✓ INDEX:

- **WSTĘP**

We wstępie nie będę się rozpisywał dużo. Wynika to z tego, iż struktura programu w Bascomie jest bardzo prosta. Na początku umieszczamy wszelkie deklaracje dotyczące np. ustawień wyświetlacza, linii I²C lub częstotliwości rezonatora (jeśli w programie było to wcześniej prawidłowo ustawione, nie trzeba dodatkowo pisać w programie) oraz deklaracje zmiennych, stałych lub konfiguracji timerów. Następnie znajduję się treść programu głównego oraz podprogramy.

!!! Uwaga !!!
Jeśli przeprowadzamy symulację programu na początku należy to zasygnalizować umieszczając **$SIM**.
Natomiast jeśli już kompilujemy program z przeznaczeniem dla procesora ten znacznik musi być całkowicie usunięty lub odznaczony (**'$SIM**).

- **KONFIGURACJA PROGRAMU BASCOM8051**

Konfiguracja programu Bascom8051 jest bardzo prosta. Po zainstalowaniu i uruchomieniu aplikacji otwieramy okno OPTIONS > COMPILER > OUTPUT i zaznaczamy pola plików jakie ma utworzyć program po skompilowaniu. Do poprawnej pracy programatora należy zaznaczyć wszystkie pola z wyjątkiem OLD INTEL HEX FILE.
Następnie zakładka COMMUNICATION, w której ustawiamy prędkość komunikacji przy transmisji szeregowej (domyślnie na 2400) oraz częstotliwość stosowanego rezonatora kwarcowego (standardowo 11,059MHz lub 12MHz).
W okienkach I²C oraz LCD ustawiamy linie portów wykorzystywane przez magistralę oraz wyświetlacz alfanumeryczny.
W zakładce MISC wybieramy konfigurację rejestru dla odpowiedniego procesora, BYTE END ustawiamy adres ostatniej komórki pamięci RAM jaka może być wykorzystana przez program (np. 7F – bo w HEX) oraz opcję SIZE WARNING informującą czy kompilator ma ostrzegać jeśli zostanie przekroczona określona pojemność programu (pamięci Flash).
Zakładka COMMUNICATION określa paramentry komunikacji komputera z programatorem przez RS. Tutaj nic nie zmieniamy, ponieważ będziemy stosować komunikację przez port równoległy LPT (dla programatora AVT – 2502).
W zakładce ENVIORONMENT - ustawienia samego programu, wg. naszej wygody.
W HARDWARE SIMULATOR ustawiamy adres portu przez który będzie odbywać się komunikacja podczas symulacji oraz typ symulatora. Ponieważ na razie nie będziemy przeprowadzać symulacji, więc nie zmieniamy domyślnych parametrów.
Okienko PROGRAMMER jest odpowiedzialne, jak sama nazwa wskazuje, za programowanie procesora (jeśli programujemy bezpośrednio z Bascoma przy użyciu AVT-2502). W niej w PARALLEL(port równoległy) ustawiamy adres portu (najczęściej 378), PORT DELAY (opóźnienie portu) dla komputera z procesorem 300MHz wynosi ok. 50
i maleje wraz z szybszym komputerem. Wybieramy jeszcze programator MCS FLASH PROGRAMMER i zaznaczamy pola AUTO FLASH oraz AUTO VERIFY.
No nareszcie można powiedzieć....konfiguracja zakończona.

