Tani kompilator firmy CCS

Kiedy po raz pierwszy zetknąłem się mikrokontrolerami PIC16xx firmy Microchip przeczytałem w dokumentacji firmowej, że mają one tylko 35 instrukcji. Nieźle - pomyślałem - łatwo się będzie tego nauczyć. Ale rzeczywistość okazała się troszkę inna. Po pierwszych próbach praktycznego wykorzystania instrukcji BTFSC, BTFSS, INCFSZ itp. zacząłem z rozrzewnieniem wspominać listę rozkazów Z-80, czy nawet 8051.

Takie były początki. Potem było już znacznie lepiej, ale zacząłem poszukiwać narzędzi pozwalających znacznie szybciej tworzyć programy. Po zbadaniu zasobów Internetu okazało się, że istnieje kilka firm oferujących kompilatory języka C dla mikrokontrolerów Microchipa. Najłatwiej jest znaleźć taki kompilator dla najbardziej popularnego, 14-bitowego rdzenia (rodziny 16xx), ale oferowane są też kompilatory dla rodzin 12xx, 17xx i 18xx. Jedną z firm, oferujących kompilatory C dla wszystkich rodzin mikrokontrolerów PIC (z wyjątkiem PIC 17xx), jest *Custom Computer Services*, bardziej znana jako CCS.

W ofercie tej firmy znajdują się trzy podstawowe (**tab. 1**) kompilatory, oznaczone jako:
- PCB dla rdzenia 12-bitowego PIC 12xx,
- PCM dla rdzenia 14-bitowego PIC16xx,
- PCH dla rdzenia 16-bitowego PIC18xx.

Każdy z tych kompilatorów występuje oddzielnie i trzeba je osobno zakupić.

Rys. 1.

Użytkownicy zespolonych środowisk projektowych, takich jak Bascom, czy kompilatory C firm Keil lub IAR, byliby najprawdopodobniej nieco zawiedzeni przy pierwszym zetknięciu z tymi trzema produktami CCS. Kompilatory w wersjach PCB, PCM i PCH są wywoływane przez ręcznie wpisywane komendy. Nie ma tu żadnego dedykowanego edytora, menedżera projektu, czy - rzecz już prawie standardowa - programowego symulatora wykonywanego programu. Jednak wymienione niedogodności nie dyskwalifikują tych kompilatorów.

Producenci oprogramowania narzędziowego dla mikrokontrolerów Microchipa są w tej komfortowej sytuacji, że mają do dyspozycji bardzo dobre i darmowo rozprowadzane środowisko projektowe MPLAB. Zapewnia ono możliwość edycji plików źródłowych, tworzenia własnych projektów oraz zawiera doskonały symulator programowy z możliwością krokowego wykonywania programu, podglądania wartości zmiennych, rejestrów i obszarów pamięci RAM, ROM i EEPROM użytkownika. Współpraca MPLAB z kompilatorami

Rys. 2.

Rys. 3.

Właściwości kompilatora PIC C firmy CCS:

- Wbudowane biblioteki obsługi RS-232, I²C, przetwornika A/D, bitowej obsługi I/O, generowania opóźnień, obsługi układów licznikowych, SSP, PSP, USB.
- Możliwość integracji z MPLAB-em oraz innymi edytorami i symulatorami umożliwiającymi debbuging kodu źródłowego. Format pliku HEX i pliku debug zapewnia kompatybilność z większością dostępnych programatorów i debuggerów.
- Funkcja *printf* umożliwia formatowanie i wyświetlanie w postaci dziesiętnej lub heksadecymalnej.
- Dzięki mechanizmowi wywoływania funkcji istnieje możliwość głębszego zagnieżdżania, niż wynika to z głębokości stosu sprzętowego.
- Dostępność źródłowych plików procedur do obsługi typowych LCD, klawiatur w postaci matryc, szeregowych EEPROM-ów serii 24XXX oraz 93XXX, zegarów czasu rzeczywistego DS2223 i NJU6355, szeregowych SRAM-ów, przetworników A/D, cyfrowych potencjometrów i innych.
- 1/8/32-bitowe zmienne stałoprzecinkowe oraz 32-bitowe zmienne zmiennoprzecinkowe.
- Instrukcje asemblerowe mogą być wstawiane *inline* oraz mogą odnosić się do wszystkich zmiennych.
- Stałe (również łańcuchy i tablice) są pamiętane w pamięci programu.
- Standardowy typ bitowy (*Short Int*) umożliwia generowanie bardzo wydajnego, bitowo zorientowanego kodu.
- #BIT i #BYTE pozwalają na zorientowanie zmiennych C na komórki pamięci o określonym adresie bezwzględnym. Umożliwia to obrazowanie rejestrów specjalnych poprzez zmienne C.

