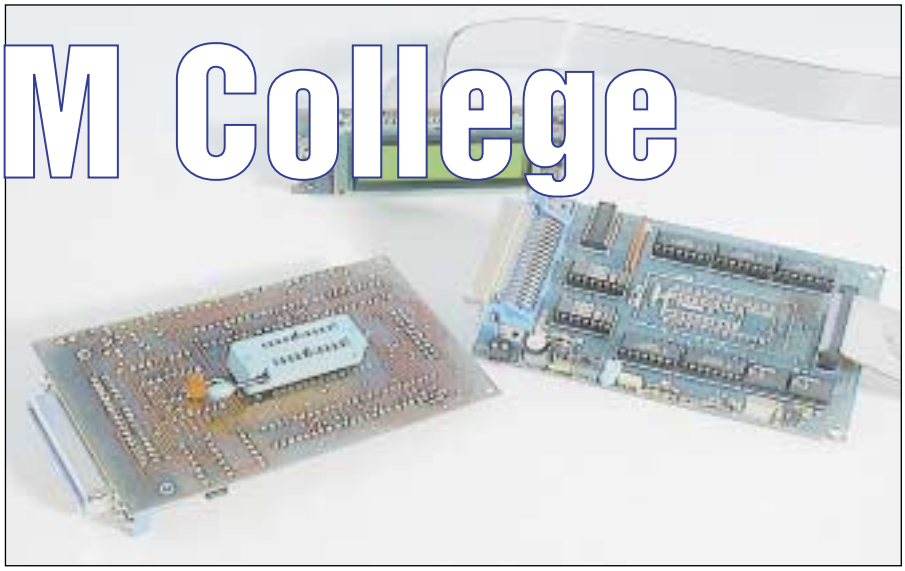
BASCOM College



Wykład 4

Witam Studentów **BASCOM College** na kolejnym wykładzie. Poprzedni wykład rozpocząłem od stwierdzenia, że jestem w znakomitym nastroju, spowodowanym mailami od Czytelników, którzy aktywnie włączyli się w nasz program zajęć i owocnie próbują samodzielnego myślenia i pisania programów. Minął miesiąc i mój nastrój polepszył się jeszcze bardziej. Powód jest ten sam: nie przypuszczałem nawet, że tak wielu Kolegów osiągnie tak dobre rezultaty już w pierwszym okresie istnienia **BASCOM College**. Właściwie, to może powinienem się martwić, bo jeżeli uczniowie zaczynają prześcigać mistrzów.... Ale po kolei, pora na mały rachunek sumienia!

Z pewnością wielu Czytelników czytając wykład o sterowaniu wyświetlaczy alfanumerycznych LCD zauważyło, że ten tekst miejscami trochę się "nie kleił". Od razu zaczęliśmy od "dużych" wyświetlaczy, pomijając najprostsze, jakimi są wyświetlacze 16*1 znaków. Wyraźnie autor kluczył dookoła tego tematu, ale nigdy go nie poruszał. Tak właśnie było, lawirowałem jak mogłem, ponieważ obsługa tych właśnie wyświetlaczy napotyka jeszcze w BASCOMie na spore trudności, nie spotykane przy wyświetlaczach 16*2 i jakichkolwiek innych. Nasz mistrz, Mark, przesiedział nad tym problemem niejedną noc, ja także dołożyłem swój wkład pracy w próby jego rozwiązania. Niestety, nasze wysiłki spełzły na niczym. Wyświetlanie statycznych napisów działa poprawnie, natomiast problemy zaczynają się przy przesuwaniu napisu lub kursora. Nie pomaga używanie polecenia konfiguracyjnego CONFIG LCD = 16*1,a po wydaniu polecenia SHIFTLCD napis dzieli się na dwie połowy, z których każda "zaczyna żyć własnym życiem". Powód tego jest dość oczywisty: wyświetlacz 16*1 jest z punktu widzenia sterującego nim programu wyświetlaczem 8*1. Mark badał ten problem, ale słusznie stwierdził, że producenci wyświetlaczy 16*1 stosują dziwne procedury ich obsługi i że nie bardzo jest w stanie zaproponować jakieś sensowne rozwiązanie problemu z przewijaniem napisów. Problem w gruncie rzeczy nie jest zbyt poważny: wyświetlacze 16*2, działające bez najmniejszych problemów, można kupić w tej samej cenie, co 16*1, a nawet niejednokrotnie taniej. Co jednak mają zrobić ci, którzy nabyli już displaye 16*1?

Rozwiązanie kłopotliwego problemu nadesłał mi jeden z Was, kolega **Artur Krajewski**. Wielkie brawa za błyskotliwe rozwiązanie problemu przewijania napisu na nieszczęsnym wyświetlaczu.

Kolega Artur nadesłał mi także bardzo "elegancko" napisany **program sterowania silnikiem krokowym**. Nie on jedyny, takich programów otrzymałem wiele i prawie wszystkie napisane były poprawnie, a w każdym razie wszystkie działały.

Listy Kolegi Artura i innych Czytelników dały mi wiele do myślenia i zainspirowały do stworzenia w **BASCOM College** działu, który nazwaliśmy **BASCOM Forum**. Jeżeli ktoś z Was wpadł na pomysł jakiegoś ciekawego rozwiązania programowego, jakiejś interesującej sztuczki, za pomocą której można w prosty sposób rozwiązać pozornie skomplikowany problem programistyczny, to bardzo proszę nadsyłajcie swoje opracowania do **BASCOM Forum**, najlepiej na mój adres e-mail. Chciałbym jednak, abyśmy się dobrze zrozumieli: nie chodzi mi o skomplikowane programy, ale wyłącznie o "tips & tricks", malutkie programiki w stylu opracowanych przez Kolegę Artura. Takie listingi będziemy systematycznie zamieszczać w **BASCOM Forum**, oczywiście z podaniem nazwiska autora. Gorąco zachęcam Was do tej działalności: pamiętajcie, że dla programisty najważniejsze są inteligencja i wyobraźnia i nie musicie się wobec tego stresować swoim małym doświadczeniem.

