Koder Morse'a współpracujący z klawiaturą AT

kit 3121 AVT

Układ umożliwia edukującą zabawę wynalezionym prawie półtora wieku temu kodem telekomunikacyjnym.

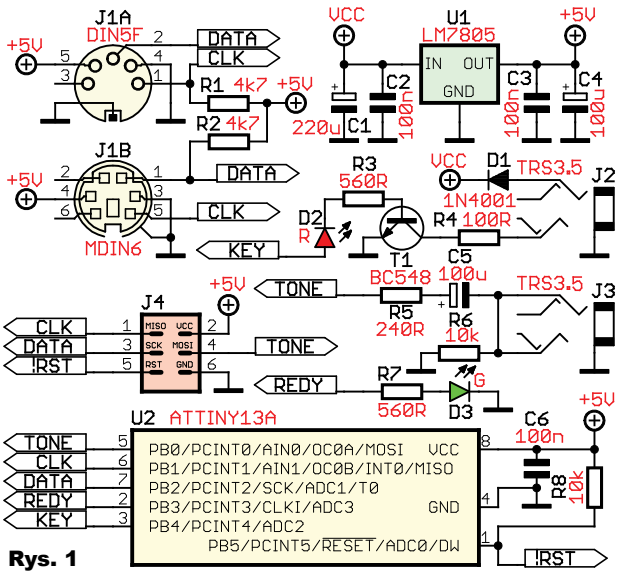
Do czego to służy?

Kod wynaleziony w pierwszej połowie XIX wieku przez Samuela Morse'a, mimo zakończenia długiej „kariery" w komunikacji profesjonalnej, jest nadal wykorzystywany przez amatorów (krótkofalowców). Wprawne posługiwanie się kluczem sztorcowym czy dwudźwigniowym jest nie lada sztuką. By nauczyć się posługiwania kluczem, wymagane jest wcześniejsze zaznajomienie się z poszczególnymi kodami znaków Morse'a, co w praktyce oznacza osłuchanie się z ich „melodyką". W zasadzie do nauki kodu Morse'a wystarczy odpowiednie oprogramowanie na komputer PC. Rozwiązanie sprzętowe ma jednak tę zaletę, że uniezależnia proces nauki od komputera PC. Podane informacje pozwolą też wykorzystać klawiaturę PC do innych celów.

Jak to działa?

Na **rysunku 1** przedstawiono schemat ideowy. Napięcie zasilania (+12V) z gniazda J2, przez D1 zabezpieczającą przed błędną polaryzacją, doprowadzane jest do stabilizatora U1 pracującego w typowej konfiguracji. Stabilizator, prócz zasilania obwodów układu napięciem +5V, zasila również dołączaną do złącza J1 klawiaturę AT.

Każdemu klawiszowi klawiatury AT przypisany jest unikalny kod i na jego podstawie można rozróżnić dwie podstawowe grupy klawiszy: standardowe, specjalne. Standardowe posiadają przypisany kod jednobajtowy. Kontroler klawiatury rozróżnia trzy stany klawisza tj. naciśnięcie, przytrzymanie i puszczenie. Naciśnięcie powoduje wysłanie kodu klawisza. Jego puszczenie wygeneruje kod klawisza poprzedzony bajtem o wartości $F0. Przykładowo naciśnięcie klawisza „A" spowoduje przesłanie bajtu o wartości $1C, a jego zwolnienie dwóch bajtów: $F0$1C. Inaczej jest dla klawiszy specjalnych, gdzie kod jest dwubajtowy i zaczyna się od wartości $E0. Przykładowo naciśnięcie klawisza „R-ALT" spowoduje wysłanie bajtów: $E0$11, puszczenie: $E0$F0$11. Dla obu grup klawiszy ich przytrzymanie powoduje powtarzanie przesyłania ich kodu (po niewielkim opóźnieniu od naciśnięcia). Wyjątkami od opisanych reguł są dwa klawisze o kodach wielobajtowych, tj. Print Screen, Pause/Break.



