Scan Engineering Telecom SPb

SVR-420. Загрузчик IBL

Руководство пользователя

Версия 1.1

Код документа: UG-SVR-420-IBL
Дата сборки: 14 марта 2016 г.
Листов в документе: 47

http://www.setdsp.ru

История ревизий

Ревизия	Дата	Изменения
1.1	14.03.2016	Исправлена ошибка в листинге A-2. Исправлена ошибка в приложении B.
1.0	—	Начальная версия

Содержание

Перечень рисунков

Перечень таблиц

Перечень листингов

Перечень процедур

Перечень сокращений и условных обозначений

BOOTP	Bootstrap Protocol	31, 39
CCS	Code Composer Studio	7, 8, 11–15, 18, 33, 34, 43, 44
CGT	Code Generation Tools	37
CIO	Console Input/Output	3, 15, 17, 34, 43
COFF	Common Object File Format	31
DDR	Double Data Rate	14, 18, 35, 36
DSO	Dynamic Shared Object	35
DSS	Debug Server Scripting	12, 13
EEPROM	Electrically Erasable Programmable Read-Only Memory	3–5, 7, 8, 11–16, 18, 19, 23–26, 28, 36, 40, 42
ELF	Executable and Linkable Format	31, 35
GEL	General Extension Language	4, 11, 26–28
I^2C	Inter-Integrated Circuit	8, 16, 19, 23, 36
IBL	Intermediate Boot Loader	3–5, 7, 8, 10, 11, 14, 18, 19, 23–26, 28, 31, 35, 36, 39, 46
IP	Internet Protocol	31
JSON	Java Script Object Notation	37
MAC	Media Access Control	31
MAD	Multicore Application Deployment	3, 4, 35–37, 39
MAP	Multiple Application Pre-linker	3, 35–39
MCSDK	MultiCore Software Development Kit	35, 37, 39
NML	No Man’s Land	37
NOR	Not OR	3–5, 7, 13–16, 18, 19, 24, 36, 39, 40
OFD	Object File Dump	37
ROMFS	Read-Only Memory File System	35–38
ROM	Read-Only Memory	36
TFTP	Trivial File Transfer Protocol	7, 14, 36, 39
TI	Texas Instruments	8
UART	Universal Asynchronous Receiver-Transmitter	4, 14, 18–20
ОС	Операционная Система	7

1 Общие сведения

Загрузчик IBL (Intermediate Boot Loader) позволяет выполнять загрузку приложений на процессоры модуля SVR-420 с NOR флеш памяти или по Ethernet с TFTP сервера в локальной сети. Конкретный режим загрузки выбирается при помощи переключателей на плате модуля SVR-420 отдельно для каждого процессора (см. приложение B).

В данном документе дано описание основных возможностей загрузчика IBL, описан процесс сборки образов IBL из исходных кодов для записи в EEPROM память процессоров модуля SVR-420 (раздел 2), описана процедура записи загрузчика IBL в EEPROM модуля SVR-420 (раздел 3.4), описан процесс конфигурирования уже записанного в EEPROM память загрузчика.

Также, в документе описан процесс записи образов приложений в NOR флеш память модуля SVR-420 для последующий загрузки этих образов загрузчиком IBL.

В данном документе описана работа со средой разработки CCS (Code Composer Studio) версии 5.4.0.00091. Установочный файл среды разработки CCS можно скачать в сети интернет с официального сайта[1], где доступны версии CCS для Windows и Linux систем. На сопроводительном диске к модулю SVR-420 в папке «Install» имеется установочный файл для среды разработки CCS версии 5.4.0.00091 для Windows системы (файл «ccs_setup_5.4.0.00091.exe»).

В данном руководстве предполагается, что среда разработки CCS установлена в папку «C:\ti» и используется компьютер с установленной 64-х разрядной версией ОС Windows 7.

Внимание

Для установки некоторых программ при помощи установочных дистрибутивов, содержащихся на сопроводительном диске к модулю SVR-420, может потребоваться подключение к сети интернет.

[1] http://processors.wiki.ti.com/index.php/Download_CCS

2 Сборка образа IBL из исходных кодов

Для сборки IBL потребуется установленный компилятор для процессоров Texas Instruments серии C6000. Данный компилятор входит в состав среды разработки CCS компании TI.

Результатом сборки IBL из исходных кодов является образ загрузчика готовый кзаписи в I^2C EEPROM модуля SVR-420.

Внимание

Перед выполнением сборки загрузчика, описанной в данном разделе, перепишите с сопроводительного диска к модулю SVR-420 папку «ibl» на жесткий диск компьютера. Далее, предполагается, что все содержимое папки «ibl» с сопроводительного диска переписано в папку «D:/Dev/Modules/SVR-420/IBL».

Для сборки загрузчика IBL в Windows системе, кроме компилятора C6000 процессоров, необходима GNU система сборки MinGW. Скачать последнюю версию MinGW можно на официальном сайте[1].

В данном документе описана работа с MinGW версии 20120426. Установочный дистрибутив MinGW версии 20120426 можно найти на сопроводительном диске к модулю SVR-420 в папке «Install» (файл «mingw-get-inst-20120426.exe»).

2.1 Установка MinGW в Windows системе

В данном разделе описан процесс установки MinGW версии 20120426 с установочного дистрибутива, содержащегося на сопроводительном диске к модулю SVR-420 (файл «Install/mingw-get-inst-20120426.exe»).

При установке MinGW в окне выбора каталога репозиториев (рисунок 2-1) необходимо выбрать пункт «Use pre-packaged repository catalogues» (использовать каталог репозитариев с заранее собранными пакетами).

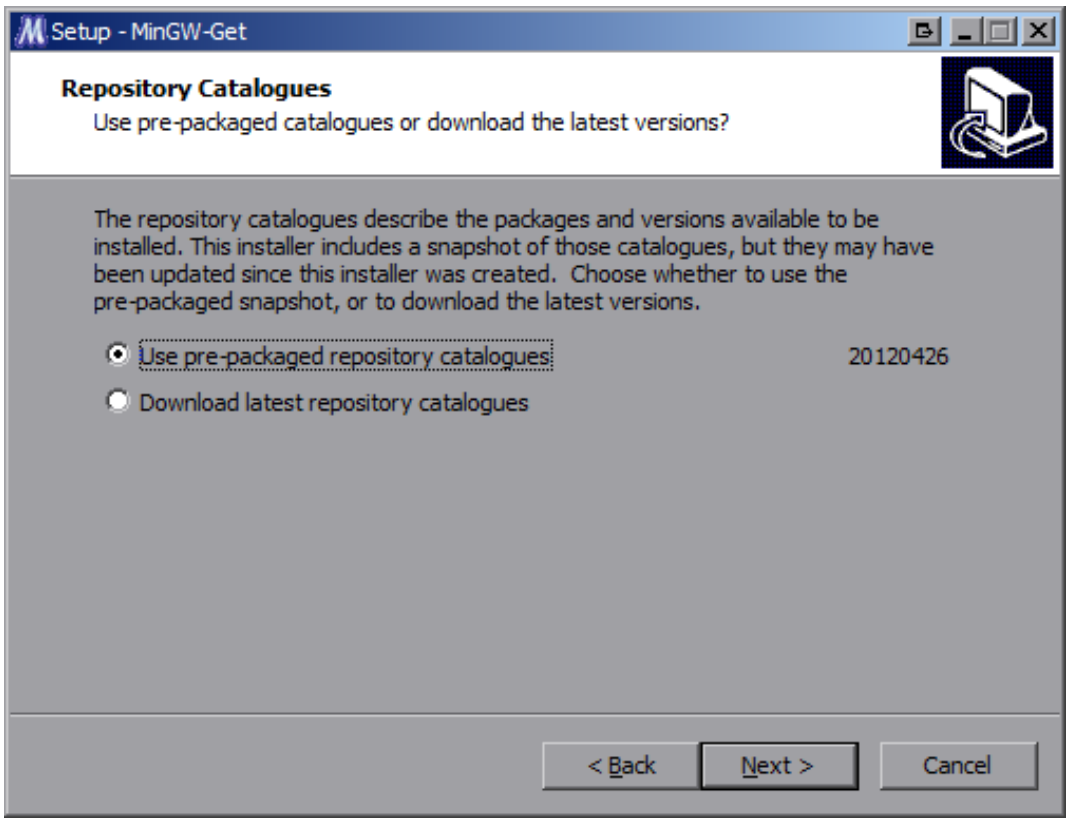


Рисунок 2-1: Установка MinGW. Выбор каталога репозитариев

В окне согласия с лицензией (рисунок 2-2), прочитайте текст лицензии, и если вы согласны со всем, что там написано, выберите пункт «I accept the agreement» и нажмите кнопку «Next».

[1] http://www.mingw.org

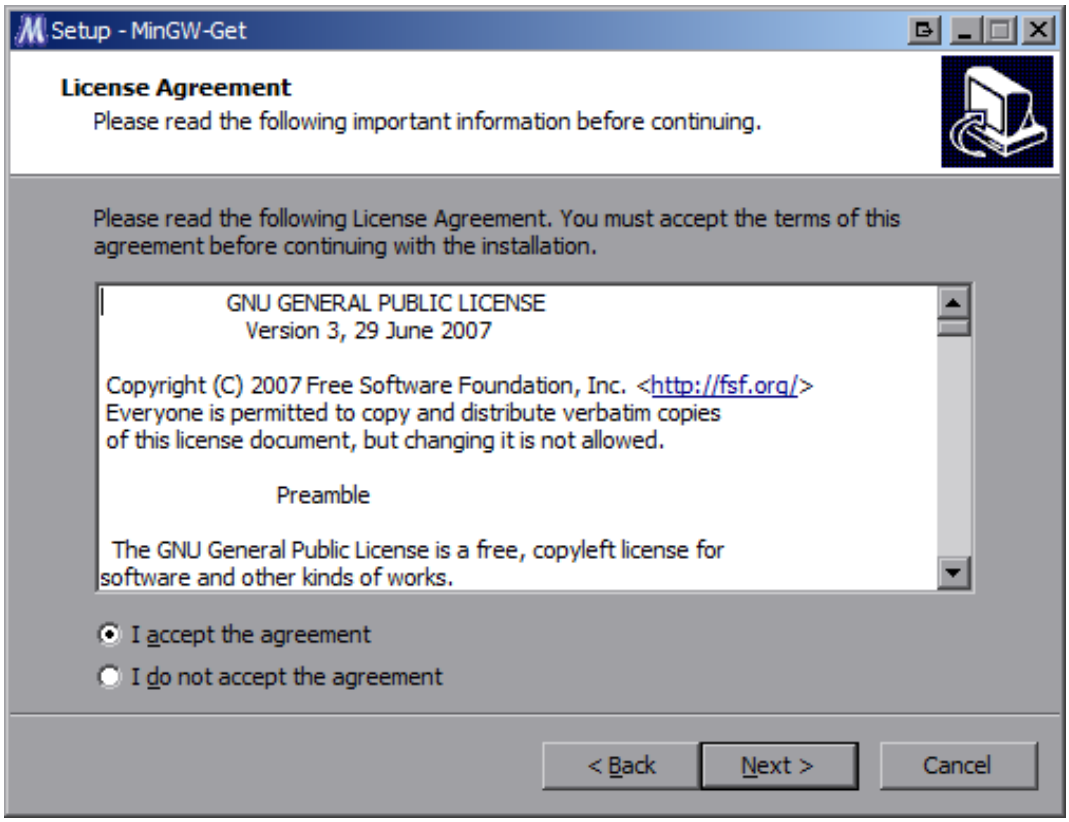


Рисунок 2-2: Установка MinGW. Окно согласия с лицензией

Путь установки MinGW (рисунок 2-3) рекомендуется оставить по умолчанию («C:/MinGW»).

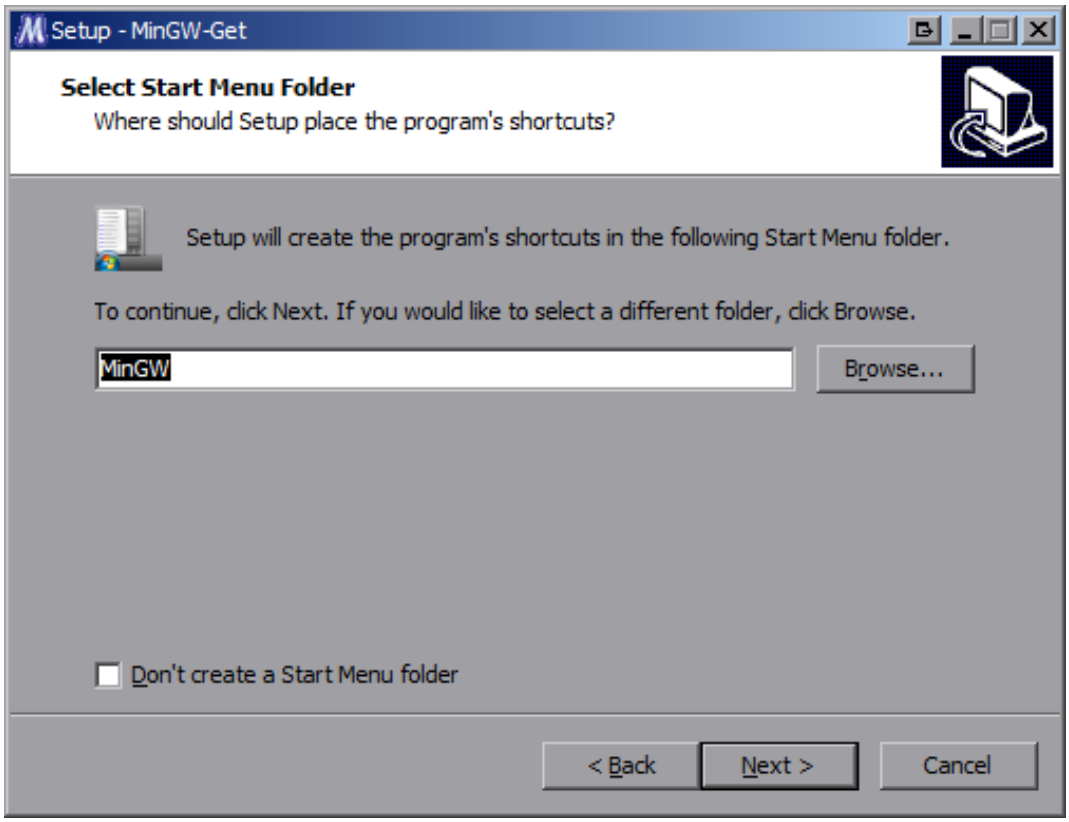


Рисунок 2-3: Установка MinGW. Выбор пути установки

В окне выбора устанавливаемых компонентов (рисунок 2-4) необходимо обязательно отметить следующие компоненты:

- «C Compiler»;
- «MSYS Basic System»;
- «MinGW Developer ToolKit».