Także Czytelnicy, którzy napotykają problemy podczas pisania swoich programów, proszeni są o nadsyłanie dręczących Ich pytań. Opublikujemy je w **BASCOM Forum** i być może któryś z Kolegów znajdzie na nie odpowiedź? Ta forma wymiany informacji przeznaczona jest przede wszystkim dla tych z Was, którzy nie posiadają jeszcze stałego dostępu do Internetu i nie mogą korzystać z istniejących tam list dyskusyjnych.

Jeśli Wasza aktywność będzie bardzo duża, jeśli otrzymamy dużo listów z opisami problemów oraz rozwiązaniami problemów, jeśli ilość tego materiału przerośnie ramy rubryki **BASCOM Forum**, to mamy już gotową koncepcję wydawania **„Biuletynu BASCOM Forum"** - specjalnego dodatku do EdW, tylko dla prenumeratorów EdW (ok. 98% studentów **BASCOM College** to prenumeratorzy EdW). Łamy tego Biuletynu będą służyć do wymiany informacji między studentami **BASCOM College**.

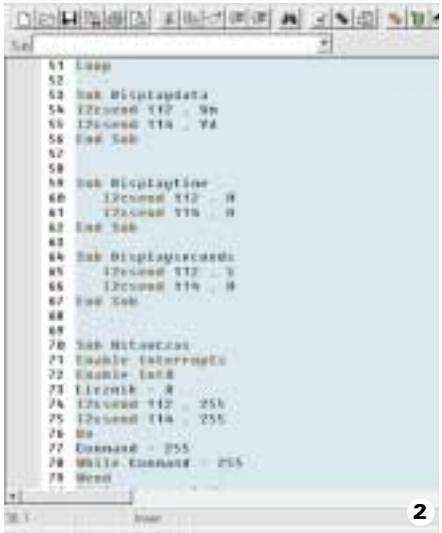
Chciałbym teraz dokończyć "rachunek sumienia", czyli zakomunikować Wam dwie sprawy, o których zapomniałem wspomnieć na poprzednich wykładach. Otóż, opisując konfigurowanie programu BASCOM, nie wspomniałem o bardzo ważnej opcji, przyspieszającej pracę i czyniącej ją bardziej komfortową. Na usprawiedliwienie mam tylko jedno: Mark ukrył ją w niezbyt konsekwentny sposób w panelu konfigurowania otoczenia, a nie w okienku, w którym ustawiamy parametry pracy programatora. Ponadto, opracowując wykład traktujący o konfigurowaniu BASCOM-a, znałem dobrze tylko jego poprzednią wersję, w której opisane niżej opcje nie występowały.

A zatem, po uruchomieniu BASCOM-a otwórzcie panel OPTIONS, a następnie ENVIRONMENT i kliknijcie na przycisk IDE (**rys.1**). Następnie zaznaczcie "ptaszkiem" opcję PROGRAM AFTER COMPILE. Od tego momentu nie będziemy już musieli po skompilowaniu programu wywoływać panelu programatora, ani nawet naciskać klawisza F4. Po kompilacji wywołanej klawiszem

F7 program automatycznie przystąpi do programowania procesora, wyświetlajc stosowane komunikaty na dole ekranu. Fajne, prawda? Żeby tak jeszcze dało się programować nasze poczciwe "pięćdziesiątki jedynki" bezpośrednio w systemie, tak jak procesory AVR! To dopiero byłby komfort pracy: po jednym naciśnięciu klawisza moglibyśmy od razu testować napisany program w jego naturalnym otoczeniu. Szkoda, że procesory "X051 nie mogą być programowane w trybie ISP (In System Programming). Nie mogą, jesteście tego pewni? No to zajrzyjcie do naszej stałej rubryki opisującej nowości z ostatniej chwili, przygotowałem tam dla Was niezłą niespodziankę!


1

Popatrzmy jeszcze chwilę na panel IDE. Mamy tam dwie interesujące opcje. "Zaptaszkujmy" okienko z napisem DISPLAY LINE NUMBERS, wyjdźmy na chwilę z powrotem do okienka edycji programu. Po zaznaczeniu wymienionej opcji wygląda ono tak, jak pokazano na **rysunku 2**. Wszystkie linie programu zostały ponumerowane, z tym, że numeracja ta ma znaczenie jedynie informacyjne i zostanie zignorowana przez kompilator. Mnie ta forma przedstawiania edytowanego programu nie bardzo się podoba, ponieważ zbyt przypomina ekran edytora archaicznych dialektów BASIC-a, stosowanych jeszcze na komputerach 8-bitowych. Ale może dla kogoś taka numeracja będzie ułatwieniem w pracy?


2

Znacznie ciekawsza jest opcja DISPLAY HINTS (wyświetl podpowiedzi). Zaznaczmy ją zatem i powróćmy do edycji programu. W wolnej linii napiszmy jakiekolwiek polecenie języka MCS BASIC, np. CONFIG. Natychmiast po napisaniu ostatniej litery na ekranie ukazał się napis (**rys.3**) informujący, do jakich wartości może odnosić się to polecenie.


