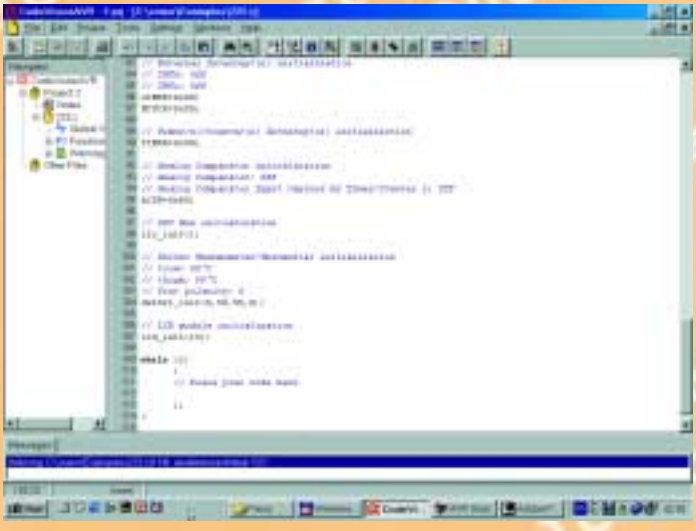
CodeVisionAVR C

Kompilator języka C dla procesorów AVR

W artykule został zaprezentowany dość ciekawy kompilator języka C dla procesorów rodziny AVR. Mimo dosyć egzotycznego pochodzenia (Rumunia), jest to kompilator zasługujący na uwagę projektantów systemów mikroprocesorowych. Może on być szczególnie atrakcyjny dla tych, którym potrzebne jest sprawne narzędzie do pisania programów w języku C, natomiast nie chcą przy tym wydać fortuny na powszechnie znane systemy takich producentów, jak Keil czy IAR. W artykule postaramy się zatem odpowiedzieć na pytanie, czy za stosunkowo niewielkie pieniądze można zostać posiadaczem pełnowartościowego narzędzia do pisania programów w języku C.

CodeVisionAVR C to narzędzie programistyczne pracujące w środowisku Windows, umożliwiające tworzenie kodu w języku C z rozszerzeniem uwzględniającym specyfikę procesorów AVR firmy Atmel. Okno programu jest podzielone na trzy części (**rys. 1**). Z lewej strony znajduje się okno nawigatora. W oknie tym, w postaci drzewa, są wyświetlane nazwy plików wchodzących w skład projektu, nazwy zmiennych globalnych i funkcji deklarowanych w poszczególnych plikach źródłowych oraz błędy i ostrzeżenia kompilacji. Po naciśnięciu nazwy pliku, w oknie głównym (po prawej stronie) otwiera się okno edycji kodu źródłowego. Kod źródłowy jest wyświetlany w różnych kolorach, w zależności od znaczenia tekstu, co znacznie zwiększa czytelność programu. Niestety, w trakcie edycji programu nie ma możliwości korzystania z systemu pomocy w sposób zależny od kontekstu. Dodanie takiej opcji, spotykanej w wielu kompilatorach, z pewnością podniosłoby wygodę pracy w *CodeVisionAVR C*. Na dole ekranu jest wyświetlane okno komunikatów, w którym kompilator umieszcza informacje o błędach i ostrzeżeniach kompilacji programu.

Jednym z ważniejszych elementów zintegrowanego środowiska programisty jest debugger, czyli system wspomagający uruchomianie i testowanie pisanych programów. Pierwszą rzeczą, jaka rzuca się w oczy po uruchomieniu *CodeVisionAVR C* jest brak jakiegokolwiek debuggera. Na szczęście producent umożliwił współpracę ze znakomitym debuggerem *AVR Studio* firmy Atmel, poprzez generowanie plików w formacie COFF. Niestety taka praca nie jest zbyt wygodna, wymagane jest ciągłe przełączanie między programami, co przy dłuższej pracy wprowadza dekoncentrujący chaos.

