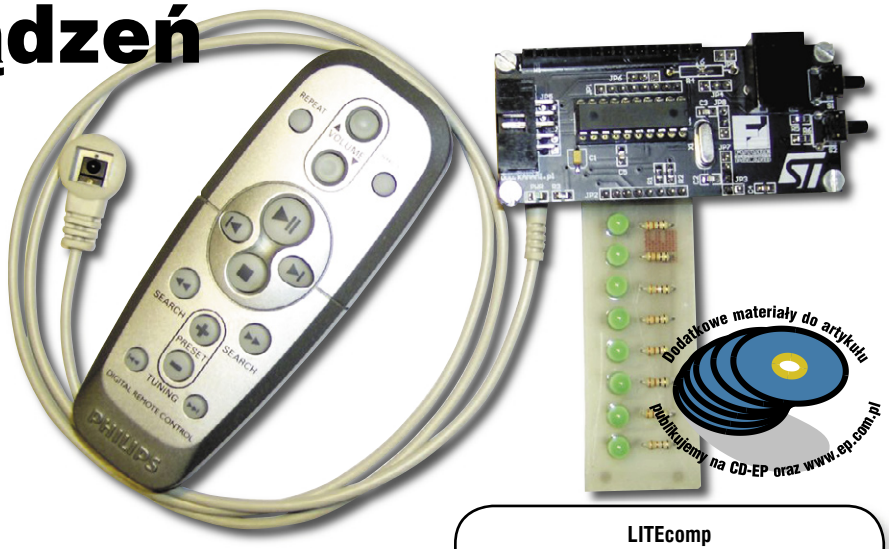
LITEcomp – aplikacje

Zdalnie sterowany włącznik ośmiu urządzeń



Pilot zdalnego sterowania używany jest od wielu lat do sterowania domowym sprzętem RTV. Ostatnimi czasy piloty stosowane są również do sterowania w systemach automatyki domowej, jak również w najróżniejszych urządzeniach budowanych przez elektroników hobbystów. W artykule przedstawiamy projekt ośmiokanałowego włącznika sterowanego pilotem pracującym w kodzie RC5 zbudowanego na bazie modułu LITEcomp.

st7.ep.com.pl

Prezentowany włącznik ten może sterować maksymalnie ośmioma wyjściami. W zależności od typu sterowanych urządzeń jako stopnie wyjściowe można zastosować tranzystory mocy, przekaźniki czy też triaki (koniecznie w tym przypadku należy dodać obwody optoizolacji). Można układ potraktować jako urządzenie typowo „edukacyjne" i zamiast stopni mocy zastosować diody LED symulujące sterowane urządzenia.

Schemat układu przedstawiono na **rys. 1**. Wyświetlacz LCD należy odłączyć od modułu LITEcomp, gdyż nie będzie wykorzystywany w tym przykładzie. Jako element odbiorczy zastosowano układ TSOP1736, który odbiera sygnał podczerwieni modulowany częstotliwością 36 kHz. Należy pamiętać, że układ ten daje na wyjściu zanegowany, w stosunku do odebranego, przebieg sygnału.

Przykładowa ramka danych (adres:0; rozkaz:0) transmitowana w kodzie RC5 na wyjściu układu TSOP1736 została przedstawiona na

