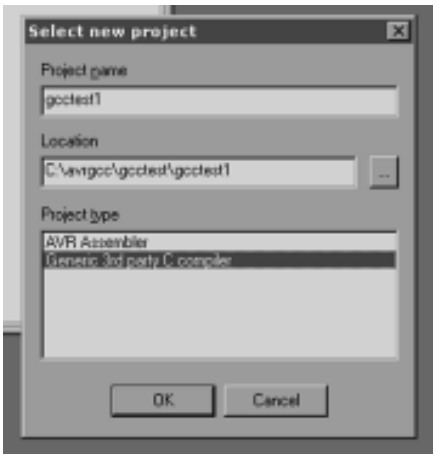
Bezpłatny kompilator C dla mikrokontrolerów AVR

Integracja GCC z AVR Studio

Popularność kompilatora języka C, który jest dostępny na stronie internetowej AVR Freaks, ciągle rośnie. Dzieje się tak z dwóch powodów. Po pierwsze, mikrokontrolery AVR coraz częściej pojawiają się w różnych aplikacjach. Po drugie, nieczęsto zdarza się taka gratka, że tak wartościowe narzędzie udostępniane jest za darmo.

Kompilator języka C dla AVR jest wynikiem jednego z tak zwanych projektów *open source*. Jest to po prostu program przygotowany przez nieformalny zespół programistów. Postać źródłową programu kompilatora można pobrać z Internetu i przystąpić do zespołu, jeśli tylko ma się coś wartościowego do dodania. GCC został napisany jako kompilator mogący pracować zarówno w środowisku DOS, jak i Windows. Jego najnowsza wersja wykorzystuje 32-bitowe biblioteki funkcji i przeznaczona jest do pracy w systemie Windows. Można ją znaleźć na stronie AVR Freaks pod adresem *http://www.avrfreaks.net/AVRGCC/index.php*.



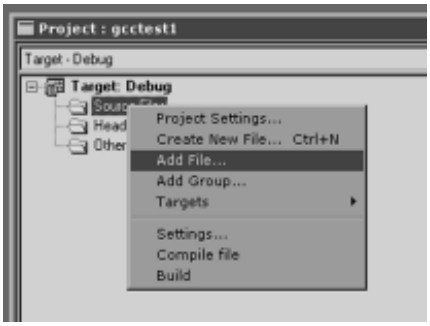
Rys 1

Do czego jest potrzebne AVR Studio?

Inaczej niż w przypadku kompletnego środowiska służącego do przygotowania programów, GCC jest tylko kompilatorem. Nie ma wbudowanego edytora i symulatora. Edycję można przeprowadzić za pomocą dowolnego edytora tekstowego. Najlepiej jest, jeżeli do tego celu używa się AVR Studio firmy Atmel. Obecnie dostępna jest już wersja 4.05 tego doskonałego narzędzia, jednak nie jest ona najlepsza do przygotowywania programów w języku C. Nie rozróżnia bowiem składni tego języka. Do współpracy z GCC najlepiej jest użyć wersji 3.55 dostępnej na wielu stronach internetowych, między innymi na AVR Freaks pod adresem *http://www.avrfreaks.net/Tools/showtools?ToolId=104*.

Z własnej praktyki wiem, że połączone ze sobą GCC i AVR Studio tworzą bardzo wartościowe narzędzie, za które normalnie trzeba by było słono zapłacić. Możemy je mieć za darmo.

Edytor tekstowy, w który wyposażone jest AVR Studio, przystosowano do składni języka C oraz asemblera. Rozróżnia on słowa kluczowe, zmienne i komentarze, oznaczając je innym kolorem lub przez zmianę czcionki. Dodatkowo, do najnowszej wersji GCC dołączony został program (był również we wcześniejszych wersjach, ale jako odrębny, zewnętrzny) tworzący zbiór przeznaczony dla programowego symulatora wbudowanego w AVR Studio. Dzięki temu możliwa jest symulacja działania programu i jego uruchomienie, zanim zostanie zapisany w pamięci mikrokontrolera.