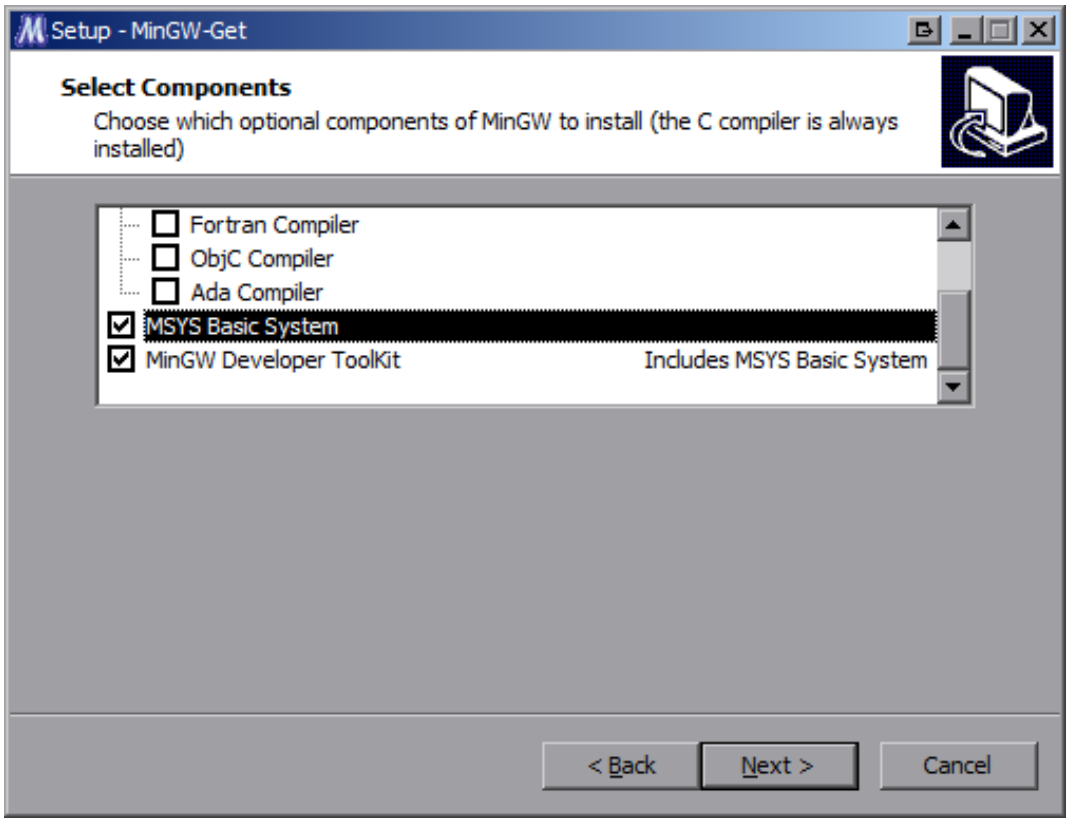


Рисунок 2-4: Установка MinGW. Выбор устанавливаемых компонентов

Остальные компоненты можно отметить на собственное усмотрение. На дальнейший процесс сборки загрузчика IBL их наличие или отсутствие никак не повлияет.

Остальные параметры установки, которые не описаны в данном руководстве, можно оставить в виде, предлагаемом установщиком по умолчанию.

После установки MinGW, через меню «Пуск», запустите «MinGW Shell» (рисунок 2-5). Все последующие действия по сборке IBL будут производиться путем ввода команд в MinGW Shell.

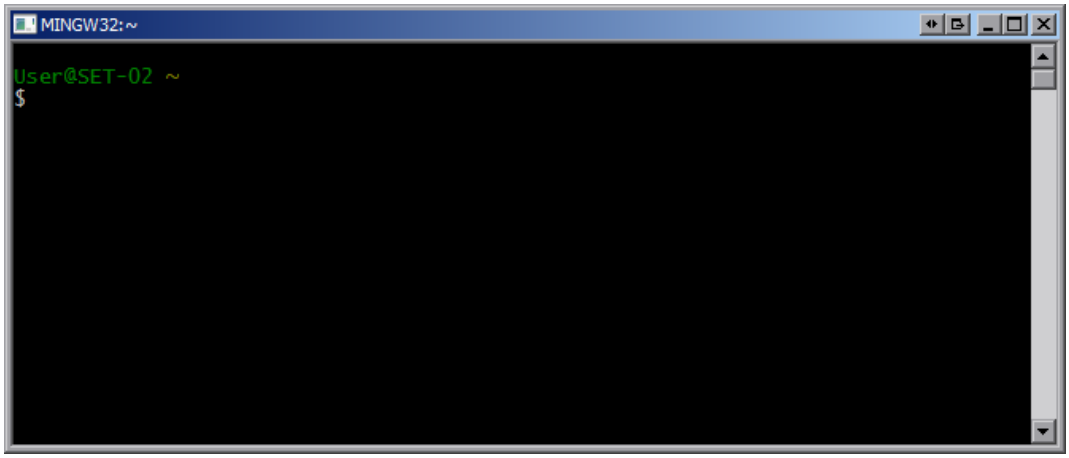


Рисунок 2-5: Приглашение командной строки MinGW Shell

2.2 Конфигурация окружения сборки

В данном разделе предполагается, что исходные коды загрузчика IBL переписаны с сопроводительного диска к модулю SVR-420 в папку «D:/Dev/Modules/SVR-420/IBL», а файлы компилятора для процессоров C6000 серии расположен в папке «C:/ti/ccsv5/tools/compiler/c6000_7.4.2». В том случае, если путь к расположению файлов компилятора для процессоров C6000 отличается, необходимо выполнить соответствующие изменения в скрипте конфигурации окружения сборки «D:/Dev/Modules/SVR-420/IBL/src/make/setupenvMsys.sh». В данном файле необходимо правильно указать путь к файлам компилятора для процессоров C6000 серии.

На рисунке 2-6 приведен снимок экрана MinGW Shell с открытым файлом «setupenvMsys.sh» в редакторе vim. На данном рисунке пути установлены в соответствии с путями указанными выше.

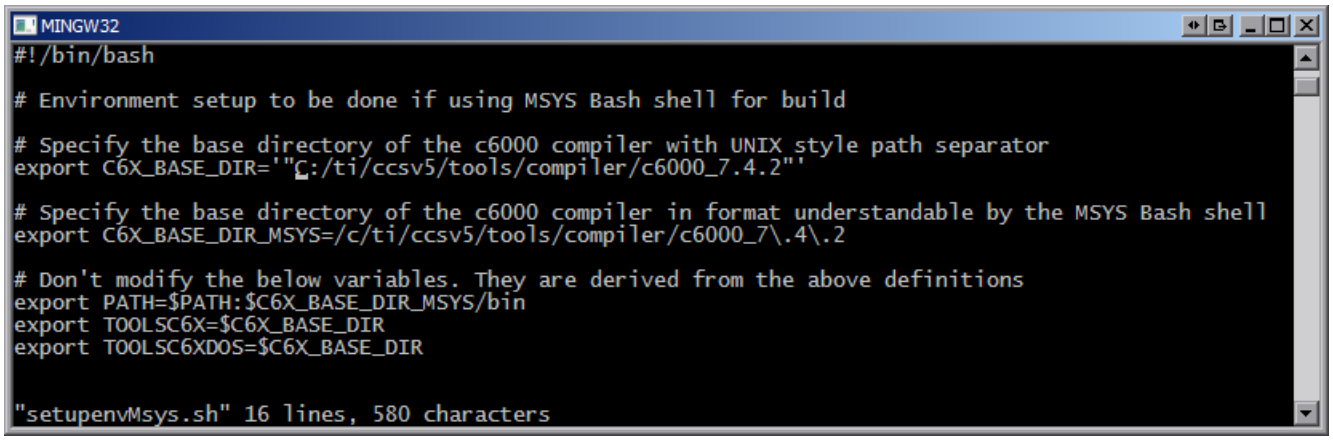


Рисунок 2-6: Редактирование файла «setupenvMsys.sh» в редакторе vim

Для редактирования файла «setupenvMsys.sh» в редакторе vim выполните в MinGW Shell команду:

```
vim /d/Dev/Modules/SVR-420/IBL/src/make/setupenvMsys.sh
```

В файле «D:/Dev/Modules/SVR-420/IBL/src/make/setupenvMsys.sh» путь к файлам компилятора для процессоров C6000 серии необходимо указать в качестве значений двух переменных — «C6X_BASE_DIR» и «C6X_BASE_DIR_MSYS». При этом, в значении переменной «C6X_BASE_DIR_MSYS» такие символы как пробел, точка, открывающая и закрывающая круглые скобки должны обязательно предваряться (экранироваться) символом обратной косой черты «\».

2.3 Сборка загрузчика

Для запуска процесса сборки загрузчика IBL для модуля SVR-420, выполните в MinGW Shell последовательно следующие команды:

```
cd /d/Dev/Modules/SVR-420/IBL/src/make/
source setupenvMsys.sh
make svr420
```

После выполнения этих команд, будет запущена сборка загрузчика IBL для модуля SVR-420, которая может занять до 30 минут (в зависимости от производительности системы).

В таблице 2-1 перечислены файлы, которые создаются в папке «D:/Dev/Modules/SVR-420/IBL/src/make/bin» после выполнения сборки загрузчика IBL для модуля SVR-420.

Таблица 2-1: Файлы, создаваемые при сборке загрузчика

Файл	Описание
«i2cConfig.gel»	GEL файл конфигурации IBL для CCS (см. раздел 4)
«i2cparam_0x50_svr420_le_0x500.out»	Бинарный образ программы конфигурации IBL для процессора TMS320C6678 модуля SVR-420 (см. раздел 4)
«i2crom_0x50_svr420_le.bin»	Бинарный образ IBL для процессора TMS320C6678 для загрузки в EEPROM память модуля SVR-420 (см. раздел 3.4)
«i2crom_0x50_svr420_le.dat»	Бинарный образ IBL для процессора TMS320C6678 модуля SVR-420 в формате CCS.

Для записи в EEPROM память модуля SVR-420 предназначен файл «i2crom_0x50_svr420_le.bin».

Запись образа загрузчика в EEPROM память проивзодится при помощи специального DSS скрипта для среды разработки CCS, работа с которым рассмотрена в разделе 3.

3 Скрипт для записи EEPROM и NOR флеш памяти модуля SVR-420

Для записи образов в EEPROM или NOR флеш память процессоров модуля SVR-420 предназначен специальный DSS скрипт «svr420_program.js», который расположен в папке «Program» на сопроводительном диске к модулю SVR-420.

Перед использованием скрипта «svr420_program.js» для записи EEPROM или NOR флеш памяти, необходимо скопировать с сопроводительного диска к модулю SVR-420 папки «Program», «TargetConfigurations» и «GEL» со всем содержимым в папку «D:/Dev/Modules/SVR-420»[1].

Скрипт «svr420_program.js» требует для своей работы установленную систему разработки CCS. Дистрибутив сетевой установки CCS имеется на сопроводительном диске к модулю SVR-420 (см. раздел 1).

Для запуска скрипта «svr420_program.js» в папке «Program» имеются вспомогательные скрипты запуска для Windows и Linux систем:

- «svr420_program.bat»: скрипт для запуска в Windows системе (см. листинг 3-1);
- «svr420_program.sh»: скрипт для запуска в Linux системе (см. листинг 3-2).

Листинг 3-1: Скрипт «svr420_program.bat»

```
@echo off
set PROGRAM_SVR_420_TARGET_CONFIG_FILE=../TargetConfigurations/SVR-420-LAN560v2-SD.ccxml
set DSS_SCRIPT_DIR="C:\ti\ccsv5\ccs_base\scripting\bin"
call %DSS_SCRIPT_DIR%\dss.bat svr420_program.js %*
```

Листинг 3-2: Скрипт «svr420_program.sh»

```
#!/bin/sh
export PROGRAM_SVR_420_TARGET_CONFIG_FILE=../TargetConfigurations/SVR-420-LAN560v2-SD.ccxml
export DSS_SCRIPT_DIR=~/ti/ccsv5/ccs_base/scripting/bin
$DSS_SCRIPT_DIR/dss.sh svr420_program.js $@
```

Как видно из листингов 3-1 и 3-2, в скриптах задаются значения для двух переменных окружения:

- «PROGRAM_SVR_420_TARGET_CONFIG_FILE»: путь к файлу описания целевой конфигурации модуля SVR-420;
- «DSS_SCRIPT_DIR»: путь к бинарным файлам DSS среды разработки CCS.

Важная информация

Только для Windows: В скрипте «svr420_program.bat» значение переменной «DSS_SCRIPT_DIR» обязательно должно быть записано в кавычках, а значение переменной «PROGRAM_SVR_420_TARGET_CONFIG_FILE», наоборот, не должно содержать кавычек.

После определения значений для переменных окружения, в скрипте происходит вызов скрипта кроссплатформенного DSS скрипта «svr420_program.js», с передачей ему аргументов командной строки.

Примечание

В данном документе рассматривается работа со скриптом «svr420_program.bat» для Windows системы. Для выполнения аналогичных действий в Linux системе, достаточно заменить вызовы скрипта «svr420_program.bat» на вызовы скрипта «svr420_program.sh».

В разделе 3.1 описывается структура каталога «Program», в котором работает скрипт «svr420_program.js». Далее, в разделе 3.2, описывается непосредственная работа со скриптом.

[1] Конечная папка может отличаться. В данном документе предполагается, что копирование выполнено именно в эту папку

3.1 Структура каталога «Program»

В листинге 3-3 представлена структура каталога «Program» с сопроводительного диска модуля SVR-420, в котором располагаются все необходимые файлы для выполнения записи в EEPROM или NOR флеш память процессоров модуля SVR-420.

Листинг 3-3: Структура каталога «Program»

```
Program
 +- bin
 |  +- ibl
 |  |  +- i2crom_0x50_svr420_le.bin
 |  |  +- i2cConfig.gel
 |  |  +- i2cparam_0x50_svr420_le_0x500.out
 |  |
 |  +- platform_test
 |  |  +- platform_test_svr420.bin
 |  |
 |  +- webserver
 |  |  +- websrv_svr420.bin
 |  |
 |  +- eeprom_0x50_c6678.bin
 |  +- nor_c6678.bin
 |
 +- logs
 |
 +- writers
 |  +- eepromwriter_c6678.out
 |  +- norwriter_c6678.out
 |
 +- svr420_program.bat
 +- svr420_program.sh
 +- svr420_program.js
```

Кроме файлов скриптов «svr420_program.bat», «svr420_program.sh» и «svr420_program.js», в каталоге «Program» имеется два вложенных каталога:

В каталоге «bin» расположены файлы образов готовые для записи в EEPROM и NOR флеш память процессоров модуля SVR-420. Пользовательские файлы, подготовленные для записи в EEPROM или NOR флеш память, должны быть помещены в каталог «bin».

Непосредственно в каталоге «bin» расположены два файла, которые используются скриптом «svr420_program.js» для записи по умолчанию:

- «eeprom_0x50_c6678.bin»: образ для записи в EEPROM память процессора TMS320C6678. По умолчанию, данный файлы представляет собой собранный образ загрузчика IBL, который можно получить путем выполнения шагов, описанных в разделе 2 данного документа.
- «nor_c6678.bin»: образ для записи в NOR флеш память процессора TMS320C6678. По умолчанию, данный файлы представляет собой собранный образ демонстрационного приложения веб-сервера. Подробное описание по сборке и запуску демонстрационного приложения веб-сервера дано в документе [1].

В каталогах «bin/webserver» и «bin/platform_test» расположены образы приложений, готовые для записи в NOR флеш память процессоров. Данные образы, также могут применяться для загрузки с TFTP сервера.

В каталоге «bin/platform_test» размещены образы приложения теста платформы (файл «platform_test_svr420.bin»), которые выполняют следующие тесты: UART, NOR флеш память, EEPROM память, DDR память и тест светодиодов на передней панели.

Важная информация

При запуске приложения теста платформы следует иметь в виду, что если во время выполнения тестов EEPROM памяти или NOR флеш памяти отключить питание модуля, содержимое тестируемого типа памяти может быть повреждено.

В каталоге «bin/webserver» находится образ собранного приложения демонстрационного веб-сервера (файл «websrv_svr420.bin»). Подробное описание данного приложение дано в документе [1].

Исходные коды приложения теста платформы находятся проекте «PlatformTest_C6678» рабочего пространства CCS. Папку с рабочим пространством можно найти на сопроводительном диске к модулю SVR-420 в папке «CCS_Workspace».

В каталоге «writers» расположены файлы программ, для осуществления записи в EEPROM и NOR флеш память процессоров. Данные файлы получены путем сборки проектов «EEPROM_Writer» (файл «eepromwriter_c6678.out») и NOR_Writer» (файл «norwriter_c6678.out») рабочего пространства CCS для модуля SVR-420. Папку с рабочим пространством со всеми проектами можно найти на сопроводительном диске в папке «CCS_Workspace».

