FlowCode i E-blocks (5)

Przykładowy projekt zegarka z wyświetlaczem LED

W poprzednim odcinku cyklu opisaliśmy podstawy multipleksowania wyświetlaczy LED. Teraz nauczymy się, jak wykorzystać tę umiejętność w praktyce do wyświetlania wartości liczby. Posłużymy się przykładem nieskomplikowanego programu zegara czasu rzeczywistego, odmierzającego czas z użyciem systemu przerwań mikrokontrolera. Wspomniane przerwanie będzie współistniało z użytym wcześniej przerwaniem obsługi wyświetlacza.

Zegarek to aplikacja najchętniej wykonywana przez początkujących adeptów sztuki programowania. Jeśli do jego budowy zastosujemy moduł wyświetlacza LCD, to wykonanie oprogramowania jest stosunkowo łatwe, ponieważ ten moduł wymaga jedynie przesłania danych do wyświetlenia. Nieco gorzej jest z wyświetlaczem LED, którego obsługą od ogółu do detalu musi zająć się tworzone przez nas oprogramowanie. Niemniej, jego wykonanie za pomocą FlowCode jest bajecznie łatwe.

Ćwiczenie 3: Zegar z wyświetlaczem 4-cyfrowym

Sposób włączenia i sterowania multipleksowanych wyświetlaczy LED był omówiony poprzednio. Teraz proponuję takie zmodyfikowanie programu, aby powstała użyteczna aplikacja zegara. W związku z tym, że najbardziej zależy mi na opisie sposobu funkcjonowania systemu przerwań oraz zaprzęgnięcia go do pracy, zdecydowałem się na wykonanie bardzo uproszczonego sposobu wprowadzania nastawy czasu, dobrze znanego z dalekowschodnich radiobudzików. Jeśli na wejściu PORTA_L(0) występuje poziom wysoki, to zegar wchodzi do trybu nastaw. Jeśli na doprowadzeniach PORTA_L(1) i PORTA_L(2) występują poziomy wysokie, to – odpowiednio – zwiększane są godziny lub minuty. Tak więc ustawienie czasu wymaga naciśnięcia i przytrzymania przycisku dołączonego do wejścia PORTA_L(0) – nazwijmy go umownie SW0 – i jednoczesnego naciskania (trzymając wciśnięty SW0) pozostałych dwóch – nazwijmy je SW1 (ustawianie godzin) i SW2 (ustawianie minut). Po zwolnieniu przycisku SW0 licznik sekund jest zerowany, a zegar rozpoczyna odmierzanie czasu.

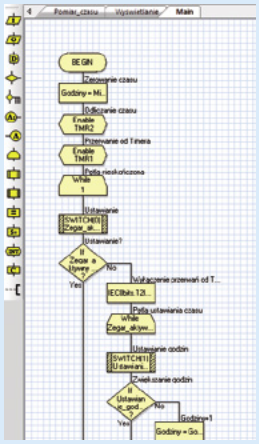
Do wykonania aplikacji z tego ćwiczenia użyłem płytek EB0064-00-2 (bazowa z mikrokontrolerem) oraz EB0008-00-1 (wyświetlacz 4-cyfrowy, 7-segmentowy LED). Na płytce bazowej zainstalowałem mikrokontroler dsPIC33FJ32GP202. Jego wybór był podyktowany liczbą wyprowadzeń I/O oraz dostępnością w przysłowiowej szufladzie. Normalnie do wykonania zegarka oczywiście nie potrzeba aż tak szybkiego mikrokontrolera.

Do portu AL dołączyłem przyciski przylutowane do złącza DSUB9. Płytkę z wyświetlaczami dołączyłem do złącz PORT_BL i PORT_BH. Oprócz tego połączenia, należy za pomocą odrębnego przewodu dołączyć zasilanie od terminatora J1 na płytce bazowej (styku oznaczonego +14 V) do terminatora J3 na płytce wyświetlacza (do styku oznaczonego jako +V). Jak poprzednio, całość zasiliłem z zasilacza stabilizowanego dostarczającego napięcie 12 V przy obciążeniu do 1 A. Mikrokontroler jest taktowany sygnałem generowanym na bazie zewnętrznego rezonatora kwarcowego o częstotliwości 12 MHz.

