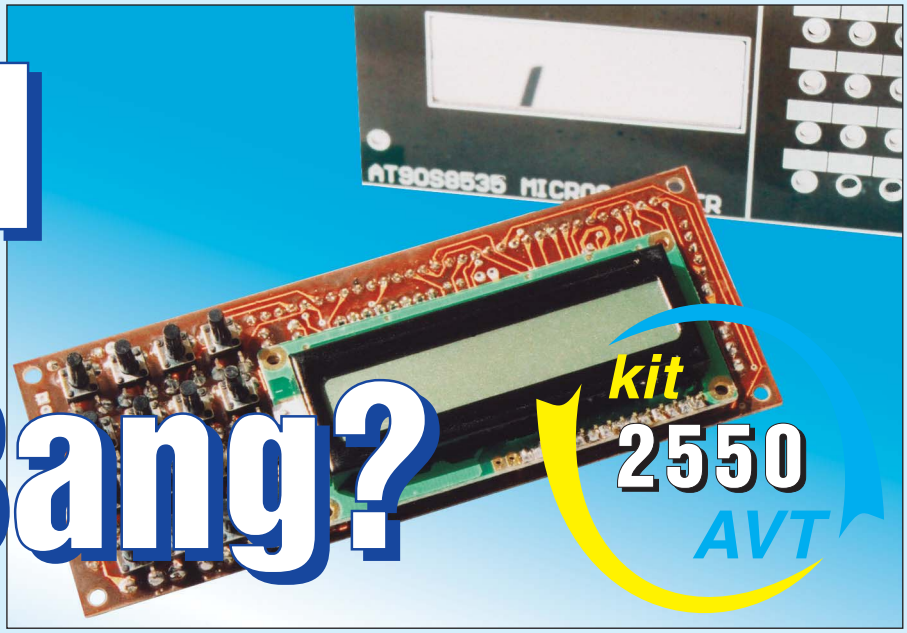
Pecel a może Big Bang?



część 1

Od redakcji:

Tego artykułu nie wolno przeoczyć. To absolutnie największe wydarzenie na łamach EdW w 2001 roku. W roku 2000 uczyliśmy się BASCOMa, a w tym roku naturalną tego konsekwencją jest opracowanie mikrokomputera, stanowiącego samodzielne urządzenie wyposażone we wszystkie elementy będące składnikami "dużych" komputerów. Posiada on monitor, klawiaturę, wszystkie stosowane w minikomputerach rodzaje pamięci, kilka portów do komunikowania się z otoczeniem (I^2C, 1WIRE i RS232) oraz to co chyba najważniejsze: możliwość rozbudowywania systemu tak, jak to ma miejsce w komputerach klasy PC.

Oto przedstawiamy taki mikrokomputer. Od dziś, jeśli będziesz potrzebował wyposażyć konstruowane przez Ciebie urządzenie w tak zwaną inteligencję - masz gotowe rozwiązanie. Nie zrobisz tego lepiej i prościej, niż programując przy pomocy BASCOMa nasz minikomputer. Jest to **m**inikomputer **o**sobisty **k**ażdego **e**lektronika, dlatego nazwaliśmy go **Pecel** (Personal Computer for Electronicians). Ktoś zaczął też w redakcji lansować nazwę **"Big Bang"**, wyrażając - z pewną, przyznajmy, egzaltacją- przekonanie, że ten minikomputer otwiera nową epokę w konstrukcjach elektronicznych. Jakoś trzeba ten komputer nazwać, więc chwilowo pozostaliśmy przy nazwie Pecel, ale zapraszamy wszystkich Czytelników EdW do zgłaszania własnych propozycji w konkursie na najlepszą nazwę tego minikomputera.

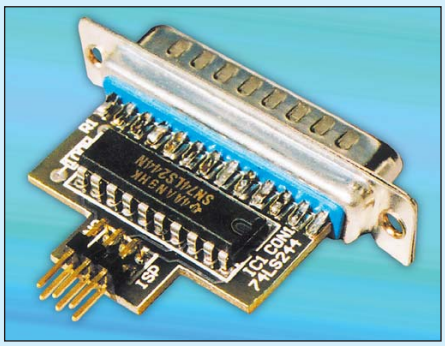
Chciałbym zaproponować Czytelnikom *Elektroniki dla Wszystkich* budowę komputera. No może trochę przesadziłem: nie komputera, ale mikrokomputera, który jednak będzie posiadał wszystkie elementy właściwe dla „dużych" maszyn, z komputerami klasy PC włącznie.

Budowanie minikomputerów ma już na łamach EdW pewne tradycje. Przebojem wśród kitów AVT był przez wiele lat i jest nim nadal słynny *Komputer Edukacyjny*, skonstruowany przez Sławka Surowińskiego. Maszyna ta umożliwiła nam pierwszy kontakt z techniką mikroprocesorową i trudno byłoby policzyć, ilu z Was zapoznało się dzięki temu układowi z programowaniem procesorów w języku asemblera. Komputer AVT-2250 jest układem typowo edukacyjnym i eksperymentalnym, chociaż istnieje także możliwość zastosowania go do celów praktycznych. Natomiast proponowany przeze mnie układ ma być przede wszystkim urządzeniem o rozlicznych zastosowaniach praktycznych, a jego walory edukacyjne wynikają z samych założeń konstrukcyjnych, narzucających użytkownikowi konieczność samodzielnego napisania programu sterującego mikrokomputerem.

W odróżnieniu od poprzednio budowanych minikomputerów i płyt testowych (np. AVT-2500) nasz nowy minikomputer jest urządzeniem w pełni funkcjonalnym, posiadającym wyjątkowo zwartą budowę, a nawet polecaną mu płytę czołową wyposażoną w stosowne napisy. Elementami decydującymi o wymiarach urządzenia był wyświetlacz alfanumeryczny i klawiatura, tak więc wymiary minikomputera niewiele wykraczają poza obrys wyświetlacza.

Sercem naszego mikrokomputera jest nowoczesny, wykonany w technologii RISC, bardzo szybki i wyposażony w wielką liczbę funkcji dodatkowych procesor typu AT90S8535. Oceniam, choć może to być ocena nieco subiektywna, że procesor ten należy do najlepszych w swojej klasie, a relacja pomiędzy jego możliwościami i ceną przedstawia się wyjątkowo korzystnie. Już sam fakt umieszczenia w strukturze procesora nieulotnej pamięci danych EEPROM, ośmiu przetworników analogowo-cyfrowych czy zegara czasu rzeczywistego, co zwalnia nas z konieczności stosowania wielu elementów zewnętrznych, przesądza o celowości zastosowania właśnie tego typu procesora.

Jest jeszcze jeden powód, dla którego AT90S8535 doskonale nadaje się do prac hobbystycznych: znaczny obszar pamięci programu, jaki mamy do dyspozycji i ogromna łatwość jej programowania. W strukturze tego procesora znalazło się miejsce na 8kB pamięci EEPROM. Jest to ogromny obszar pamięci i zręczny programista potrafi na tym



procesorze zrealizować prawdziwe cuda. Niestety, nie wszyscy jesteśmy wykwalifikowanymi programistami i taka pojemność pamięci programu to prawdziwy dar niebios także dla początkujących. Uwalnia to ich bowiem od ustawicznych stresów związanych z przekroczeniem rozmiaru programu przeznaczonego do umieszczenia w procesorze z 2, czy nawet 4kB EEPROM-em.

Wszystkie procesory AVR, w tym oczywiście nasz '8583 mogą być programowane w języku MCS BASIC, zaimplementowanym w pakiecie BASCOM AVR, bracie bliźniaku znanego nam BASCOM-a 8051. Dialekt MCS BASIC stosowany w pakiecie BASCOM AVR praktycznie nie różni się od języka stosowanego w BASCOM-ie 8051. Różnice wynikają głównie z odmiennego nazewnictwa wyprowadzeń procesora i znacznie bogatszego zestawu funkcji zaszytych w strukturach procesorów AVR.

Doszliśmy w tym momencie do jeszcze jednego powodu, który wpłynął na decyzję o wyborze typu procesora zastosowanego w naszym minikomputerze. Pakiet BASCOM AVR stał się ostatnio „okrętem flagowym" firmy MCS Electronics, co oczywiście nie oznacza, że zaprzestano prac nad doskonaleniem BASCOM-a 8051. Można powiedzieć, że Mark doprowadził do perfekcji swoją ideę: „Co tu wymyślić, aby inni nie musieli myśleć?". Opracowane ostatnio najnowsze polecenia języka MCS BASIC dla procesorów AVR sprowadzają wiele trudnych problemów programistycznych, nad którymi ja sam przesiedziałem kilka nocy, do wydania jednego polecenia systemowego. W dalszej części opisu minikomputerka zapoznamy się z najnowszymi „fajerwerkami", za pomocą których nawet bardzo skomplikowany program można napisać w ciągu kilku minut.

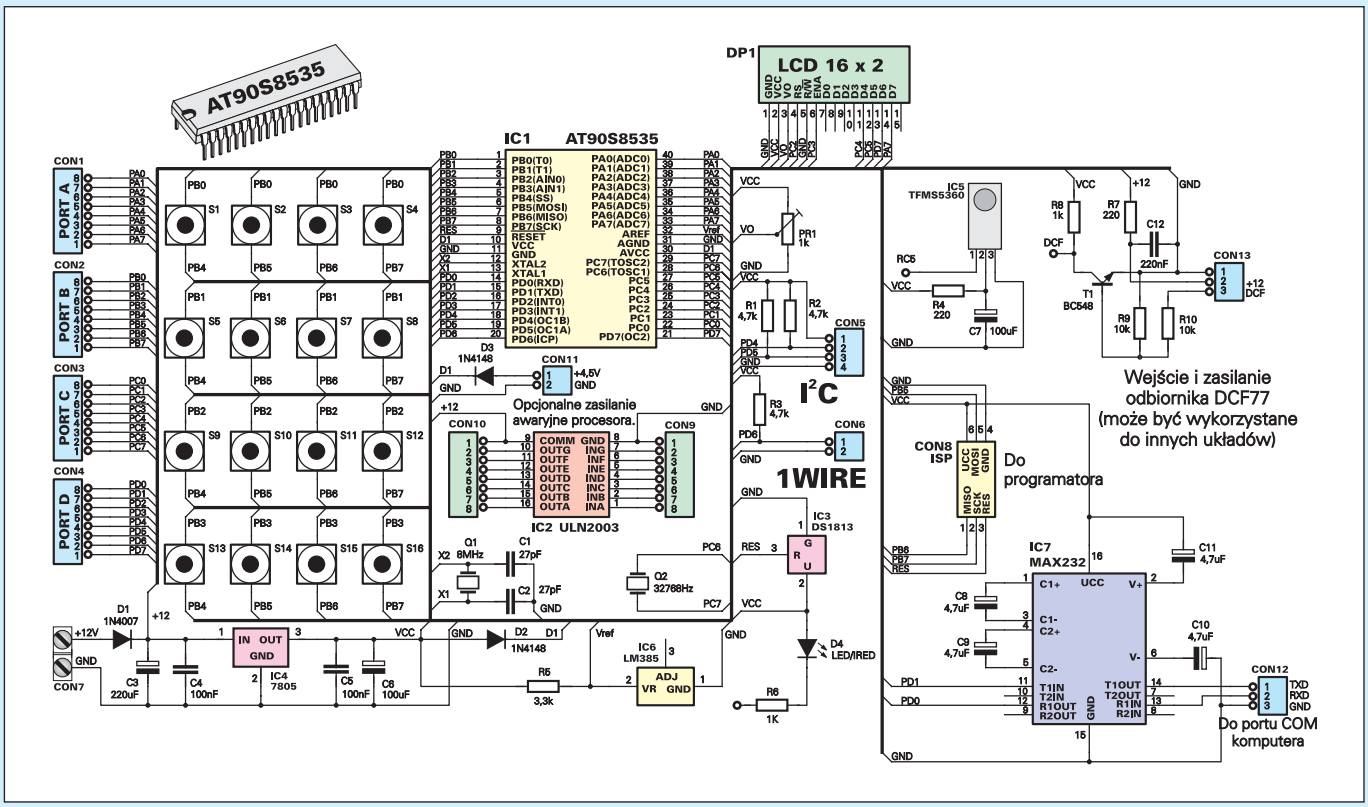
Z pewnością wielu z Was z niepokojem myśli już o jednej, niesłychanie ważnej podczas tworzenia systemu mikroprocesorowego sprawie: o programowaniu procesora. Być może napisanie programu jest sprawą prostą, ale jak wprowadzić go do pamięci CPU? Programatory procesorów są z zasady urządzeniami bardzo skomplikowanymi i kosztownymi i co nam przyjdzie z posiadania minikomputera i napisanego dla niego programu, jeżeli nie będziemy mieli możliwości wprowadzenia go do pamięci procesora? Bardzo się mylicie, Moi Drodzy! Programator procesorów AVR, w tym procesora 90S8535, jest urządzeniem banalnie prostym i składającym się tylko z jednego standardowego układu TTL! Powiem więcej: można w ogóle obyć się bez programatora podłączając interfejs SPI umieszczony w strukturze procesora bezpośrednio do portu drukarkowego! Jest to jednak rozwiązanie awaryjne i na co dzień będziemy się posługiwać prostym programatorkiem, którego płytka PCB mieszcząca się we wnętrzu typowej obudowy wtyku drukarkowego będzie dołączana za darmo do kitu minikomputera.

Bez najmniejszej przesady mogę stwierdzić, że dysponując naszym komputerkiem i niekiedy jednym czy dwoma opracowanymi dla niego urządzeniami dodatkowymi, będziemy mogli zbudować **KAŻDY system mikroprocesorowy**, jaki tylko przyjdzie nam do głowy. Jeżeli nawet napotkamy problemy związane z ograniczeniami sprzętowymi, jakie istnieją w każdym bez wyjątku komputerze czy minikomputerze, to od czego bogowie dali nam ręce i głowę? Jeżeli zatem zajdzie rzeczywista potrzeba, to zbudowane zostaną nowe karty, rozszerzające i tak już bogate możliwości minikomputera.

Chciałbym jeszcze powrócić do jednej, uprzednio poruszonej ogólnikowo sprawy: wartości edukacyjnej proponowanego układu. Chciałbym, a mam nadzieję, że Czytelnicy zgodzą się ze mną, aby opis minikomputera był w jakimś sensie kontynuacją kursu BASCOM College. Wykłady BASCOM College zakończyły się dość szybko. Wiele ciekawych problemów związanych z programowaniem procesorów nie zostało nawet zasygnalizowanych, nie mówiąc nawet o ich dokładnym omówieniu. Mogę tu przykładowo wymienić sprawy związane z transmisją danych pomiędzy komputerem a procesorem, wykorzystującą interfejs RS232, o której opis domagało się wielu Czytelników. Nadarza się zatem dokonała okazja, aby dysponując odpowiednim hardware (płytka AVT-2500 nie miała urządzeń sprzętowych niezbędnych do przeprowadzenia transmisji RS232) opisać dokładnie tę i inne sprawy.

Kurs BASCOM College zakończył się już dawno, kilka miesięcy temu, a przy tempie rozwoju oprogramowania oferowanego przez MCS Electronics jest to cała epoka. Podam

Rys. 1 Schemat ideowy



Wam tylko jeden przykład: na jednym z wykładów BC opisywaliśmy metody odczytywania numerów seryjnych i zawartości rejestrów układów 1WIRE produkowanych przez firmę DALLAS. Przerobiliśmy tylko proste przykłady, a ja sam nie wiedziałem, jak np. odczytać numery wielu układów 1WIRE jednocześnie dołączonych do tej samej linii. Wiedziałem tylko, że da się to zrobić, ale że sprawa jest dość skomplikowana i że już kilku dobrych programistów połamało sobie na tym zęby. A jak ten problem wygląda w czerwcu 2001, kiedy piszę ten artykuł? Mark sprowadził go do kilku prostych poleceń, które zastosować potrafi nawet początkujący programista:

```
var = 1WIRECOUNT( )- podaj liczbę układów 1WIRE
                          dołączonych do magistrali
var = 1WSEARCHFIRST( )        - podaj numer
                          seryjny pierwszego układu
var = 1WSEARCHNEXT( )         - podaj numery
                          seryjne dowolnej liczby układów
```

Czego właściwie będziemy potrzebować, aby rozpocząć korzystanie z naszego minikomputera? Baza sprzętowa i programowa są w końcu nieraz rzeczami najważniejszymi, ponieważ decydują o kosztach, jakie będziemy musieli ponieść przed rozpoczęciem pracy i nauki. Na szczęście sprzęt, jakiego będziemy potrzebować, nie jest ani bardziej skomplikowany, ani kosztowny od wyposażania, które musieliśmy mieć do dyspozycji podczas przerabiania programu BASCOM College.

1. Komputer klasy PC. Nie musi to być wcale jednostka z PENTIUM taktowanym zegarem 1GHz, ale jakikolwiek komputer, na którym można uruchomić system WINDOWS95/98/NT/2000. A zatem, w skrajnym przypadku możemy się zadowolić nawet maszyną z procesorem PENTIUM I! Oczywiście, jest to absolutne minimum, ale wymagania BASCOM-a są tak niewielkie, że w ostateczności możemy posłużyć się nawet takim muzealnym zabytkiem.

2. Programator procesorów AVR. Schemat tego programatora zostanie pokazany w dalszej części artykułu. Jak już wspominałem, jest to układ wręcz śmiesznie prosty i tani.

3. Kabelek do programatora, czyli zwykły kabel płaski zakończony dwoma zaciskanymi wtykami 10 pin.

4. Bardzo przydatny może okazać się jeszcze jeden kabelek. Taki niezbyt długi, 1 ... 2mb, z trzema przewodami i zakończony żeńskim wtykiem DB9. Umożliwi on komunikację pomiędzy naszym komputerkiem a maszyną klasy PC. Komunikacja ta będzie odbywać się poprzez łącze RS232, a jak bardzo może okazać się użyteczna także podczas pisania i testowania programów, dowiecie się w dalszej części tego artykułu.

5. No i wreszcie najważniejsze: software! Potrzebny Wam będzie pakiet BASCOM AVR, brat bliźniak dobrze Wam już znanego BASCOM-a 8051. Na początek wystarczy BASCOM AVR Demo, którego możliwości w obecnej chwili nie ustępują możliwościom jego wersji komercyjnej. Nawet ograniczenie długości kodu wynikowego zostało ostatnio zmniejszone i wynosi obecnie 2kB. Niestety, jest to za mało dla pełnego wykorzystania możliwości procesora ‘8535 i dlatego w przyszłości będziecie musieli pomyśleć o zakupie wersji komercyjnej. Jej cena wynosi niezmiennie 69USD + koszta, czyli przy zakupie bezpośrednio w MCS Electronics około 85USD. Nie jest to mało, ale cena BASCOM-a w relacji do jego możliwości jest po prostu rewelacyjnie niska! Sprawdźcie, ile kosztują inne kompilatory, np. Keil! A przecież BASCOM jest nie tylko kompilatorem, ale całym, potężnym zestawem narzędziowym. Pakiet BASCOM AVR DEMO jest dostępny za darmo na stronie internetowej EdW. Został on także umieszczony na płycie CD-EP wrześniowej Elektroniki Praktycznej.

Zajmijmy się teraz dalszymi konkretami, czyli możliwościami oferowanymi przez minikomputer.