В каталоге «logs» сохраняются логи вывода CIO с процессоров в случае запуска скрипта с параметром `log` (см. таблицу 3-1).

3.2 Работа со скриптом записи NOR флеш и EEPROM памяти

Работа со скриптом записи EEPROM и NOR флеш памяти модуля SVR-420 осуществляется из командной строки. Выбор того или иного режима работы скрипта осуществляется при помощи параметров командной строки, передаваемых скрипту при запуске.

Если запустить скрипт «svr420_program.bat» без параметров, будет выдана краткая справка по возможным параметрами командной строки (см. листинг 3-4).

Листинг 3-4: Вывод скрипта «svr420_program.bat», запущенного без параметров

```
D:\Dev\Modules\SVR-420\Program>svr420_program.bat

Syntax: svr420_program.js <NOR|EEPROM> <options>

Mode:
  NOR    : program NOR flash memory
  EEPROM : program I2C EEPROM memory

Options:
  dsp<n>               : program DSP<n>
  all                  : program all DSP's (DSP1 and DSP2)
  image=<image>        : image file for all DSP's
  image_dsp<n>=<image> : individual image for DSP<n>
                         (image must be in 'bin' folder)
  bus_dsp<n>=<address> : I2C bus EEPROM address for DSP<n>
                         Adress must be a decimal number
                         Only for EEPROM mode. Default is 80 (0x50)
  bus=<address>        : I2C bus EEPROM address for all DSP's
                         Adress must be a decimal number
                         Only for EEPROM mode. Default is 80 (0x50)
  log                  : Enable CIO logging

D:\Dev\Modules\SVR-420\Program>
```

Таким образом, первый параметр командной строки задает режим работы скрипта и может принимать одно из значений:

- `NOR`: режим записи NOR флеш памяти;
- `EEPROM`: режим записи EEPROM памяти.

Количество остальных параметров командной строки может варьироваться. Доступные параметры приведены в таблице 3-1.

Таблица 3-1: Параметры командной строки скрипта «svr420_program.js»

Параметр	Описание
`dsp<n>`	Включает запись образа в память для процессора с номером `<n>`. Значение `<n>` может быть от 1 до 2: • 1: DSP1_TMS320C6678; • 2: DSP2_TMS320C6678.
`all`	Включает запись образа в память сразу для 2-х процессоров модуля SVR-420. Эквивалентно заданию параметров командной строки `dsp<n>` для всех `<n>`.
`image=<image>`	Параметр задает имя файла (`<image>`) образа для записи на обоих процессорах. Может использоваться в том случае, когда на все процессоры необходимо записать один и тот же образ.
`image_dsp<n>=<image>`	Параметр позволяет указать имя файла (`<image>`) образа для конкретного процессора `<n>`.
`bus_dsp<n>=<address>`	Параметр позволяет указать адрес (`<address>`) I^2C шины EEPROM памяти для записи образа для конкретного процессора `<n>`. Значение данного параметра учитывается только в режиме `EEPROM`. По умолчанию, если параметр не задан, используется адрес шины 80 (0x50).
`bus=<address>`	Параметр позволяет указать адрес (`<address>`) I^2C шины EEPROM памяти для записи образа сразу для всех 2-х процессоров. Значение данного параметра учитывается только в режиме `EEPROM`. По умолчанию, если параметр не задан, используется адрес шины 80 (0x50).

Продолжение таблицы на следующей странице

Продолжение таблицы 3-1

Параметр	Описание
`log`	Параметр включает запись вывода CIO со всех процессоров модуля SVR-420 в файлы, которые будут помещены в каталог «logs». Создаваемые файлы лога имеют формат «`SVR-420-DSP<n>-<time>-cio.txt`», где `<n>` — номер процессора модуля SVR-420 (от 1 до 4), `<time>` — время запуска скрипта «svr420-program.js». По умолчанию, запись вывода CIO в файлы отключена.

Важная информация

Значения адреса шины для параметров `bus_dsp<n>` и `bus` обязательно должны указываться в десятичном виде. Соответственно, для адреса шины 0x50 (значение по умолчанию) необходимо указывать значение 80, а для адреса шины 0x51 — значение 81.

3.3 Запись образов в NOR флеш память

Для записи образа в NOR флеш память модуля SVR-420 выполните шаги процедуры 3-1.

Внимание

Запись в NOR флеш память рекомендуется выполнять при включенном режиме «NOBOOT» на соответствующем процессоре (см. приложение B).

Процедура 3-1. Запись образов в NOR флеш память

1. Выполните подготовку образа(ов).

   Внимание

   Следует помнить, что объем NOR флеш памяти, установленной на модуле SVR-420, составляет 16 Мбайт на каждый процессор. Таким образом, ограничение на размер образов, записываемых в NOR флеш память, составляет 16 Мбайт.

2. Скопируйте подготовленный(е) для записи образ(ы) в каталог «D:/Dev/Modules/SVR-420/Program/bin».
3. Перейдите в каталог «D:/Dev/Modules/SVR-420/Program».
4. Запустите скрипт «svr420_program.bat» в режиме `NOR` с требуемыми параметрами командной строки (см. таблицу 3-1 раздела 3.2).

В процедуре 3-1 представлены общие шаги, необходимые для записи образа в NOR флеш память процессоров модуля SVR-420.

В качестве примера, в процедуре 3-2 подробно описаны шаги, необходимые для выполнения записи в NOR флеш память всех двух процессоров модуля SVR-420 образа приложения демонстрационного веб-сервера [1]. начиная от сборки приложения, и заканчивая его загрузкой с NOR флеш памяти при помощи загрузчика IBL.

Важная информация

Записываемые в NOR флеш память файлы образов обязательно должны иметь расширение «.bin».

При сборке приложений в CCS, по умолчанию, файлы имеют расширение «.out». Перед выполнением записи, необходимо переименовать файлы таким образом, чтобы они имели расширение «.bin».

Процедура 3-2. Запись образа демонстрационного приложения веб-сервера в NOR флеш память и его загрузка

1. Выполните сборку приложения веб-сервера для конфигураций сборки «Debug» (процесс сборки приложения веб-сервера подробно описан в документе [1]).

   В результате сборки приложений веб-сервера должен быть получен файл «D:/Dev/Modules/SVR-420/CCS_Workspace/WebServer_C6678/Debug/websrv_svr420.out»:

   Примечание

   Модуль SVR-420 поставляется с записанным приложением теста платформы, в котором выполняются последовательно тест светодиодов на передней панели модуля, частичный тест EEPROM памяти, частичный тест NOR флеш памяти и полный тест внешней DDR памяти. Кроме того, записанный в NOR флеш память тест выводит различную информацию о модуле. Пример вывода в UART теста платформы приведен в листинге 3-6.

2. Скопируйте файл, полученный при выполнении шага 1 в папку «D:/Dev/Modules/SVR-420/Program/bin». Для этого выполните в терминале последовательно следующие команды (при копировании, также выполняется изменение расширения «.out» на «.bin», что необходимо для правильной записи):

```
d:
cd D:\Dev\Modules\SVR-420
copy CCS_Workspace\WebServer_C6678\Debug\websrv_svr420.out Program\bin\websrv_svr420.bin
```

3. Установите для всех процессоров модуля SVR-420 режим «NOBOOT» в соответствии с данными таблицы B-1 приложения B.
4. Подключите отладчик к модулю SVR-420 и к компьютеру (предполагается использование отладчика Spectrum Digital XDS560v2 STM LAN).
5. Включите модуль SVR-420.
6. Перейдите в каталог «D:/Dev/Modules/SVR-420/Program». Для этого необходимо выполнить в терминале команды:

```
d:
cd D:\Dev\Modules\SVR-420\Program
```

7. Запустите скрипт «svr420_program.bat» выполнив в терминале команду:

```
svr420_program.bat NOR all image=websrv_svr420.bin
```

Будет запущен процесс записи образов в NOR флеш память процессоров модуля SVR-420. При этом, вывод в терминал должен быть аналогичен тому, что приведен в листинге A-1 приложения A. Весь процесс записи должен занимать не более 5 минут.

После вывода в терминал сообщения «`Executing writers on DSP(s)...`» на процессорах модуля SVR-420 начинается процесс записи данных образов в NOR флеш память. При этом, светодиоды в соответствующей группе («DSP1» и/или «DSP2») на передней панели модуля SVR-420 будут иметь определенное состояние в зависимости от хода процесса записи (см. рисунок 3-1).

Рисунок 3-1: Состояния светодиодов на передней панели модуля SVR-420 во время записи

При нормальном процессе записи и верификации записанных данных, должен гореть только зеленый светодиод «LD1». Желтый светодиод «LD2» гореть не должен.

Если во время записи или верификации происходит ошибка, то загорается желтый светодиод «LD2», зеленый светодиод «LD1» гаснет (см. рисунок 3-1). При этом, работа программы записи прекращается.

В случае успешного завершения записи и верификации записанных данных, загораются оба светодиода «LD1» и «LD2».

8. Выключите модуль SVR-420.
9. Если в EEPROM память процессоров модуля SVR-420 еще не записан образ загрузчика IBL, выполните шаги процедуры 3-4.

Примечание

Модуль SVR-420 поставляется с уже записанным образом загрузчика IBL в EEPROM память (адрес I^2C шины 0x50) обоих процессоров модуля. Поэтому, если в EEPROM память (адрес I^2C шины 0x50) не производилась запись каких либо других образов, выполнять запись загрузчика не требуется.

10. Установите для всех процессоров модуля SVR-420 режим «NOR» в соответствии с данными таблицы B-1 приложения B.
11. Включите модуль SVR-420.

После выполнения всех шагов процедуры 3-2, вывод в UART с процессоров модлуя SVR-420 при его включении должен выглядеть подобно приведенному в листинге 3-5. В листинге 3-5 приведен вывод в UART загрузки демонстрационного приложения веб-сервера на процессоре «DSP1_C6678». Вывод в UART с других процессоров должен выглядеть аналогично.

Листинг 3-5: Вывод в UART загрузки демонстрационного приложения веб-сервера с NOR флеш памяти

```
1 IBL INIT
2 I2C boot
```

```
EDC enabled
-- Device is C6678
Boot table processor executed

IBL version: 1.0.0.16-5
Device is SVR-420
b4:99:4c:0d:ea:05
IBL: PLL and DDR Initialization Complete
IBL Result code 00
Multiboot enabled
SPI initialized
IBL: Booting from NOR
Booting from NOR flash from address
  = 0x8157c8e0
Platform initialized
Start BIOS 6
QMSS successfully initialized
CPPI successfully initialized
PA successfully initialized
HUA version 2.00.00.04
Setting hostname to setdemo
MAC Address: B4-99-4C-0D-EA-05
EMAC ports count: 2
Configuring DHCP client
PASS successfully initialized
Ethernet subsystem successfully initialized
Ethernet eventId : 48 and vectId (Interrupt) : 7
Registration of the EMAC Successful, waiting for link up ..
Service Status: DHCPC    : Enabled  :          : 000
Service Status: THTTP    : Enabled  :          : 000
Service Status: DHCPC    : Enabled  : Running  : 000
Network Added: If-1:192.168.2.85
Service Status: DHCPC    : Enabled  : Running  : 017
```

В листинге 3-6 приведен пример вывода в UART запуска приложения теста платформы модуля SVR-420 (файл «Program/bin/platform_test_svr420.bin»).

Листинг 3-6: Вывод в UART при запуске теста платформы

```
Platform test for module
  _______      _______        _  _ ___   ___
 / ____\ \    / /  __ \      | || |__ \ / _ \
| (___  \ \  / /| |__) |_____| || |_ ) | | | |
 \___ \  \ \/ / |  _  /______|__   _/ /| | | |
 ____) |  \  /  | | \ \         | |/ /_| |_| |
|_____/    \/   |_|  \_\        |_|____|\___/

p_info->version                            = 2.00.00.14
p_info->board_name                         = SVR-420
p_info->board_id                           = SVR-420 v1.0
p_info->serial_nbr                         = 5290102
p_info->slot_n                             = 62
p_info->dsp_n                              = DSP1
p_info->cpu.core_count                     = 8
p_info->cpu.name                           = TMS320C6678
p_info->cpu.id                             = 21
p_info->cpu.revision_id                    = 0
p_info->cpu.endian                         = 1
p_info->frequency                          = 999 MHz
p_info->led[PLATFORM_USER_LED_CLASS].count = 2
p_info->emac.port_count                    = 2
  -> EMAC port 0
     MAC address = b4:99:4c:0d:ea:05
  -> EMAC port 1
     MAC address = b4:99:4c:0d:ea:05

NOR device info:
p_device->device_id                        = 0xbb18
p_device->manufacturer_id                  = 0x20
p_device->width                            = 8
p_device->block_count                      = 256
p_device->page_count                       = 256
p_device->page_size                        = 256
p_device->spare_size                       = 0
p_device->handle                           = 0xbb18
p_device->flags                            = 0
p_device->bboffset                         = 0
```

```
EEPROM device (I2C address 0x50) info:
p_device->device_id                         = 0x50
p_device->manufacturer_id                   = 0x1
p_device->width                             = 8
p_device->block_count                       = 1
p_device->page_count                        = 1
p_device->page_size                         = 65536
p_device->spare_size                        = 0
p_device->handle                            = 0x50
p_device->flags                             = 0
p_device->bboffset                          = 0

EEPROM device (I2C address 0x51) info:
p_device->device_id                         = 0x51
p_device->manufacturer_id                   = 0x1
p_device->width                             = 8
p_device->block_count                       = 1
p_device->page_count                        = 1
p_device->page_size                         = 65536
p_device->spare_size                        = 0
p_device->handle                            = 0x51
p_device->flags                             = 0
p_device->bboffset                          = 0

Current core id is 0

----------
 LED test
----------
Testing USER class leds (2 leds)...
- LED 0 ON
- LED 0 OFF
- LED 0 ON
- LED 0 OFF
- LED 1 ON
- LED 1 OFF
- LED 1 ON
- LED 1 OFF
LED test complete

-------------
 EEPROM test
-------------
test_eeprom: Partial EEPROM test mode
test_eeprom: 0x50: Saving EEPROM contents...
test_eeprom: 0x50: Writing 1024 bytes...
test_eeprom: 0x50: Read and check data...
test_eeprom: 0x50: Restoring EEPROM contents...
test_eeprom: 0x50: Test passed
test_eeprom: 0x51: Saving EEPROM contents...
test_eeprom: 0x51: Writing 1024 bytes...
test_eeprom: 0x51: Read and check data...
test_eeprom: 0x51: Restoring EEPROM contents...
test_eeprom: 0x51: Test passed
EEPROM test complete

----------
 NOR test
----------
test_nor: Partial NOR test mode
test_nor: NOR flash sector size is 65536 bytes
test_nor: Test block size is 65536 bytes
test_nor: Saving original data from NOR...
test_nor: Writing pattern data to NOR...
test_nor: Reading pattern data from NOR...
test_nor: Comparing data...
test_nor: Data is equal. Test passed.
test_nor: Write back original data to NOR...
test_nor: Test passed
NOR test complete

----------------------
 External memory test
----------------------
Memory test: start = 0x80000000, end = 0xffffffff
Memory test: Write a pattern...
Memory test: Read and check pattern...
```

```
Memory test: Read and check pattern...
Memory test: Write a pattern for complementary values...
Memory test: Read and check pattern...
External memory test passed
External memory test complete

Test completed
```

3.4 Запись образов в EEPROM память

Для записи образа в EEPROM память модуля SVR-420 выполните шаги процедуры 3-3.

Внимание

Запись в EEPROM память рекомендуется выполнять при включенном режиме «NOBOOT» на соответствующем процессоре (см. приложение B).

