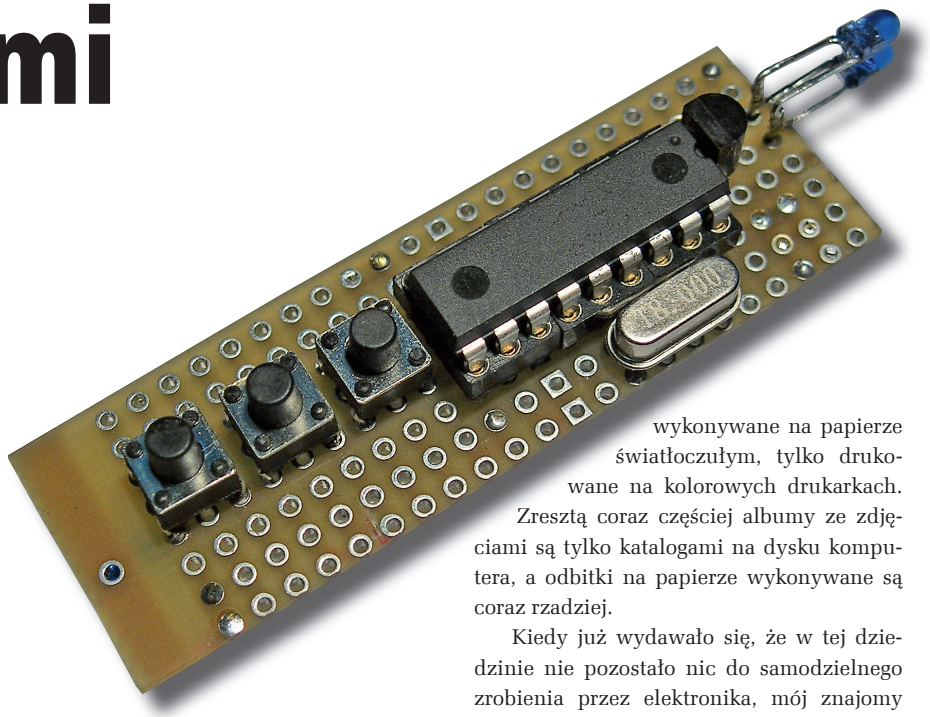
Pilot do zdalnego sterowania lustrzankami cyfrowymi



W EP8/2009 opisaliśmy narzędzie do graficznego tworzenia programów o nazwie Flowcode. Teraz prezentujemy projekt wykonany przy jego zastosowaniu. Wykonane urządzenie ma duże walory zarówno edukacyjne, jak i użytkowe. Patrząc na schemat, można by powiedzieć, że to Miniprojekt, ale za jego wykonaniem stoi ogromna wiedza teoretyczna. To jeszcze jedno urządzenie z tych, których siła tkwi nie tyle w sprzęcie, ile w oprogramowaniu.

Rekomendacje: *praktyczny kurs programowania we Flowcode oraz użyteczne urządzenie dla pasjonatów fotografii.*

Trudno zerwać ze starymi nawykami

W czasach, kiedy nieznane było pojęcie fotografii cyfrowej, amatorzy często sami wykonywali obróbkę zdjęć we własnej ciemni. Wywoływanie, szczególnie błon i odbitek barwnych, wymagało ścisłego zachowania temperatur chemikaliów i czasów poszczególnych etapów procesu. Dlatego popularne były elektroniczne programowane zegary ciemniowe pozwalające na precyzyjne odmierzanie tych czasów. Takie zegary były projektowane i wykonywane również przez elektroników hobbystów. Teraz, kiedy fotografia jest całkowicie cyfrowa, zdjęcia są zapisywane na karatach pamięci flash (najczęściej SD), potem przesyłane do komputera, gdzie mogą być poddane obróbce w programach graficznych. Odbitki nie są wykonywane na papierze światłoczułym, tylko drukowane na kolorowych drukarkach. Zresztą coraz częściej albumy ze zdjęciami są tylko katalogami na dysku komputera, a odbitki na papierze wykonywane są coraz rzadziej.

Kiedy już wydawało się, że w tej dziedzinie nie pozostało nic do samodzielnego zrobienia przez elektronika, mój znajomy poprosił mnie o wykonanie pilota zdalnego sterowania do lustrzanki cyfrowej firmy Pentax. W pierwszym momencie zacząłem się zastanawiać, po co zdalnie sterować aparatem cyfrowym? Okazało się, że powody są co najmniej dwa. Pierwszy to wykonanie autoportretu. Stajemy przed aparatem, naciskamy przycisk pilota i robimy sobie zdjęcie. Drugi powód to dążenie do perfekcyjnej ostrości wykonywanego zdjęcia. W momencie, kiedy wykonywane jest zdjęcie, niezbędne jest, aby aparat się nie poruszał. Dlatego najlepsze technicznie zdjęcia wykonywane są, kiedy jest on umocowany na statywie. Jednak żeby wykonać zdjęcie, trzeba nacisnąć spust migawki, a to jest źródłem drgań aparatu. Aby ich uniknąć, zwalnia się migawkę bezdotykowo za pomocą pilota zdalnego sterowania. Cyfrowe lustrzanki wyposażone są w możliwość takiego sterowania – mają odbiorniki podczerwieni i akceptują dedykowane komendy różniące się od komend systemów stosowanych w sprzęcie RTV.

