

Wprowadzenie do Code Composer Studio v.12

Pierwszy projekt z wykorzystaniem modułu Beaglebone AI

Kolejne kroki prowadzące do włączenia zasilania modułu Beaglebone AI (BB AI), utworzenia oraz uruchomienia pierwszego projektu w środowisku Code Composer Studio (CCS) są opisane poniżej w kolejnych punktach. Krótka charakterystyka modułu BB AI dostępna jest na stronie:

<https://beagleboard.org/boards/beaglebone-ai>

a krótka charakterystyka środowiska CCS na stronie:

<https://www.ti.com/tool/CCSTUDIO>.

Podręcznik użytkownika (User's Guide) – w dalszej części niniejszej instrukcji nazywany w skrócie Podręcznikiem CCS – dla ostatniej wersji CCS jest dostępny na stronie:

https://software-dl.ti.com/ccs/esd/documents/users_guide/index.html

Podręcznik CCS dostępny jest również w aplikacji CCS w menu Help -> 'Help Contents' -> 'Code Composer Studio User's Guide'.

1. Włączenie do pracy modułu BB AI

Moduł BB AI jest umieszczony w jednej obudowie z przezroczystej plexi wraz z emulatorem XDS200, kodekiem audio WM8960 oraz złączami BNC na płycie czołowej. Całość połączona jest na stałe dwoma kablami USB z komputerem PC. Jeden z tych kabli jest wykorzystywany do zasilania BB AI i kodeka z portu USB i na tym kablu znajduje się elektroniczny przełącznik zasilania z timerem. Drugi kabel służy do połączenia emulatora XDS200 z komputerem PC.

UWAGA: Oba kable USB modułu podłączone są na stałe do komputera i nie wolno ich rozłączać. Stałe połączenie zapewnia pewną wspólną masę, co zapobiega możliwemu uszkodzeniu któregoś z podzespołów spowodowanym stanami przejściowymi przy rozłączaniu i włączaniu kabli do gniazd USB w komputerze. **Tylko prowadzący może wyjmować kable i je ponownie podłączać do portów USB. Włączanie i wyłączanie zasilania modułu odbywa się przełącznikiem elektronicznym na zasilającym kablu USB.**

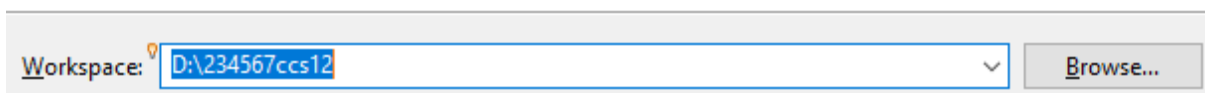
W pierwszym kroku należy włączyć zasilanie przełącznikiem elektronicznym na kablu USB i ustawić timer np. na 2 lub 3 godziny (2h lub 3h). W czasie pracy można wydłużyć ten czas o kolejne godziny (na końcu zajęć wyłączenie modułu odbywa się tym samym wyłącznikiem po zamknięciu sesji 'Debug' i środowiska CCS). Po włączeniu zasilania i podstawowej konfiguracji modułu przez oprogramowanie startowe, na zakończenie zaświeci się dioda D2 (ok. 3 sekundy od włączenia zasilania).

2. Uruchomienie CCS i wybór katalogu roboczego (workspace)

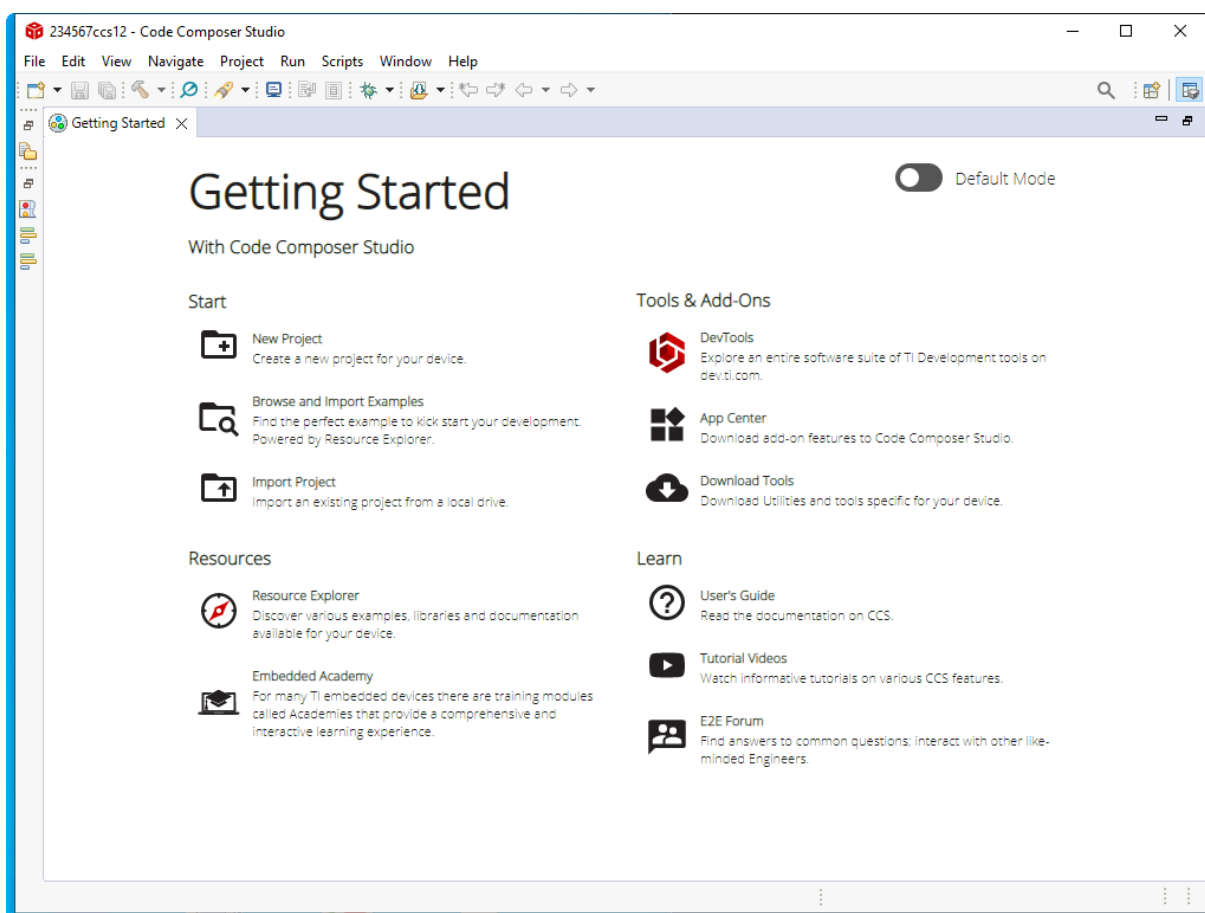
Po uruchomieniu CCS pojawia się pytanie o ścieżkę do katalogu roboczego, gdzie wpisujemy dysk D:\ i jako nazwę katalogu własny numer indeksu z dopiskiem ccs12:

Select a directory as workspace

Code Composer Studio uses the workspace directory to store its preferences and development artifacts.



Potwierdzamy wybór przyciskiem Launch i po dłuższej chwili pojawia się okno główne CCS:



W tym momencie, jeszcze przed utworzeniem pierwszego projektu, workspace zawiera następujące katalogi:

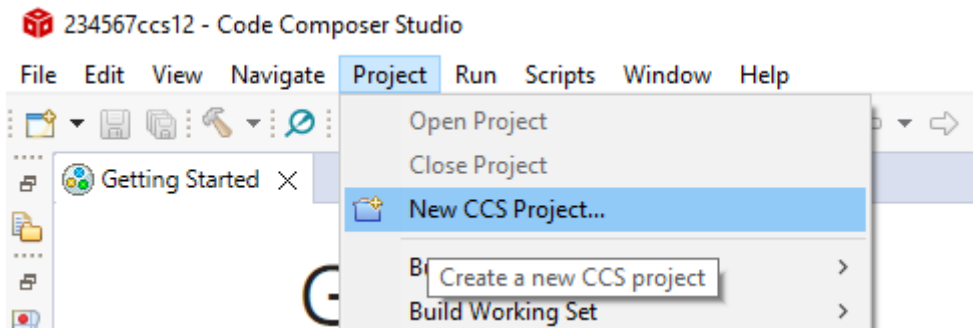
.jxbrowser.userdata
.dvt

.metadata
RemoteSystemsTempFiles

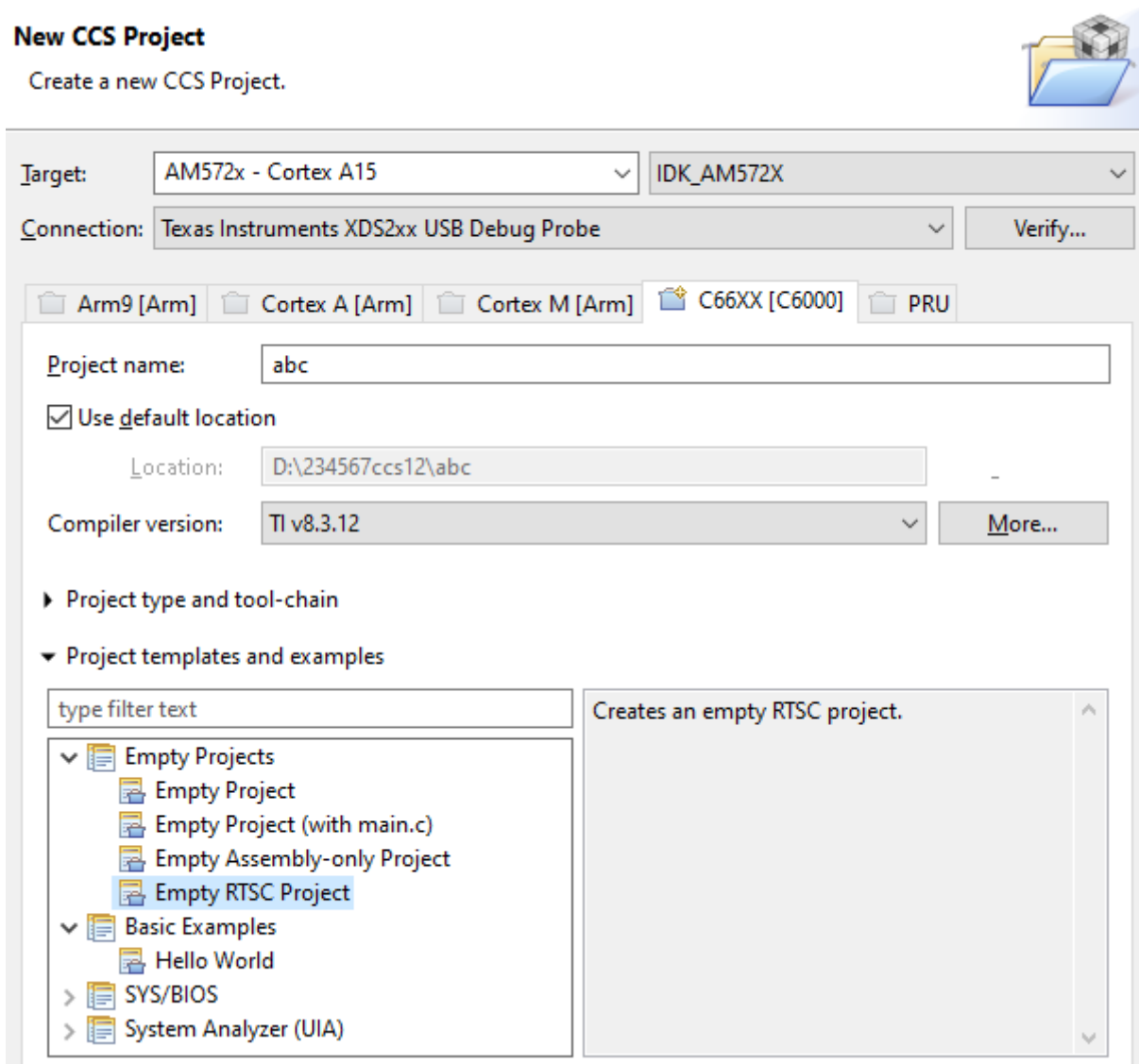
Zawartość katalogu .jxbrowser.userdata zawiera ok. 20 MB danych, katalog .metadata ok. 0.5MB, pozostałe dwa katalogi są puste.