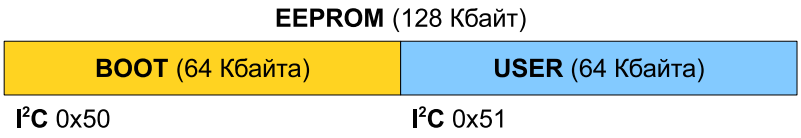
Процедура 3-3. Запись образов в EEPROM память

1. Выполните подготовку образа(ов).

   Внимание

   Следует помнить, что общий объем EEPROM памяти, установленной на модуле SVR-420, составляет 128 Кбайт на каждый процессор. EEPROM память, установленная на модуле SVR-420, разделена на две части (каждая объемом по 64 Кбайта): адресу 0x50 шины I^2C соответствует первая часть, а адресу 0x51 шины I^2C — вторая часть (см. рисунок 3-2). При этом, загрузка модуля SVR-420 возможна только из первой части EEPROM с адресом шины I^2C 0x50. Таким образом, ограничение на размер образов, записываемых в EEPROM память, составляет 64 Кбайт.

2. Скопируйте подготовленный(е) для записи образ(ы) в каталог «D:/Dev/Modules/SVR-420/Program/bin».
3. Перейдите в каталог «D:/Dev/Modules/SVR-420/Program».
4. Запустите скрипт «svr420_program.bat» в режиме `EEPROM` с требуемыми параметрами командной строки (см. таблицу 3-1 раздела 3.2).



BOOT — область загрузчика IBL;
USER — область пользовательских данных

Рисунок 3-2: Организация EEPROM памяти на модуле SVR-420

В процедуре 3-3 представлены общие шаги, необходимые для записи образа в EEPROM память процессоров модуля SVR-420.

Важная информация

Записываемые в EEPROM память файлы образов обязательно должны иметь расширение «.bin».

В качестве примера, в процедуре 3-4 подробно описаны шаги, необходимые для выполнения записи в EEPROM память всех четырех процессоров модуля SVR-420 образа загрузчика IBL.

Процедура 3-4. Запись образа загрузчка IBL в EEPROM память и его загрузка

1. Выполните сборку загрузчика IBL (процесс сборки IBL подробно описан в разделе 2).

   В результате сборки загрузчика IBL должен быть получен файл «D:/Dev/Modules/SVR-420/ibl/src/make/bin/i2crom_0x50_svr420_le.bin».

Примечание

Модуль SVR-420 поставляется с уже записанным образом загрузчика IBL в EEPROM память обоих процессоров. Образ, записанный в EEPROM память при изготовлении модуля SVR-420 содержится в файле «i2crom_0x50_svr420_le.bin». Данный файл расположен в каталоге «D:/Dev/Modules/SVR-420/Program/bin/ibl».

2. Скопируйте файл, полученный при выполеннии шага 1 в папку «D:/Dev/Modules/SVR-420/Program/bin». Для этого выполните в терминале последовательно следующие команды:

```
d:
cd D:\Dev\Modules\SVR-420
copy ibl\src\make\bin\i2crom_0x50_svr420_le.bin Program\bin
```

3. Установите для всех процессоров модуля SVR-420 режим «NOBOOT» в соответствии с данными таблицы B-1 приложения B.
4. Подключите отладчик к модулю SVR-420 и к компьютеру (предполагается использование отладчика Spectrum Digital XDS560v2 STM LAN).
5. Включите модуль SVR-420.
6. Перейдите в каталог «D:/Dev/Modules/SVR-420/Program». Для этого необходимо выполнить в терминале команды:

```
d:
cd D:\Dev\Modules\SVR-420\Program
```

7. Запустите скрипт «svr420_program.bat» выполнив в терминале команду:

```
svr420_program.bat EEPROM all image=i2crom_0x50_svr420_le.bin
```

   Будет запущен процесс записи образа в EEPROM память процессоров модуля SVR-420. При этом, вывод в терминал должен быть аналогичен тому, что приведен в листинге A-2 приложения A. Весь процесс записи должен занимать не более 5 минут.

   Состояния светодиодов «DSP» на передней панели модуля SVR-420 во время записи EEPROM памяти соответствуют состояниям светодиодов во время записи NOR флеш памяти, которые описаны в шаге 7 процедуры 3-2.
8. Выключите модуль SVR-420.
9. Установите для всех процессоров модуля SVR-420 требуемый режим загрузки («NOR», «TFTP» или «PCI-E») в соответствии с данными таблицы B-1 приложения B.
10. Включите модуль SVR-420.

4 Конфигурация IBL

Записанный в EEPROM память загрузчик IBL хранит блок с конфигурационными параметрами в этой же EEPROM памяти с определенным смещением. Конфигурация IBL заключается в изменении данных блока с конфигурационными параметрами в EEPROM памяти.

Для облегчения процесса изменения данных в блоке конфигурационных данных загрузчика, при сборке IBL (см. раздел 2), создается образ специальной программы «i2cparam_0x50_svr420_le_0x500.out» (см. таблицу 2-1), которая предназначены для загрузки на процессорах модуля SVR-420 для конфигурации загрузчика.

Примечание

Значение «0x500» в имени файлов «i2cparam_0x50_svr420_le_0x500.out» определяет величину смещения расположения блока конфигурационных параметров загрузчика IBL относительно начала EEPROM памяти.

Для запуска программы конфигурации загрузчика IBL необходимо, в первую очередь, запустить целевую конфигурацию модуля SVR-420, как показано в разделе 5.

После запуска целевой конфигурации, как показано в разделе 5, выполните подключение к требуемому процессору (в данном примере рассматрена конфигурация загрузчика на процессоре «DSP1_C6678»). Для этого, нажмите правой кнопкой мыши на названии ядра «DSP1_C6678_0» и выберите пункт меню «Connect Target» (см. рисунок 4-1).

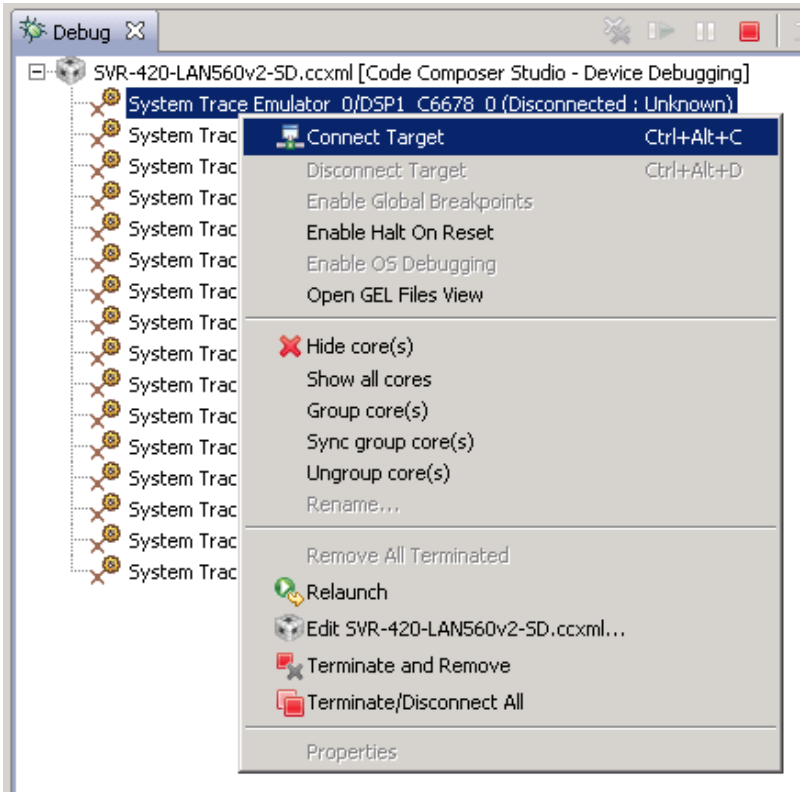


Рисунок 4-1: Подлкючение к процессору «DSP1_C6678»

Выберите ядро «DSP1_C6678_0» в окне «Debug» (щелкните левой кнопкой мыши по названию ядра). Выберите пункт главного меню «Run > Load > Load Program...». В открывшемся окне (рисунок 4-2) нажмите на кнопку «Browse».

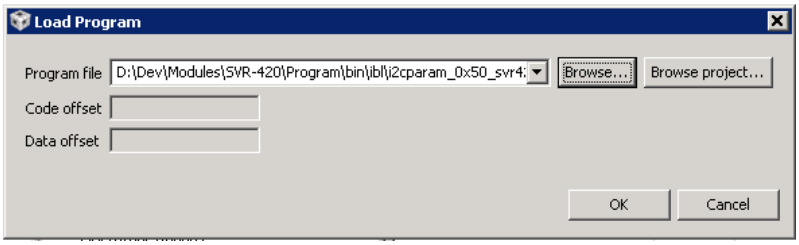


Рисунок 4-2: Окно загрузки кода на ядро процессора

В открывшемся окне выбора файлов необходимо выбрать файл «i2cparam_0x50_svr420_le_0x500.out» (см. таблицу 2-1) и нажать на кнопку «OK». Данный файл также можно найти в папке «D:/Dev/Modules/SVR-420/Program/bin/ibl».

После загрузки кода на ядро процессора, окно «Debug» должно выглядеть, как показано на рисунке 4-3.

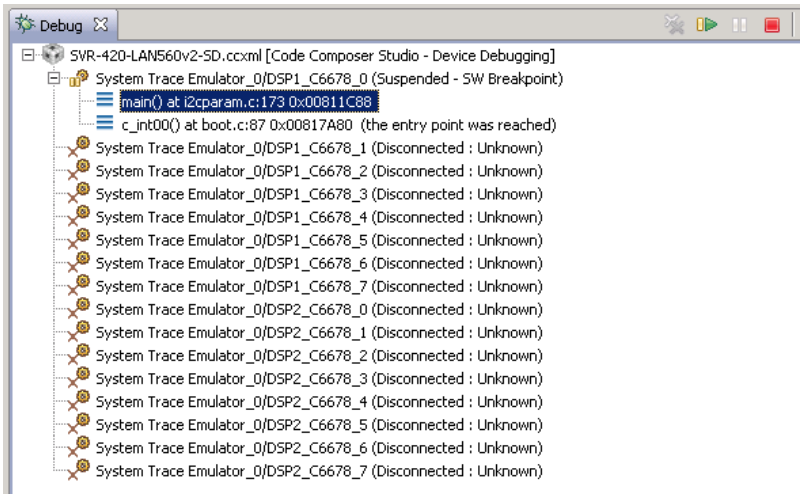


Рисунок 4-3: Внешний вид окна «Debug» после загрузки кода на ядра процессоров

Выделите в окне «Debug» ядро «DSP1_C6678_0» щелкнув по нему левой кнопкой мыши. и запустите выполнение кода, нажав на кнопку («Resume»). Кнопка («Resume») находится в верхней части окна «Debug» (см. рисунок 4-3).

После запуска приложения конфигурации загрузчика, в окно «Console» будет выведено сообщение «Run the GEL for the device to be configured, press return to programm the I2C» (см. рисунок 4-4).

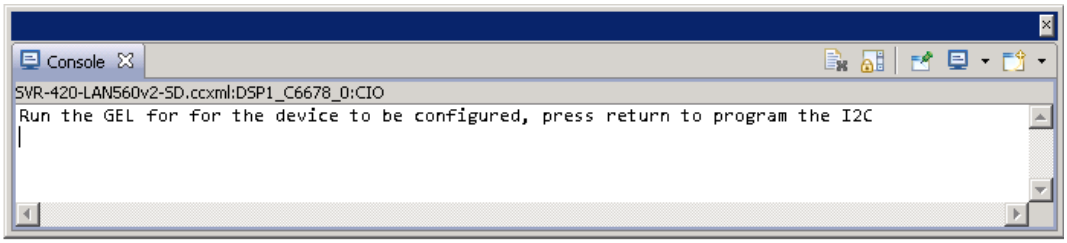


Рисунок 4-4: Внешний вид окна «Console» после запуска кода на процессоре

Данное сообщение означает, что программа ожидает пока будет произведена конфигурация параметров загрзучика IBL и предлагает нажать на клавишу «Enter» для записи выполненой конфигурации в EEPROM память.

Конфигурация параметров загрузчика осуществляется при помощи специального GEL файла «i2cConfig.gel», который можно найти в папке «D:/Dev/Modules/SVR-420/Program/bin/ibl».

Для вызова окна управления GEL файлами, выберите пункт главного меню «Tools > GEL Files», как показано на рисунке 4-5.

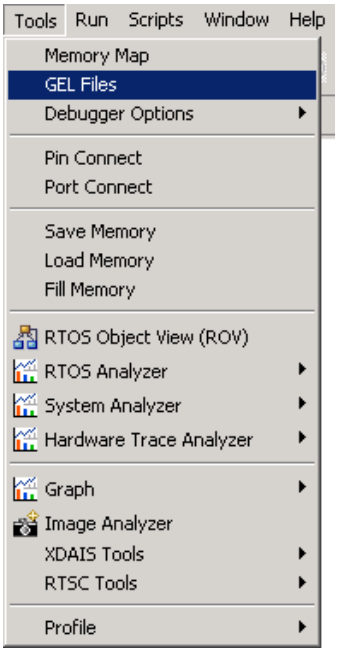


Рисунок 4-5: Пункт главного меню для вызова окна управления GEL файлами

В открывшемся окне «GEL Files» (рисунок 4-6) нажмите правой кнопкой мыши в области со списком загруженных GEL файлов (изначально там должен быть только один файл «svr420.gel») и выберите пункт меню «Load GEL...».

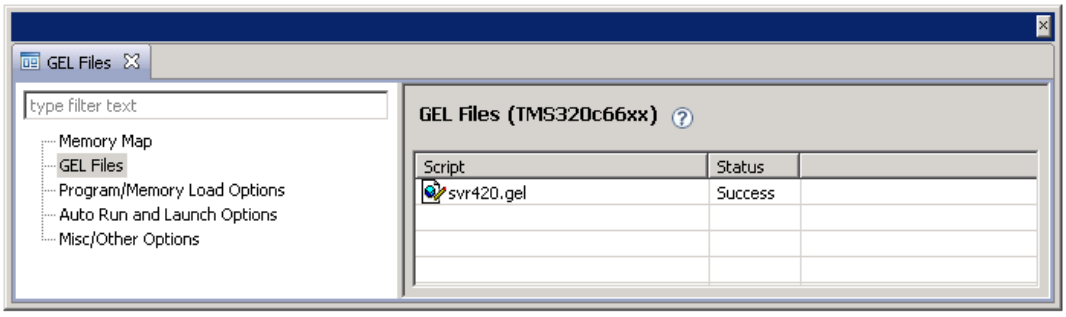


Рисунок 4-6: Окно управления GEL файлами

В открывшемся окне необходимо выбрать файл «D:/Dev/Modules/SVR-420/Program/bin/ibl/i2cConfig.gel». После чего, в списке загруженных файлов окна «GEL Files» должен появиться файл «i2cConfig.gel», как показано на рисунке 4-7.

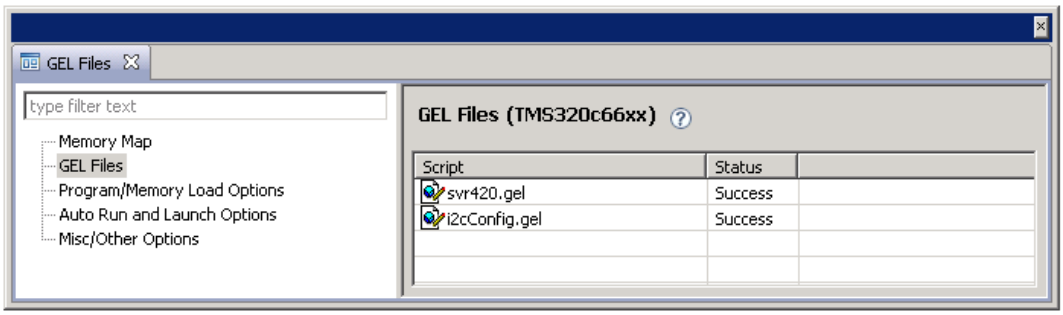


Рисунок 4-7: Окно управления GEL файлами с загруженным файлом «i2cConfig.gel»

После выполнения данных действий, появится пункт меню «Scripts > SET SVR-420 IBL > setConfig_svr420_main» при выборе которого будет применена конфигурация загрузчика (см. рисунок 4-8).

