

Krótką instrukcja obsługi debuggera HI-WAVE

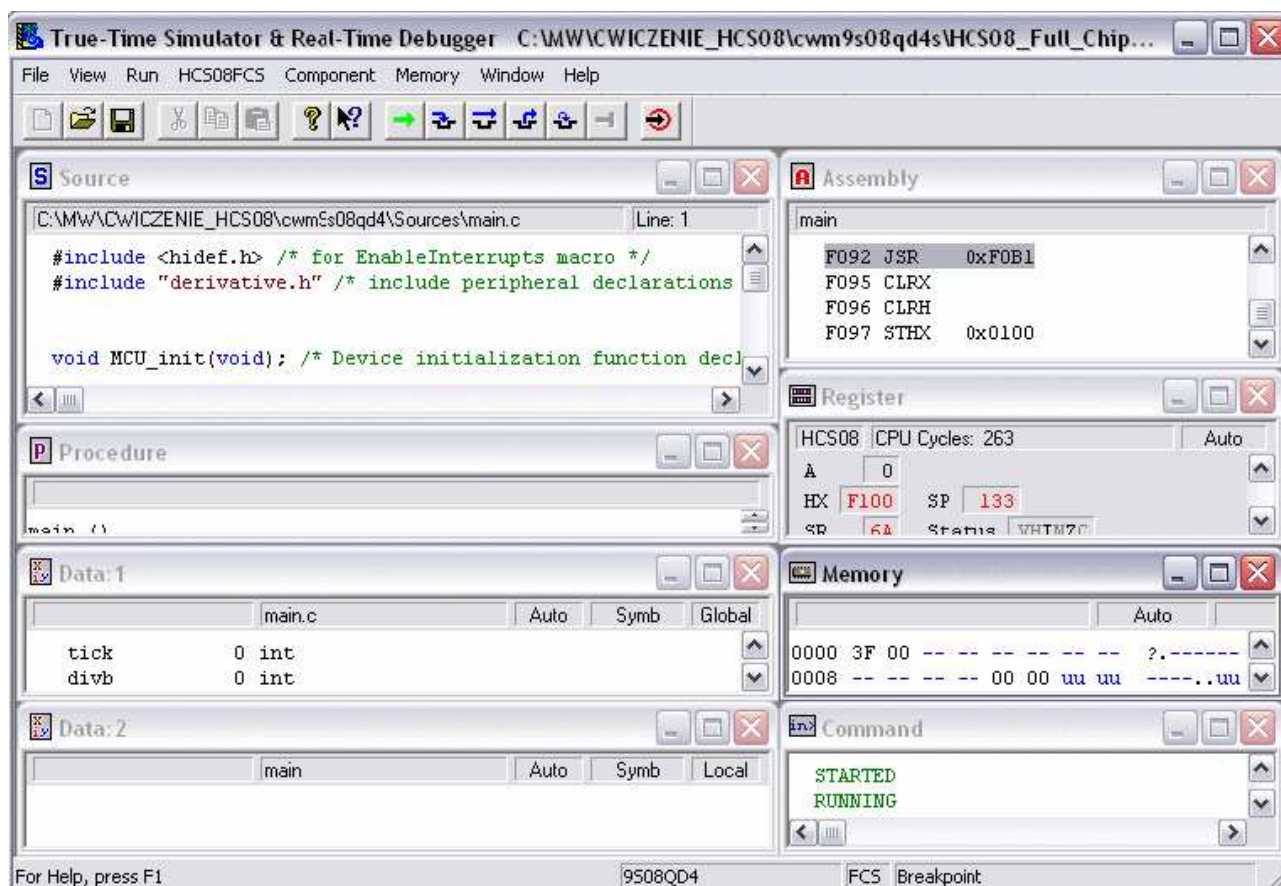
opracowano na podstawie:

”HI-WAVE Product Manual”, Hiware, 1999

Marek Wnuk
Wrocław, 2002-2012

1 Uruchamianie HI-WAVE

W środowisku Code Warrior debugger HI-WAVE uruchamia się z menu (**Project | Debug**) lub ikoną z paska narzędziowego. Katalogiem roboczym jest katalog aktualnie otwartego projektu.



Rysunek 1: Okno debuggera HI-WAVE

1.1 Automatyzowanie startu HI-WAVE

Komendy debuggera, które są zawsze powtarzane po starcie można zautomatyzować przez ich wpisanie do pliku *startup.cmd*, który jest uruchamiany przy ładowaniu interfejsu systemu docelowego

(*target component*). Na przykład:

```
load hello.abs
bs &main t
g
```

spowoduje załadowanie aplikacji *hello*, ustawienie tymczasowej (jednorazowej) pułapki (*temporary breakpoint*) na początku funkcji *main* i uruchomienie aplikacji. W ten sposób uzyskamy wykonanie kodu startowego oraz inicjalizacji i zatrzymanie programu na początku funkcji *main*.

