Emulator odbiornika DCF

AVT-5050

Emulator może być przydatny przygotowującym oprogramowanie obsługujące odbiorniki sygnału DCF. Gdy program nie działa poprawnie nie wiemy jaka jest przyczyna: czy odbiornik odbiera zakłócony sygnał, czy występuje błąd w programie? Dzięki emulatorowi mamy pewność, że nadawany przez niego sygnał jest prawidłowy.

Na sygnał synchronizacji z odbiornika DCF trzeba czekać nawet minutę, dzięki emulatorowi czas ten wynosi tylko kilka sekund. Emulator odbiornika DCF opisano już na łamach EP (AVT-423). Jego budowa była oparta na EPROM-ie, co powodowało konieczność zastosowania wielu dodatkowych układów. Opisany w tym artykule emulator jest o wiele prostszy. Zawiera tylko jeden układ scalony (nie licząc stabilizatora). Ponadto o wiele łatwiej wpisać dane przesyłane DCF.

Budowa i działanie

Jak widać na schemacie elektrycznym (**rys. 1**), urządzenie jest banalnie proste. Napięcie zasilania z gniazda DB9 (w takie samo gniazdo są wyposażone odbiorniki DCF) jest obniżane do 5V za pomocą scalonego stabilizatora US1. Kondensatory C1...C4 filtrują napięcie zasilające. Napięcie 5V służy do zasilania procesora. Elementy C5 i R1 formują sygnał zerujący po włączeniu zasilania. Mikrokontroler można wyzerować także ręcznie, naciskając przełącznik RESET. Elementy C6, C7 i Q1 ustalają częstotliwość sygnału generatora zegarowego taktującego mikrokontroler. Tranzystory T1 i T2, wraz z elementami towarzyszącymi, tworzą obwód konwertujący sygnał o poziomach TTL na sygnał o poziomach zgodnych ze standardem RS232C. Zastosowana konfiguracja tego konwertera zabezpiecza emulator przed uszkodzeniami powodowanymi zwarciem wyjścia do masy czy szyny zasilania. Emulator wytwarza zarówno napięcia bipolarne jak i unipolarne, co jest zależne od budowy stopnia wejściowego zegara DCF. Z napięciami bipolarnymi będziemy mieli do czynienia, gdy emulator podłączymy do portu RS232C (np. komputer), a napięcia unipolarne są stosowane przy innych rozwiązaniach w innych konstrukcjach (np. AVT-322, AVT-5022). Dioda D3 świeci się w momentach, gdy na wyjściu RS pojawia się impuls. Na wyjściu OUT_TTL pojawiają się impulsy w standardzie TTL. Wyjście to może być użyteczne, gdy będziemy programować zegar, a nie zbudowaliśmy jeszcze części sprzętowej. Wówczas wyjście OUT_TTL łączymy w wybranym wyprowadzeniem procesora, w którym zapisano dekodującym sygnał DCF. Emulator jest zasilany z portu RS232C. Jeśli jednak wykorzystamy wyjście OUT_TTL emulator trzeba zasilić inaczej. Możemy podać napięcie +8...15V na wejście 12V lub napięcie +5V na wejście 5V.

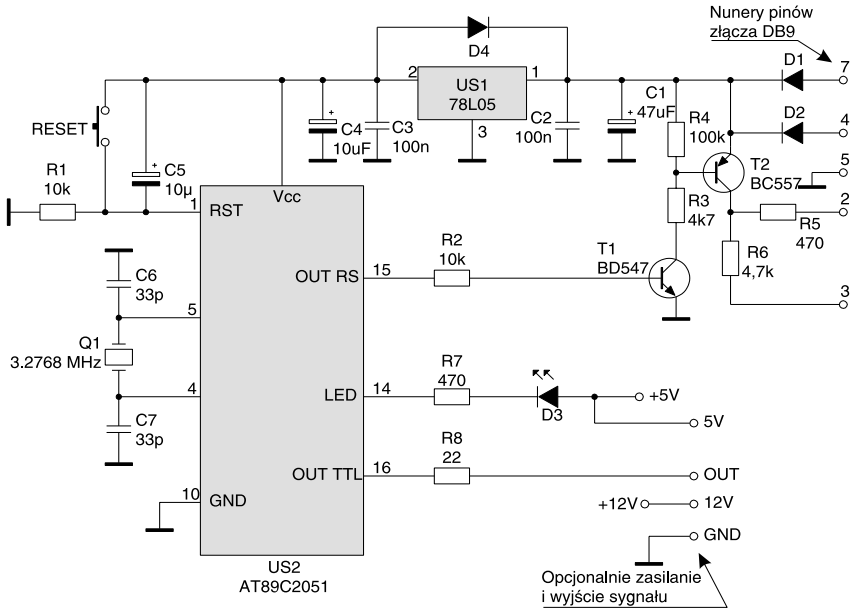
W pamięci procesora zapisano program generujący sygnał DCF. Program obsługujący zegar jest dość krótki, dlatego zdecydowałem się na umieszczenie jego treści w artykule (**list. 1**).

