CodeVisionAVR

Kompilator C dla AVR na każdą kieszeń

Gdy na rynku pojawiły się pierwsze mikrokontrolery z rdzeniem ARM przeznaczone dla szerokiego grona odbiorców (tzn. atrakcyjne cenowo, nie wymagające ogromnych nakładów finansowych na rozpoczęcie z nimi prac oraz możliwe do kupienia w niewielkich ilościach), pojawiły się głosy, że nadchodzi koniec ery mikrokontrolerów 8–bitowych. Od tamtej pory minęło kilka lat, układy 8–bitowe nadal cieszą się dużą popularnością i nic nie wskazuje, żeby w najbliższym czasie miało się to zmienić. Co więcej, producenci wciąż opracowują nowe układy z rodzin obecnych na rynku od wielu lat, a ceny tych układów są coraz atrakcyjniejsze.

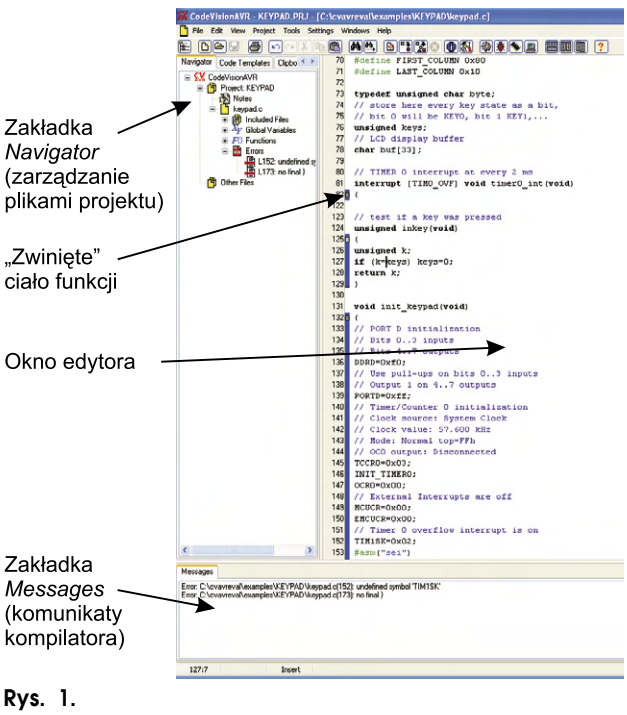
W Polsce, w zastosowaniach popularnych, nieustannie prym wiodą mikrokontrolery z rodziny AVR. Na rynku można znaleźć bogatą ofertę zestawów uruchomieniowych, programatorów oraz kompilatorów różnych języków programowania przeznaczonych dla tych właśnie układów. Dużą i zasłużoną popularnością cieszy się kompilator języka Basic firmy MCS Electronics (Bascom AVR) – jest on łatwy do przyswojenia, producent dostarcza mnóstwo przykładów oraz bogate biblioteki do obsługi urządzeń peryferyjnych, a programy pisze się w nim szybko i stosunkowo łatwo. No i oczywiście można liczyć na pomoc licznego grona użytkowników tego języka w Polsce.

Jednak dla wielu programistów ograniczenia tego języka są trudne do zaakceptowania, np. brak możliwości wykonywania działań na więcej niż dwóch argumentach (obliczenie *a=b+c+d* wymaga najpierw obliczenia *a=b+c*, a następnie *a=a+d*). Podczas budowania większych aplikacji trudno również nie docenić możliwości pracy z projektami, której w Bascomie niestety nie ma (chociaż można znaleźć przykłady rozbudowanych aplikacji napisanych w Bascomie, np. wykorzystujących Ethernet do przesyłania danych).