LITEcomp

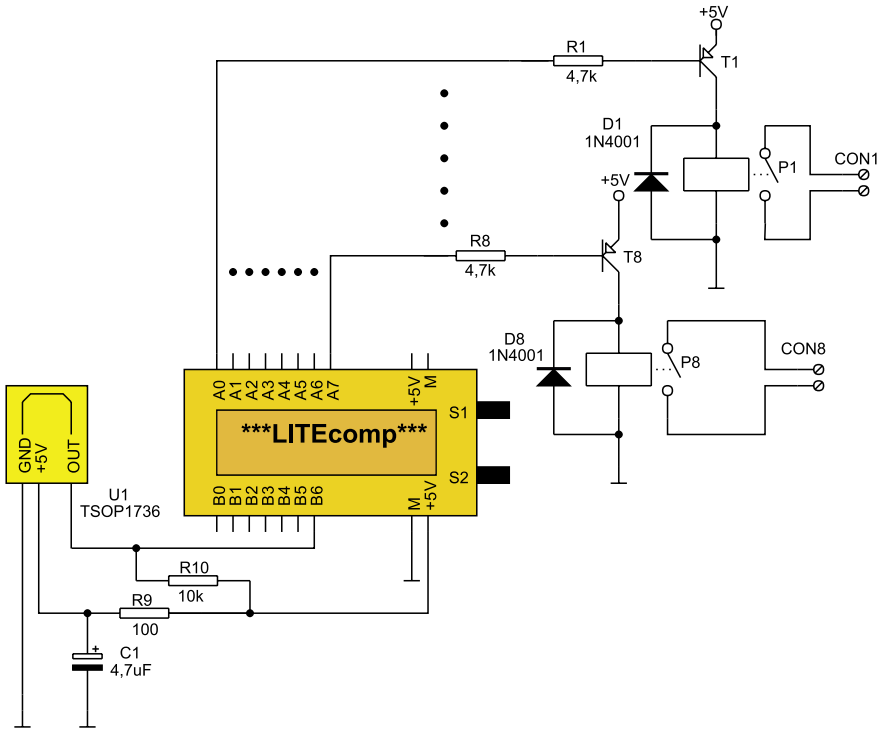
LITEcomp jest prostym komputerkiem wykonanym na mikrokontrolerze ST7FLITE19. LITEcomp jest w ramach promocji dodawany bezpłatnie do książki „Mikrokontrolery ST7LITE w praktyce" (autor Jacek Bogusz). Książka jest dostępna w *sklep.avt.pl* (numer katalogowy KS–260905).

R E K L A M A

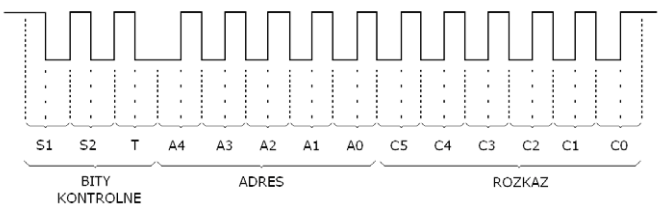
Robot na LITEcompie

W kolejnej EP przedstawimy opis prostego samobieżnego robota, którego "mózgiem" jest ***LITEcomp***.

Prezentacje wideo będą dostępne na *www.ep.com.pl*.



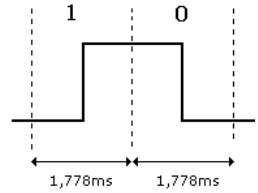
Rys. 1. Schemat podłączenia LITEcompa



Rys. 2. Ramka RC5

rys. 2. Jeden bit, niezależnie od transmitowanej wartości trwa 1,778 ms. Ramka danych składa się z 14 bitów, z czego użyteczną informację niosą bity 4…14. Dwa pierwsze bity mają stałą wartość „1” i służą jako znacznik początku ramki, natomiast bit trzeci ma za zadanie wskazywać, czy kolejne transmitowane ramki pochodzą od kolejnych naciśnięć przycisku na pilocie, czy są skutkiem jego dłuższego przytrzymania. Dane przesyłanie w standardzie RC5 są zakodowane w kodzie Manchester. Logiczna „1” jest zakodowana jako przejście ze stanu niskiego do wysokiego, natomiast logiczne zero jako przejście ze stanu wysokiego do niskiego, co pokazano na **rys. 3**.

Wyjście danych odbiornika podczerwieni podłączone jest do wyprowadzenia PB6 modułu LITEcomp. Opadające zbocze na tym wyprowadzeniu sygnalizuje początek transmitowanej ramki danych i wywołuje przerwanie. Przerwanie to występuje podczas transmitowanej ramki tylko raz – po jego pierwszym wystąpieniu następuje jego zablokowanie na czas trwania ramki danych. W przerwaniu tym następuje również włączenie timera AT2 oraz zezwolenie na przerwanie od niego. Dane są odbierane w ramach procedury obsługi przerwania timera taktowanego sygnałem o częstotliwości 8 MHz (Fosc/2). Przerwanie to występuje co 444 mikrosekundy, a więc czterokrotnie w podczas jednego bitu. Moment wystąpienia