CCS musi być poprzedzona jego odpowiednią konfiguracją.

Po utworzeniu własnego projektu, z zakładki *Options->Development Mode* trzeba wybrać rodzaj procesora i narzędzie do symulacji (**rys. 1**). Następnie trzeba zainstalować kompilator, który później może być używany jako standardowo dla języka opisu projektu (**rys. 2**). W nowo utworzonym projekcie trzeba określić rodzaj kompilatora, ścieżki dostępu do plików nagłówkowych i bibliotecznych oraz typy plików źródłowych (**rys. 3**).

Klikając na nazwie pliku z rozszerzeniem *hex* w polu *Project Files*, a następnie na podświetlonym przycisku *Node Properties*, uzyskujemy dostęp do ustawień samego kompilatora (**rys. 4**). Można tu ustawić rodzaj używanego kompilatora i włączyć generowanie pliku *Tree*. W polu *Additional Command Line Options* można wpisać dowolne komendy kompilatora, szczegółowo opisane w dokumentacji firmowej.

Od tego momentu MPLAB jest skonfigurowany i można rozpocząć tworzenie własnego programu. Za jego pomocą można także symulować działanie tworzonego programu korzystając z wbudowanego programowego symulatora (**rys. 5**) lub sprzętowego emulatora, np. opisywanego już w EP zestawu MPLAB-IDE 2000.

Firma CCS ma w swojej ofercie również własne środowisko IDE-PCW. Działa ono w systemie operacyjnym Windows 95/98 (**rys. 6**) i umożliwia tworzenie własnych projektów oraz edycję plików źródłowych z możliwością włączenia kontroli syntaktycznej (wyświetlanie w różnych kolorach słów kluczowych i komentarzy). Standardowo, pakiet ma wbudowane kompilatory PCB i PCM, a za dodatkową opłatą można doinstalować kompilator PCH.

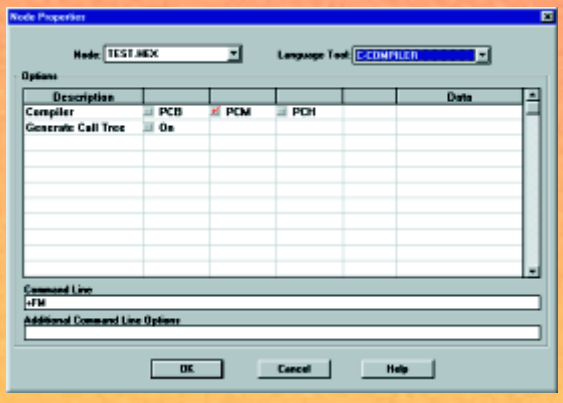
Kreator projektów ma wbudowaną funkcję automatycznego tworzenia pliku nagłówkowego o nazwie projektu. W pliku źródłowym automatycznie przygotowywane jest wywołanie szeregu funkcji bibliotecznych obsługujących peryferia mikrokontrolera. Pakiet PCW umożliwia również oglądanie danych katalogowych wybranego w projekcie mikrokontrolera i deasemblację plików z rozszerzeniem **.hex*. Z zakładki *Tools* można uruchomić narzędzie *Device Editor* umożliwiające edycję pliku bazy danych mikrokontrolerów *devices.dat*. Znajduje się tam też *Serial Port Monitor* (umożliwiający komunikację poprzez port szeregowy RS232 pomiędzy tworzoną aplikacją a komputerem), disasembler oraz programy umożliwiające porównywanie plików, wybór mikrokontrolera, szybkie przejście do pakietu MPLAB i zaprogramowanie mikrokontrolera. Ta ostatnia czynność wymaga oczywiście odpowiedniego programatora. Kompilator, w trakcie kompilacji pliku źródłowego (oprócz pliku wynikowego z rozszerzeniem **.hex*) generuje też kilka plików pomocniczych. Z zakładki *View* można otwierać i analizować plik listingu *c/asm*. Zawiera on listing programu źródłowego w C wraz z odpowiadającymi mu sekwencjami w asemblerze. Plik ten jest szczególnie przydatny w sytuacji, kiedy należy przeanalizować fragment jakiegoś programu.

