Code Composer Studio v4 (3)

Debugowanie projektu

Debugowanie projektu w środowisku CCSv4 jest wykonywane przez program debuggera będący elementem środowiska. Debugger pracuje na komputerze PC i łączy się z układem sprzętowym nazywanym emulatorem sprzętowym. Połączenie typowo wykorzystuje łącze USB. Emulator jest dołączony do łącza JTAG docelowego układu scalonego (układu procesorowego).

Dotychczas w EP na temat CCS
EP9/2011 „Code Composer Studio v4 – zintegrowane środowisko projektowe”
EP10/2011 „Code Composer Studio v4 – praca z projektem”

Dodatkowe informacje:
Strona produktu *http://www.ti.com/ccs*
Bazowa strona dokumentacji *http://processors.wiki.ti.com/index.php/CCSv4*
Pomoc środowiska CCSv4 z menu Help → Help Contents.

Dodatkowe materiały na CD/FTP:
ftp://ep.com.pl, user: 15352, pass: 760hp6s5
• poprzednie części artykułu

Debugowanie projektu wymaga zdefiniowania konfiguracji sprzętowej systemu docelowego (*target configuration setting*). Konfiguracja systemu docelowego jest wpisywana do pliku w formacie XML o rozszerzeniu „ccxml”. Plik może być kopiowany, usuwany i otwierany. Konfiguracja określa niezależnie dwa elementy:

- Połączenie (connection) – typ emulatora lub symulatora.
- Docelowy układ/płytka – typ układu scalonego (układu procesorowego) lub struktura płytki drukowanej.

Interfejs graficzny do definiowania konfigurowania systemu docelowego jest obecnie częścią środowiska CCSv4. Użytkownik może sam określić kombinację emulatora i układu procesorowego.

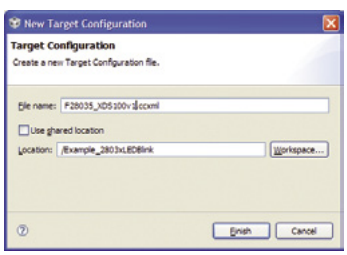
Plik konfiguracji systemu docelowego dla projektu może być:

- Wpisany (dodany) do foldera projektu.
- Dołączony (bez przepisywania) do projektu z lokalizacji domyślnej.

Tworzenie nowej konfiguracji sprzętowego systemu docelowego

Rozpoczęcie tworzenia nowej konfiguracji systemu docelowego można wykonać na kilka sposobów:

- Wybór z menu opcji *Target* → *New Target Configuration*.
- Wybór z menu opcji *File* → *Target* → *New Target Configuration*.
- Kliknięcie prawym klawiszem myszy na linię projektu w oknie *C/C++ Projects*. Z menu kontekstowego należy wybrać pozycję *New*, a następnie *Target Configuration File*.



Rysunek 1. Wybór nazwy i foldera dla pliku konfiguracji systemu docelowego

Otwierane jest wtedy okno edycyjne *New Target Configuration* (**rysunek 1**). W polu *File Name* należy wpisać nazwę dla pliku konfiguracji systemu docelowego z rozszerzeniem „ccxml”. Najlepiej użyć nazwy dobrze opisującej zastosowany układ sprzętowy.

W polu *Location* należy podać miejsce umieszczenia pliku. Można wskazać dwa różne miejsca umieszczenia pliku konfiguracji:

- Lokalizacja pliku konfiguracji we wspólnym (domyślnym) folderze – z wybraną opcją *Use shared location*.
- Lokalizacja pliku konfiguracji w folderze projektu – przy wyłączeniu opcji *Use shared location*. Plik będzie dodany do projektu.

Plik konfiguracji systemu docelowego musi być określony dla każdego indywidualnego projektu środowiska CCSv4.

Lokalizacja pliku konfiguracji w folderze projektu

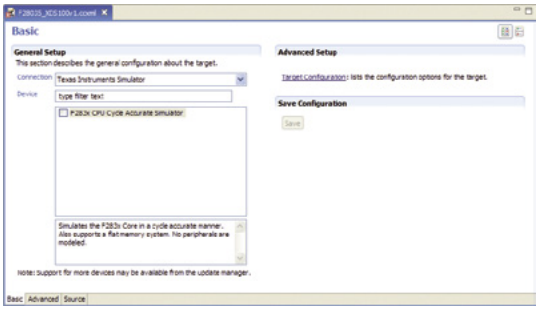
W oknie *New Target Configuration* należy wyłączyć (odznaczyć) opcję *Use shared location* (**rysunek 1**). Następnie należy kliknąć na przycisk *Browse*. W otworzonym oknie pokazywany jest folder, w którym umieszczony jest projekt. Wystarczy kliknąć na przycisk *OK*.