3

Niezły bajerek, ale o jego stosowaniu musicie już sami zadecydować. Ważne jest to, że podpowiedzi pojawiające się na ekranie możemy dowolnie edytować, dostosowując je do swoich potrzeb, a także dodawać nowe, związane z dowolnymi słowami. Plik z podpowiedziami znajduje się w głównym katalogu BASCOM-a i posiada nazwę: BASCOM.KRF. Poniżej zamieszczony jest fragment tego pliku wraz z drobnymi zmianami, jakie pozwoliłem sobie w nim poczynić. Oczywiście, nie ma najmniejszego znaczenia, w jakim języku redagowany jest ten plik i czy stosowane są w nim polskie znaki diakrytyczne.

```
DISABLE|intsource
PRINTHEX|var
SWAP|var,var
DELAY| krótkotrwałe opóźnienie
TO|end [STEP var]
DOWNTO|end [STEP var]
REM|remark
ENABLE|intsource
I2CSEND|slaveaddress,var [,bytes]
I2CRECEIVE|slaveaddress,var [,bytes to write,bytes to read]
START|timerx
LOAD|timerx
COUNTER| = wartość
EdW| bardzo fajne pismo!
```

Tyle rachunku sumienia na dzień dzisiejszy. Rozpoczynamy wreszcie nasz wykład, którego głównym tematem będzie symulacja programowa i sprzętowa.

Emulacja programowa i symulacja sprzętowa w programie BASCOM 8051

Możliwość przetestowania napisanego programu bez konieczności programowania procesora i umieszczania go w uruchamianym układzie jest jedną z największych zalet pakietu BASCOM. Z listów, jakie otrzymałem od Czytelników wiem, że korzystanie z symulatorów sprawia im nieco kłopotów i że nie wszystkie opisy zwarte w helpie są dla nich jasne i zrozumiałe. Najwyższa więc pora, aby uporządkować nasze wiadomości o symulatorach pakietu BASCOM i umożliwić wszystkim Czytelnikom korzystanie z tych znakomitych narzędzi.

Zacznijmy od uporządkowania terminologii, niezbyt jasno zdefiniowanej w BASCOM-ie. Podczas tego wykładu i następnych przyjmiemy następujące określenia:

Emulacja programowa - badanie działania programu z wykorzystaniem wbudowanego w pakiet BASCOM symulatora. W BASCOM-ie stosowane jest określenie "HARDWARE EMULATION". Do wykonania tej symulacji nie są potrzebne jakiekolwiek narzędzia sprzętowe.

Symulacja sprzętowa - badanie działania programu w środowisku, w którym będzie następnie pracował zaprogramowany procesor. W BASCOM-ie używane jest określenie "REAL HARDWARE SIMULATION". Tyz piknie, jak mawiają starzy górale, ale nasze określenia bardziej mi się podobają.

W poniższej tabeli zamieszczone zostało zestawienie funkcji, jakie możemy testować w każdym z rodzajów symulacji.

Funkcja	Emulacja programowa	Symulacja sprzętowa
Dwukierunkowe operacje na portach i pojedynczych wyprowadzeniach portów	Tak	Tak
Obsługa LCD	Tak	Tak
Odbiór kodu RC5	Nie	Nie
Operacje matematyczne	Tak	Tak
Przerwania	Tak	Tak, z ograniczeniami
Timery	Tak, z ograniczeniami	Tak
Transmisja	1WIRE Nie	Tak, na szybkich maszynach
Transmisja I2C	Nie wykorzystuje się	Tak
Wszelkie operacje w czasie rzeczywistym	Nie	Nie

Generalnie możemy przyjąć następujące zasady:

1. Wydarzenia zachodzące podczas symulacji, obojętne czy programowej, czy sprzętowej, zawsze **zachodzą znacznie wolniej** niż w zaprogramowanym procesorze (co niekiedy jest ogromną zaletą!). To spowolnienie biegu czasu zależy oczywiście od procesora komputera, na którym prowadzona jest symulacja, ale nawet najszybszy PENTIUM nie będzie w stanie umożliwić symulacji wydarzeń rozgrywających się w czasie rzeczywistym.

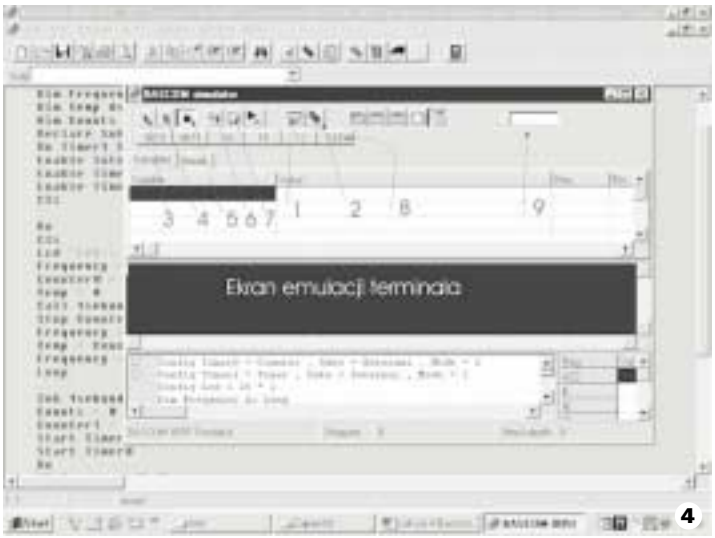
2. Z wyżej wspomnianych powodów nie jest możliwe symulowanie np. odbioru kodu RC5, który nadawany jest przez piloty ze stałą szybkością, której w żaden sposób nie możemy zmienić. Także symulacja transmisji 1WIRE napotyka na poważne trudności i możliwa jest tylko na najszybszych komputerach.