Rys. 2

Pamiętam swoje wrażenie, gdy rok temu po raz pierwszy skorzystałem z kompilatora GCC. Wydawał mi się on wtedy czymś zupełnie nie do użytku. Obecnie jestem zdumiony jego możliwościami i całkowicie zmieniłem swoją opinię o nim.

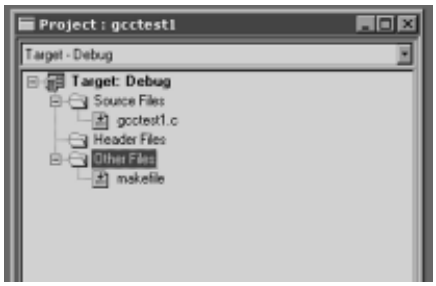
Instalacja

Po pobraniu z witryny AVR Freaks (GCC ma rozmiar około 11MB, AVR Studio około 4MB) lub skorzystania z płyty CD-EP10/2002B, należy zainstalować obydwa programy. Instalacja nie nastręcza żadnych problemów. AVR Studio to typowy program systemu Windows - jego wpis pojawi się w menu *Start* jako *ATMEL AVR Tools*.

Inaczej jest z GCC. Jak wcześniej wspomniałem, jest to tak zwany kompilator zewnętrzny i chociaż jego wpisy pojawią się w menu *Start*, to trudno będzie go wywołać z tego menu.

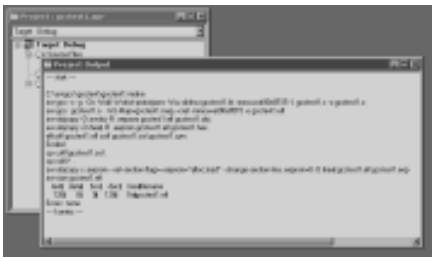
Konfiguracja

Po zainstalowaniu należy skonfigurować AVR Studio tak, aby można było pisać i uruchamiać programy w języku C. Po pierwsze, musimy na dysku komputera założyć katalog (folder), w którym przechowywane będą programy napisane w GCC. Ja używam katalogu *C:\SRC*, w którym umieszczam wszystkie programy źródłowe w podkatalogach dla danego kompilatora, aby przy tworzeniu kopii zapasowej na dysku archiwalnym łatwo było wszystkie te zbiory skopiować, kopiując tylko jeden katalog. Programy źródłowe napisane w GCC zapisują w katalogu *C:\SRC\AVR-GCC*, ale każdy użytkownik może utworzyć własny o dowolnej

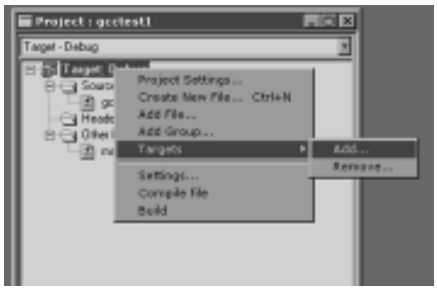


Rys. 3

Rys. 4



Rys. 5



Rys. 6

nazwie. Dobrze jest dla każdego projektu utworzyć oddzielny podkatalog.

W celu wyjaśnienia przebiegu kompilacji skorzystajmy z przykładowego programu dostarczanego wraz z GCC. Znajdziemy go na dysku twardym w katalogu *C:\AVRGCC\GCCTEST\GCCTEST1* pod nazwą *gcctest1.c*.

Tworzenie nowego zbioru projektu

Aby przeprowadzić kompilację, należy uruchomić AVR Studio. Z menu *Project* wybrać opcję *New* (**rys. 1**), wpisać nazwę projektu oraz wskazać katalog, w którym znajdą się pliki źródłowe oraz wynikowe utworzone przez kompilator. Następnie należy zaznaczyć *Generic 3rd party C compiler* i nacisnąć OK, po czym zapisać projekt na dysku wybierając *Project - Save*.