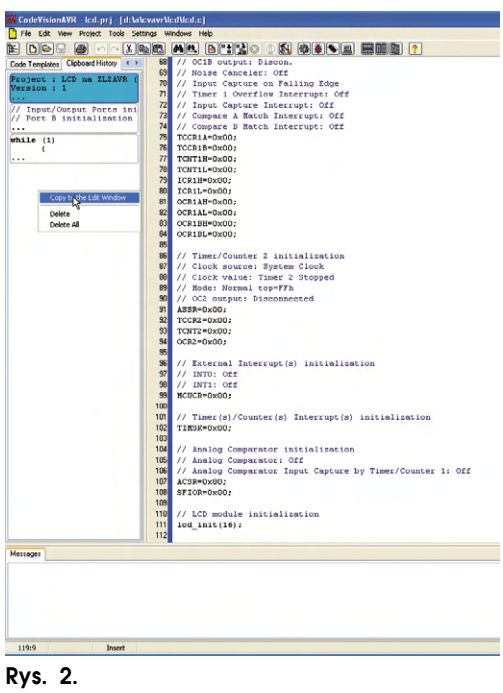
Tego typu niedogodności jest pozbawiony język C, który w świecie mikrokontrolerów jest jedynym powszechnie wykorzystywanym, uniwersalnym językiem programowania. Dosyć często jest spotykana opinia, że jest to język trudny do opanowania, jednak oceniając chociażby liczbę różnorodnych przykładów dostępnych w Internecie, trudno się z tym zgodzić. Odkładając na bok dywagacje na temat wyższości jednego języka programowania nad innym, warto zwrócić uwagę na jeden niekwestionowany przez nikogo fakt: kompilatory języka C są dostępne dla wszystkich rodzin mikrokontrolerów i mikroprocesorów. Wynikają z tego dwie potężne zalety języka C – po pierwsze, znaczną część raz napisanego kodu można przenosić między różnymi projektami (nawet wykonywanymi na całkiem odmiennym sprzęcie!). Po drugie, czas poświęcony na naukę programowania z pewnością będzie procentował w życiu zawodowym.

Biorąc pod uwagę wymienione argumenty wydawałoby się, że nikt nie powinien mieć wątpliwości wybierając język programowania i wszyscy

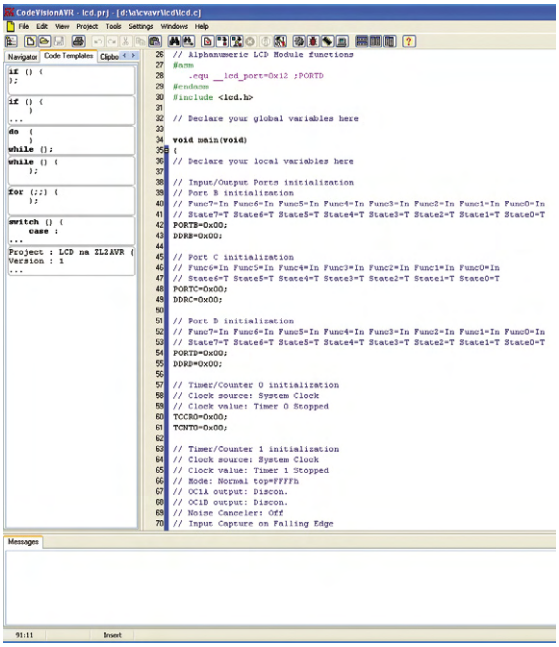
Wersja demonstracyjna CodeVisionAVR jest do pobrania ze strony producenta: *http://www.hpinfotech.ro/*.
Ograniczenia wersji demonstracyjnej:
– kod wynikowy nie większy niż 2 kB;
– brak bibliotek do obsługi: PCF8563, PCF8583, DS1302, DS1307, wyświetlacza LCD 4x40.



Rys. 1.



Rys. 2.