3. Prosta konstrukcja naszego symulatora sprzętowego nie umożliwia testowania urządzeń, które wykorzystują wejścia analogowe procesora. Przez nasz symulator "przechodzą" jedynie sygnały cyfrowe. Oczywiście, budowa symulatora realizującego tę funkcję jest możliwa, ale byłoby to urządzenie bardzo skomplikowane i kosztowne. Nie wykluczam jednak powstania takiego symulatora w przyszłości.

4. Biorąc pod uwagę powyższe zastrzeżenia, podczas emulowania pracy procesora należy

skupić się na badaniu zjawisk zachodzących "wewnątrz" procesora, badać poprawność pętli programowych, obliczeń dokonywanych przez procesor, a także transmisji I2C, która może być symulowana sprzętowo bez najmniejszych problemów.

Na **rysunku 4** zostało pokazane otwarte okno emulatorów. Zaznaczone zostały najważniejsze funkcje emulatorów, dotyczące procesorów 'X051:



1. Uruchamianie symulacji programowej. Kliknięcie tego przycisku powoduje wyświetlenie panelu zawierającego wyświetlacz LCD, wyświetlacz siedmiosegmentowy i zobrazowane jako diody LED wyprowadzenia portu 1 i 3 (**rys. 5**).

2. Uruchamianie emulacji sprzętowej. Przycisk ten uruchamia emulator programowy, o ile ten jest dołączony do komputera. **Tu bardzo ważne uwagi: włączenie emulacji sprzętowej bez dołączenia emulatora programowego połączonego z badanym układem drastycznie spowalnia pracę programu, a w pewnych przepadkach może nawet spowodować zawieszenie się komputera. Symultaniczna praca obydwu symulatorów jest możliwa, ale także znacząco spowalnia pracę.**

3. Start testowanego programu. Naciśnięcie tego przycisku powoduje rozpoczęcie wykonywania **ostatnio skompilowanego programu**. Program jest wykonywany aż do samoczynnego zakończenia lub przerwania emulacji.

4. Przerwa w testowaniu. Przycisk pauzy, która może ułatwić przeanalizowanie ostatnio wykonanych linii programu.

5. Zatrzymanie symulacji. Powoduje zatrzymanie wykonywanego programu, bez możliwości wznowienia od momentu zatrzymania.

6. Praca krokami - polecenie po poleceniu. Bardzo użyteczna funkcja pozwalająca na dokładne prześledzenie działania wybranych fragmentów programu. Każde naciśnięcie tego przycisku powoduje wykonanie kolejnej linii programu

7. Wykonanie programu od początku do bieżącej linii (linii zaznaczonej żółtą strzałką na marginesie okna programu). **Tu ważna uwaga: na niektórych komputerach ta strzałka zachowuje się dziwnie: pojawia się i znika**. Nie wiem, dlaczego tak się dzieje, ale pisałem już do Marka w tej sprawie.

8. Przyciski służące do generacji przerwań w symulacji programowej i sprzętowej. Bardzo ważna funkcja, szczególnie podczas testowania programów wykorzystujących timery. Symulacja tak bardzo zwalnia pracę programów, że oczekiwanie na wystąpienie przerwania timera może być nieco nużące. Za pomocą przycisków T0 lub T1 możemy to oczekiwanie znacznie skrócić.

9. Suwak regulacji szybkości emulacji. Czasami spowolnienie pracy programu wywoływane samym faktem symulacji okazuje się niewystarczające. Suwakiem tym możemy jeszcze bardziej zwolnić pracę programu i w ten sposób dokładnie przeanalizować interesujący nas jego fragment.

10. Wyświetlanie zawartości pamięci programu. Opcja dla zaawansowanych: powoduje wyświetlenie okienka zawierającego kod aktualnie testowanego programu.

11. Okienko "podglądu" zmiennych występujących w programie i rejestrów emulowanego procesora. Fantastyczna sprawa: daje możliwość podglądania tego, co dzieje się w rejestrach timerów i jak zachowują się zastosowane w programie zmienne. Napiszmy sobie prosty programik:

```
Dim X As Byte
Dim Z As Byte
Dim Y As Byte
Do
  Incr X
  Decr Z
  Y = X - Z
Loop
```

Większego sensu ten program nie ma, ale chodzi nam tylko o pokazanie możliwości podglądu zmiennych występujących w programie. Po skompilowaniu tego wybitnego dzieła programistycznego uruchamiamy symulacje programową, a w okienko VARIABLES wpisujemy (po naciśnięciu klawiszy CTRL + ALT + V) zmienne X, Y i Z. Naciskamy przycisk uruchamiania programu, a efekt możemy zobaczyć na **rysunku 6**.

12. Okienko kontroli zawartości rejestrów. Funkcja tego okienka jest nieco podobna do roli pełnionej przez podgląd zmiennych. Umożliwia ono jednak obserwację wszystkich rejestrów wewnętrznych procesora. Opcja raczej dla zaawansowanych.

Sądzę, że najlepszą metodą poznania zarówno emulatora programowego, jak i symulatora sprzętowego będzie, jak zwykle, doświadczenie praktyczne. Zajmijmy się teraz okienkiem emulacji terminala, wykorzystywanym zarówno w symulacji programowej, jak i w emulacji sprzętowej. Jest to jedno z najsilniejszych narzędzi pakietu BASCOM i musimy poświecić mu chwilę uwagi.

Procesory 'X051 posiadają wbudowaną obsługę interfejsu RS232, a transmisja informacji poprzez to łącze jest z poziomu języka MCS BASIC szczególnie łatwa. Nie musimy się nawet zastanawiać, jaki jest protokół transmisji portu szeregowego: po prostu piszemy np. "PRINT [coś tam] i mamy absolutną pewność, że to nasze "coś tam" zostanie wysłane do portu RS232. Podobnie proste jest odczytywanie danych z tego portu, sprowadzające się do wydania jednego lub kilku poleceń. Jednak ekran terminala nie służy wyłącznie testowaniu programów, które komunikują się z komputerem lub innym urządzeniem za pomocą interfejsu RS232. Jest on jeszcze jednym narzędziem umożliwiającym "podglądanie" testowanego programu i sprawdzania, czy wykonywany jest on zgodnie z naszymi oczekiwaniami.