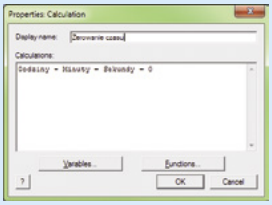
Tworzenie programu można rozpocząć od wybrania z okienka dialogowego opcji *Create a new FlowCode chart* lub modyfikując poprzednio wykonany program do obsługi wyświetlacza LED. W tym drugim wypadku, najlepiej aby „stary program" zapisać nadając mu nową nazwę. Ja zapamiętałem go jako „Zegar_Led7seg4" i pod tą nazwą można znaleźć go wśród materiałów dodatkowych na płycie CD. Jest to rozwiązanie łatwiejsze, unikniemy potrzeby tworzenia od nowa procedury obsługi wyświetlania.

Na początek, jeśli tworzymy program od nowa, na panelu należy ułożyć wyświetlacz LED (komponent *led7seg4* z menu *Outputs*) i dołączyć go w sposób opisany wcześniej, w odcinku kursu nt. multipleksowania wyświetlacza.

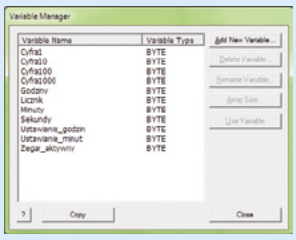
Zacznijmy od zmian, które musimy wykonać na zakładce *Main*. Pierwsze pole, umieszczone tuż za *BEGIN* (**rysunek 1**) należy zmienić w sposób pokazany na **rysunku 2**. Wcześniej, za pomocą przycisków *Variables* oraz *Add New Variable…* należy utworzyć zmienne: *Godziny, Minuty, Sekundy*. Warto przy okazji założyć zmienne *Zegar_aktywny, Ustawianie_godzin* i *Ustawianie_minut*, którymi posłużymy się przy odczycie położenia przełączników służących do ustawiania zegara. Zmienne: *Cyfra1, Cyfra10, Cyfra100,*



Rysunek 1. Zakładka *Main* – początek programu głównego

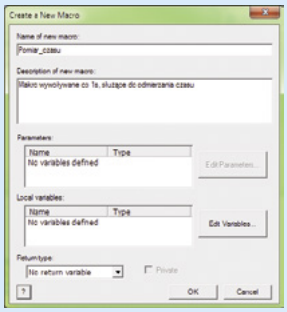


Rysunek 2. Zerowanie wskazań czasu za pomocą instrukcji języka C

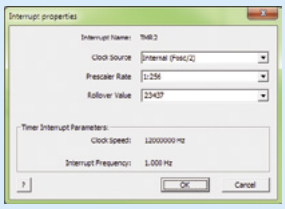


Rysunek 3. Lista zmiennych używanych w programie (nie obejmuje zmiennej lokalnej Kropka_dzies)

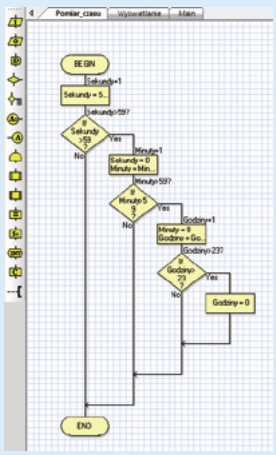
Rysunek 4. Własności przerwania Timera 2 służącego do odmierzania czasu (przerwanie co 1 sekundę)



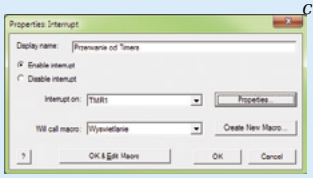
Rysunek 5. Parametry makra służącego do odmierzania czasu



Rysunek 6. Parametry Timera 2 (źródło sygnału, preskaler, wartość odświeżająca rejestr timera)



Rysunek 7. Wygląd procedury służącej do odmierzania czasu (zakładka *Pomiar_czasu*)



Rysunek 8. Własności przerwania Timera 1 służącego do obsługi wyświetlacza LED

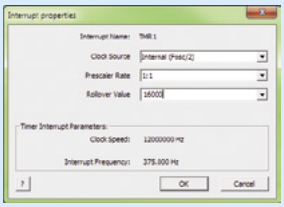
Cyfra1000 oraz *Licznik* powinny nam pozostać z poprzedniego ćwiczenia. Kompletną listę zmiennych pokazano na **rysunku 3**.