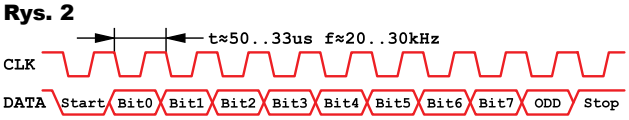
Rys. 1

Dane z klawiatury AT przesyłane są przez dwuprzewodowy, synchroniczny interfejs szeregowy. Obie linie (zegarowa, danych) magistrali są podciągnięte rezystorami R1, R2 do +5V. Ramka protokołu widoczna na **rysunku 2** jest jedenastobitowa i składa się kolejno: z bitu startu (wartość L), ośmiu bitów danych w kolejności od najmniej znaczącego bitu, bitu kontroli parzystości typu ODD (przyjmuje on wartość H, gdy liczba jedynek w przesyłanym bajcie jest parzysta) i bitu stopu (wartość H). Za wytwarzanie sygnału zegarowego odpowiedzialna jest zawsze klawiatura. Twórcy interfejsu przewidzieli możliwość przesyłania danych do klawiatury, jednak opis komunikacji „do klawiatury" wykracza poza ramy niniejszego artykułu.

Program dla mikrokontrolera U2 napisany został w kompilatorze Bascom-AVR. Większość kodu napisana jest jako wstawka asemblerowa, jedynie definicje stałych, zmiennych oraz instrukcje opóźnienia są wysokopoziomowe. Takie podejście podyktowane jest tym, by kod wynikowy zmieścił się w „skromnej" pamięci flash mikrokontrolera ATtiny13A.

Choć w Bascom AVR istnieje funkcja (GetAtKbdRaw) umożliwiająca odczyt nieprzetworzonych kodów z klawiatury AT, to w przypadku chęci buforowania odebranych bajtów ma istotną wadę. Funkcja nie ma zaimplementowanego mechanizmu przeterminowania czasu oczekiwania, dlatego wstrzymuje działanie programu do czasu odebrania bajtu z klawiatury. Co prawda możliwe jest wcześniejsze zakończenie jej działania przez ustawienie zmiennej Err w przerwaniu np. od przepełnienia licznika. Jednak gdy do wykrywania przychodzących bajtów (zbocza sygnału zegarowego) użyć przerwania zewnętrznego INT0, które ma najwyższy priorytet (zaraz po Reset), to w trakcie jego trwania niemożliwa jest obsługa innego przerwania, co przekreśla sens takiego rozwiązania.