Rys. 3.

programowaliby w C. Niestety, dobry i wygodny w używaniu kompilator języka C dla jednej rodziny mikrokontrolerów to wydatek wielu tysięcy złotych (np. IAR), co jest wystarczającą przeszkodą dla większości chętnych. Na przeciwległym końcu znajduje się kompilator GCC, który co prawda jest dostępny bezpłatnie, jednak jego używanie w większości przypadków wiąże się z dosyć trudną konfiguracją i niezbyt przyjemnym środowiskiem pracy. W większości przypadków, bowiem Atmel umożliwił w AVR Studio pełną współpracę z kompilatorem AVR–GCC (zalecana przez producenta dystrybucja to WinAVR). Dzięki temu programiści mogą poświęcić swój czas na pisanie programów, a nie na pokonywanie kolejnych trudności związanych z przygotowywaniem stanowiska pracy. Jedyną niedogodnością pracy w takim tandemie (AVR Studio + AVR–GCC) jest konieczność tolerowania, niekiedy zaskakujących, niespodzianek związanych z działaniem kompilatora, np. irytująca potrzeba modyfikowania programów przygotowywanych we wcześniejszych wersjach GCC (czasami albo się nie kompilują, albo działają niezgodnie z oczekiwaniami).

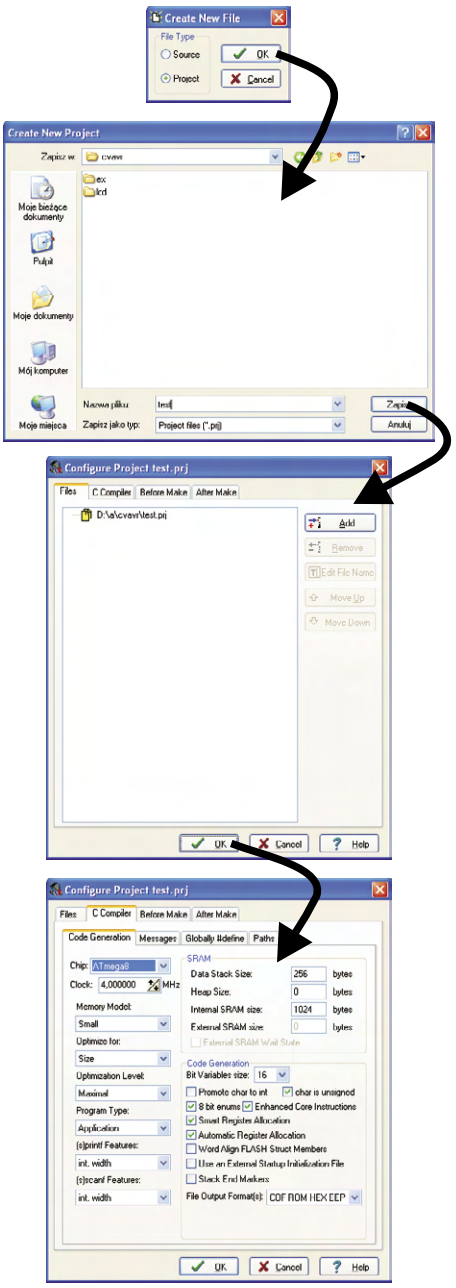
Rynek nie toleruje pustki, więc miejsce pomiędzy drogimi kompilatorami o największych możliwościach i bezpłatnym AVR–GCC zajęły kompilatory (a właściwie kompilatory z IDE) będące rozsądnym kompromisem pomiędzy ceną produktu i jego możliwościami. W segmencie kompilatorów *low cost* można wymienić kilkanaście produktów wartych uwagi, jednak największe uznanie odbiorców zdobyły *ICCV7 for AVR* firmy ImageCraft (ceny od 249 USD netto) oraz CodeVisionAVR firmy HP InfoTech (cena 150 EUR netto). W artykule zaprezentujemy najważniejsze cechy drugiego z wymienionych.