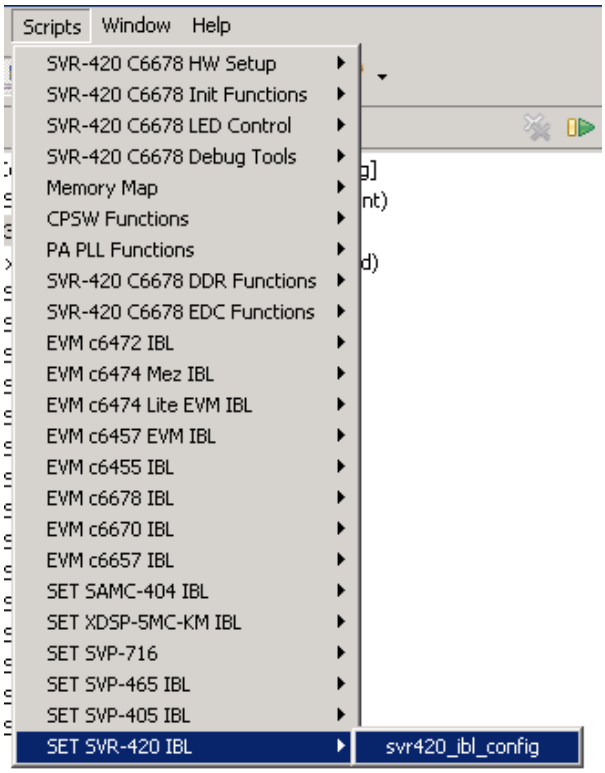


Рисунок 4-8: Пункт главного меню «Scripts > SET SVR-420 IBL»

Далее, необходимо шелкнуть мышкой в окне «Console» и нажать клавишу «Enter» для выполнения записи конфигурационных параметров загрузчика в EEPROM память. После чего, в окне «Console» должно появиться сообщение «I2c table write complete» (см. рисунок 4-9).

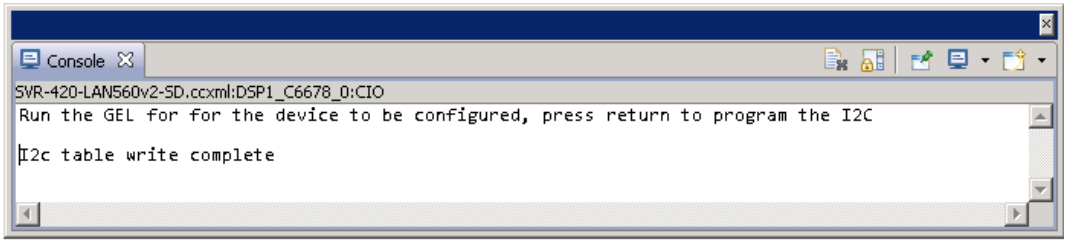


Рисунок 4-9: Внешний вид окна «Console» после выполнения записи конфигурации в EEPROM память

Конфигурационные параметры загрузчика IBL задаются в загружаемом файле «i2cConfig.gel» путем его редактирования. Редактирование данного файла должно производиться до его загрузки в окно «GEL Files». В том случае, если файл редактируется после того, как он загружен в окне «GEL Files», после редактирования, необходимо щелкнуть правой кнопкой мыши по названию файла в окне «GEL Files» и выбрать пункт меню «Reload» (см. рисунок 4-6).

В листинге 4-1 приведен фрагмент файла «i2cConfig.gel» с конфигурационными параметрами загрузчика для модуля SVR-420.

Листинг 4-1: Фрагмент GEL файла «i2cConfig.gel» с конфигурационными параметрами загрузчика IBL для модуля SVR-420

```
1691 menuitem "SET SVR-420 IBL";
1692
1693 hotmenu svr420_ibl_config()
1694 {
1695     ibl.iblMagic = ibl_MAGIC_VALUE;
1696
1697     ibl.iblEvmType = ibl_SVR_420;
1698
1699     /* Main PLL: 122.88 MHz reference, 1000MHz output */
1700     ibl.pllConfig[ibl_MAIN_PLL].doEnable      = 1;
1701     ibl.pllConfig[ibl_MAIN_PLL].prediv        = 29;
1702     ibl.pllConfig[ibl_MAIN_PLL].mult          = 472;
1703     ibl.pllConfig[ibl_MAIN_PLL].postdiv       = 2;
1704     ibl.pllConfig[ibl_MAIN_PLL].pllOutFreqMhz = 1000;
1705
1706     /* DDR PLL: 66.67 MHz reference, 1333 MHz output */
```

```
    /* outFreqMhz = 2 * inFreq * mult / postdiv = 66.667 * 2 * 20 / 2 = 1333.3 */

    /* Net PLL: 122.88 MHz reference, 1044 MHz output (followed by a built in divide by 3 to give 348 MHz to PA) */
    /* outFreqMhz = (inFreq * mult / postdiv) / 3 = (122.88 * 17 / 2) / 3 = 1044.48 / 3 = 348.16 */

    /* DDR PLL: */
    ibl.pllConfig[ibl_DDR_PLL].doEnable       = 1;
    ibl.pllConfig[ibl_DDR_PLL].prediv         = 1;
    ibl.pllConfig[ibl_DDR_PLL].mult           = 20;
    ibl.pllConfig[ibl_DDR_PLL].postdiv        = 2;
    ibl.pllConfig[ibl_DDR_PLL].pllOutFreqMhz  = 1333;

    /* Net PLL: 122.88 MHz reference, 1044 MHz output (followed by a built in divide by 3 to give 350 MHz to PA) */
    ibl.pllConfig[ibl_NET_PLL].doEnable       = 1;
    ibl.pllConfig[ibl_NET_PLL].prediv         = 1;
    ibl.pllConfig[ibl_NET_PLL].mult        = 17;
    ibl.pllConfig[ibl_NET_PLL].postdiv        = 2;
    ibl.pllConfig[ibl_NET_PLL].pllOutFreqMhz  = 1044;


    ibl.ddrConfig.configDdr = 1;
    ibl.ddrConfig.uEmif.emif4p0.registerMask = ibl_EMIF4_ENABLE_sdRamConfig | ibl_EMIF4_ENABLE_sdRamRefreshCtl |
      ← ibl_EMIF4_ENABLE_sdRamTiming1 | ibl_EMIF4_ENABLE_sdRamTiming2 | ibl_EMIF4_ENABLE_sdRamTiming3 |
      ← ibl_EMIF4_ENABLE_ddrPhyCtl1;

    // NOTE: this values is not used for configuration on SVP-465
    // Configuration for EMIF is hardcoded in 'hw/ddrs/emif4/emif4.c'
    ibl.ddrConfig.uEmif.emif4p0.sdRamConfig         = 0x63062A32;
    ibl.ddrConfig.uEmif.emif4p0.sdRamConfig2            = 0;
    ibl.ddrConfig.uEmif.emif4p0.sdRamRefreshCtl     = 0x00005161;
    ibl.ddrConfig.uEmif.emif4p0.sdRamTiming1            = 0x1113783C;
    ibl.ddrConfig.uEmif.emif4p0.sdRamTiming2            = 0x30B37FE3;
    ibl.ddrConfig.uEmif.emif4p0.sdRamTiming3            = 0x559F8ADF;
    ibl.ddrConfig.uEmif.emif4p0.lpDdrNvmTiming          = 0;
    ibl.ddrConfig.uEmif.emif4p0.powerManageCtl          = 0;
    ibl.ddrConfig.uEmif.emif4p0.iODFTTestLogic          = 0;
    ibl.ddrConfig.uEmif.emif4p0.performCountCfg     = 0;
    ibl.ddrConfig.uEmif.emif4p0.performCountMstRegSel   = 0;
    ibl.ddrConfig.uEmif.emif4p0.readIdleCtl         = 0;
    ibl.ddrConfig.uEmif.emif4p0.sysVbusmIntEnSet        = 0;
    ibl.ddrConfig.uEmif.emif4p0.sdRamOutImpdedCalCfg    = 0;
    ibl.ddrConfig.uEmif.emif4p0.tempAlterCfg            = 0;
    ibl.ddrConfig.uEmif.emif4p0.ddrPhyCtl1          = 0x0010010f;
    ibl.ddrConfig.uEmif.emif4p0.ddrPhyCtl2              = 0;
    ibl.ddrConfig.uEmif.emif4p0.priClassSvceMap     = 0;
    ibl.ddrConfig.uEmif.emif4p0.mstId2ClsSvce1Map       = 0;
    ibl.ddrConfig.uEmif.emif4p0.mstId2ClsSvce2Map       = 0;
    ibl.ddrConfig.uEmif.emif4p0.eccCtl                  = 0;
    ibl.ddrConfig.uEmif.emif4p0.eccRange1               = 0;
    ibl.ddrConfig.uEmif.emif4p0.eccRange2               = 0;
    ibl.ddrConfig.uEmif.emif4p0.rdWrtExcThresh          = 0;

        // Rear VPX RP3/5
    ibl.sgmiiConfig[0].configure     = 0;
    ibl.sgmiiConfig[0].adviseAbility = 0x9801;  // 1
    ibl.sgmiiConfig[0].control       = 32;      // 1
    ibl.sgmiiConfig[0].txConfig      = 0x108a1; // 0x108a1
    ibl.sgmiiConfig[0].rxConfig      = 0x700621;
    ibl.sgmiiConfig[0].auxConfig     = 0x51;

    // PHY RJ-45
    ibl.sgmiiConfig[1].configure     = 1;
    ibl.sgmiiConfig[1].adviseAbility = 0x9801;//1;  //
    ibl.sgmiiConfig[1].control       = 1;//1;
    ibl.sgmiiConfig[1].txConfig      = 0x108a1;//0x108a1;
    ibl.sgmiiConfig[1].rxConfig      = 0x700621;
    ibl.sgmiiConfig[1].auxConfig     = 0x51;

    ibl.mdioConfig.nMdioOps = 0;

    ibl.spiConfig.addrWidth  = 24;
    ibl.spiConfig.nPins      = 5;
    ibl.spiConfig.mode       = 1;
    ibl.spiConfig.csel       = 2;
    ibl.spiConfig.c2tdelay   = 8;
    ibl.spiConfig.busFreqMHz = 20;

    ibl.emifConfig[0].csSpace    = 2;
    ibl.emifConfig[0].busWidth   = 8;
    ibl.emifConfig[0].waitEnable = 0;

    ibl.emifConfig[1].csSpace    = 0;
    ibl.emifConfig[1].busWidth   = 0;
    ibl.emifConfig[1].waitEnable = 0;

    ibl.bootModes[0].bootMode = ibl_BOOT_MODE_NOR;
    ibl.bootModes[0].priority = ibl_HIGHEST_PRIORITY;
    ibl.bootModes[0].port     = 0;
```

```
1793
1794      ibl.bootModes[0].u.norBoot.bootFormat   = ibl_BOOT_FORMAT_ELF;
1795      ibl.bootModes[0].u.norBoot.bootAddress[0][0]    = 0;        /* Image 0 NOR offset byte address in LE mode */
1796      ibl.bootModes[0].u.norBoot.bootAddress[0][1]    = 0x800000; /* Image 1 NOR offset byte address in LE mode  */
1797      ibl.bootModes[0].u.norBoot.bootAddress[1][0]    = 0;        /* Image 0 NOR offset byte address in BE mode */
1798      ibl.bootModes[0].u.norBoot.bootAddress[1][1]    = 0x800000; /* Image 1 NOR offset byte address in BE mode  */
1799      ibl.bootModes[0].u.norBoot.interface    = ibl_PMEM_IF_SPI;
1800      ibl.bootModes[0].u.norBoot.blob[0][0].startAddress  = 0x80000000;       /* Image 0 load start address in LE mode */
1801      ibl.bootModes[0].u.norBoot.blob[0][0].sizeBytes     = 0x800000;         /* Image 0 size (10 MB) in LE mode */
1802      ibl.bootModes[0].u.norBoot.blob[0][0].branchAddress = 0x80000000;       /* Image 0 branch address after loading in LE mode
            ← */
1803      ibl.bootModes[0].u.norBoot.blob[0][1].startAddress  = 0x80000000;       /* Image 1 load start address in LE mode */
1804      ibl.bootModes[0].u.norBoot.blob[0][1].sizeBytes     = 0x800000;         /* Image 1 size (10 MB) in LE mode */
1805      ibl.bootModes[0].u.norBoot.blob[0][1].branchAddress = 0x80000000;       /* Image 1 branch address after loading in LE mode
            ← */
1806      ibl.bootModes[0].u.norBoot.blob[1][0].startAddress  = 0x80000000;       /* Image 0 load start address in BE mode */
1807      ibl.bootModes[0].u.norBoot.blob[1][0].sizeBytes     = 0x800000;         /* Image 0 size (10 MB) in BE mode */
1808      ibl.bootModes[0].u.norBoot.blob[1][0].branchAddress = 0x80000000;       /* Image 0 branch address after loading in BE mode
            ← */
1809      ibl.bootModes[0].u.norBoot.blob[1][1].startAddress  = 0x80000000;       /* Image 1 load start address in BE mode */
1810      ibl.bootModes[0].u.norBoot.blob[1][1].sizeBytes     = 0x800000;         /* Image 1 size (10 MB) in BE mode */
1811      ibl.bootModes[0].u.norBoot.blob[1][1].branchAddress = 0x80000000;       /* Image 1 branch address after loading in BE mode
            ← */
1812
1813      ibl.bootModes[1].bootMode = ibl_BOOT_MODE_NONE;
1814
1815      ibl.bootModes[2].bootMode = ibl_BOOT_MODE_TFTP;
1816      ibl.bootModes[2].priority = ibl_HIGHEST_PRIORITY+1;
1817      ibl.bootModes[2].port     = 1;//ibl_PORT_SWITCH_ALL;
1818
1819      ibl.bootModes[2].u.ethBoot.doBootp          = TRUE;
1820      ibl.bootModes[2].u.ethBoot.useBootpServerIp = TRUE;
1821      ibl.bootModes[2].u.ethBoot.useBootpFileName = TRUE;
1822      ibl.bootModes[2].u.ethBoot.bootFormat       = ibl_BOOT_FORMAT_ELF;
1823
1824
1825      SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.ipAddr,    192,168,1,3);
1826      SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.serverIp,  192,168,1,2);
1827      SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.gatewayIp, 192,168,1,1);
1828      SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.netmask,   255,255,255,0);
1829
1830      /* Use the e-fuse value */
1831      ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[0] = 0;
1832      ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[1] = 0;
1833      ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[2] = 0;
1834      ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[3] = 0;
1835      ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[4] = 0;
1836      ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[5] = 0;
1837
1838
1839      ibl.bootModes[2].u.ethBoot.ethInfo.fileName[0]  = 's';
1840      ibl.bootModes[2].u.ethBoot.ethInfo.fileName[1]  = 'v';
1841      ibl.bootModes[2].u.ethBoot.ethInfo.fileName[2]  = 'r';
1842      ibl.bootModes[2].u.ethBoot.ethInfo.fileName[3]  = '4';
1843      ibl.bootModes[2].u.ethBoot.ethInfo.fileName[4]  = '2';
1844      ibl.bootModes[2].u.ethBoot.ethInfo.fileName[5]  = '0';
1845      ibl.bootModes[2].u.ethBoot.ethInfo.fileName[6]  = '.';
1846      ibl.bootModes[2].u.ethBoot.ethInfo.fileName[7]  = 'b';
1847      ibl.bootModes[2].u.ethBoot.ethInfo.fileName[8]  = 'i';
1848      ibl.bootModes[2].u.ethBoot.ethInfo.fileName[9]  = 'n';
1849      ibl.bootModes[2].u.ethBoot.ethInfo.fileName[10] = '\0';
1850      ibl.bootModes[2].u.ethBoot.ethInfo.fileName[11] = '\0';
1851      ibl.bootModes[2].u.ethBoot.ethInfo.fileName[12] = '\0';
1852      ibl.bootModes[2].u.ethBoot.ethInfo.fileName[13] = '\0';
1853      ibl.bootModes[2].u.ethBoot.ethInfo.fileName[14] = '\0';
1854
1855      ibl.bootModes[2].u.ethBoot.blob.startAddress  = 0x80000000;       /* Load start address */
1856      ibl.bootModes[2].u.ethBoot.blob.sizeBytes     = 0x20000000;
1857      ibl.bootModes[2].u.ethBoot.blob.branchAddress = 0x80000000;       /* Branch address after loading */
1858
1859      ibl.chkSum = 0;
1860  }
```

В таблице 4-1 дано краткое описание основных конфигурационных параметров, из файла «i2cConfig.gel», их назначение и возможные значения.

