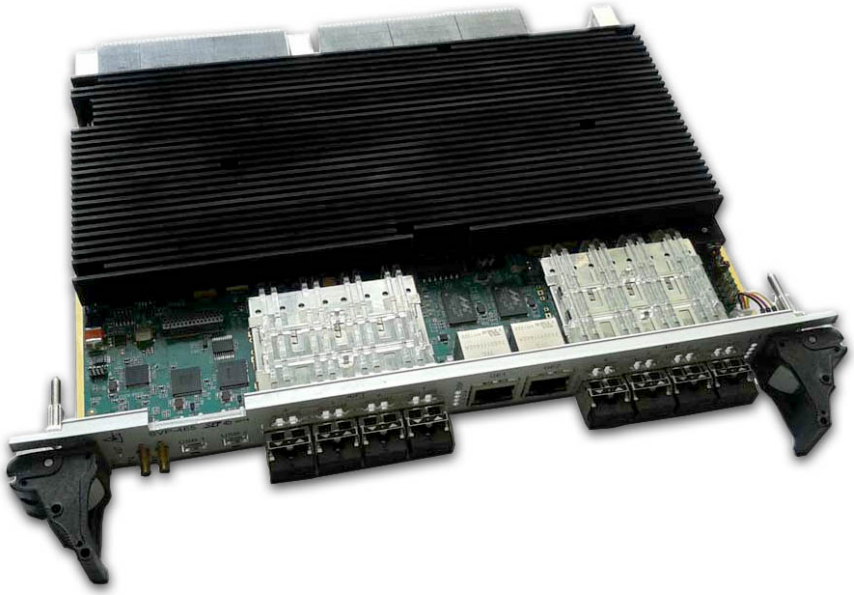
Scan Engineering Telecom SPb

SVP-465. Загрузчик IBL

Руководство пользователя

Версия 1.2



Код документа: UG-SVP-465-IBL
Дата сборки: 27 мая 2015 г.
Листов в документе: 51

http://www.setdsp.ru

История ревизий

Ревизия	Дата	Изменения
1.2	18.11.2014	В разделы 3.3 и 3.4 добавлена информация об обязательном расширении «.bin» для записываемых файлов образов.
1.1	11.07.2014	Добавлен раздел 6 по подготовке образов для загрузки приложений на несколько ядер процессора при помощи MAD Utils. Добавлено описание дополнительных параметров в таблице 4-1.
1.0	23.06.2014	Начальная версия

Содержание

Список рисунков

Список таблиц

Список листингов

Список процедур

Перечень сокращений и условных обозначений

BOOTP	Bootstrap Protocol	31, 32, 40
CCS	Code Composer Studio	7, 8, 12, 13, 15, 18, 35, 47, 48
CGT	Code Generation Tools	38
CIO	Console Input/Output	3, 15, 17, 35, 47
COFF	Common Object File Format	31
DDR	Double Data Rate	14, 18, 36, 38
DSO	Dynamic Shared Object	36
DSS	Debug Server Scripting	12, 13
EEPROM	Electrically Erasable Programmable Read-Only Memory	3–5, 7, 8, 12–17, 19, 20, 23–26, 28, 37, 42, 45
ELF	Executable and Linkable Format	31, 36
GEL	General Extension Language	4, 12, 27, 29
I^2C	Inter-Integrated Circuit	8, 17, 20, 23, 37
IBL	Intermediate Boot Loader	3–5, 7, 8, 10–12, 14, 18–20, 23–26, 29, 31, 36–38, 40, 50
IP	Internet Protocol	31
JSON	Java Script Object Notation	38, 39
MAC	Media Access Control	31
MAD	Multicore Application Deployment	2–4, 36–38, 40
MAP	Multiple Application Pre-linker	3, 36–41
MCSDK	MultiCore Software Development Kit	36, 38, 40
NML	No Man’s Land	38
NOR	Not OR	3–5, 7, 13–16, 18–20, 24, 38, 40–42
OFD	Object File Dump	38
ROMFS	Read-Only Memory File System	36–39
ROM	Read-Only Memory	37
SRIO	Serial RapidIO	41
TFTP	Trivial File Transfer Protocol	7, 38, 40, 41
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 SVP-465 с NOR флеш памяти или по Ethernet с TFTP сервера в локальной сети. Конкретный режим загрузки выбирается при помощи переключателей на плате модуля SVP-465 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 SVP-465 (раздел 2), описана процедура записи загрузчика IBL в EEPROM модуля SVP-465 (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 SVP-465 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 SVP-465 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 SVP-465, 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 SVP-465.

Внимание

Перед выполнением сборки загрузчика, описанной в данном разделе, перепишите с сопроводительного диска к модулю SVP-465 папку «ibl» на жесткий диск компьютера. Далее, предполагается, что все содержимое папки «ibl» с сопроводительного диска переписано в папку «D:/Dev/Modules/SVP-465/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 SVP-465 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 SVP-465 (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).

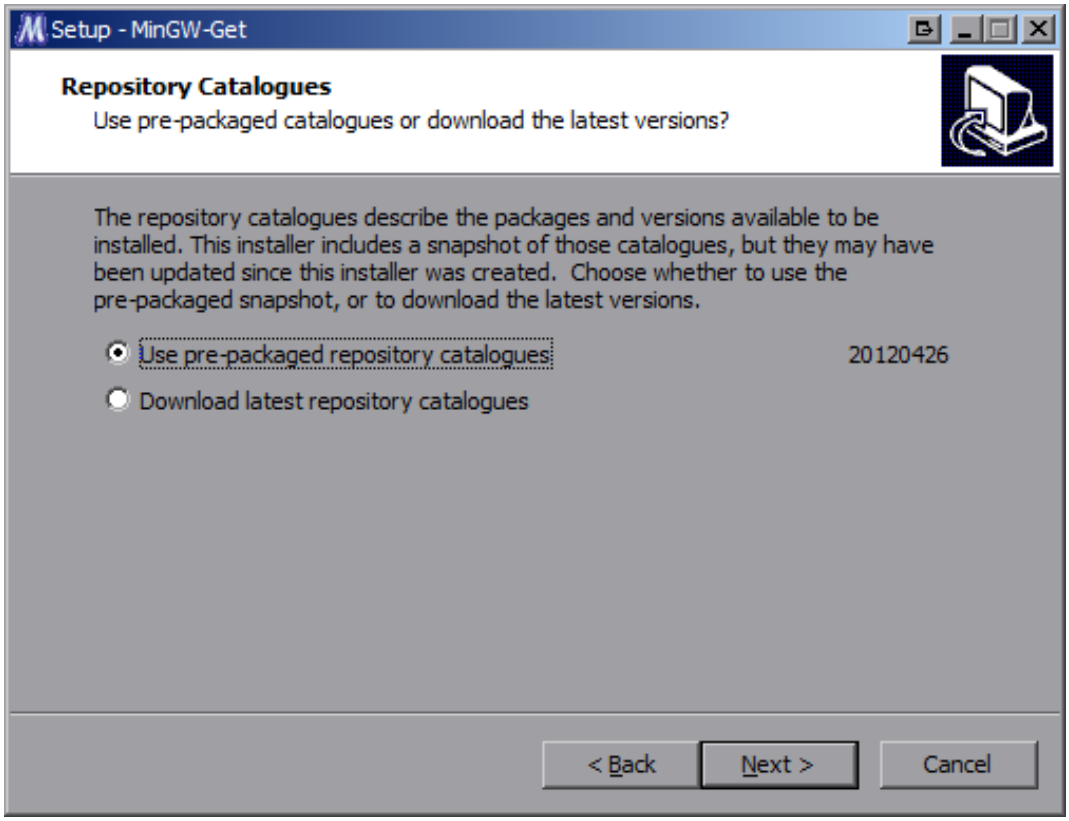


Рисунок 2-1: Установка MinGW. Выбор каталога репозитариев

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

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».

