Zegar z termometrem

Przykład aplikacji we FlowCode

na CD

AVT 5384

Flowcode jest postrzegane przede wszystkim jako środowisko do tworzenia programów przez osoby nieznające żadnego języka programowania. Wbudowane makra sprzętowe zapewniają szybkie tworzenie algorytmów sterujących bez konieczności poznawania budowy układów peryferyjnych mikrokontrolera i w pewnym stopniu – dołączanych układów zewnętrznych. Typowym przykładem takiego podejścia jest obsługa alfanumerycznego wyświetlacza LCD. Użytkownik nie musi również martwić się o znajomość algorytmów obsługi klawiatury, ważniejszych magistral szeregowych itp. ***Rekomendacje:*** *funkcjonalny zegar z termometrem, który jest jednocześnie ciekawym przykładem zastosowania kompilatora FlowCode.*

Po „napisaniu" kilku prostych programów postanowiłem spróbować swoich sił w trochę „większym" zadaniu. Z założenia miało to być przydatne urządzenie z czymś w rodzaju menu i funkcjami ustawień parametrów. Każdy, kto projektował taki układ wie, że najwięcej zasobów mikrokontrolera i czasu poświeconego na oprogramowanie zajmuje interfejs użytkownika: tworzenie menu, funkcje ustawiania i wyświetlania parametrów, obsługa manipulatora (klawiatury, enkodera obrotowego), wyświetlanie komunikatów itp. Te zadania muszą wykonywać algorytmy napisane przez programistę, a Flowcode może tylko (lub aż) zapewnić wsparcie w postaci obsługi wyświetlacza LCD.

Postanowiłem skonstruować i wykonać zegar z termometrem. Na jego przykładzie takiego opiszę, w jaki sposób poradziłem sobie z zadaniem z zadaniem z użyciem FlowCode. Przed zaprojektowaniem układu elektrycznego przyjąłem następujące założenia:

- Zegar ma być wyposażony w czytelny wyświetlacz LCD z podświetlaniem. Wybrałem taki o organizacji 1 wiersz na 6 znaków, ponieważ jego ekran jest stosunkowo duży i wyświetlany czas oraz temperatura są dobrze widoczne z większej odległości.
- Jako czujnik temperatury zastosowałem popularny układ DS18B20. Wybór był podyktowany częściowym wsparciem obsługi tego termometru, a szczególnie wsparciem obsługi interfejsu 1-wire przez Flowcode.
- Zdecydowałem się na użycie zewnętrznego układu zegara RTC. Wybrałem układ MCP79512 produkowany przez Microchipa. Układ komunikuje się z systemem nadrzędnym za pomocą interfejsu SPI.
- Do obsługi menu użytkownika posłuży enkoder (impulsator) ze stykiem zwieranym przyciśnięciem oski.
- Ponieważ mam FlowCode dla mikrokontrolerów PIC16F i PIC18F, to wybór musiał być ograniczony do tych rodzin. Ostatecznie wybór padł na PIC18F88. Moim zdaniem to mikrokontroler o bardzo dobrym stosunku ceny do możliwości, a poza tym bardzo dobrze pasował do założeń projektowych.

Schemat zegara z termometrem pokazano na **rysunku 1**. Wyświetlacz dołączono do mikrokontrolera przez 4 bitową magistralę równoległą. Linie danych D0…D3 są połączone z liniami portu PORTA (RA0…RA3). Sygnał RS jest połączony z linią RA5, a EN z linią RA7. Oprócz tego, wyświetlacz ma typowy układ regulowania kontrastu z potencjometrem R3.

Termometr DS18B20 jest zasilany napięciem +3,3 V. Należy dołączyć go do konektora U1 . Niezbędne podciąganie do plusa zasilania linii magistrali 1-wire (linia portu RA6) jest realizowane za pomocą rezystora R2 (1 kΩ). Zegarek RTC składa się z układu U3 (MCP79512), obwodu oscyla-

W ofercie AVT*
AVT-5384 A **AVT-5384 B**
AVT-5384 UK

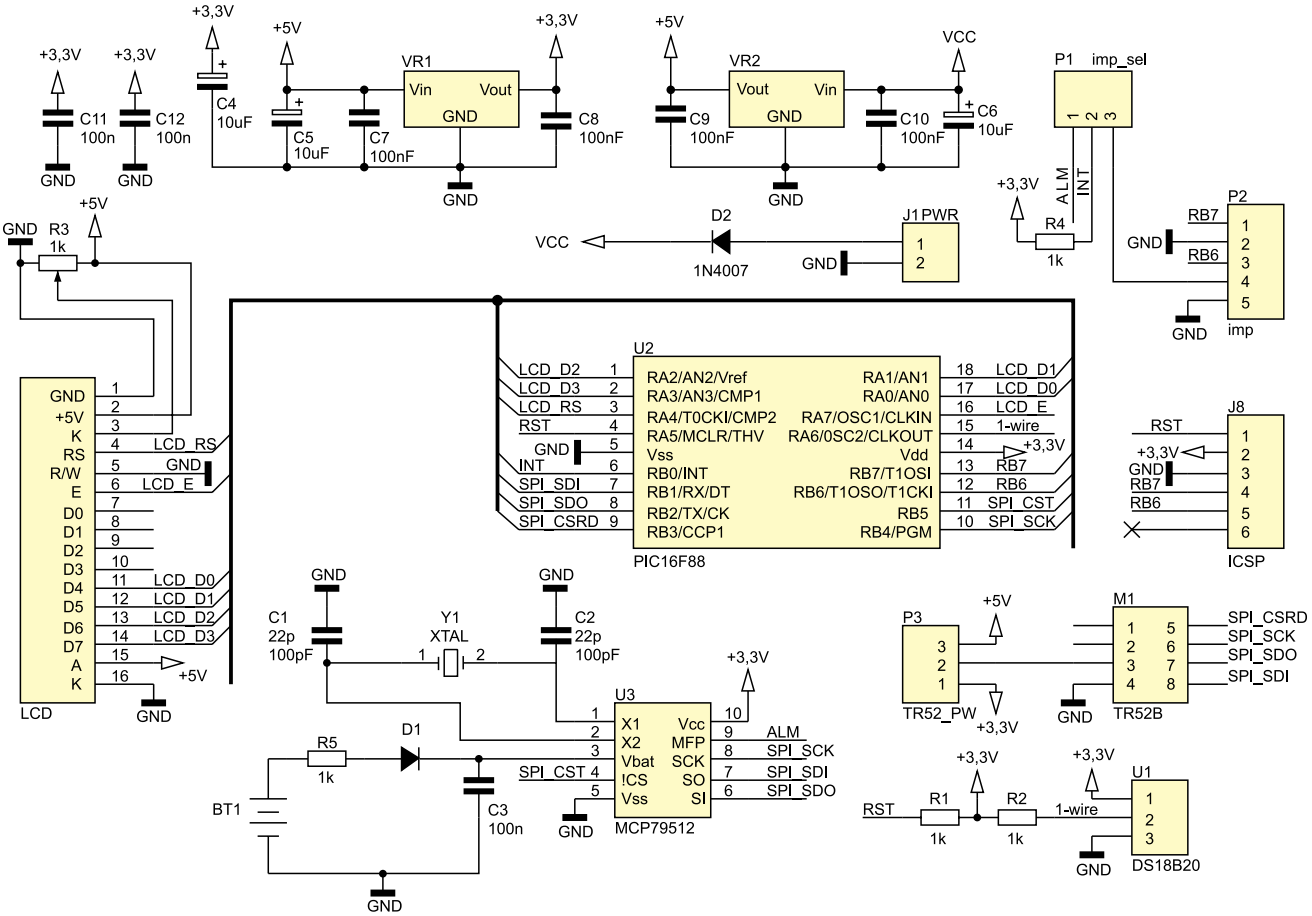
Podstawowe informacje:
- Napięcie zasilające 9…12 V DC/200 mA.
- Mikrokontroler PIC16F88, układ zegara MCP79512 z podtrzymaniem bateryjnym.
- Wyświetlacz 1 linia × 8 znaków.
- Wyświetlanie czasu (minuty i godziny) i temperatury.
- Oprogramowanie we FlowCode v4.0.