Kompilator generuje też plik *Call Tree*, w którym pokazane są (w postaci drzewa) wszystkie funkcje wraz z ich wywołaniami. Dodatkowo zawarta jest tam informacja o użyciu przez funkcje pamięci programu i pamięci RAM. Z zakładki *View* można też oglądać plik wynikowy w postaci hexadecymalnej lub ASCII, plik MAP określający mapę zajętości pamięci RAM w ostatnio kompilowanym programie, plik z rozszerzeniem COD używanym przez symulator programowy, plik *Valid Fuses* informujący o stanie rejestru bezpieczników oraz pliki *Valid Interrupts* i *Status Line*.

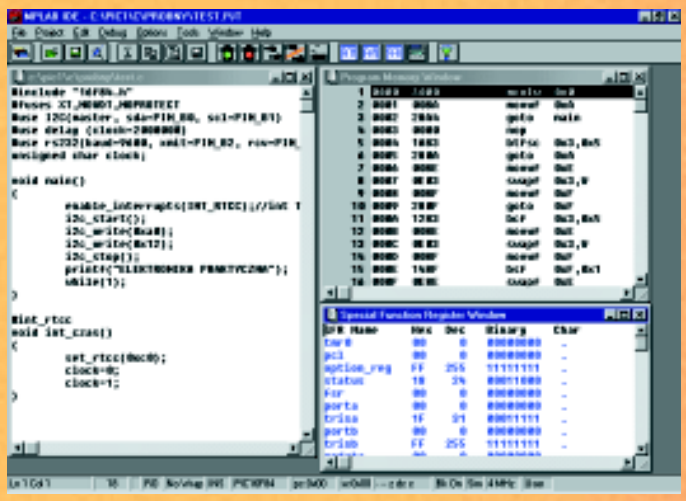
Dość przydatną może być funkcja *Data Sheet*, również uruchamiana z *View*. Umożliwia ona przeglądanie danych katalogowych w trakcie pisania programu. Jest to przydatna funkcja, szczególnie w momencie tworzenia aplikacji niewystarczająco poznanego mikrokontrolera.

Całość jest dość użytecznym środowiskiem projektowym, ale ma jedną dość istotną wadę - brak jest tam jakiegokolwiek symulatora programowego. W pewnym sensie lukę tę wypełnia odwołanie się do pakietu MPLAB. MPLAB uruchamia się wtedy z otwartym projektem przygotowanym z poziomu PCW i można korzystać wtedy z symulatora.

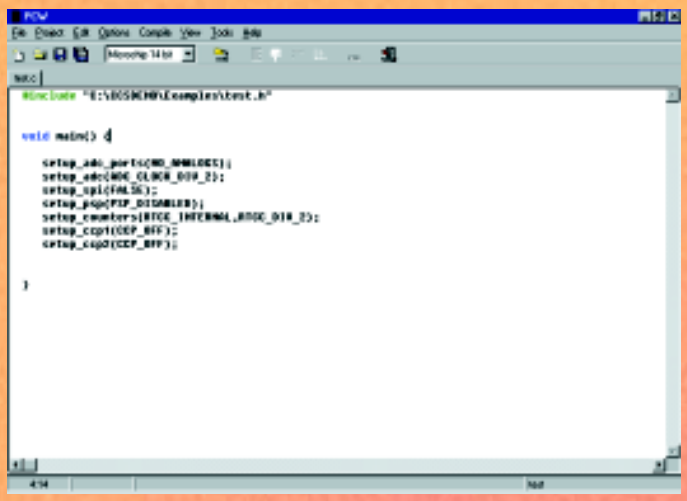
Większość producentów kompilatorów dla mikrokontrolerów deklaruje zgodność swoich produktów z normą ANSI. Norma ta określa przede wszystkim składnię deklaracji i definicji funkcji, cechy arytmetyki, możliwości preprocesora, definicję standardowej biblioteki itp. Znormalizowanie języka umożliwia w miarę proste przenoszenie oprogramowania. Z mikrokontrolerami sprawa nie jest tak prosta. Oprócz standardowych konstrukcji językowych, kompilatory dla mikrokontrolerów muszą być wyposażone w rozszerzenia pozwalające przede wszystkim na zaprogramowanie układów peryferyjnych. W przypadku kompilatora CCS rozszerzenia te mogą być realizowa-



Rys. 4.



Rys. 5.



Rys. 6.

ne poprzez komendy preprocesora i niestandardowe funkcje biblioteczne.

Preprocesor, oprócz standardowych komend typu *#IF*, *#ELSE*, *#LIST* itp., zawiera komendy pozwalające na sterowanie rozszerzeniami spotykanymi tylko w mikrokontrolerach PIC.