W związku z tym, że diagram Flow Code jest tłumaczony na język C, to są akceptowane niektóre wyrażenie z tego języka, takie jak *Zmienna_1 = Zmienna_2 = Wartość*. Oczywiście w pokazanej sytuacji obu zmiennym zostanie nadana ta sama wartość np. początkowa, która może być 0.

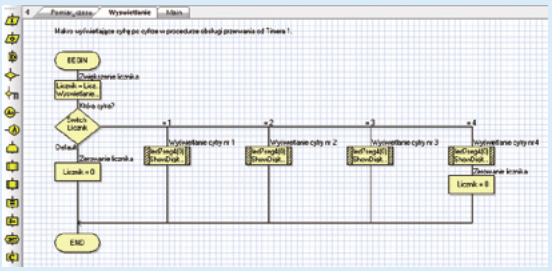
Do odmierzania czasu użyto przerwań generowanych przez Timer 2. Identycznie jak w poprzednim przykładzie, Timer 1 jest używany do wywoływania procedury wyświetlania cyfr. Tuż za polem zerowania zmiennych czasu, ikonę *INT* przeciągniętą na diagram z paska narzędziowego. Po dwukrotnym kliknięciu na symbolu przerwania, wypełniamy pole parametrów przerwania zgodnie z **rysunkiem 4**, jednak zanim wybierzemy makro wywoływane przez przerwanie (pole *Will call macro:*), należy je utworzyć przez kliknięcie na przycisku *Create New Macro…* Puste okienko wypełniamy zgodnie z **rysunkiem 5**. W efekcie, po zamknięciu okienka własności przerwania poprzez kliknięcie na przycisk *OK & Edit Macro*, zostanie utworzona nowa zakładka, o nazwie wpisanej w polu *Name of new macro:*.

Tymczasem wybieramy nowoutworzone makro jako parametr pola *Will call macro:* i naciskamy klawisz *Properties*. Okienko *Interrupt properties* należy wypełnić zgodnie z **rysunkiem 6** – *Clock Source* i *Prescaler Rate* wybieramy z listy *combo*, wartość *Rollover Value* wpisujemy ręcznie, tak aby częstotliwość wywoływania przerwania wynosiła 1 Hz. Pamiętamy, że częstotliwość taktowania rdzenia mikrokontrolera wynosi 12 MHz. Stąd częstotliwość wywoływania przerwania: 12 MHz/2/256/23437=1 Hz.

Klikamy na *OK* zamykając okienko *Interrupt Properties*, a następnie na *OK & Edit Macro* tworząc nową zakładkę – w moim wypadku o nazwie *Pomiar_czasu*. Wygląd podprogramu umieszczonego na tym arkuszu pokazano na **rysunku 7**. Na początku, za pomocą pola *Calculations* zwiększamy liczbę sekund o 1. Niestety, nie można w tym wypadku użyć wygodnego zapisu z języka C (np. *Sekundy++*), ale należy wpisać całe wyrażenie Sekundy=Sekundy+1. Kolejne pole, odpowiednik dyrektywy *If*, służy do sprawdzenia czy liczba sekund przekroczyła 59. Jeśli tak, to jest zwiększana liczba minut, natomiast liczba sekund jest zerowana. Po zwiększeniu liczby minut jest wykonywane sprawdzenie



Rysunek 9. Parametry przerwania Timera 1 (częstotliwość wywołania 375 Hz)

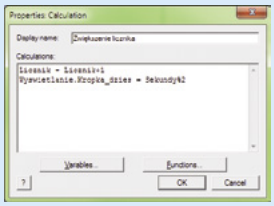


Rysunek 10. Wygląd procedury obsługi przerwania od Timera 1 (zakładka *Wyswietlanie*)

czy przekroczyła ona 59. Jeśli tak, to jest zwiększana liczba godzin, natomiast liczba minut jest zerowana. I znowuż, po zwiększeniu liczby godzin jest wykonywane sprawdzenie czy jest ona większa od 23. Jeśli tak, to liczba godzin jest zerowana. Łatwo zauważyć, że program do obsługi zegara pracuje w systemie 24-godzinnym.

