KURS
Bascom'a

Autor
Paweł Klaja

Korekta

v1.5

Spis treści

1 Wstęp

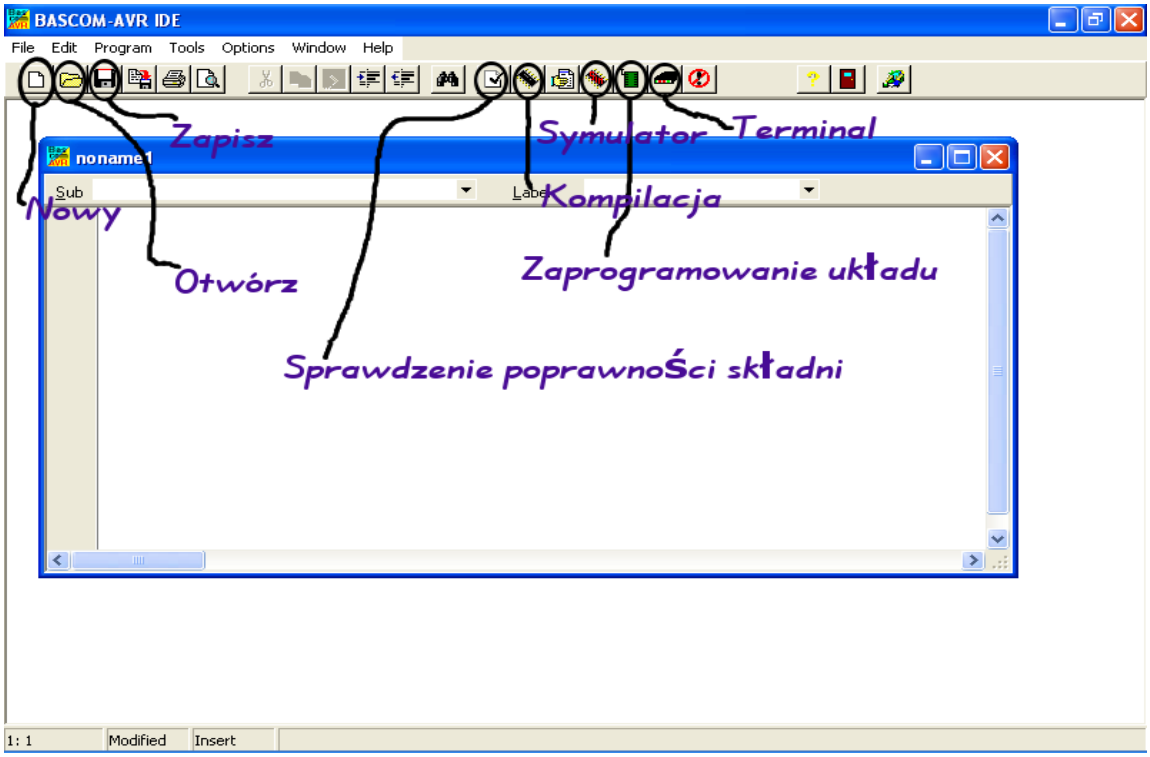
W tym kursie postaram się przedstawić podstawy języka i obsługi programu Bascom AVR. Jest to język programowania wysokiego rzędu. Co oznacza, że napisanie programu zajmuje dużo mniej niż w asemblerze, a przy tym jest mało skomplikowany. Kurs jest tak napisany, aby w jak najprostszy sposób wyjaśnić wszystkie zagadnienia. Najlepszą metodą nauki są przykłady, więc starałem się umieścić ich jak najwięcej. Jeśli coś jest niejasno napisane lub jakiś temat został w ogóle pominięty to napisz pklaja@o2.pl.

2 Instalacja i obsługa programu Bascom AVR

Bascom to nie tylko nazwa języka programowania, ale również nazwa programu, w którym tworzymy kod, a następnie go kompilujemy. Jest to podstawowa funkcja tego programu. Zacznijmy od zakupu programu lub ściągnięcia darmowej wersji i instalacji. Darmową wersję programu możemy ściągnąć ze strony producenta(http://www.mcselec.com/download_avr.htm). Trzeba ściągnąć wszystkie pliki instalacyjne (bcavrd_*.zip), a następnie rozpakować w jednym katalogu. Wersja demo w zasadzie niczym się nie różni od pełnej wersji. Poza tym, że kompiluje kod do 2kB, co na początku nie jest przeszkodą. Jeśli już posiadamy pliki instalacyjne to uruchamiamy plik setup.exe, który włącza instalację. Przebieg instalacji powinien wyglądać następująco:

- Ukazuje się nam okienko powitania w instalacji Bascom AVR, klikamy przycisk „Next”
- Ukazuje się nam okienko z plikiem readme, który powinniśmy przeczytać, klikamy przycisk „Next”
- Ukazuje się nam okienko z licencją, którą powinnyśmy przeczytać, i jeśli się z nią zgadzamy to przyciskamy przycisk „Yes”(w zasadzie musimy to zrobić, bo inaczej nie zainstalujemy programu)
- Ukazuje się nam okienko, w którym możemy zmienić domyślną ścieżkę instalacji programu, klikamy przycisk „Next”
- Ukazuje się nam okienko, w którym możemy zadecydować czy instalator ma zrobić kopię zapasową(yes-tak,no-nie), klikamy przycisk „Next”
- Klikamy przycisk „Next”
- Ukazuje się nam okienko, w którym pytają się nas czy jesteśmy pewni, że chcemy zainstalować ten program, klikamy przycisk „Next”
- Ukazuje się nam okienko, w którym widzimy jak się instaluje program
- Ukazuje się nam okienko zakończenia instalacji, klikamy przycisk „Finish”

Po zainstalowaniu programu uruchamiamy go. Powinno się nam ukazać okno takie jak na poniższym obrazku. Dodatkowo opisałem na nim znaczenie ikon.



Proponuje na początek zapoznać się z menu Bascoma. Jeśli to zrobimy to zobaczymy, że to nie tylko edytor tekstu i kompilator. Bascom posiada symulator (procesorów, wyświetlacza LCD i innych), różne przydatne narzędzia, a nawet obsługę programatora. Teraz zapoznajmy się z podstawami obsługi. Żeby zacząć pisać program musimy nacisnąć ikonkę *nowy* lub z menu *File->New* albo skrótem klawiszowym *ctrl+n*. Jak już się nam otworzy nowe okno to możemy zacząć pisać program. Zanim go skompilujemy powinniśmy dostosować ustawienia kompilatora do naszych potrzeb. Robimy to wybierając z menu *Options->Compiler->Chip*. Ukaże się nam nowe okno, w którym możemy wybrać procesor (chip), dla którego ma być skompilowany nasz program, a także czy będziemy używać zewnętrznej pamięci RAM(XRAM). Ta opcja nie jest dostępna dla wszystkich procesorów. W zakładce *Output* możemy zaznaczyć, bądź odznaczyć jakie pliki mają powstawać po kompilacji. Dalsze zakładki na razie pominiemy, gdyż opcje tam zawarte lepiej ustawiać poprzez deklarację na początku programu, co omówimy w dalszej części książki. Teraz możemy skompilować nasz program. Robimy to przez naciśnięcie klawisza *kompiluj* lub z menu *Program->Compile* albo skrótem klawiszowym F7. W razie gdybyśmy popełnili jakiś błąd to kompilator nie skompiluje nam programu i wyświetli listę błędów, które musimy poprawić, aby poprawnie skompilować program.

3 Stałe i zmienne

Jak wiemy podstawową sprawą w każdym języku są zmienne. W tym rozdziale dowiesz się jakie mamy dostępne typy zmiennych oraz jak ich się używa.

3.1 Typy zmiennych

Język Bascom oferuje nam sześć typów zmiennych, co wystarcza nawet przy skomplikowanych programach. W tabelce przedstawiono typy oraz ich zakres.

Typ	***Zakres***
Bit	0 lub 1
Byte	0...255
Word	0...65535
Integer	32767 … 32768
Long	-2147483648 ... 2147483647
Single	1.5×10^{-45} ... 3.4×10^{38}
STRING	zmienna tekstowa

Tabela 1

3.2 Deklaracja zmiennych

Przed użyciem zmiennej trzeba ją zadeklarować. Robi się to prawie na samym początku programu. W następujący sposób:
Dim nazwa_zmiennej *As* nazwa_tupu_zmiennej

Przykład 1

```
Dim licznik As byte
```

Przy deklaracji zmiennej typu string trzeba również podać jej długość. Więc deklaracja wygląda tak:
Dim nazwa_zmiennej*długość *As String*

Przykład 2

```
Dim kom*16 As String
```

3.3 Stałe

Czasami program wymaga użycia stałej. Polecam używanie stałej do przechowywania komunikatów, które będą wyświetlane lub wysyłane, gdyż wtedy łatwo poprawić błąd albo przetłumaczyć komunikat na inny język i nie musimy przeszukiwać programu tylko mamy

wszystko w jednym miejscu. Tak deklaruje się stałą i zarazem nadaje się jej wartość:
CONST nazwa_stałej = wartość. Żeby nasza stała przechowywała nam tekst to bierzemy go w cudzysłów.