Napiszmy sobie zatem kolejny, nieco dłuższy program. Właściwie powinniśmy uczynić

to nieco później, na ćwiczeniach, ale nie mogę wprost się powstrzymać, aby nie pokazać Wam czegoś ciekawego. Naprawdę, to będzie rzeczywiście ciekawe, choć nieco wyprzedzające nasz program nauczania. W programie tym po raz pierwszy zastosujemy transmisję I²C, która będzie tematem dopiero następnej lekcji i dlatego te fragmenty programu pozostawimy na razie bez komentarza.

$sim

Metapolecenie $SIM umieszczamy na początku programu, który ma być testowany w symulacji programowej lub emulacji sprzętowej. PRZED KOMPILACJĄ GOTOWEGO PROGRAMU, KTÓRY MA ZOSTAĆ UMIESZCZONY W PROCESORZE, POLECENIE TO NALEŻY BEZWZGLĘDNIE USUNĄĆ!

```
Config Sda = P1.6
Config Scl = P1.7
Declare Sub Write_eeprom(adres As By-
                              te , Value As Byte)
Declare Sub Read_eeprom(adres As Byte
                             , Value As Byte)
Dim R As Byte
Dim Value As Byte , Adres As Byte
For R = 0 To 5
Print "Podaj wartość " ; R ; "[0...255]";
Input Value
Call Write_eeprom , R , Value
Next R

Print "Weryfikacja zapisu:"
For R = 0 To 5
        Call Read_eeprom , R , Value
        Print "Wartość " ; R ; " = " ; Value
Next R
End

Sub Write_eeprom(adres As Byte , Value
                                  As Byte)
        I2cstart
        I2cwbyte 160
        I2cwbyte Adres
        I2cwbyte Value
        I2cstop
        Waitms 10
  End Sub

Sub Read_eeprom(adres As Byte , Value
                                  As Byte)
        I2cstart
        I2cwbyte 160
        I2cwbyte Adres
        I2cstart
        I2cwbyte 161
        I2crbyte Value , 9
        I2cstop
End Sub
```

Po "wklepaniu" powyższego programu do edytora tekstowego BASCOM-a kompilujemy go i .... poczekajcie jeszcze moment, musimy przygotować sobie hardware. Dołączamy do komputera emulator sprzętowy, a jego wtyk emulacyjny łączymy z naszą płytką testową. Na płytce musi być umieszczony tylko jeden element: pamięć szeregowa EEPROM, czyli IC2. Następnie zakładamy jumpery JP2 i JP3 i powracamy do naszego programu. Wywołujemy okienko emulatora i przyciskiem z rysuneczkiem układu scalonego włączamy emulację sprzętową.

Po uruchomieniu programu zostaniemy poproszeni o podanie pięciu wartości z zakresu od 0 do 255, czyli liczb ośmiobitowych. Wartości podajemy z klawiatury, potwierdzając każdą z nich naciśnięciem ENTER. Sekundę po podaniu ostatniej liczby na ekranie pojawia się komunikat o dokonaniu weryfikacji poprawności zapisu danych w pamięci EEPROM (**rys. 7**).

Czemu właściwie miała służyć ta demonstracja? Chciałem pokazać Wam, jak użyteczna może być funkcja PRINT, nawet jeżeli nie jest ona stosowana zgodnie ze swoim podstawowym przeznaczeniem. Operacje zapisu i odczytu danych z pamięci EEPROM wykonywane są zwykle przez procesor "po cichu", bez wysyłania jakichkolwiek komunikatów. W przypadku złego działania programu, często nie jesteśmy w stanie stwierdzić, czy zapis lub odczyt danych przebiegał poprawnie. W takiej sytuacji możemy program lub jego fragment przetestować za pomocą emulatora sprzętowego, dodając we właściwych miejscach instrukcje PRINT, które po ewentualnym poprawieniu błędów możemy usunąć. Wygodne, prawda?

Moi Drodzy, zastanówmy się teraz, co właściwie najlepszego uczyniliśmy, pisząc ten program? Będzie to kolejna dygresja ale koniecznie chcę Wam to powiedzieć. A więc, co zrobiliśmy, proste ćwiczenie? Nie, zrobiliśmy ni mniej, ni więcej tylko ... **programator szeregowych pamięci EEPROM**! Przecież za pomocą tego programu, po zmianie wartości w pętli FOR .... NEXT (z 5 na 255 lub, w przypadku "mniejszych" pamięci, na 127) możemy zaprogramować pamięć danymi, które mogą być wykorzystane w jakimś budowanym przez nas urządzeniu. Co więcej, ten programator nie będzie kosztował nas ani grosza, ponieważ jedynym elementem potrzebnym do jego budowy jest posiadana już przez nas płytka testowa!

Jasne, że taki "ręczny" programator jest urządzeniem dość prymitywnym i nie nadającym się do profesjonalnej pracy. Nic jednak nie stoi na przeszkodzie, aby zestaw danych pozyskanych z jakiegoś programu obliczeniowego przenieść przez clipboard do edytora BASCOM-a i następnie ... nie - dość już tych dygresji, jeżeli tak dalej pójdzie, to na kolejnej lekcji będziecie mieli innego wykładowcę, a ja dam do gazet ogłoszenie: "Przyjmę każdą pracę, może być fizyczna"! Wracajmy do naszego emulatora!