Po wykonaniu podprogramu obsługi przerwania Timera 2, wracamy do zakładki *Main* i kontynuujemy tworzenie programu głównego. Jako kolejne pole umieszczamy na diagramie pole zezwalające na załączenie przerwania od Timera 1 – posłuży nam ono do obsługi wyświetlacza w sposób wytłumaczony uprzednio. Jeśli korzystamy z poprzedniego przykładu, to nie ma potrzeby konfigurowania przerwania i można pominąć lekturę tego akapitu. Po dwukrotnym kliknięciu na symbolu otworzy się okienko właściwości przerwania, które należy wypełnić jak na **rysunku 8**. Zakładka *Wyswietlanie* jest tworzona po kliknięciu na przycisk *Create New Macro…*, a następnie na *OK & Edit Macro*. Pole parametrów przerwania pokazano na **rysunku 9**. Jest ono wywoływane z częstotliwością 325 Hz, co powoduje odświeżanie pojedynczej cyfry z częstotliwością około 81 Hz, a więc co 12 ms. Przy takiej częstotliwości odświeżania, w normalnych warunkach użytkownik nie zauważy żadnego migotania.

Tworząc procedurę obsługi przerwania trzeba dodać zmienną globalną lub lokalną o nazwie Kropka_dzies. Ja utworzyłem zmienną lokalną procedury obsługi przerwania, tworzoną przy jego wywoływaniu i niszczoną tuż przed powrotem do programu głównego, stąd przed nazwą zmiennej jest umieszczona nazwa zakładki, do której ona przynależy. Ta zmienna posłuży nam ona do naprzemiennego zaświecania lub gaszenia segmentu kropki za drugą cyfrą wyświetlacza. W ten sposób będziemy mogli stwierdzić, że nasz zegar „chodzi" tj. pracuje w trybie odmierzania czasu, ale nie jej rola jest czysto „kosmetyczna" i nie potrzebujemy dostępu do niej w innych funkcjach. Oczy-



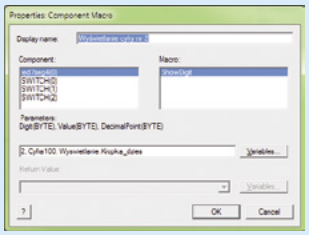
Rysunek 11. Operacje wykonywane na początku procedury wyświetlającej: zwiększenie licznika i obliczenie reszty z dzielenie licznika sekund

wiście, możemy też nie tworzyć nowej zmiennej i nie wykonywać migającej kropki, co uprości program i zmniejszy wielkość kodu wynikowego.

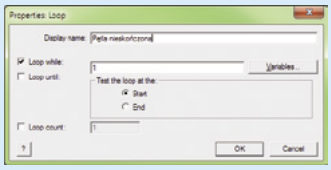
Całą procedurę obsługi wyświetlacza LED pokazano na **rysunku 10**. Poza zmianami dokonanymi na początku w polu *Calculations* i w trakcie rozpatrywania zmiennej Licznik w polu „=2”, nie wymaga ona jakiś większych zmian w porównaniu do tej opisywanej przy okazji omawiania sposobu wyświetlania multipleksowanego.

Na początku, tuż za polem *BEGIN*, jest umieszczone pole *Calculations*, za pomocą którego jest zwiększana wartość licznika oraz jest wyliczana zmienna Kropka_dzies (**rysunek 11**) jako reszta z dzielenie liczby sekund przez 2. Jak łatwo domyślić się, dla nieparzystej liczby sekund zmienna przyjmuje wartość 1, a dla parzystej 0. Po podstawieniu jako parametr wywołania makra *Wyświetlanie cyfry nr 2* (**rysunek 12**) odpowiada to zaświeceniu (1) lub zgaszeniu (0) segmentu kropki dziesiętnej.