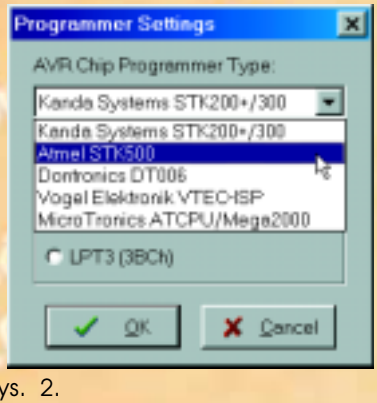


Rys. 1.

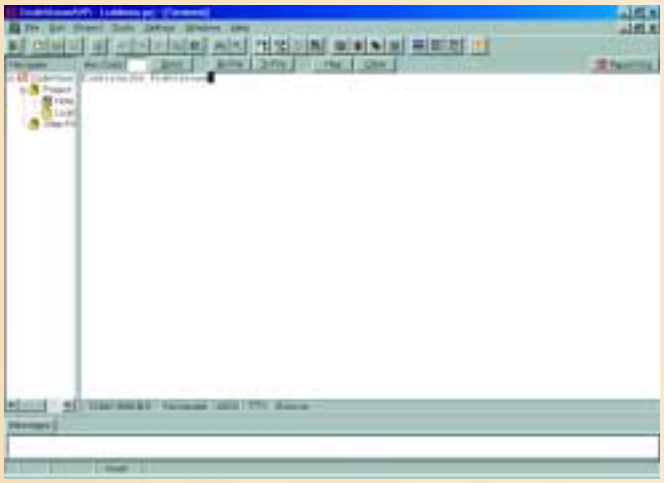
Za pomocą *CodeVisionAVR C* możliwe jest sterowanie jednym z pięciu programatorów mikroprocesorów (**rys. 2**). Jest to bardzo wygodne w końcowym etapie pisania programu, gdy jest on już testowany w docelowym układzie elektronicznym. Dzięki temu zaprogramowanie układu, po wprowadzonych zmianach, jest bardzo szybkie. Na etapie uruchamiania układu bardzo wygodne niekiedy okazuje się korzystanie z wbudowanego w *CodeVisionAVR C* terminala (**rys. 3**), dzięki któremu można na przykład śledzić komunikaty odbierane przez port szeregowy komputera.

Na szczególną uwagę zasługuje *CodeWizardAVR* (**rys. 4**), czyli generator kodu źródłowego. Dzięki tej aplikacji można łatwo i szybko wygenerować kod źródłowy inicjujący układy peryferyjne wbudowane w wybrany mikroprocesor (porty we/wy, timery, liczniki, układ watchdoga, przetworniki A/C i inne) oraz elementy dołączone do wybranych portów sterownika (wyświetlacz LCD, interfejs szeregowy, I^2C i inne). Listę mikroprocesorów obsługiwanych przez *CodeVisionAVR C* zawarto w **tab. 1**.

Na **rys. 5** pokazano okno ustawiania opcji kompilacji programu. Można wybrać typ mikroprocesora docelowego dla realizowanego projektu, rodzaj optymalizacji stosowanej



Rys. 2.



Rys. 3.