CodeVisionAVR

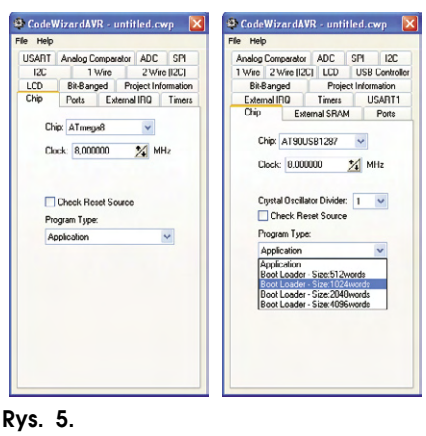
Opisywana wersja CodeVisionAVR jest oznaczona numerem 1.25.9. Wersja demonstracyjna jest dostępna na stronie producenta (*http://www.hpinfotech.ro/*) i umożliwia nielimitowaną w czasie pracę i kompilowanie programów, których kod wynikowy jest mniejszy niż 2 kB. Program jest przeznaczony do pracy pod kontrolą systemów Windows (95/98/2000/XP/Vista), jego instalacja przebiega typowo dla aplikacji przeznaczonych do tych systemów.

Na **rys. 1** pokazano okno programu z otwartym przykładowym projektem obsługi klawiatury matrycowej. Największa część ekranu to oczywiście okno edytora, który oferuje możliwości typowe dla edytorów przeznaczonych dla programistów, m.in. kolorowanie składni, numerowanie linii programu, możliwość „zwijania" (ukrywania) bloków programu.

Na dole okna z rys. 1 znajduje się okno *Messages*, w którym są wyświetlane komunikaty generowane przez kompilator, np. o błędach w programie (kliknięcie w komunikat informujący o błędzie powoduje przeniesienie do odpowiedniej linii programu źródłowego). Z lewej strony znajduje się zakładka *Navigator* ułatwiająca zarządzanie plikami wchodzącymi w skład projektu. Zamiast tej zakładki można wyświetlić zakładkę *Clipboard History* (**rys. 2**), która umożliwia zarządzanie fragmentami tekstów ostatnio przechowywanymi w schowku systemowym. Rozwiązanie takie pozwala np. łatwo przenosić wiele fragmentów programów między sobą. Na **rys. 3** pokazano zakładkę *Code Templates*, w której warto umieścić najczęściej wykorzystywane fragmenty programów, np. typowy dla wykonywanych projektów nagłówek programu lub najczęściej stosowane konstrukcje instrukcji, i później wygodnie je wstawiać do kolejnych programów.



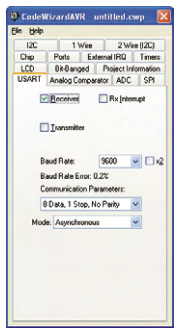
Rys. 4.



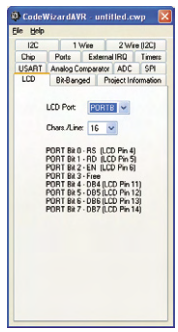
Rys. 5.

Rys. 6.

Rys. 7.



Rys. 8.



Rys. 9.

W CodeVisionAVR można stworzyć projekt w tradycyjny sposób, czyli dodając istniejące lub tworząc nowe pliki źródłowe, określając typ mikrokontrolera, częstotliwość taktowania itp. (**rys. 4**). Można również (i warto) skorzystać z wbudowanego generatora szkieletu programu CodeWizardAVR. W zależności od wybranego typu mikrokontrolera (**rys. 5**) można definiować funkcje poszczególnych wyprowadzeń, sposób pracy wbudowanych układów peryferyjnych (SPI, UART, CAN, USB, watchdog itd.), sposób pracy obsługiwanych zewnętrznych układów peryferyjnych (np. zewnętrznej pamięci danych czy alfanumerycznego wyświetlacza LCD). Wykaz obsługiwanych mikrokontrolerów znajduje się w **tab. 1**.