Przykład 3

```
Const kom1="Naciśnij klawisz"
Const pi=3.14
```

3.4 Tablice

Oczywiście nim zaczniemy używać tablic, musimy je zadeklarować. Odbywa się to w podobny sposób, jak w przypadku zmiennych:
Dim nazwa_tablicy (długość tablicy) *As* nazwa_typu_zmiennej

Przykład 4

```
Dim tab (12) As byte
```

Wpisywanie wartości do tablicy odbywa się tak samo jak w przypadku zmiennej, z tą różnicą, że wskazujemy, do którego elementu tablicy wpisujemy wartość.

Przykład 5

```
Dim tab (12) As byte
tab(10)="a"
tab(3)=1
```

4 Działania

Jak już zadeklarujemy zmienną to chcemy wykonywać na nich jakieś operacje. Niektóre z nich to oczywiście działania arytmetyczne. W tabelce przedstawiłem jakie działania oferuje nam Bascom:

Znak	***Działanie***
+	Dodawanie
-	Odejmowanie
*	Mnożenie
/	Dzielenie

Tabela 2

Działania możemy wykonywać na zmiennych, stałych bądź podanych wprost liczbach.

Przykład 6

```
Dim A As Byte
Dim B As Byte
Dim wynik As Byte
wynik=A+B          'suma zmiennej A i B
wynik=A/5          'zmienną A dzielimy przez 5
wynik=wynik-1      'zmniejszamy zmienną wynik o jeden
```

W przykładzie pokazałem jak wykonywać działania arytmetyczne. Warto zwrócić uwagę na ostatnią operację, czyli dekrementację (zmniejszenie wartości o jeden) zmiennej *wynik*. Tą operacje powinniśmy zastąpić rozkazem *DECR*. To samo dotyczy się inkrementacji, czyli zwiększenia wartości zmiennej o jeden, zamiast działania używamy rozkazu *INCR*.

Przykład 7

```
Dim A AS Byte
Incr A       'stosujemy to zamiast A=A+1
Decr A       'stosujemy to zamiast A=A-1
```

5 Operatory

Teraz zapoznajmy się z operatorami. Będą one nam bardzo przydatne w pętlach oraz rozkazach warunkowych. W tabelce przedstawiono wszystkie dostępne operatory w języku Bascom. Oczywiście nie wszystkich będziemy używać. Na co dzień używa się tylko paru z nich, lecz bywają sytuacje, że przydają się także nietypowe, więc podaję wszystkie.

Operator	***Znaczenie***
=	równy
>	większy
<	mniejszy
<>	różny
<=	mniejsze lub równe
>=	większe lub równe

Tabela 3

6 Operacje na bitach

Nauczyliśmy się już parę rzeczy. Teraz czas na zapoznanie się z operacjami na bitach, a dokładnie na portach procesora. Oczywiście zaczynamy od konfiguracji.

6.1 Konfiguracja Portów

Jak wiemy w procesorach z rodziny AVR musimy się zdecydować czy dany pin/port będzie wejściem czy wyjściem. A w przypadku jak będzie wejściem czy ma być podciągnięty do Vcc. Do konfigurowania portów służy polecenie o następującej składni *Config Portx = output/input,* gdzie zamiast x wstawiamy literkę portu. Jeśli po znaku „=" napiszemy *output* to port będzie wyjściowy, a jeśli *input* to będzie wejściowy. Jeśli potrzebujemy skonfigurować tylko jeden pin to prostu robimy to następująco: Config Pinx.y = *output/input,* gdzie zamiast x wstawiamy literkę portu, a zamiast y numer pinu. W przypadku kiedy pin/port jest wyjściowy to możemy go podciągnąć do Vcc. Postępujemy tak w przypadku, kiedy czekamy na sygnał niski. Czyli na przykład: do Portu 1 jest podpięty przycisk, który w momencie naciśnięcia daje sygnał niski(Gnd) na ten pin. Jak to zrobić pokazuje poniższy przykład.

Przykład 8

```
Config Pind.1=Output          'Pind.1 jako wyjściowy
Set Portd.1                   'Podciągniecie Pind.1 do Vcc
```

Jeśli byśmy nie podciągnęli tego pinu to przy sprawdzaniu jego stanu okazałoby się, że jest zero, a przecież przycisk jest naciśnięty.

6.2 Operacje na bitach

Gdy chcemy wysłać coś na port, to używamy nazwy Potrx.y, a jeśli pobieramy coś z portu to używamy nazwy Pinx.y. Teraz poznajmy polecenia, dzięki którym będziemy sterować stanami na portach. Aby wysłać coś na port to po prostu piszemy jego nazwę, znak = i wartość lub zmienną(koniecznie 8-bitową). Natomiast jeśli chcemy sterować pojedynczym bitem to robimy to przy pomocy poleceń: *Set Portx.y* - aby ustawić stan wysoki, *Reset Portx.y* - aby ustawić stan niski. Jest jaszcze jedno bardzo pożyteczne polecenie służące do sterowania bitami. Mianowicie *toggle,* które zmienia stan bitu na przeciwny.

Przykład 9

```
Dim A As Byte
Config Portb=Output
Portb=A                       'zawartość zmiennej
                              'A będzie wystawiona
                                  'na Portb
Set Portb.1                   'ustawienie Portb.1
Reset Portb.7                 'wyzerowanie Portb.7
```

Bascom posiada jeszcze jedną operacje na bitach, czyli przypisywanie aliasów. Alias to jakby nazwa (etykieta), którą przypisujemy pinowi. Jest to bardzo pożyteczne, bo łatwiej zapamiętać nazwę niż numer portu. Poza tym, gdy piszemy długi i skomplikowany program, używanie aliasów umożliwia szybką zmianę nazwy pinu w przypadku pomyłki. Na przykładzie pokażę

jak się przypisuje Alias.

Przykład 10

```
led Alias Portb.1                'przypisania aliasu Portb.1
napis Alias Portd.5              'przypisania aliasu Portb.1
Set napis
Reset led
```

7 Pętle

Pętle to kolejna rzecz, bez której nie można napisać programu. Pomijając prosty program. Język Bascom oferuje nam trzy rodzaje pętli. W tym rozdziale zapoznamy się z nimi.

7.1 Pętla Do ...Loop

Jest to pętla bezwarunkowa, w której krąży program.Ta pętla nie sprawdza żadnego warunku, tylko stale krąży. Wyjść z niej można tylko poprzez wydanie komendy *Exit Do*. Teraz zapoznajmy się ze składnią tej pętli:

Do
.
. *'program*
.
Loop

W tej pętli umieszczamy nasz główny program. Poza tym taka pętla przydaje się w różnych sytuacjach, które pokażę w późniejszych przykładach, a teraz tylko taki przykład. Jeśli chcemy wyjść z tej pętli stosujemy rozkaz *Exit Do*.

Przykład 11

```
Config Pinb.1=Output
Do
Waitms 50                  'odczekaj 50ms
Toggle Portb.1             'zmień stan bitu na przeciwny
Loop
End
```

Efekt działania programu jest taki, że co 50ms zmienia się stan na pinie Portb.1. Możemy to sprawdzić poprzez podpięcie diody led do tego pinu.

7.2 Pętla While...Wend

Pętla while krąży dopóki jest spełniony warunek, który jest zawsze sprawdzany po każdym przejściu pętli. Oczywiście może się tak zdarzyć, że pętla nie przeleci ani razu, bo warunek nie będzie spełniony. Tak wygląda składnia pętli while:

while warunek
.
.
.
wend

Jeśli chcemy wyjść z tej pętli, mimo tego, że warunek jest nadal spełniony stosujemy rozkaz *Exit While*. Poniższy przykład ilustruje jak użyć tej pętli. Jeśli podepniemy diodę do pinu Portb.1 to zobaczymy jak 25 razy mrugnie.

Przykład 12

```
Dim A As Byte
Config Pinb.1 = Output
A=0
While A < 50
Waitms 50
Toggle Portb.1
Incr A
Wend
End
```

7.3 Pętla For

To jedna z ważniejszych pętli, a zarazem - moim zdaniem - najbardziej skomplikowana. Warunek tej pętli wygląda następująco: od podanej wartości jakieś zmienne „j" do podanej wartości tej samej zmiennej wykonuj pętle. A tak wygląda składnia pętli:
For A=wartość To watrość2
.
.
.
Next A
Tej pętli używamy na przykład przy odbieraniu lub wysyłaniu jakieś konkretnej ilości danych. Jeśli chcemy wyjść z tej pętli, mimo tego że warunek jest nadal spełniony stosujemy rozkaz *Exit For.*

Przykład 13

```
Dim a As Byte
For a = 1 To 10
Print a
Next a
```

Pętle *for* bardzo często wykorzystuje się przy pracy na tablicach. Dobrym przykładem będzie prosty program do kopiowania tablic. W poniższym przykładzie kopiujemy zawartość tablicy *tab_1* do *tab*.

Przykład 14

```
Dim tab (12) As byte
Dim tab_1 (12) As byte
Dim a As Byte
For a=0 To 11
tab(a)=tab_1(a)
Next a
```

8 Instrukcje warunkowe

To kolejna podstawa każdego programu. Dzięki instrukcjom warunkowym możemy sprawdzić m.in. stan wejścia w procesorze. Tymi instrukcjami sprawdzamy również równości zmiennych.

8.1 Instrukcja warunkowa IF

Jest podstawową instrukcją, dzięki której program decyduje za nas, oczywiście jeśli mu na to pozwolimy. Oto składnia tego polecenia:

If warunek Then

.

