FlowCode i E-blocks (2)

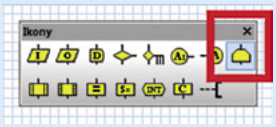
Obsługa klawiatury matrycowej, deklarowanie zmiennych, symulator

Nowoczesne języki programowania ewoluują w stronę, która umożliwi łatwe napisanie programu nie tylko elicie programistów, ale dosłownie każdemu. Pozwolą na to kompilatory graficzne, których aktualnie używają z powodzeniem zarówno profesjonaliści z różnych dziedzin, jak i… dzieci programujące klocki Lego Mindstorm. Przykładem środowiska programistycznego przeznaczonego równie dobrze dla profesjonalistów, jak i amatorów, służącego do programowania graficznego różnych rodzin mikrokontrolerów, jest produkt brytyjskiej firmy Matrix Multimedia – FlowCode. W tej części kursu pokażemy, w jaki sposób można obsłużyć klawiaturę matrycową, porty wejścia/wyjścia, zdefiniować zmienną oraz nauczymy się korzystania z symulatora.

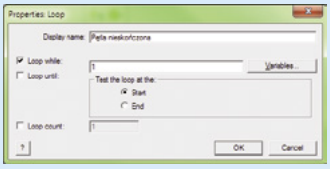
Podstawową częścią niemal każdego interfejsu użytkownika jest klawiatura – wirtualna lub wykonana na bazie zestyków mechanicznych. Dla potrzeb tego ćwiczenia zastosowałem moduł EB014-00-1 dostępny w ofercie Matriksa. Zawiera on typową klawiaturę matrycową przylutowaną do płytki drukowanej ze złączem DSUB9 oraz sieciami rezystorowymi. Aby łatwo można było zademonstrować jej działanie, zdecydowałem się na użycie modułu z diodami świecącymi – EB004-00-2.

Obsługa klawiatury matrycowej

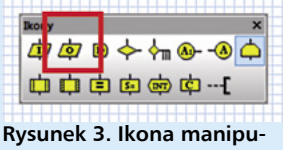
Klawiatura matrycowa składa się z łączników mechanicznych połączonych w wiersze i kolumny – łącznik jest umieszczany na przecięciu się kolumny i wiersza. Moduł EB014 ma klawiaturę składającą się z 3 kolumn i 4 wierszy. Łatwo policzyć, że jest dostępne 12 klawiszy (3×4), od „0” do „9” oraz symbole gwiazdki i krzyżyka. Kolumny są dołączone do złącza DSUB9 w taki sposób, że odpowiadają bitom 0..2, natomiast wiersze 4…7.



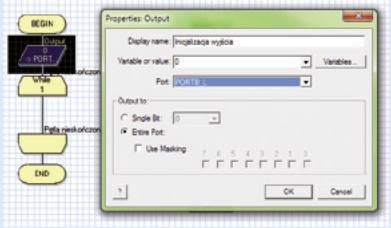
Rysunek 1. Ikona pętli *Loop*



Rysunek 2. Okno właściwości pętli



Rysunek 3. Ikona manipulowania wyjściami *Outputs*



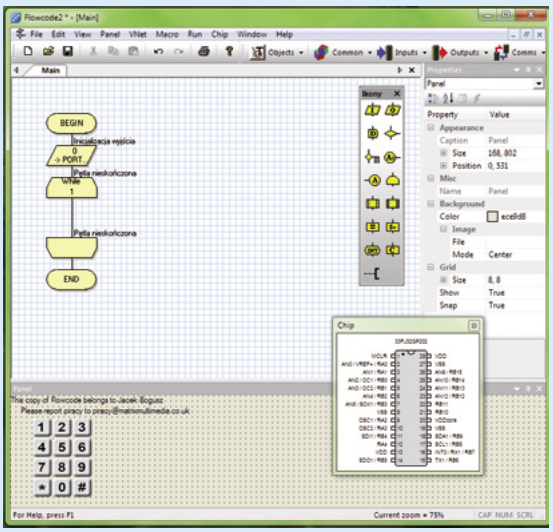
Rysunek 4. Umieszczenie pola *Output* na diagramie programu

