

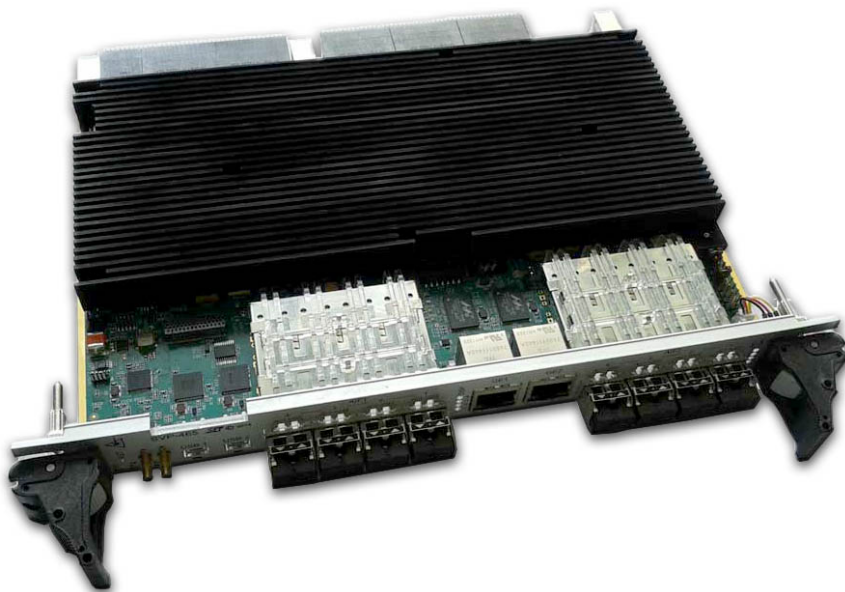


Scan Engineering Telecom SPb

SVP-465. Сборка и запуск теста HyperLink

Руководство пользователя

Версия 1.0



Код документа: UG-SVP-465-HL
Дата сборки: 27 мая 2015 г.
Листов в документе: 23

© 2015, ООО «Скан Инжиниринг Телеком - СПб»
<http://www.setdsp.ru>

Содержание

Список рисунков	3
Список таблиц	3
Список листингов	3
Список процедур	3
Перечень сокращений и условных обозначений	4
1 Общие сведения	5
2 Конфигурация рабочего пространства	7
3 Конфигурация проекта	8
3.1 Конфигурация количества линий HyperLink	8
3.2 Конфигурация рабочей скорости HyperLink	8
3.3 Включение режима локальной петли HyperLink	9
4 Сборка проекта	10
5 Импорт и запуск целевой конфигурации модуля	12
6 Загрузка кода	15
7 Запуск теста HyperLink	18
Приложение А: Разделение вывода сообщений (CIO) ядер процессоров	21

Список рисунков

1-1	Структурная схема модуля SVP-465	5
2-1	Выбор пути рабочего пространства (workspace) в CCS	7
2-2	Проект «HyperLink_Test» и его конфигурации сборки	7
	а) Обзорщика проектов и окно конфигураций сборки проекта	7
	б) Выбор активной конфигурации сборки проекта	7
4-1	Пункт меню для сборки проекта	10
4-2	Окно выбора конфигураций для сборки	10
4-3	Сборка проекта теста производительности	11
5-1	Пункт меню для отображения окна целевых конфигураций	12
5-2	Меню импорта целевой конфигурации	12
5-3	Окно выбора файла для импорта целевой конфигурации	13
5-4	Окно выбора способа импорта файла целевой конфигурации	13
5-5	Запуск целевой конфигурации	13
5-6	Список ядер процессоров модуля SVP-465	14
6-1	Группировка ядер процессоров	15
6-2	Ядра процессоров после выполнения подключения	15
6-3	Меню загрузки кода на ядро процессора	16
6-4	Окно загрузки кода на ядро процессора	16
6-5	Окно выбора файла для загрузки на ядро процессора	16
	а) Процессор TMS320C6670	16
	б) Процессор TMS320C6678	16
6-6	Внешний вид окна «Debug» после загрузки кода на ядра процессоров	17
A-1	Контекстное меню целевой конфигурации	21
A-2	Окно настроек целевой конфигурации	21
A-3	Открытие второго окна «Console»	22
A-4	Два окна «Console»	22
A-5	Выбор ядра для отображения вывода в окне «Console»	22
A-6	Вывод сообщений с двух ядер в два окна «Console»	23

Список таблиц

3-1	Макроопределения количества линий HyperLink	8
3-2	Макроопределения рабочей скорости HyperLink	8

Список листингов

3-1	Файл «hylinkLLDCfg.h» (конфигурация количества линий HyperLink)	8
3-2	Файл «hylinkLLDCfg.h» (конфигурация рабочей скорости HyperLink)	8
3-3	Файл «hylinkLLDCfg.h» (включение режима локальной петли HyperLink)	9
7-1	Вывод в консоль теста HyperLink с ядра «DSP1_C6670_0»	18
7-2	Вывод в консоль теста HyperLink с ядра «DSP1_C6678_0»	19

Список процедур

7-1	Запуск теста HyperLink	18
-----	------------------------	----

Перечень сокращений и условных обозначений

CCS	Code Composer Studio	6, 7, 14, 18, 21, 22
CIO	Console Input/Output	2, 14, 21
EDMA	Enhanced Direct Memory Access	6
IPC	Inter Process Communication	6
LLD	Low Level Driver	6
MCSDK	MultiCore Software Development Kit	6
PDK	Platform Development Kit	6
TI	Texas Instruments	6
UART	Universal Asynchronous Receiver-Transmitter	18
ОС	Операционная Система	6

1 Общие сведения

В данном документе описан процесс сборки из исходных кодов и запуск теста HyperLink на модуле SVP-465, Тест выполняет передачу данных по интерфейсу HyperLink между процессорами TMS320C6670 и TMS320C6678. На модуле SVP-465 посредством HyperLink каждый процессор TMS320C6670 соединен с процессором TMS320C6678 (см. рисунок 1-1).

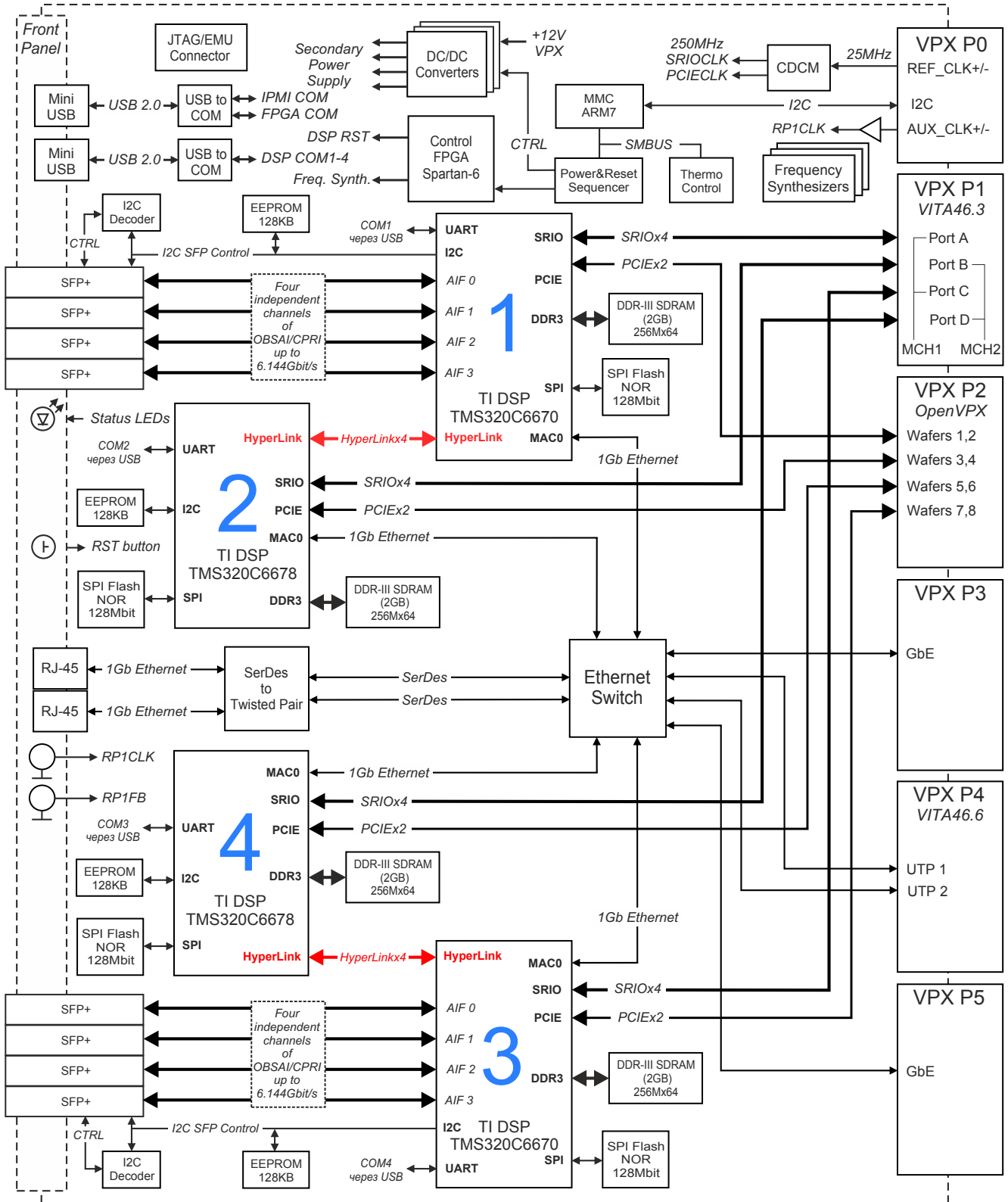


Рисунок 1-1: Структурная схема модуля SVP-465

Для сборки и запуска теста HyperLink требуется установленная система разработки CCS (Code Composer Studio) компании TI. Сборка и запуск теста HyperLink были проверены на CCS версии 5.4.0.00091 под ОС Windows 7 (64-bit). Среду разработки CCS можно бесплатно скачать с сайта производителя¹.

