Czytnik-programator kart chipowych, część 1

PROJEKT Z OKŁADKI

kit AVT-468

Artykuł prezentujący konstrukcję czytnika-programatora kart chipowych będzie składał się z dwóch części. W pierwszej z nich przedstawiamy zasadę działania kart chipowych, sposób ich programowania oraz konstrukcję elektryczną urządzenia. Za miesiąc przybliżymy sposób sterowania pracą programatora-czytnika oraz bardzo efektowną, przykładową aplikację.

Tematyka kart chipowych stała się w ostatnich miesiącach bardzo modna zarówno wśród elektroników, jak i ogromnej rzeszy „szarych“ ludzi. Jest to efekt coraz szerszego ich stosowania - każdy telefon komórkowy sieci GSM jest wyposażony w taką kartę, coraz większa liczba automatów telefonicznych „woli“ pobierać opłaty za rozmowy z kart chipowych, a nie magnetycznych. Także posiadacze kont bankowych zostaną wkrótce wyposażeni w takie karty. Z wyglądu karta chipowa przypomina zwykłą kartę kredytową lub bankomatową, z tą różnicą, że na jej jednej stronie znajdują się efektownie wyglądające złocone pola stykowe.

Cóż to więc, tak naprawdę, jest ta „karta chipowa“? Wbrew pozorom nie jest łatwo odpowiedzieć na to pytanie. Najprostszą jest odpowiedź, że jest to karta wyposażona w układ scalony, który najczęściej spełnia rolę elementu pamięciowego. Rzadko jednak spotyka się karty integrujące w wewnętrznym chipie samą tylko pamięć. Zazwyczaj karty chipowe są wyposażone w mniej lub bardziej zaawansowane mechanizmy zabezpieczające ich zawartość. Najprostsze takie rozwiązania polegają na zastosowaniu haseł dostępu do poszczególnych bloków logicznych (partycji) pamięci, nieco bardziej zaawansowane wykorzystują liczniki liczby błędnie podanych haseł, a zawartość najlepiej zabezpieczonych kart jest chroniona przez mikrokontrolery zintegrowane z kryptokonrolerami, których pokonanie jest niezwykle trudne.

Pomimo opracowania wielu niezłych standardów opisujących strukturę fizyczną i sposób pracy interfejsu łączącego wnętrze karty ze światem, panuje w tej dziedzinie dość duży „bałagan“, który w znacznym stopniu utrudnia zaprojektowanie uniwersalnego programatora do wszystkich typów kart chipowych. Dodatkową, naprawdę niebagatelną, trudnością jest utajnienie fragmentów dokumentacji przez producentów kart.

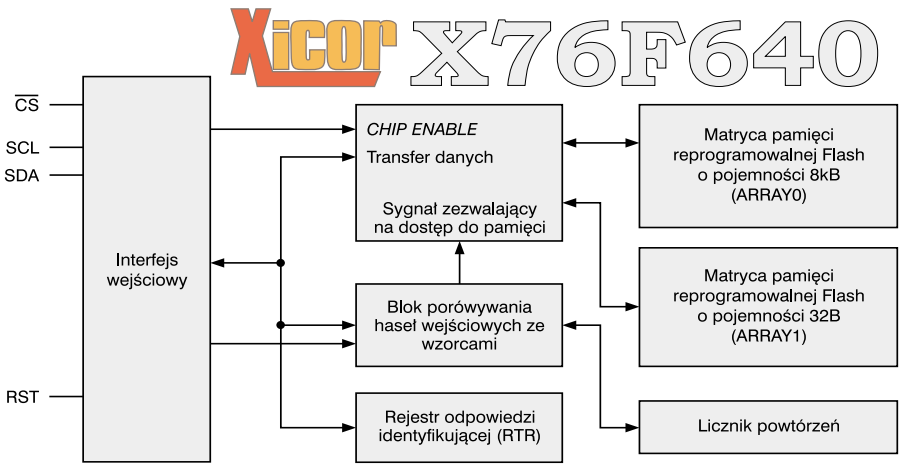
Dlaczego to robią? Produkcja kart to dobry biznes, ale ich „rozkuwanie“ jest jeszcze lepszym.