Dodatkowe materiały na CD/FTP:
ftp://ep.com.pl, user: 63048, pass: 632vmey5
- wzory płytek PCB
- karty katalogowe i noty aplikacyjne elementów oznaczonych w Wykazie elementów kolorem czerwonym

* Uwaga:
Zestawy AVT mogą występować w następujących wersjach:

AVT xxxx UK	to zaprogramowany układ. Tylko i wyłącznie. Bez elementów dodatkowych.
AVT xxxx A	płytka drukowana PCB (lub płytki drukowane, jeśli w opisie wyraźnie zaznaczono), bez elementów dodatkowych.
AVT xxxx A+	płytka drukowana i zaprogramowany układ (czyli połączenie wersji A i wersji UK) bez elementów dodatkowych.
AVT xxxx B	płytka drukowana (lub płytki) oraz komplet elementów wymieniony w załączniku pdf
AVT xxxx C	to nic innego jak zmontowany zestaw B, czyli elementy wlutowane w PCB. Należy mieć na uwadze, że o ile nie zaznaczono wyraźnie w opisie, zestaw ten nie ma obudowy ani elementów dodatkowych, które nie zostały wymienione w załączniku pdf
AVT xxxx CD	oprogramowanie (nieczęsto spotykana wersja, lecz jeśli występuje, to niezbędne oprogramowanie można ściągnąć, klikając w link umieszczony w opisie kitu)

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! (UK, A, A+, B lub C). **http://sklep.avt.pl**

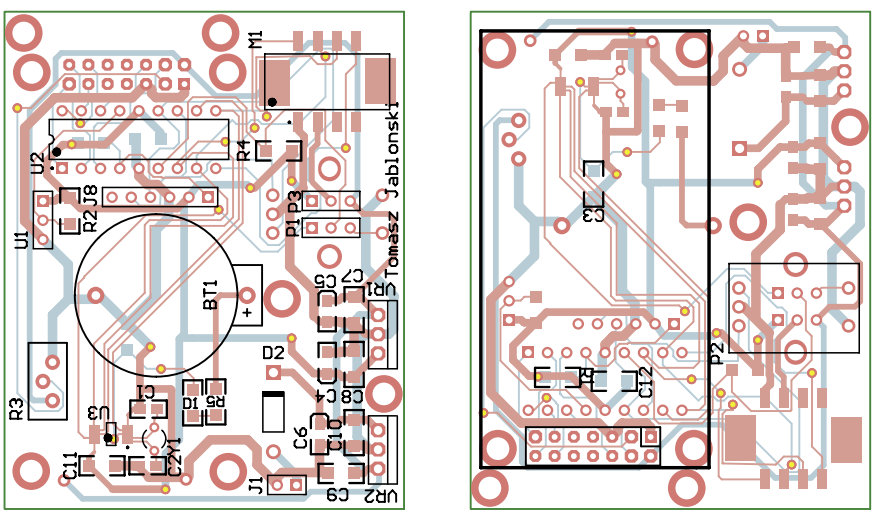


Rysunek 1. Schemat zegara z termometrem

tora kwarcowego z kwarcem zegarkowym X1 oraz kondensatorami C1 i C2, i obwodu podtrzymującego zasilanie liczników zegara po zaniku napięcia głównego (dioda D1, rezystor R5, kondensator C3 i bateria litowa o napięciu 3 V). Linie A i B enkodera połączono z liniami portów RB6 i RB7, a zwierany styk z linią RB0.

Układ zasilania dostarcza napięcie +5 V (układ stabilizatora VR2 typu 7805) do zasilania wyświetlacza LCD i +3,3 V (układ stabilizatora z układem VR1 typu LM2937-3,3) do zasilania mikrokontrolera PIC18F88, zegara RTC MCP79512 oraz termometru DS18B20.

Rozmieszczenie elementów na płytce zegara pokazano na **rysunku 2**. Płytkę drukowaną zaprojektowano w taki sposób, że elementy są montowane z jej obu stron. Na umownej stronie elementów jest umieszczony wyświetlacz LCD i impulsator. Na umownej stronie druku (**fotografia 3**) umieszczono pozostałe elementy. Wyświetlacz należy przykręcić za pomocą 4 wkrętów M2.5 z plastykowymi tulejkami dystansowymi i połączyć z płytką przewodami. Jedynym elementem, który może sprawić problem mniej wprawnym elektronikom jest układ zegarka MCP79512 o gęstym rastrze wyprowadzeń. Mikrokontroler ma obudowę typu DIP i dobrze jest dla niego przewidzieć podstawkę. PIC16F88 można zaprogramować w podstawce zewnętrznego programatora lub w systemie. Do tego celu przewidziano złącze J8. Ponieważ do linii RB6 i RB7 wykorzystywanych do programowania jest dołączony impulsator, to może okazać się, że jedna z nich będzie na stałe zwarta do masy. Dlatego mikrokontroler powinien być zaprogramowany przed wlutowaniem impulsatora. Po zmontowaniu płytki dołączamy do złącza U1 (bezpośrednio lub za



Rysunek 2. Płytka drukowana zegara z termometrem

Na CD: karty katalogowe i noty aplikacyjne elementów oznaczonych w wykazie elementów kolorem czerwonym

Wykaz elementów

Rezystory:
R1, R2, R4, R5: 1 kΩ (SMD 1206)
R3: potencjometr 4,7 kΩ
Kondensatory:
C1, C2: 22 pF (SMD 0805)
C3, C7, C8… C12: 100 nF (SMD 1206)
C4…C6 1 µF/16 V (SMD 3216)
Półprzewodniki:
D1: BAV21 (SMD)
D2: 1N4007
U1: DS18B20
U2: PIC16F88 I/P
U3: MCP79512
VR1: LM2937-3.3 (lub odpowiednik)
VR2: 7805
Inne:
Kwarc zegarkowy 32768 kHz
Wyświetlacz LCD 1×6 znaków
Podstawka 18 pin
Gniazdo baterii
Płytka drukowana
Listwa godpinów 2,54 mm
Impulsator Alps EC11 ze stykiem

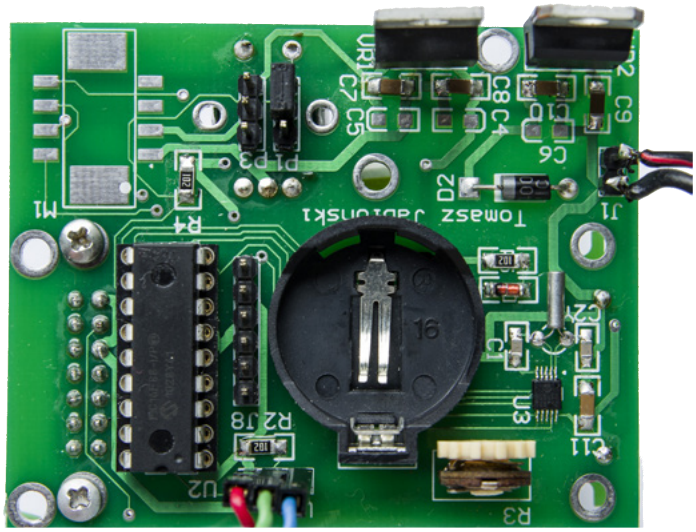
pomocą przewodów) termometr DS18B20 i układ jest gotowy do testowania poprawności działania.

Poza ustawieniem kontrastu potencjometrem R3 i zwarciem pinów 2 z 3 złącza P1 nie trzeba specjalnych czynności uruchomieniowych. Do złącza J1 dołączamy zasilacz 9...12 V DC i obciążalności ok. 200 mA. Jeżeli wszystko jest zmontowane prawidłowo, to na wyświetlaczu powinna wyświetlić się zmierzona temperatura otoczenia. Przez ogrzanie termometru DS18B20 można sprawdzić, czy pomiar się zmienia. Trzeba tylko pamiętać, że pomiar jest wykonywany co 1 sekundę. Czas odliczany przez zegarek jest wyświetlany po naciśnięciu i przytrzymaniu oski impulsatora. Oskę przytrzymujemy tak długo, aż zostanie wyświetlony czas. Powtórne przyciśnięcie oski powoduje przejście do wyświetlania temperatury. Kiedy zegar wyświetla czas, to obrót oski impulsatora powoduje wejście do ustawiania czasu. Najpierw kręcąc oską ustawiamy godziny. Kiedy są już ustawione naciskamy oskę i program przechodzi do ustawiania minut. Minuty są ustawiane tak jak godziny. Ustawiony czas jest wpisywany do rejestrów zegarka i zliczanie jest uruchamiane po naciśnięciu oski.