Rys. 3. Sposób kodowania bitu w RC5

przerwania od timera oraz momenty próbkowania sygnału przedstawiono na **rys. 4**. Wynika z niego, że w zasadzie każdy bit może zostać spróbkowany maksymalnie dwa razy, gdyż co drugie wystąpienie przerwania od timera następuje w chwili potencjalnej zmiany stanu na wejściu. W naszym przypadku jedynym bitem próbkowanym dwa razy jest bit S2. Dzięki temu możemy stwierdzić, czy nadawana ramka jest w standardzie RC5 poprzez sprawdzenie czy w chwili 3 na wejściu będzie panował stan wysoki. Jest to charakterystyczna cecha kodu RC5, która pozwoli na odróżnienie czy odebrany sygnał jest nadawany w kodzie RC5 czy też w jakimś innym kodzie. Gdybyśmy zrezygnowali ze sprawdzenia sygnału w chwili 3 i ograniczyli się wyłącznie do chwil 1 i 5 mogłaby zaistnieć sytuacja, w której sygnał nadawany np. w kodzie NEC (charakteryzujący się niskim poziomem logicznym przez pierwsze 9,4 ms) zostałby rozpoznany jako kod RC5. W przypadku stwierdzenia niezgodności odebranej ramki ze standardem RC5 ustawiane są bity błędów. Każda z trzech pierwszych próbek ma własny bit błędu w zmiennej statusowej. Ustawienie któregokolwiek bitu błędu świadczy o tym, że należy odebrane dane zignorować.

List. 1. Deklaracje zmiennych wykorzystywanych w programie

```
BYTES
     SEGMENT ‚ram0’
tickCount       ds.b 1   ; licznik wystąpień przerwania od timera
RC5Command   ds.b 1  ; odebrany rozkaz RC5
RC5Address   ds.b 1  ; odebrany adres RC5
RC5Status    ds.b 1  ; status
RC5StatusT   ds.b 1  ; status pomocniczy
OutNumber ds.b 1  ; numer aktywnego wyjścia
     WORDS
     SEGMENT ‚eeprom’
OutAddress   ds.b 8  ; tablica adresów
OutCommand   ds.b 8  ; tablica rozkazów
IsProgrammed ds.b 1  ;
```

List. 2. Procedura obsługi przerwania zewnętrznego

```
._EI2Vect
  CLR     RC5Command ; wyzerowanie poprzednio odebranego rozkazu
  CLR     RC5Address ; wyzerowanie poprzednio odebranego adresu
  LD    A, #%10100000;
  LD    RC5Status, A ; ustawienie początkowej wartości bajtu statusu
  BRES     PBOR, RC5Pin ; zablokowanie przerwania zewnętrznego
  LD    A, #18     ; włączenie timera oraz zezwolenie na
  LD    ATCSR, A   ; przerwanie od jego przepełnienia
  IRET
```

List. 3. Procedura obsługi przerwania od timera AT2

```
._ATOvfVect
  LD    A, ATCSR      ; zerowanie flagi przerwania
  INC     tickCount       ; zwiększenie licznika wystąpień przerwania
  LD    A, tickCount    ;
  CP    A, #57        ; czy to już koniec ramki?
  JREQ    EndFrame        ; jeśli tak to skok do EndFrame
  BTJT    RC5Port, RC5Pin, Exit; jeśli wejście RC5 jest w stanie wysokim
                  ; to skok do Exit
; w przeciwnym razie....
;Początek ramki sygnału RC5
BitS1               ; bit S1
  CP    A, #1
  JRNE    BitS20
  BRES    RC5Status, RC5ErrS1
  IRET
BitS20                ; pierwsza połówka bitu S2
  CP    A, #3
  JRNE    BitS2
  BSET    RC5Status, RC5ErrS20
  IRET
BitS2               ; druga połówka bitu S2
  CP    A, #5
  JRNE    BitT
  BRES    RC5Status, RC5ErrS21
  IRET
BitT              ; bit T
  CP    A, #9
  JRNE    BitA4
  BSET    RC5Status, RC5Toggle
  IRET
BitA4               ; bit A4
  CP    A,#13
  JRNE    BitA3
  BSET    RC5Address, #4
  IRET
BitA3               ; bit A3
  CP    A,#17
  JRNE    BitA2
  BSET    RC5Address, #3
  IRET
BitA2               ; bit A2
  CP    A,#21
  JRNE    BitA1
  BSET    RC5Address, #2
  IRET
BitA1               ; bit A1
  CP    A, #25
  JRNE    BitA0
  BSET    RC5Address, #1
  IRET
BitA0               ; bit A0
  CP    A, #29
  JRNE    BitC5
  BSET    RC5Address, #0
  IRET
BitC5               ; bit C5
  CP    A, #33
  JRNE    BitC4
  BSET    RC5Command, #5
  IRET
BitC4               ; bit C4
  CP    A, #37
  JRNE    BitC3
  BSET    RC5Command, #4
  IRET
BitC3               ; bit C3
  CP    A, #41
  JRNE    BitC2
  BSET    RC5Command, #3
  IRET
BitC2               ; bit C2
  CP    A, #45
  JRNE    BitC1
  BSET    RC5Command, #2
  IRET
BitC1               ; bit C1
  CP    A, #49
  JRNE    BitC0
  BSET    RC5Command, #1
  IRET
BitC0               ; bit C0
  CP    A, #53
  JRNE    EndFrame
  BSET    RC5Command, #0
Exit
  IRET
EndFrame              ; koniec ramki sygnału RC5
  LD    A, tickCount
  CP    A, #57        ; sprawdzenie, czy aby na pewno
  JRNE    Exit        ; jeśli nie to skok do Exit
  CLR     tickCount       ; wyzerowanie licznika wystąpień przerwania
  CLR     ATCSR     ; zatrzymanie timera
  LD    A, RC5Status
  AND     A, #%11100000     ; sprawdzenie czy wystąpiły błędy
  JRNE    NoReady       ; jeśli tak to skok do NoReady
  BSET    RC5Status,RC5Ready  ; w przeciwnym razie ustawienie flagi gotowości
NoReady
  BSET    PBOR, RC5Pin    ; zezwolenie na przerwanie zewnętrzne EI2
  JRA     Exit
```