Funkcje mikrokomputera realizowane bezpośrednio przez procesor AT90S8535:

- Wewnętrzna pamięć programu o pojemności 8kB. Porównując nasz mikrokomputerek z maszynami klasy PC, jest to nic, ale dla systemu mikroprocesorowego, którym w istocie jest nasz układ taka ilość pamięci to prawie nieograniczone możliwości rozbudowy programu. Przypomnijcie sobie, ile ciekawych układów zaprojektowaliśmy wykorzystując procesory z 2kB pamięci (AT89C2051 czy AT90S2313), i wyobraźcie sobie, co można zdziałać mając do dyspozycji aż taki obszar pamięci.

- Wewnętrzna nieulotna pamięć danych EEPROM o pojemności 512B, która w większości przypadków pozwala na rezygnację ze stosowania pamięci zewnętrznych, prawie zawsze dodawanych do układów z procesorami ‘2051. W pamięci tej możemy zapisać 512 bajtów, czyli np. 512 różnych liczb. W dalszej części tego artykułu dowiecie się, że procedura zapisu i odczytu danych z EEPROM-a jest banalnie prosta i sprowadza się do wydania tylko jednego polecenia.

- Wewnętrzna pamięć danych SRAM o pojemności także 512B. Napisałem naprawdę sporo bardzo rozbudowanych programów na ten procesor, ale jeszcze nigdy „nie udało” mi się zapełnić tej pamięci nawet w połowie.

- ośmiokrotny 10-bitowy przetwornik analogowo-cyfrowy. Przetwornik korzysta z zewnętrznego źródła napięcia odniesienia, równego lub mniejszego od napięcia zasilania. A zatem, wszelkie operacje związane z pomiarem wartości analogowych za pomocą naszego komputera będą wymagały w najgorszym wypadku tylko tego jednego elementu zewnętrznego, a pomiarów będziemy mogli dokonywać aż w ośmiu punktach jednocześnie. Dokładność pomiarów jest w większości przypadków aż nadto wystarczająca. Dla przykładu: dokonując pomiaru w zakresie do 5V dysponujemy rozdzielczością 4,8mV.

- sprzętowy UART, czyli układ umożliwiający transmisję danych z wykorzystaniem protokołu RS232. A więc, nasz mikrokomputer może bezpośrednio „porozumiewać się” z dużymi maszynami klasy PC, a także z każdym innym komputerem lub systemem mikroprocesorowym wyposażonym w interfejs RS232. UART umieszczony w strukturze procesora wspomagany jest sprzętowo przez dodatkowy hardware umieszczony na płycie głównej naszego mikrokomputera.

- RTC – zegar czasu rzeczywistego, a właściwie osobny oscylator + timer, który z zewnętrznym kwarcem 32768Hz automatycznie generuje przerwania co 1 sekundę. Zegar czasu rzeczywistego otrzymał ostatnio potężne wsparcie software’owe w języku MCS BASIC. Czy wiecie, jak teraz wygląda programowa konstrukcja zegara, pokazującego aktualny czas i datę? Ano, tak: LCD TIME$ i LCD DATE$. To wszystko.

- Trzy sprzętowe timery, w tym jeden (Timer2) mogący współpracować z dodatkowym zewnętrznym rezonatorem kwarcowym. Ten właśnie timer wykorzystywany jest do realizacji funkcji zegara czasu rzeczywistego. Wszystkie timery są wyposażone w bardzo rozbudowane funkcje, takie jak preskalery, sprzętową generację PWM i inne.

- Dwa zewnętrzne źródła przerwań sprzętowych: INT0 i INT1

- Wbudowany sprzętowy interfejs SPI. Jest to jedna z największych zalet procesora ‘8535. Interfejs SPI umożliwia nie tylko komunikację z innymi układami i systemami mikroprocesorowymi, ale **także programowanie procesora bez konieczności wyjmowania go z podstawki**. Do złącza CON8 umieszczonego na płycie głównej mikrokomputera możemy dołączyć prosty programator ISP (In System Programming) i po napisaniu programu bądź jego fragmentu wprowadzić go do pamięci procesora naciskając tylko jeden klawisz. Programowanie w systemie nie tylko znakomicie upraszcza i przyspiesza pracę nad nowym programem, ale także eliminuje ryzyko uszkodzenia wyprowadzeń procesora podczas częstego wyjmowania i wkładania go w podstawkę.

- Sprzętowy watchdog, czyli dodatkowy, wyspecjalizowany timer skutecznie zabezpieczający procesor przed „zawieszeniem się” na przykład na skutek wystąpienia silnych zakłóceń zewnętrznych.

Funkcje mikrokomputera realizowane przy współpracy z hardware umieszczonym na płycie głównej:

- Wyświetlacz alfanumeryczny LCD. Element ten jest podstawowym układem służącym przekazywaniu informacji opracowanej przez komputer i przetłumaczonej na „ludzki" język cyfr, liter i znaków specjalnych. W mikrokomputerze można zastosować, zależnie od potrzeb i możliwości finansowych wyświetlacz 16*1 lub 16*2 znaki, z podświetlaniem lub bez.

- Klawiatura szesnastoprzyciskowa. Klawiatura ta, zbudowana z tanich i łatwych do nabycia przycisków, po odpowiednim oprogramowaniu umożliwia wprowadzanie do komputera nie tylko liczb, ale także wszystkich znaków alfanumerycznych. Programowa obsługa klawiatury zostanie szczegółowo omówiona w dalszej części artykułu, ale już teraz mogę Wam powiedzieć, że sprowadza się ona do jednego polecenia języka MCS BASIC: GETKBD! Tym Czytelnikom, którym nie wystarczy taka prosta klawiatura i którzy chcieliby wprowadzać do minikomputera dane za pomocą typowej konsoli od PC, mogę już teraz powiedzieć, że dołączenie takiej klawiatury do naszego układu jest sprawą banalnie prostą, a wprowadzanie z niej danych odbywa się przy pomocy tylko jednego polecenia programowego.

- Magistrala I^2C jest jednym z najważniejszych elementów naszego mikrokomputera, który umożliwia praktycznie nieograniczoną rozbudowę systemu. Większość układów peryferyjnych opracowanych dla mikrokomputera sterowana jest poprzez magistralę I^2C, a ponadto do dyspozycji będziemy mieli także ogromną ilość modułów I^2C spełniających najróżniejsze funkcje, których opisy zostały opublikowane w Elektronice Praktycznej.

- Magistrala 1WIRE obsługująca układy opracowane przez firmę DALLAS, w tym termometry cyfrowe, przełączniki i oczywiście „magiczne" tabletki z serii i-BUTTON.

- Moduł odbiornika kodu RC5 lub innego transmitowanego w podczerwieni, z nośną o częstotliwości zbliżonej do 36kHz. Moduł ten może okazać się użyteczny nie tylko do odbierania informacji z pilota od sprzętu RTV, ale i do komunikowania się z innymi procesorami lub wspomnianym sprzętem. Po opracowaniu przez MCS Electronics polecenia SENDRC5 generowanie kodu sterowania na zakresie podczerwieni stało się naprawdę banalnie proste, podobnie jak generacja kodu DTMF (DTMFOUT), wykorzystywanego w telefonii.

- Bezpośrednie sterowanie odbiornikami prądu stałego umożliwia umieszczony na płycie głównej mikrokomputera układ typu ULN2003, zawierający w swojej strukturze siedem driverów mocy, o maksymalnym prądzie do 500mA każdy. Z wyjść tych driverów możemy bezpośrednio sterować silniczkami elektrycznymi DC, w tym czterofazowymi silnikami krokowymi, przekaźnikami, elektromagnesami, a także żarówkami na napięcie 12V i girlandami diod świetlnych.

- Jednym z najważniejszych układów umożliwiających komunikację komputerka z innymi urządzeniami elektronicznymi jest pełny interfejs RS232 zrealizowany na popularnym układzie MAX232. Za pomocą tego układu, wspieranego przez sprzętowy UART wbudowany w strukturę procesora AT90S8535, możemy nawiązać łączność z dowolnym komputerem wyposażonym w interfejs RS232 (czyli z każdą maszyną klasy PC) lub innym urządzeniem elektronicznym.

- Na płycie głównej został także umieszczony dodatkowy rezonator kwarcowy o częstotliwości podstawowej 32768Hz, czyli popularny „kwarc zegarkowy". Element ten umożliwia uruchomienie wewnętrznego sprzętowego generatora czasu rzeczywistego, bloku wręcz bezcennego nie tylko dla konstruktorów zegarów, ale także innych urządzeń wymagających pomiaru czasu, w tym mierników częstotliwości.

- Jak już wiemy, procesor AT90S8535 został wyposażony w wewnętrzny ośmiokanałowy przetwornik analogowo-cyfrowy o rozdzielczości 10 bitów. Niewiele jest on jednak wart bez zewnętrznego, wysokostabilnego źródła napięcia odniesienia, które na szczęście zostało umieszczone na płycie głównej. Jako źródło napięcia odniesienia 2,5V został zastosowany układ LM385.

- Każdy, kto choćby trochę zapoznał się z zasadami konstruowania układów zrealizowanych w technice mikroprocesorowej, wie, jakie znaczenia ma prawidłowy start procesora po włączeniu zasilania. Zamontowany na płytce reset sprzętowy typu DS1813 nie tylko zapewnia właściwe warunki startu procesora, ale także nadzoruje poziom napięcia zasilającego. Spadek tego napięcia poniżej poziomu dopuszczalnego dla procesora AT90S8535 mógłby, w przypadku dalszej pracy procesora, mieć nieobliczalne następstwa, polegające głównie na uszkodzeniu zawartości pamięci danych EEPROM.

- Wiem, że bardzo lubicie konstruować zegary. Nasz minikomputer daje w tym zakresie ogromne, wręcz nieograniczone możliwości. Możecie zbudować zarówno prosty zegarek, jak i bardzo rozbudowane układy nadzorujące w funkcji czasu dziesiątki urządzeń peryferyjnych. Tylko że dokładność takiego zegara będzie taka, jaka będzie dokładność zastosowanego w nim rezonatora kwarcowego 32768Hz, czyli niekiedy niezbyt wielka. A co powiecie, Moi Drodzy, na zegar, którego dokładność będzie wynosić **1 sekundę na ... pięć milionów lat**? Taki właśnie zegar lub sterownik pracujący z absolutną z ludzkiego punktu widzenia precyzją, będziecie mogli zbudować, wykorzystując dodatkowe elementy umieszczone na płytce minikomputera oraz zewnętrzny odbiorniczek radiowy. Reszta to tylko kilkanaście, no, powiedzmy, kilkadziesiąt linijek programu, dokładnie omówionego w dalszych częściach tego artykułu.

- Z przyczyn, o których wspomnimy w dalszej części artykułu, procesor sterujący pracą naszego minikomputerka zasilany jest napięciem obniżonym o 0,6V w stosunku do napięcia zasilającego resztę układu (+5VDC). Obniżenie napięcia zostało zrealizowane za pomocą diody krzemowej, która jednocześnie separuje zasilanie procesora od reszty układu. Nic więc prostszego, aby dodając dodatkowa diodę i złącze umożliwić sobie awaryjne zasilanie samego tylko procesora z dodatkowego źródła, np. baterii 4,5V. Procesor pobiera znikomo mały prąd, szczególnie po wprowadzeniu go w stan IDLE lub POWER DOWN i takie rozwiązanie może być niezwykle cenne np. w konstrukcjach zegarów.

Funkcje mikrokomputera realizowane za pomocą specjalnie dla niego opracowanego, dodatkowego sprzętu.

- Na płycie głównej naszego mikrokomputera zostały umieszczone drivery mocy umożliwiające sterowanie odbiornikami prądu stałego, o poborze prądu nie przekraczającym 500mA. Nie jest to zbyt wiele i dlatego zaprojektowana została oddzielna karta rozszerzająca, na której można umieścić do 8 przekaźników typu RM-86. Każdy z nich posiada dwie pary przełączanych styków o obciążalności prądowej do 8A. Karta może służyć do zasilania urządzeń prądem stałym lub przemiennym o napięciu do 250VAC.

- Komputer bez karty dźwiękowej? To chyba niemożliwe i dlatego nasz układ został wyposażony w kartę, na której umieszczony został dobrze wszystkim znany „silikofon", czyli ISD25120. Karta sterowana jest za pomocą magistrali I^2C i umożliwia nagrywanie i odtwarzanie sekwencji akustycznych o łącznym czasie trwania do 2 minut. Należy sądzić, że karta ta okaże się bardzo użyteczna dla konstruktorów, którzy zajmą się konstruowaniem „mówiących" zegarów czy innych układów domowej automatyki.

- W praktyce konstruktora hobbysty bardzo często spotykamy się z koniecznością „ożywiania" wykonanych konstrukcji i do tego celu najczęściej wykorzystujemy silniki elektryczne różnych typów. Silniki prądu stałego o małej mocy możemy sterować bezpośrednio z płyty głównej mikrokomputera, ale ograniczeniem jest tu pobierany z niej maksymalny prąd i napięcie. Ponadto, do sterowania np. krokowym silnikiem czterofazowym niezbędne byłoby wykorzystanie aż czterech wyjść procesora, a sterowanie silnikami krokowymi dwufazowymi jest w ogóle niemożliwe. Dlatego też został zaprojektowany dodatkowy moduł rozszerzający możliwości

komputera, sterowany magistralą I²C, za pomocą którego możemy zasilać:
• Dwa silniki krokowe dwufazowe,
• Dwa silniki krokowe czterofazowe,
• Cztery silniki komutatorowe DC z możliwością zmiany kierunku i prędkości obrotowej,
• Osiem silników DC bez możliwości zmiany kierunku obrotów lub osiem dowolnych urządzeń zasilanych prądem stałym o napięciu do 35V i pobieranym prądzie nie większym niż 500mA.

- Jak wiadomo, apetyt rośnie w miarę jedzenia. Procesor AT90S8535 to potężna jednostka, o ogromnych możliwościach i obszernej pamięci programu. Jednak nie zdziwiłbym się, gdyby bardziej ambitnym Konstruktorom nawet jego możliwości pewnego dnia przestały wystarczać i okazałoby się np., że napisany skomplikowany program wykracza swoimi rozmiarami ponad 8kB kodu wynikowego. Dla tych Konstruktorów przygotowałem chyba miłą niespodziankę: co powiecie na procesor o pamięci programu ... 128kB, sześciu portach wejściowo-wyjściowych, 4kB pamięci EEPROM i 4kB pamięci SRAM? Procesorem tym jest kolejny produkt firmy ATMEL: AT MEGA103. Jednak umieszczenie tego procesora w podstawce na płycie głównej naszego mikrokomputera jest absolutnie niemożliwe, i to z dwóch powodów: posiada on aż 64 wyprowadzenia i produkowany jest wyłącznie w obudowie przeznaczonej do montażu SMD. Jednak nie takie trudności już przezwyciężaliśmy: zaprojektowałem dla Was specjalną kartę rozszerzającą, dołączaną do podstawki procesora na płycie głównej mikrokomputera. Na karcie tej umieszczony zostanie procesor AT MEGA103 i złącza do dwóch dodatkowych portów.

Programowanie minikomputera

- Nasz minikomputer bez sterującego nim programu może być co najwyżej niezbyt efektowną ozdobą na biurko. Program można napisać w dowolnym języku posiadającym kompilator umożliwiający utworzenie kodu binarnego przeznaczonego do umieszczenia w pamięci procesora AVR. Szczególnie jednak polecam pakiet BASCOM AVR, ze względu na łatwość programowania i ogromny komfort pracy. Napisany program musi oczywiście zostać umieszczony w pamięci procesora i do tego celu potrzebny będzie programator obsługujący transmisję SPI i umożliwiający zaprogramowanie procesora bez konieczności wyjmowania go z podstawki. Na szczęście, w przeciwieństwie do programatorów równoległych taki programator jest urządzeniem wręcz śmiesznie prostym i tanim. Do naszego minikomputera został opracowany specjalny programator (**rysunek 2**), zrealizowany z wykorzystaniem zaledwie jednego układu scalonego z rodziny TTLLS, mieszczący się w typowej obudowie wtyku DB25. **Płytka tego programatora będzie za darmo dodawana do kitu minikomputera**. W dalszej części artykułu podane zostaną liczne przykłady programowana w MCS BASIC, dialekcie przeznaczonym dla procesorów AVR.

- Co jednak mają czynić ci Koledzy, którzy nie posiedli jeszcze umiejętności programistycznych w stopniu wystarczającym do napisania dość skomplikowanego programu? Dla nich przygotowywana jest specjalna niespodzianka: otóż w kicie nie będzie, jak można by się było spodziewać, dostarczany „czysty" procesor, ale zaprogramowany układ umożliwiający natychmiastowe korzystanie z minikomputerka. Jakie funkcje będzie wykonywał ten „fabryczny" minikomputer dowiecie się w dalszej części artykułu. Ważne jest jedno: jeżeli jego funkcje przestaną Wam wystarczać albo jeżeli będziecie chcieli stworzyć zupełnie nowe urządzenie, to sterujący nim program **nie będzie w żaden sposób zabezpieczony przed kopiowaniem** i zawsze będziecie mogli zapisać go na dysku, zaprogramować procesor po swojemu i w dowolnym momencie powrócić do „fabrycznego" programu. Program, który dla Was napisałem (jego funkcje są jeszcze niespodzianką) jest bardzo, ale to bardzo rozbudowany i zajmuje praktycznie całą pamięć EEPROM. Mogłoby to wywołać obawy, że posiadacze wersji Demo BASCOM-a AVR nie będą mogli skopiować go na dysk i ponownie zaprogramować nim procesora. Na szczęście takie obawy byłyby absolutnie bezzasadne: **ograniczenia długości kodu wynikowego występujące w wersji Demo dotyczą wyłącznie samej kompilacji programu, a nie operacji na już skompilowanych plikach.**

- W dalszej części artykułu omówimy szczegółowo wszystkie ważniejsze polecenia języka MCS BASIC, specyficzne dla procesorów AVR i nie omawiane podczas kursu BASCOM College.

Jak to działa?

Podobnie jak płytka testowa używana podczas kursu BACOM College w ogóle nie działa i działać będzie dopiero po zaprogramowaniu procesora, czy to programem napisanym samodzielnie, czy programem fabrycznym, zaszytym w procesorze dostarczonym w kicie. Jednak działanie tego programu omówimy później, a na razie zajmiemy się opisem hardware pokazanego na schemacie. A zatem, zmieniamy tytuł tego fragmentu artykułu na:

Z czego to się składa?

Zanim jednak rozpoczniemy tę pracę, chciałbym zwrócić się do zupełnie początkujących Czytelników i wyjaśnić im pewną sprawę. Bardziej doświadczeni konstruktorzy proszeni są o opuszczenie tego fragmentu artykułu. Chodzi mi o sposób rysowania schematu, nieco odmienny od tego, do którego jesteście przyzwyczajeni. Na większości schematów publikowanych w EdW wszystkie połączenia pomiędzy elementami zaznaczone były jako osobne linie. Jest to metoda dobra, ale jedynie w przypadku prostych układów. Przy rysowaniu schematów układów bardziej rozbudowanych, a w szczególności układów cyfrowych i mikroprocesorowych, do łączenia elementów używamy tzw. BUS, czyli jakby arterii komunikacyjnych, od których rozchodzą się odgałęzienia, każde zaopatrzone w indywidualną nazwę. Wiecie, do czego to można porównać? Do rozbebeszonej elektrycznej instalacji samochodowej! Tam także mamy grube, oplecione taśmą izolacyjną wiązki przewodów, od której odchodzą pojedyncze kable prowadzące do różnych elementów samochodowej instalacji. Różnica polega na tym, że w samochodzie przewody łączące ze sobą wspólne punktu układu oznaczone są kolorami, a na naszym schemacie tzw.etykietami, czyli niepowtarzalnymi nazwami.