Oprogramowanie

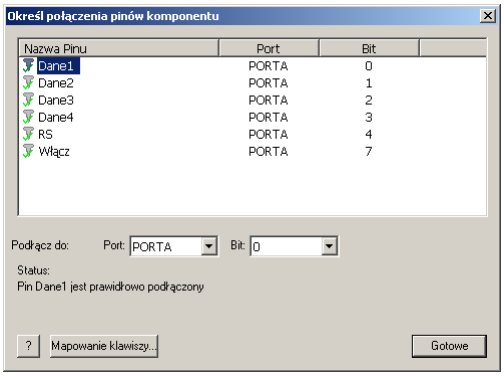
Omawianie programu sterującego zegarem zacznijmy od zdefiniowania konfiguracji sprzętowej. FlowCode będzie za pomocą makr wpierało obsługę dwóch urządzeń peryferyjnych: wyświetlacza LCD i interfejsu 1-wire. Interfejs SPI będzie obsługiwany „na piechotę” przez specjalnie do tego celu napisane makro programowe.

Po ułożeniu na panelu symbolu wyświetlacza można otworzyć okno służące do definiowania połączeń z liniami portów mikrokontrolera – pokazano je na **rysunku 4**. Okno służące do definiowania wyprowadzeń interfejsu 1-wire pokazano na **rysunku 5**. Będziemy korzystali z makr obsługi magistrali 1-wire oraz z makr wspomagających obsługę termometru DS18B20. Obie te definicje nie są elementami właściwego programu i należy je wykonać przed rozpoczęciem programowania.

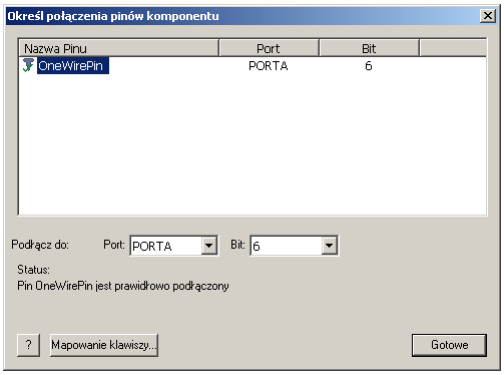


Fotografia 3. Widok elementów od strony lutowania

Wykonanie programu obsługi zegara jest dosyć „dużym” zadaniem ze względu na obszerne rysunki zajmujące sporo miejsca na ekranie. Narysowanie go na jednej planszy spowoduje, że późniejsza analiza i wprowadzanie poprawek będzie bardzo niewygodne. Dlatego program podzielono na bloki funkcjonalne. Każdy z takich bloków umieszczono w oddzielnym makrze programowym będącym odpowiednikiem funkcji w języku C. Do makra można przekazywać argumenty, może ono również zwracać rezultaty. Oprócz tego makra mogą operować na zmiennych globalnych.



Rysunek 4. Okno definicji połączeń wyświetlacza LCD

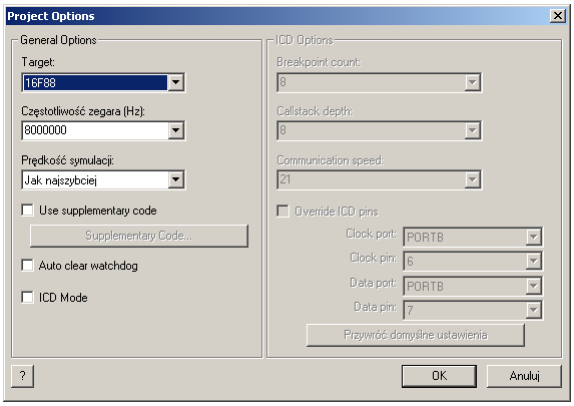


Rysunek 5. Okno definicji połączeń interfejsu 1-wire

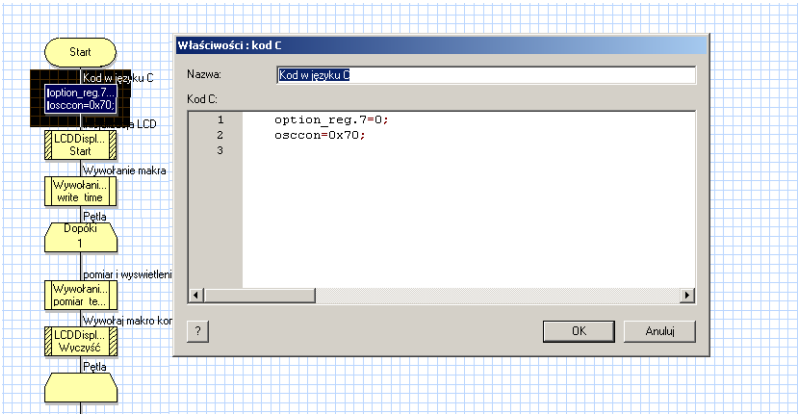
Makro *pomiar_temp*

Zadaniem makra jest odczytanie temperatury z układu DS18B20, a następnie jej wyświetlenie. Sposób obsługi interfejsu 1-wire był wielokrotnie, dokładnie opisywany na łamach Elektroniki Praktycznej i nie będę go powtarzał. Przypomnę tylko, że wymiana danych składa się z sekwencji: inicjowania, zapisu bitu (zera logicznego i jedynki logicznej) oraz odczytu bitu (zera i jedynki logicznej). Sekwencje mają dokładnie zdefiniowane szczeliny czasowe. Programowa implementacja obsługi 1-wire nie jest trudna, ale wymaga zachowania reżimów czasowych. Dlatego FlowCode musi „znać” częstotliwość, z którą jest taktowany mikrokontroler, aby wyliczyć czasy trwania sekwencji czasowych na magistrali.

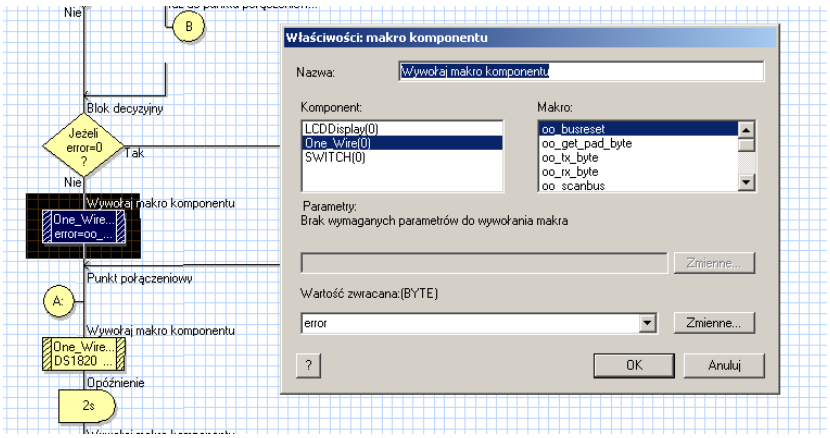
Mikrokontroler PIC16F88 może być taktowany za pomocą zewnętrznego oscylatora kwarcowego dołączonego do wyprowadzeń RA6 i RA7. Ponieważ w projekcie potrzebo-



Rysunek 6. Ustawienie częstotliwości taktowania dla symulatora oraz procedur odmierzania czasu

