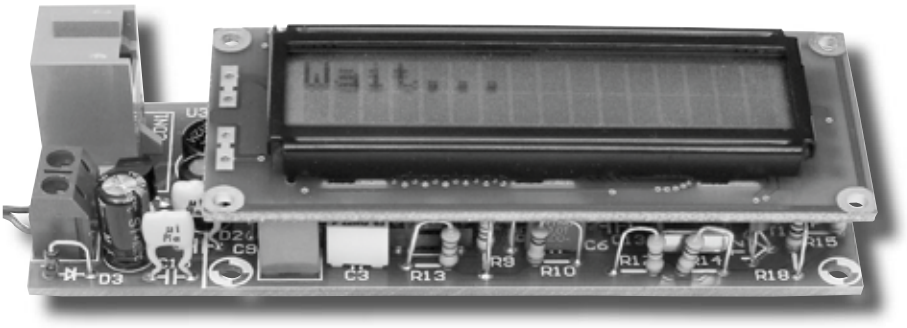
Programowy dekoder CLIP (FSK), część 1

AVT-590



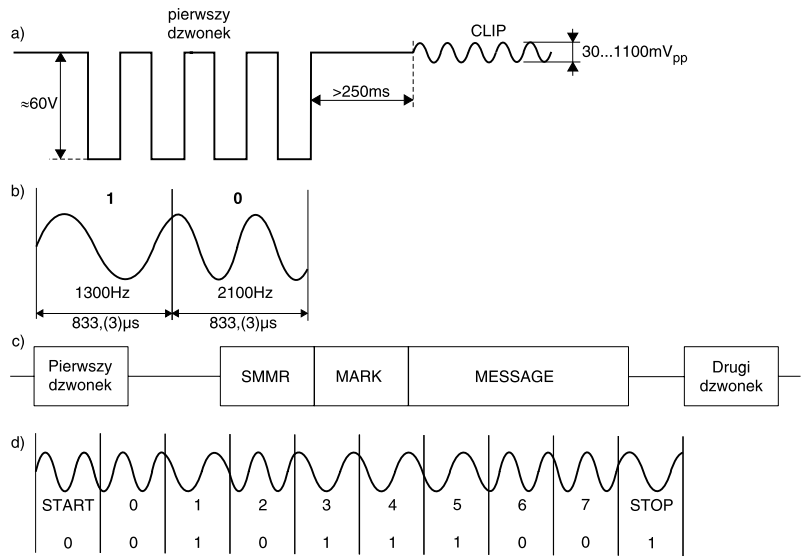
W artykule prezentujemy układ wyświetlający numer abonenta dzwoniącego (CLIP - Calling Line Identification Presentation). W przeciwieństwie do poprzednio opisywanych urządzeń, nie wymaga żadnego specjalizowanego układu scalonego.

Rekomendacje:

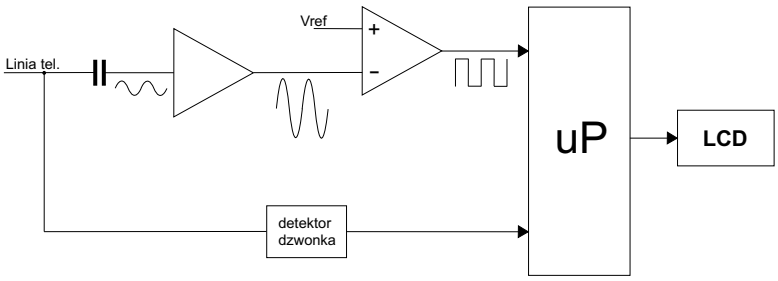
możliwość odczytu numeru abonenta dzwoniącego jest bardzo cenna. Umożliwia użytkownikowi telefonu chociażby uniknięcia niechcianych rozmów. Dla elektroników budujących wszelkiego rodzaju układy „telefoniczne" funkcja CLIP jest bezcenna, bo pozwala budować urządzenia o ogromnych możliwościach, często przewyższających możliwości urządzeń fabrycznych. Mogą to być wszelkiego rodzaju rejestratory rozmów telefonicznych, taryfikatory itp.

Usługa prezentacji numeru dzwoniącego jest standardem w sieciach komórkowych i systemie ISDN. Abonenci analogowi także mogą z niej skorzystać pod warunkiem, że przyłączeni są do centrali cyfrowej. Korzystanie z usługi wymaga zwykle jej zamówienia w Biurze Obsługi Klienta operatora telekomunikacyjnego. W przypadku linii analogowych do przesyłania informacji CLIP stosuje się dwie podstawowe metody. Pierwsza polega na wysłaniu numeru dzwoniącego za pomocą typowych sygnałów DTMF. Druga – znacznie częściej stosowana (jak mnie poinformowano, dominujący na rynku polskim operator korzysta tylko z tej metody) – wykorzystuje do tego celu system modulacji V.23 (FSK). W Elektronice Praktycznej znalazł się już opis dekodera CLIP pracującego w systemie V.23. Jednak użyty w nim specjalizowany układ scalony nie jest tani, a przede wszystkim – nie jest łatwo dostępny. To samo dotyczy innych dekoderów CLIP i demodulatorów FSK.

Poniżej prezentuję projekt programowego dekodera CLIP FSK, który po dzwonku pokazuje na wyświetlaczu LCD numer abonenta dzwoniącego. Artykuł zawiera szczegółowy opis przebiegów FSK i przykład ich programowego dekodowania za pomocą prostego mikrokontrolera AVR (90S2313). Eliminuje to konieczność stosowania sprzętowego dekodera, co pozwala obniżyć koszt urządzenia o ok. 10 USD (nie wspominając o uniknięciu trudności zdobycia takich kości). Układ stanowi także przejaw aktualnej tendencji: w dobie rosnących pojemności pamięci i możliwości mikrokontrolerów oraz spadku ich cen



Rys. 1. Zasada przesyłania sygnału CLIP. a) napięcie w linii podczas przesyłania sygnału CLIP, b) sposób kodowania bitów, c) pakiet transmisji CLIP, d) ramka przesyłania bajtu 00111010b



Rys. 2. Schemat blokowy dekodera

dąży się do ograniczenia stosowania dekoderów sprzętowych. Jeszcze kilka lat temu do RC5 itp. stosowano np. SAA3049, dziś takie rozwiązanie jest co najmniej archaiczne.