Rys. 2



Zaprezentowana niskopoziomowa obsługa przerwania INT0 sprawdza czas upływający między zboczami sygnału zegarowego, tworząc skuteczny mechanizm przeterminowania czasu oczekiwania na dane z klawiatury AT. Odbiór danych nie wstrzymuje działania programu i umożliwia buforowanie wyliczonego, jednego kodu Morse'a „do przodu" (w trakcie generowania poprzedniego znaku), co w tym zastosowaniu jest wystarczające. Po wykryciu zbocza opadającego na wejściu INT0 (CLK) program przechodzi do obsługi przerwania (włączona opcja Nosave oznacza brak automatycznej obsługi stosu przez kompilator). Jedynym rejestrem, który jest modyfikowany przez program główny i obsługę przerwania INT0, jest rejestr statusu SREG i jego zawartość jest odkładana na stos. Ponieważ obie części programu korzystają z „dedykowanych sobie" (różnych) rejestrów roboczych, nie istnieje potrzeba „zapamiętywania" ich zawartości na stosie. Następnie wyłączane jest zezwolenie na obsługę przerwania INT0 (bit 6 rejestru GIMSK). Stan linii PINB.2, czyli wartość odebranego bitu jest kopiowany do znacznika C w rejestrze statusu SREG i wsuwany do rejestru roboczego z lewej strony mnemonikiem ROR. Po odebraniu jednego bitu następuje oczekiwanie kolejno na zbocze narastające i gdy bit nie jest bitem stopu, zbocze opadające sygnału zegarowego. Podczas oczekiwania inkrementowana jest zawartość rejestru R19 i porównywana ze stałą TimeOut. Zrównanie zawartości R19 z tą stałą oznacza przekroczenie czasu oczekiwania na zbocze (przeterminowanie) i zakończenie obsługi przerwania. Oczywiście przerwanie zakończy się, jeżeli podczas sprawdzania bitów startu, parzystości, stopu wystąpi błąd. We wszystkich ww. przypadkach przerwanie przekaże do pętli głównej wartość zero. Jeżeli odczyt bajtu zakończył się sukcesem, to w zależności od jego wartości podejmowane są odpowiednie działania. Jeżeli odebrano „zwiastun" klawisza specjalnego $E0, to fakt ten zostanie zapamiętany i bajt odebrany w następnej obsłudze przerwania interpretowany będzie jako kolejny bajt kodu klawisza specjalnego. Podobnie jest w przypadku odebrania zwiastuna zwolnienia klawisza $F0, bajt odebrany w następnej obsłudze przerwania interpretowany będzie jako kolejny bajt kodu klawisza puszczonego (włącznie ze zwiastunem klawisza specjalnego). Na podstawie tablicy stałych KbdRawData wyliczany jest indeks kodu znaku Morse'a w tablicy MorseaData. Jeżeli któryś z obu klawiszy SHIFT lub prawy klawisz ALT jest wciśnięty, to stosownie do wciśniętego klawisza „znakowego", do wyliczonego indeksu (dla tablicy MorseaData) dodawany jest wektor przesunięcia. W ten sposób realizowana jest obsługa diakrytyków (R-ALT) oraz znaków alternatywnych (SHIFT). „Mapę" obsługiwanych klawiszy obrazuje **rysunek 3**. Przed zakończeniem obsługi przerwania kasowany jest bit żądania jego obsługi (bit 6 rejestru GIFR), w czasie wykonywania obsługi przerwania wystąpiło bowiem dziesięć zboczy opadających sygnału zegarowego ustawiających tę flagę. Na koniec włączane jest zezwolenie na obsługę przerwania INTO (bit 6 rejestru GIMSK) i zdjęcie ze stosu zapamiętanej przed obsługą przerwania zawartości rejestru statusu.

Pobrana w obsłudze przerwania wartość z tablicy MorseaData jest przekazywana przez zmienną Code do pętli głównej. Gdy przyjmuje wartość zero nie są podejmowane są żadne działania. Gdy wartość odpowiada naciśnięciu obsługiwanego klawisza funkcyjnego, program realizuje modyfikację obowiązujących ustawień. I tak, za regulację tempa kodowania znaków (czas trwania kropki w zakresie 40...120ms) odpowiadają klawisze F5, F6. Regulacja częstotliwości tonu generatora (≈0,5...≈2,5kHz) odsłuchu możliwa jest za pomocą klawiszy F7 i F8. F2 zapisuje aktualne ustawienia tempa i tonu w pamięci EEPROM. Natomiast ich odczyt (ustawienie ich jako obowiązujących) realizowany jest naciśnięciem F3. Obsługiwane klawisze funkcyjne oraz przypisania specjalnych kodów Morse'a widoczne są w tabelach na **rysunku 4**. Naciśnięcie SPACE jest interpretowane jedynie podczas generowania zbuforowanego poprzednio znaku i powoduje przedłużenie „ciszy" po zakończaniu znaku do długości trzech interwałów kreski, odpowiadającej pauzie między słowami.

W innych przypadkach wartość zmiennej interpretowana jest według algorytmu, który zobrazowano (dla litery C) na **rysunku 5**. Poszczególne znaki Morse'a zakodowane są na jednym bajcie, gdzie binarna jedynka odpowiada kropce, a zero kresce. Ponieważ poszcze-

