MATERIAŁY INFORMACYJNE
STEROWNIK KNEST

Opracowali na podstawie własnych przemyśleń
i materiałów z internetu:
Piotr Tomczuk
Maciej Rudziński

Warszawa 2007

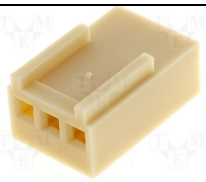
☺

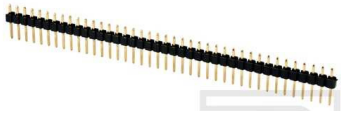
v.1.3

MATERIAŁY DO WYKONANIA STEROWNIKA KNEST

Lista elementów			
L.P.	**Nazwa**	**Ilość**	**Foto/orientacyjna cena/szt**
Procesor			
1	Procesor **ATMEGA8535-16PU** dil 40	1	 12 zł IC5
2	Podstawka dip 40 precyzyjna	1	 3 zł
Blok Zasilacza			
3	Stabilizator LM7805 1,5A	1	 0,6 zł IC2
4	Kondensator 470 uF / 25V	2	 0,8 zł C3, C4
5	Kondensator 100 nF / 50V	2	 0,15 zł C5, C6
6	Dioda świecąca zielona 5mm matowa	1	 0,3 zł LED1
7	Rezystor 560 om	1	 0,1 zł R1
Blok generatora			
8	Kwarc 11,059MHz	1	 2,2 zł Q2
9	Kondensator 27 pF (od 20p do 35pF)	2	 0,1 zł C1, C2

Blok Resetu			
10	Kondensator 10uF / 16V	1	 0,6 zł C7
11	Rezystor 10 kΩ (0,125W)	1	 0,1 zł R5
12	Dioda 1N4148	1	 0,1 zł D1
13	Przycisk Reset 6x6mm h = 5mm Normalnie otwarty	1	 0,5 zł S1
Złącze Programowarki na płytce			
14	Gniazdo NS25-W5K (lub WF-05S) kątowe do druku raster 2,54mm	1	 0,4 zł
Programowarka (kabel komputer – procesor)			
15	Rezystor 100 Ω 0,125W	5	 0,1 zł
16	Obudowa wtyku żeńskiego NS25 (lub HU-05) raster 2,54mm	1	 0,6 zł
17	Terminal gniazda NS 25 (lub WF)	8	 0,01 zł
18	Obudowa D-SUB plastikowa prosta 25pin szara	1	 0,9 zł
19	Wtyk męski D-SUB do wlutowania przewodów 25pin	1	 1 zł

20	Przewód linka komputerowy 0,5 mm^2 5 żył w oplocie lub 10 żyłowa taśma kolorowa 0,5 mm^2	1,5m	 3,5/mb
Elementy potrzebne do zmontowania elementów portu RS – 232 OPCJA			
21	Przewód 0,5 mm^2 3 żyły w oplocie do RS232	3 m	 3,5/mb
22	Max 232 Układ scalony Transceiver RC232 DIP16	1	 1,6 zł IC4
23	Podstawka dip 16 precyzyjna	1	 1,25 zł
24	Kondensator elektrolityczny 4,7 uF/25V	4	 0,35 zł C8,C9,C10,C11
25	Wtyk NS25 W3P (lub WF-3) Raster 2,54mm	1	
26	Wtyk złącza D-Sub 9 żeński lut na kabel	1	 1,2 zł
27	Obudowa D-SUB plastikowa prosta 9pin	2	 1,5 zł
28	Gniazdo NS25 W3P (lub WF-3) Raster 2,54mm	1	 1,5zł
Wyświetlacz LCD OPCJA			
29	Wyświetlacz LCD 2*16 1602 (biało-nieb) z podświetleniem (np.HY-1602 F4) lub WC 1602 (zielony) z podświetleniem (np. JM162ASF)	1	 25zł

30	Potencjometr 10 kΩ	1	 0,15 zł R8
31	Piny - gniazdo 1x40 proste o rozstawie złącza wyświetlacza	1	 1,2 zł
32	Piny proste 1x40o rozstawie złącza 2.54mm -wyświetlacza	1	 1,2 zł
Elementy dodatkowe			
33	Złącza AK (LE lub TB) rozstaw 5mm	1	 0,6
34	Podstawka dip 18 precyzyjna	1	 1,25 zł
35	Układ scalony 8x Darlington Driver ULN2803A DIP18		 1,2 zł IC3
36	Dioda świecąca prostokąt 2x5mm	2szt	 LED 4 LED 3
37	Koszulka termokurczliwa 5 mm		0,5 zł
38	Koszulka termokurczliwa 3,5 mm		0,5 zł
39	Przycisk (taki sam jak reset) 6x6mm h = 5mm Normalnie otwarty	2	 0,5 zł S2 , s\S3
40	Rezystor 560 Ω 0,125W	2	 0,1 zł R2, R3
41	Rezystor 4,7 kΩ 0,125W	2	 0,1 zł R6, R7
42	Piny proste 2x40o rozstawie złącza 2,54mmwyświetlacza	1	 1,5 zł
43	Kołki dystansowe + nakrętki	4	Dowolne, zalecane metalowe 12 mm wysokości
44	Laminat jednostronny i środek trawiący	1	Laminat 10 x 8 cm ok. 16zł

STEROWNIK KNEST

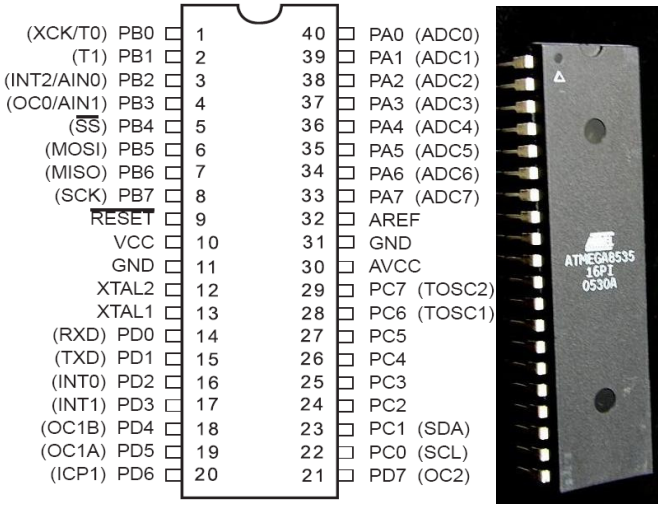
Dzięki rozwojowi technologii powstały układy z wbudowaną pamięcią flash i możliwością programowania mikroprocesora zamontowanego w układzie, co pozwala pominąć układy wykorzystujące pamięci typu Eprom lub EEprom. Mikroprocesor firmy Atmel ATMEGA 8535 umożliwia programowanie wewnętrznej pamięci flash w trybie szeregowym SPI (Serial Programming Interface). Wykorzystywany jest w tym celu port równoległy LPT dowolnego komputera klasy PC, na którym zainstalowane jest środowisko MCS BASCOM AVR DEMO.

Budowa mikroprocesorowego sterownika opartego o procesor ATMEGA 8535 firmy Atmel

AVR ATMEGA 8535 opiera się na architekturze harwardzkiej. Rozdzielono w niej przestrzeń adresowej pamięci programu i przestrzeni adresowej pamięci danych. Uzyskano to dzięki zastosowaniu oddzielnych magistral adresowych. Dzięki temu możliwe było zastosowanie słowa o różnej szerokości dla pamięci programu i pamięci danych. Chroni to przed przypadkowym odczytaniem danej i interpretowanie jej jako instrukcji.

Mikroprocesory AVR należą do grupy układów o architekturze RISC (Reduced Instruction Set Computer). Cechuje je to, że większość rozkazów RISC jest realizowana w jednym takcie zegara co zapewnia szybsze wykonanie programu. Programy pisane dla procesorów RISC charakteryzują się większą spójnością, a co za tym idzie mniejszym kodem wynikowym. (Architektura procesora 89S8252 jest określona nazwą CISC (Complex Instruction Set Computer) gdzie wykonanie jednego rozkazu CISC wymagało zazwyczaj wykonania wielu operacji, co zwykle trwało kilka taktów zegara.

Cechą wyróżniającą mikrokontrolery AVR jest również zaimplementowanie wielu rejestrów wewnętrznych, z których każdy może pełnić rolę akumulatora podczas wykonywania operacji arytmetycznych i logicznych. Minimalizuje to liczbę przesłań między rejestrowych, co korzystnie wpływa na szybkość wykonywania programu. Opis wyprowadzeń procesora oraz jego zdjęcie zaprezentowano na rys ...