1.2 Ładowanie aplikacji

1. wybranie **Target | Load ...** spowoduje otwarcie okna dialogowego **LoadObjectFile (Target)** na rysunku 1 jest to HCS08FCS - *HCS08 Full Chip Simulator*),
2. wybranie aplikacji (np. *hello.abs*),
3. naciśnięcie **OK** spowoduje zamknięcie okna dialogowego i załadowanie aplikacji.

Okno **Source** zawiera tekst źródłowy modułu, w którym znajduje się punkt startowy aplikacji (zazwyczaj *startup*) z podświetlonym wyrażeniem stanowiącym ten punkt.

Okno **Assembly** zawiera odpowiedni fragment zdeasemblowanego kodu maszynowego z podświetlonym punktem wejściowym.

Okno **Global Data** zawiera listę zmiennych globalnych określonych w module startowym.

Okno **Local Data** jest (na razie) puste.

Rejestr **PC** w oknie **Register** jest zainicjalizowany wartością adresu startowego aplikacji.

1.3 Uruchamianie aplikacji

Są dwa sposoby:

- wybranie **Run | Start/Continue** z menu,
- naciśnięcie ikony **Start/Continue** na pasku narzędziowym HI-WAVE.

Działanie aplikacji jest sygnalizowane przez **RUNNING** na pasku statusowym.

Aplikacja będzie działać dotąd, aż:

- użytkownik ją zatrzyma (patrz dalej),
- zostanie osiągnięta pułapka,
- nastąpi zdarzenie wyjątkowe (*exception*).

1.4 Zatrzymywanie aplikacji

Są dwa sposoby:

- wybranie **Run | Halt** z menu,
- naciśnięcie ikony **Halt** na pasku narzędziowym HI-WAVE.

Zatrzymanie aplikacji jest sygnalizowane przez **HALTED** na pasku statusowym.

W oknach **Source** i **Assembly** są podświetlone linie odpowiadające następnej instrukcji do wykonania.

Okno **Global Data** zawiera listę zmiennych globalnych określonych w aktualnym module (nazwa modułu jest w nagłówku okna).

Okno **Local Data** zawiera listę zmiennych globalnych określonych w aktualnej funkcji (nazwa funkcji jest w nagłówku okna).

2 Praca krokowa na poziomie kodu źródłowego

2.1 Instrukcja po instrukcji (*Single Step*)

- wybranie **Run | Single Step** z menu,
- naciśnięcie ikony **Single Step** na pasku narzędziowym HI-WAVE.

Zatrzymanie po wykonaniu kroku jest sygnalizowane przez **STEPPED** na pasku statusowym.

W oknach **S** i **Assembly** są podświetlone linie odpowiadające następnej instrukcji do wykonania.

W oknach **Register**, **Memory** i **Data** kolorem czerwonym są zaznaczone elementy, których wartość uległa zmianie w wyniku wykonania instrukcji.

2.2 Przeskakując wywołania funkcji (*Step Over*)

- wybranie **Run | Step Over** z menu,
- naciśnięcie ikony **Step Over** na pasku narzędziowym HI-WAVE.

Zatrzymanie po wykonaniu kroku jest sygnalizowane przez **STEPPED OVER** na pasku statusowym.

2.3 Wychodząc z wywołanej funkcji (*Step Out*)

- wybranie **Run | Step Out** z menu,
- naciśnięcie ikony **Step Out** na pasku narzędziowym HI-WAVE.

Zatrzymanie po wykonaniu kroku jest sygnalizowane przez **STOPPED** na pasku statusowym.

2.4 Po jednej instrukcji kodu asemblerowego

- wybranie **Run | Assembly Step** z menu,
- naciśnięcie ikony **Assembly Step** na pasku narzędziowym HI-WAVE.

Zatrzymanie aplikacji po wykonaniu kroku jest sygnalizowane przez **TRACED** na pasku statusowym.

3 Operacje na zmiennych

3.1 Wyświetlanie zmiennych lokalnych z wybranej funkcji

- metoda "przeciągania" (*drag-and-drop*):
 1. "przeciągnięcie" nazwy funkcji z okna **Procedure** do okna **Local Data**;
- metoda "podwójnego pstryknięcia" (*double-click*):
 1. "podwójne pstryknięcie" na nazwie funkcji w oknie **Procedure**.