Opis protokołu CLIP V.23

Dane o numerze abonenta dzwoniącego wraz z aktualną datą i godziną przesyłane są pomiędzy pierwszym, a drugim sygnałem dzwonienia. Mają one postać sygnału sinusoidalnego o amplitudzie 15 mV...560 mV i składowej stałej równej napięciu panującemu w linii telefonicznej w stanie spoczynku (odłożona słuchawka), które wynosi 40...70 V (**rys. 1a**). Dane cyfrowe kodowane są z wykorzystaniem modulacji FSK (*Frequency Shift Keying*). Polega ona na szeregowym (asynchronicznym) przesyłaniu kolejnych bitów informacji z prędkością 1200 bodów (±1%), przy czym wartości bitów kodowane są za pomocą częstotliwości sygnału. Jedynce logicznej (*mark*) odpowiada częstotliwość 1300 Hz (±1,5%), zaś logicznemu zeru (*space*) – 2100 Hz (±1,5%). Jest to pokazane na **rys. 1b**.

Cały pakiet CLIP składa się z trzech następujących po sobie bloków pokazanych na **rys. 1c**. Są to:

- **SMMR**. Sygnał służący do nastrojenia odbiornika. Składa się z ciągu 300 bitów o wartościach zmieniających się naprzemiennie: 0,1,0,...,
- **MARK**. Sygnał ten składa się z ciągu 180 bitów - wszystkie o wartości 1,
- **MESSAGE**. Jest to główna część pakietu niosąca w sobie informacje m.in. o numerze abonenta dzwoniącego. Składa się z sekwencji bajtów. Bajty przesyłane są, podobnie jak w standardzie RS232, w postaci ramek zawierających 1 bit startu o wartości 0, 8 bitów informacyjnych oraz 1...10 bitów stopu o wartości 1. Bity informacyjne przesyłane są w kolejności od najmniej do najbardziej znaczącego. Brak jest bitu kontroli parzystości.

Rys. 1d przedstawia przykładową ramkę przy przesyłaniu bajtu o wartości 00111010b (58 dec).

Znaczenie kolejnych grup bajtów bloku MESSAGE (wiadomość) jest następujące:

- **T1** – rodzaj danych. W przypadku przesyłania sygnału identyfikacji numeru dzwoniącego bajt ten ma wartość: 10000000,
- **L1** – liczba wszystkich bajtów wiadomości z wyjątkiem T1, L1 i CHECKSUM. Wartość zależy od długości numeru abonenta dzwoniącego,
- **T2** – określenie rodzaju danych: czas i data. Bajt ten ma stałą wartość: 00000001,
- **L2** – liczba bajtów kodujących czas i datę. Bajt ma stałą wartość: 00001000,
- **V2** – osiem bajtów, które zawierają czas i datę zapisane w kodzie ASCII,
- **T3** – określenie rodzaju danych: numer abonenta. Bajt ten ma stałą wartość: 00000010,
- **L3** – liczba bajtów numeru,
- **V3** – numer abonenta w postaci kolejnych cyfr zapisanych w kodzie ASCII,
- **CHECKSUM** – suma kontrolna. Bajt ten zawiera dopełnienie do dwóch sumy (modulo 2) wszystkich bajtów wiadomości począwszy od **T1** (z wyłączeniem samej sumy kontrolnej). W przypadku poprawnej wiadomości dodanie do siebie modulo 2 wszystkich jej bajtów łącznie z sumą kontrolną da w wyniku zero. Przy dodawaniu przeniesienie z najbardziej znaczącego bitu jest ignorowane (suma ma być 8-bitowa).

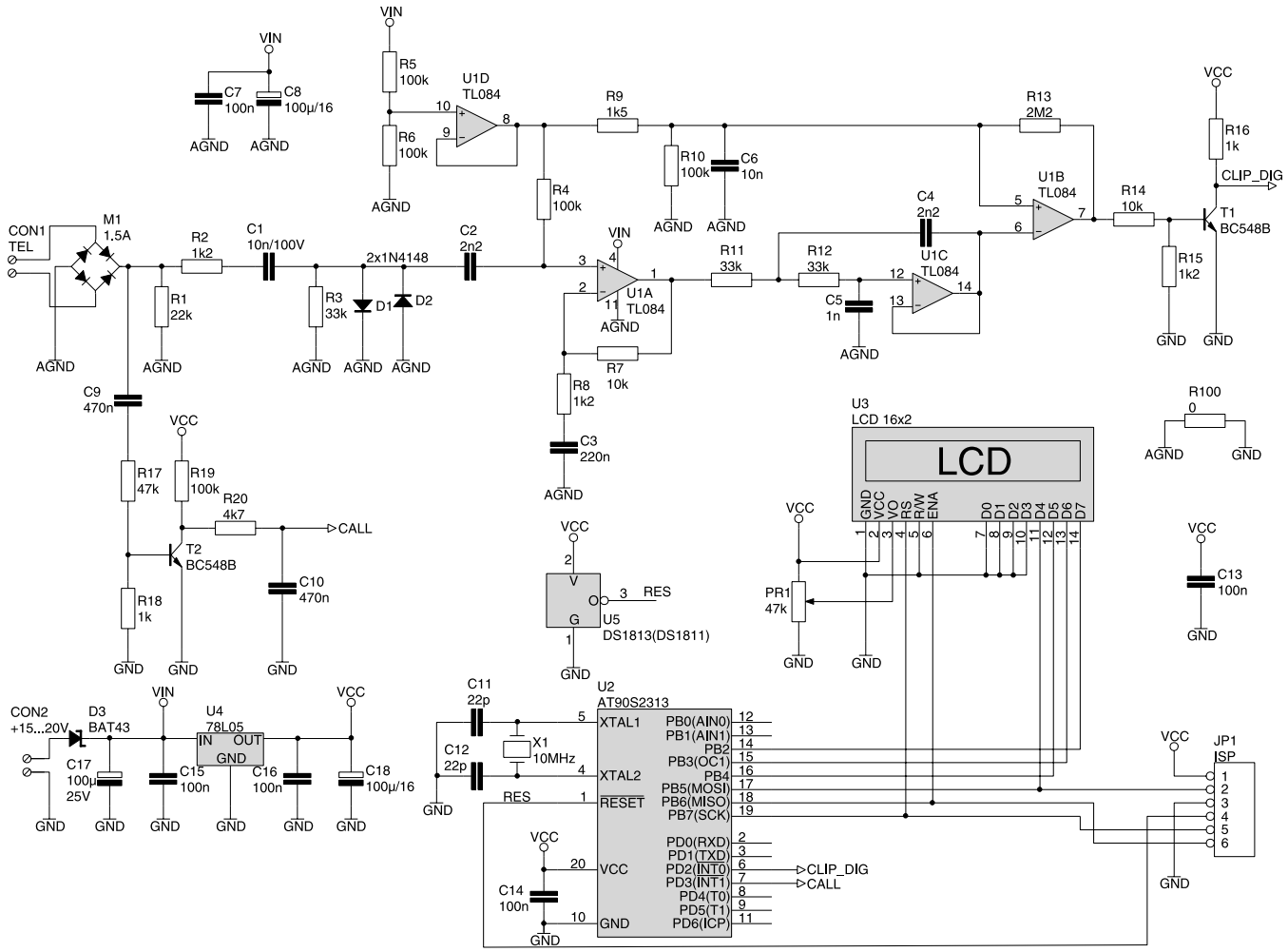
Poniżej przedstawiona jest przykładowa postać bloku MESSAGE jaki zostanie przesłany gdy abonent o numerze 0600123456 zadzwoni do nas o godzinie 13:42 dnia 12 maja:

T1: 10000000 -> CLIP
L1: 00010110 -> 22 bajty od T2 do końca V3
T2: 00000001 -> czas i data
L2: 00001000 -> 8 bajtów na czas i datę
V2:
00110000 -> '0'
00110101 -> '5'
00110001 -> '1'
00110010 -> '2'
00110001 -> '1'
00110011 -> '3'
00110100 -> '4'
00110010 -> '2'
T3: 00000010 -> numer abonenta
L3: 00001010 -> 10 bajtów na numer (10 cyfr)
V3:
00110000 -> '0'
00110110 -> '6'
00110000 -> '0'
00110000 -> '0'
00110001 -> '1'
00110010 -> '2'
00110011 -> '3'
00110100 -> '4'
00110101 -> '5'
00110110 -> '6'
CHECKSUM: 11001000

Opis układu

Schemat blokowy dekodera znajduje się na **rys. 2**. Ogólna zasada działania układu jest następująca: sygnał dzwonienia pojawiający się w linii jest wykrywany przez blok detektora dzwonka, który utrzymuje na swoim wyjściu stan aktywny (niski) przez cały czas trwania sygnału dzwonienia. Nie reaguje on przy tym na inne przebiegi występujące w linii. Napięcie z wyjścia detektora wyzwala proces analizy i dekodowania. Po ponad 250 ms pojawia się sygnał CLIP (rys. 1a). Po „wyłuskaniu" sygnału CLIP z linii telefonicznej jest on wzmacniany (wraz z filtracją pasmową), a następnie za pomocą komparatora porównywany z odpowiednio wytworzonym napięciem stałym V_{ref}. Powstały na wyjściu komparatora przebieg prostokątny (cyfrowy) ma tę samą częstotliwość co sygnał wejściowy. Czasy trwania stanów wysokiego i niskiego są (niemal – o tym w dalszej części) takie same jak czasy trwania półokresów sygnału CLIP. Otrzymany przebieg cyfrowy jest następnie dekodowany przez mikrokontroler. Na koniec odczytany numer abonenta oraz aktualna data i godzina trafiają na wyświetlacz alfanumeryczny LCD.

Schemat ideowy układu jest pokazany na **rys. 3**. Układ składa się

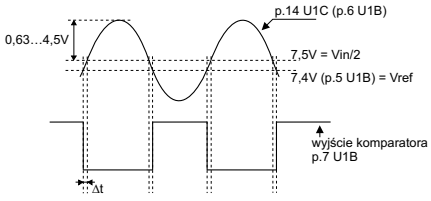


Rys. 3. Schemat elektryczny dekodera

z wyraźnie wyodrębnionych części cyfrowej, którą tworzy układ U2 wraz z elementami współpracującymi z nim, oraz części analogowej tworzonej przez poczwórny wzmacniacz operacyjny U1 i współpracujące z nim elementy dyskretne. Część cyfrowa zasilana jest z prostego zasilacza zbudowanego na U4. Masa układu jest rozdzielona na część analogową i cyfrową. Ścieżki masy na płytce łączą się przy zasilaniu poprzez R100 o wartości 0R (zwora). Zapobiega to przede wszystkim mieszaniu się prądów analogowych z cyfrowymi, a w konsekwencji wzmacniania szybkich impulsów cyfrowych przez układy analogowe. Opisywany dekoder nie należy wprawdzie do urządzeń ultra-precyzyjnych, ale rozdzielenie mas jest zawsze właściwym rozwiązaniem, gdy w układzie współpracują ze sobą bloki cyfrowe i analogowe.