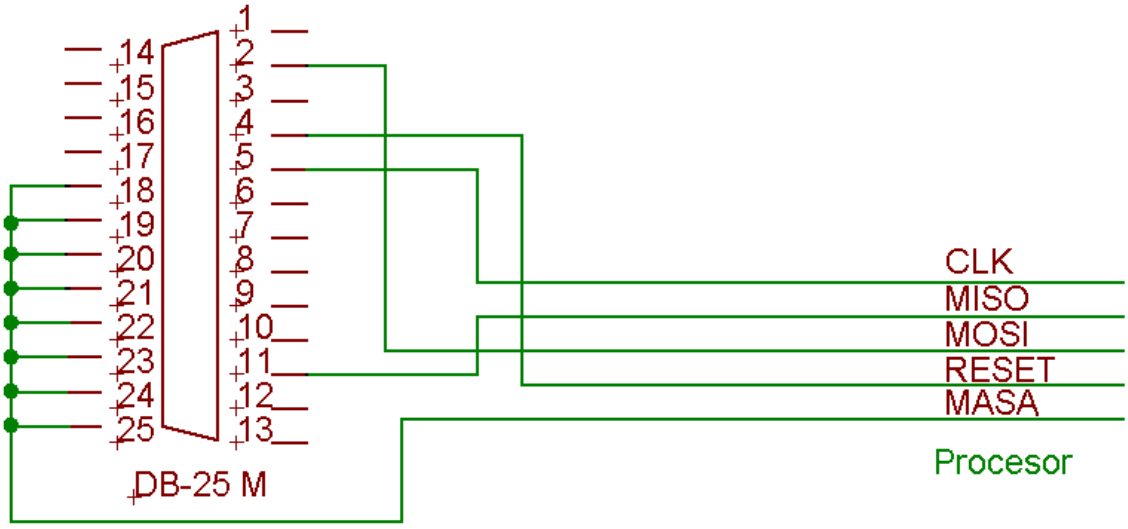


Rys.... Opis wyprowadzeń oraz widok procesora ATMEGA 8535

Mikrokontroler ATmega8 ma następujące parametry oraz cechy funkcjonalne:
• Mały pobór mocy
• Zaawansowana architektura RISC (Reduced Instruction Computer) charakteryzująca się:

- 130 instrukcjami, z których większość jest wykonywana w jednym cyklu maszynowym
- 32 rejestrami 8-bitowymi ogólnego przeznaczenia
- Dużą wydajnością 16 MIPS (miliona operacji na sekundę) przy częstotliwości zegara 16 MHz
• Pamięć:
- 8 kB nieulotnej pamięci Flash o trwałości 10 000 zapisów/kasowań,
- opcjonalne miejsce na Bootloader
- 512 B pamięci EEPROM o trwałości 100 000 zapisów/kasowań (może przechowywać dane do 10 lat)
- 1 kB pamięci SRAM,
- Wbudowane programowalne zabezpieczenia pamięci programu przed odczytem i zapisem
• Układy peryferyjne:
- dwa 8 – bitowe czasomierze/liczniki z oddzielnymi preskalerami,
- jeden 16 – bitowy czasomierz/licznik z oddzielnym preskalerem, możliwość pracy z w trybie Capture oraz Compare
- zegar czasu rzeczywistego z oddzielnym oscylatorem
- trzy kanały PWM (OC1, OC1B, OC2)
- 8-kanałowy przetwornik A/C – o rozdzielczości 10 bitów
- programowalny USART do transmisji przez RS232
- szeregowy interfejs Master/Slave SPI
- programowalny Watchdog z oddzielnym oscylatorem
- wbudowany komparator analogowy
• Specjalne wyposażenie mikrokontrolera:
- zerowanie po włączeniu mikrokontrolera
- wewnętrzny kalibrowany oscylator RC
- wewnętrzne oraz zewnętrzne źródła sygnałów przerwań
- pięć trybów uśpienia mikrokontrolera
- 32 linie I/O dowolnego wykorzystania
• Zakresy napięć zasilania:
- 2,7 V…5,5 V (ATmega 8535L)
- 4,5 V…5,5 V (Atmega 8535)
• Zakres częstotliwości sygnału taktującego mikroprocesor:
- 0…8 MHz (ATmega 8535L)
- 0…16 MHz (Atmega 8535)
• Pobierany prąd przy częstotliwości sygnału taktowania 4 MHz i przy napięciu zasilania 3V (ATmega8535L):
- w stanie aktywnym: 3,6 mA
- w trybie Idle: 1,0 mA
- w trybie Power-down: 0,5 uA

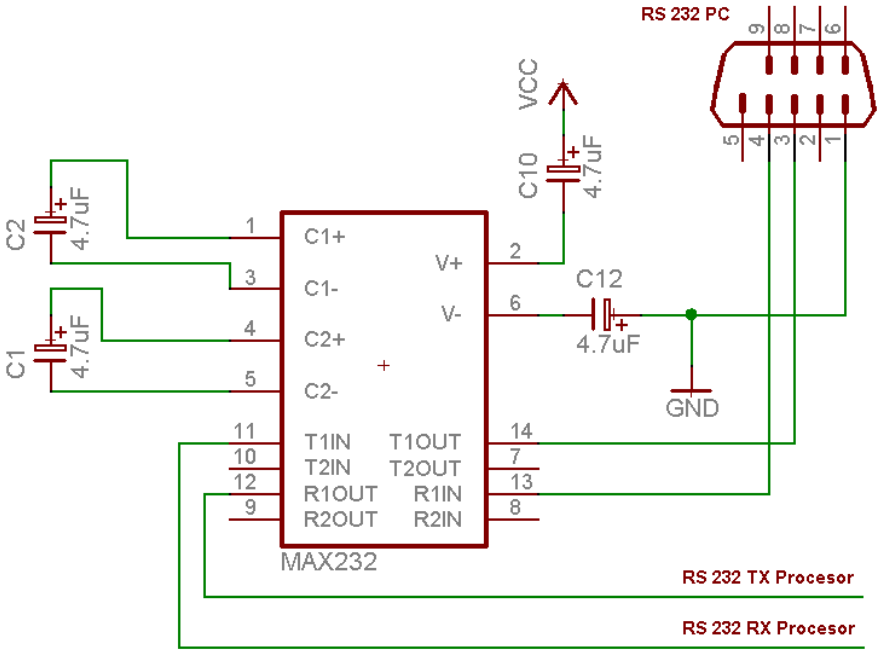
Najistotniejszym argumentem dla zastosowania tego typu mikroprocesora w konstrukcji sterownika był sposób jego programowania. Fakt ten przyczynia się do skrócenia czasu potrzebnego na przygotowanie i testowanie oprogramowania. Ułatwia dokonywanie zmian w programie i pozwala ocenić możliwości wykorzystania testowanego algorytmu. Schemat złącza programowarki SPI prezentuje poniższy rysunek :



Rys. Złącze programujące ISP

Dzięki zainstalowaniu złącza szeregowego typu RS232, sterownik może komunikować się z dowolnym urządzeniem, wykorzystującym komunikację szeregową w standardzie RS232C np. komputer PC. Do komunikacji wybrano dedykowany układ scalony MAX 232. Układ ten jest podwójnym nadajnikiem/odbiornikiem interfejsu RS232C który spełnia wszystkie wymagania normy EIA RS232C. Wymaga pojedynczego napięcia zasilania +5V i zawiera dwa konwertery dostarczające napięcia +10V i -10V. Wejścia układów są kompatybilne z TIL/CMOS, wyjścia mają ograniczenie szybkości narastania napięcia wyjściowego i wyjściową impedancję 300Ω w stanie wyłączenia zasilania. Odbiorniki mogą pracować z napięciem wejściowym do ±30V; impedancja wejściowa zawiera się w przedziale 3 do 7kΩ wejścia mają histerezę polepszającą odporność na zakłócenia.

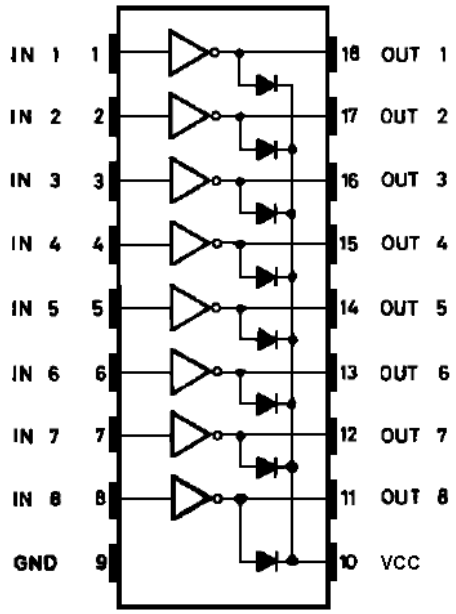
Schemat interfejsu komunikacyjnego przedstawiono na poniższym rysunku:



Rys. Złącze komunikacyjne

Układ bufora pełni rolę przedwzmacniacza dla układów wykonawczych większej mocy. Blok bufora napięciowego opartego na układzie ULN2803N umożliwia bezpośrednie sterowanie takich elementów jak np. przekaźniki czy silniki. Zastosowanie bloku pośredniczącego pomiędzy procesorem a blokiem

kluczy tranzystorowych zapobiega nadmiernemu obciążeniu prądowemu wyjść mikrokontrolera. Układ wykonany jest w układzie tranzystorów Darlingtona. Cechuje się maksymalnym napięciem zasilania równym 50V, obciążony może być prądem 500mA, posiada wbudowane diody zwrotne co pozwala na uzyskanie czasów przełączania rzędu 1us.
Blokowy schemat wzmacniacza mocy prezentuje rys:

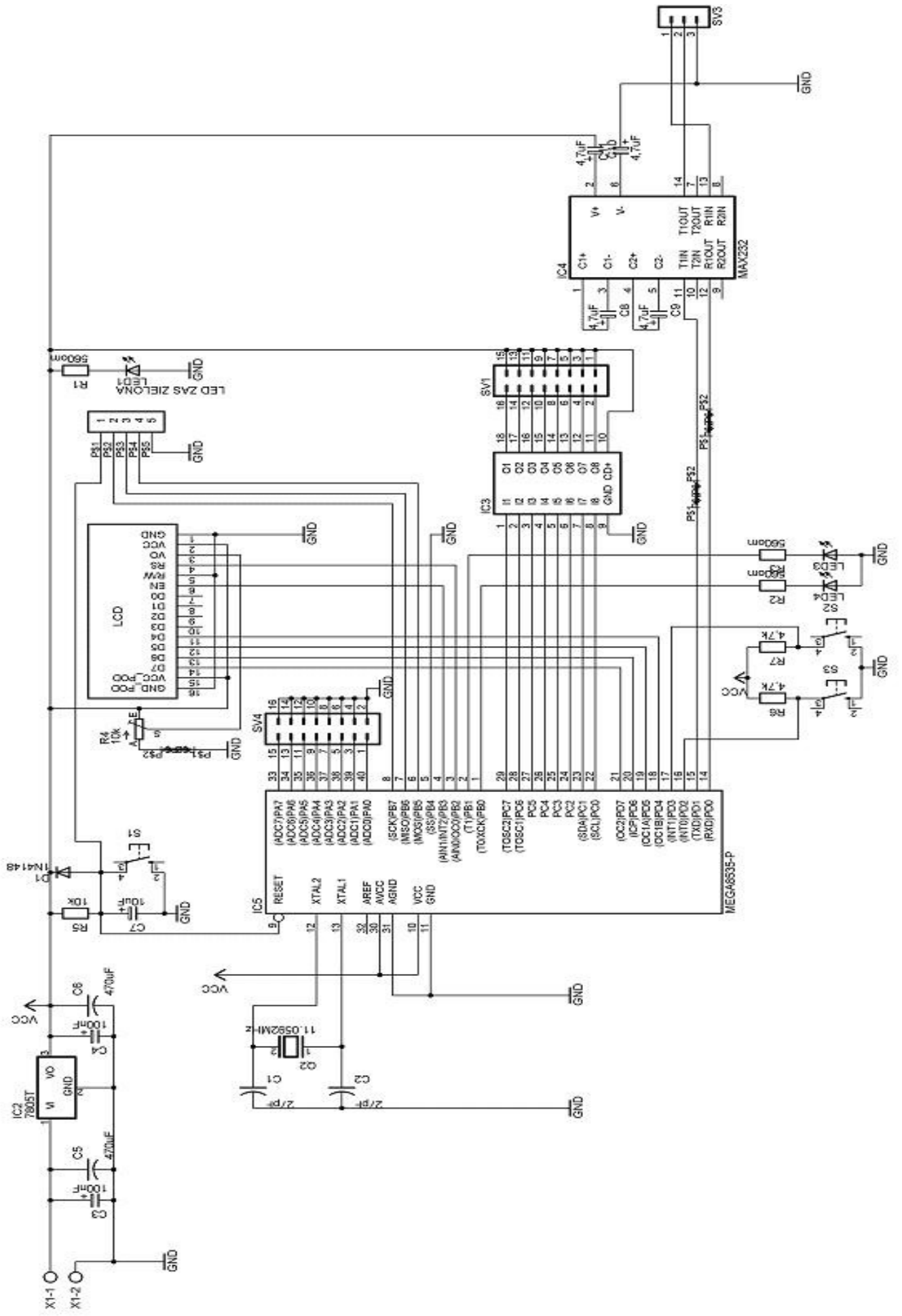


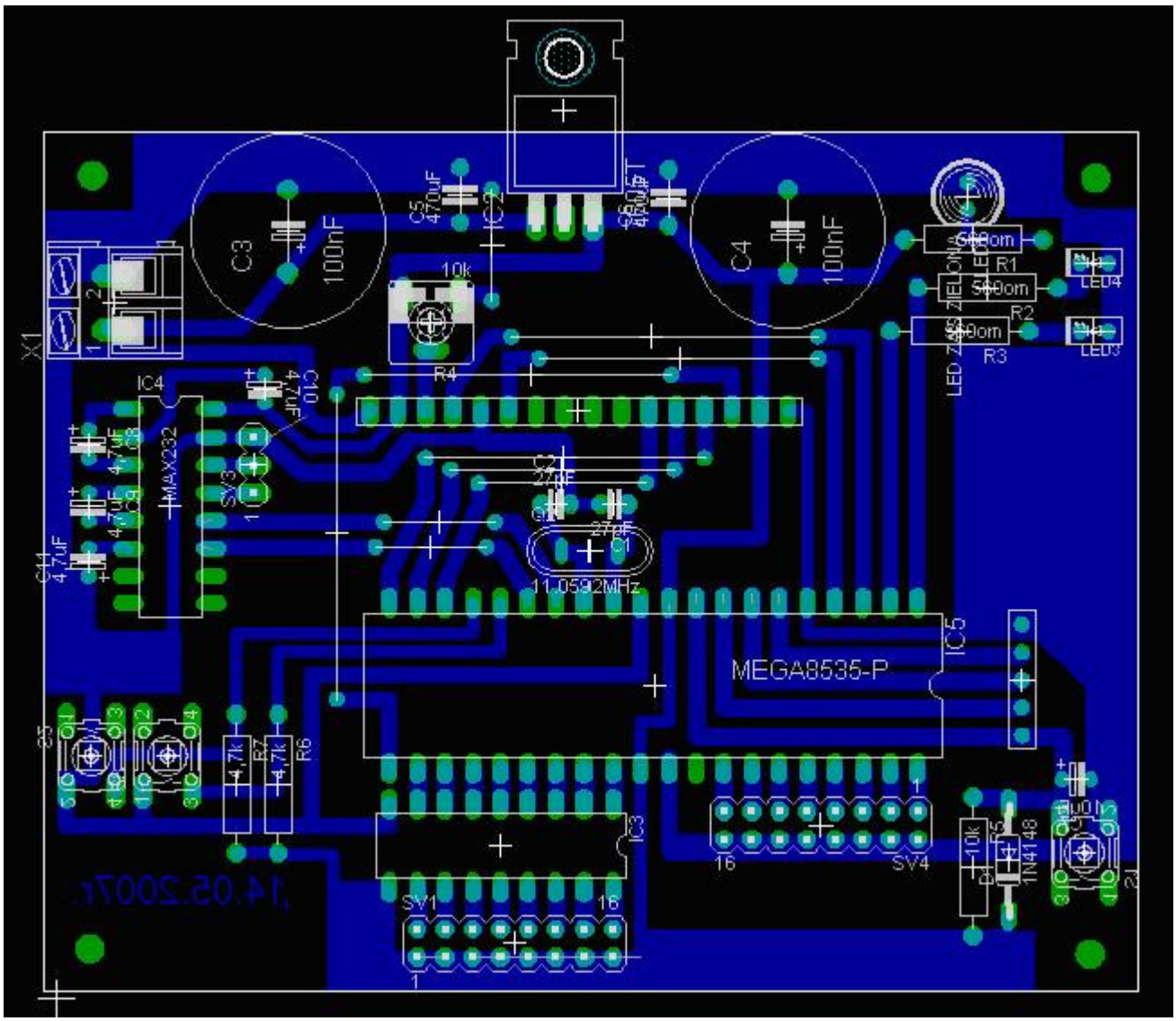
Rys. Schemat wewnętrzny drivera ULN2803

Schemat kompletnego sterownika prezentuje załącznik nr 1.