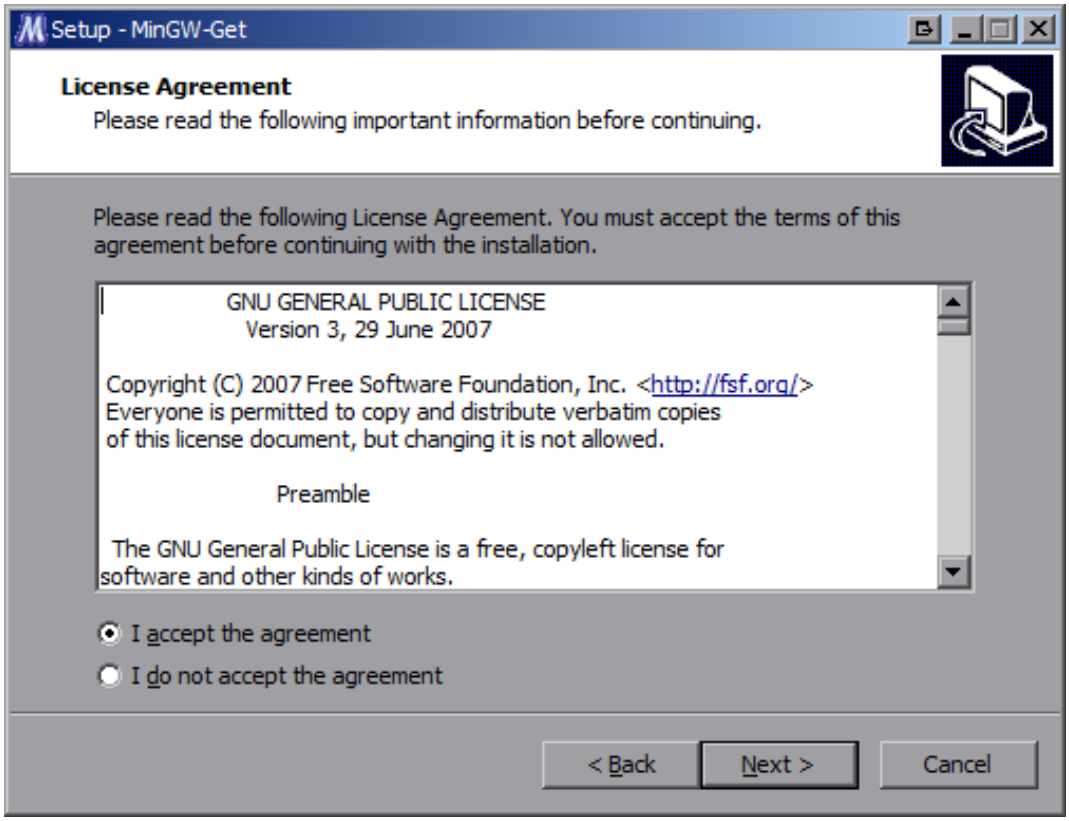


Рисунок 2-2: Установка MinGW. Окно согласия с лицензией

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

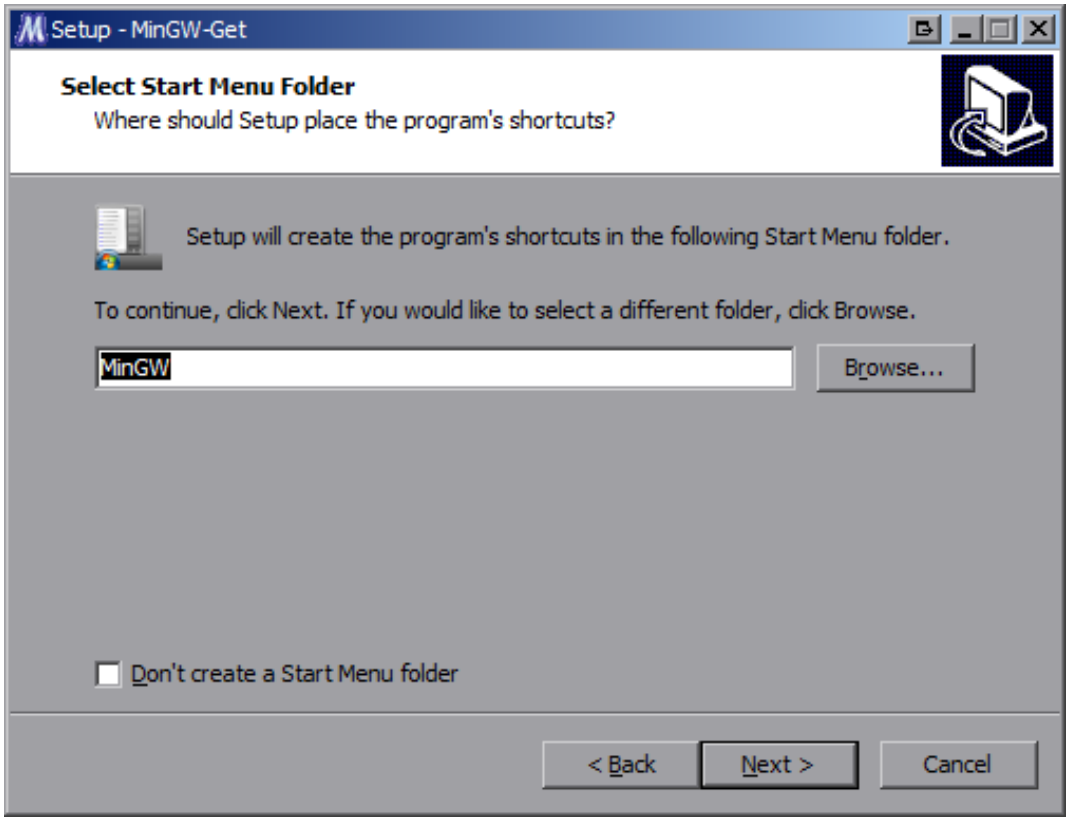


Рисунок 2-3: Установка MinGW. Выбор пути установки

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

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

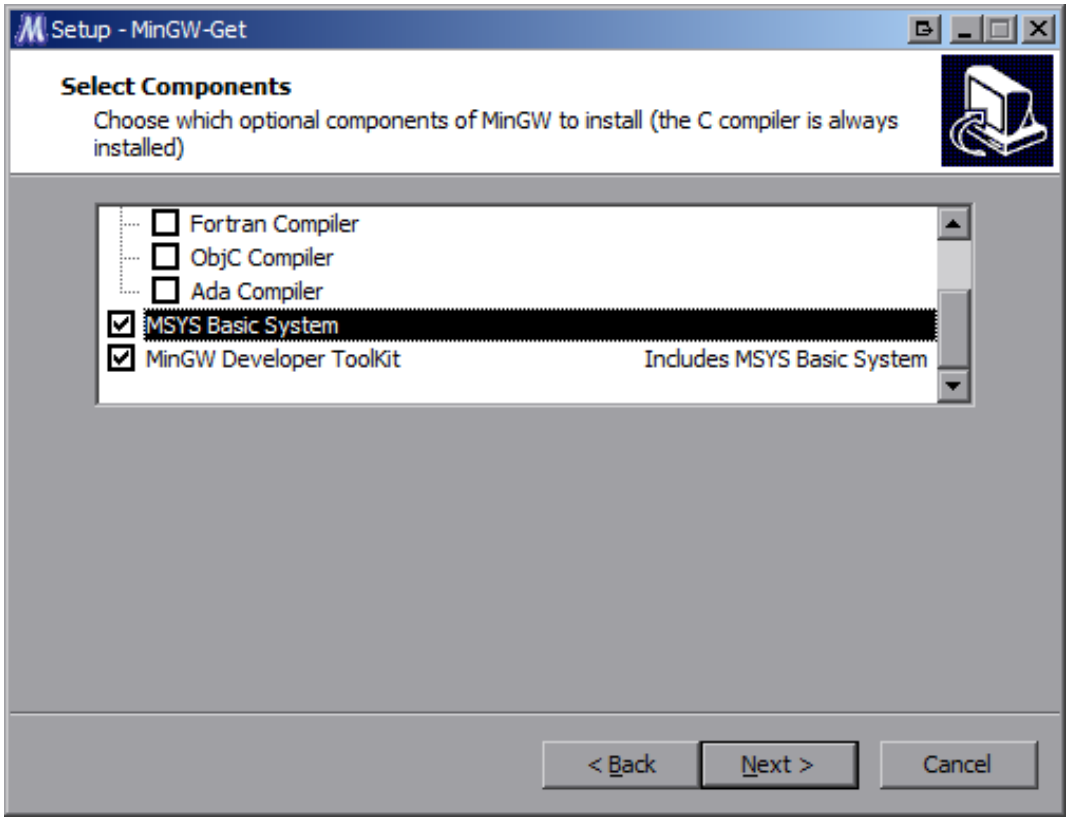


Рисунок 2-4: Установка MinGW. Выбор устанавливаемых компонентов

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.

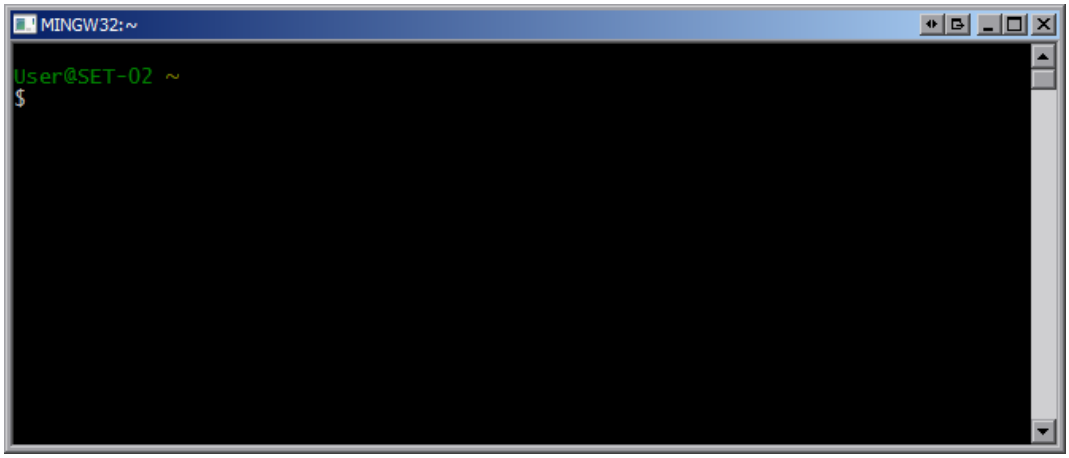


Рисунок 2-5: Приглашение командной строки MinGW Shell

2.2 Конфигурация окружения сборки

В данном разделе предполагается, что исходные коды загрузчика IBL переписаны с сопроводительного диска к модулю SVP-465 в папку «D:/Dev/Modules/SVP-465/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/SVP-465/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.

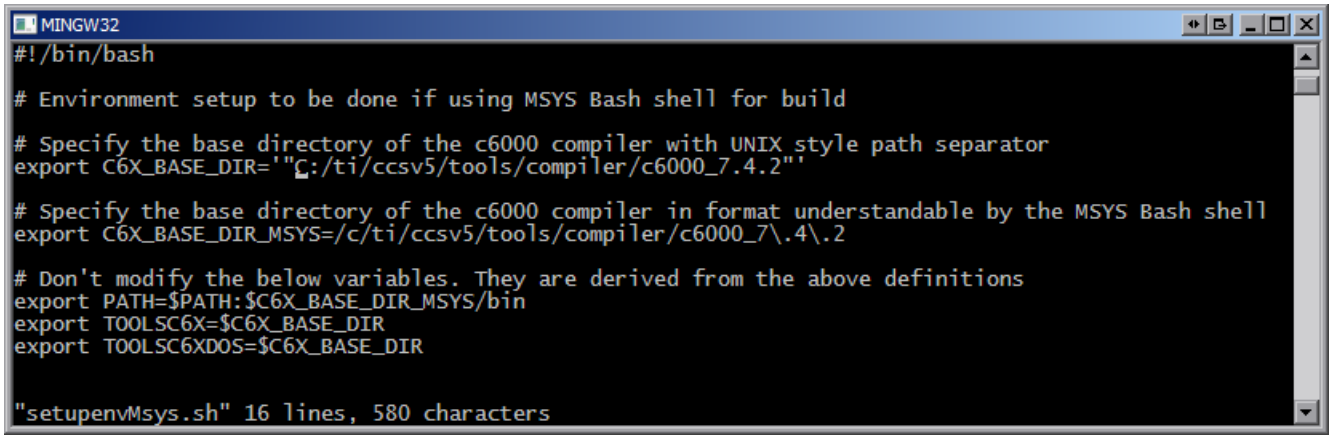


Рисунок 2-6: Редактирование файла «setupenvMsys.sh» в редакторе vim

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

```
vim /d/Dev/Modules/SVP-465/IBL/src/make/setupenvMsys.sh
```

В файле «D:/Dev/Modules/SVP-465/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 SVP-465, выполните в MinGW Shell последовательно следующие команды:

```
cd /d/Dev/Modules/SVP-465/IBL/src/make/
source setupenvMsys.sh
make svp465
```

После выполнения этих команд, будет запущена сборка загрузчика IBL для процессоров TMS320C6670 и TMS320C6678 модуля SVP-465, которая может занять до 30 минут (в зависимости от производительности системы). Имеется возможность выполнить сборку загрузчика IBL только для процессора TMS320C6670 или TMS320C6678 модуля SVP-465. Для сборки IBL только для процессора TMS320C6670 необходимо выполнять команду

```
make svp465_c6670
```

Для сборки IBL только для процессора TMS320C6678 необходимо выполнять команду

```
make svp465_c6678
```

В таблице 2-1 перечислены файлы, которые создаются в папке «D:/Dev/Modules/SVP-465/IBL/src/make/bin» после выполнения сборки загрузчика IBL для модуля SVP-465.

Таблица 2-1: Файлы, создаваемые при сборке загрузчика

Файл	Описание
«i2cConfig.gel»	GEL файл конфигурации IBL для CCS (см. раздел 4)
«i2cparam_0x50_svp465_c6670_le_0x500.out»	Бинарный образ программы конфигурации IBL для процессоров TMS320C6670 модуля SVP-465 (см. раздел 4)
«i2cparam_0x50_svp465_c6678_le_0x500.out»	Бинарный образ программы конфигурации IBL для процессоров TMS320C6678 модуля SVP-465 (см. раздел 4)
«i2crom_0x50_svp465_c6670_le.bin»	Бинарный образ IBL для процессоров TMS320C6670 для загрузки в EEPROM память модуля SVP-465 (см. раздел 3.4)
«i2crom_0x50_svp465_c6678_le.bin»	Бинарный образ IBL для процессоров TMS320C6678 для загрузки в EEPROM память модуля SVP-465 (см. раздел 3.4)
«i2crom_0x50_svp465_c6670_le.dat»	Бинарный образ IBL для процессоров TMS320C6670 модуля SVP-465 в формате CCS.
«i2crom_0x50_svp465_c6678_le.dat»	Бинарный образ IBL для процессоров TMS320C6678 модуля SVP-465 в формате CCS.

Для записи в EEPROM память модуля SVP-465 предназначены файлы «i2crom_0x50_svp465_c6670_le.bin» и «i2crom_0x50_svp465_c6678_le.bin» для процессоров TMS320C6670 и TMS320C6678 соответственно.

Запись образов 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 SVP-465

Для записи образов в EEPROM или NOR флеш память процессоров модуля SVP-465 предназначен специальный DSS скрипт «svp465_program.js», который расположен в папке «Program» на сопроводительном диске к модулю SVP-465.

Перед использованием скрипта «svp465_program.js» для записи EEPROM или NOR флеш памяти, необходимо скопировать с сопроводительного диска к модулю SVP-465 папки «Program», «TargetConfigurations» и «GEL» со всем содержимым в папку «D:/Dev/Modules/SVP-465»[1].

Скрипт «svp465_program.js» требует для своей работы установленную систему разработки CCS. Дистрибутив сетевой установки CCS имеется на сопроводительном диске к модулю SVP-465 (см. раздел 1).

Для запуска скрипта «svp465_program.js» в папке «Program» имеются вспомогательные скрипты запуска для Windows и Linux систем:

- «svp465_program.bat»: скрипт для запуска в Windows системе (см. листинг 3-1);
- «svp465_program.sh»: скрипт для запуска в Linux системе (см. листинг 3-2).

Листинг 3-1: Скрипт «svp465_program.bat»

```
@echo off
set PROGRAM_SVP_465_TARGET_CONFIG_FILE=../TargetConfigurations/SVP-465-USB560v2.ccxml
set DSS_SCRIPT_DIR="C:\ti\ccsv5\ccs_base\scripting\bin"
call %DSS_SCRIPT_DIR%\dss.bat svp465_program.js %*
```

Листинг 3-2: Скрипт «svp465_program.sh»

```
#!/bin/sh
export PROGRAM_SVP_465_TARGET_CONFIG_FILE=../TargetConfigurations/SVP-465-USB560v2.ccxml
export DSS_SCRIPT_DIR=~/ti/ccsv5/ccs_base/scripting/bin
$DSS_SCRIPT_DIR/dss.sh svp465_program.js $@
```

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

- «PROGRAM_SVP_465_TARGET_CONFIG_FILE»: путь к файлу описания целевой конфигурации модуля SVP-465;
- «DSS_SCRIPT_DIR»: путь к бинарным файлам DSS среды разработки CCS.

Важная информация

Только для Windows: В скрипте «svp465_program.bat» значение переменной «DSS_SCRIPT_DIR» обязательно должно быть записано в кавычках, а значение переменной «PROGRAM_SVP_465_TARGET_CONFIG_FILE», наоборот, не должно содержать кавычек.

После определения значений для переменных окружения, в скрипте происходит вызов скрипта кроссплатформенного DSS скрипта «svp465_program.js», с передачей ему аргументов командной строки.

Примечание

В данном документе рассматривается работа со скриптом «svp465_program.bat» для Windows системы. Для выполнения аналогичных действий в Linux системе, достаточно заменить вызовы скрипта «svp465_program.bat» на вызовы скрипта «svp465_program.sh».