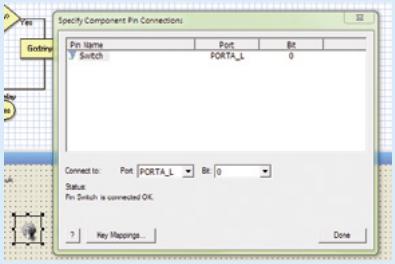
Teraz wracamy do zakładki *Main*, na której trzeba wykonać całą procedurę obsługi klawiszy nastaw. Program główny pracuje w pętli nieskończonej, w której oczekuje na przerwania od Timera 1 i Timera 2 oraz odczytuje stan przełącznika *Ustawianie* (SW0) podejmując odpowiednią akcję. Tworzenie dalszej części diagramu rozpoczynamy od ułożenia w obszarze panelu symboli przycisków. Trzy przełączniki wybieramy z paska *Inputs -> SWITCH* (czynność należy wykonać 3 razy). Następnie klikając prawym klawiszem na symbolu każdego przełącznika z menu kontekstowego wybieramy *Connections* i dołączamy do odpowiedniego doprowadzenia PORTA_L(0…2). Przykładowe połączenie zamieszczono na **rysunku 14**. Następnie z paska ikon przeciągamy symbol pętli *While*. Tworzymy pętlę nieskończoną – jej parametry pokazano na **rysunku 15**. Jako pierwszą instrukcję w pętli umieszczamy makro odczytu stanu przełącznika SW0, który możemy nazwać *Ustawianie/Praca*. Jego stan (0 wyłączony lub 1 aktywny) jest zapamiętywany w zmiennej Zegar_aktywny. Następne pole, warunkowe, sprawdza czy ta zmienna ma wartość „1” i jeśli tak, to przechodzi do dalszej części pętli, w której są wyliczane i podstawiane wartości poszczególnych cyfr wyświetlacza. Pole zawierające wspomniane obliczenia nazwałem *Konwersja* (**rysunek 16**). Stan pierwszej cyfry wyświetlacza jest zapamiętany w zmiennej Cyfra1000. Aby zegar wyświetlał tam dziesiątki godzin, należy ich liczbę (zmienna Godziny) podzielić przez 10. Stan drugiej cyfry określa zawartość zmiennej Cyfra100. Reprezentuje ona jednostki godzin i jest wyliczana jako reszta z dzielenia liczby godzin przez 10. Identycznie są wyliczane wskazania minut i wstawiane do zmiennych Cyfra10 i Cyfra1 (zmienna Minuty). Te obliczenia są wykonywane przy każdym przebiegu pętli głównej oraz podczas wykonywania nastaw.



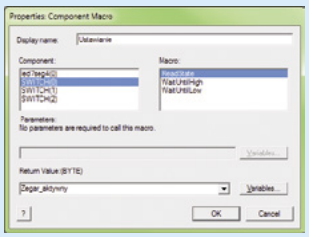
Rysunek 12. Zmodyfikowane okno wyświetlania cyfry numer 2 (=2)



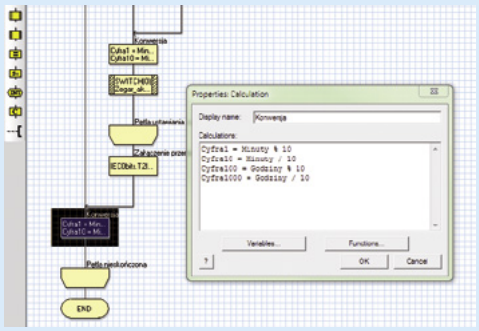
Rysunek 13. Parametry nieskończonej pętli głównej programu



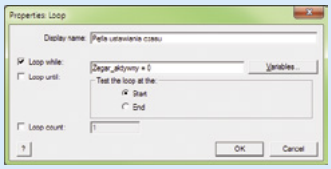
Rysunek 14. Przykładowy sposób dołączenia komponentu *SWITCH*



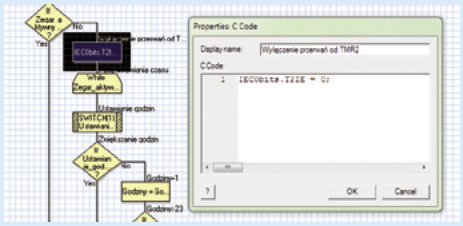
Rysunek 15. Makro zapamiętujące stan komponentu *SWITCH(0)* w zmiennej *Zegar_aktywny*



Rysunek 16. Wyznaczanie cyfr wyświetlanych przez zegar



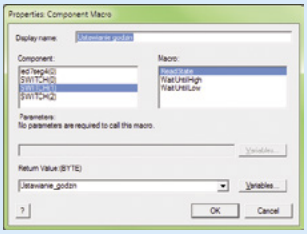
Rysunek 17. Własności pętli nastaw



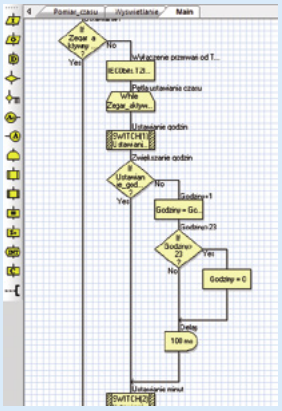
Rysunek 18. Wyłączenie przerwań Timera 2 za pomocą bloku instrukcji języka C