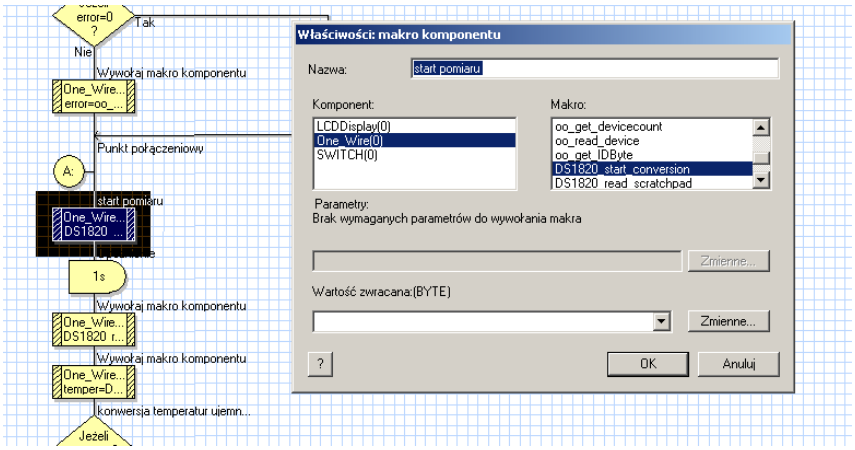


Rysunek 7. Ustawienie częstotliwości taktowania mikrokontrolera w programie

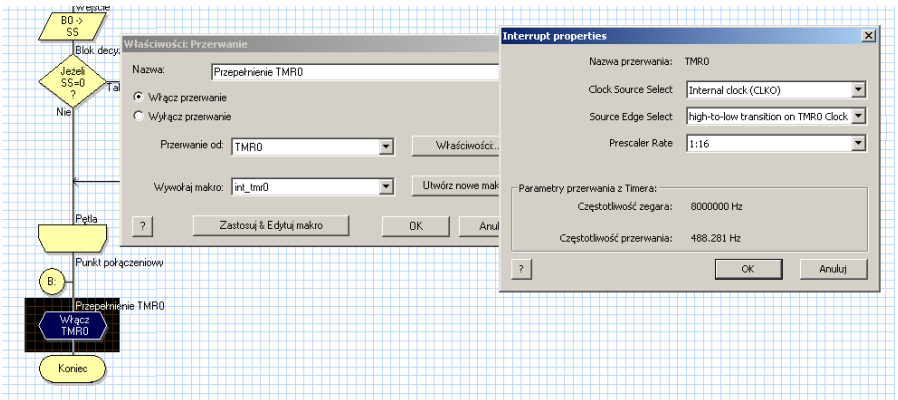


Rysunek 8. Funkcja inicjalizowania interfejsu 1-wire

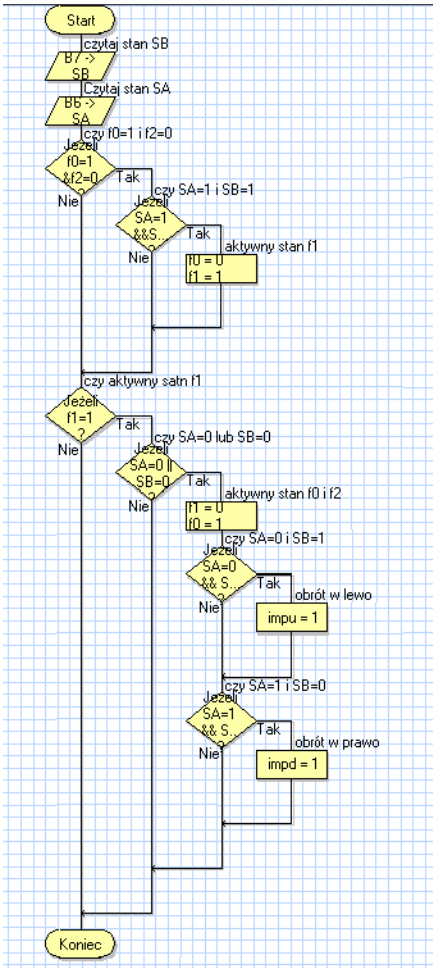
wałem tych wyprowadzeń do innych celów, to zdecydowałem się na taktowanie z użyciem wewnętrznego, precyzyjnego oscylatora RC. Wytwarza on sygnał o częstotliwości 8 MHz, ale użytkownik może go zmieniać przez ustalenie stopnia podziału za pomocą rejestru OSCCON. Po restarcie mikrokontrolera zawartość OSCCON jest zerowana i mikrokontroler jest taktowany z częstotliwością 31,25 kHz. Z tego powodu, na początku programu trzeba wpisać do OSCCON wartość 0x70 – wtedy mikrokontroler jest taktowany z częstotliwością 8 MHz. Ale aby FlowCode o tym „wiedział", trzeba tę częstotliwość wpisać w oknie „Częstotliwość zegara" zakładki opcji projektu *Widok -> Project Options* (**rysunek 6**). To ustawienie jest też wykorzystywane miedzy innymi do obliczeń opóźnień w obsłudze wyświetlacza LCD. Zapisanie rejestru OSCCON jest wykonywane w bloku wstawek w języku C. Blok oraz jego zawartość pokazano na **rysunku 7**. Oprócz zmiany zawartości rejestru OSCCON jest też zerowany bit RPBU (bit 7 rejestru *Option Register*). Powoduje to automatyczne podciąganie do plusa zasilania przez wewnętrzne rezystory wszystkich linii portu PORTB ustawionych jako wejściowe. Jest to niezbędne do prawidłowego odczytu poziomów linii A i B enkodera oraz styku zwieranego po naciśnięciu oski.



Rysunek 9. Początek pomiaru temperatury



Rysunek 10. Włączenie i zdefiniowanie przerwania od TMR0



Rysunek 11. Makro obsługi impulsatora

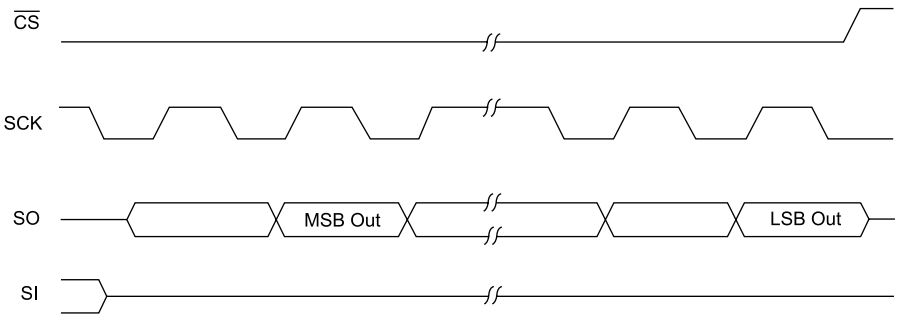
Wróćmy do pomiaru temperatury. Kiedy na magistrali 1-wire nie jest wykonywana żadna operacja, to linia danych jest w stanie wysokiej impedancji. Wtedy linia 1-wire (RA6) będzie na poziomie wysokim, bo jest podciągana do plusa przez rezystor R2. Wymiana informacji z termometrem musi rozpoczynać się od wymuszenia przez mikrokontroler sekwencji inicjalizacji – *Reset and presence pulses*. Wykonuje to funkcja wsparcia obsługi magistrali 1-wire oo_busreset pokazana na **rysunku 8**. Program czeka w pętli aż ta sekwencja zostanie zakończona. Dalej będziemy już korzystali z gotowych funkcji obsługi termometru DS18B20 wykonanych przez producenta kompilatora.

Pomiar temperatury rozpoczynamy funkcją *DS1820_start_conversion* (**rysunek 9**) polegającą na wysłaniu na magistralę komendy *ConvertT*. Czas trwania konwersji jest zależny od zaprogramowanej rozdzielczości. FlowCode ustawia rozdzielczość 12-bitową. Wtedy pomiar trwa ok. 750 ms. Zakończenie pomiaru można stwierdzić sprawdzając czy na magistrali wystąpi poziom wysoki. Można również nie testować stanu magistrali, lecz po prostu poczekać przez odpowiedni czas. W naszym programie czekamy na pomiar przez 1 sekundę, a potem jest wykonywana funkcja *DS1820_read_scratchpad*. Właściwe odczytanie temperatury realizuje funkcja *DS1820_get_temp* zwracająca 16-bitową wartość zapisywana do rejestru *temper*. Wartość z rejestru *temper* jest liczbą typu int ze znakiem. Aby stwierdzić czy odczytana temperatura jest ujemna, wystarczy sprawdzić, czy temper jest liczbą ujemną. Liczby ujemne są zapisane w kodzie U2, więc aby poprawnie wyświetlić temperaturę trzeba zanegować wszystkie bity i dodać jeden.

