



SVP-716. Загрузчик IBL

Руководство пользователя

Версия 1.0



Код документа: UG-SVP-716-IBL
Дата сборки: 27 мая 2015 г.
Листов в документе: 43

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

Содержание

Список рисунков	3
Список таблиц	3
Список листингов	3
Список процедур	3
Перечень сокращений и условных обозначений	5
1 Общие сведения	6
2 Сборка образа IBL из исходных кодов	7
2.1 Установка MinGW в Windows системе	7
2.2 Конфигурация окружения сборки	10
2.3 Сборка загрузчика	10
3 Скрипт для записи EEPROM и NOR флеш памяти модуля SVP-716	12
3.1 Структура каталога «Program»	13
3.2 Работа со скриптом записи NOR флеш и EEPROM памяти	15
3.3 Запись образов в NOR флеш память	16
3.4 Запись образов в EEPROM память	21
4 Конфигурация IBL	23
5 Импорт и запуск целевой конфигурации модуля	30
6 Подготовка образов приложений для загрузки на нескольких ядрах процессора	33
6.1 Компоненты MAD Utils	33
6.2 Режимы работы MAD Utils	34
6.3 Работа с MAD Utils в режиме «Prelinker bypass mode»	34
6.4 Конфигурация MAP Tool	35
6.5 Конфигурационный файл развертывания	35
6.6 Запуск MAP Tool	37
6.7 Загрузка образа	37
Приложение А: Примеры работы скрипта записи EEPROM и NOR флеш памяти	38
Приложение Б: Выбор режима загрузки IBL	42
Список литературы	43

Список рисунков

2-1	Установка MinGW. Выбор каталога репозитариев	7
2-2	Установка MinGW. Окно согласия с лицензией	8
2-3	Установка MinGW. Выбор пути установки	8
2-4	Установка MinGW. Выбор устанавливаемых компонентов	9
2-5	Приглашение командной строки MinGW Shell	9
2-6	Редактирование файла «setupenvMsys.sh» в редакторе vim	10
3-1	Состояния светодиодов на передней панели модуля SVP-716 во время записи	17
3-2	Организация EEPROM памяти на модуле SVP-716	21
4-1	Подключение к процессору «DSP1_C6670»	23
4-2	Окно загрузки кода на ядро процессора	24
4-3	Внешний вид окна «Debug» после загрузки кода на ядра процессоров	24
4-4	Внешний вид окна «Console» после запуска кода на процессоре	24
4-5	Пункт главного меню для вызова окна управления GEL файлами	25
4-6	Окно управления GEL файлами	25
4-7	Окно управления GEL файлами с загруженным файлом «i2cConfig.gel»	25
4-8	Пункт главного меню «Scripts > SET SVP-716»	26
4-9	Внешний вид окна «Console» после выполнения записи конфигурации в EEPROM память	26
5-1	Пункт меню для отображения окна целевых конфигураций	30
5-2	Меню импорта целевой конфигурации	30
5-3	Окно выбора файла для импорта целевой конфигурации	31
5-4	Окно выбора способа импорта файла целевой конфигурации	31
5-5	Запуск целевой конфигурации	31
5-6	Список ядер процессора модуля SVP-716	32
6-1	Схема работы MAD в режиме «Prelinker bypass mode»	34

Список таблиц

2-1	Файлы, создаваемые при сборке загрузчика	10
3-1	Параметры командной строки скрипта «svp716_program.js»	15
4-1	Основные конфигурационные параметры файла «i2cConfig.gel»	28
6-1	Параметры конфигурационного файла MAP Tool	35
6-2	Параметры файла конфигурации развертывания	36
Б-1	Положение переключателей модуля SVP-716 для установки режимов загрузки IBL	42

Список листингов

3-1	Скрипт «svp716_program.bat»	12
3-2	Скрипт «svp716_program.sh»	12
3-3	Структура каталога «Program»	13
3-4	Вывод скрипта «svp716_program.bat», запущенного без параметров	15
3-5	Вывод в UART загрузки демонстрационного приложения веб-сервера с NOR флеш памяти	18
3-6	Вывод в UART при запуске теста платформы	18
4-1	Фрагмент GEL файла «i2cConfig.gel» с конфигурационными параметрами загрузчика IBL для модуля SVP-716	26
6-1	Пример конфигурационного файла для MAP Tool	35
6-2	Пример файла конфигурации развертывания	36
A-1	Вывод в терминал при запуске команды «svp716_program.bat NOR all»	38
A-2	Вывод в терминал при запуске команды «svp716_program.bat EEPROM all»	40

Список процедур

3-1	Запись образов в NOR флеш память	16
3-2	Запись образа демонстрационного приложения веб-сервера в NOR флеш память и его загрузка	16
3-3	Запись образов в EEPROM память	21
3-4	Запись образа загрузка IBL в EEPROM память и его загрузка	21

Перечень сокращений и условных обозначений

BOOTP	Bootstrap Protocol	28, 29, 37
CCS	Code Composer Studio	6, 7, 10–14, 16, 32
CGT	Code Generation Tools	35
CIO	Console Input/Output	14, 15
COFF	Common Object File Format	29
DDR	Double Data Rate	13, 16, 33, 35
DSO	Dynamic Shared Object	33
DSS	Debug Server Scripting	11, 12
EEPROM	Electrically Erasable Programmable Read-Only Memory	2–4, 6, 7, 10–15, 17, 18, 21–24, 26, 34, 38, 40
ELF	Executable and Linkable Format	29, 33
GEL	General Extension Language	3, 10, 24–26
I²C	Inter-Integrated Circuit	7, 15, 18, 21, 34
IBL	Intermediate Boot Loader	2–4, 6, 7, 9–11, 13, 16–18, 21–24, 26, 28, 29, 33–35, 37, 42
IP	Internet Protocol	29
JSON	Java Script Object Notation	35, 36
MAC	Media Access Control	29
MAD	Multicore Application Deployment	2, 3, 33–35, 37
MAP	Multiple Application Pre-linker	2, 33–37
MCSDK	MultiCore Software Development Kit	33, 35, 37
NML	No Man's Land	35
NOR	Not OR	2–4, 6, 12–18, 22, 35, 37, 38
OFD	Object File Dump	35
ROMFS	Read-Only Memory File System	33–36
ROM	Read-Only Memory	34
TFTP	Trivial File Transfer Protocol	6, 35, 37
TI	Texas Instruments	7
UART	Universal Asynchronous Receiver-Transmitter	3, 13, 16, 18
ОС	Операционная Система	6

1 Общие сведения

Загрузчик IBL (Intermediate Boot Loader) позволяет выполнять загрузку приложений на процессоры модуля SVP-716 с NOR флеш памяти или по Ethernet с TFTP сервера в локальной сети. Конкретный режим загрузки выбирается при помощи переключателей на плате модуля SVP-716 отдельно для каждого процессора (см. приложение Б).

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-716 (раздел 2), описана процедура записи загрузчика IBL в EEPROM модуля SVP-716 (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-716 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 сайта¹, где доступны версии CCS для Windows и Linux систем. На сопроводительном диске к модулю SVP-716 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-716, 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 к записи в I²C EEPROM модуля SVP-716.

Внимание



Перед выполнением сборки загрузчика, описанной в данном разделе, перепишите с сопроводительного диска к модулю SVP-716 папку «ibl» на жесткий диск компьютера. Далее, предполагается, что все содержимое папки «ibl» с сопроводительного диска переписано в папку «D:/Dev/Modules/SVP-716/IBL».

Для сборки загрузчика IBL в Windows системе, кроме компилятора C6000 процессоров, необходима GNU система сборки MinGW. Скачать последнюю версию MinGW можно на официальном сайте¹.

В данном документе описана работа с MinGW версии 20120426. Установочный дистрибутив MinGW версии 20120426 можно найти на сопроводительном диске к модулю SVP-716 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-716 (файл «Install/mingw-get-inst-20120426.exe»).