Rys. 3

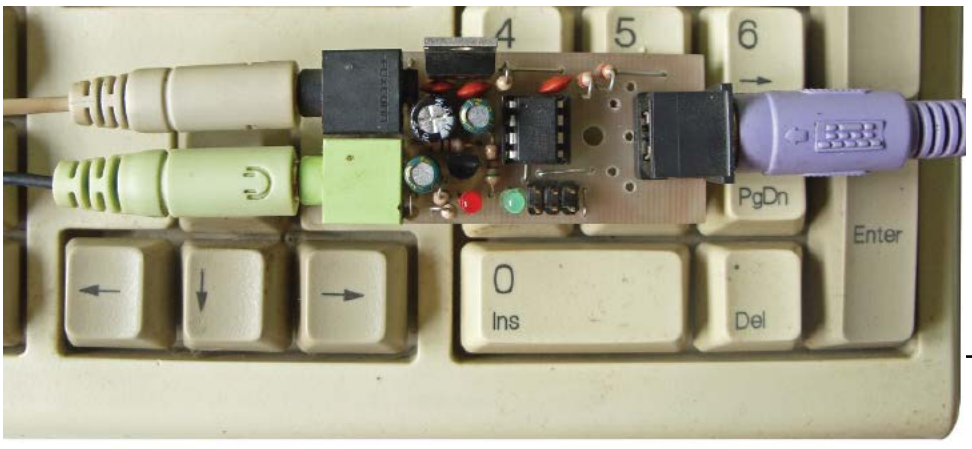
Klawisz	Funkcja
F2	Zapis ustawień do EEPROM
F3	Odczyt ustawień z EEPROM
F5	Zmniejszenie tempa kodowania
F6	Zwiększenie tempa kodowania
F7	Zmniejszenie częstotliwości tonu
F8	Zwiększenie częstotliwości tonu

Klawisz	Kod	Znaczenie
ESC	•••–•–	Koniec pracy
K	–•–	Wezwanie
BKSP	••••••••	Błąd
Ł	•–••–	Znak rozdziału
!	–•–•– –	KW
@	•– –•–•	AC
$	•••–••–	SX
&	•–•••	Czekaj
?	••– –••	Prośba o powtórzenie
+	•–•–•	Koniec nadawania
KP -	–•••–	Początek nadawania
KP +	•••–•	Zrozumiano
KP 0..9	.....	Kody skrócone cyfr
SPACE		Pauza między słowami

Rys. 4

```
C –•–•
  01011000
Start:
– 01011000
   ← LSL
• 10110000
   ← LSL
– 01100000
   ← LSL
• 11000000
   ← LSL
  10000000
CPI &H80
BRNE Start
Stop:
```

