Asynchroniczny konwerter RS232<->Midi

AVT-842

W artykule prezentujemy układ, który umożliwia współpracę standardowych interfejsów RS232 i MIDI.

Nie wszystkie zagadnienia, które mogłyby być proste, takimi właśnie są. Tak ma się sprawa z interfejsem MIDI i interfejsem szeregowym w PC. Mimo że oba interfejsy są szeregowe, to nie można w prosty sposób przekazywać informacji z jednego interfejsu do drugiego. Nie chodzi tu o różnice parametrów elektrycznych charakteryzujących oba standardy.

Podstawowy problem tkwi w szybkości transmisji, z jaką MIDI przekazuje informację (31250 bodów). Jest ona niemożliwa do uzyskania w standardzie RS232 PC-ta. Najbliższe wartości to 28800 lub 38400 bodów.

Pragnę w niniejszym projekcie przedstawić dwukierunkowy konwerter szybkości transmisji szeregowej, umożliwiający podłączenie urządzenia MIDI do PC-ta poprzez port szeregowy.

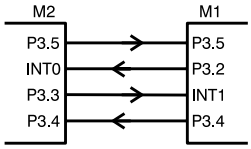
Oprogramowanie

W Internecie na stronie *http://www.cistron.nl/~nctnico/midi.htm* opublikowano projekt o nazwie „Ultra cheap MIDI interface", w którym, aby uzyskać pożądaną prędkość transmisji szeregowej, zaproponowano zmianę częstotliwości pracy generatora taktującego układ portu szeregowego na dodatkowo zainstalowanej karcie COM. Rozwiązanie dosyć proste, ale nie każdy może sobie na to pozwolić - duży problem będą mieli np. właściciele laptopów.

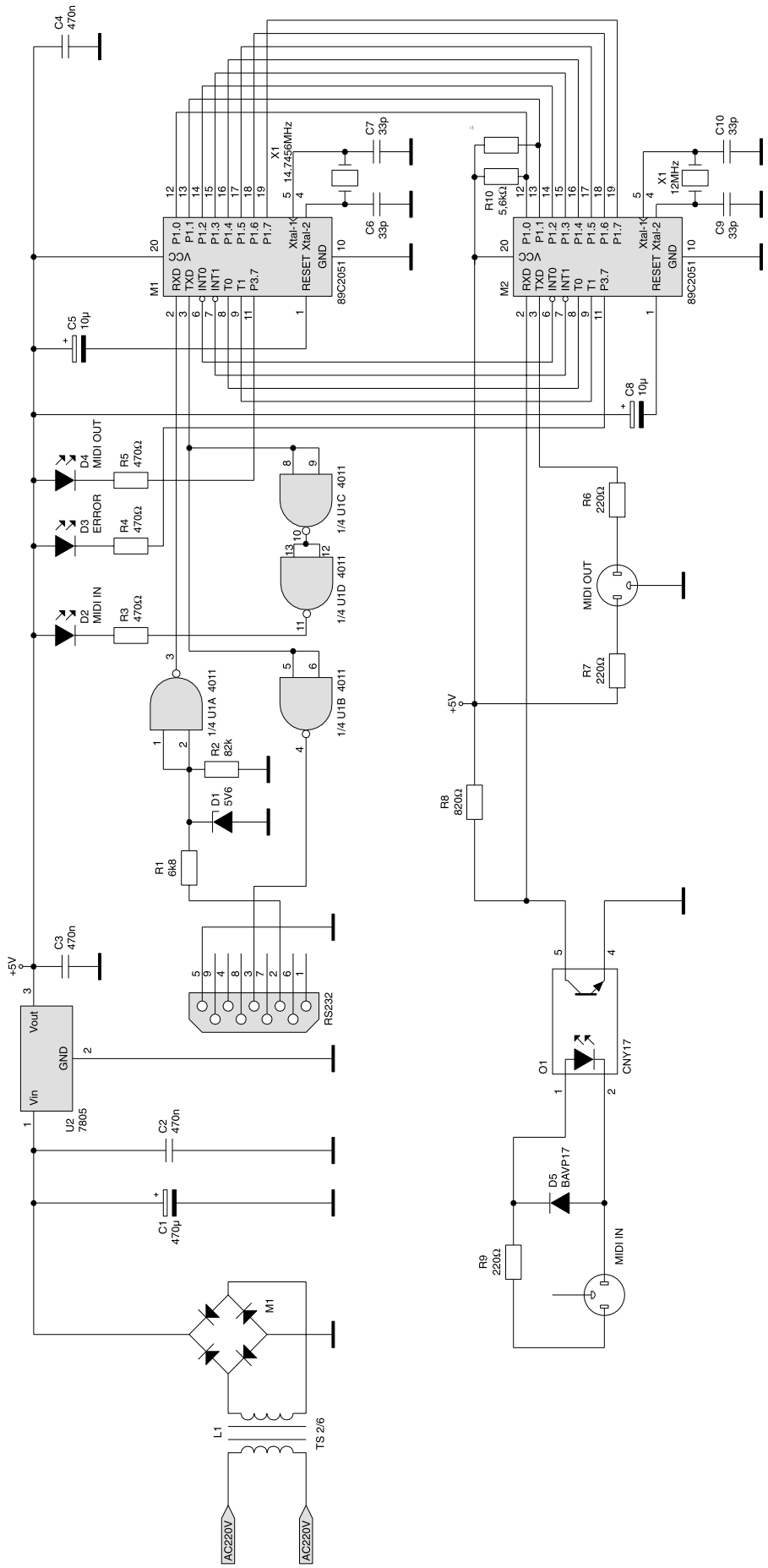
Wraz z projektem publikowany jest programowy driver, pracujący pod Windows 3.xx i Windows 95/98, który przesyła informacje MIDI z portu i do portu szeregowego z prędkością 38400 bodów. Z powyższego projektu wykorzystamy tylko driver. Funkcję konwertera szybkości transmisji szeregowej będą pełnić dwa mikroprocesory jednoukładowe 89C2051 wyposażone w nadajnik i odbiornik do transmisji szeregowej.