Pierwszym problemem, z jakim przyszło mi się zmierzyć w trakcie opracowywania koncepcji pilota, było generowanie sygnału nośnej. Załóżmy, że nośna będzie miała częstotliwość 36 kHz. Okres takiego przebiegu to w przybliżeniu 28 µs. W czasie jednego okresu linia wyjściowa portu sterującego nadajnik musi mieć stan niski przez 14 µs i wysoki przez 14 µs. Najprościej byłoby ge-

AVT–5202

W ofercie AVT:
AVT–5202A – płytka drukowana

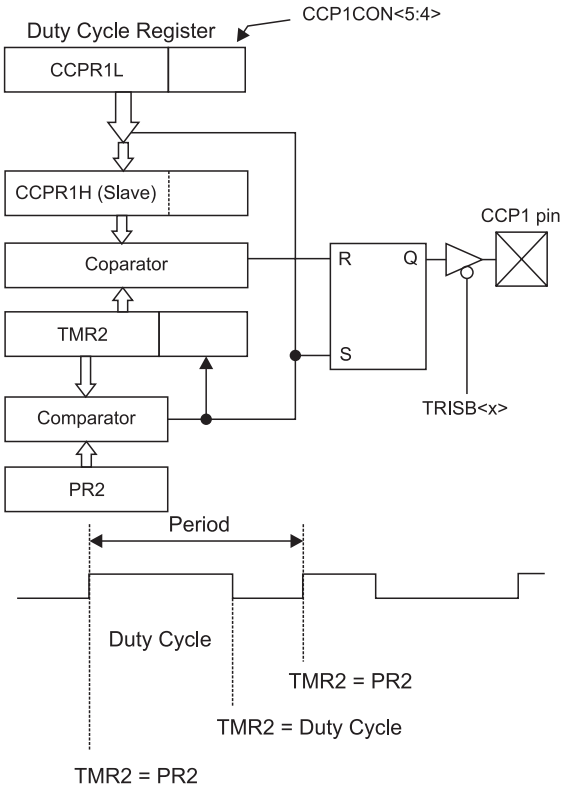
PODSTAWOWE PARAMETRY

- Płytka o wymiarach: 16×73 mm
- Zasilanie: np. 3,6 V (3 akumlatory AA)
- Do użytku z aparatami Canon, Nikon, Pentax
- Mikrokontroler PIC16F628A

PROJEKTY POKREWNE wymienione artykuły są w całości dostępne na CD

Tytuł artykułu	Nr EP/EdW	Kit
Wyzwalacz flesza fotograficznego	EdW 1/1997	—

Na CD karty katalogowe i noty aplikacyjne elementów oznaczonych na Wykazie Elementów kolorem czerwonym



Rys. 1. Schemat blokowy modułu PWM i sygnał wyjściowy

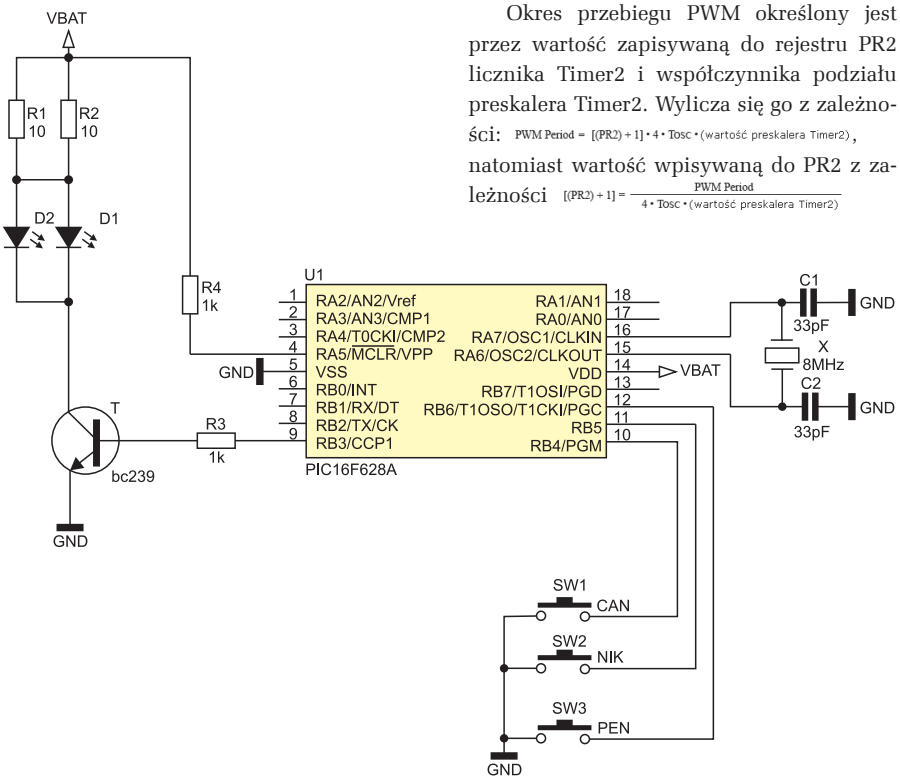
nerować przebieg w przerwaniu zegara zgłaszanym co 14 µs.