Zajmijmy się teraz dolnym okienkiem panelu emulatora, tym, w którym wyświetlany jest tekst testowanego programu. Wygląda ono niepozornie, ale kryje w sobie pewne możliwości. Uruchommy jeszcze raz nasz program testowania zapisu do EEPROM (bardzo proszę, zachowajcie go do następnej lekcji o I2C, będzie BARDZO potrzebny) i zatrzymajmy go, na przykład po wprowadzeniu 2 liczby. Przez cały czas pracy programu tekst w dolnym okienku przesuwał się, a obecnie znieruchomiał (**rys. 8**). Najedźmy teraz kursorem myszki na jakąś zmienną umieszczoną w programie, np. na VALUE. W tym samym momencie w lewym dolnym rogu okienka pojawił się komunikat: "VALUE = 24". Co to oznacza? To proste, przecież zatrzymaliśmy program po wprowadzeniu drugiej liczby, którą było właśnie ... 24! Naprowadźmy teraz kursor na inną zmienną, na R. Natychmiast w dolnym rogu okna wyświetlił się napis: "R = 2". A więc doszliśmy do kolejnego ułatwienia oferowanego nam przez emulator BASCOM-a: w okienku z tekstem programu zmienne przybierają takie wartości, jakie są im nadawane przez pracujący program i wartości te, po zatrzymaniu programu, możemy w każdej chwili odczytać i odpowiednio zinterpretować. No, niech ktoś jeszcze powie, że nasz BASCOM nie jest najwygodniejszym narzędziem programistycznym, jakie kiedykolwiek powstało!

No dobrze, ale program wykonywany jest bardzo szybko, nawet w symulacji, i pewnie trzeba mieć znakomity refleks, aby zatrzymać go we właściwym momencie? Ależ nic podobnego, aby zatrzymać program dokładnie w tym miejscu, gdzie chcemy, nie musimy wcale posiadać refleksu teksańskiego rewolwerowca. Metod jest kilka: możemy na przykład w tekst programu

wprowadzić dodatkowe polecenia BREAK (przerwij). Po napotkaniu takiego polecenia emulator przerywa pracę i należy go powtórnie uruchomić przyciskiem START (program wykonywany będzie od momentu wystąpienia instrukcji BREAK). Jednak wpisywanie dodatkowych poleceń w tekst programu niesie z sobą pewne ryzyko: jeżeli zapomnimy je usunąć przed ostateczną kompilacją i programowaniem procesora, to biada nam: procesor z pewnością zgłupieje (można wstrzymać wykonywanie poleceń BREAK za pomocą metapolecenia $NOBREAK, umieszczonego na początku programu). Jest jednak znacznie wygodniejsza metoda, pozwalająca na umieszczanie i usuwanie znaczników chwilowego przerywania pracy programu. Ustawmy teraz kursor na początku tej linijki programu, przy której chcemy wstrzymać jego działanie i naciśnijmy klawisz F9. Z pewnością zauważyliście, że na marginesie okienka pojawiła się w tym momencie czerwona kropka. Naciśnijmy jeszcze raz F9 - kropka zniknęła. Przenieśmy teraz kursor na początek innej linii, znowu naciśnijmy F9 i uruchommy program. Zauważymy,że program zatrzyma się na linii, przy której została umieszczona czerwona kropka. W ten sposób możemy podczas symulacji zatrzymywać program w dowolnych momentach, analizować zmienne lub rejestry i ponownie uruchamiać. Bardzo ważne jest też to, że wstawiane w ten sposób polecenia przerwania egzekucji programu nie "przedostają się" do jego kodu źródłowego i nie grozi nam wpadka spowodowana nieusunięciem poleceń BREAK przed programowaniem procesora.

Moi Drodzy, nie wyczerpaliśmy całego materiału dotyczącego emulacji sprzętowej i symulacji programowej. Czeka na Was jednak wyjątkowo ciekawe ćwiczenie i nie chciałbym, abyście przystąpili do jego wykonywania zbyt zmęczeni. Do tematu symulatorów BASCOM-a będziemy jeszcze niejednokrotnie powracać, a na razie, jak zwykle proponuję filiżankę kawy i bierzemy się za ćwiczenie praktyczne.

Zbigniew Raabe
e-mail: **zbigniew.raabe@edw.com.pl**

KONKURS

BASCOM College

na ciekawe aplikacje zaprojektowane na płytce testowej AVT-2500

Płytka testowa AVT-2500 może służyć nie tylko do nauki i testowania mniej skomplikowanych programów, ale także może stać się uniwersalnym narzędziem warsztatowym o ogromnych możliwościach. Zobaczyliśmy jak można wykonać prostą nawijarkę do cewek. Równie dobrze można przy pomocy płytki testowej wykonać w parę minut miernik częstotliwości, zegar, itd.

Takich urządzeń może być wiele i to niekoniecznie pracujących w naszym warsztacie. Płytka testowa może też być swojego rodzaju terminalem komputera: możemy przecież skompilować napisany program i uruchomić go z poziomu emulatora sprzętowego, bez konieczności programowania procesora.

Podam Wam prosty, banalny przykład. Zbliżają się święta Bożego Narodzenia i potrzebujemy w prosty sposób, bez ponoszenia kosztów zbudować urządzenie sterujące w efektowny sposób lampkami na choinkę. Na płytce testowej mamy driver mocy (ULN2803), mogący sterować ośmioma odbiornikami energii pobierającymi prąd o wartości do 500mA. Cóż wiec prostszego: należy dołączyć do tego drivera girlandy lampek, jego wejścia wysterować z wyprowadzeń procesora i napisać program, którego efekt działania będzie zależeć wyłącznie od naszej wyobraźni.