В разделе 3.1 описывается структура каталога «Program», в котором работает скрипт «svp465_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 SVP-465, в котором располагаются все необходимые файлы для выполнения записи в EEPROM или NOR флеш память процессоров модуля SVP-465.

Листинг 3-3: Структура каталога «Program»

```
Program
 +- bin
 |  +- led_test
 |  |  +- led_test_c6670.bin
 |  |  +- led_test_c6678.bin
 |  |
 |  +- platform_test
 |  |  +- platform_test_c6670.bin
 |  |  +- platform_test_c6678.bin
 |  |
 |  +- eeprom_0x50_c6670.bin
 |  +- eeprom_0x50_c6678.bin
 |  +- nor_c6670.bin
 |  +- nor_c6678.bin
 |
 +- logs
 |
 +- writers
 |  +- eepromwriter_c6670.out
 |  +- eepromwriter_c6678.out
 |  +- norwriter_c6670.out
 |  +- norwriter_c6678.out
 |
 +- svp465_program.bat
 +- svp465_program.sh
 +- svp465_program.js
```

Кроме файлов скриптов «svp465_program.bat», «svp465_program.sh» и «svp465_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 SVP-465. Пользовательские файлы, подготовленные для записи в EEPROM или NOR флеш память, должны быть помещены в каталог «bin».

Непосредственно в каталоге «bin» расположены четыре файла, которые используются скриптом «svp465_program.js» для записи по умолчанию:

- «eeprom_0x50_c6670.bin» и «eeprom_0x50_c6678.bin»: образы для записи в EEPROM память процессоров TMS320C6670 и TMS320C6678 соответственно. По умолчанию, данные файлы представляют собой собранные образы загрузчика IBL, которые можно получить путем выполнения шагов, описанных в разделе 2 данного документа.
- «nor_c6670.bin» и «nor_c6678.bin»: образы для записи в NOR флеш память процессоров TMS320C6670 и TMS320C6678 соответственно. По умолчанию, данные файлы представляют собой собранные образы демонстрационного приложения веб-сервера. Подробное описание по сборке и запуску демонстрационного приложения веб-сервера дано в документе [1].

В каталогах «bin/led_test» и «bin/platform_test» расположены образы приложений, готовые для записи в NOR флеш память процессоров TMS320C6670 и TMS320C6678.

В каталоге «bin/platform_test» размещены образы приложения теста платформы (файлы «platform_test_c6670.bin» и «platform_test_c6678.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/led_test» находятся образы такого же приложения теста платформы, но в котором отключены все тесты кроме теста светодиодов (файлы «led_test_c6670.bin» и «led_test_c6678.bin»).

Исходные коды приложения теста платформы находятся проектах «PlatformTest_C6670» и «PlatformTest_C6678» рабочего пространства CCS. Папку с рабочим пространством можно найти на сопроводительном диске к модулю SVP-465 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» (файлы «eepromwriter_c6670.out» и «eepromwriter_c6678.out») и NOR_Writer» (файлы «norwriter_c6670.out» и «norwriter_c6678.out») рабочего пространства CCS для модуля SVP-465. Оба данных проекта имеют по две цели сборки «C6670» и «C6678» для процессоров TMS320C6770 и TMS320C6678 соответственно. 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 SVP-465 осуществляется из командной строки. Выбор того или иного режима работы скрипта осуществляется при помощи параметров командной строки, передаваемых скрипту при запуске.

Если запустить скрипт «svp465_program.bat» без параметров, будет выдана краткая справка по возможным параметрами командной строки (см. листинг 3-4).

Листинг 3-4: Вывод скрипта «svp465_program.bat», запущенного без параметров

```
D:\Dev\Modules\SVP-465\Program>svp465_program.bat

Syntax: svp465_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, DSP2, DSP3 and DSP4)
  image=<image>        : image file for all DSP's
  image_c6670=<image>  : image file for TMS320C6670 DSP's (DSP1 and DSP3)
                         (image must be in 'bin' folder)
  image_c6678=<image>  : image file for TMS320C6678 DSP's (DSP2 and DSP4)
                         (image must be in 'bin' folder)
  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

Example #1: svp465_program.js NOR dsp2 image_dsp2=user_nor.bin dsp3 dsp4
  - Program NOR flash on DSP2 with image 'bin/user_nor.bin'
  - Program NOR flash on DSP3 with default NOR image
    (bin/nor_c6670.bin)
  - Program NOR flash on DSP4 with default NOR image
    (bin/nor_c6678.bin)

Example #2: svp465_program.js EEPROM dsp1 dsp2 image_dsp1=user.bin bus_dsp1=81
  - Program I2C 0x50 EEPROM on DSP1 with image 'bin/user.bin'
  - Program I2C 0x51 EEPROM on DSP2 with default IBL image
    (bin/eeprom_0x50_c6678.bin)

Example #3: svp465_program.js EEPROM all
  - Program I2C 0x51 EEPROM on all DSP's with default IBL images
    (bin/eeprom_0x50_c6670.bin for TMS320C6670 and
     bin/eeprom_0x50_c6678.bin for TMS320C6678 processors)

D:\Dev\Modules\SVP-465\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 «svp465_program.js»

Параметр	Описание
dsp<n>	Включает запись образа в память для процессора с номером <n>. Значение <n> может быть от 1 до 4: • 1: DSP1_TMS320C6670; • 2: DSP2_TMS320C6678; • 3: DSP3_TMS320C6670; • 4: DSP4_TMS320C6678.
all	Включает запись образа в память сразу для всех 4-х процессоров модуля SVP-465. Эквивалентно заданию параметров командной строки dsp<n> для всех <n>.
image=<image>	Параметр задает имя файла (<image>) образа для записи на всех четырех процессорах. Может использоваться в том случае, когда на все процессоры необходимо записать один и тот же образ. Как правило, образы для процессоров TMS320C6670 и TMS320C6678 различные, в этом случае рекомендуется использовать параметры image_c6670=<image> и image_c6678=<image> для указания образов для процессоров TMS320C6670 и TMS320C6678 раздельно.
image_c6670=<image>	Параметр задает имя файла (<image>) образа для записи на процессорах TMS320C6670 (DSP1 и DSP3). Файл <image> должен быть помещен в каталог «bin». По умолчанию, если данный параметр не задан, используются следующие файлы образов: • для режима NOR: файл «nor_c6670.bin»; • для режима EEPROM: файл «eeprom_0x50_c6670.bin».
image_c6678=<image>	Параметр задает имя файла (<image>) образа для записи на процессорах TMS320C6678 (DSP2 и DSP4). Файл <image> должен быть помещен в каталог «bin». По умолчанию, если данный параметр не задан, используются следующие файлы образов: • для режима NOR: файл «nor_c6678.bin»; • для режима EEPROM: файл «eeprom_0x50_c6678.bin».
image_dsp<n>=<image>	Параметр позволяет указать имя файла (<image>) образа для конкретного процессора <n>. Если заданы параметры image_c667x и image_dsp<n>, то будет использован образ, который указан в параметре image_dsp<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 4-х процессоров. Значение данного параметра учитывается только в режиме EEPROM. По умолчанию, если параметр не задан, используется адрес шины 80 (0x50).
log	Параметр включает запись вывода CIO со всех процессоров модуля SVP-465 в файлы, которые будут помещены в каталог «logs». Создаваемые файлы лога имеют формат «SVP-465-DSP<n>-<time>-cio.txt», где <n> — номер процессора модуля SVP-465 (от 1 до 4), <time> — время запуска скрипта «svp465-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 SVP-465 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 SVP-465, 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/SVP-465/Program/bin».
3. Перейдите в каталог «D:/Dev/Modules/SVP-465/Program».
4. Запустите скрипт «svp465_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 SVP-465.

В качестве примера, в процедуре 3-2 подробно описаны шаги, необходимые для выполнения записи в NOR флеш память всех четырех процессоров модуля SVP-465 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 процессоров TMS320C6670 и TMS320C6678 для конфигураций сборки «Debug» (процесс сборки приложения веб-сервера подробно описан в документе [1]).

   В результате сборки приложений веб-сервера для обоих типов процессоров (TMS320C6670 и TMS320C6678) должно быть получено два файла «.out»:

   - «D:/Dev/Modules/SVP-465/CCS_Workspace/WebServer_C6670/Debug/websrv_c6670.out»;
   - «D:/Dev/Modules/SVP-465/CCS_Workspace/WebServer_C6678/Debug/websrv_c6678.out».

   Примечание

   Модуль SVP-465 поставляется с записанным приложением теста платформы, в котором выполняются последовательно тест светодиодов на передней панели модуля и 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 файлы, полученные в шаге 1, в папку «D:/Dev/Modules/SVP-465/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\SVP-465
copy CCS_Workspace\WebServer_C6670\Debug\websrv_c6670.out Program\bin\websrv_c6670.bin
copy CCS_Workspace\WebServer_C6678\Debug\websrv_c6678.out Program\bin\websrv_c6678.bin
```

3. Установите для всех процессоров модуля SVP-465 режим «NOBOOT» в соответствии с данными таблицы B-1 приложения B.
4. Подключите отладчик к модулю SVP-465 и к компьютеру (предполагается использование отладчика Blackhawk XDS560v2-USB System Trace Emulator).
5. Включите модуль SVP-465.
6. Перейдите в каталог «D:/Dev/Modules/SVP-465/Program». Для этого необходимо выполнить в терминале команды:

```
d:
cd D:\Dev\Modules\SVP-465\Program
```

7. Запустите скрипт «svp465_program.bat» выполнив в терминале команду:

```
svp465_program.bat NOR all image_c6670=websrv_c6670.bin image_c6678=websrv_c6678.bin
```

Будет запущен процесс записи образов в NOR флеш память процессоров модуля SVP-465. 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 SVP-465 начинается процесс записи данных образов в NOR флеш память. При этом, светодиоды «DSP» на передней панели модуля SVP-465 должны загореться желтым цветом для соответствующих процессоров (см. рисунок 3-1).

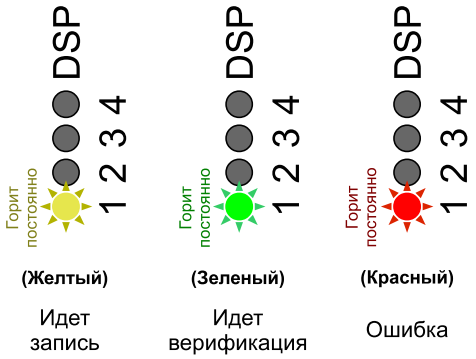


Рисунок 3-1: Состояния светодиодов на передней панели модуля SVP-465 во время записи

После завершения записи, начинается процесс верификации записанных данных, при этом светодиод «DSP» для соответствующего процессора загорается зеленым цветом (см. рисунок 3-1). О начале процесса верификации на каком либо из процессоров, свидетельствуют сообщения вида «`Reading and verifying sector 3 (196608 bytes of 4665508)`» выводимые в теримнал.

Если во время записи или верификации происходит ошибка, то светодиод «DSP» для соответствующего процессора загорается красным цветом (см. рисунок 3-1) и процесс записи для данного процессора прекращается.

В случае успешного завершения записи и верификации записанных данных, светодиод «DSP» для соответствующего процессора должен погаснуть.

8. Выключите модуль SVP-465.
9. Если в EEPROM память процессоров модуля SVP-465 еще не записан образ загрузчика IBL, выполните шаги процедуры 3-4.

Примечание

Модуль SVP-465 поставляется с уже записанным образом загрузчика IBL в EEPROM память (адрес I^2C шины 0x50) всех четырех 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 SVP-465 режим «NOR» в соответствии с данными таблицы B-1 приложения B.
11. Включите модуль SVP-465.

После выполнения всех шагов процедуры 3-2, вывод в UART с процессоров модлуя SVP-465 при его включении должен выглядеть подобно приведенному в листинге 3-5. В листинге 3-5 приведен вывод в UART загрузки демонстарционного приложения веб-сервера на процессоре «DSP1_C6670». Вывод в UART с других процессоров должен выглядеть аналогично.

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

```

IBL INIT
I2C boot
EDC enabled
Boot table processor executed

IBL version: 1.0.0.16-4
Device is SVP-465
00:17:ea:ed:d0:84
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: 00-17-EA-ED-D0-84
EMAC ports count: 1
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-1:10.0.1.197
Service Status: DHCPC    : Enabled  : Running  : 017
```

В листинге 3-6 приведен пример вывода в UART запуска приложения теста платформы для процессоров TMS320C6670 модуля SVP-465 (файл «Program/bin/platform_test_c6670.bin»).

