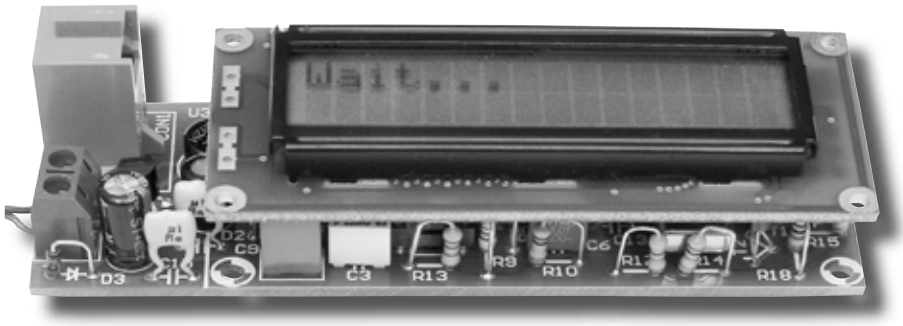
Programowy dekoder CLIP (FSK), część 2

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.



Dodatkowe materiały do artykułu publikujemy na CD-EP oraz www.ep.com.pl

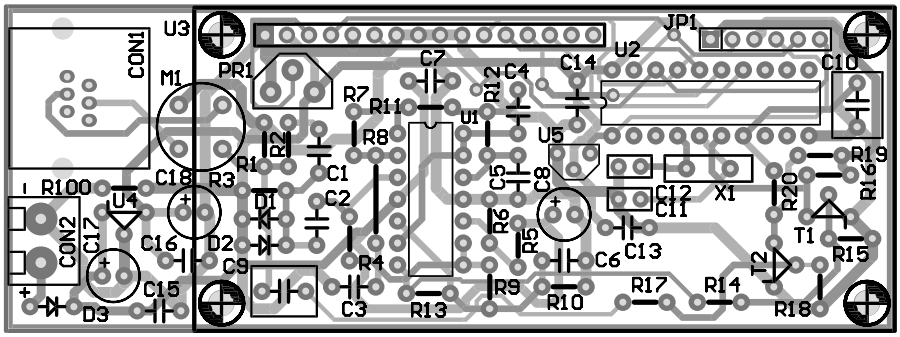
Działanie dekodera

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 *MesError* 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