Параметры, которые не описаны в таблице 4-1, связаны с аппаратной конфигурацией оборудования модуля SVR-420 и их изменение не рекомендуется.

Таблица 4-1: Основные конфигурационные параметры файла «i2cConfig.gel»

Параметр	Описание
ibl.bootModes[2].u.ethBoot.doBootp	Для режима загрузки TFTP. Если равен TRUE, IBL будет пытаться получить сетевую конфигурацию по протоколу BOOTP. Если равно FALSE, то будут использованы параметры конфигурации сети описанные ниже.
ibl.bootModes[2].u.ethBoot.bootFormat	Задает формат загружаемого образа. Может принимать одно из следующих значений: • ibl_BOOT_FORMAT_COFF — объектный формат COFF. Загружается через встроенный в IBL загрузчик COFF файлов; • ibl_BOOT_FORMAT_ELF — объектный формат ELF. Загружается через встроенный в IBL загрузчик ELF файлов; • ibl_BOOT_FORMAT_BBLOB — бинарный формат готовый к загрузке на модуле (не требующий соответствующего загрузчика); • ibl_BOOT_FORMAT_AUTO — автоматическое определение формата по сигнатуре файла; • ibl_BOOT_FORMAT_NAME — автоматическое определение формата по расширению загружаемого файла («.out» — COFF, «.elf» — ELF, «.bin» — BBLOB).
ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[0...5]	Задает значение аппаратного MAC адреса сетевого интерфейса. Если все значения равны 0, используется встроенный производителем в процессор MAC-адрес.
ibl.bootModes[2].u.ethBoot.ethInfo.fileName[0...63]	Задает имя файла для загрузки. Максимальная длина имени файла составляет 64 символа. Последним символом имени файла загрузки обязательно должен быть символ «\0».
ibl.bootModes[2].u.ethBoot.ethInfo.ipAddr	Определяет значение фиксированного IP адреса. Октеты IP адреса в файле «i2cConfig.gel» записываются через запятую. Значение параметр используется в случае, если ibl.bootModes[2].u.ethBoot.doBootp = FALSE.
ibl.bootModes[2].u.ethBoot.ethInfo.serverIp	Задает IP адрес сервера загрузки в случае, если ibl.bootModes[2].u.ethBoot.doBootp = FALSE.
ibl.bootModes[2].u.ethBoot.ethInfo.gatewayIp	Задает IP адрес основного шлюза в случае, если ibl.bootModes[2].u.ethBoot.doBootp = FALSE.
ibl.bootModes[2].u.ethBoot.ethInfo.netmask	Задает маску подсети в том случае, если ibl.bootModes[2].u.ethBoot.doBootp = FALSE.
ibl.bootModes[2].u.ethBoot.useBootpServerIp	Если равен FALSE, то в качестве IP адреса сервера загрузки будет использовано значение параметра ibl.bootModes[2].u.ethBoot.ethInfo.serverIp. Если равен TRUE, то будет использован IP адрес сервера загрузки, указанный в BOOTP ответе от BOOTP сервера. Данный параметр имеет значение только в том случае, когда ibl.bootModes[2].u.ethBoot.doBootp = TRUE.
ibl.bootModes[2].u.ethBoot.useBootpFileName	Если равен FALSE, то в качестве имени файла для загрузки будет использовано значение параметра ibl.bootModes[2].u.ethBoot.ethInfo.fileName[0...63]. Если равен TRUE, то будет использовано имя файла загрузки, указанное в BOOTP ответе от BOOTP сервера. Данный параметр имеет значение только в том случае, когда ibl.bootModes[2].u.ethBoot.doBootp = TRUE.
ibl.bootModes[2].u.ethBoot.blob.startAddress	Адрес области памяти куда будет осуществляться загрузка образа.
ibl.bootModes[2].u.ethBoot.blob.sizeBytes	Максимальный допустимый объем образа, который допускается загрузить.
ibl.bootModes[2].u.ethBoot.blob.branchAddress	Адрес точки входа, куда будет осуществлен переход после завершения загрузки образа приложения.

5 Импорт и запуск целевой конфигурации модуля

Для загрузки кода приложениий на модуль SVR-420, в первую очередь, необходимо запустить целевую конфигурацию модуля SVR-420. В папке «TargetConfigurations» сопроводительного диска к модулю SVR-420находятся файлы целевых конфигураций для различных отладчиков. В данном документе рассматривается загрузка кода через отладчик Spectrum Digital XDS560v2 STM LAN. Данному отладчику соответствует файл целевой конфигурации «SVR-420-LAN560v2-SD.ccxml», который необходимо импортировать в рабочее пространство.

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

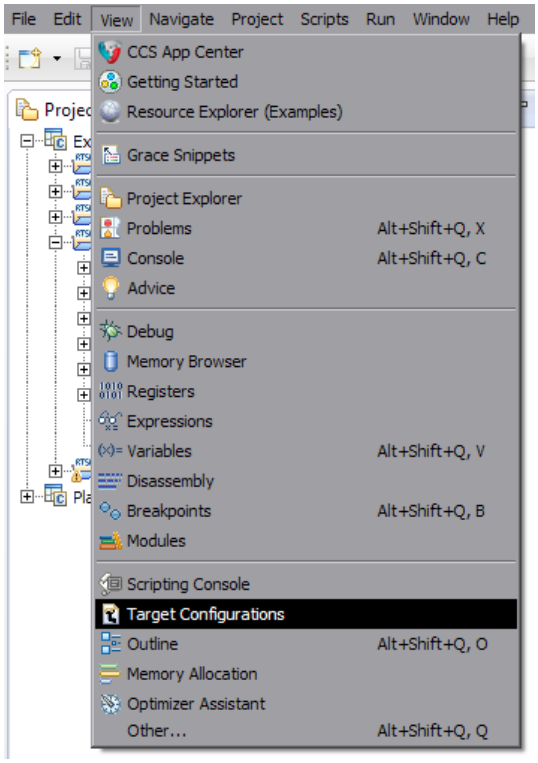


Рисунок 5-1: Пункт меню для отображения окна целевых конфигураций

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

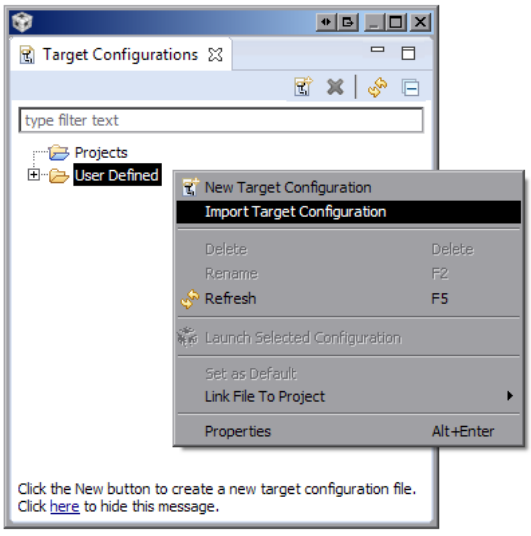


Рисунок 5-2: Меню импорта целевой конфигурации

В появившемся окне выбора файла (см. рисунок 5-3) необходимо выбрать файл «SVR-420-LAN560v2-SD.ccxml» и нажать на кнопку «Открыть». В данном документе предполагается, что папка «TargetConfigurations» с сопрово-

дительного диска к модулю SVR-420, где расположен файл «SVR-420-LAN560v2-SD.ccxml», скопирована в папку «D:/Dev/Modules/SVR-420/TargetConfigurations».

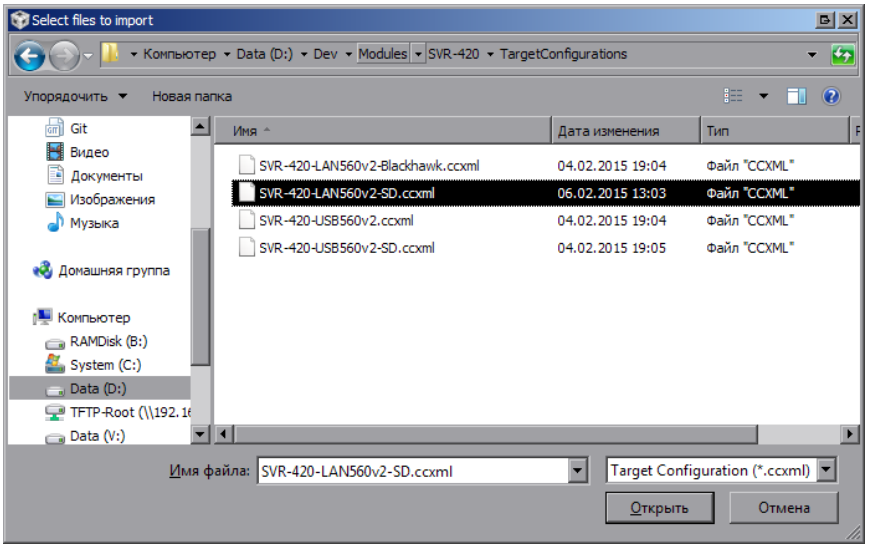


Рисунок 5-3: Окно выбора файла для импорта целевой конфигурации

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 «OK».

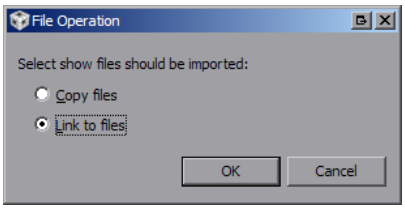


Рисунок 5-4: Окно выбора способа импорта файла целевой конфигурации

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

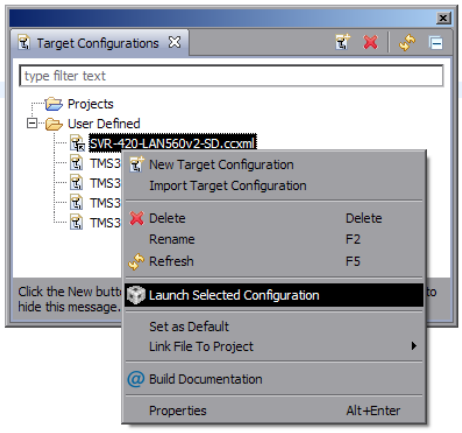


Рисунок 5-5: Запуск целевой конфигурации

После запуска целевой конфигурации модуля SVR-420, среда разработки CCS перейдет в режим отладки и в окне «Debug» будет выведен список ядер обоих процессоров модуля SVR-420, как показано на рисунке 5-6.

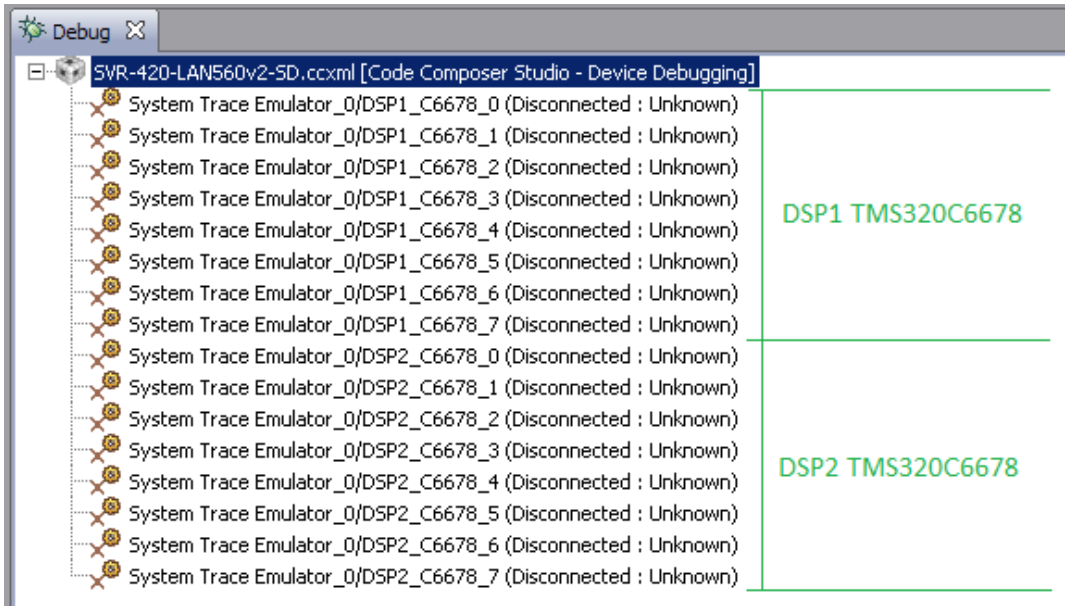


Рисунок 5-6: Список ядер процессоров модуля SVR-420

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 Б даны инструкции по настройке раздельного вывода (CIO) каждого ядра процессора в отдельное окно «Console».

6 Подготовка образов приложений для загрузки на нескольких ядрах процессора

Для подготовки образов приложений, которые необходимо загрузить сразу на несколько ядер процессора, используется набор специальных утилит MAD разработанных компанией Texas Instruments.

Набор утилит MAD входит в состав MCSDK (MultiCore Software Development Kit). Более подробную информацию по возможностям и приемам использования набора утилит MAD можно получить ознакомившись с документом [2].

6.1 Компоненты MAD Utils

MAD Utils предоставляет набор утилит для достижения следующих целей:

- необходимость выполнения загрузки множества приложений на несколько ядер;
- необходимость сохранения памяти, путем выделения общего кода многоядерных приложений.

MAD Utils состоит из пяти основных утилит, которые можно разделить на две категории: «Утилиты времени сборки» и «Утилиты времени исполнения».

Утилиты времени сборки (build time utilities) включают в себя:

- Static linker — статический линковщик, предназначенный для линковки приложений и зависимых динамических общих объектов DSO (Dynamic Shared Object).
- Prelink Tool — используется для назначения сегментам ELF файла виртуальных адресов.
- MAP Tool — используется для назначения виртуальных адресов сегментам многоядерных приложений. Пользователь определяет нужные разделы памяти для устройства и высокоуровневые инструкции сегмена размещения в MAP Tool. Основываясь на данной информации, MAP Tool определяет виртуальные и физические адреса времени исполнения для каждого сегмента ELF файла для каждого приложения. Затем вызывается Prelink Tool для выделения области для хранения всех приложений и их DSO. MAP Tool также генерирует набор активационных записей для загрузки приложения на определенное ядро. Активационные записи — это инструкции загрузчика времени исполнения, которые выполняют следующие действия:
  - настройка карты виртуальной памяти и аттрибутов доступа и защиты областей памяти;
  - копирование и инициализация загружаемых сегментов памяти на их адреса времени исполнения.