Do sterowania chciałem wykorzystać mikrokontroler PIC16 taktowany częstotliwością 8 MHz. Cykl maszynowy trwa w nim 0,5 µs. Łatwo obliczyć, że pomiędzy kolejnymi przerwaniami mikrokontroler może wykonać 28 rozkazów. Jest to zdecydowanie za mało, aby obsłużyć przerwania i wykonywać jakieś czynności poza nimi. Można było zwiększyć częstotliwość taktowania – PIC16 można taktować sygnałem o częstotliwości do 20 MHz, jednak również może to się okazać zbyt mało do poprawnego działania. Ponadto, zwiększenie częstotliwości taktowania powoduje zwiększenie poboru prądu, co nie jest korzystne dla urządzenia zasilanego bateryjnie.

Pozostało porzucenie idei wykorzystania przerwań i rozważenie odliczania opóźnień przez zliczanie w pętli cyklów rozkazowych. Dla 14 µs trzeba odliczyć 28 cykli przy fxtal=8 MHz. Takie rozwiązanie jest proste i skuteczne, ale dla precyzyjnego odmierzania czasu trzeba procedurę napisać w asemblerze. Już przystąpiłem do realizacji tego pomysłu, kiedy przyszedł mi do głowy inny pomysł.

Microchip ma w swojej ofercie bardzo dobrze wyposażone w układy peryferyjne mikrokontrolery PIC16F. Do sprzętowego generowania fali prostokątnej o programowanej częstotliwości można wykorzystać wyjście PWM. W mikrokontrolerach PIC16F sygnał PWM jest generowany w rozbudowanym module sprzętowym CCP (*Capture/Compare PWM*). Schemat blokowy modułu zaprogramowanego do pracy jako generator PWM pokazano na **rys. 1**.

Okres przebiegu PWM określony jest przez wartość zapisywaną do rejestru PR2 licznika Timer2 i współczynnika podziału preskalera Timer2. Wylicza się go z zależności: $\text{PWM Period} = [(PR2)+1] \cdot 4 \cdot Tosc \cdot (\text{wartość preskalera Timer2})$, natomiast wartość wpisywaną do PR2 z zależności $[(PR2)+1] = \frac{\text{PWM Period}}{4 \cdot Tosc \cdot (\text{wartość preskalera Timer2})}$



Rys. 2. Schemat elektryczny pilota

WYKAZ MATERIAŁÓW

Rezystory
R1, R2: 10 Ω/0,25 W, SMD 1206
R3, R4: 1 kΩ, SMD 1206

Kondensatory
C1, C2: 33 pF

Półprzewodniki
U1: PIC16F628A I/P zaprogramowany
D1,D2: diody LED IR
T: BC239

Inne
X – rezonator kwarcowy 8 MHz
Mikroprzyciski 3 szt.
Pojemnik na baterie

Dla 36 kHz PWM Period=28 µs, a dla Fosc=8 MHz 4×Tosc=0,5 µs. Do PR2 trzeba wpisać 28 µs /0,5 µs – 1=55, przy założeniu, że preskaler jest wyłączony (równy 1). W ten sposób ustalany jest okres przebiegu PWM. W następnym kroku trzeba ustalić współczynnik wypełnienia przebiegu na 50%. Wartość tego współczynnika programuje się z rozdzielczością 10 bitów, zapisując 8-bitowy rejestr CCPR1L (8 młodszych bitów) i 2 bity <5:4> rejestru CCP1CON według zależności

$\text{PWM Duty Cycle} = (CCPR1L:CCP1CON<5:4>) \cdot 4 \cdot Tosc \cdot (\text{wartość preskalera Timer2})$

w której *PWM Duty Cycle* to czas trwania stanu wysokiego. Jeżeli preskaler będzie wyłączony, to otrzymamy

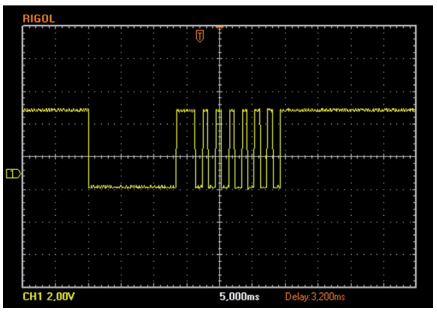
$(CCPR1L:CCP1CON<5:4>) = \frac{\text{PWM Duty Cycle}}{4 \cdot Tosc}$