O czym w artykule nie będę pisał i dlaczego

Od razu się zastrzegam - urządzenie prezentowane w artykule nie zostało opracowane z myślą o ładowaniu „lewych“ impulsów do kart telefonicznych, czy też zapisywaniu sobie do karty kredytowej nieograniczonych limitów pieniędzy do wydania. Co więcej - zapewniam Was, że podejmowanie takich prób zdecydowanie nie ma sensu. Zastosowane we współczesnych kartach zabezpieczenia (biorąc dodatkowo pod uwagę trudności, a w zasadzie niemożność, zdobycia kompletnej dokumentacji) zapewniają bardzo duże bezpieczeństwo informacjom tam

Podstawowe parametry i możliwości programatora:

- ✓ programuje i odczytuje zawartość pamięci kart chipowych X76F100 i X76F640;
- ✓ sterowany jest przez dowolny program terminalowy;
- ✓ wymiana danych odbywa się poprzez port szeregowy RS232 (19200/8N1 lub 19200/8N2);
- ✓ rozmiar bufora danych: 32B;
- ✓ czas programowania sektora pamięci (32 bajty): poniżej 11ms;
- ✓ zasilanie: 8..12VDC lub AC, pobór prądu ok. 20mA.



Rys. 1. Schemat blokowy karty chipowej X76F640.

zapisanym. Co prawda karty telefoniczne nie zawierają w sobie mikroprocesora z wyszukanymi algorytmami zabezpieczającymi, ale zastosowane w nich bardzo proste zabezpieczenia gwarantują ich kompletne skasowanie (w przypadku struktur EEPROM) - otrzymujemy w ten sposób bardzo drogą pustą kartę, którą można później wykorzystać we własnej aplikacji (jeżeli znany jest producent karty) lub wyzerowanie dostępnych jednostek - a można to zrobić bez trudu (w przypadku struktur EPROM).

Dla porządku wyjaśnię, aby zapobiec traceniu czasu na łamanie „szyfrów" Telekomunikacji Polskiej lub banków, dlaczego przełamanie nawet najprostszych zabezpieczeń jest mało prawdopodobne.

Karty, których struktura jest zgodna ze standardem przemysłowym (np. ODS, Gemplus, S&O, ORGA), wymagają do poprawnego wykorzystania wpisania do matrycy pamięciowej kilku znaków charakteryzujących ich wydawcę (np. bank lub firmę telekomunikacyjną). Znaki te są zapamiętywane zazwyczaj w matrycy EPROM, do której skasowania niezbędne jest zastosowanie promieniowania ultrafioletowego. Matryca EPROM jest zabezpieczona przed niepowołanym odczytem. Zastosowanie takiej procedury modyfikacji karty wymagałoby jej bardzo precyzyjnego demontażu (struktury są zazwyczaj zalane masą syntetyczną nie przepuszczającą światła).

Po serii prób (przecież nie mogę się przyznać w EP, że nie potrafiłem przełamać zabezpieczeń w kartach Centertela!) doszedłem do wniosku, że w czasie zmarnowanym na śledzenie zachowania karty po kolejnych próbach dostępu, więcej można zarobić sprzedając te karty.

Tak więc, z jednej strony kusi eksperymentowanie, z drugiej strony nie bardzo się to opłaca! Wyciągnięcie wniosków pozostawiam Wam.

Co wobec tego?

Wszystkie argumenty zniechęcające Was do marnowania czasu, które przedstawiłem powyżej, nie mają na celu zamknięcie tematu i zakończenie artykułu! Są bowiem dostępne na rynku bardzo interesujące karty chipowe, które można samodzielnie zastosować w ciekawych aplikacjach. Co więcej - jest do nich dostępna niezła dokumentacja, a cena samych kart nie powoduje udaru u potencjalnych klientów.

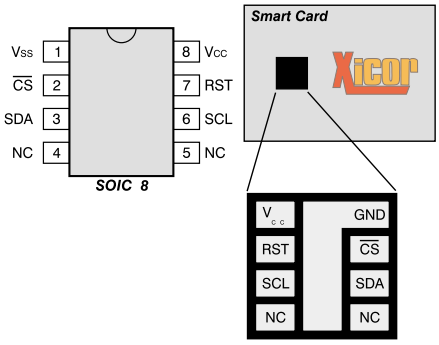
Schemat blokowy karty wykorzystanej w prezentowanym projekcie przedstawiono na **rys. 1**.