.

.

End if

Zasada jest prosta. Jeśli jest spełniony warunek to wykonuje instrukcje, które się znajdują między *then* a *end if*. W przeciwnym wypadku te instrukcje są omijane.

Przykład 15

```
Dim A as byte
A=0
Do
If A > 10 Then
A=0
end if
Loop
```

8.2 Instrukcja warunkowa IF ...ELSE

To kolejna często używana instrukcja. Różni się od poprzednich tym, że jeśli warunek jest niespełniony to wtenczas są wykonywane instrukcje znajdujące się po *else*. Składnia tej instrukcji wygląda następująco:

If warunek then

.

.

.

Else

.

.

.

End if

Czyli instrukcja wykonuje następujące czynności: sprawdza warunek, jeśli jest spełniony to wykonuje instrukcje znajdujące się po *then.* Natomiast, jeśli warunek nie jest spełniony to wykonuje rozkazy po *else*.

Przykład 16

```
Dim A As Byte
A=0
Do
If A>10 Then
A=0
Else
Incr A
Loop
```

8.3 Instrukcja warunkowa CASE

Tej instrukcji używa się bardzo rzadko, aczkolwiek warto ją znać. Oto składnia:

```
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
```

9 Opóźnienia czasowe

Czasem program musi się zatrzymać na określony czas. Do tego wykorzystujemy właśnie opóźnienia czasowe. Wyobraźmy sobie sytuację, w której chcemy, aby mrugnęła dioda podpięta do pierwszego pinu portu „b". Jeśli napisalibyśmy taki program to dioda tak szybko by mrugnęła, że byśmy tego nawet nie zauważyli.

```
Set portb.1        'zapalenie diody
Reset portb.1      'zgaszenie diody
```

W takiej sytuacji musimy zastosować opóźnienie czasowe. Możemy to zrobić na różne sposoby, ale ponieważ w tym przypadku nie potrzebujemyndokładnego czasu opóźnienia to proponuję wykorzystać opóźnienia, które oferuje nam Bascom. Jest to opóźnienie *Waitms,* które robi opóźnienie tak długie jak podamy w ms. Natomiast *Wait* robi opóźnienie tak długie jak podamy w s.

Przykład 17

```
Dim A As byte
Set portb.1          'zapalenie diody
Wait 2               'opóźnienie 2s
Reset portb.1        'zgaszenie diody
For A = 1 To 250
Set Portb.1           'zapalenie diody
Waitms A             'opóźnienie A ms
Reset portb.1         'zgaszenie diody
Next A
Watims 20            'opóźnienie 20 ms
Set Portb.1
End
```

10 Podprogramy

Przy dłuższych programach używanie podprogramów to konieczność dla przejrzystości kodu. Podprogramów używa się również do zmniejszenia rozmiaru kodu wynikowego. Najpierw podprogram musimy zadeklarować następującym poleceniem: *Declare Sub* nazwa_podprogramu. Później po programie głównym tworzymy nasz podprogram w następujący sposób

Sub Nazwa_podprogramu

.

.

.

End Sub

Po deklaracji i napisaniu podprogramu, możemy go wywołać. Język Bascom przedstawia nam dwa sposoby. W pierwszym wpisujemy nazwę podprogramu. A następną nazwę podprogramu poprzedzamy poleceniem *Call*.

Przykład 18

```
Dim A As Byte
Declare Sub inkrementuj
Do
If A > 10 Then
A = 0
Else
inkrementuj
End If
Loop
End
Sub inkrementuj
Incr A
End Sub
```

11 Przerwania

Im bardziej skomplikowany program tym częściej musimy korzystać z układu przerwań. Język Bascom bardzo ułatwia nam korzystanie z układu przerwań. Oto nazwy przerwań:

Nazwa przerwania w Bascomie	***Opis***
Int0,Int1,...	Kolejne przerwania zewnętrzne
Timer0,Timer1,Timer2	Przepełnienie licznika 0,1, lub 2
URxC	odebranie danej
UTxC	wysłanie danej
ADC	Przetwornik A/C

Tabela 4

Teraz sposób obsługi przerwań, który jest nadzwyczaj prosty. Przypisujemy, do której etykiety procesor ma przechodzić, gdy wystąpi przerwanie. Robimy to poleceniem *ON nazwa przerwania nazwa etykiety.* Oczywiście, aby układ przerwań działał, należy go odblokować, czyli *Enable Interrupts,* a następnie odblokować konkretne interesujące nas przerwanie *Enable nazwa przerwania.* Przykład najlepiej ilustruje zasadę działania. Program przedstawiony w przykładzie służy do odbioru danych z portu szeregowego przy użyciu przerwania.

Przykład 19

```
Baud = 4800
Enable Interrupts          'uaktywnienie przerwania odbiru danej z portu
                           szeregowego
Enable Urxc
Dim Tekst As String * 1
On Urxc Odbierz            'przypisanie obsługo przerwania
Do


Loop
Odbierz:
Tekst = Inkey()
 Return
 Return
```

12 Obsługa wyświetlacza LCD

Będzie tu mowa o wyświetlaczach ze sterownikiem HD44780 lub kompatybilnym.

12.1 Konfiguracja wyświetlacza lcd

Na początek musimy podać, do których pinów procesora jest podpięty wyświetlacz. Robimy to następująco:
Config Lcdpin = Pin , Db4 = Portb.4 , Db5 = Portb.5 , Db6 = Portb.6 , Db7 = Portb.7 , E = Portb.1 , Rs = Portb.0
Następnym krokiem jest podanie jakiej organizacji wyświetlacz posiadamy: Config Lcd = konfiguracja. Dostępne konfigurację: 1*8,2*8,1*16,2*16,1*20,2*20,4*20,2*40. Pierwsza liczba mówi ile mamy wierszy, a druga ile znaków w każdej linijce. Możemy skonfigurować również kursor tzn. czy ma być widoczny, a jak tak to czy ma mrugać. Robimy to poprzez rozkaz o następującej składni: *CURSOR ON / OFF BLINK / NOBLINK*. gdzie *ON / OFF* - włączenie(ON) lub wyłączenie(OFF) kursora, a *BLINK / NOBLINK* - czy kursor ma mrugać(BLINK) czy nie(NOBLINK).

Przykład 20

```
Config Lcdpin = Pin , Db4 = Portb.4 , Db5 = Portb.5 , Db6 = Portb.6 ,
Db7 = Portb.7 , E = Portb.1 , Rs = Portb.0
 Config Lcd =2*16
Cursor On Blink
```

W przykładzie powyżej skonfigurowałem wyświetlacz dwu - wierszowy, po szesnaście znaków w linijce. Dodatkowo skonfigurowałem tak kursor, aby był widoczny i aby mrugał.

12.2 Wyświetlanie znaków i zmiennych tekstowych

W momencie, gdy umiemy skonfigurować wyświetlacz to możemy zacząć coś na nim wyświetlać. Wyświetlenie znaku lub zmiennej odbywa się dzięki poleceniu *Lcd*. Sposób korzystania z polecenia *Lcd* najlepiej pokaże przykład.

Przykład 21

```
Config Lcdpin = Pin , Db4 = Portb.4 , Db5 = Portb.5 , Db6 = Portb.6
, Db7 = Portb.7 , E = Portb.1 , Rs = Portb.0
Config Lcd =2*16
Dim A As byte
A=10
Lcd "Napis"          'wyświetlenie tekstu na wyświetlaczu
Lcd A                'wyświetlenie wartości zmiennej A na wyświetlaczu
```

12.3 Wyświetlanie zmiennych liczbowych

Zmienne tekstowe wyświetla się w prosty sposób. Niestety ze zmiennymi typu integer, single i lond jest więcej kłopotów .Na szczęście istnieją polecenia które konwertują te zmienne na zmienne tekstowe, a wtedy już możemy je bez problemów wyświetlić. Do konwersji będą potrzebne polecenia: *str* - która zamienia zmienną liczbową na zmienną tekstową, *format* - która odpowiednio formatuje nam liczbę. Funkcje format opisze trochę dokładniej: pełna składnia tego

polecenia wygląda następująco *format(zmienna,"maska")*, dokładnego opisania wymaga maska, która mówi funkcji jak sformatować zmienną. Dozwolone maski na przykładzie liczby 987:

- spacje „ „ tyle ile spacji, tyle zostanie przesunięta liczba w prawo: efekt „ 987"
- znak „+" zostanie dodany znak + : efekt „+987"
- znak „0" zostaną dodane zera: efekt (maski „0000") „0987"
- znak „."znak kropki wstawia kropkę w odpowiednim miejscu: efekt (maski „00.00") „09.87"

A oto zastosowanie tych poleceń w przykładzie, który możemy wykorzystać do wyświetlania temperatury odczytanej z czujnika DS1820.