3.2 Wyświetlanie zmiennych globalnych z wybranego modułu

- otwarcie okna **Module**:
 1. wybranie **Component | Open** z menu (lista dostępnych okien),
 2. "podwójne pstryknięcie" na **Module** (otwarcie okna z listą wszystkich modułów aplikacji),
 3. "przeciągnięcie" nazwy modułu z okna **Module** do okna **Global Data**;
- otwarcie lokalnego menu (*popup menu*):
 1. przyciśnięcie prawego przycisku myszy w oknie **Global Data** (otwarcie lokalnego menu),
 2. wybranie **Open Module** (okno dialogowe z nazwami modułów),
 3. "podwójne pstryknięcie" na nazwie modułu.

3.3 Zmiana formatu wyświetlania wartości zmiennych

1. przyciśnięcie prawego przycisku myszy w oknie **Data** (otwarcie lokalnego menu),
2. wybranie **Format** (okno dialogowe z dostępnymi formatami),
3. "podwójne pstryknięcie" na wybranym formacie.

Dostępne formaty:

Hex - szesnastkowy,

Oct - osemkowy,

Dec - dziesiętny,

UDec - dziesiętny bez znaku,

Bin - binarny,

Symbolic - zależny od typu zmiennej:

1. zmienne wskaźnikowe - szesnastkowy,
2. wskaźniki na funkcje - nazwy funkcji,
3. zmienne znakowe - ASCII i dziesiętny,
4. inne zmienne - dziesiętny z lub bez znaku.

3.4 Zmiana wartości zmiennej

”Podwójne pstryknięcie” na wartości zmiennej w oknie **Data** powoduje jej podświetlenie jej i umożliwienie edycji. Format wyświetlania jest zgodny z regułami ANSI C dla stałych (przedrostek 0x - szesnastkowo, przedrostek 0 - ósemkowo, brak przedrostka - dziesiętnie). Edycję liczby kończy się przez *ENTER* lub *TAB*. Zakończenie edycji przez *TAB* powoduje automatyczne przejście do edycji następnej zmiennej. Naciśnięcie *ESC* powoduje przywrócenie poprzedniej wartości.

3.5 Znajdowanie adresu i rozmiaru zmiennej

”Podwójne pstryknięcie” na nazwie zmiennej powoduje wyświetlenie w nagłówku okna adresu i rozmiaru obszaru pamięci przydzielonego na tę zmienną.

3.6 Przeglądanie pamięci przydzielonej na zmienną

- metoda ”przeciągania” (*drag-and-drop*):
 1. ”przeciągnięcie” nazwy zmiennej z okna **Data** do okna **Memory**;
- użycie klawisza 'A':
 1. wskazanie nazwy zmiennej w oknie **Data**,
 2. naciśnięcie lewego przycisku myszy i klawisza 'A' spowoduje przewijanie okna **Memory** do czasu osiągnięcia adresu alokacji wybranej zmiennej.

Obszar pamięci przydzielony wybranej zmiennej jest podświetlony w oknie **Memory**.

3.7 Ładowanie adresu zmiennej do rejestru

”Przeciągnięcie” nazwy zmiennej do pola wybranego rejestru w oknie **Register**.

4 Działanie na rejestrach

4.1 Zmiana formatu wyświetlania rejestrów

1. przyciśnięcie prawego przycisku myszy w oknie **Register** (otwarcie lokalnego menu),
2. wybranie w **Options** formatu szesnastkowego lub binarnego.

4.2 Zmiana zawartości rejestru

- rejestry liczbowe (adresowe, indeksowe lub danych):
 1. "podwójne pstryknięcie" na rejestrze (podświetlenie jego zawartości i umożliwienie edycji),
 2. edycję kończy się przez *enter* lub *tab* (*tab* powoduje automatyczne przejście do edycji następnego rejestru),
 3. *ESC* powoduje przywrócenie poprzedniej wartości.
- rejestry bitowe (statusowe) - znaki literowe bitów ustawionych na '1' są wyświetlane w kolorze czarnym, bitów '0' - w szarym; "podwójne pstryknięcie" na znaku bitu powoduje zmianę jego wartości.

4.3 Wyświetlenie pamięci wskazywanej przez zawartość rejestru

- przez "przecignięcie" rejestru do okna **Memory**;
- przez menu lokalne (dla dowolnego adresu):
 1. przyciśnięcie prawego przycisku myszy w oknie **Memory** (otwarcie lokalnego menu),
 2. wybranie **Address...** (okno dialogowe **Memory...**),
 3. wpisanie adresu i potwierdzenie **OK**.

