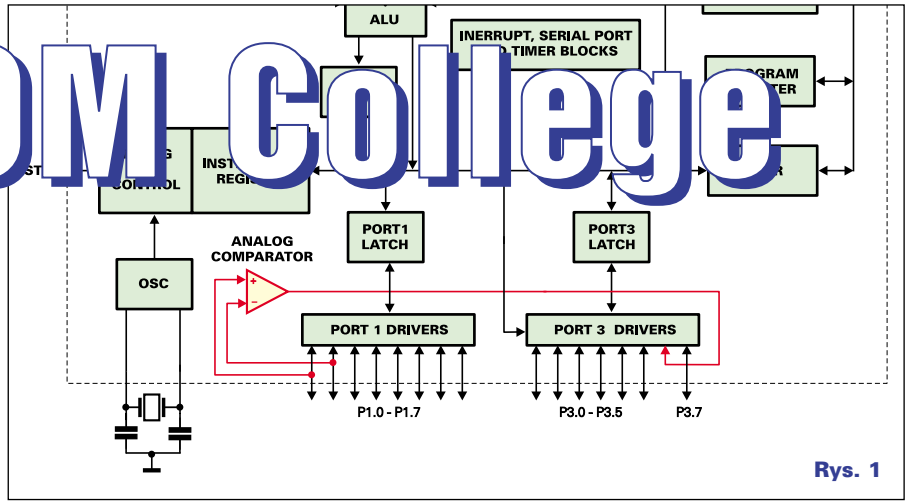
BASCOM College



Rys. 1

Wykład 7

Procesor a pomiary wartości analogowych oraz inne, drobne sprawy

Miło mi spotkać się z Wami na kolejnym wykładzie BASCOM College. Poprzedni wykład, traktujący o magistrali I²C był wyjątkowo trudny i ważny, tak więc dzisiejsza lekcja będzie miała nieco wypoczynkowy charakter. Więcej, będzie ona czymś w rodzaju "koncertu życzeń", czyli poruszać będziemy tematy, o które Studenci BASCOM College najczęściej pytają w e-mailach.

Zadałem sobie nieco trudu, przejrzałem całą do tej pory otrzymaną pocztę i postarałem się posegregować listy tak, aby dowiedzieć się, jakie tematy sprawiają Wam najwięcej problemów i na opracowanie jakich czekacie ze szczególną niecierpliwością. Znalazłem też wiele listów z prośbami o wytłumaczenie działania poszczególnych poleceń języka MCS BASIC. Te tematy, które powtarzały się najczęściej zostaną omówione na dzisiejszej lekcji.

Dzisiejsza lekcja będzie miała nieco nietypową formę, ponieważ zostanie połączonaw jedną całość z ćwiczeniami. Spowodowane jest to dużą różnorodnością poruszanych na wykładzie tematów i chęcią skupienia całego przerabianego materiału w jedną całość.

Wracajmy zatem do Waszych listów i problemów nad jakimi się zastanawiacie. Muszę przyznać, że byłem nieco zaszokowany wynikami przeprowadzonej przeze mnie analizy. Okazuje się, że jednym z największych problemów jest dla Was ... wykonywanie przez procesor pomiarów wartości analogowych! Co, mało Wam było tych oporniczków, kondensatorków, wzmacniaczy operacyjnych i innego elektronicznego śmiecia? Żarty, żartami, ale pytania o pomiar napięcia, a pośrednio także wartości nieelektrycznych były jak najbardziej na miejscu. Wprawdzie dysponujemy już ogromną gamą najrozmaitszych czujników, mierzących temperaturę, ciśnienie czy poziom oświetlenia i dostarczających informację o wynikach pomiarów bezpośrednio w postaci cyfrowej, ale ich stosowanie nie zawsze ma sens ekonomiczny. Trudno przecież angażować relatywnie kosztowny termometr cyfrowy DS1820 do zgrubnego pomiaru temperatury czy też instalować w systemie przetwornik ADC w celu stwierdzenia, czy zapadł już zmierzch. Wiele prostych pomiarów można wykonać bezpośrednio za pomocą "gołego" procesora, co najwyżej wyposażonego w jeden czy dwa elementy dyskretne.

W kilku listach zapytywaliście, jak zmierzyć napięcie za pomocą procesora. Pytania te zadane były nieściśle: brak było definicji typu procesora. Możliwości naszego '2051 w zakresie pomiaru wartości analogowych nie są imponujące, ale mamy przecież do dyspozycji procki innych typów. Wymienię tu dla przykładu rewelacyjny chip AT90S8535, wyposażony w osiem przetworników analogowo-cyfrowych o rozdzielczości 10 bitów, lub też ADUC812, "analogowy" procesor z przetwornikami o rozdzielczości .... 24 bitów! Używając takich chipów wystarczy z poziomu MCS BASIC wydać polecenie GETAD, i po sprawie! Sadzę jednak, że zadając pytanie o pomiary wartości analogowych mieliście raczej na myśli nasz poczciwy '2051.

Pomiar wartości analogowych za pomocą procesora 89C2051

Odpowiedź na pytanie o pomiar wartości analogowych za pomocą procesora '2051 chciałbym połączyć z drugą odpowiedzią, tym razem na pytanie o nieco dziwne nazwanie wyprowadzeń portu 3 naszego '2051. Rzeczywiście, z pozoru wygląda to nieco nielogicznie: mamy najpierw pięć pinów ponumerowanych od 0 do 5, a następnie od razu pin 7. A gdzie pin 6? Wiadomo, że nie dałoby się wyprowadzić wszystkich wyjść dwóch portów w procesorze zapakowanym w dwudziestopinową obudowę, ale dlaczego taka dziwna numeracja? Otóż, wynika to z tego, że w rzeczywistości wyprowadzenie 6 portu 3 istnieje i jest dostępne do operacji odczytu! Jednak, aby to udowodnić, musimy najpierw zajrzeć do środka procesora '2051.