Po tej małej dygresji przystąpmy wreszcie do inwentaryzacji dóbr widocznych na schemacie.

Złącza:

1. Sercem naszego minikomputera jest, oczywiście opisany już wyżej procesor typu AT90S8535 – IC1. Jednak sam procesor, bez niezbędnej mu eskorty niewiele by zdziałał.
2. Q1, C1, C2 są elementami niezbędnymi do funkcjonowania wewnętrznego oscylatora systemowego procesora. W układzie zastosowano rezonator kwarcowy o częstotliwości podstawowej 8MHz, czyli najwyższej dopuszczalnej dla procesora '8583.
3. CON1, CON2, CON3 i CON4 są złączami, do których doprowadzone zostały wszystkie aktywne wyprowadzenia procesora, czyli porty A, B, C i D. Do złącz tych możemy podłączyć ewentualne układy peryferyjne, a także aparaturę pomiarową.
4. CON5 jest jednym z najważniejszych elementów mikrokomputera. Umożliwia ono komunikację z dosłownie setkami układów peryferyjnych sterowanych magistralą I²C. Do tego samego złącza możemy dołączyć także klawiaturę od komputera PC, która wprawdzie nie jest układem I²C, ale wymaga identycznych połączeń.
5. CON6 spełnia podobną rolę co CON5 i obsługuje magistralę 1WIRE, czyli umożliwia kontakt z DOWOLNĄ liczbą układów produkcji firmy DALLAS. Do tego jednego wyprowadzenia możemy dołączać termometry, zdalnie sterowane przełączniki i „magiczne tabletki" DALLAS-a.
6. CON7 jest punktem, do którego doprowadzamy napięcie zasilające minikomputer, czyli 12VDC. Dioda D1 zabezpiecza układ przed katastrofalnymi skutkami odwrócenia polaryzacji napięcia zasilającego.
7. CON8, czyli coś dla wygodnych i dbających o bezpieczeństwo procesora. Jest to

złącze pełniące szczególnie ważną funkcję: umożliwia ono wielokrotne programowanie procesora bez konieczności wyjmowania go z podstawki. W dalszej części artykułu dowiecie się, jak nieprawdopodobny komfort pracy zapewnia to małe złącze!

8. CON9 i CON10 są wejściami i wyjściami bufora mocy ULN2803

9. CON11 - złącze alternatywnego zasilania procesora. Warto tu zwrócić uwagę na nietypowy sposób zasilania procesora, który jest dołączony do szyny zasilającej VCC o napięciu +5VDC za pośrednictwem diody D2. W związku z tym napięcie zasilania procesora jest zmniejszone o ok. 0,6V i wynosi tylko ok. 4,4VDC. Co spowodowało zastosowanie tak nietypowego rozwiązania? Otóż, jest to cała historia. Podczas uruchamiania kilku układów z procesorem AT90S8535 wykorzystujących wbudowany w jego strukturę oscylator i generator przerwań RTC napotkałem na nieoczekiwane i dziwaczne trudności. W niektórych układach oscylator nie działał w ogóle, a w innych pracował w niekontrolowany sposób, włączając się i wyłączając w nieoczekiwanych momentach. Ani sprawdzania części hardware'owej układu ani kodu napisanego programu nie dawało rezultatu, podobnie jak wertowanie karty katalogowej procesora. Na rozwiązanie problemu natknąłem się dopiero podczas lektury erraty do karty katalogowej, gdzie firma ATMEL umieściła wręcz kuriozalne stwierdzenie:

„When using an external 32 kHz crystal as asynchronous clock source for Timer2, the timer may count incorrectly at voltages above 4.0V. Keep the supply voltage below 4.0V when clocking Timer2 from an external crystal."

No comments! Nie wnikam, dlaczego budowa generatora kwarcowego 32768Hz o napięciu zasilania 5V okazała się zbyt trudna dla konstruktorów ATMEL-a. Ważne jest tylko to, że obniżenie napięcia o 0,6V spowodowało natychmiastowe usunięcie problemów ze sprzętowym RTC. W naszym układzie dioda D2 jest elementem opcjonalnym: jeżeli nie będziecie wykorzystywać sprzętowego zegara czasu rzeczywistego, to można po prostu zastąpić ją zworą.

10. CON12 pełni także niesłychanie ważną funkcję. Umożliwia ono połączenie naszego minikomputera z portem szeregowym komputera PC lub innego urządzenia elektronicznego wyposażonego w sprzętowy interfejs RS232.

11. CON13 jest złączem o szczególnym charakterze. Nie było go na płytce pierwszego prototypu naszego komputera i zostało dodane później. Jego zadaniem jest umożliwienie dołączenia do minikomputera typowego odbiornika sygnału DCF77 nadawanego na falach długich. Odebranie i zdekodowanie tego sygnału umożliwi nam budowę układów wykorzystujących atomowy wzorzec czasu o dokładności 1 sekundy na 5 milionów lat.

Układy scalone:

IC1 – najważniejszy element konstrukcji minikomputera, z którym dobrze się zapoznamy w najbliższym czasie.

IC2 – zawiera w swojej strukturze osiem driverów mocy. Każdy z nich może zasilać od strony masy układu pobierające prąd o wartości do 500mA. Zadaniem układu ULN2003 jest bezpośrednie sterowanie przekaźnikami, żarówkami, silnikami i innymi urządzeniami wykonawczymi.

IC3 – jest układem resetujacym procesor w przypadku spadku napięcia zasilającego poniżej wartości minimalnej, a także zapewnia pewny start procesora po włączeniu zasilania.

IC4 – to zwykły, znany Wam bardzo dobrze scalony stabilizator napięcia +5VDC.

IC5 – to także nasz dobry znajomy. Jego zadaniem jest odbieranie transmisji danych nadawanych w podczerwieni z częstotliwością nośną ok. 36kHz.

IC6 – to wzorzec napięciowy. Na jego wyprowadzeniu VR występuje napięcie dokładnie równe 2500mV, stabilne w funkcji temperatury i napięcia zasilającego. Jest to bardzo ważny element układu minikomputera w przypadku, kiedy będziemy chcieli dokonywać pomiaru wartości analogowych i wykorzystywać zawarty w strukturze procesora '8535 ośmiokrotny przetwornik analogowo-cyfrowy.

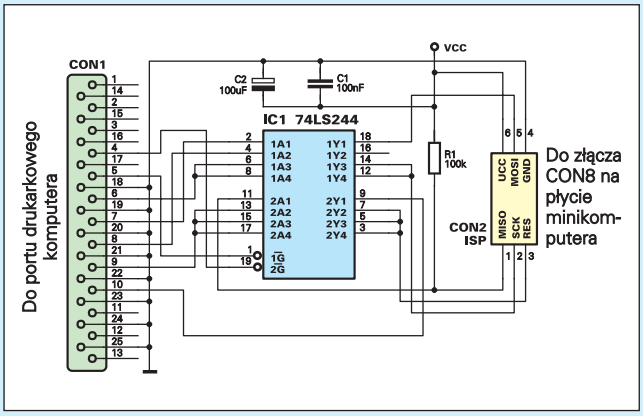
IC7 - zapewnia nawiązanie łączności pomiędzy naszym minikomputerem a maszynami klasy PC lub innymi urządzeniami elektronicznymi wyposażonymi w interfejs RS232.

Pozostałe elementy konstrukcji komputera:

- Klawiatura, składająca się z szesnastu klawiszy S1 .... S16 połączonych w matrycę 4x4 umożliwia wprowadzanie do komputera wszelkiego typu danych, tak liczb jak i tekstów.

- Wyświetlacz alfanumeryczny LCD – DP1. Możemy zastosować dwa typy wyświetlaczy: 16x1 i 16x2 znaki. Zdecydowanie polecałbym wyświetlacz drugiego typu (taki będzie dostarczany w kicie). Natomiast sprawą do dyskusji jest to, czy wyświetlacz ma być podświetlany, czy nie. Wyświetlacz z podświetlaniem jest z pewnością bardziej efektowny i lepiej czytelny, ale za te zalety trzeba zapłacić nieco większą cenę, a także liczyć się ze zwiększonym poborem prądu.

Rys. 2 Schemat ideowy programatora



Montaż i uruchomienie

Na **rysunku 3** została pokazana płytka obwodu drukowanego naszego minikomputera. Ze względu na znaczną komplikację połączeń płytka została zaprojektowana na laminacie dwustronnym z metalizacją.

Montaż minikomputera nie powinien nastręczyć nikomu większych trudności, pod warunkiem że będziecie przestrzegać kilku wskazówek, których za chwilę Wam udzielę. Przede wszystkim pamiętajcie o

Wykaz elementów

Rezystory:

PR1	1kΩ
R1, R2, R3	4,7kΩ
R4, R7	220Ω
R5	3,3kΩ
R6, R8	1kΩ
R9, R10	10kΩ

Kondensatory:

C1, C2	27pF
C3	220µF
C4, C5	100nF
C6, C7	100µF
C8, C9, C10, C11	4,7µF
C12	220nF

Półprzewodniki:

D1	1N4007
D2, D3	1N4148
D4	LED
IC1	AT90S8535
IC2	ULN2003
IC3	DS1813
IC4	7805
IC5	TFMS5360
IC6	LM385 2,5V
IC7	MAX232
T1	BC548

Pozostałe:

CON1, CON2, CON3, CON4, CON9, CON10	
8 x goldpin	CON5 4 x goldpin
CON6, CON11	2 x goldpin
CON7	ARK2 (3,5mm)
CON8	3x2 goldpin
CON12, CON13	3 x goldpin
DP1	wyświetlacz alfanumeryczny LCD 16x2
Listwa goldpinów 16x1	
Q1	rezonator kwarcowy 8MHz
Q2	rezonator kwarcowy 32768Hz
S1 ... S16	microswitch 10 mm

Komplet podzespołów z płytką jest dostępny w sieci handlowej AVT jako kit szkolny AVT-2550

mądrej zasadzie „Festina lente” i wszystkie czynności podczas montażu komputerka wykonujcie powoli i z rozmysłem. Wylutowywanie nieprawidłowo zamontowanych elementów z płytki dwuwarstwowej nie należy do przyjemności!

No tak, postraszyłem Was trochę, a tak naprawdę to wszystkie te zastrzeżenia dotyczą tylko dwóch elementów, na które musicie zwrócić szczególną uwagę: klawiatury i szeregu goldpinów, do których następnie będziecie musieli przylutować wyświetlacz alfanumeryczny. Elementy te zostawmy sobie na sam koniec pracy, a teraz zamontujmy pozostałe. Ten etap montażu wykonujemy „po Bożemu”, rozpoczynając od wlutowania w płytkę rezystorów i podstawek pod układy scalone, a kończąc na kondensatorach elektrolitycznych i stabilizatorze napięcia. Bez obaw **lutujemy także te elementy, które znajdują się w obrębie klawiatury: jej przyciski będą zamontowane po przeciwnej stronie płytki**.

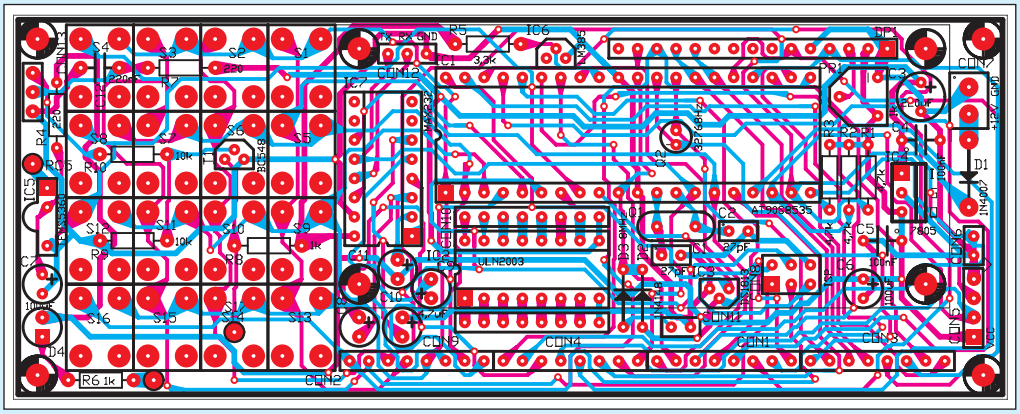
Po wlutowaniu tych elementów odwracamy płytkę na drugą stronę i lutujemy szereg 16 goldpinów od strony (umownej) ścieżek, czyli po przeciwnej stronie co pozostałe elementy. Goldpiny lutujemy do szeregu punktów lutowniczych oznaczonych na płytce jako DP1.

Pora teraz na najważniejszą czynność, jaką musimy wykonać przed zamontowaniem wyświetlacza: na **kilkukrotne sprawdzenie poprawności montażu już wlutowanych elementów**. Pamiętajmy, **że po zamontowaniu wyświetlacza utracimy dostęp do wielu punktów lutowniczych i dokonanie jakichkolwiek poprawek będzie bardzo, bardzo trudne**!

Jeżeli stwierdziliśmy ponad wszelką wątpliwość, że montaż został przeprowadzony poprawnie, to możemy teraz przylutować wyświetlacz. Czynność tę musimy przeprowadzić wyjątkowo starannie, montując wyświetlacz jak najbliżej powierzchni płytki i idealnie do niej równolegle. Aha, przed wlutowaniem wyświetlacza musimy jeszcze zagiąć metalowe łapki, mocujące obudowę wyświetlacza do jego płytki, tak aby ściśle przylegały do jej powierzchni. Czynność tę należy wykonać bardzo delikatnie, za pomocą małych kombinerek.

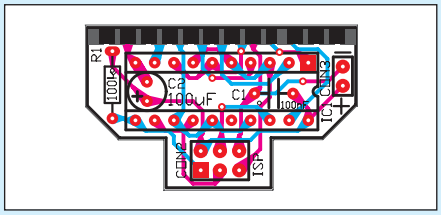
Ostatnią czynnością montażową będzie budowa szesnastkowej klawiatury, a jej przebieg zależeć będzie od rodzaju zastosowanego wyświetlacza. Jeżeli użyliśmy wyświetlacza bez podświetlania, to przyciski lutujemy tak, jak inne elementy, oczywiście podobnie jak złącze wyświetlacza od spodniej strony płytki. Na pewne problemy napotkamy jedynie w przypadku zastosowania wyświetlacza z podświetlaniem, który ze swej natury jest o kilka milimetrów grubszy od swojego uboższego krewnego nie świecącego w ciemności. W takim wypadku microswitche muszą zostać odsunięte jak najdalej od powierzchni płytki, tak aby ich przyciski wystawały co najmniej o 2milimetry ponad powierzchnię wyświetlacza. A zatem, jeżeli zdecydowaliśmy się na wyświetlacz z podświetlaniem, to przyciski lutujemy „powierzchniowo”, nie przewlekając ich końcówek przez otwory w punktach lutowniczych.

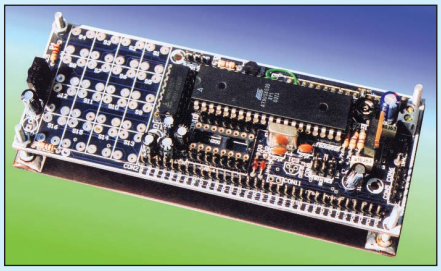
Montaż klawiatury jest ostatnią czynnością jaką musimy wykonać podczas budowy płyty głównej naszego komputera. Dla tych, którzy przebrnęli przez ten etap będzie fraszką połączenie płyty głównej z płytą czołową komputerka, widoczną na wkładce w środku numeru. To tego celu najlepiej wykorzystać tulejki dystansowe lub po prostu cztery śrubki M3, których łebki lutujemy do punktów lutowniczych płyty czołowej i łączymy całość z płytą główną za pomocą ośmiu nakrętek M3.



Rys. 3

Rys. 4 Płytka drukowana programatora





W dalszych częściach artykułu zapoznam Was z następującymi sprawami:

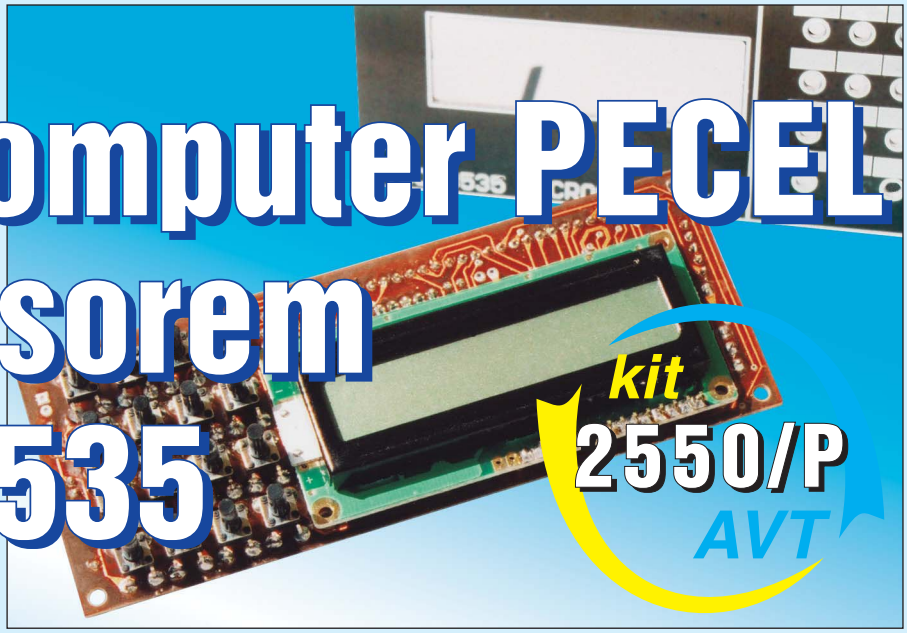
1. Budową programatorka ISP, stanowiącego nierozłączną całość z naszym minikomputerem.

2. Metodami programistycznymi potrzebnymi do „ożywienia” wykonanego układu i z najnowszymi „fajerwerkami” pakietu BASCOM AVR.

3. „Fabrycznym” programem obsługi minikomputera.

Zbigniew Raabe
zbigniew.raabe@edw.com.pl

Mikrokomputer PECEL z procesorem AT90S8535



Część 2

Nasz minikomputer jest w zasadzie gotowy. Do wykonania pozostały już tylko drobiazgi: banalnie prosty programator, sterowany z pakietu BASCOM AVR, który będzie służył do wprowadzania napisanego programu do pamięci minikomputera oraz dwa kabelki: jeden do programatora, a drugi do łączenia PECEL-a z portem szeregowym komputera. Zacznijmy od montażu programatorka.

Schemat elektryczny układu programatora AVR został pokazany na **rysunku 4**. Jak łatwo zauważyć ponieważ cała inteligencja programatora skupiona została w jego części software'owej układ został maksymalnie uproszczony i zawiera tylko aktywny element: IC1 - 74HCT244.

Układ jest zmodyfikowaną wersją programatora STK200 firmy Kanda, bardzo popularnego wśród elektroników. W Internecie można znaleźć sporo oprogramowania obsługującego ten programator, dostępnego jako shareware. Nic więc dziwnego, że wiedząc o popularności tej tysiące razy sprawdzonej konstrukcji, Mark zapewnił jej wsparcie software'owe z poziomu pakietów BASCOM AVR i BASCOM 8051. Widoczny na schemacie układ programatora nie jest urządzeniem związanym na śmierć i życie z naszym minikomputerem: za jego pomocą możemy zaprogramować każdy procesor AVR i niektóre procesory z rodziny '51 (np. AT89S8252). Procesory można programować zarówno w podstawkach wyposażonych w niezbędny rezonator kwarcowy, jak i w systemie, za pomocą specjalnego złącza ISP. Ponieważ nie wszystkie układy bazujące na procesorach AVR posiadają takie złącza, pozwoliłem sobie zaprojektować specjalne konektory umożliwiające programowanie w systemie bez konieczności dolutowywania przewodów, ani dołączania ich za pomocą chwytaków. Rozmieszczenie wyprowadzeń tych złączy (CON2 na schemacie programatora i CON8 na schemacie minikomputera) jest zgodne ze standardem zalecanym przez firmę ATMEL.