Komenda *#bit* pozwala na definiowanie zmiennej bitowej w dowolnym rejestrze. Rejestrem tym może być komórka SFR lub obszaru pamięci GPR. Sterowanie liniami portu może się odbywać poprzez ich zdefiniowanie, np. *#bit led=6.2*, gdzie 6 to adres rejestru SFR PORTB. Wyzerowanie tej linii wygląda bardzo prosto: *led = 0;*.

Bitowe flagi definiuje się tak samo. Obsługa przerwań jest zawsze związana z rozszerzeniem języka C. W przypadku kompilatora CCS funkcje obsługi przerwania są poprzedzone komendą *#INT_xx*, gdzie *xx* jest parametrem określającym przyczynę zaistnienia przerwania. Na przykład, komenda *#int_rtcc* poprzedzająca procedurę powoduje, że w przypadku wystąpienia przerwania od licznika T0 zostanie wywołana ta właśnie procedura.

Wielu Czytelników zapewne doceniło wbudowane w kompilator Bascom funkcje obsługi magistrali I²C, wyświetlacza LCD czy nawet niektórych układów scalonych. Do tej pory, producenci kompilatorów C niechętnie umieszczali takie funkcje w swoich pakietach projektowych. Przykładem może być opisywany na łamach EP bardzo dobry kompilator dla 8051 firmy Keil, który nie ma żadnej bibliotecznej funkcji wspomagającej obsługę chociażby magistrali I²C, za wyjątkiem standardowych funkcji znakowych np. *printf*. Zaczyna się to powoli zmieniać, czego przykładem jest opisywany tutaj kompilator.

Komendy preprocesora *#use delay clock*, *#use I²C*, *#use RS232* pozwalają na wykorzystanie funkcji bibliotecznych obsługujących magistralę I²C i programowy port szeregowy RS232. Szczególnie ten ostatni wydaje się przydatny, ponieważ wiele PIC-ów z rodziny 16xx nie posiada szeregowego portu sprzętowego.

Ściśle ze środowiskiem sprzętowym związane są komendy *#DEVICE* określające typ mikrokontrolera i *#FUSES* pozwalające programować łączniki (bezpieczniki) konfiguracyjne (typ oscylatora, protekcję odczytu itp.). Komenda *#ORG* pozwala na umieszczenie kodu programu w określonej przestrzeni pamięci programu.

Dołączana standardowo do kompilatora biblioteka zawiera, oprócz powszechnie spotykanych funkcji matematycznych i wejścia wyjścia, wiele dodatkowych funkcji pozwalających szybko i łatwo tworzyć programy obsługujące port szeregowy, magistralę I²C i SPI oraz *Parallel Slave Port* (PSP). Dość rozbudowane są funkcje pozwalające w prosty sposób „manipulować" bitami - ustawiać jako wejście lub wyjście, odczytywać poszczególne bity, ustawiać rejestry TRIS itp. Dodatkowo, są dostępne funkcje rotacji i przesuwania bitów w słowie. Ze sterowaniem mikrokontrolerem związane są funkcje *sleep()* pozwalające uzyskać rozkaz *sleep*, *reset_cpu()*, *disable_interrupts*, *enable_interrupts()* czy *ext_int_edge()*. Dostępne są też funkcje sterowania timerami i watchdogiem, przetwornikiem A/D i analogowym komparatorem. Możliwe jest także łatwe zapisywanie i odczytywanie wewnętrznej pamięci EEPROM dzięki funkcjom *read_eeprom* i *write_eeprom*.

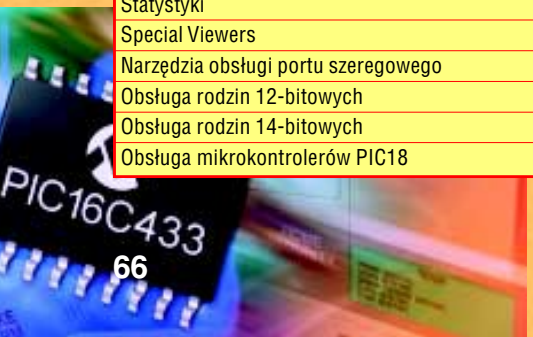
Ważniejsze funkcje biblioteczne, związane ze specyficzną obsługą mikrokontrolera i magistrali, pokazano w **tab. 2**.

Korzystając z wbudowanych funkcji, można szybko tworzyć własne aplikacje. Trzeba tu jednak pamiętać, że z przeniesieniem tak napisanych programów na mikrokontrolery innych producentów będą tym większe trudności im więcej korzystamy z wbudowanych, niestandardowych funkcji kompilatora. Oczywiście, zawsze decyzja należy do programisty: czy wykorzysta np. funkcje obsługi magistrali I²C, czy napisze sobie własne, łatwo przenoszone procedury.