Для сборки проекта теста HyperLink необходимы следующие установленные пакеты:

- PDK C6670 версии 1.1.2.6;
- PDK C6678 версии 1.1.2.6;
- IPC (Inter Process Communication) версии 1.24.3.32;
- EDMA3 LLD версии 2.11.5 (входит в состав PDK);
- HyperLink LLD версии 1.0.1.5 (входит в состав PDK);
- SYS/BIOS версии 6.33.6.50.
- XDCtools версии 3.25.5.94 (дистрибутив для Windows систем имеется на сопроводительном диске в папке «Install»).

В данном списке указаны версии пакетов, на которых производилось тестирование. Сборка проекта возможна с использованием пакетов более ранних версий, однако, в этом случае, не гарантируется правильная работа теста HyperLink.

Тест передачи данных через интерфейс HyperLink написан на основе исходного кода теста передачи данных из состава библиотек TI MCSDK (MultiCore Software Development Kit) для процессоров TMS320C6670 и TMS320C6678.

Как видно из структурной схемы модуля SVP-465 (рисунок 1-1), посредством HyperLink соединены между собой процессоры:

- процессор «DSP1_C6670» соединен с процессором «DSP2_C6678»;
- процессор «DSP3_C6670» соединен с процессором «DSP4_C6678».

В данном документе рассматривается загрузка кода и запуск теста HyperLink на процессорах «DSP1_C6670» и «DSP2_C6678». Для запуска кода на процессорах «DSP3_C6670» и «DSP4_C6678», при чтении документа, следует считать процессор «DSP1_C6670» процессором «DSP3_C6670», а процессор «DSP2_C6678» заменять на процессор «DSP4_C6678».

¹ <http://www.ti.com/tool/ccstudio>

2 Конфигурация рабочего пространства

Перед началом сборки и запуска теста передачи данных необходимо с сопроводительного диска к модулю SVP-465 скопировать на жесткий диск компьютера папку рабочего пространства CCS (папка «CCS_Workspace» на диске). В данном документе предполагается, что содержимое папки «CCS_Workspace» было скопировано с сопроводительного диска на жесткий диск компьютера в папку «D:/Dev/Modules/SVP-465/CCS_Workspace».

Кроме папки рабочего пространства, необходимо скопировать с сопроводительного диска папки «TargetConfigurations», «RTSC» и «GEL» со всем содержимым. При этом, важно, чтобы обе данные папки находились на одном уровне. В данном документе предполагается, что содержимое папки «TargetConfigurations» скопировано в папку «D:/Dev/Modules/SVP-465/TargetConfigurations», содержимое папки «GEL» скопировано в папку «D:/Dev/Modules/SVP-465/GEL», а содержимое папки «RTSC» скопировано в папку «D:/Dev/Modules/SVP-465/RTSC».

После запуска среды разработки CCS необходимо указать путь к папке рабочего пространства как показано на рисунке 2-1 («D:/Dev/Modules/SVP-465/CCS_Workspace»).

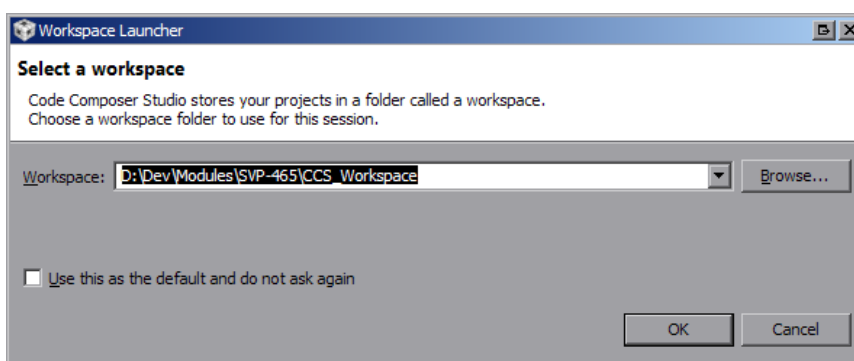
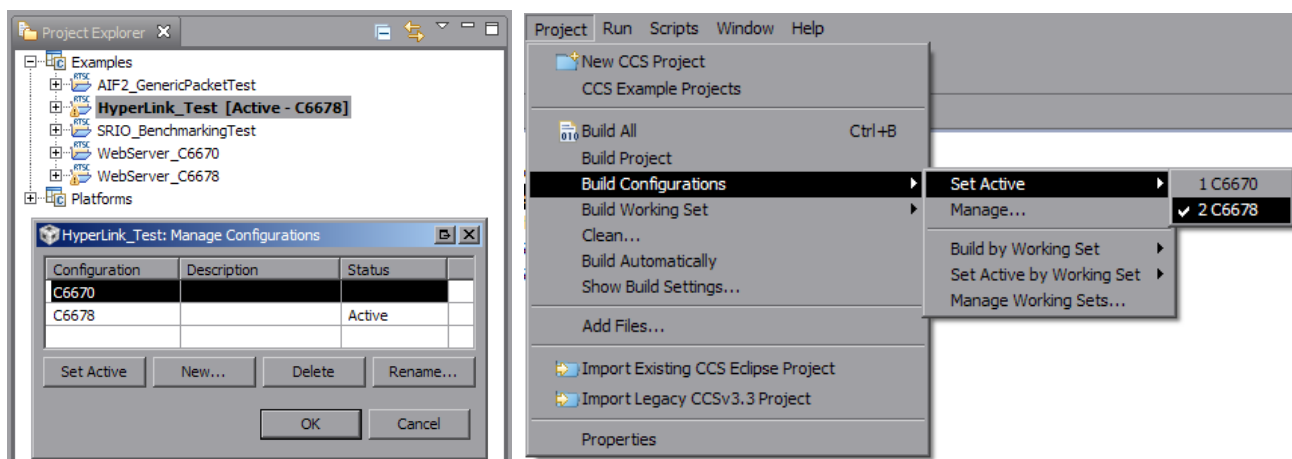


Рисунок 2-1: Выбор пути рабочего пространства (workspace) в CCS

Код теста HyperLink содержится в проекте «HyperLink_Test» (см. рисунок 2-2а) рабочего пространства.



а) Обзорателя проектов и окно конфигураций сборки проекта

б) Выбор активной конфигурации сборки проекта

Рисунок 2-2: Проект «HyperLink_Test» и его конфигурации сборки

Проект «HyperLink_Test» имеет две конфигурации сборки (в нижней части рисунка 2-2а показано окно управления конфигурациями сборки):

- конфигурация «C6670» — проект для запуска на первом процессоре TMS320C6670;
- конфигурация «C6678» — проект для запуска на втором процессоре TMS320C6670.

Для выбора активной конфигурации сборки необходимо выбрать пункт главного меню «Project > Build Configurations > Set Active > *Имя_нужной_конфигурации_сборки*» (см. рисунок 2-2б).

3 Конфигурация проекта

Конфигурация проекта осуществляется путем редактирования содержимого файла «hypLnkLLDCfg.h» проекта «HyperLink_Test».

Для применения выполненных изменений в конфигурации, необходимо выполнить сборку проекта в соответствии с процедурой, описанной в разделе 4.

3.1 Конфигурация количества линий HyperLink

Количество линий Hyperlink устанавливается одним из макроопределений, указанных в таблице 3-1.

Таблица 3-1: Макроопределения количества линий HyperLink

Макроопределение	Количество линий
hypLnk_EXAMPLE_ALLOW_1_LANE	1
hypLnk_EXAMPLE_ALLOW_4_LANES	4 (по умолчанию)

Для выбора необходимого количества линий HyperLink, в файле «hypLnkLLDCfg.h», необходимо раскомментировать необходимое макроопределение (см. листинг 3-1).

Листинг 3-1: Файл «hypLnkLLDCfg.h» (конфигурация количества линий HyperLink)

```
81 //#define hypLnk_EXAMPLE_ALLOW_1_LANE
82 #define hypLnk_EXAMPLE_ALLOW_4_LANES
```

3.2 Конфигурация рабочей скорости HyperLink

Рабочая скорость Hyperlink устанавливается одним из макроопределений, указанных в таблице 3-2.

Таблица 3-2: Макроопределения рабочей скорости HyperLink

Макроопределение	Скорость, ГГц
hypLnk_EXAMPLE_SERRATE_01p250	1.25
hypLnk_EXAMPLE_SERRATE_03p125	3.125
hypLnk_EXAMPLE_SERRATE_06p250	6.25 (по умолчанию)
hypLnk_EXAMPLE_SERRATE_07p500	7.5
hypLnk_EXAMPLE_SERRATE_10p000	10.0
hypLnk_EXAMPLE_SERRATE_12p500	12.5

Для выбора необходимой рабочей скорости HyperLink, в файле «hypLnkLLDCfg.h», необходимо раскомментировать необходимое макроопределение (см. листинг 3-2).