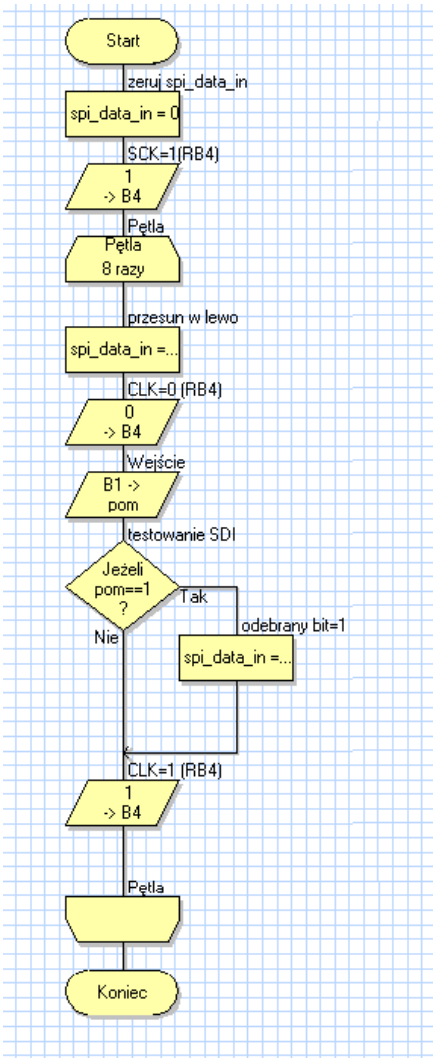
Jeżeli mamy już wykonane sprowadzenie czy liczba jest dodatnia, czy ujemna oraz wykonaną konwersję z kodu U2, to będzie konieczne przekształcenie odczytanej wartości na postać możliwą do wyświetlenia. Dla konwersji 12-bitowej część całkowita temperatury jest wyliczana z wyrażenia Tmp = temper/16. Przy założeniu rozdzielczości 0,5°C część ułamkowa jest wyliczana z równania Tmpu=(temper MOD 16)*6. Wyliczone wartości są wyświetlane przez makro *disp_temp*. Zależnie od tego czy temperatura jest dodatnia, czy ujemna jest wyświetlany znak, a później 2 cyfry części całkowitej, kropka dziesiętna, 1 cyfra części ułamkowej z literą C np. +21.5C. Użycie gotowych makr obsługi 1-wire, termometru DS18B20 i wyświetlacza LCD spowodowało, że program nie jest rozbudowany i łatwo go przeanalizować.

Obsługa enkodera

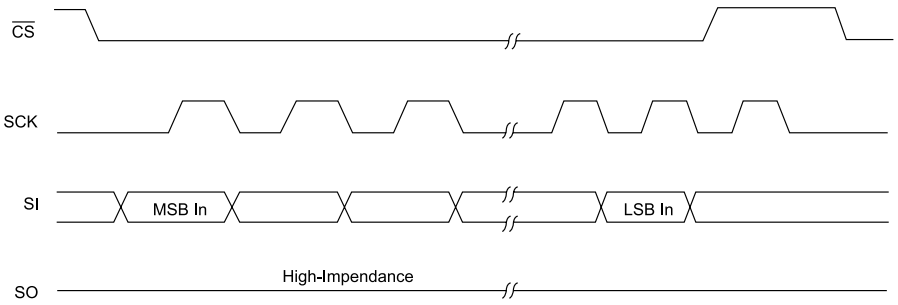
Obrotowy enkoder (impulsator) to element chętnie stosowany w interfejsie użytkownika. Jednak jego obsługa nie jest banalna. Główny problem tkwi w tym, że przy pokręcaniu osią są generowane sygnały przesunięte w fazie i program obsługi nie może „zgubić" ich zboczy. Ponadto, jest konieczne określenie kolejności występowania zboczy. Dlatego jest trudno napisać



Rysunek 12. Przebiegi czasowe przy odczytywaniu danych z układu scalonego zegara



Rysunek 13. Makro *spi_in*



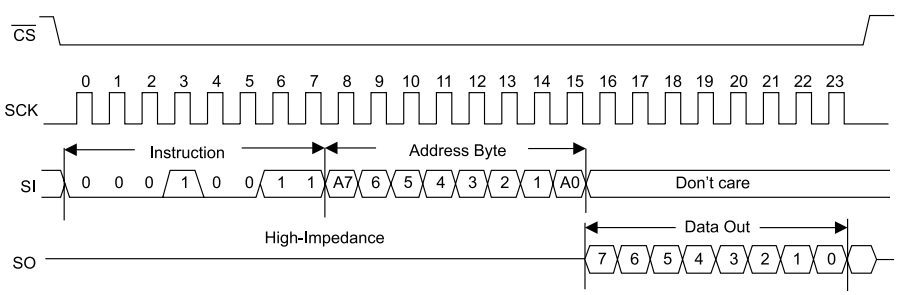
Rysunek 14. Przebiegi czasowe przy zapisywaniu danych do układu scalonego zegara

uniwersalną procedurę pracującą na zasadzie poollingu (odpytywania), bo zazwyczaj czytanie stanu impulsatora jest przerywane innymi czynnościami wymagającymi czasu, np. wyświetlaniem informacji. Dlatego do jego obsługi głównie wykorzystuje się dwie metody oparte o przerwania. Jedna z nich polega na wykorzystaniu zgłaszania przerwania zewnętrznego przez zbocza generowane przez enkoder. Jest ona nieskomplikowana, ale wymaga użycia dwóch wejść przerwań zewnętrznych. Mikrokontroler PIC16F ma możliwość generowania przerwań za pomocą PORTB i można użyć tej metody. Druga metoda, to użycie przerwań od licznika zgłaszanych co 1 ms. W obsłudze przerwania jest wywoływana funkcja maszyny stanu wykrywającej zbocza i ich kolejność.

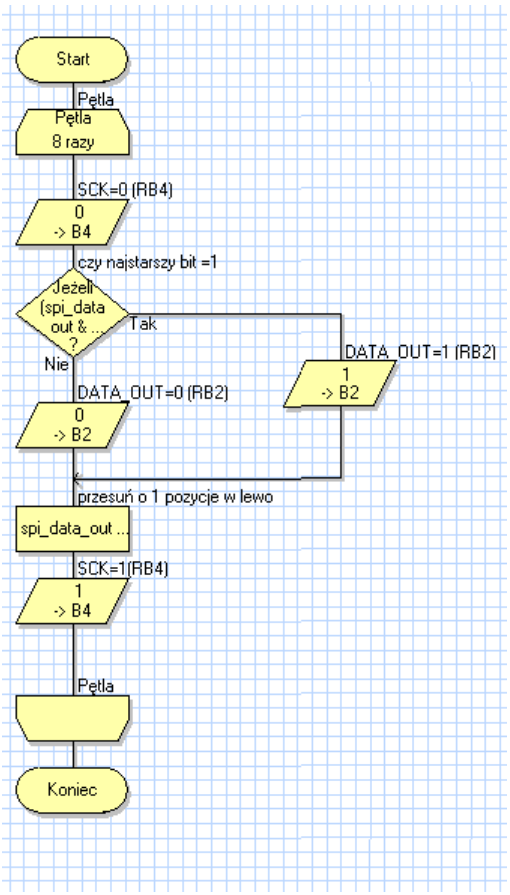
Twórcy FlowCode docenili uniwersalność enkodera i wbudowali makra wspierające jego obsługę, ale dopiero od wersji V5. Ja dysponuję wersją V4 i dlatego funkcję obsługi napisałem sobie sam. Jest oparta na metodzie maszyny stanów i przerwania zgłaszanego przez licznik TMR0 co 1 ms.

Najpierw trzeba skonfigurować licznik, tak aby zliczał cykle maszynowe po podzieleniu przez 4. Jak pamiętamy mikrokontroler jest taktowany sygnałem o częstotliwości 8 MHz, zatem licznik ma zliczać impulsy o częstotliwości 8 MHz/4=2 MHz. Dla 8-bitowego licznika to zbyt dużo, aby na jego wyjściu otrzymać sygnał przepełnienia co 1 ms i dlatego na wejście licznika włączymy preskaler o podziale 1/16. Wtedy będzie on zliczał impulsy o częstotliwości 2 MHz/16=125 kHz. Jeżeli teraz po każdym przepełnieniu wpiszemy do licznika wartość 255-125=130, to przerwania będą zgłaszane co 1 ms. Na **rysunku 10** pokazano konfigurację przerwania od TMR0. W oknie „Przerwanie od" jest wybierane źródło – w naszym przypadku jest to TMR0. W oknie „Wywołaj makro" wybieramy makro obsługi przerwania. Po kliknięciu na belkę „Właściwości" możemy zdefiniować licznik: Internal Clock – zliczanie cykli rozkazowych i „Prescaler rate" – w naszym wypadku 1:16.