W oknie *New Target Configuration* wybrana jest ścieżka bieżącego projektu (rys. 1). Wszystkie elementy potrzebne do pracy z projektem znajdują się wtedy w tym samym folderze. Po wcześniejszym wpisaniu odpowiedniej nazwy pliku *.ccxml w oknie *New Target Configuration* i ustawieniu ścieżki należy kliknąć na przycisk *Finish*.

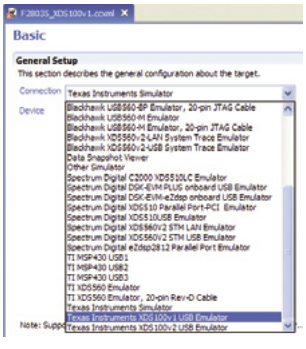
Wybór typu emulatora sprzętowego i układu procesorowego

Otwierane jest okno edycyjne zatytułowane tak jak nazwa utworzonego pliku konfiguracji systemu docelowego (**rysunek 2**).

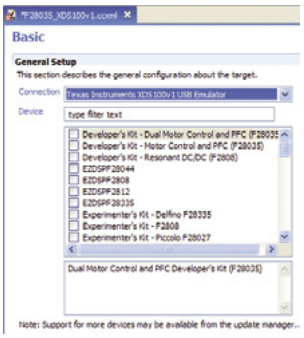
Okno jest zorganizowane jako trzy zakładki. Na zakładce *Basic* w panelu *General Setup* należy wybrać odpowiednie opcje. Najpierw na liście rozwijanej pola *Connection* trzeba wybrać typ emulatora, np. Texas Instruments XDS100v1 USB Emulator (**rysunek 3**). Następnie należy poczekać do chwili, gdy w oknie *Device* zostanie wyświetlona lista układów procesorowych i płytek uruchomieniowych (**rysunek 4**). Teraz trzeba suwakiem przewinąć listę w oknie *Device*. Właściwie jest to lista sprzętowych systemów docelowych, więc jeden typ układu procesorowego może występować wielokrotnie, np. samodzielnie lub jako element płytki eZdsp. Po znalezieniu nazwy odpowied-



Rysunek 2. Okno edycyjne pliku konfiguracji systemu docelowego środowiska CCSv4 (ustawienia domyślne)



Rysunek 3. Wybór emulatora sprzętowego w oknie edycyjnym konfiguracji systemu docelowego środowiska CCSv4



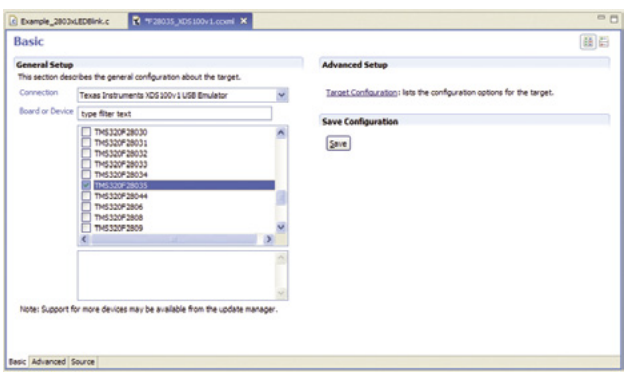
Rysunek 4. Lista sprzętowych systemów docelowych w oknie edycyjnym konfiguracji systemu docelowego środowiska CCSv4

niego układu procesorowego należy kliknąć (zaznaczyć) na jeden (tylko) kwadracik na lewo od wybranego elementu. Następnie należy odczekać, aż linia zostanie zaznaczona na ciemno (**rysunek 5**).

Na zakładce *Advanced* można zobaczyć (i zmienić) szczegółowe ustawienia konfiguracji. Zakładka *Source* umożliwia oglądanie (i edycję) wygenerowanego pliku XML. Na koniec należy na zakładce *Basic* kliknąć na przycisk *Save*. Należy poczekać na zakończenie zapisywania. Wtedy przycisk zostanie wyświetlony w szarym kolorze.

Tylko jeden plik konfiguracji systemu docelowego związany z projektem może zostać ustawiony (i oznaczony) jako aktywny (*Active*). Dodany nowy plik konfiguracji systemu docelowego jest automatycznie ustawiany jako aktywny. Można zmieniać wybór pliku aktywnego w oknie *C/C++ Project*. Należy kliknąć na nazwę pliku konfiguracyjnego prawym klawiszem myszy i wybrać opcję *Set as Active Target Configuration*. Konfiguracja systemu docelowego aktywna jest używana domyślnie po wydaniu polecenia *Target → Debugg Active Project*.