3. Tworzenie nazwy pierwszego projektu, deklaracja podstawowych danych projektu, prosty program i kompilacja

Wybieramy 'New CCS Project' (jeśli projekt już jest zdefiniowany wybieramy 'Open Project'):



Wybór podstawowych danych projektu jest następujący:



Nazwa projektu (tutaj 'abc') jest jednocześnie nazwą podkatalogu w bieżącym katalogu workspace. Wybieramy 'Next' i pojawia się okno z informacją o dostępnych modułach programowych środowiska CCS. W sekcji 'XDCtools settings' wpisujemy w pozycji 'Platform' nazwę ti.platforms.evmAM572X (ważne aby zwrócić uwagę na wielkość liter – małe/duże – w nazwie):

Products

Edit product selections and XDCtools configuration settings



Target:

Connection:

C66XX [C6000]

Products and repositories

- Products
 - XDCtools [3.55.2.22_core]
 - am57xx OpenMP [2.6.3.00]
 - am57xx PDK [1.0.19]
 - CTools Library [2.2.0.00]
 - EDMA3 Low Level Driver [2.12.5]
 - Framework Components [3.40.2.07]
 - IMGLIB C66x [3.1.1.0]
 - IPC [3.50.4.08]
 - SYS/BIOS [6.76.3.01]
 - System Analyzer (UIA Target) [2.30.1.02]
 - XDAIS [7.24.0.04]
- XDCpath Package Repositories
 - \${COM_TI_OMP_INSTALL_DIR}/packages
 - \${COM_TI_PDK_INSTALL_DIR}/packages
 - \${COM_TI_CTOOLSLIB_INSTALL_DIR}/packages
 - \${COM_TI_EDMA3_LLD_INSTALL_DIR}/packages

Buttons: Add..., Edit..., Expand, Exclude, Remove, Up, Down, Details...

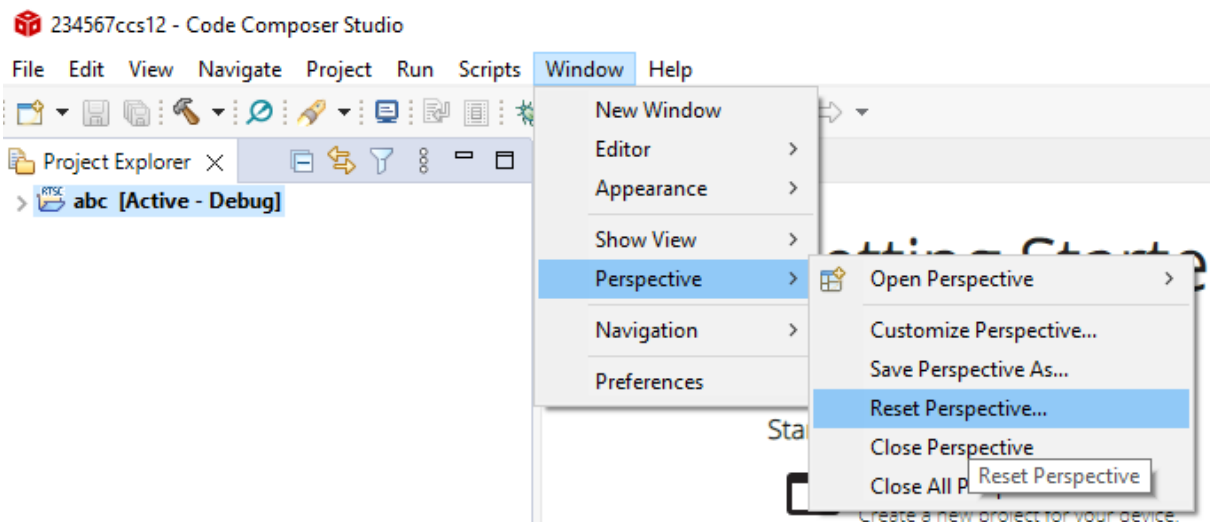
XDCtools settings

Target:

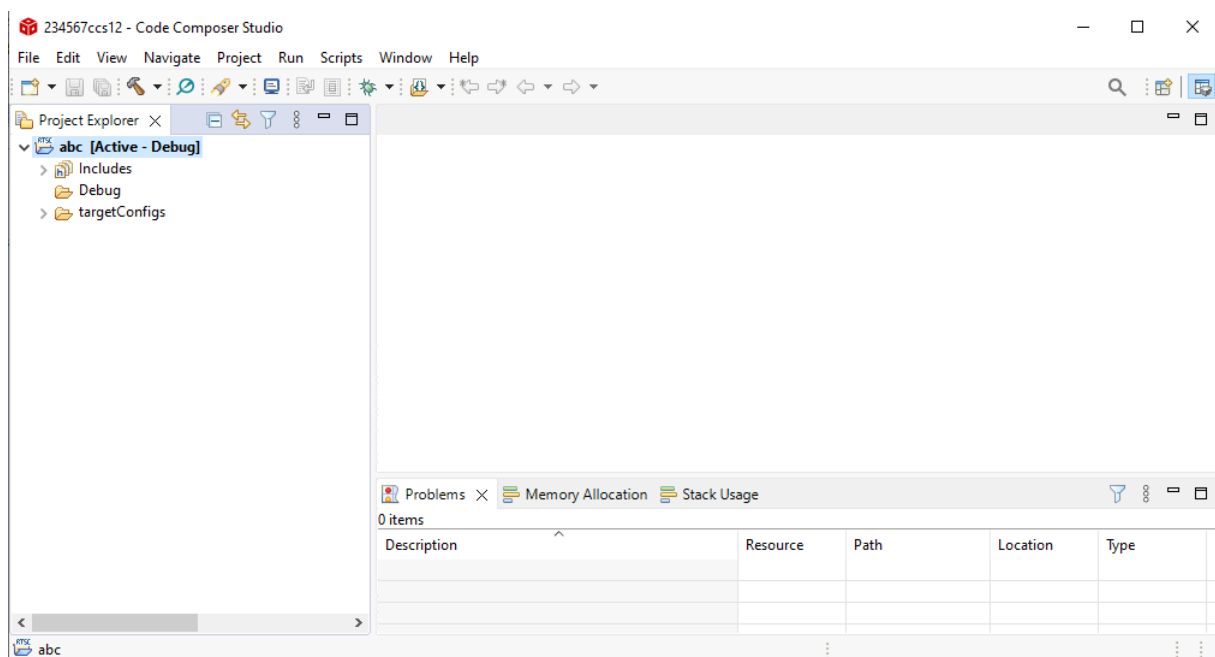
Platform:

Build-profile:

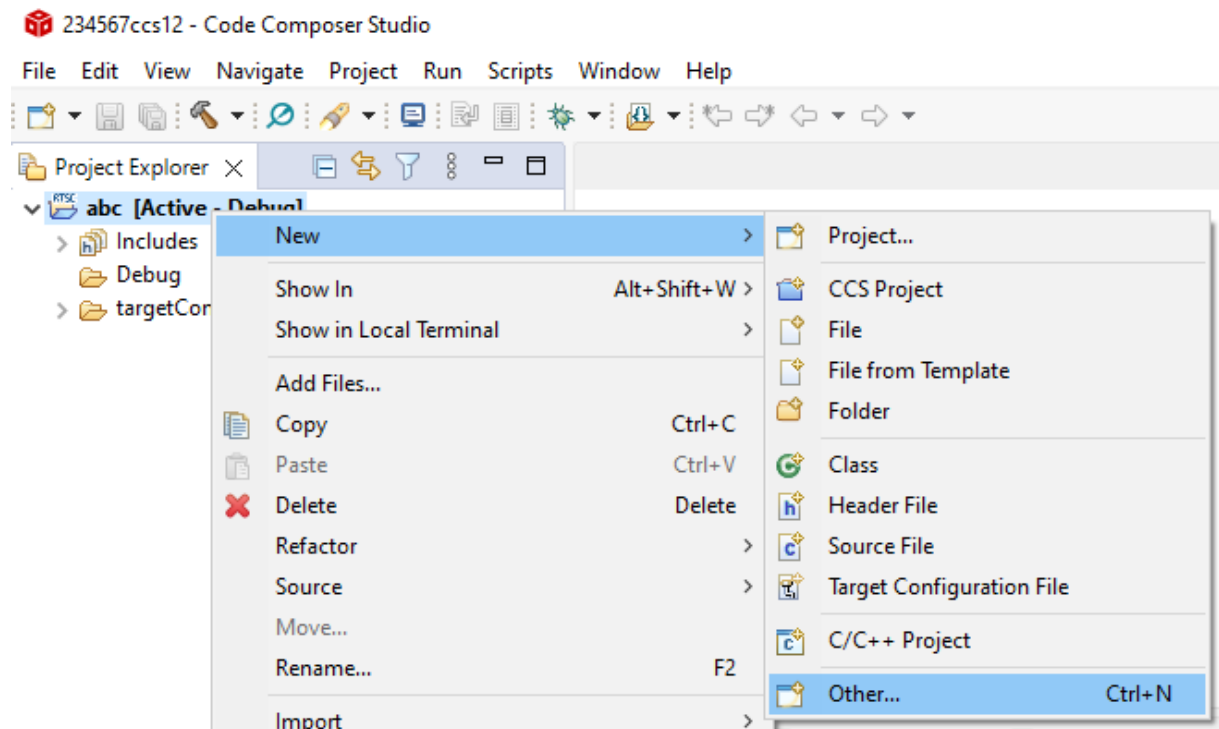
Wybieramy 'Finish' i pojawia się podstawowy widok ('Perspective') typu 'Edit'. Każdy z ćwiczących może organizować własny układ takiego widoku, ale zawsze można powrócić do widoku standardowego poprzez reset widoku:



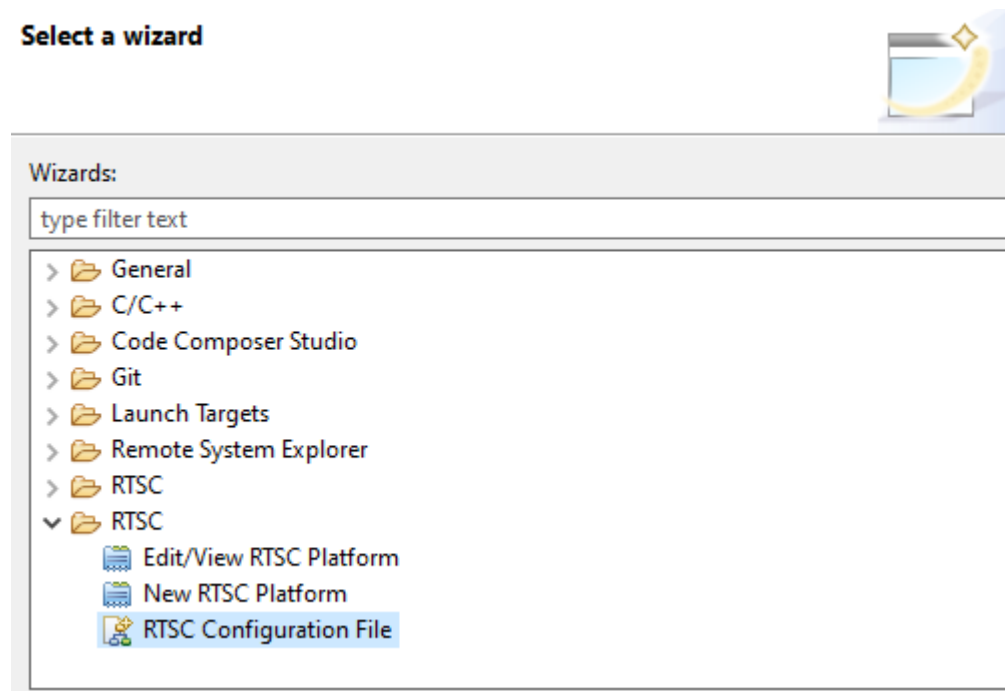
Po wybraniu 'Reset Perspective' widok 'Edit' zawiera oprócz menu trzy pola, w tym 'Project Explorer' po lewej z listą projektów w workspace (tutaj tylko projekt 'abc'). Tylko jeden projekt jest aktywny ('Active – Debug'):



Definiujemy nowy plik konfiguracyjny RTOS (poprzez prawy klawisz myszy):

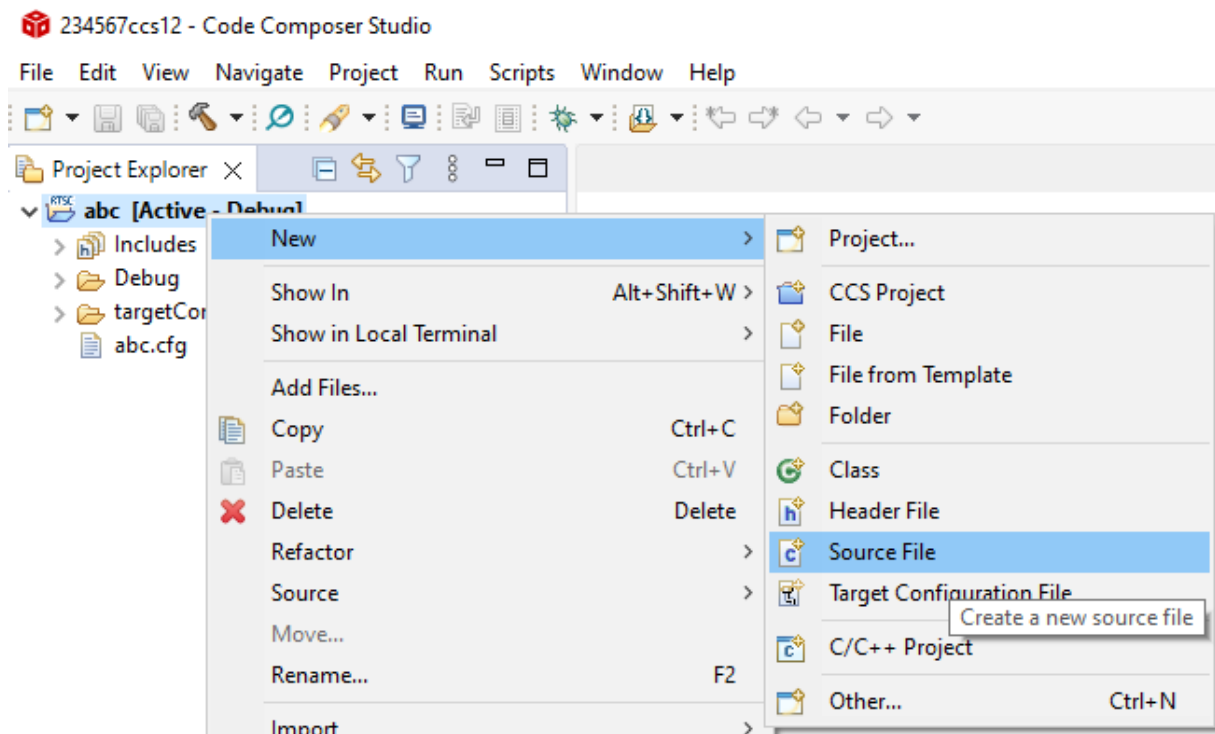


Select a wizard



Wybieramy 'Next' i 'Finish'. Plik konfiguracyjny RTOS z rozszerzeniem cfg pojawia się na liście w 'Project Explorer'.

Analogicznie deklarujemy nowy plik źródłowy dla języka C (zwrócić uwagę na zmianę template z 'Default C++...' na 'Default C...'):



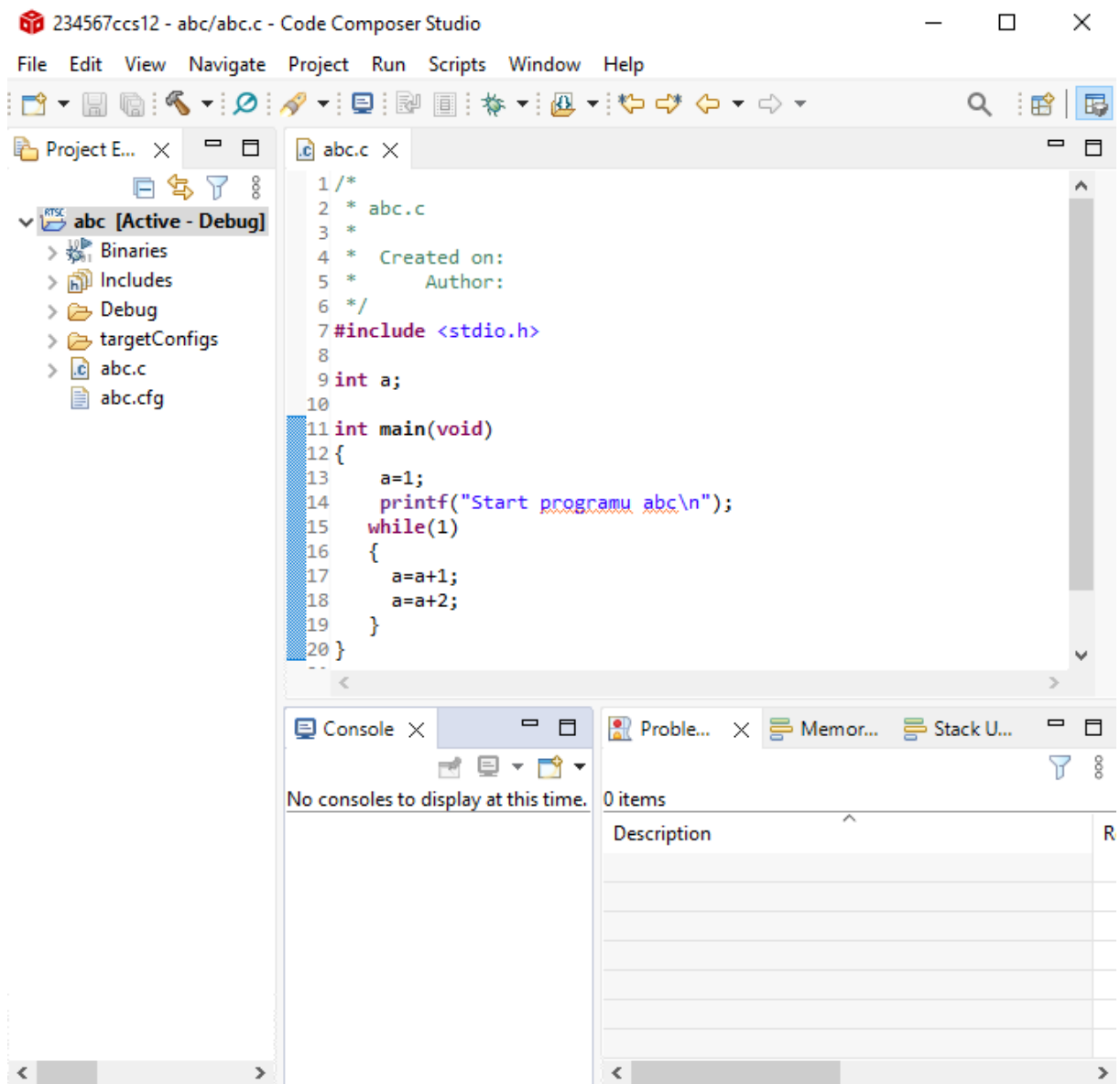
Source File

Create a new source file.