Na pierwszy rzut oka karta X76F640 to zwykła pamięć EEPROM z interfejsem I²C! Ale tylko na pozór! Układ X76F640 rzeczywiście integruje w swojej strukturze dwa niezależne bloki pamięci EEPROM (jeden o pojemności 8kB, drugi o pojemności 32B), a oprócz nich kilka modułów dodatkowych:

- Blok porównywania haseł wejściowych ze wzorcami. Odpowiada on za weryfikację haseł wpisywanych do układu przez zewnętrzny sterownik. Warto zwrócić uwagę, że dostęp do każdego z obszarów pamięciowych wymaga osobnego hasła (osobne dla zapisu/odczytu). Także inne polecenia, nie związane bezpośrednio z operacjami na matrycy pamięciowej, wymagają do uaktywnienia odpowiedniego hasła. W sumie dostępu do układu X76F640 chroni aż pięć haseł o długości 64 bitów każde.
- Licznik powtórzeń, który stanowi bardzo ważny element zabezpieczający zawartość pamięci. Jeżeli wystąpi kilka (osiem) nieuprawnionych (bez podania odpowiedniego hasła) prób dostępu do jakiejkolwiek funkcji pamięci, następuje automatyczne zerowanie obydwu matryc pamięciowych. Dzięki temu można z prawdopodobieństwem bliskim pewności założyć, że osoby niepowołane nie będą w stanie podejrzeć informacji zapisanych w matrycach pamięciowych.
- Rejestr odpowiedzi identyfikującej, który jest uzupełnieniem układu X76F640. Wbudowanie tego rejestru w strukturę karty gwarantuje spełnienie wymogów standardu ISO7816 (opisuje sposób przekazywania danych z i do karty). Rejestr ten jest faktycznie zaprogramowaną przez producenta „na sztywno" 32-bitową matrycą pamięciową, która pozwala jednoznacznie określić czytnikowi z jakim typem układu ma do czynienia.

W celu zachowania zgodności z jedynym liczącym się w świecie standardem mechanicznym, opisującym rozmieszczenie elementów stykowych w kartach chipowych, karta X76F640 ma wyprowadzenia jak na **rys. 2**. W pewnym uproszczeniu można przyjąć, że ich rozkład jest identyczny z zaleceniami standardu ISO7816.

Warto wspomnieć, że układ X76F640 występuje w kilku wer-



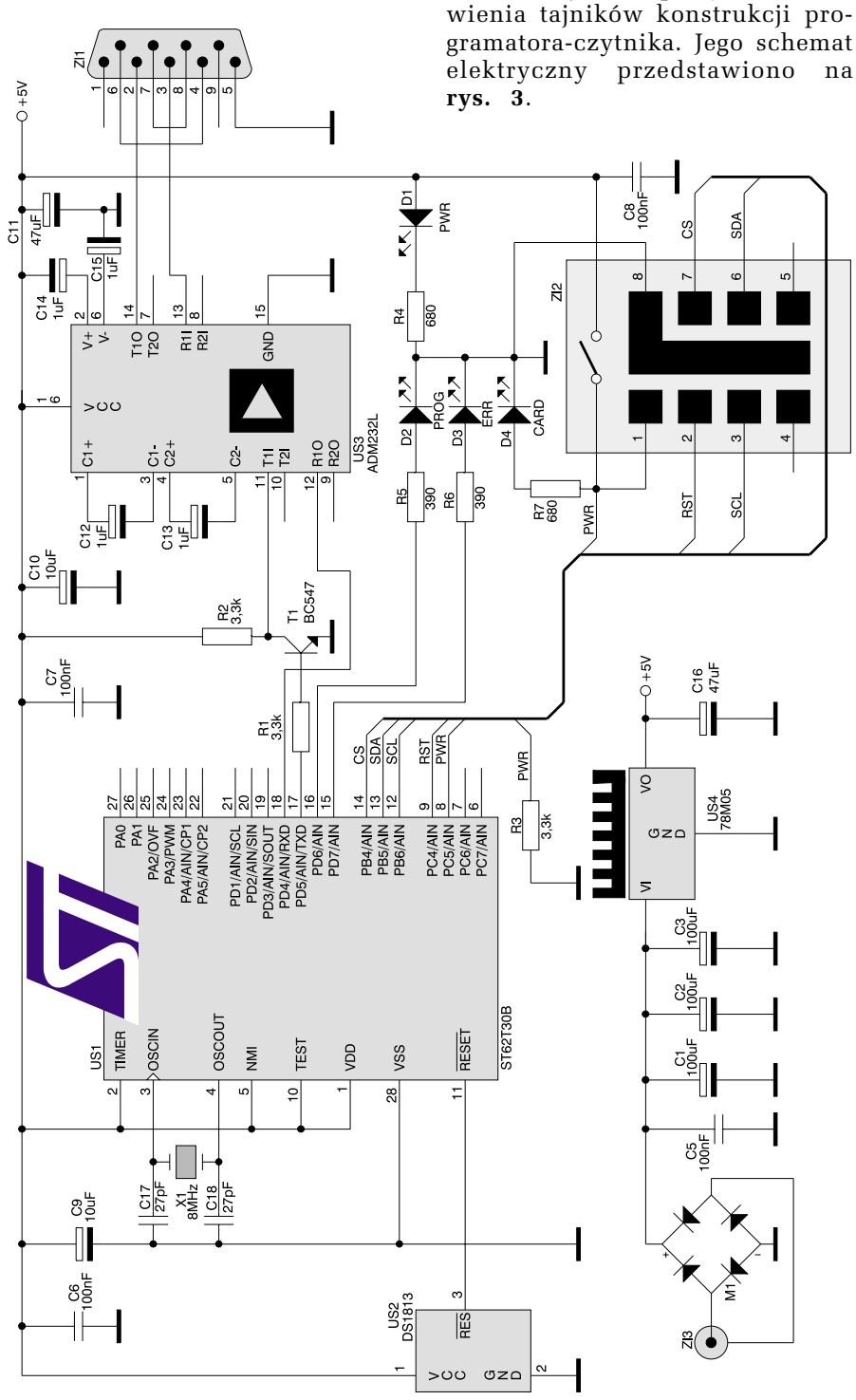
Rys. 2. Wyprowadzenia układu X76F640.