Makro *int_tmr0* jest wywoływane co 1 ms i ma za zadanie wpisać do TMR0 wartość początkową 130 oraz uruchamiać pracę maszyny stanów poprzez wywołanie makra *rot_enc*. Maszyna ma zdefiniowane 3 stany oznaczone jako f0, f1 i f2. Na początku programu, przy inicjalizacji, ustawia się aktywny stan f0=1, a pozostałe są nieaktywne (f1=0 i f2=0). Zależnie od odczytywanych poziomów wejść SA (RB6) i SB (RB7) ma-

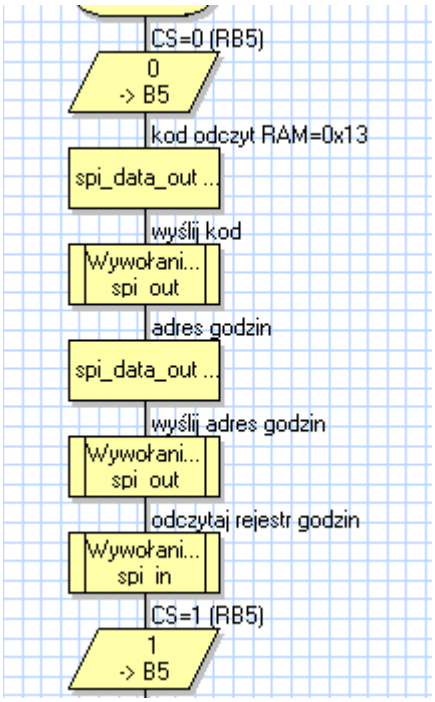


Rysunek 16. Przebiegi czasowe komendy READ



Rysunek 15. Makro *spi_out*

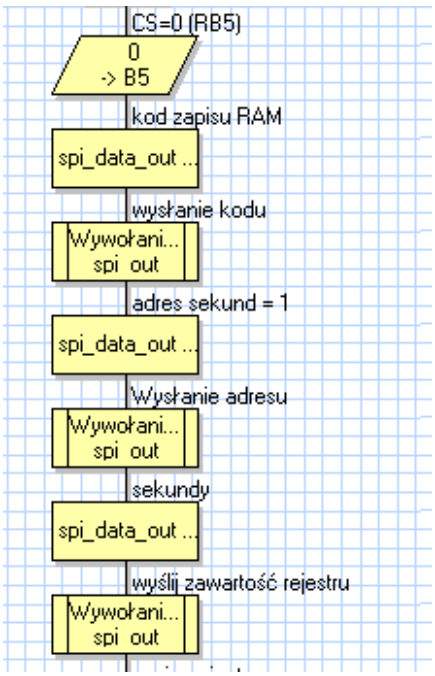
szyna zmienia swoje stany. Jeżeli stan f2 stanie się aktywny, to oznacza, że wykryto obrót i jego kierunek. Kierunek obrotu jest zapisany w zmiennych *impu* (obrót w prawo) i *impd* (obrót w lewo). Pozostałe stany są automatycznie zerowane. Makro obsługi



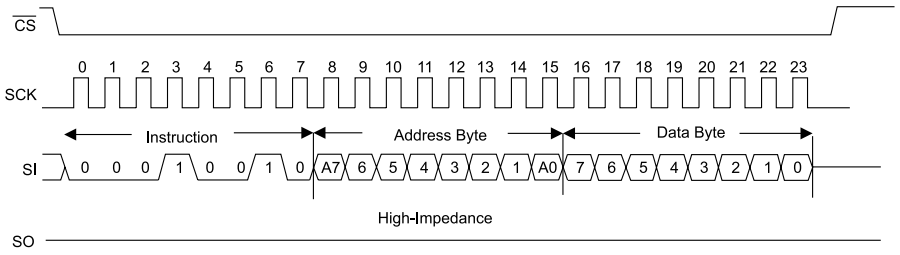
Rysunek 17. Makro odczytujące licznik godzin

impulsatora pokazano na **rysunku 11**. Do momentu, gdy w programie głównym nie zostanie wyzerowany stan f2, impulsator nie jest obsługiwany. Kiedy cały ten mechanizm jest uruchomiony i działa prawidłowo, wystarczy testowanie stanu f2. Jeżeli jest ustawiony, to odczytujemy wartość *impu* oraz *impd* i na tej podstawie określamy kierunek obrotu. Kiedy potrzebujemy kolejnej informacji o obrocie ośki, to zerujemy f2 i czekamy aż zostanie ponownie ustawiony.

Przy okazji warto wspomnieć, że włączone przerwanie od TMR0 zakłóca prawidłowe działanie wbudowanych funkcji



Rysunek 19. Fragment programu zapisującego licznik sekund



Rysunek 18. Makro zapisujące rejestr zegarka

magistrali 1-wire. Najprawdopodobniej wtedy nie jest możliwe prawidłowe odmierzanie krótkich czasów potrzebnych do prawidłowego transferu bitów na magistrali. Dlatego kiedy jest uruchomiony pomiar temperatury, przerwanie od TMR0 jest blokowane.

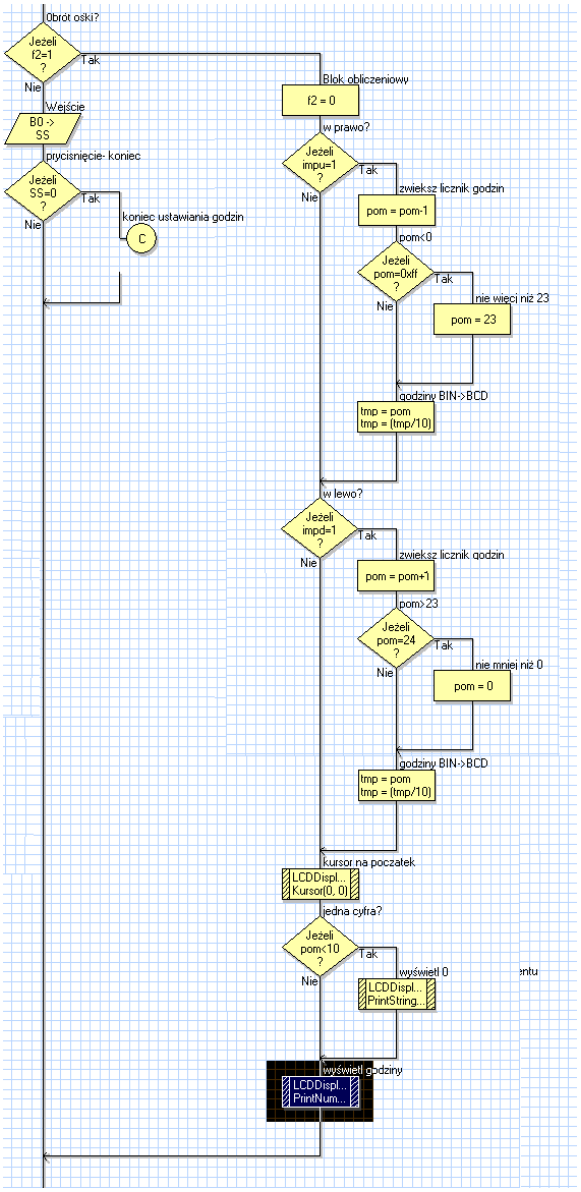
Enkoder będzie używany do ustawienia czasu.