Przykład 22

```
Config Lcdpin = Pin , Db4 = Portb.4 , Db5 = Portb.5 , Db6 = Portb.6 , Db7 = Portb.7 ,
E = Portb.1 , Rs = Portb.0
Config Lcd = 16 * 2

Dim T As Integer

Cls
Lcd "Temp: "
Lcd Format(str(t) , "00.0") ; "C"
end
```

12.4 Inne polecenia

Oto polecenia, które przydadzą się przy korzystaniu z wyświetlacza Lcd:

Polecenie	Opis
cls	czyszczenie wyświetlacza
homeline	pierwsza linijka wyświetlacza
lowerline	druga linijka wyświetlacza
locate y,x	Umieszcza kursor w podanym miejscu .y-wiersz,x-znak
Shift LEFT/RIGHT	przesuwa napis na wyświetlaczu w prawo(RIGHT) lub w lewo(LEFT) o jedno miejsce

Tabela 5

Zastosowanie tych poleceń pokaże na przykładzie wyświetlania czasu i daty na wyświetlaczu. Czas wyświetlany będzie na środku pierwszej linijki, a data na początku drugiej linijki.

Przykład 23

```
Config Lcdpin = Pin , Db4 = Portb.4 , Db5 = Portb.5 , Db6 = Portb.6 , Db7 = Portb.7
, E = Portb.1 , Rs = Portb.0
Config Lcd = 16 * 2

Dim czas*8 As string, data*8 as string

Locate 1 , 5
lcd czas
lowerline
lcd data
```

13 Liczniki

Słowo licznik to próba przetłumaczenia słowa timer. Ja używam tego słowa jako odpowiednika, ale nie wszyscy się z tym zgadzają i twierdzą, że należy używać słowa timer, gdyż nie można dokładnie przetłumaczyć znaczenia. Licznik to układ liczący o rozdzielczości 8 lub 16 bitów (w przypadku procesorów z rodziny AVR). Większość procesorów z rdzeniem AVR posiada dwa liczniki, jeden 8 bitowy, a drugi 16 bitowy. W zasadzie również jest regułą, że liczniki 16 bitowe posiadają więcej funkcji. Liczniki najczęściej wykorzystuje się do odmierzania czasu (np. trwania impulsu) lub generowania impulsów.

13.1 Konfiguracja licznika

Moduły licznikowe w mikrokontrolerach AVR mogą pracować w dwóch podstawowych trybach: jako timer (czasomierz) lub jako counter (licznik). Jeśli chcemy, żeby licznik pracował jako counter to robimy to następująco: *CONFIG TIMERx = COUNTER, EDGE=RISING/FALLING*, gdzie zamiast *x* wstawiamy numer licznika, który konfigurujemy(np. *Config Timer0 = Counter*) i określamy parametr *EDGE* tzn czy licznik będzie zwiększany wraz z pojawieniem się zbocza opadającego(*FALLING*) na wejściu licznika czy rosnącego(*RISING*). Natomiast jeśli licznik ma pracować jako czasomierz to konfigurujemy go następująco: *CONFIG TIMERx = TIMER, PRESCALE=1/8/64/256/1024,* gdzie zamiast *x* wstawiamy numer licznika, który konfigurujemy i określamy parametr *PRESCALE,* który określa współczynnik podziału częstotliwości sygnału zegarowego (czyli z jaką częstotliwością ma być taktowany licznik). W przykładzie pokazano jak skonfigurować licznik 0 w trybie counter, a licznik1 w trybie czasomierza.

Przykład 24

```
Config Timer0=COUNTER, EDGE= RISING
Config Timer1=Timer, Prescale=64
```

13.2 Obsługa licznika

Liczniki w mikrokontrolerach AVR możemy startować i zatrzymywać w dowolnej chwili. W Bascomie robimy to za pomącą poleceń:

- Start licznika: *Start Timerx*
- Zatrzymanie licznika: *Stop Timerx*

Gdzie zamiast x wstawiamy numer licznika (np. *Start Timer0*)
Liczniki mogą powodować trzy zdarzenia:

- Przepełnienie licznika
- Spełnienie warunku porównania
- Przechwyt wartości

Możemy kontrolować stan licznika na dwa sposoby tzn. albo badać flagi, albo używać przerwań. Zdecydowanie polecam ten drugi mimo tego, że na początek może się wydać bardziej skomplikowany, ale posiada wiele zalet. Najważniejszą z nich to nie blokowanie pracy programu (co ma miejsce w przypadku sprawdzania flagi). Przerwania pozwalają nam poświęcić minimalną ilość czasu na obsługę danego zdarzenia.

13.2.1 Przepełnienie licznika

Zacznę od wyjaśnienia co oznacza przepełnienie licznika. Jest to zdarzenie wywoływane, gdy licznik osiągnie maksymalną wartość i w następnym cyklu zegarowym zostanie wyzerowany. Pamiętajmy, że liczniki w AVR liczą albo do 0xFF (255) w przypadku licznika 8-bitowego lub do 0xFFFF(65536) w przypadku licznika 16-bitowego. Ustawia się flaga *TOVx* (x-numer licznika) lub wywołuje przerwanie nazwane w Bascomie *Timerx* (x-numer licznika). Teraz tylko jeszcze powinniśmy nauczyć się jak obliczać czas, który ma odmierzać licznik. Tworzymy to następująco: dzielimy częstotliwość zegara taktującego przez wartość ustawionego preskalera, a następnie dzielimy przez maksymalną wartość jaka może być wpisana do licznika(8bit-256,16bit-65537).Przykład pokaże jak to zrobić w praktyce. Mamy następującą sytuacje: używamy licznika 8 bitowego, procesor jest taktowny kwarcem 8 Mhz, a preskaler jest ustawiony na 256.

Przykład 25

(8MHz/256)/256=122

W czasie jednej sekundy licznik przepełni się około 122 razy.

Myślę, że przykład rozwiał wszelkie wątpliwości co do obliczeń. Teraz przykład, który przedstawia program, który wykorzystuje licznik 0 i odmierza czas 1s, wykorzystując do tego przerwania.

Przykład 26

```
$crystal = 8000000
Config Timer0 = Timer , Prescale = 256
On Timer0 Przerwanie

Dim Licz As Byte

Enable Interrupts
Enable Timer0

Do
Loop
End

Przerwanie:
Incr Licz
If Licz > 122 Then
   Licz = 0                                   'znaczy że już odmierzył 1s
End If
Return
```

14 Port szeregowy

Obsługa portu szeregowego w asemblerze staje się czasem naprawdę uciążliwa. Na szczęście twórcy Bascoma zadbali oto, aby transmisja danych była przyjemnością. Na początek musimy ustalić prędkość taktowania transmisji. Aby to zrobić musimy: po pierwsze podać wartość kwarcu taktującego procesor, a następnie prędkość transmisji danych. Prędkość kwarcu podajemy w następujący sposób: *$crystal* wartość_ w _hercach, a prędkość transmisji: *Boud* prędkość. W poniższym przykładzie pokazałem przykładową konfigurację. Pamiętajmy, że przy danej częstotliwości taktowania nie zawsze można uzyskać pożądaną prędkość transmisji. Bascom sygnalizuje nam, jeśli istnieje możliwość uzyskania danej prędkości transmisji danych.

Przykład 27

```
$crystal = 10000000
Baud = 9600
```

Teraz trzeba się zapoznać z poleceniami nadawania i odbierania.

14.1 Nadawanie

Do nadawania służy tylko jedno polecenie, dzięki któremu możemy wysyłać znaki, zmienne, oraz tekst. Polecenie *Print,* - bo o nim mowa - jest bardzo uniwersalne. Oto przykład, który przedstawi parę jego zastosowań:

Przykład 28

```
$crystal = 10000000
Baud = 9600
Dim zmienna As Byte
Print zmienna                'wyśle zmienną
Print "To jest tekst"        'wyśle tekst który jest w
                             ' cudzysłowiu
Print Hex(zmienna)           'wyśle zmienne w postaci
heksadecymalne
```

14.2 Odbieranie

Do odbierania danych Bascom posiada kilka instrukcji, które przedstawiłem w tabelce. Szczegółowe omówienie znajduje się w następnych podpunktach.

Polecenie	***Opis***
INPUT "napis",zmienna	najpierw wysyła napis, następnie odbiera zmienną lub zmienne
INPUTBIN zmienna	odbiera tyle znaków ile jest w definicji zmiennej
INKEY()	Pobrania z bufora odbiorczego

14.2.1 INPUT

Składnia tego polecenia wygląda następująco: *Input "tekst", zmienna, zmienna2*. To polecenie jest jednym z najbardziej pożytecznych ponieważ najpierw wysyła tak zwaną

"zachętę", a później odbiera dane. Możemy go wykorzystać w sytuacjach, kiedy chcemy aby ktoś wpisał z konsoli na komputerze jakąś daną. Dokładne działanie przedstawię na przykładzie. Napiszmy program, który będzie odbierał daną (liczbę z zakresu 0..255), następnie sprawdzał czy jest większą, mniejszą lub równa 100.

Przykład 29

```
$crystal = 10000000
Baud = 9600
Dim zmienna As Byte
Input "Podaj liczbe z zakresu 0..255: " , Zmienna
If Zmienna = 100 Then
Print "Liczba " Zmienna ; "jest równa 100."
If Zmienna > 100 Then
Print "Liczba " ; Zmienna ; "jest wieksza od 100."
Else
Print "Liczba " ; Zmienna ; "jest mniejsza od 100."
End If
```

Sądzę, że funkcja INPUT jest zrozumiała. Ma ona jeszcze jedną zaletę Kończy odbieranie danych po odebraniu znaku CR (#13-ENTER). Czyli idealnie nadaje się do odbierania danych z komputera.

