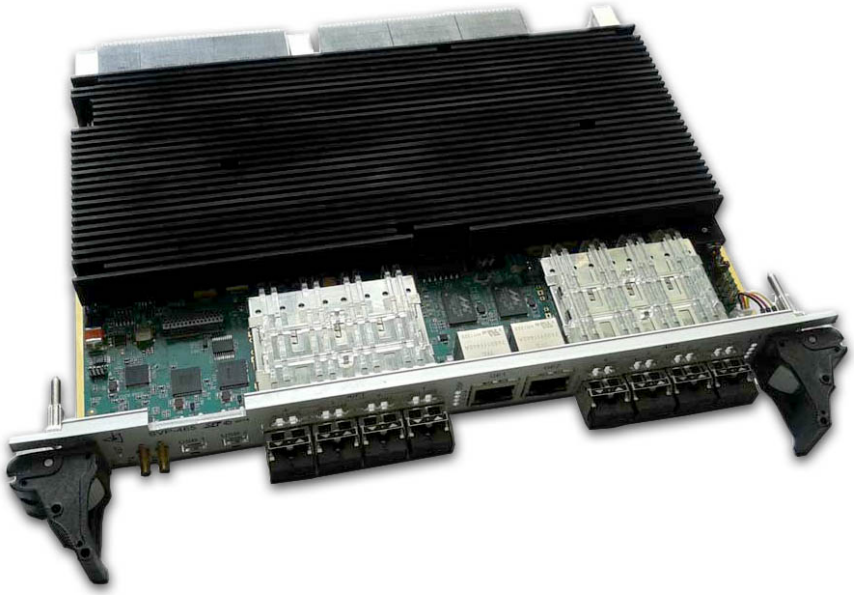
Scan Engineering Telecom SPb

SVP-465. Сборка и запуск теста AIF2

Руководство пользователя

Версия 1.0



Код документа: UG-SVP-465-AIF2
Дата сборки: 27 мая 2015 г.
Листов в документе: 25

http://www.setdsp.ru

Содержание

Список рисунков

Список таблиц

Список листингов

Список процедур

Перечень сокращений и условных обозначений

AIF2	Antenna Interface 2	5–7, 9–13, 18, 20–22
CCS	Code Composer Studio	6, 7, 9, 17, 23, 24
CIO	Console Input/Output	17, 23
CPRI	Common Public Radio Interface	5, 11, 12
DIO	Direct Input/Output	11, 12
EDMA	Enhanced Direct Memory Access	6
FPGA	Field-Programmable Gate Array	6
IPC	Inter Process Communication	6
LLD	Low Level Driver	6, 9
MCSDK	MultiCore Software Development Kit	6
OBSAI	Open Base Station Architecture Initiative	5, 11, 12
PDK	Platform Development Kit	6, 9
TI	Texas Instruments	6
ОС	Операционная Система	6

1 Общие сведения

В данном документе описан процесс сборки из исходных кодов и запуск теста передачи данных через интерфейс AIF2 на модуле SVP-465. Тест выполняет передачу данных по интерфейсу AIF2 с одного процессора TMS320C6670 на второй процессор TMS320C6670, которые расположены на одном модуле SVP-465. В данном документе, предполагается, что процессоры TMS320C6670 модуля SVP-465 соединены между собой оптическими кабелями через приемо-передающие оптические каналы OBSAI/CPRI на передней панели, как показано на рисунке 1-1.

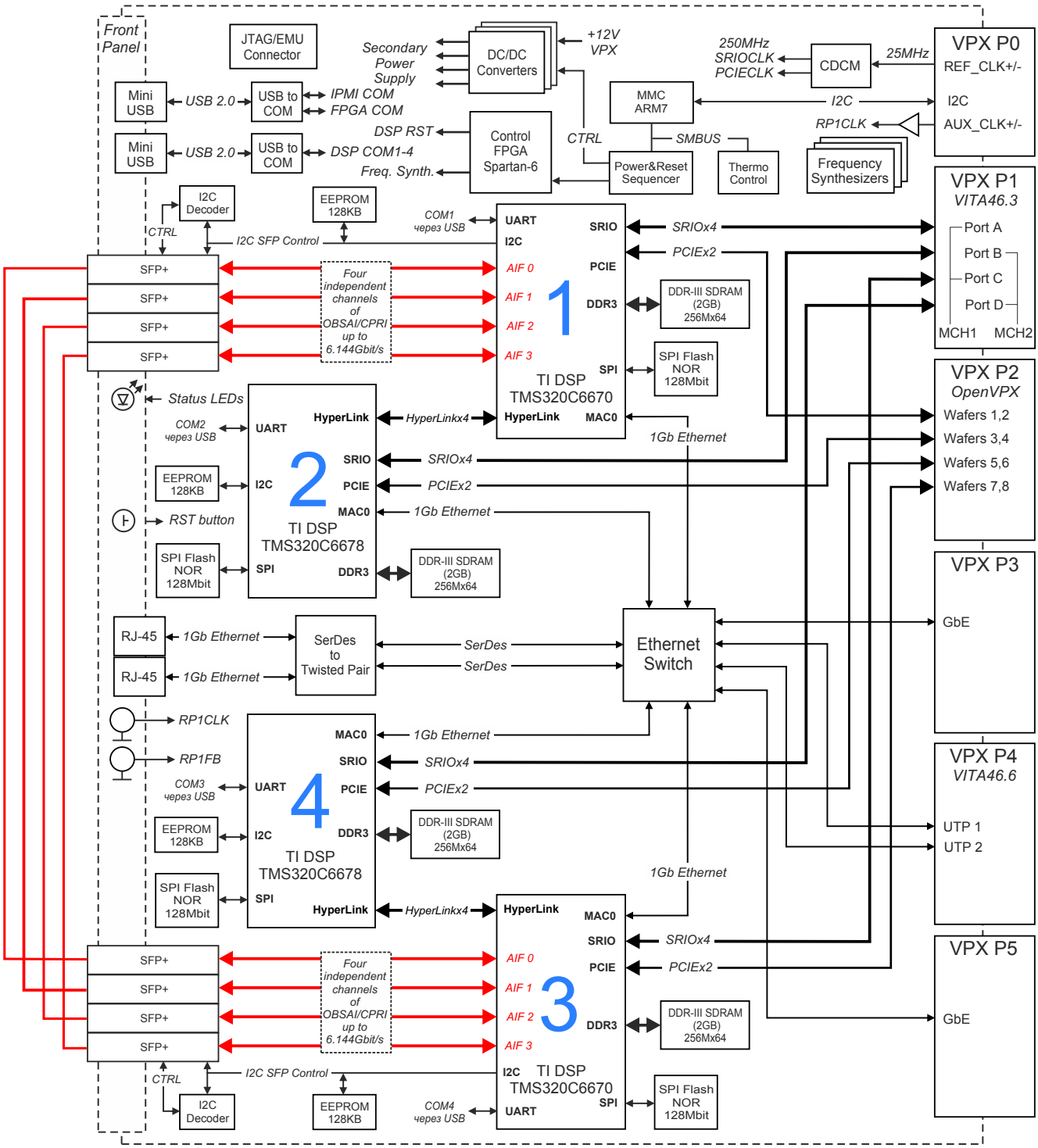


Рисунок 1-1: Структурная схема модуля SVP-465

Для сборки теста передачи данных AIF2 требуется установленная система разработки CCS (Code Composer Studio) компании TI. Сборка и запуск теста передачи данных через интерфейс AIF2 были проверены на CCS версии 5.4.0.00091 под ОС Windows 7 (64-bit). Среду разработки CCS можно бесплатно скачать с сайта производителя[1].

Тест передачи данных через интерфейсу AIF2 написан на основе кода теста передачи данных из состава библиотеки TI MCSDK (MultiCore Software Development Kit) для процессора TMS320C6670.

Для сборки проекта теста передачи данных AIF2 необходимы следующие установленные пакеты:

- PDK C6670 версии 1.1.2.6;
- IPC (Inter Process Communication) версии 1.24.3.32;
- EDMA3 LLD версии 2.11.5 (входит в состав PDK C6670);
- AIF2 LLD версии 1.1.0.2 (имеется на сопроводительном диске к модулю SVP-465);
- 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 и запуск проекта возможен с использованием пакетов более ранних версий, однако, в этом случае, не гарантируется правильная работа теста передачи данных через интерфейс AIF2.

Внимание

В том случае, если используемая версия AIF2 LLD ниже 1.1.0.2, описываемый в данном документе тест передачи данных будет работать только на AIF2 Link 0. Инструкция по установке AIF2 LLD версии 1.1.0.2 приведена в разделе 3.

Для правильной работы теста передачи данных AIF2 на модуле SVP-465 должна выполняться схема синхронизации подсистем AIF2 двух процессоров TMS320C6670, показанная на рисунке 1-2. Данная схема синхронизации обеспечивается заводской прошивкой в FPGA.