W układzie zamiast wymaganego (dla zapewnienia standardu RS232) układu MAX232 zastosowałem bramki CMOS 4011. To rozwiązanie nie tylko upraszcza układ i obniża koszt, ale jest dopuszczalne we wszystkich obecnie stosowanych komputerach. Prawie wszystkie modele powyżej AT akceptują na wejściu RS232 sygnały TTL. Ta właściwość pozwala nam na dokonanie uproszczenia.

Nie będę w niniejszym artykule zamieszczał szczegółów dotyczących MIDI, ponieważ już były publikowane w EP, a zainteresowanym polecam obejrzenie internetowej strony: *http://godot.tuniv.szczecin.pl/~mjaskula/tomi/music.html#midi*.



Rys. 1. Sposób komunikowania się procesorów w urządzeniu.



Rys. 2. Schemat elektryczny konwertera.

Konwersja szybkości transmisji z niższej na wyższą nie przedstawia problemu. Zupełnie inaczej jest, gdy chcemy przejść z wyższej na niższą, ponieważ pojawia się nadmiar danych. Aby nie utracić informacji, należy zastosować bufor. Informacje MIDI nie są przesyłane ciągle, ale raczej cyklicznie i dlatego nie jest wymagane stosowanie bufora o dużej pojemności. W zupełności wystarcza wewnętrzna pamięć mikroprocesora (ok. 100 bajtów).

Trochę inaczej może być w przypadku transmisji tzw. „System Exclusive“, kiedy to przesyłane są dane w trybie ciągłym. Nie mogę zagwarantować, że podczas takiej transmisji z komputera do każdego urządzenia MIDI zastosowany bufor będzie wystarczający.