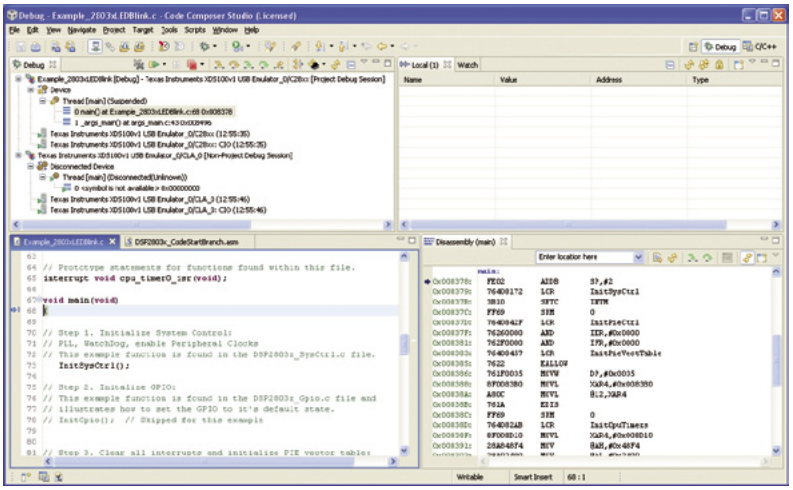
Tylko jeden plik konfiguracji systemu docelowego związany z projektem może zostać ustawiony jako domyślny (*Default*). Można zmieniać wybór pliku domyślnego w tym oknie. Należy kliknąć na nazwę pliku konfiguracyjnego prawym klawiszem myszy i wybrać opcję *Set as Active Default Configuration*.



Rysunek 5. Okno edycyjne pliku konfiguracji systemu docelowego środowiska CCSv4 po wybraniu emulatora sprzętowego i typu układu procesorowego

Tabela 1 Polecenia (najważniejsze) rozpoczęcia/zakończenia (sesji) debugowania

Ikona	Nazwa	Opis	Umiejscowienie
	New Target Configuration	Tworzenie nowego pliku konfiguracji sprzętowej systemu docelowego	File New Menu Target Menu
	Debug Active Project	Uruchomienie sesji debugowej z użyciem aktywnego (active) projektu	Debug Toolbar Target Menu
	Launch TI Debugger	Uruchomienie debuggera z użyciem domyślnej (target) konfiguracji systemu docelowego	Debug Toolbar Target Menu
	Debug	Uruchomienie sesji debugowej z użyciem bieżącej konfiguracji. Przycisk ze strzałką otwiera listę wyboru	Debug Toolbar Target Menu
	Connect Target	Dołączenie debuggera do układu procesorowego	TI Debug Toolbar Target Menu Debug View Context Menu
	Terminate All	Zamyka wszystkie sesje debugowe	Target Menu Debug View Toolbar



Rysunek 6. Okno perspektywy Debug środowiska CCSv4 po wykonaniu w perspektywie C/C++ polecenia *Debug Active Project* (budowanie projektu, uruchomienie debuggera, załadowanie kodu)

Jeden plik może być wybrany jako *Active/Default*. Konfiguracja systemu docelowego oznaczona jako domyślna jest używana po wydaniu polecenia *Target → Launch TI Debugger*.

Uruchomienie debuggera

Wykonanie w perspektywie C/C++ polecenia z menu *Target → Lunch TI Debugger* powoduje uruchomienie debuggera i otworzenie perspektywy *Debug*. Można teraz ręcznie dołączyć układ procesorowy (system docelowy) poleceniem *Connect Target*. Można również ręcznie załadować (zaprogramować) kod do pamięci układu procesorowego.

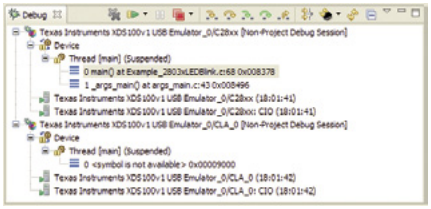
Zestawienie poleceń rozpoczęcia/zakończenia (sesji) debugowania jest zamieszczone w **tabeli 1**. Wykonanie w perspektywie C/C++ polecenia *Debug Active Project* (menu *Target → Debug Active Project*) lub przyciśnięcie przycisku *Debug* powoduje automatyczne rozpoczęcie sesji debugowania. Powoduje to wykonanie budowania inkrementacyjnego aktywnego projektu, uruchomienie debugera, automatyczne dołączenie debugera do docelowego układu procesorowego (systemu), załadowanie kodu wynikowego (programu) do układu procesorowego i otworzenie (przełączenie) perspektywy *Debug*.