Dla lepszego wyobrażenia sobie wygody korzystania z CodeWizard

Tab. 1. Wykaz mikrokontrolerów obsługiwanych przez CodeVisionAVR (wersja 1.25.9)

ATtiny13
ATtiny22
ATtiny2313
ATtiny24, ATtiny44, ATtiny84
ATtiny25, ATtiny45, ATtiny85
ATtiny26, ATtiny46, ATtiny86, ATtiny166
ATtiny261, ATtiny461, ATtiny861
AT90S2313
AT90S2323, AT90S2343
AT90S2333, AT90S4433
AT90S4414, AT90S8515
AT90S4434, AT90S8535
AT90S8534
AT90CAN32, AT90CAN64, AT90CAN128
AT90PWM2, AT90PWM2B, AT90PWM3,
AT90PWM3B, AT90PWM216, AT90PWM316
AT90USB1286, AT90USB1287, AT90USB646,
AT90USB647, AT90USB162, AT90USB82
ATmega103
ATmega128, ATmega1280, ATmega1281
ATmega161
ATmega162
ATmega163
ATmega164, ATmega164P
ATmega165
ATmega169
ATmega2560, ATmega2561
ATmega32
ATmega323
ATmega324, ATmega324P
ATmega325, ATmega325P, ATmega3250,
ATmega3250P
ATmega329, ATmega329P, ATmega3290,
ATmega3290P
ATmega406
ATmega48, ATmega48P, ATmega88,
ATmega88P, ATmega168, ATmega168P,
ATmega328P
ATmega603
ATmega64
ATmega640
ATmega644, ATmega644P
ATmega645, ATmega6450
ATmega649, ATmega6490
ATmega8, ATmega16
ATmega8515
ATmega8535
FPSLIC AT94K05, AT94K10, AT94K20,
AT94K40
AT43USB355
AT76C711
AT86RF401

List. 1.

```
1  /*****************************************************
2  This program was produced by the
3  CodeWizardAVR V1.25.9 Evaluation
4  Automatic Program Generator
5  © Copyright 1998-2008 Pavel Haiduc, HP InfoTech s.r.l.
6  http://www.hpinfotech.com
7
8  Project : Test_EP
9  Version : 1
10   Date    : 2008-04-09
11   Author  : AK
12   Company :
13   Comments:
14   Program testowy dla EP
15   Odbieranie danych przez USART
16   i wyświetlanie ich na LCD
17
18
19   Chip type           : ATmega8
20   Program type        : Application
21   Clock frequency     : 8,000000 MHz
22   Memory model        : Small
23   External SRAM size  : 0
24   Data Stack size     : 256
25   *****************************************************/
26
27   #include <mega8.h>
28
29   // Alphanumeric LCD Module functions
30   #asm
31      .equ __lcd_port=0x18 ;PORTB
32   #endasm
33   #include <lcd.h>
34
35   // Standard Input/Output functions
36   #include <stdio.h>
37
38   // Declare your global variables here
39
40   void main(void)
41   {
42   // Declare your local variables here
43
44   // Input/Output Ports initialization
45   // Port B initialization
46   // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
47   // State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
48   PORTB=0x00;
49   DDRB=0x00;
50
51   // Port C initialization
52   // Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
53   // State6=T State5=T State4=T State3=T State2=T State1=T State0=T
54   PORTC=0x00;
55   DDRC=0x00;
56
57   // Port D initialization
58   // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
59   // State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
60   PORTD=0x00;
61   DDRD=0x00;
62
63   // Timer/Counter 0 initialization
64   // Clock source: System Clock
65   // Clock value: Timer 0 Stopped
66   TCCR0=0x00;
67   TCNT0=0x00;
```