Jeżeli okres PWM trwa 28 µs, to dla wypełnienia 50% *PWM Duty Cycle* będzie trwał 14 µs. W takim wypadku wartość wpisywana do (CCPR1L:CCP1CON<5:4>) będzie równa 14 µs/0,5 µs=28. Tak skonfigurowany moduł generuje przebieg prostokątny o częstotliwości 36 kHz i wypełnieniu 50% **sprzętowo** i nie zajmuje czasu mikrokontrolera.

Do wykonania pilota wybrałem tani i dostępny mikrokontroler PIC16F628A. Ma wszystko, co potrzeba: moduł CCP i niewielką obudowę z 18 wyprowadzeniami. Schemat pilota pokazano na **rys. 2**.

„Rysowanie" programu

Oscylogram komendy zwolnienia migawki aparatu Pentax pokazano na **rys. 3**. Komenda rozpoczyna się od stanu niskiego trwającego 13,2 ms. Potem jest stan wysoki trwający 2,9 ms i 7 cykli składających się ze



Rys. 3. Oscylogram komendy zwalniania migawki Pentax

stanu niskiego trwającego 1 ms i wysokiego trwającego 900 µs.

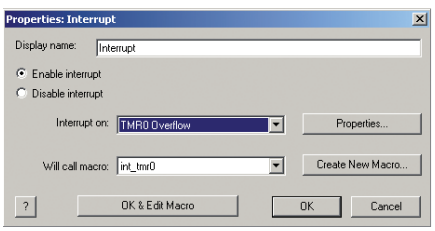
Początkowo miałem napisać program sterujący w języku C, jednak w końcu zamiast pisać narysowałem go w środowisku Flowcode firmy MatrixMultimedia. Flowcode był już opisywany na łamach „Elektroniki Praktycznej", a ten projekt jest przykładem zastosowania tego bardzo ciekawego narzędzia.

Żeby ułatwić sobie pracę, podzieliłem zadania na funkcjonalne fragmenty. W języku C fragmenty realizujące poszczególne zadania są umieszczane w procedurach wywoływanych z programu głównego. Większe zadania można też umieścić w oddzielnych plikach łączonych po kompilacji przez linker. We Flowcode odpowiednikami funkcji są tak zwane makra. Makro to fragment programu narysowany na oddzielnej planszy. Może ono mieć argumenty wejściowe (więcej niż jeden) i może zwracać zmienne. Oprócz makr programowych w pakiet zostały wbudowane wykonane przez producenta gotowe makra komponentów, pozwalające programować układy peryferyjne bez konieczności żmudnego „grzebania" w rejestrach. Będziemy również korzystać z takich makr przy programowaniu modułu PWM i licznika TMR0.

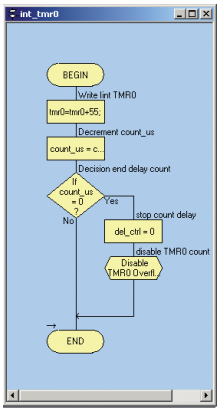
Jeżeli spojrzymy na oscylogram na rys. 3, to zobaczymy, że konieczne będzie odliczanie opóźnień z rozdzielczością 100 µs. Flowcode ma wbudowaną funkcję odliczania opóźnień z rozdzielczością 1 ms, jednak jest to zbyt mało nasze potrzeby. Do odliczania opóźnień można wykorzystać na przykład licznik Timer0 zgłaszający przerwania co 100 µs.

Licznik Timer0 będzie zliczać impulsy o częstotliwości Fosc/4. Żeby przepełniał się co 100 µs, musi zliczyć 200 impulsów o okresie 0,5 µs, bo tyle trwa jeden okres przebiegu o częstotliwości 2 MHz (8 MHz/4). Flowcode ma możliwość obsługi przerwań i w oparciu o przerwanie od przepełnienia TMT0 zostanie zbudowane makro odliczające opóźnienie.

