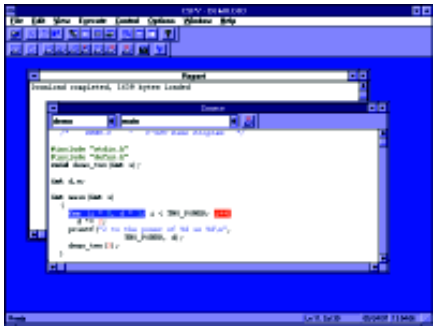
Realizacja projektów na 8051 przy pomocy oprogramowania firmy IAR SYSTEMS

W poprzednim odcinku przybliżyliśmy Czytelnikom sposób posługiwania się kompilatorem C firmy IAR podczas tworzenia nowego projektu oraz jego elementów. Po utworzeniu kodu wynikowego czas na jego analizę czyli - „debugging".

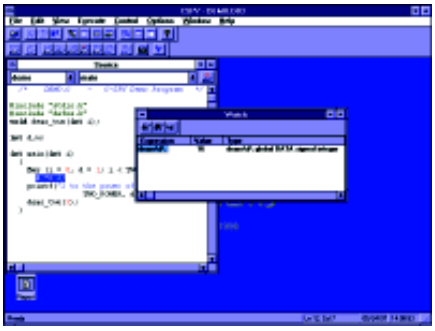
Rys. 1.



Rys. 2.

Uruchomienie debugera C-SPY następuje poprzez kliknięcie ikony w systemie Embedded Workbench, lub poprzez wybranie opcji „Debugger" z menu „Project".

Program źródłowy ze wszystkimi komponentami zostanie automatycznie skompilowany i skonsolidowany przez kompilator, po czym zostanie uruchomiony debuger. Aby rozpocząć sesję należy teraz jedynie otworzyć zbiór debugera utworzony na etapie kompilacji, w naszym przypadku będzie to „demo.d03".



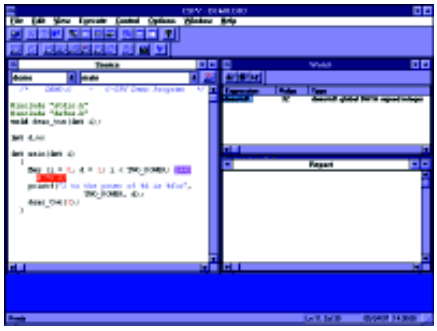
Rys. 3.

Tak więc wybieramy opcję „Open" z menu „File", a następnie klikamy na zbiór „demo.d03". Debugger otworzy zbiór, następnie pokaże puste okno „Source", ponieważ w naszym przykładzie nie zdefiniowano kodu związanego bezpośrednio z modułem CSTARTUP. Należy więc wybrać jeden z dwóch (w naszym przypadku) źródeł, są to programy : „demo" i „demo_two". Wybieramy „demo", w oknie „Source" pojawi się listing źródłowy „demo.s03" (**rys. 2**).

Analiza na poziomie „C"

Analizę rozpoczniemy od trybu „krok po kroku", w tym celu wystarczy kliknąć na odpowiednią ikonę lub wybrać opcję „Step" z menu „Execute". Program rozpoczyna pracę zawsze od funkcji „main", tak więc w naszym przypadku ekran debugera będzie wyglądał jak na **rys. 5**.

Rys. 4.

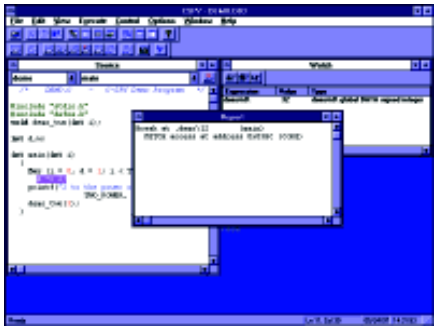


Rys. 5.

Aktualna pozycja debugera jest zaznaczona jako wyróżniony kolorem tekst w oknie program źródłowego., w naszym przypadku jest to linia:

```
for (i = 0, d = 1; i < TWO_POWER; i++)
......
```

Wykonywanie kolejnych kroków programu polega na dalszym klikaniu na ikonę „Step".



Rys. 6.

Podgląd wybranych zmiennych

C-SPY jak każdy program tego typu pozwala na bieżącą obserwację zmiennych zawartych w kodzie źródłowym programu oraz w zależności od potrzeby ich modyfikację.

Aby otworzyć okno służące tej funkcji wystarczy wybrać opcję „Watch" z menu „Window" lub kliknąć na odpowiadającą jej ikonę u góry ekranu. Na ekranie pojawi się okno „Watch", należy teraz wprowadzić wybraną zmienną podając jej nazwę.