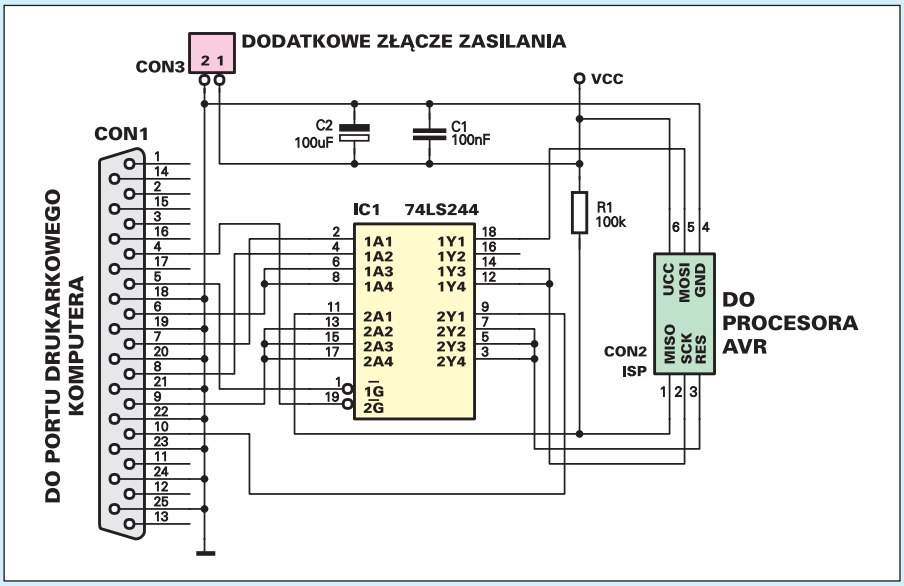
Dla szczególnie dociekliwych Czytelników podaję teraz uproszczony algorytm programowania procesorów AVR. Ci spośród Was, których zagłębianie się w teoretyczne podstawy działania procesorów zbytnio nie interesuje, mogą spokojnie pominąć ten fragment artykułu.

Aby zaprogramować pamięci procesora, programator musi wykonać następujące czynności:

1. Podczas włączania zasilania wymusić stan niski na wejściach RESET! i SCK procesora. Nie wszystkie programatory (w tym opisywany) są w stanie wykonać tę czynność i w taki przypadku konieczne jest, po wymuszeniu stanu niskiego na wejściu SCK, podanie na wejście RESET! dodatniego impulsu o czasie trwania dwóch cykli zegarowych.

2. Po upływie co najmniej 20 ms programator musi wysłać do procesora instrukcję zezwolenia na programowanie. Składnia tej i innych instrukcji podana jest w tabeli poniżej.

Rys. 4



Komentarz:
a - wyższe bity adresu
b - niższe bity adresu
H=0 - niższy bajt, H=1 - wyższy bajt
o - odczyt danych
i - zapis danych
x - bez znaczenia
A - bit zabezpieczający 1
B - bit zabezpieczający 2

Instrukcja	Format instrukcji				Działanie
	Bajt 1	Bajt 2	Bajt 3	Bajt 4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Zezwolenie na programowanie
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Kasowanie obu pamięci
Read Program Memory	0010 H000	xxxx xx**aa**	**bbbb bbbb**	**oooo oooo**	Odczyt górnej lub dolnej (H) części danych spod adresu **a:b**
Write Program Memory	0100 H000	xxxx xx**aa**	**bbbb bbbb**	**iiii iiii**	Zapis górnej lub dolnej (H) części danych spod adresu **a:b**
Read EEPROM Memory	1010 0000	xxxx xxxx	x**bbb bbbb**	oooo oooo	Odczyt z pamięci danych spod adresu **b**
Write EEPROM Memory	1100 0000	xxxx xxxx	x**bbb bbbb**	**iiii iiii**	Zapis do pamięci danych pod adres **b**
Write Lock Bits	1010 1100	111x x**AB**x	xxxx xxxx	xxxx xxxx	Zapis bitów zabezpieczających A i B
Read Signature Bits	0011 0000	xxxx xxxx	xxxx xx**bb**	oooo oooo	Odczyt typu układu **o** spod adresu **b**

3. Kolejną czynnością będzie sprawdzenie poprawności transmisji. Po wysłaniu przez programator drugiego bajtu instrukcji Programming Enable, procesor powinien odpowiedzieć „odesłaniem" do programatora wartości tego bajtu. Jeżeli tak się stanie, to należy uznać, że transmisja jest prawidłowa i przystąpić do wykonywania kolejnych instrukcji. Jeżeli jednak programator nie otrzymał „echa" od procesora, to należy powtórzyć próby nawiązania transmisji. Brak „echa" po 32 próbie świadczy o niemożności zsynchronizowania układów.
4. Po nawiązaniu transmisji programator powinien wysyłać do procesora kolejne instrukcje, przewidziane dla aktualnie wykonywanego zadania. Możliwe jest zaprogramowanie zarówno pamięci danych, jak i programu, odczyt ich zawartości oraz zabezpieczenie pamięci programu za pomocą dwóch bitów zabezpieczających.

Po zakończeniu programowania ustawienie stanu wysokiego na wejściu RESET procesora umożliwia jego poprawną pracę.

Jak widać, algorytm programowania poprzez złącze SPI jest dość skomplikowany. Na szczęście, nie musimy go znać na pamięć, ponieważ jest on automatycznie i bez naszego udział realizowany przez „mądrego" BASCOM-a.

Na **rysunku 5** została pokazana mozaika ścieżek płytki obwodu drukowanego, wykonanego na laminacie dwustronnym z metalizacją. Płytka programatorka została tak zwymiarowana, że po zmontowaniu i przylutowaniu do złącza DB25M mieści się „lekko na wcisk" w typowej obudowie DB25. Taki sposób obudowania programatora daje nam dużą wygodę w posługiwaniu się urządzeniem, które z pewnością będzie dość często podłączane i odłączane do komputera, na zmianę z kablem drukarkowym.

Płytkę montujemy dość nietypowo, ponieważ ze względu na drastyczne ograniczenie jej wymiarów musimy montować elementy po obydwu jej stronach. Montaż rozpoczynamy od połączenia płytki ze złączem DB-25.

Na dłuższej krawędzi płytki programatora, po obydwu jej stronach został umieszczony szereg punktów lutowniczych, rozmieszczonych identycznie do wyprowadzeń złącza DB-25. Płytkę wsuwamy „na wcisk" pomiędzy końcówki lutownicze złącza tak, aby wyprowadzenia konektora DB25 pokryły się dokładnie z punktami lutowniczymi. Podczas dopasowywania płytki do złącza właściwie nie można się pomylić, ponieważ na jednej stronie płytki mamy 12, a na drugiej 13 punktów lutowniczych. Po dokładnym dopasowaniu obu elementów do siebie lutujemy wyprowadzenia złącza, używając lutownicy o cienkim grocie.

Kolejną czynnością będzie wlutowanie w płytkę kondensatorów. Oba te elementy lutujemy od strony druku, oczywiście w przypadku płytki dwustronnej jest to strona umowna. Dla ułatwienia: w płytkach AVT maska lutownicza na stronie ścieżek jest zawsze zabarwiona na czerwonawo. Kondensatory montujemy na płask, równolegle do powierzchni płytki, a po przylutowaniu obcinamy jak najkrócej ich końcówki.

Układ IC1 montujemy „po bożemu" na stronie elementów płytki drukowanej. Odstępstwem od reguł jest rezygnacja ze stosowania podstawki, której użycie uniemożliwiłoby umieszczenie płytki w maleńkiej obudowie.

Montaż elektryczny kończymy na przylutowaniu do płytki jedynego rezystora i złącza CON2. Podwójny szereg kątowych goldpinów lutujemy tak, aby jego wolne końcówki znalazły się jak najbliżej płaszczyzny powierzchni płytki.

A więc, programator mamy już w zasadzie gotowy! Pozostaje tylko wyposażyć go w kabel łączący go z programowanym procesorem. Kabel ten wykonujemy z odcinka dziesięciożyłowego przewodu taśmowego o długości ok. 50 cm, zaciskając na jego końcach dwa wtyki 10-pinowe. Niestety, nie są produkowane takie wtyki o sześciu końcówkach i w naszym kablu cztery przewody pozostaną niewykorzystane.

Na zakończenie umieszczamy płytkę programatora wraz z dołączonym do niej kablem w przeznaczonej dla niej obudowie od wtyku DB-25 i skręcamy całość śrubkami.

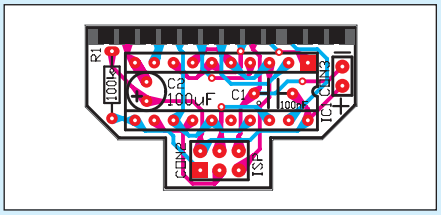
Uwieńczeniem naszej pracy będzie teraz połączenie programatora z minikomputerem i komputerem PC, na którym został zainstalowany pakiet BASCOM AVR. Pamiętajcie, że połączenia te musimy zawsze wykonywać przy wyłączonym zasilaniu obu urządzeń. Natomiast **przypadkowe, odwrotne połączenie przewodu prowadzącego od programatora do złącza ISP na płycie minikomputera nie grozi żadnymi przykrymi konsekwencjami! Złącze ISP zostało przez ATMEL-a tak sprytnie zaprojektowane, że po zmianie kierunku jego włączenia nie może dojść do uszkodzenia ani procesora, ani programatora i jedynym objawem będzie nieprawidłowe działanie całości**. W praktyce, odwrotne połączenie tego kabla będzie sygnalizowane komunikatem o niemożności zidentyfikowania dołączonego do programatora procesora.

Chciałbym jeszcze wyjaśnić sprawę widocznego na schemacie złącza CON3, o którego roli jak dotąd nie wspominaliśmy. Jest to złącze nie używane podczas pracy programatora z naszym minikomputerem, ponieważ programator jest tu zasilany z płyty minikomputera za pośrednictwem złącza ISP. Mam jednak nadzieję, że wykorzystacie zbudowany programator nie tylko do programowania PECEL-a, ale także podczas budowy innych układów. Może wtedy okazać się korzystne, aby testowany układ zasilany był z programatora (w każdym razie ja często stosuję tę metodę, wygodną podczas pracy nad kilkoma prototypami naraz). Do złącza CON3 należy wtedy doprowadzić napięcie o wartości +5VDC, którego idealnym źródłem może być np. game port komputera, a w ostateczności dowolny inny zasilacz o podanym napięciu i maksymalnym prądzie dostosowanym do wymagań uruchamianego układu.

Czy wiecie, moi Drodzy, do jakiego etapu pracy doszliśmy w tym momencie? Prawdę mówiąc, zakończyliśmy już budowę minikomputera PECEL i potrzebnego do jego programowania hardware! Pozostał nam jeszcze wprawdzie jeden kabelek do wykonania, ale możemy odłożyć tę pracę na później, do czasu kiedy zajmiemy się komunikacją nawiązywaną przez nasz minikomputer z „dużym" PC-tem za pośrednictwem portu RS232. Co zatem teraz zrobimy? Powinniśmy zająć się teraz opisem metod programistycznych służących ożywieniu PECEL-a, ale wiem, na co macie bardziej ochotę! Zapewne chcielibyście wypróbować programator i minikomputer i byłoby z mojej strony okrucieństwem, gdybym kazał Wam na to czekać. A zatem, do dzieła!

Wiecie co? Strasznie mi ten artykuł zaczyna się „rozłazić" i mam nadzieję, że

Rys. 5 Płytka drukowana programatora



połapiecie się w tych licznych dygresjach! Przecież zanim wykonamy pierwsze próby programowania procesora naszego minikomputera musimy coś zrobić z programem, który jest już umieszczony w jego pamięci. W kicie AVT-2550 dostarczany jest procesor z umieszczonym w jego pamięci EEPROM programem, który dla Was napisałem. Działanie tego programu zostanie opisane w dalszej części artykułu i nieskromnie mam nadzieję, że zyska on Wasze uznanie. Pamięć procesora nie została w jakikolwiek sposób zabezpieczona przed kopiowaniem, a listing programu został opublikowany na internetowej stronie Elektroniki dla Wszystkich (www.edw.com). Program stanowi zatem Waszą niepodzielną własność, ale co zrobić, jeżeli posiadamy tylko jeden, dostarczony w kicie procesor AT90S8535? Jakakolwiek próba programowania procesora spowoduje nieodwołalne zniszczenie zapisanego w jego pamięci „fabrycznego" programu. Arcydzieło sztuki programowania to chyba nie jest, ale może warto go zachować na przyszłość?

Na szczęście mamy już gotowy programator, który bynajmniej nie służy tylko do programowania procesora. Ile użytecznych funkcji może on jeszcze spełniać, dowiecie się w najbliższej przyszłości, a na razie, trochę wbrew logice, zajmijmy się nie programowaniem, ale odczytywaniem programu już zapisanego w pamięci EEPROM procesora.

O instalacji i ogólnym konfigurowaniu pakietu BASCOM AVR nie będę pisał, ponieważ praktycznie nie różnią się one od obsługi znanego już Wam pakietu BASCOM 8051. Wspomnijmy tylko o konfigurowaniu programatora, ponieważ nie mieliśmy z tym jeszcze do czynienia.

Po uruchomieniu BASCOM-a AVR klikamy na pasek OPTIONS i z rozwiniętego menu wybieramy opcję PROGRAMMER. Ukaże się nam wtedy panel pokazany na **rysunku 6**. W okienku PROGRAMMER wybieramy teraz **typ programatora, którym musi być STK200/STK300 Programmer**. Następnie zamykamy okienko i naciskamy klawisz F4, co owocuje pojawieniem się okienka programatora pokazanego na **rysunku 7**. Na wszelki wypadek naciskamy jeszcze na przycisk CHIP, a następnie IDENTIFY. Po tym zabiegu w małym okienku obok napisu CHIP powinien pokazać się napis informujący o typie zidentyfikowanego procesora, czyli w naszym przypadku AT90S8535.

Może się jednak zdarzyć, że programator nie będzie w stanie zidentyfikować typu procesora i na ekranie ukaże się mało sympatyczny napis widoczny na **rysunku 8**. Jeżeli jesteśmy całkowicie pewni, że montaż minikomputera i programatora przeprowadziliśmy poprawnie, to zapewne przyczyną jest nieprawidłowe podłączenie kabla łączącego programator z minikomputerem. Po sprawdzeniu tego połączenia i ewentualnym odwróceniu wtyku o 180 stopni wszystko powinno zacząć działać normalnie.

Zajmijmy się teraz zachowaniem dla potomności programu zapisanego w EEPROM-ie dostarczonego w kicie procesora. Po identyfikacji typu procesora klikamy na pasek CHIP, a następnie wybieramy opcję READ CHIPCODE INTO BUFFER (załaduj kod zawarty w pamięci procesora do bufora). W tym momencie rozpocznie się proces odczytywania zawartości pamięci EEPROM, który niestety potrwa chwilę, no powiedzmy dłuższą chwilę. Tak to już jest: zawsze „coś za coś" i za liczne udogodnienia związane z programowaniem ISP płacimy zwiększonym czasem trwania szeregowej transmisji danych. Ręczę jednak, że to się Wam opłaci!

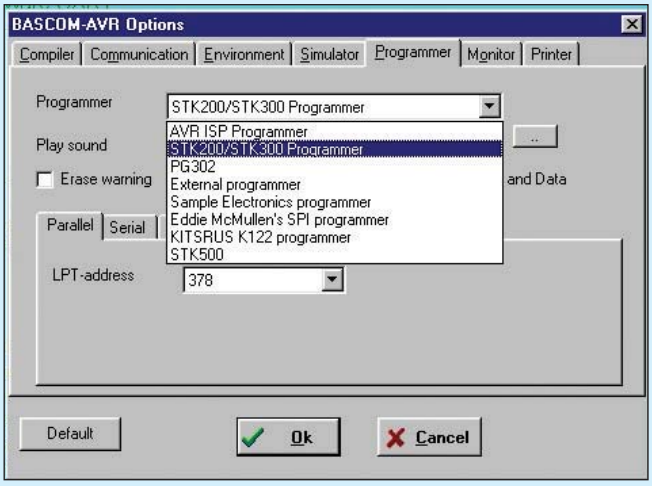
Mam nadzieję, że programator uporał się już z mozolnym odczytywaniem programu z pamięci procesora. A zatem, następną czynnością będzie zapisanie tego programu na dysku, pod dowolną nazwą i w dowolnym, wskazanym katalogu. W tym celu klikamy na pasek BUFFER, wybieramy opcję SAVE TO FILE i podajemy nazwę pliku, w którym ma być zapisany program w formacie binarnym (**rysunek 9**). Proponuję wykonać przynajmniej jedną kopię zapasową tego pliku i zapisać ją w innym katalogu, niż oryginał.

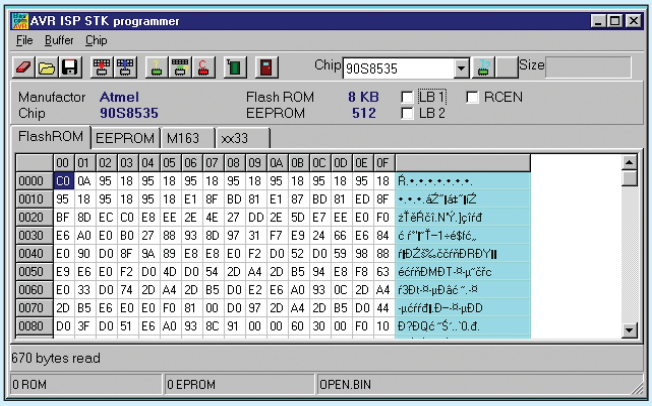
Nadeszła wreszcie pora, aby sprawdzić działanie zbudowanego układu w jego podstawowej, ale nie jedynej funkcji, jaką jest wprowadzanie programu do pamięci EEPROM minikomputera PECEL. Ponieważ niewiele jeszcze wiemy o programowaniu procesora AT90S8535, wykonamy tylko proste testy, wykorzystujące wiedzę nabytą podczas kursu BASCOM College. No, może pokażę Wam coś więcej...

Zanim napiszemy nasz pierwszy program na minikomputer PECEL, musimy jeszcze dokończyć konfigurowanie programatora. W jego okienku konfiguracyjnym zaznaczamy dodatkowo opcje AUTO FLASH i AUTO VERIFY, tak jak pokazano na **rysunku 10**.