przez kompilator (*size/speed*), przydzielić pamięć na potrzeby stosu itd. Jak widać do wyboru są tylko dwa modele pamięci: *tiny* i *small*. W modelu *tiny* pamięć SRAM jest adresowana za pomocą 8 bitów (w takim przypadku możliwy jest dostęp tylko do pierwszych 256 bajtów pamięci SRAM), natomiast w modelu *small* za pomocą 16 bitów. Adresowanie pamięci Flash i EEPROM, niezależnie od wybranego modelu pamięci, jest zawsze 16-bitowe. Deklarując zmienną, która ma być umieszczona w pamięci Flash, należy poprzedzić jej nazwę słowem kluczowym `flash` (np. `int flash liczba`). Identyczna sytuacja jest ze zmiennymi umieszczonymi w pamięci EEPROM (np. `char eeprom tekst []="Elektronika Praktyczna"`). Zmienne, których nazwy nie poprzedza żadne słowo kluczowe są umieszczane w pamięci SRAM. Ze względu na specyfikę procesorów AVR, język C rozszerzono między innymi o zmienne bitowe, bitowy dostęp do rejestrów oraz obsługę przerwań sprzętowych. *CodeVisionAVR C* umożliwia mieszanie kodu w języku C z fragmentami napisanymi w asemblerze. Dzięki temu możliwa jest ścisła kontrola działania programu w jego krytycznych czasowo fragmentach. Pewne zastrzeżenia można mieć do sposobu weryfikacji kodu źródłowego przez kompilator. Nie znajduje on wielu błędów związanych z przekroczeniem dopuszczalnych wartości adresów. Błędy te są wychwytywane podczas kompilacji pośredniego pliku w asemblerze i są sygnalizowane lakonicznym komunikatem bez wskazania miejsca powstania błędu. Wychwycenie takiego błędu jest niekiedy trudne, a niemożność zlokalizowania błędów tego rodzaju przez *CodeVisionAVR C* wynika z zastosowania asemblera pracującego niezależnie od kompilatora języka C. Kompilacja kończona jest podsumowaniem (**rys. 6**), w którym są zawarte informacje m.in. o procesorze docelowym, modelu pamięci, wybranym rodzaju optymalizacji, liczbie błędów i ostrzeżeń, obszarach pamięci przeznaczonych na stos.

W ramach *CodeVisionAVR C* producent dostarczył wiele funkcji bibliotecznych (nie wszystkie zostały zamieszczone w wersji ewaluacyjnej). Zostały one pogrupowane w następujące biblioteki:

- bibliotekę funkcji znakowych, np. `unsigned char` **`isascii`**`(char c)`, `char` **`tolower`**`(char c)`;
- bibliotekę standardowych funkcji wejścia/wyjścia, np. `char` **`getchar`**`(void)`, `void` **`printf`**`(char flash *fmtstr [, arg1, arg2,...])`;
- bibliotekę funkcji standardowych, np. `int` **`atoi`**`(char *str)`, `int` **`rand`**`(void)`;
- bibliotekę funkcji matematycznych, np. `unsigned int` **`abs`**`(int x)`, `float` **`log`**`(float x)`;
- bibliotekę funkcji tekstowych, np. `char` **`*strcat`**`(char *str1, char *str2)`, `signed char` **`strcmp`**`(char *str1, char *str2)`;
- bibliotekę funkcji konwertujących kod BCD, np. `unsigned char` **`bcd2bin`**`(unsigned char n)`;
- bibliotekę funkcji konwertujących kod Gray'a, np. `unsigned char` **`gray2bin`**`(unsigned int n)`;

Tab. 1. Lista układów obsługiwanych przez CodeVisionAVR C:

- ATtiny22
- AT90S2313
- AT90S2323/2343
- AT90S2333/4433
- AT90S4414/8515
- AT90S4434/8535
- AT90S8534
- ATmega603/103
- ATmega128
- ATmega161
- ATmega163
- ATmega323 (ATmega32)
- ATmega8/16
- FPSLIC AT94K05/10/20/40

- bibliotekę funkcji dostępu do pamięci, np. `void` **`pokeb`**`(unsigned int addr, unsigned char data)`, `unsigned char` **`peekb`**`(unsigned int addr)`;
- bibliotekę funkcji opóźnień czasowych, np. `void` **`delay_us`**`(unsigned int n)`, `void` **`delay_ms`**`(unsigned int n)`;
- bibliotekę funkcji do obsługi alfanumerycznych wyświetlaczy LCD zgodnych ze standardem Hitachi HD44780, np. `void` **`_lcd_write_data`**`(unsigned char data)`;
- bibliotekę funkcji do obsługi interfejsu I²C, np. `unsigned char` **`I2C_read`**`(unsigned char ack)`, `void` **`I2C_stop`**`(void)`;
- bibliotekę funkcji do obsługi czujnika temperatury LM75, np. `int` **`lm75_temperature_10`**`(unsigned char chip)`;
- bibliotekę funkcji do obsługi czujnika temperatury/termostatu DS1621, np. `unsigned char` **`ds1621_get_status`**`(unsigned char chip)`;
- bibliotekę funkcji do obsługi zegara czasu rzeczywistego PCF8563, PCF8583, DS1302 i DS1307 np. `unsigned char` **`rtc_read`**`(unsigned char address)`;
- bibliotekę funkcji do obsługi protokołu 1 Wire, np. `unsigned char` **`w1_write`**`(unsigned char data)`;

Rys. 4.