Podstawowe bloki wchodzące w skład sterownika to:

- Blok zasilania
- Blok procesora wraz z układem zegarowym i resetującym
- Blok układu programowarki
- Blok komunikacji RS232
- Blok wzmacniacza mocy

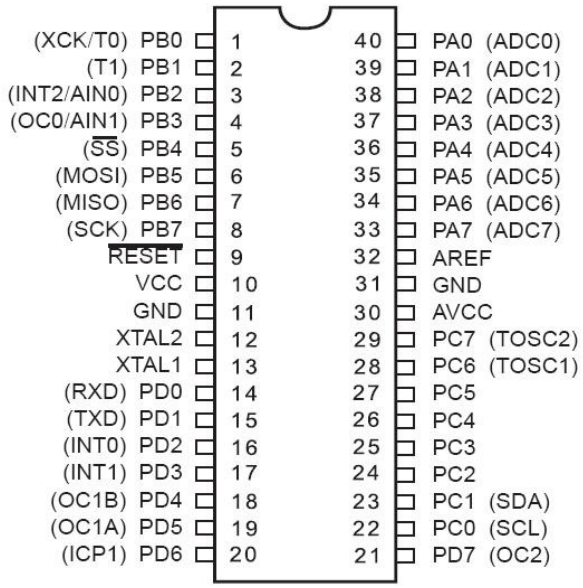

SV3
GND
4,7uF
C11
C10
4,7uF
IC4
MAX232
C1+
C1-
C2+
C2-
V+
V-
T1IN
T2IN
R1OUT
R2OUT
T1OUT
T2OUT
R1IN
R2IN
4,7uF
C8
4,7uF
C9
560om
R1
LED ZAS ZIELONA
LED1
SV1
IC3
GND
P31
P32
P33
P34
P35
LCD
GND
VCC
VO
RS
RW
EN
D0
D1
D2
D3
D4
D5
D6
D7
VCC_PCD
GND_PCD
R4
10k
SV4
560om
R2
LED2
560om
R3
LED3
4,7k
R7
S3
4,7k
R8
VCC
S2
1N4148
D1
S1
R5
10k
C7
10uF
IC5
MEGA8535-P
RESET
XTAL2
XTAL1
AREF
AVCC
AGND
VCC
GND
11.0592MHz
Q2
C1
22pF
C2
22pF
VCC
GND
IC2
7805T
VI
VO
GND
C6
470uF
C4
100nF
C5
470uF
C3
100nF
X1-1
X1-2
GND

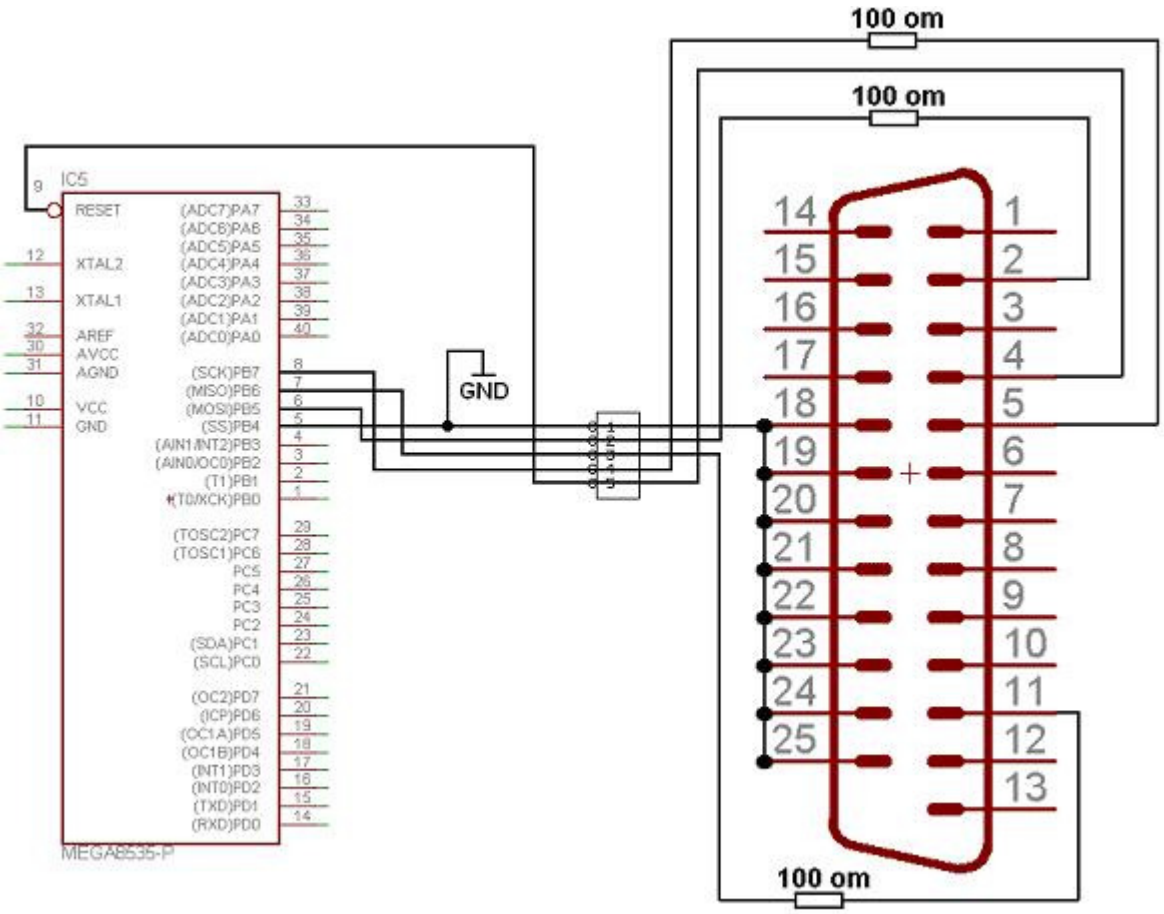

C3
100nF
C4
100nF
C5
470uF
C6
470uF
IC2
10k
R4
X1
IC4
MAX232
C10
4,7uF
C11
4,7uF
C12
4,7uF
C13
4,7uF
C14
4,7uF
SV3
C2
27pF
C1
27pF
11,0592MHz
MEGA8535-P
IC5
IC3
SV1
SV4
R1
330om
R2
330om
R3
330om
LED1
LED2
LED3
LED4
R7
4,7k
R6
4,7k
R5
10k
1N4148
D1
P1
P2

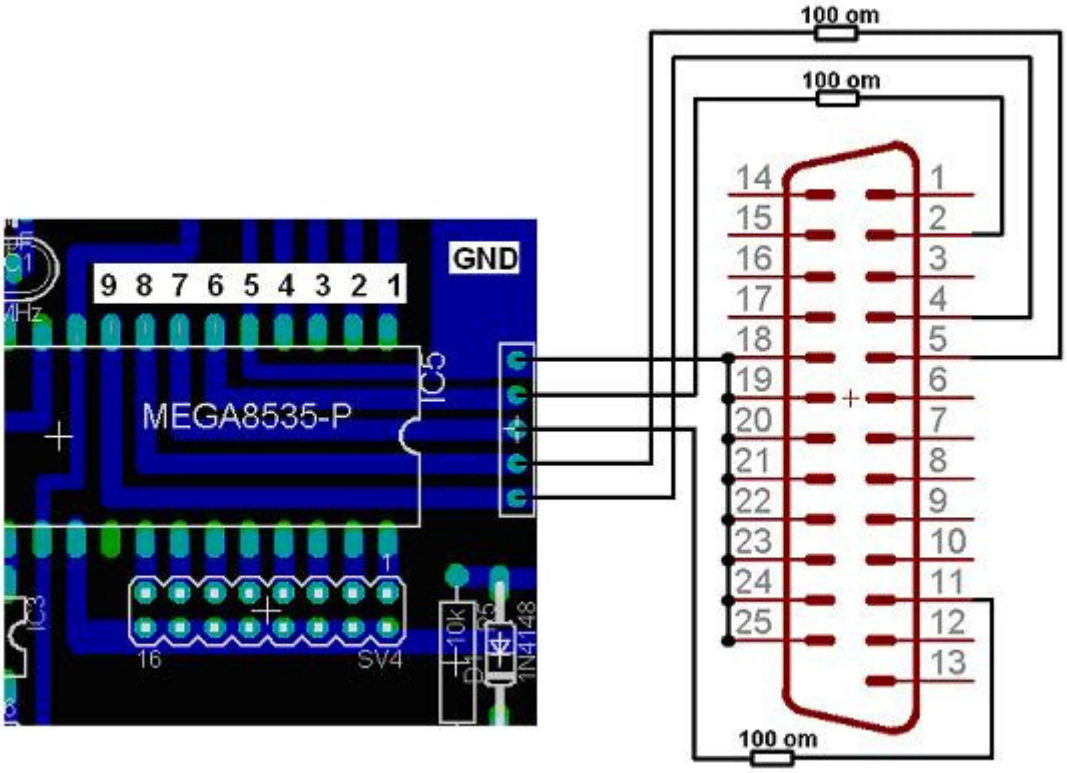
SCHEMAT PROGRAMOWARKI (Sample Electronics cable programmer)

Złącze D-Sub 25 pin		ATMEGA8535-16PU	
Nr pin	**Funkcja**	**Nr pin**	**Funkcja**
2	**D0**	**6**	**PB 5 - MOSI**
4	**D2**	**9**	**RESET**
5	**D3**	**8**	**PB 7 - CLOCK**
11	**BUSY**	**7**	**PB 6 - MISO**
18-25	**GND**	**11**	**GND**
18-25	**GND**	**5**	**PB 4 - SS (DO GND)**