Zegar RTC

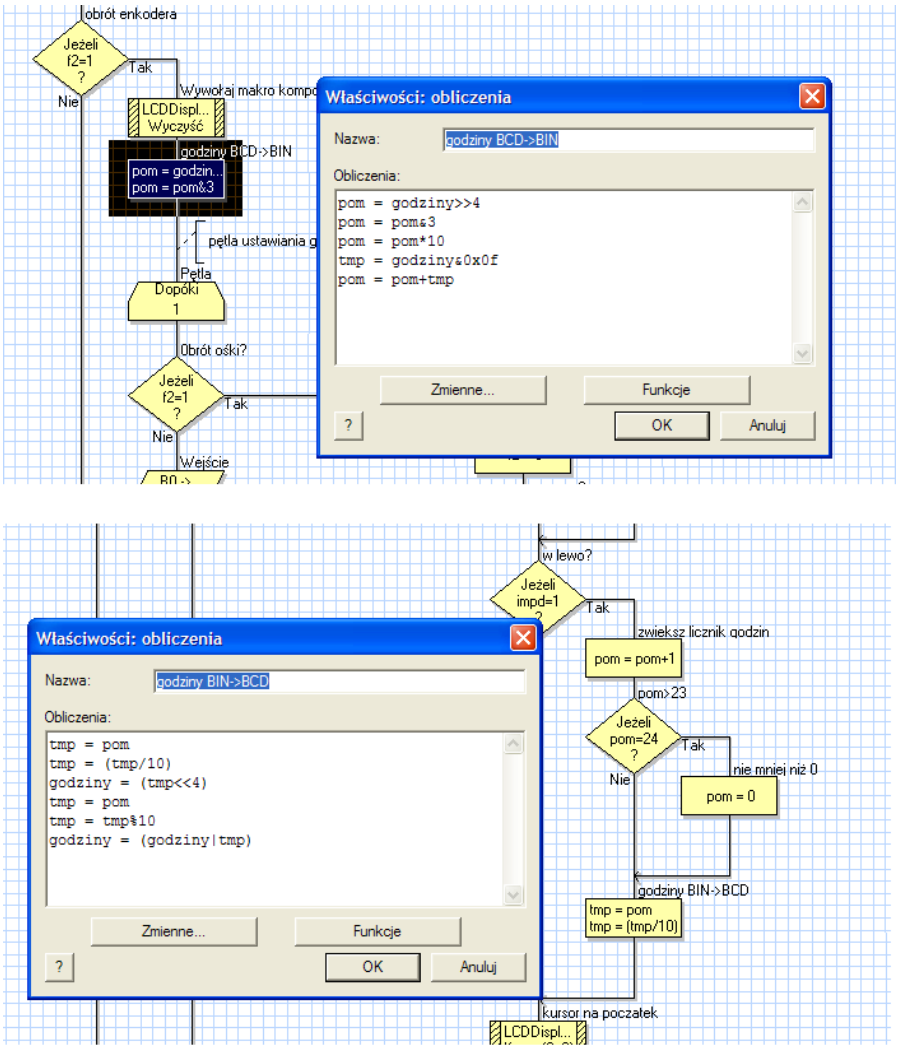
Układ scalony zegara RTC typu MCP79512 jest taktowany za pomocą kwarcu zegarkowego o częstotliwości 32768 Hz. W strukturze układu zawarto liczniki zliczające godziny, minuty sekundy i setne części sekund oraz kalendarz zbudowany z liczników dni tygodnia, miesięcy i lat. Dwa banki dodatkowych rejestrów czasu i rejestrów dnia tygodnia, dnia miesiąca i numeru miesiąca pozwalają na zaprogramowanie dwóch niezależnych alarmów. Oprócz rejestrów licznikowych umieszczonych w pamięci SRAM, użytkownik ma do dyspozycji 64 bajty pamięci SRAM i 1 kbit (256 bajtów) pamięci EEPROM.

Ważną cechą zegara jest możliwość programowego kalibrowania dokładności odliczanego czasu w zakresie ±255 ppm z krokiem co 1 ppm. W kalibrowaniu pomaga wyprowadzenie sygnału generatora o programowanej częstotliwości: 1 Hz, 4096 Hz, 8192 Hz lub 32768 Hz. Można ją mierzyć za pomocą dokładnego częstościomierza i precyzyjnie wyregulować odmierzanie czasu. Układ ma możliwość zasilania z baterii w razie zaniku głównego napięcia zasilającego. Dostęp do rejestrów zegara, kalendarza, obszaru pamięci SRAM użytkownika oraz pamięci EEPROM jest realizowany poprzez interfejs SPI złożony z linii:

- CS – wyzerowanie tej linii aktywuje interfejs SPI zegara. W czasie transferu danych do i z zegara sygnał CS musi być wyzerowany. Ustawienie CS wprowadza interfejs w stan *standby*. Pracują wtedy tylko liczniki czasu, a układ pobiera znikomy prąd. Linia wyjścia danych SO przechodzi w stan wysokiej impedancji umożliwiając przesyłanie magistralą danych innym urządzeniom. Jest połączona z linią portów mikrokontrolera RB5.
- SO – wyjście danych zegara RTC, połączone z linią danych wejściowych mikrokontrolera (port RB1). Dane są



Rysunek 20. Fragment procedury ustawiania godzin.



Rysunek 21. Konwersje BCD na BIN i BIN na BCD

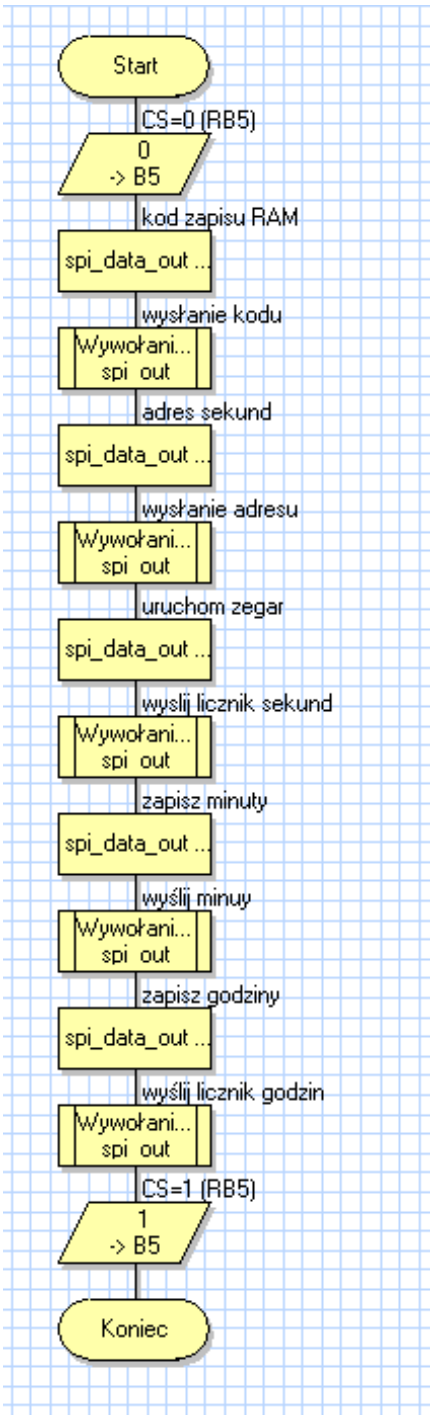
wyprowadzane na SO przy zboczu opadającym sygnału CLK.

- SI – wejście danych zegara RTC, połączone z linią danych wyjściowych mikrokontrolera (port RB2). Dane są zatrzaskiwane przy narastającym zboczu sygnału CLK.
- CLK – sygnał zegarowy interfejsu SPI.

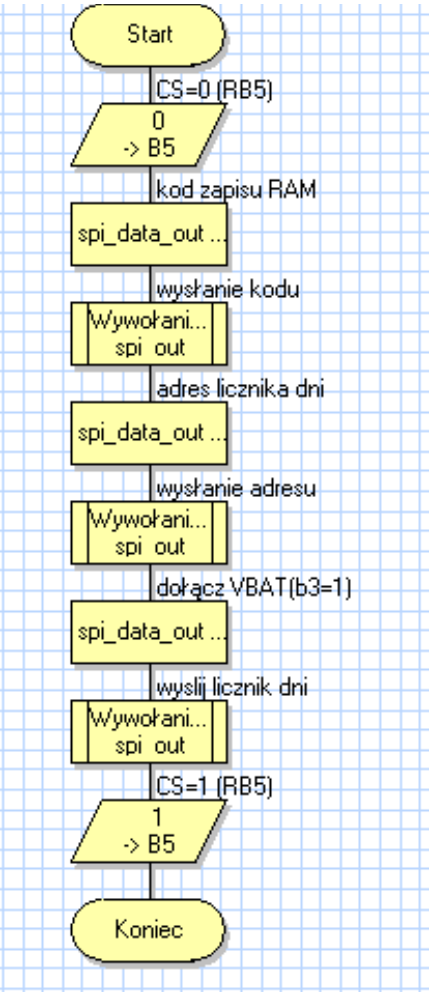
Mikrokontroler PIC16F88 ma wbudowany sprzętowy moduł synchronicznej magistrali szeregowej SSP mogący pracować w trybie SPI Master. Działanie modułu jest powiązane z systemem przerwań i można zaprogramować komunikację z zewnętrznymi urządzeniami pracującą w tle programu głównego. Tutaj nie ma takiej potrzeby i dlatego interfejs SPI emulowany programowo. Do tego celu utworzyłem dwa makra: *spi_in* odczytujące bajt danych z układu zegarka oraz *spi_out* zapisujące bajt danych do układu zegarka. Na **rysunku** 12 pokazano przebiegi czasowe przy odczytywaniu danych z zegarka. Sekwencja rozpoczyna się od wyzerowania linii CS. Dane na linii SO są ważne po opadającym zboczy sygnału CLK. Makro *spi_in* pokazano na **rysunku 13**. Bity odczytywane z wyjścia SO są wpisywane na najmłodszą pozycję (bit b0) zmiennej *spi_data_in* i przesuwane o 1 pozycję w lewo. Po 8 taktach zegarowych w *spi_data_in* jest umieszczony kompletny bajt. Makro nie zmienia stanu linii CS. Stan tej linii jest modyfikowany przy pełnym cyklu zapisu/odczytu rejestrów zegarka.