14.2.2 INPUTBIN

Zdarzają się takie sytuacje, gdy komunikujemy się z urządzeniem, które nie wysyła znaku CR - kiedy kończy transmisję. Z tego powodu w Bascomie mamy do dyspozycji polecenie INPUTBIN, które odbiera tyle znaków ile jest w definicji zmiennej, nie posiada również tekstu zachęty. Tak wygląda jego składnia: INPUTBIN zmienna1,zmienna2.... A poniżej przykład który powinien wszystko wyjaśnić.

Przykład 30

```
Dim tekst As String*16
INPUTBIN tekst        'polecenie będzie czekało na 16 znaków a
                      ' po ich odebraniu zakończy odbieranie
```

Tym poleceniem możemy także odbierać określoną liczbę elementów, które zostaną zapisane do tablicy, co możemy zaobserwować na poniższym przykładzie.

Przykład 31

```
Dim tab (12) As byte
INPUTBIN tab(6),5          'do tablicy od elementu 6 zostanie
                           'zapisanych 5 odebranych bajtów
```

14.2.3 INKEY

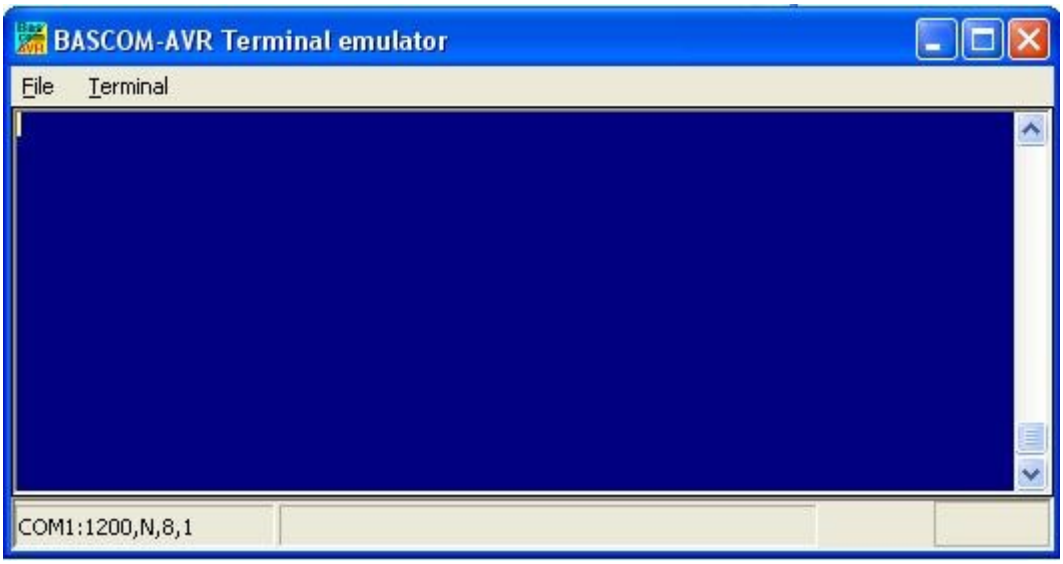
Ta instrukcja pobiera daną z bufora odbiorczego. Tego polecenia używamy wtedy, gdy używamy przerwania Urxc.

Przykład 32

```
$crystal = 10000000
Baud = 4800
Dim tekst As Byte
Enable Interrupts
Enable Urxc

On Urxc Tekstcior
Do

loop
end

Tekstcior:                    'Podprogram
Tekst = Inkey()
 Return
```

14.3 Obsługa terminalu

Terminal włączamy naciskając ikonkę *terminal* lub z menu *Tools-> Terminal Emulator.* Ukazuje się nam nowe okienko (jak na rys. poniżej), w którym będziemy wysyłać i otrzymywać dane przesyłane przez port szeregowy.



Nim zaczniemy to robić musimy oczywiście skonfigurować terminal. W lewym dolnym rogu możemy zobaczyć aktualną konfigurację, która jest napisana według następującej kolejności: *numer portu* : *prędkość transmisji*,*bity parzystości*,*ilość bitów danych*, *długość bitu stopu*. Na rys. Powyżej mamy następująco skonfigurowany terminal. Korzystamy z portu COM1 z prędkością 1200 baud, bez bitu parzystości z 8 bitową daną i jednym bitem stopu. Aby wprowadzić interesujące nas parametry transmisji wybieramy z menu *Terminal->settings w* zakładce *Communication,* ustawiamy numer portu, do którego jest podłączone urządzenie do komunikacji oraz z jaką prędkością ma się odbywać transmisja. Po dokonaniu odpowiednich ustawień, możemy zacząć pracę z terminalem. Jeśli urządzenie wyśle coś na port to wyświetli się to w niebieskim polu w postaci znaków ASIC. Jeśli chcemy wysłać coś przez port to po

prostu piszemy to i naciskamy ENTER. Jeśli terminal nie wyświetla nam znaków (a powinien) albo wyświetla „głupoty”, to sprawdź jeszcze raz prędkość transmisji. Bo to jest najczęstsza przyczyna. Mogą to być również zakłócenia.

15 Eeprom

Większość mikroprocesorów z rodziny AVR posiada wbudowaną pamięć Eeprom, która jest przydatna w wielu sytuacjach. Jak wiemy, pamięć Eeprom nie traci danych po wyłączeniu zasilania, a do tego jest programowalna i kasowana elektrycznie. Jeśli potrzebujemy przechowywać dane, których wartość nie zmienia się często, a chcemy, aby dana była przechowywana nawet po wyłączeniu zasilania to umieszczamy ją w pamięci Eeprom. Bascom posiada polecenia, które ułatwiają korzystanie z pamięci eeprom. Mianowicie dwa: jedno do zapisu, a drugie do odczytywania wartości komórki pamięci. Bascom umożliwia nam dodatkowo zadeklarowane zmiennej, której wartość będzie przechowywana w pamięci Eeprom.

15.1 Zapisywanie

Zapisywanie do wewnętrznej pamięci Eeprom odbywa się dzięki poleceniu: *WRITEEEPROM dana, adres.* Zamiast *dana* piszemy wartość lub nazwę zmiennej, a zamiast *adres* adres komórki pamięci, do której ma być zapisana dana.

Przykład 33

```
Dim zmienna As Byte
Writeeeprom Zmienna , 1              'zapisanie zmienna do pamięci
                                     ' eeprom pod adresem 1
End
```

Ten przykład pokazał jak zapisać daną do pamięci Eeprom. Do pamięci możemy zapisywać tablice, ale pamiętajmy, że Bascom sam inkrementuje numer elementu tablicy. Poniższy przykład pokazuje jak poprawnie zapisać tablice do pamięci.

Przykład 34

```
Dim Wpis(16) As Byte
Dim X As Byte
Dim Adres As Integer
For X = 1 To 16
  Writeeeprom Wpis(x) , Adres
  Incr Adres
Next
```

15.2 Odczytywanie

Do odczytu posiadamy jedno polecenie o następującej składni: *READEEPROM dana,adres.* Zamiast *dana* piszemy nazwę zmiennej, do której ma być zapisana wartość komórki, a zamiast *adres* adres komórki, z której ma być odczytana wartość.

Przykład 35

```
Dim zmienna As Byte
Readeeprom Zmienna , 1        'odczytanie komórki pamięci Eeprom z
                              'adresu 1 do zmienna
End
```

15.3 Zmienna w pamięci eeprom

Bascom umożliwia nam takie zadeklarowanie zmiennej, aby jej wartość nie była przechowywana w SRAM, tylko w Eeprom. Robimy to następująco: *Dim* nazwa_zmiennej *As ERAM* nazwa_tupu_zmiennej. Jak widzimy deklaracja takiej zmiennej różni się tylko słowem ERAM od deklaracji zwykłej zmiennej. Przykład 27 pokazuje jak dokładnie działa taka zmienna.

Przykład 36

```
Dim B As Byte , A As Byte , Wynik As Byte
Dim Wynike As Eram Byte
B= 1
A = 3
Wynik = A + B
Wynike = Wynik                'zapisanie wyniku do pamięci Eeprom
Wynik = Wynike                'odczytanie wyniku z pamięci Eeprom
Print Wynik
```

Jeśli używamy tak zadeklarowanej zmiennej to należy uważać, żeby jej wartość nie była za często zmieniana, bo pamięć Eeprom ma 100000 cykli zapisu/kasowania. Może się wydawać, że to dużo, bo tak jest jeśli używamy tej pamięci w sposób przemyślany i kontrolowany. Natomiast jeśli wartość takiej zmiennej będzie się zmieniać co przejście pętli to bardzo szybko się jej pozbędziemy.

16 Przetwornik A/C

W tym rozdziale nauczymy się jak korzystać z wbudowanego, w niektóre AVR 'ery przetwornika analogowo-cyfrowego. Proces przetwarzania odbywa się metodą kolejnych przybliżeń (successive approximation) co powoduje, że czas przetwarzania nie jest stały. Wynosi od 125 do 520 ms i zależy od przetwarzanej wielkości analogowej. Przetwornik powinien być taktowany zegarem o częstotliwości 50-200kHz, niezależnie od częstotliwości zastosowanego kwarcu. Dlatego przetwornik został wyposażony w preskaler z o następujących stopniach stopniami podziału: 2, 4, 8, 16, 32, 64 i 128. Przetwornik może pracować w dwóch trybach: SINGLE oraz FREE. W trybie SINGLE użytkownik inicjuje rozpoczęcie konwersji. Po zakończeniu konwersji przetwornik umieszcza jej rezultat w buforze i czeka na następny sygnał rozpoczęcia konwersji.
W trybie FREE konwersja inicjowana jest automatycznie, czyli po zakończeniu jednego procesu przetwarzania i przerwie trwającej 1,5 taktu zegara przetwornika, rozpoczyna się następny cykl. W buforze znajduje się zawsze rezultat ostatnio wykonanej konwersji.