Środowisko CCSv4 również wspiera uruchamianie programów z zastosowaniem symulatora oraz systemu operacyjnego czasu rzeczywistego DSP/BIOS.

Perspektywa Debug

Standardowa perspektywa *Debug* środowiska CCSv4 jest automatycznie otwierana po uruchomieniu debuggera (**rysunek 6**). Po dołączeniu debugera do układu procesorowego oraz po załadowaniu do układu procesorowego kodu programu typowo otwierane jest okno *Debug*, okna danych *Local* i *Watch* oraz okno edytora z kodem źródłowym pliku zawierającego funkcję *main()*. Przy uruchamianiu projektu kolejny raz mogą być również otwierane inne okna, np. *Disassembly*.

Po lewej stronie okna edycyjnego jest pasek wyboru (*Selection Margin*). Na pasku wyboru pokazywana jest kolorowa strzałka, gdy załadowany jest plik wynikowy programu do



Rysunek 7. Okno *Debug* perspektywy *Debug* w środowisku CCSv4 po załadowaniu kodu

pamięci układu procesorowego i wykonywanie programu jest zatrzymane w obrębie kodu pokazywanego w oknie. Jest ona powiązana z zawartością licznika rozkazów PC. Wskazuje na linię kodu języka C związanego z instrukcją (asemblerową) spod adresu w PC.

Dołączanie/odłączanie sprzętu do CCSv4

CCSv4 umożliwia dynamiczne dołączanie i odłączanie docelowego układu procesorowego (systemu). Na początku pracy CCSv4 jest domyślnie odłączony od układu procesorowego. Pozwala to na pracę środowiska bez fizycznej obecności układu procesorowego. Uruchomienie debuggera środowiska CCSv4 (np. z menu *Target* → *Lunch TI Debugger*) nie powoduje automatycznego dołączenia do docelowego układu procesorowego.

Dołączenie CCSv4 do docelowego układu procesorowego jest możliwe po wcześniejszym zdefiniowaniu konfiguracji docelowego systemu sprzętowego. Oznacza to określenie typu emulatora sprzętowego dołączonego do komputera PC oraz typu układu procesorowego docelowego dołączonego do emulatora poprzez łącze JTAG.

Dołączenie do docelowego układu procesorowego jest wykonywane po kliknięciu na przycisk *Connect Target* na pasku narzędzi perspektywy *Debug*. Ponowne kliknięcie przycisku, nazywanego obecnie *Disconnect Target,* powoduje odłączenie od docelowego układu procesorowego.

Po poprawnym wykonaniu polecenia *Connect* środowisko CCSv4 komunikuje się z procesorem docelowym. Automatycznie otwierane jest okno *Debug* oraz okno deasemblacji *Disassembly*.

Okno *Debug*

Okno *Debug* jest dostępne w perspektywie *Debug* środowiska CCSv4 (**rysunek 7**). W oknie jest pokazywany stan stosu dla aktualnie debugowanego układu procesorowego. Są one omówione w **tabeli 2**. Po zatrzymaniu działania programu jest pokazywana w perspektywie *Debug* zawartość okien dla ramki umieszczonej na szczycie stosu. Można także oglądać stan dla innych ramek stosu. Dla procesorów wielordzeniowych, jak TMS320F28035 Piccolo, w oknie *Debug* pokazywany jest stan obu rdzeni (rys. 7). Na pasku narzędzi okna *Debug* są dostępne podstawowe przyciski wykonywania programu.

Tabela 2 Polecenia (najważniejsze) ładowania programu/symboli

Ikona	Nazwa	Opis	Umiejscowienie
	Load Program	Ładowanie kodu wynikowego budowania projektu z pliku typu COFF (*.out) do układu procesorowego (lub symulatora)	Target Menu Debug Toolbar Modules View Toolbar
	Load Symbols	Ładowanie symboli dla pliku wynikowego umieszczonego w pamięci nieulotnej typu Flash lub ROM. Powoduje wyczyszczenie tablicy symboli i załadowanie nowej zawartości	Target Menu Modules View Toolbar
	Reload Program	Ponowne załadowanie kodu wynikowego budowania projektu z pliku typu COFF (*.out) do układu procesorowego (lub symulatora)	Target Menu Debug Toolbar Modules View Toolbar