Dodawanie zbiorów do projektu

Należy ustawić kursor myszy na *Source Files* w okienku *Project*, nacisnąć prawy klawisz myszy i wybrać *Add File* (**rys. 2**), po czym wskazać zbiór *gcctest1.c*. Po dodaniu można otworzyć ten zbiór, klikając dwukrotnie na jego nazwie lub wybierając polecenie *Open*.

Nowy projekt wymaga dodania zbioru konfiguracyjnego programu *Make*. Zbiór o nazwie *makefile* znajdziesz w tym samym katalogu. Należy postępować z nim podobnie jak poprzednio: za pomocą kursora wskazać w okienku *Project* linię *Other Files* (**rys. 3**) i nacisnąć prawy klawisz myszy. Następnie z okienka menu należy wybrać *Add File* i wskazać zbiór o nazwie *makefile*. Wymaga on kilku słów wyjaśnienia.

Na **list. 1** zamieszczono przykład prostego zbioru konfiguracyjnego, który jest na tyle uniwersalny, że prawdopodobnie jedynymi liniami, które będzie trzeba czasami zmienić są:

- Typ mikrokontrolera, czyli linia *MCU = at90s8515*. Oczywiście należy tu wpisać typ używanego w projekcie mikrokontrolera.
- Linia *TRG = gcctest1*. Należy wpisać nazwę zbioru źródłowego zawierającego program w języku C (bez rozszerzenia). W połączeniu z następną linią *SRC=$(TRG).c* oznacza, że zarówno zbiór źródłowy, jak i wynikowe będą się różniły pomiędzy sobą tylko typem (rozszerzeniem nazwy).

List. 1.

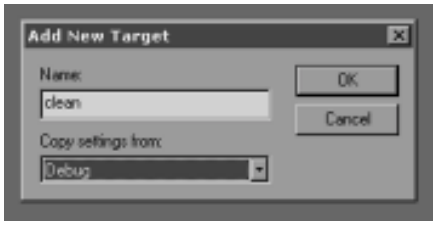
```
# Simple Makefile Volker Oth (c) 1999
# edited by AVRfreaks.net nov.2001

############ change these lines according to your project ##################

#tu wstaw nazwę mikrokontrolera (at90s8515, at90s8535, attiny22, atmega603 etc.)
   MCU = at90s8515

#tu wpisz nazwę zbioru wynikowego kompilacji (bez rozszerzenia!)
   TRG = gcctest1

#tu podaj nazwę zbioru źródłowego, tekstu programu w języku C
   SRC = $(TRG).c

#dodatkowe zbiory źródłowe w języku asembler
   ASRC    =

#dodatkowe zbiory bibliotek
   LIB =

#dodatkowe zbiory dołączane do zbioru źródłowego do kompilacji
   INC =

#parametry dla kompilatora asembler
   ASFLAGS = -Wa, -gstabs

#parametry dla kompilatora GCC
   CPFLAGS= -g  -Os -Wall -Wstrict-prototypes -Wa,-ahlms=$(<:.c=.lst)

#parametry linkera
   LDFLAGS = -Wl,-Map=$(TRG).map,-cref


############ tej linii przypuszczalnie nigdy nie będziesz zmieniał #############
include $(AVR)/avrfreaks/avr_make

###### tu dodawane są zależności pomiędzy plikami ####################
$(TRG).o: $(TRG).c
```

Kompilowanie programu źródłowego projektu

Jak wcześniej wspomniano, kompilator GCC jest kompilatorem zewnętrznym. Uruchamia się go z poziomu linii komend, podając jako parametr wywołania nazwę pliku tekstowego zawierającego program źródłowy. W taki właśnie sposób kompilator zostanie uruchomiony przez AVR Studio. AVR Studio umożliwia uruchomienie tylko jednego programu. Niestety, GCC wymaga wydania wielu komend przed jego uruchomieniem. Nasuwa się bardzo proste rozwiązanie tego problemu - należy utworzyć zbiór wsadowy komend, tak zwany *batch file* (**.bat*).