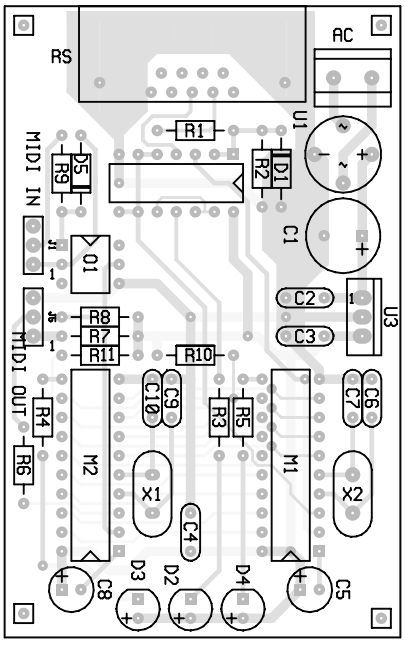
Komunikacja mikroprocesorów

Port P1 mikroprocesorów jest używany do dwukierunkowego przesyłania danych, zaś linie P3.2, P3.3, P3.4, P3.5 stanowią magistralę sterującą (**rys. 1**). Komunikacja między procesorami odbywa się w trybie hand-shake. Mikroprocesor, który odebrał bajt z portu szeregowego, wpisuje go do portu P1, a następnie generuje przerwanie w drugim mikroprocesorze poprzez podanie stanu niskiego na wejściu INT0 (w M1) lub INT1 (w M2). Następnie oczekuje potwierdzenia przez niski poziom sygnału na wyjściach P3.4 (M2) lub P3.5 (M1), że bajt został odczytany.

Jedno z przerwań musi mieć wyższy priorytet, gdyż dwa procesory mogłyby zgłosić żądanie przerwania w tym samym czasie. Nastąpiłoby wtedy nieskończenie długie oczekiwanie każdego z nich na potwierdzenie otrzymania bajtu. Priorytet ma INT0 w M1, które jest wyzwalane ujemnym zboczem sygnału z wyjścia P3.2 mikroprocesora M2.

Opis programu

Wysłanie bajtu przez M1 rozpoczyna w M2 obsługę procedury przerwania INT0, która wpisuje wartość otrzymanego bajtu do bufora okrężnego procedurą *put* (**list. 1**). Następnie, jeżeli nie jest ustawiony bit *busy*, wywo-



Rys. 3. Rozmieszczenie elementów na płytce drukowanej.

ływana jest procedura *send*, która inicjuje wysyłanie bajtu z bufora okrężnego do portu szeregowego MIDI. Bufor okrężny zrealizowany jest w obszarze pamięci określonym przez stałe *wskp*, *wske*. Zmienna *pocz* jest wskaźnikiem określającym adres, pod który zostanie wpisany następny otrzymany bajt z M1. Zmienna *kon* wskazuje bajt, który jako pierwszy ma być wysłany przez port szeregowy M2. Bufor jest pusty, gdy oba wskaźniki są sobie równe.

Procedura *put* wpisuje kolejny bajt do bufora oraz zwiększa wartość zmiennej *pocz*. Jeżeli wartość wskaźnika *pocz* „dogoni" wartość *kon*, to jest ustawiany bit *full*, który oznacza zapełnienie bufora. Próba kolejnego wpisania bajtu do bufora jest sygnalizowana świeceniem się diody „ERROR". Jest to informacja, że nastąpiło przepełnienie bufora i utrata danych. Skasowanie świecenia diody następuje po wyzerowaniu mikroprocesora.

Procedura *send* najpierw sprawdza, czy nie jest ustawiony bit *emp*. Jeżeli nie, to z bufora jest pobierany bajt wskazany przez *kon* i wpisywany do rejestru *SBUF* portu szeregowego inicjując transmisję MIDI. Następnie zwiększana jest wartość wskaźnika *kon* i ustawiany bit *busy*, informujący o transmisji szeregowej w toku. Jeżeli bufor jest pusty, to ustawiany jest bit *emp*.

Kiedy cały bajt zostanie wysłany, mikroprocesor wywołuje procedurę przerwania portu szeregowego z ustawionym bitem *ti*. Procedura ta w przypadku gdy bufor