Na **rysunku 1** (powyżej) widać fragment "wnętrzności" naszego ulubionego procesorka. Nie zajmowaliśmy się nimi do tej pory, ale nadszedł czas, kiedy musimy wreszcie to zrobić. Zajmiemy się zresztą tylko małym kawałeczkiem struktury procesora, tym zaznaczonym kolorem czerwonym. No i co my tu widzimy? Nie nic innego, jak wzmacniacz operacyjny, z wejściami dołączonymi do wyprowadzeń 0 i 1 portu 0 i z wyjściem, no właśnie, wyjście wzmacniacza to ten "zagubiony", szósty pin portu trzeciego! Niestety, jak już wiemy pin P3.6 nie został wyprowadzony na zewnątrz procesora i nie mamy możliwości zastosowania w tym wzmacniaczu jakichkolwiek sprzężeń zwrotnych. A zatem, może on pracować włącznie jako komparator napięciowy. Oczywiście, fakt wykorzystania pinów P1.0 i P1.1 jako wejść wzmacniacza operacyjnego w niczym nie przeszkadza podczas używania jako normalnych wejść lub wyjść cyfrowych. Musimy jednak pamiętać, że są to wyprowadzenia typu OPEN COLLECTOR i jeżeli mamy zamiar używać ich jako wyjść, to najczęściej niezbędne będzie dołączenie do nich zewnętrznych rezystorów podciągających, tzw. pullupów.

Nie wiem jak Was, ale mnie ten rysuneczek zamieszczony w karcie katalogowej procesora ‘2051 zupełnie nie przekonał. Narysować można wszystko, a my przecież kierujemy się słuszną zasadą sprawdzania podanych nam informacji w praktyce. A zatem napiszemy sobie banalnie prosty programik, który po skompilowaniu umieścimy we wnętrzu procesora. Tu ważna uwaga: wszystkie ćwiczenia związane z pomiarami wartości analogowych musimy wykonywać wyłącznie z zaprogramowanym procesorem. Symulacja programowa nie miałaby sensu, a nasz prosty i tani emulator sprzętowy nie jest w stanie “przepuszczać” danych analogowych. Nie mamy zatem wyjścia i musimy zaprogramować procesor następującym, wyjątkowo skomplikowanym programem:

```
Config Lcd = 16 * 1a
Cursor Off
Dim X As Bit
Cls
Do
            X = P3.6
            Lcd " Pin p3.6= " ; X
            Waitms 200
            Cls
Loop
```

A więc próbujemy dokonać czegoś, co do niedawna wydawało się niemożliwe: odczytać stan wyprowadzenia 6 portu 3! Zanim jednak włożymy zaprogramowany procesor w podstawkę, musimy dokonać jeszcze jednej operacji. Za pomocą dwóch z naszych zakończonych wtykami kabelków połączmy pin P1.0 z plusem zasilania, a pin P1.1 z masą i wreszcie uruchommy nasz program.

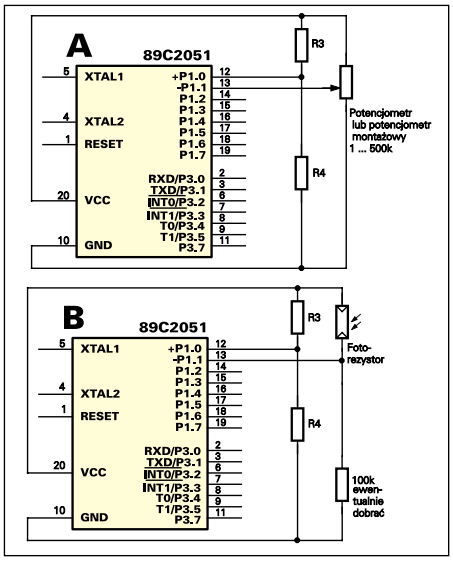
Na ekranie wyświetlacza ukazała się informacja, że na “niewidzialnym” wyjściu portu 3 mamy logiczny stan wysoki. To jakby się zgadzało, ale co się stanie, jeżeli zamienimy kabelki miejscami i na wejściu P1.0 wymusimy stan niski, a na P1.1 stan wysoki? Przeprowadzony błyskawicznie eksperyment wykazał, że zgodnie z oczekiwaniami na wyjściu 6 portu 3 pojawił się stan niski.

Jak dotąd nie uzyskaliśmy jeszcze wiele, ponieważ w zasadzie porównujemy ze sobą dwa różne stany logiczne, a nie napięcia. A zatem dokonajmy drobnej modyfikacji naszego hardware’u zgodnie z **rysunkiem 2A**. Jako rezystory dołączone do pinu P1.0 wykorzystamy dwa “wolne” oporniki, umieszczone przezornie na naszej płytce testowej, a potencjometr (lub potencjometr montażowy) z pewnością znajdziemy w czeluściach naszych szuflad z elektronicznymi skarbami. Wartość tego potencjometru może być w zasadzie dowolna. Po ponownym uruchomieniu programu i powolnym pokręcaniu potencjometrem, z pewnością okaże się, że stan wyjścia P3.6 zmienia się mniej więcej w połowie pełnego kąta obrotu potencjometru, wtedy gdy napięcie na jego środkowym wyprowadzeniu jest równe połowie napięcia zasilania (rezystory na płytce testowej mają równą wartość). Jest to już dowód na to, że producent ‘2051 nie oszukiwał nas i we wnętrzu procesora umieszczony jest pełnosprawny komparator analogowy.