Tabela 3. Polecenia (najważniejsze) wykonania programu

Ikona	Nazwa	Opis	Umiejscowienie
	Halt	Zatrzymanie wykonywania programu przez układ procesorowy. Okna perspektywy *Debug* są automatycznie odświeżane przy zastosowaniu pobranych nowych danych.	Target Menu Debug View Toolbar
	Run	Wznowienie wykonanie programu, który jest aktualnie załadowany do pamięci układu procesorowego od bieżącego stanu licznika rozkazów PC. Wykonywanie programu jest kontynuowane, aż do napotkania pułapki	Target Menu Debug View Toolbar
	Run to Line	Wznowienie wykonanie programu, który jest aktualnie załadowany do pamięci układu procesorowego od bieżącego stanu licznika rozkazów PC. Wykonywanie programu jest kontynuowane aż do osiągnięcia wskazanej linii kodu źródłowego/asemblerowego	Target Menu Disassembly Context Menu Source Editor Context Menu
	Go to Main	Uruchamia program do początku funkcji main().	Debug View Toolbar
	Step Into	Praca krokowa poziomu kodu źródłowego C/C++ z wchodzeniem do zaznaczonej instrukcji (funkcji) i zatrzymaniem wykonania	Target Menu Debug View Toolbar
	Step Over	Praca krokowa poziomu kodu źródłowego C/C++ z wykonaniem zaznaczonej instrukcji (funkcji) i zatrzymaniem wykonania na pierwszej instrukcji po powrocie na ten sam poziom	Target Menu Debug View Toolbar
	Step Return	Wykonanie bieżącej funkcji do końca i zatrzymanie wykonania na pierwszej instrukcji po powrocie na wyższy poziom	Target Menu Debug View Toolbar
	Reset	Przerwanie wykonywania bieżącego programu i wymuszenie operacji RESET układu procesorowego	Target Menu Debug View Toolbar
	Restart	Ustawia licznik rozkazów PC na wartość punktu wejścia programu aktualnie załadowanego do pamięci układu procesorowego. Następnie program jest wykonywany do określonego miejsca, typowo do początku funkcji main()	Target Menu Debug View Toolbar
	Assembly Step Into	Praca krokowa poziomu kodu asemblerowego z wchodzeniem do zaznaczonej instrukcji (funkcji) i zatrzymaniem wykonania	TI Explicit Stepping Toolbar Target Advanced Menu
	Assembly Step Over	Praca krokowa poziomu kodu źródłowego C/C++ z wykonaniem zaznaczonej instrukcji (funkcji) i zatrzymaniem wykonania na pierwszej instrukcji po powrocie na ten sam poziom	TI Explicit Stepping Toolbar Target Advanced Menu

Okno *Disassembly*

W oknie deasemblacji (*Disassembly*) jest pokazywany kod asemblerowy (i maszynowy) programu znajdującego się w pamięci układu procesorowego po wykonaniu deasemblacji (rys. 6). Okno deasemblacji może być pokazywane w obu perspektywach *C/C++* oraz *Debug*. W tym oknie pokazywane są informacje symboliczne dla adresów funkcji i zmiennych języka C (o ile są dostępne). W oknie można przejść do dowolnego poprawnego adresu. Deasemblacja zawartości pamięci programu (kodu źródłowego) wykonywana jest dynamicznie podczas zmiany adresu w oknie. Na pasku tytułu okna pokazywany jest najbliższy symbol adresowy i przesuniecie względem niego pierwszej linii pokazywanej w oknie.

Okno deasemblacji można otworzyć z menu *View* → *Disassembly*. Podczas wykonywania programu w pracy krokowej zawartość okna deasemblacji jest dynamicznie aktualizowana.

Po poprawnym wykonaniu polecenia *Connect* środowisko CCSv4 komunikuje się z procesorem docelowym. Automatycznie otwierane jest okno deasemblacji (rys. 6). Pokazywana jest

w nim zawartość kodu od adresu zatrzymania układu procesorowego, czyli od początku kodu bootowania układu procesorowego z pamięci Boot-ROM:
F2802x Piccolo:
```
0x3ff7bb: 28AD0004 MOV @SP,#0x0004
```
F2803x Piccolo:
```
0x3ff8a1: 28AD0004 MOV @SP,#0x0004
```

Kod załadowany do pamięci może być pokazywany w oknie deasemblacji na dwa sposoby. Tryb kodu źródłowego *Source Mode* – pokazywany jest tylko kod maszynowy po deasemblacji. Jest to domyślny tryb pracy.Tryb *Mixed Mode* – pokazywany jest kod źródłowy w języku C lub asemblerowym (o ile kod jest dostępny) oraz jego kod maszynowy po deasemblacji. W oknie deasemblacji są również pokazywane pułapki. Są one identyfikowane ikonkami na lewo od adresu instrukcji. Ich znaczenie jest takie samo jak w oknie edycji.