AVR wykonajmy prosty projekt, w którym na wyświetlaczu LCD będą wyświetlane znaki odbierane przez moduł UART mikrokontrolera. Pierwszym krokiem jest wybranie wykorzystywanego mikrokontrolera oraz jego częstotliwości taktowania (**rys. 6**). Następnie, dla zachowania porządku, warto wypełnić pola w zakładce *Project Information* (**rys. 7**). W zakładce *USART* (**rys. 8**) zaznaczamy *Receiver*, natomiast *Transmitter* pozostawiamy niezaznaczone, zadaniem mikrokontrolera jest tylko odbieranie danych. Należy wybrać również odpowiednie parametry transmisji (np. 9600 b/s, 8 bitów danych, 1 bit stopu, bez kontroli parzystości). W przykładowym projekcie będziemy również wykorzystywać wyświetlacz LCD, więc w zakładce *LCD* (**rys. 9**) należy wybrać port, do którego jest on dołączony. Po wybraniu z menu *File>Generate, Save and Exit* otrzymamy szkielet naszego programu pokazany na **list. 1**. Z programu tego można usunąć inicjalizację układów peryferyjnych niewykorzystywanych w przykładowej aplikacji – w efekcie powstanie znacznie krótszy program źródłowy pokazany na **list. 2** i jest to gotowy szkielet programu. Przykładowy program powinien pracować w nieskończonej pętli `while(1)`, która została dodana przez CodeWizardAVR. Kod niezbędny do zrealizowania postawionego zadania należy wpisać w miejsce oznaczone `// Place your code here`. Jak widać napisanie całej naszej aplikacji to tylko CZTERY linijki programu, pozostała część została wygenerowana przez CodeWizardAVR.

Ponieważ kompilator generuje pliki w formacie *coff*, to możliwe jest debuggowanie programu w AVRStudio (uruchamianego z poziomu CodeVisionAVR) zarówno w postaci programu źródłowego w języku C (**rys. 10**), jak i po deasemblacji (**rys. 11**).

W CodeVisionAVR zintegrowano również aplikację do programowania w systemie mikrokontrolerów AVR o nazwie CodeVisionAVR Chip Programmer. Program ten współpracuje z większością popularnych programatorów – warto zwrócić uwagę, że z poziomu IDE można programować zarówno za pomocą popularnych programatorów zgodnych z STK200, jak i z wykorzystaniem takich narzędzi jak AVRISP MkII (USB), AVR Dragon czy JTAGICE MkII.

O popularności kompilatorów z segmentu *low cost* decydują dostarczane

List. 1. c.d.

```
68
69   // Timer/Counter 1 initialization
70   // Clock source: System Clock
71   // Clock value: Timer 1 Stopped
72   // Mode: Normal top=FFFFh
73   // OC1A output: Discon.
74   // OC1B output: Discon.
75   // Noise Canceler: Off
76   // Input Capture on Falling Edge
77   // Timer 1 Overflow Interrupt: Off
78   // Input Capture Interrupt: Off
79   // Compare A Match Interrupt: Off
80   // Compare B Match Interrupt: Off
81   TCCR1A=0x00;
82   TCCR1B=0x00;
83   TCNT1H=0x00;
84   TCNT1L=0x00;
85   ICR1H=0x00;
86   ICR1L=0x00;
87   OCR1AH=0x00;
88   OCR1AL=0x00;
89   OCR1BH=0x00;
90   OCR1BL=0x00;
91
92   // Timer/Counter 2 initialization
93   // Clock source: System Clock
94   // Clock value: Timer 2 Stopped
95   // Mode: Normal top=FFh
96   // OC2 output: Disconnected
97   ASSR=0x00;
98   TCCR2=0x00;
99   TCNT2=0x00;
100  OCR2=0x00;
101
102  // External Interrupt(s) initialization
103  // INT0: Off
104  // INT1: Off
105  MCUCR=0x00;
106
107  // Timer(s)/Counter(s) Interrupt(s) initialization
108  TIMSK=0x00;
109
110  // USART initialization
111  // Communication Parameters: 8 Data, 1 Stop, No Parity
112  // USART Receiver: On
113  // USART Transmitter: Off
114  // USART Mode: Asynchronous
115  // USART Baud Rate: 9600
116  UCSRA=0x00;
117  UCSRB=0x10;
118  UCSRC=0x86;
119  UBRRH=0x00;
120  UBRRL=0x33;
121
122  // Analog Comparator initialization
123  // Analog Comparator: Off
124  // Analog Comparator Input Capture by Timer/Counter 1: Off
125  ACSR=0x80;
126  SFIOR=0x00;
127
128  // LCD module initialization
129  lcd_init(16);
130
131  while (1)
132        {
133        // Place your code here
134
135        };
136  }
```