Zmienne używane przez program

Program używa 6 bajtów pamięci RAM oraz 17 bajtów pamięci EEPROM. W pamięci RAM przechowywane są następujące zmienne:

tickCount – licznik wystąpień przerwania od timera,

RC5Command – odebrany rozkaz kodu RC5,

RC5Address – odebrany adres kodu RC5,

RC5Status – status ostatniej transmisji RC5,

RC5StatusT – status pomocniczy (przechowywana jest w nim wartość poprzednio odebranego bitu T),

OutNumber – numer sterowanego wyjścia.

W pamięci EEPROM znajdują się dwie 8–bajtowe tablice:

OutAddress – przechowująca adresy RC5 odpowiadające poszczególnym wyjściom, oraz

OutCommand – przechowująca rozkazy kodu RC5 odpowiadające poszczególnym wyjściom.

Stan wyjścia zostanie zmieniony tylko wtedy, gdy zarówno odebrany adres jak i rozkaz będzie identyczny z zapamiętanym w pamięci EEPROM. W pamięci EEPROM znajduje się również zmienna **IsProgrammed**, przechowująca informacje, czy w pamięci EEPROM zapamiętane zostały kody dla poszczególnych wyjść. Deklaracje zmiennych przedstawiono na **list. 1**.

Procedura obsługi przerwania zewnętrznego EI2

W ramach tej procedury następuje wyzerowanie zmiennych **RC5Address**, **RC5Command** oraz nadanie zmiennej **RC5Status** wartości początkowej. Następnie dokonywane jest zablokowanie przerwania zewnętrznego oraz włączenie timera i zezwolenie na przerwanie od jego przepełnienia. Kod procedury obsługi przerwania zewnętrznego przedstawiono na **list. 2**.

Ustawienie dwóch bitów błędu w zmiennej statusowej podyktowane jest algorytmem działania procedury obsługi przerwania od timera i wyjaśnione zostanie przy okazji jej omawiania.

Procedura obsługi przerwania od timera AT2

Jest to najważniejsza procedura, mająca za zadanie zdekodowanie transmitowanej ramki RC5. Ze

List. 4. Procedura zapisu danych do pamięci EEPROM

```
RegisterRemoteCodes
  CALL WaitForRC5    ; oczekiwanie na odebranie kodu
  LD Y, OutNumber    ; załadowanie do Y numeru wyjścia
  LD A, RC5Address      ; załadowanie do A odebranego adresu
  BSET EECSR, #EECSR_E2LAT  ; ustawienie bitu E2LAT -> zapis
  LD (OutAddress,Y), A     ; zapis wartości do pamięci EEPROM
  BSET EECSR, #EECSR_E2PGM  ; start operacji zapisu
  BTJT EECSR, #EECSR_E2PGM, *  ; oczekiwanie na zakończenie zapisu
  LD A, RC5Command   ; załadowanie do A odebranego rozkazu
  BSET EECSR, #EECSR_E2LAT  ; ustawienie bitu E2LAT -> Zapis
  LD (OutCommand,Y),A      ; zapis wartości do pamięci EEPROM
  BSET EECSR, #EECSR_E2PGM  ; start operacji zapisu
  BTJT EECSR, #EECSR_E2PGM, *  ; oczekiwanie na zakończenie zapisu
  BRES EECSR, #EECSR_E2LAT
  RET
```