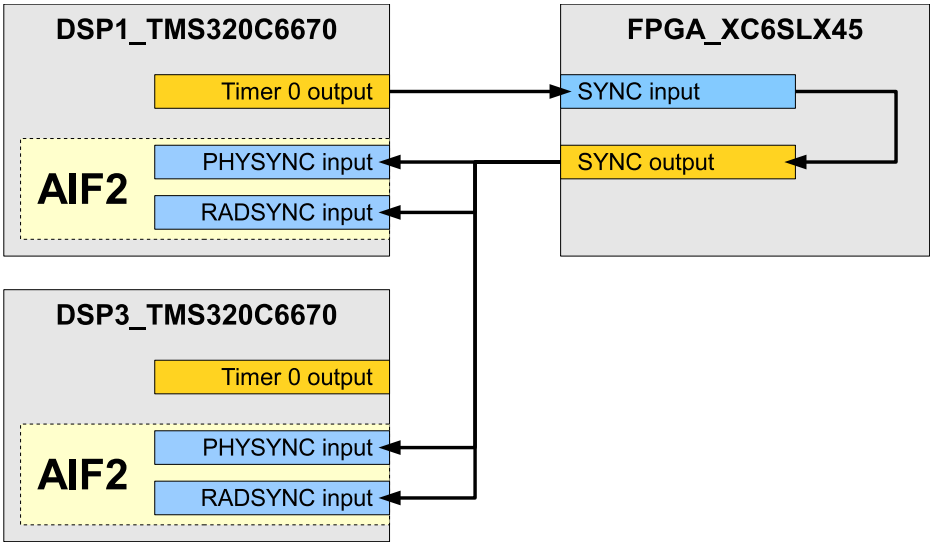


Рисунок 1-2: Схема синхронизации подсистем AIF2 двух процессоров TMS320C6670 модуля SVP-465

[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»).

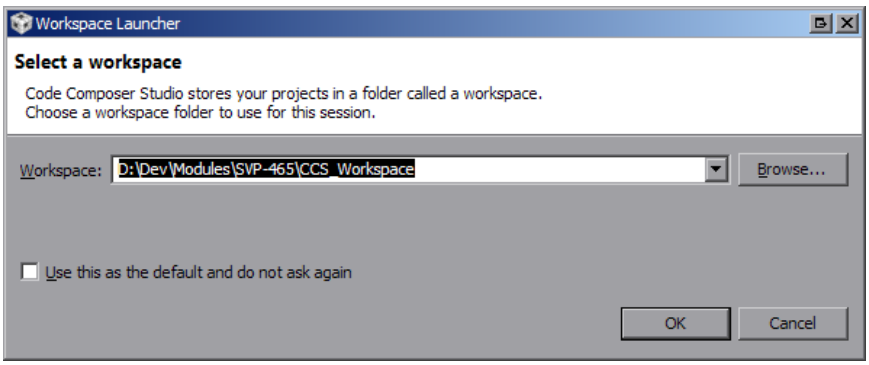
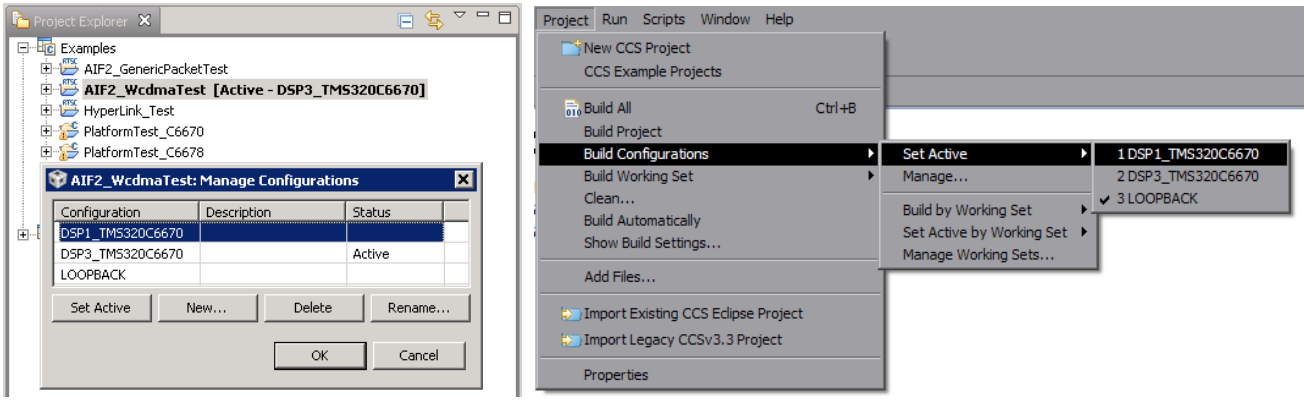


Рисунок 2-1: Выбор пути рабочего пространства (workspace) в CCS

Исходные коды тест передачи данных через интерфейс AIF2 расположены в проекте «AIF2_WcdmaTest» (см. рисунок 2-2а). Проект «AIF2_WcdmaTest» предназначен для запуска только на процессорах TMS320C6670 (на процессорах TMS320C6678 отсутствует AIF2).



а) Обозревателя проектов и окно конфигураций сборки проекта

б) Выбор активной конфигурации сборки проекта

Рисунок 2-2: Проект «AIF2_WcdmaTest» и его конфигурации сборки

Проект «AIF2_WcdmaTest» имеет три конфигурации сборки (в нижней части рисунка 2-2а показано окно управления конфигурациями сборки):

- конфигурация «LOOPBACK» — проект с включенной локальной петлей на AIF2;
- конфигурация «DSP1_TMS320C6670» — проект для запуска на первом процессоре TMS320C6670;
- конфигурация «DSP3_TMS320C6670» — 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 Установка AIF2 LLD версии 1.1.0.2

В состав PDK C6670 версии 1.1.2.6 входит AIF2 LLD версии 1.0.0.6, который позволяет запускать тест передачи данных через инетрфейс AIF2 только с использованием AIF2 Link 0.

На сопроводительном диске к модулю SVP-465 имеется архив «AIF2_KeyStone1_1_1_0_2_engineering.rar» в папке «Install», который содержит AIF2 LLD версии 1.1.0.2. В данной версии драйвера исправлены ошибки, не позволяющие работать с другими линками, кроме AIF2 Link 0.

Для его установки, необходимо распаковать данный архив в папку «C:/ti/pdk_c6670_1_1_2_6/packages» с заменой всех файлов (предполагается, что CCS установлена в папку «C:/ti»).

4 Конфигурация проекта

Для применения выполненных изменений в конфигурации, необходимо выполнить сборку проекта в соответствии с процедурой, описанной в разделе 5.

Конфигурация тестов AIF2 определяется в файле «wcdma.c» проекта «AIF2_WcdmaTest». Фрагмент файла «wcdma», отвечающий за конфигурацию тестов AIF2 приведен в листинге 4-1.