Листинг 3-2: Файл «hypLnkLLDCfg.h» (конфигурация рабочей скорости HyperLink)

```
87 //#define hypLnk_EXAMPLE_SERRATE_01p250
88 //#define hypLnk_EXAMPLE_SERRATE_03p125
89 #define hypLnk_EXAMPLE_SERRATE_06p250
90 //#define hypLnk_EXAMPLE_SERRATE_07p500
91 //#define hypLnk_EXAMPLE_SERRATE_10p000
92 //#define hypLnk_EXAMPLE_SERRATE_12p500
```


3.3 Включение режима локальной петли HyperLink

По умолчанию, в проекте «HyperLink_Test» режим локальной петли HyperLink отключен. Для того, чтобы включить режим локальной петли HyperLink, необходимо в файле «hypLnkLLDCfg.h» раскомментировать строку с определением макроса «hypLnk_EXAMPLE_LOOPBACK» (см. листинг 3-3).

Листинг 3-3: Файл «hypLnkLLDCfg.h» (включение режима локальной петли HyperLink)

```
72 /*****
73  * Select internal loopback or use the SERDES connection
74  *****/
75 //define hypLnk_EXAMPLE_LOOPBACK
```

Примечание

В данном документе описан процесс сборки и запуска теста HyperLink для передачи данных между двумя процессорами. Запуск теста HyperLink с включенной локальной петлей в данном документе не рассмотрен.

4 Сборка проекта

Для сборки проекта нажмите правой кнопкой мыши по названию проекта «HyperLink_Test» в окне обозревателя проектов («Project Explorer») для вызова контекстного меню и выберите пункт меню «Build Configurations > Build Selected...» (рисунок 4-1).

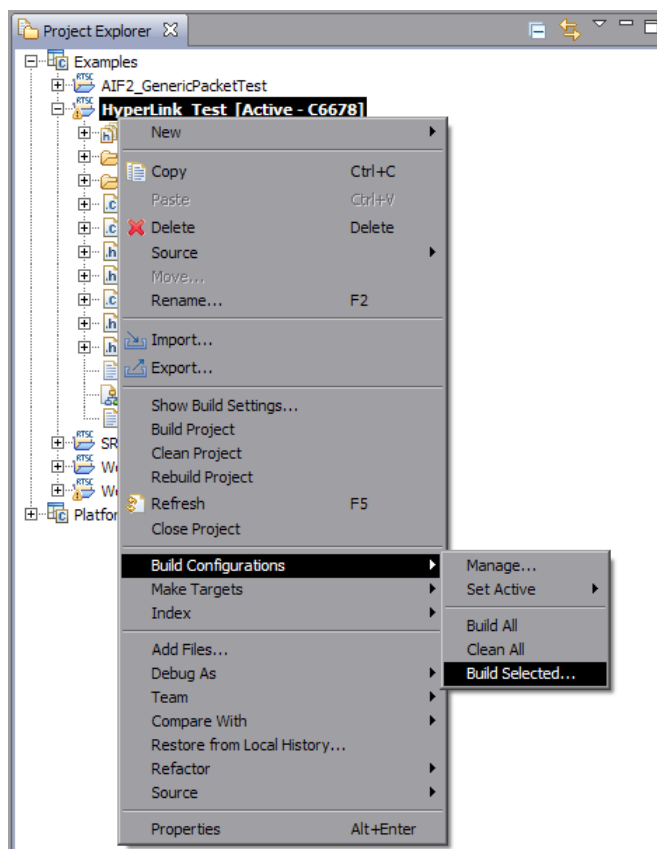


Рисунок 4-1: Пункт меню для сборки проекта

В открывшемся окне (рисунок 4-2) нужно отметить конфигурации сборки, для которых необходимо выполнить сборку и нажать на кнопку «ОК».

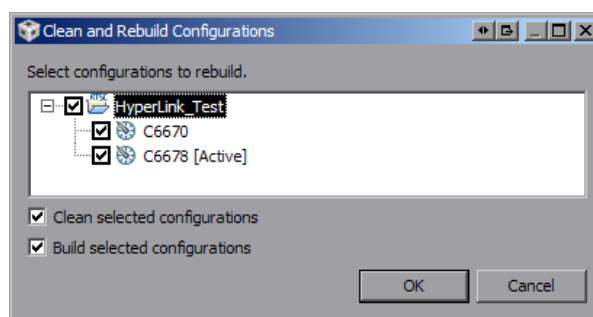


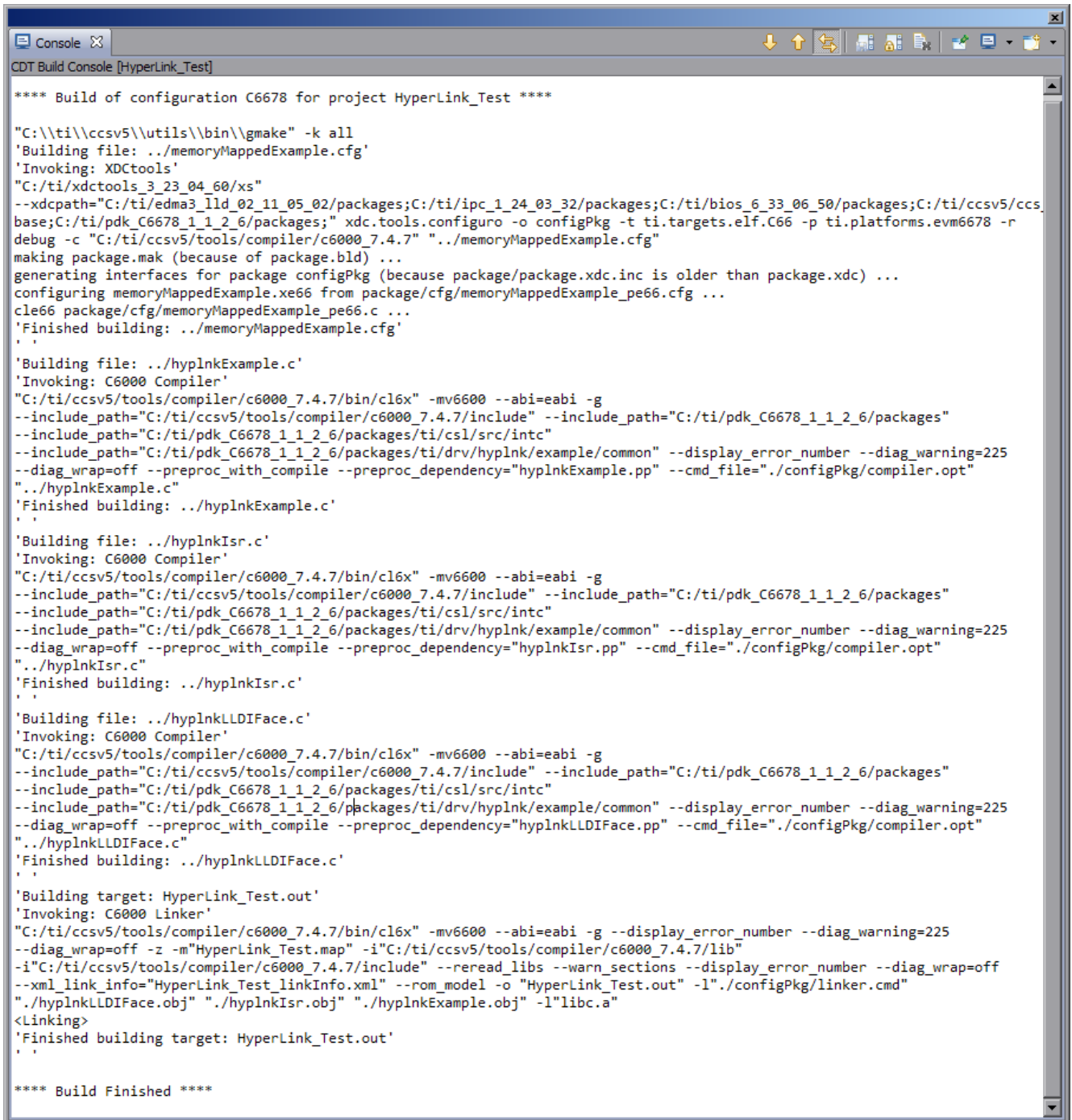
Рисунок 4-2: Окно выбора конфигураций для сборки

Проект «HyperLink_Test» имеет три конфигурации сборки:

- конфигурация «C6670» — проект для запуска на первом процессоре TMS320C6670;
- конфигурация «C6678» — проект для запуска на втором процессоре TMS320C6670.

Для запуска теста HyperLink необходимо выполнить сборку обеих конфигураций — «C6670» и «C6678».