16.1 Konfiguracja przetwornika

Konfiguracja przetwornika odbywa się przez wydania polecenia:*CONFIG ADC = SINGLE | FREE , PRESCALER = dzielnik | AUTO , [REFERENCE = OFF | AVCC | INTERNAL]*.
Teraz wyjaśnię poszczególne części polecenia. Zaczynamy od wyboru trybu pracy czyli *ADC = SINGLE | FREE,* tryby opisałem juz wcześniej. Następnie wybieramy dzielnik częstotliwości zegara taktującego: robimy to dzięki opcji *PRESCALER = dzielnik,* gdzie zamiast *dzielnik* wpisujemy: *2, 4, 8, 16, 32, 64 lub 128.* W bascom'ie możemy wybrać jeszcze jedną opcje tzn. „AUTO", która spowoduje, że podczas kompilacji Bascom sam określi właściwy stopień podziału. Jednak musimy ustawić dyrektywę **$CRYSTAL**, aby kompilator wiedział jaką częstotliwością jest taktowny procesor**.** W niektórych typach procesorów możemy także zadecydować co będzie napięciem odniesienia. Opcje podaję w tabelce.

Opcja	***Opis***	***Sposób podłączenia***
Off	Napięcie podaje się z zewnątrz	Napięcie podłączamy do końcówki AREF
AVCC	Napięciem odniesienia jest napięcie zasilające część analogową	między AREF a AGND należy dołączyć kondensator 10uF
Internal	Wewnętrzne napięcie odniesienia 2,56V	między AREF a AGND należy dołączyć kondensator 10uF

Tabela 6

Przykładowa konfiguracja przetwornika A/C:

Przykład 37

```
Config Adc = Single , Prescaler = Auto, Reference = Internal
```

16.2 Odczytywanie przetwarzania

Do odczytania używamy polecenia: *Getadc(nr.kanału),* gdzie nr.kanału to numer kanału, z którego ma być pobierany pomiar. Funkcja ta działa tylko wtenczas, gdy przetwornik pracuje w trybie single. Do włączenia przetwornika używamy polecenia: *start adc,*polecenie to powoduje włączenie zasilania modułu przetwornika. Do wyłączenia przetwornika służy polecenie: *stop* adc. Poniższy przykład powinien pokazać jak stosować te polecenia.

Przykład 38

```
Config Adc = Single , Prescaler = Auto
Start Adc
Dim W As Word , Kanal As Byte
Kanal = 0
'odczytujemy przetworzoną wartość z kolejnych kanałów
Do
  W = Getadc(Kanal)
  Print "Z kanału " ; Kanal ; "odczytałem wartość " ; W
  Incr Kanal
  If Kanal > 7 Then Kanal = 0
Loop
End
```

17 Emulacja magistrali I²C

To jest jedna z najlepszych rzeczy w języku Bascom tzn. gotowe emulacje najpopularniejszych magistrali. Teraz nie musisz się, że np. napisana przez ciebie emulacja zawiodła czy może wystąpił inny problem. Zaczniemy od magistrali I²C: najpierw trzeba wskazać, który Pin będzie taktujący Config Scl = Portx.y, a który danych Config Sda = Portx.y. Później możemy zaczynać używać magistrali. W tabelce poniżej przedstawiłem polecenia, które służą do sterowania magistralą.

Polecenie	***Opis***
I2cstart	Generuje sygnał startu magistrali
I2cwbyte	wysyłanie danej
I2crbyte	odbieranie danej
I2cstop	zatrzymanie magistrali

Tabela 7

Przykład 39 ilustruje korzystanie z magistrali I²C - może się wam przydać, gdy zechcecie korzystać z układu 2404, a nawet całej rodziny układów 24. Te układy to pamięci Eeprom, które są sterowane za pomocą magistrali I²C.Są bardzo wygodne w użyciu, a przy tym nie drogie i powszechnie dostępne.

Przykład 39

```
Dim A As Byte
Const Adresw = &hAE    'adres zapisywania 2404 koniecznie zapisany szesnastkowo
Const Adresr = &hAF    'adres czytania 2404 koniecznie zapisany szesnastkowo
I2cstart             'start magistrali
I2cwbyte Adresw       'wysłanie adresu
I2cwbyte 1           'wysłanie adresu komórki pamięci Eeprom do zapisania
I2cwbyte 3           'wysłanie wartości
I2cstop              'stop magistrali
Waitms 10            'czekanie przez 10ms, aż układ zapisze daną
I2cstart                    'start magistrali
I2cwbyte Adresw               'wysłanie adresu
I2cwbyte 1                  'wysłanie adresu komórki pamięci Eeprom do odczytania
I2cstart                    'start magistrali
I2cwbyte Adresr               'wysłanie adresu
I2crbyte A , Nack             'odebranie danej z magistrali
I2cstop                     'stop magistrali
End
```

18 Emulacja magistrali 1-Wire

W tym rozdziale zajmiemy się emulacją magistrali 1-Wire. Jest dość ciekawą magistralą, gdyż transmisja odbywa się dzięki jednemu przewodowi. Żeby użyć magistrali musimy podać, który pin będzie odpowiadał za wymianę danych *Config 1wire = Portx.y.* Teraz możemy już używać magistrali. W tabelce przedstawiłem wszystkie polecenia do obsługi magistrali.

Polecenie	***Opis***
1wreset	reset magistrali
1wwrite(ilość bitów)	wysłanie danej
1wread(ilość bitów)	odbiór danej

Tabela 8

Polecenia do odbioru/wysyłania danych zostały tak skonstruowane, aby ułatwić prace z tablicami tzn. zamiast pisać taki program:
For I = 1 To 8
Ar(i) = 1wread()
Next
możemy zastąpić go tylko jedną linijką: *Ar(1) = 1wread(8)* . Efekt działania będzie ten sam, czyli zostanie odebrane 8 bajtów i zapisane w tablicy *Ar.* W drugim przypadku tablica jest inkrementowana automatycznie. Przykład najlepiej zilustruje jak prosto, dzięki bascom, można skorzystać z magistrali 1 wire. Transmisje zaczynamy od zresetowania magistrali, następnie wysyłamy dane bądź rządania transmisji, później możemy odbierać dane. Poniższy przykład to fragment programu do odczytu temperatury z czujnika DS1620.

Przykład 40

```
Dim Bd(9) As Byte

1wreset                         'reset magistrali
1wwrite &HCC                        'wysyłanie przez magistrale
1wwrite &HBE                        ' rządanie
Waitms 250
Bd(1) = 1wread(9)                   ' sczytywanie 9 danych z magistrali
1wreset                         'reset magistrali, koniec transmisji
```

19 Ćwiczenia

19.1 Ćwiczenie 1 - Krążąca jedynka

Celem ćwiczenia jest napisanie programu, który będzie wysyłał na port takie sygnały, żeby podłączone do niego diody świeciły w następujący sposób: zapalamy tylko jedna diodę, następnie czekamy określony czas. Teraz gasimy diodę i zapalamy następną. Znowu czekamy określony czas i gasimy diodę i zapalamy następną i tak stale. To właśnie jest krążąca jedynka. Proponuje podpiąć diody do jednego portu, gdyż wtenczas będzie dużo prostszy program. Jeśli udało ci się już napisać taki program to dopisz jeszcze obsługę prędkości zapalania diod. Chodzi o to, aby po pojawieniu się stanu niskiego, na którymkolwiek pinie prędkość przełączania diod zwiększyła się dwukrotnie.

19.2 Ćwiczenie 2 -Krążąca jedynka + LCD

Wykorzystaj poprzedni program do zrobienia krążącej jedynki. Teraz dopisz do programu obsługę wyświetlacza LCD. Na wyświetlaczu ma się wyświetlać aktualna prędkość przełączania diod. Dodatkowo przycisk ma przełączać między następującymi prędkościami: 50,150,250.