FlowCode ma gotowe makro służące do obsługi klawiatury matrycowej. Można je znaleźć w menu *Inputs -> Keypad*. Po kliknięciu na tę pozycję menu klawiatura pojawia się na panelu. Następnie, znanym nam już sposobem, trzeba dołączyć klawiaturę do mikrokontrolera: klikamy prawym przyciskiem myszy na klawiaturze i z menu kontekstowego wybieramy *Connections*. Dokładnie omówię sposób jej dodania przy okazji opisu wykonania programu.

Dla potrzeb tego ćwiczenia wybrałem mikrokontroler dsPIC33FJ32GP202 w obudowie 28-nóżkowej. Jedynymi kryteriami były – dostępność w szufladzie i odpowiednia liczba wyprowadzeń I/O. Klawiaturę (moduł EB014) dołączyłem do portu *BH*, natomiast płytkę z diodami LED (moduł EB004) do portu *BL*. Jak płytkę nadrzędną zastosowałem moduł *up-stream* EB064-00-2. Całość zasiliłem z portu USB komputera PC – w tym celu zworkę J1 (PSU/USB) ustawiłem w pozycji USB.

Ewentualnego przełączenia wymagają jeszcze zwory J16…J18 (*PICKIT/USB*), J5…J7, J19 (*PICKIT/USB*) oraz J8 (*A*), J9 (*B*), J15 (*C*). Zworki *A…C* noszą nazwę *Programming Pins*. W związku z tym, że używam programatora wbudowanego na płytce drukowanej, to zworki J16…J18 ustawiłem w pozycji *USB*. Mimo iż noszą one odrębne oznaczenia, to umieszczono je w jednym rzędzie i są one łączone za pomocą 4-pozycyjnego mostka, a więc wszystkie jednocześnie, bez możliwości „przekombinowania”. Aby móc programować mikrokontroler dsPIC33FJ32 należy zewrzeć parę zworek J9 (pozycja *B*). Na koniec należy ustawić zwory J16…J18 w pozycji *VRAIL*. Teraz można umieścić mikrokontroler w podstawce U2 oraz dołączyć płytkę za pomocą kabla USB do komputera PC.

Ćwiczenie 1: obsługa klawiatury matrycowej



Rysunek 5. Symbol klawiatury matrycowej w obszarze *Panel*

Aby wykonać program, z menu wybieramy opcję *File -> New*. Z okienka *Choose a Target* wybieramy mikrokontroler, który jest używany. Oczywiście, do ćwiczenia można wykorzystać dowolnego PIC'a, nie musi to być identyczny jak mój, ale trzeba zwrócić uwagę na liczbę dostępnych wyprowadzeń portów. Do obsługi klawiatury jest wymagane 7 linii I/O, natomiast diod LED – 8 linii I/O. Jest nam więc potrzebny co najmniej jeden pełny port, np. port *B* w wypadku użycia dsPIC33GP202 (15 bitów z 16-bitowego portu).

Po wskazaniu mikrokontrolera i kliknięciu *OK*, zostanie wyświetlony ekran roboczy zawierający obszary *Panel*, *Properties* (właściwości), *Chip* (rozmieszczenie wyprowadzeń mikrokontrolera) oraz arkusz roboczy (*Main*). Na arkuszu roboczym zostaną wyświetlone symbole *BEGIN* i *END* oznaczające początek i koniec aplikacji użytkownika – pomiędzy nimi będziemy umieszczali czynności wykonywane przez program.

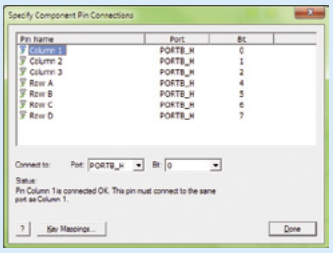
Program z tego ćwiczenia będzie działał typowo: instrukcje zostaną umieszczone wewnątrz pętli nieskończonej, w której będzie odczytywana klawiatura i zależnie od wciśniętego klawisza będą zaświecane odpowiednie diody LED. Naciśnięcie klawiszy 1…9 będzie powodowało odpowiednie zaświecenie diod LED0…LED8.