Листинг 3-6: Вывод в UART при запуске теста платформы на процессоре TMS320C6670

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

p_info->version                          = 2.00.00.15
p_info->board_name                       = SVP-465
p_info->board_id                         = SVP465
p_info->serial_nbr                       = 4160501
p_info->slot_n                           = 9
p_info->dsp_n                            = DSP1
```

```
p_info->cpu.core_count                      = 4
p_info->cpu.name                            = TMS320C6670
p_info->cpu.id                              = 21
p_info->cpu.revision_id                     = 0
p_info->cpu.endian                          = 1
p_info->frequency                           = 983 MHz
p_info->led[PLATFORM_USER_LED_CLASS].count = 1
p_info->led[PLATFORM_AIF_LED_CLASS].count  = 4
p_info->emac.port_count                     = 1
  -> EMAC port 0
     MAC address = 00:17:ea:ed:d7:27
Current core id is 0

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 (1 leds)...
- LED 0 GREEN
- LED 0 RED
- LED 0 GREEN and RED
- LED 0 OFF
Testing AIF class leds (4 leds)...
- LED 0 GREEN
- LED 0 RED
- LED 0 GREEN and RED
- LED 0 OFF
- LED 1 GREEN
- LED 1 RED
- LED 1 GREEN and RED
- LED 1 OFF
- LED 2 GREEN
- LED 2 RED
- LED 2 GREEN and RED
- LED 2 OFF
- LED 3 GREEN
- LED 3 RED
- LED 3 GREEN and RED
- LED 3 OFF
LED test complete

-------------
```

```
 EEPROM test
-------------
test_eeprom: Full EEPROM test mode
test_eeprom: 0x50: Saving EEPROM contents...
test_eeprom: 0x50: Writing 65536 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 65536 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: Full NOR test mode
test_nor: NOR flash total size is 16777216 bytes
test_nor: NOR flash sector size is 65536 bytes
test_nor: Test block size is 65536 bytes
test_nor: Sector 0/255...
test_nor: Sector 1/255...
test_nor: Sector 2/255...
test_nor: Sector 3/255...
test_nor: Sector 4/255...
test_nor: Sector 5/255...
test_nor: Sector 6/255...
test_nor: Sector 7/255...
test_nor: Sector 8/255...
test_nor: Sector 9/255...
test_nor: Sector 10/255...
test_nor: Sector 11/255...

ВЫРЕЗАНО: Вырезаны аналогичные сообщения для других секторов NOR флеш памяти

test_nor: Sector 240/255...
test_nor: Sector 241/255...
test_nor: Sector 242/255...
test_nor: Sector 243/255...
test_nor: Sector 244/255...
test_nor: Sector 245/255...
test_nor: Sector 246/255...
test_nor: Sector 247/255...
test_nor: Sector 248/255...
test_nor: Sector 249/255...
test_nor: Sector 250/255...
test_nor: Sector 251/255...
test_nor: Sector 252/255...
test_nor: Sector 253/255...
test_nor: Sector 254/255...
test_nor: Sector 255/255...
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 SVP-465 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 SVP-465, составляет 128 Кбайт на каждый процессор. EEPROM память, установленная на модуле SVP-465, 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 SVP-465 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/SVP-465/Program/bin».
3. Перейдите в каталог «D:/Dev/Modules/SVP-465/Program».
4. Запустите скрипт «svp465_program.bat» в режиме `EEPROM` с требуемыми параметрами командной строки (см. таблицу 3-1 раздела 3.2).

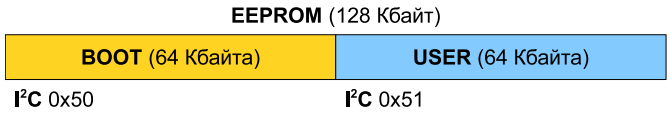


Рисунок 3-2: Организация EEPROM памяти на модуле SVP-465

В процедуре 3-3 представлены общие шаги, необходимые для записи образа в EEPROM память процессоров модуля SVP-465.

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 SVP-465 образа загрузчика IBL.

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

1. Выполните сборку загрузчика IBL для процессоров TMS320C6670 и TMS320C6678 (процесс сборки IBL подробно описан в разделе 2).

   В результате сборки загрузчика IBL для обоих типов процессоров (TMS320C6670 и TMS320C6678) должно быть получено два файла «.bin»:

   - «D:/Dev/Modules/SVP-465/ibl/src/make/bin/i2crom_0x50_svp465_c6670_le.bin»;
   - «D:/Dev/Modules/SVP-465/ibl/src/make/bin/i2crom_0x50_svp465_c6678_le.bin».

Примечание

Модуль SVP-465 поставляется с уже записанным образами загрузчика IBL в EEPROM память. Образы, записанные в EEPROM память при изготовлении модуля SVP-465 содержатся в файлах «eeprom_0x50_c6670.bin» (для процессоров TMS320C6670) и «eeprom_0x50_c6678.bin» (для процессоров TMS320C6678). Данные файлы располагаются в каталоге «D:/Dev/Modules/SVP-465/Program/bin».

2. Скопируйте файлы, полученные в шаге 1, в папку «D:/Dev/Modules/SVP-465/Program/bin». Для этого выполните в терминале последовательно следующие команды:

```
d:
cd D:\Dev\Modules\SVP-465
copy ibl\src\make\bin\i2crom_0x50_svp465_c6670_le.bin Program\bin
copy ibl\src\make\bin\i2crom_0x50_svp465_c6678_le.bin Program\bin
```

3. Установите для всех процессоров модуля SVP-465 режим «NOBOOT» в соответствии с данными таблицы B-1 приложения B.
4. Подключите отладчик к модулю SVP-465 и к компьютеру (предполагается использование отладчика Blackhawk XDS560v2-USB System Trace Emulator).
5. Включите модуль SVP-465.
6. Перейдите в каталог «D:/Dev/Modules/SVP-465/Program». Для этого необходимо выполнить в терминале команды:

```
d:
cd D:\Dev\Modules\SVP-465\Program
```

7. Запустите скрипт «svp465_program.bat» выполнив в терминале команду:

```
svp465_program.bat EEPROM all image_c6670=i2crom_0x50_svp465_c6670_le.bin image_c6678=
    ← i2crom_0x50_svp465_c6678_le.bin
```

   Будет запущен процесс записи образов в EEPROM память процессоров модуля SVP-465. 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 SVP-465 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 SVP-465.
9. Установите для всех процессоров модуля SVP-465 требуемый режим загрузки («NOR», «TFTP» или «PCI-E») в соответствии с данными таблицы B-1 приложения B.
10. Включите модуль SVP-465.

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), создаются образы специальных программ «i2cparam_0x50_svp465_c6670_le_0x500.out» и «i2cparam_0x50_svp465_c6678_le_0x500.out» (см. таблицу 2-1), которые предназначены для загрузки на процессорах TMS320C6770 и TMS320C6678 модуля SVP-465 соответственно.

Примечание

Значение «0x500» в имени файлов «i2cparam_0x50_svp465_c667x_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 SVP-465, 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_C6670»). Для этого, нажимте правой кнопкой мыши на названии ядра «DSP1_C6670_0» и выберите пункт меню «Connect Target» (см. рисунок 4-1).

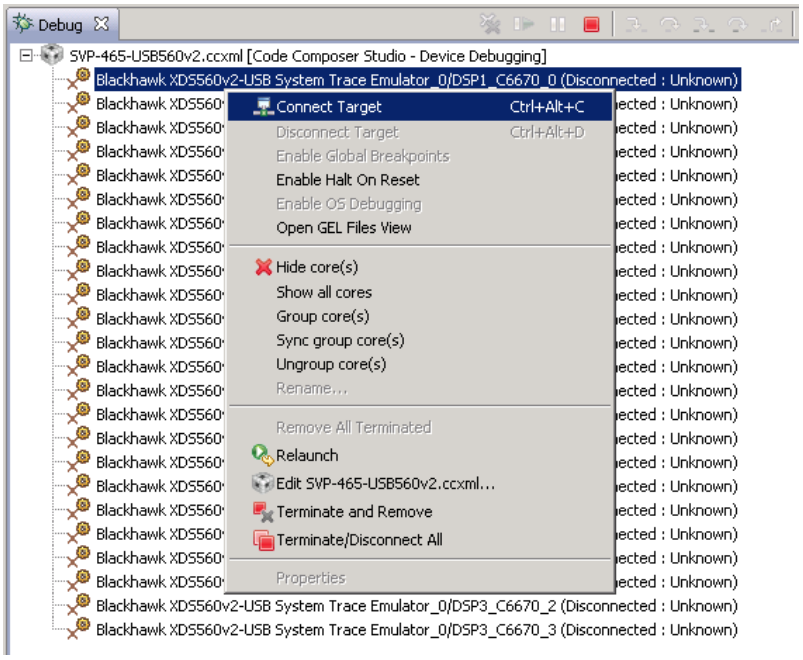


Рисунок 4-1: Подлкючение к процессору «DSP1_C6670»

Выберите ядро «DSP1_C6670_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».

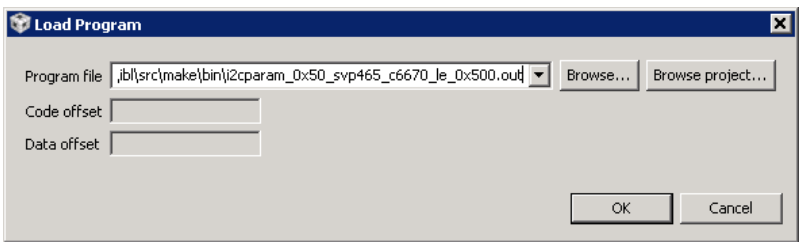


Рисунок 4-2: Окно загрузки кода на ядро процессора

В открывшемся окне выбора файлов необходимо выбрать файл «i2cparam_0x50_svp465_c6670_le_0x500.out» (см. таблицу 2-1) и нажать на кнопку «OK».

Примечание

Для процессоров TMS320C6678 необходимо выбирать файл «i2cparam_0x50_svp465_c6678_le_0x500.out».

После загрузки кода на ядро процессора TMS320C6670, окно «Debug» должно выглядеть, как показано на рисунке 4-3.

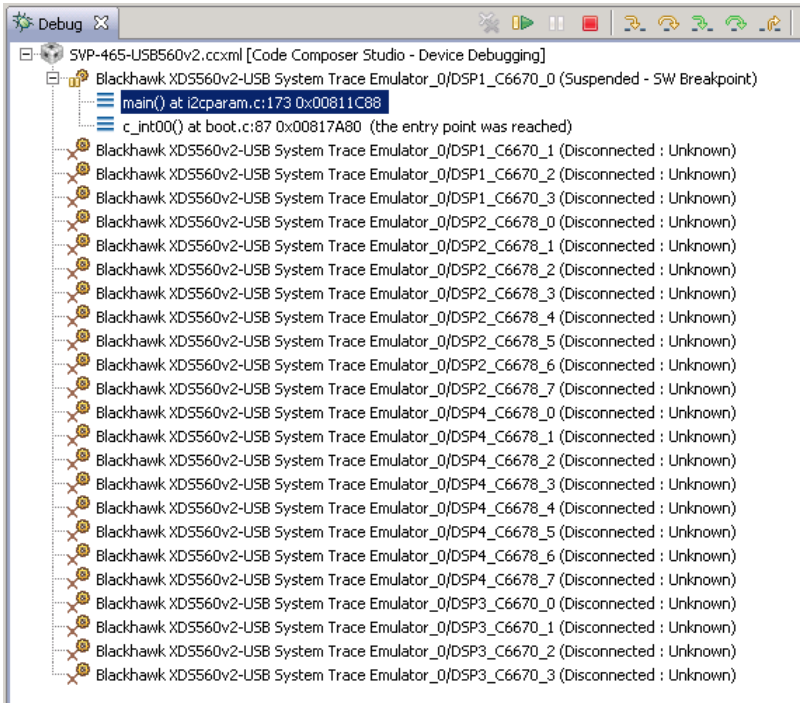


Рисунок 4-3: Внешний вид окна «Debug» после загрузки кода на ядра процессоров

Выделите в окне «Debug» ядро «DSP1_C6670_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).

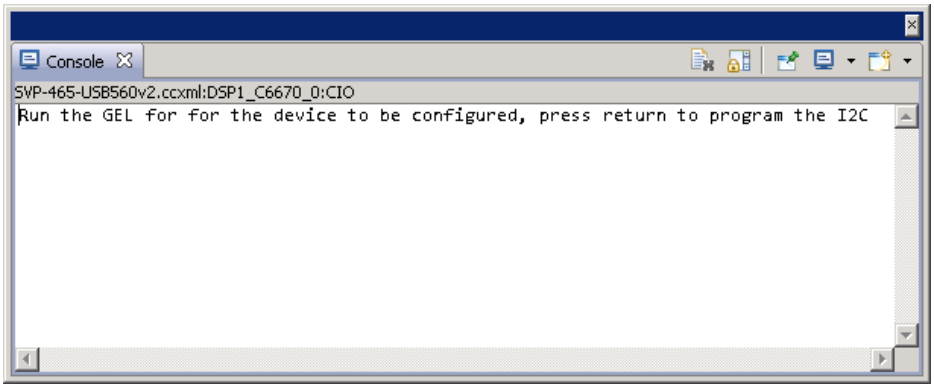


Рисунок 4-4: Внешний вид окна «Console» после запуска кода на процессоре

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/SVP-465/ibl/srv/make/bin» (см. таблицу 2-1) после выполнения сборки загрузчика (см. раздел 2).