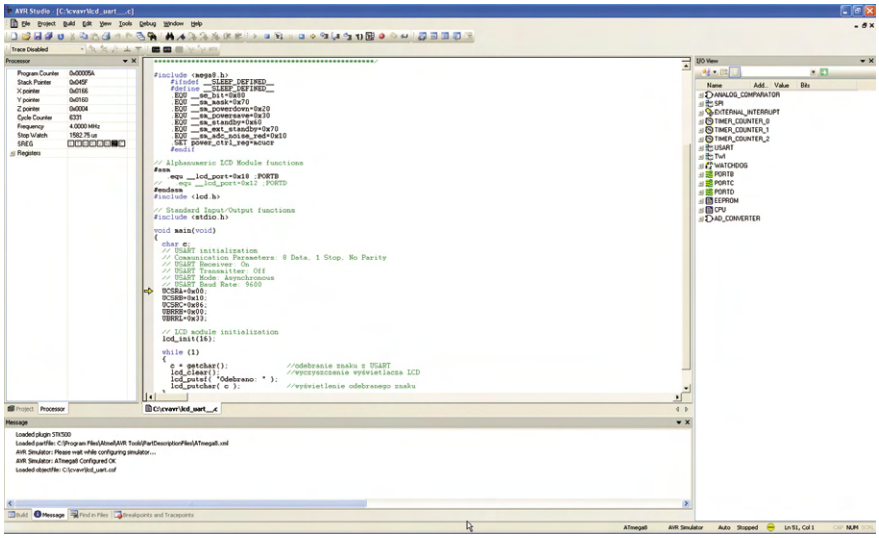
List. 2.

```
1  /*****************************************************
2  Project : Test_EP
3  Version : 1
4  Date    : 2008-04-09
5  Author  : AK
6  Comments:
7  Program testowy dla EP
8  Odbieranie danych przez USART
9  i wyświetlanie ich na LCD
10
11   Chip type           : ATmega8
12   Program type        : Application
13   Clock frequency     : 8,000000 MHz
14   Memory model        : Small
15   External SRAM size  : 0
16   Data Stack size     : 256
17   *****************************************************/
18
19   #include <mega8.h>
20
21   // Alphanumeric LCD Module functions
22   #asm
23      .equ __lcd_port=0x18 ;PORTB
24   #endasm
```

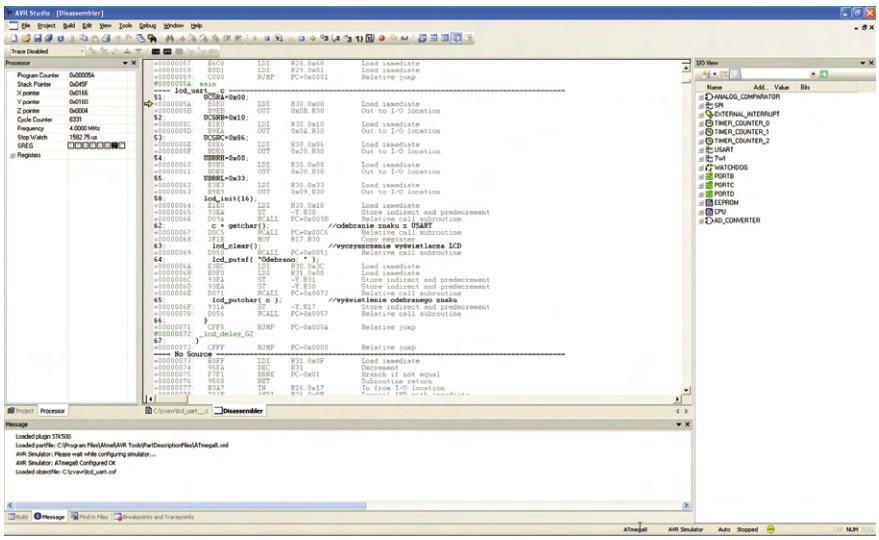
List. 2. c.d.

```
25   #include <lcd.h>
26
27   // Standard Input/Output functions
28   #include <stdio.h>
29
30   void main(void)
31   {
32   // USART initialization
33   // Communication Parameters: 8 Data, 1 Stop, No Parity
34   // USART Receiver: On
35   // USART Transmitter: Off
36   // USART Mode: Asynchronous
37   // USART Baud Rate: 9600
38   UCSRA=0x00;
39   UCSRB=0x10;
40   UCSRC=0x86;
41   UBRRH=0x00;
42   UBRRL=0x33;
43
44   // LCD module initialization
45   lcd_init(16);
46
47   while (1)
48         {
49         // Place your code here
50
51         };
52   }
```