  Полученный образ приложения и активационные записи запаковываются в образ ROMFS, который предназначен для загрузки на целевом устройстве.

Утилиты времени исполнения (run time utilities) включают в себя:

- Утилита начальной загрузки. В качестве данной утилиты выступает загрузчик IBL, который предоставляет функциональность загрузки образа ROMFS в общую внешнюю память устройства (DDR).
- MAD Loader — утилита загрузки времени исполнения, которая обеспечивает функционал запуска приложений на заданном ядре. Данная утилиты выполняет следующие действия для обеспечения запуска приложения на ядре:
  - конфигурация карты виртуальной памяти для ядра;
  - конфигурация атрибутов и режимов доступа для каждого раздела памяти;
  - копирование сегментов памяти с локального адреса в адрес времени исполнения;
  - выполнение преинициализационных функций приложения;
  - выполнение инициализационных функций зависимых библиотек и приложений;
  - запуск приложения (переход по адресу точки входа).

6.2 Режимы работы MAD Utils

MAD Utils позволяет работать в двух режимах:

- Prelinker bypass mode. В данном режиме утилита MAP Tool не выполняет назначения адресов сегментам приложения и Prelink Tool не вызывается. Данный режим подходи в тех случаях, когда просто требуется выполнить загрузку приложения или нескольких приложений на конкретном ядре или ядрах.
- Prelinker mode. В данном режиме утилита MAP Tool выполняет назначение адресов сегментам приложения и вызывает Prelink Tool. Данный режим походит в тех случаях, когда разработчику требуется, чтобы MAP Tool выполнил присвоение адресов для общего кода между несколькими ядрами, на которых должно работать приложение.

Внимание

В данном документе рассматривается только работа MAD Utils в режиме работы «Prelinker bypass mode». Информацию по работе с MAD Utils в режиме «Prelinker mode» можно получить обратившись к документу [2].

6.3 Работа с MAD Utils в режиме «Prelinker bypass mode»

На рисунке 6-1 изображена схема работы MAD Utils в режиме «Prelinker bypass mode».

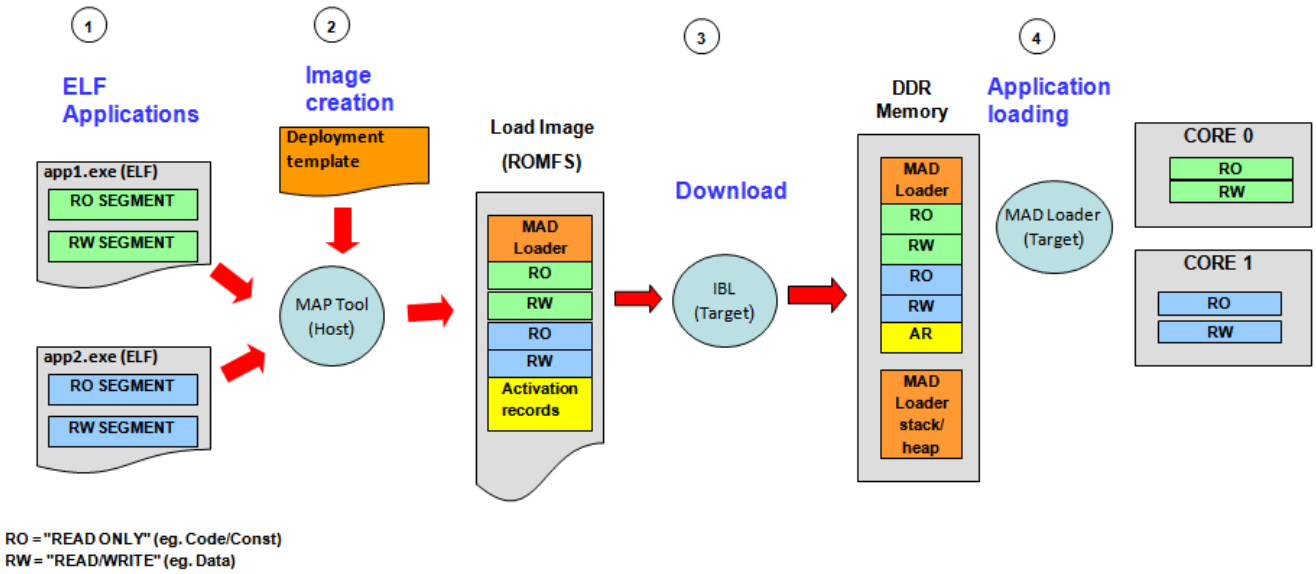


Рисунок 6-1: Схема работы MAD в режиме «Prelinker bypass mode»

Работа с MAD Utils может быть разделена на два этапа — этап подготовки образа для загрузки и этап загрузки образа на целевом устройстве.

Этап подготовки образа состоит из следующих шагов:

- Сборка приложения (статическая линковка по адресам исполнения);
- Создание файла конфигурации развертывания для MAP Tool с определением приложения для каждого ядра.
- Запуск MAP Tool с файлом конфигурации развертывания в качестве входных данных.
- MAP Tool создает образ для загрузки (в формате ROMFS), содержащий в себе активационные записи для каждого приложения.

Этап загрузки образа на целевом устройстве выглядит следующим образом:

- При загрузке, в первую очередь, на устройстве выполняется код загрузчика из ROM памяти. Этот загрузчик выполняет загрузку IBL, который должен быть записан в I^2C EEPROM память устройства.
- IBL выполняет загрузку MAD образа с TFTP сервера или NOR флеш памяти в DDR память. При этом, IBL должен быть сконфигурирован таким образом, что бы переход осуществлялся по адресу точки входа MAD Loader.

- MAD Loader выполняет разбор образа ROMFS и выполняет загрузку сегментов приложения по их адресам времени исполнения для каждого ядра.
- MAD Loader выполняет переход по адресу точки входа для каждого ядра, запуская таким образом выполнения приложения на ядрах целевого устройства.

6.4 Конфигурация MAP Tool

Входными данными для MAP Tool является конфигурационный файл в формате JSON. Данный конфигурационный файл должен содержать параметры, приведенные в таблице 6-1.

Таблица 6-1: Параметры конфигурационного файла MAP Tool

Параметр	Описание
`deploymentCfgFile`	Путь к конфигурационному файлу развертывания.
`LoadImageName`	Имя файла образа, который будет сгенерирован. Данный файл образа (в формате ROMFS) будет помещен в папку «images».
`prelinkExe`	Путь к запускаемому файлу Prelink Tool. Данная утилита является частью CGT (Code Generation Tools).
`stripExe`	Путь к запускаемому файлу Strip Tool. Данная утилита является частью CGT.
`ofdTool`	Путь к запускаемому файлу OFD Tool. Данная утилита является частью CGT.
`malApp`	Путь к файлу приложения MAD Loader.
`nmlLoader`	Путь к файлу приложения NML Loader. NML Loader является составной частью MAD Loader.

В листинге 6-1 приведен пример конфигурационного файла для MAP Tool.

Листинг 6-1: Пример конфигурационного файла для MAP Tool

```
{
    "deploymentCfgFile" : ".\\deploy.json",
    "LoadImageName"     : "mad_test.bin",
    "prelinkExe"        : "C:\\ti\\ccsv5\\tools\\compiler\\c6000_7.4.2\\bin\\prelink6x",
    "stripExe"          : "C:\\ti\\ccsv5\\tools\\compiler\\c6000_7.4.2\\bin\\strip6x",
    "ofdTool"           : "C:\\ti\\ccsv5\\tools\\compiler\\c6000_7.4.2\\bin\\ofd6x.exe",
    "malApp"            : "C:\\ti\\mcsdk_2_01_02_06\\tools\\boot_loader\\mad-utils\\mad-loader\\bin\\C6670\\le\\mal_app.exe",
    "nmlLoader"         : "C:\\ti\\mcsdk_2_01_02_06\\tools\\boot_loader\\mad-utils\\mad-loader\\bin\\C6670\\le\\nml.exe"
}
```

В листинге 6-1 указаны пути с учетом следующих предположений:

- используется MCSDK версии 2.01.02.06 установленная в папку «C:/ti/mcsdk_2_01_02_06»;
- используется компилятор версии 7.4.2 установленный в папку «C:/ti/ccsv5/tools/compiler/c6000_7.4.2»;
- файл конфигурации развертывания «deploy.json» находится в той же папке.

6.5 Конфигурационный файл развертывания

В режиме «Prelinker bypass mode» конфигурационный файл развертывания определяет следующую информацию:

- области памяти для загрузки;
- данные приложений для развертывания.

Файл конфигурации развертывания является файлом в формате JSON. Для режима «Prelinker bypass mode» файл должен содержать параметры, приведенные в таблице 6-2.

Таблица 6-2: Параметры файла конфигурации развертывания

Параметр	Описание
`deviceName`	JSON объект, определяющий целевое устройство. Может принимать следующие значения: • `C6657` — 2-х ядерный процессор TMS320C6657; • `C6670` — 4-х ядерный процессор TMS320C6670; • `C6678` — 8-и ядерный процессор TMS320C6678;

Продолжение таблицы на следующей странице

Продолжение таблицы 6-2

Параметр	Описание
`partitions`	Данный параметр идентифицирует разделы памяти, которые будут загружены из ROMFS образа в память устройства при загрузке. Определение каждого из разделов должно содержать следующие параметры: • `name` — имя раздела. Используется для идентификации раздела памяти в отладочном выводе MAP Tool; • `vaddr` — виртуальный адрес раздела памяти; • `size` — размер области памяти в байтах; • `loadPartition` — определяет будет ли данный раздел загружен или нет.
`applications`	Данный параметр идентифицирует приложения для развертывания. Определение каждого приложения должно содержать следующие параметры: • `name` — имя приложения. Используется для идентификации в отладочном выводе; • `fileName` — имя файла образа приложения полученного в результате сборки приложения; • `allowedCores` — номера ядер, на которых разрешен запуск данного приложения.
`appDeployment`	Данный параметр определяет приложения для каждого из ядер целевого устройства. Представляет из себя массив (размер массива должен соответствовать количеству ядер целевого устройства), каждый элемент которого должен содержать имя приложения, которое необходимо загрузить на соответствующем ядре целевого устройства. В том случае, если на каком либо ядре загружать приложение не требуется, соответствующий элемент данного массива задается в виде пустой строки (”“).

Листинг 6-2: Пример файла конфигурации развертывания

```
{
    "deviceName" : "C6670",
    "partitions" : [
        {
            "name"  : "load-partition",
            "vaddr" : "0x9e000000",
            "size"  : "0x2000000",
            "loadPartition" : true
        }
    ],
    "applications" : [
        {
            "name"          : "app1",
            "fileName"      : "../build/app_1.out",
            "allowedCores"  : [0,1,2,3]
        },
        {
            "name"          : "app2",
            "fileName"      : "../build/app_2.out",
            "allowedCores"  : [0,1,2,3]
        }
    ],
    "appDeployment" : [
        "app1",
        "app1",
        "",
        "app2"
    ]
}
```

6.6 Запуск MAP Tool

Для запуска MAP Tool потребуется установленный Python интерпретатор, который можно бесплатно скачать с официального сайта[1].

Запуск MAP Tool в режиме «Prelinker bypass mode» осуществляется путем выполнения команды:

```
python maptool.py <файл_конфигурации> bypass-prelink
```

Внимание

Следует помнить, что образ, предназначенный для загрузки на целевом устройсве, будет создан в папке «images». Имя файла образа задается в конфигурационном файле MAP Tool (см. раздел 6.4).

В качестве параметра <файл_конфигурации> необходимо указать имя файла конфигурации MAP Tool (см. раздел 6.4). Python скрипт «maptool.py» располагается в папке «tools/boot_loader/mad-utils/map-tool» относительно папки установки MCSDK («C:/ti/mcsdk_2_01_02_06»). Для запуска MAP Tool непосредственно из папки установки MCSDK необходимо выполнять команду указав полный путь к файлу «maptool.py»:

```
python "C:\ti\mcsdk_2_01_02_06\tools\boot_loader\mad-utils\map-tool\maptool.py" <файл_конфигурации>
    ← bypass-prelink
```

Примечание

Инструкции по записи полученного образа в NOR флеш память приведены в разделе 3.3 данного документа.

Инструкции по установке и настройке служб BOOTP и TFTP, необходимых для осуществления загрузки образа по Ethernet с TFTP сервера, можно найти в документе [3].

6.7 Загрузка образа

Для того, чтобы загрузчик IBL мог корректно загружать образы, подготовленные при помощи MAD Utils, необходимо выполнить его конфигурацию. Подробное описание процедуры конфигурации загрузчика IBL приведено в разделе 4 данного документа.

В том случае, если подготовленный образ планируется загружать по Ethernet с TFTP сервера, то необходимо установить следующие параметры:

```
ibl.bootModes[2].u.ethBoot.bootFormat         = ibl_BOOT_FORMAT_BBLOB;
ibl.bootModes[2].u.ethBoot.blob.startAddress  = 0x9e000000;
ibl.bootModes[2].u.ethBoot.blob.sizeBytes     = 0x20000000;
ibl.bootModes[2].u.ethBoot.blob.branchAddress = 0x9e001040;
```

Если же загрузка подготовленного образа будет происходить с NOR флеш памяти, то необходимо установить параметры:

```
ibl.bootModes[0].u.norBoot.bootFormat               = ibl_BOOT_FORMAT_BBLOB;
ibl.bootModes[0].u.norBoot.blob[0][0].startAddress  = 0x9e000000;
ibl.bootModes[0].u.norBoot.blob[0][0].sizeBytes     = 0x800000;
ibl.bootModes[0].u.norBoot.blob[0][0].branchAddress = 0x9e001040;
```

Дополнительное описание конфигурационных параметров приведено в таблице 4-1 раздела 4 данного документа.

[1] https://www.python.org/download/

Приложение А Примеры работы скрипта записи EEPROM и NOR флеш памяти

В листинге А-1 приведен пример вывода в терминал запуска скрипта «svr420_program.bat» для записи в NOR флеш память образов по умолчанию на все четыре процессора модуля SVR-420.

Примечание

В целях сокращения объема листингов, приведенных в данном приложении, некоторые их части вырезаны. В местах вырезанных данных листингов, помещен текст в виде «ВЫРЕЗАНО: Описание вырезанного текста».