Jaki jednak możemy mieć praktyczny pożytek z tego elementu architektury ‘2051? Z pewnością znajdzie się takich zastosowań wiele, a my rozpatrzmy tylko jeden przykład. Budujemy jakieś urządzenie (np. robota na konkurs!), które musi wiedzieć, jaki jest stan oświetlenia w pomieszczeniu, w którym się znajduje. Zajrzyjcie teraz do waszych zapasów i poszukajcie jakiegoś fotorezystora lub fototranzystora. Typ zupełnie dowolny, byle tylko działały! Jeśli znaleźliście już te elementy i do tego jeszcze jeden rezystorek o rezystancji mniej więcej równej rezystancji posiadanego fotoopornika w średnich warunkach oświetleniowych, to zmontujcie sobie na płytce testowej układzik pokazany na **rysunku 2B**, a następnie zmodyfikujmy nieco nasz program:

```
Config Lcd = 16 * 1a
Cursor Off
Dim X As Bit
Cls
Do
  X = P3.6
  If X = 0 Then
  Lcd "    Ale jasno!"
  Else
  Lcd "Oj, jak ciemno!"
  End If
  Waitms 200
  Cls
Loop
```

Rys. 2



Po zaprogramowaniu procesora i uruchomieniu programu możemy chwilę poeksperymentować, zakrywając i odsłaniając fotorezystor.

Skutek działania tak napisanego programu nie jest może zbyt odkrywczy, ale przecież to tylko przykład. Pamiętajcie, że zamiast fotorezystora możecie przecież zastosować np. termistor lub precyzyjny termometr w rodzaju LM35, a jeden z rezystorów w dzielniku napięciowym zastąpić potencjometrem regulacyjnym. Uzyskamy wtedy możliwość wywołania określonej reakcji programu procesora na dokładnie określoną zmianę warunków zewnętrznych. A to już jest coś!

Z pewnością wielu z Was, Drodzy Studenci BASCOM College, to ”coś” jednak nie zadowoli! W listach pytaliście przecież, o możliwość POMIARU wartości elektrycznych i nieelektrycznych, a nie ich porównywanie! Za chwilę przejdziemy do przeprowadzenia próby dokonania pomiaru wartości rezystancji lub pojemności, lecz najpierw muszę Was o czymś uprzedzić. Dokonywanie takich pomiarów jest możliwe, lecz przy zastosowaniu prostych środków i nieskomplikowanego hardware’u nie będą to pomiary zbyt dokładne. Oczywiście, ich dokładność możemy zwiększyć przez rozbudowywanie oprogramowania lub dodanie dodatkowych elementów sprzętowych, ale z pewnością w pewnym momencie zadamy sobie pytanie o sens dalszych tego typu działań. Dojdziemy bowiem do momentu, w którym najprostszym rozwiązaniem okaże się dodanie zewnętrznego przetwornika analogowo-cyfrowego lub wręcz zastosowanie procesora z wbudowanym takim przetwornikiem (np. ‘80515, ‘80535, ‘80517, ‘80535 lub jeden z procesorów rodziny AVR). Niemniej, jeżeli nie jest wymagana zbyt wielka dokładność pomiaru, możemy sobie poradzić nawet dysponując “gołym” ‘2051.

Z pewnością większość z Was zauważyła już tajemnicze złącza umieszczone z prawej strony naszej płytki testowej. Złącza te oznaczone są symbolami R i C i mają także wyprowadzone końcówki zasilania i masy. W ich bezpośrednim sąsiedztwie znajduje się jeszcze styk, za pomocą którego możemy połączyć to złącze z procesorem. Mogę już teraz powiedzieć Wam, że są to elementy służące do przeprowadzania eksperymentów z pomiarem rezystancji i pojemności, a pośrednio także wielkości nieelektrycznych. Zmontujmy sobie zatem prosty układzik, pokazany na **rysunku 3,** i napiszmy następujący program:

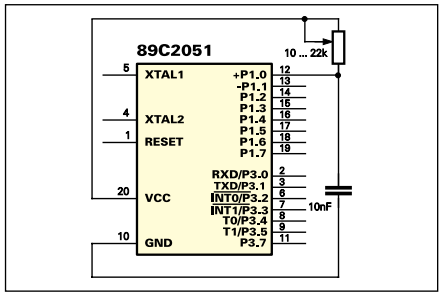
```
$crystal = 11059200  ‘określenie częstotliwości rezonatora kwarcowego
Config Timer0 = Timer , Gate = Internal , Mode = 1
                ‘konfiguracja timera niezbędnego do
  ‘prawidłowego funkcjonowania polecenia GETRC
Dim W As Word

Do
                W = Getrc(p1.0)
‘określenie pinu, do którego zostały dołączone mierzone elementy
                Lcd W
                Waitms 200
                Cls
Loop
```

Po uruchomieniu programu poobserwujmy chwilę dane pojawiające się na wyświetlaczu LCD, pokręcając jednocześnie potencjometrem. Jeżeli dysponujemy omomierzem, to możemy sporządzić sobie tabelkę przeliczeniową, w rodzaju tej przygotowanej przez MCS Elec-

tronics i sprawdźmy, czy otrzymywane dane są proporcjonalne do mierzonej wartości.

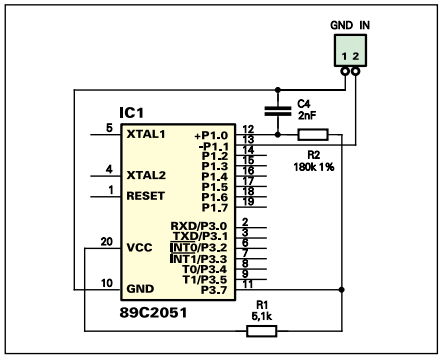
Wynik pomiaru	Rzeczywista wartość R
250	10900
198	9020
182	8040
166	7000
154	6020
138	5040
122	4040
106	3060
86	2160
54	1000
22	198
18	150
10	104
6	1



Rys. 3

Możemy także wykonać kilka eksperymentów z pomiarem względnej wartości kondensatorów, przy znanej i nie zmienianej wartości rezystancji.