5 Operacje na pamięci

5.1 Zmiana formatu wyświetlania zawartości pamięci

1. przyciśnięcie prawego przycisku myszy w oknie **Memory** (otwarcie lokalnego menu),
2. wybranie **Format** (okno dialogowe z dostępnymi formatami),
3. "podwójne pstryknięcie" na wybranym formacie.

Dostępne formaty:

Hex - szesnastkowy,

Oct - osemkowy,

Dec - dziesiętny,

UDec - dziesiętny bez znaku,

Bin - binarny.

5.2 Zmiana zawartości pamięci

”Podwójne pstryknięcie” na wartości komórki pamięci w oknie **Memory** powoduje jej podświetlenie i umożliwienie edycji. Wpisanie wartości powoduje przejście do edycji następnej komórki. Edycję kończy się przez *ENTER* lub *TAB*. Naciśnięcie *ESC* powoduje przywrócenie poprzedniej wartości i wyjście z trybu edycji.

6 Praca z kodem źródłowym

6.1 Deasemblacja wybranej linii kodu źródłowego

- metoda ”przeciągania” (*drag-and-drop*):
 1. zaznaczenie przy pomocy myszy obszaru w oknie **Source**,
 2. ”przeciągnięcie” zaznaczonego obszaru do okna **Assembly**;
- użycie klawisza 'R':
 1. wskazanie instrukcji w oknie **Source**,
 2. naciśnięcie lewego przycisku myszy i klawisza 'R'.

Odpowiednie rozkazy w oknie **Assembly** są podświetlone na szaro.

6.2 Wyświetlanie adresu i kodu deasemblacji

- użycie menu lokalnego:
 1. przyciśnięcie prawego przycisku myszy w oknie **Assembly** (otwarcie lokalnego menu),
 2. wybranie **Display Code** lub **Display Address**;
- użycie menu **Assembly**:
 1. ”pstryknięcie” na pasku tytułowym okna **Assembly** (otwarcie menu **Assembly**),
 2. wybranie **Assembly | Display Code** lub **Assembly | Display Address**.

6.3 Ustawianie pułapek *breakpoints*)

1. wskazanie linii w oknie **Source** lub **Assembly** i przyciśnięcie prawego przycisku myszy (otwarcie lokalnego menu),
2. wybranie **Set Breakpoint** ustawia pułapkę (oznaczoną czerwoną strzałką w linii kodu),
3. aktualnie ustawione pułapki można obejrzeć i zmodyfikować wybierając z lokalnego menu **Show Breakpoints**.

6.4 Usuwanie pułapek i ich przełączanie

1. wskazanie linii w oknie **Source** lub **Assembly**, w której jest ustawiona pułapka i przyciśnięcie prawego przycisku myszy (otwarcie lokalnego menu),
2. wybranie **Delete Breakpoint** usuwa pułapkę,
3. wybranie **Disable Breakpoint** wyłącza pułapkę nie zmieniając jej parametrów,
4. wybranie **Enable Breakpoint** włącza wyłączoną wcześniej pułapkę.

6.5 Zwijanie bloków kodu źródłowego

W celu poprawienia czytelności programów, bloki kodu źródłowego w oknie **Source** można ukrywać (zwijać) i pokazywać (rozwijając) dzięki znacznikom wskazującym ich granice:

- "pstryknięcie" na znaczniku (trójkąt w kwadracie) ograniczającym blok powoduje zwinięcie bloku i zastąpienie go w oknie **Source** znacznikiem w formie kwadratu w kwadracie,
- "pstryknięcie" na znaczniku zwiniętego bloku powoduje jego rozwinięcie.

7 Systemowe skrypty dla HI-WAVE

Pliki: *startup.cmd*, *reset.cmd*, *preload.cmd*, *postload.cmd* są systemowymi skryptami dla debuggera, są automatycznie rozpoznawane i wykonywane:

startup.cmd jest wykonywany po załadowaniu interfejsu systemu docelowego, (*target component*) umieszczonego w *project.ini* lub z menu: **Component | Set Target**,

reset.cmd jest wykonywany przy resetowaniu systemu docelowego (np. **Targrt | Reset**,

preload.cmd jest wykonywany przed załadowaniem aplikacji (w formie *.abs*, lub *.sx*) przy wybraniu np. **Target | Load**,

postload.cmd jest wykonywany po załadowaniu aplikacji (w formie *.abs* lub *.sx*) przy wybraniu np. **Target | Load**.

Inne pliki typu *.cmd* mogą być rozpoznawane i wykonywane dla różnych interfejsów systemów docelowych.