Листинг 4-1: Файл «wcdma.c» (фрагмент с конфигурацией тестов)

```
87  #define TEST_NUM        4
88
89  volatile unsigned char testEnable[TEST_NUM] =
90  {  // Each of those need to be run individually
91      1,
92      1,
93      1,
94      1
95  };
96
97  typedef struct
98  {
99      const unsigned char      name[64];                          // Test name
100     CSL_Aif2LinkProtocol     protocol;                          // AIF protocol used
101     Uint32                   linkEnable[AIF_MAX_NUM_LINKS];     // AIF link enable or disable, 1 or 0
102     Uint32                   linkRate[AIF_MAX_NUM_LINKS];       // AIF link rate (1=1x; 2=2x; 4=4x; 8=8x)
103     CSL_Aif2LinkDataType     outboundDataType[AIF_MAX_NUM_LINKS];   // AIF link data type for outbound burst traffic
104     CSL_Aif2DataWidth        outboundDataWidth[AIF_MAX_NUM_LINKS];  // AIF link data width for outbound burst traffic
105     CSL_Aif2LinkDataType     inboundDataType[AIF_MAX_NUM_LINKS];    // AIF link data type for inbound burst traffic
106     CSL_Aif2DataWidth        inboundDataWidth[AIF_MAX_NUM_LINKS];   // AIF link data width for inbound burst traffic
107     CSL_Aif2DioEngineIndex   dioEngine[AIF_MAX_NUM_LINKS];          // AIF link DIO engine used
108     Uint32                   nbLinkDio[AIF2_MAX_NUM_DIO_ENGINE];    // AIF number of links per DIO engine
109 } TestObj;
110
111 volatile TestObj testObjTab[TEST_NUM] = {
112     {   // OBSAI 8x for UL WCDMA - Links 0--3 - DIO 0 and DIO 1
113         "OBSAI multi-link 4x UL 8-bit (4 x 3.072 Gbps)", // test name
114         CSL_AIF2_LINK_PROTOCOL_OBSAI,
115         // link0        link1          link2          link3          link4          link5
116         {1,             1,             1,             1,             0,             0           },  // link enable
117         {4,             4,             4,             4,             4,             4           },  // link rate
118         {DATA_TYPE_UL,  DATA_TYPE_UL,  DATA_TYPE_UL,  DATA_TYPE_UL,  DATA_TYPE_UL,  DATA_TYPE_UL },  // outboundDataType
119         {DATA_WIDTH_8,  DATA_WIDTH_8,  DATA_WIDTH_8,  DATA_WIDTH_8,  DATA_WIDTH_8,  DATA_WIDTH_8 },  // outboundDataWidth
120         {DATA_TYPE_UL,  DATA_TYPE_UL,  DATA_TYPE_UL,  DATA_TYPE_UL,  DATA_TYPE_UL,  DATA_TYPE_UL },  // inboundDataType
121         {DATA_WIDTH_8,  DATA_WIDTH_8,  DATA_WIDTH_8,  DATA_WIDTH_8,  DATA_WIDTH_8,  DATA_WIDTH_8 },  // inboundDataWidth
122         {CSL_AIF2_DIO_ENGINE_0, CSL_AIF2_DIO_ENGINE_0, CSL_AIF2_DIO_ENGINE_1, CSL_AIF2_DIO_ENGINE_1, CSL_AIF2_DIO_ENGINE_0, CSL_AIF2_DIO_ENGINE_0}, // DIO engine
123         {2,             2,             0},
124     },
125     {   // OBSAI 8x for UL WCDMA - Link 0 - DIO 0
126         "OBSAI one link 8x UL 8-bit (6.250 Gbps)", // test name
127         CSL_AIF2_LINK_PROTOCOL_OBSAI,
128         // link0        link1          link2          link3          link4          link5
129         {1,             0,             0,             0,             0,             0           },  // link enable
130         {8,             8,             8,             8,             8,             8           },  // link rate
131         {DATA_TYPE_UL,  DATA_TYPE_UL,  DATA_TYPE_UL,  DATA_TYPE_UL,  DATA_TYPE_UL,  DATA_TYPE_UL },  // outboundDataType
132         {DATA_WIDTH_8,  DATA_WIDTH_8,  DATA_WIDTH_8,  DATA_WIDTH_8,  DATA_WIDTH_8,  DATA_WIDTH_8 },  // outboundDataWidth
133         {DATA_TYPE_UL,  DATA_TYPE_UL,  DATA_TYPE_UL,  DATA_TYPE_UL,  DATA_TYPE_UL,  DATA_TYPE_UL },  // inboundDataType
134         {DATA_WIDTH_8,  DATA_WIDTH_8,  DATA_WIDTH_8,  DATA_WIDTH_8,  DATA_WIDTH_8,  DATA_WIDTH_8 },  // inboundDataWidth
135         {CSL_AIF2_DIO_ENGINE_0, CSL_AIF2_DIO_ENGINE_0, CSL_AIF2_DIO_ENGINE_1, CSL_AIF2_DIO_ENGINE_1, CSL_AIF2_DIO_ENGINE_0, CSL_AIF2_DIO_ENGINE_0}, // DIO engine
136         {1,             0,             0},
137     },
138     {   // OBSAI 8x for UL WCDMA - Link 0 - DIO 0
139         "OBSAI one link 8x DL 16-bit (6.250 Gbps)", // test name
140         CSL_AIF2_LINK_PROTOCOL_OBSAI,
141         // link0        link1          link2          link3          link4          link5
142         {1,             0,             0,             0,             0,             0           },  // link enable
143         {8,             8,             8,             8,             8,             8           },  // link rate
144         {DATA_TYPE_DL,  DATA_TYPE_DL,  DATA_TYPE_DL,  DATA_TYPE_DL,  DATA_TYPE_DL,  DATA_TYPE_DL },  // outboundDataType
145         {DATA_WIDTH_16, DATA_WIDTH_16, DATA_WIDTH_16, DATA_WIDTH_16, DATA_WIDTH_16, DATA_WIDTH_16 },  // outboundDataWidth
146         {DATA_TYPE_DL,  DATA_TYPE_DL,  DATA_TYPE_DL,  DATA_TYPE_DL,  DATA_TYPE_DL,  DATA_TYPE_DL },  // inboundDataType
147         {DATA_WIDTH_16, DATA_WIDTH_16, DATA_WIDTH_16, DATA_WIDTH_16, DATA_WIDTH_16, DATA_WIDTH_16 },  // inboundDataWidth
148         {CSL_AIF2_DIO_ENGINE_0, CSL_AIF2_DIO_ENGINE_0, CSL_AIF2_DIO_ENGINE_0, CSL_AIF2_DIO_ENGINE_0, CSL_AIF2_DIO_ENGINE_0, CSL_AIF2_DIO_ENGINE_0}, // DIO engine
149         {1,             0,             0},
150     },
151     {   // CPRI 8x for UL WCDMA - Link 0 - DIO 0
152         "CPRI one link 8x DL 16-bit (4.9152 Gbps)", // test name
153         CSL_AIF2_LINK_PROTOCOL_CPRI,
154         // link0        link1          link2          link3          link4          link5
155         {1,             0,             0,             0,             0,             0           },  // link enable
156         {8,             8,             8,             8,             8,             8           },  // link rate
157         {DATA_TYPE_DL,  DATA_TYPE_DL,  DATA_TYPE_DL,  DATA_TYPE_DL,  DATA_TYPE_DL,  DATA_TYPE_DL },  // outboundDataType
158         {DATA_WIDTH_16, DATA_WIDTH_16, DATA_WIDTH_16, DATA_WIDTH_16, DATA_WIDTH_16, DATA_WIDTH_16 },  // outboundDataWidth
159         {DATA_TYPE_DL,  DATA_TYPE_DL,  DATA_TYPE_DL,  DATA_TYPE_DL,  DATA_TYPE_DL,  DATA_TYPE_DL },  // inboundDataType
160         {DATA_WIDTH_16, DATA_WIDTH_16, DATA_WIDTH_16, DATA_WIDTH_16, DATA_WIDTH_16, DATA_WIDTH_16 },  // inboundDataWidth
161         {CSL_AIF2_DIO_ENGINE_0, CSL_AIF2_DIO_ENGINE_0, CSL_AIF2_DIO_ENGINE_0, CSL_AIF2_DIO_ENGINE_0, CSL_AIF2_DIO_ENGINE_0, CSL_AIF2_DIO_ENGINE_0}, // DIO engine
162         {1,             0,             0},
163     }
164 };
```

Макроопределением «TEST_NUM» задается общее количество проводимых тестов AIF2 (строка 87 листинга 4-1). По умолчанию определено четыре теста AIF2.

Значения массива «testEnable» (строка 89 листинга 4-1) позволяют отключить конкретный тест при запуске приложения теста. Значение «1» означает, что конкретный тест будет запущен, значение «0» отключает запуск конкретного теста. По умолчанию все определенные тесты включены (все значения массива «testEnable» установлены в «1»).

Далее, в строке 97 листинга 4-1 определена структура «TestObj», которая определяет конфигурацию конкретного теста AIF2. Описание полей данной структуры дано в таблице 4-1.

Таблица 4-1: Описание полей структуры «TestObj» конфигурации тестов

Поле	Описание
`name[64]`	Строка, содержащая имя теста.
`protocol`	Протокол, используемый тестом. Может принимать одно из следующих значений: • `CSL_AIF2_LINK_PROTOCOL_OBSAI` — OBSAI протокол; • `CSL_AIF2_LINK_PROTOCOL_CPRI` — CPRI протокол.
`linkEnable[]`	Определяет используется ли конкретный AIF2 линк данным тестом или нет. Может принимать одно из следующих значений: • `0` — тест не использует указанный линк; • `1` — тест использует указанный линк.
`linkRate[]`	Скорость линка. Может принимать одно из следующих значений: • `1` — скорость x1; • `2` — скорость x2; • `4` — скорость x4; • `8` — скорость x8.
`outboundDataType[]`	Тип исходящих данных линка. Может принимать одно из следующих значений: • `DATA_TYPE_DL` — «downlink» тип данных; • `DATA_TYPE_UL` — «uplink» тип данных.
`outboundDataWidth[]`	Формат исходящих данных линка. Может принимать одно из следующих значений: • `DATA_WIDTH_7` — 7 бит; • `DATA_WIDTH_8` — 8 бит; • `DATA_WIDTH_15` — 15 бит; • `DATA_WIDTH_16` — 16 бит.
`inboundDataType[]`	Тип входящих данных линка. Может принимать одно из следующих значений: • `DATA_TYPE_DL` — «downlink» тип данных; • `DATA_TYPE_UL` — «uplink» тип данных.
`inboundDataWidth[]`	Формат входящих данных линка. Может принимать одно из следующих значений: • `DATA_WIDTH_7` — 7 бит; • `DATA_WIDTH_8` — 8 бит; • `DATA_WIDTH_15` — 15 бит; • `DATA_WIDTH_16` — 16 бит.
`dioEngine[]`	Определяет DIO движок, который будет использован для передачи данных данного линка. Может принимать одно из значений: • `CSL_AIF2_DIO_ENGINE_0` — DIO движок 0; • `CSL_AIF2_DIO_ENGINE_1` — DIO движок 1; • `CSL_AIF2_DIO_ENGINE_2` — DIO движок 2.
`nbLinkDio[]`	Массив из трех элементов, каждый элемент которого соответствует определенному DIO движку. Определяет количество линков, назначенных на конкретный DIO движок. Например, если AIF2 Link 0 использует DIO движок 0, а AIF2 Link 1 использует DIO движок 1, при этом остальные линки отключены, то массив `nbLinkDio[]` должен быть задан в виде `{ 1, 1, 0 }`.