После нажатия на кнопку «ОК» будет запущен процесс сборки выбранных конфигураций сборки. В случае успешной сборки, в окне консоли (Console) должны быть выведены сообщения, идентичные показанным на рисунке 4-3.



```

CDT Build Console [HyperLink_Test]

**** Build of configuration C6678 for project HyperLink_Test ****

"C:\ti\ccsv5\utils\bin\gmake" -k all
'Building file: ../memoryMappedExample.cfg'
'Invoking: XDCtools'
"C:/ti/xdctools_3_23_04_60/xs"
--xdcpath="C:/ti/edma3_1ld_02_11_05_02/packages;C:/ti/ipc_1_24_03_32/packages;C:/ti/bios_6_33_06_50/packages;C:/ti/ccsv5/ccs
base;C:/ti/pdk_C6678_1_1_2_6/packages;" xdc.tools.configuro -o configPkg -t ti.targets.elf.C66 -p ti.platforms.evm6678 -r
debug -c "C:/ti/ccsv5/tools/compiler/c6000_7.4.7" "../memoryMappedExample.cfg"
making package.mak (because of package.bld) ...
generating interfaces for package configPkg (because package/package.xdc.inc is older than package.xdc) ...
configuring memoryMappedExample.xe66 from package/cfg/memoryMappedExample_pe66.cfg ...
cle66 package/cfg/memoryMappedExample_pe66.c ...
'Finished building: ../memoryMappedExample.cfg'
'
'
'Building file: ../hypLnkExample.c'
'Invoking: C6000 Compiler'
"C:/ti/ccsv5/tools/compiler/c6000_7.4.7/bin/cl6x" -mv6600 --abi=eabi -g
--include_path="C:/ti/ccsv5/tools/compiler/c6000_7.4.7/include" --include_path="C:/ti/pdk_C6678_1_1_2_6/packages"
--include_path="C:/ti/pdk_C6678_1_1_2_6/packages/ti/csl/src/intc"
--include_path="C:/ti/pdk_C6678_1_1_2_6/packages/ti/drv/hypLnk/example/common" --display_error_number --diag_warning=225
--diag_wrap=off --preproc_with_compile --preproc_dependency="hypLnkExample.pp" --cmd_file="./configPkg/compiler.opt"
"../hypLnkExample.c"
'Finished building: ../hypLnkExample.c'
'
'
'Building file: ../hypLnkIsr.c'
'Invoking: C6000 Compiler'
"C:/ti/ccsv5/tools/compiler/c6000_7.4.7/bin/cl6x" -mv6600 --abi=eabi -g
--include_path="C:/ti/ccsv5/tools/compiler/c6000_7.4.7/include" --include_path="C:/ti/pdk_C6678_1_1_2_6/packages"
--include_path="C:/ti/pdk_C6678_1_1_2_6/packages/ti/csl/src/intc"
--include_path="C:/ti/pdk_C6678_1_1_2_6/packages/ti/drv/hypLnk/example/common" --display_error_number --diag_warning=225
--diag_wrap=off --preproc_with_compile --preproc_dependency="hypLnkIsr.pp" --cmd_file="./configPkg/compiler.opt"
"../hypLnkIsr.c"
'Finished building: ../hypLnkIsr.c'
'
'
'Building file: ../hypLnkLLDIface.c'
'Invoking: C6000 Compiler'
"C:/ti/ccsv5/tools/compiler/c6000_7.4.7/bin/cl6x" -mv6600 --abi=eabi -g
--include_path="C:/ti/ccsv5/tools/compiler/c6000_7.4.7/include" --include_path="C:/ti/pdk_C6678_1_1_2_6/packages"
--include_path="C:/ti/pdk_C6678_1_1_2_6/packages/ti/csl/src/intc"
--include_path="C:/ti/pdk_C6678_1_1_2_6/packages/ti/drv/hypLnk/example/common" --display_error_number --diag_warning=225
--diag_wrap=off --preproc_with_compile --preproc_dependency="hypLnkLLDIface.pp" --cmd_file="./configPkg/compiler.opt"
"../hypLnkLLDIface.c"
'Finished building: ../hypLnkLLDIface.c'
'
'
'Building target: HyperLink_Test.out'
'Invoking: C6000 Linker'
"C:/ti/ccsv5/tools/compiler/c6000_7.4.7/bin/cl6x" -mv6600 --abi=eabi -g --display_error_number --diag_warning=225
--diag_wrap=off -z -m"HyperLink_Test.map" -i"C:/ti/ccsv5/tools/compiler/c6000_7.4.7/lib"
-i"C:/ti/ccsv5/tools/compiler/c6000_7.4.7/include" --reread_libs --warn_sections --display_error_number --diag_wrap=off
--xml_link_info="HyperLink_Test_linkInfo.xml" --rom_model -o "HyperLink_Test.out" -l"./configPkg/linker.cmd"
"../hypLnkLLDIface.obj" "../hypLnkIsr.obj" "../hypLnkExample.obj" -l"libc.a"
<Linking>
'Finished building target: HyperLink_Test.out'
'
'
**** Build Finished ****

```

Рисунок 4-3: Сборка проекта теста производительности

5 Импорт и запуск целевой конфигурации модуля

Для загрузки кода приложений на модуль SVP-465, в первую очередь, необходимо запустить целевую конфигурацию модуля SVP-465. В папке «TargetConfigurations» сопроводительного диска к модулю SVP-465 находятся файлы целевых конфигураций для различных отладчиков. В данном документе рассматривается загрузка кода через отладчик Blackhawk USB560 v2 System Trace. Данному отладчику соответствует файл целевой конфигурации «SVP-465-USB560v2.ccxml», который необходимо импортировать в рабочее пространство.

Выберите пункт главного меню «View > Target Configurations» (рисунок 5-1)

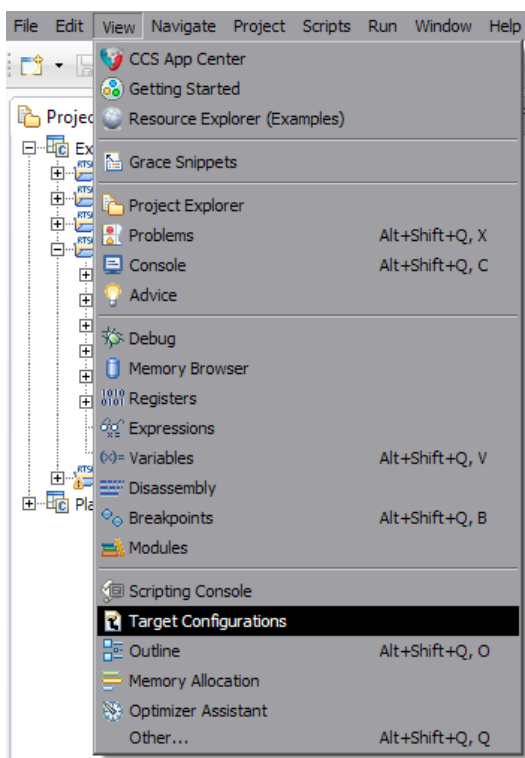


Рисунок 5-1: Пункт меню для отображения окна целевых конфигураций

В окне целевых конфигураций («Target Configurations»), нажмите правой кнопкой мыши на свободной области для вызова контекстного меню и выберите пункт «Import Target Configuration» (см. рисунок 5-2).

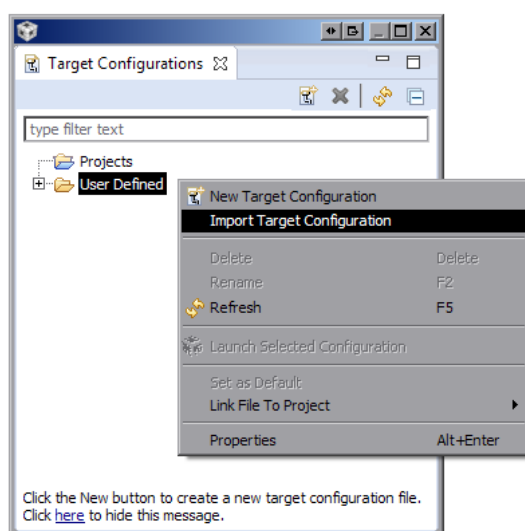


Рисунок 5-2: Меню импорта целевой конфигурации

В появившемся окне выбора файла (см. рисунок 5-3) необходимо выбрать файл «SVP-465-USB560v2.ccxml» и нажать на кнопку «Открыть». В данном документе предполагается, что папка «TargetConfigurations» с сопроводительного диска к модулю SVP-465, где расположен файл «SVP-465-USB560v2.ccxml», скопирована в папку «D:/Dev/Modules/SVP-465/TargetConfigurations».

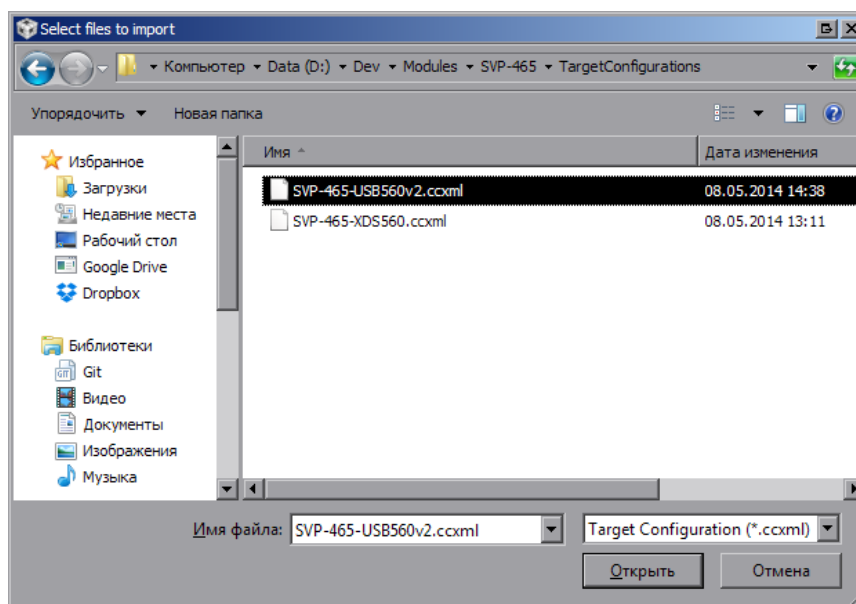


Рисунок 5-3: Окно выбора файла для импорта целевой конфигурации

После нажатия на кнопку «Открыть» появится окно выбора способа импорта файла целевой конфигурации (рисунок 5-4). Необходимо выбрать способ «Link to files» и нажать на кнопку «ОК».