Zacznijmy od czegoś bardzo prostego, pamiętając że mamy tylko przetestować programator, a na naukę programowania PECEL-a przyjdzie czas trochę później. Napiszmy zatem:

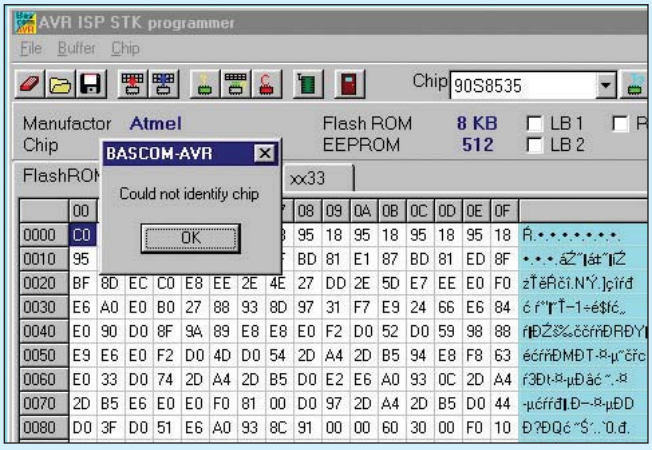
Rys. 6

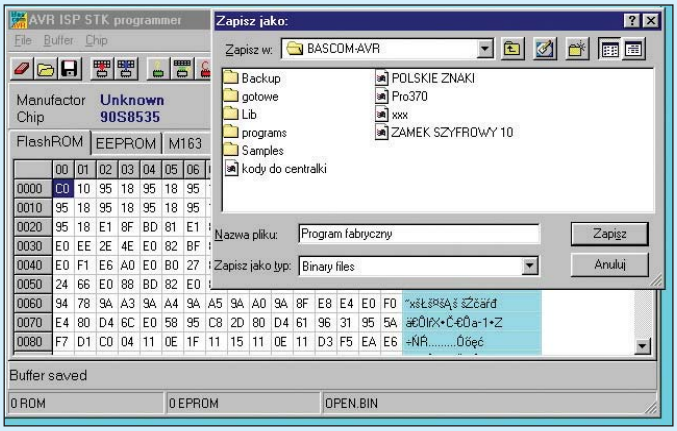




Rys. 7

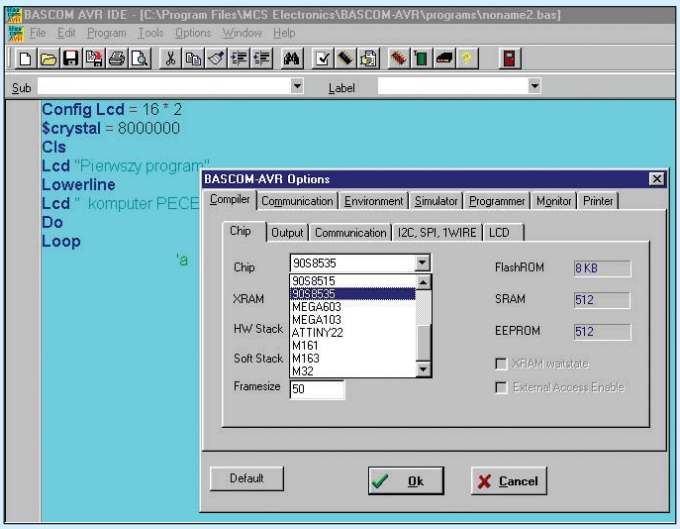
Rys. 8





Rys. 9

Rys. 10



'Listing 1

```
Config Lcd = 16 * 2
$crystal = 8000000
Cls
Lcd "Pierwszy program"
Lowerline
Lcd "komputer PECEL"
Do
Loop
```

Przed kompilacją tego programiku musimy jeszcze poinstruować kompilator, jakiego typu procesora będziemy używać. Otwieramy zatem okienko OPTIONS, a następnie COMPILER i CHIP (**rysunek 11**) i zaznaczamy procesor typu AT90S8535.

Naciskamy teraz „magiczny" klawisz F7. Dlaczego nadałem mu taki przydomek? Ano dlatego, że po jego naciśnięciu napisany przez nas program został nie tylko skompilowany, ale natychmiast umieszczony w pamięci procesora. Procesor został następnie zresetowany, a efekt działania pierwszego programu na PECEL-a możemy zobaczyć na jego wyświetlaczu alfanumerycznym. Nie musieliśmy wyjmować kosztownego i łatwego do uszkodzenia procesora z podstawki, wkładać go do programatora, a następnie ponownie umieszczać w minikomputerze. W tym, między innymi, tkwi siła programowania ISP połączona z fantastycznymi możliwościami BASCOM-a! Fajne, prawda?

Najwyższy czas, aby rozpocząć systematyczną naukę programowania minikomputera PECEL. Ale nawet mnie samego korci, aby awansem pokazać Wam jakiś „fajerwerk", jakiś spektakularny przykład możliwości naszych nowych urządzeń. Wspomniałem uprzednio, że zbudowany przed chwilą programator może służyć do wielu celów, często nie bardzo związanych z samym procesem programowania. Może pamiętacie z kursu BASCOM College lub z własnych doświadczeń, w jaki sposób zapisywaliśmy dane w zewnętrznych pamięciach danych EEPROM? Było z tym trochę problemów, trzeba było napisać kilkanaście linijek programu, nie mówiąc o konieczności dodawania do systemu dodatkowego układu – zewnętrznej pamięci danych EEPROM. No to popatrzcie, jak to wygląda w naszym minikomputerze wspartym siłą BASCOM-a!

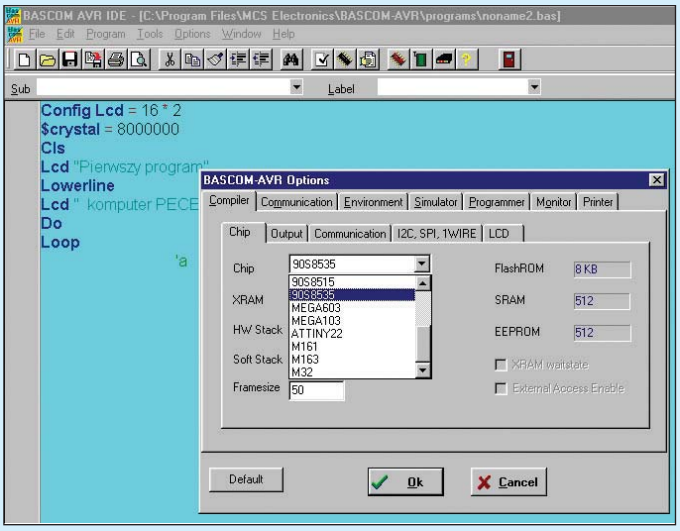
Pisząc ostatnie zdanie zauważyłem, że narobiło się trochę bałaganu w stosowanym w artykule nazewnictwie i że początkujący Koledzy mogą mieć z tym trochę kłopotu. Procesor AT90S8535 posiada aż trzy rodzaje pamięci i musimy dokładnie uprzytomnić sobie, do czego każda z nich służy.

1. Pamięć **programu EEPROM** służy do zapisywania treści programu sterującego pracą procesora. Jej pojemność wynosi 8kB i może być programowana wyłącznie za pomocą zewnętrznego programatora. Jakiekolwiek zmiany w jej zawartości bez stosowania programatora są niemożliwe. Pamięć programu może być przeprogramowywana do 1000 razy.

2. Pamięć **danych EEPROM** służy do zapisywania tych informacji, które nie mogą być utracone po wyłączeniu zasilania. Pamięć ta programowana jest przez odpowiednie polecenia obsługującego procesor programu. Istnieje także możliwość zaprogramowania pamięci danych EEPROM za pomocą zbudowanego przed chwilą programatora, a także odczytania jej zawartości. Pamięć danych EEPROM może być przeprogramowywana do 100 000 razy.

3. Pamięć **danych RAM** służy do chwilowego przechowywania danych, a jej zawartość jest bezpowrotnie tracona po wyłączeniu zasilania.

Rys. 11

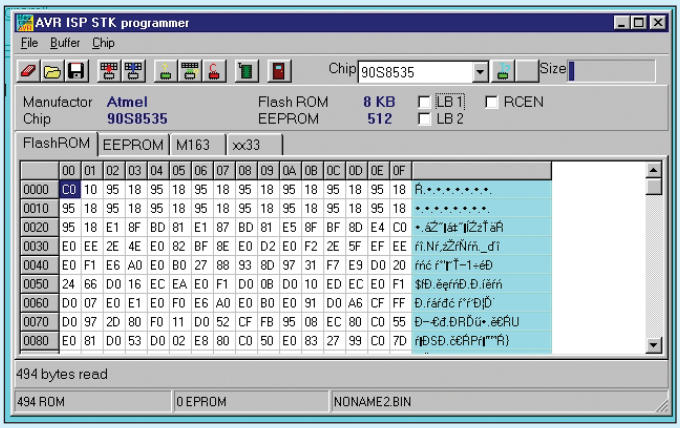


Do napisanego uprzednio programu dopiszmy trzy linijki, tak aby całość wyglądała tak, jak na listingu 2 (dodatkowe linie zaznaczono pogrubionym drukiem). Nie będziemy na razie tłumaczyć sobie znaczenia nowych poleceń i wspomnę tylko, że dodatkowym zadaniem programu jest teraz zapisanie w wewnętrznej pamięci danych EEPROM pod adresem 1 wartości zmiennej X, czyli 214.

'Listing 2
```
Config Lcd = 16 * 2
$crystal = 8000000
Dim X As Byte
Cls
Lcd "Pierwszy program"
Lowerline
Lcd "komputer PECEL"
X = 214
Writeeeprom X , 1
Do
Loop
```

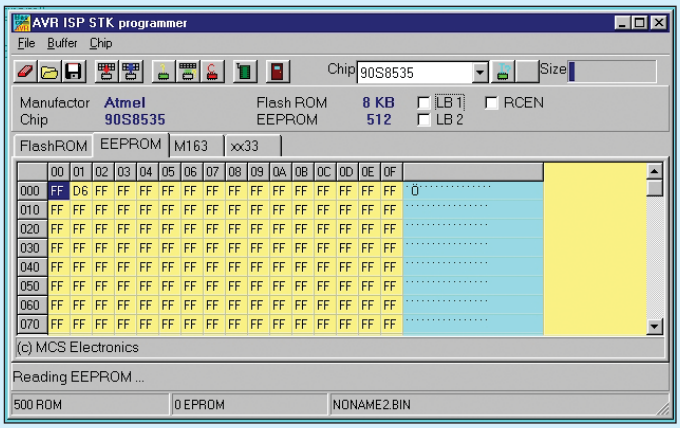
Ponownie naciskamy magiczny klawisz i... właściwie nic nowego się nie stało. Napis został wyświetlony, ale czy wartość zmiennej rzeczywiście znalazła się w pamięci? No, to popatrzcie teraz, jak wygodne narzędzia dostaliście do ręki i jak w przyszłości ułatwią nam one testowanie napisanych programów. Rzeczywiście, kontrolowanie, czy dane są zapisywane i czy lokowane są pod takim adresem, pod jakim byśmy chcieli, nie jest sprawą prostą. Podczas posługiwania się „normalnym“ oprogramowaniem znalezienie ewentualnych błędów może „trochę“ potrwać i kosztować „trochę“ nerwów. Przecież pamięci danych EEPROM nie można zobaczyć! Nie można? W BASCOM-ie wszystko można!

Zmieńmy teraz trochę konfigurację BASCOM-a, usuwając zaznaczenie opcji AUTO FLASH w okienku konfiguracyjnym programatora. Naciśnijmy następnie klawisz F4, co zaowocuje pojawieniem się na ekranie okienka programatora. No i co w tym nowego? Popatrzcie na **rysunek 12**: mniej więcej w jednej trzeciej wysokości okienka programatora znajdują się dodatkowe przyciski, a wszystko wskazuje, że w obecnej chwili aktywny jest pierwszy z nich, oznaczony jako FLASHROM. Naciśnijmy zatem drugi z przycisków, ten, na którym widnieje napis EEPROM, słusznie przypuszczając, że może on mieć coś wspólnego z wewnętrzną nieulotną pamięcią danych EEPROM.



Rys. 12

Rys. 13



Rzeczywiście, wydarzyło się coś nowego: na ekranie pojawiła się nowa tabelka, tylko że w niej zapisane są same wartości FF(HEX), czyli dziesiętnie 255. Do czasu! Powtórzmy teraz operację, którą wykonywaliśmy podczas kopiowania „fabrycznego“ programu PECEL-a, czyli klikamy na pasek CHIP, a następnie wybieramy opcję READ CHIPCODE INTO BUFFER. Tylko że tym razem do bufora ładowana będzie nie zwartość pamięci programu, ale interesująca nas pamięć danych EEPROM!

Efekt naszych poczynań jest widoczny na **rysunku 13**. No i co? Nie można zobaczyć pamięci danych EEPROM? Sprawdźmy jeszcze: pod adresem 1 widoczna jest tam wartość D6h, zapisana w formacie heksadecymalnym. Po przeliczeniu na format dziesiętny mamy: $D6_{(HEX)} = 214_{(DEC)}$.

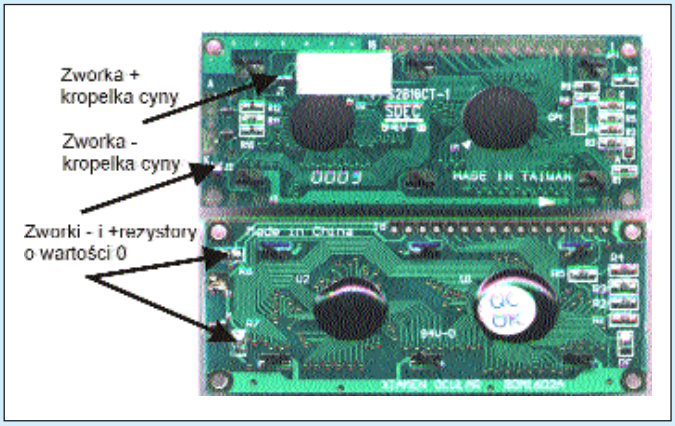
Chciałbym jeszcze na chwilę powrócić do charakterystyk różnych pamięci, jakimi dysponuje procesor AT90S8535, serce komputera PECEL. Zgodnie z danymi zawartymi w karcie katalogowej tego układu podałem, że pamięć programu może być zapisywana do 1000 razy, a pamięć danych EEPROM do 100000 razy. Są to liczby ogromne i trudno chyba obawiać się ich przekroczenia. Jednak w przypadku, kiedy PECEL byłby używany przez grupę użytkowników, np. na zajęciach w Technikum Elektronicznym, można obawiać się przekroczenia liczby dozwolonych programowań pamięci programu. Z doświadczenia jednak wiem, że dane podane przez producenta zostały obliczone mocno „na wyrost“, najprawdopodobniej z uwzględnieniem najbardziej krytycznych warunków pracy procesora. Nie testowałem nigdy, jaką maksymalną liczbę cykli zapisu może wytrzymać pamięć programu, ale dokonałem barbarzyńskiego eksperymentu z pamięcią danych EEPROM. Napisałem program, którego jedynym zadaniem było nieustanne zapisywanie całego obszaru tej pamięci coraz to nowymi danymi. Każda operacja zapisu była zliczana a wynik przekazywany do komputera. I wicie, co się zdarzyło? Po 324 567 cyklach zapisu dałem sobie spokój z dalszym prowadzeniem eksperymentu, uznając procesor ATMEL-a za produkt najwyższej klasy, a dane podawane przez tę firmę za więcej niż wiarygodne.

Musimy jednak skończyć z tym chaotycznym działaniem i licznymi dygresjami. Rozpoczynamy systematyczną naukę programowania minikomputera PECEL, a tym samym wszystkich procesorów AVR, poszerzając przy okazji nasze wiadomości o chipach z rodziny ’51.

Zbigniew Raabe
zbigniew.raabe@edw.com.pl

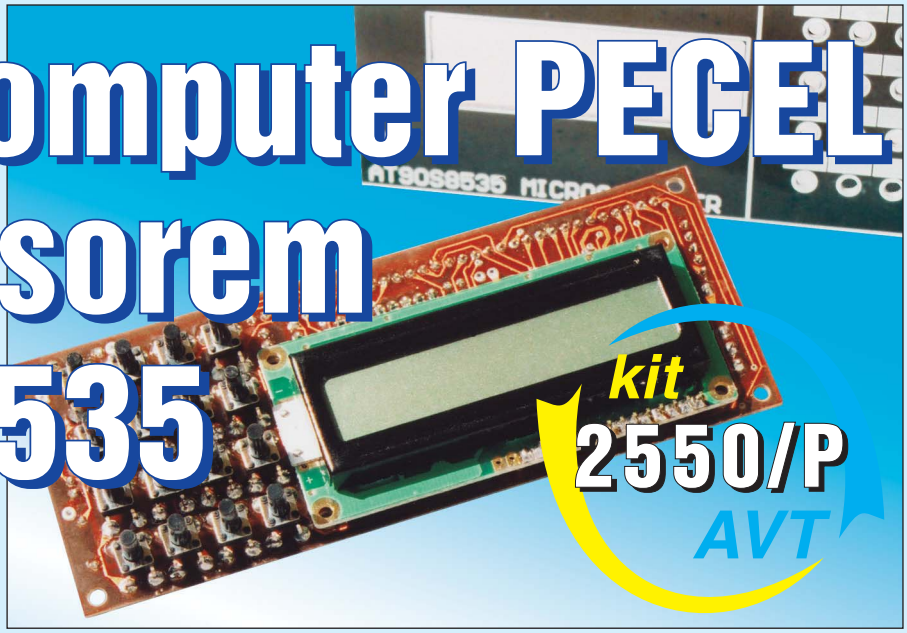
Chciałbym przy okazji rozwiać pewne wątpliwości, które jak wiem z listów e-mailowych nurtują od dawna Czytelników EdW i EP. Ponieważ na podobne trudności mogą natrafić także użytkownicy minikomputera PECEL, chciałbym do końca wyjaśnić sprawę podświetlanych wyświetlaczy LCD.

W ofercie handlowej AVT znajdują się obecnie relatywnie tanie wyświetlacze 16*2 i 16*1 z podświetlaniem. Takie wyświetlacze znajdować się będą także w kicie zawierającym części do budowy naszego minikomputera. Problem, na jaki napotkali Czytelnicy polega na... niemożności wyłączenia podświetlania w tych wyświetlaczach. Rzeczywiście, wyświetlacze te skonstruowane są trochę niewygodnie i sterowanie podświetlaniem wymaga w nich pewnych, zresztą drobnych przeróbek. Popatrzcie na rysunek, na którym przedstawione zostały płytki dwóch najpopularniejszych wyświetlaczy LCD 16*2 i 16*1. Strzałkami oznaczono tam elementy, których wlutowanie przesądziło o permanentnym włączeniu podświetlania! Elementami tymi mogą być "niby rezystory SMD", czyli po prostu zworki lub... najzwyklejsze kropelki cyny. Chcąc wyłączyć wyświetlanie, musimy usunąć jedną ze zworek. Nie sądzę jednak, aby komukolwiek zależało na usunięciu podświetlania na stałe. Najlepszym rozwiązaniem byłoby zapewnienie sobie możliwości włączania i wyłączania podświetlania w zależności od aktualnych potrzeb. Jest to możliwość tym bardziej atrakcyjna, że będziemy mogli dokonywać tych operacji programowo, a nawet płynnie regulować jasność świecenia wyświetlacza. A zatem, musimy najpierw zlokalizować tę zworkę, która odcina zasilanie podświetlania wyświetlacza od strony minusa zasilania. Czynność tę najprościej wykonać za pomocą omomierza, jeszcze przed wylutowywaniem jakichkolwiek elementów z płytki wyświetlacza. Następnie wylutowujemy zlokalizowaną zworkę i podłączamy prowizorycznie zasilanie do samego tylko wyświetlacza (+5VDC - nóżka 2, masa - nóżka 1). Po włączeniu zasilania wyświetlacz nie powinien dawać żadnych "oznak życia", co w tym momencie jest zjawiskiem prawidłowym. Spróbujmy teraz zewrzeć wyprowadzenie 15 wyświetlacza z masą, co powinno spowodować włączenie podświetlania. Jeżeli tak się nie stanie, to może to oznaczać, że wylutowaliśmy niewłaściwą zworkę. Jeżeli jednak podświetlanie włączyło się, to możemy przejść do dalszego etapu montażu komputerka.