List. 1.

```
;RS232-MIDI-M1-strona RS
$MOD51
;Podlaczenia
;M1 M2
;p1.0        p1.0
;p1.1        p1.1
;p1.2        p1.2
;p1.3        p1.3
;p1.4        p1.4
;p1.5        p1.5
;p1.6        p1.6
;p3.2        p3.2
;p3.3        p3.3
;p3.4        p3.4
;p3.5        p3.5
;M1 p3.0     RS232 IN
;M1 p3.1     RS232 OUT
;M2 p3.0     MIDI IN
;M2 p3.1     MIDI OUT
;M2 p3.7     LED ERROR

;adresy zmiennych w M2
pocz         EQU 021h ;wskaznik poczatka bufora
kon          EQU 022h ;wskaznik konca bufora
wskp         EQU 023h ;adres poczatka bufora
wske         EQU 080h ;adres konca bufora+1

;20h - bajt pomocniczy adresowany bitowo
full         EQU 000h ;bufor pelny
emp          EQU 001h ;bufor pusty
busy         EQU 002h ;RS w toku


;*********************** poczatek programu ***********************
program:
    org 000h          ;RESET
    jmp inic
    org 003h          ;INT0
    jmp inte
    org 023h          ;MIDI
    jmp sint

    ;reset procedure

inic:       ;inicjalizacja
    ;konfiguracja MIDI
    mov scon,#01010000b         ;set-RS 1st mode, RXD activation
    mov tmod,#00100010b         ;t1->M1,M0=10-tryb2, t0-M1,M0=01>tryb2
                                ;gate1,C/T1,M1,M0,gate0,C/T0,M1,M0
    mov pcon,#10000000b         ;transmision rate smod=1

    mov th1,#254     ;f=(fosc*2^smod)/(384*(256-th1)) ==31250
                                ;fosc=12MHz
    mov tl1,th1
    mov tcon,#01000000b         ;T1-on  tf1,tr1,tf0,tr0,ie1,it1,ie0,it0
    mov IP,#00000000b;priorytet -PT2,PS,PT1,PX1,PT0,PX0
    mov IE,#10010001b;maska EA,EAD,ET2,ES,ET1,EX1,ET0,EX0
    clr busy
    setb emp
    clr full
    mov pocz,#wskp
    mov kon,#wskp

loop:
    nop                         ;petla glowna
    nop
    nop
    nop
    nop
    sjmp    loop

;************Bufor okrezny************************************
;kon-adres nast.bajtu do wyslania
;pocz-tutaj wpisz otrzymany bajt

put:                ;wpisuje bajt do bufora
    jnb full,pu0
    clr p3.7        ;ERROR LED
    ret
pu0: mov r0,pocz
    mov @r0,a       ;wpisz do bufora
    mov a,pocz
    inc a
    cjne a,#wske,pu1
    mov a,#wskp
pu1: cjne   a,kon,pu2
    setb full       ;bufor pelny
    ret
pu2: mov pocz,a     ;uaktualnij pocz
    clr emp         ;bufor nie pusty
    ret

send:               ;wysyla bajt do MIDI
    jnb emp,se1     ;zakoncz-bufor pusty
    ret
se1: mov r0,kon
    mov SBUF,@r0    ;wyslij bajt
    setb busy       ;wysylanie w toku
    clr full
    mov a,kon
    inc a
    cjne a,#wske,se2
    mov a,#wskp
se2: mov kon,a
    cjne a,pocz,se3
    setb emp        ;bufor pusty
se3: ret

sint:               ;przerwanie  MIDI
    jb ri,sin1      ;bajt wyslano do MIDI
    clr ti          ;clear int poiter
    jb emp,sine
    acall send      ;wyslij nast.bajt z bufora
    reti
sine: clr busy      ;wyslano wszystkie bajty
    reti
sin1:               ;bajt otrzymano z MIDI
    jnb p3.4,sin1   ;czekaj az wysle poprzedni
    mov p1,SBUF
    clr ri
    clr p3.3        ;int1 in M1
sin2: jb p3.4,sin2  ;potwierdzenie otrzymania bajtu
    setb p3.3       ;bajt wyslano do M1
    reti


inte:               ;przerwanie INT0
    mov p1,#255
    mov a,p1
    clr p3.5
    acall put
    jb busy,inte1
    acall send
inte1: setb p3.5
    reti

end
```