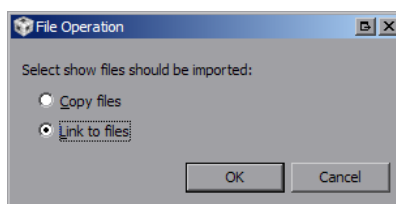


Рисунок 5-4: Окно выбора способа импорта файла целевой конфигурации

Для запуска целевой конфигурации, в окне целевых конфигураций («Target Configurations»), необходимо нажать правой кнопкой мыши на целевой конфигурации и выбрать пункт меню «Launch Selected Configuration» (см. рисунок 5-5).

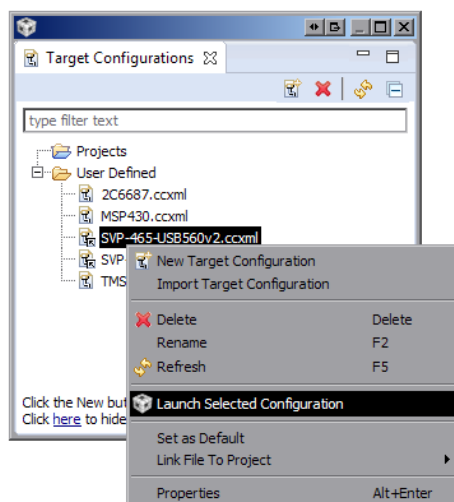


Рисунок 5-5: Запуск целевой конфигурации

После запуска целевой конфигурации модуля SVP-465, среда разработки CCS перейдет в режим отладки и в окне «Debug» будет выведен список ядер всех четырех процессоров модуля SVP-465, как показано на рисунке 5-6.

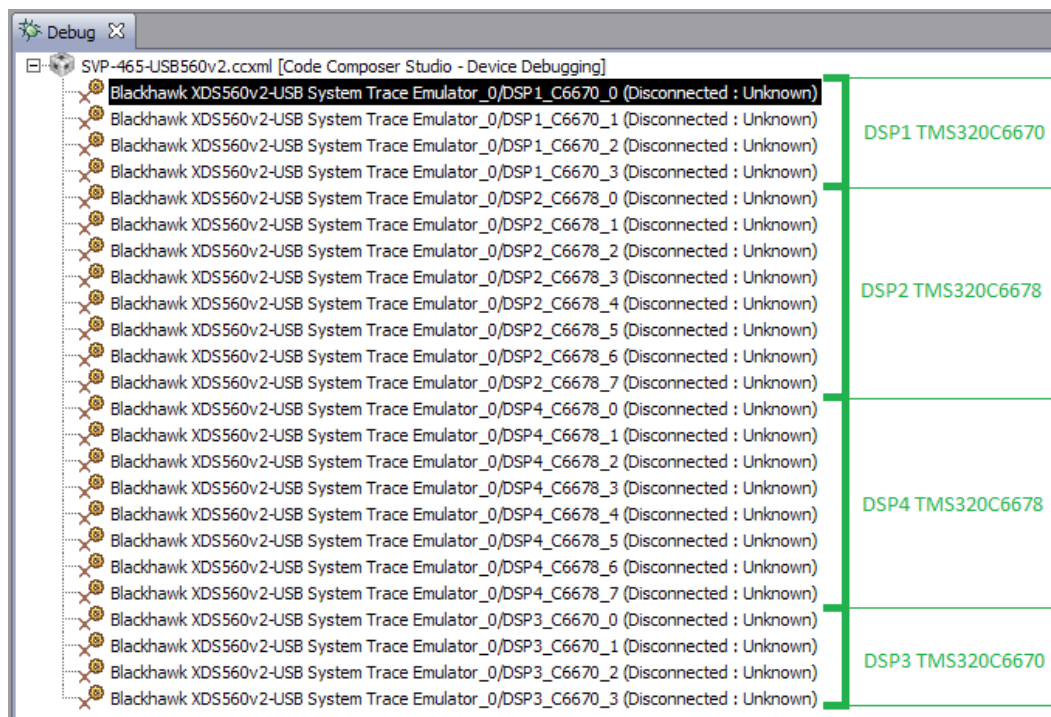


Рисунок 5-6: Список ядер процессоров модуля SVP-465

Примечание

По умолчанию, среда CCS настроена таким образом, что при запуске кода сразу на нескольких процессорах или нескольких ядрах одного процессора, вывод (C/O) со всех ядер процессоров будет выводиться в одно и то же окно «Console». В приложении A даны инструкции по настройке отдельного вывода (C/O) каждого ядра процессора в отдельное окно «Console».

6 Загрузка кода

Запустите целевую конфигурацию модуля SVP-465, как описано в разделе 5.

Перед загрузкой кода на ядра процессора, необходимо выполнить группировку тех ядер, на которых будет выполняться код теста HyperLink. В данном случае, это ядра «DSP1_C6670_0» и «DSP2_C6678_0».

Выберите ядра «DSP1_C6670_0» и «DSP2_C6678_0» в окне «Debug» (щелкните левой кнопкой мыши на ядре «DSP1_C6670_0», затем, зажав на клавиатуре клавишу Ctrl, щелкните на ядре «DSP2_C6678_0»). Щелкните правой кнопкой мыши на последнем выбранном ядре, и выберите пункт меню «Group core(s)» (см. рисунок 6-1).

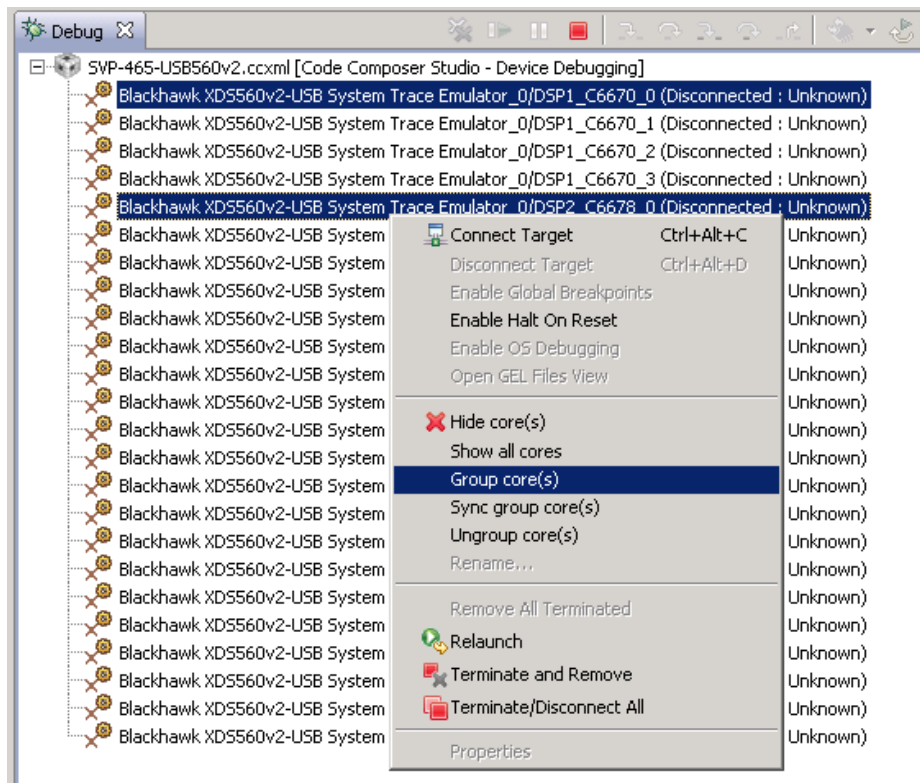


Рисунок 6-1: Группировка ядер процессоров

Щелкните правой кнопкой мыши на названии группы «Group 1» и выберите пункт меню «Connect Target». После выполнения подключения к ядрам процессоров, окно «Debug» должно выглядеть, как показано на рисунке 6-2.

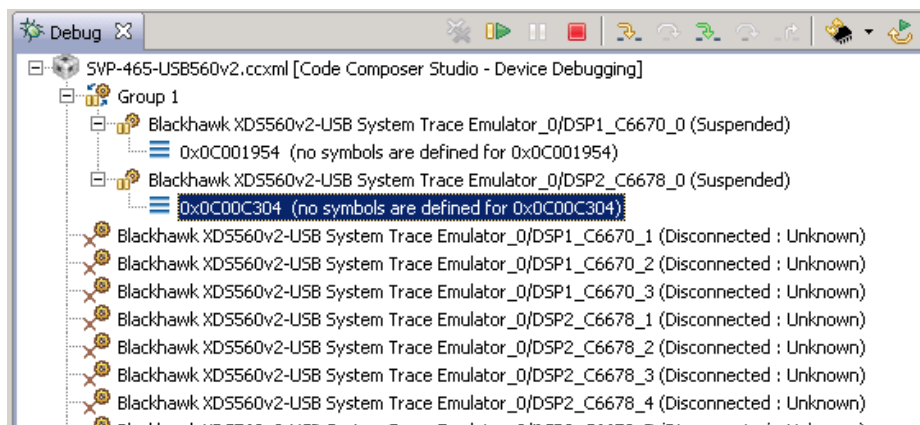


Рисунок 6-2: Ядра процессоров после выполнения подключения

Выберите ядро «DSP1_C6670_0» в окне «Debug» (щелкните левой кнопкой мыши по названию ядра). Выберите пункт главного меню «Run > Load > Load Program...» (рисунок 6-3).