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

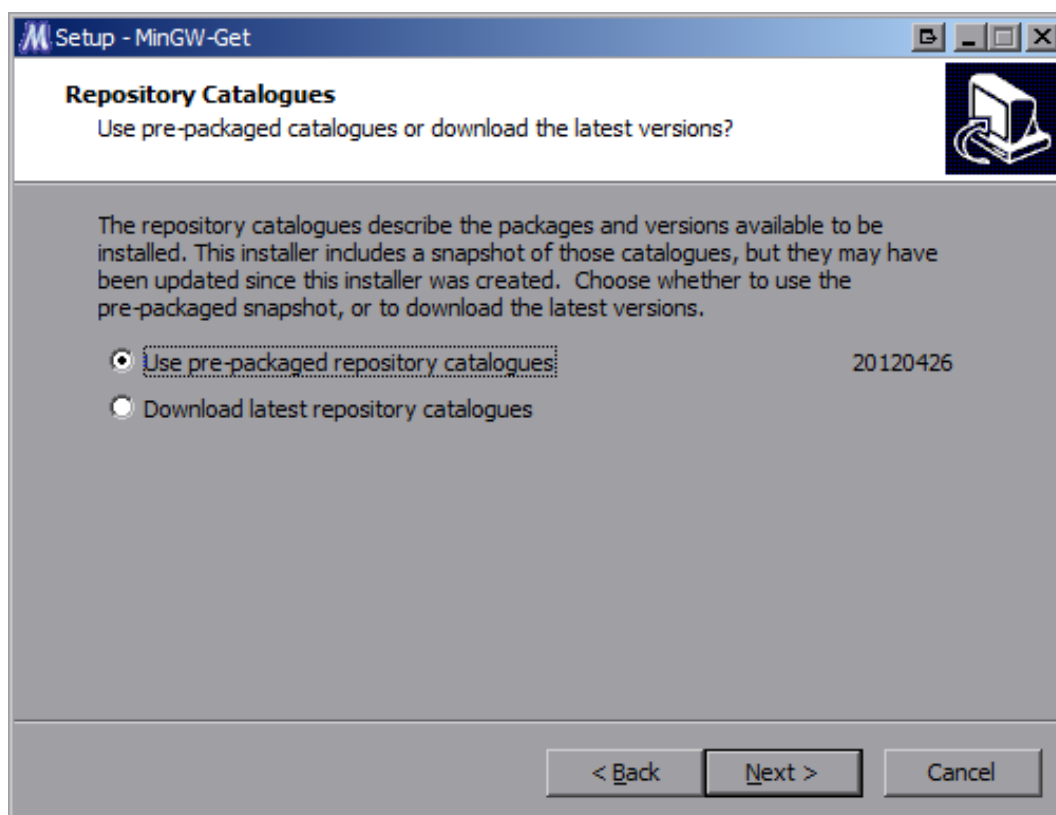


Рисунок 2-1: Установка MinGW. Выбор каталога репозитория

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».