- **ZMIENNE**

DIM [nazwa_zmiennej_1] AS BIT - zakres zmiennej 0 lub 1
DIM [nazwa_zmiennej_2] AS BYTE - zakres zmiennej 0...255
DIM [nazwa_zmiennej_3] AS WORD - zakres zmiennej 0...65535
DIM [nazwa_zmiennej_4] AS INTEGER - zakres zmiennej -32767..32768
DIM [nazwa_zmiennej_5]*10 AS STRING - zmienna tekstowa o długości np. 10

- **STAŁE**

CONST [nazwa_stałej_1] = 15 - cyfra 15 zapisana dziesiętnie
CONST [nazwa_stałej_1] = &B1111 - cyfra 15 zapisana binarnie
CONST [nazwa_stałej_1] = &HF - cyfra 15 zapisana szesnastkowo
CONST [nazwa_stałej_2] = -1000
CONST [nazwa_stałej_3] = 1.1
CONST [nazwa_stałej_4] = "tekst"

Przy przypisywaniu zmiennym wartości należy pamiętać o zakresach zadeklarowanych im typów.

Np. nie możemy dla zmiennej zadeklarowanej:
DIM ZMIENNA_1 AS BYTE
Przypisać:
ZMIENNA = 300

- **TABLICE**

DIM [nazwa_tablicy](ilość) AS [rodzaj_zmiennej]
[nazwa_tablicy](5) = 146 - zapianie do 5 elementu tablicy liczby 146

Przykład:

```
Dim Tablica(10) As Byte
Tablica(5) = 146
```

- **PRZYPISANIE NAZWY DO LINII LUB PORTU**

[nowa_nazwa] ALIAS [linia/port]
Jest to funkcja bardzo praktyczna i ułatwiająca pisanie programu, gdyż zamiast używać np. P1.7 w programie, tworzymy alias i używamy tej nazwy.

Przykład:

```
LED ALIAS P1.7
DO
  RESET LED        - wyzerowanie LED, a właściwie linii P1.7
  WAIT 1           - poczekaj 1 sekundę
  SET LED          - ustawienie linii 7 z portu 1
  WAIT 1
LOOP
END
```

Przedstawiony program zapala i gasi diodę LED podpiętą do linii 7 w porcie 1 z częstotliwością 0,5Hz i współczynnikiem wypełnienia 50% .

Przykład 2:

```
PORT1 ALIAS P1
DO
  PORT1 = 255      - ustawienie całego portu 1
  WAIT 1           - poczekaj 1 sekundę
  PORT1 = 0        - wyzerowanie całego portu 1
  WAIT 1
LOOP
END
```

- **PĘTLE**

Pętla DO...LOOP:

```
DO          - początek pętli
...         - operacje i polecenia wykonywane w pętli
LOOP        - koniec pętli i powrót do jej początku
END
```

Pętla WHILE...WEND:

```
WHILE [warunek]  - początek pętli
 ...             - ciąg_instrukcji
WEND             - koniec pętli i wyjście jeśli warunek spełniony, powrót do jej początku
                   jeśli nie
```

Przykład:

```
WHILE ZMIENNA < 10
  INCR ZMIENNA
WEND
```

- **PODPROGRAMY**

Na początku wpisujemy deklaracje podprogramu:
DECLARE SUB [nazwa_podprogramu]

Następnie w programie piszemy, co konkretnie podprogram ma wykonywać.
Robimy to w następujący sposób:

```
SUB [nazwa_podprogramu]      - początek podprogramu
...                          - operacje i polecenia wykonywane w podprogramie
END SUB                      - koniec podprogramu
```

Aby wywołać podprogram piszemy:
CALL [nazwa_podprogramu]

Natomiast, jeśli w podprogramie będziemy wykonywać operacje na zmiennej globalnej, deklarujemy ją na początku oraz przy wywoływaniu podprogramu.
Na początku wpisujemy deklaracje zmiennej i podprogramu:
DIM [zmienna] AS [typ_zmiennej]
DECLARE SUB [nazwa_podprogramu]([zmienna] AS [typ_zmiennej])

A w programie piszemy, co konkretnie podprogram ma wykonywać. Robimy to w następujący sposób:

```
SUB [nazwa_podprogramu]( [zmienna] AS [typ_zmiennej] )   - początek podprogramu
...                          - operacje i polecenia wykonywane w podprogramie
END SUB                      - koniec podprogramu
```

Aby wywołać podprogram piszemy:
CALL [nazwa_podprogramu]([zmienna])