19.3 Ćwiczenie 3 – Prosty interpreter poleceń

Napisz program, który będzie odbierał polecenia przez port szeregowy, a następnie je wykonywał. Jeśli odbierze nieznane polecenie to wysyła przez port komunikat o błędnym poleceniu. Załóżmy, że procesor ma interpretować następujące 6 poleceń:

Polecenie	***Znaczenie***
Set led1	Ustawić stan wysoki na którymś wyjściu
Reset led1	Ustawić stan niski na którymś wyjściu
Set led2	Ustawić stan wysoki na którymś wyjściu
Reset led2	Ustawić stan niski na którymś wyjściu
Set led3	Ustawić stan wysoki na którymś wyjściu
Reset led3	Ustawić stan niski na którymś wyjściu

Tabela 9

19.4 Rozwiązania

19.4.1 Ćwiczenie1

```
$crystal = 10000000                 'procesor taktowany 10MHz
Config Portb = Output               'portb wyjściowy
Config Portd = Input                'portd wejściowy
Portd = &HFF                        'portd podciągnięty przez wewnętrzny opornik
                                        do Vcc
Dim Nr As Byte                      'zmienna która przechowuje numer pinu portu
Dim Czas As Byte                    'zmienna przechowująca czas przełączania
diod
Declare Sub Przyciski               'podprogram - sprawdzanie przycisku
Czas = 150                          'początkowy czas przełączania diod
Set Portb.nr                        'ustawienie stanu wysokiego na odpowiednim pinie
```

```
Waitms Czas                    'odczekiwanie czasu
Do
  Reset Portb.nr               'ustawienie stanu niskiego na odpowiednim
                               pinie
 Incr Nr                       'następny pin
 If Nr > 7 Then                'jeśli numer pinu jest większy od 7 to wtenczas
                               wpisuje 0
  Nr = 0
 End If
  Set Portb.nr                 'ustawienie stanu wysokiego na odpowiednim
                                     pinie
Waitms Czas                    'odczekiwanie czasu
Call Przyciski
Loop
End                            'end program

Sub Przyciski
If Pind.2 = 0 Then             'jeśli na pinie 2 portu d pojawi się stan niski to
                               zwiększa szybkość przełączania
Czas = 75
End If
End Sub Przyciski
```

19.4.2 Ćwiczenie2

```
Config Lcdpin = Pin , Db4 = Portd.3 , Db5 = Portd.4 , Db6 = Portd.5 , Db7 = Portd.6 , E =
Portd.2 , Rs = Portd.1
Config Portb = Output
Config Pind.0 = Input
Dim Nr As Byte
Dim Czas As Byte

Declare Sub Przyciski
Czas = 150
Config Lcd = 16 * 2
Cursor Off
Cls
Lcd "Predkosc: "
Lcd Czas ; " ms"
Do
  Reset Portb.nr               'ustawienie stanu niskiego na odpowiednim
                                     pinie
  Incr Nr                      'następny pin
  If Nr > 7 Then               'jeśli numer pinu jest większy od 7 to
                                     wtenczas wpisuje 0
  Nr = 0
  End If
  Set Portb.nr                 'ustawienie stanu wysokiego na odpowiednim
                                     pinie
  Waitms Czas                  'odczekiwanie czasu
Call Przyciski
Loop
```

```
End

Sub Przyciski:
If Pind.0 = 0 Then
Waitms 250
  If Czas = 150 Then
  Czas = 250
  Else
    If Czas = 250 Then
    Czas = 50
    Else
    Czas = 150
    End If
  End If

Cls
Lcd "Predkosc: "
Lcd Czas ; " ms"
End If
End Sub Przyciski
```

19.4.3 Ćwiczenie3

```
$crystal = 10000000
$baud = 9600                          'prędkość transmisji danych
Config Portb = Output
Config Pind.6 = Output
Config Pind.5 = Output
Config Pind.4 = Output
Reset Portd.6
Dim Polecenie As String * 9
Dim Flaga As Byte                      'jeśli jest ustawiona to znaczy, że nie
                                          rozpoznano polecenia

Do
Flaga = 1
Input ">" , Polecenie                  'odbieranie polecenia

If Polecenie = "set led1"then
Set Portd.6
Flaga = 0
End If

If Polecenie = "clr led1"then
Reset Portd.6
Flaga = 0
End If

If Polecenie = "set led2"then
Set Portd.5
Flaga = 0
End If
```

```
If Polecenie = "clr led2"then
Reset Portd.5
Flaga = 0
End If

If Polecenie = "set led3"then
Set Portd.4
Flaga = 0
End If

If Polecenie = "clr led3"then
Reset Portd.4
Flaga = 0
End If

If Flaga = 1 Then
Print "zle polecenie"
Lcd "blad"
End If
Loop
```

20 Projekty

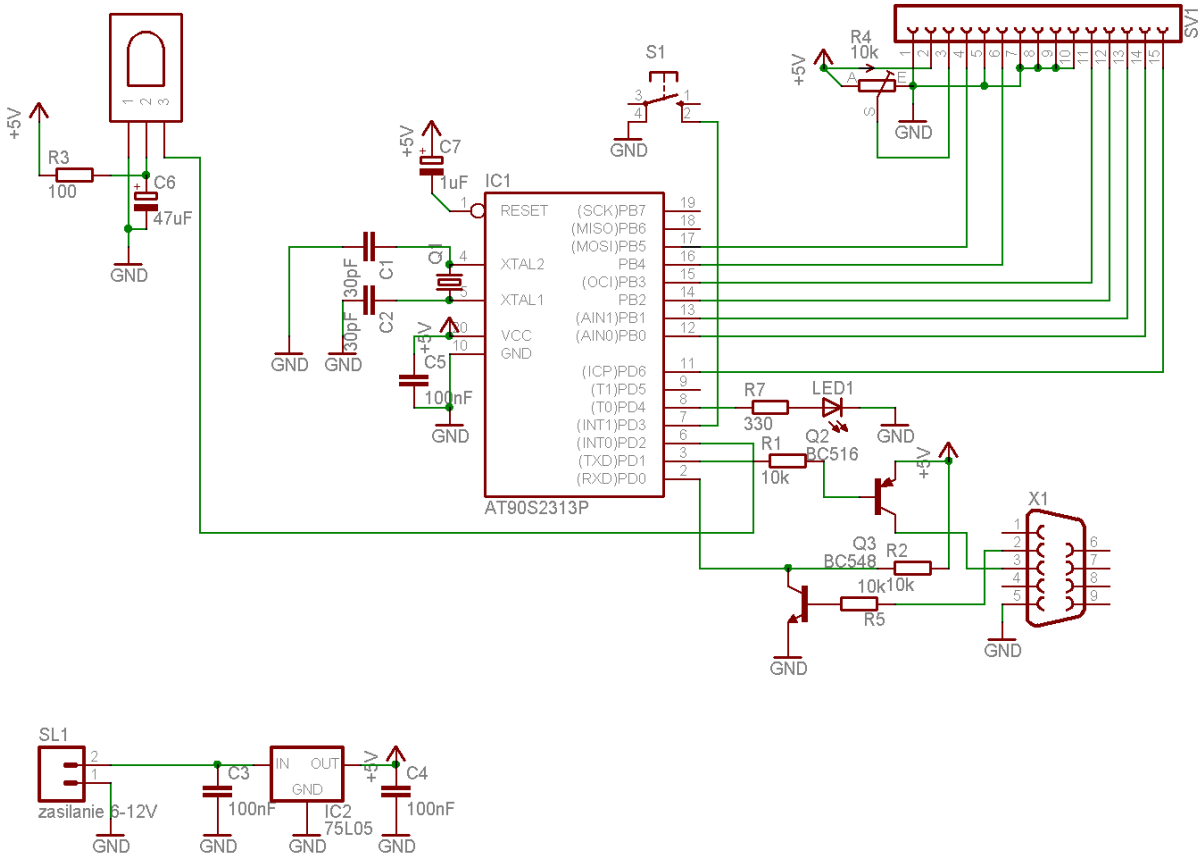
Nareszcie trochę praktyki. W końcu wykorzystamy naszą wiedzę do zaprogramowania procesora, który będzie robił pożyteczne rzeczy. W tym dziale znajdziesz gotowe projekty, które możesz sam wykonać. Projekty, czyli pełen schemat, zaprojektowana płytka(w programie eagle), przykładowe oprogramowanie, skompilowane oprogramowanie(w katalogu projekty). Życzę miłej zabawy. Wszystkie projekty są mojego autorstwa, i nie wolno ich udostępniać bez mojej zgody!!!

20.1 Pilot do winamp'a

Jest to projekt, który może wykonać nawet początkujący, jest on stosunkowo prosty do wykonania i nie powinno być problemów z dostaniem części do wykonania urządzenia. Nazwa projektu może być trochę myląca, ponieważ nie jest to pilot tylko odbiornik kodu Rc5, który steruje winamp'em (popularny odtwarzacz mp3). Odbiornik nie współpracuje z wszystkim pilotami, na pewno z uniwersalnymi. Teraz parę słów o samym układzie całym sterowaniem oraz dekodowaniem zajmuje się procesor AT90S2313. Do konwersji sygnału RS232 - ttl zostały użyte dwa tranzystory - zamiast max232 ze względu na obniżenie kosztów. Układ odczytuje kod wysłany przez pilot, następnie wysyła go przez port szeregowy. Procesor ma także za zadanie pobrać przez port szeregowy dane i wyświetlić je na wyświetlaczu. Układ współpracuje z wtyczką do winamp'a, którą trzeba odpowiednio skonfigurować. Sposób konfiguracji jest opisany w późniejszej części.

20.1.1 Schemat

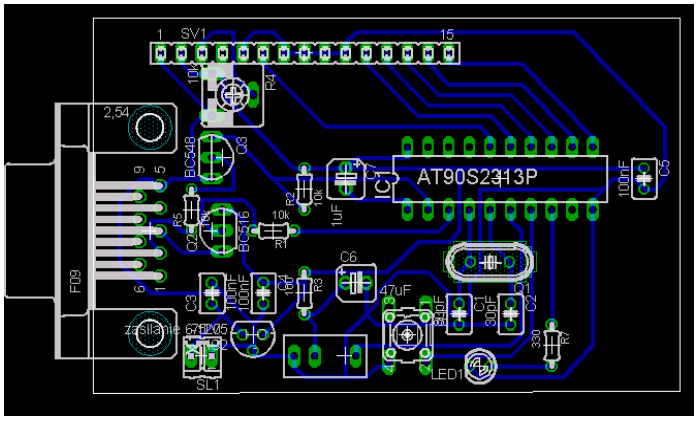
Na rysunku znajduje się schemat odbiornika. Został on zaprojektowany w programie Eagle.



