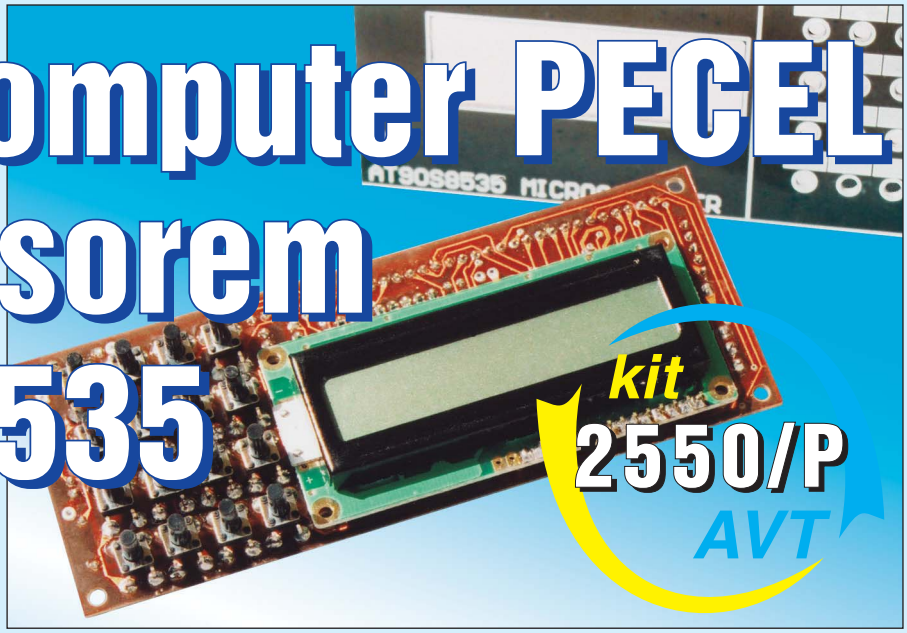
Mikrokomputer PECEL z procesorem AT90S8535



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.

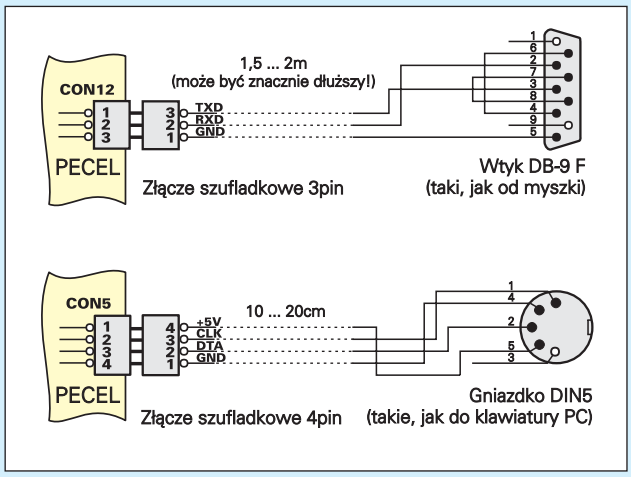
Jednocześnie zapraszam wszystkich Czytelników EdW do współpracy w tworzeniu oprogramowania dla PECEL-a.

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:

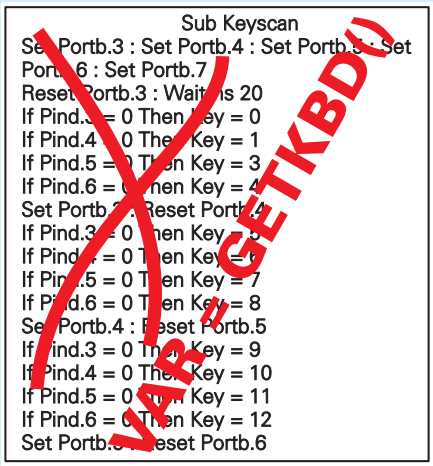
CONFIG KBD = PORT[numer portu]

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.

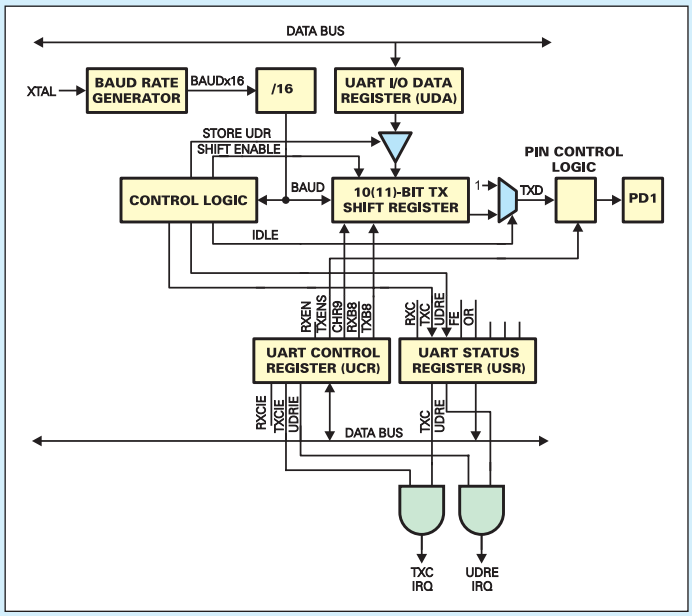
To jednak jeszcze nie wszystko: za chwilę czeka Was prawdziwa niespodzianka, i to taka, o jakiej pewnie nawet nie marzyliście! Za moment dostaniecie do ręki wyjątkowo potężne narzędzia programowe i sprzętowe, które mogą uczynić programowanie PECEL-a nie tylko dziecinnie łatwym zadaniem, ale także prawdziwą 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:

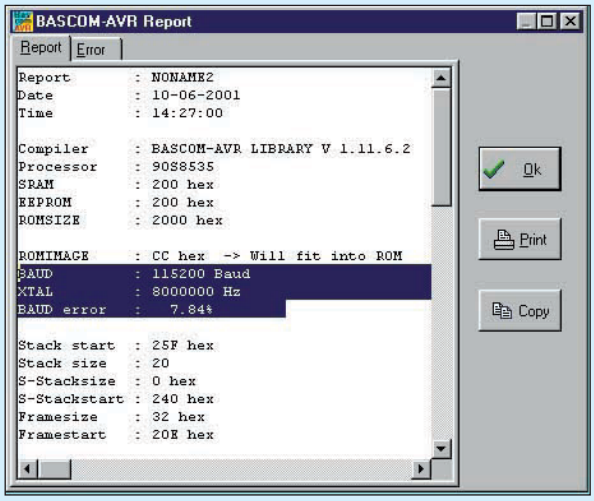
```
$crystal = 8000000
$baud = 19200
Do
Print " PECEL wita Czytelników Elektroniki dla
Wszystkich!"
Wait 1
Loop
End
```

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.

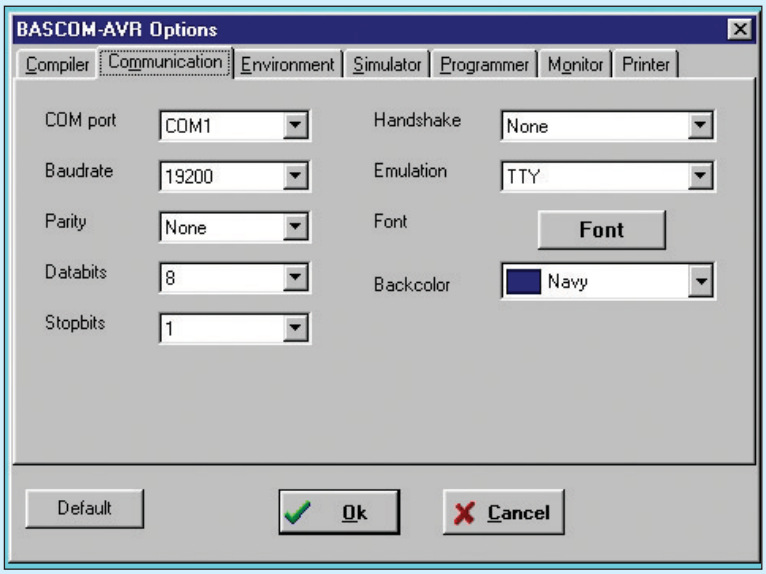
A zatem nadeszła dawno oczekiwana chwila! Klikamy na przycisk TOOLS i następnie wybieramy opcję TERMINAL EMULATOR, lub po prostu naciskamy kombinację klawiszy CTRL + T. Jeżeli wszystkie opisane uprzednio czynności wykonaliśmy poprawnie, to nasze oczy powinien ucieszyć widok pokazany na **rysunku 19**.

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.

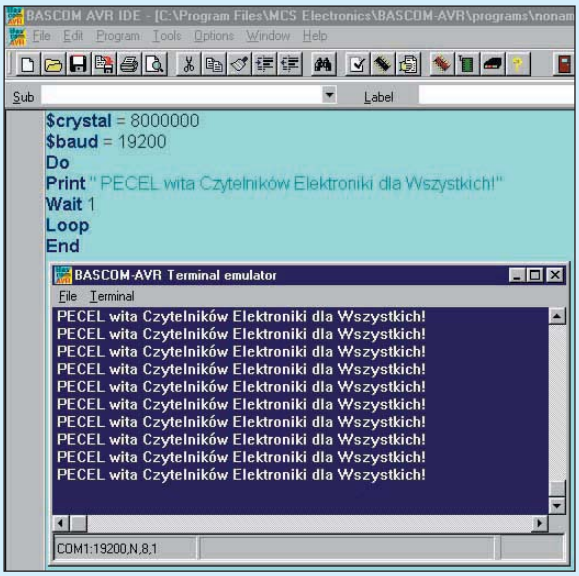
Ciąg dalszy w następnym numerze EdW

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.