1. Instalacja WinAVR

Strona główna: http://winavr.sourceforge.net/
Do pobrania z: http://sourceforge.net/project/showfiles.php?group_id=68108

Instalujemy ze standardowymi ustawieniami.

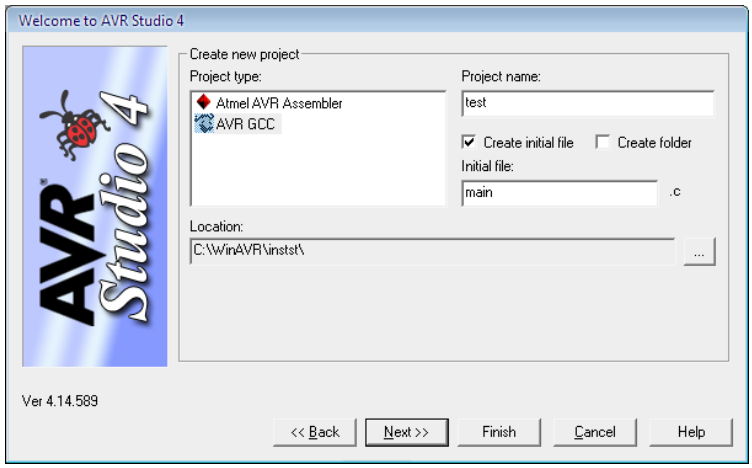
2. Instalacja AVR Studio 4

Strona główna: http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725
Do pobrania z: http://www.atmel.com/dyn/resources/prod_documents/aStudio4b589.exe

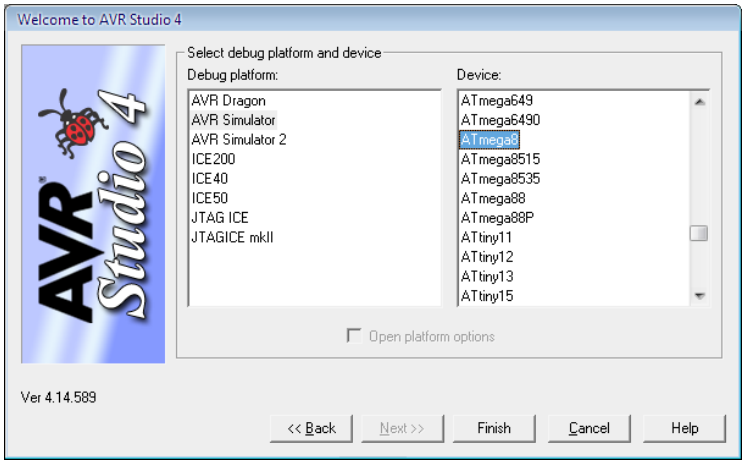
Instalujemy ze standardowymi ustawieniami.

3. Utworzenia nowego projektu

1. Uruchamiamy AVR Studio
2. Klikamy „New project”
3. Project type: AVR GCC
4. Project name: test
5. Zaznaczone Create initial file
6. Initial file: main.c
7. Location: jakiś katalog
8. Next >>