List. 2.

```
;RS232-MIDI-M1-strona RS
$MOD51
;Podlaczenia
;M1 M2
;p1.0       p1.0
;p1.1       p1.1
;p1.2       p1.2
;p1.3       p1.3
;p1.4       p1.4
;p1.5       p1.5
;p1.6       p1.6
;p3.2       p3.2
;p3.3       p3.3
;p3.4       p3.4
;p3.5       p3.5
;M1 p3.0    RS232 IN
;M1 p3.1    RS232 OUT

;***************** poczatek programu ***********************
program:
    org 000h          ;RESET
    jmp inic
    org 013h          ;INT1
    jmp inte
    org 023h          ;MIDI
    jmp sint

    ;reset procedure

inic:       ;inicjalizacja
    ;konfiguracja MIDI
    mov scon,#01010000b         ;set-RS 1st mode, RXD activation
    mov tmod,#00100010b         ;t1->M1,M0=10-tryb2, t0-M1,M0=01>tryb2
                                ;gate1,C/T1,M1,M0,gate0,C/T0,M1,M0
    mov pcon,#10000000b         ;transmision rate smod=1

    mov th1,#254     ;f=(fosc*2^smod)/(384*(256-th1)) ==38400
                                ;fosc=14.7456MHz
    mov tl1,th1
    mov tcon,#01000100b         ;T1-on  tf1,tr1,tf0,tr0,ie1,it1,ie0,it0
    mov IP,#00000100b           ;priorytet -PT2,PS,PT1,PX1,PT0,PX0
    mov IE,#10010100b           ;maska EA,EAD,ET2,ES,ET1,EX1,ET0,EX0
    mov r1,#00

loop: nop           ;petla glowna
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    inc r1
    cjne r1,#00,loop1
    setb p3.7
loop1: sjmp loop

sint:
    jb ri,sin1      ;bajt wyslano do RS
    clr ti          ;clear int poiter
    setb p3.4
    reti
sin1:               ;bajt otrzymano z RS
    jnb p3.5,sin1   ;czekaj az wysle poprzedni bajt
    movc B,SBUF
    clr ri
    mov p1,B
    clr p3.7        ;LED control
    mov r1,#01
    clr p3.2        ;int0 in M2
sin2: jb p3.5,sin2  ;potwierdzenie otrzymania bajtu
    setb p3.2                   ;bajt wyslano do M2
    mov B,#255
    mov p1,B
    reti


inte:
    mov p1,#255
    mov SBUF,p1
    clr p3.4        ;odebrano bajt
    mov p1,B
    reti

end
```

jest pusty zeruje bit *busy*, a w innym przypadku inicjuje wysyłanie kolejnego bajtu wywołując procedurę *send*.

Pojawienie się bajtu danych w M1 jest sygnalizowane świeceniem diody sterowanej z wyjścia P3.7. Dioda świeci przez około 3ms, a czas podtrzymania naliczany jest w pętli głównej programu.

Częstotliwość transmisji danych portu szeregowego mikroprocesora zależy od częstotliwości pracy jego oscylatora. W obu mikroprocesorach port szeregowy pracuje w trybie drugim (rejestr *tmod*), a częstotliwość w tym trybie możemy obliczyć ze wzoru:

$$f_{tr} = (f_{osc} * 2^{smod})/(384*(256-t_{h1}))$$

gdzie:

f_{tr} - szybkość transmisji [bodów/s]

f_{osc} - częstotliwość kwarcu

smod - bit w rejestrze *pcon*

t_{h1} - bardziej znaczący bajt zegara T1

W mikroprocesorze M1 przyjąłem f_{osc}=14,756MHz i t_{h1}=254, zaś w M2 f_{osc}=12MHz i t_{h1}=254.

Uruchamianie

Montaż elektryczny nie wymaga szczegółowego omówienia. Schemat montażowy znajduje się na **rys. 3**.

Najpierw należy zainstalować driver w PC-cie. W Windows 95 w Panelu Sterowania wybieramy „Dodaj nowy sprzęt“, a następnie ręcznie (opcja *Nie*) wybieramy *Kontrolery video, dźwięku i gier* i *Z dysku* instalujemy driver. Program instalujący zapyta jeszcze o port COM. W Windows 3.xx w Panelu Sterowania wybieramy ikonę *Drivers*. Naciskamy na przycisk *Add* i wybieramy sterowniki nie będące na liście. Wpisujemy ścieżkę dostępu, skąd program instaluje driver. Wszelkich późniejszych zmian możemy dokonać w programie *system.ini* w sekcji [cbxt3.drv], która po zainstalowaniu domyślnie przyjmuje następujące wartości:

SysExWait=40
comport=1
Buffersize=10240
MIDIINPersistence=50

SysExWait - czas w [ms] oczekiwania systemu na odebranie całej paczki danych *SystemExclusive*, po którym nastąpi przesłanie otrzymanych informacji do programu MIDI.

comport - numer portu COM.

Buffersize - rozmiar bufora danych potrzebnego do przesyłania paczek danych *SystemExclusive*. Może przyjmować wartości od 1 do 65535.

MIDIINPersistence - określa liczbę bajtów, które driver odczyta zanim odda kontrolę Systemowi Windows. Wartość ta nie dotyczy przesyłania paczek danych *SystemExclusive*.