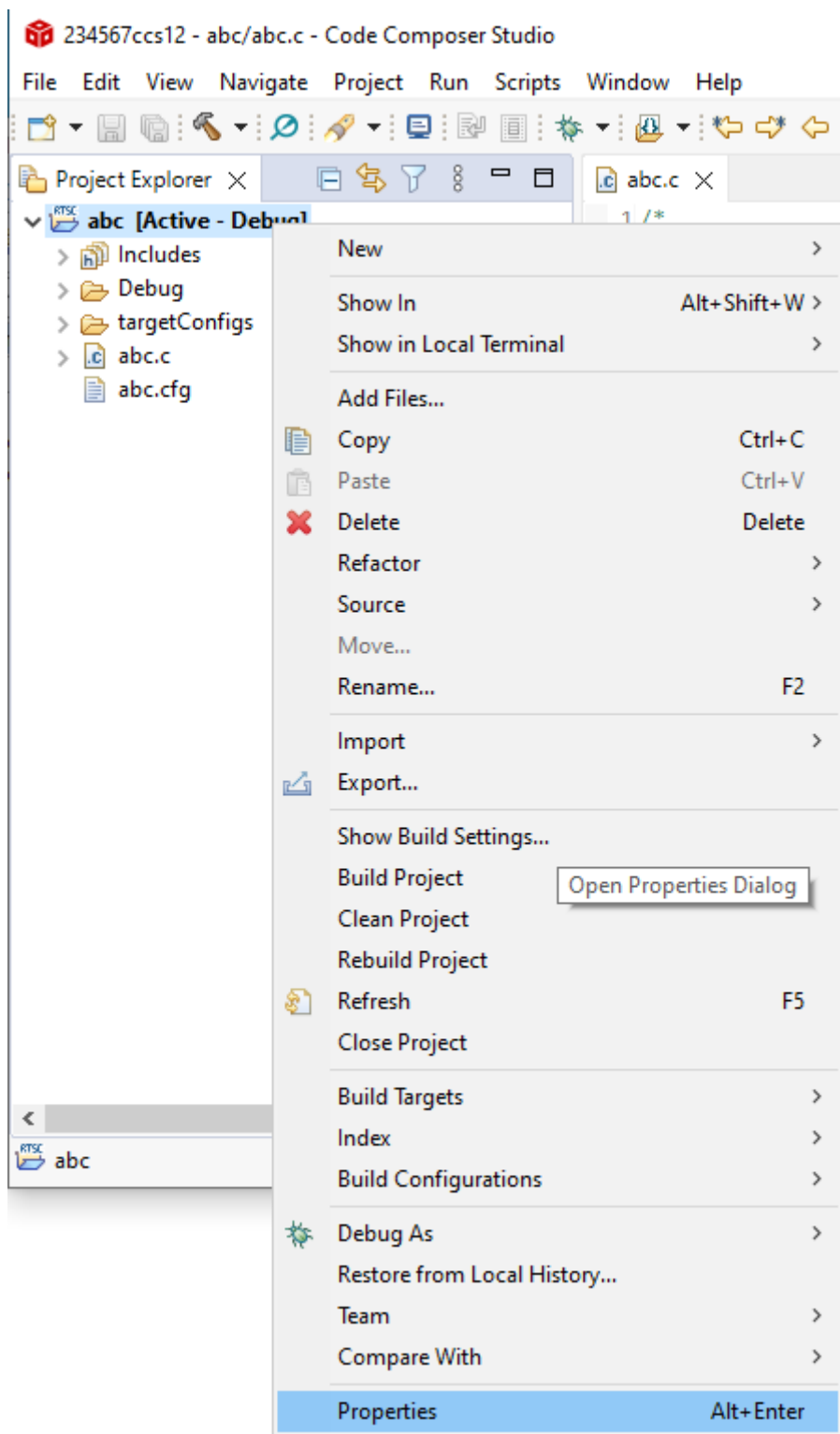


Source folder:	<input type="text" value="abc"/>	<input type="button" value="Browse..."/>
Source file:	<input type="text" value="abc.c"/>	
Template:	<input type="text" value="Default C source template"/>	<input type="button" value="Configure..."/>

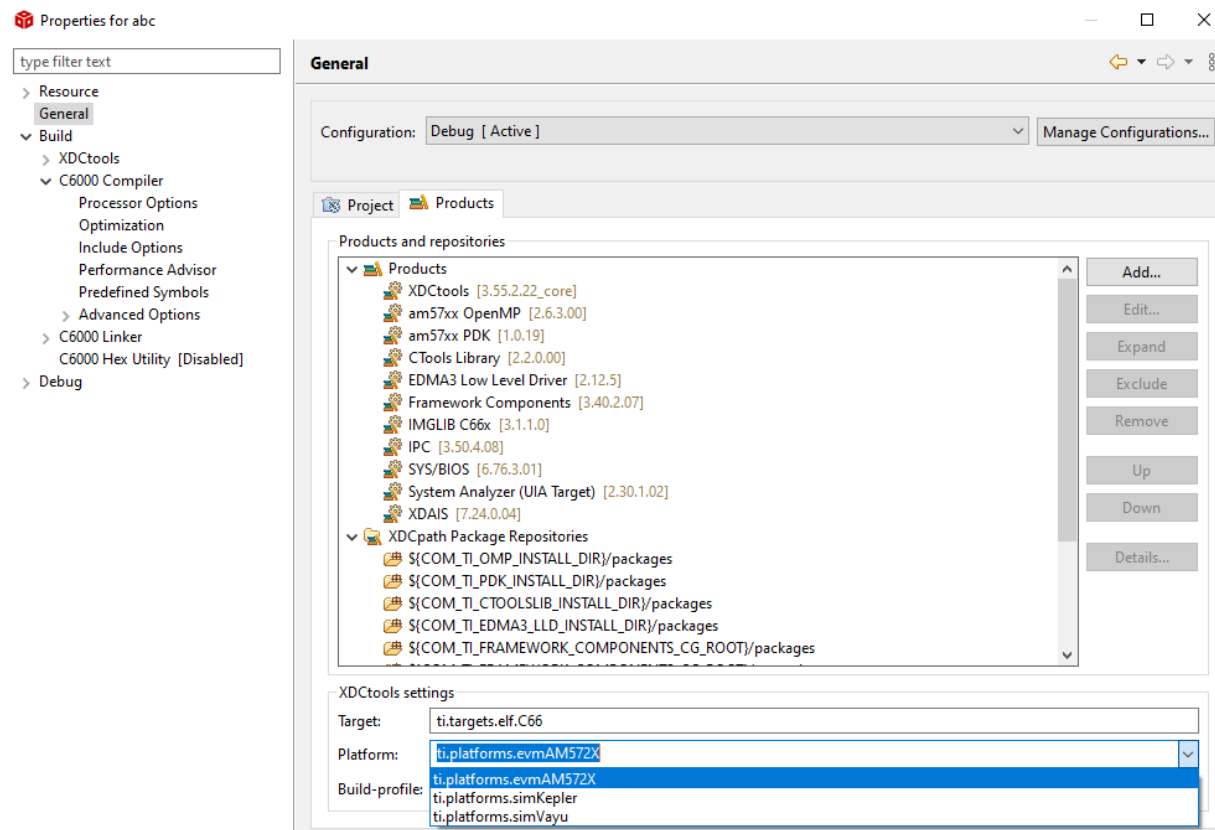
Wybieramy 'Finish' i uzupełniamy plik źródłowy w edytorze o kilka wierszy prostego kodu:



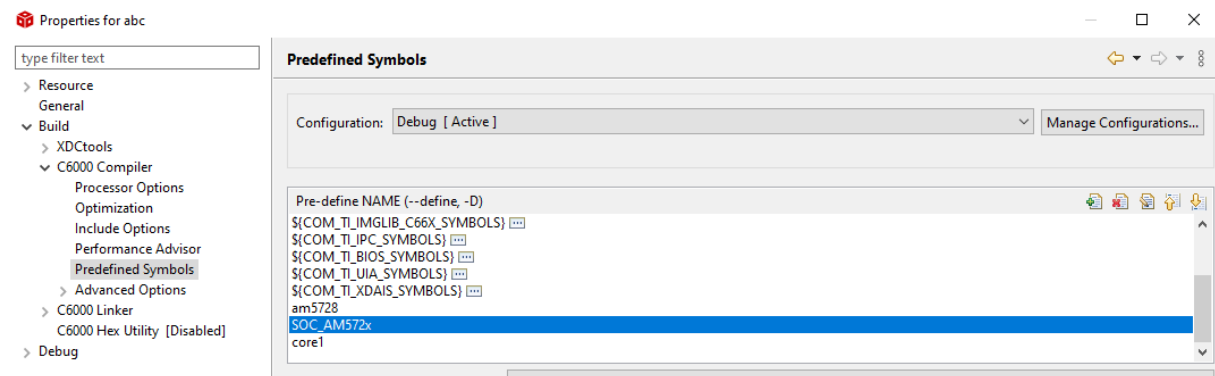
Dokonujemy jeszcze kilku sprawdzeń i zmian we właściwościach projektu:



Sprawdzenie, czy jest wpisana prawidłowo platforma:



Dodanie (przy zaznaczonym am5728) za pomocą nowego symbolu predefiniowanego SOC_AM572x wymaganego przez niektóre biblioteki RTOS dla procesorów AM572x (zwrócić uwagę na wielkość liter w symbolu):



Wybieramy 'Apply and Close'.

Możemy teraz dokonać próby kompilacji przez 'Build Project' lub 'Build All' (inaczej Ctrl+B) jeśli mamy jeden otwarty projekt.

W przypadku komunikatów o błędach należy je poprawić w kodzie źródłowym. Jeśli nie ma błędów, to kompilacja powinna zakończyć się komunikatem o prawidłowym utworzeniu pliku wynikowego typu out:

```
Finished building target: "abc.out"
```





lub:

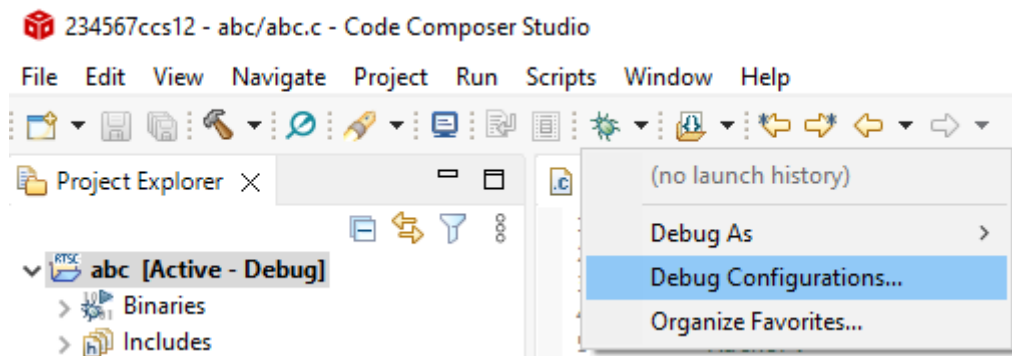
```
gmake[1]: 'abc.out' is up to date.
```

```
**** Build Finished ****
```

```
**** Build Finished ****
```