Zbigniew Raabe
zbigniew.raabe@edw.com.pl

Mikrokomputer PECEL z procesorem AT90S8535



Część 3

Od dnia dzisiejszego zestaw części do budowy minikomputera PECEL będzie dostarczany z zaprogramowanym wstępnie procesorem. Pierwsza wersja Beta programu realizuje funkcję najdokładniejszego zegara Świata, pracującego z precyzją jednej sekundy na pięć milionów lat!

Program PECEL Ver. 1.0.0 Beta jest wspólną własnością Autora i Czytelników Elektroniki dla Wszystkich. Jako taki będzie poddawany stałym modyfikacjom i ulepszeniom. Wszystko wskazuje też na to, że w najbliższym czasie mogą powstać zupełnie nowe wersje tego programu, różniące się znacznie od pierwowzoru i pełniące zupełnie nowe funkcje. Mogłoby to spowodować sytuację, w której nabywcy pierwszej partii kitów mogliby mimowolnie zostać skrzywdzeni, posiadając pierwotną, najmniej doskonałą wersję programu. Aby uniknąć takiej sytuacji kody źródłowe WSZYSTKICH kolejnych wersji programu PECEL będą zamieszczane na stronie internetowej Elektroniki dla Wszystkich.

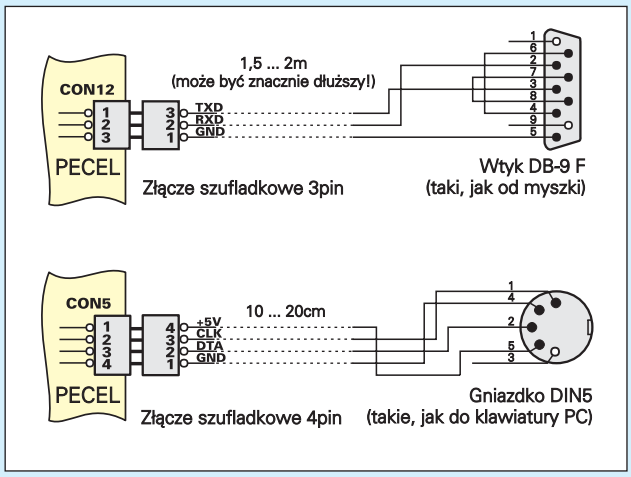
Jednocześnie zapraszam wszystkich Czytelników EdW do współpracy w tworzeniu oprogramowania dla PECEL-a.

Mam do Was teraz jedną, w właściwie nawet dwie prośby. Chodzi mi o wykonanie jeszcze dwóch prostych kabelków: jeden z nich ma posłużyć do połączenia PECEL-a z portem szeregowym komputera, a drugi umożliwi wykorzystywanie typowej klawiatury PC AT do wprowadzania danych do naszego minikomputera. Schematy połączeń obydwu kabelków pokazane są na **rysunku 14**. Do wykonania pierwszego kabla potrzebować będziemy odcinka przewodu trójżyłowego praktycznie dowolnego typu. Nie musi to być (ale może) kabel ekranowany. Także długość tego przewodu nie jest praktycznie niczym ograniczona i powinna być dostosowana do aktualnych potrzeb. Sądzę, że najlepszy będzie przewód o długości 1,5 ... 2mb. Z przylutowaniem do przewodów złącza DB9 i gniazdka DIN5 nie będziemy mieli z pewnością najmniejszych problemów. Nieco inaczej wygląda jednak sprawa z dołączeniem przewodów do PECEL-a. Najprostszą metodą byłoby ich przylutowanie do złącz CON12 i CON5. Jednak takie rozwiązanie utrudniłoby szybką zamianę konfiguracji systemu, szczególnie w przypadku „zablokowania" przylutowanymi przewodami złącza magistrali I^2C. Polecałbym Wam inną, wielokrotnie sprawdzoną metodę, polegającą na zakończeniu przewodów odpowiednio przyciętymi kawałkami złącz tzw. szufladkowych, czyli „żeńskich" odpowiedników konektorów goldpin. Uzyskamy w ten sposób możliwość bezproblemowego odłączania przewodów od płyty głównej minikomputera, co może mieć szczególne znaczenie podczas emula.... no, tak o mało się nie wygadałem i przedwcześnie nie zdradziłem przygotowanej dla Was **NIESPODZIANKI**.

Przygotowanie pierwszego z przewodów, którego zadaniem będzie połączenie PECEL-a z portem RS-232 komputera PC możemy sobie znacznie ułatwić jeżeli dysponujemy przewodem od uszkodzonej, że nie powiem „zdechłej" myszki. Odpadnie nam wtedy konieczność lutowania złącza DB25, a potrzebne nam przewody będziemy mogli zidentyfikować za pomocą omomierza.

Zastanawiałem się, od czego rozpocząć opisywanie metod programistycznych, które posłużą do tchnięcia życia w nasz minikomputer. Początkowo miałem zamiar rozpocząć od najważniejszych poleceń i funkcji BASCOM-a, których **nie omawialiśmy**

Rys. 14



podczas kursu BASCOM College. Jednak doszedłem do wniosku, że tak właściwie to nie ma „ważnych" i „mniej ważnych" elementów języka programowania: każdy fragment jego składni może okazać się w pewnych sytuacjach najważniejszy. Dlatego też przyjąłem inny porządek pisania tego artykułu: mam zamiar zredagować go tak, abyście jak najszybciej mogli zobaczyć pierwsze, efektowne rezultaty Waszej pracy i już w najbliższych chwilach ożywić minikomputer. Zacznijmy więc od tych funkcji BASCOM-a, które są niezbędne do realizowania komunikacji minikomputera z otoczeniem. Za to, że szybko zobaczycie rezultaty Waszej pracy i że będą one bardziej efektowne, niż się spodziewacie, ręczę głową, na której mi co nieco zależy!

Obsługa wyświetlacza LCD

Sposoby wysyłanie danych do wyświetlacza alfanumerycznego LCD zostały już wyczerpująco omówione podczas kursu BASCOM College. Dlatego też przypomnijmy sobie tylko ważniejsze polecenia służące wysyłaniu tekstów na ekran LCD, oraz sposób konfigurowania wyświetlacza, nieco odmienny od sposobu używanego podczas pracy z płytką testową AVT2500.

Pierwszą czynnością jaką będziemy musieli wykonać, zanim jeszcze spróbujemy wysłać cokolwiek na ekran jest poinstruowanie kompilatora o parametrach zastosowanego wyświetlacza i sposobu jego dołączenia do wyprowadzeń procesora. A zatem użyjmy dwóch, znanych już Wam poleceń:

CONFIG LCD = LCDtype [40 * 4,16 * 1, 16 * 2, 16 * 4, 16 * 4, 20 * 2, 20 * 4 lub 16 * 1a]
i
CONFIG LCDPIN = PIN , DB4= PN,DB5=PN, DB6=PN, DB7=PN, E=PN, RS=PN [gdzie PN oznacza numer pinu portu, do którego dołączone są wyprowadzenia wyświetlacza]

Z określeniem typu wyświetlacza nie będziemy mieli najmniejszego kłopotu. PECEL wyposażony jest w wyświetlacz dwuliniowy 2* 16 znaków. A zatem, piszemy: Config LCD = 16*2.

Trochę bardziej skomplikowane będzie poinstruowanie kompilatora do których pinów procesora zostały dołączone poszczególne wyprowadzenia wyświetlacza. Na naszej prostej płytce testowej AVT2500 wszystkie wyprowadzenia wyświetlacza były dołączone do jednego portu. W minikomputerze, z różnych względów nie było to możliwe i połączenia z wyświetlaczem prowadzą do różnych portów, pozornie chaotycznie. W dalszej części artykułu poinformuję Was, co wymusiło taki, a nie inny układ tych połączeń, a na razie popatrzmy uważnie na schemat i zredagujmy polecenie konfiguracyjne wyświetlacza LCD. Będzie ono miało postać: Config Lcdpin = Pin, Db4 = Portc.4, Db5 = Portc.5, Db6 = Portd.7, Db7 = Porta.7, E = Portc.3, Rs = Portc.2

A zatem, mamy już cztery pierwsze linijki, od których będzie zaczynał się każdy program napisany na nasz minikomputer.

```
$regfile = "8535def.dat"    'poinformowanie kompilatora o typie zastosowanego procesora
$crystal = 8000000          'poinformowanie kompilatora o częstotliwości oscylatora systemowego
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portd.7 , Db7 = Porta.7 , E = Portc.3 , Rs = Portc.2
```

Za chwilę dodamy do nich dalsze, ale najpierw musimy dobrze zapamiętać pewną cechę dialektu języka MCS BASIC stosowanego w pakiecie BASCOM AVR:

Pisząc program na procesor AVR poszczególne piny portów nazywamy PIN[port][numer pinu], czyli na przykład PINB.1, PIND.3 itd.
Podczas wysyłania danych na poszczególne piny używamy składni: PORT[numer portu][pin portu]
np. : SET PORTB.1 lub RESET PORTB.5.
Podczas odczytu stanu pinów portów używamy składni: PIN[port][numer pinu]
np. X= PINB.1 lub IF PIND.3 = 1 THEN .....

Polecenia służące obsłudze wyświetlacza alfanumerycznego były szczegółowo omówione podczas kursu BASCOM College. Przypomnijmy tylko najważniejsze z nich:
LCD [zmienna lub tekst] wysyła podaną wartość na ekran wyświetlacza
LOCATE [rząd, kolumna] ustawia kursor na wskazanej pozycji
CLS czyści ekran wyświetlacza
CURSOR ON / CURSOR OFF włącza i wyłącza wyświetlanie kursora
SHIFTLCD [RIGHT/LEFT, ilość pozycji] „przewija" napis na ekranie LCD

Obsługa klawiatury szesnastkowej komputera PECEL

Jak każdy szanujący się komputer także nasz PECEL wyposażony jest w klawiaturę. Nie jest to może klawiatura o możliwościach konsoli PC, ale do naszych celów będzie zupełnie wystarczająca. Nie zapominajmy, że w razie absolutnej konieczności będziemy mogli skorzystać także z klawiatury PC, o czym jeszcze będziemy mówić w dalszej części artykułu.

Klawiatura PECEL-a składa się z szesnastu klawiszy połączonych w matrycę czterech rzędów i czterech kolumn. Matryca została dołączona do portu B procesora, zajmując wszystkie jego 8 pinów. W tym miejscu chciałbym wyjaśnić jedną sprawę: to że dołączyliśmy klawiaturę do portu B nie oznacza bynajmniej, że nie będziemy mogli wykorzystywać jego wyprowadzeń do innych celów. W momencie, kiedy klawiatura nie jest skanowana wszystkie piny portu B „wiszą w powietrzu" i mogą być wykorzystane do sterowania dowolnymi układami. Musimy jedynie zwrócić uwagę, aby żaden z tych układów nie zwierał wejść portu B ani do masy ani do plusa zasilania w czasie korzystania z klawiatury.

No właśnie, w jaki sposób mamy dowiedzieć się czy i jaki klawisz został naciśnięty? Myślę, że wielu z Was już się domyśla: należy po prostu programowo cyklicznie ustawiać stan niski na kolejnych rzędach klawiatury i za każdym razem badać, czy któryś z pinów, do których dołączone są kolumny matrycy nie znalazł się w stanie niskim. Tak, jest to dobra metoda, a listing programu napisanego według tej zasady został przedstawiony na **rysunku 15** wraz z ... przekreślającym go znakiem! Taki program oczywiście by działał, ale po co mamy pisać tekst nie mieszczący się nawet na stronie ekranowej, jeżeli Mark już o wszystkim pomyślał? Do skanowania szesnastoznakowej klawiatury służy w MCS BASIC jedno proste polecenie:

Zmienna = GETKBD()

poprzedzone dyrektywą konfiguracyjną:

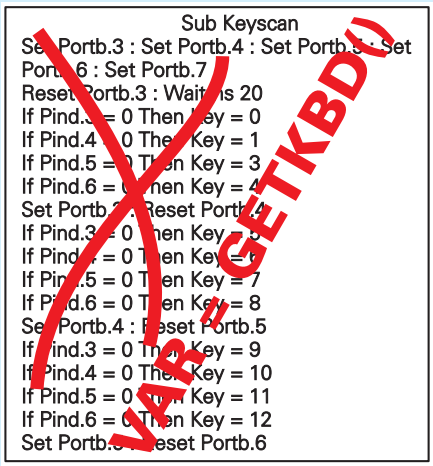
CONFIG KBD = PORT[numer portu]

po którego wydaniu klawiatura jest automatycznie przeszukiwana, a podana zmienna przyjmuje umowną wartość naciśniętego klawisza. Za chwilę wyjaśnimy sobie pojęcie „umowną", a na razie poproszę Was o zapamiętanie jednej własności polecenia GETKBD(), której przeoczenie mogłoby spowodować poważne komplikacje podczas pisania programu wykorzystującego to polecenie:

Jeżeli żaden klawisz nie został naciśnięty, to polecenie GETKBD() zwraca zawsze wartość 16

A zatem, wiemy już wszystko, co jest potrzebne do rozpoczęcie nauki obsługi klawiatury minikomputera. Napiszmy sobie zatem prosty programik demonstracyjny, którego zadaniem będzie jedynie doświadczalne potwierdzenie zdobytej przed chwilą wiedzy teoretycznej.

Rys. 15



```
$regfile = "8535def.dat"
$crystal = 8000000
Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Po-
rtc.5 , Db6 = Portd.7 , Db7 = Porta.7 , E = Po-
rtc.3 , Rs = Portc.2
Config Kbd = Portb
Dim Key As Byte
Cls
Lcd "Test klawiatury"
Lowerline
Lcd " szesnastkowej"
Do
Key = 16
Key = Getkbd()
If Key < 16 Then
                              Cls
                              Lcd Key
End If
Loop
```

Jeżeli skompilujemy ten program i zaprogramujemy nim procesor, to po każdym naciśnięciu klawisza na wyświetlaczu LCD będzie ukazywał się jego kod, czyli wartość całkowicie umowna. Ponadto z pewnością zauważyliście już coś niepokojącego, coś co w niektórych sytuacjach może spowodować, że nasza klawiatura będzie miała nieco egzotyczny rozkład przycisków. Otóż, okazało się że kody naciskanych kolejno klawiszy układają się w następujący sposób:

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

co w najmniejszym nawet stopniu nie odpowiada ani napisom na płycie czołowej minikomputera, ani ogólnie przyjętym zasadom konstruowania klawiatur numerycznych. Czyżbym popełnił jakiś błąd? Na szczęście wszystko jest w porządku. Przecież kody odbierane z klawiatury są wartościami umownymi i tylko od programu zależeć będzie, w jaki sposób będą interpretowane. Rozkład przycisków na klawiaturze został wymuszony podczas projektowania płytki obwodu drukowanego, na której ze względu na chęć obniżenia kosztów liczył się każdy milimetr kwadratowy. Nie przywiązywałem najmniejszej wagi do rozkładu klawiszy na płytce obwodu drukowanego ponieważ metodami programistycznymi można bez problemów zmienić kody poszczególnych klawiszy. Przeróbmy odrobinę napisany program:

```
' .....................
Do
Key = 16
Key = Getkbd()
If Key < 16 Then
                              Key = Lookup(key ,
Keyboard_decoding)
                              Cls
                                    Lcd Key
End If
Loop
' .....................
Keyboard_decoding:
Data 7 , 4 , 1 , 0 , 8 , 5 , 2 , 10 , 9 , 6 , 3 , 11 , 15
, 14 , 13 , 12
```

i natychmiast zauważymy, że kody klawiszy ułożyły się w następujący, całkowicie zgodny z napisami na płycie czołowej, sposób:

7	8	9	15
4	5	6	14
1	2	3	13
0	10	11	12

Najmilsza chwila poranka: dwóch komputerów pogadanka.

To, co za chwilę przeczytacie stanowi z pewnością najciekawszy fragment tej części artykułu opisującego metody programistyczne stosowane przy tworzeniu software dla minikomputera PECEL. Chciałbym wreszcie teraz poruszyć sprawę komunikacji pomiędzy układami mikroprocesorowymi, a komputerami klasy PC, a także pomiędzy dwoma minikomputerami. Zauważcie, że w tym momencie otwiera się przed nami zupełnie nowy obszar zastosowań naszego minikomputera. Może on być samodzielnym systemem mikroprocesorowym równie dobrze jak terminalem komputera, układem współpracującym ściśle z PC. Więcej, jak się w najbliższej przyszłości okaże, to komputer może być niekiedy terminalem PECEL-a, skwapliwie wykonując wysyłane przez minikomputerek rozkazy. Mamy przed sobą wręcz oszałamiające perspektywy: będziemy mogli budować np. przyrządy pomiarowe (pamiętajmy o ośmiu wejściach analogowych procesora ‘8535) mogące pracować jako samodzielne urządzenia, a także jako terminale przekazujące komputerowi PC dane do dalszej obróbki. Poruszymy szerzej ten temat w dalszych częściach artykułu, a jak na razie zapraszam Was do lektury EP 9/01 i 10/01, gdzie opisano właśnie konstrukcję wielofunkcyjnego miernika częstotliwości współpracującego z komputerem PC, zbudowanego także w oparciu o procesor ‘8535.

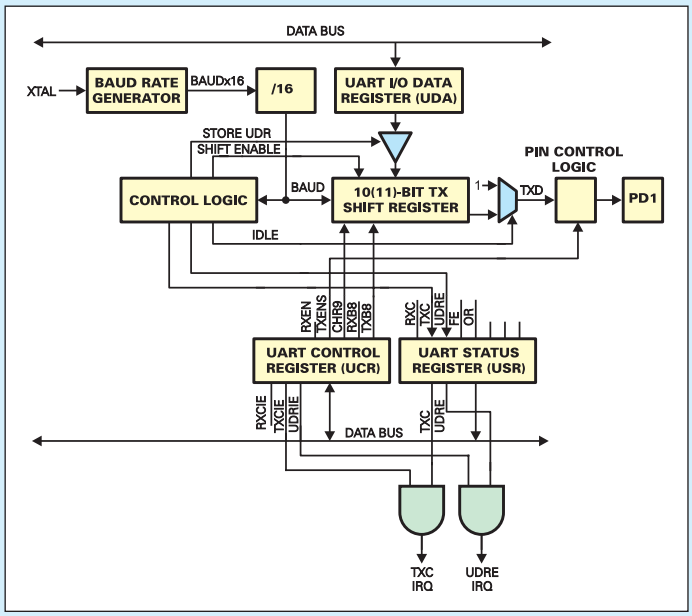
To jednak jeszcze nie wszystko: za chwilę czeka Was prawdziwa niespodzianka, i to taka, o jakiej pewnie nawet nie marzyliście! Za moment dostaniecie do ręki wyjątkowo potężne narzędzia programowe i sprzętowe, które mogą uczynić programowanie PECEL-a nie tylko dziecinnie łatwym zadaniem, ale także prawdziwą przyjemnością i relaksującą rozrywką. O opisanie procedur umożliwiających wymianę danych za pośrednictwem łącza RS232 dopominali się już Studenci BASCOM College. A więc proszę: macie co chcecie i jeszcze trochę więcej!