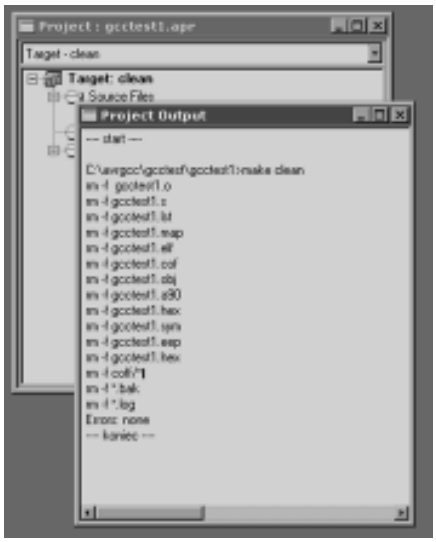
W tym celu w dowolnym edytorze tekstowym, na przykład Notepadzie, należy wpisać następującą listę komend:

```
@echo *** start kompilacji ***
@set AVR=c:\AVRGCC
@set CC=avr-gcc
@set PATH=c:\AVRGCC\bin
make %1
@echo *** koniec kompilacji ***
```



Rys. 7

Rys. 8



Rys. 9

Następnie należy zapisać ten zbiór w dowolnej lokalizacji, ale znajdującej się na liście ścieżek przeszukiwanych przez system Windows (komenda *PATH* w *autoexec.bat*). Ja umieściłem go w katalogu *c:\WINDOWS*. Należy nazwać go na przykład *gcc_cmp.bat*. i powrócić do AVR Studio i otwartego okna projektu, po czym wskazać *Target: Debug*, nacisnąć prawy klawisz myszy i z menu wybrać polecenie *Settings*.

Najpierw w wyświetlonym oknie (**rys. 4**) należy ustawić wartości pól otoczonych ramką *Linker/Build Stage Settings*. Należy wyłączyć opcję *Run compile on each* i zaznaczyć *Run linker/bulid*. W okienku *Command line* wpisz *gcc_cmp.bat*. W polach otoczonych ramką *Run Stage Settings* należy pozostawić wpis *Errors: none* oraz *obj*, zaznaczyć opcję *Run code* i wybrać OK.

Teraz kompilator jest gotowy do skompilowania projektu. Wskazując kursorem na *Target: debug* należy nacisnąć prawy klawisz myszy, a następnie wybrać *Build* (**rys. 5**).

Uwagi na temat kompilacji

Plik o nazwie *makefile* zawiera szereg poleceń kompilatora. Między innymi jest w nim linia:

```
#compiler flags
 CPFLAGS = -g -Os -Wall -Wstrict-
prototypes -Wa,-ahlms=$(<:.c=.lst)
```

Rys. 10

Opcja *-O* określa, jak będzie optymalizowany kod wynikowy. Domyślnie ustawiona jest na *-Os*, co oznacza, że kod będzie optymalny pod względem rozmiaru. Zazwyczaj jest to nastawa zadowalająca przeciętnego programistę. Jednak można włączyć również inne sposoby oraz poziomy optymalizacji. Szczegółowe informacje na ten temat można znaleźć na stronie internetowej AVR Freaks. Nie chcę ich tutaj przytaczać, ponieważ parametry te mogą się zmienić wraz z wersją kompilatora.

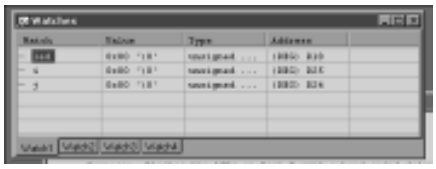
Kompilator tworzy na dysku kilka plików wynikowych. Nie są one usuwane po utworzeniu zbioru HEX. Doprowadza to do sytuacji, w której podczas następnego wywołania kompilatora, program *Make* może stwierdzić, że nie jest konieczna ponowna kompilacja pliku źródłowego. Autorzy zalecają więc przed ponownym uruchomieniem kompilatora usunąć z dysku wszystkie zbiory wynikowe. Można to zrobić w bardzo prosty sposób: jako parametr wywołania *gcc_cmp.bat* podać polecenie *clean*. W tym celu również możemy posłużyć się opcją *Target*.