Pawel Klaja	
TITLE: pilot do winampa01	
Document Number:	REV:
Date: 2005-04-08 14:33:06	Sheet: 1/1

20.1.2 Płytka

Na poniższym rysunku znajduje się projekt płytki. Jeśli chcecie wykonać płytkę samodzielnie to wraz z kursem otrzymałeś projekt płytki w postaci pliku brd.Plik ten zawiera projekt płytki, znajdujący się w katalogu *projekty* i obsługiwany przez program Eagle (www.cadsoft.de).



20.1.3 Lista części

Tu zamieszczam spis części oraz ich wartości:

C1 30pF

C2 30pF

C3 100nF

C4 100nF

C5 100nF

C6 47uF

C7 1uF

IC1 AT90S2313P

IC2 75L05

LED1 LED3MM

Q1 10MHz

Q2 BC516

Q3 BC548

R1 10k

R2 10k

R3 100

R4 10k

R5 10k

R7 330

S1 10-XX switch-omron

SL1 2 zaciskowe gniazdo

SV1 listwa 15 na goldpiny

U$1 TFMS5360

X1 gniazdo com DSUB-9

20.1.4 Przykładowe oprogramowanie

Poniżej znajduje się przykładowe oprogramowanie mikrokontrolera. Do odbioru kodu rc5 używamy polecenia *Getrc5(address , Command),* gdzie *address* to zmienna, w której będzie przechowywany adres pilota, który wysłał sygnał. *Command* to zmienna, w której będzie się znajdowała dana przesłana przez pilot. Nim użyjemy tego polecenia musimy poinformować, do którego pinu podpięty jest odbiornik. Robimy to następująco: *Config Rc5 = Pinx.y* gdzie *x* - port, *y* – numer pinu. Program starałem się pisać w sposób jak najbardziej przejrzyście i umieszczać komentarze. Teraz krótko o działaniu programu. Program posiada zabezpieczenie przed użyciem innego pilota. Zapisuje adres pilota, z którym ma współpracować w pamięci eeprom. Procesor steruje również wyświetlaczem lcd, na którym wyświetla tekst wysłany z portu szeregowego, gdy napis jest za długi (znaczy dłuższy niż 16 znaków) to zostaje on przesuwany, aby można było przeczytać cały tekst, a nie tylko pierwsze 16 liter. Ponadto procesor steruje podświetlaniem wyświetlacza. Zapala również diodę led podczas odbierania kodu rc5. Dodawanie/zmiana pilota polega na naciśnięciu przycisku. Zachęcam do przeanalizowania tego kodu i oczywiście do jego ulepszania, a najlepiej napisania własnego. Poniższy kod jest w katalogu projekty, jest również skompilowany.

```
'led czytanie kodu Portd.4
'Rc5 Portd.2
'podswietlanie Portd.6
'przycisk dodawania pilota Portd.3
$crystal = 10000000
Baud = 4800


Config Lcdpin = Pin , Db4 = Portb.3 , Db5 = Portb.2 , Db6 = Portb.1 , Db7 = Portb.0 , E = Portb.4 , Rs = Portb.5
Config Rc5 = Pind.2
Config Pind.4 = Output
Config Pind.6 = Output
Config Pind.3 = Input
Config Lcd = 16 * 2


Enable Interrupts                    'uaktywnienie przerwania odbioru danej z portu szeregowego
Enable Urxc
```

```
On Urxc Tekstcior                    'podprogram Tekstcior obsługujue przerwanie

Dim Address As Byte , Command As Byte , Ilosc As Integer
Dim Address_zarejestrowny As Byte          'adres pilota w eeprom
Dim Pruba As Byte
Dim Tekst As String * 1

Set Portd.3
Cls
Readeeprom Address_zarejestrowny , 1
Upperline
Lcd "Czekam na RC5"
Lowerline
Lcd "Pilot:" ; Address_zarejestrowny
Set Portd.4
Wait 2
Reset Portd.4
If Address_zarejestrowny = 255 Then

Cls
    If Address_zarejestrowny < 255 Then
      Lcd "Zmiana pilota"
    Else
      Lcd "Dodanie pilota"
    End If
    Lowerline
    Lcd "Nacisnij klawisz"
    Waitms 250
    Do
    Getrc5(address , Command)
      If Address < 255 Then              'zapisywanie kodu pilota w pamieci eeprom
      Writeeeprom Address , 1
      Address_zarejestrowny = Address
      Cls
```

```
      Lcd "Pilot zostal dodany"
      Wait 2
      Cls
      Exit Do
      End If
    Loop


End If
Do

 If Pind.3 = 0 Then                          'dodawanie pilota
 Cls
    If Address_zarejestrowny < 255 Then
      Lcd "Zmiana pilota"
    Else
      Lcd "Dodanie pilota"
    End If
    Lowerline
    Lcd "Nacisnij klawisz"
    Waitms 250
    Do
    Getrc5(address , Command)
      If Address < 255 Then                   'zapisywanie kodu pilota w pamięci eeprom
      Writeeeprom Address , 1
      Address_zarejestrowny = Address
      Cls
      Lcd "Pilot zostal dodany"
      Wait 2
      Cls
      Exit Do
      End If
    Loop
 End If
```

```
 If Ilosc > 16 Then                    'ile odebrał znaków z winampa jeśli więcej niż 16 to
                                       przesuwa napis
    Shiftlcd Left
    Waitms 80
 End If

 Getrc5(address , Command)             'odbieranie kodu
 If Address = Address_zarejestrowny Then
   Command = Command And &B01111111
   Print Address ; "  " ; Command
   Lowerline
   Lcd Address
   Lcd "  "
   Lcd Command
   Set Portd.4
Waitms 50
Reset Portd.4
 End If
Loop
End

Tekstcior:                             'Podprogram odbierania tekstu
Tekst = Inkey()
  Pruba = Asc(tekst)
  If Pruba > 0 Then
   If Pruba = 35 Then
   Cls
   Ilosc = 0
    Else
    Incr Ilosc
    Lcd Tekst
  End If
 End If
 Return
 Return
```

20.1.5 Konfiguracja komputera

Najpierw przegrywamy plik gen_serialcontrol.dll (znajduje sie w katalogu projekty) do katalogu, gdzie winamp trzyma wtyczki (domyślnie Program Files\Winamp\Plugins). Następnie włączamy winamp'a i w preferencjach konfigurujemy wtyczkę. Ustalamy, do którego portu com jest podłączony odbiornik. Dalsza konfiguracja jest opisana dla programu powyżej. W zakładce “COM port” prędkość transmisji ustalamy, 8 bitowe dane i jeden bit stopu oraz brak kontroli parzystości. Teraz otwieramy port szeregowy przyciskiem “Open now” oraz zaznaczamy opcje auto otwierania portu podczas startu winampa “Auto open”. Teraz możemy juz uczyć winamp'a poleceń. Przechodzimy do zakładki Winamp, zaznaczamy na liście funkcję, którą chcemy, aby wykonywał winamp po przyciśnięciu danego przycisku pilota, klikamy “Learn” i naciskamy przycisk pilota. W analogiczny sposób robimy z innymi funkcjami. Teraz przechodzimy do zakładki “Output”. Zaznaczamy “Output on”, do pola “Output format” wpisujemy znak #, a następnie według uznania zgodnie z legendą poniżej. Przykładowy format: #1% 5%. Po zakończeniu konfiguracji na wyświetlaczu lcd powinny się wyświetlać ustawione napisy.

20.1.6 Podsumowanie

Starałem się opisać projekt jak najjaśniej mam nadzieję, że mi się udało, ale ponieważ to jest pierwszy projekt jaki opisałem, więc zachęcam do przysyłania wszelkich sensownych uwag, co pomoże uniknąć w przyszłości popełniania tych samych błędów.

21 Zmiany

V1.5:
- dodanie rozdziału: Projekty
- rozszerzenie rozdziału:Emulacja magistrali 1-Wire, Przerwania
- poprawienie czytelności tekstu
- dodano nowe przykłady

V1.4:
- dodanie rozdziału: Przetwornik ADC
- rozszerzenie rozdziału: Obsługa wyświetlacza LCD
- dodano nowe przykłady

V1.3:
- dodanie spisu treści
- dodanie bookmark w pdf
- dodano rozdziały: **eeprom, liczniki**
- dodano nowe przykłady(np. Do magistrali I^2C)
- poprawienie wyglądu całego dokumentu
- poprawki błędów sygnalizowane przez użytkowników

22 Podsumowanie

Cóż to już koniec tego kursu mam nadzieję, że zaspokoił on twoją wiedzę. Jeśli nie to napisz a kolejna wersja może będzie rozszerzona o problem, który cię nurtuje. Oto mój adres e-mail pklaja@o2.pl, na który proszę kierować wszelkie uwagi oraz informacje o błędach. Pozdrawiam!