Najczęściej programiści piszący w językach wysokiego poziomu do tworzenia pętli nieskończonych używają instrukcji pętli warunkowych z warunkiem zatrzymania, który nigdy nie zostanie spełniony. Pomiędzy początek a koniec takiej pętli wstawia się instrukcje wykonywane przez mikrokontroler lub umieszcza ją na końcu aplikacji po to, aby mikrokontroler nie „poszedł w maliny" po zakończeniu wykonywania programu głównego. Często robi się tak „na wszelki wypadek", pomimo np. umieszczenia na końcu programu instrukcji *Power Down*. Pamiętajmy, że w niewielkim mikrokontrolerze najczęściej nie ma systemu operacyjnego, który byłby w stanie przejąć kontrolę nad CPU po zakończeniu wykonywania programu głównego i dlatego to programista sam musi zadbać o sposób zakończenia aplikacji. W języku C najczęściej do tworzenia pętli nieskończonych są używane instrukcje *while(1)* lub *for(;;)*.

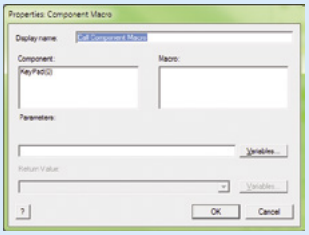
We FlowCode różne pętle są dostępne pod symbolem *Loop* pokazanym na **rysunku 1**. Ten symbol należy przenieść metodą *przeciągnij i upuść* na linię pomiędzy *BEGIN* a *END* na arkuszu roboczym. Po umieszczeniu symbolu na arkuszu, klikamy na nim dwukrotnie w celu wyświetlenia okna właściwości (**rysunek 2**). Dla potrzeb tego przykładu posłużymy się pętlą *while(1)* wykonywaną po spełnieniu warunku, który jest spełniony zawsze – oznacza to „1" będąca argumentem pętli. Dla potrzeb tego przykładu w okienku *Properties: Loop* należy wpisać identyczną zawartość, jak pokazana na rys. 1.

Moduł EB-004 z diodami LED jest dołączony do portu *BL*. Zaświecenie diody wymaga poziomu wysokiego na wejściu modułu, zgaszenie poziomu niskiego. Przed rozpoczęciem obsługi diod warto zainicjować port sterujący na wypadek, gdyby pomiędzy odczytem klawiatury a początkiem programu mikrokontroler miał wykonać jeszcze jakieś czynności.

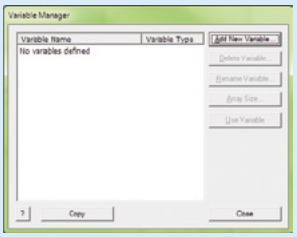
We FlowCode dioda LED jest wśród komponentów wyjściowych (*Outputs*) i można ją ułożyć na panelu obok klawiatury, dołączyć do odpowiedniego wyprowadzenia mikrokontrolera za pomocą kliknięcia prawym klawiszem myszy i wyboru *Connections*, a następnie obsługiwać z użyciem *Component Macro*, identycznie jak klawiaturę czy inne urządzenia z grup *Inputs* lub *Outputs*. Do dyspozycji są dwie instrukcje, z których jedna zaświeca (*LedOn*), a druga gasi (*LedOff*) diodę LED. Ja jednak zdecydowałem się na manipulowanie całymi wartościami portów, zamiast pojedynczymi bitami. W tym celu posłużyłem się ikoną *Output* (**rysunek 3**). Posługując się techniką *przeciągnij i upuść* umieszczamy ją pomiędzy polem *START* a symbolem pętli *While* (**rysunek 4**). Teraz możemy już zająć się klawiaturą. W menu *Inputs* na pasku urządzeń wskazujemy komponent *KeyPad*. Zostanie on umieszczony w oknie *Panel*, co widać na dolnej części **rysunku 5**. Klawiaturę trzeba dołączyć do odpowiednich wyprowadzeń mikrokontrolera. W tym celu klikamy prawym klawiszem na symbolu klawiatury w oknie *Panel* i wybieramy *Connections*. Zostanie wyświetlone okno, jak na **rysunku 6**. Wybierając w górnej części okna odpowiednie wyprowadzenie (np. *Column 1*, *Row A*) dołączamy je za pomocą dolnej linijki, wybierając odpowiedni port i numer bitu. Dla potrzeb tego przykładu, trzeba wprowadzić nastawy zgodne z rys. 6 i nacisnąć klawisz *Done*.