Obawiam się jednak, że analiza otrzymanych wyników pomiarów doprowadziła Was raczej do smutnych wniosków. Wyniki pomiarów są bardzo nieliniowe i ich wykorzystanie wymagałoby albo zastosowania rozbudowanych tablic przeliczeniowych, albo rozbudowy hardware'u. Zastosowanie tabel wiąże się ze znacznym zwiększeniem obszaru zajmowanego przez program w pamięci procesora. Tabele przeliczeniowe służące do konwersji wyników pomiaru napięcia, z którymi zapoznamy się za chwilę, zajmują w pamięci ROM aż 620 bajtów, czyli prawie jedną trzecią całego jej obszaru. Jeżeli więc uprzemy się, aby dokonywać dokładnych pomiarów za pomocą naszego '2051, to chyba najlepszą metodą byłoby rozbudowanie warstwy sprzętowej o dodatkowy element - źródło prądowe o regulowanej i stabilizowanej wartości wypływającego z niego prądu. Przykład takiego rozwiązania pokazuje **rysunek 4**.



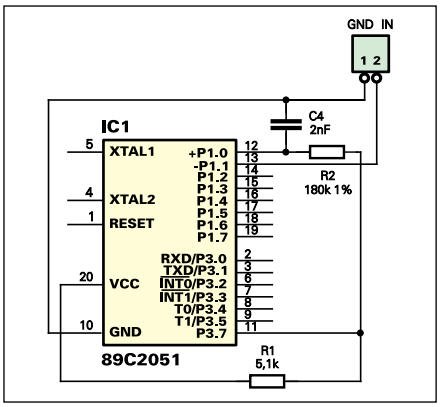
Rys. 4

Zajmijmy się teraz kolejnym ciekawym tematem, jakim jest bezpośredni pomiar napięcia wykonywany za pomocą procesora '2051. Mark przygotował specjalne polecenie służące temu celowi: GETAD2051, dostępne jak na razie w wersji beta, przeznaczonej do testowania. Powiem Wam coś szczerze: moim zdaniem cała ta zabawa nie ma większego sensu. Nie tyle sam pomiar, co przeliczenie otrzymanych w jego efekcie danych jest tak kłopotliwe i zajmuje tak wiele miejsca w pamięci, że moim zdaniem gra nie jest warta świeczki, szczególnie przy obecnych cenach prostych przetworników analogowo-cyfrowych, a nawet procesorów z wbudowanymi takimi przetwornikami. Jednak nie wszystko, czego się uczymy musi mieć natychmiastowe zastosowanie w praktyce, tak więc zapoznamy się z poleceniem GETAD2051 bez dalszych komentarzy.

Do odczytywania wartości napięcie przyłożonego do pinu P1.1 procesora '2051 napiszemy krótki program, pozornie załatwiający wszystkie problemy związane z pomiarem napięcia. Przedtem jednak musimy przygotować sobie prosty układ, którego schemat został pokazany na **rysunku 5**.

```
$regfile = "89c2051.dat"
Dim A As Byte
Do
 A = Getad2051()
Lcd A
Loop
```

Tak napisany program możemy skompilować i uruchomić w zaprogramowanym procesorze. Jednak skutki jego działania będą, jak na razie, mizerne. Wyświetlane wyniki nijak się mają do rzeczywistego napięcia istniejącego na wejściu P1.1 procesora, po dokonaniu kilku pomiarów i "ręcznym" przeliczeniu ich wartości łatwo możemy stwierdzić, że ich liniowość także pozostawia wiele do życzenia. Aby uzyskać poprawne wyniki pomiaru, należy dodać do programu liczącą ... 159 linii tabelę przeliczeniową i instrukcję odczytywania zawartych w niej danych: A = Lookup(a, Dta), umieszczoną bezpośrednio przed poleceniem LCD A. Jak już wspomniałem, wszystko to razem zajmuje ponad 600 bajtów pamięci ROM.



Rys. 5

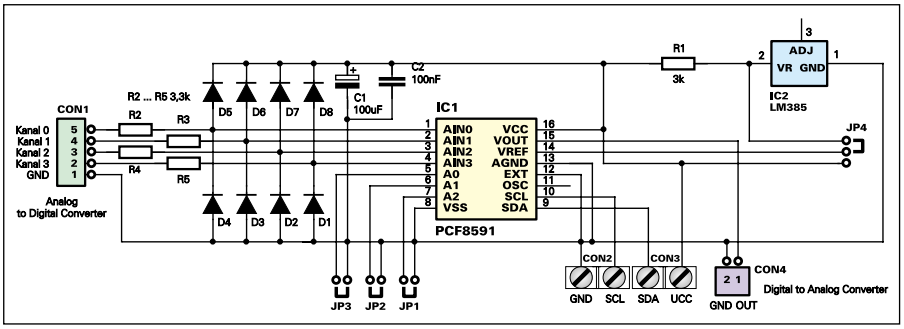
Dla zainteresowanych i lubiących śmiałe, chociaż nie zawsze użyteczne eksperymenty podaję dane zawarte w tabeli przeliczeniowej (w ramce poniżej).

