i PIC16

Po skompilowaniu tworzony jest plik wynikowy z rozszerzeniem hex, a z menu *Chip –> View Asm* można zobaczyć źródło programu w języku asembler.

Wybierając *Chip –> Compile to Chip*, powodujemy, że program jest kompilowany i jeżeli nie ma błędów, to kod wynikowy jest przesyłany przez programator PPP do pamięci mikrokontrolera umieszczonego w module E-Block. Ponieważ w trakcie testów dysponowałem tym modułem, sprawdziłem również działanie tej opcji. Jest to bardzo wygodne rozwiązanie, bo jednym kliknięciem myszki można program skompilować i wynik przesłać do mikrokontrolera.



Rys. 15. Okno zaawansowanych ustawień bitów konfiguracyjnych dla PIC18F2680

Podsumowanie

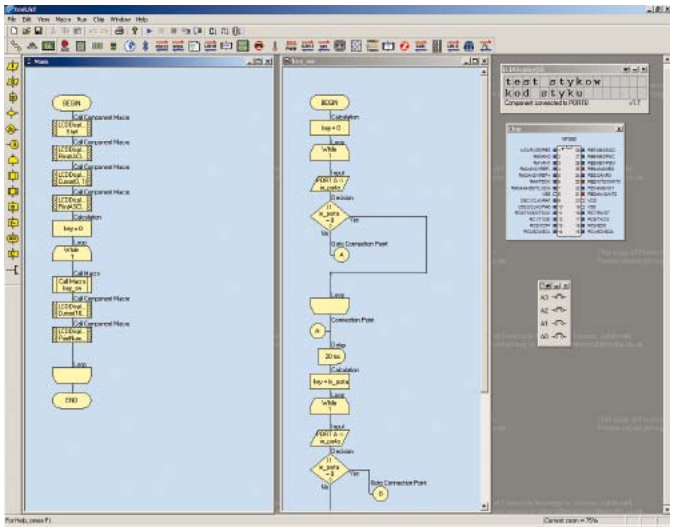
Flowcode jest narzędziem na pewno godnym uwagi. Ma wiele trudnych do przecenienia zalet. Najważniejsza z nich to szybkość i łatwość tworzenia programów. Jest też łatwy do opanowania, bo używanych jest tylko 13 ikon (plus ikona komentarza). Początkujący programista może bez problemu poruszać się po doskonale przygotowanych przez producenta gotowych elementach: ikonach i komponentach sprzętowych. W miarę poznawania Flowcode można tez wprowadzać bardziej zawansowane mechanizmy, na przykład przerwania i elementy języka C.

Ci wszyscy, którzy na co dzień piszą programy w języku C, muszą się przestawić na inną filozofię programowania. Z mojego doświadczenia wynika, że nie jest to trudne. Takie schematy blokowe jak w Flowcode rysuje się też (najczęściej na kartce) dla bardziej zaawansowanych projektów pisanych w językach wyższego rzędu – przynajmniej ja tak robię. Na **rys. 16** pokazano działający program czytający stan 4 styków i wyświetlający kod styku na wyświetlaczu. W trakcie rysowania uczyłem się od podstaw mechanizmów programowania, bo to było pierwsze moje zetknięcie z Flowcode. Wszystko to zajęło mi nie więcej niż 2...3 godziny, praca składała się głównie ze studiowania plików pomocy. Rysowanie bardziej ambitnego zadania zajęłoby więcej czasu, ale i tak szybkość tworzenia aplikacji w zupełnie nieznanym środowisku jest imponująca, co świadczy o dobrze przemyślanej idei całego przedsięwzięcia.

Jeżeli ktoś wyznaje zasadę, że nieważne jest w czym programujesz, byleby program działał dobrze i był napisany tak szybko, jak to tylko możliwe, to na pewno zainteresuje się Flowcode.

Jak na razie nie znam ani polskiego dystrybutora, ani ceny tego narzędzia. Ten ostatni element na pewno w naszych realiach warunkowałby jego powodzenie. Producent na swoich stronach internetowych oferuje dwie wersje dla mikrokontrolerów PIC: profesjonalną (tę testowałem) za 99 funtów i studencką/do użytku domowego za 39 funtów angielskich. Szczególnie ta druga opcja wydaje się atrakcyjna dla hobbystów i może stanowić potencjalne zagrożenie dla popularnego Bascoma

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



Rys. 16. Program czytający stan styków i wyświetlający kod klawisza