W spoczynku na wyjściu mostka M1 panuje napięcie ok. 60 V. Gdy ktoś do nas dzwoni przychodzący sygnał dzwonienia w postaci przebiegu prostokątnego o amplitudzie ok. 60 V trafia na detektor dzwonka, który tworzą elementy T2, C9, C10, R1... R20. Kondensator C9 odcina składową stałą sygnału. Dzielnik R17, R18 zapewnia, że T2 otworzy się dopiero przy napięciu przekraczającym 22 V, co zapobiega włączaniu T2 spowodowanemu występowaniem zakłóceń. W każdej nieparzystej połówce okresu tranzystor T2 nasyca się i przez rezystor R20 rozładowuje C10. Napięcie na C10 (punkt CALL) spada niemal do zera, a więc poniżej progu przełączania portów procesora U2. Układ R20 - C10 zapobiega przedostawaniu się przez detektor szybkich impulsów (szpilek), co dodatkowo poprawia jakość jego pracy. Stała czasowa R20, C10 wynosi ok. 2 ms i jest wystarczająco mała aby C10 zdążył się rozładować w czasie połowy okresu (20 ms). Podczas parzystych półokresów T2 jest zatkany i C10 ładuje się przez R19 i R20, jednak stała czasowa wynosi w tym przypadku znacznie więcej niż 20 ms. Dzięki temu napięcie w punkcie CALL utrzymuje się w stanie niskim przez cały czas trwania dzwonka i jeszcze przez ok. 100 ms po jego zakończeniu. Sygnał dzwonka przedostaje się jednocześnie poprzez R2 i C1 do części odpowiedzialnej za obróbkę sygnału CLIP, ale jest on w tym przypadku całkowicie ignorowany przez mikrokontroler.

Co najmniej 250 ms po pierwszym dzwonku, w linię wysyłany jest sygnał CLIP. Przedostaje się on przez mostek M1 (mimo przeznaczenia do pracy z przebiegami 50 Hz mostki doskonale przenoszą sygnały o częstotliwościach kilku kHz). Trafia następnie przez R2 i C1 na ogranicznik napięciowy złożony z diod D1, D2. Diody te mają na celu ograniczenie napięcia podawanego na wejście wzmacniacza U1A (TL084) podczas występowania sygnału dzwonienia o amplitudzie kilkudziesięciu woltów. W przypadku sygnału CLIP, którego amplituda nie przekracza 600 mV, diody te nie maja żadnego znaczenia. Kondensator C1 odcina składową stałą i jednocześnie wraz z rezystorem R3 tworzy prosty filtr górnoprzepustowy o częstotliwości granicznej



Rys. 4. Zasada wytwarzania przebiegu cyfrowego

ok. 700 Hz. Wstępnie odfiltrowuje on niskoczęstotliwościowe śmieci, jak na przykład przydźwięk sieciowy. Następnie sygnał przez kondensator C2 trafia na wzmacniacz nieodwracający U1A. Jego wejście spolaryzowane jest napięciem stałym równym połowie napięcia zasilającego V_{in}, które jest wytwarzane przez R5, R6 i wtórnik U1D. Na wejściu U1A pojawia się sygnał CLIP o składowej stałej równej $V_{in}/2$ i amplitudzie 15...560 mV. Wzmocnienie U1A wyznacza stosunek R7 do R8 i impedancji C3. Dla przebiegów o częstotliwości 1...2 kHz wynosi ono ok. 8 razy. Jednocześnie R8 i C3 określają dolną częstotliwość graniczną wzmacniacza równą ok. 700 Hz, co dodatkowo eliminuje zakłócenia niskoczęstotliwościowe.

Dla napięć stałych U1A jest wtórnikiem, więc na jego wyjściu pojawia się wzmocniony przebieg CLIP o takiej samej składowej stałej jaką ma przebieg wejściowy, czyli $V_{in}/2$. Przy maksymalnej przewidzianej w specyfikacji amplitudzie sygnału CLIP (560 mV) sygnał z wyjścia U1A ma amplitudę 4,5 V. Ze względu na fakt, że kostka TL084 zasilana napięciem pojedynczym nie może pracować z napięciami wejściowymi (wspólnymi) mniejszymi niż 3 V, napięcie V_{in} zasilające dekoder musi wynosić co najmniej 15 V. Zastosowanie w miejsce TL084 nowoczesnego wzmacniacza Rail-to-Rail na wejściach (i najlepiej na wyjściach) pozwoliłoby zasilać układ napięciem od 10 V w górę. Ja jednak w ramach obniżania kosztów urządzenia wybrałem TL084.

Z wyjścia U1A sygnał podawany jest na dolnoprzepustowy filtr Sallena-Keya zbudowany na U1C. Jest to aktywny filtr drugiego rzędu (spadek charakterystyki –40 dB/dekadę) o górnej częstotliwości granicznej wynoszącej 3,5 kHz. Filtr ten bez tłumienia przepuszcza sygnał CLIP, za to bardzo mocno tłumi przebiegi o częstotliwościach przekraczających 20 kHz – o ponad 30 dB czyli ok. 30 razy. Umożliwia on pracę dekodera także wtedy, gdy do linii telefonicznej dołączone jest oprócz telefonu inne urządzenie jak np. modem ADSL itp. Układy takie pracują w paśmie ponad 20 kHz. W sumie elementy R3--C1, R8...C3 oraz R11, R12, C4, C5, U1C tworzą nieco rozproszony filtr pasmowy przepuszczający przebiegi o częstotliwościach 700 Hz...3,5 kHz. Charakterystyka tego filtru w lewej części jest niezbyt dobra (łagodne kolano) jednak w tym zastosowaniu nie ma to żadnego znaczenia.