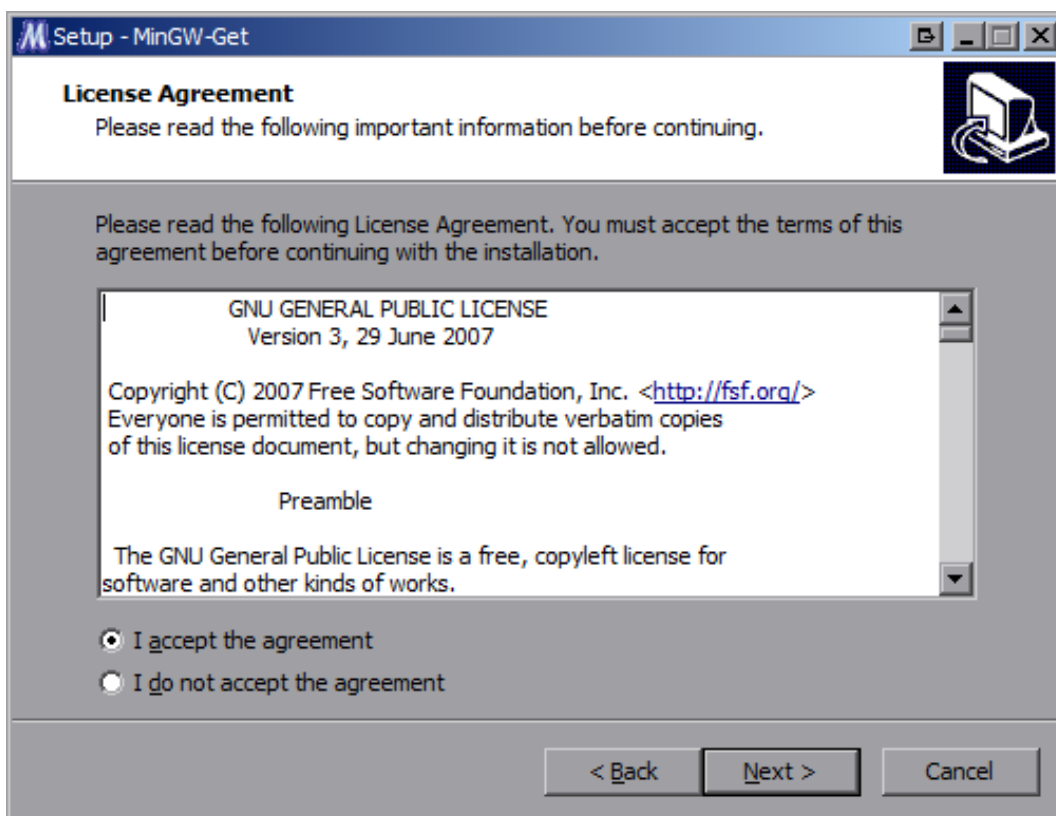


Рисунок 2-2: Установка MinGW. Окно согласия с лицензией

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

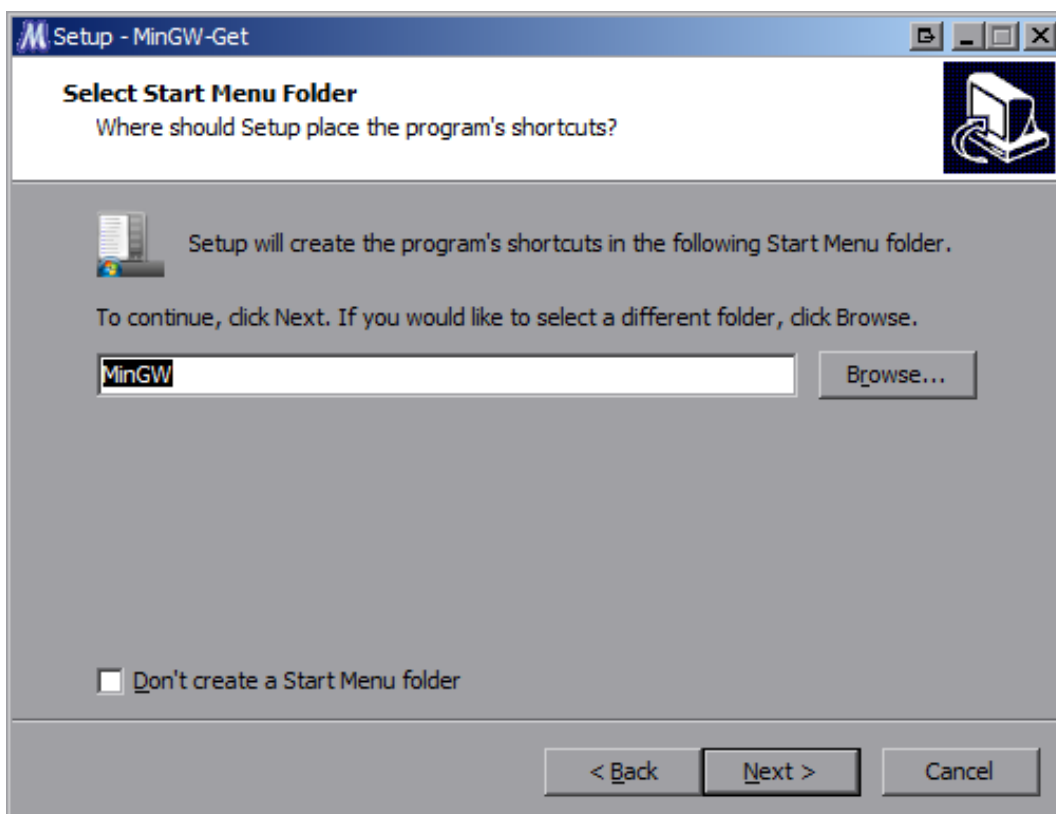


Рисунок 2-3: Установка MinGW. Выбор пути установки

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

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

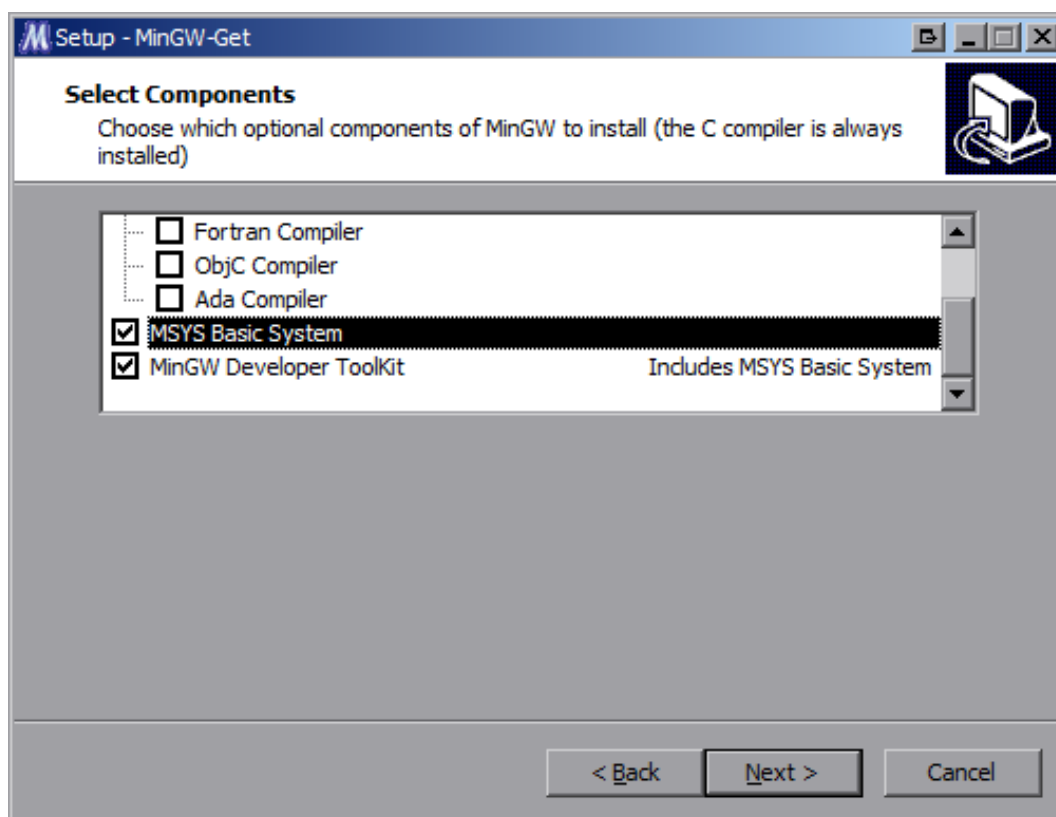


Рисунок 2-4: Установка MinGW. Выбор устанавливаемых компонентов

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.

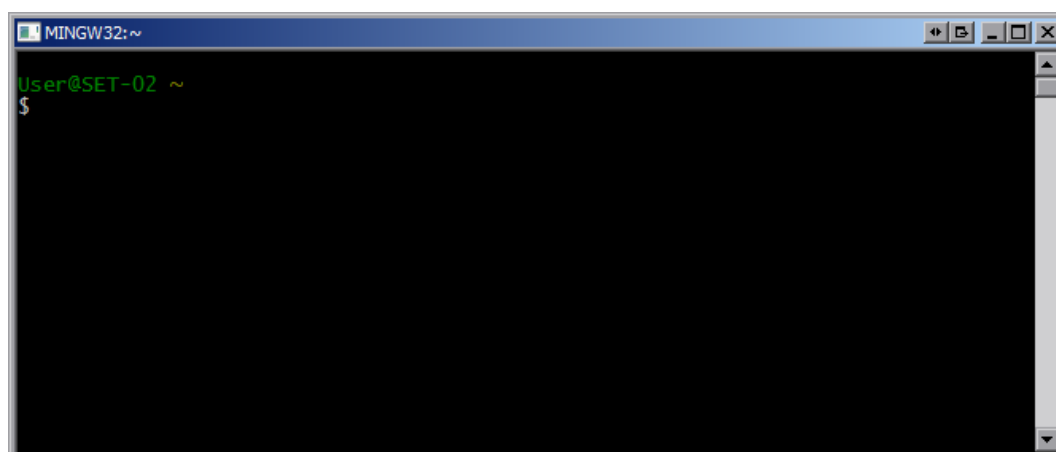


Рисунок 2-5: Приглашение командной строки MinGW Shell

2.2 Конфигурация окружения сборки

В данном разделе предполагается, что исходные коды загрузчика IBL переписаны с сопроводительного диска к модулю SVP-716 в папку «D:/Dev/Modules/SVP-716/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-716/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.

```

MINGW32
#!/bin/bash

# Environment setup to be done if using MSYS Bash shell for build

# Specify the base directory of the c6000 compiler with UNIX style path separator
export C6X_BASE_DIR="/C:/ti/ccsv5/tools/compiler/c6000_7.4.2"

# Specify the base directory of the c6000 compiler in format understandable by the MSYS Bash shell
export C6X_BASE_DIR_MSYS=/c/ti/ccsv5/tools/compiler/c6000_7\4.2

# Don't modify the below variables. They are derived from the above definitions
export PATH=$PATH:$C6X_BASE_DIR_MSYS/bin
export TOOLSC6X=$C6X_BASE_DIR
export TOOLSC6XD0S=$C6X_BASE_DIR

"setupenvMsys.sh" 16 lines, 580 characters

```

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-716/IBL/src/make/setupenvMsys.sh
```

В файле «D:/Dev/Modules/SVP-716/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-716, выполните в MinGW Shell последовательно следующие команды:

```
cd /d/Dev/Modules/SVP-716/IBL/src/make/
source setupenvMsys.sh
make svp716
```

После выполнения этих команд, будет запущена сборка загрузчика IBL для модуля SVP-716, которая может занять до 30 минут (в зависимости от производительности системы).

В таблице 2-1 перечислены файлы, которые создаются в папке «D:/Dev/Modules/SVP-716/IBL/src/make/bin» после выполнения сборки загрузчика IBL для модуля SVP-716.

Таблица 2-1: Файлы, создаваемые при сборке загрузчика

Файл	Описание
«i2cConfig.gel»	GEL файл конфигурации IBL для CCS (см. раздел 4)
«i2cparam_0x50_svp716_le_0x500.out»	Бинарный образ программы конфигурации IBL для процессора TMS320C6670 модуля SVP-716 (см. раздел 4)
«i2crom_0x50_svp716_le.bin»	Бинарный образ IBL для процессора TMS320C6670 для загрузки в EEPROM память модуля SVP-716 (см. раздел 3.4)

Продолжение таблицы на следующей странице

Продолжение таблицы 2-1

Файл	Описание
«i2crom_0x50_svp716_le.dat»	Бинарный образ IBL для процессора TMS320C6670 модуля SVP-716 в формате CCS.

Для записи в EEPROM память модуля SVP-716 предназначен файл «i2crom_0x50_svp716_le.bin».

Запись образа загрузчика в EEPROM память производится 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-716

Для записи образов в EEPROM или NOR флеш память процессоров модуля SVP-716 предназначен специальный DSS скрипт «svp716_program.js», который расположен в папке «Program» на сопроводительном диске к модулю SVP-716.

Перед использованием скрипта «svp716_program.js» для записи EEPROM или NOR флеш памяти, необходимо скопировать с сопроводительного диска к модулю SVP-716 папки «Program», «TargetConfigurations» и «GEL» со всем содержимым в папку «D:/Dev/Modules/SVP-716»¹.

Скрипт «svp716_program.js» требует для своей работы установленную систему разработки CCS. Дистрибутив сетевой установки CCS имеется на сопроводительном диске к модулю SVP-716 (см. раздел 1).

Для запуска скрипта «svp716_program.js» в папке «Program» имеются вспомогательные скрипты запуска для Windows и Linux систем:

- «svp716_program.bat»: скрипт для запуска в Windows системе (см. листинг 3-1);
- «svp716_program.sh»: скрипт для запуска в Linux системе (см. листинг 3-2).

Листинг 3-1: Скрипт «svp716_program.bat»