Rysunek 6. Okno połączeń klawiatury matrycowej

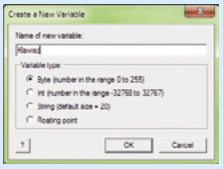


Rysunek 8. Okno właściwości makra obsługi komponentu



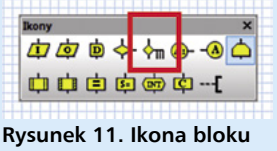
Rysunek 9. Okno menadżera zmiennych *Variable Manager*

Klawiatura znajduje się w grupie urządzeń wejścia/wyjścia, a więc jest obsługiwana za pomocą makra *Component Macro*. Klikając na ikonie pokazanej na **rysunku 7** umieszczamy makro tuż za instrukcją *While* i dwukrotnie klikamy na jego symbolu. Otworzy się okno, jak na **rysunku 8**. Dla ułatwienia wpisujemy w nim nazwę symboliczną (*Display name* np. Odczyt klawiatury), klikamy na nazwie komponentu (pole *Component*) *KeyPad(0)* oraz w polu *Macro* na nazwie *GetKeyPadNumber*. W dolnej części okna, nad pustą linią, pojawił się komunikat *Return Value: (BYTE)*, co oznacza, że funkcja zwraca jednobajtową wartość, którą należy przechować w celu dalszej oceny. Dla tego przeznaczenia trzeba ją zapamiętać w zmiennej w pamięci operacyjnej RAM.

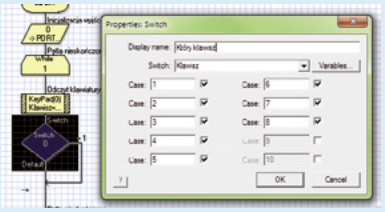


Rysunek 10. Okno właściwości zmiennej *Klawisz*

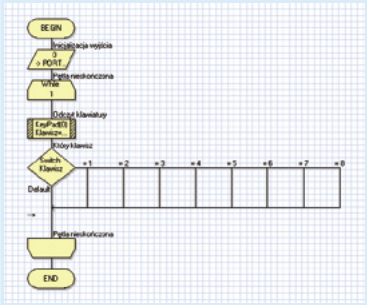
Zmienne deklaruje się po kliknięciu na przycisk *Variables*. Zostanie wyświetlone okno *Variable Manager*, jak na **rysunku 9**. Następnie klikamy na przycisk *Add New Variable* i wpisujemy nazwę zmiennej – ja posłużyłem się nazwą *Klawisz*. Nowe okienko o nazwie *Create New Variable* umożliwia nam również określenie typu zmiennej – jak pamiętamy funkcja *GetKeyPadNumber* zwraca zmienną typu bajt (*Byte*), więc wybrałem zmienną o tym samym zakresie (**rysunek 10**). Po kliknięciu na *OK* wskazujemy nazwę nowoutworzonej zmiennej na liście i klikamy na klawisz *Use Variable*. Nazwa zmiennej pojawi się w pustej linii okna z rys. 8. Następnie klikamy na *OK* i na tym możemy uznać proces tworzenia obsługi klawiatury za zakończony. Teraz trzeba zająć się rozpa-



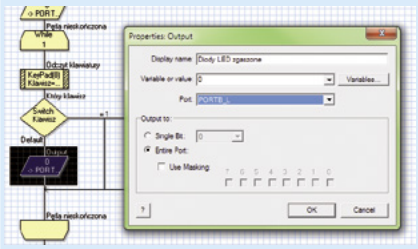
Rysunek 11. Ikona bloku warunkowego *Switch*