Z wyjścia filtru niezmieniony sygnał CLIP podawany jest na odwracające wejście komparatora U1B. I tu ujawnia się sedno zasady działania całego dekodera. Dzielnik R9--R10 wytwarza napięcie referencyjne dla komparatora (V_{ref} na rys. 2). Przy wartościach R9-R10 jak na schemacie V_{ref} jest o ok. 100 mV mniejsze od napięcia występującego na wyjściu U1D, które z kolei jest równe składowej stałej przebiegu z wyjścia filtru (bo U1A i U1C dla napięć stałych są wtórnikami). W związku z tym komparator porównuje ze sobą napięcie chwilowe wzmocnionego przebiegu CLIP z napięciem ciut mniejszym (o 100 mV) od jego składowej stałej. Na wyjściu komparatora występuje przebieg prostokątny dokładnie odzwierciedlający sygnał CLIP. Ilustruje to szczegółowo **rys. 4** przy założeniu, że napięcie zasilania wynosi 15 V (oczywiście zasada nie zmienia się dla większych napięć zasilania). Za pośrednictwem T1 dobrej jakości sygnał cyfrowy trafia na wejście INT0 mikrokontrolera. Kondensator C6 filtruje napięcie V_{ref}. Rezystor R13 wprowadza niewielką histerezę do charakterystyki komparatora, co zapobiega występowaniu wszelkich szybkozmiennych przebiegów na jego wyjściu w okolicy poziomu przełączania. Choć ryzyko wystąpienia takich przebiegów istnieje gdy komparator pracuje z sygnałami o bardzo małej częstotliwości - a tu tak nie jest, to jednak niewielka histereza jest bardzo pożądana.

Jak widać z rys. 4 naturalną konsekwencją opisanej zasady wytwarzania przebiegu cyfrowego jest to, że jego wypełnienie nie wynosi dokładnie 50%. Czasy trwania stanów niskiego i wysokiego różnią się o $2*\Delta t$ od połowy okresu sygnału CLIP. Czas Δt jest czasem, w którym wzmocniony sygnał CLIP zmienia się od napięcia V_{ref} do swojej składowej stałej. Podczas dekodowania (więcej w części „Działanie dekodera") stany przebiegu cyfrowego mierzone są z rozdzielczością 10 punktów, co w zupełności wystarcza do poprawnego odróżnienia od siebie sygnałów kodujących 0 i 1. Aby dekodowanie przebiegało poprawnie czas $2*\Delta t$ powinien być mniejszy od 10% połowy okresu CLIP, czyli od 5% całego okresu. Dzięki temu błąd pomiaru spowodowany niedoskonałością metody będzie na poziomie rozdzielczości tego pomiaru, a więc nic nie zaszkodzi. Wynika stąd, że czas Δt powinien być mniejszy od 2,5% czasu trwania całego okresu. Stanowi to kąt równy $0{,}025 \cdot 2 \cdot \pi = 0{,}157$. Sinus w okolicach zera jest funkcją niemal tożsamościową, a więc wartość 100 mV stanowi 0,157 minimalnej amplitudy poprawnego sygnału występującego na wyjściu wzmacniacza U1A. Jest więc ona równa ok. 630 mV. Ponieważ wzmocnienie U1A wynosi 8, minimalna amplituda poprawnie dekodowanego sygnału CLIP wyniesie ok. 80 mV. Ze względu na rozrzuty, przyjmuję pesymistycznie okrągłą wartość 100 mV. Jest to czułość mojego dekodera.

Powyższe rozważanie oznacza, że opisywany dekoder pracuje z przebiegami CLIP o amplitudzie 100...560 mV. Jak widać z **tab. 1** i rys. 1, zakres amplitud V.23 wynosi 15...560 mV. Ze względu na tak szeroki zakres napięć w profesjonalnych układach tego typu stosuje się układy ARW (automatycznej regulacji wzmoc-

Tab. 1. Parametry sygnału dla protokołu V.23

Parametr	Wartość
Mark (Logiczne 1)	1300 Hz ±1,5%
Space (Logiczne 0)	2100 Hz ±1,5%
Poziom sygnału mark	–40 dBV do –8 dBV (10...398,1 mV RMS)
Poziom sygnału space	–40 dBV do –8 dBV (10...398,1 mV RMS)
Różnice poziomów	6 dB max (2 razy)
Poziom sygnałów niepożądanych	max –20 dB poniżej poziomu sygnału (300 – 3400 Hz)
Prędkość transmisji	1200 baud ±1%
Format danych	Transmisja szeregowa asynchroniczna. 1 bit START, 8 bitów danych, 1...10 bitów STOP. Bit START 0. Bit STOP 1.

Tab. 1.

```
// Przerwanie co 26us.
SIGNAL (SIG_OUTPUT_COMPARE1A)
{
   // Deklaracje i definicje zmiennych statycznych.
   // BitNumber - jedyna zmienna lokalna automatyczna.
   BaudCounter++;

   // Pomiar dlugosci trwania impulsow (zarowno stanow H i L).
   // IN - alias dla pinu PD2(INT0).
   if(tmp==IN)
   {
      cou++;
   }
   else
   {
      tmp=IN;
      LastPulseMem=LastPulseTime;
      LastPulseTime=cou;
      cou=0;
   }
   // Tutaj mamy dlugosc ostatniego impulsu w zmiennej LastPulseTime,
   // zas poprzedniego w zmiennej LastPulseMem.
   if(ClipState==CLIPSTATE_MESS)
   {
      if(BaudCounter>304 && cou==0)
      {
         // Po srodku bitu stopu -
         // - oczekiwanie na bit startu nastepnego bajtu.
         // (304*26us)/833,3us=9,5 (polowa bitu stopu).
         return;
      }

      if(BaudCounter==304)
      {
         // Srodek bitu stopu -
         // zapamietanie odczytanego bajtu (CurrentByte)
         // w tablicy Table[].
         // Jesli odczytano odpowiednio wiele bajtow -
         // - zakonczenie dekodowania.

         return;
      }

      if(BaudCounter>32 && BaudCounter<288)
      {
         // Bity LSB...MSB
         // BitNumber - nr. bitu (0...7)
         // Liczby okreslaja momenty
         // pod koniec nadawania kolejnych bitow.
         // np. (123*26us)/833,3us=3,8 -> BitNumber=2
         // (bo odrzucamy bit START i numerujemy od 0).
         BitNumber=255;
         if(BaudCounter==59)
            BitNumber=0;
         if(BaudCounter==91)
            BitNumber=1;
         if(BaudCounter==123)
            BitNumber=2;
         if(BaudCounter==155)
            BitNumber=3;
         if(BaudCounter==187)
            BitNumber=4;
         if(BaudCounter==219)
            BitNumber=5;
         if(BaudCounter==251)
            BitNumber=6;
         if(BaudCounter==283)
            BitNumber=7;


         if(BitNumber==255)
            return;

         // Zapalenie lub wyzerowanie kolejnego bitu
         // w zmiennej CurrentByte.

         if(LastPulseTime>(ThresholdTime+1))
            CurrentByte|=(1<<BitNumber);
         else if(LastPulseTime<(ThresholdTime-1))
            CurrentByte&=~(1<<BitNumber);
         else
         {
            if(LastPulseMem>ThresholdTime)
               CurrentByte|=(1<<BitNumber);
            else
               CurrentByte&=~(1<<BitNumber);
         }

         return;
      }
      return;
   }

   if(ClipState==CLIPSTATE_SMMR && cou==0)
   {
      // Pomiar sredniego czasu trwania impulsu
      // w czasie 128 impulsow kodujacych naprzemiennie 0 i 1.
      // Nadanie wartosci zmiennej ThresholdTime.
      // Na koniec przejscie do stanu CLIPSTATE_MARK

      return;
   }

   if(ClipState==CLIPSTATE_MARK && cou==0)
   {
      // Oczekiwanie na przyjscie zera (impulsy krotsze od ThresholdTime)
      // i przejscie do stanu CLIPSTATE_MESS.
      return;
   }
}
```