Wspominałem już Wam o nowym programatorze, który sprowadza wszystkie operacje związane z programowaniem procesora i testowaniem programu w uruchamianym układzie do jednego naciśnięcia klawisza F7 (możecie go obejrzeć na załączonej fotografii na stronie 22 inicjującej wykład). Zmontowałem kilka prototypów tego układu: jeden jest w Pracowni Konstrukcyjnej AVT, drugi pojechał do Holandii, trzeci pozostawiłem sobie do dalszych testów. Pozostały jeszcze dwa, w których, tak jak w pozostałych nie trzeba było wykonywać jakichkolwiek poprawek. Może komuś z Was mogą się przydać?

Te dwa programatory przeznaczyłem na nagrodę dla tych dwóch Czytelników, którzy przyślą najciekawsze pomysły na nowe, praktyczne zastosowania naszej płytki testowej AVT-2500. Oczywiście, sam pomysł jest najważniejszy, ale chciałbym także sprawdzić Wasze umiejętności i obejrzeć listingi napisanych programów. Nie ma żadnych ograniczeń, co do rodzaju urządzeń zbudowanych na płytce testowej. Narzędzia warsztatowe, układy automatyki domowej, wszystko co wymyślicie może być zaakceptowane. Ograniczenie jest tylko jedno: nie można stosować żadnych dodatkowych układów elektronicznych, z wyjątkiem najprostszych, takich jak oporniki, fotorezystory czy pojedyncze tranzystory. Nie ma natomiast ograniczeń jeżeli chodzi o układy elektromechaniczne: silniki DC i krokowe, serwomechanizmy, elektromagnesy, cały ten złom jest mile widziany.

Ogłaszamy zatem konkurs na ciekawe pomysły użytecznych aplikacji płytki testowej. Ponieważ mamy wakacje, więc rozstrzygnięcie konkursu planujemy dopiero po 30 września.

Oprócz dwóch wymienionych już nagród głównych przewidujemy opublikowanie na łamach EdW najciekawszych projektów. Każdy opublikowany projekt nagrodzimy darmową prenumeratą roczną EdW.

Zbigniew Raabe

Z ostatniej chwili

1. Współpraca pomiędzy MCS Electronics a redakcją Elektroniki Praktycznej, reprezentowaną przez moją skromną osobę, zaowocowała ostatnio skonstruowaniem, ośmielam się twierdzić, rewelacyjnego urządzania przeznaczonego do współpracy z pakietem BASCOM8051 SEfEP. Jest nim ... programator ISP procesorów podrodziny 89CX051! Oczywiście nie jest to prawdziwe ISP, ale z powodzeniem go udaje. Procesor włożony jest w podstawkę programatora i połączony kablem z wtykiem emulacyjnym umieszczonym w uruchamianym układzie. Przed rozpoczęciem programowania procesor zostaje odłączony od przeznaczonego dla niego układu, zaprogramowany i ponownie połączony ze swoim naturalnym środowiskiem. Wszystkie te czynności odbywają się całkowicie automatycznie i kilka sekund po naciśnięciu przycisku F7 możemy już sprawdzić działanie programu w rzeczywistych warunkach. W chwili, kiedy piszę te słowa, programator odbywa właśnie podróż do Holandii, gdzie zostanie dodatkowo przetestowany przez Marka. Jeżeli wyniki testów okażą się pomyślne, co jest prawie pewne, to opis układu zostanie opublikowany w jednym z najbliższych numerów Elektroniki Praktycznej.

2. MCS Electronics wyraźnie nabrała tempa w doskonaleniu młodszego "brata" naszego BASCOM-a, czyli pakietu BASCOM AVR, przeznaczonego do procesorów RISC, produkowanych przez ATMEL'a. W ciągu ostatniego miesiąca ukazały się kolejno dwie nowe edycje tego programu, a także udoskonalona wersja DEMO, dostępna na stronie *www.ep.com.pl* <*http://www.ep.com.pl*> lub *www.mcselec.com* <*http://www.mcselec.com*>. Rozbudowany pakiet zawiera nową wersję symulatora programowego, niestety jeszcze nie oczyszczonego ze wszystkich błędów. Jednak Mark obiecuje, że do końca roku BASCOM AVR stanie się w pełni profesjonalnym programem, nie ustępującym BASCOM-owi 8051.

3. 31 maja. Nowa wersja, a właściwie uaktualnienie BASCOM-a 8051 na stronie www.mcselec.com <*http://www.mcselec.com*> ! Jest to wersja Beta i Autor bynajmniej nie twierdzi, że jest pozbawiona błędów w nowo dodanych funkcjach (w końcu po to tworzy się wersję Beta!). Mimo to warto spróbować! Po przetestowaniu napiszę, co o tym sądzę.

BASCOM Forum

Szanowny Panie Profesorze!

*Trzy podane przez Pana przykłady działają bez zarzutu, jeśli wykonuje się symulację programową. Jednak gdy zacząłem symulować na sprzęcie, zaczęły się też kłopoty. Wyświetlacz LCD, który otrzymałem w kicie, zachowuje się dziwnie. W konfiguracji 16*1 polecenie Lcd "Elektronika dla Wszystkich" wyświetli tylko "Elektron", czyli 8 znaków. W konfiguracji 16*1a to samo polecenie wyświetli poprawnie 16 znaków napisu. I wszystko jest OK do momentu użycia Shiftlcd - obie połówki wyświetlacza przewijają się niezależnie. Po kilku eksperymentach zauważyłem, że pierwsze 8 znaków to jakby pierwsza linia wyświetlacza, pozostałe zaś to jakby druga linia. Obie linie posiadają 40 znaków pamięci. Zatem aby móc używać Shiftlcd i korzystać z całego wyświetlacza, musiałem użyć małej sztuczki. Oto listing programu:*

```
Config Lcd = 16 * 1                          'konfiguracja LCD: 1 linia 16 znaków
Cls
Lcd "Test wyswietlacza LCD, Artur Krajewski. " ' 40 znaków, na wyświetlaczu 8
pierwszych
Lowerline                    'użycie drugiej linii, i jak to się ma do konfiguracji?
Lcd "wietlacza LCD, Artur Krajewski. Test wys"      'przesuniecie o 8 znaków
CursorOff
Do
    Shiftlcd Left
    Waitms 1                                  ' opóźnienie wg potrzeby
 Loop
 End
                                              ' dziwne? a jednak działa...
```