Przedstawiony tutaj kompilator wydaje się być dobrym narzędziem dla większości profesjonalnych i oczywiście amatorskich (ze względu na atrakcyjną cenę!) zastosowań. Niewątpliwie na

Tab. 1. Porównanie różnych wersji kompilatorów oferowanych przez CCS.

Właściwości	PCB	PCM	PCH	PCW	PCWH
Command Line Compiler	✓	✓	✓	✓	✓
Built-in Functions	✓	✓	✓	✓	✓
Example Programs	✓	✓	✓	✓	✓
Device Drivers	✓	✓	✓	✓	✓
MPLAB Interface	✓	✓	✓	✓	✓
Windows IDE				✓	✓
C Aware Editor			✓	✓	
Wbudowany kreator projektu				✓	✓
Zaawansowana optymalizacja kodu wynikowego				✓	✓
Device Selector/Editor				✓	✓
Call Tree and Memory Map	✓	✓	✓	✓	✓
Statystyki				✓	✓
Special Viewers				✓	✓
Narzędzia obsługi portu szeregowego				✓	✓
Obsługa rodzin 12-bitowych	✓			✓	✓
Obsługa rodzin 14-bitowych		✓		✓	✓
Obsługa mikrokontrolerów PIC18			✓		✓



uwagę zasługuje rozbudowana biblioteka procedur i dość dobra dokumentacja. Nie bez znaczenia jest też fakt, że dostępne są trzy, prawie nie różniące się w obsłudze, kompilatory dla głównych rodzin mikrokontrolerów Microchip'a w dość przystępnej cenie. Każdy z kompilatorów można dostać u krajowego dystrybutora za cenę zbliżoną do ceny pakietu Bascom. Jak już wspomniałem, jest to tylko sam kompilator, ale możliwość pełnej współpracy z darmowym MPLAB całkowicie tę niedogodność eliminuje.

Wersja PCW jest już zdecydowanie droższa, ale daje możliwość korzystania ze wszystkich trzech kompilatorów w jednolitym środowisku IDE. Wersja testowa pakietu PCW rozprowadzana jest razem z pakietem MPLAB. Można ją też ściągnąć ze strony Microchipa, co też jest pewną rekomendacją produktów CCS. Pewną niedogodnością kompilatorów CCS jest brak możliwości kompilowania i linkowania więcej niż jednego pliku. Przy rosnących rozmiarach pamięci programu, pliki źródłowe mogą osiągać duże rozmiary i dzielenie ich na mniejsze części znacznie ułatwiają pracę.

Na stronie *www.ccsinfo.com* można znaleźć wiele interesujących, dodatkowych informacji dotyczących kompilatora. Godny polecenia jest mały test porównawczy kompilatorów CCS z produktami innych firm, znajdujący się na stronie *www.ccsinfo.com/compare.html*.

Tomasz Jabłoński, AVT
tomasz.jablonski@ep.com.pl

Tab. 2. Zestawienie ważniejszych funkcji bibliotecznych.

RS232	I²C	SPI I/O	Parallel Slave
Getc()	I2C_start()	Setup_spi()	Setup_psp()
Putc()	I2C_stop()	Spi_read()	Psp_input_full()
Puts()	I2_read()	Spi_write()	Psp_output_full()
Printf()	I2C_write()	Spi_data_is_in()	Psp_overflow()
Kbhit()	I2C_poll()		
Set_uart_speed()			
Timers&delay	**EEPROM**	**Procesor**	**Operacje bitowe**
Delay_us()	Read_eeprom()	Sleep()	Shift_right()
Delay_ms()	Write_eeprom()	Reset_cpu()	Shift_left()
Delay_cycles()	Read_calibration ()	Disable_interrupts()	Rotate_right()
Setup_timer_x()		Enable_interrupts()	Rotate_left()
Set_timer_x()		Ext_int_edge()	Bit_clear()
Get_timer()			Bit_set()
Setup_counters()			Bit_test()
Setup_wdt()			Swap()
Restart_wdt()			

Ceny kompilatorów

PCB dla 12 bit	99USD
PCM dla 14 bit	99USD
PCH dla PIC18	99USD
PCW dla 12 i 14 bit	350USD

Dodatkowe informacje

Kompilatory firmy CCS udostępniła redakcji firma Gamma, tel. (22) 663-83-76, www.gamma.pl.

Dodatkowe informacje o ofercie firmy CCS można znaleźć w Internecie pod adresem: www.ccsinfo.com.