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

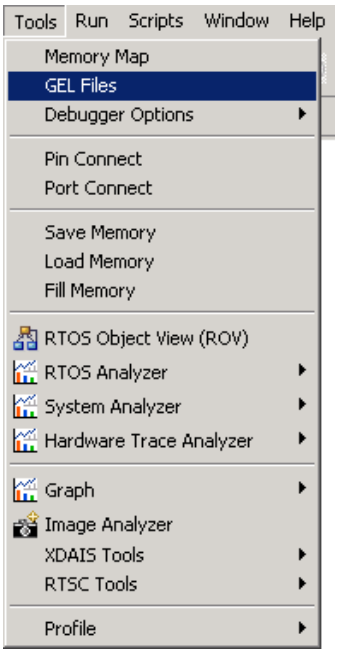


Рисунок 4-5: Пункт главного меню для вызова окна управления GEL файлами

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 «svp465_c6670.gel») и выберите пункт меню «Load GEL...».

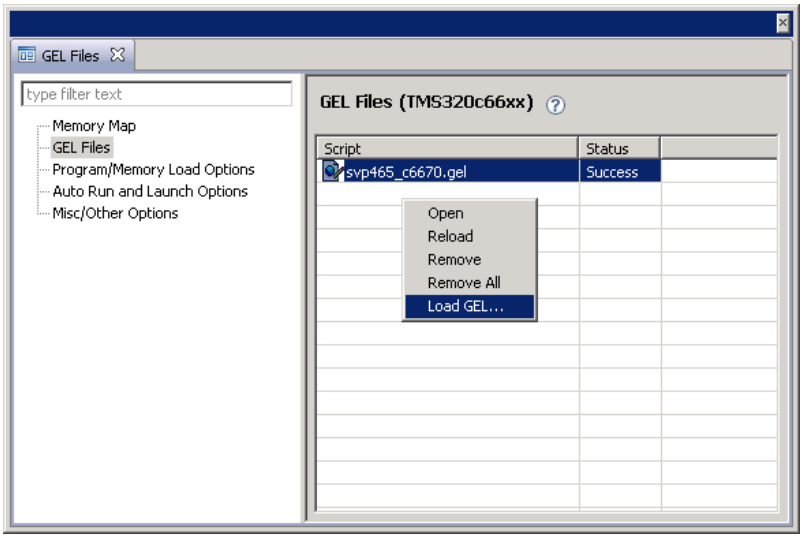


Рисунок 4-6: Окно управления GEL файлами

В открывшемся окне необходимо выбрать файл «D:/Dev/Modules/SVP-465/ibl/src/make/bin/i2cConfig.gel». После чего, в списке загруженных файлов окна «GEL Files» должен появиться файл «i2cConfig.gel», как показано на рисунке 4-7.

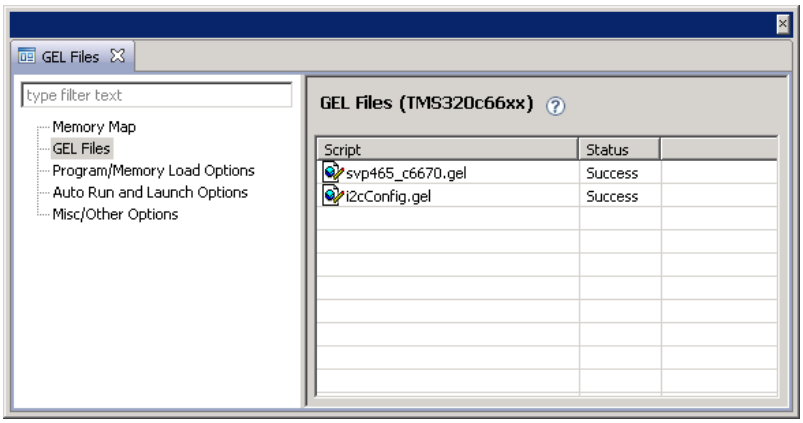


Рисунок 4-7: Окно управления GEL файлами с загруженным файлом «i2cConfig.gel»

После выполнения данных действи, должно появиться два пункта меню «Scripts > SET SVP-465 IBL > setConfig_svp465_c6670_main» и «Scripts > SET SVP-465 IBL > setConfig_svp465_c6678_main» при выборе которых будет применена конфигурация загрузчика (см. рисунок 4-8).

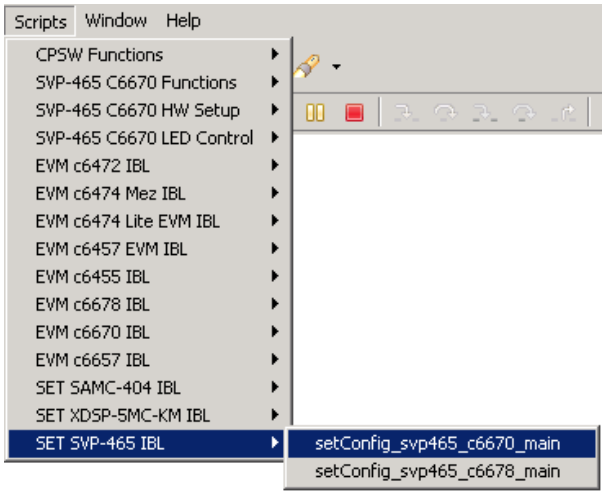


Рисунок 4-8: Пункт главного меню «Scripts > SET SVP-465 IBL»

При конфигурации загрузчика на процессорах TMS320C6670 следует выбирать пункт меню «setConfig_svp465_c6670_main», а на процессорах TMS320C6678, сооветственно, пункт меню «setConfig_svp465_c6678_main».

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).

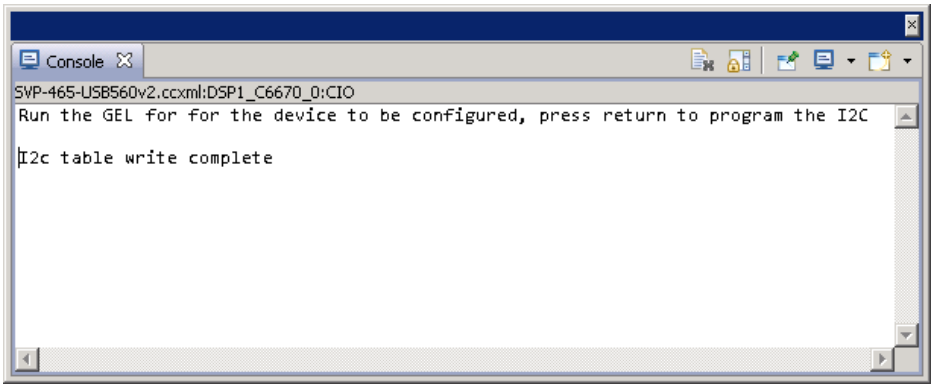


Рисунок 4-9: Внешний вид окна «Console» после выполнения записи конфигурации в EEPROM память

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 SVP-465.

Листинг 4-1: Фрагмент GEL файла «i2cConfig.gel» с конфигурационными параметрами загрузчика IBL для модуля SVP-465

```
menuitem "SET SVP-465 IBL";

hotmenu setConfig_svp465_c6670_main()
{
    ibl.iblEvmType = ibl_SVP_465_C6670;

    /* Main PLL: 122.88 MHz reference, 983MHz output */
    ibl.pllConfig[ibl_MAIN_PLL].doEnable      = 1;
    ibl.pllConfig[ibl_MAIN_PLL].prediv        = 1;
    ibl.pllConfig[ibl_MAIN_PLL].mult          = 16;
    ibl.pllConfig[ibl_MAIN_PLL].postdiv       = 2;
    ibl.pllConfig[ibl_MAIN_PLL].pllOutFreqMhz = 983;

    svp465_ibl_config();
}

hotmenu setConfig_svp465_c6678_main()
{
    ibl.iblEvmType = ibl_SVP_465_C6678;

    /* Main PLL: 122.88 MHz reference, 1000MHz output */
    ibl.pllConfig[ibl_MAIN_PLL].doEnable      = 1;
    ibl.pllConfig[ibl_MAIN_PLL].prediv        = 29;
    ibl.pllConfig[ibl_MAIN_PLL].mult          = 472;
    ibl.pllConfig[ibl_MAIN_PLL].postdiv       = 2;
    ibl.pllConfig[ibl_MAIN_PLL].pllOutFreqMhz = 1000;

    svp465_ibl_config();
}

svp465_ibl_config()
{
    ibl.iblMagic = ibl_MAGIC_VALUE;

    /* 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;

    ibl.sgmiiConfig[0].configure     = 1;
    ibl.sgmiiConfig[0].adviseAbility = 0x9801;
    ibl.sgmiiConfig[0].control       = 32;
    ibl.sgmiiConfig[0].txConfig      = 0x106a1;
    ibl.sgmiiConfig[0].rxConfig      = 0x700621;
    ibl.sgmiiConfig[0].auxConfig     = 0x51;

    ibl.sgmiiConfig[1].configure     = 0;
    ibl.sgmiiConfig[1].adviseAbility = 1;
    ibl.sgmiiConfig[1].control       = 1;
    ibl.sgmiiConfig[1].txConfig      = 0x106a1;
    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;

    ibl.bootModes[0].u.norBoot.bootFormat               = ibl_BOOT_FORMAT_ELF;
    ibl.bootModes[0].u.norBoot.bootAddress[0][0]        = 0;            /* Image 0 NOR offset byte address in LE mode */
    ibl.bootModes[0].u.norBoot.bootAddress[0][1]        = 0x800000;     /* Image 1 NOR offset byte address in LE mode  */
    ibl.bootModes[0].u.norBoot.bootAddress[1][0]        = 0;            /* Image 0 NOR offset byte address in BE mode */
    ibl.bootModes[0].u.norBoot.bootAddress[1][1]        = 0x800000;     /* Image 1 NOR offset byte address in BE mode  */
    ibl.bootModes[0].u.norBoot.interface                = ibl_PMEM_IF_SPI;
    ibl.bootModes[0].u.norBoot.blob[0][0].startAddress  = 0x80000000;   /* Image 0 load start address in LE mode */
    ibl.bootModes[0].u.norBoot.blob[0][0].sizeBytes     = 0x800000;     /* Image 0 size (10 MB) in LE mode */
    ibl.bootModes[0].u.norBoot.blob[0][0].branchAddress = 0x80000000;   /* Image 0 branch address after loading in LE mode */
    ibl.bootModes[0].u.norBoot.blob[0][1].startAddress  = 0x80000000;   /* Image 1 load start address in LE mode */
    ibl.bootModes[0].u.norBoot.blob[0][1].sizeBytes     = 0x800000;     /* Image 1 size (10 MB) in LE mode */
    ibl.bootModes[0].u.norBoot.blob[0][1].branchAddress = 0x80000000;   /* Image 1 branch address after loading in LE mode */
    ibl.bootModes[0].u.norBoot.blob[1][0].startAddress  = 0x80000000;   /* Image 0 load start address in BE mode */
    ibl.bootModes[0].u.norBoot.blob[1][0].sizeBytes     = 0x800000;     /* Image 0 size (10 MB) in BE mode */
    ibl.bootModes[0].u.norBoot.blob[1][0].branchAddress = 0x80000000;   /* Image 0 branch address after loading in BE mode */
    ibl.bootModes[0].u.norBoot.blob[1][1].startAddress  = 0x80000000;   /* Image 1 load start address in BE mode */
    ibl.bootModes[0].u.norBoot.blob[1][1].sizeBytes     = 0x800000;     /* Image 1 size (10 MB) in BE mode */
    ibl.bootModes[0].u.norBoot.blob[1][1].branchAddress = 0x80000000;   /* Image 1 branch address after loading in BE mode */

    ibl.bootModes[1].bootMode = ibl_BOOT_MODE_NONE;

    ibl.bootModes[2].bootMode = ibl_BOOT_MODE_TFTP;
    ibl.bootModes[2].priority = ibl_HIGHEST_PRIORITY + 1;
    ibl.bootModes[2].port     = 0;

    ibl.bootModes[2].u.ethBoot.doBootp          = TRUE;
    ibl.bootModes[2].u.ethBoot.useBootpServerIp = TRUE;
    ibl.bootModes[2].u.ethBoot.useBootpFileName = TRUE;
    ibl.bootModes[2].u.ethBoot.bootFormat       = ibl_BOOT_FORMAT_ELF;

    SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.ipAddr,    192,168,1,3);
    SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.serverIp,  192,168,1,2);
    SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.gatewayIp, 192,168,1,1);
    SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.netmask,   255,255,255,0);

    /* Use the e-fuse value */
    ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[0] = 0;
    ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[1] = 0;
    ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[2] = 0;
    ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[3] = 0;
    ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[4] = 0;
    ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[5] = 0;

    ibl.bootModes[2].u.ethBoot.ethInfo.fileName[0]  = 's';
    ibl.bootModes[2].u.ethBoot.ethInfo.fileName[1]  = 'v';
    ibl.bootModes[2].u.ethBoot.ethInfo.fileName[2]  = 'p';
```

```
    ibl.bootModes[2].u.ethBoot.ethInfo.fileName[3]  = '4';
    ibl.bootModes[2].u.ethBoot.ethInfo.fileName[4]  = '6';
    ibl.bootModes[2].u.ethBoot.ethInfo.fileName[5]  = '5';
    ibl.bootModes[2].u.ethBoot.ethInfo.fileName[6]  = '.';
    ibl.bootModes[2].u.ethBoot.ethInfo.fileName[7]  = 'b';
    ibl.bootModes[2].u.ethBoot.ethInfo.fileName[8]  = 'i';
    ibl.bootModes[2].u.ethBoot.ethInfo.fileName[9]  = 'n';
    ibl.bootModes[2].u.ethBoot.ethInfo.fileName[10] = '\0';

    ibl.bootModes[2].u.ethBoot.blob.startAddress  = 0x80000000;   /* Load start address */
    ibl.bootModes[2].u.ethBoot.blob.sizeBytes     = 0x20000000;
    ibl.bootModes[2].u.ethBoot.blob.branchAddress = 0x80000000;   /* Branch address after loading */

    ibl.chkSum = 0;
}
```

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 SVP-465 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`.