Jedną z największych zalet procesorów produkowanych przez firmę ATMEL jest wbudowany w strukturę tych układów sprzętowy UART, umożliwiający stosunkowo łatwą realizacje transmisji danych w standardzie RS232. Połączenie systemu mikroprocesorowego z komputerem pozwala na budowę najróżniejszego typu terminali do PC, aparatury pomiarowej z której dane można przekazywać i poddawać dalszej obróbce w komputerze. Magistrala komunikacyjna RS232 jest chyba najlepszym sposobem na połączenie ze sobą dwóch układów mikroprocesorowych i umożliwienie im „rozmowy” nawet na bardzo duże odległości.

Nie są to jednak jedyne zastosowania transmisji RS232 odbywającej się pomiędzy procesorem i komputerem. Za chwilę dowiemy się, jak bardzo ta możliwość może okazać się użyteczna podczas uruchamiania i testowania programów dla systemów mikroprocesorowych, które ... nawet nie będą nigdy wykorzystywać transmisji szeregowej podczas normalnej pracy.

Zarówno procesory ’51 produkcji ATMEL jak i prawie wszystkie (wyjątkami są: AT TINY22, AT90S2343 i AT90S2333) chipy AVR wyposażone są w sprzętowy układ UART (Universal Asynchronous Receiver and Transmitter), umożliwiający realizację transmisji RS232 na drodze sprzętowej. W artykule, który w tej chwili czytacie, przyjęliśmy zasadę podobną do reguł obowiązujących w BASCOM College: jak najmniej

Rys. 16



teorii: praktyka, praktyka i jeszcze raz praktyka! Dlatego też nie będziemy szczegółowo opisywać budowy UART i zadowolimy się jedynie pokazaniem na **rysunku 16** poglądowego schematu, przedstawiającego jego liczniki i rejestry. Koledzy pragnący pogłębić swoją wiedzę o UART znajdą wszelkie potrzebne dane w karcie katalogowej dowolnego procesora AVR (www.atmel.com), a my będziemy traktować ten układ o dość skomplikowanej budowie jako małą „czarną skrzynkę", która po prostu wykonuje wydawane w języku MCS BASIC polecenia.

Przekazywanie danych za pomocą łącza szeregowego jest w przypadku procesorów ATMEL-a szczególnie łatwa, a przy korzystaniu z pakietów BASCOM wręcz dziecinnie prosta. Oczywiście, zanim rozpoczniemy transmitowanie danych musimy przygotować odpowiednie środowisko sprzętowe, a następnie poinstruować kompilator o naszych zamierzeniach. Środowisko sprzętowe już posiadamy: minikomputer PECEL, komputer klasy PC oraz przygotowany przed chwilą przewód, z pomocą którego połączymy ze sobą obydwie maszyny. A zatem, bierzmy się za pisanie pierwszego programu.

Najważniejszą sprawą, jaką musimy załatwić przed rozpoczęciem pracy nad każdym programem wykorzystującym transmisję RS232 jest prawidłowe określenie szybkości przekazywania danych. Zaniedbanie tej czynności bądź przeprowadzenie jej w niewłaściwy sposób zawsze prowadzi do totalnej katastrofy czyli niemożności nawiązania kontaktu pomiędzy komputerami. Z listów e-mail od Czytelników wiem, że właśnie nieprawidłowe zadeklarowanie szybkości transmisji lub nie podanie jej w ogóle jest najczęstszą przyczyną problemów pojawiających się podczas uruchamiania programów wykorzystujących łącze RS232.

Szybkość transmisji danych określana jest za pomocą dyrektywy:

Kwarc	1000000Hz	4000000Hz	7372800Hz	8000000Hz	11059200Hz
Baudrate					
2400	**0.2%**	0,2%	0,0%	0,2%	0,0%
4800	**0,2%**	0,2%	0,0%	0,2%	0,0%
9600	7,5%	**0,2%**	0,0%	0,2%	0,0%
14400	7,8%	2,1%	**0,0%**	0,8%	0,0%
19200	7,8%	**0,2%**	0,0%	0,2%	0,0%
28800	7,8%	3,7%	**0,0%**	2,1%	**0,0%**
38400	22,9%	7,5%	**0,0%**	0,2%	0,0%
57600	7,8%	7,8%	**0,0%**	3,7%	**0,0%**
76800	22,9%	7,8%	**0,0%**	7,5%	**0,0%**
115200	84,3%	7,8%	**0,0%**	7,8%	**0,0%**

Tabela 1 Błąd częstotliwości w zależności od częstotliwości kwarcu

$Baud= X [2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 76800 lub 115200]

A więc, wydawało by się, że wszystko jest bardzo proste: ustawiamy po prostu największą prędkość transmisji i przystępujemy do pisania programu. No dobrze, możemy spróbować, napiszmy sobie najprostszy programik, którego zadaniem jest jedynie wysłanie prostego tekstu do komputera:

```
$crystal = 8000000
$baud = 115200
Print " PECEL wita Czytelników Elektroniki dla
Wszystkich!"
End
```

Wyjaśnieniem działania nowego polecenia PRINT zajmiemy się za chwilę, a teraz zastanówmy się, czy ten program ma choćby najmniejszą szansę na poprawne działanie. Z góry mogę Wam powiedzieć, że nie ma!

Wyłączmy teraz na chwilę opcję „PROGRAM AFTER COMPILE" z menu OPTIONS\ENVIRONMENT i skompilujmy napisany programik. Kompilacja programu, w którym BASCOM nie znalazł błędu składni przebiegła, oczywiście prawidłowo, co jednak nie oznacza że program będzie działa poprawnie. Kliknijmy teraz na przycisk PROGRAM, a następnie wybierzmy opcję SHOW RESULTS, co spowoduje otworzenie nowego okienka z całą kopalnią bezcennych informacji o naszym programie (**rysunek 17**). Później zajmiemy się bardziej szczegółowym ich opisem, a na razie zwróćmy uwagę tylko na trzy linijki wyświetlonego tekstu:

```
BAUD        : 115200 Baud
XTAL        : 8000000 Hz
BAUD error  : 7.84%
```

Okazuje się, że przy częstotliwości oscylatora systemowego równej 8MHz, a taki właśnie kwarc został dołączony do naszego PECEL-a błąd generacji częstotliwości zegarowej UART wynosi aż 7,8% co praktycznie uniemożliwia prawidłowe przeprowadzenie transmisji danych. Nie będziemy tu wdawać się w dość skomplikowane obliczenia i badać jaką częstotliwość zegarową UART możemy wygenerować przy częstotliwości zegara systemowego równej 8MHz. Nie obciążajmy się zbytnio teorią, zainteresowanych odsyłam do karty katalogowej dowolnego procesora AVR, a my posłużmy się teraz gotową tabelką, skopiowaną z takiej właśnie strony.

Z tabeli tej wynika niezbicie, że przy częstotliwości zegara systemowego wynoszącej 8MHz nie uda nam się wygenerować większej szybkości transmisji RS232 niż 38400, w ostateczności 57600Baud. Błąd generacji częstotliwości nie większy niż 4% pozwala jeszcze mieć nadzieję na prawidłową wymianę danych. Jednak jest to zabieg dość ryzykowny i lepiej pozostać przy mniejszej częstotliwości, np. 19200Baud.

A zatem przeróbmy trochę nasz program testowy, który będzie teraz wyglądał następująco:

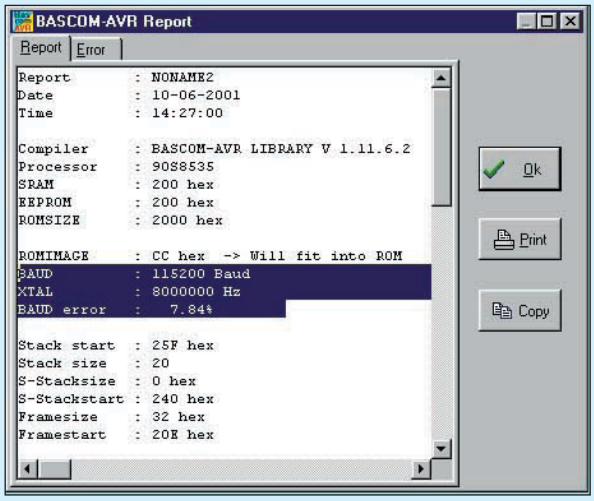
```
$crystal = 8000000
$baud = 19200
Do
Print " PECEL wita Czytelników Elektroniki dla
Wszystkich!"
Wait 1
Loop
End
```

Raport wygenerowany przez BASCOM-a wygląda teraz także zupełnie inaczej: możemy mieć całkowitą pewność, że transmisja danych będzie przebiegać poprawnie.

```
BAUD        : 19200 Baud
XTAL        : 8000000 Hz
BAUD error  :  0.16%
```

Nadszedł teraz doniosły moment przeprowadzenia pierwszego eksperymentu z transmisją danych z PECEL-a do komputera PC.

Rys. 17



Możemy już zaprogramować procesor i... zamiast podziwiać rezultaty naszej pracy wziąć się za konfigurowanie środowiska programowego odpowiedzialnego za porozumiewanie się z PECEL-em.

Bardzo ważne jest prawidłowe ustawienie szybkości transmisji w urządzeniu, z którym procesor ma nawiązać łączność. Takim urządzeniem najczęściej będzie monitor interfejsu szeregowego, najlepiej ten, który został wbudowany w pakiety BASCOM. Po raz kolejny możemy teraz przekonać się, jak wspaniałym zestawem narzędzi jest nasz BASCOM. W pakiecie tym zaszyte są bowiem wszystkie funkcje pozwalające nie tylko na monitorowanie portu RS232, ale i na dwukierunkowe przekazywani danych pomiędzy PC a innym urządzeniem wyposażonym w port komunikacyjny RS232. Monitor konfigurujemy po otwarciu okienka OPTIONS\COMMUNICATION, tak jak pokazano na **rysunku 18**. **Musimy także zawsze pamiętać, że po każdej zmianie szybkości transmisji w układzie, który ma współpracować z komputerem musimy zmienić także ustawienia monitora obsługującego tę transmisję. Uwaga ta dotyczy nie tylko monitora zawartego w pakiecie BASCOM, ale także wszystkich innych powszechnie stosowanych monitorów portów RS232**.

Oprócz szybkości transmisji musimy także określić, przez który port szeregowy ma się ona odbywać. Oczywiście, musi to być ten port, do którego nie jest podłączona myszka. W przypadku mojego komputera był to port COM1. Pozostałe parametry w okienku konfiguracyjnym pozostawiamy bez zmian, tak jak jest to widoczne na rysunku 18.

Większość współcześnie użytkowanych komputerów PC posiada „fabrycznie" zainstalowane dwa porty szeregowe: COM1 i COM2, i do jednego z nich jest na stałe dołączona myszka. Drugi port pozostaje najczęściej niewykorzystany i do niego właśnie dołączymy przewód transmitujący dane do i z PECEL-a. Jednak po uruchomieniu programu monitora może się zdarzyć, że np. myszka umieszczona została w porcie COM1 i na ten sam port został skonfigurowany monitor. Taka sytuacja prowadzi do natychmiastowego zawieszenia pracy myszy, a my mamy wtedy dwa wyjścia z sytuacji. Możemy przenieść myszkę do drugiego portu i ponownie uruchomić komputer, lub wykorzystując tylko klawiaturę skonfigurować monitor do śledzenia wolnego aktualnie portu.

Podsumujmy teraz wykonane czynności i ich następstwa:

1. W procesorze minikomputera PECEL znajduje się napisany przez nas programik wysyłający do komputera komunikat powitalny
2. Monitor portu szeregowego pakietu BASCOM został odpowiednio skonfigurowany, określona została szybkość transmisji numer wykorzystywanego do niej portu COM.

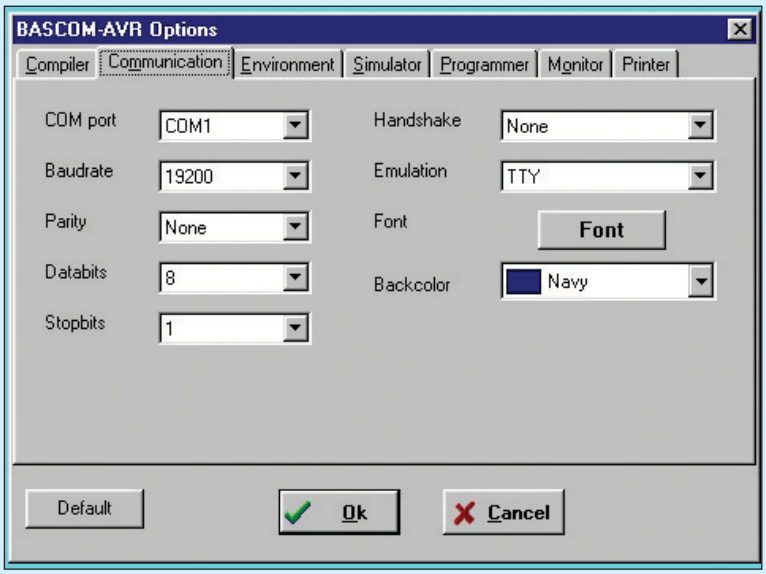
A zatem nadeszła dawno oczekiwana chwila! Klikamy na przycisk TOOLS i następnie wybieramy opcję TERMINAL EMULATOR, lub po prostu naciskamy kombinację klawiszy CTRL + T. Jeżeli wszystkie opisane uprzednio czynności wykonaliśmy poprawnie, to nasze oczy powinien ucieszyć widok pokazany na **rysunku 19**.

No i tak, Moi Drodzy, dokonaliśmy wielkiej rzeczy! Jak wielkiej, pokaże najbliższa przyszłość. W każdym razie jest to milowy krok na drodze do budowy inteligentnego terminala komputerowego o ogromnych możliwościach, jakim może stać się nasz PECEL.

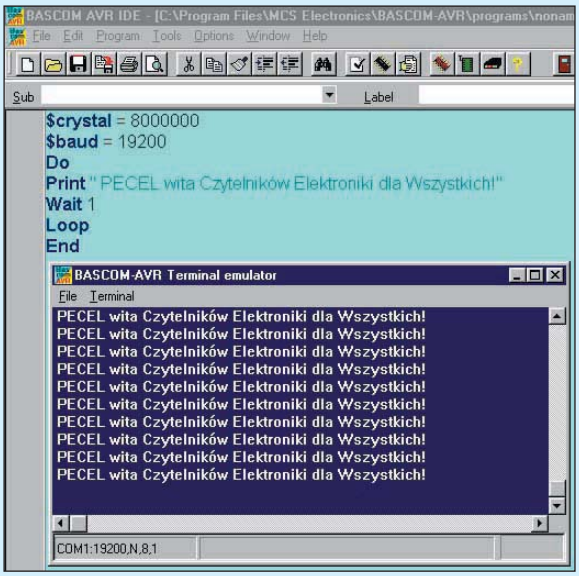
Ciąg dalszy w następnym numerze EdW

Zbigniew Raabe,
zbigniew.raabe@edw.com.pl

Rys. 18

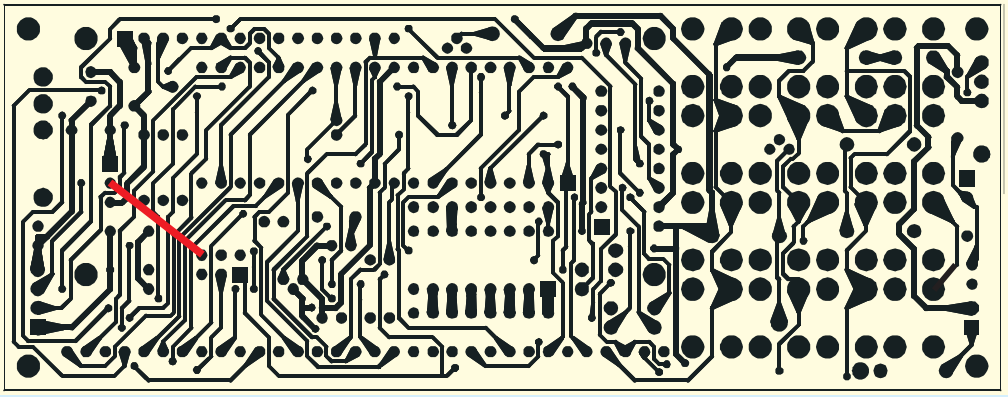


Rys. 19



UWAGA! BARDZO WAŻNE!

Kilka dni temu stwierdziliśmy, że w czasie procesu produkcyjnego jednej z serii płytek obwodów drukowanych do minikomputera PECEL powstała przerwa w obwodzie masy. Na szczęście wada ta jest bardzo łatwa do naprawienia: wystarczy połączyć za pomocą odcinaka przewodu dwa punkty: pin 6 złącza CON8 i pin 2 stabilizatora napięcia IC4. Sposób wykonania dodatkowego połączenia został pokazany na rysunku.



Mikrokomputer PECEL z procesorem AT90S8535

Część 4

Oczywiście, program przedstawiony w poprzednim odcinku ma tylko jedno zastosowanie: pokazanie że nawiązanie łączności pomiędzy dwoma komputerami jest możliwe i sprawdzenie poprawności konfiguracji używanych do transmisji narzędzi. Poza tym nie służy on do niczego. Aby jednak móc napisać bardziej rozbudowany program, musimy wreszcie zapoznać się choćby z podstawowymi poleceniami programowymi języka MCS BASIC służącymi przekazywaniu informacji poprzez łącze RS232.

Podstawowymi poleceniami języka MCS BASIC stosowanymi podczas wymiany danych poprzez interfejs RS232 są:

PRINT [zmienna, wartość, zmienna tekstowa lub tekst]

pozwalające wysłać do portu szeregowego komputera dowolną wartość, zmienną liczbową lub tekstową, oraz

INPUT [opcjonalny tekst zachęty], [zmienna liczbowa lub tekstowa]

które umożliwią „ręczne" przesłanie informacji poprzez port szeregowy do systemu mikroprocesorowego. Jeżeli polecenie INPUT zostanie zastosowane łącznie z tekstem zachęty podanym w cudzysłowie, to tekst ten zostanie wyświetlony na ekranie terminala komputera.

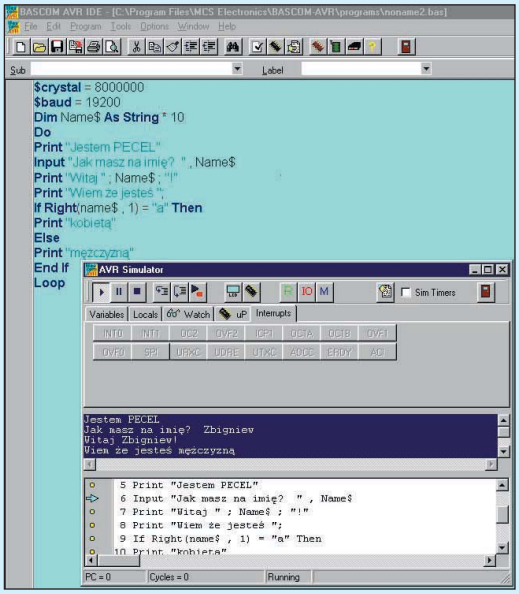
Czy jednak to drugie polecenie naprawdę działa? Aby to sprawdzić, napiszmy sobie prosty programik:

```
$crystal = 8000000
$baud = 19200
Dim Name$ As String * 10
Do
Print " Jestem PECEL"
Input "Jak masz na imię?  " , Name$
Print "Witaj " ; Name$ ; "!"
Print "Wiem że jesteś ";
If Right(name$ , 1) = "a" Then
Print "kobietą"
Else
Print "mężczyzną"
End If
Loop
```

Napisany program kompilujemy i na wszelki wypadek testujemy w symulacji programowej (**rysunek 20**). Uruchamianie emulatora programowego niczym się nie różni od podobnej operacji dokonywanej w środowisku BACOM-a 8051 i którą opisywaliśmy w BASCOM College. Jeżeli nie popełniliśmy błędu, to powinniśmy nawiązać z PECEL-em dwustronną łączność.