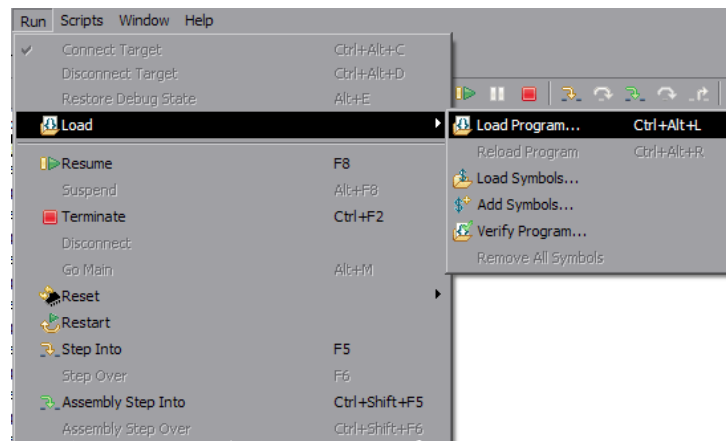


Рисунок 6-3: Меню загрузки кода на ядро процессора

В открывшемся окне (рисунок 6-4) нажмите на кнопку «Browse project...».

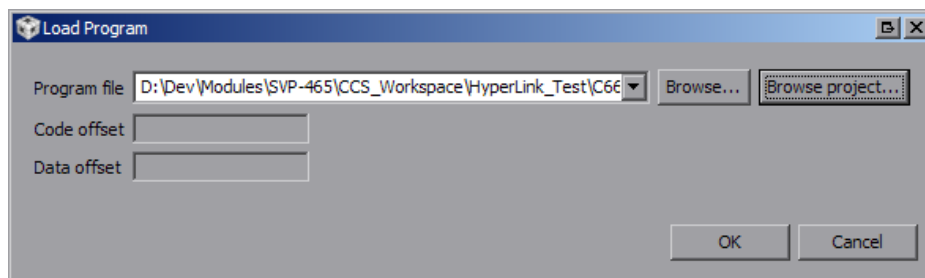
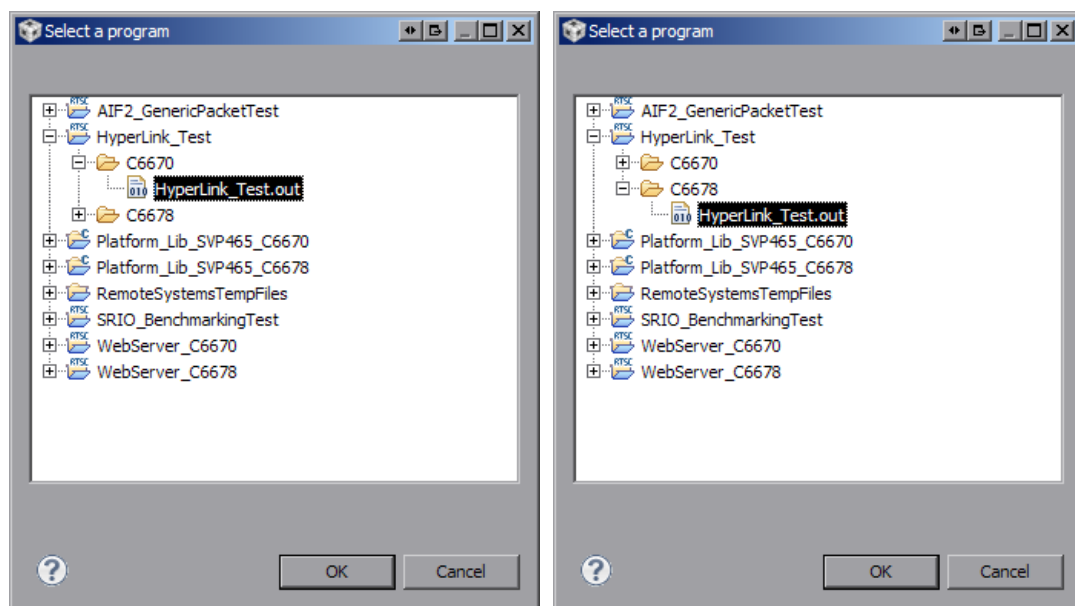


Рисунок 6-4: Окно загрузки кода на ядро процессора

Выберите файл «HyperLink_Test.out» конфигурации сборки «C6670» из проекта «HyperLink_Test», как показано на рисунке 6-5а, и нажмите на кнопку «ОК». В окне, показанном на рисунке 6-4, также нажмите на кнопку «ОК».



а) Процессор TMS320C6670

б) Процессор TMS320C6678

Рисунок 6-5: Окно выбора файла для загрузки на ядро процессора

После загрузки кода на процессор TMS320C6670 необходимо, также, выполнить загрузку кода на процессор TMS320C6678. Для этого, выберите ядро «DSP2_C6678_0» в окне «Debug» и аналогичным образом выполните загрузку файла «HyperLink_Test.out», но из конфигурации сборки «C6678» (см. рисунок 6-56).

После загрузки кода теста HyperLink на ядра процессоров TMS320C6670 и TMS320C6678, окно «Debug» должно выглядеть, как показано на рисунке 6-6.

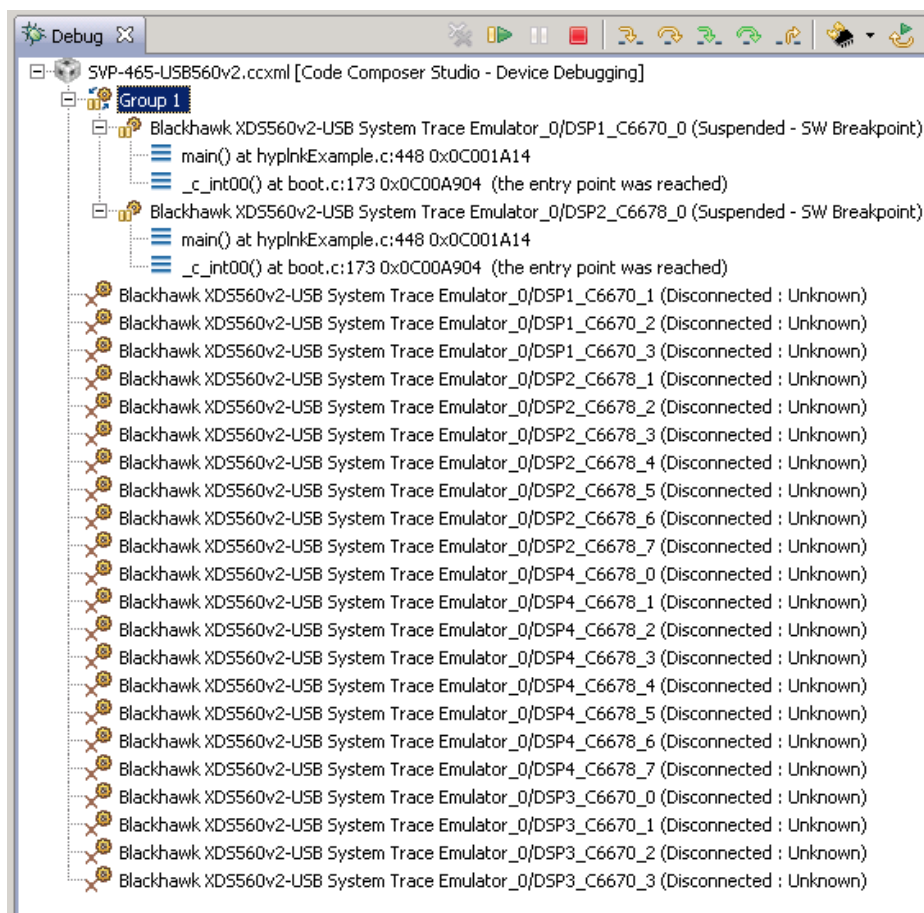



Рисунок 6-6: Внешний вид окна «Debug» после загрузки кода на ядра процессоров


7 Запуск теста HyperLink

Для запуска теста HyperLink выполните шаги, описанные в процедуре 7-1.

Процедура 7-1. Запуск теста HyperLink

1. Выполнить загрузку кода теста HyperLink на ядра процессоров, как описано в разделе 6.
2. Выбрать в окне «Debug» группу «Group 1» щелкнув по ней левой кнопкой мыши.
3. Запустить выполнение кода, одновременно на обоих процессорах, нажав на кнопку  («Resume»). На рисунке 6-6 кнопка «Resume» расположена в верхнем правом углу.

В случае успешного запуска теста HyperLink, вывод с ядра «DSP1_C6670_0» в окно «Console» должен выглядеть аналогично приведенному в листинге 7-1, а вывод с ядра «DSP2_C6678_0» должен выглядеть аналогично приведенному в листинге 7-2.

В листингах 7-1 и 7-2 приведен вывод только шести итераций теста. В случае сборки и запуска теста HyperLink с конфигурацией по умолчанию, тест будет выполняться бесконечно. Для остановки выполнения теста HyperLink, выберите в окне «Debug» группу «Group 1» и нажмите на кнопку  («Suspend»).

Внимание



Тест HyperLink осуществляет вывод только в отладочную консоль CCS. В UART вывод теста не дублируется.

Примечание

В приложении A приведена информация по настройке отдельного вывода с ядер процессоров. В том случае, если не будет выполнена настройка отдельного вывода с ядер процессоров, сообщения с обоих ядер будут выводиться в одном общем окне «Console».