Sądzę, że wyczerpaliśmy już temat dokonywania pomiarów analogowych za pomocą procesora '2051. Do tematu takich pomiarów będziemy jeszcze wielokrotnie wracać, ale w kontekście wykorzystywania przetworników analogowo-cyfrowych wbudowanych w procesory lub będących samodzielnymi

```
Dta:
Data 0     ' 0 0.000
Data 1     ' 1 0.047
Data 1     ' 2 0.093
Data 2     ' 3 0.138
Data 2     ' 4 0.184
Data 3     ' 5 0.229
Data 3     ' 6 0.273
Data 3     ' 7 0.317
Data 4     ' 8 0.361
Data 4     ' 9 0.404
Data 5     ' 10 0.447
Data 5     ' 11 0.489
Data 6     ' 12 0.531
Data 6     ' 13 0.573
Data 6     ' 14 0.614
Data 7     ' 15 0.655
Data 7     ' 16 0.696
Data 8     ' 17 0.736
Data 8     ' 18 0.776
Data 8     ' 19 0.815
Data 9     ' 20 0.854
Data 9     ' 21 0.893
Data &H10  ' 22 0.931
Data &H10  ' 23 0.969
Data &H10  ' 24 1.006
Data &H11  ' 25 1.044
Data &H11  ' 26 1.080
Data &H11  ' 27 1.117
Data &H12  ' 28 1.153
Data &H12  ' 29 1.189
Data &H12  ' 30 1.224
Data &H13  ' 31 1.260
Data &H13  ' 32 1.295
Data &H13  ' 33 1.329
Data &H14  ' 34 1.363
Data &H14  ' 35 1.397
Data &H14  ' 36 1.431
Data &H15  ' 37 1.464
Data &H15  ' 38 1.497
Data &H15  ' 39 1.530
Data &H16  ' 40 1.562
Data &H16  ' 41 1.594
Data &H16  ' 42 1.626
Data &H17  ' 43 1.657
Data &H17  ' 44 1.688
Data &H17  ' 45 1.719
Data &H18  ' 46 1.750
Data &H18  ' 47 1.780
Data &H18  ' 48 1.810
Data &H19  ' 49 1.840
Data &H19  ' 50 1.869
Data &H19  ' 51 1.898
Data &H19  ' 52 1.927
Data &H20  ' 53 1.956
Data &H20  ' 54 1.984
Data &H20  ' 55 2.012
Data &H21  ' 56 2.040
Data &H21  ' 57 2.068
Data &H21  ' 58 2.095
Data &H21  ' 59 2.122
Data &H22  ' 60 2.149
Data &H22  ' 61 2.176
Data &H22  ' 62 2.202
Data &H22  ' 63 2.228
Data &H23  ' 64 2.254
Data &H23  ' 65 2.279
Data &H23  ' 66 2.305
Data &H23  ' 67 2.330
Data &H24  ' 68 2.355
Data &H24  ' 69 2.379
Data &H24  ' 70 2.404
Data &H24  ' 71 2.428
Data &H25  ' 72 2.452
Data &H25  ' 73 2.476
Data &H25  ' 74 2.499
Data &H25  ' 75 2.523
Data &H26  ' 76 2.546
Data &H26  ' 77 2.569
Data &H26  ' 78 2.591
Data &H50  ' 79 5.000
Data &H49  ' 80 4.953
Data &H49  ' 81 4.907
Data &H48  ' 82 4.862
Data &H48  ' 83 4.816
Data &H47  ' 84 4.771
Data &H47  ' 85 4.727
Data &H47  ' 86 4.683
Data &H46  ' 87 4.639
Data &H46  ' 88 4.596
Data &H45  ' 89 4.553
Data &H45  ' 90 4.511
Data &H44  ' 91 4.469
Data &H44  ' 92 4.427
Data &H44  ' 93 4.386
Data &H43  ' 94 4.345
Data &H43  ' 95 4.304
Data &H42  ' 96 4.264
Data &H42  ' 97 4.224
Data &H42  ' 98 4.185
Data &H41  ' 99 4.146
Data &H41  ' 100 4.107
Data &H40  ' 101 4.069
Data &H40  ' 102 4.031
Data &H40  ' 103 3.994
Data &H39  ' 104 3.956
Data &H39  ' 105 3.920
Data &H39  ' 106 3.883
Data &H38  ' 107 3.847
Data &H38  ' 108 3.811
Data &H38  ' 109 3.776
Data &H37  ' 110 3.740
Data &H37  ' 111 3.705
Data &H37  ' 112 3.671
Data &H36  ' 113 3.637
Data &H36  ' 114 3.603
Data &H36  ' 115 3.569
Data &H35  ' 116 3.536
Data &H35  ' 117 3.503
Data &H35  ' 118 3.470
Data &H34  ' 119 3.438
Data &H34  ' 120 3.406
Data &H34  ' 121 3.374
Data &H33  ' 122 3.343
Data &H33  ' 123 3.312
Data &H33  ' 124 3.281
Data &H32  ' 125 3.250
Data &H32  ' 126 3.220
Data &H32  ' 127 3.190
Data &H31  ' 128 3.160
Data &H31  ' 129 3.131
Data &H31  ' 130 3.102
Data &H31  ' 131 3.073
Data &H30  ' 132 3.044
Data &H30  ' 133 3.016
Data &H30  ' 134 2.988
Data &H29  ' 135 2.960
Data &H29  ' 136 2.932
Data &H29  ' 137 2.905
Data &H29  ' 138 2.878
Data &H28  ' 139 2.851
Data &H28  ' 140 2.824
Data &H28  ' 141 2.798
Data &H28  ' 142 2.772
Data &H27  ' 143 2.746
Data &H27  ' 144 2.721
Data &H27  ' 145 2.695
Data &H27  ' 146 2.670
Data &H26  ' 147 2.645
Data &H26  ' 148 2.621
Data &H26  ' 149 2.596
Data &H26  ' 150 2.572
Data &H25  ' 151 2.548
Data &H25  ' 152 2.524
Data &H25  ' 153 2.501
Data &H25  ' 154 2.477
Data &H24  ' 155 2.454
Data &H24  ' 156 2.431
Data &H24  ' 157 2.409
```