ATMEGA8535-16PU



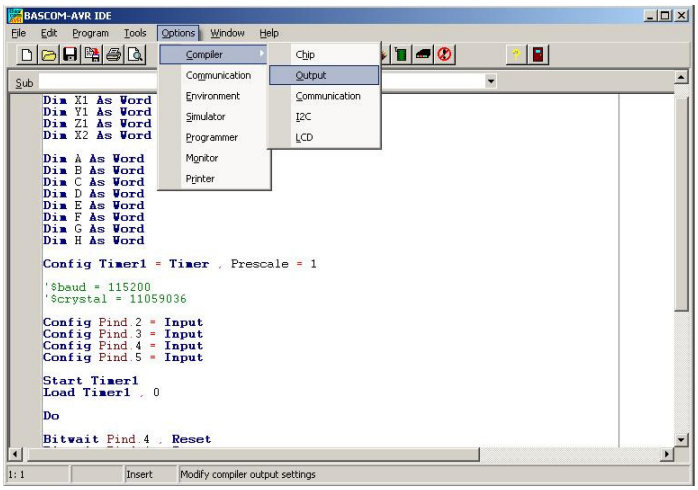

100 om
100 om
IC5
RESET
XTAL2
XTAL1
AREF
AVCC
AGND
VCC
GND
(ADC7)PA7
(ADC6)PA6
(ADC5)PA5
(ADC4)PA4
(ADC3)PA3
(ADC2)PA2
(ADC1)PA1
(ADC0)PA0
(SCK)PB7
(MISO)PB6
(MOSI)PB5
(SS)PB4
(AIN1/INT2)PB3
(AIN0/OC0)PB2
(T1)PB1
(T0/XCK)PB0
(TOSC2)PC7
(TOSC1)PC6
PC5
PC4
PC3
PC2
(SDA)PC1
(SCL)PC0
(OC2)PD7
(ICP)PD6
(OC1A)PD5
(OC1B)PD4
(INT1)PD3
(INT0)PD2
(TXD)PD1
(RXD)PD0
MEGA8535-P
GND
100 om


100 om
100 om
GND
9 8 7 6 5 4 3 2 1
MEGA8535-P
IC5
IC3
16
SV4
10k
1N4148
100 om

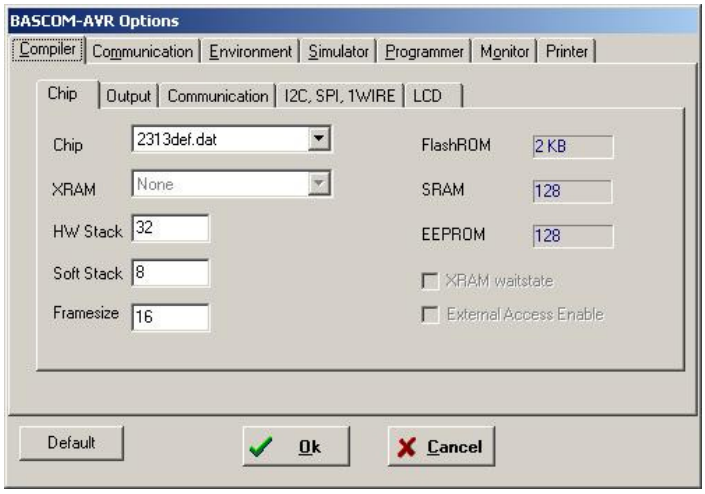
Konfiguracja wstępna programu Bascom AVR

Konfiguracja programu Bascom AVR jest stosunkowo prosta i nie powinna nastręczyć większych kłopotów.

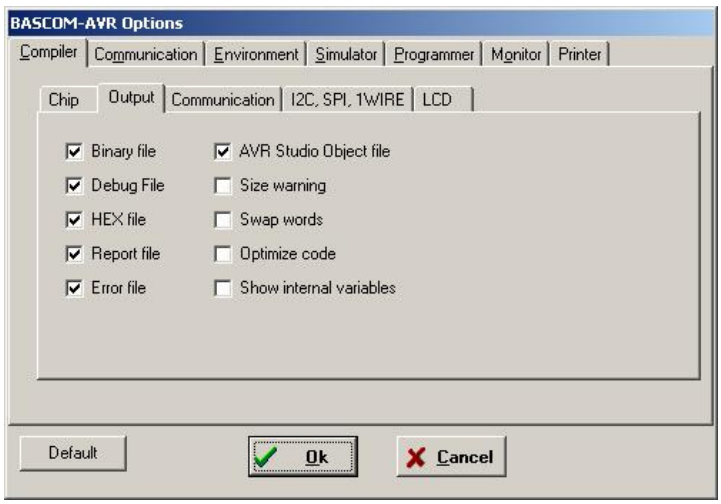
1. Po zainstalowaniu i uruchomieniu aplikacji otwieramy okno
   OPTIONS > COMPILER > OUTPUT



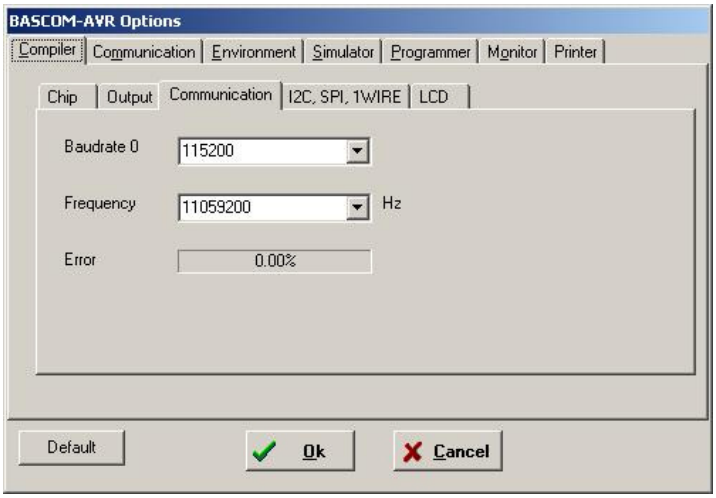
2. Na wstępie wybieramy procesor którego będziemy używali z dostępnych na liście:
(W naszym przypadku zamiast attiny2313.dat wybieramy mega8535.dat lub m8535.dat - w zależności od wersji programu oznaczenie może się różnić)



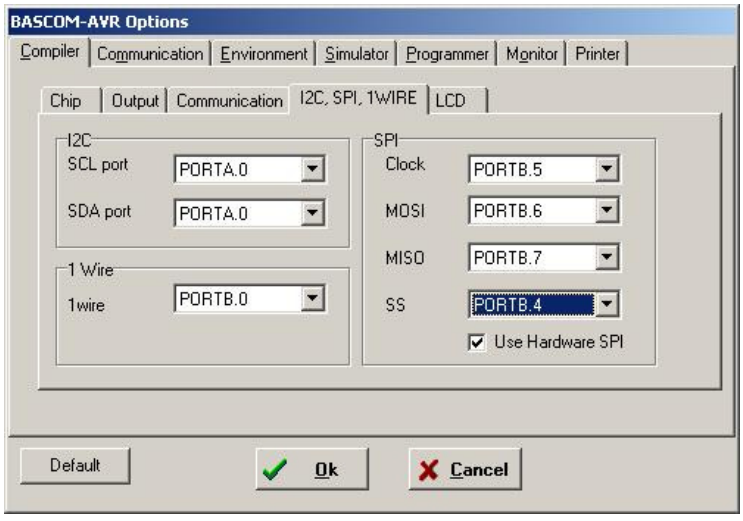
2. Następnie w zakładce OUTPUT zaznaczamy pola plików jakie mają być utworzone przez Bascom po skompilowaniu programu. Do poprawnej pracy programatora należy zaznaczyć wszystkie pola z poniższego rysunku (domyślnie są tak zaznaczone):



3. W zakładce COMMUNICATION ustawiamy prędkość komunikacji przy transmisji szeregowej (domyślnie na 115200) oraz częstotliwość stosowanego rezonatora kwarcowego (zalecane 11,0592 MHz lub 8 MHz).

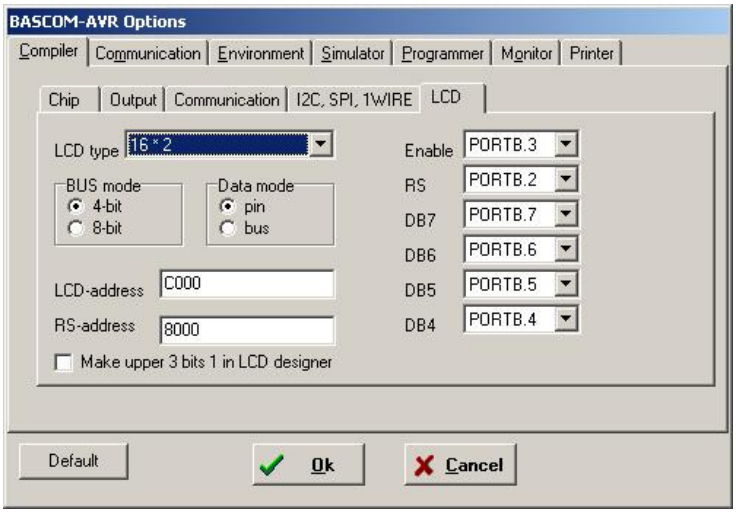


4. W polach zakładki I²C, SPI, 1WIRE ustawiamy linie portów procesora wykorzystywane przez magistrale komunikacyjne do transmisji. W przypadku używania programatora „Sample Elektronic Programmer" zalecane jest załączenie opcji „ Use Hardware SPI"



5. W polach zakładki LCD ustawiamy linie portów procesora wykorzystywane przez wyświetlacz LCD do wyświetlania informacji:

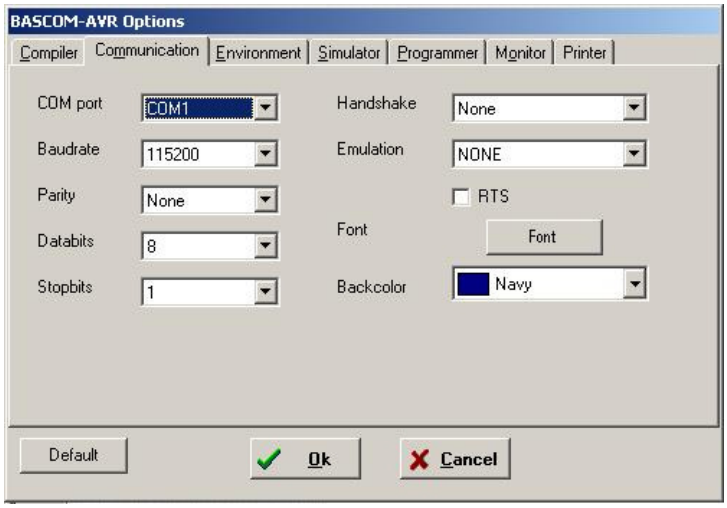
LCD type 16 * 2
ENABLE = Portb.3,
Rs = Portb.2
Db7 = Portd.7
Db6 = Portd.6
Db5 = Portd.5
Db4 = Portd.4



Na tym kończy się konfiguracja ustawień procesora

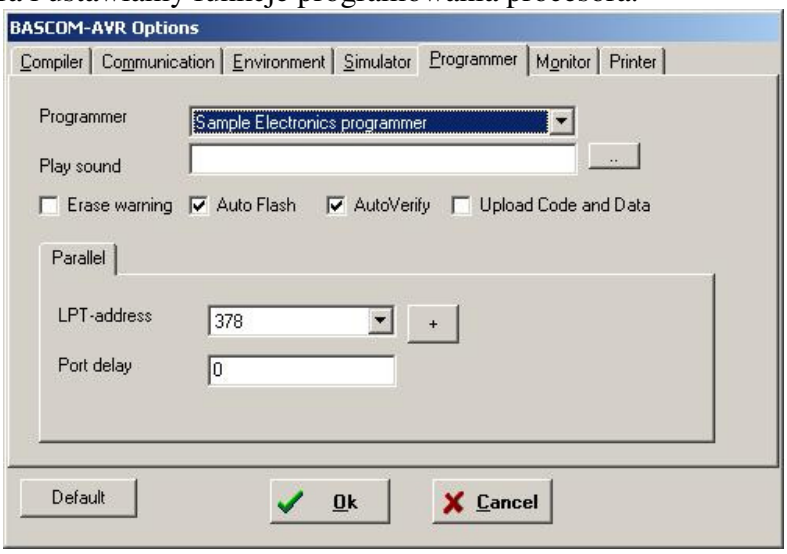
W dalszej części ustawiamy:

1. W zakładce COMMUNICATION określamy parametry komunikacji komputera z procesorem przez port RS 232 od strony komputera. Konfigurujemy ustawienie według naszych potrzeb i zgodnie z wartościami ustawionymi w komunikacji ze strony procesora.



2. W zakładce ENVIORONMENT ustawiamy parametry użytkowe samego programu, wg. naszej wygody.

3. W zakładce 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.
4. Okienko PROGRAMMER jest odpowiedzialne za programowanie procesora. W nim wybieramy rodzaj programatora i ustawiamy funkcje programowania procesora.



Programator zalecany to „Sample Electronic Programmer". Dodatkowo zaznaczamy pola AUTO FLASH oraz AUTO VERIFY. Port LPT , port szeregowy i reszty ustawień nie zmieniamy.

Uff ale trudne. ☺
Nareszcie można stwierdzić .... konfiguracja zakończona.

Materiały ze strony http://www.edw.com.pl/ea/bascom_avr_inst.html

OPIS IKONEK

Pasek menu

W skład paska menu wchodzą zakładki, których znaczenie wyjaśniono poniżej:

File - ta zakładka wraz z jej poleceniami jest z pewnością wszystkim znana z innych programów windowsowych:

 - otwiera nowe okno edytora dla nowego pliku, w którym można zapisywać kod programu - skrót klawiszowy **Ctrl+N**,

 - otwiera poprzednio zapisany plik - skrót klawiszowy **Ctrl+O**,

 - zamyka otwarty plik - skrót klawiszowy **Ctrl+F4**,

 - zapisuje otwarty i edytowany właśnie plik, o nadanej wcześniej nazwie - skrót klawiszowy **Ctrl+S**,

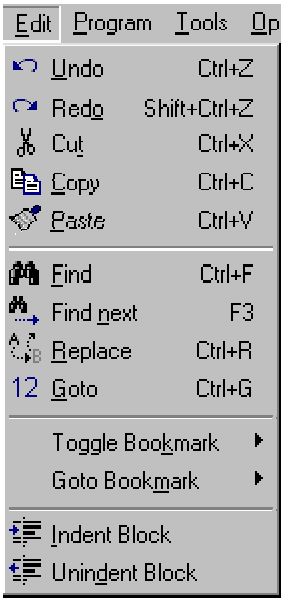
 - zapisuje otwarty i edytowany właśnie plik z poleceniem nadania nazwy,

- umożliwia podgląd wydruku otwartego pliku,

- polecenie drukowania otwartego pliku - skrót klawiszowy **Ctrl+P**,

- wyjście z pakietu Bascom-AVR - skrót klawiszowy **Alt+F4**

Edit - również typowa zakładka programów windowsowych zawierająca znane polecenia takie jak kopiowanie, wycinanie, wklejanie itp:



- powoduje cofnięcie wykonanej ostatnio zmiany w programie - skrót klawiszowy **Ctrl+Z**,

- przywraca ponownie poprzednio wykonaną, a następnie cofniętą zmianę w programie - skrót klawiszowy **Shift+Ctrl+Z**,

- wycina zaznaczony tekst i przechowuje go w schowku - skrót klawiszowy **Ctrl+X**,

- kopiuje zaznaczony tekst do schowka - skrót klawiszowy **Ctrl +C**,

- wkleja zawartość schowka w miejsce wskazane kursorem - skrót klawiszowy - **Ctrl+V**,

- wyszukiwanie w programie fragmentu tekstu, który wpisuje się w okienko dialogowe *"Find Text"*, w okienku tym można poustawiać różne opcje dotyczące wyszukiwania - skrót klawiszowy **Ctrl+F**,

- wyszukiwanie następnego wystąpienia tekstu wpisanego w okienko dialogowe *"Find Text"* - skrót klawiszowy **F3**,

- zastępuje wszystkie znalezione słowa wpisane w okienku dialogowym *"Replace Text"* nowymi słowami również wpisanymi w tym okienku, w okienku tym można poustawiać różne opcje dotyczące wyszukiwania i zastępowania - skrót klawiszowy **Ctrl+R**,

- skok do wybranej linii programu, którą to linię wpisuje się w okienku dialogowym *"Go to line Number"* - skrót klawiszowy **Ctrl+G**,

Toggle Bookmark - ustawia znaczniki w programie, widoczne są one z lewej strony okienka edytora, znaczników może być maksymalnie 8, ułatwiają one szybkie przemieszczanie się po tekście programu,

Goto Bookmark - umożliwia skok do wybranego znacznika - skrót klawiszowy **Ctrl+x**, gdzie x to numer znacznika,

- przesuwa zaznaczony fragment o ilość pozycji jaka jest ustawiona w zakładce *"Options->Environment->Editor->Tab-size"*,

 - cofa wstawiony akapit,

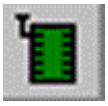
Program - polecenia z tej zakładki uruchamiają programy służące do kompilacji, sprawdzania składni, symulacji i wysyłania programu do mikrokontrolera:

 - uruchamia kompilację programu - skrót klawiszowy **F7**,

 - uruchamia wykrywanie błędów składni w pisanym programie - skrót klawiszowy **Ctrl+F7**,

 - pokazuje raport z kompilacji - skrót klawiszowy **Ctrl+W**,

 - uruchamia symulację programu - skrót klawiszowy **F2**,

 - uruchamia program umożliwiający zaprogramowanie mikrokontrolera - skrót klawiszowy - **F4**,

W procesorach AVR na początku zawsze należy skonfigurować porty określając dla każdego wyprowadzenia dwa parametry:
- funkcję jaką ma pełnić - wejścia czy wyjścia
- stan spoczynkowy jaki ma przyjąć - "0" czy "1"
Jeżeli porty nie zostaną skonfigurowane, to przy starcie mikrokontrolera (po wyzerowaniu) rejestry PORTx i DDRx (x w zależności od portu będzie zastąpiony literką A, B, C lub D) zostaną wyzerowane co oznacza, że wszystkie wyprowadzenia portów będą wejściami w stanie wysokiej impedancji (wejścia pływające)

W **helpie do BASCOM-AVR** (wersja w języku polskim) są dokładnie opisane wszystkie instrukcje i inne elementy języka BASCOM dla AVR, dlatego nie ma sensu abym w tym miejscu bardziej szczegółowo opisywać te elementy, umieszczone będą tylko niezbędne dla zrozumienia tematu informacje i wyjaśnienia.