4. Konfiguracja sesji 'Debug'

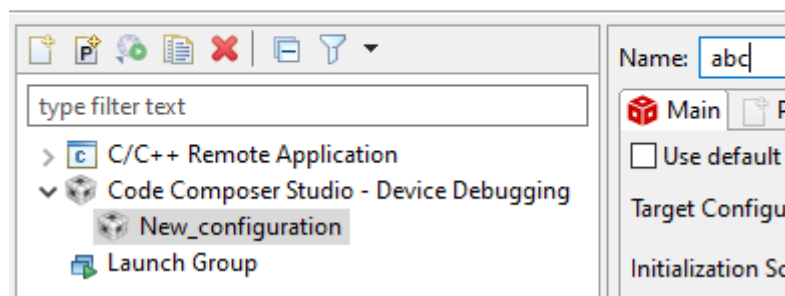
Przed przełączeniem się w sesję 'Debug' należy utworzyć plik konfiguracyjny tej sesji. W przypadku większej liczby projektów każdy z nich może mieć inny plik konfiguracyjny sesji 'Debug' lub ten sam, jednak w celu uniknięcia błędów w konfiguracji przyjmujemy w laboratorium, że nazwa pliku konfiguracyjnego sesji 'Debug' jest taka sama, jak nazwa projektu. Plik konfiguracyjny tworzymy przez 'Debug Configurations' wybierając  obok , (uwaga: błędne wybranie w tym momencie  zamiast  spowoduje trudne do poprawienia błędy w konfiguracji sesji 'Debug'):



Deklaracja nowej konfiguracji (dwukrotny klik na 'Code Composer Studio – Device Debugging') i wpisanie w polu 'Name' nazwy konfiguracji takiej samej, jak nazwa projektu (tutaj abc) i potwierdzenie przez 'Apply':

Create, manage, and run configurations

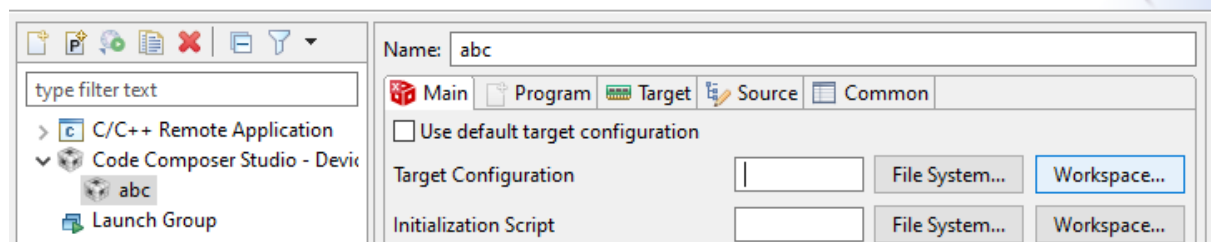
Start CCS Debug launch

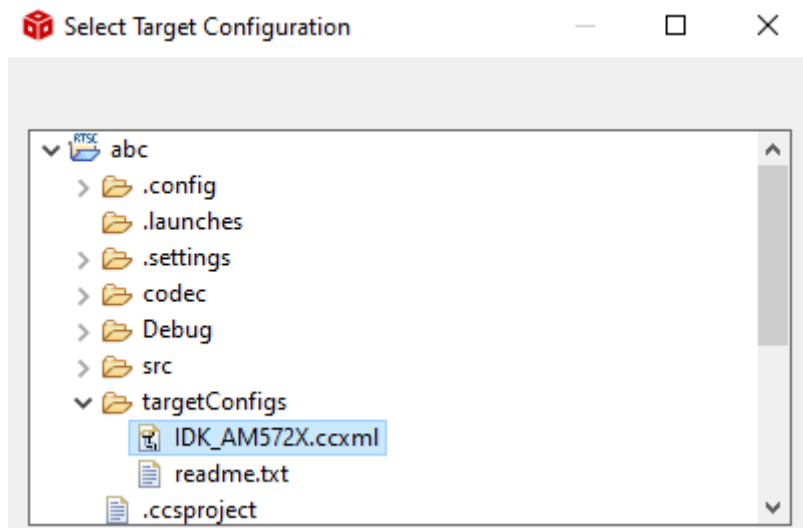


W karcie 'Main' w wierszu 'Target Configuration' wybieramy 'Workspace' i zaznaczamy z katalogu bieżącego projektu (tutaj abc) plik konfiguracyjny typu cxml z podkatalogu targetConfigs:

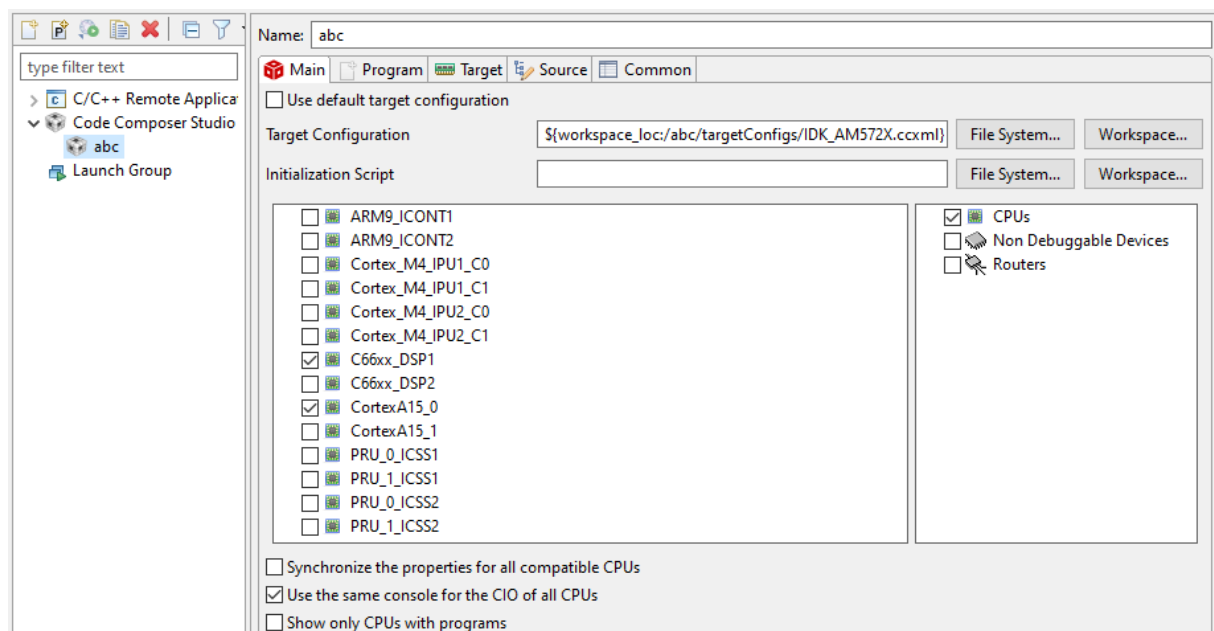
Create, manage, and run configurations

Specify a Target Configuration

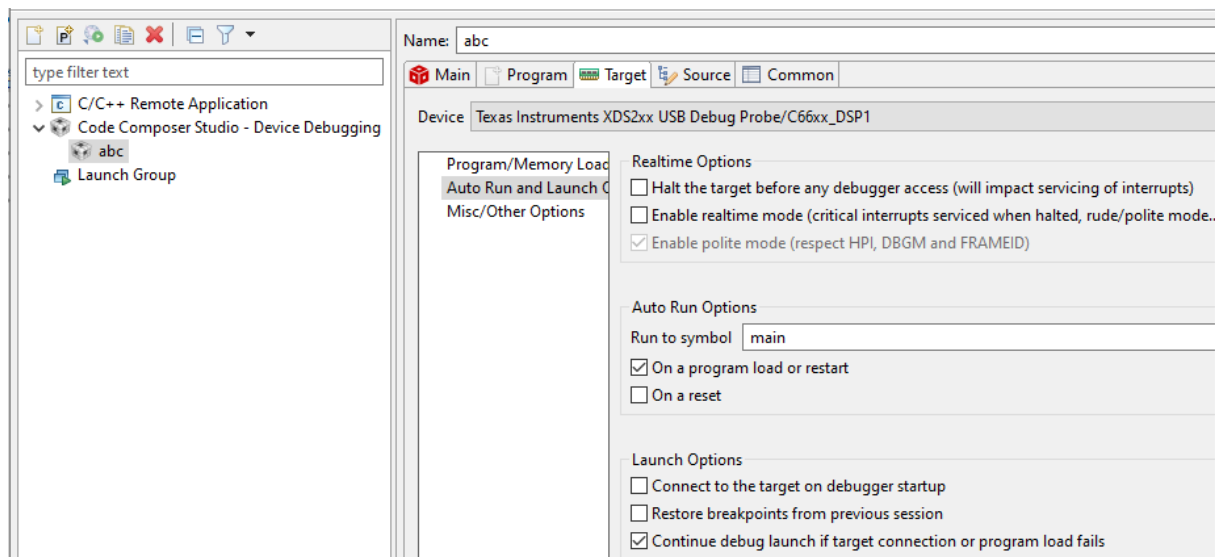




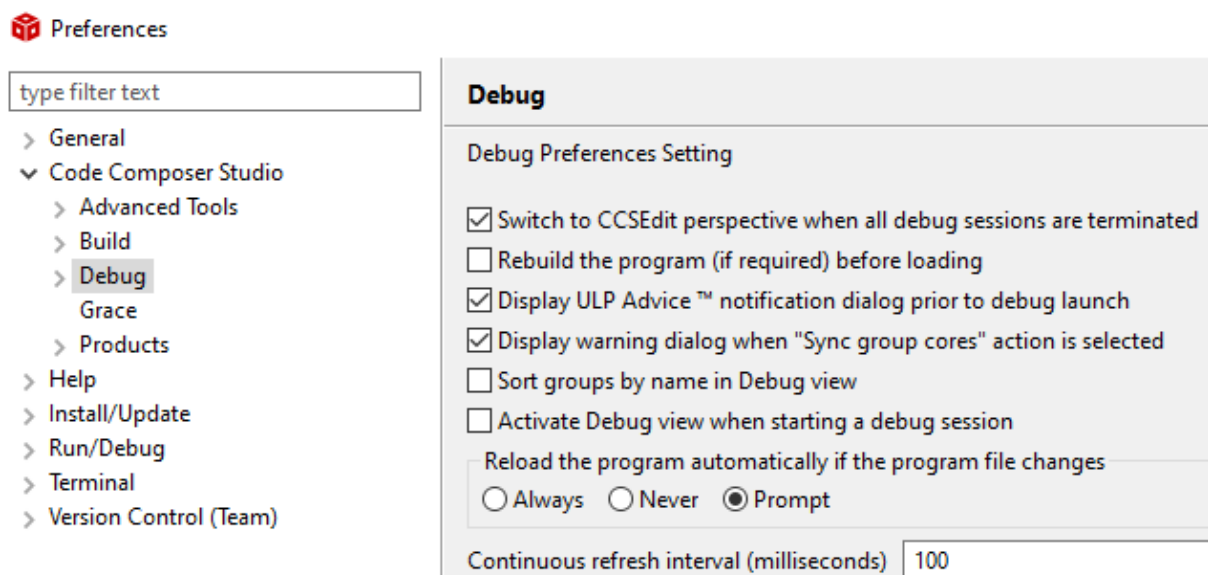
Zaznaczamy tylko rdzenie 'C66xx_DSP1' oraz 'CortexA15_0' (lista aż do PRU_1_ICSS2 !):