nienia). Jednak w praktyce operatorzy starają się dostarczać sygnał CLIP o amplitudzie co najmniej 100 mV ze względu na duże ryzyko zakłócania go przy amplitudach rzędu kilkudziesięciu mV. Dlatego projektując dekoder zdecydowałem się na prosty wzmacniacz nieodwracający w miejsce układu ARW. Kto chce może zwiększyć czułość dekodera. Najprostszym sposobem będzie zastosowanie w miejsce TL084 nowoczesnego wzmacniacza z wejściami i wyjściami typu Rail-to-Rail mogącego pracować z napięciami zasilania ponad 20 V i zasilić dekoder napięciem np. 20 V. Wzmacniacze *Rail-to-Rail* mogą pracować z napięciami wejściowymi na poziomie zarówno 0 V jak i V_{dd}. Dlatego stosując taki wzmacniacz można, przy zasilaniu 20 V, zwiększyć wzmocnienie U1A do ok. 17...18 (pamiętając o tym, że reaktancja C3 wynosi dla 1 kHz ok. 700 Ω i jest porównywalna z rezystancją R8). Zapewni to czułość ok. 630 mV/18=35 mV. Ewentualnej niewielkiej korekty wymagać będzie dzielnik R9-R10 (wystarczy zmiana R9 na 1...1,2 kΩ). Nie polecam natomiast zwiększania czułości poprzez zbliżanie napięcia V_{ref} do połowy napięcia zasilania. Różnica między nimi powinna zapewnić margines zakłóceń na poziomie co najmniej 50 mV przy uwzględnieniu napięcia niezrównoważenia wzmacniacza, które dla kostki TL084 może wynosić nawet 15 mV. Oznacza to, że różnica ta powinna wynosić co najmniej 65 mV. Wartość 100 mV jest tu moim zdaniem optymalna. Dla kostek o mniejszym napięciu niezrównoważenia (nowoczesne wzmacniacze mają je na poziomie dużo poniżej 1 mV) można pokusić się o zmniejszenie tej różnicy, ale nie bardziej niż do 50...60 mV.

Część cyfrowa dekodera jest dość typowa. Tworzy ją mikrokontroler U2 wraz z wyświetlaczem LCD. Ze względu na sposób dekodowania konieczne było zapewnienie dużej szybkości pracy procesora, stąd obecność kwarcu 10 MHz w obwodzie wytwarzania przebiegu zegarowego. Poprawny reset procesora po włączeniu napięcia zasilającego (lub ewentualnych szkodliwych spadkach tego napięcia) zapewnia zewnętrzny układ Brown-out-Reset U5 typu DS1813 (lub DS1811). Zastosowanie w miejsce 90S2313 jego nowszej wersji ATtiny2313, która w porównaniu z protoplastą posiada wewnętrzny układ BOR, uwalnia nas od konieczności stosowania DS1813. W takim przypadku podczas programowania ATtiny2313 należy ustawić odpowiedni bezpiecznik (*fusebit*).

Arkadiusz Antoniak
arkadiusz.antoniak@wp.pl

WYKAZ ELEMENTÓW

Rezystory
R1: 22 kΩ
R2, R8, R15: 1,2 kΩ
R3, R11, R12: 33 kΩ
R4, R5, R6, R10, R19: 100 kΩ
R7, R14: 10 kΩ
R9: 1,5 kΩ
R13: 2,2 MΩ
R16, R18: 1 kΩ
R17: 47 kΩ
R20: 4,7 kΩ
R100: 0 Ω (zwora)
PR1: 47 kΩ trymer stojący

Kondensatory
C1, C6: 10 nF
C2, C4: 2,2 nF
C3: 220 nF
C5: 1 nF
C7, C13, C14, C15, C16: 100 nF
C8, C18: 100 µF/16 V
C9, C10: 470 nF
C11, C12: 22 pF
C17: 100 µF/25 V

Półprzewodniki
U1: TL084
U2: AT90S2313 lub ATtiny2313
U3: LCD 16*2
U4: 78L05
U5: DS1813 lub DS1811
D1, D2: 1N4148
D3: BAT85
T1, T2: BC548B
M1: mostek 1,5 A

Inne
CON1: złącze telefoniczne do druku
CON2: ARK2
X1: 10 MHz

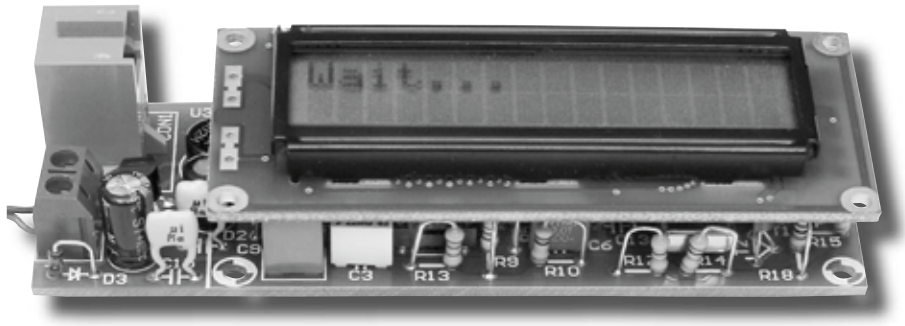
Programowy dekoder CLIP (FSK), część 2

AVT-590

W artykule prezentujemy układ wyświetlający numer abonenta dzwoniącego (CLIP - Calling Line Identification Presentation). W przeciwieństwie do poprzednio opisywanych urządzeń, nie wymaga żadnego specjalizowanego układu scalonego.