```
1 @echo off
2 set PROGRAM_SVP_716_TARGET_CONFIG_FILE=../TargetConfigurations/SVP-716-LAN560v2.ccxml
3 set DSS_SCRIPT_DIR="C:\ti\ccsv5\ccs_base\scripting\bin"
4 call %DSS_SCRIPT_DIR%\dss.bat svp716_program.js %*
```

Листинг 3-2: Скрипт «svp716_program.sh»

```
1 #!/bin/sh
2 export PROGRAM_SVP_716_TARGET_CONFIG_FILE=../TargetConfigurations/SVP-716-LAN560v2.ccxml
3 export DSS_SCRIPT_DIR=~/.ti/ccsv5/ccs_base/scripting/bin
4 $DSS_SCRIPT_DIR/dss.sh svp716_program.js $@
```

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

- «PROGRAM_SVP_716_TARGET_CONFIG_FILE»: путь к файлу описания целевой конфигурации модуля SVP-716;
- «DSS_SCRIPT_DIR»: путь к бинарным файлам DSS среды разработки CCS.

Важная информация



Только для Windows: В скрипте «svp716_program.bat» значение переменной «DSS_SCRIPT_DIR» обязательно должно быть записано в кавычках, а значение переменной «PROGRAM_SVP_716_TARGET_CONFIG_FILE», наоборот, не должно содержать кавычек.

После определения значений для переменных окружения, в скрипте происходит вызов скрипта кроссплатформенного DSS скрипта «svp716_program.js», с передачей ему аргументов командной строки.

Примечание

В данном документе рассматривается работа со скриптом «svp716_program.bat» для Windows системы. Для выполнения аналогичных действий в Linux системе, достаточно заменить вызовы скрипта «svp716_program.bat» на вызовы скрипта «svp716_program.sh».

В разделе 3.1 описывается структура каталога «Program», в котором работает скрипт «svp716_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-716, в котором располагаются все необходимые файлы для выполнения записи в EEPROM или NOR флеш память процессоров модуля SVP-716.

Листинг 3-3: Структура каталога «Program»

```
Program
+- bin
| +- ibl
| | +- i2crom_0x50_svp716_le.bin
| | +- i2cConfig.gel
| | +- i2cparam_0x50_svp716_le_0x500.out
| |
| +- platform_test
| | +- platform_test_svp716.bin
| |
| +- srio
| | +- srio_svp716.bin
| |
| +- webservice
| | +- webservice_svp716.bin
| |
| +- eeprom_0x50_c6670.bin
| +- nor_c6670.bin
|
+- logs
|
+- writers
| +- eepromwriter_c6670.out
| +- norwriter_c6670.out
|
+- svp716_program.bat
+- svp716_program.sh
+- svp716_program.js
```

Кроме файлов скриптов «svp716_program.bat», «svp716_program.sh» и «svp716_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-716. Пользовательские файлы, подготовленные для записи в EEPROM или NOR флеш память, должны быть помещены в каталог «bin».

Непосредственно в каталоге «bin» расположены два файла, которые используются скриптом «svp716_program.js» для записи по умолчанию:

- «eeprom_0x50_c6670.bin»: образ для записи в EEPROM память процессора TMS320C6670. По умолчанию, данный файл представляет собой собранный образ загрузчика IBL, который можно получить путем выполнения шагов, описанных в разделе 2 данного документа.
- «nor_c6670.bin»: образ для записи в NOR флеш память процессора TMS320C6670. По умолчанию, данный файл представляет собой собранный образ демонстрационного приложения веб-сервера. Подробное описание по сборке и запуску демонстрационного приложения веб-сервера дано в документе [1].

В каталоге «bin/platform_test» размещен образ приложения теста платформы (файл «platform_test_svp716.out»), 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.

Исходные коды приложения теста платформы находятся в проекте «PlatformTest_C6670» рабочего пространства CCS. Папку с рабочим пространством можно найти на сопроводительном диске к модулю SVP-716 в папке «CCS_Workspace».

В каталоге «writers» расположены файлы программ, для осуществления записи в EEPROM и NOR флеш память процессора. Данные файлы получены путем сборки проектов «EEPROM_Writer» (файл «eepromwriter_c6670.out») и NOR_Writer» (файл «norwriter_c6670.out») рабочего пространства CCS для модуля SVP-716. 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-716 осуществляется из командной строки. Выбор того или иного режима работы скрипта осуществляется при помощи параметров командной строки, передаваемых скрипту при запуске.

Если запустить скрипт «svp716_program.bat» без параметров, будет выдана краткая справка по возможным параметрами командной строки (см. листинг 3-4).

Листинг 3-4: Вывод скрипта «svp716_program.bat», запущенного без параметров

```

1 D:\Dev\Modules\SVP-716\Program>svp716_program.bat
2
3 Syntax:  svp716_program.js <NOR|EEPROM> <options>
4
5 Mode:
6   NOR    : program NOR flash memory
7   EEPROM : program I2C EEPROM memory
8
9 Options:
10  image=<image>      : image file for all DSP's
11  bus=<address>      : I2C bus EEPROM address for all DSP's
12                    : Address must be a decimal number
13                    : Only for EEPROM mode. Default is 80 (0x50)
14  log               : Enable CIO logging
15
16 D:\Dev\Modules\SVP-716\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 «svp716_program.js»

Параметр	Описание
image=<image>	Параметр задает имя файла (<image>) образа для записи.
bus=<address>	Параметр позволяет указать адрес (<address>) I ² C шины EEPROM памяти для записи. 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 с процессора модуля SVP-716 в лог-файл, который будет помещен в каталог «logs». Создаваемые файлы лога имеют формат «SVP-716-DSP<n>-<time>-cio.txt», где <n> — номер процессора модуля SVP-716 (всегда 1), <time> — время запуска скрипта «svp716-program.js». По умолчанию, запись вывода CIO в файлы отключена.

Важная информация



Значения адреса шины для параметра bus обязательно должно 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-716 выполните шаги процедуры 3-1.

Внимание



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

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-716, 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-716/Program/bin».
3. Перейдите в каталог «D:/Dev/Modules/SVP-716/Program».
4. Запустите скрипт «svp716_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-716.

В качестве примера, в процедуре 3-2 подробно описаны шаги, необходимые для выполнения записи в NOR флеш память всех четырех процессоров модуля SVP-716 образа приложения демонстрационного веб-сервера [1], начиная от сборки приложения, и заканчивая его загрузкой с NOR флеш памяти при помощи загрузчика IBL.

Важная информация



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

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

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

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

В результате сборки приложения веб-сервера должен быть получен файл «D:/Dev/Modules/SVP-716/CCS_Workspace/WebServer/Debug/webserver_svp716.out»;

Примечание

Модуль SVP-716 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 файл, полученный при выполнении шага 1 в папку «D:/Dev/Modules/SVP-716/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-716
copy CCS_Workspace\WebServer\Debug\webserver_svp716.out Program\bin\webserver_svp716.bin
```

3. Установите для процессора модуля SVP-716 режим «NOBOOT» в соответствии с данными таблицы Б-1 приложения Б.
4. Подключите отладчик к модулю SVP-716 и к компьютеру (предполагается использование отладчика Spectrum Digital XDS560v2 STM LAN).
5. Включите модуль SVP-716.
6. Перейдите в каталог «D:/Dev/Modules/SVP-716/Program». Для этого необходимо выполнить в терминале команды:

```
d:
cd D:\Dev\Modules\SVP-716\Program
```

7. Запустите скрипт «svp716_program.bat» выполнив в терминале команду:

```
svp716_program.bat NOR image=webserv_svp716.bin
```

Будет запущен процесс записи образов в NOR флеш память процессоров модуля SVP-716. При этом, вывод в терминал должен быть аналогичен тому, что приведен в листинге А-1 приложения А. Весь процесс записи должен занимать не более 5 минут.

После вывода в терминал сообщения «Executing writers on DSP(s)...» на процессоре модуля SVP-716 начинается процесс записи данных образа в NOR флеш память. При этом, светодиоды «DSP1» и «DSP2» на передней панели модуля SVP-716 будут иметь определенное состояние в зависимости от хода процесса записи (см. рисунок 3-1).

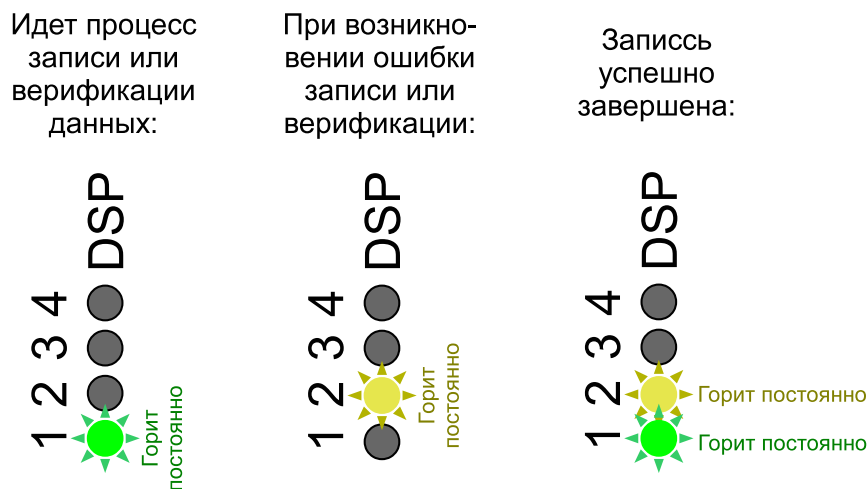


Рисунок 3-1: Состояния светодиодов на передней панели модуля SVP-716 во время записи

При нормальном процессе записи и верификации записанных данных, должен гореть только зеленый светодиод «DSP1». Желтый светодиод «DSP2» гореть не должен.

Если во время записи или верификации происходит ошибка, то загорается желтый светодиод «DSP2», зеленый светодиод «DSP1» гаснет (см. рисунок 3-1). При этом, работа программы записи прекращается.

В случае успешного завершения записи и верификации записанных данных, должны гореть оба светодиода «DSP1» и «DSP2».

8. Выключите модуль SVP-716.
9. Если в EEPROM память процессоров модуля SVP-716 еще не записан образ загрузчика IBL, выполните шаги процедуры 3-4.

Примечание

Модуль SVP-716 поставляется с уже записанным образом загрузчика IBL в EEPROM память (адрес I²C шины 0x50). Поэтому, если в EEPROM память (адрес I²C шины 0x50) не производилась запись каких либо других образов, выполнять запись загрузчика не требуется.

10. Установите для процессора модуля SVP-716 режим «NOR» в соответствии с данными таблицы Б-1 приложения Б.
11. Включите модуль SVP-716.

После выполнения всех шагов процедуры 3-2, вывод в UART с процессоров модуля SVP-716 при его включении должен выглядеть подобно приведенному в листинге 3-5. В листинге 3-5 приведен вывод в UART загрузки демонстрационного приложения веб-сервера.

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

```

1
2 IBL INIT
3 I2C boot
4 EDC enabled
5 Boot table processor executed
6
7 IBL version: 1.0.0.16-5
8 Device is SVP-716
9 Initializing GbE Switch Subsystem...
10 GbE Switch Subsystem initialized
11 00:17:ea:e8:f1:a4
12 IBL: PLL and DDR Initialization Complete
13 IBL Result code 00
14 Multiboot enabled
15 SPI initialized
16 IBL: Booting from NOR
17 Booting from NOR flash from address
18   = 0x8157c8e0
19 QMSS successfully initialized
20 CPPI successfully initialized
21 PA successfully initialized
22 HUA version 2.00.00.04
23 Setting hostname to tidemo-
24 MAC Address: 00-17-EA-E8-F1-A4
25 Configuring DHCP client
26 PASS successfully initialized
27 Ethernet subsystem successfully initialized
28 Ethernet eventId : 48 and vectId (Interrupt) : 7
29 Registration of the EMAC Successful, waiting for link up ..
30 Service Status: DHCPDC   : Enabled   :           : 000
31 Service Status: THTTP    : Enabled   :           : 000
32 Service Status: DHCPDC   : Enabled   : Running  : 000
33 Network Added: If-1:10.0.6.218
34 Service Status: DHCPDC   : Enabled   : Running  : 017

```

В листинге 3-6 приведен пример вывода в UART запуска приложения теста платформы для процессоров TMS320C6670 модуля SVP-716 (файл «Program/bin/platform_test_c6670.out»).

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

```

1 Platform test for module
2
3  /-----\ /-----\ |-----\ |-----\
4 | (  \ \ / / | ) |-----\ / / | / /
5  \ \ / / | |-----\ / / | | \
6  ) | \ / | | |-----\ / / | | ( ) |
7 |-----\ \ | | /-----\ / / |-----\
8
9 p_info->version           = 2.00.00.15
10 p_info->board_name        = SVP-716
11 p_info->serial_nbr        =
12 p_info->cpu.core_count    = 4
13 p_info->cpu.name          = TMS320C6670
14 p_info->cpu.id            = 21
15 p_info->cpu.revision_id   = 0
16 p_info->cpu.endian        = 1

```

```
17 p_info->frequency = 1000 MHz
18 p_info->led[PLATFORM_USER_LED_CLASS].count = 2
19 p_info->led[PLATFORM_SYSTEM_LED_CLASS].count = 0
20 p_info->emac.port_count = 1
21   -> EMAC port 0
22     MAC address = 00:17:ea:e8:f1:a4
23
24 NOR device info:
25 p_device->device_id = 0xbb18
26 p_device->manufacturer_id = 0x20
27 p_device->width = 8
28 p_device->block_count = 256
29 p_device->page_count = 256
30 p_device->page_size = 256
31 p_device->spare_size = 0
32 p_device->handle = 0xbb18
33 p_device->flags = 0
34 p_device->bboffset = 0
35
36 EEPROM device (I2C address 0x50) info:
37 p_device->device_id = 0x50
38 p_device->manufacturer_id = 0x1
39 p_device->width = 8
40 p_device->block_count = 1
41 p_device->page_count = 1
42 p_device->page_size = 65536
43 p_device->spare_size = 0
44 p_device->handle = 0x50
45 p_device->flags = 0
46 p_device->bboffset = 0
47
48 EEPROM device (I2C address 0x51) info:
49 p_device->device_id = 0x51
50 p_device->manufacturer_id = 0x1
51 p_device->width = 8
52 p_device->block_count = 1
53 p_device->page_count = 1
54 p_device->page_size = 65536
55 p_device->spare_size = 0
56 p_device->handle = 0x51
57 p_device->flags = 0
58 p_device->bboffset = 0
59 Current core id is 0
60
61 -----
62 LED test
63 -----
64 Testing USER class leds (2 leds)...
65 - LED 0 ON
66 - LED 0 OFF
67 - LED 0 ON
68 - LED 0 OFF
69 - LED 1 ON
70 - LED 1 OFF
71 - LED 1 ON
72 - LED 1 OFF
73 LED test complete
74
75 -----
76 EEPROM test
77 -----
78 test_eeprom: Partial EEPROM test mode
79 test_eeprom: 0x50: Saving EEPROM contents...
80 test_eeprom: 0x50: Writing 1024 bytes...
81 test_eeprom: 0x50: Read and check data...
82 test_eeprom: 0x50: Restoring EEPROM contents...
83 test_eeprom: 0x50: Test passed
84 test_eeprom: 0x51: Saving EEPROM contents...
85 test_eeprom: 0x51: Writing 1024 bytes...
86 test_eeprom: 0x51: Read and check data...
87 test_eeprom: 0x51: Restoring EEPROM contents...
88 test_eeprom: 0x51: Test passed
89 EEPROM test complete
90
91 -----
92 NOR test
93 -----
94 test_nor: Partial NOR test mode
```