Примечание: *Поля `linkEnable[]`, `linkRate[]`, `outboundDataType[]`, `outboundDataWidth[]`, `inboundDataType[]`, `inboundDataWidth[]` и `dioEngine[]` представляют собой массивы из шести элементов, каждый элемент которого соответствует определенному AIF2 линку. Например, `linkEnable[0]` соответствует AIF2 Link 0, `linkEnabled[1]` соответствует AIF2 Link 1 и т. д.*

Начиная со строки 111 листинга 4-1 описывается массив структур «TestObj» для всех четырех тестов AIF2, определенных по умолчанию:

1) OBSAI тест объединенных четырех линков AIF2 Link 0, 1, 2 и 3 на скорости 4x (формат данных 8 бит);
2) OBSAI тест линка AIF2 Link 0 на скорости 8x (формат данных 8 бит);
3) OBSAI тест линка AIF2 Link 0 на скорости 8x (формат данных 16 бит);
4) CPRI тест линка AIF2 Link 0 на скорости 8x (формат данных 16 бит).

Внимание

При задании пользовательской конфигурации тестов AIF2 необходимо помнить о том, что в SVP-465 используются только AIF2 Link 0, 1, 2 и 3. Линки AIF2 Link 4 и 5 в модуле SVP-465 не подключены.

5 Сборка проекта

Для сборки проекта нажмите правой кнопой мыши по названию проекта «AIF2_GenericPacketTest» в окне обозревателя проектов («Project Explorer») для вызова контекстного меню и выберите пункт меню «Build Configurations > Build Selected...» (рисунок 5-1).

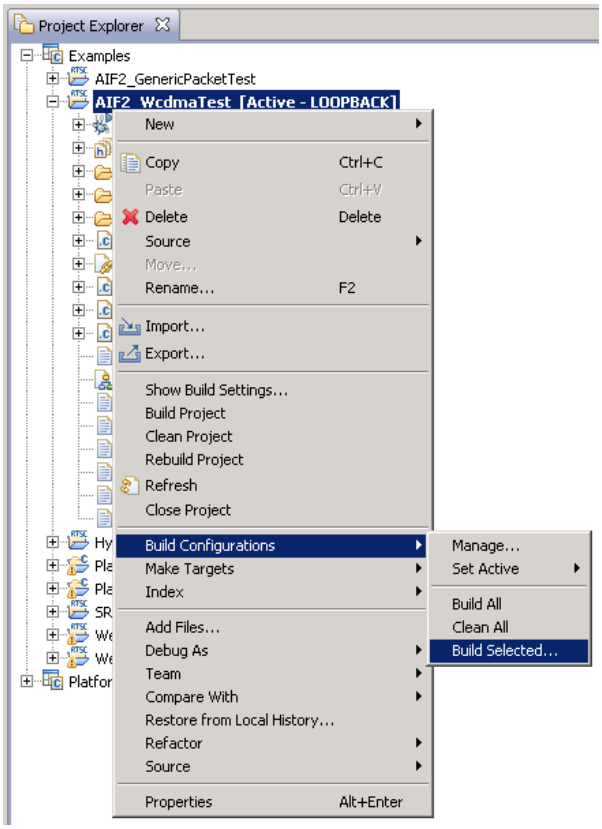


Рисунок 5-1: Пункт меню для сборки проекта

В открывшемся окне (рисунок 5-2) нужно отметить конфигурации сборки, для которых необходимо выполнить сборку и нажать на кнопку «OK».

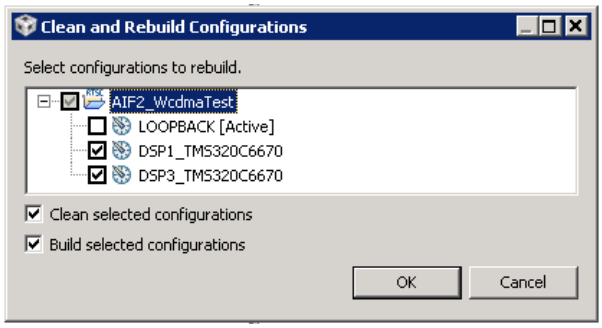


Рисунок 5-2: Окно выбора конфигураций для сборки

Проект «AIF2_GenericPacketTest» имеет три конфигурации сборки:

- конфигурация «LOOPBACK» — проект с включенной локальной петлей на AIF2;
- конфигурация «DSP1_TMS320C6670» — проект для запуска на первом процессоре TMS320C6670;
- конфигурация «DSP3_TMS320C6670» — проект для запуска на втором процессоре TMS320C6670.

Если тест передачи данных через AIF2 планируется запускать на одном процессоре с включенной локальной петлей на AIF2 необходимо выбрать одну конфигурацию сборки «LOOPBACK». Если тест будет запускаться на двух процессорах (передача данных между двумя процессорами), то необходимо отметить для сборки две конфиуграции — «DSP1_TMS320C6670» и «DSP3_TMS320C6670».

После нажатия на кнопку «OK» будет запущен процесс сборки выбранных конфигураций сборки. В случае успешной сборки, в окне консоли (Console) должны быть выведены сообщения, идентичные показанным на рисунке 5-3.

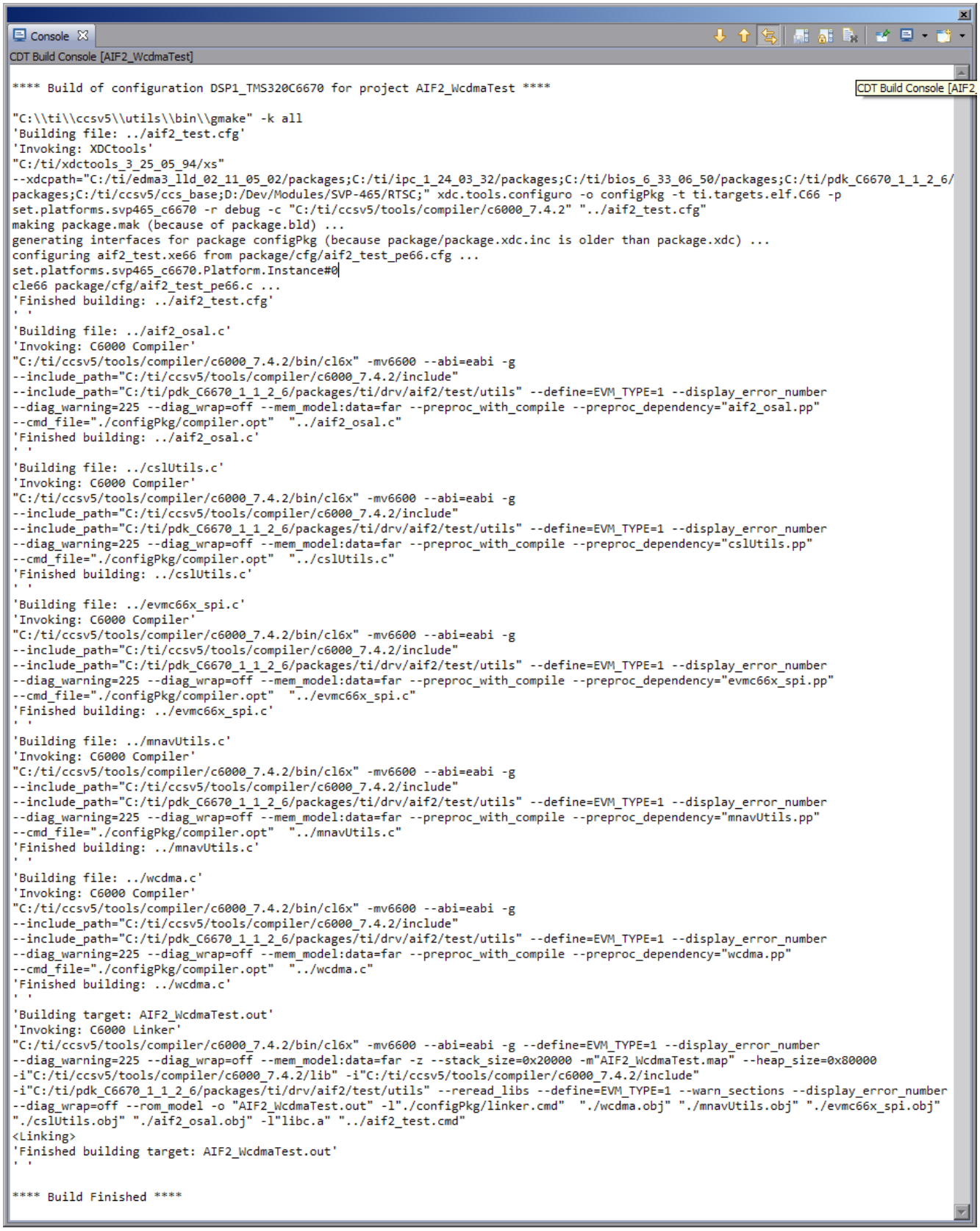


Рисунок 5-3: Сборка проекта теста производительности

6 Импорт и запуск целевой конфигурации модуля

Для загрузки кода приложениий 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 6-1)