układami. Z dostępnych na rynku przetworników najbardziej godnym polecenia wydaje się być układ Philipsa - PCF8591, zawierający w swojej strukturze aż cztery niezłej jakości ośmiobitowe przetworniki ADC i jakby tego było mało, na dodatek jeszcze jeden przetwornik DAC (Digital to Analog Converter), także o rozdzielczości ośmiobitowej. Wszystko to ładnie zapakowane w 14 pinową obudowę i sterowane poprzez magistralę I²C. Dla zainteresowanych podaję na **rysunku 6** schemat prostego układu, umożliwiającego pomiar wartości napięcia w czterech kanałach i wysłania na jedno wyjście wartości analogowej. Chciałbym też zwrócić Waszą uwagę, że dokładny opis tego układu zostanie opublikowany w jednym z najbliższych numerów Elektroniki Praktycznej.

Rys. 6



Wybrane polecenia i funkcje programu BASCOM8051

Co jeszcze sprawia Wam problemy i z jakimi problemami zwracaliście się do mnie? Kolejna grupa listów zawiera pytania o ... korzystanie z asemblera w programach pisanych w MCS BASIC. Pytania jak najbardziej sensowne: w programie pisanym w języku wysokiego poziomu wstawki asemblerowe mogą niekiedy być stosowane. Połączenie tych dwóch języków może pomóc nam zoptymalizować pisany program i osiągnąć jak najmniejszą długość kodu wynikowego. Oczywiście, BASCOM umożliwia takie operacje, a nawet napisanie całego programu w asemblerze. Jest tylko jeden problem: nie czuję się predestynowany do tłumaczenia Wam spraw związanych z pisaniem programów w tym języku. Asembler oczywiście znam, ale szczerze i z całego serca go nie cierpię. Jest to jeszcze jeden powód, dlaczego z takim entuzjazmem powitałem pojawienie się tak wspaniałego narzędzia, jak BASCOM.

Jeżeli pytań na temat techniki pisania podprogramów asemblerowych i implementowania ich w program napisany w MCS BASIC będzie więcej, to porozmawiam z naszym ekspertem od asemblera, redaktorem Sławkiem Surowińskim. Może to on napisze jeden wykład, przypominający nam asembler i zaprezentuje możliwości wykorzystywania tego języka w naszej pracy. Ja podam Wam tylko podstawowe zasady wklejania wstawek asemblerowych w "basicowy" program i niezbędne do tego celu polecenia. Zanim jednak to uczynię, chciałbym uprzytomnić Wam, ile możecie stracić stosując zbyt często asemblerowe "łaty" (oj, nie lubię ja tego asemblera, nie lubię!).

Jedną z najwspanialszych cech języków wysokiego poziomu, a w szczególności MCS BASIC jest to, że program napisany na jakiś typ procesora będzie działał bez większych przeróbek nawet z procesorem z zupełnie innej rodziny. Jeżeli więc napiszemy program np. na '2051, a w pewnym momencie okaże się, że zbyt wolna "pięćdziesiątka jedynka" nie wyrabia się z niektórymi funkcjami, to w każdej chwili, praktycznie bez przeróbek sprzętowych możemy zastosować wielokrotnie szybszy i nowocześniejszy procesor AVR AT90S2313, "pinowy" odpowiednik naszego '2051. Jednak nie kompatybilność sprzętowa jest tu najważniejsza: znacznie większe znaczenia ma to, że praktycznie nie będziemy musieli wprowadzać zmian w programie. Te zmiany, których w żaden sposób nie będziemy mogli uniknąć, najczęściej będziemy mogli wykonać "z automatu", stosując FIND\REPLACE w edytorze tekstowym. O resztę będzie martwił się kompilator.

Jeżeli jednak w programie zastosowaliśmy wstawki asemblerowe, to będziemy musieli napisać je od nowa, ponieważ asemblery procesorów z dwóch rodzin najczęściej znacznie się różnią od siebie.

Asemblerową "łatę" możemy umieścić w dowolnym miejscu programu napisanego w MCS BASIC. Początek podprogramu napisanego w asemblerze oznaczamy:

$ASM,

a koniec:

$END ASM

Kompilator BASCOM'a rozpoznaje wszystkie polecenia asemblerowe, których spis znajduje się w helpie, w rozdziale "ASM PROGRAMMING". Wyjątkiem jest polecenie **"SWAP"**, które jest także poleceniem języka MCS BASIC. Jeżeli chcemy użyć tego polecenia w wstawce asemblerowej, to musi ono zostać poprzedzone znakiem **"!"**, czyli napiszemy: **!SWAP**.

Przed chwilą zapoznaliśmy się z prostym programem służącym do dokonywania pomiarów napięcia za pomocą procesora '2051 wyposażonego jedynie w prosty hardware. Program ten, bez tablic przeliczeniowych, miał w pewnym uproszczeniu postać:

```
Do
 A = Getad2051()
Lcd A
Loop
```

Napiszmy jeszcze raz ten program, stosując zamiast polecenia GETAD2051 napisanego w języku wysokiego poziomu, jego odpowiednik asemblerowy:

```
$regfile = "89c2051.dat"
Dim A As Byte
Do
 $Asm
Clr P3.7
Mov a,#3
Lcall waitms
Clr a
Setb P3.7
Nop
Nop
_ad1:
Jb p3.6, _ad4
Inc a
Cjne a,#79,_ad1
Push acc
Mov a, #3
Lcall waitms
Pop acc
Clr a
Clr P3.7
Nop
Nop
_ad2:
Jnb p3.6, _ad3
Inc a
Cjne a, #79, _ad2
Dec a
_ad3:
Add a, #79
_ad4:
Ret
$End asm

Lcd A
Loop
```

Pozostawię porównanie obydwu programów bez komentarzy, co nie oznacza, że zniechęcam kogokolwiek do pisania programów w asemblerze.