Przebiegi czasowe przy wysłaniu danych do zegarka pokazano na **rysunku 14**, natomiast makro *spi_data_out* na **rysunku 15**. Dane z wejścia SI są wpisywane do zegarka przy narastającym zboczu sygnału CLK. Linia SO przechodzi w stan wysokiej impedancji. Nie jest to więc typowy interfejs SPI, w którym zapisywanie danych do wejścia SI powoduje wyprowadzenie danych na wyjściu SO.

Oba makra tylko zapisują i odczytują bajty. Zapisywanie i odczytywanie zwartości rejestrów zegarka wymaga sekwencji przesłania kilku bajtów. Ponieważ oprócz rejestrów zegara/kalendarza do dyspozycji jest też pamięć SRAM i EEPROM, to wprowadzono komendy sterujące procesem dostępu. Sekwencja zapisu lub odczytu rejestrów rozpoczyna się o wyzerowania linii CS. Pierwszym bajtem jest zawsze kod komendy. Dokładny opis komend i mapa rejestrów znajduje się dokumentacji układu.



Rysunek 22. Makro write_time – zapisanie rejestrów zegarka



Rysunek 23. Makro en_vbat – włączenie zasilania Vbat

Nas będzie interesowała komenda READ o kodzie 0x13 odczytująca rejestr o zadanym adresie i komenda WRITE o kodzie 0x12 zapisująca rejestr o zadanym adresie oraz rejestry o adresach 0x01 (licznik sekund), 0x02 (licznik minut) i 0x03 (licznik godzin).

Przebiegi czasowe komendy READ pokazano na **rysunku 16**. Po wysłaniu kodu 0x13 wysyłany jest adres rejestru. Oba bajty są wysyłane przez makro *spi_out*. Później jest wywoływane makro *spi_in* odczytujące zawartość zaadresowanego rejestru. Na **rysunku 17** pokazano fragment programu odczytujący rejestr godzin z zegarka. Odczytane godziny są umieszczone w zmiennej *spi_data_in*, a potem w zmiennej *godziny*. Analogicznie można odczytywać zawartość pozostałych liczników. Wystarczy tylko zmienić adres. Zapisanie licznika zegarka jest jeszcze łatwiejsze. Po wysłaniu kodu 0x12 jest wysyłany adres rejestru, a po nim zapisywana dana, jak pokazano na **rysunkach 18** i **19**.

Umiemy już zapisywać i odczytywać rejestry zegarka, co pozwala na odczytywanie zliczanego czasu. Ale aby zegarek działał prawidłowo trzeba go najpierw ustawić. Procedury interfejsu użytkownika pochłaniają sporo pamięci programu i energii programisty. Nie inaczej jest i w tym wypadku. Musimy napisać procedurę, która:

- Odczyta rejestry minut i godzin.
- Zatrzyma liczniki zegarka.
- Przekonwertuje wartość liczników z kodu BCD na wartość binarną.
- Zmieni i wyświetli zmienioną wartość licznika godzin.
- Zmieni i wyświetli zmienioną wartość licznika minut.
- Przekonwertuje wartość liczników z wartości binarnej na BCD.
- Zapisze rejestry zegarka i uruchomi zliczanie.

Do zmiany wartości liczników jest używany impulsator. Kręcąc jego ośką w prawo lub lewo zwiększa się lub zmniejsza ustawianą wartość.

Na **rysunku 20** pokazano fragment procedury ustawiania godzin. Wcześniej program odczytał liczniki godzin i minut i umieścił je w zmiennych godziny i minuty. Potem została wykonana konwersja z kodu BCD (bo w tym kodzie liczą liczniki zegarka) na wartość binarną a wynik tej konwersji zapisano w zmiennej *pom*.

Jeżeli zostanie wykryty obrót ośki (zmienna f2=1), to są testowane zmienne *impu* i *impd* określające kierunek obrotu. Jeżeli jest ustawiona zmienna *impu*, to licznik godzin (przepisany do zmiennej *pom*) jest dekrementowany. Gdy po tej operacji ma wartość 0xff, to program wpisuje do niego wartość 23 (maksymalną wartość licznika godzin). Ustawienie zmiennej *impd* skutkuje inkrementowaniem licznika godzin. Tu również jest przeprowadzana korekta. Po osiągnięciu wartości 24 licznik jest zerowany. Po zmodyfikowaniu zmiennej jej nowa wartość jest zapisywana do układu zegarka. W tym celu jest ona konwertowana na BCD i zapisywana do licznika godzin, a następnie wyświetlana na ekranie LCD. Po przyciśnięciu osi impulsatora program kończy ustawianie godzin i przechodzi do bliźniaczej procedury ustawiania minut. Różni się od tej pokazanej na rys. 19, że zlicza modulo 59 a wynik zapisuje do zmiennej *minuty*.

Konwersje BCD->BIN i BIN->BCD są wykonywane za pomocą bloku obliczeń (**rysunek 21**). Użycie dodatkowej zmiennej tmp i taki, a nie inny zapis wynikają z pewnych ograniczeń kompilatora języka C użytego w pakiecie FlowCode. Musimy pamiętać, że algorytmy rysowane na planszy są tłumaczone na język C, a potem standardowo kompilowane. Naciśnięcie ośki po ustawieniu minut powoduje, że procedura ustawiania kończy działanie i zmodyfikowane liczniki już po konwersji do postaci BCD są zapisywane do zegarka. Zatrzymany zegarek wymaga uruchomienia przez zapisanie wyzerowanego licznika sekund z ustawionym bitem b7 (ST), uruchamiającym odmierzanie czasu. Zapisywanie rejestrów zegarka realizuje makro *write_time* pokazane na **rysunku 22**.

Podtrzymanie bateryjne zegarka

Układ MCP79512 ma wbudowany układ przełączania na zasilanie bateryjne po zaniku głównego napięcia zasilania. Przełączenie następuje w momencie, gdy VCC maleje i osiągnie poziom +1,5 V. Domyślnie doprowadzenie baterii jest wewnętrznie odłączone. Ma to na celu ograniczenie rozładowywania baterii. W praktyce nie wiadomo, kiedy napięcie zasilania zaniknie i trzeba Vbat dołączyć na stałe przez ustawienie bitu VBATEN umieszczonego na pozycji b3 w rejestrze licznika dni o adresie 0x04. Jest to wykonywane przez makro *en_vbat* pokazane na **rysunku 23**.

Podsumowanie

Głównym założeniem tego projektu było zbudowanie przydatnego urządzenia zaprogramowanego we Flowcode. Jak należało się spodziewać, program sterujący jest dość rozbudowany. Napisanie (narysowanie) takiego programu nie sprawiało dodatkowych trudności tylko z powodu stosowania FlowCode. Przeciwnie – wsparcie w postaci wbudowanych funkcji jeszcze ułatwiło programowanie. Jednak bezwzględnie trzeba pamiętać o korzystaniu z możliwości komentowania funkcji wykonywanych przez program i używać możliwości dodawania dodatkowych komentarzy wpisywanych w algorytm. Bez tego, nawet po krótkiej przerwie w pracy nad programem, trudno się rozeznać w bardziej rozbudowanych makrach.

Prezentowany tutaj program miał być w założeniu tak prosty, jak to tylko było możliwe. Dlatego nie optymalizowałem ilości użytych zmiennych tym bardziej, że pozostało dużo wolnej pamięci RAM. Nie wbudowałem też kilku potencjalnie przydatnych funkcji. Jedną z nich mogłaby by być funkcja kalibracji oscylatora kwarcowego. Można też się pokusić o funkcję alarmu, ale tu trzeba będzie rozwiązać problem wejścia do bardziej rozbudowanego menu ustawień. Mój zegar w porównaniu z dokładnym zegarem DCF77 spieszył się o ok. 1 sekundę na dobę. To dość sporo, bo po miesiącu daje to już 30 sekund. Wydaję się, że kalibracja mogłaby zmniejszyć tę niedogodność. Ponieważ „kod źródłowy" jest ogólnie dostępny, to każdy użytkownik dysponujący pakietem FlowCode będzie mógł go sobie zmodyfikować według własnych potrzeb. Procedury napisane przeze mnie wykorzystują mniej niż połowę pamięci programu mikrokontrolera PIC16F88 i dzięki temu można sobie „poszaleć".

Tomasz Jabłoński, EP