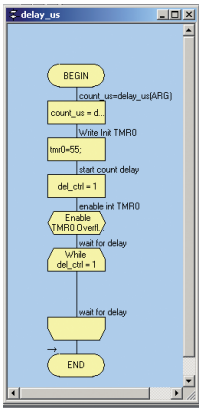
Obsługę przerwania rozpoczniemy od jego odblokowania. Na schemacie stawiamy ikonę INT i z jej menu (otwieranego po kliknięciu prawym klawiszem) wybieramy zakładkę Properties (**rys. 4**). Konfiguracja polega na wybraniu odblokowania (*Enable interrupt*), wybraniu rodzaju przerwania



Rys. 4. Odblokowanie i konfiguracja przerwania od przepełnienia TMR0



Rys. 5. Makro obsługi przerwania od Timer0

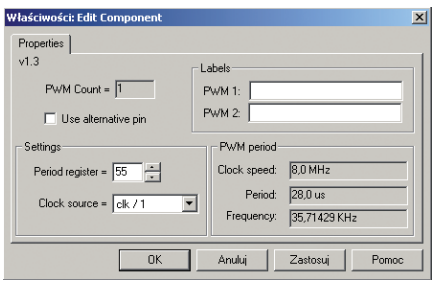


Rys. 6. Makro funkcji odliczania opóźnień

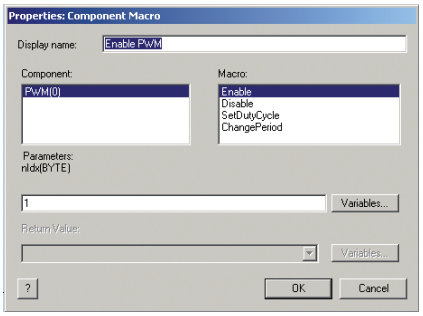
(*TMR0 Overflow*) i skojarzeniu go z makrem wykonywanym w momencie wywołanie, przerwania. Cała operacja trwa kilka sekund. Jeżeli naciśniemy przycisk *OK&Edit Macro*, to program otworzy planszę nowego makra o nazwie *int_tmr0*, w którym będzie można narysować obsługę przerwania.

O przechowanie wartości rejestrów i programowe zerowanie flagi przerwania troszczy się Flowcode i nie musimy się tym zajmować. Kompletne makro obsługi przerwania pokazano na **rys. 5**. Na początku, do rejestru licznika wpisywana jest wartość 55, tak by licznik przepełnił się po zliczeniu 200 impulsów (licznik liczy w przód). Każde przerwanie powoduje dekrementację zmiennej *count_us*. Po wyzerowaniu *count_us* program zeruje zmienną *del_ctrl* i wywołuje makro blokujące przyjmowanie przerwań. Idea odliczania opóźnień wygląda następująco: po wpisaniu do zmiennej *count_us* liczby odmierzanych interwałów 100-µsekundowych, wpisuje się do zmiennej *del_ctr* jedynkę, odblokowuje przerwania i czeka, aż zmienna *del_ctrl* zostanie wyzerowana i przerwania zostaną zablokowane. Makro funkcji odliczania opóźnień pokazano na **rys. 6**. Do zmiennej *count_us* wpisywana jest wartość argumentu funkcji reprezentowanej przez makro. To, czy makro ma argumenty, definiuje się w trakcie tworzenia makra, w zakładce *Macro –> New*.

Konfiguracja układu CCP w trybie PWM również nie jest skomplikowana. Na pasku komponentów umieszczona jest ikona PWM.



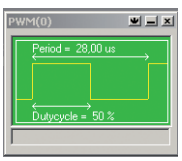
Rys. 7. Okno właściwości komponentu PWM



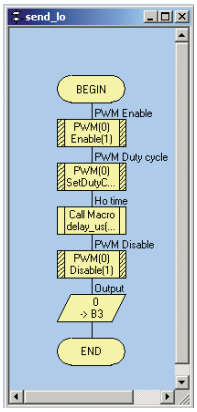
Rys. 8. Okno właściwości komponentu PWM

Po kliknięciu na nią otwiera się okienko PWM i Flowcode automatycznie udostępnia funkcje odblokowania i konfiguracji interfejsu. Najpierw skonfigurujemy czas trwania okresu przebiegu PWM w okienku właściwości – **rys. 7**. Mając na uwadze wcześniej przeprowadzone wyliczenia, w okienku Period Register wpisujemy 55. Ta wartość zostanie wpisana do rejestru PR2. Preskaler Timer2 ustawiamy w okienku *Clock source* (preskaler=1). Obok, w polu *PWM Period*, jest automatycznie wyliczany okres przebiegu i częstotliwość dla zegara równego 8 MHz. Po tej definicji możemy postawić na planszy makro komponentu i z jego właściwości wybrać jedną z funkcji przypisanych do komponentu PWM – **rys. 8**. Interfejs PWM trzeba odblokować funkcją *Enable* z parametrem 1 (numer odblokowanego PWM), a potem ustalić współczynnik wypełnienia funkcją *SetDutyCycle* z parametrami 1 (numer aktywnego PWM) i wartością 28, ustalającą czas trwania czasu wysokiego na 14 µs. W wyniku otrzymamy przebieg o okresie 28 µs i czasie trwania stanu wysokiego równym 14 µs (współczynnik wypełnienia 50%). Wynik tych działań można zobaczyć w okienku komponentu – **rys. 9**.