W celu dołączenia do programu wstawki asemblerowej możemy także użyć wygodnego polecenia **$INCLUDE**, zawierającego jako parametr nazwę pliku z podprogramem asemblerowym i ścieżkę dostępu do niego. Na polecenie **$INCLUDE**, możemy także wstawić w tekst programu fragmenty napisane w języku MCS BASIC. Stosowanie tego polecenia może w wielu przepadkach znacznie przyspieszyć pisanie programu i pozwolić na uniknięcie wielu denerwujących błędów.

Każdy programista gromadzi sobie podprogramy, realizujące najrozmaitsze funkcje. Są to gotowe procedury, które można wstawiać w pisane programy. Im więcej takich procedur mamy, tym lepiej i tym szybciej będzie szła nam praca nad nowym programem. Najlepiej przechowywać je w osobnym podkatalogu wraz z dokładnym opisem pełnionych przez nie funkcji. Jednak nasze procedury nie są zwykle niezmienne i są stale doskonalane i ulepszane, zgodnie z ustawicznym rozwojem języka MCS BASIC. Jeżeli więc jakaś procedura została zmieniona, to musimy pamiętać także o modyfikacji każdego programu, w którym została użyta. Od konieczności pamiętania o tym zwalnia nas właśnie polecenie **$INCLUDE**. Zamiast kopiować całą sprawdzoną procedurę do tworzonego programu, piszemy:

```
$INCLUDE [ścieżka dostępu procedura]
np.:
$include C:\program files\mcs electronics\
        bascom-8051\procedury\obsluga eeprom.bas
```

Linia programu z poleceniem $INCLUDE jest równoważna wstawieniu w jego tekst kompletnej procedury, zajmującej nieraz wiele

miejsca. Powoduje to zwiększenie czytelności programu (ale w żadnym wypadku nie zmniejsza długości kodu wynikowego). Najważniejsze jest jednak co innego: jeżeli w danej procedurze dokonamy jakichkolwiek zmian lub ulepszeń, to zostaną one automatycznie uwzględnione w każdym programie, w którym umieściliśmy wywołanie tej procedury.

A teraz coś dla początkujących Studentów BASCOM College i Tych, którzy dołączyli do nas w ostatnim czasie. Z analizy listów, które od Was dostaję wynika, że wielu początkujących programistów nie za bardzo radzi sobie z najbardziej podstawowymi funkcjami MCS BASIC, czyli z deklaracją zmiennych i tablic. W przysyłanych przez Was listingach programów, niejednokrotnie napisanych bardzo pomysłowo, często występują błędy na poziomie podstawowym. Często w miarę poprawnie napisany program "wywala się" lub zabiera zbyt wiele miejsca w pamięci RAM właśnie z powodu tych elementarnych błędów i nieumiejętnej deklaracji zmiennych i tablic. Często duże problemy mają ci Studenci, którzy ... dobrze znają "komputerowy" QBASIC i nie zapoznali się z różnicami występującymi pomiędzy tym językiem a MCS BASIC. A zatem przypomnijmy sobie sprawy podstawowe, które być może potraktowałem na pierwszej lekcji zbyt ogólnikowo.

W przeciwieństwie do QBASICa MCS BASIC **wymaga deklarowanie każdej zmiennej**, a nie tylko tablic zawierających więcej niż dziesięć składników. Właśnie nieprzestrzeganie tej zasady było przyczyną wielu problemów, na jakie natrafili początkujący programiści. Deklarowanie każdej bez wyjątku zmiennej jest logiczną konsekwencją konieczności maksymalnego oszczędzanie pamięci RAM procesora. Każda zmienna musi mieć swoją własną, niepowtarzalną nazwę, składającą się z dowolnych znaków dostępnych z klawiatury. Nazwa zmiennej może mieć długość do 255 znaków.

W języku MCS BASIC występuje 7 rodzajów zmiennych. Ich rozróżnienie i właściwe stosowanie jest fundamentem powodzenia podczas pisania programu i właściwego zagospodarowania niezbyt pojemnej pamięci RAM, w której przechowywane są zmienne. Musicie pamiętać, że w przypadku procesora 'X051 macie do dyspozycji zaledwie 128 bajtów tej pamięci i że w związku z tym każdy zajęty niepotrzebnie bit może doprowadzić do jej przepełnienia. Musicie pamiętać, że program, w którym zmienna zadeklarowana jest jako bajt, a której nadajemy jedynie wartości 0 i 1, będzie pracował poprawnie, ale zupełnie niepotrzebnie zmarnowaliśmy siedem z 128 bitów RAM. Takich sytuacji musimy bezwzględnie unikać i nigdy nie deklarować zmiennych "na zapas".

Raz jeszcze przypomnijmy sobie rodzaje zmiennych języka MCS BASIC, a potem pomówimy, jak można redukować liczbę użytych zmiennych i odzyskiwać obszary pamięci RAM.