sjach obudów. W prezentowanym urządzeniu wykorzystywane będą układy w obudowach standardowych kart chipowych (pełne oznaczenie układu X76F640Y), ale dostępne są także układy w obudowach:
- SOIC8 (ozn. X76F640A);
- nieobudowanej struktury (ozn. X76F640H i W);
- 8-pinowego modułu, bez obudowy w postaci karty nośnej (ozn. X76F640X).

Xicor produkuje karty przystosowane do pracy w temperaturach standardowych (0..+70°C) oraz rozszerzonych (-20..+85°C), co jest cechowane literą E w oznaczeniu. Dostępne są także wersje pracujące z niskimi napięciami zasilania (2,7..3,6V). Są one oznaczone dodatkowym sufiksem „-2,7".

Opis urządzenia

Możemy teraz przejść do omówienia tajników konstrukcji programatora-czytnika. Jego schemat elektryczny przedstawiono na **rys. 3**.



Rys. 3. Schemat elektryczny urządzenia.

Jak widać konstrukcja elektryczna urządzenia jest stosunkowo prosta. „Sercem" programatora jest jeden z najnowszych mikrokontrolerów rodziny ST62, układ noszący oznaczenie ST62T30B (US1). Jest on wyposażony w dużą pamięć programu (8kB), wewnętrzną pamięć EEPROM, RAM oraz szereg interesujących peryferyjnych modułów wewnętrznych, spośród których w projekcie wykorzystano port komunikacji szeregowej UART, timer oraz watchdog.

Program wpisany do pamięci procesora odpowiada za poprawną pracę całego programatora, konfigurację portów, obsługę przerwań (wykorzystano dwa spośród dostępnych) itp. Na **list. 1** przedstawiono fragment tego programu, zawierający procedurę inicjalizacji rejestrów i urządzeń wewnętrznych procesora oraz dwie procedury przedstawiające sposób realizacji odczytu 32 bitów słowa *Response To Reset*, które umożliwia rozpoznanie typu karty włożonej do czytnika. Kolejne odczytane bajty słowa *RTR* są zwracane przez procedurę *read_byte* w akumulatorze, a procedura *info_a* (pominięto ją, ze względu na długość listingu) odpowiada za wysyłanie komunikatów diagnostycznych do komputera PC.

Ponieważ jednym z najważniejszych problemów na jakie napotykają konstruktorzy systemów mikroprocesorowych jest poprawne wyzerowanie procesora po włączeniu zasilania i blokadę jego pracy przy napięciu o nieprawidłowej wartości, w programatorze zastosowano scalony generator sygnału zerującego firmy Dallas, który nosi oznaczenie DS1813 (US2). Schemat blokowy przybliżający jego budowę wewnętrzną przedstawiono na **rys. 4**. Możliwości tego układu są większe od wymagań aplikacji - ponieważ nie jest wykorzystywany układ wspomagający zerowanie ręczne. Ze względu na niemal identyczną cenę układu DS1813 z układami DS1811 (i podobnymi) wybór padł na układ bardziej elastyczny.

W aplikacji wykorzystano tylko 8 linii I/O procesora. Zastosowanie stosunkowo dużego procesora z rodziny ST62 może wywoływać wobec tego pewne wątpliwości.