Należy ustawić kursor myszy na *Target: Debug*. Z menu wybrać *Targets*, a następnie *Add...* (**rys. 6**). Pojawi się okienko *Add New Target* (**rys. 7**). W polu nazwy należy wpisać na przykład *clean*, a w polu *Copy settings from* wybrać *Debug*. To spowoduje, że nowo utworzony zbiór nastaw odziedziczy je po *Debug*. Pierwsza linia w okienku *Project* to lista nastaw. Należy wybrać z niej *Target: clean*. Następnie wskazać poniżej *Target: clean* kursorem i naciśnąć prawy klawisz myszy - z menu wybrać *Settings*. Można zauważyć, że wszystkie właściwości poprzednio wpisywane w *Target: Debug* zostały skopiowane. Jedyne, co należy zrobić, to w okienku komend za wywołaniem *gcc_cmp.bat* dopisać plecenie *clean* (**rys. 8**), następnie wybrać OK, aby zapamiętać nastawy.

Uruchomienie polecenia *Build* dla *Target: clean* (po naciśnięciu prawego klawisza) usunie wszystkie zbiory pochodzące z poprzedniej kompilacji (**rys. 9**).

Symulacja pracy programu

Wcześniejsze wersje kompilatora GCC i AVR Studio wymagały wykonania dosyć skomplikowanej procedury przy uruchamianiu symulacji programu. Osobno otwierało się zbiory wynikowe, osobno zbiór źródłowy w języku C. Teraz trzeba tylko wybrać z menu AVR Studio *File* oraz *Open* i otworzyć zbiór o nazwie *gcctest1.cof*. Jest on tworzony w trakcie kompilacji.



Rys. 11

Po tej operacji pojawi się okienko, w którym konieczne będzie wybranie typu mikrokontrolera oraz wpisanie parametrów środowiska (**rys. 10**). Po naciśnięciu OK, symulator jest gotów do pracy.

Symulator AVR Studio oferuje nam wiele różnych możliwości. Ich opis z nadmiarem przekraczałby ramy niniejszego artykułu. Wspomnę więc może w skrócie tylko o tych najważniejszych.

Kombinacja klawiszy *SHIFT+F5* działa tak jak zerowanie. Po nim mamy do wyboru różne tryby wykonywania programu, łącznie z pracą w trybie automatycznym (tak zwanym animowanym). Naciśnięcie klawisza *F11* powoduje wykonanie pojedynczej linii programu, *F10* pominięcie na przykład wywołania procedury, a *F5* uruchomienie programu tak, jak gdyby był uruchamiany w mikrokontrolerze. Program będzie wykonywany do napotkania pułapki oznaczonej przez *F9* lub jego zakończenia w normalny sposób. Okienko o nazwie *Watches* (**rys. 11**) umożliwia podgląd i modyfikację zmiennych. Można zobaczyć w jakim rejestrze dana zmienna się znajduje i jaką ma wartość. Można również ją zmodyfikować (w zależności od potrzeb uruchamianej aplikacji), po prostu wskazując w okienku *Watches* wartość zmiennej i po pojawieniu się kursora wpisując nową.

Podsumowanie

Mam nadzieję, że ten artykuł ułatwi zainteresowanym pierwsze kroki z kompilatorem GCC oraz umożliwi im tworzenie własnych aplikacji. Reszta to już tylko kwestia praktyki i doświadczenia. Polecam również lekturę materiałów dostępnych na stronie internetowej AVR Freaks (*http://www.avrfreaks.net/*) oraz grupy dyskusyjne związane z GCC (również można je znaleźć na AVR Freaks). Jest to kopalnia wiedzy na tematy związane z programowaniem mikrokontrolerów AVR.

Jacek Bogusz, AVT
jacek.bogusz@ep.com.pl

Dodatkowe informacje

Oprogramowanie opisane w artykule (tzn. kompilator GCC i AVR Studio 3.55) są dostępne w Internecie na stronie www.avrfreaks.com, a także na płycie CD-EP10/2002B.