Листинг А-1: Вывод в терминал при запуске команды «svr420_program.bat NOR all»

```
D:\Dev\Modules\SVR-420\Program>svr420_program.bat NOR all

SVR-420 program script started
Copyright (c) 2015, Scan Engineering Telecom SPb

Writing to DSP1 ENABLED
Writing to DSP2 ENABLED

Target configuration file:
  "../TargetConfigurations/SVR-420-LAN560v2-SD.ccxml"

Images for DSP's:
  DSP1: "bin/nor_c6678.bin" (564107 bytes)
  DSP2: "bin/nor_c6678.bin" (564107 bytes)

Configuring debug server for SVR-420 board...
Done!
Opening a debug session for 2 DSP(s)...
Loaded FPGA Image: C:\ti\ccsv5\ccs_base\common\uscif\dtc_top.jbc
Done!
Connecting to DSP(s)...

ВЫРЕЗАНО: Вырезана часть с GEL инициализацией ядер процессоров

Done!
Loading binary images to DSP(s) memory...
  Loading file "bin/nor_c6678.bin" (564107 bytes)...
  Loading file "bin/nor_c6678.bin" (564107 bytes)...
Done!
Configuring writers on DSP(s)...
Done!
Executing writers on DSP(s)...
NOR Writer Utility Version 02.00.00.00
Copyright (c) 2014, Scan Engineering Telecom SPb

Write size:    564107 bytes
Start address: 0x0

Flashing sector 0 (0 bytes of 564107)
NOR Writer Utility Version 02.00.00.00
Copyright (c) 2014, Scan Engineering Telecom SPb

Write size:    564107 bytes
Start address: 0x0

Flashing sector 0 (0 bytes of 564107)
Flashing sector 1 (65536 bytes of 564107)
Flashing sector 1 (65536 bytes of 564107)
Flashing sector 2 (131072 bytes of 564107)
Flashing sector 2 (131072 bytes of 564107)
Flashing sector 3 (196608 bytes of 564107)
Flashing sector 3 (196608 bytes of 564107)
Flashing sector 4 (262144 bytes of 564107)
Flashing sector 4 (262144 bytes of 564107)
Flashing sector 5 (327680 bytes of 564107)
Flashing sector 5 (327680 bytes of 564107)
Flashing sector 6 (393216 bytes of 564107)
Flashing sector 6 (393216 bytes of 564107)
Flashing sector 7 (458752 bytes of 564107)
Flashing sector 7 (458752 bytes of 564107)
Flashing sector 8 (524288 bytes of 564107)
Flashing sector 8 (524288 bytes of 564107)
```

```
Reading and verifying sector 0 (0 bytes of 564107)
Reading and verifying sector 1 (65536 bytes of 564107)
Reading and verifying sector 2 (131072 bytes of 564107)
Reading and verifying sector 3 (196608 bytes of 564107)
Reading and verifying sector 0 (0 bytes of 564107)
Reading and verifying sector 4 (262144 bytes of 564107)
Reading and verifying sector 1 (65536 bytes of 564107)
Reading and verifying sector 5 (327680 bytes of 564107)
Reading and verifying sector 2 (131072 bytes of 564107)
Reading and verifying sector 6 (393216 bytes of 564107)
Reading and verifying sector 3 (196608 bytes of 564107)
Reading and verifying sector 7 (458752 bytes of 564107)
Reading and verifying sector 4 (262144 bytes of 564107)
Reading and verifying sector 8 (524288 bytes of 564107)
Reading and verifying sector 5 (327680 bytes of 564107)
NOR programming completed successfully
Reading and verifying sector 6 (393216 bytes of 564107)
Reading and verifying sector 7 (458752 bytes of 564107)
Reading and verifying sector 8 (524288 bytes of 564107)
NOR programming completed successfully
Done!
Terminating debug sessions on DSP(s)...
Done!
Stopping debug server...
Done!


D:\Dev\Modules\SVR-420\Program>
```

В листинге A-2 приведен пример вывода в терминал запуска скрипта «svr420_program.bat» для записи в EEPROM память образов по умолчанию на все четыре процессора модуля SVR-420.

Листинг A-2: Вывод в терминал при запуске команды «svr420_program.bat EEPROM all»

```

D:\Dev\Modules\SVR-420\Program>svr420_program.bat EEPROM all

SVR-420 program script started
Copyright (c) 2015, Scan Engineering Telecom SPb

Writing to DSP1 ENABLED
Writing to DSP2 ENABLED

Target configuration file:
  "../TargetConfigurations/SVR-420-LAN560v2-SD.ccxml"

Images for DSP's:
  DSP1: "bin/eeprom_0x50_c6678.bin" (57468 bytes)
  DSP2: "bin/eeprom_0x50_c6678.bin" (57468 bytes)

Configuring debug server for SVR-420 board...
Done!
Opening a debug session for 2 DSP(s)...
Loaded FPGA Image: C:\ti\ccsv5\ccs_base\common\uscif\dtc_top.jbc
Done!
Connecting to DSP(s)...

ВЫРЕЗАНО: Вырезана часть с GEL инициализацией ядер процессоров

Loading binary images to DSP(s) memory...
  Loading file "bin/eeprom_0x50_c6678.bin" (57468 bytes)...
  Loading file "bin/eeprom_0x50_c6678.bin" (57468 bytes)...
Done!
Configuring writers on DSP(s)...
Done!
Executing writers on DSP(s)...
EEPROM Writer Utility Version 02.00.00.00
Copyright (c) 2014, Scan Engineering Telecom SPb

Write size:        57468 bytes
I2C bus address:   0x50
I2C start address: 0x0
Swap data:         no

Writing 57468 bytes from DSP memory address 0x0c000100 to EEPROM (0x0050)...
Reading 57468 bytes from EEPROM (0x0050) to DSP memory address 0x0c010100...
Verifying data read...
EEPROM programming completed successfully
EEPROM Writer Utility Version 02.00.00.00
Copyright (c) 2014, Scan Engineering Telecom SPb

Write size:        57468 bytes
I2C bus address:   0x50
I2C start address: 0x0
Swap data:         no

Writing 57468 bytes from DSP memory address 0x0c000100 to EEPROM (0x0050)...
Reading 57468 bytes from EEPROM (0x0050) to DSP memory address 0x0c010100...
Verifying data read...
EEPROM programming completed successfully
Done!
Terminating debug sessions on DSP(s)...
Done!
Stopping debug server...
Done!


D:\Dev\Modules\SVR-420\Program>
```

Приложение Б 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 Б-1).

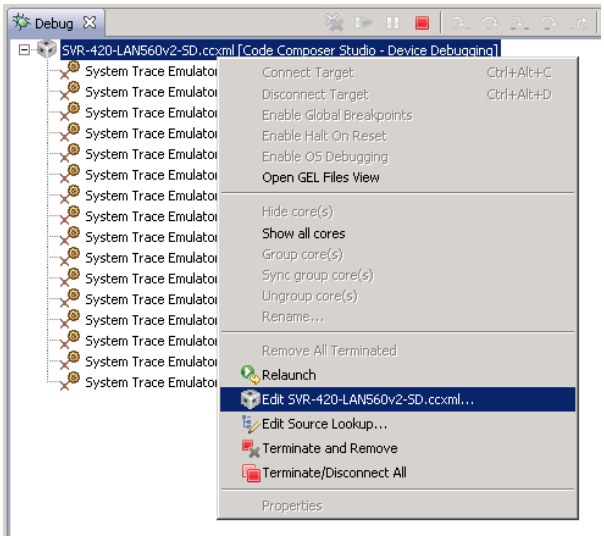


Рисунок Б-1: Контекстное меню целевой конфигурации

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

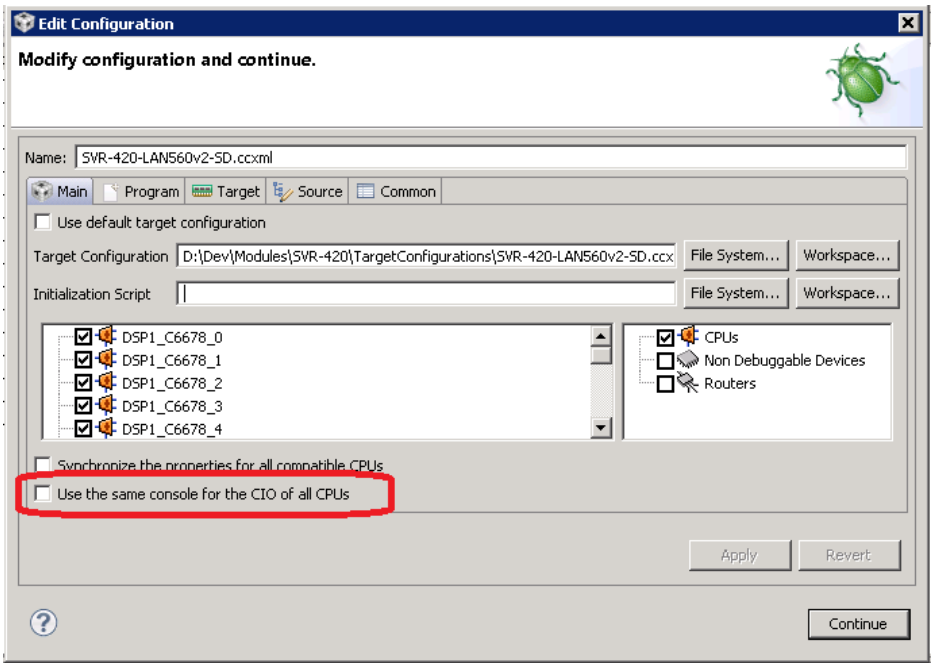


Рисунок Б-2: Окно настроек целевой конфигурации

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

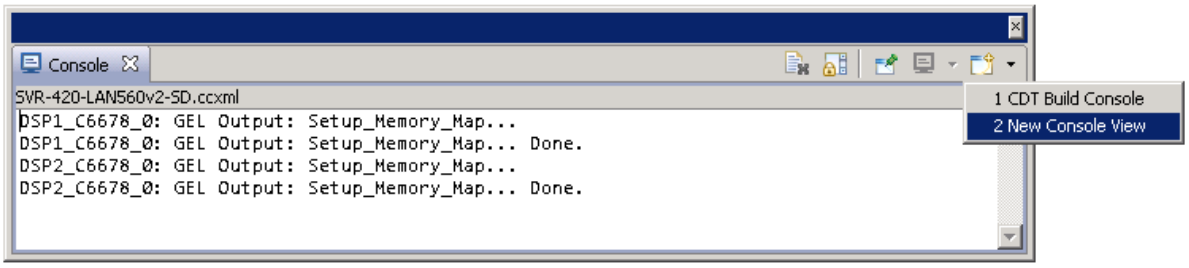


Рисунок Б-3: Открытие второго окна «Console»

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

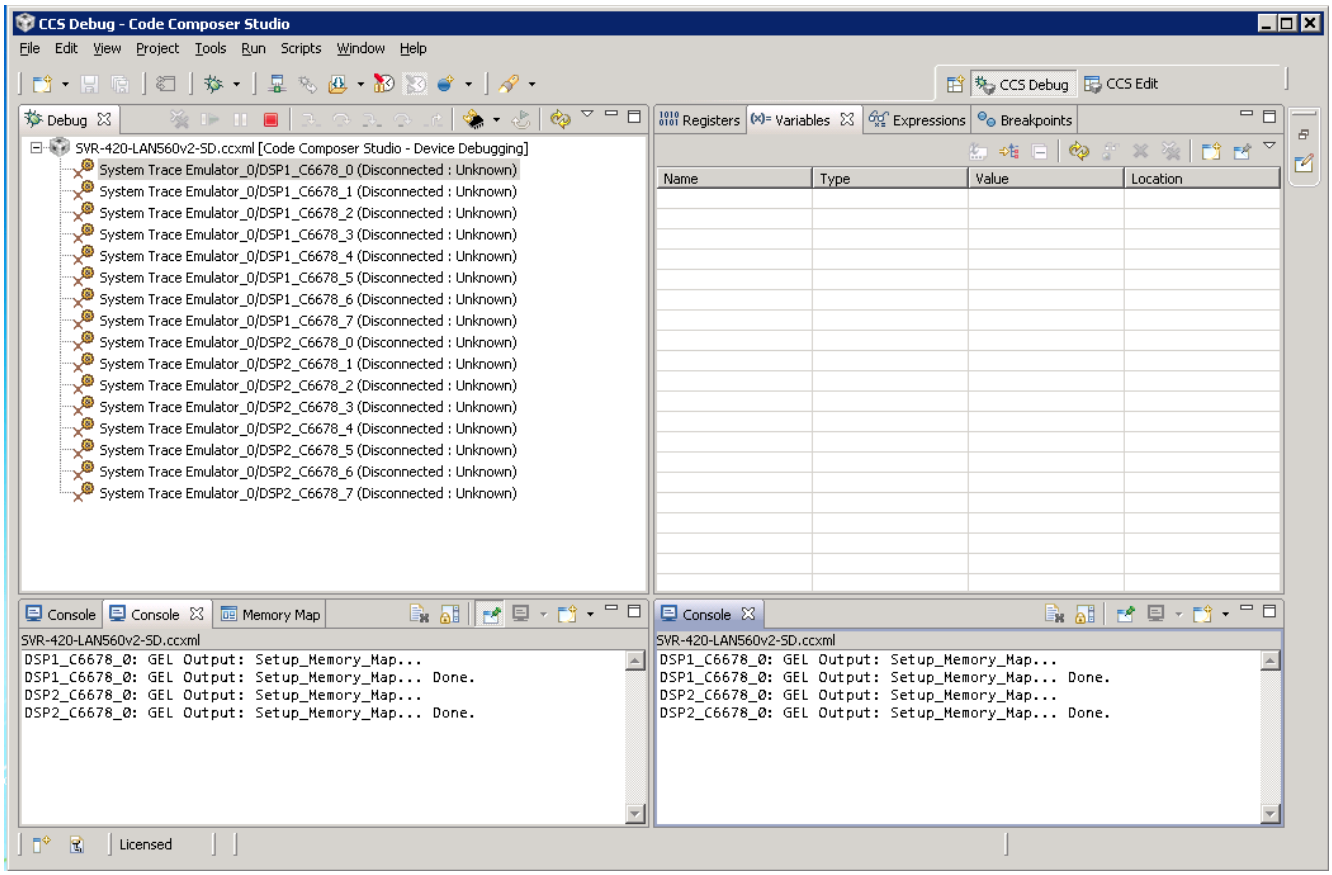


Рисунок Б-4: Два окна «Console»

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 Б-5).

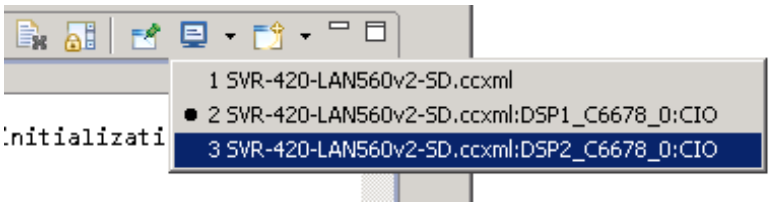


Рисунок Б-5: Выбор ядра для отображения вывода в окне «Console»

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

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

Приложение B Выбор режима загрузки IBL

Выбор режима загрузки IBL осуществляется при помощи переключателей X8.1–4 на плате модуля SVR-420.

В таблице B-1 представлены положения переключателей X8.1–4 для всех возможных режимов загрузки IBL для каждого из двух процессоров модуля SVR-420.

Таблица B-1: Положение переключателей модуля SVR-420 для установки режимов загрузки IBL

Режим	DSP1_C6678		DSP2_C6678	
	X8.1	X8.2	X8.3	X8.4
NOBOOT (DEBUG)	Разомкнут	Разомкнут	Разомкнут	Разомкнут
NOR	Разомкнут	**Замкнут**	Разомкнут	**Замкнут**
TFTP (BOOTP)	**Замкнут**	Разомкнут	**Замкнут**	Разомкнут

Для режима NOBOOT устанавливаются значения `BOOTMODE[7:0] = 00000000b` для соответствующего процессора. Для режима NOR устанавливаются значения `BOOTMODE[7:0] = 00000101b` для соответствующего процессора. Для режима TFTP устанавливаются значения `BOOTMODE[7:0] = 00100101b` для соответствующего процессора. для соответствующего процессора.

Примечание

`BOOTMODE[7:0]` является частью регистра `DEVSTAT[8:1]`, а `PCIESSMODE[1:0]` является частью регистра `DEVSTAT[15:14]`. В процессоре TMS320C6678 32-х битный регистр `DEVSTAT` доступен по адресу 0x02620020.

Список литературы

1. SVR-420. Сборка и запуск приложения веб-сервера. Руководство пользователя. UG-SVR-420-WEB (цит. на с. 14, 18).
2. Multicore Application Deployment (MAD) Utilities. User's Guide.
URL: http://processors.wiki.ti.com/index.php/MAD_Utils_User_Guide (цит. на с. 35, 36).
3. Установка и настройка сервера сетевой загрузки (BOOTP и TFTP). Руководство пользователя.
UG-CMN-BOOTP-TFTP (цит. на с. 39).