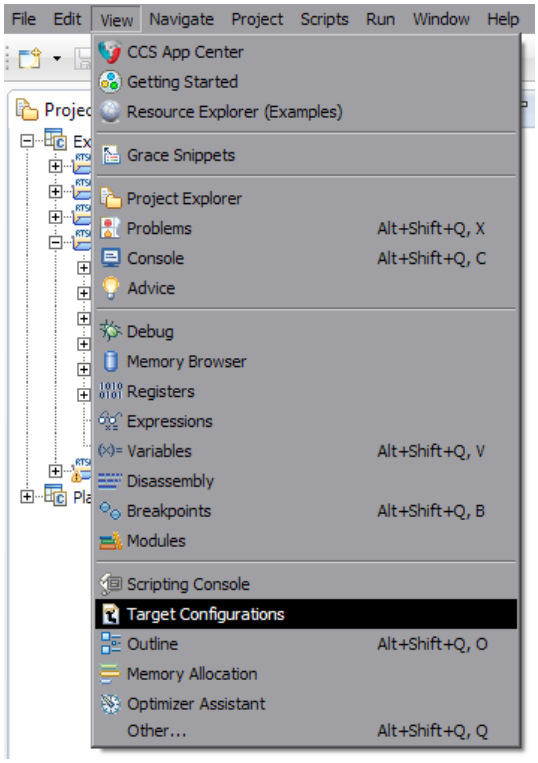


Рисунок 6-1: Пункт меню для отображения окна целевых конфигураций

В окне целевых конфигураций («Target Configurations»), нажмите правой кнопкой мыши на свободной области для вызова контекстного меню и выберите пункт «Import Target Configration» (см. рисунок 6-2).

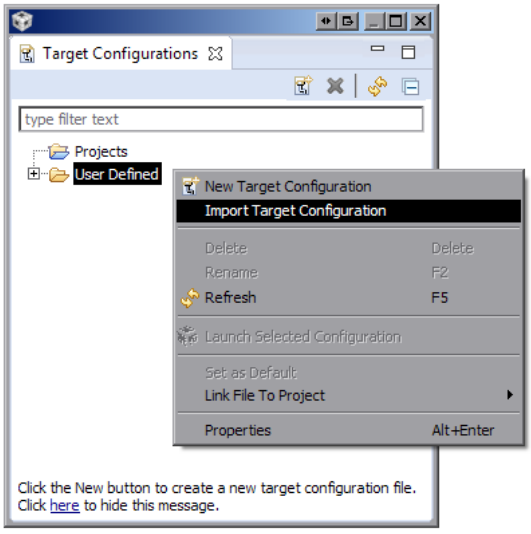


Рисунок 6-2: Меню импорта целевой конфигурации

В появившемся окне выбора файла (см. рисунок 6-3) 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».

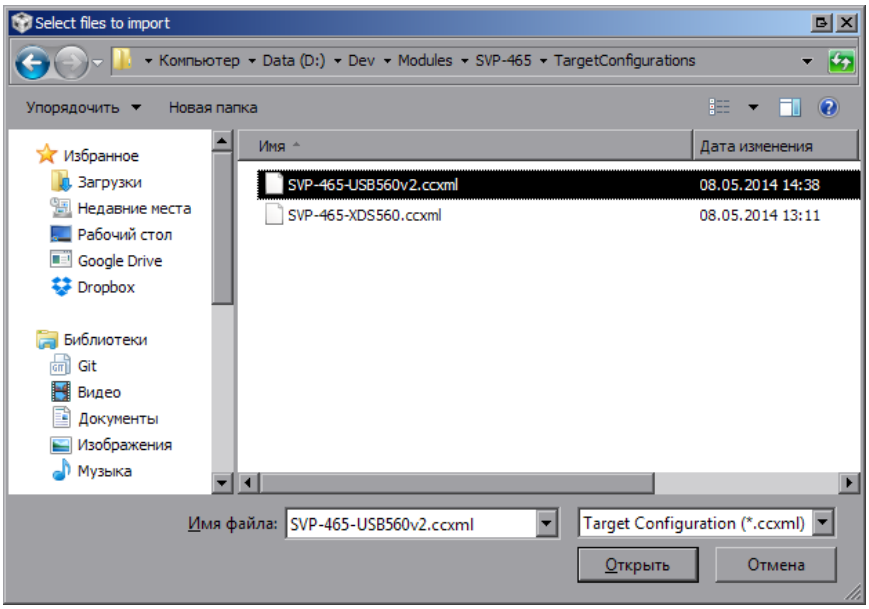


Рисунок 6-3: Окно выбора файла для импорта целевой конфигурации

После нажатия на кнопку «Открыть» появится окно выбора способа импорта файла целевой конфигурации (рисунок 6-4). Необходимо выбрать способ «Link to files» и нажать на кнопку «OK».

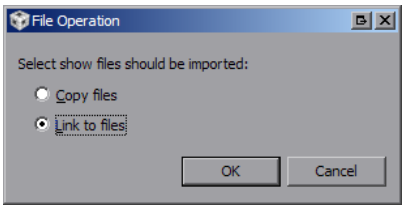


Рисунок 6-4: Окно выбора способа импорта файла целевой конфигурации

Для запуска целевой конфигурации, в окне целевых конфигураций («Target Configurations»), необходимо нажать правой кнопкой мыши на целевой конфиуграции и выбрать пункт меню «Launch Selected Configuration» (см. рисунок 6-5).

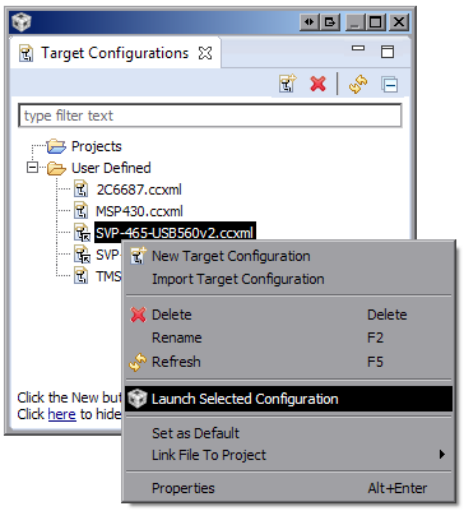


Рисунок 6-5: Запуск целевой конфигурации

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 6-6.

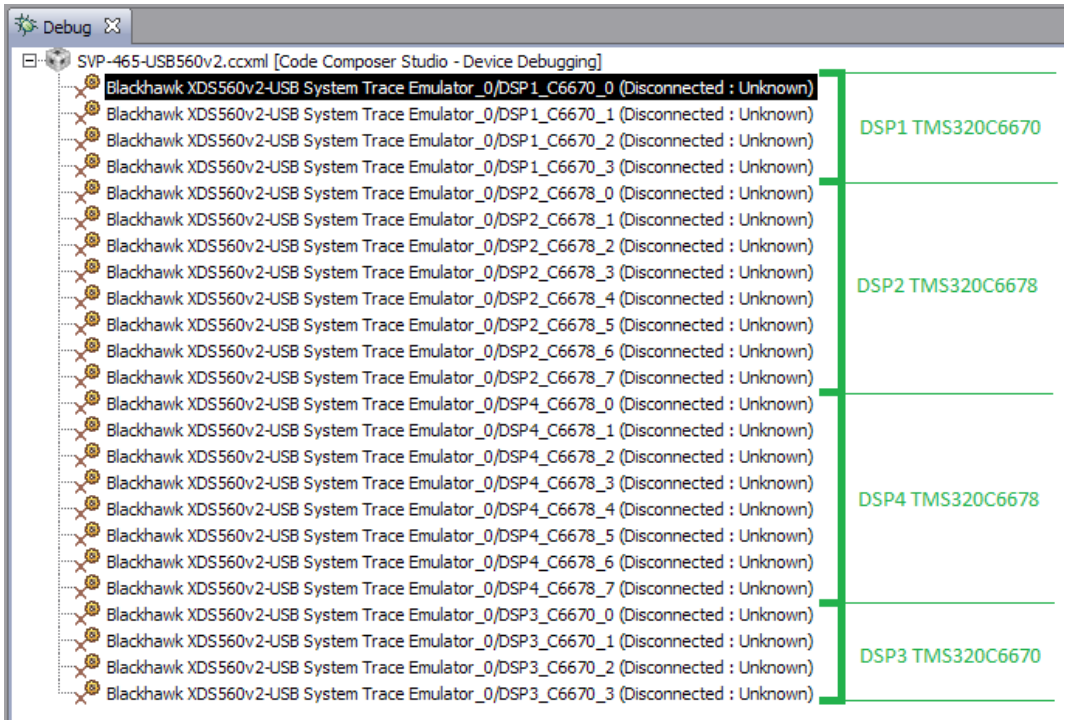


Рисунок 6-6: Список ядер процессоров модуля SVP-465

Примечание

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

7 Загрузка кода

Запустите целевую конфигурацию модуля SVP-465, как описано в разделе 6.

Перед загрузкой кода на ядра процессора, необходимо выполнить группировку тех ядер, на которых будет выполняться код теста AIF2. В данном случае, это ядра «DSP1_C6670_0» и «DSP3_C6670_0».