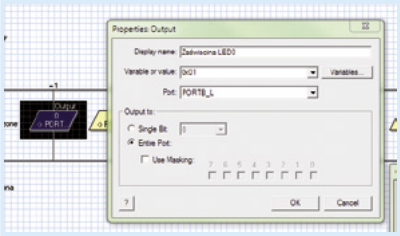
Rysunek 12. Okno właściwości bloku *Switch*



Rysunek 13. Poprawnie skonfigurowany blok *Switch* umieszczony w obszarze roboczym



Rysunek 14. Umieszczenie instrukcji wykonywanych domyślnie

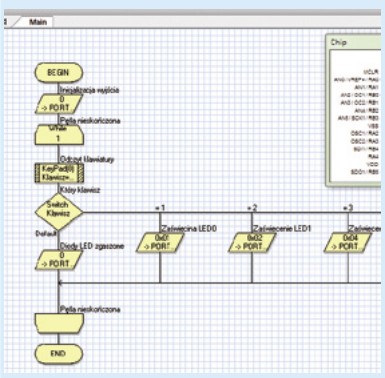


Rysunek 15. Umieszczenie instrukcji wykonywanych dla warunków =1, =2 itd.

trywaniem wartości zmiennej *Klawisz*.

Do rozpatrywania wartości zmiennych i podejmowania akcji zależnie od niej w typowych językach programowania służą instrukcje warunkowe. W języku C są to instrukcje *if* oraz *switch … case*. FlowCode ma oba te mechanizmy, jednak dla uproszczenia programu ten drugi wydaje się być bardziej atrakcyjny, ponieważ zamiast pojedyncze „rozważania" zajmujące sporo miejsca na ekranie zostają zgrupowane w czytelny blok. Z paska wybieramy ikonę *Switch* (**rysunek 11**) i przeciągamy ją za makro odczytujące klawiaturę. Następnie klikamy dwukrotnie na symbolu *Switch* i zaznaczamy liczbę rozpatrywanych warunków – w naszym wypadku będzie ich aż 8 (równoważne 8 diodom LED). Za pomocą przycisku *Variables* wskazujemy zmienną *Klawisz* jako tę rozpatrywaną. Okno właściwości *Switch* powinno wyglądać jak na **rysunku 12**, natomiast po kliknięciu *OK* powinien się pokazać obraz, jak na **rysunku 13**. Łatwo zauważyć, że nad poszczególnymi odnogami umieszczonymi z boku pola *Switch* są umieszczone liczby (=1, =2, =3 itd.) oznaczające, dla jakiego warunku nastąpi przejście daną ścieżką. Jeśli dla przykładu zostanie spełniony warunek *Klawisz = 1*, to CPU wykona instrukcje umieszczone w ścieżce „=1", jeśli *Klawisz = 2*, to CPU wykona instrukcje umieszczone w ścieżce „=2" itd. Liczba odnóg oraz rozpatrywane wartości zależą od „ptaszków" zaznaczonych w oknie *Switch*.

Jak pamiętamy, program ma zaświecać diodę zależnie od numeru wciśniętego klawisza. Domyślnie diody są zgaszone i są zaświecane za pomocą przyciśnięcia klawisza. Jak pierwsze wykonajmy więc instrukcje realizowane w wypadku domyślnym – *Default*. Odpowiedni blok modyfikujący port *BL* należy umieścić pod polem *Switch*, jak na **rysunku 14**. Następnie należy umieścić na odpowiednich liniach pozostałe akcje, analogicznie jak na **rysunku 15**, zmieniając jedynie wartości wpisywane w polu *Variable or value* w taki sposób, że LED0 odpowiada 0x01, LED1 – 0x02, LED2 – 0x04, … , LED7 – 0x80. Ja posłużyłem się liczbami szesnastkowymi, jednak można wpisać liczby dziesiętne lub inne, np. w formacie binarnym, używając formatu akceptowanego przez język C. Fragment gotowego programu pokazano na **rysunku 16**. W oknie *Panel* można (korzystając z paska urządzeń) umieścić diody LED i dołączyć je do wyprowadzeń *Port BL* 0…7 mikrokontrolera. Wówczas, podczas symulowania pracy programu, uzyskamy efekt „zaświecających się" wirtualnych diod LED. Sposób umieszczenia i dołączenia diod jest taki sam, jak klawiatury i nie będę go opisywał, pozostawiając tę czynność samodzielnej inwencji Czytelnika. Będzie to dobry test na czytanie ze zrozumieniem, a pomocnym w uzyskaniu pozytywnych rezultatów może być **rysunek 17**.