Rys. 5.

- bibliotekę funkcji do obsługi czujników temperatury DS1820 i DS1821, np. `int` **`ds1820_temperature_10`**`(unsigned char *addr);`
- bibliotekę funkcji do obsługi zewnętrznych pamięci EEPROM DS2430 i DS2433, np. `unsigned char` **`ds2430_write_appreg_block`**`(unsigned char *romcode, unsigned char *source,unsigned char addr,unsigned char size);`
- bibliotekę funkcji do obsługi SPI, np. `unsigned char` **`spi`**`(unsigned char data);`
- bibliotekę funkcji zarządzających poborem energii, np. `void` **`idle`**`(void)`, `void` **`powersave`**`(void)`.

W poznaniu zawartości tych bibliotek pomaga plik pomocy *CodeVisionAVR C*. Zarówno elementy języka C, jak i funkcje biblioteczne są tam wyczerpująco zaprezentowane.

Do głównych zalet *CodeVisionAVR C* należy zaliczyć atrakcyjną cenę oraz biblioteki zawierające funkcje pozwalające małym nakładem pracy uzyskać efektowny program. Dużą

Rys. 6.

niedogodnością jest konieczność stosowania debuggera niezależnego producenta (na szczęście *AVR Studio* jest programem darmowym), ale obecnie takie praktyki są często stosowane - identyczne rozwiązanie można spotkać w kompilatorze języka C ImageCraft *ICCAVR V6*, który oferując podobne możliwości jest jednak trochę droższy niż *CodeVisionAVR C*. Odpowiadając na pytanie postawione na wstępie artykułu można stwierdzić, że za stosunkowo niewielką kwotę można zostać posiadaczem kompilatora języka C dla procesorów AVR. Porównując jednak *CodeVisionAVR C* z systemami z górnej półki (IAR, Keil) należy dostrzec przepaść dzielącą te dwa światy. Na korzyść tych drugich przemawiają przede wszystkim całkowicie zintegrowane środowisko programisty (m.in. wbudowany debugger), możliwość samodzielnego rozbudowywania bazy obsługiwanych układów, znacznie wyższy komfort pracy programisty w środowisku zintegrowanym, praca z systemem pomocy ułatwiającym pisanie programów, bardzo rozbudowana dokumentacja (często zawierająca noty katalogowe procesorów), jednorodne środowisko pracy dla różnych rodzin procesorów, wspomaganie budowania systemów czasu rzeczywistego itd. Czym zatem, oprócz atrakcyjnej ceny, mogą przyciągnąć programistów takie produkty jak *CodeVisionAVR C*? Przede wszystkim łatwością pisania programów do najczęściej spotykanych zastosowań. Dołączone do *CodeVisionAVR C* biblioteki pozwalają szybko i łatwo tworzyć programy wykorzystujące na przykład protokoły I^2C, RS-232C, 1-Wire oraz ułatwiają, dzięki odpowiednim funkcjom, stosowanie w projektach popularnych układów scalonych (np. DS1820).

Wersja ewaluacyjna, w porównaniu do pełnej wersji, jest pozbawiona niektórych bibliotek m.in. obsługi wyświetlaczy LCD 4x40, układów z interfejsem 1-Wire (PCF8563, PCF8583, DS1302, DS1307) oraz czujników temperatury (DS1820/DS1822). Dodatkowo zostało wprowadzone ograniczenie wielkości kodu wynikowego.

Paweł Zbysiński

Dodatkowe informacje

Więcej informacji można znaleźć na stronie internetowej producenta CodeVisionAVR C: http://infotech.ir.ro.