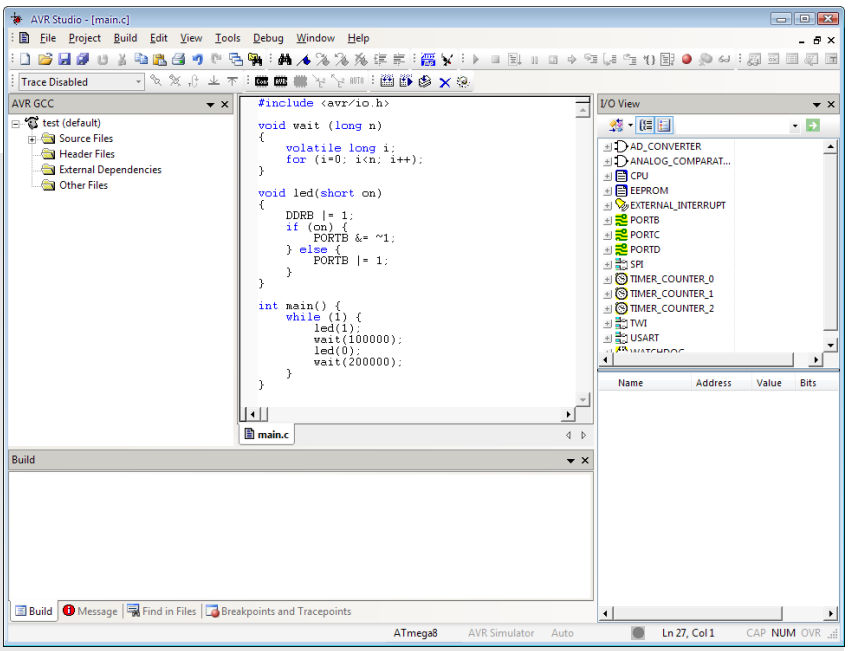


9. Debug platform: AVR Simulator
10. Device: ATmega8
11. Finish



4. Przykładowy program

Wklejamy przykładowy program.




```
#include <avr/io.h>

void wait (long n)
{
        volatile long i;
        for (i=0; i<n; i++);
}

void led(short on)
{
        DDRB |= 1;
        if (on) {
                PORTB &= ~1;
        } else {
                PORTB |= 1;
        }
}

int main() {
        while (1) {
                led(1);
                wait(1000);
                led(0);
                wait(2000);
        }
}
```

W ustawieniach projektu (Project > Configuration Options) ustawiamy Frequency na 1000000 hz i stopień optymalizacji na -O0, czyli brak optymalizacji. Jest to konieczne do symulacji, finalny program powinien zostać skompilowany z optymalizacją -Os, czyli optymalizacja ukierunkowana na dużą szybkość działania oraz mały rozmiar.

5. Kompilacja

Klikamy: Build > Build. W oknie Build powinna się pokazać informacja o prawidłowym zakończeniu kompilacji oraz dodatkowo rozmiar skompilowanego programu i danych.

6. Symulacja

Klikamy: Debug > Start debugging. Pojawiła się żółta strzałka na początku programu – jest to aktualna pozycja wykonywanego kodu. Pasek narzędzi Debug:

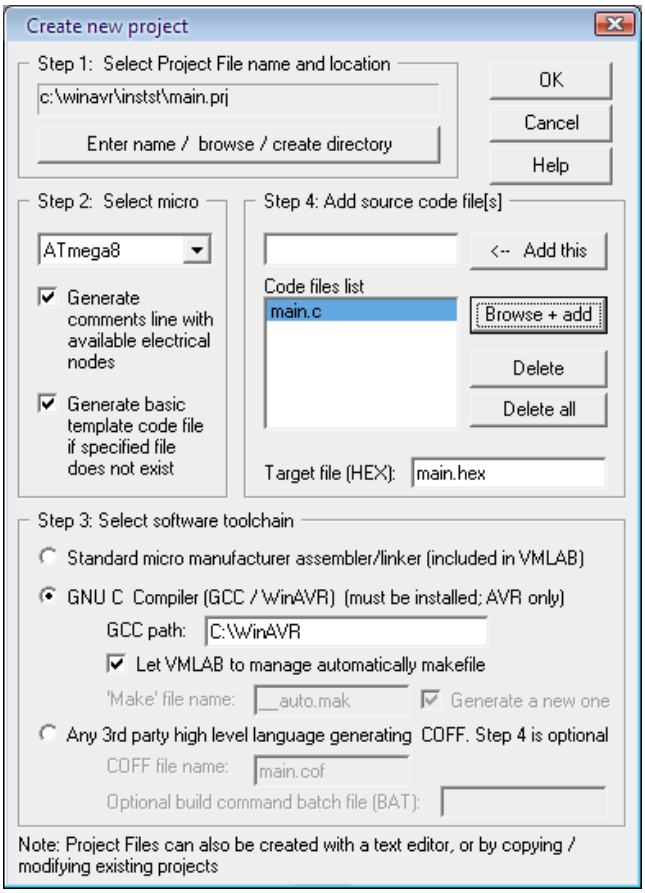
- uruchamia symulację (Debug > Start debugging)
- zakańcza symulację
- wykonuje program bez nadzoru, aż do momentu wstrzymania lub breakpointa.
- tymczasowe wstrzymanie wykonywania programu.
- restart symulacji
- pokazuje aktualną pozycję (jeżeli nie jest widoczna)
- wykonuje jeden krok (jedną linie programu) ze wskakiwaniem do funkcji
- wykonuje jeden krok nie wskakując do funkcji
- wyskakuje z aktualnej funkcji
- uruchamia program, aż do momentu przejścia do pozycji wskazanej kursorem
- uruchamia program pokazując każdy krok
- dodaje breakpoint, program zawsze się zatrzyma przed wykonaniem linii tak oznaczonej
- usuwa breakpoint
- dodaje zmienną wskazywaną przez kursor, której wartość można zobaczyć w oknie „Watch”