List. 5. Procedura rejestrująca kody sterujące dla wszystkich wyjść

```
RegisterOutputs
  LD Y, OutNumber    ; załadowanie do Y numeru wyjścia
  LD A, (OutMask,Y)     ; załadowanie do A maski wyjścia
  CPL  A       ; zanegowanie A
  LD PADR, A      ; zapis do portu
  CALL RegisterRemoteCodes  ; zapamiętanie kodu dla aktualnego numeru wyj-
ścia
INC  OutNumber    ; inkrementacja numeru wyjścia
  LD A, #8      ;
  CP A, OutNumber    ; sprawdzenie, czy już wszystkie wyjścia
  JRNE RegisterOutputs    ; jeśli nie to skok na początek
  LD A, #$FF      ;
  LD PADR, A      ; dezaktywacja wszystkich wyjść
  BSET EECSR, #EECSR_E2LAT  ; E2LAT = 1 -> zapis
  LD A, #$55      ;
  LD IsProgrammed, A    ; zapamiętanie bajtu kontrolnego
  BSET EECSR, #EECSR_E2PGM  ; start operacji zapisu
  BTJT EECSR, #EECSR_E2PGM, *  ; oczekiwanie na zakończenie zapisu
  RET
```

względu na ograniczony zakres odmierzanego przez timer czasu przerwanie występuje czterokrotnie podczas trwania bitu. W związku z tym każde wystąpienie timera powoduje inkrementowanie wartości zmiennej **tickCount**. Na początku procedura dokonuje zerowania flagi przerwania poprzez odczyt rejestru ATCSR. Odczytana wartość w tym przypadku nie ma znaczenia i nie jest brana pod uwagę. Następnie inkrementowany jest licznik wystąpień przerwania a jego wartość porównywana jest z liczbą 57, która wskazuje na koniec ramki danych. Jeśli licznik jest różny od 57, to następuje sprawdzenie stanu linii wejściowej. W przypadku, gdy na linii wejściowej panuje stan wysoki, następuje opuszczenie procedury obsługi przerwania. W przypadku stwierdzenia stanu niskiego podejmowana jest w zależności od wartości licznika przerwań odpowiednia akcja, polegająca na ustawieniu odpowiednich bitów. Wyjątkiem są dwie flagi błędów, które w tym przypadku są zerowane, należy je więc na początku ramki ustawić. Czynność ta wykonywana jest w ramach procedury obsługi przerwania zewnętrznego, które sygnalizuje początek ramki. Rozwiązanie to zostało podyktowane chęcią maksymalnego uproszczenia procedury obsługi przerwania, której działanie sprowadza się do sprawdzenia linii wejściowej na okoliczność występowania na niej stanu niskiego. Ponieważ niski stan na wejściu w chwili 1 i 5 (patrz rys. 4) jest stanem poprawnym, następuje więc zerowanie ustawionych na początku ramki bitów błędów. Stwierdzenie stanu niskiego w chwili 3 oznacza, że transmitowana ramka nie jest nadawana w standardzie RC5, więc następuje ustawienie flagi błędu. W przypadku bitów A4:0 oraz C5:0 sprawa jest już oczywista: stwierdzenie na wejściu w odpowiadających im chwilach stanu niskiego powoduje ich ustawienie. Jako że na początku każdej ramki zmienne **RC5Address** i **RC5Command** są zerowane nie ma konieczności podejmowania działania w reakcji na niski stan na linii wejściowej.

Zapamiętanie odebranego kodu w pamięci EEPROM

Kody sterujące poszczególnymi wyjściami zapamiętywane są w nieulotnej pamięci EEPROM. Dla każdego wyjścia zapamiętywany jest adres oraz polecenie RC5, co pozwoli na wykorzystanie dowolnego pilota od dowolnego urządzenia sterowanego kodem RC5. Po włączeniu zasilania modułu LITEcomp jest sprawdzana wartość zmiennej **IsProgrammed**, przechowywanej również w pamięci EEPROM, i w przypadku, gdy jest ona różna od liczby 55h następuje wywołanie procedury uczenia kodów. Kod procedury zapamiętującej odebrany kod przedstawiono na **list. 4**.