Rekomendacje:

możliwość odczytu numeru abonenta dzwoniącego jest bardzo cenna. Umożliwia użytkownikowi telefonu chociażby uniknięcia niechcianych rozmów. Dla elektroników budujących wszelkiego rodzaju układy „telefoniczne" funkcja CLIP jest bezcenna, bo pozwala budować urządzenia o ogromnych możliwościach, często przewyższających możliwości urządzeń fabrycznych. Mogą to być wszelkiego rodzaju rejestratory rozmów telefonicznych, taryfikatory itp.



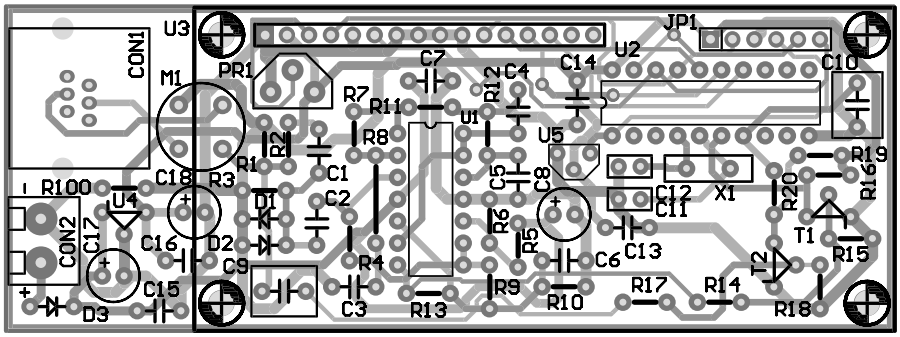
Działanie dekodera

Program dekodujący napisałem w C używając kompilatora AVR-GCC (kompilacja 2002-06-25) ze względu na jego popularność i dostępność za darmo. Kody źródłowe, makefile i kody wynikowe dostępne są na stronie internetowej EP.

W pętli głównej procesor przy odblokowanym przerwaniu INT1 oczekuje w stanie IDLE na nadejście sygnału dzwonka. Po pojawieniu się dzwonka (CALL) oczekuje na jego zakończenie, a następnie blokuje przerwanie INT1 i odblokowuje INT0. W tym stanie procesor ponownie wchodzi w stan IDLE i oczekuje na przyjście przebiegu prostokątnego wytworzonego z sygnału CLIP. Wszystkie oczekiwania objęte są odpowiednimi timeout-ami, co zapewnia zawsze poprawną reakcję na zakłócenia jakie mogą się pojawić (procesor nigdy nie wejdzie w nieskończoną pętlę oczekiwania na coś, co się nie stanie). Do realizacji jednego z tych timeout-ów użyłem Timera1 w trybie zliczania f_{clk}/1024 i włączonym przerwaniu od przepełnienia. Ten sam timer z innymi ustawieniami służy potem do dekodowania sygnału CLIP. Procedury obsługi przerwań INT0 i INT1 są puste, gdyż przerwania te (oba wyzwalane zboczem opadającym) służą jedynie do sygnalizacji nadejścia sygnałów CLIP i CALL i wyrwania procesora ze stanu uśpienia.

Odczyt wartości kolejnych bitów z sygnału CLIP odbywa się poprzez pomiar czasów trwania stanów wysokiego i niskiego (oba nazywam dalej impulsami) przebiegu prostokątnego, który pojawia się na nóżce PD2(INT0). Wykorzystałem do tego 16-bitowy Timer1 ustawiony w tryb CTC (*Clear Timer on Compare*), który zlicza impulsy bezpośrednio z generatora taktującego (10 MHz). Pomiar odbywa się w procedurze obsługi przerwania od porównania stanu timera z wartością rejestru OCR1, która wynosi 260 (0x0104) i jest stała przez cały czas dekodowania. Oznacza to, że przerwanie wywoływane jest ze stałym interwałem 26 μs. Przy częstotliwościach kodowania bitów rzędu 1...2 kHz (**rys. 1b**) pozwala to uzyskać rozdzielczość pomiaru rzędu 10 punktów co w zupełności wystarcza do odróżnienia zera od jedynki. Jednocześnie wartość 26 μs stanowi bardzo dokładną podwielokrotność okresu taktowania transmisji równego 833,(3) μs i nadaje się do odmierzania czasu w obrębie pojedynczej ramki bajta (błąd względny wynosi mniej niż 0,2% i jest jak najbardziej dopuszczalny).