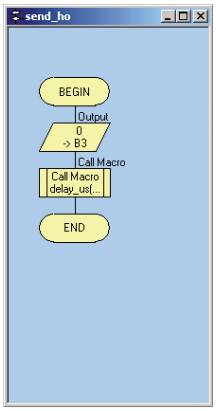
Do wygenerowania komendy spustu migawki pokazanej na oscylogramie (rys. 3) potrzebne będą 2 makra funkcyjne. Pierwsze to *send_lo* wysyłające stan niski przez określony czas, a drugie *send_ho* wysyłające stan wysoki przez określony czas. Stan niski na wyjściu odbiornika podczerwieni jest wymuszany, kiedy odbiornik odbiera zmodulowany sygnał o częstotliwości 36 kHz. Wysyłanie stanu niskiego będzie polegało na wysyłaniu sygnału 36 kHz przez czas określony w argumencie wywołania. Najpierw jest odblokowywany moduł PWM i programowany współczynnik wypełnienia. Potem makro odmierza czas określony przez swój argument. Po zakończeniu odmierzania interfejs



Rys. 9. Okno zaprogramowanego komponentu PWM



Rys. 10. Makro wysyłania zera

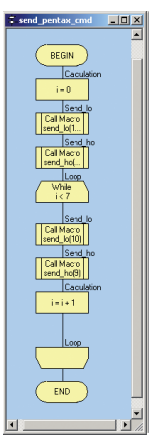


Rys. 11. Makro wysyłania jedynki

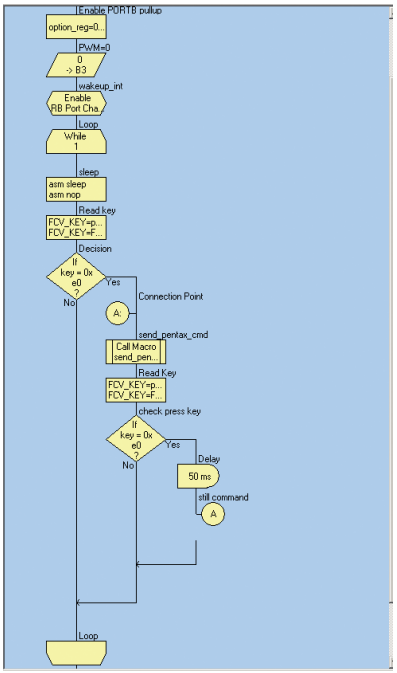
PWM jest blokowany. Blokowanie interfejsu PWM jest konieczne, ponieważ gdy nie jest wysyłane zero logiczne, to na wyjściu PWM musi być stan niski, aby diody nadawcze LED się nie świeciły. Makro wysyłające zera logiczne *send_lo* pokazano na **rys. 10**, a wysyłające jedynkę na **rys. 11**. Funkcję wysyłania komendy spustu migawki aparatu Pentax pokazano na **rys. 12**.

Skoro mamy już wszystkie niezbędne makra, to można pomyśleć o narysowaniu pętli głównej programu. W tej pętli będzie sprawdzane, czy został naciśniety przycisk wysłania komendy. Jeżeli tak, to komenda zostanie wysłana. Po wysłaniu program sprawdza, czy dalej jest wciśnięty przycisk – jeżeli tak, to program odlicza opóźnienie i cyklicznie wysyła komendy, aż do momentu puszczenia przycisku. W tym momencie można by zakończyć pracę na projektem, gdyby nie jeden problem. Pilot zdalnego sterowania musi być zasilany z baterii. Ciągle pracujący mikrokontroler, nawet kiedy pozornie nic nie robi, pracując w nieskończonej pętli, pobiera z baterii zdecydowanie za duży prąd. Ponieważ przez większość czasu program nic nie robi, to aby zmniejszyć pobór mocy, można uśpić mikrokontroler.

W mikrokontrolerach PIC16 tryb oszczędzania energii jest wprowadzany rozkazem *sleep*. Układ można wybudzić ze stanu uśpienia przez sprzętowe zerowanie (stan wysoki na MCLR), lub przez przerwania: zewnętrzne INT, od zmiany stanów na wejściach portu PORTB i od układów peryferyjnych odblokowywanych bitem PEIE. W naszym przypadku idealnym będzie przerwanie od zmian na liniach portu PORTB. Żeby było to możliwe, trzeba odblokować przerwanie i w procedurze obsługi odczytać stan linii portów PORTB. Odblokowanie przerwania i tworzenie makra obsługi przerwania robi się tak samo, jak w przypadku opisywanego już przerwania od przepełnienia licznika TMR0. Na **rys. 13** pokazano pętlę główną programu wysyłającego komendę zwolnienia migawki Pentax. Przed jej wywołaniem odblokowane jest przerwanie od zmian na liniach PORTB oraz wymuszany jest stan niski na linii RB3 (wyjście PWM). Pozostawienie tej linii jako wejściowej powoduje, że układ pobiera ok. 30 mA prądu, bo tranzystor T1 przechodzi w stan pracy liniowej, a powinien być odcięty. Przez diody D1 i D2 płynie prąd do czasu naciśnięcia klawisza i zainicjowania modułu PWM, który w stanie nieaktywnym ustawia RB3 w stan niski.