Jeśli przycisk SW0 jest włączony (zmienna Zegar_aktywny=0), to program wchodzi do pętli nastaw. Przypomnijmy, że pozostanie w niej wymaga trzymania przycisku wciśniętego, a ten powinien zewrzeć doprowadzenie PORTA_L(0) z masą. Pętla jest wykonywana dotąd, aż przełącznik SW0 zostanie zwolniony, tj. zmienna Zegar_aktywny przyjmie wartość 1. Sposób wpisania parametrów pętli pokazano w okienku na **rysunku 17**. Na początku pętli są blokowane przerwania Timera 2 służące do odmierzania czasu. Tu występuje pewna niedogodność, o której będzie mowa dalej. Przerwania są blokowane przez wyzerowanie bitu odpowiedzialnego za ich wywoływanie, co zrobiono za pomocą instrukcji języka C. Ikonę oznaczoną „C” należy przeciągnąć z paska ikon na diagram, a po dwukrotnym kliknięciu wpisać instrukcję pokazaną na **rysunku 18**.

W kolejnym kroku jest odczytywany stan przycisku SW1 (ustawianie godzin) i zapamiętywany w zmiennej Ustawianie_godzin (**rysunek 19**). Jeśli jest on przyciśnięty, to jest zwiększana liczba godzin i wykonywane sprawdzenie czy godziny są większe od 23. Przekroczenie tej wartości powoduje wyzerowanie liczby godzin i cały cykl powtarza się. Wygląd części programu głównego odpowiedzialnej za ustawianie godzin pokazano na **rysunku 20**. Identycznie jest wykonywana nastawa minut, z tym, że jest odczytywany przełącznik SW2, jego stan jest zapamiętywany w zmiennej Ustawianie_minut i jest wykonywane sprawdzenie czy liczba minut nie przekroczyła 59. Na odczytaniu stanu przełączników SW1 i SW2 program odczekuje przez 100 ms, co ma na celu zmniejszenie szybkości pracy klawiszy. Opóźnienie programowe można zastąpić np. makrem oczekiwania na zwolnienie przycisku. Pozo-



Rysunek 19. Odczyt stanu przełącznika *SWITCH(1)* i zapamiętanie go w zmiennej *Ustawianie_godzin*



Rysunek 20. Wygląd części programu głównego – tu ustawianie godzin (podobnie wykonano ustawianie minut)

stawiam to do inwencji osoby modyfikującej program.

Po zakończeniu nastaw, czego oznaką jest zmiana położenia SW0 z „wyzerowanej" na „ustawioną", przerwania są załączane za pomocą instrukcji języka C IEC0bits. T2IE = 1;, liczba sekund jest zerowana i zegar rozpoczyna odmierzanie czasu od nastawionej wartości godzin oraz minut.

Na koniec

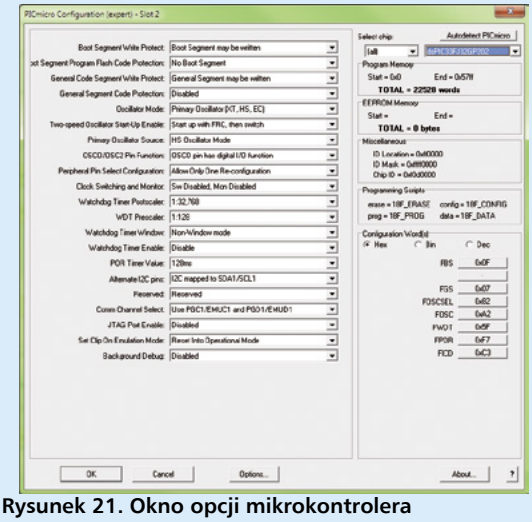
Gwoli przypomnienia i wzorem poprzedniego przykładu, aby częstotliwości wywoływania przerwań były właściwie wyświetlane przez symulator, należy ustawić częstotliwość zegarową mikrokontrolera. Jak wspomniano, do jego taktowania wykorzystano sygnał o częstotliwości 12 MHz generowany na bazie oscylatora kwarcowego. Jego użycie wymaga ustawienia trybu pracy mikrokontrolera oraz parametrów dla IDE. Parametry mikrokontrolera ustawia się za pomocą menu *Chip -> Configure*. W oknie zatytułowanym *PICmicro Configuration – Slot 2* wybieramy:

- Oscillator mode: Primary Oscillator (XT, HS, EC).
- Two-speed Oscillator Start-Up Enable: Start up with FRC, then switch.
- Primary Oscillator Source: HS Oscillator Mode.
- OSCO/OSC2 Pin Function: OSCO pin Has Digital I/O function.
- Watchdog Timer Enable: Disable.
- JTAG Port Enable: Disabled.