Выберите ядра «DSP1_C6670_0» и «DSP3_C6670_0» в окне «Debug» (щелкните левой кнопкой мыши на ядре «DSP1_C6670_0», затем, зажав на клавиатуре клавишу Ctrl, щелкните на ядре «DSP3_C6670_0»). Щелкните правой кнопкой мыши на ядре «DSP1_C6670_0» и выберите пункт меню «Group core(s)» (см. рисунок 7-1).

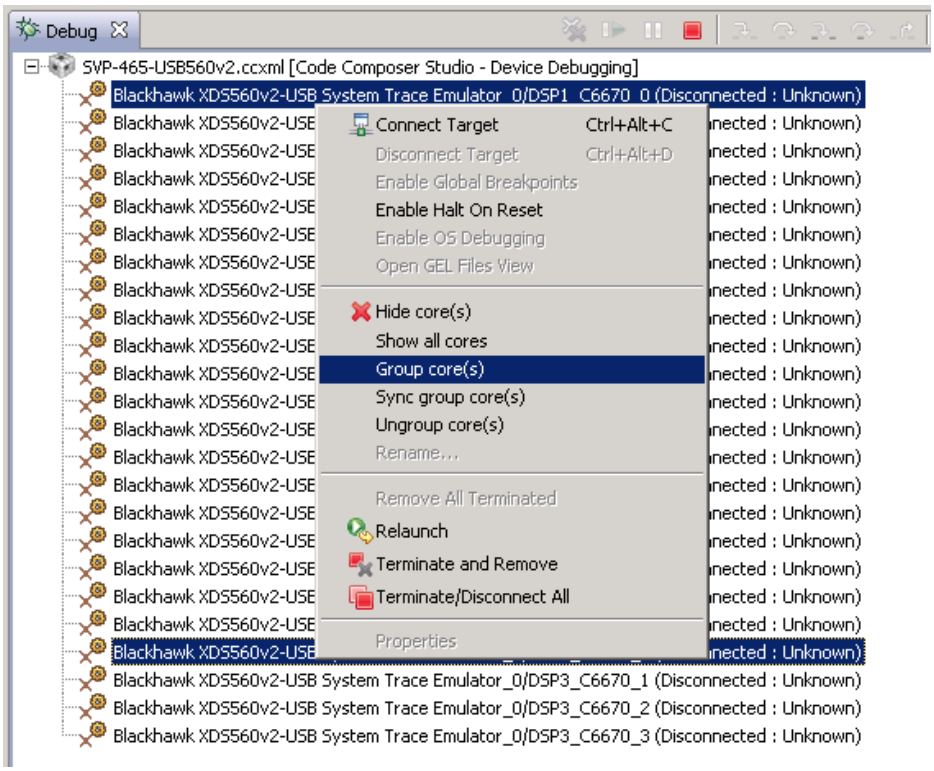


Рисунок 7-1: Группировка ядер процессоров

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 7-2.

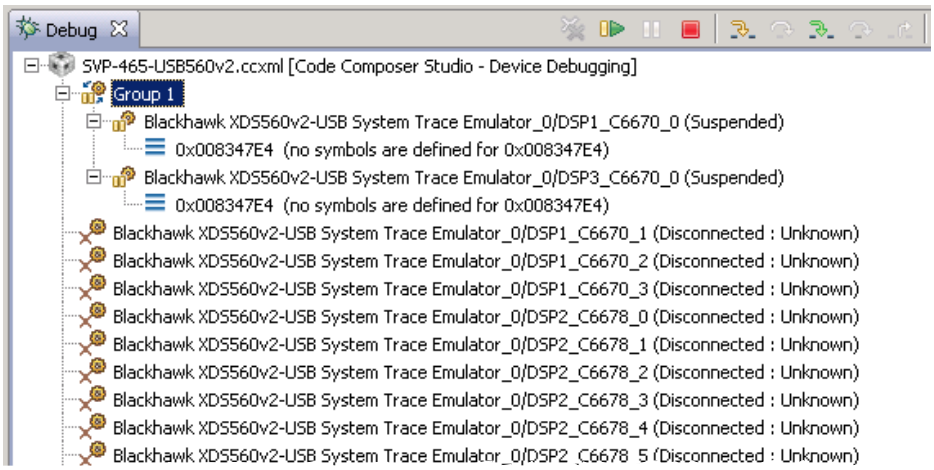


Рисунок 7-2: Ядра процессоров после выполнения подключения

Выберите ядро «DSP1_C6670_0» в окне «Debug» (щелкните левой кнопкой мыши по названию ядра). Выберите пункт главного меню «Run > Load > Load Program...» (рисунок 7-3).

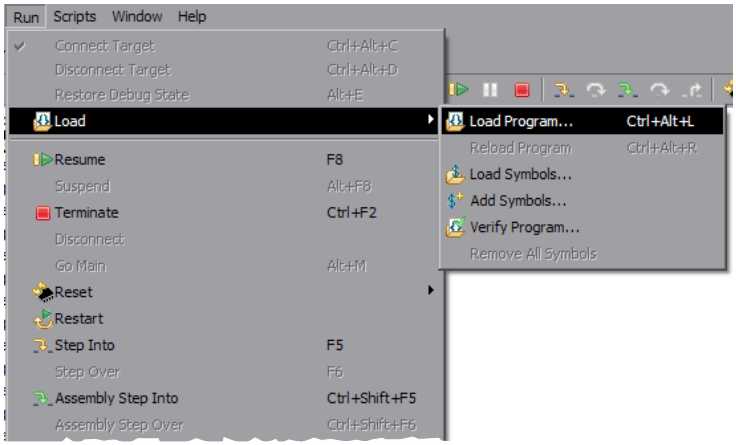


Рисунок 7-3: Меню загрузки кода на ядро процессора

В открывшемся окне (рисунок 7-4) нажмите на кнопку «Browse project...».

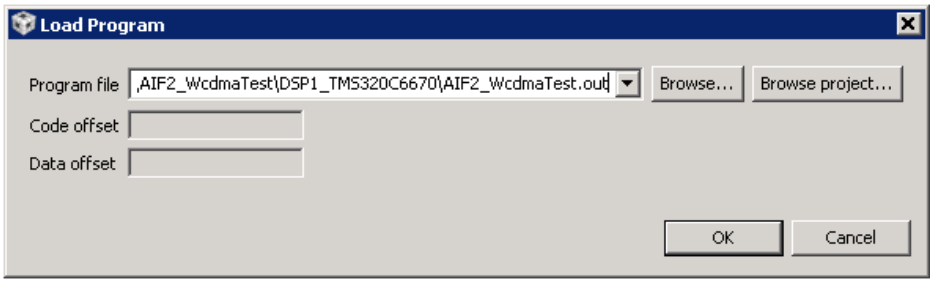
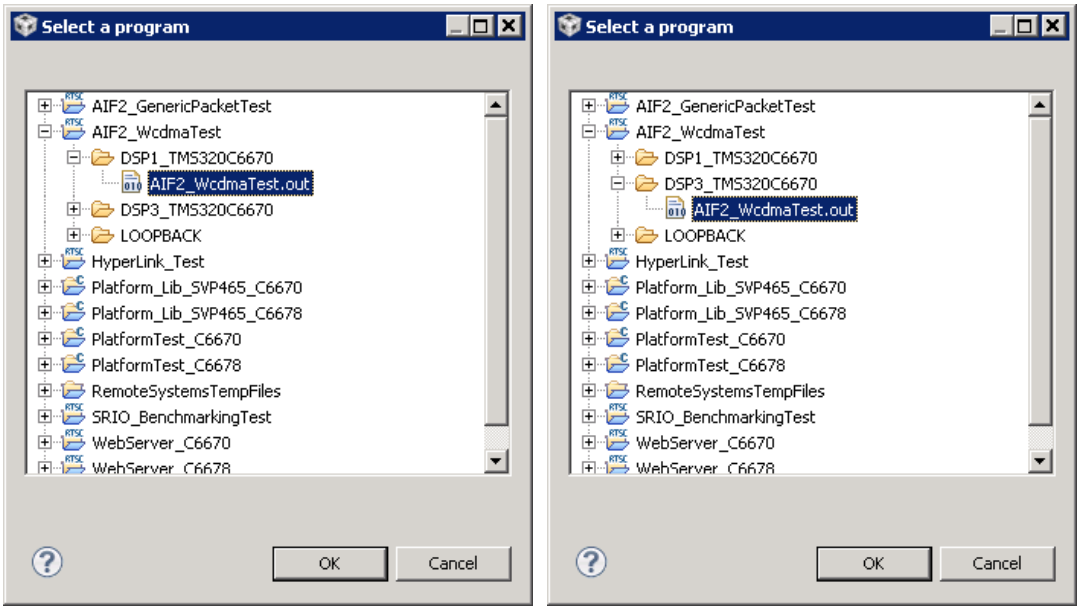


Рисунок 7-4: Окно загрузки кода на ядро процессора

Выберите файл «AIF2_WcdmaTest.out» конфигурации сборки «DSP1_TMS320C6670» из проекта «AIF2_WcdmaTest», как показано на рисунке 7-5а, и нажмите на кнопку «OK». В окне, показанном на рисунке 7-4, также нажмите на кнопку «OK».