Rys. 12. Funkcja wysyłania komendy

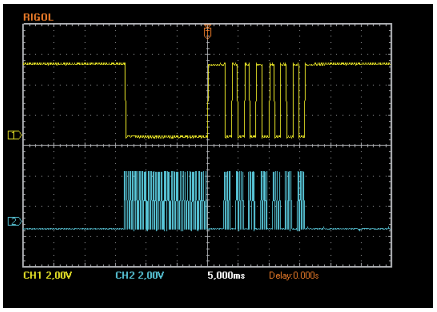


Rys. 13. Pętla główna programu

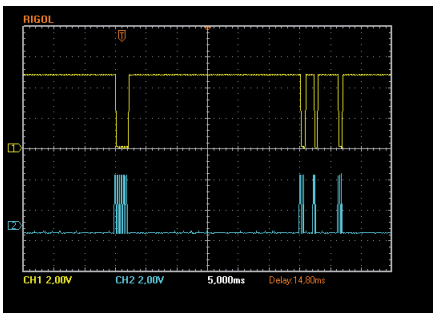
Konfiguracja PWM i przerwania od Timer0 jest wykonywana przez makra programowe. Pierwszą instrukcją pętli jest rozkaz *sleep*. Jest on umieszczany pod symbolem ikony wprowadzania kodu C, pod którą można również wprowadzić polecenia zapisane w asemblerze: *asm sleep*. Po wykonaniu rozkazu *sleep* mikrokontroler jest usypiany i zmniejsza pobór prądu. Naciśnięcie przycisku podłączonego do linii PORTB powoduje zgłoszenie przerwania i wybudzenie mikrokontrolera. Jeżeli zostanie zidentyfikowane naciśnięcie przycisku podłączonego do RB6, to wysyłana jest komenda wyzwolenia migawki. Jeżeli po wysłaniu komendy klawisz jest nadal wciśnięty, to po odczekaniu opóźnienia wysyłane są cyklicznie komendy aż do stwierdzenia puszczenia klawisza. Mikrokontroler ponownie wchodzi w stan uśpienia i czeka na kolejne przerwanie.

Budowa układu

Modelowy pilot został wykonany na kawałku płytki uniwersalnej. Mała liczba elementów i połączeń powoduje, że taki montaż jest prosty i szybki. Uruchomienie pilota nie powinno nastręczać większych problemów. Po włożeniu w podstawkę lub przylutowaniu zaprogramowanego mikrokontrolera trzeba włączyć zasilanie. W trakcie testów układ był zasilany z trzech akumulatorków AAA połączonych szeregowo. Daje to napięcie zasilania ok. 3,5…3,8 V. Prąd pobierany z baterii, kiedy przyciski pilota nie są naciskane, nie powinien przekraczać 50...60 µA. W czasie, kiedy pilot wysyła zmodulowany sygnał komendy, pobór prądu wzrasta do ok. 11 mA.



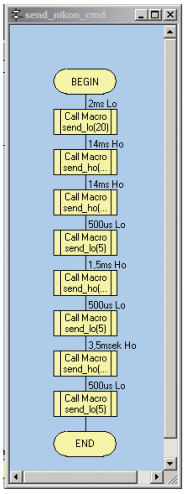
Rys. 14. Oscylogramy odbieranej i nadawanej komendy do aparatu Pentax



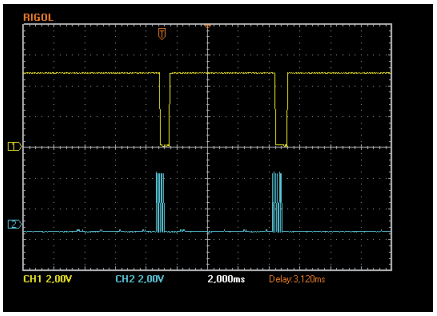
Rys. 15. Oscylogram komendy dla aparatu Nikon

Pracę urządzenia można sprawdzić za pomocą dwukanałowego oscyloskopu. Na **rys. 14** pokazano oscylogram, w którym w górnym kanale jest sygnał z wyjścia odbiornika TSOP1736, a w dolnym kanale sygnał z wyjścia PWM (RB3) mikrokontrolera. Końcowy test jest przeprowadzany w trakcie współpracy z aparatem fotograficznym. Zasięg nadawania jest dosyć spory. Testowany egzemplarz wyzwalał migawkę z odległości ok. 10 m.