```
95 test_nor: NOR flash sector size is 65536 bytes
96 test_nor: Test block size is 65536 bytes
97 test_nor: Saving original data from NOR...
98 test_nor: Writing pattern data to NOR...
99 test_nor: Reading pattern data from NOR...
100 test_nor: Comparing data...
101 test_nor: Data is equal. Test passed.
102 test_nor: Write back original data to NOR...
103 test_nor: Test passed
104 NOR test complete
105
106 -----
107 External memory test
108 -----
109 platform_external_memory_test(start = 0x80000000, end = 0x8fffffff) called
110 External memory test passed
111 External memory test complete
112
113 Test completed
```

3.4 Запись образов в EEPROM память

Для записи образа в EEPROM память модуля SVP-716 выполните шаги процедуры 3-3.

Внимание



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

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-716, составляет 128 Кбайт. EEPROM память, установленная на модуле SVP-716, разделена на две части (каждая объемом по 64 Кбайта): адресу 0x50 шины I²C соответствует первая часть, а адресу 0x51 шины I²C — вторая часть (см. рисунок 3-2). При этом, загрузка модуля SVP-716 возможна только из первой части EEPROM с адресом шины I²C 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-716/Program/bin».
3. Перейдите в каталог «D:/Dev/Modules/SVP-716/Program».
4. Запустите скрипт «svp716_program.bat» в режиме EEPROM с требуемыми параметрами командной строки (см. таблицу 3-1 раздела 3.2).

EEPROM (128 Кбайт)



Рисунок 3-2: Организация EEPROM памяти на модуле SVP-716

В процедуре 3-3 представлены общие шаги, необходимые для записи образа в EEPROM память процессоров модуля SVP-716.

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-716 образа загрузчика IBL.

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

1. Выполните сборку загрузчика IBL (процесс сборки IBL подробно описан в разделе 2).
В результате сборки загрузчика IBL должен быть получен файл «D:/Dev/Modules/SVP-716/ibl/src/make/bin/i2crom_0x50_svp716_le.bin».

Примечание

Модуль SVP-716 поставляется с уже записанным образом загрузчика IBL в EEPROM память. Образ, записанный в EEPROM память при изготовлении модуля SVP-716 содержится в файле «i2crom_0x50_svp716_le.bin». Данный файл расположен в каталоге «D:/Dev/Modules/SVP-716/Program/bin/ibl».

2. Скопируйте файлы, полученные в шаге 1, в папку «D:/Dev/Modules/SVP-716/Program/bin». Для этого выполните в терминале последовательно следующие команды:

```
d:  
cd D:\Dev\Modules\SVP-716  
copy ibl\src\make\bin\i2crom_0x50_svp716.bin Program\bin
```

3. Установите для процессора модуля SVP-716 режим «NOBOOT» в соответствии с данными таблицы Б-1 приложения Б.
4. Подключите отладчик к модулю SVP-716 и к компьютеру (предполагается использование отладчика Spectrum Digital XDS560v2 STM LAN).
5. Включите модуль SVP-716.
6. Перейдите в каталог «D:/Dev/Modules/SVP-716/Program». Для этого необходимо выполнить в терминале команды:

```
d:  
cd D:\Dev\Modules\SVP-716\Program
```

7. Запустите скрипт «svp716_program.bat» выполнив в терминале команду:

```
svp716_program.bat EEPROM all image=i2crom_0x50_svp716_le.bin
```

Будет запущен процесс записи образов в EEPROM память процессоров модуля SVP-716. При этом, вывод в терминал должен быть аналогичен тому, что приведен в листинге А-2 приложения А. Весь процесс записи должен занимать не более 5 минут.

Состояния светодиодов «DSP» на передней панели модуля SVP-716 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-716.
9. Установите для всех процессоров модуля SVP-716 требуемый режим загрузки в соответствии с данными таблицы Б-1 приложения Б.
10. Включите модуль SVP-716.

4 Конфигурация IBL

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

Для облегчения процесса изменения данных в блоке конфигурационных данных загрузчика, при сборке IBL (см. раздел 2), создается образ специальной программы «i2cparam_0x50_svp716_le_0x500.out» (см. таблицу 2-1), которая предназначена для загрузки на процессоре TMS320C6770 модуля SVP-716 соответственно.

Примечание

Значение «0x500» в имени файлов «i2cparam_0x50_svp716_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-716, как показано в разделе 5.

После запуска целевой конфигурации, как показано в разделе 5, выполните подключение требуемому процессору. Для этого, нажмите правой кнопкой мыши на названии ядра «DSP1_C6670_0» и выберите пункт меню «Connect Target» (см. рисунок 4-1).

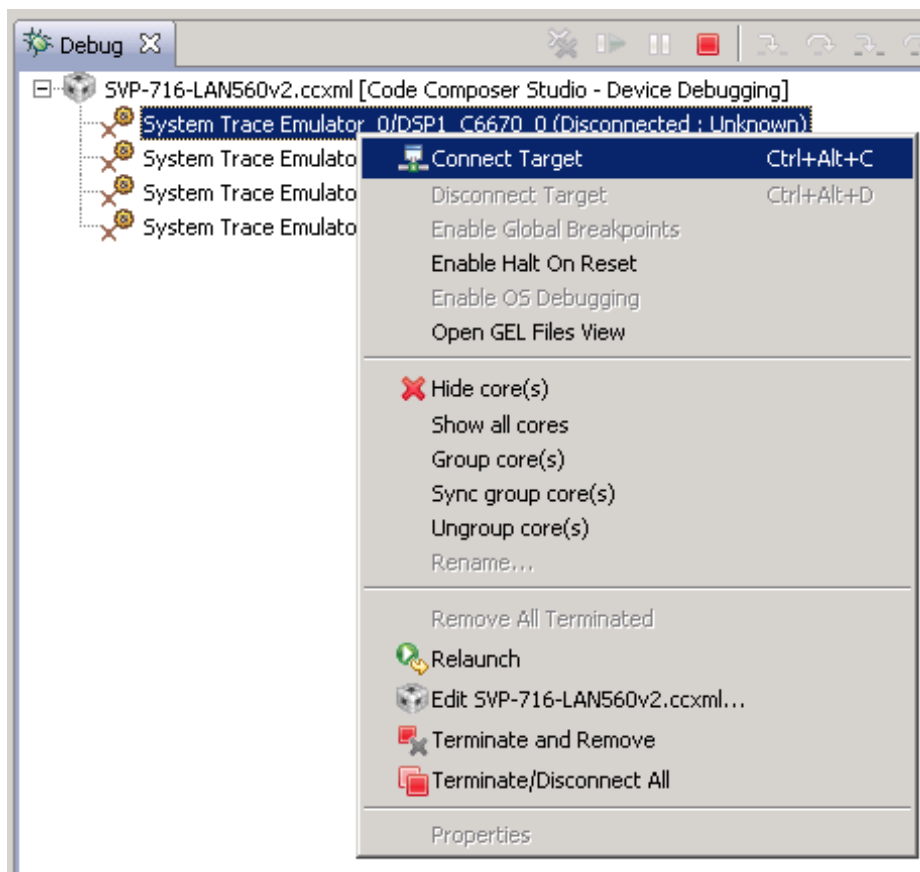


Рисунок 4-1: Подключение к процессору «DSP1_C6670»

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».

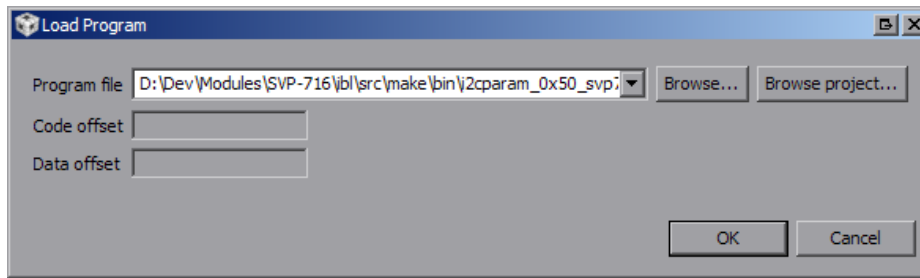


Рисунок 4-2: Окно загрузки кода на ядро процессора

В открывшемся окне выбора файлов необходимо выбрать файл «i2cparam_0x50_svp716_le_0x500.out» (см. таблицу 2-1) и нажать на кнопку «ОК».

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

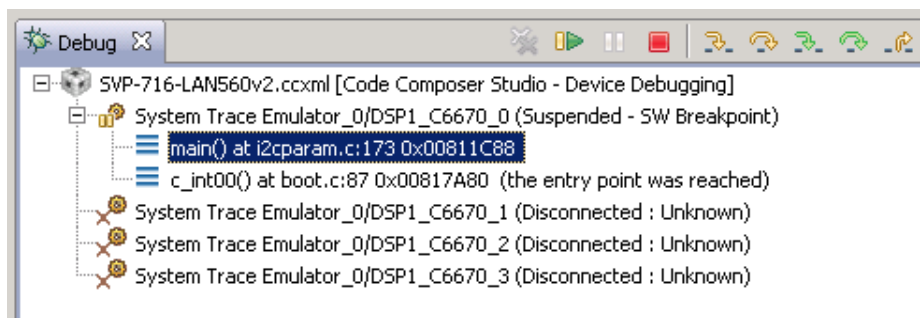




Рисунок 4-3: Внешний вид окна «Debug» после загрузки кода на ядра процессоров

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 program the I2C» (см. рисунок 4-4).

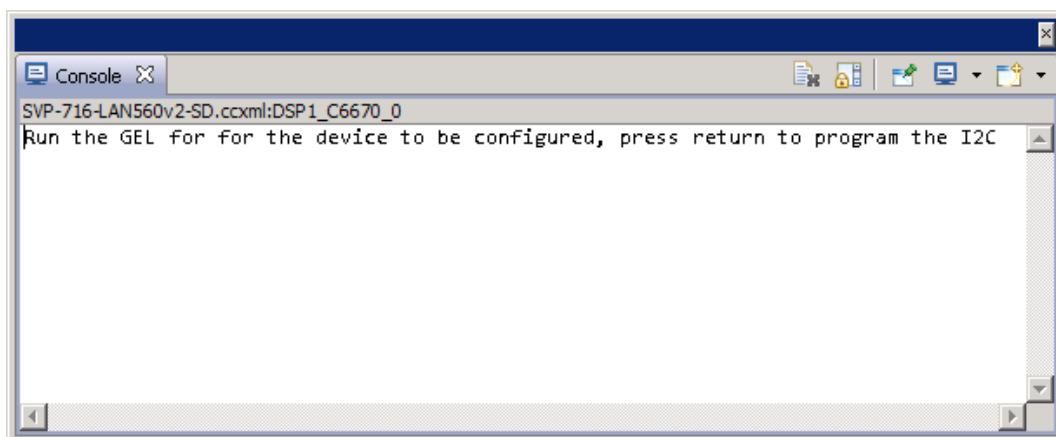


Рисунок 4-4: Внешний вид окна «Console» после запуска кода на процессоре

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

Конфигурация параметров загрузчика осуществляется при помощи специального GEL файла «i2cConfig.gel», который можно найти в папке «D:\Dev\Modules\SVP-716\ibl\src\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.

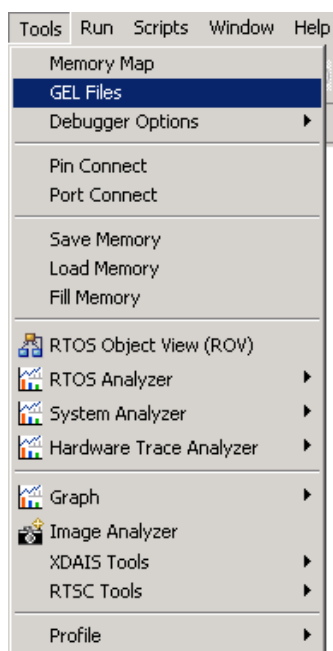


Рисунок 4-5: Пункт главного меню для вызова окна управления GEL файлами

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 «svp716_c6670.gel») и выберите пункт меню «Load GEL...».

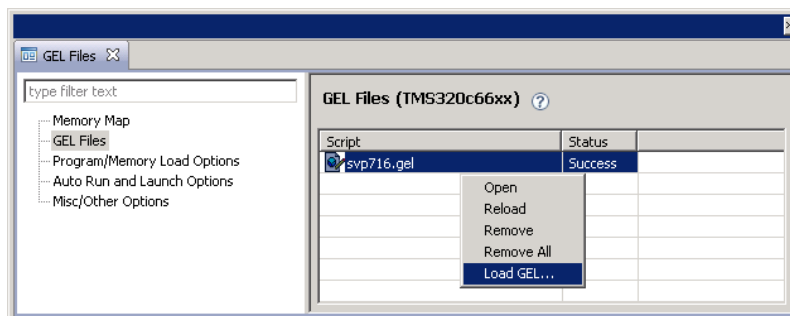


Рисунок 4-6: Окно управления GEL файлами

В открывшемся окне необходимо выбрать файл «D:/Dev/Modules/SVP-716/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.

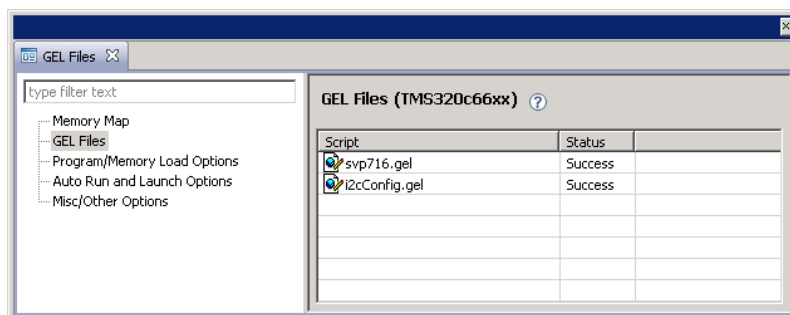


Рисунок 4-7: Окно управления GEL файлами с загруженным файлом «i2cConfig.gel»

После выполнения данных действий, должен появиться пункт меню «Scripts > SET SVP-716 > setConfig_svp716_main» при выборе которого будет применена конфигурация загрузчика (см. рисунок 4-8).

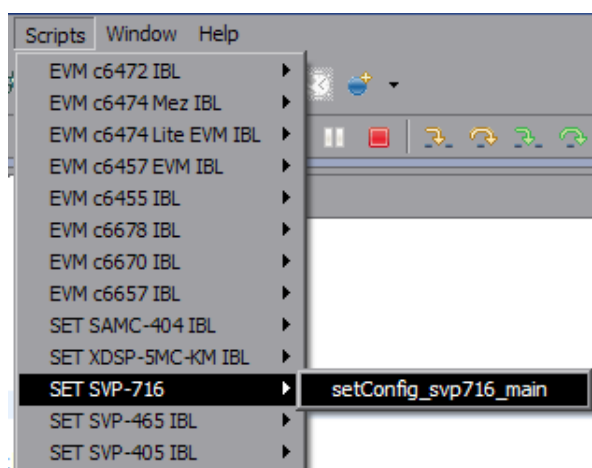


Рисунок 4-8: Пункт главного меню «Scripts > SET SVP-716»

Далее, необходимо щелкнуть 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).

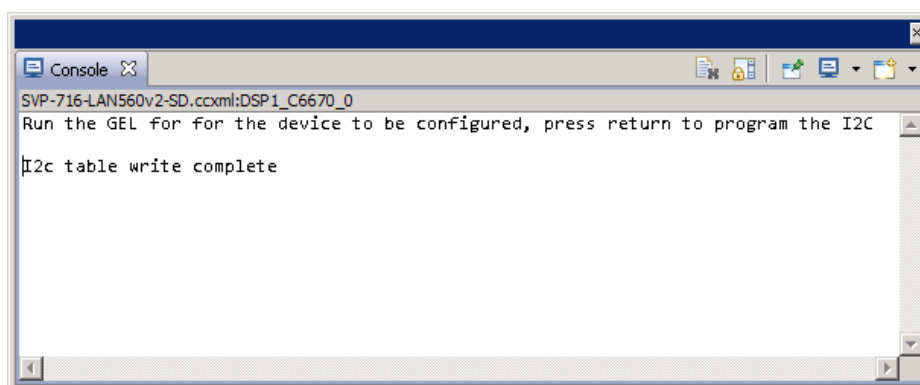


Рисунок 4-9: Внешний вид окна «Console» после выполнения записи конфигурации в EEPROM память

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-716.

Листинг 4-1: Фрагмент GEL файла «i2cConfig.gel» с конфигурационными параметрами загрузчика IBL для модуля SVP-716