Продолжение таблицы на следующей странице

Продолжение таблицы 4-1

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

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

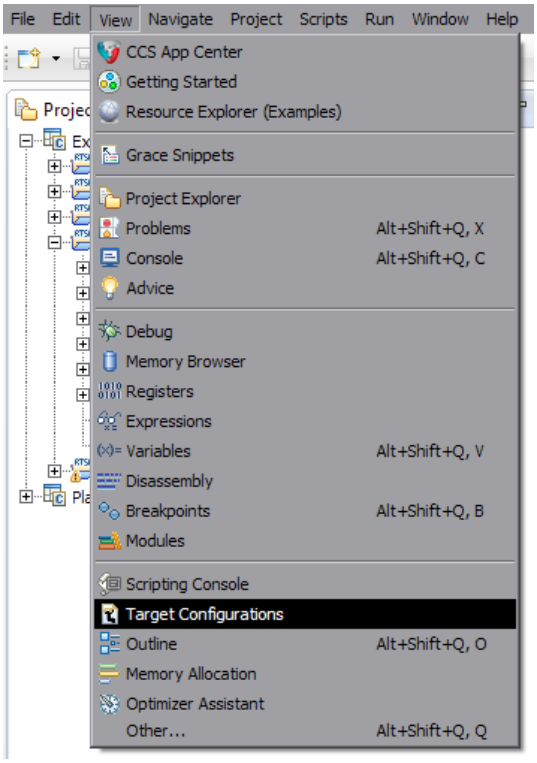


Рисунок 5-1: Пункт меню для отображения окна целевых конфигураций

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

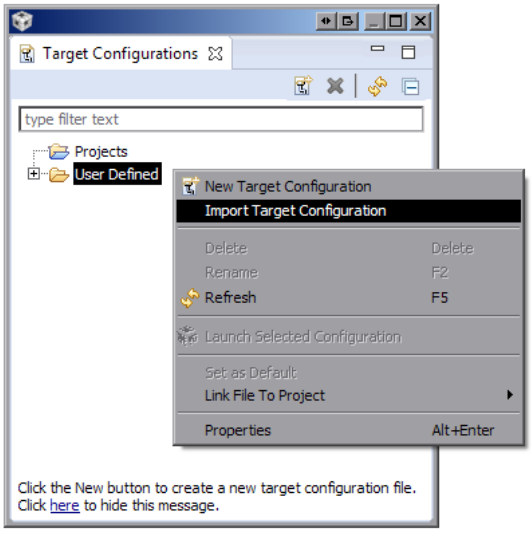


Рисунок 5-2: Меню импорта целевой конфигурации

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

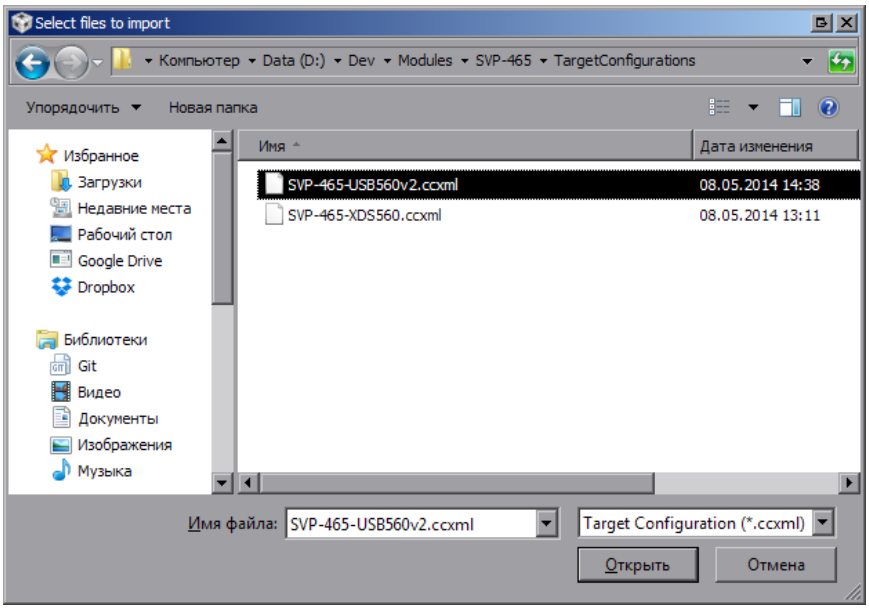


Рисунок 5-3: Окно выбора файла для импорта целевой конфигурации

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

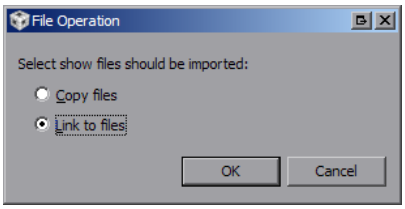


Рисунок 5-4: Окно выбора способа импорта файла целевой конфигурации

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).

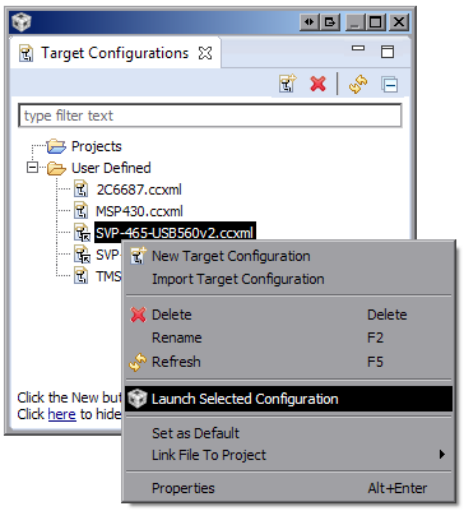


Рисунок 5-5: Запуск целевой конфигурации

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

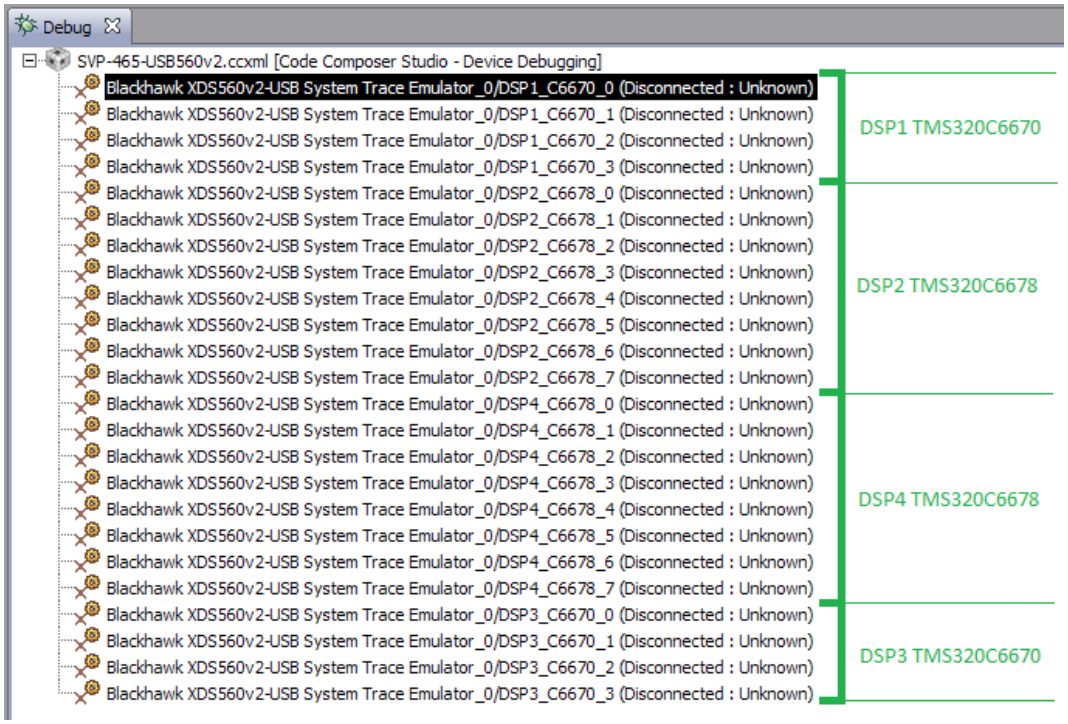


Рисунок 5-6: Список ядер процессоров модуля SVP-465

Примечание

По умолчанию, среда CCS настроена таким образом, что при запуске кода сразу на нескольких процессорах или нескольких ядрах одного процессора, вывод (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».

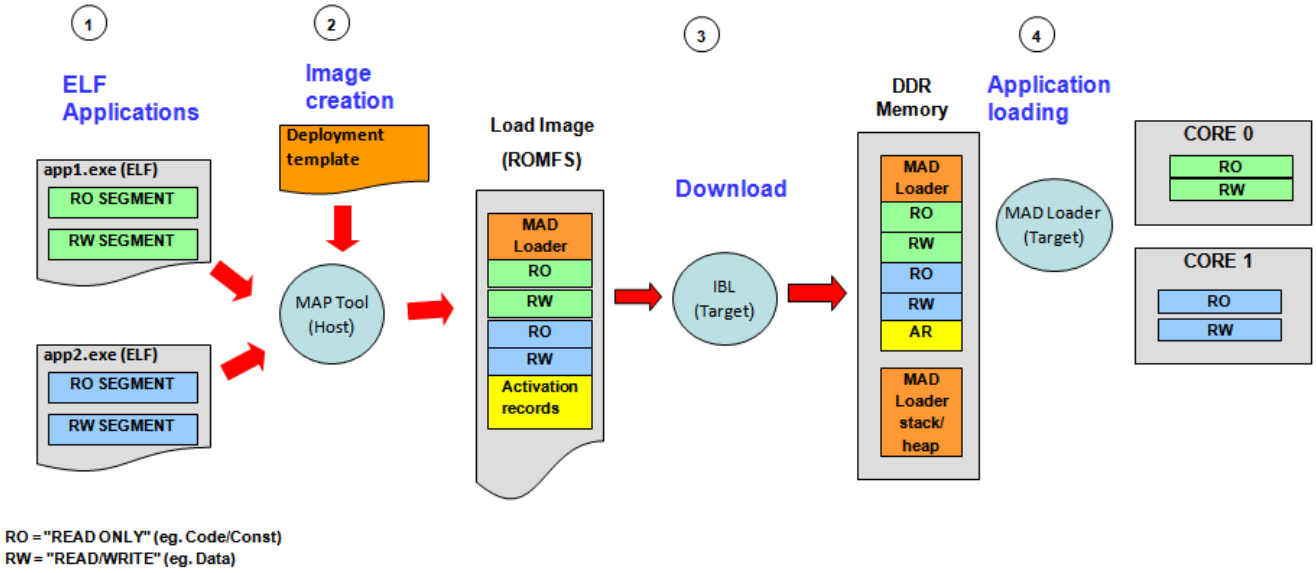


Рисунок 6-1: Схема работы MAD в режиме «Prelinker bypass mode»

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;
`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/

6.8 Пример использования

В качестве примера, в данном разделе будет выполнена подготовка образа приложения теста производительности SRIO (Serial RapidIO) для генератора данных [4] для запуска на процессоре TMS320C6678 модуля SVP-465.

Согласно документу [4], приложение генератора данных необходимо обязательно запускать на втором ядре процессора.