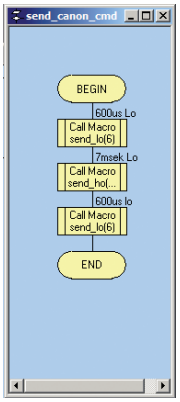
Aparaty firmy Pentax, mimo że cieszą się uznaniem, nie są najbardziej popularne na rynku. Częściej spotykane są dwie inne marki: Canon i Nikon. Skoro pilot wysyła komendę do Pentaksa, to nic nie stoi na przeszkodzie, żeby wysy-



Rys. 16. Makro komendy aparatu Nikon



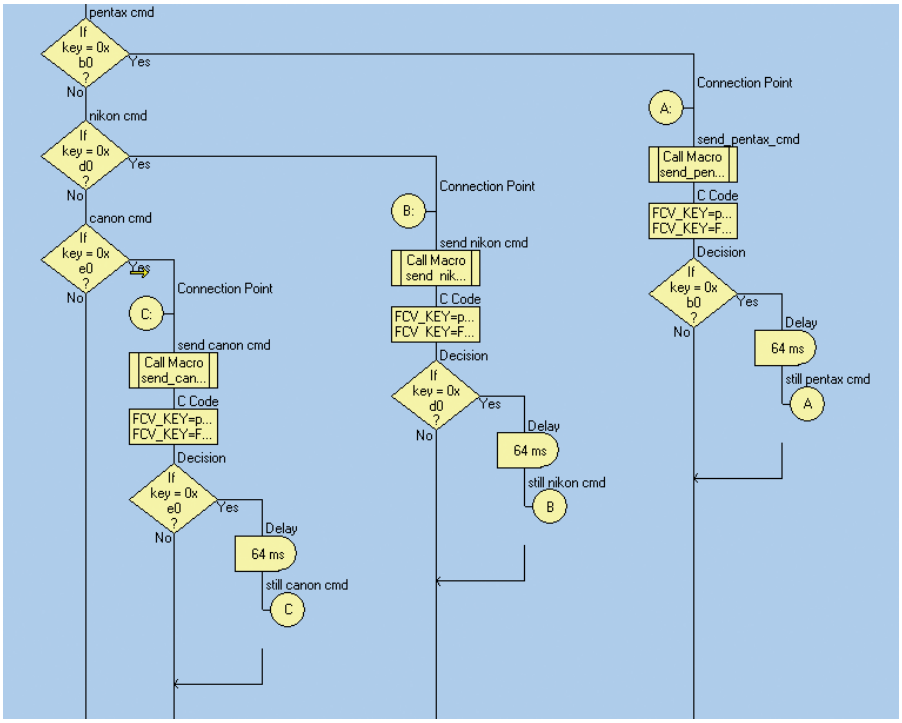
Rys. 17. Oscylogram komendy aparatu Canon



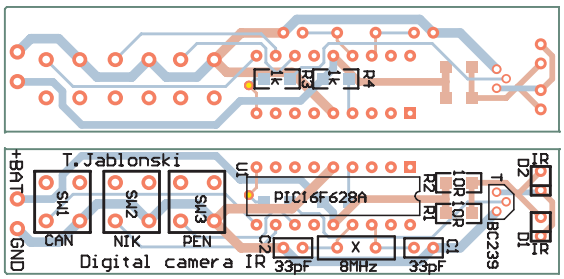
Rys. 18. Makro komendy aparatu Canon

łać komendy do innych aparatów. W trakcie projektowania układu przewidziano trzy przyciski do wysyłania trzech różnych komend. Każdy z przycisków miał być dedykowany dla komend różnych producentów aparatów. Na oscylogramie na **rys. 15** pokazano komendę dla aparatu Nikon. Rozpoczyna ją stan niski trwający ok. 2 ms. Potem jest stan wysoki trwający 28 ms; stan niski 500 µs; wysoki 1,5 ms; niski 500 µs; wysoki 3,5 ms i sekwencja kończy się stanem niskim trwającym 500 µs. Narysowanie tak wykonywanej komendy, kiedy już mamy mechanizmy odliczania opóźnień i generowania nośnej, nie sprawi żadnego problemu. Przykład realizacji przedstawiono na **rys. 16**. Podobnie jest z komendą dla aparatu Canon. Jej oscylogram przedstawiono na **rys. 17**, a przykład realizacji na **rys. 18**.

Na **rys. 19** pokazano kompletną główną pętlę programu. Po naciśnięciu przycisku i wybudzeniu ze stanu uśpienia program testuje, który klawisz został naciśnięty i wysyła przypisaną mu komendę.



Rys. 19. Kompletna pętla główna dla trzech aparatów



Rys. 20. Schemat montażowy układu

Zastosowanie mikrokontrolera umożliwia dalszą ewentualną rozbudowę pilota. Można na przykład dodać funkcję ciągłego wysyłania kodów przez określony czas, przydatną dla funkcji B. Można też dodać moduł radiowy i zwiększyć zakres działania urządzenia do kilkuset metrów. Jest to przydatne na przykład przy fotografowaniu dzikich zwierząt.

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