Pozostałe parametry można pozostawić domyślne. Okno opcji mikrokontrolera pokazano na **rysunku 21**. Teraz ustawimy częstotliwość oscylatora. Można to zrobić wybierając z menu *View* opcję *Project Options*. Tu z listy *Clock speed (Hz)* wybieramy *12000000* i naciskamy klawisz *OK*.

Zauważone błędy

Dla potrzeb funkcji ustawiania czasu, w tak zwanym „pierwszym podejściu", wyłączyłem przerwania za pomocą ikony *INT* wybierając *TMR2* i zaznaczając opcję *Disable interrupt*. Następnie, aby włączyć przerwania, ponownie użyłem ikony *INT* zaznaczając *Enable interrupt* oraz wskazując funkcję Pomiar_czasu. Taki sposób wykonania programu został bez żadnych zastrzeżeń przyjęte przez symulator. Niestety, próba zaprogramowania mikrokontrolera skończyła się wygenerowaniem komunikatu o błędzie. A więc powtarza się sytuacja bardzo dobrze znana również z innych symulatorów, gdy program napisany dla mikrokontrolera działa poprawnie w symulatorze i „sypie się" w rzeczywistym środowisku pracy. Na szczęście, błąd pojawił się już na etapie kompilowania programu i nie wymagał żmudnej analizy pracy mikrokontrolera, a jedynie analizy wykonanego programu. Niby logicznie wszystko było w porządku, ale czy na pewno?



Rysunek 21. Okno opcji mikrokontrolera

Gdy tworzymy program w asemblerze lub języku C, to na samym początku są definiowane wektory przerwań. Są one umieszczane w pewnej **stałej lokalizacji w pamięci programu**, ponieważ typowy kontroler przerwań działa w taki sposób, że po odebraniu żądania obsługi przerwania pobiera wektor do podprogramu obsługi z pewnej stałej, predefiniowanej lokalizacji, której położenia nie można zmieniać w trakcie wykonywania programu. A więc dwukrotne włączenie i wyłączenie przerwania w sposób dostępny za pomocą ikony *INT* powoduje próbę dwukrotnego zaprogramowania tej samej lokalizacji w pamięci *Flash*, co jest oczywistym błędem. Owszem, w tym wypadku był to adres tego samego programu obsługi, ale ocena tego przez kompilator jest niemożliwa, ponieważ dwukrotne wywołanie makra definiującego przerwanie powoduje dwukrotne utworzenie tablicy wektorów przerwań, co z definicji jest niewykonalne. Tym samym muszę się przyznać do błędu logicznego, chociaż początkowo nie rozumiałem dlaczego kompilator „buntuje się" przy tak oczywistej operacji.

Lekarstwem była możliwość zajrzenia do kodu w języku C, odszukanie linii odpowiedzialnej za załączenie przerwań od Timera 2 i skopiowanie jej do okienka bloku umożliwiającego wstawianie instrukcji w C. Wyłączając przerwania wystarczyło zmienić „1" na „0", natomiast włączenie przerwania nie wymagało żadnej zmiany. A to wszystko bez dokładnej znajomości nazw rejestrów (sic!) programowanego mikrokontrolera…

Podsumowanie

W miarę tworzenia kolejnych programów, Flow Code coraz bardziej mi się podoba. Szczerze mówiąc początkowo podchodziłem do „rysowania programów" z dużą rezerwą, jednak w miarę tworzenia kolejnych aplikacji nie sposób nie docenić łatwości użytkowania tego narzędzia. Dodatkowymi zaletami są możliwość współpracy IDE z popularnymi programatorami mikrokontrolerów oraz elastyczność konfigurowania połączeń mikrokontrolera. Wadą – konieczność posiadania monitora o dużej rozdzielczości, ponieważ rysunki zajmują sporo miejsca na ekranie.

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.