Dla przykładu: dodajemy do Watch zmienną *on* w funkcji *led*, ustawiamy breakpoint w pierwszej linii, uruchamiamy program. Po dotarciu do breakpointa wykonujemy program krok po kroku. W oknie IO View możemy podglądnąć wartości wszystkich rejestrów. W oknie procesor mamy czas symulacji oraz aktualny takt zegara.

7. Instalacja programiu VMLAB

Strona główna: http://www.amctools.com/vmlab.htm
Do pobrania z: http://www.amctools.com/download.htm

8. Nowy projekt



Klikamy Project > New project.
Step 1:
Zapisujemy projekt w naszym katalogu
Step 2:
Wybieramy mikrokontroler
Step 3:
GNU C Compiler
Wskazujemy katalog, gdzie zainstalowaliśmy WinAVR
Zaznaczamy Let VMLAB to ma... itd.
Step 4:
Klikamy Browse + add i wybieramy nasz main.c
Target file: main.hex

9. Symulacja

Musimy skompilować nasz projekt: Project>Build
W menu Run oraz na pasku narzędzi mamy podobne narzędzia jak w AVR Studio do sterowaniem działania programu. Breakpointy dodaje się przez kliknięcie kwadracika obok linii. W menu view możemy otworzyć kilka okiem przydatnych podczas debugowania.

10. Urządzenia zewnętrzne

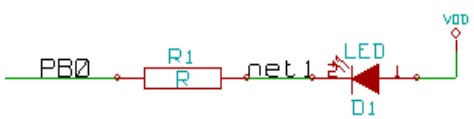
Ten symulator ma taką zaletę, że można podłączać wirtualne urządzenia do mikrokontrolera. Wpisuje się je w pliku projektu (na końcu pliku musi być przynajmniej jedna pusta linia). Porty oznaczone są przez PB0-PB7, PC0-PC6, itd. Przykłady:

Oscyloskop (wykres napięcia na PB0):

```
.PLOT V(PB0)
```

Dioda LED (podłączona przez rezystor do PB0):

```
D1 VDD net1
R1 net1 PB0 100
```



Dioda D1 jest widoczna w oknie „View>Control Panel”. W pliku podłączamy ją do VDD i do nowego przewodu net1. Rezystor R1 (100 ohm) został podłączony do PB0 oraz net1.

Przycisk (podłączony do PB1 oraz GND):

```
K0 PB1 GND
```

Inne urządzenia i przykłady można znaleźć w „Help>Contensts>Hardware Componets”

11. Programowanie

Kopiujemy i konfigurujemy Makefile (ze strony lub z katalogu WinAVR\sample) tak jak to było na warsztatach. Wpisujemy „make clean”, aby upewnić się, że nie ma żadnych śmieci. Następnie „make”, a na koniec „make program”.

Aby programowanie się powiodło, w systemie musi być zainstalowany filtr dla libusb. Do pobrania z http://sourceforge.net/project/showfiles.php?group_id=78138, plik o nazwie „libusb-win32-filter-bin-0.1.12.1.exe”.