Rys. 5

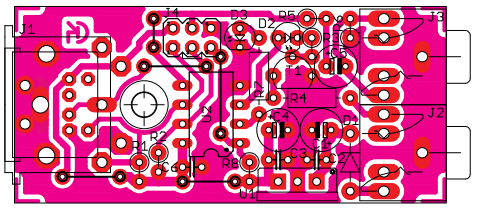


gólne znaki różnią się liczbą impulsów, w bajcie na pierwszej wolnej pozycji za kodem znaku „wstawiona" jest nieznacząca jedynka a po niej zera. W pętli generującej kod Morse'a następuje cykliczne przesuwanie rejestru roboczego R20 o jeden bit w lewo. Najbardziej znaczący bit rejestru wyznacza czas trwania impulsu (kropka lub kreska) na nóżce PB.4 (KEY) oraz czas trwania tonu na wyjściu OC0A (TONE). Podczas przesuwania do najmniej znaczącego bitu „wsuwane" jest zero. Dzieje się tak do uzyskania wartości &B10000000 w R20, co oznacza koniec przetwarzania znaku. Taka metoda umożliwia generowanie znaków o długości maksymalnie siedmiu impulsów. Generowanie znaku „błąd" przypisanego klawiszowi Backspace (osiem kropek), jako wyjątku, obsłużono manipulacją znacznikiem T w rejestrze SREG. Znacznik T jest ustawiany przed pętlą przetwarzania znaku. W pierwszym jej wykonaniu (po pierwszym przesunięciu w lewo zawartości R20), jeżeli tylko wartość w rejestrze R20 jest równa &B11111110, stan znacznika jest kopiowany do najmniej znaczącego bitu R20 i zmienia zawartość R20 na &H11111111. Następnie znacznik T jest kasowany i przepisywanie jego wartości do najmniej znaczącego bitu R20 w następnych przebiegach pętli nie zmienia już zawartości R20. Między impulsami wstawiane są pauzy zgodnie ze specyfikacją kodu Morse'a, tj. czas „ciszy" między impulsami znaku równy interwałowi kropki, czas pauzy po znaku (pauza między znakami) równy czasowi trwania kreski. W czasie trwania impulsów znaku Morse'a licznik T/C0 pracuje w trybie zerowania licznika przy zgodnym porównaniu (CTC – Clear Timer on Compare Match). Licznik z interwałem zależnym od ustawionego podziału zwiększa zawartość rejestru TCNT0 i porównuje z wpisaną wcześniej do rejestru OCR0A wartością. W momencie zrównania zawartości obu rejestrów rejestr TIMER0 zaczyna liczyć od zera, a stan wyjścia OC0A zmienia się na przeciwny (ustawiony bit COM0A0 w rejestrze TCCR0A), generując w ten sposób ton akustyczny o częstotliwości wyznaczanej przez zawartość rejestru OCR0A. Przydatny do odsłuchu generowanego kodu przebieg przez elementy R5, C5 jest doprowadzony do gniazda J3, gdzie można podłączyć wzmacniacz m.cz. lub słuchawki (z regulacją tłumienia, potencjometr). C5 odcina składową stałą, a wartość R5 zabezpiecza nóżkę procesora przed przepływem znaczącego prądu w przypadku zwarcia w przewodach. R6 zapobiega „trzaskom" przy dołączaniu wzmacniacza m.cz. do zasilanego układu. Jak wyżej wspomniano, bufor kodów Morse'a umożliwia wczytanie kodu w trakcie generowania poprzedniego znaku. Zaświecenie LED D3 sygnalizuje gotowość na naciśnięcie klawisza klawiatury. Płynne „pisanie morsem" w praktyce oznacza cykliczne gaszenie LED D5 przez naciśnięcie klawisza. Obserwacja D3 zapewnia poprawność czasową generowanego kodu. Takie rozwiązanie zapewnia nadążanie osoby piszącej za „melodią" generowanego kodu. Podczas trwania impulsu kodowanego znaku Morse'a (kropka lub kreska) prąd wypływający z nóżki PB.4 (KEY) zaświeca LED D2 w takt impulsów generowanych znaków i otwiera T1 pracujący w konfiguracji OC, który może sterować (z gniazda J2) zewnętrznym obwodem wykonawczym np. LED (przy nadawaniu optycznym sygnałów). Osoby posiadające odpowiednią licencję na nadawanie mogą wypróbować koder, podłączając wyjście do wejścia klucza w radiostacji (sterowanego masą). Celem trzysekundowego opóźnienia na początku programu (po podłączeniu zasilania) jest zignorowanie (nieprzetwarzanie) kodu ($AA) zwracanego przez klawiaturę AT po wykonaniu procedury inicjującej tzw. selftest.

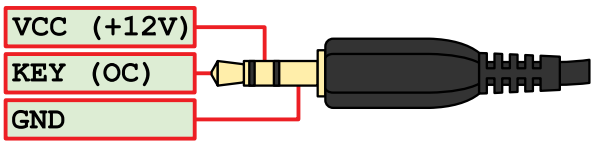
Montaż i uruchomienie

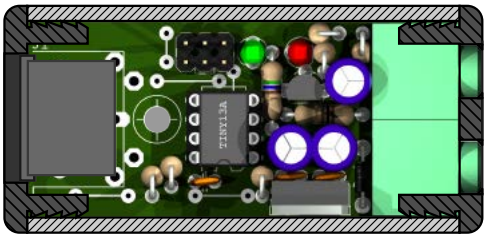
Jednowarstwowy obwód drukowany widoczny jest na **rysunku 6**. Po sprawdzeniu PCB na występowanie zwarć i pęknięć można przystąpić do montażu. W pierwszej kolejności należy zamontować pięć zwór. Pozostałe elementy warto montować, stosując kryterium gabarytowe tj. w kolejności od najmniejszych do największych. Wskazany jest montaż U2 w podstawce. Większość rezystorów montowana jest pionowo (tzw. łany zboża). Typ montowanego gniazda J1 (DIN5F, Mini DIN6) zależeć będzie od wtyku współpracującej klawiatury AT. Jako złącza J2, J3 zastosowano gniazda TRS 3,5mm (pot. minijack stereo)