а) Первый процессор TMS320C6670 б) Второй процессор TMS320C6670

Рисунок 7-5: Окно выбора файла для загрузки на ядро процессора

После загрузки кода на ядро «DSP1_C6670_0» необходимо, также, выполнить загрузку кода на ядро «DSP3_C6670_0» второго процессора TMS320C6670. Для этого, выберите ядро «DSP3_C6670_0» в окне «Debug» и аналогичным образом выполните загрузку файла «AIF2_WcdmaTest.out», но из конфигурации сборки «DSP3_TMS320C6670» (см. рисунок 7-5б).

После загрузки кода теста AIF2 на оба процессора TMS320C6670, окно «Debug» должно выглядеть, как показано на рисунке 7-6.

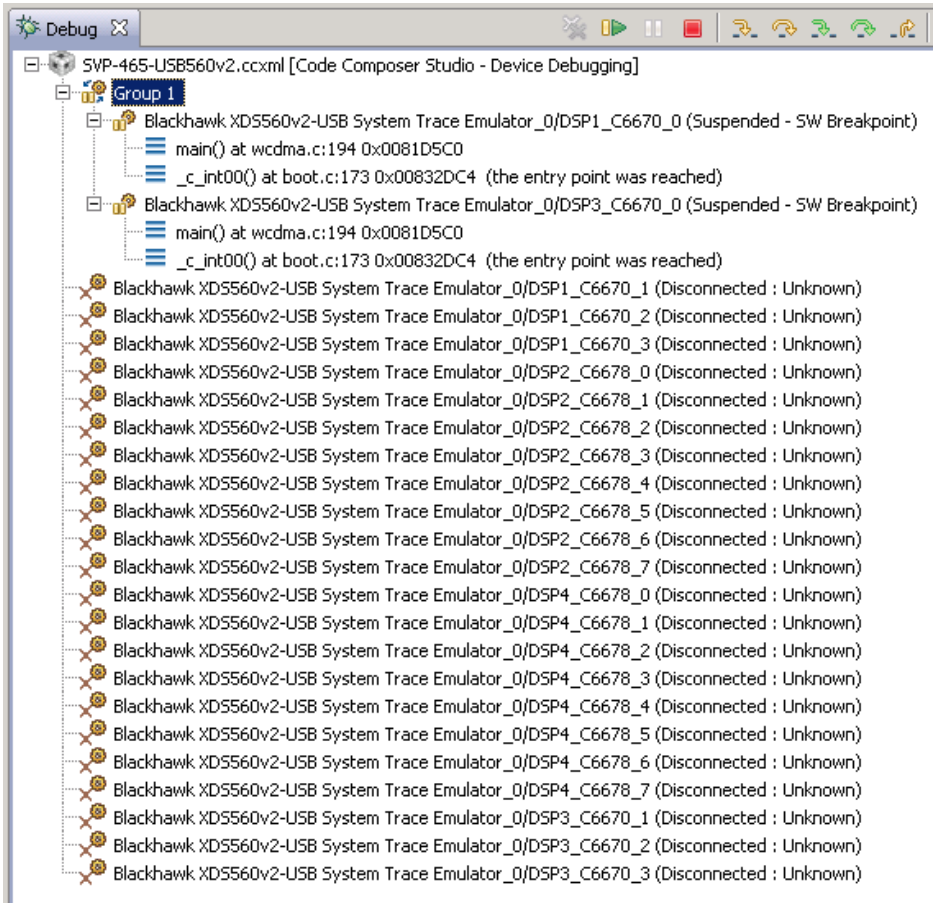


Рисунок 7-6: Внешний вид окна «Debug» после загрузки кода на ядра процессоров

8 Запуск теста передачи данных AIF2

Для запуска теста AIF2 выполните шаги, описанные в процедуре 8-1.

Процедура 8-1. Запуск теста AIF2

1. Выполнить загрузку кода теста AIF2 на ядра процессоров, как описано в разделе 7.
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 7-6 кнопка «Resume» расположена в верхнем правом углу.

В случае успешного запуска теста AIF2, вывод с ядра «DSP1_C6670_0» в окно «Console» должен выглядеть аналогично приведенному в листинге 8-1, а вывод с ядра «DSP3_C6670_0» должен выглядеть аналогично приведенному в листинге 8-2.

В листингах 8-1 и 8-2 приведен вывод для тестирования AIF2 Link 0. Для тестирования AIF2 Link 1, AIF2 Link 2 и AIF2 Link 3 необходимо изменить конфигурацию проекта в соответствии с разделом 4, выполнить сборку проекта согласно разделу 5 и перезапустить его. Вывод с ядер должен быть аналогичным тому, что приведены в листингах 8-1 и 8-2.

Примечание

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

Листинг 8-1: Вывод в консоль теста AIF2 с ядра «DSP1_C6670_0»

```
Beginning AIF2 WCDMA test:
test: OBSAI multi-link 4x UL 8-bit (4 x 3.072 Gbps)
AIF OBSAI mode
link 0 runs at 4x rate
        outbound data type: UL_RSA
        outbound data width: 8 bit
        inbound data type: UL_RSA
        inbound data width: 8 bit
link 1 runs at 4x rate
        outbound data type: UL_RSA
        outbound data width: 8 bit
        inbound data type: UL_RSA
        inbound data width: 8 bit
link 2 runs at 4x rate
        outbound data type: UL_RSA
        outbound data width: 8 bit
        inbound data type: UL_RSA
        inbound data width: 8 bit
link 3 runs at 4x rate
        outbound data type: UL_RSA
        outbound data width: 8 bit
        inbound data type: UL_RSA
        inbound data width: 8 bit
link 4 is disabled
link 5 is disabled

 Test OBSAI multi-link 4x UL 8-bit (3.072 Gbps x 4) on dsp 1: SUCCESS

test: OBSAI one link 8x UL 8-bit (6.250 Gbps)
AIF OBSAI mode
link 0 runs at 8x rate
        outbound data type: UL_RSA
        outbound data width: 8 bit
        inbound data type: UL_RSA
        inbound data width: 8 bit
link 1 is disabled
```

```
link 2 is disabled
link 3 is disabled
link 4 is disabled
link 5 is disabled

 Test OBSAI one link 8x UL 8-bit (6.250 Gbps) on dsp 1: SUCCESS

test: OBSAI one link 8x DL 16-bit (6.250 Gbps)
AIF OBSAI mode
link 0 runs at 8x rate
        outbound data type: DL
        outbound data width: 16 bit
        inbound data type: DL
        inbound data width: 16 bit
link 1 is disabled
link 2 is disabled
link 3 is disabled
link 4 is disabled
link 5 is disabled

 Test OBSAI one link 8x DL 16-bit (6.250 Gbps) on dsp 1: SUCCESS

test: CPRI one link 8x DL 16-bit (4.9152 Gbps)
AIF CPRI mode
link 0 runs at 8x rate
        outbound data type: DL
        outbound data width: 16 bit
        inbound data type: DL
        inbound data width: 16 bit
link 1 is disabled
link 2 is disabled
link 3 is disabled
link 4 is disabled
link 5 is disabled

 Test CPRI one link 8x DL 16-bit (4.9152 Gbps) on dsp 1: SUCCESS

All tests have passed
```

Листинг 8-2: Вывод в консоль теста AIF2 с ядра «DSP3_C6670_0»

```
Beginning AIF2 WCDMA test:
test: OBSAI multi-link 4x UL 8-bit (4 x 3.072 Gbps)
test: OBSAI one link 8x UL 8-bit (6.250 Gbps)
test: OBSAI one link 8x DL 16-bit (6.250 Gbps)
test: CPRI one link 8x DL 16-bit (4.9152 Gbps)
All tests have passed
```

Приложение A 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 раздельного 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).

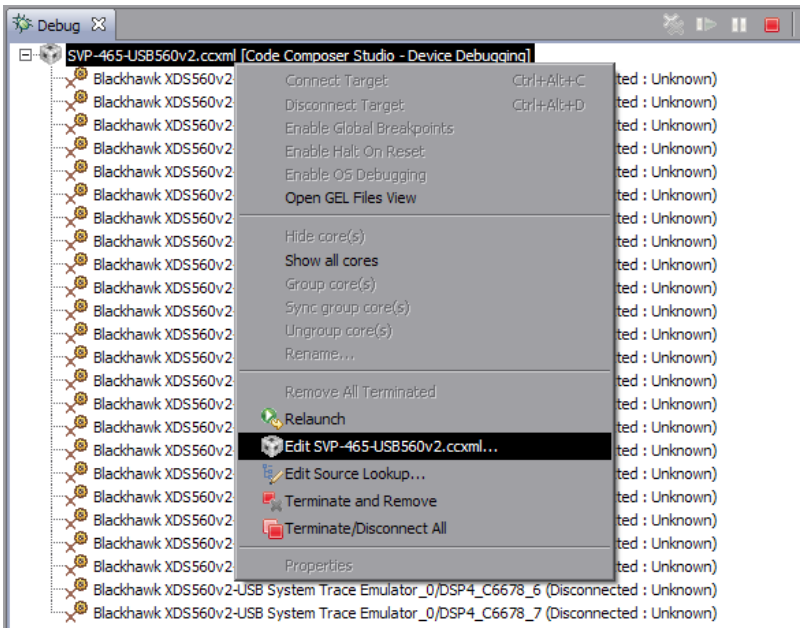


Рисунок A-1: Контекстное меню целевой конфигурации

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».