Wybieramy 'Apply' i przechodzimy do karty 'Target'. W pozycji 'Auto Run and Launch Options' ustawienia powinny być następujące:

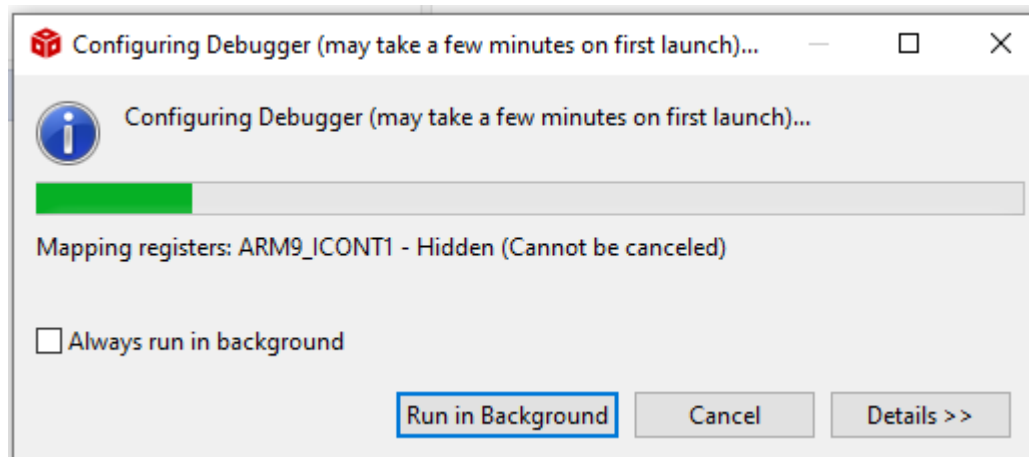


Warto jeszcze ustawić w Window -> Preferences -> 'Code Composer Studio' -> 'Debug' wartość 100 dla 'Continuous refresh interval (milliseconds)':

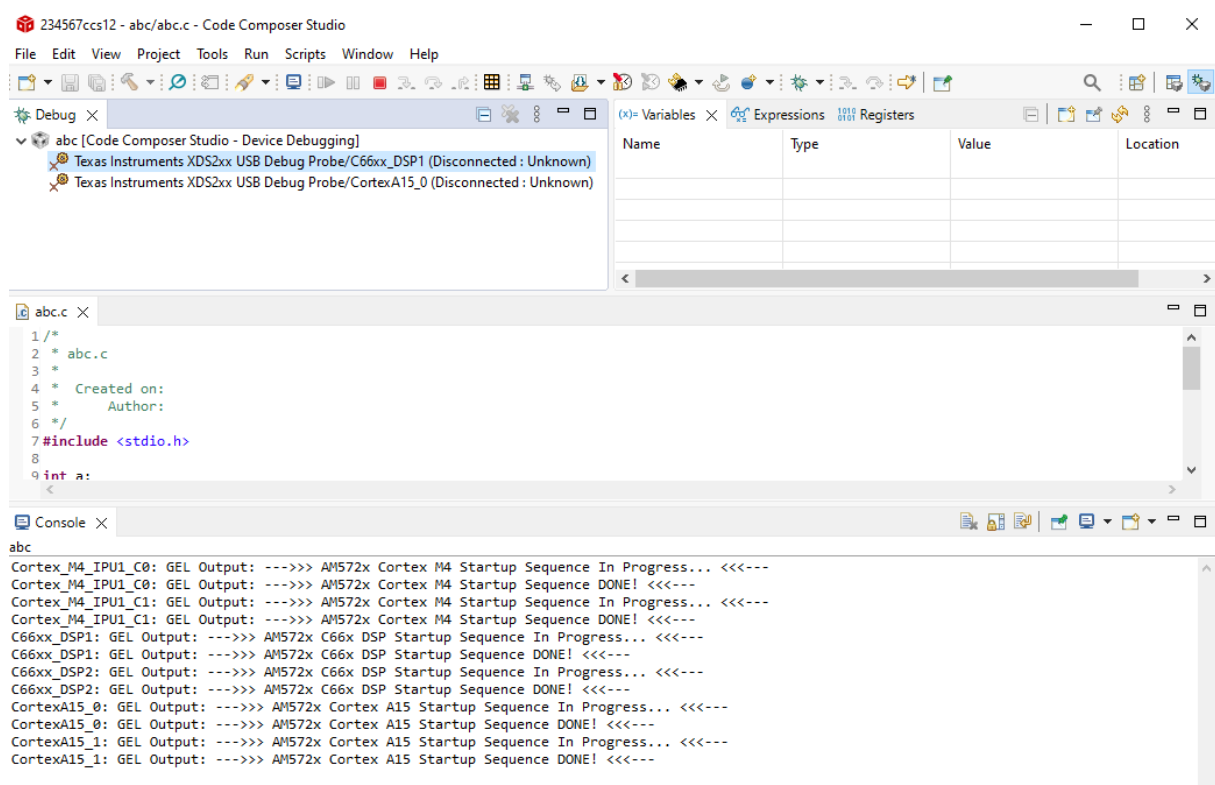


5. Otwarcie aktywnej sesji 'Debug', połączenie z rdzeniem A15 i DSP oraz załadowanie programu

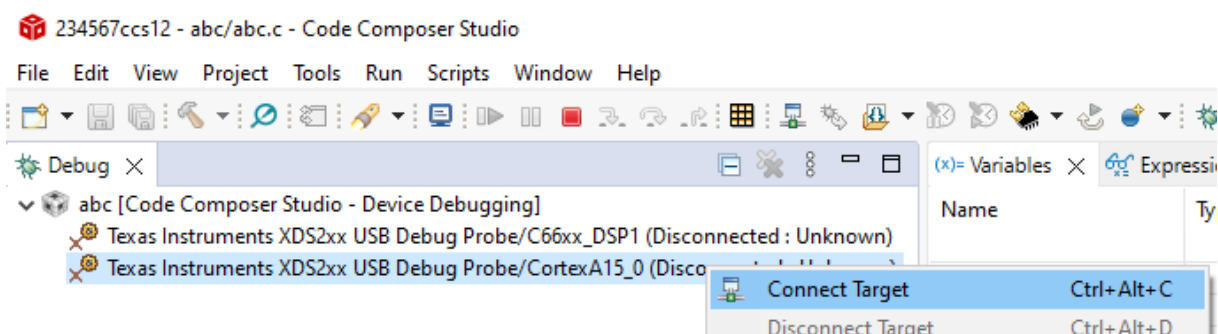
Uruchamiamy sesję poprzez  - pojawia się wówczas:



i po dłuższej chwili następuje przejście do widoku (Perspective) 'Debug':



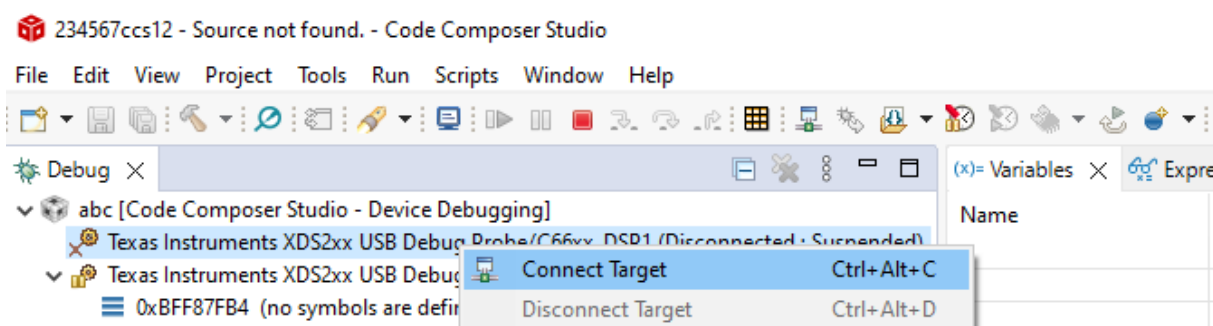
W okienku 'Debug' (lewy górny róg powyżej) komunikaty 'Disconnected: Unknown' oznaczają, że emulator XDS nie połączył się jeszcze z A15 i DSP1. To połączenie uruchamiamy najpierw koniecznie dla CortexA15_0:



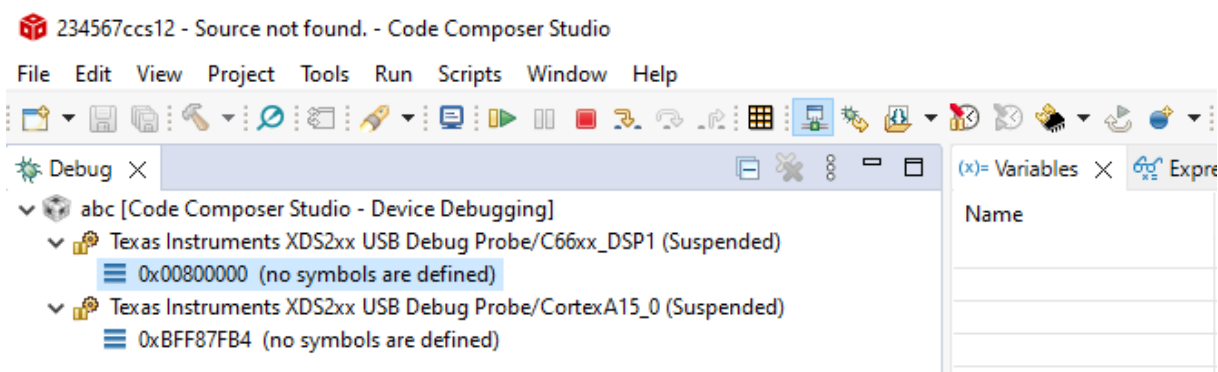
Należy teraz zaczekać, aż pojawi się w oknie 'Console' komunikat:

CortexA15_0: GEL Output: --->>> DSP1SS Initialization is DONE! <<<---

Dopiero teraz można uruchamiać połączenie z C66xx_DSP1:




W efekcie uzyskujemy stan, w którym XDS ma połączenie z obydwojema rdzeniami, które są w stanie wstrzymania ('Suspended'):