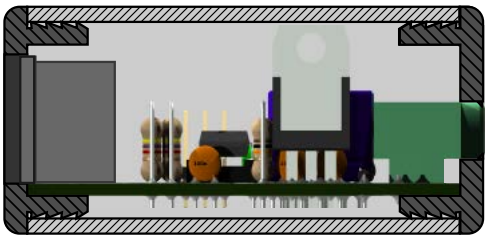
Rys. 6



Rys. 7







Rys. 8

pochodzące z „wylutu” z płyt głównych komputerów PC, kart dźwiękowych. Gniazda tego typu są dobrej jakości, czego nie można powiedzieć o popularnych złączach dostępnych w handlu. Kod programu źródłowy, maszynowy oraz wsad EEPROM udostępniono na Elportalu w materiałach do niniejszego artykułu. Do zaprogramowania U2 przewidziano złącze J4 w standardzie Atmel ISP. Podczas programowania U2 do gniazda J1 nie powinna być podłączona klawiatura. Układ zmontowany ze sprawnych elementów (z zaprogramowanym mikrokontrolerem) wymaga jedynie sprawdzenia poprawności działania. Wymagane jest wyłączenie dzielnika zegara systemowego tj. fabrycznie ustawionego bitu konfiguracyjnego (fuses) CKDIV8. Wskazane jest również włączenie modułu nadzorującego napięcie zasilające (BOD), BODLEVEL=4,3V. W celu sprawdzenia układu do gniazd należy podłączyć: J1–klawiaturę, J3–słuchawki, J2–napięcie zasilania według **rysunku 7**. Podczas naciskania klawiszy klawiatury obie LED powinny zachowywać się zgodnie z opisem w poprzednim śródtytule, a w słuchawkach powinien być słyszalny generowany kod. Wciskając klawisze F5...F8, należy sprawdzić działanie regulacji tempa „nadawania” i częstotliwości tonu. Następnie ustawienia zapisać F2, po ich zmianie wczytać, naciskając F3. Po odłączeniu i ponownym włączeniu zasilania ustawieniami obowiązującymi powinny być te ostatnio zapisane. Sprawdzenie działania wyjścia klucza polega na dołączeniu LED z szeregowym rezystorem (1kΩ) między wyjście KEY a VCC. Jeżeli dołączona LED „zachowuje” się jak LED D2 układ można uznać za uruchomiony. Układ zasilany napięciem 12V pobiera około 14mA (bez dołączonej klawiatury). Pobór prądu przez dołączoną klawiaturę może zawierać się w przedziale od kilku do kilkudziesięciu mA w zależności od jej modelu. Wymiary PCB umożliwiają montaż układu w duraluminiowej rurze o przekroju kwadratowym a = 30mm i grubości ścianek 1,5...2mm, co ilustruje **rysunek 8**. Oprócz przycięcia kształtownika do długości 57mm wymagane jest wywiercenie dwóch otworów (d = 3mm) dla LED. Obróbka mechaniczna dostępnych w handlu zaślepek z tworzyw sztucznych polega wycięciu, wywierceniu otworów na gniazda, wycięciu szczelin utrzymujących PCB w odległości uniemożliwiającej zwarcie z metalową rurą, co nie powinno sprawić większych trudności Szanownym Czytelnikom.

Wykaz elementów

Rezystory:

R4	100Ω
R5	240Ω
R3, R7	560Ω
R1, R2	4,7kΩ
R6, R8	10kΩ

Kondensatory:

C1	220u/25V
C2, C3, C6	100n ceramiczny
C4, C5	100u/16V

Półprzewodniki:

D1	1N4001
D2	LED R 3mm
D3	LED G 3mm
T1	BC548
U1	LM7805
U2	ATtiny13A

Inne:

J1	DIN5F lub MDIN6
J2, J3	Jack 3,5mm

Komplet podzespołów z płytką jest dostępny w sieci handlowej AVT jako kit szkolny AVT-3121.

Cyprian Kamil Kowalski
c4v2@o2.pl