W każdym programie muzycznym, przy wyborze Urządzenia MIDI, driver ten pojawi się jako: „Yamaha CBX-T3 1.3“, oczywiście wtedy, gdy występuje zadeklarowany port COM w komputerze i jest nie wykorzystywany w danej chwili przez inne urządzenie.

Aby uruchomić sterownik potrzebny będzie instrument MIDI, który jest w stanie generować i czytać komunikaty MIDI. Zmontowany układ nie wymaga żadnych regulacji, więc jeżeli nie popełniliśmy błędu, powinien od razu działać poprawnie. Podajemy mu zasilanie i podłączamy między instrument a port szeregowy. W komputerze uruchamiamy program muzyczny i ustawiamy zainstalowany sterownik w *MIDI Setup* jako wejściowy i wyjściowy.

Większość programów muzycznych (Cubase, CakeWalk) ma kontrolki sygnalizujące otrzymywanie i wysyłanie komunikatów MIDI. Poczas naciskania na klawiaturę instrumentu, w czasie właściwej pracy sterownika kontrolka powinna się zaświecić. Następnie sprawdzamy, czy podłączony instrument reaguje na rozkazy wysyłane z programu muzycznego. Pamiętajmy o ustawieniu takiego samego kanału MIDI.

Jeżeli nie następuje transmisja, to w lokalizacji błędu pomogą nam diody sterownika. Migająca dioda D4 informuje, że M1 poprawnie odczytuje informacje z komputera. Dioda D2 zapala się, kiedy przekazywane są komunikaty MIDI z dołączonego instrumentu, ale miganie nie gwarantuje poprawności odczytu informacji przez M2. Kiedy będą problemy w przesyłaniu informacji w tym kierunku, to proponuję zmienić program w mikroprocesorze M2 tak, aby bit portu P3.7 ustawiany był jak w M1. Wtedy zapalająca się dioda upewni nas, że mikroprocesor poprawnie odczytuje informacje z instrumentu. Nie musimy mieć oscyloskopu, aby stwierdzić, czy następuje transmisja - wystarczy zwykły miernik, najlepiej cyfrowy ze względu na wysoką impedancje wejściową. Napięcie przy występującej transmisji nieznacznie różni się od napięcia w stanie statycznym. W ten sposób możemy określić przyczynę błędnej pracy sterownika sprawdzając kolejno miejsca w torze transmisji sygnału. Mierząc napięcie na nóżkach XTAL2 mikroprocesorów możemy także sprawdzić poprawność działania oscylatorów. Napięcie to nie powinno być bliskie zasilania lub zera. Kiedy następują przekłamania w informacjach dochodzących z instrumentu do komputera - co może się objawiać przez np. odczytywanie nie tych dźwięków przyporządkowanych klawiszom, które naciskamy na klawiaturze instrumentu - można próbować dobrać inną wartość rezystora R8 (z zakresu 470Ω..2kΩ), podciągającego napięcie na transoptorze. Wartość, która jest dobrana (820Ω) zapewnia poprawną transmisję dla większości transoptorów typu CNY17. Transoptor ten pracuje na granicy swojego pasma i najlepszy byłby transoptor PC900 firmy Sharp, który jest stosowany w większości profesjonalnych urządzeń MIDI.

Układ zmontowałem na dwóch płytkach, osobno niestabilizowany, popularny zasilacz sieciowy z wyjściem 9..12[V] DC i osobno stabilizator z resztą układu.

Piotr Swadźba
pswadzba@friko6.onet.pl

WYKAZ ELEMENTÓW

Rezystory
R1: 6,8kΩ
R2: 82kΩ
R3, R4, R5: 470Ω
R6, R7, R9: 220Ω
R8: 820Ω
R10, R11: 5,6kΩ

Kondensatory
C1: 470µF/25V
C2, C3, C4: 470nF
C5, C8: 10µF/10V
C6, C7, C9, C10: 33pF

Półprzewodniki
M1, M2: AT89C2051 zaprogramowane
U1: 4011
U2: 7805
D1: dioda Zenera 5,1V/100mW
D2, D3, D4: diody LED
D5: BAVP17 (1N4148)
Q1: CNY17
M1: mostek 1A/50V

Różne
L1: TS2/6
RS232: gniazdo do druku DB9M
X1: 14,7456MHz
X2: 12MHz