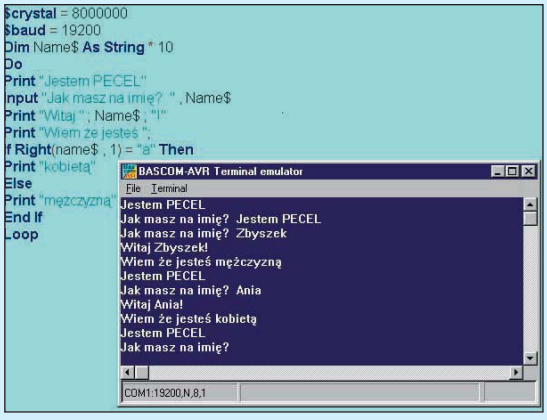
Aby się upewnić, czy nasz program działa poprawnie otwieramy okienko monitora portu RS232 i odpowiadamy na pytanie zadane przez PECEL-a. Tekst wprowadzamy z klawiatury komputera, a następnie potwierdzamy podanie imienia za pomocą klawisza ENTER (**rysunek 21**). Czytelnikom pozostawiam odpowiedź na pytanie, w jaki sposób procesor określa płeć rozmówcy i jakie trzy męskie imiona mogą wprowadzić go w błąd.

No tak, przełomowy moment mamy już za sobą: PECEL potrafi porozumiewać się z PECET-em! Pozostaje jednak otwarte pytanie, do czego to można wykorzystać? Przecież chyba nie do pisania prostych, zabawkowych programików? Otóż, z pewnością komunikacja pomiędzy dwoma komputerami nie będzie wykorzystywana tylko do błahych spraw. Wprost przeciwnie, wykorzystując narzędzia, z którymi zapoznaliśmy się przed chwilą, będziemy mogli zbudować, a właściwie zaprogramować wiele „bardzo poważnych" urządzeń, a przede wszystkim zestaw przyrządów laboratoryjnych o ogromnych możliwościach. Tematowi temu poświęcona będzie „większa połowa" kolejnej części tego artykułu, na razie, pamiętając że musimy jeszcze poruszyć temat obiecanej niespodzianki, podam Wam tylko jeden przykład. Obiecuję, będzie to przykład wyjątkowo spektakularny!



Rys. 20

Rys. 21



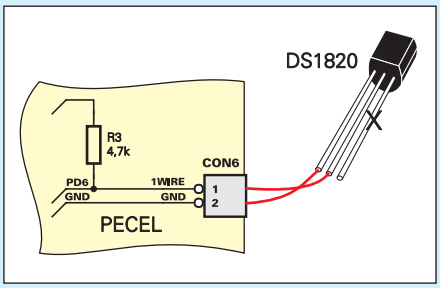
Mam nadzieję, że dysponujecie choćby jednym egzemplarzem popularnego termometru cyfrowego typu DS1820? Jeżeli nie, to warto zakupić nawet kilka sztuk tych tanich i niezwykle użytecznych elementów. Przydadzą się nam wielokrotnie, nie tylko podczas nauki programowania PECEL-a. Jeden taki termometr podłączamy do złącza CON6 minikomputera, dokładnie tak, jak pokazano na **rysunku 22**. Następnie piszemy kolejny program, kompilujemy go i programujemy nim procesor. Programu tego nie będę komentował, ponieważ procedury odczytu danych z termometrów DS1820 zostały już opisane w ramach kursu BASCOM College.

W tym memencie muszę wspomnieć o jednej, dość wstydliwej sprawie. Moje, legendarne już roztargnienie dało jeszcze raz o sobie, tym razem owocując przeoczeniem pewnego elementu, którego umieszczenie w konstrukcji PECEL-a byłoby jak najbardziej wskazane. Mam tu na myśli przycisk służący do ręcznego resetowania procesora. W warunkach normalnej eksploatacji taki element nie byłby specjalnie użyteczny, ale podczas prowadzenia eksperymentów jego zastosowanie może znacznie usprawnić pracę. Wyłączanie i ponowne włączanie zasilania w celu rozpoczęcia pracy programu jest dość uciążliwe, a dodanie przycisku RESET, niekoniecznie umieszczonego na płycie czołowej będzie czynnością bardzo prostą. Jako taki element możemy wykorzystać zwykły microswitch lub dowolny inny przycisk monostabilny o niewielkich wymiarach, dołączony do PECEL-a zgodnie ze schematem pokazanym na **rysunku 23**.

```
$crystal = 8000000
$baud = 19200
Config 1wire = Pind.6
Dim Bd(9) As Byte
Dim Tmp As Byte
Dim T As Integer
Dim T1 As Integer
Dim Reg_no(8) As Byte
Declare Sub Read1820
Input "Naciśnij klawisz
ENTER", tmp
' Uwaga! Wydanie
polecenia INPUT
powoduje wstrzymanie
pracy programu do
momentu naciśnięcia
klawisza ENTER
Do
  1wwrite &HCC
  1wwrite &H44
  Wait 1
  Read1820
  Print T
  Loop
Sub Read1820
  1wreset
  1wwrite &HCC
  1wwrite &HBE
  Bd(1) = 1wread(9)
  1wreset
    Tmp = Bd(1) And 1
    If Tmp = 1 Then
Decr Bd(1)
    T = Makeint(bd(1) ,
Bd(2))
    T = T * 50
    T = T - 25
    T1 = Bd(8) - Bd(7)
    T1 = T1 * 100
    T1 = T1 / Bd(8)
    T = T + T1
    T = T / 10
End Sub
```

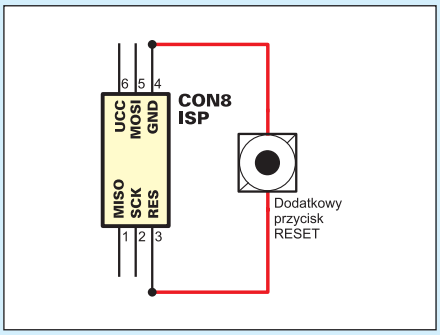
Otwieramy teraz po raz kolejny okienko terminala RS232 i albo naciskamy dodany do układu przycisk RESET, albo wyłączamy i ponownie włączamy zasilanie PECEL-a. Na ekranie monitora ukaże się napis zachęcający do naciśnięcia klawisza ENTER, co też bez obaw możemy natychmiast uczynić.

Zgodnie z przewidywaniami na ekranie rozpoczęło się cykliczne wyświetlanie zmierzonej przez DS1820 temperatury. Ponieważ w programie zabrakło poleceń przeliczających wynik pomiaru, wyświetlane wartości są dziesięciokrotnie zawyżone, tj. zamiast np. 28,3 stopni wyświetlane jest 283. Nie przejmujmy się tym jednak, za chwilę okaże się, że w niczym nie będzie nam to przeszkadzać. Pobawmy się teraz chwilę tak wykonanym termometrem, sprawdzając czy prawidłowo reaguje na podgrzanie i ochładzanie czujnika.



Rys. 22

Rys. 23

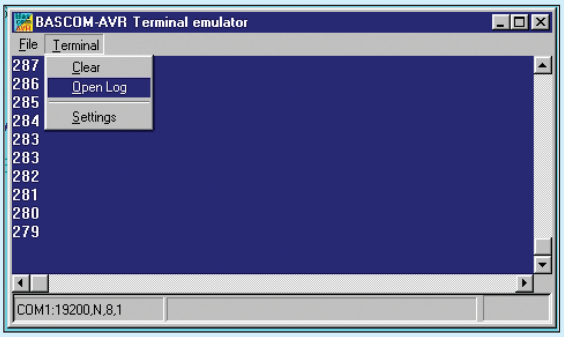


Myślę, że część Czytelników jest nieco rozczarowana: tyle zachodu, aby zbudować prosty termometr! Poczekajcie jednak chwilę, a już teraz mogę Wam przypomnieć, że zbudowaliśmy termometr, ale nie taki znowu byle jaki: do PECEL-a możemy przecież dołączyć absolutnie dowolną ilość czujników DS1820, pracujących na jednym, wspólnym przewodzie. To tego tematu powrócimy jeszcze w przyszłości, a na razie chciałbym pokazać Wam coś innego.

Otwórzmy teraz po raz kolejny okienko terminala portu RS232 i zajmijmy się dodatkowymi przyciskami umieszczonymi na jego górnej krawędzi (**rysunek 24**). Najbardziej powinien nas zainteresować przycisk OPEN LOG, ponieważ otwiera on drogę do nieznanych dotąd, rewelacyjnych możliwości BASCOM-a. Wykonajmy kolejno następujące czynności:

1. Zresetujmy minikomputer bądź za pomocą dodanego przycisku, bądź za pomocą wyłączenia i włączenia zasilania.
2. Kliknijmy na przycisk OPEN LOG. Spowoduje to pojawienie się na ekranie okienka, w którym musimy podać nazwę pliku, do którego zapisywane będą wszelkie dane przechodzące przez monitorowany port COM.
3. Po nadaniu nazwy pliku zamykamy okienko OPEN LOG i naciskamy klawisz ENTER. Od tego momentu wszystkie informacje ukazujące się na ekranie monitora będą także zapisywane w pliku o podanej przez nas nazwie.
4. Pomęczmy teraz trochę nasz termometr. Osobiście polecałbym serię sadystycznych eksperymentów polegających na przypiekaniu go lutownicą lub innym gorącym przedmiotem. Zwracajmy jednak uwagę, aby temperatura czujnika nie przekroczyła 120 stopni (na ekranie liczba 1200!).
5. Po upływie 1 ... 2 minut kliknijmy ponownie na przycisk na krawędzi terminala i tym razem wybierzmy opcję CLOSE LOG.

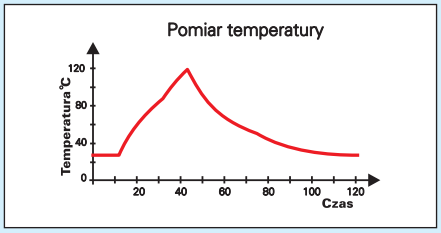
Rys. 24



Możemy teraz zapoznać się z treścią pliku, w którym zapisaliśmy wyniki pomiarów temperatury. To już zaczyna być interesujące: mamy tam dokładny zapis zmian temperatury odbywającego się na określonym odcinku czasu, a pomiary dokonywane były mniej więcej co 1 sekundę. Oczywiście, ten skromny sposób zapisu możemy metodami programistycznymi dowolnie rozbudować. Możemy do każdego pomiaru dodać informację o czasie jego dokonania, możemy dowolnie zmieniać częstotliwość dokonywanych pomiarów, możemy też wreszcie zastosować dowolną ilość czujników i ich wyniki zapisywać w oddzielnych kolumnach. Teraz chyba mogliście zorientować się, jakie możliwości daje transmisja danych z PECEL-a do komputera PC. W najbliższej przyszłości zaprogramujemy PECEL-a tak, aby stał się wszechstronnym laboratoryjnym przyrządem pomiarowym. Połączenie z komputerem da nam wtedy możliwość zapisywania wyników wszelkich pomiarów i archiwizowania ich w celu późniejszego wykorzystania. Ale czy tylko archiwizowania? Przecież dane uzyskane z PECEL-a możemy poddać dalszej obróbce, wykorzystując w tym celu arkusze kalkulacyjne czy też edytory graficzne pracujące pod kontrolą MS WINDOWS. Nie mogę się po prostu powstrzymać, aby nie zaprezentować Wam kolejnego, spektakularnego pokazu możliwości, jakie dostaliśmy do rąk.

Dane opisujące barbarzyński eksperyment z przypiekaniem nieszczęsnego czujnika lutownicą przeniosłem jako plik ASCII do arkusza kalkulacyjnego MS EXCEL. Następnie wyniki pomiarów zostały podzielone przez 10. Wykonanie z tak przetworzonych danych wykresu sprowadziło się już tylko do kilku kliknięć myszką, a efekt wszystkich tych operacji nie trwających dłużej niż minutę został pokazany na **rysunku 25**. Wygląda ciekawie, prawda? Na wykresie widać nawet krótki moment wahania, w którym chciałem dać już spokój dręczonemu czujnikowi, ale ostatecznie postanowiłem torturować go nadal.

Rys. 25



Moi Drodzy, to tylko prosty, najprostszy przykład możliwości PECEL-a używanego w roli inteligentnego terminala komputerowego. W najbliższym czasie zajmiemy się szerzej tym tematem, ale dopiero po omówieniu wszystkich (lub prawie wszystkich) metod programistycznych stosowanych przy pisaniu programów na nasz minikomputer.

Do tej pory mówiliśmy o dwukierunkowej transmisji danych pomiędzy komputerem a procesorem wyłącznie w kontekście ewentualnej budowy urządzeń wykorzystujących taką wymianę informacji. Istnieje jednak jeszcze jedno zastosowanie łącza RS232, genialnie upraszczające odpluskwianie i testowanie pisanego oprogramowania. Transmisję szeregową możemy wykorzystać jako narzędzie do „podglądania" pracującego programu, i to „na żywo", w jego naturalnym środowisku. Wystarczy nieraz, w punkcie programu, którego działania nie jesteśmy pewni, dopisać instrukcje wysyłające na ekran terminala np. informacje o wartości pewnych zmiennych, od których w decydujący sposób zależy działanie programu. Z kolei, jeżeli chcemy przetestować program zmieniając „zdalnie" parametry jego pracy, to stosując instrukcję INPUT możemy w wybranych momentach zmieniać wartości wybranych zmiennych.

Omawianie podstawowych zagadnień związanych z transmisją danych poprzez złącze RS232 zajęło nam tyle miejsca, że już niewiele go pozostało na opisanie niespodzianki, jaką dla Was przygotowałem. Tak więc z konieczności omówimy tę sprawę w największym skrócie, pozostawiając resztę do następnego numeru EdW.

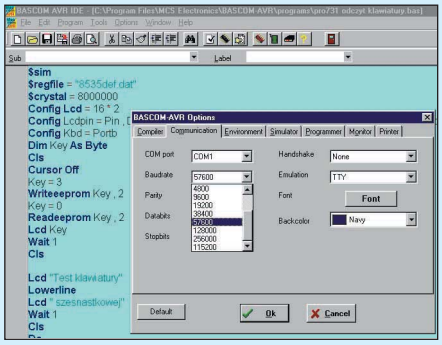
Opis PECEL-a nie jest BASCOM College i nie mam prawa zadawać Wam jakichkolwiek ćwiczeń czy prac domowych do odrobienia. Mam jednak prośbę: może zechcielibyście w wolnej chwili przepisać do edytora BASCOM-a program, którego listing został zamieszczony poniżej.

Program ten nie jest mojego autorstwa, nie mogę wiec go ani zmieniać ani komentować. Tak wiec, po prostu przepiszcie go, nie wnikając na razie w jego treść. Następnie poproszę Was o skompilowanie tego programu i wpisanie go do pamięci procesora. Uprzedzam, że bezpośrednio po wykonaniu tej czynności spotka Was przykre rozczarowanie: PECEL nie będzie dawał żądnych widocznych z zewnątrz oznak „życia". Następnie połączcie ponownie PECEL-a z komputerem za pomocą złącza RS232 i na wszelki wypadek odłączmy od niego kabel programatora.

```
'MONITOR SYMULACJI
SPRZĘTOWEJ
$regfile = "8535def.dat"
$crystal = 8000000
$baud = 57600
Dim Krk As Byte
Dim Adr As Word
Dim Adrl As Byte , Adrh
As Byte
Dim Vl As Byte
Do
  Krk = Inkey()
  If Krk = "T" Then
    Print Chr(13);
  Elseif Krk = "W" Then
    Adr = Waitkey()
    Vl = Waitkey()
    Out Adr , Vl
    Print Chr(13);
  Elseif Krk = "R" Then
    Adr = Waitkey()
    Vl = Inp(adr)
    Print Chr(vl);
  Elseif Krk = "O" Then
    Adrl = Waitkey()
    Adrh = Waitkey()
    Vl = Waitkey()
    Adr = Adrh * 256
    Adr = Adr + Adrl
    Out Adr , Vl
    Print Chr(13);
  End If
Loop
```

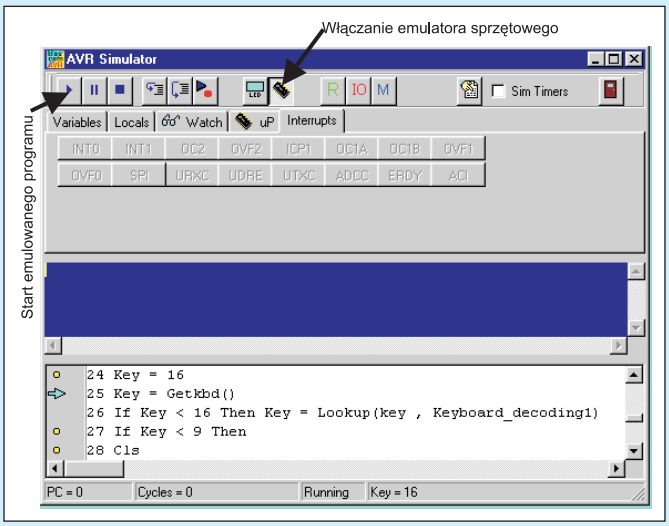
Kolejną czynnością będzie napisanie prostego programu, wykorzystującego np. wyświetlacz alfanumeryczny LCD i klawiaturę szesnastkową. Nie musi to być jakiś nowy i skomplikowany program, na początek zupełnie wystarczy prosty programik, który już wykorzystywaliśmy do demonstracji obsługi wyświetlacza LCD i klawiatury. **Tu bardzo ważna uwaga: na samym początku programu napiszcie "$SIM"!** Następnie skompilujcie ten program, ale **w żadnym wypadku nie ładujcie go do pamięci procesora**!

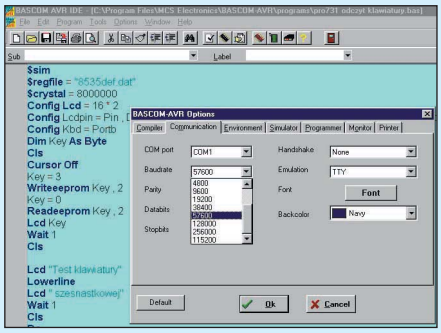
Kolejną czynnością będzie wykonanie drobnej zmiany w konfiguracji BASCOM-a. Musimy zmienić uprzednio ustawioną prędkość transmisji danych poprzez złącze RS232 z 19200 na 57600 baud (**rysunek 26**). No i wreszcie dochodzimy do końca: pozostało nam już tylko otworzenie okienka symulatora (klawisz F2), kliknięcie na ikonkę włączania symulacji sprzętowej (**rysunek 27**) i naciśnięcie na strzałkę włączającą symulację. Następnie możemy już przeprowadzić testy klawiatury za pomocą uniwersalnego emulatora sprzętowego, który w tym momencie dostaliście do dyspozycji. Nie będzie już potrzebne wielokrotne przeprogramowywanie procesora w celu przetestowania drobnych zmian w programie. Wszystkie testy będziemy mogli przeprowadzić w „real world"



Rys. 26

Rys. 27





Rys. 28

wyłącznie w symulacji sprzętowej.

Poruszony temat jest tak obszerny, że jego kontynuację odkładamy do dalszej części artykułu o minikomputerze PECEL.

Zbigniew Raabe,
zbigniew.raabe@edw.com.pl