Ważnym fragmentem programu są deklaracje pojawiające się za etykietą „dane:“. Każda deklaracja

List. 2.

```
;Listing 1
;-------------------------------------
    include 'INCL:8051/8051.def'
    include 'INCL:8051/piny_89c2051.def'

speed_h     equ     256-107    ; 100ms = 100000[ s]*(3.2768[MHz]/12) =
speed_l     equ     256-170    ; 27306 cykli = 106*256+170

LED         equ     Pin14
OUT_TTL     equ     Pin16
OUT_RS      equ     Pin15

;-------------------------------------
start:
    mov     SP,#$30             ;Ustawienie stosu
    mov     TMOD,#%0001         ;timer 0 liczy czasy
    mov     TH0,#speed_h        ;załadowanie timerów 100ms
    mov     TL0,#speed_l
    setb    TR0                 ;start timera

    acall   impulszero          ;generuje trzy impulsy zero
    acall   impulszero
    acall   impulszero

datainit:
    mov     dptr,#dane          ;adres początku danych
    mov     R0,#60              ;liczba danych
repeat:
    mov     a,#0                ;pobranie danej
    movc    a,@a+dptr
    cjne    a,#0,rep1           ;czy "0" ?
    acall   jeden               ;generowanie zera
    mov     a,#1
    acall   wait100ms
    acall   zero
    mov     a,#9
    acall   wait100ms
    sjmp    rep4

rep1:
    cjne    a,#1,rep2           ;czy "1" ?
    acall   jeden               ;generowanie jedynki
    mov     a,#2
    acall   wait100ms
    acall   zero
    mov     a,#8
    acall   wait100ms
    sjmp    rep4

rep2:
    cjne    a,#'s',error        ;czy synchro ?
    acall   zero                ;generowanie synchronizacji
    mov     a,#18
    acall   wait100ms
    sjmp    rep4

error:
    mov     a,#1                ;błąd
    acall   wait100ms
    acall   zero
    mov     a,#1
    acall   wait100ms
    acall   jeden
    sjmp    error

rep4:
    inc     dptr                ;zwiększ wskaźnik danych
    djnz    R0,repeat           ;czy koniec
    sjmp    datainit

;-------------------------------------
impulszero:
    acall   jeden               ;generowanie zera
    mov     a,#1
    acall   wait100ms
    acall   zero
    mov     a,#9
    acall   wait100ms
    ret

zero:
    setb    LED                 ;LED - zgaszona
    clr     OUT_TTL             ;OUT = 0V
    clr     OUT_RS              ;RS  = +12V
    ret

jeden:
    clr     LED                 ;LED - świeci
    setb    OUT_TTL             ;OUT = +5V
    setb    OUT_RS              ;RS  = -12V

    ret

; Odczekanie czasu 100ms*ACC
wait100ms:
    jnb     TF0,wait100ms
    mov     TH0,#speed_h        ;załadowanie timerów 100ms
    mov     TL0,#speed_l
    clr     TF0                 ;zerowanie flagi
    djnz    ACC,wait100ms
    ret

;-------------------------------------
; Dane zawierają:
;   23:21 27.05.01  Niedziela

dane:
    byte    's'     ; 60    Synchronizacja    (brak impulsu)
    byte    0       ;  1    Start                (0)
    byte    0       ;  2    14 zer
    byte    0       ;  3
    byte    0       ;  4
    byte    0       ;  5
    byte    0       ;  6
    byte    0       ;  7
    byte    0       ;  8
    byte    0       ;  9
    byte    0       ; 10
    byte    0       ; 11
    byte    0       ; 12
    byte    0       ; 13
    byte    0       ; 14
    byte    0       ; 15
    byte    0       ; 16    Typ anteny
    byte    0       ; 17    Zapowiedź zmiany czasu
    byte    1       ; 18    LSB Czas letni/zimowy   %01
    byte    0       ; 19    MSB Czas letni/zimowy
    byte    0       ; 20    Korekta czasu           %0
    byte    1       ; 21    Początek informacji     %1

    byte    1       ; 22    LSB jedności minut      %0001, #1
    byte    0       ; 23        jednostki minut
    byte    0       ; 24        jednostki minut
    byte    0       ; 25    MSB jednostki minut
    byte    0       ; 26    LSB dziesiątki minut    %010,  #2
    byte    1       ; 27        dziesiątki minut
    byte    0       ; 28    MSB dziesiątki minut
    byte    0       ; 29    Bit parzystości minut

    byte    1       ; 30    LSB jednostki godzin     %0011, #3
    byte    1       ; 31        jednostki godzin
    byte    0       ; 32        jednostki godzin
    byte    0       ; 33    MSB jednostki godzin
    byte    0       ; 34    LSB dziesiątki godzin   %10, #2
    byte    1       ; 35    MSB dziesiątki godzin
    byte    1       ; 36    Bit parzystości godzin

    byte    1       ; 37    LSB jednostki dnia miesiąca      %0111, #7
    byte    1       ; 38        jednostki dnia miesiąca
    byte    1       ; 39        jednostki dnia miesiąca
    byte    0       ; 40    MSB jednostki dnia miesiąca
    byte    0       ; 41    LSB dziesiątki dnia miesiąca    %10, #2
    byte    1       ; 42    MSB dziesiątki dnia miesiąca
    byte    1       ; 43    LSB dnia tygodnia               %111, #7
    byte    1       ; 44        dnia tygodnia
    byte    1       ; 45    MSB dnia tygodnia
    byte    1       ; 46    LSB jednostki miesiąca   %0101, #5
    byte    0       ; 47        jednostki miesiąca
    byte    1       ; 48        jednostki miesiąca
    byte    0       ; 49    MSB jednostki miesiąca
    byte    0       ; 50        dziesiątki miesiąca     %0,    #0
    byte    1       ; 51    LSB jednostki roku       %0001, #1
    byte    0       ; 52        jednostki roku
    byte    0       ; 53        jednostki roku
    byte    0       ; 54    LSB jednostki roku
    byte    0       ; 55    LSB dziesiątki roku     %0000, #0
    byte    0       ; 56        dziesiątki roku
    byte    0       ; 57        dziesiątki roku
    byte    0       ; 58    LSB dziesiątki roku
    byte    0       ; 59    Bit parzystości bitów 37...58
```