Przykład:

```
DIM ZMIENNA AS BYTE
DECLARE SUB PODPROGRAM(ZMIENNA)

DO
  ZMIENNA = 10                  - przypisanie wartości 10
  CALL PODPROGRAM(ZMIENNA)      - wywołanie naszego podprogramu
LOOP                            - koniec pętli
END                             - koniec programu głównego

SUB PODPROGRAM(ZMIENNA AS BYTE)
  HOME                          - ustawienie kursora w 1 linii 1 znak
  LCD ZMIENNA                   - wyświetlenie zmiennej czyli 10
END SUB                         - koniec podprogramu
```

- **WYŚWIETLACZ ALFANUMERYCZNY LCD**

W Bascomie sterowanie wyświetlaczy alfanumerycznych LCD jest banalnie proste. Wystarczy tylko poinformować kompilator, do których linii portu będzie on podłączony (OPTIONS/COMPILER/LCD) i zadeklarować na początku programu rodzaj podłączanego wyświetlacza:

CONFIG LCD=[ilość_znaków]*[ilość_linii]
gdzie ilość znaków i ilość linii może być następująca:
40*4, 40*2, 16*1, 16*1a, 16*2, 16*4, 16*4, 20*2, 20*4,40*4a

W przypadku problemów ze sterowaniem 16*1 lub 40*4 należy zmienić je na 16*1a lub 40*4a

Bascom pozwala na wykonywanie następujących operacji na wyświetlaczu:

LCD "TEKST" - wyświetlenie napisu TEKST
LCD [zmienna] - wyświetlenie zmiennej
LCD "tekst"; [zmienna] - wyświetlenie w jednej linii napisu TEKST i wartości zmiennej
CLS - czyszczenie wyświetlacz
CURSOR ON/OFF, BLINK/NOBLINK - operacje na kursorze
gdzie ON/OFF - włącz lub wyłącz kursor, BLINK/NOBLINK - włącz lub wyłącz miganie kursora, polecenia ta mogą działać oddzielnie, czyli możemy np. napisać:
CURSOR BLINK

LOCATE [linia],[pozycja_kursora] - ustawienie kursora na linii i pozycji
SHIFTCURSOR LEFT/RIGHT - przesuwanie kursora w lewo lub wprawo
SHIFTLCD LEFT/RIGHT - przesuwanie tekstu w lewo lub w prawo
DISPLAY ON/OFF - włączenie lub wyłączenie wyświetlacza
UPPERLINE - przejście kursora do wyższej linii
LOWERLINE - przejście kursora do niższej linii
THIRDLINE - przejście kursora do trzeciej linii
FOURTHLINE - przejście kursora do czwartej linii
DEFLCDCHAR [numer_banku] , 31 , 17 , 17 , 17 , 17 , 17 , 31 , 0 - deklarowanie własnego znaku np. polskich liter, gdzie numer_banku: 0..7. Aby stworzyć dowolny znak należy wybrać z menu TOOLS/LCD DESIGNER, a następnie narysować żądany znak. Bascom automatycznie wstawi polecenie DEFLCDCHAR do programu. My tylko musimy podać numer banku wywołując poleceniem CHR([nr_banku]).
Np.
LCD CHR(0)

- **WYŚWIETLACZ ALFANUMERYCZNY LCD**

Porady:

Jeśli chcesz wyświetlić tekst np. na środku wyświetlacza 1x16, zamiast pisać:
```
Home
Lcd "      wyraz       "
```
użyj poleceń:
```
Locate 1 , 7
Lcd "wyraz"
```

Drugi przykład jest bardziej optymalny - wykonanie jego zajmuje mniej czasu.