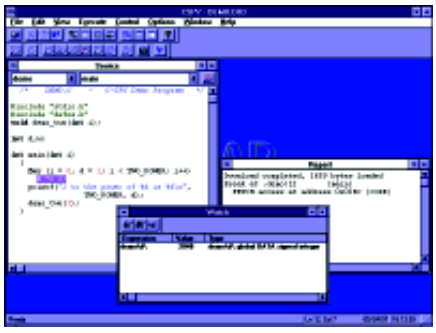
Rys. 7.

W naszym przypadku wybraliśmy zmienną typu „integer“ : „d“. Każda zmienna w oknie „Watch“ jest opisana także nazwą funkcji w której jest użyta, podany jest jej typ oraz aktualna wartość.

Wykonajmy teraz dla przykładu sześć kolejnych kroków pętli, aby zaobserwować zmiany w oknie „Watch“. W tym celu, zamiast używać ikony „Step“, posłużymy się opcją „Mutli Step...“ z menu „Execute“, wprowadzając liczbę kroków, u nas będzie to 6.

Ustawianie pułapek

Użytkownik w prosty sposób może ustawić dowolna ilość „Breakpoint'ów“ - czyli pułapek w kodzie źródłowym C lub na poziomie asemblera 8051, przez podanie numeru linii, lub nazwy funkcji. Można to zrobić także w prostszy sposób, ustawiając kursor na wybranej linii i wydając komendę „Toggle Breakpoint“.



Rys. 8.

W tym celu należy otworzyć okno „Report“, które posłuży do przedstawienia informacji na temat wszystkich ustawionych później pułapek. W naszym przykładzie ustawimy pułapkę w linii:

```
d *= 2;
```

Breakpoint zostanie ustawiony, co widoczne jest wyróżnieniem linii kolorem czerwonym w oknie „Source“, okno „Report“ jest na razie puste, bowiem nie wykonaliśmy kolejnego kroku programu.

Wykonanie programu do pułapki

Aby wykonać program do kolejnego brakpointu należy wybrać opcję „Go“ z menu „Execute“, lub kliknąć odpowiadającą tej komendzie ikonę. Program zostanie wykonany do napotkania pułapki, która została wcześniej ustawiona, co zostanie potwierdzone odpowiednim komunikatem w oknie „Report“.

Program C-SPY pozwala użytkownikowi na definiowanie bardziej rozbudowanych pułapek warunkowych. Dla przykładu ustawimy pułapkę w programie, która zdefiniuje przekroczenie wartości 2000 dla zmiennej „d“.

Rys. 9.

W tym celu należy wybrać opcję „Edit Breakpoints...“ z menu „Control“, po tym zostanie wyświetlone okno edycji pułapek (**rys. 7**).

Wprowadzamy w linii „Condition“ nasz warunek „d>2000“ i wybieramy w linii „Condition Type“ opcję „Condition True“. Następnie potwierdzamy operacje klikając na klawisz „Modify“, wreszcie zamykamy okno „Breakpoints“ wybierając „Close“.

Rys. 10.

Teraz możemy wykonać program aż do momentu spełnienia warunku pułapki, używając podobnie jak poprzednio opcji „Go“ z menu „Execute“. Program automatycznie zatrzyma się w momencie kiedy zmienna "d" przyjmie wartość 2048 - warunek został spełniony. W oknie „Watch“ zobaczyć można zmienną d oraz jej aktualną wartość (**rys. 8**).

Wykonanie programu „do kursora“

Inną pożyteczną opcją podczas sesji z debugerem C-SPY jest wykonanie instrukcji aż do tej, która aktualnie jest wskazywana przez kursor.

Najpierw należy jednak otworzyć okno „Terminal I/O“ z menu „Window“, aby w efekcie uzyskać standardowe wyjście funkcji „printf“ w tym oknie. Pamiętajmy jednak że aby uaktywnić je należy zlinkować moduły programu z opcją „-rt“, która uaktywnia standardowe wyjście funkcji I/O dla potrzeb debugera C-SPY.

Rys. 11.

Wykonajmy dla przykładu program do instrukcji

```
demo_two(3);
```

W tym celu ustawiamy kursor na tej linii, a następnie wykonujemy komendę „Goto Cursor“. Program zostanie wykonany do tej linii czego efektem będzie wypisanie komunikatu w oknie „Terminal I/O“, wygenerowanym przez instrukcję „printf“ w naszym przykładzie (**rys.9**).

Wykonanie programu do funkcji

Możliwe jest także wykonanie programu aż do wywołania interesującej nas funkcji: komenda „Step into a function“ z menu „Execute“.

W naszym przykładzie po wywołaniu tej komendy okno „Source“ pokaże teraz wskazaną funkcję „demo_two(3);“ z pliku DEMO_TWO.C.

Przy analizie programu podczas wywołań sekwencji funkcji, często niezbędne staje się ich śledzenie. W programie C-SPY do tego celu służy okno „Calls“ w menu „Window“ (**rys.11**).

Sławomir Surowiński, AVT

Oprogramowanie udostępniła redakcji firma RK-System.