„byte" to informacja dla programu czy wygenerować „zero", „jeden" czy sygnał synchronizacji. Przyjrzyjmy się bliżej formatowi przesyłania informacji w kodzie DCF. Jedynka jest reprezentowana przez impuls trwający 200ms, z przerwą 800ms. Sygnał reprezentujący zero jest impulsem trwającym 100ms z przerwą 900ms. Jak z tego wynika nadawanie jednego bitu trwa dokładnie 1 sekundę. Bit synchronizacji jest reprezentowany przez brak impulsu, czyli przerwę trwającą 1800ms (1,8 sekundy). Po impulsie synchronizacji (w pierwszej sekundzie) występuje bit startu 0, po nim 14 zer. 16. sekunda (16. bit) określa typ anteny (0-normalna, 1-zapasowa). Bit 17. zawiera informację o zapowiedzi zmiany czasu. Na godzinę przed zmianą czasu bit ten równy jest jeden. Bity 18. i 19. zawierają informację o czasie {letni czy zimowy). Bity o wartościach %10- w kolejności starszy bit 19. młodszy 18. - oznaczają czas zimowy, a %01 oznacza czas letni. Bit 20. równy „1" oznacza zapowiedź dodatkowej sekundy. Bit 21. zawsze jest ustawiony na jeden (bit startu informacji). Bity 22...25. (w kolejności 22. bit-LSB, 25. bit-MSB) zawierają jedności minut w formacje BCD. Bity 26...28. dziesiątki minut, bit 28. jest bitem parzystości bitów 22...28. Bit parzystości oblicza się bardzo łatwo. Jeżeli liczba „jedynek" w sprawdzanym rekordzie będzie parzysta, bit sumy kontrolnej przyjmuje wartość 0. Jeżeli liczba jedynek będzie nieparzysta, bit sumy kontrolnej przyjmuje wartość 1. Bity 30...33. zawierają jedności godzin, bity 34...35. dziesiątki godzin, bit 36. parzystość



Rys. 1. Schemat elektryczny emulatora.

dla bitów 30...35. Bity 37...40. - jedności dni miesiąca, bity 41...42. dziesiątki dni miesiąca. Bity 43...45. nr dnia tygodnia (1-poniedziałek, 2-wtorek, 3-środa, itd.). Bity 46...49. - jedności miesiąca, bit 50. - dziesiątki miesiąca. Bity 51...54. - jednostki lat, bity 55...58. dziesiątki lat. W informacji DCF nadawane są ostatnie dwie cyfry roku. Bit 59. jest bitem parzystości bitów 37...58. Na tym kończy się nadawanie informacji DCF. Po 59. bicie następuje impuls synchronizacji i ponowne nadawanie informacji. Znając ten format możemy wpisać w programie dowolną datę i godzinę. W programie przykładowym wpisano dane zawierające datę 27 maja 2001 roku, niedziela, godzina 23:21.