Rysunek 16. Fragment gotowego programu

Symulator programowy

Działanie nowoutworzonego programu można przetestować za pomocą symulatora. Jeśli połączenia zostały zdefiniowane poprawnie, to symulator będzie pracował jak rzeczywiste urządzenie, klawisze można będzie „naciskać", a diody LED będą odpowiednio „zaświecały się".

Symulator uruchamia się za pomocą menu *Run*. Po jego wybraniu dostępne są następujące opcje:

- *Go/Continue* (klawisz *F5*) służąca do uruchomienia programu lub wznowienia wykonywania przerwanego za pomocą *Pause*.
- *Step Into* (klawisz *F8*) powodująca wykonanie pojedynczego kroku.
- *Step Over* (klawisze *Shift+F8*) powodująca zaniechanie wykonania makro czy instrukcji.
- *Pause* (klawisz *F7*) wstrzymująca wykonywanie programu.
- *Stop* (klawisze *Shift+F5*) przerywająca wykonywanie programu.

Obserwowanie pracy naszego programu krok po kroku jest możliwe po kolejnych naciśnięciach F8. Jeśli nie chcemy czekać np. na odbiór danych, to należy nacisnąć klawisze Shift+F8, co spowoduje przejście do kolejnego bloku, bez wykonywania instrukcji zawartych w aktualnie podświetlonym. Praca programu jest sygnalizowana za pomocą czerwonej, przemieszczającej się ramki, widocznej jedynie w trybie pracy krokowej (F8, Shift+F8) lub po wstrzymaniu pracy programu (F7). Oprócz samej ramki w okienku *Variables* można wyświetlić podgląd stanu zmiennych – w tym wypadku jest pokazywana wartość zmiennej *Klawisz*. Zmienne do okienka można dodawać za pomocą wskazania, kliknięcia prawym klawiszem myszy i wybrania w menu kontekstowym opcji *Add variable*. Po jej aktywowaniu ukaże się znane nam z wcześniejszego opisu okno menedżera, w którym trzeba wskazać zmienną i kliknąć *Use variable*. Efekt pracy symulatora programowego pokazano na **rysunku 18**. Stan wyprowadzeń mikrokontrolera jest sygnalizowany nie tylko za pomocą wirtualnych diod LED, ale również poprzez zmianę koloru wyprowadzeń na symbolu mikrokontrolera umieszczonym w okienku *Chip*. Wyprowadzenia wyzerowane mają kolor niebieski, natomiast te ustawione – kolor czerwony.

Konfigurowanie mikrokontrolera

Jedną z najtrudniejszych czynności przy programowaniu we FlowCode jest konfigurowanie mikrokontrolera. Nie jest ono trudne ze względu na niedociągnięcia kompilatora, ale z powodu liczby dostępnych opcji oraz filozofii przyjętej przez firmę Microchip. Prawidłowe skonfigurowanie mikrokontrolera i jego peryferiów wymaga wiedzy, której nie można nabyć inaczej, jak czytając dokumentację techniczną mikrokontrolera lub posługując się innymi opracowaniami, jak chociażby artykuły z Elektroniki Praktycznej.