Ładowanie kodu programu

Ładowanie kodu (load) oznacza otworzenie wskazanego pliku formatu COFF (*.out) będącego rezultatem budowania projektu i wpisanie (załadowanie) go do układu procesorowego docelowego. Ładowanie kodu jest dostępne w konfiguracji *Debug*, czyli po uruchomieniu debuggera i po dołączeniu do systemu docelowego. Zestawienie poleceń ładowania programu/symboli jest zamieszczone w **tabeli 3**. Ładowanie kodu wynikowego programu jest wykonywane automatycznie po wykonaniu polecenia *Debug Active Project* (menu *Target* → *Debug Active Project*). Ładowanie kodu wynikowego programu można wykonać ręcznie z menu *Target* → *Load Program*. Jest też polecenie ponownego załadowania programu *Target* → *Reload Program*.

W zależności od rodzaju pamięci RAM lub Flash, do której ładowany jest kod, operacja wykonywana jest w różny sposób.

Po wydaniu polecenia *Target* → *Load Program* otwierane jest okno wyboru pliku z kodem wynikowym (**rysunek 8**). Przycisk *Browse Project* pozwala na automatyczne ustawienie ścieżki do foldera /*Debug* aktywnego projektu w polu *Program file*. Przycisk *Browse* pozwala na wybranie pliku z tego foldera.

Po przyciśnięciu przycisku *OK* rozpoczyna się automatyczne ładowanie kodu do układu procesorowego. Postęp operacji jest pokazywany w osobnym oknie (**rysunek 9**). Typowo, po wykonaniu polecenia ładowania kodu do pamięci program jest uruchamiany od punktu wejścia języka C (entry point) i wykonywany do początku funkcji main(). Oznacza to wykonanie inicjacji środowiska języka C (rys. 6). Licznik rozkazów (PC) jest ustawiony na pierwszą instrukcję funkcji *main()*. Pokazują to strzałki w oknie *Disassembly* oraz oknie edycyjnym pliku z funkcją *main()*.

Rysunek 8. Okno wyboru pliku z kodem wynikowym po wykonaniu polecenia *Load Program* w środowisku CCSv4

Ładowanie pliku wynikowego do pamięci RAM

Ładowanie kodu do pamięci RAM powoduje wpisanie kodu do pamięci RAM układu procesorowego poprzez jego łącze JTAG. W trakcie ładowania kodu typowo jest wykonywana weryfikacja (domyślnie włączona opcja *Perform verification during Program Load*). Załadowane dane są odczytywane z pamięci i porównywane z oryginałem. Wystąpienie różnic jest sygnalizowane.

Ładowanie pliku wynikowego do pamięci Flash

Jeśli kod wynikowy programu ma zostać umieszczony w wewnętrznej pamięci Flash układu procesorowego, to jest wymagane zaprogramowanie tej pamięci. Środowisko CCSv4 automatycznie rozpoznaje, które sekcje pliku COFF mają zostać umieszczone w pamięci Flash. Programator pamięci Flash zostaje uruchomiony automatycznie.

Operacja programowania pamięci Flash jest wykonywana w trzech krokach: kasowanie, programowanie i weryfikowanie pamięci Flash. Dalej ładowanie przebiega tak samo jak do pamięci RAM. Postęp poszczególnych kroków operacji jest pokazywany w osobnym oknie (rys. 9). Widok tego okna może być bardzo różny w zależności np. od tego, czy jest to pierwsze ładowanie kodu po uruchomieniu sesji debugowej czy następne.

Ustawienia narzędzia do programowania pamięci Flash można oglądać i zmieniać. Należy wybrać z menu *Tools* → *On-Chip Flash*. Otwierane jest okno *Control Panel* z zakładką *On-Chip Flash* (**rysunek 10**).

Zawartość okna jest zależna od używanego w projekcie układu procesorowego (konfiguracja systemu docelowego). Można w nim zmieniać parametry oraz wykonywać dodatkowe operacje jak jednorazowe kasowanie pamięci Flash i programowanie hasła dostępu.

Należy uważać, aby ustawienia nie spowodowały przekroczenia maksymalnej częstotliwości zegara systemowego SYSCLKOUT używanego układu procesorowego.