Dokładnie, jak już skończyłem się dziwić, pomyślałem, że jeśli ja mam taki wyświetlacz (SDEC 94V-0 P-S1C16CT), to pewnie inni "koledzy z roku" mogą mieć taki sam i postanowiłem napisać do Profesora tego maila.

Pozdrowienia
Artur Krajewski

Phone: **+48 (0) 601 72 99 16**
Pager: **+48601729916@text.plusgsm.pl**
<mailto: +48601729916@text.plusgsm.pl>
E-mail: **artek@artek.priv.pl**

Szanowny Panie Zbigniewie!

Nazywam się Michał Waśkiewicz i mam 13 lat. EdW czytam od numeru 2/1999. Gdy przeczytałem o BASCOM College, od razu się zapisałem do tej szkoły. Gdy zacząłem wykonywać i ćwiczyć Wykład 2 zauważyłem, że BASCOM nie ma automatycznego startu z pliku, tj. nie uruchamia się po dwukrotnym kliknięciu na plik. Na pewno nie tylko mnie to przeszkadza, ale również innym uczniom BASCOM College. Poniżej przedstawiam rozwiązanie tego problemu. Należy włączyć okno "Mój komputer", a następnie kliknąć Widok/Opcje folderów i wybrać zakładkę "Typy plików".Wybieramy "Nowy typ...". W polu "Opis typu" należy wpisać jakąś nazwę kojarzącą się z programem (np. BASCOM 8051).W polu "Skojarzone rozszerzenie" wpisujemy "bas", a w polu "Typ zawartości (MIME)" wpisujemy "dokument BASCOM". Następnie wybieramy "Nowy..." gdzie w polu akcja wpisujemy "open" a w polu "Aplikacja używana do wykonania akcji" wpisujemy ścieżkę do programu BASCOM-a po tej ścieżce "%1"(łącznie z cudzysłowem). Następnie wciskamy OK i znowu OK. Zamykamy kolejne okno wciskając OK i mamy ustawiony automatyczny start plików z rozszerzeniem "bas". Pliki te otrzymały ikonę programu BASCOM 8051.W taki sposób w Windows 98 można rozwiązać ten problem.

Z poważaniem ***Michał Waśkiewicz***
(mwaskiew@go2.pl <mailto:mwaskiew@go2.pl>)

P.S. Zapomniałem powiedzieć, że nie odrobiłem pracy domowej z BASCOM. :)

```
Rem Bascom College - Wyklad 2 - Zadanie domowe
Rem Program obslugi silnika krokowego

Rem (C) 2000 by Artur Krajewski
$sim
P3 = 0

Rem Zerowanie portu P3
Rem zapobiegnie podaniu napiecia na wszystkie 4 fazy silnika

    Dim Tab_full(8) As Byte
      Tab_full(1) = 241
      Tab_full(2) = 242
      Tab_full(3) = 244
      Tab_full(4) = 248
      Tab_full(5) = 241
      Tab_full(6) = 242
      Tab_full(7) = 244
      Tab_full(8) = 248

    Rem Tabela wartosci podawanych na port P3 przy sterowaniu "pelnym"
    Rem Wartosci 1-4 sa takie same jak 5-8, co teoretycznie jest
    Rem marnotrastwem 4 bajtow pamieci RAM, jednak umozliwia znaczne
    Rem uproszczenie konstrukcji programu co zaoszczedza sporo ROM'u

    Dim Tab_half(8) As Byte
      Tab_half(1) = 241
      Tab_half(2) = 243
      Tab_half(3) = 242
      Tab_half(4) = 246
      Tab_half(5) = 244
      Tab_half(6) = 252
      Tab_half(7) = 248
      Tab_half(8) = 249

Rem Tabela wartosci podawanych na port P3 przy sterowaniu "polowkowym"

Rem Istotne sa tylko 4 mniej znaczace bity portu P3 (sterowanie)
Rem Bitom bardziej znaczacym przypisywana jest "1" co powoduje
Rem wprowadzenie ich w tryb z mozliwoscia odczytu
Dim Krok As Byte
  Krok = 1

Rem Zmienna pamietajaca "polozenie" silnika

Rem Glowna petla programu

Do

  If P3.4 = 0 Then
    P3 = Tab_half(krok)
  Else
    P3 = Tab_full(krok)
  End If
  Rem Sprawdzenie trybu pracy silnika

  If P3.5 = 0 Then
    Krok = Krok -  1
  Else
    Krok = Krok + 1
  End If

  Rem Sprawdzenie kierunku pracy silnika

  If Krok = 0 Then
    Krok = 8
  End If

  If Krok = 9 Then
    Krok = 1
  End If

  Rem Utrzymanie zmiennej "krok" w przedziale <1,8>
  Rem Konstrukcja nie jest moze zbyt "szybka" ale za to czytelna

  If P3.7 = 0 Then
    Wait 1
  Else
    Waitms 50
  End If

  Rem Sprawdzenie "szybkości" pracy silnika

Loop

Rem That's all, Folks...
```

Artur Krajewski