Создайте файл «maptool_c6678_producer.json» в папке «D:/Dev/Modules/SVP-465/CCS_Workspace/SRIO_-BenchmarkingTest» со следующим содержимым:

```
{
    "deploymentCfgFile" : ".\\deploy_c6678_producer.json",
    "LoadImageName"     : "srio_c6678_producer.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"
}
```

Создайте файл «deploy_c6678_producer.json» в папке «D:/Dev/Modules/SVP-465/CCS_Workspace/SRIO_-BenchmarkingTest» со следующим содержимым:

```
{
    "deviceName" : "C6678",
    "partitions" : [
        {
            "name"  : "load-partition",
            "vaddr" : "0x9e000000",
            "size"  : "0x2000000",
            "loadPartition" : true
        }
    ],
    "applications" : [
        {
            "name" : "srio",
            "fileName" : "./C6678_Producer/srio_c6678_producer.out",
            "allowedCores" : [0,1,2,3,4,5,6,7]
        }
    ],
    "appDeployment" : [
        "",
        "srio",
        "",
        "",
        "",
        "",
        "",
        ""
    ]
}
```

Запустите MAP Tool выполнив команду:

```
python "C:\ti\mcsdk_2_01_02_06\tools\boot_loader\mad-utils\map-tool\maptool.py" "
 ← maptool_c6678_producer.json" bypass-prelink
```

В папку «images» будет помещен файл «srio_c6678_producer.bin», который можно записывать в NOR флеш память или выполнять загрузку данного файла по Ethernet с TFTP сервера.

При загрузке образа, приложение будет запущено только на 2-м ядре процессора.

Приложение А Примеры работы скрипта записи EEPROM и NOR флеш памяти

В листинге А-1 приведен пример вывода в терминал запуска скрипта «svp465_program.bat» для записи в NOR флеш память образов по умолчанию на все четыре процессора модуля SVP-465.

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 «svp465_program.bat NOR all»

```
D:\Dev\Modules\SVP-465\Program>svp465_program NOR all

SVP-465 program script started
Copyright (c) 2014, Scan Engineering Telecom SPb

Writing to DSP1 ENABLED
Writing to DSP2 ENABLED
Writing to DSP3 ENABLED
Writing to DSP4 ENABLED

Target configuration file:
  "../TargetConfigurations/SVP-465-USB560v2.ccxml"

Images for DSP's:
  DSP1: "bin/nor_c6670.bin" (4675295 bytes)
  DSP2: "bin/nor_c6678.bin" (4665508 bytes)
  DSP3: "bin/nor_c6670.bin" (4675295 bytes)
  DSP4: "bin/nor_c6678.bin" (4665508 bytes)

Configuring debug server for SVP-465 board...
Done!
Opening a debug session for 4 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_c6670.bin" (4675295 bytes)...
  Loading file "bin/nor_c6678.bin" (4665508 bytes)...
  Loading file "bin/nor_c6670.bin" (4675295 bytes)...
  Loading file "bin/nor_c6678.bin" (4665508 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:    4665508 bytes
Start address: 0x0

Flashing sector 0 (0 bytes of 4665508)
NOR Writer Utility Version 02.00.00.00
Copyright (c) 2014, Scan Engineering Telecom SPb

Write size:    4675295 bytes
Start address: 0x0

Flashing sector 0 (0 bytes of 4675295)
NOR Writer Utility Version 02.00.00.00
NOR Writer Utility Version 02.00.00.00
Flashing sector 1 (65536 bytes of 4665508)
Copyright (c) 2014, Scan Engineering Telecom SPb
Copyright (c) 2014, Scan Engineering Telecom SPb


Write size:    4675295 bytes
```

```
Write size:    4665508 bytes
Start address: 0x0
Start address: 0x0


Flashing sector 0 (0 bytes of 4675295)
Flashing sector 0 (0 bytes of 4665508)
Flashing sector 1 (65536 bytes of 4675295)
Flashing sector 2 (131072 bytes of 4665508)
Flashing sector 1 (65536 bytes of 4665508)
Flashing sector 1 (65536 bytes of 4675295)

ВЫРЕЗАНО: Вырезаны аналогичные сообщения для других секторов

Flashing sector 67 (4390912 bytes of 4665508)
Flashing sector 71 (4653056 bytes of 4665508)
Flashing sector 69 (4521984 bytes of 4675295)
Flashing sector 67 (4390912 bytes of 4675295)
Reading and verifying sector 0 (0 bytes of 4665508)
Flashing sector 68 (4456448 bytes of 4665508)
Flashing sector 70 (4587520 bytes of 4675295)
Reading and verifying sector 1 (65536 bytes of 4665508)
Reading and verifying sector 2 (131072 bytes of 4665508)
Reading and verifying sector 3 (196608 bytes of 4665508)
Reading and verifying sector 4 (262144 bytes of 4665508)
Reading and verifying sector 5 (327680 bytes of 4665508)
Reading and verifying sector 6 (393216 bytes of 4665508)
Flashing sector 68 (4456448 bytes of 4675295)
Flashing sector 69 (4521984 bytes of 4665508)
Reading and verifying sector 7 (458752 bytes of 4665508)
Flashing sector 71 (4653056 bytes of 4675295)
Reading and verifying sector 8 (524288 bytes of 4665508)
Reading and verifying sector 9 (589824 bytes of 4665508)
Reading and verifying sector 10 (655360 bytes of 4665508)
Reading and verifying sector 11 (720896 bytes of 4665508)
Reading and verifying sector 12 (786432 bytes of 4665508)
Reading and verifying sector 13 (851968 bytes of 4665508)
Flashing sector 69 (4521984 bytes of 4675295)
Flashing sector 70 (4587520 bytes of 4665508)
Reading and verifying sector 14 (917504 bytes of 4665508)
Reading and verifying sector 0 (0 bytes of 4675295)
Reading and verifying sector 15 (983040 bytes of 4665508)
Reading and verifying sector 1 (65536 bytes of 4675295)
Reading and verifying sector 16 (1048576 bytes of 4665508)
Reading and verifying sector 2 (131072 bytes of 4675295)
Reading and verifying sector 17 (1114112 bytes of 4665508)
Reading and verifying sector 3 (196608 bytes of 4675295)
Reading and verifying sector 18 (1179648 bytes of 4665508)
Reading and verifying sector 4 (262144 bytes of 4675295)
Reading and verifying sector 19 (1245184 bytes of 4665508)
Flashing sector 70 (4587520 bytes of 4675295)
Reading and verifying sector 5 (327680 bytes of 4675295)
Flashing sector 71 (4653056 bytes of 4665508)
Reading and verifying sector 20 (1310720 bytes of 4665508)
Reading and verifying sector 6 (393216 bytes of 4675295)
Reading and verifying sector 21 (1376256 bytes of 4665508)
Reading and verifying sector 7 (458752 bytes of 4675295)
Reading and verifying sector 22 (1441792 bytes of 4665508)
Reading and verifying sector 8 (524288 bytes of 4675295)
Reading and verifying sector 23 (1507328 bytes of 4665508)
Reading and verifying sector 9 (589824 bytes of 4675295)
Reading and verifying sector 24 (1572864 bytes of 4665508)
Reading and verifying sector 10 (655360 bytes of 4675295)
Flashing sector 71 (4653056 bytes of 4675295)
Reading and verifying sector 25 (1638400 bytes of 4665508)
Reading and verifying sector 11 (720896 bytes of 4675295)
Reading and verifying sector 0 (0 bytes of 4665508)
Reading and verifying sector 26 (1703936 bytes of 4665508)

ВЫРЕЗАНО: Вырезаны аналогичные сообщения для других секторов

Reading and verifying sector 71 (4653056 bytes of 4665508)
Reading and verifying sector 56 (3670016 bytes of 4675295)
Reading and verifying sector 41 (2686976 bytes of 4675295)
Reading and verifying sector 46 (3014656 bytes of 4665508)
NOR programming completed successfully
Reading and verifying sector 57 (3735552 bytes of 4675295)
Reading and verifying sector 42 (2752512 bytes of 4675295)
```

```
Reading and verifying sector 47 (3080192 bytes of 4665508)
Reading and verifying sector 58 (3801088 bytes of 4675295)

ВЫРЕЗАНО: Вырезаны аналогичные сообщения для других секторов

Reading and verifying sector 60 (3932160 bytes of 4665508)
Reading and verifying sector 71 (4653056 bytes of 4675295)
Reading and verifying sector 56 (3670016 bytes of 4675295)
Reading and verifying sector 61 (3997696 bytes of 4665508)
NOR programming completed successfully
Reading and verifying sector 57 (3735552 bytes of 4675295)
Reading and verifying sector 62 (4063232 bytes of 4665508)
Reading and verifying sector 58 (3801088 bytes of 4675295)
Reading and verifying sector 63 (4128768 bytes of 4665508)

ВЫРЕЗАНО: Вырезаны аналогичные сообщения для других секторов

Reading and verifying sector 70 (4587520 bytes of 4665508)
Reading and verifying sector 66 (4325376 bytes of 4675295)
Reading and verifying sector 71 (4653056 bytes of 4665508)
Reading and verifying sector 67 (4390912 bytes of 4675295)
NOR programming completed successfully
Reading and verifying sector 68 (4456448 bytes of 4675295)
Reading and verifying sector 69 (4521984 bytes of 4675295)
Reading and verifying sector 70 (4587520 bytes of 4675295)
Reading and verifying sector 71 (4653056 bytes of 4675295)
NOR programming completed successfully
Done!
Terminating debug sessions on DSP(s)...
Done!
Stopping debug server...
Done!

D:\Dev\Modules\SVP-465\Program>
```

В листинге A-2 приведен пример вывода в терминал запуска скрипта «svp465_program.bat» для записи в EEPROM память образов по умолчанию на все четыре процессора модуля SVP-465.