Należy ściągnąć sobie tego helpa w wersji pl i często tam zaglądać!

Zgodnie z tym co jest napisane powyżej (w ramce) należy najpierw skonfigurować porty procesora. Do konfiguracji urządzeń sprzętowych w BASCOM-AVR służy instrukcja ***CONFIG***, którą się używa razem z nazwą konfigurowanego sprzętu, czyli w przypadku konfiguracji portu nazwą będzie ***PORTx***, a w przypadku pojedynczej końcówki nazwą będzie ***PINx.y***, gdzie x to port B lub D (dla innych mikrokontrolerów może być jeszcze A lub C), y to numer końcówki z danego portu (0 ... 6 lub 7). Za stan portów w mikrokontrolerach AVR odpowiedzialne są trzy rejestry DDRx, PORTx i PINx (x to: A, B, C lub D). Instrukcja CONFIG ustawia cały port lub wybraną końcówkę portu w tryb pracy wejścia lub wyjścia. Inaczej mówiąc ustawia odpowiednio rejestr kierunku czyli DDRx. Jeżeli do każdego bitu rejestru wpiszemy "1" to wszystkie wyprowadzenia portu będą wyjściami, natomiast jeżeli do każdego bitu rejestru DDRx wpiszemy "0" to wszystkie wyprowadzenia będą wejściami. W przykładach poniżej stosuję kolorystykę składni poleceń taką jaką mam ustawioną w BASCOM-AVR. Kolorem zielonym i kursywą wyróżniam komentarze które zawsze muszą być poprzedzone znakiem apostrofu ' lub instrukcją ***REM***. Warto przyzwyczaić się do opatrywania swoich programów komentarzami gdyż to po pewnym czasie ułatwia analizę wcześniej napisanych programów:

```
' - to jest komentarz
```

Rem - i to też jest komentarz Konfiguracja całego portu B jako wyjście lub wejście:

```
Config Portb = Output     ' cały port B jako wyjście
Config Portb = Input      ' cały port B jako wejście
```

można również skonfigurować każde wyprowadzenie (każdy bit) osobno:

```
Config Pinb.0 = Output    ' wyprowadzenie PB0 portu B jako wyjście
Config Pinb.1 = Output    ' wyprowadzenie PB1 portu B jako wyjście
Config Pinb.2 = Output    ' wyprowadzenie PB2 portu B jako wyjście
Config Pinb.3 = Output    ' wyprowadzenie PB3 portu B jako wyjście
Config Pinb.4 = Output    ' wyprowadzenie PB4 portu B jako wyjście
Config Pinb.5 = Output    ' wyprowadzenie PB5 portu B jako wyjście
Config Pinb.6 = Output    ' wyprowadzenie PB6 portu B jako wyjście
Config Pinb.7 = Output    ' wyprowadzenie PB7 portu B jako wyjście
Config Pinb.0 = Input     ' wyprowadzenie PB0 portu B jako wejście
Config Pinb.1 = Input     ' wyprowadzenie PB1 portu B jako wejście
Config Pinb.2 = Input     ' wyprowadzenie PB2 portu B jako wejście
Config Pinb.3 = Input     ' wyprowadzenie PB3 portu B jako wejście
Config Pinb.4 = Input     ' wyprowadzenie PB4 portu B jako wejście
Config Pinb.5 = Input     ' wyprowadzenie PB5 portu B jako wejście
Config Pinb.6 = Input     ' wyprowadzenie PB6 portu B jako wejście
Config Pinb.7 = Input     ' wyprowadzenie PB7 portu B jako wejście
```

powyższego zapisu nie polecam z oczywistych powodów, lepiej przedstawić to samo korzystając z tego, że bajt można przedstawić jako 8 bitów i od razu będzie widać, który bit jest wyjściem, a który wejściem:

```
Config Portb = &B11111111    ' cały port B jako wyjście
Config Portb = &B00000000    ' cały port B jako wejście
```

w tym przypadku wszystkie wyprowadzenia są wyjściami lub wejściami, gdyby zapisać:

```
Config Portb = &B11111100    ' wyprowadzenia PB0 i PB1 to wejścia,
                               ' PB2 do PB7 to wyjścia
```

oznaczałoby to, że bit 0 i bit 1 są ustawione na 0 co oznacza, że wyprowadzenie PB0 i PB1 portu B są wejściami, a pozostałe wyjściami.
Prefiks **&B** oznacza w BASCOM-AVR, że liczba występująca po tym znaku jest zapisana w postaci dwójkowej, prefiks **&H** oznacza liczbę zapisaną w kodzie szesnastkowym, bez prefiksu liczba w kodzie dziesiętnym.
Przypisanie funkcji dla portu to pierwsza rzecz, druga natomiast to określenie stanu spoczynkowego wyprowadzeń portów mikrokontrolera po starcie pogramu. W tym celu do każdego bitu rejestru PORTx należy wpisać "0" lub "1". Można tego dokonać dla każdego bitu osobno, lub dla całego rejestru od razu, podobnie jak to miało miejsce przy przypisaniu funkcji wejścia lub wyjścia.

Ustawienie stanu spoczynkowego portu B:

```
Portb = &B11111111     ' podciągnięcie wszystkich wyprowadzeń
                         ' portu B do "1"
```

dla portu B skonfigurowanego jako wyjście będzie to oznaczało, że stanem spoczynkowym wszystkich wyprowadzeń jest jedynka (stan wysoki), dla portu B skonfigurowanego jako wejście, to oznacza podciągnięcie wszystkich wejść do jedynki. Gdy zapiszemy:

```
Portb = &B00000000     ' dla portu B jako wyjście oznacza to
                         ' ustawienie na wszystkich wyprowadzeniach
                         ' stanu "0",
                         ' dla portu B jako wejście oznacza to
                         ' pozostawienie wszystkich wejść
                         ' pływających
```

będzie to oznaczało dla portu B skonfigurowanego jako wyjście ustawienie na wszystkich wyprowadzeniach stano logicznego "0", natomiast dla portu B skonfigurowanego jako wejście oznacza to, że wszystkie wejścia pozostają w stanie wysokiej impedancji (wejścia pływające). Warto pamiętać, że dla wyprowadzenia portu pełniącego rolę wejścia, do którego podpięty jest przełącznik zwierający go do zera (po naciśnięciu), ważnym jest aby to wejście było podciągnięte do jedynki.
Podobnie jak konfigurowanie pojedynczego wyprowadzenia portu można zrobić ustawienie stanu

spoczynkowego dla pojedynczego wyprowadzenia, co ilustruje poniższy zapis:

```
Portb.5 = 1    ' wyprowadzenie PB5 portu B podciągnięte do "1"
Portb.5 = 0    ' wejście PB5 portu B pływające lub
                ' wyjście PB5 portu B ustawione w stan"0"
```

Po tych wszystkich wyjaśnieniach wracamy do naszego układu i przygotujemy porty mikrokontrolera do pracy zgodnie z założeniami.

```
Config Portb = &B11111111  ' cały port B jako wyjście
       Portb = &B11111111  ' wszystkie wyjścia w stanie "1"
Config Portd = &B1111100   ' PD0 i PD1 - wejścia, pozostałe - wyjścia
       Portd = &B1111111   ' PD0 do PD6 podciągnięte do "1"
```

Nasz mikrokontroler przygotowany jest do pracy, SW1 i SW4 podpięte są do wejść PD0 i PD1, które podciągnięte są do jedynki, pozostałe wyprowadzenia portu D są wyjściami (co dla działania układu jest obojętne), cały port B jest wyjściem i stan spoczynkowy jest "1", co w przypadku diody LED1 podpiętej do PB0 powoduje, że dioda nie świeci. Aby program działał zgodnie z założeniami, to należy sterować stanem wyjścia PB0 poprzez zmianę najmłodszego bitu rejestru PORTB czyli Portb.0 w zależności od stanu wejść PD0 i PD1. Zmieniać stan bitu rejestru PORTB (i innych rejestrów PORTA, PORTD, PORTC również) można w BASCOM-AVR na dwa sposoby, aby dioda zaświeciła należy napisać:

```
       Portb.0 = 0  ' PB0 w stanie "0" - dioda świeci
```
lub
```
Reset Portb.0  ' PB0 w stanie "0" - dioda świeci
```

w drugim sposobie użyłem instrukcji ***RESET***, która ustawia określony bit w stan "0".
Aby dioda przestała świecić należy napisać:

```
       Portb.0 = 1  ' PB0 w stanie "1" - dioda nie świeci
```
lub
```
Set Portb.0  ' PB0 w stanie "1" - dioda nie świeci
```

tutaj w drugim sposobie użyłem instrukcji ***SET***, która Ustawia określony bit w stan "1".
Czas na najważniejsze, a więc co trzeba zrobić aby program ciągle sprawdzał stan wejść, do których są podpięte przełączniki SW1, SW4 i w zależności od ich stanów zmieniał stan wyjścia PB0 powodując świecenie bądź gaszenie diody LED1. Aby program wykonywał w kółko jakąś operację należy zastosować w programie nieskończoną pętlę. W BASCOM-AVR może do tego celu posłużyć instrukcja ***DO ... LOOP***, która powtarza blok programu dopóki warunek końcowy nie będzie spełniony. Gdy warunek nie zostanie podany (a tak będzie w naszym przypadku) pętla będzie się wykonywać w nieskończoność. Zapisać to więc można tak jak poniżej:

```
Do     ' początek nieskończonej pętli
   ciąg wykonywanych instrukcji
Loop   ' powrót do początku pętli
```