Листинг 7-1: Вывод в консоль теста HyperLink с ядра «DSP1_C6670_0»

```

1  Version #: 0x01000105; string HYPLNK LLD Revision: 01.00.01.05:Nov 19 2012:16:17:16
2  About to do system setup (PLL, PSC, and DDR)
3  Power domain is already enabled. You probably re-ran without device reset (which is OK)
4  Constructed SERDES configs: PLL=0x00000064; RX=0x0046c495; TX=0x000ccc315
5  system setup worked
6  About to set up HyperLink Peripheral
7  ===== begin registers before initialization =====
8  Revision register contents:
9    Raw      = 0x4e901900
10 Status register contents:
11   Raw      = 0x00002004
12 Link status register contents:
13   Raw      = 0x00000000
14 Control register contents:
15   Raw      = 0x00000001
16 Control register contents:
17   Raw      = 0x00000000
18 ===== end registers before initialization =====
19 SERDES_STS (32 bits) contents: 0x00000001; lock = 1
20 ===== begin registers after initialization =====
21 Status register contents:
22   Raw      = 0x04400005
23 Link status register contents:
24   Raw      = 0xccf00cf0
25 Control register contents:
26   Raw      = 0x00006200
27 ===== end registers after initialization =====
28 Waiting 5 seconds to check link stability
29 Precursors 0 Analysis: 1,0,1,0,1,0,0,1
30 Postcursors: 19 Analysis: 1,0,0,1,1,0,0,1

```

```

31 Link seems stable
32 About to try to read remote registers
33 ===== begin REMOTE registers after initialization =====
34 Status register contents:
35   Raw      = 0x0440000b
36 Link status register contents:
37   Raw      = 0xfdf0bdf0
38 Control register contents:
39   Raw      = 0x00006200
40 ===== end REMOTE registers after initialization =====
41 Peripheral setup worked
42 About to read/write once
43 Single write test passed
44 About to pass 65536 tokens; iteration = 0
45 === this is not an optimized example ===
46 Link Speed is 4 * 6.25 Gbps
47 Passed 65536 tokens round trip (read+write through hyplnk) in 16224 Mcycles
48 Approximately 247558 cycles per round-trip
49 === this is not an optimized example ===
50 Checking statistics
51 About to pass 65536 tokens; iteration = 1
52 === this is not an optimized example ===
53 Link Speed is 4 * 6.25 Gbps
54 Passed 65536 tokens round trip (read+write through hyplnk) in 16224 Mcycles
55 Approximately 247558 cycles per round-trip
56 === this is not an optimized example ===
57 Checking statistics
58 About to pass 65536 tokens; iteration = 2
59 === this is not an optimized example ===
60 Link Speed is 4 * 6.25 Gbps
61 Passed 65536 tokens round trip (read+write through hyplnk) in 16224 Mcycles
62 Approximately 247559 cycles per round-trip
63 === this is not an optimized example ===
64 Checking statistics
65 About to pass 65536 tokens; iteration = 3
66 === this is not an optimized example ===
67 Link Speed is 4 * 6.25 Gbps
68 Passed 65536 tokens round trip (read+write through hyplnk) in 16224 Mcycles
69 Approximately 247558 cycles per round-trip
70 === this is not an optimized example ===
71 Checking statistics
72 About to pass 65536 tokens; iteration = 4
73 === this is not an optimized example ===
74 Link Speed is 4 * 6.25 Gbps
75 Passed 65536 tokens round trip (read+write through hyplnk) in 16224 Mcycles
76 Approximately 247558 cycles per round-trip
77 === this is not an optimized example ===
78 Checking statistics
79 About to pass 65536 tokens; iteration = 5

```

Листинг 7-2: Вывод в консоль теста HyperLink с ядра «DSP1_C6678_0»

```

1 Version #: 0x01000105; string HYPLNK LLD Revision: 01.00.01.05:Nov 19 2012:16:04:15
2 About to do system setup (PLL, PSC, and DDR)
3 Power domain is already enabled. You probably re-ran without device reset (which is OK)
4 Constructed SERDES configs: PLL=0x00000064; RX=0x0046c495; TX=0x000cc315
5 system setup worked
6 About to set up HyperLink Peripheral
7 ===== begin registers before initialization =====
8 Revision register contents:
9   Raw      = 0x4e901900
10 Status register contents:
11   Raw      = 0x00002004
12 Link status register contents:
13   Raw      = 0x00000000
14 Control register contents:
15   Raw      = 0x00000001
16 Control register contents:
17   Raw      = 0x00000000
18 ===== end registers before initialization =====

```

```
19 SERDES_STS (32 bits) contents: 0x00000001; lock = 1
20 ===== begin registers after initialization =====
21 Status register contents:
22   Raw      = 0x04400005
23 Link status register contents:
24   Raw      = 0xccf00cf0
25 Control register contents:
26   Raw      = 0x00006200
27 ===== end registers after initialization =====
28 Waiting 5 seconds to check link stability
29 Precursors 0 Analysis: 1,0,1,0,1,0,1,0
30 Postcursors: 19 Analysis: 1,0,0,0,0,0,0,0,1
31 Link seems stable
32 About to try to read remote registers
33 ===== begin REMOTE registers after initialization =====
34 Status register contents:
35   Raw      = 0x0440080f
36 Link status register contents:
37   Raw      = 0xfdf0bdf0
38 Control register contents:
39   Raw      = 0x00006200
40 ===== end REMOTE registers after initialization =====
41 Peripheral setup worked
42 About to read/write once
43 Single write test passed
44 About to pass 65536 tokens; iteration = 0
45 === this is not an optimized example ===
46 Link Speed is 4 * 6.25 Gbps
47 Passed 65536 tokens round trip (read+write through hyplnk) in 16503 Mcycles
48 Approximately 251826 cycles per round-trip
49 === this is not an optimized example ===
50 Checking statistics
51 About to pass 65536 tokens; iteration = 1
52 === this is not an optimized example ===
53 Link Speed is 4 * 6.25 Gbps
54 Passed 65536 tokens round trip (read+write through hyplnk) in 16503 Mcycles
55 Approximately 251827 cycles per round-trip
56 === this is not an optimized example ===
57 Checking statistics
58 About to pass 65536 tokens; iteration = 2
59 === this is not an optimized example ===
60 Link Speed is 4 * 6.25 Gbps
61 Passed 65536 tokens round trip (read+write through hyplnk) in 16503 Mcycles
62 Approximately 251827 cycles per round-trip
63 === this is not an optimized example ===
64 Checking statistics
65 About to pass 65536 tokens; iteration = 3
66 === this is not an optimized example ===
67 Link Speed is 4 * 6.25 Gbps
68 Passed 65536 tokens round trip (read+write through hyplnk) in 16503 Mcycles
69 Approximately 251826 cycles per round-trip
70 === this is not an optimized example ===
71 Checking statistics
72 About to pass 65536 tokens; iteration = 4
73 === this is not an optimized example ===
74 Link Speed is 4 * 6.25 Gbps
75 Passed 65536 tokens round trip (read+write through hyplnk) in 16503 Mcycles
76 Approximately 251827 cycles per round-trip
77 === this is not an optimized example ===
78 Checking statistics
79 About to pass 65536 tokens; iteration = 5
```

Приложение А Разделение вывода сообщений (CIO) ядер процессоров

По умолчанию, среда CCS настроена таким образом, что при запуске кода сразу на нескольких процессорах или нескольких ядрах одного процессора, вывод (CIO) со всех ядер процессоров будет выводиться в одно и то же окно «Console». В данном приложении даны инструкции по настройке отдельного вывода (CIO) каждого ядра процессора в отдельное окно «Console».

После запуска целевой конфигурации, в окне «Debug», нажмите правой кнопкой мыши на названии файла целевой конфигурации и выберите пункт меню «Edit X...», где X — имя файла целевой конфигурации (см. рисунок A-1).

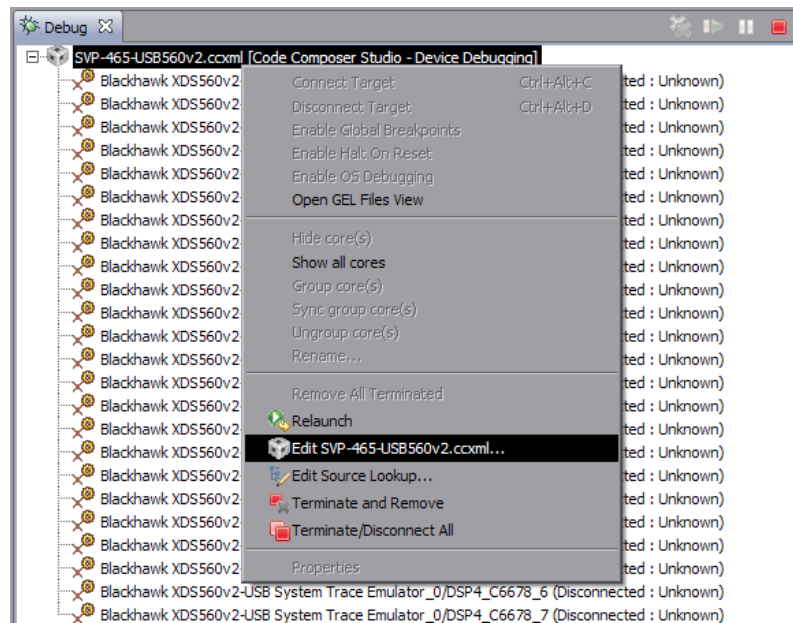


Рисунок A-1: Контекстное меню целевой конфигурации

В открывшемся окне, снимите галочку с опции «Use the same console for the CIO of all CPUs» (см. рисунок A-2) и нажмите на кнопки «Apply» и «Continue».