Листинг A-2: Вывод в терминал при запуске команды «svp465_program.bat EEPROM all»

```
D:\Dev\Modules\SVP-465\Program>svp465_program EEPROM all

SVP-465 program script started
Copyright (c) 2014, Scan Engineering Telecom SPb

Writing to DSP1 ENABLED
Writing to DSP2 ENABLED
Writing to DSP3 ENABLED
Writing to DSP4 ENABLED

Target configuration file:
  "../TargetConfigurations/SVP-465-USB560v2.ccxml"

Images for DSP's:
  DSP1: "bin/i2crom_0x50_svp465_c6670_le.bin" (57252 bytes)
  DSP2: "bin/i2crom_0x50_svp465_c6678_le.bin" (57084 bytes)
  DSP3: "bin/i2crom_0x50_svp465_c6670_le.bin" (57252 bytes)
  DSP4: "bin/i2crom_0x50_svp465_c6678_le.bin" (57084 bytes)

Configuring debug server for SVP-465 board...
Done!
Opening a debug session for 4 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/i2crom_0x50_svp465_c6670_le.bin" (57252 bytes)...
  Loading file "bin/i2crom_0x50_svp465_c6678_le.bin" (57084 bytes)...
  Loading file "bin/i2crom_0x50_svp465_c6670_le.bin" (57252 bytes)...
  Loading file "bin/i2crom_0x50_svp465_c6678_le.bin" (57084 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:        57084 bytes
I2C bus address:   0x50
I2C start address: 0x0
Swap data:         no

Writing 57084 bytes from DSP memory address 0x0c000100 to EEPROM (0x0050)...
EEPROM Writer Utility Version 02.00.00.00
EEPROM Writer Utility Version 02.00.00.00
EEPROM Writer Utility Version 02.00.00.00
Copyright (c) 2014, Scan Engineering Telecom SPb
Copyright (c) 2014, Scan Engineering Telecom SPb
Copyright (c) 2014, Scan Engineering Telecom SPb



Write size:        57252 bytes
Write size:        57084 bytes
Write size:        57252 bytes
I2C bus address:   0x50
I2C bus address:   0x50
I2C bus address:   0x50
I2C start address: 0x0
I2C start address: 0x0
I2C start address: 0x0
Swap data:         no
Swap data:         no
Swap data:         no



Writing 57252 bytes from DSP memory address 0x0c000100 to EEPROM (0x0050)...
Writing 57084 bytes from DSP memory address 0x0c000100 to EEPROM (0x0050)...
```

```
Writing 57252 bytes from DSP memory address 0x0c000100 to EEPROM (0x0050)...
Reading 57084 bytes from EEPROM (0x0050) to DSP memory address 0x0c010100...
Reading 57252 bytes from EEPROM (0x0050) to DSP memory address 0x0c010100...
Reading 57084 bytes from EEPROM (0x0050) to DSP memory address 0x0c010100...
Reading 57252 bytes from EEPROM (0x0050) to DSP memory address 0x0c010100...
Verifying data read...
EEPROM programming completed successfully
Verifying data read...
Verifying data read...
Verifying data read...
EEPROM programming completed successfully
EEPROM programming completed successfully
EEPROM programming completed successfully
Done!
Terminating debug sessions on DSP(s)...
Done!
Stopping debug server...
Done!

D:\Dev\Modules\SVP-465\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).

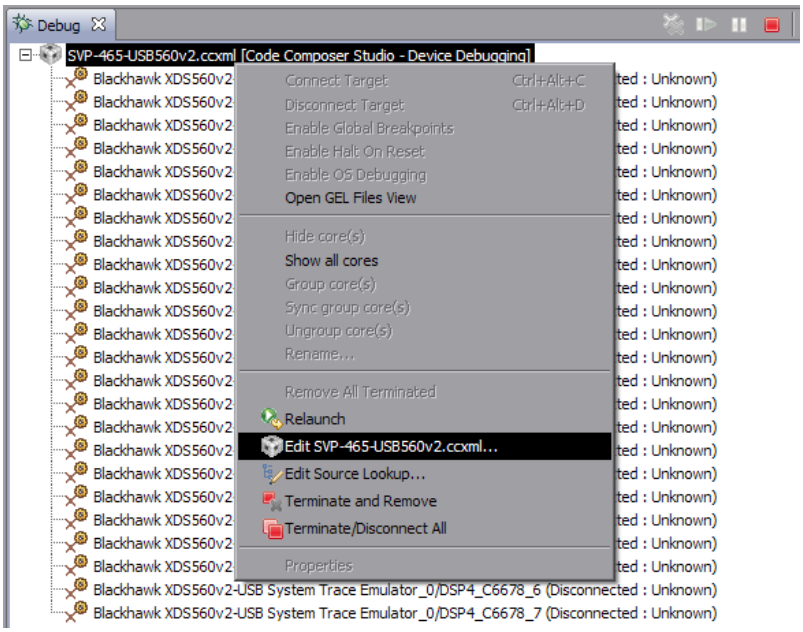


Рисунок Б-1: Контекстное меню целевой конфигурации

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».

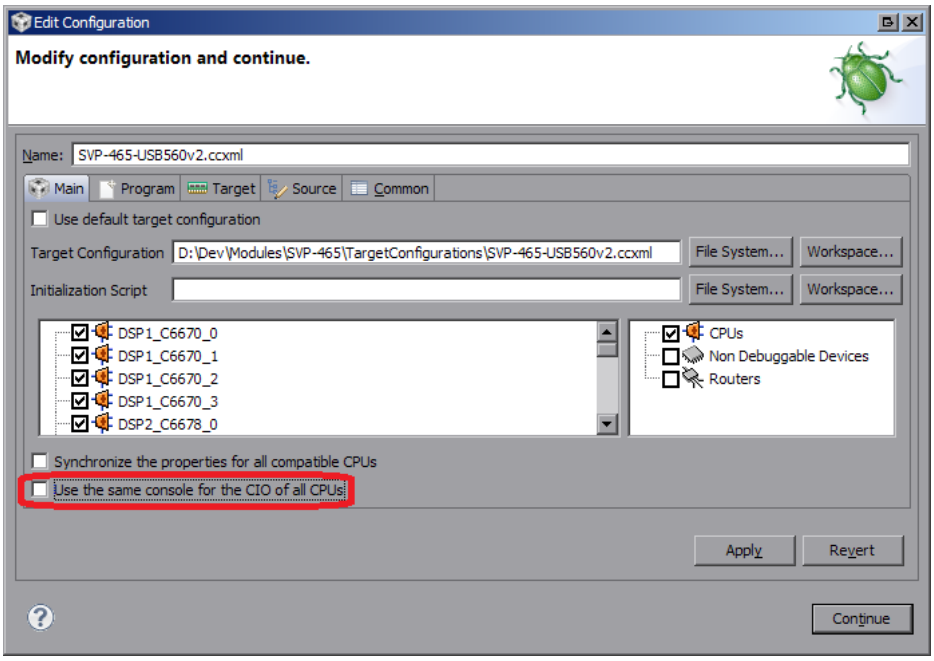


Рисунок Б-2: Окно настроек целевой конфигурации

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

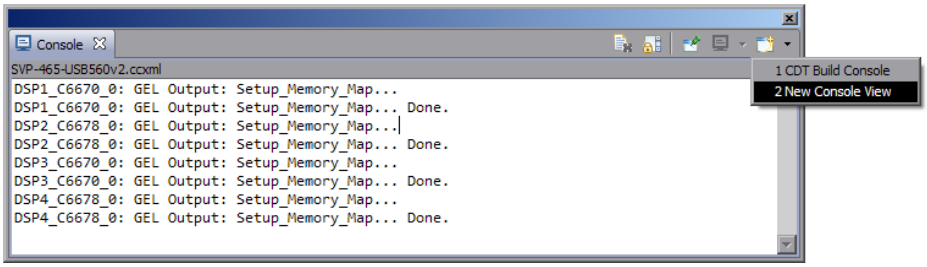


Рисунок Б-3: Открытие второго окна «Console»

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

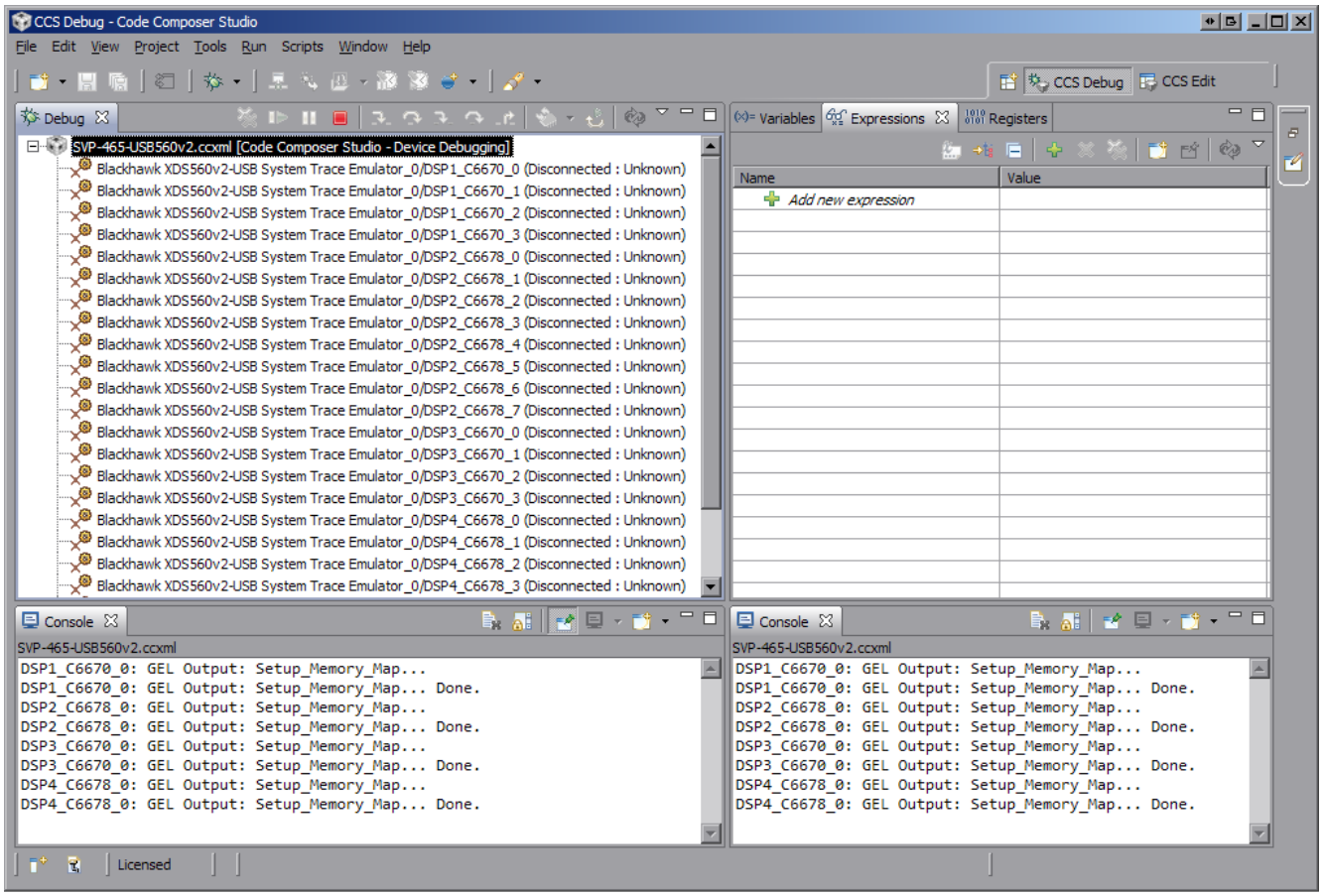


Рисунок Б-4: Два окна «Console»

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).

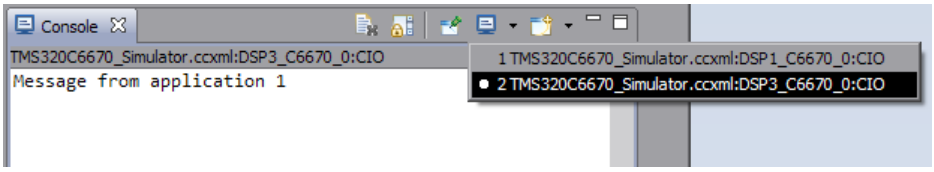


Рисунок Б-5: Выбор ядра для отображения вывода в окне «Console»

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

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 Б-5, а вывод в два окна будет выглядеть, как показано на рисунке Б-6.

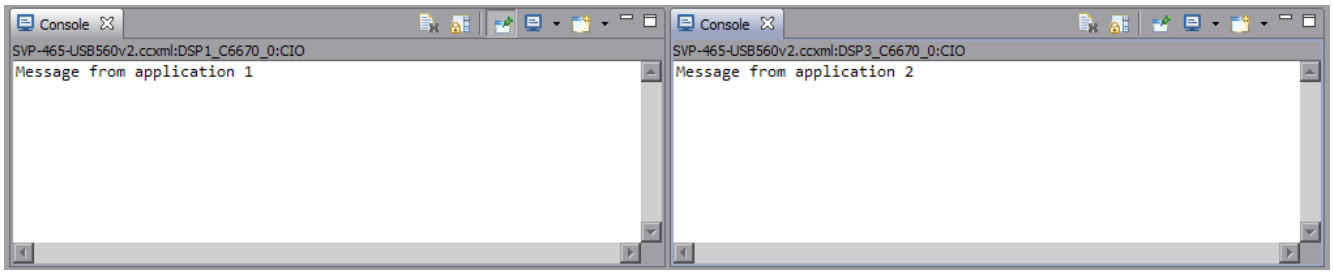


Рисунок Б-6: Вывод сообщений с двух ядер в два окна «Console»

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

Приложение B Выбор режима загрузки IBL

Выбор режима загрузки IBL осуществляется при помощи установки джамперов на переключатели SW1–SW8 на плате модуля SVP-465. Переключатели SW1–SW4 расположены на разъёме X16, а переключатели SW5–SW8 на разъёме X17 платы модуля SVP-465.

В таблице B-1 представлены положения переключателей SW1–SW8 для всех возможных режимов загрузки IBL для каждого из четырех процессоров модуля SVP-465.

Таблица B-1: Положение переключателей модуля SVP-465 для установки режимов загрузки IBL

Режим	Разъём X16				Разъём X17			
	DSP1_C6670		DSP2_C6678		DSP3_C6670		DSP4_C6678	
	SW1	SW2	SW3	SW4	SW5	SW6	SW7	SW8
NOBOOT (DEBUG)	Разомкнут	Разомкнут	Разомкнут	Разомкнут	Разомкнут	Разомкнут	Разомкнут	Разомкнут
NOR	Разомкнут	**Замкнут**	Разомкнут	**Замкнут**	Разомкнут	**Замкнут**	Разомкнут	**Замкнут**
TFTP (BOOTP)	**Замкнут**	Разомкнут	**Замкнут**	Разомкнут	**Замкнут**	Разомкнут	**Замкнут**	Разомкнут
PCI-E	**Замкнут**	**Замкнут**	**Замкнут**	**Замкнут**	**Замкнут**	**Замкнут**	**Замкнут**	**Замкнут**

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 режима PCI-E устанавливаются значения `BOOTMODE[7:0] = 00000100b` и `PCIESSMODE[1:0] = 00b` для соответствующего процессора.

Примечание

`BOOTMODE[7:0]` является частью регистра `DEVSTAT[8:1]`, а `PCIESSMODE[1:0]` является частью регистра `DEVSTAT[15:14]`. В процессорах TMS320C6670 и TMS320C6678 32-х битный регистр `DEVSTAT` доступен по адресу 0x02620020.

Список литературы

1. SVP-465. Сборка и запуск приложения веб-сервера. Руководство пользователя. UG-SVP-465-WEB (цит. на с. 14, 18).
2. Multicore Application Deployment (MAD) Utilities. Texas Instruments. URL: http://processors.wiki.ti.com/index.php/MAD_Utils_User_Guide (цит. на с. 36, 37).
3. Установка и настройка сервера сетевой загрузки (BOOTP и TFTP). Руководство пользователя. UG-CMN-BOOTP-TFTP (цит. на с. 40).
4. SVP-465. Сборка и запуск теста производительности Serial RapidIO. Руководство пользователя. UG-SVP-465-SRIO (цит. на с. 41).