Okno konfigurowania mikrokontrolera dsPIC-33FJ32GP202 pokazano na **rysunku 19**. Za jego pomocą programista może określić, co stanie się z segmentem bootloadera, który oscylator będzie używany do taktowania mikrokontrolera i w jaki sposób, jak będzie pracował Watchdog, jak będą mapowane wyprowadzenia mikrokontrolera, czy będzie używany JTAG i wiele, wiele innych opcji. Niestety, o ile da się wytłumaczyć znaczenie poszczególnych nastaw dla danego mikrokontrolera lub rodziny mikrokontrolerów, o tyle trudno jest podać jakąś uniwersalną receptę dla wszystkich PIC-ów. Co prawda kompilator w momencie tworzenia nowego programu wprowadza domyślne ustawienia, jednak nie zawsze będą one dobre dla naszego programu.

Błędy i poprawki

Jakież było moje zdziwienie, gdy okazało się, że program pracujący prawidłowo w symulatorze nie działa po przesłaniu do mikrokontrolera. Analiza kodu w języku C wykazała, że niestety – twórcy FlowCode nie ustrzegli się błędów.

Moduł klawiatury i moduł diod LED są tak wykonane, że przy użyciu mikrokontrolera w obudowie 28-nóżkowej nie da się dołączyć klawiatury do portu *BL*, natomiast diod do portu *BH*. Uniemożliwia to kształt płytki drukowanej modułu klawiatury EB014. Liczne eksperymenty wykazały, że klawiatura dołączona do „niższej" połowy portu (bity 0…7) funkcjonuje prawidłowo, natomiast po dołączeniu do bardziej znaczącej połowy portu (bity 8…15) funkcja jej obsługi nieodmiennie zwraca 0xFF. Doświadczenie podpowiedziało mi, że winny jest zakres zmiennych. Łatwo powiedzieć, trudniej odnaleźć błąd.

Próba zmiany typu zmiennej *Klawisz* nie dała rezultatu. Pozostało „dokopanie się" do funkcji obsługi klawiatury, aby zobaczyć co tam się dzieje. Wyświetlenie prototypu funkcji obsługi jest możliwe po kliknięciu prawym klawiszem myszy na symbolu klawiatury w oknie *Panel* i wybraniu opcji *Custom Code*. Następnie wskazujemy makro *GetKeypadNumber* i klikamy na przycisk *Edit Code*. Jednak wcześniej, za pomocą *Chip -> View C* warto zajrzeć, która linia będzie aktywna w nieco przydługim fragmencie kodu kompilowanym warunkowo, więc teraz zamykamy wyświetlone okno.

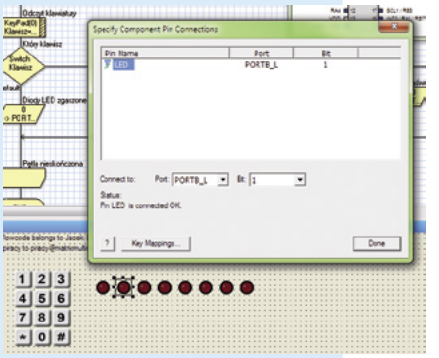
Po wyświetleniu kodu (ja mam w tym celu skonfigurowany Notepad+) odszukujemy ciało funkcji *FCD_KeyPad0_GetKeypadNumber()*. Fragment programu za dyrektywą *#if* jest wykonywany tylko w wypadku spełnienia warunku. Wśród licznych znajdujemy oczywisty *4==4*. Odpowiadają mu deklaracje:

```
#if (4 == 4)
  #define KPAD_ROW_MTX
{4096,8192,16384,32768}
  #define KPAD_ROW_MASK
(4096|8192|16384|32768)
#endif
```

Znając język C łatwo domyślić się, że ujęcie liczb w nawias klamrowy sugeruje ich użycie do inicjowania tablicy. Można też zorientować się, że liczby z zakresu 4096…32768 ujętego w klamry wymagają zmiennych 16-bitowych. Analizując dalszą część programu zauważamy, że liczby są wstawiane do tablicy zawierającej zmienne typu *char*, a więc 8-bitowe:

```
const char mtxKeysAsNumbers[] =
{1,4,7,10,2,5,8,0,3,6,9,11};
const char mtxCols[] =
KPAD_COL_MTX;
const char mtxRows[] =
KPAD_ROW_MTX;
```

To ewidentny błąd twórców programu. Owszem, klawiatura dołączona do mikrokontrolera 8-bitowego lub mniej znaczącej połowy portu 16-bitowego będzie działała prawidłowo, ale w wypadku takim, jak w naszym przykładzie – nie ma szans! Uzbrojeni w tę wiedzę wracamy do edycji makra obsługi klawiatury.