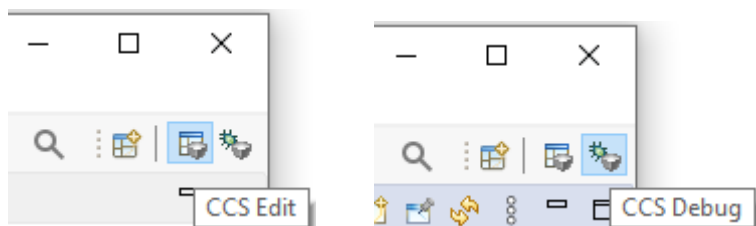



W konsoli (w dolnej części okna aplikacji) znajduje się długa historia wykonania plików typu GEL (charakterystycznych dla środowiska CCS), w których – przy opisanym w niniejszym opracowaniu podejściu – nie powinno być informacji w kolorze czerwonym informującym o błędzie. Końcowa część tego raportu jest następująca:

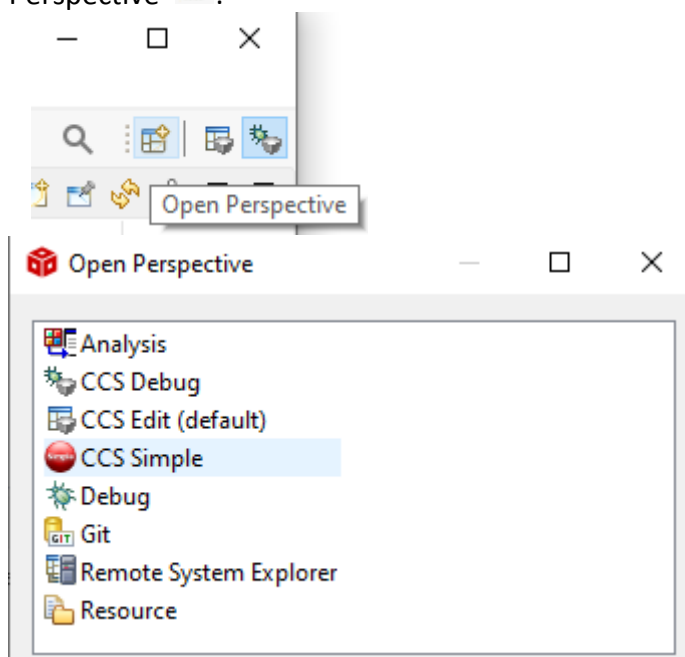
```
CortexA15_0: GEL Output: --->>> DSP1SS Initialization is in progress ... <<<---
CortexA15_0: GEL Output: DEBUG: Clock is active ...
CortexA15_0: GEL Output: DEBUG: Checking for data integrity in DSPSS L2RAM ...
CortexA15_0: GEL Output: DEBUG: Data integrity check in GEM L2RAM is successful!
CortexA15_0: GEL Output: --->>> DSP1SS Initialization is DONE! <<<---
```

W przypadku wystąpienia komunikatów w kolorze czerwonym (o wystąpieniu błędu) należy ten fakt zgłosić prowadzącemu. Najczęściej konieczne jest wówczas wykonanie procedury resetu wg instrukcji „Opis resetu BBAI” umieszczonej na pulpicie systemu Windows.

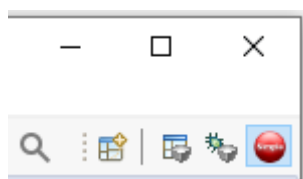
Ze względu na długi proces startu sesji 'Debug' zaleca się nie używać w czasie pracy funkcji zakończenia sesji 'Debug' (tj. funkcji 'Terminate') uzyskiwanej poprzez  (za wyjątkiem sytuacji na zakończenie pracy z modułem, tj. przed zamknięciem aplikacji CCS lub konieczności wykonania resetu). Dobrym wyjściem jest przechodzenie do widoku 'Edit' bez zakończenia sesji 'Debug'. Dobrym rozwiązaniem jest również wprowadzić wówczas rdzeń DSP w stan 'Suspended' lub pozostawić program zatrzymany w Breakpoincie. Przejście między widokiem 'Edit' (w którym widoczne są narzędzia przydatne do edycji i kompilacji) oraz widokiem 'Debug' (w którym widoczne są narzędzia przydatne przy uruchamianiu programu) bez zakończenia sesji 'Debug' odbywa się poprzez dwie ikonki w prawym górnym rogu aplikacji CCS:



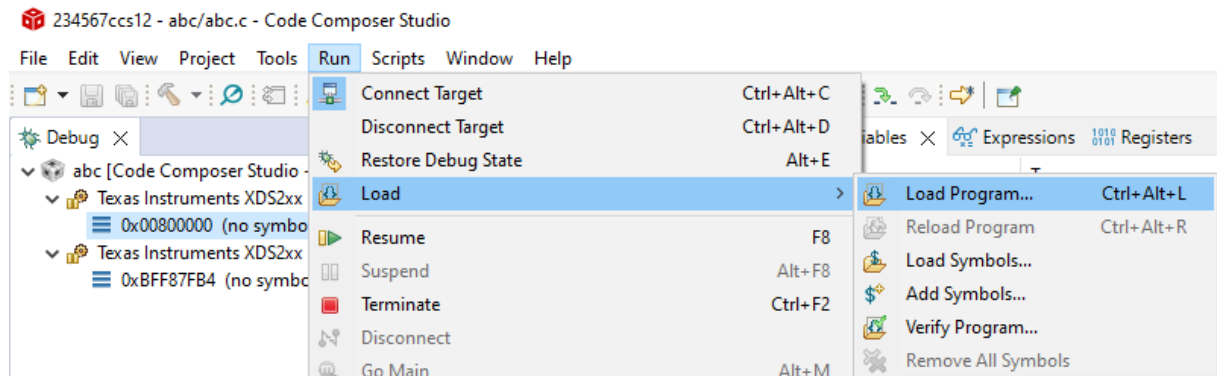
Przydatnym może być również włączenie widoku (Perspective) typu 'Simple', który jest połączeniem wybranych narzędzi z widoku 'Edit' i 'Debug', co pozwala na dłuższą pracę w jednym widoku. Włączenie ikonki dla widoku 'Simple' odbywa się poprzez ikonkę 'Open Perspective' :



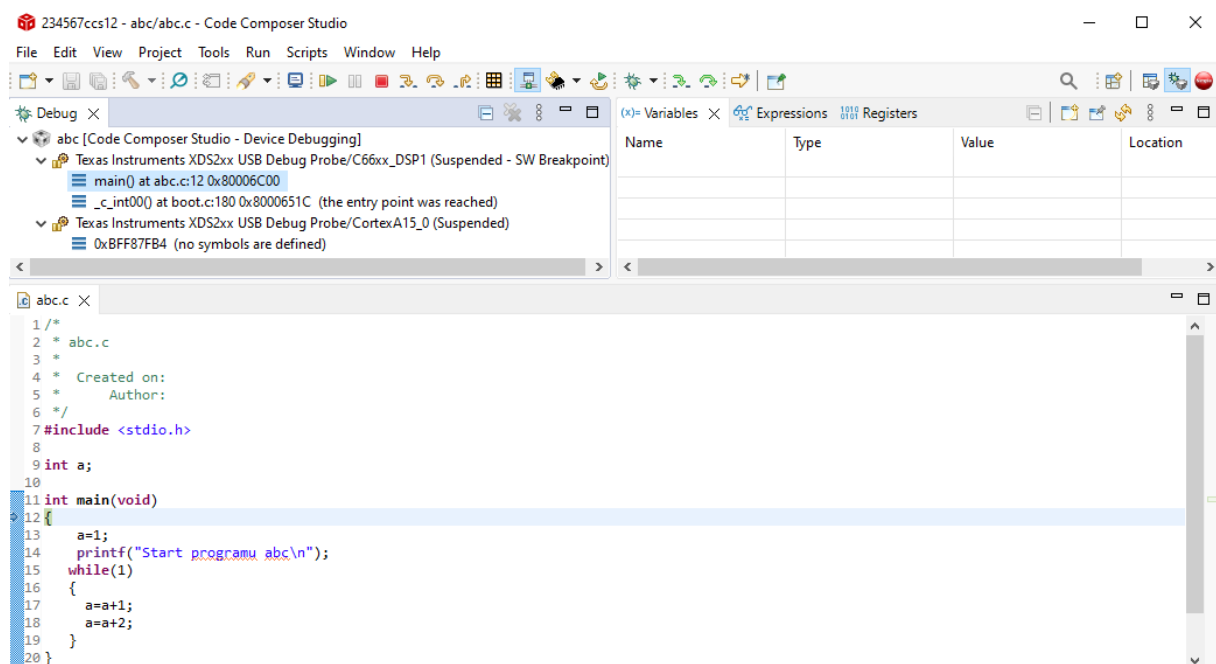
W efekcie uzyskuje się dostęp do trzech widoków ('Edit', 'Debug', 'Simple'):



Załadowanie programu skompilowanego wcześniej do pliku typu out odbywa się (przy wybranym rdzeniu DSP1) poprzez Run -> Load -> 'Load Program' (lub Ctrl+Alt+L):



i dalej 'Browse' i po odnalezieniu w workspace i podkatalogu 'nazwa projektu'\Debug (tutaj abc\debug) pliku typu out (tutaj abc.out) i dalej 'Otwórz' i 'Ok', program powinien zostać załadowany, ale jeszcze nie uruchomiony, a licznik rozkazów powinien ustawić się na początek programu main:



Program jest gotowy do uruchomienia.

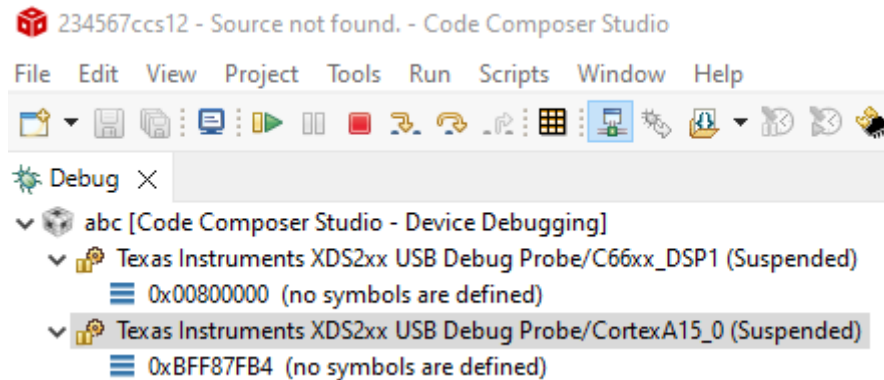
6. Uruchomienie programu, praca krokowa, zakładanie pułapek, sprawdzanie wartości zmiennych

Uruchomienie programu, praca krokowa, zakładanie pułapek, sprawdzanie wartości zmiennych uzyskuje się podobnie jak w standardowych narzędziach dla różnych mikroprocesorów. Ćwiczącym pozostawia się zadanie samodzielnego (a w razie problemów z pomocą prowadzącego zajęcia) sprawdzenia działania poszczególnych funkcji uruchomieniowych.