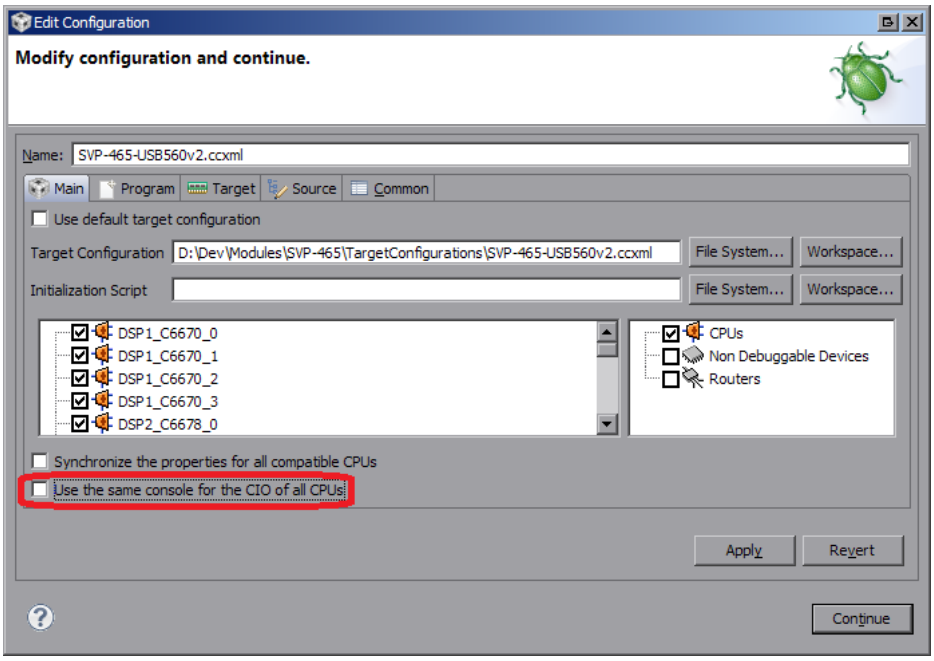


Рисунок A-2: Окно настроек целевой конфигурации

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

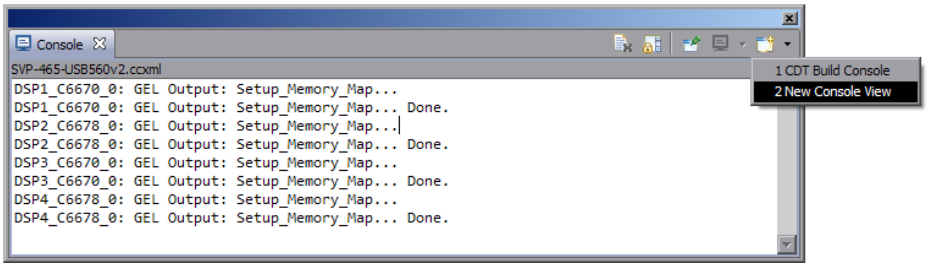


Рисунок A-3: Открытие второго окна «Console»

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

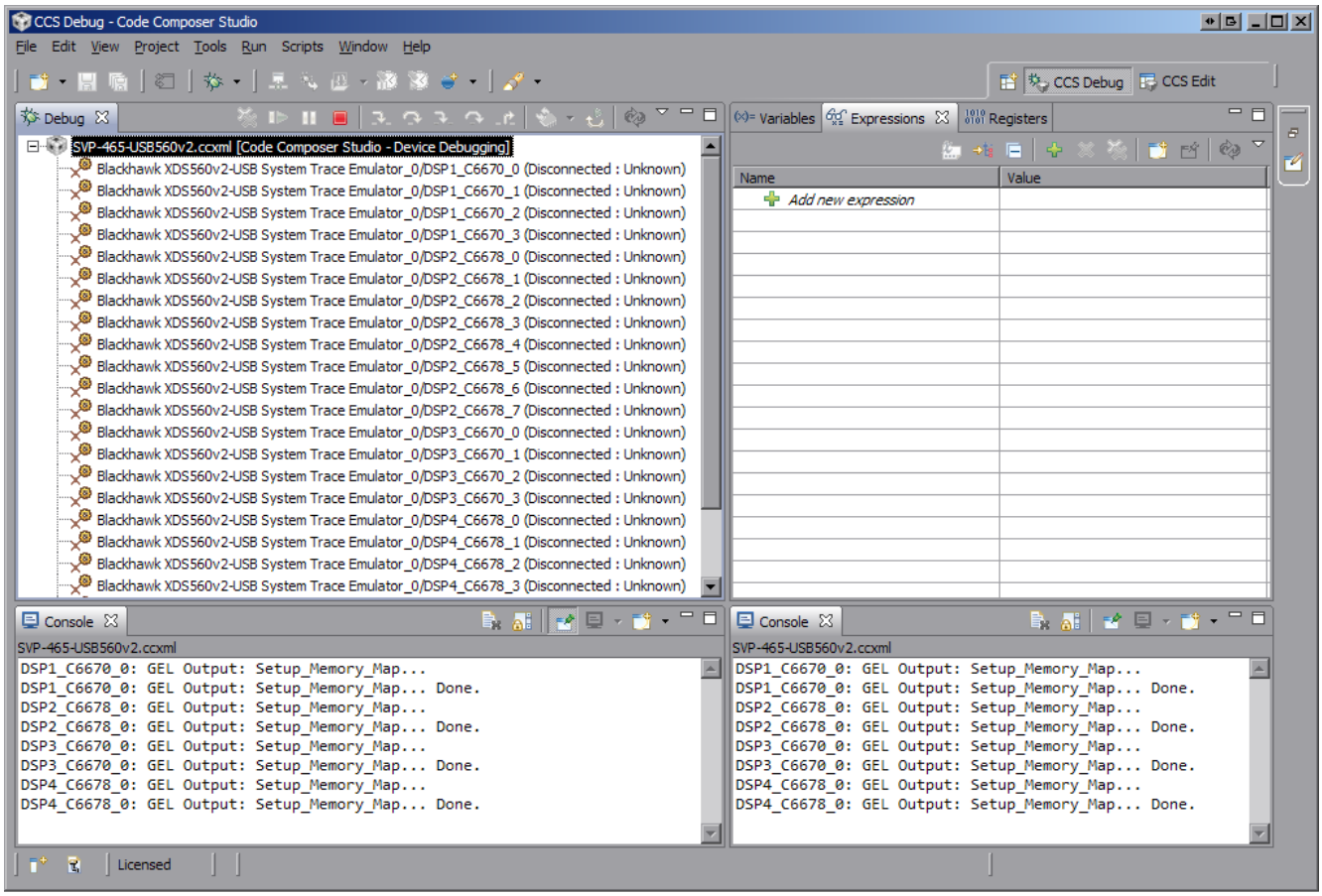


Рисунок A-4: Два окна «Console»

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).

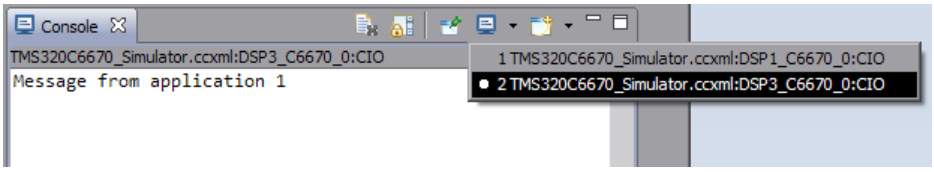


Рисунок A-5: Выбор ядра для отображения вывода в окне «Console»

Следует иметь ввиду, 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.

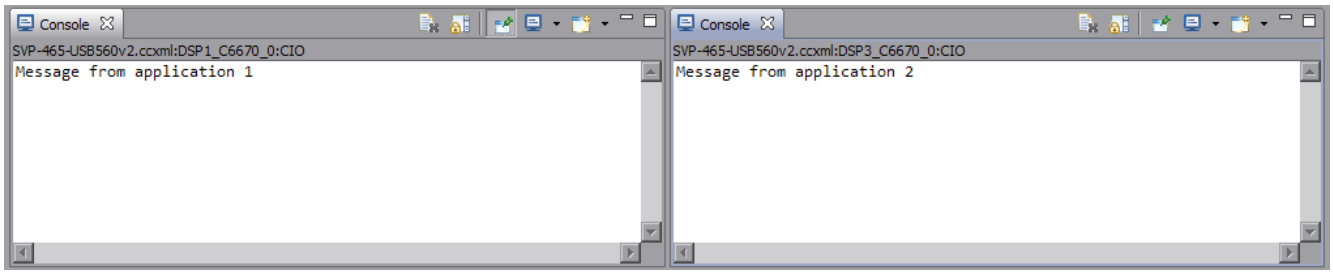


Рисунок A-6: Вывод сообщений с двух ядер в два окна «Console»

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