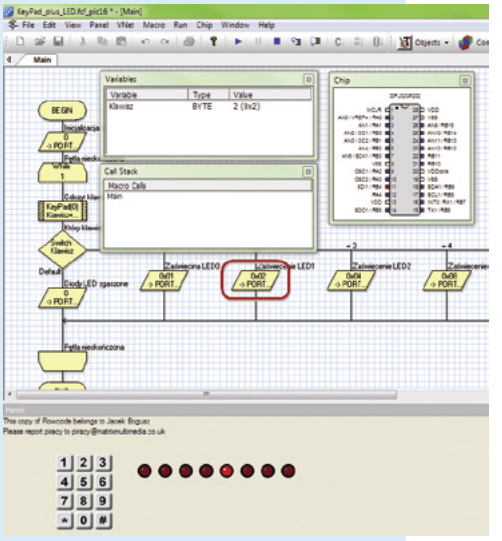


Rysunek 17. Okno własności komponentu LED

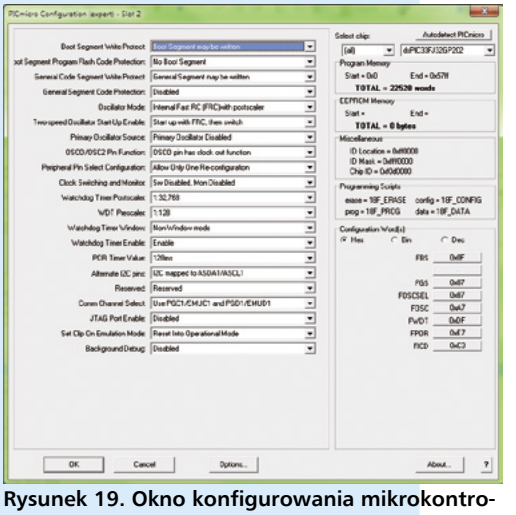
Klikamy prawym klawiszem myszy na klawiaturze, wybieramy *Custom Code*, następnie makro *GetKeypadNumber* oraz *Edit Code*. Odszukujemy wspomniane deklaracje i zmieniamy na:

```
const unsigned int
mtxKeysAsNumbers[]
= %b;
const unsigned int
mtxCols[] = KPAD_
COL_MTX;
const unsigned int
mtxRows[] = KPAD_
ROW_MTX;
```

Następnie kompilujemy program i wgrywamy go do pamięci mikrokontrolera. Od tego momentu możemy cieszyć się prawidłowo funkcjonującym kodem.



Rysunek 18. Efekt działania symulatora



Rysunek 19. Okno konfigurowania mikrokontrolera dsPIC33FJ32GP202

Podsumowanie

Niestety, jak udowodniło to proste ćwiczenie, każdy program to konstrukcja logiczna i jako taka zawiera błędy. Na szczęście pakiet FlowCode ma mechanizmy, które umożliwiają łatwą analizę kodu i korektę zauważonych niedociągnięć. Mam nadzieję, że w wersji 5 pakietu poprawiono opisane błędy w bibliotece (kurs opiera się na FlowCode v4). Mniej doświadczeni użytkownicy w wypadku napotkania problemów, takich jak opisywany wyżej, mogą napisać do inżynierów wsparcia technicznego lub zadać pytanie na forum.

Jacek Bogusz, EP

Redakcja Elektroniki Praktycznej dziękuje firmie TME z Łodzi, dystrybutorowi firmy Matrix Multimedia, za udostępnienie zestawu E-blocks oraz mikrokontrolerów używanych w kursie programowania FlowCode.