Listing 1.

```
;*************************************************************
;*              Czytnik kart chipowych z RS232
;*************************************************************
        .title "Chip_Card"
        .input "rejestry.a62"
        .vers "st6230"
        .romsize 8
        .pp_on
        .dp_on
        .w_on

rst     .equ 4                  ; portc
insert  .equ 5                  ; portc
cs      .equ 4                  ; portb
sda     .equ 5                  ; portb
scl     .equ 6                  ; portb
ledprog .equ 6                  ; portd
lederr  .equ 7                  ; portd


        .section 1
        .org 0h

;*************************************************************
;*      Inicjalizacja procesora
;*************************************************************
begin   ldi ddrd,11100000b      ; wyjscie dla UARTa i LEDow
        ldi ord,11100000b       ;
        ldi drd,00000000b

        ldi ddrc,00010000b      ; ustala wyjscie dla RST
        ldi orc,00010000b
        ldi drc,00000000b

        ldi ddrb,01010000b      ; wyjscia SCL, CS,
                                ; SDA na razie "wejscie"
        ldi orb,01010000b               ; zalezy od bitu PB5
        ldi drb,00010000b

        reti

        set 7,drd               ; gasi LEDa ERR
        set 6,drd               ; gasi LEDa PROG

        ldi ior,00010000b               ; wlacza przerwania
        ldi uartcr,00101001b    ; ustala, ze:
                                ; - szybkosc: 19200b/s (8n1 lub 2)
                                ; - aktywne przerwanie od RECEIVERa


;*************************************************************
;*      Odczytuje z karty RESPONSE TO RESET i wysyla do PC
;*              Wykorzystany bufor TEMP i V, X!
;*************************************************************
card    res ledprog,drd         ; opoznienie po wlaczeniu zasilania
        call wait
        ldi x,0                         ; zeruje bajt kontrolny czy
00/FF
        set ledprog,drd
        res cs,drb
        set rst,drc
        set scl,drb             ; impuls zegara inicjujacy R_T_R
        nop                     ; niezbedne opoznienie (katalog!)
        nop
        nop
        res scl,drb
        res rst,drc
        ldi v,4                         ; ilosc bajtow do odczytania
rd_rtr  call read_byte
        ld uartdr,a
        cpi a,0
        jrz dodaj
        cpi a,0ffh
        jrz dodaj
        jp card1
dodaj   inc x                   ; zwieksza stan licznika bajtow 00/FF
card1   jrr 6,uartcr,card1      ; wysyla odczytany bajt do PC
        dec v
        jrz card3
        jp rd_rtr
card3   set cs,drb
        ld a,x
        cpi a,4
        jrz card2
        jp card5
card2   ldi drwr,rtr_err.w
        ldi y,rtr_err.d
        call info_a
card6   res lederr,drd
        call wait
        set lederr,drd
        call wait
        jrs insert,drc,card6
card5   ret


;*************************************************************
;* Procedura INFO - wysyla 16 znakow komunikatu przez RS232
;*************************************************************
info_a  ******************


;*************************************************************
;*            Procedura odczytu bajtu R_T_R
;*          Odczytany bajt znajduje sie w A
;* Informacje odbierane sa w kolejnosci: BYTE0 LSB..MSB, BYTE1...
;*************************************************************
read_byte:
        res ledprog,drd
        clr a
        set scl,drb
        jrr sda,drb,b7_1        ; oznacza, ze odebrany bit jest rowny 1
        set 0,a
b7_1    res scl,drb
        set scl,drb
        jrr sda,drb,b6_1
        set 1,a
b6_1    res scl,drb
        set scl,drb
        jrr sda,drb,b5_1
        set 2,a
b5_1    res scl,drb
        set scl,drb
        jrr sda,drb,b4_1
        set 3,a
b4_1    res scl,drb
        set scl,drb
        jrr sda,drb,b3_1
        set 4,a
b3_1    res scl,drb
        set scl,drb
        jrr sda,drb,b2_1
        set 5,a
b2_1    res scl,drb
        set scl,drb
        jrr sda,drb,b1_1
        set 6,a
b1_1    res scl,drb
        set scl,drb
        jrr sda,drb,b0_1
        set 7,a
b0_1    res scl,drb
        set ledprog,drd
        ret

wait    ldi v,0ffh
12      ldi a,0ffh
11      dec a
        jrnz 11
        dec v
        jrnz 12
        ret

        .section 32
        .org 00h
        jp uart_int  ; skok do obslugi przerwania od REC UART

        .org 0eh
reset   jp begin
```

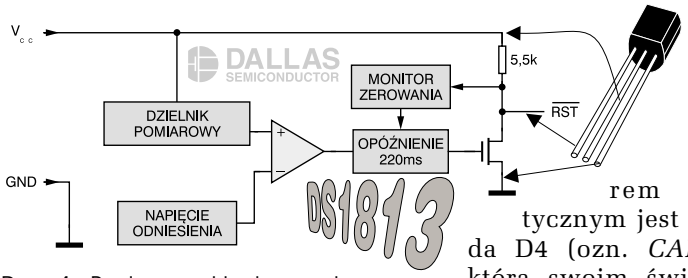
Wybór na ST62T30B padł z dwóch zasadniczych powodów.