Ładowanie symboli dla pliku wynikowego umieszczonego w pamięci nieulotnej

Jest również pewna postać ładowania kodu dla pamięci nielotnej typu Flash, ROM lub EEPROM. Jeśli pamięć Flash układu procesorowego została zaprogramowana kodem wynikowym programu, to CCSv4 pozwala na załadowanie danych symbolicznych dla debugowania tego programu. Wtedy środowisko CCSv4 czyta dane debugowe, np. wartość symboli adresowych i są one używane do debugowania. Nie jest wymagane ponowne programowanie pamięci Flash. W celu załadowania danych symbolicznych należy z menu wybrać *Target* → *Load Symbols*. Otwierane jest wtedy typowe okno dialogowe (rys. 8), w którym należy wybrać plik wynikowy *.out (typowo np. w folderze *Debug*). Po wykonaniu tej operacji debugger środowiska CCSv4 uwzględnia informację symboliczną tak jak po zaprogramowaniu pamięci Flash układu procesorowego.

Uruchamianie programu, praca krokowa i polecenie Reset

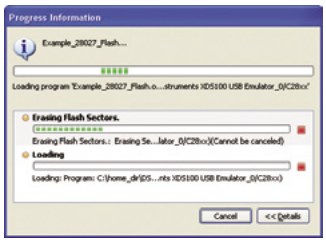
Układy procesorowe rodziny TMS320C2000 mają bardzo rozbudowany sprzętowy układ wspomagania procesu uruchamia kodu. Środowisko CCSv4 umożliwia przeprowadzenie uruchamiania programu na różne sposoby. Do uruchamiania programu służy perspektywa *Debug*. Udostępnia ona wiele poleceń przy użyciu: ikonek na pasku narzędzi perspektywy *Debug*, ikonek na pasku narzędzi okna *Debug*, poleceń w menu kontekstowym okna *Debug* lub poleceń z menu *File* oraz *Target*. Można podzielić polecenia na:

- polecenia rozpoczęcia/zakończenia sesji debugowania,
- polecenia ładowania programu/symboli,
- polecenia wykonania programu.

Uruchamianie programu

Jest kilka poleceń w perspektywie Debug, które dotyczą wykonywania programu (tab. 3):

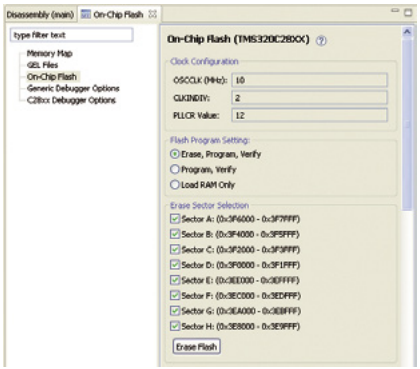
- *Target* → *Reset* → *Reset CPU* – Polecenie powoduje przerwanie wykonywania bieżącego programu i wymuszenie operacji RESET układu procesorowego. Polecenie jest dostępne po dołączeniu debuggera do układu procesorowego. Jest to wykonywane poprzez łącze JTAG i nie do końca jest realizowane tak samo jak RESET sprzętowy. Nie są np. zmieniane ustawienia modułu PLL. Typowo układ procesorowy jest następnie zatrzymywany na pierwszej instrukcji programu BOOT-loadera .
- *Target* → *Reset* → *Reset Emulator* – Polecenie jest dostępne po dołączeniu emulatora do komputera bez konieczności dołączania debuggera do układu procesorowego.
- *Target* → *Restart* – Polecenie typowo jest używane po poleceniu *Reset CPU*. Jego działanie jest określone w opcji *Auto Run* debuggera. Typowo powoduje ono uruchomienie układu procesorowego, wykonanie kodu inicjacji środowiska języka C, czyli od adresu c_int00 i zatrzymanie wykonania na początku funkcji *main()*. Zatrzymanie jest realizowane jako pułapka programowa.
- *Target* → *Run* – Polecenie powoduje uruchomienie układu procesorowego po jego poprzednim zatrzymaniu. Układ procesorowy pracuje, aż do ponownego zatrzymania wykonanego automatycznie



Rysunek 9. Okno postępu programowania pamięci Flash (faza kasowania) po wykonaniu polecenia *Load Program* dla projektu skonfigurowanego do pracy z pamięcią Flash układu procesorowego w środowisku CCSv4

po uauktywnieniu się pułapki lub po wykonaniu polecenia *Halt*.