Jeśli zechcemy wpisać inną datę czy godzinę, należy pamiętać o odpowiednim ustawieniu bitów parzystości. Przypomnę tylko, że deklaracja „byte 0" spowoduje wygenerowanie 0, „byte 1" jedynki, natomiast „byte „s" spowoduje wygenerowanie impulsu synchronizacji.

Montaż i uruchomienie

Schemat montażowy płytki drukowanej pokazano na **rys. 2**. Montaż zaczynamy od elementów najmniejszych, kończymy na największych. Po włączeniu zasilania dioda D3 powinna sygnalizować migotaniem krótkie impulsy.

Emulator podłączamy w miejsce przeznaczone na odbiornik sygnału DCF. Po uruchomieniu zegara DCF naciskamy przycisk RESET na emulatorze. Emulator generuje trzy impulsy „zero", następnie impuls synchronizacji i informację o czasie. Po odebraniu sygnału synchronizacji zegar DCF powinien zacząć dekodować informację. Po odebraniu całej informacji (pojawieniu się kolejnego impulsu synchronizacji), zegar powinien wyświetlić datę i czas. Jeśli czas nie pojawił się, to oznacza, że popełniliśmy błąd w programie. Należy go odszukać, a po usunięciu program ponownie skompilować i zapisać w pamięci procesora lub w pamięci uruchamianego systemu (EPROM, emulator EPROM, czy emulator CPU). Po wystartowaniu programu należy nacisnąć RESET na emulatorze DCF. Dzięki temu nie będziemy długo czekać na pojawienie się impulsu synchronizacji. Dzięki emulatorowi można znacznie skrócić czas pisania programu. Jeśli program poprawnie działa z emulatorem, należy sprawdzić go z odbiornikiem DCF. Zależnie od terytorium na którym mieszkamy, odbieranie poprawnej informacji DCF może zająć trochę czasu. Ten czas zależy od zakłóceń, poziomu sygnału, itp.

Jeszcze uwaga dla piszących program obsługi DCF. Emulator generuje czysty sygnał, w którym impuls „jedynki" trwa dokładnie 200ms, „zera" 100ms. Rzeczywisty sygnał jest zazwyczaj zniekształcony, czas trwania impulsu „zera" może wahać się w granicach 80...120ms, dlatego najlepiej napisać program tak aby:

- za jedynkę uznawać impuls trwający od 50 do 150ms,
- za zero uznawać impuls trwający od 150 do 250ms,
- za synchronizację uznawać brak impulsu przez ponad 1500ms.

Tą procedurę można uprościć:

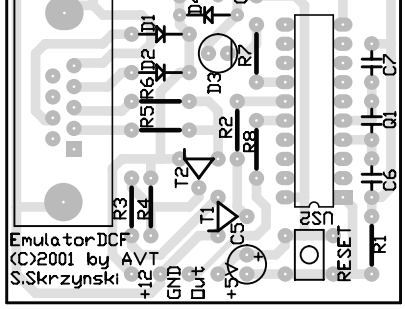
- za jedynkę uznawać impuls trwający mniej niż 150ms,
- za zero uznawać impuls trwający ponad 150ms,
- za sygnał synchronizacji uznawać brak impulsu przez ponad 1500ms.

Sławomir Skrzyński, AVT
slawomir.skrzynski@ep.com.pl

WYKAZ ELEMENTÓW

Rezystory
R1, R2: 10kΩ
R3, R6: 4,7kΩ
R4: 100kΩ
R5, R7: 470Ω
R8: 22Ω
Kondensatory
C1: 47µF/25V
C2, C3: 100nF
C4, C5: 10µF/16V
C6, C7: 33pF
Półprzewodniki
US1: 78L05
US2: AT89C2051
T1: BC547
T2: BC557
D1, D2, D4: 1N4148
D3: LED
Różne
Złącze DB9F do druku
Q1: 3,2768MHz
RESET: mikroprzełącznik

Wzory płytek drukowanych w formacie PDF są dostępne w Internecie pod adresem: ***http://www.ep.com.pl/?pdf/luty02.htm*** *oraz na płycie CD-EP02/2002B w katalogu* ***PCB****.*



Rys. 2. Rozmieszczenie elementów na płytce drukowanej.