1. Ma on dużą pamięć dla programu, co pozwoliło na dość swobodne (czytaj rozrzutne) gospodarowanie nią. Program sterujący po „doszlifowaniu“ zajmuje ok. 6,3kB, przy czym z pewnością dałoby się go jeszcze nieco zmniejszyć, ale nie w takim stopniu, aby zejść poniżej krytycznego progu 4kB (takie „kwanty“ pojemności pamięci dostępne są w rodzinie ST62).
2. Procesory ST62T30B mają wbudowany sprzętowy port szeregowy, dzięki czemu obsługa szeregowej transmisji danych jest prosta, niezawodna i możliwe jest wykorzystanie stosunkowo dużych szybkości (do 38,4kb/s), co jest niewykonalne w przypadku programowej emulacji intefejsu.

Tak więc zwyciężyła wygoda projektanta - co wziąwszy po uwagę cenę układu - jest wystarczającym uzasadnieniem w tym procesorze.

Układ US3 spełnia rolę konwertera napięciowego, pośredniczącego pomiędzy mikrokontrolerem ST62T30B i linią RS232. W strukturę tego układu wbudowano oprócz elementów tworzących interfejs także pompę ładunkową, dzięki której napięcie 5V jest przetwarzane do poziomu ±9..12V, co w zupełności spełnia wymagania stawiane przez standard RS232.

W egzemplarzu modelowym ADM232L, który charakteryzuje się wbudowanymi doskonałymi zabezpieczeniami antyprzepięciowymi, niewielkim poborem mocy, a do poprawnej pracy wystarczają mu cztery niewielkie kondensatory o pojemności ok. 1µF. Można oczywiście zastosować zamiast ADM232L dowolną inną wersję



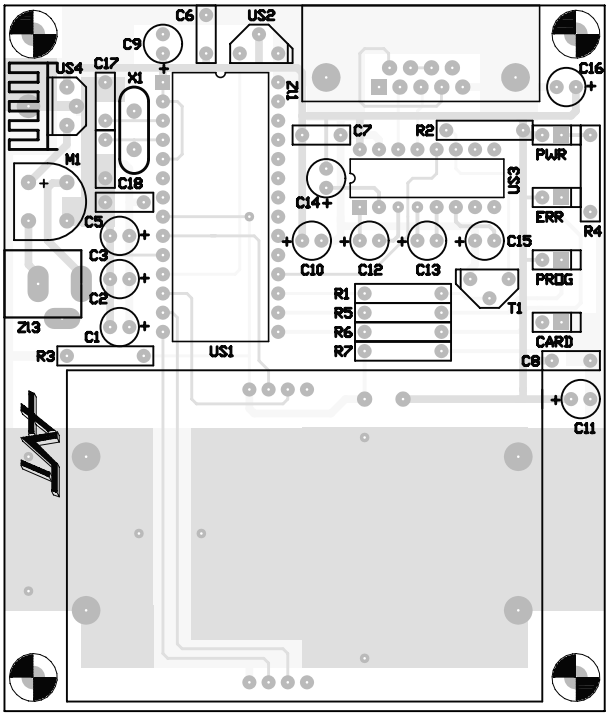
Rys. 4. Budowa układu zerującego DS1813.

tego popularnego układu, ale wymaga to zazwyczaj dobrania pojemności kondensatorów C12..15 (patrz uwagi w wykazie elementów).

Dość intrygująco (przyznacie wszyscy) wygląda na schemacie inwerter, wykonany w oparciu o tranzystor T1 i dwa rezystory: R1, R2. Zastosowanie tego skomplikowanego układu miało na celu zlikwidowanie niedoróbki (tak to niestety wygląda!) projektantów interfejsu UART w układzie ST62T30B. Niedoróbka ta polega na wysyłaniu na pin *TxD* informacji w postaci zanegowanej, co zdecydowanie uniemożliwia pracę interfejsu. Tak więc, po zastosowaniu inwertera przed układem AD232 problemy z poprawnością transmisji danych zniknęły. W egzemplarzu modelowym zastosowano pojedynczy inweter serii *TinyLogic*, ale ze względu na trudny montaż (SMD) i stosunkowo wysoką cenę układu zdecydowanie lepszym rozwiązaniem okazał się inwerter tranzystorowy.