- *Target → Halt* – Polecenie powoduje zatrzymanie układu procesorowego.
- *Target → Go to Main* – Polecenie typowo jest używane po poleceniu *Reset*. Powoduje ono uruchomienie układu procesorowego i wykonanie całej funkcji inicjacji środowiska języka C. Układ procesorowy jest następnie zatrzymywany na pierwszej instrukcji funkcji *main()*.
- *Target → Run to Line* – Polecenie powoduje uruchomienie układu procesorowego i wykonanie kodu aż do miejsca kodu w którym został umieszczony kursor.



Rysunek 10. Okno *Control Panel* z zakładką *On-Chip Flash* środowiska CCSv4 dla układu procesorowego TMS320F28035 (fragment)

- *Set PC to Cursor* – Polecenie dostępne po kliknięciu prawym klawiszem myszy na linię kodu w oknie edycyjnym z otwartym kodem źródłowym. Z podręcznego menu należy wybrać pozycję *Set PC to Cursor*. To samo można wykonać w oknie *Disassembly*. Polecenie powoduje ustawienie licznika rozkazów układu procesorowego (PC) na adres kodu w miejscu, na którym został umieszczony kursor. Pozwala to na uruchomienie wykonania kodu od tego miejsca. Uwaga: Polecenie jest bardzo przydane do powtarzania wykonania kawałka kodu bez konieczności ponownego uruchamiania układu procesorowego. Bardzo to ułatwia przeprowadzanie prób zachowania się fragmentu kodu z bieżącą zawartością zmiennych. Trzeba jednak pamiętać, aby nie zaburzyć struktury wywoływania funkcji. Czyli jest to przydatne raczej w ramach jednego kontekstu (ramki stosu) języka C.

Praca krokowa

Środowisko CCSv4 pozwala na wykonywanie pracy programu w sposób krokowy. Są dwa sposoby wykonywania pracy krokowej:

- Dla kolejnych instrukcji (linii kodu źródłowego) języka C.
- Dla kolejnych instrukcji (linii kodu maszynowego) języka asemblerowego.

Praca krokowa na poziomie asemblerowym może być obserwowana jednocześnie w oknie edycyjnym pliku źródłowego języka C/C++ oraz, oknie deasemblacji. Na pasku tytułu okna deasemblacji podana jest nazwa pliku źródłowego, którego dotyczy kod asemblerowy.

Dodatkowo dla każdego z tych sposobów wykonywania pracy krokowej można:

- Wykonać pracę krokową z wchodzeniem do funkcji i zatrzymanie wykonania na jej pierwszej instrukcji (*Step Into*, *Assembly Step Into*).
- Wykonać wykonanie funkcji i zatrzymanie na pierwszej instrukcji po powrocie (*Step Over*, *Assembly Step Over*).
- Wykonać operację wyjścia *Step Out* powodującą wykonanie kodu bieżącej funkcji do końca i zatrzymanie wykonania na pierwszej instrukcji po powrocie na wyższy poziom.

Pułapki

Pułapki są podstawowym elementem każdego środowiska do debugowania kodu programu. Pułapka podejmuje akcję, gdy określone dla niej warunki zostaną spełnione. Debugger środowiska CCSv4 ma kilka typów pułapek. Ich dostępność zależy od dołączonego rodzaju układu procesorowego.

Okna podglądu, pamięci, rejestrów, stosu i symboli

Wiele okien w perspektywie *Debug* środowiska CCSv4 włącza się automatycznie przy uruchomieniu sesji debuggowej, np. okno *Debug*, panel podglądu zmiennych z oknami *Watch* i *Local(1)* oraz okno edycyjne z kodem funcji *main()*. Włączanie pozostałych okien jest wykonywane poprzez wybór pozycji z menu *View*. Są to: okno pamięci *Memory*, okno rejestrów *Register*, okno przerwań *Breakpoints* i okno deasemblacji *Disassembly*.

Wizualizacja danych

Środowisko CCSv4 udostępnia możliwość graficznej prezentacji danych. Podstawowe typy okien prezentacji graficznej to: Single Time, Dual Time, FFT Magnitude, Complex FFT, FFT Magnitude & Phase oraz FFT Waterfall. Okno wizualizacji można otworzyć poprzez wybór z menu *Tools->Graph*, a następnie wybrać określony typ okna prezentacji. Otwierane jest wtedy okno dialogowe *Graph Properties*. Należy w tym oknie ustawić odpowiednie parametry.

Aktualne informacje dotyczące debugowania w środowisku CCSv4 znajdują się w pomocy (Help) tego środowiska.

Henryk A. Kowalski, EP
henryk.kowalski@ep.com.pl