Nazwa zmiennej	Zakres wartości	Zajmowany RAM	Zastosowanie
Bit	0....1	1 bit	Tylko 0 i 1
Byte	0 ... 255	1 bajt	Liczby dodatnie w podanym zakresie
Integer	-32768 ... 32768	2 bajty	Liczby dodatnie i ujemne w podanym zakresie
Word	0 ... 65535	2 bajty	Liczby dodatnie w podanym zakresie
Long	-2147483648 ... 2147483647	4 bajty	Liczby dodatnie i ujemne w podanym zakresie
Single		4 bajty	Operacje na liczbach zmiennoprzecinkowych
String	do 254 znaków	do 255 bajtów	Definiowanie tekstów

Sposób stosowania pierwszych pięciu rodzajów zmiennych jest chyba oczywisty: zawsze deklarować możliwie najmniej "pamięciożerną" zmienną! Natomiast komentarza wymagają zmienne typu SINGLE i STRING.

Zmienna typu SINGLE została dodana do języka MCS BASIC stosunkowo niedawno i nie występowała w ogóle w zapomnianym już BASCOM-ie LT. Umożliwia ona dokonywanie operacji na liczbach zmiennoprzecinkowych, czyli mówiąc w uproszczeniu obliczanie liczb niecałkowitych. Za pomocą zmiennych typu SINGLE możemy dokonywać nawet bardzo skomplikowanych obliczeń, włącznie z funkcjami trygonometrycznymi. Ale, ... zawsze jest jakieś ale, musimy bardzo uważać na takie obliczenia, jeżeli dysponujemy procesorem z niezbyt wielką pamięcią programu! Wprawdzie zmienne typu SINGLE zajmują tylko, lub aż 4 bajty RAM, ale do wykonywania obliczeń potrzebna jest także spora ilość pamięci programu. Podam Wam prosty przykład podprogramu, za pomocą którego możemy obliczyć funkcję sinus podanej wartości (w radianach) kąta:

```
Dim Angle As Single , S1 As Single , S2 As Single , S3 As Single
 Angle = .1                    'wartość podana w radianach

 Gosub Sin                     'oblicz wartość sinusa od Angle
Print Angle
End

Sin:
S1 = Angle
            Angle = Angle * Angle
            S2 = Angle
            Angle = Angle * 0.01388888899236917
            Angle = Angle - 1
            Angle = Angle * S2
            Angle = Angle * 0.02380952425301075
            Angle = Angle + 1
            Angle = Angle * S2
            Angle = Angle * .05
            Angle = Angle - 1
            Angle = Angle * S2
            Angle = Angle * 0.1666666716337204
            Angle = Angle + 1
            Angle = Angle * S1
Return
```

Pozornie wszystko w porządku, program działa poprawnie, ale po jego skompilowaniu otwórzcie sobie okienko PROGRAM\SHOW RESULT:
Used ROM : &H53E 1342 (dec) > Ok

Horrendum, ten prosty programik zajął w pamięci 1342 bajty, czyli znacznie więcej niż połowę pamięci, jaką ogółem mamy do dyspozycji! Płynie stąd smutny wniosek, że jeżeli mamy zamiar dokonywać skomplikowanych obliczeń matematycznych, to powinniśmy się zaopatrzyć co najmniej w procesor '4051.

Podobnie ma się sprawa ze zmiennymi typu STRING. Musimy używać ich z największą rozwagą, pamiętając, że w definiowanym tekście każdy znak zajmuje dokładnie 1 bajt i że do każdej zmiennej tekstowej dodawany jest jeszcze jeden bajt kończący ciąg znaków. A zatem jeżeli zadeklarujemy zmienną:

DIM X AS STRING * 2

to zajmie ona w pamięci RAM trzy bajty.

Na zakończenie podam Wam jeszcze koło ratunkowe, bo tak można nazwać polecenie **ERASE**, usuwające z pamięci wykorzystane i już niepotrzebne zmienne, zwalniając w ten sposób cenny obszar pamięci RAM. Wydanie polecenia:

Erase [zmienna]

Usunie z pamięci wskazaną zmienną i zwolni zajmowane przez nią miejsce. Musimy jednak pamiętać, że usunąć możemy tylko ostatnio zadeklarowane zmienne, w kolejności "od tyłu". Najlepiej posłużyć się przykładem:

```
DIM A as byte, C as byte, D as byte, E as byte
```

Teraz możemy usuwać zmienne w podanej kolejności:

```
Erase E
Erase D
Erase C
```

Natomiast np. wydanie polecenia "Erase C" bezpośrednio po zadeklarowaniu zmiennych nie da żadnego rezultatu. Napiszmy sobie krótki programik, w którym tymczasowo zawiesimy wykonanie polecenia ERASE.

Po skompilowaniu program zadziałał w symulacji programowej prawidłowo, dwukrotnie wysyłając na ekran terminala zadeklarowane wartości. Jednak po dodaniu polecenia ERASE, już w czasie kompilacji wystąpił komunikat o błędzie, ponieważ powtórnie chcieliśmy użyć zmiennej już usuniętej z pamięci.

```
Dim A As Byte : A = 23
Dim C As Byte : C = 47
Dim D As Byte : D = 68
Dim E As Byte : E = 98
Print A
Print C
Print D
Print E
            'Erase E
Print A
Print C
Print D
Print E
```

A zatem pamiętajcie: nie tylko rodzaj deklarowanych zmiennych należy dokładnie przemyśleć. Istotna jest także kolejność ich deklarowania i zmienne, które ewentualnie będziemy mogli po wykorzystaniu usunąć z pamięci, deklarujemy jako ostatnie.

To wszystko na dzisiaj, pomimo że nie omówiliśmy nawet drobnej części poruszanych w Waszych listach tematów. Na następnym wykładzie zajmiemy się wprawdzie obsługą magistrali 1WIRE, ale jeżeli starczy czasu, to wrócimy jeszcze do trapiących Was problemów.

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