Załóżmy, że po dzwonku nadszedł poprawny sygnał CLIP i na INT0 pojawia się odpowiedni przebieg prostokątny. Procesor wychodzi z drugiego stanu IDLE, ustawia Timer1 jak opisałem wyżej i włącza go. Następnie oczekuje w pętli głównej na zakończenie dekodowania (zapalenie flagi *fEnd*). Cały proces dekodowania odbywa się w przerwaniu. Na **list. 1** przedstawiona jest procedura jego obsługi, z wyłączeniem szczegółów nieistotnych z punktu widzenia zasady działania. Do pomiaru długości impulsów służy zmienna *cou* inkrementowana za każdym razem gdy stan PD2 nie zmienił się od poprzedniego przerwania. Jeśli się zmienił, wartość jej przepisywana jest do zmiennej *LastPulseTime*, a jej wartość z kolei do zmiennej *LastPulseMem*. Dzięki temu informacja o długościach dwóch ostatnich impulsów



Rys. 5. Schemat montażowy płytki drukowanej

jest cały czas znana. Ważną rolę pełni zmienna *ClipState* przechowująca informację o tym, który blok pakietu CLIP (SMMR, MARK czy MESSAGE) jest aktualnie analizowany. W stanie CLIPSTATE_SMMR (strumień bitów na przemian 0 i 1) program nastraja się - wyznacza graniczny czas pomiędzy impulsami kodującymi 0 i 1, który umieszcza w zmiennej *ThresholdTime*. Zmienna ta wykorzystywana jest później do określania wartości bitów w bloku MESSAGE. Przechodzenie między kolejnymi stanami odbywa się na podstawie zawartości zmiennych *LastPulseTime* i *ThresholdTime*.

W stanie CLIPSTATE_MESS dokonuje się właściwa analiza przesyłanych ramek bajtów (**rys. 1d**). Zmienna *BaudCounter* (inkrementowana przy każdym wywołaniu przerwania) stanowi „zegarek" wyznaczający szybkość transmisji. Jej wartość określa, w jakiej części ramki program aktualnie się znajduje. Porównując ją z odpowiednimi liczbami można określić, czy nadszedł bit startu, któryś (i który) z bitów informacyjnych LSB...MSB czy też bit stopu. Na podstawie tej zmiennej i zmiennych *LastPulseTime* i *LastPulseMem* w odpowiednich momentach pod koniec nadawania każdego z bitów LSB...MSB modyfikowana jest zmienna *CurrentByte*, która po zakończeniu analizy ramki zawiera bajt w niej przesłany. Po wykryciu bitu stopu (znowu na podstawie *BaudCounter*) wartość *CurrentByte* wpisywana jest pod kolejnym indeksem do globalnej tablicy *Table[]*. Po zakończeniu dekodowania tablica ta zawiera wszystkie bajty bloku MESSAGE w kolejności ich wysyłania (od T1 do CHECKSUM). Cały proces kończy się gdy indeks w tablicy przekroczy wartość *Table[1]+3*, która równa jest długości (ilość bajtów) pakietu MESSAGE.

Cała procedura zabezpieczona jest przed wieloma błędami, jakie mogą tutaj wystąpić. Jeśli wszystko przebiegło poprawnie globalna flaga *MessError* nie zostaje zapalona. Zabezpieczenia nie zostały pokazane na list. 1 aby nie przysłoniły sedna zasady dekodowania. Po więcej informacji odsyłam do kompletnego kodu źródłowego. Przyczyną zakończenia dekodowania z błędem może być na przykład zbyt duża (przekraczająca 30) wartość *Table[1]*. Jeśli zaś po pierwszym dzwonku nie nadszedł poprawny sygnał CLIP lecz „śmieci" nie niosące żadnej informacji to może się zdarzyć, że program nigdy nie wejdzie w część wpisywania danych do tablicy *Table[]* gdzie jednocześnie odbywa się wykrywanie końca dekodowania. Spowodowałoby to po prostu zawieszenie się programu. Ten i wszelkie inne błędy „załatwia" globalny timeout nałożony na przerwanie, który realizuje zmienna *TimeoutCounter* typu unsigned int. Jest ona inkrementowana w każdym przerwaniu, a osiągnięcie przez nią wartości 40000 (czas ok. 1 s) powoduje zakończenie procesu z błędem (zapalenie flag *fEnd* i *MessError*).

Po zakończeniu dekodowania program przechodzi do dalszej części pętli głównej. Za pomocą funkcji *ErrorCheck()* sprawdza czy wystąpiły błędy. Jeśli nie, informacje o numerze i aktualnej dacie odzyskane z tablicy *Table[]* trafiają na wyświetlacz LCD. Funkcja *ErrorCheck()* sprawdza flagę *MessError*, a jeśli nie jest ona zapalona oblicza i zwraca sumę modulo 2 wszystkich bajtów tablicy *Table[]*. W pętli głównej sprawdzane jest także, czy wysyłany numer jest zastrzeżony lub abonent jest niedostępny (korzysta z centrali analogowej). Informacja o tym zawarta jest w pierwszym bajcie numeru (bloku V3), który zapisany jest w tablicy *Table[14]*. Jeśli ma on wartość 80 to numer jest zastrzeżony, jeśli zaś 79 – abonent jest niedostępny.

Montaż i uruchomienie

Schemat montażowy znajduje się na **rys. 5**. Sam montaż nie powinien sprawić nikomu problemu. Pod układy scalone należy zastosować podstawki, najlepiej precyzyjne. Elementy znajdujące się pod wyświetlaczem LCD powinny mieć jak najmniejszą wysokość, dotyczy to zwłaszcza kondensatorów elektrolitycznych. Rezonator kwarcowy powinien być wlutowany jako leżący, można też wlutować niski kwarc o wysokości 4 mm. Jako łączący masy analogową i cyfrową rezystor R100 należy wlutować zworkę. Po zmontowaniu układu trzeba zaprogramować mikrokontroler U2. Do tego celu można użyć dowolnego programatora ISP (np. popularnego STK200) i jakiegokolwiek współpracującego z nim programu na PC. Przy programowaniu należy pamiętać, że masa układu nie jest galwanicznie oddzielona od linii telefonicznej i na czas podłączenia kabla ISP dekoder powinien być od niej odłączony. Stosując w miejsce 90S2313 nowszy mikrokontroler ATtiny2313 musimy pamiętać o odpowiednim ustawieniu fusebitów (zewnętrzny generator kwarcowy, włączony BOR itp.).

Poprawnie zmontowany układ od razu działa poprawnie i nie wymaga żadnych czynności uruchomieniowych. Warto jednak kierując się opisem obejrzeć oscyloskopem lub rejestratorem analogowym przebiegi w kluczowych punktach układu i sprawdzić, czy są prawidłowe.

Arkadiusz Antoniak
arkadiusz.antoniak@wp.pl