Programator został wyposażony w cztery diody sygnalizacyjne LED, które w pewnym stopniu ułatwiają diagnostykę urządzenia. Dioda D1 (ozn. *PWR*) sygnalizuje świeceniem fakt dołączenia napięcia zasilającego do programatora. Zalecany dla niej kolor to zielony. Dioda D2 (ozn. *PROG*) informuje użytkownika, że procesor wymienia z kartą informację, w związku z czym nie wolno jej wyjmować z uchwytu Zl2. W egzemplarzu modelowym zastosowano diodę o żółtym kolorze świecenia. Diodę D3 (ozn. *ERR*) wykorzystano do sygnalizowania błędów występujących podczas pracy programatora. Najbardziej „logicznym" kolorem tej diody jest oczywiście czerwony. Ostatnim zastosowanym sygnalizatorem optycznym jest dioda D4 (ozn. *CARD*), która swoim świeceniem sygnalizuje włożenie karty (lub czegoś do niej mechanicznie podobnego) do uchwytu Zl2.

Pozostałe elementy urządzenia są dość typowe i nie będę ich szczegółowo omawiał. Wartko jedynie zwrócić uwagę na fakt zastosowania na wejściu zasilania mostka prostowniczego w układzie Graetza, dzięki czemu polaryzacja napięcia wejściowego może być dowolna, możliwe jest także zasilanie napięciem zmiennym.



Rys. 5. Rozmieszczenie elementów na płytce drukowanej.

Montaż i uruchomienie

Płytkę programatora zaprojektowano jako dwustronną z metalizowanymi otworami. Widok mozaiki ścieżek obydwu stron płytki przedstawiono na wkładce wewnątrz numeru, a schemat montażowy płytki na **rys. 5**.

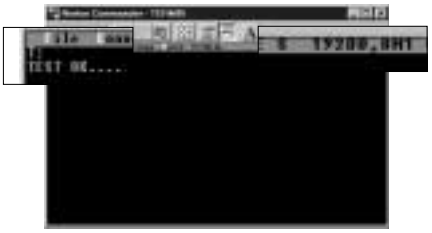
Montaż urządzenia nie ma specjalnych wymagań. Przed wlutowaniem w płytkę stabilizatora US4 należy przykręcić do niego radiator, a następnie przykręcić go do płytki drukowanej. Po wykonaniu tych czynności można przylutować końcówki stabilizatora do punktów lutowniczych. Bardzo ważne jest także - uwaga dla tych Czytelników, którzy nie zakupią zestawu - aby złącze Zl1 było żeńskie! Ze względów bezpieczeństwa warto zastosować pod układ US1 podstawkę.

Do uruchomienia układu potrzebny będzie kabel 1:1 zakończony złączami 9-stykowymi (z jednej strony męskim, z drugiej żeńskim) oraz zasilacz o napięciu wyjściowym 8..15VDC i wydajności prądowej min. 20mA.

Po włączeniu zasilania procesor wykonuje prostą procedurę testową, której wykonywanie jest sygnalizowane zewnętrznie zapaleniem na chwilę diod *ERR* i *PROG*, następnie zgaśnięciem diody *ERR* i po chwili *PROG*. Taka sekwencja oznacza, że inicjalizacja procesora przebiegła prawidłowo. Następnie należy podłączyć do programatora i komputera PC kabel RS232 i uruchomić dowolny program terminalowy. Doskonale do tego celu nadaje się Hyper Terminal (**rys. 6**), stanowiący standardowe wyposażenie Windows 95 lub TERM95 (**rys. 7**), który jest częścią składową Norton Commandera. Programy terminalowe mogą pracować w dowolnym trybie znakowym z wyłączonym echem lokalnym. Najlepszym rozwiązaniem jest wykorzystanie trybu ANSI lub HEX.

Parametry transmisji należy zadać następująco:

Rys. 6. Widok okna programu Hyper Terminal.



Rys. 8. Widok okna działającego programu TERM95.EXE.

- ramka 8-bitowa;
- szybkość transferu 19200b/s;
- bity stopu 1 lub 2;
- brak bitu parzystości.

W skrócie można je zapisać jako: 19200/8N1 lub 19200/8N2.

Po uruchomieniu i skonfigurowaniu programu terminalowego należy napisać na ekranie (co w praktyce oznacza wysłać do programatora) polecenie autotestu - *T:*. Jeżeli po kilku sekundach na ekranie terminala pojawi się odpowiedź „TEST OK....“ można przyjąć, że programator działa poprawnie. Jeżeli taki komunikat się nie pojawi należy dokładnie sprawdzić jakość montażu i zastosowanych elementów.