Teraz pozostało jeszcze spowodowanie sprawdzania stanów wejść PD0, PD1 i uzależnienia od nich stanu PB0, czyli musimy wykonać instrukcję w stylu: *"wykonaj coś pod warunkiem, że jest spełnione coś innego"*.
W BASCOM-AVR jest instrukcja ***IF ... THEN ... ELSE ... END IF***, która znakomicie się do naszych celów nadaje. Tworzy ona tzw. blok decyzyjny. Instrukcja ***IF ... THEN*** oblicza logiczną wartość podanego wyrażenia. Jeśli będzie ono prawdziwe (wynikiem będzie logiczna prawda) wykonany zostanie blok instrukcji umieszczony po instrukcji ***THEN***. Jeśli będzie ono fałszywe, to instrukcje po słowie ***THEN*** nie zostaną wykonane. Wykonane za to będą instrukcje po słowie ***ELSE***, jeśli ono występuje. W naszym przypadku musimy więc sformułować następujący blok decyzyjny: *"... jeśli PD0 jest w stanie zero, to ustaw PB0 w stan "0" jeśli tak nie jest to przejdź do następnej instrukcji, w której będzie następna decyzja - jeśli PD1 jest w stanie zero, to ustaw PB0 w stan "1", jeśli tak nie jest to wróć do początku ..."*. I tu mała niespodzianka - jak odczytać stan dowolnego wyprowadzenia portu? Umiemy wpisywać do dowolnego bitu portu jedynkę lub zero poleceniem ***PORTx.y = 1*** czy ***PORTx.y = 0***, a jak jest z odczytem? I tu okazuje się przydatny trzeci rejestr PINx (to jest rejestr tylko do odczytu; x to: A, B, C lub D). Do odczytu stanu wyprowadzenia dowolnego portu służy polecenie ***PINx.y***. Jeśli już wszystko jasne, to czas zapisać nasz blok decyzyjny:

```
Do                                  ' początek nieskończonej pętli
  If Pind.0 = 0 Then Portb.0 = 0    ' jeśli na wejściu PD0 jest 0 to wyjście
                                    ' PB0 przyjmuje stan "0"
  If Pind.1 = 0 Then Portb.0 = 1    ' jeśli na wejściu PD1 jest 0 to wyjście
                                    ' PB0 przyjmuje stan "1"
```

```
Loop                                    ' powrót do początku pętli
```

Teraz wystarczy połączyć wszystkie niezbędne elementy programu czyli *"blok konfigurujący porty"* oraz ostatnio napisany *"blok decyzyjny"*, całość zakończyć instrukcją ***END*** i otrzymamy pierwszy program:

```
Config Portb = &B11111111  ' cały port B jako wyjście
       Portb = &B11111111  ' wszystkie wyjścia w stanie "1"
Config Portd = &B1111100   ' PD0 i PD1 - wejścia, pozostałe - wyjścia
       Portd = &B1111111   ' PD0 do PD6 podciągnięte do "1"
Do                                      ' początek nieskończonej pętli
   If Pind.0 = 0 Then Portb.0 = 0    ' jeśli na wejściu PD0 jest 0 to wyjście
                                        ' PB0 przyjmuje stan "0"
   If Pind.1 = 0 Then Portb.0 = 1    ' jeśli na wejściu PD1 jest 0 to wyjście
                                        ' PB0 przyjmuje stan "1"
Loop                                    ' powrót do początku pętli
End                                     ' koniec programu
```

Teraz wystarczy zaprogramować procesor i sprawdzić działanie programu . Powodzenia!

Pierwszy program do sterownika KNEST:

```
$regfile = "8535def.dat"

$crystal = 11059200
$baud = 19200

Config Lcd = 16 * 2
Config Lcdpin = Pin , Db4 = Portd.4 , Db5 = Portd.5 , Db6 = Portd.6 , Db7 = Portd.7 , E = Portb.3 , Rs = Portb.2
Config Pinb.0 = Output
Config Pinb.1 = Output
Config Pind.2 = Input
Config Pind.3 = Input

Config Portc = Output
Config Porta = Output

P1 Alias Pind.2
P2 Alias Pind.3

Dim A As Byte



Cls
Cursor Off
Lcd "KNEST"

For A = 1 To 10
  Shiftlcd Right              'przesuniemy tekst w prawo
  Waitms 200                            'czekamy aż wykona
Next

For A = 1 To 9
  Shiftlcd Left               'przesuniemy tekst w lewo
```

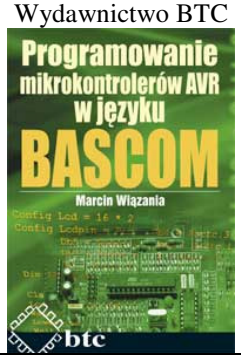
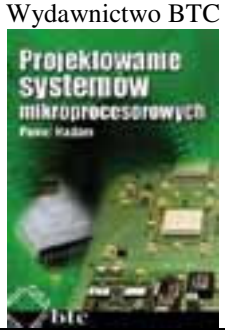
```
   Waitms 200                          'czekamy aż wykona
Next

Cls

Do

If P1 = 0 Then
Cls
Locate 2 , 2
Lcd "Dziala"

End If
If P2 = 0 Then
Cls
Locate 2 , 2
Lcd "Dobrze"

End If

Locate 1 , 2
Lcd "))))  KNEST  (((("
Print "Witam"
Set Portb.1
Set Portb.0
Portc = 255
Porta = 255
Wait 1

Cls
Reset Portb.1
Reset Portb.0
Portc = 0
Porta = 0
Wait 1
Loop
END
```

Polecane strony internetowe:

L.P.	Strona internetowa:	Opis:
1.	www.mcselec.com	Producent programu Bascom AVR
2.	www.atmel.com	Producent procesorów
3.	www.elektroda.pl	Polecane forum dyskusyjne
4.	www.elenota.pl	Noty katalogowe elementów
5.	www.tme.pl	Sklep internetowy z elektroniką
6.	www.avt.com.pl	Wydawnictwo, własne forum, sklep
7.	www.propox.com.pl	Producent modułów elektronicznych
8.	www.soyter.com.pl	Sklep z elektroniką – moduły transmisji
9.	www.analog.com	Światowy producent elektroniki

10.	www.national.com	Światowy producent elektroniki
11.	www.maxim-ic.com	Światowy producent elektroniki
12.	www.piekarz.pl	Sklep elektroniką – Warszawa ul. Wolumen
13.	www.semiconductors.com.pl	Sklep elektroniką – Warszawa ul. Hoża
14.	www.elportal.pl	Strona gazety Elektronika dla wszystkich
15.	www.sklep.avt.pl	Sklep internetowy wydawnictwa AVT
16.	www.elektronikapraktyczna.pl	Strona gazety Elektronika Praktyczna

Polecane książki i wydawnictwa:

L.P.	Nazwa:	Opis:
1.	Elektronika Plus Wydanie specjalne 1(2) 2004 BASCOM Wydawnictwo AVT 	Numer poświęcony programowaniu w BASCOM Dodatkowo płytka testowa AVT –2500 i płyta CD Cena ok. 35 zł
2.	Programowanie mikrokontrolerów AVR w języku Bascom. Marcin Wiązania. Wydawnictwo BTC 	W książce opisano sposób wykorzystywania pakietu Bascom oraz podstawy języka Bascom Basic. Korzystanie ze specyficznych instrukcji języka zilustrowano wieloma przykładami z obszernymi komentarzami. Książka jest przeznaczona dla uczniów i studentów uczelni technicznych, a także wszystkich elektroników wykorzystujących w swych projektach mikrokontrolery AVR. Cena ok. 55 zł
3.	Projektowanie systemów mikroprocesorowych. Paweł Hadam. Wydawnictwo BTC 	Idealna książka dla wszystkich elektroników projektujących urządzenia wykorzystujące mikroprocesory. W książce znajduje się wiele praktycznych informacji dotyczących projektowania systemów cyfrowych zbudowanych z wykorzystaniem mikrokontrolerów. Uwagi praktyczne dotyczące wykorzystania konkretnego mikrokontrolera w takim, a nie innym układzie pracy, odnoszą się do najpopularniejszych w naszym kraju typów mikrokontrolerów: zwłaszcza mikrokontrolerów rodziny '51 (przede wszystkim najprostszych układów firmy Atmel), a także układów PIC i AVR. Cena ok. 53 zł
4.		
5.		
6.		