Procedura zapisu kodu do pamięci EEPROM jest wywoływania kolejno dla każdego wyjścia. Kod procedury rejestrującej kody sterujące kolejno dla wszystkich wyjść przedstawiony jest na **list. 5**.

Porównanie odebranego kodu z zapamiętanym w pamięci EEPROM

Po zarejestrowaniu kodów dla wszystkich wyjść program przechodzi do oczekiwania na nadejście kodu. Po jego odebraniu następuje porównanie z zapamiętanymi w pamięci EEPROM kodami odpowiedzialnymi za sterowanie poszczególnymi wyjściami. W przypadku stwierdzenia zgodności odebranego kodu z odczytaniem z pamięci EEPROM następuje zanegowanie stanu odpowiedniego wyjścia a następnie opuszczenie procedury porównującej kody. Wynikają z tego dwa ograniczenia: nie można zaprogramować dla dwóch wyjść tego samego kodu, jak również nie można zaprogramować osobnych kodów do włączenia i do wyłączenia wyjść.

List. 6. Procedura porównania kodów: odebranego i zapamiętanego

```
CheckRemoteCodes
  LD Y, OutNumber
  LD A, (OutAddress,Y)  ; odczyt adresu z pamięci
  CP A, RC5Address; porównanie z adresem odebranym
  JRNE CRCNext    ; jeśli różne przygotowanie do porównania następnego
  LD A, (OutCommand,Y)  ; odczyt rozkazu z pamięci
  CP A, RC5Command; porównanie z rozkazem odebranym
  JRNE CRCNext    ; jeśli różne przygotowanie do porównania następnego
  LD A, PADR    ;
  XOR  A, (OutMask,Y)  ;  zanegowanie stanu wyjścia o odpowiednim numerze
  LD PADR, A    ;
CRCNext
  IN OutNumber  ; zwiększenie numeru
  LD A, #8    ;
  CP A, OutNumber  ; sprawdzenie czy równe 8
  JRNE CheckRemoteCodes   ; jeśli różne to skok na początek
  CLR  OutNumber  ; w przeciwnym razie zerowanie numeru wyjścia
  RET
```

List. 7. Pętla programu głównego

```
._reset
     CALL Init       ; inicjalizacja
     RIM           ; odblokowanie przerwań
     LD A, IsProgrammed    ; odczyt bajtu kontrolnego
     CP A, #$55       ; sprawdzenie czy nauczono kodów
     JREQ mainLoop         ; jeśli tak to skok do pętli głównej
     CALL RegisterOutputs    ; w przeciwnym razie wywołanie procedury uczenia
mainLoop
     CALL WaitForRC5    ; oczekiwanie na odebranie kodu
     CALL CheckRemoteCodes      ; sprawdzenie odebranego kodu z zapamiętanym
     JRA  mainLoop         ; pętla nieskończona
```

Program główny

Zasadnicza część programu jest bardzo prosta. Bezpośrednio po uruchomieniu programu dokonywana jest inicjalizacja portów, zmiennych oraz wykorzystywanych układów peryferyjnych. Następnie sprawdzana jest wartość zmiennej **IsProgrammed**. W przypadku, gdy ta wartość jest różna od liczby 0x55, co świadczy o tym, że w pamięci EEPROM nie zostały zapamiętane kody sterujące poszczególnymi wyjściami, następuje wywołanie procedury rejestracji kodów. W przeciwnym przypadku następuje przejście do pętli głównej, w której program oczekuje na odebranie kodu RC5 a następnie dokonuje sprawdzenia czy odebrany kod jest identyczny z jednym z ośmiu kodów zapamiętanych w pamięci EEPROM. W sytuacji, gdy odebrany kod znajduje się w pamięci EEPROM następuje uaktywnienie wyjścia, do którego kod został przypisany.

Przedstawiony w artykule program, który w rzeczywistości jest zbiorem podstawowych procedur pozwalających na odbiór sygnału zdalnego sterowania w systemie RC5. Zajmuje on niespełna 10% dostępnej pamięci programu. Do dyspozycji programisty pozostaje ogromna część pamięci programu, jak również pamięci danych, co pozwala na zaimplementowanie dowolnych dodatkowych funkcji. Dzięki temu istnieje możliwość stworzenia praktycznie dowolnego urządzenia sterowanego pilotem podczerwieni. Informacje zawarte w artykule mogą stanowić podstawę do własnych opracowań urządzeń sterowanych sygnałem w standardzie RC5.

Radosław Kwiecień, EP
radosław.kwiecien@ep.com.pl