Jeśli potrzebujesz wyczyścić tylko jedną linię wyświetlacza 2x16 użyj konstrukcji:
```
Lowerline
Lcd "                "
```

- **WARUNEK IF...THEN**

```
IF [zdarzenie] THEN
...             - to co ma się zdarzyć jeśli warunek będzie spełniony
ELSE
...             - to co ma się zdarzyć jeśli warunek nie będzie spełniony
END IF          - koniec warunku
```

Zdarzenie może mieć następującą postać:
[zmienna] = 0 - jeśli zmienna będzie równa zero
P1.0 = 1 - jeśli bit 0 portu P1 będzie równy jeden

Przykład:

```
DIM Licznik AS BYTE      - deklaracja zmiennej LICZNIK
DO                       - początek pętli
  INCR Licznik           - zwiększenie o 1 zmiennej LICZNIK
  IF licznik = 15 THEN   - jeśli LICZNIK = 15 to
    Licznik = 0          - wyzeruj zmienną licznik
  END IF                 - koniec warunku przepełnienia
LOOP                     - koniec pętli i powrót do jej początku
END
```

Oczywiście można realizować warunek pod spełnieniem innego warunku.

Przykład 2:

```
DIM licznik_1 AS BYTE    - deklaracja zmiennej LICZNIK_1
DIM licznik_2 AS BYTE    - deklaracja zmiennej LICZNIK_2
DO                       - początek pętli
  IF licznik_1 = 15 THEN - jeśli LICZNIK_1 = 15 to
    licznik_1 = 0        - wyzeruj licznik_1
    INCR licznik_2       - zwiększenie o 1 zmiennej LICZNIK_2
    IF licznik_2 = 20 THEN - jeśli LICZNIK_2 = 20 to
      licznik_2 = 0      - wyzeruj licznik_2
    END IF               - koniec warunku 2
  END IF                 - koniec warunku 1
  INCR licznik_1         - zwiększenie o 1 zmiennej LICZNIK_1
LOOP                     - koniec pętli i powrót do jej początku
END
```

Program działa następująco. Na początku żaden z warunków nie jest spełniony dopóty, dopóki ta zmienna nie osiągnie wartości 15. Po osiągnięciu tej wartości zmienna pierwsza jest zerowana, a druga (licznik_2) jest incrementowana (zwiększana o 1). Po spełnieniu warunku dla drugiej zmiennej, jest ona zerowana, a program działa tak, że LICZNIK_2 jest zerowany co 15 x 20 czyli co 300 pętli.

- **INSTRUKCJA WYBORU CASE**

Chcąc wstawić instrukcję wyboru CASE w programie piszemy:

```
SELECT CASE [zmienna]
CASE [warunek_1] :
...                 - operacje wykonywane przy spełnieniu warunku 1
CASE [warunek_2] :
...                 - operacje wykonywane przy spełnieniu warunku 2
CASE ELSE :
...                 - operacje wykonywane w przypadku, kiedy żaden z powyższych
                      warunków nie został spełniony
END SELECT          - koniec uwarunkowań
```

Przykład:

```
CONFIG LCD = 16 * 1 DIM LICZNIK AS BYTE - deklaracja zmiennej licznik
LICZNIK = 0                             - wyzerowanie zmiennej
DO                                      - początek pętli
 INCR LICZNIK                  - zwiększenie wartości licznik o 1
 SELECT CASE LICZNIK           - początek uwarunkowań
  CASE 100 :                   - jeśli zmienna ma wartość 100 to
   LCD "LICZNIK =" ; LICZNIK   - wyświetl na Lcd LICZNIK = [wartość licznika]
  CASE 200 :                   - analogicznie
   LCD "LICZNIK =" ; LICZNIK
  CASE IS < 10 :               - jeśli zmienna licznik ma wartość mniejszą od 10 to ...
   LCD "LICZNIK < 10"
  CASE 11 TO 20 :              - jeśli wartość jest z przedziału od 11 do 20 to
   LCD "11 < LICZNIK < 20"
  CASE ELSE :                  - wartość nie spełnia powyższych warunków
   LCD "LICZNIK MA INNĄ WARTOŚĆ"
 END SELECT
LOOP
```