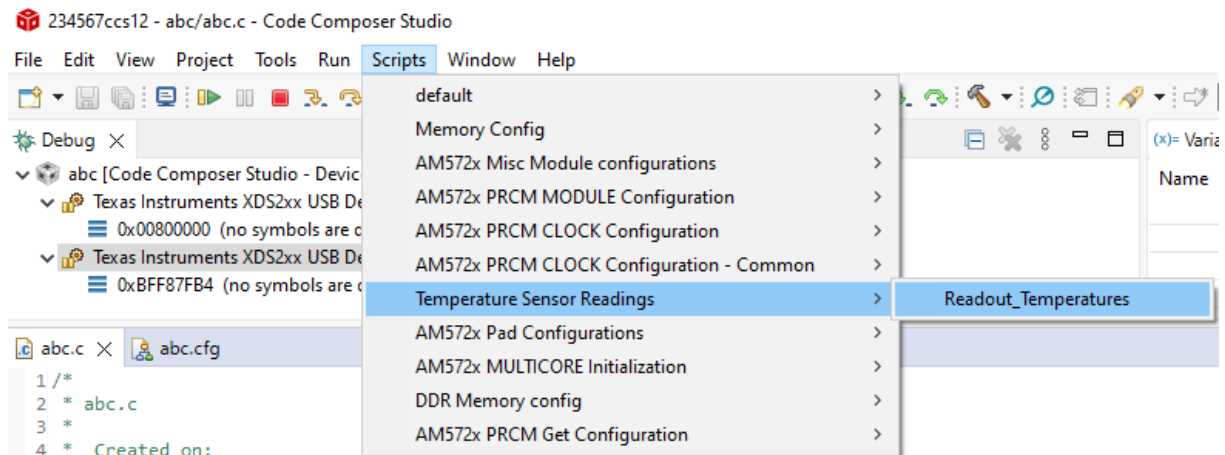
7. Przydatne narzędzia dodatkowe

7.1. Pomiar temperatury pracy

Warto od czasu do czasu sprawdzać temperaturę, w jakiej pracuje procesor AM5729 modułu BB AI (układ jest wyposażony w radiator i wentylator). Do tego celu służy skrypt GEL skonfigurowany dla rdzenia A15. W tym celu należy zaznaczyć w okienku Debug ten rdzeń (cały czas pozostanie on w trybie Suspended):



i następnie uruchomić odpowiedni skrypt GEL z menu Scripts:

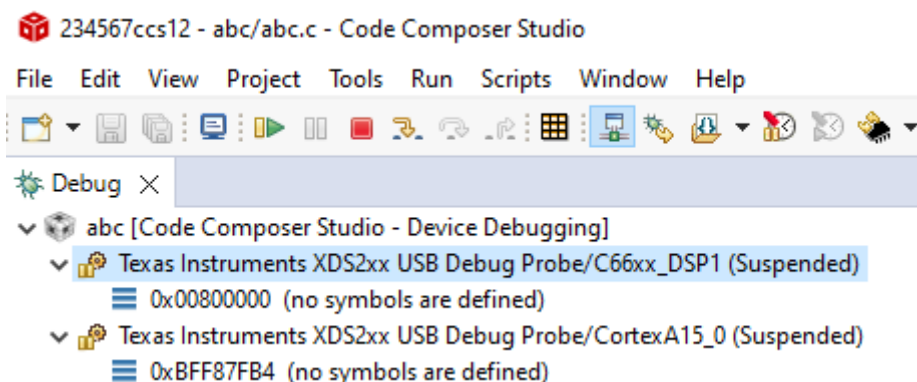


Efektem działania tego skryptu jest odczyt temperatury z czujników temperatury i wyświetlenie wyników w konsoli:

```
CortexA15_0: GEL Output: MPU Temperature: 60.7696991 degC
CortexA15_0: GEL Output: GPU Temperature: 59.5555992 degC
CortexA15_0: GEL Output: CORE Temperature: 59.9603004 degC
CortexA15_0: GEL Output: IVA Temperature: 60.3650017 degC
```

W przypadku przekroczenia temperatury 70°C należy o tym fakcie poinformować prowadzącego zajęcia.

Po zakończeniu pomiaru temperatury ponownie wybieramy rdzeń DSP w okienku Debug, co powoduje, że kolejne operacje (np. ładowanie programu) będą wykonywane dla tego rdzenia:



Pozostałych skryptów GEL nie należy uruchamiać, gdyż ich zawartość może wymagać zmiany dla modułu BB AI.

8. Uwagi dotyczące katalogu workspace, kopiowanie / archiwizacja całego projektu lub jego najważniejszych plików

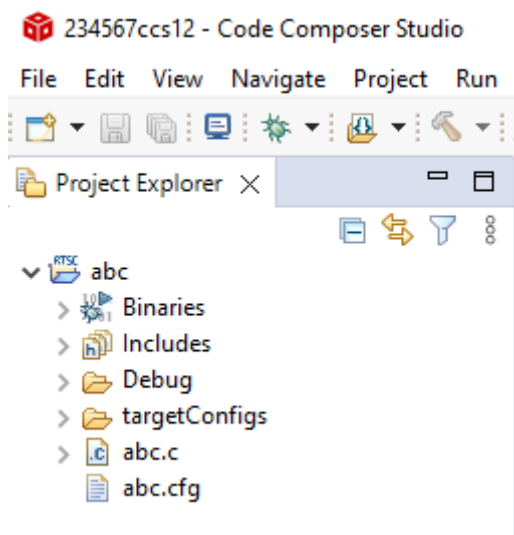
Po utworzeniu pierwszego projektu abc w katalogu workspace (czyli D:\nrindeksu\ccs12) w stosunku do sytuacji omówionej w pkt. 2 (po utworzeniu workspace, ale przed utworzeniem pierwszego projektu) został utworzony jeszcze katalog abc:



Natomiast sam katalog projektu zawiera pliki projektu:

Ten komputer > Dysk lokalny (D:) > 234567ccs12 > abc			
Nazwa	Typ	Rozmiar	
.launches	Folder plików		
.settings	Folder plików		
Debug	Folder plików		
targetConfigs	Folder plików		
.ccsproject	Plik CCSPROJECT	1 KB	
.cproject	Plik CPROJECT	41 KB	
.project	Plik PROJECT	1 KB	
.xdchelp	Plik XDCHHELP	0 KB	
abc	Plik C	1 KB	
abc	Plik CFG	0 KB	

Pliki projektu są również wyszczególnione w karcie 'Project Explorer' w CCS, ale bez katalogów i plików rozpoczynających się od znaku '.'. W przypadku projektu abc w 'Project Explorer' widoczne są katalogi Debug, targetConfigs oraz pliki abc.c i abc.cfg:

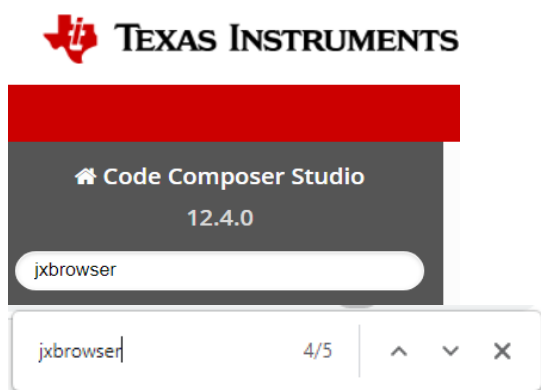


Rozdział '6.1.1.1. Workspaces' w podręczniku użytkownika CCS zawiera szereg praktycznych uwag dotyczących katalogu Workspace:

https://software-dl.ti.com/ccs/esd/documents/users_guide/ccs_project-management.html#workspaces

w tym m.in. zalecenie okresowego 'czyszczenia' tego katalogu z wcześniejszym zapisaniem jego ustawień oraz reimportem wcześniejszych projektów. Z upływem czasu bardzo wzrasta wielkość obszaru dysku zajmowanego przez katalogi '.jxbrowser.userdata' oraz '.metadata'. Informacje o tych podkatalogach można znaleźć w Podręczniku CCS wykorzystując jego wyszukiwarkę, a następnie używając funkcji wyszukaj (Ctrl-F) samej przeglądarki, np.:

← → ↻ software-dl.ti.com/ccs/esd/



9.5.1.4. Clear the CCS Browser Cache

Try deleting both the:

- Chromium browser user cache folder `[WORKSPACE FOLDER DIR]/.jxbrowser.userdata`

Ze znalezionej odpowiedzi wynika, że w celu wyczyszczenia katalogu '.jxbrowser.userdata' wystarczy go skasować i przy kolejnym uruchomieniu CCS zostanie on utworzony z zawartością początkową.

Natomiast katalogu '.metadata' nie zaleca się w ten sposób kasować, gdyż zawiera on konfigurację workspace, którą można zapisać w sposób opisany w rozdz. 6.1.1.1.1 Podręcznika

CCS. Tworzony jest wówczas plik typu epf, np. abc.epf. Po utworzeniu nowego katalogu workspace można jego wcześniejsze ustawienia zaimportować z tego pliku.

Funkcja eksportu plików projektu do pliku zip opisana jest w Podręczniku CCS w rozdz. 6.1.2.2.1, a funkcja importu z tego pliku w rozdz. 6.1.2.3. W najprostszej wersji można skopiować na dysk sieciowy lub pendrive cały podkatalog projektu (w naszym przypadku abc), czy nawet cały katalog workspace.

Jako absolutne minimum należy przyjąć częste kopiowanie na własny nośnik (sieciowy i/lub pendrive) przynajmniej następujących plików (przykład dla projektu abc): plik abc.c i inne pliki źródłowe dołączane dyrektywą #include, abc.cfg i ewentualnie abc.epf. Pozostałe ustawienia można stosunkowo prosto utworzyć od początku. Wykonywanie kopii bezpieczeństwa projektu (przynajmniej kluczowych plików umożliwiających szybkie odtworzenie projektu) tworzonych na zajęciach jest koniecznym warunkiem pozytywnego rozliczenia się ćwiczącego z pracy w trakcie semestru i pod jego koniec przy wystawianiu oceny końcowej z kursu.

Procedura na zakończenie zajęć:

1. Zamknąć sesję 'Debug' za pomocą ('Terminate'). W efekcie nastąpi przejście do sesji 'Edit'.
2. Po przejściu do sesji 'Edit' zamknąć aplikację CCS.
3. Wyłączyć zasilanie modułu przełącznikiem elektronicznym na zasilającym kablu USB.
4. Skopiować projekt lub najważniejsze pliki projektu na własny nośnik (dysk sieciowy i/lub pendrive).