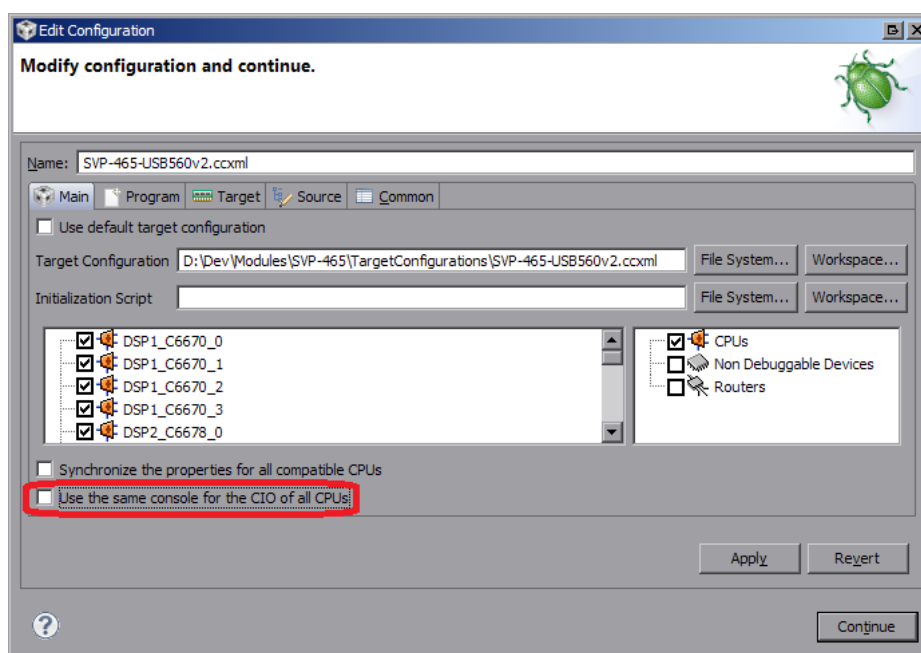



Рисунок A-2: Окно настроек целевой конфигурации

В окне «Console» нажмите на кнопку  («Open Console») и выберите пункт меню «New Console View» (см. рисунок A-3).

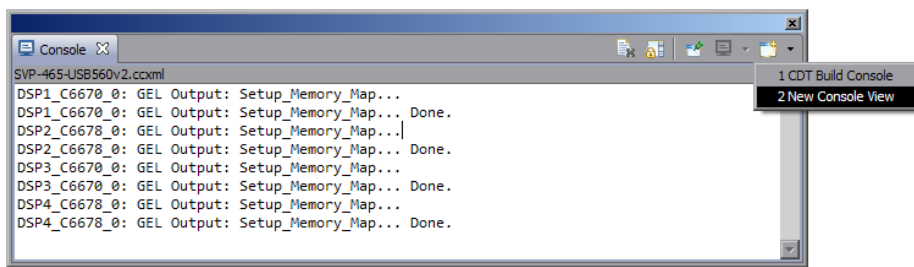


Рисунок A-3: Открытие второго окна «Console»

После этого, будет открыто еще одно окно «Console», которое можно переместить в любое удобное место окна CCS, например, как показано на рисунке A-4.

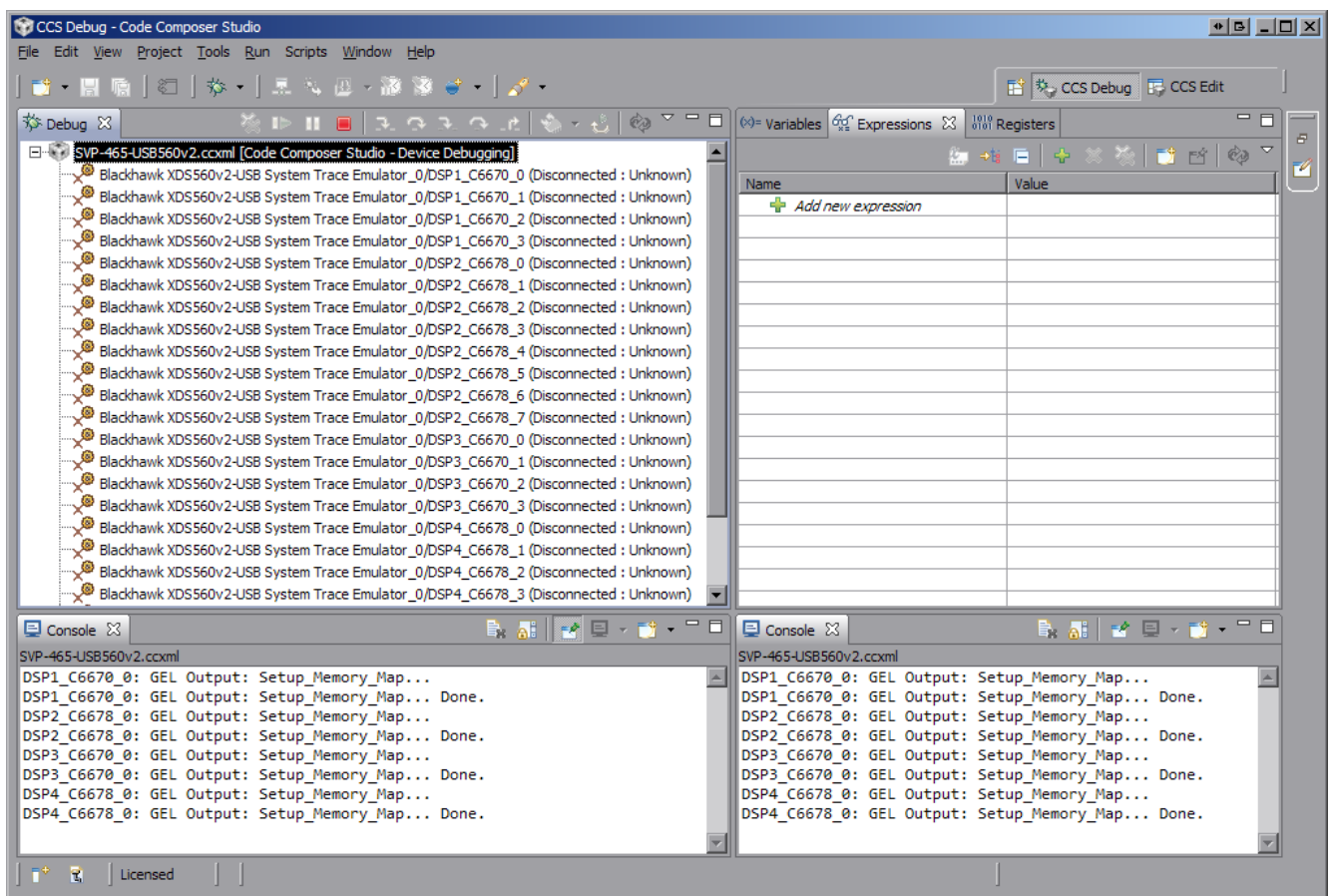



Рисунок A-4: Два окна «Console»

Теперь, если запустить приложения на различных ядрах процессоров, для вывода с каждого отдельного ядра будет происходить в отдельное окно. Для того, чтобы выбрать вывод какого ядра необходимо отображать в конкретном окне «Console», необходимо нажать на кнопку  («Display Selected Console») этого окна и выбрать пункт меню соответствующий нужному ядру (см. рисунок A-5).

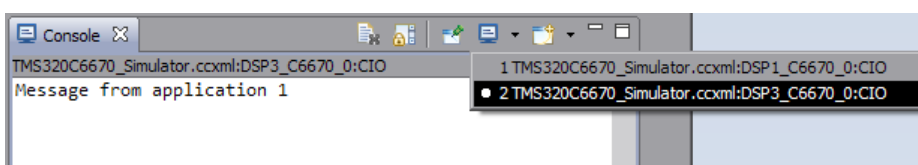


Рисунок A-5: Выбор ядра для отображения вывода в окне «Console»

Следует иметь в виду, что в данном списке (рисунок A-5) можно выбрать только те ядра, с которых уже был произведен какой либо вывод.

Например, если на ядре «DSP1_C6670_0» запустить простое приложение, выводящее сообщение «Message from application 1», а на ядре «DSP3_C6670_0» запустить второе приложение, выводящее сообщение «Message from application 2», то список доступных консолей будет соответствовать рисунку A-5, а вывод в два окна будет выглядеть, как показано на рисунке A-6.

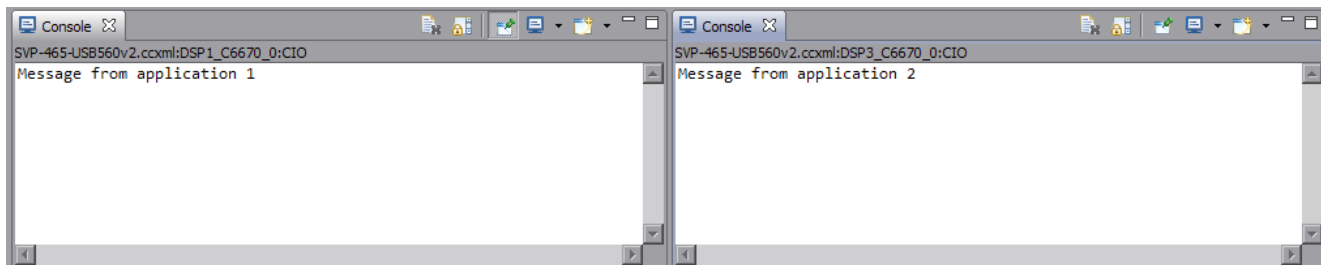


Рисунок A-6: Вывод сообщений с двух ядер в два окна «Console»

Для закрепления окна «Console» за конкретным ядром используется кнопка  («Pin Console»).