WYKAZ ELEMENTÓW

Rezystory
R1, R2, R3: 3,3kΩ
R4, R7: 680Ω
R5, R6: 390Ω

Kondensatory
C1, C2, C3: 100µF/25V
C5, C6, C7, C8: 100nF
C9, C10: 10µF/10V
C11, C16: 47µF/10V
C12, C13, C14, C15: 1µF/25V
C17, C18: 27pF

Półprzewodniki
D1, D2, D3, D4: LED
M1: mostek 500mA/50V
T1: BC547
US1: ST62T30B - zaprogramowany
US2: DS1813 (dowolna wersja w obudowie TO-92)
US3: ADM232L*
US4: 78M05 lub podobny w obudowie TO-220

Różne
X1: 8MHz - oscylator kwarcowy
Zl1: złącze kątowe, żeńskie DB9
Zl2: złącze z czujnikiem karty 7434L0825S01-08 firmy FCI lub LM08 RS-Components
Zl3: złącze zasilania
Radiator
Zasilacz 8..12VDC/100mA lub podobny
Kabel RS232/1:1 (złącza: męskie/żeńskie)

** Jeżeli zamiast układu ADM232L zostanie zastosowany standardowy układ XX232 kondensatory C12..15 powinny mieć pojemność 10µF. W przypadku zastosowania układu ADM232A kondensatory C12..15 powinny mieć pojemność 100nF każdy.*

Dla ciekawskich - jak karta i procesor ze sobą rozmawiają?

Wymiana danych pomiędzy procesorem sterującym pracą programatora i kartą jest dość złożona. Dzieje się tak pomimo zastosowania popularnego interfejsu, który w znacznym stopniu jest zgodny ze standardem I²C. Zgodność polega przede wszystkim na tym, że początek i koniec każdej przesyłanej ramki wymaga wygenerowania przez procesor (pracujący jako Master) warunków *START* i *STOP*. Wykorzystywane są także pozostałe elementy procesu transmisji danych poprzez szynę I²C, typowe dla ogólnie znanych standardów.

Na **rys. 8** przedstawiono podstawowy algorytm transmisji danych. Jak widać każde polecenie (omówimy je w drugiej części artykułu) wymaga potwierdzenia niezależnym hasłem, którego długość wynosi 64 bity. Na **rys. 9** przedstawiono sposób „odpytywania“ karty o potwierdzenie *ACK*. Taka procedura nie jest typowa dla standardowych układów I²C. Warto zwrócić uwagę, że brak potwierdzenia *ACK* nie powoduje uznania transmisji za nieważną, jeżeli trwa krócej niż 10ms (czas niezbędny do zapisania matrycy EEPROM). Na **rys. 10** przedstawiono algorytm „odpytywania“ karty po wysłaniu bajtu polecenia.

Wszelkie problemy z ograniczeniami czasowymi rozwiązuje oprogramowanie sterujące pracą mikrokontrolera.

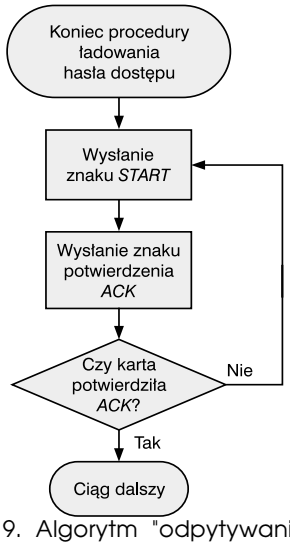
Piotr Zbysiński, AVT

Dokończenie artykułu znajdzie się w październikowym numerze EP.

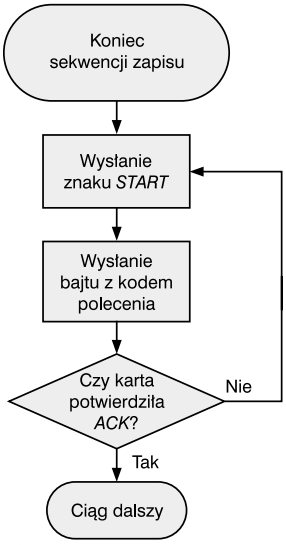
Dane katalogowe kart chipowych X76F100 i X76F640 są dostępne do końca października pod adresem: www.avt.com.pl/avt/ep/ftp.

Kody X76F640 udostępniła do testów firma E-2000 Setron.

Rys. 8. Sposób przesyłania danych.



Rys. 9. Algorytm "odpytywania" karty o znak ACK (bez bajtu polecenia).



Rys. 10. Algorytm "odpytywania" karty o znak ACK (z wykorzystaniem bajtu polecenia).