przez producenta biblioteki do obsługi najpopularniejszych układów peryferyjnych. „Goły” kompilator powoduje, że przed rozpoczęciem jakichkolwiek prac trzeba zebrać i przetestować (lub samodzielnie napisać) funkcje obsługujące podstawowe układy peryferyjne. Jest to zadanie z pewnością kształcące, jednak wymaga wiele czasu i sporo doświadczenia zarówno w programowaniu, jak i w szczegółach dotyczących pracy specyficznych układów.



Rys. 10.



Rys. 11.

Biblioteki dostarczane w CodeVisionAVR wraz z kompilatorem można podzielić na dwie grupy. Pierwszą z nich stanowią standardowe biblioteki języka C (m.in. *stdio*, *stdlib*, *ctype*, *math*) w których są zawarte funkcje pozwalające na operacje na zmiennych znakowych (np. *isprint()*, *toupper()*), tekstowych (np. *strcmp()*), zarządzanie pamięcią (np. *malloc()*), obsługę operacji wejścia–wyjścia (np. *getchar()*, *printf()*, *scanf()*), wykonywanie operacji matematycznych (np. *sqrt()*, *fmod()*).

Drugą grupę stanowią biblioteki zawierające funkcje obsługi specyficznych układów peryferyjnych. Warto tutaj wymienić bibliotekę do obsługi alfanumerycznych wyświetlaczy LCD (np. *lcd_clear()*, *lcd_gotoxy()*) czy funkcje obsługi transmisji szeregowej SPI, I²C (np. *i2c_read()*, *i2c_start()*), 1–Wire (np. *w1_read()*, *w1_search()*).

Oprócz wymienionych bibliotek ogólnego przeznaczenia w CodeVisionAVR zawarto również zestawy funkcji do obsługi konkretnych układów: czujników temperatury (LM75, DS1621, DS1820, DS18B20, DS1822), układów RTC (PCF8563, PCF8583, DS1302, DS1307) i pamięci EEPROM (DS2430, DS2433). Przykładowo do odczytania z układu PCF8583 aktualnej godziny wystarczy, po zainicjowaniu interfejsu I²C (funkcją *i2c_init()*), wywołać funkcję *rtc_get_time()*.

Mamy nadzieję, że ten krótki i bardzo pobieżny opis możliwości CodeVisionAVR, zachęci chociaż część naszych Czytelników do wypróbowania sił w programowaniu w języku C. Wydaje się, że kompilator ten (również w bezpłatnej wersji demonstracyjnej) jest idealnym narzędziem, pozwalającym na stosunkowo łatwe i bezstresowe zdobycie pierwszych doświadczeń w programowaniu w C. Z równym powodzeniem może być także wykorzystywany w zaawansowanych projektach, gdyż oferuje wiele możliwości ułatwiających pracę programisty.

Andrzej Gawryluk, EP

Dystrybutorem kompilatora CodeVisionAVR jest *Kamami.pl*.