- **OPÓŹNIENIA CZASOWE**

DELAY - opóźnienie ok. 100us dla kwarcu 12MHz
WAITMS [milisekundy] - opóźnienie w milisekundach
WAIT [sekundy] - opóźnienie w sekundach również w zakresie 8 bitów

Przykład:

```
LED ALIAS P1.7
DO
  RESET LED      - wyzerowanie LED, a właściwie linii P1.7
  WAIT 1         - poczekaj 1 sekundę
  SET LED        - ustawienie linii 7 z portu 1
  WAIT 1
LOOP
END
```

Przykład 2:

```
LED ALIAS P1.7
DO
  RESET LED      - wyzerowanie LED, a właściwie linii P1.7
  WAITMS 200     - poczekaj 200 milisekund
  SET LED        - ustawienie linii 7 z portu 1
  WAITMS 200     - poczekaj 200 milisekund
LOOP
END
```

- **PRZERWANIA**

Przerwania:
INT0 INT1, SERIAL, TIMER0, TIMER1,TIMER2, INT3, INT4, INT5, INT6, INT7, INT8
Zależy to od używanego mikroprocesora.

W przypadku procesora 89CX051 mamy do dyspozycji dwa przerwania oraz dwa timery.

ON [przerwanie] [podprogram]	- informacja dla kompilatora, który podprogram ma być wykonywany przy podanym przerwaniu
ENABLE INTERRUPTS	- odblokowanie układu przerwań
ENABLE [przerwanie]	- odblokowanie podanego przerwania
DIASBLE [przerwanie]	- wyłączenie podanego przerwania

Przykład:

```
ON INT0 PRZERWANIE
ENABLE INTERUPTS
ENABLE INT0

DO
...       - program główny
LOOP
END


PRZERWANIE:
...       - instrukcje przerwania
RETURN
```

- **MAGISTRALA I²C**

Na początku deklaracja linii danych i zegara.
Są również ustawienia tych bitów w kompilatorze, lecz deklaracja programowa ma priorytet.

```
CONFIG SDA = P3.5
CONFIG SCL = P3.7
```

Następnie w programie piszemy:

I2CSTART	- warunek startu szyny
I2CWBYTE [zmienna], ACK/NACK	- zapis zmiennej z potwierdzeniem (ACK) lub bez (NACK)
I2CRBYTE [zmienna], ACK/NACK	- odczyt zmiennej z potwierdzeniem (ACK) lub bez (NACK).
I2CSTOP	- warunek stopu szyny.

Przykład dla układu RTC – PCF8583:

```
I2CSTART              - start
I2CWBYTE 162          - zapis adresu układu
I2CWBYTE 2            - wybór rejestru
I2CSTART              - kolejny sygnał startu na magistrali
I2CWBYTE 163          - zapis adresu do odczytu
I2CRBYTE S , Ack      - odczyt danych
I2CRBYTE M , Ack
I2CRBYTE H , Ack
I2CRBYTE Day , Ack
I2CRBYTE Month , Nack
I2CSTOP
```

Przedstawiony wycinek programu przedstawia komunikację z układem PCF8583 spod adresu 162 i odczyt danych (czasu i daty).

Przykład dla układu PCF8574

```
I2CSEND 78 , DANA     - wysłanie wartości zmiennej DANA do układu
WAITMS 10             - zwłoka 10 milisekund
I2CRECEIVE 79 , DANA_R - odczyt wartości z układu do DANA_R
```

- **INFORMACJE**

Kurs Bascom8051 został napisany przez autorów serwisu **µMAX.**
Plik może być rozpowszechniany bez ograniczeń pod warunkiem zachowania jego zawartości oraz treści. Za kopiowanie kursu nie można pobierać jakichkolwiek opłat.
Zawartość treściowa kursu może ulegać zmianom i aktualizacji.

Adres: http://www.uMAX.prv.pl
Mail: scaut@umax.prv.pl