```

1691 menuitem "SET SVP-716";
1692
1693 hotmenu setConfig_svp716_main()
1694 {
1695     ibl.iblMagic = ibl_MAGIC_VALUE;
1696     ibl.iblEvmType = ibl_SVP_716;//ibl_EVM_C6670L;
1697
1698     /* Main PLL: 150 MHz reference, 1000 MHz output */
1699     /* outFreqMhz = inFreq * mult / postdiv = 150 * 20 / 3 = 1000 */
1700     ibl.pllConfig[ibl_MAIN_PLL].doEnable = 1;
1701     ibl.pllConfig[ibl_MAIN_PLL].prediv = 3;
1702     ibl.pllConfig[ibl_MAIN_PLL].mult = 40;
1703     ibl.pllConfig[ibl_MAIN_PLL].postdiv = 2;
1704     ibl.pllConfig[ibl_MAIN_PLL].pllOutFreqMhz = 1000;
1705

```

```

1706  /* DDR PLL: 65 MHz reference, 1300 MHz output */
1707  /* outFreqMhz = 2 * inFreq * mult / postdiv = 65 * 2 * 20 / 2 = 1300 */
1708  ibl.pllConfig[ibl_DDR_PLL].doEnable = 1;
1709  ibl.pllConfig[ibl_DDR_PLL].prediv = 2; // for 1333
1710  ibl.pllConfig[ibl_DDR_PLL].mult = 41;
1711  ibl.pllConfig[ibl_DDR_PLL].postdiv = 2;
1712  ibl.pllConfig[ibl_DDR_PLL].pllOutFreqMhz = 1300;
1713
1714  /* Net PLL: 150 MHz reference, 1050 MHz output (followed by a built in divide by 3 to give 350 MHz to PA) */
1715  /* outFreqMhz = (inFreq * mult / postdiv) / 3 = (150 * 14 / 2) / 3 = 1050 / 3 = 350 */
1716  ibl.pllConfig[ibl_NET_PLL].doEnable = 1;
1717  ibl.pllConfig[ibl_NET_PLL].prediv = 3;
1718  ibl.pllConfig[ibl_NET_PLL].mult = 42;
1719  ibl.pllConfig[ibl_NET_PLL].postdiv = 2;
1720  ibl.pllConfig[ibl_NET_PLL].pllOutFreqMhz = 1050;
1721
1722  ibl.ddrConfig.configDdr = 1;
1723  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;
1724
1725  ibl.ddrConfig.uEmif.emif4p0.sdRamConfig = 0x63066A32;
1726  ibl.ddrConfig.uEmif.emif4p0.sdRamConfig2 = 0;
1727  ibl.ddrConfig.uEmif.emif4p0.sdRamRefreshCtl = 0x0000514C;
1728  ibl.ddrConfig.uEmif.emif4p0.sdRamTiming1 = 0x1113783C;
1729  ibl.ddrConfig.uEmif.emif4p0.sdRamTiming2 = 0x30407FE3;
1730  ibl.ddrConfig.uEmif.emif4p0.sdRamTiming3 = 0x559F849F;
1731  ibl.ddrConfig.uEmif.emif4p0.lpDdrNvmTiming = 0;
1732  ibl.ddrConfig.uEmif.emif4p0.powerManageCtl = 0;
1733  ibl.ddrConfig.uEmif.emif4p0.iODFTTestLogic = 0;
1734  ibl.ddrConfig.uEmif.emif4p0.performCountCfg = 0;
1735  ibl.ddrConfig.uEmif.emif4p0.performCountMstRegSel = 0;
1736  ibl.ddrConfig.uEmif.emif4p0.readIdleCtl = 0;
1737  ibl.ddrConfig.uEmif.emif4p0.sysVbusIntEnSet = 0;
1738  ibl.ddrConfig.uEmif.emif4p0.sdRamOutImpdedCalCfg = 0;
1739  ibl.ddrConfig.uEmif.emif4p0.tempAlterCfg = 0;
1740  ibl.ddrConfig.uEmif.emif4p0.ddrPhyCtl1 = 0x0010010d;
1741  ibl.ddrConfig.uEmif.emif4p0.ddrPhyCtl2 = 0;
1742  ibl.ddrConfig.uEmif.emif4p0.priClassSvceMap = 0;
1743  ibl.ddrConfig.uEmif.emif4p0.mstId2ClsSvce1Map = 0;
1744  ibl.ddrConfig.uEmif.emif4p0.mstId2ClsSvce2Map = 0;
1745  ibl.ddrConfig.uEmif.emif4p0.eccCtl = 0;
1746  ibl.ddrConfig.uEmif.emif4p0.eccRange1 = 0;
1747  ibl.ddrConfig.uEmif.emif4p0.eccRange2 = 0;
1748  ibl.ddrConfig.uEmif.emif4p0.rdWrtExcThresh = 0;
1749
1750  ibl.sgmiConfig[0].configure = 1;
1751  ibl.sgmiConfig[0].adviseAbility = 0x9801;
1752  ibl.sgmiConfig[0].control = 0x20;
1753  ibl.sgmiConfig[0].txConfig = 0x108a1;
1754  ibl.sgmiConfig[0].rxConfig = 0x700621;
1755  ibl.sgmiConfig[0].auxConfig = 0x81;
1756
1757  ibl.sgmiConfig[1].configure = 1;
1758  ibl.sgmiConfig[1].adviseAbility = 0x9801; // 1
1759  ibl.sgmiConfig[1].control = 0x20; // 1;
1760  ibl.sgmiConfig[1].txConfig = 0x108a1;
1761  ibl.sgmiConfig[1].rxConfig = 0x700621;
1762  ibl.sgmiConfig[1].auxConfig = 0x81;
1763
1764  ibl.mdioConfig.nMdioOps = 0;
1765
1766  ibl.spiConfig.addrWidth = 24;
1767  ibl.spiConfig.nPins = 5;
1768  ibl.spiConfig.mode = 1;
1769  ibl.spiConfig.csel = 2;
1770  ibl.spiConfig.c2tdelay = 1;
1771  ibl.spiConfig.busFreqMHZ = 20;
1772
1773  ibl.emifConfig[0].csSpace = 2;
1774  ibl.emifConfig[0].busWidth = 8;
1775  ibl.emifConfig[0].waitEnable = 0;
1776
1777  ibl.emifConfig[1].csSpace = 0;
1778  ibl.emifConfig[1].busWidth = 0;
1779  ibl.emifConfig[1].waitEnable = 0;
1780
1781
1782  ibl.bootModes[0].bootMode = ibl_BOOT_MODE_NOR;
1783  ibl.bootModes[0].priority = ibl_HIGHEST_PRIORITY;
1784  ibl.bootModes[0].port = 0;
1785
1786  ibl.bootModes[0].u.norBoot.bootFormat = ibl_BOOT_FORMAT_ELF;//AUTO;
1787  ibl.bootModes[0].u.norBoot.bootAddress[0][0] = 0; /* Image 0 NOR offset byte address in LE mode */
1788  ibl.bootModes[0].u.norBoot.bootAddress[0][1] = 0x800000; /* Image 1 NOR offset byte address in LE mode */
1789  ibl.bootModes[0].u.norBoot.bootAddress[1][0] = 0; /* Image 0 NOR offset byte address in BE mode */
1790  ibl.bootModes[0].u.norBoot.bootAddress[1][1] = 0x800000; /* Image 1 NOR offset byte address in BE mode */

```

```

1791  ibl.bootModes[0].u.norBoot.interface    = ibl_PMEM_IF_SPI;
1792  ibl.bootModes[0].u.norBoot.blob[0][0].startAddress = 0x80000000; /* Image 0 Load start address in LE mode */
1793  ibl.bootModes[0].u.norBoot.blob[0][0].sizeBytes   = 0x800000; /* Image 0 size (10 MB) in LE mode */
1794  ibl.bootModes[0].u.norBoot.blob[0][0].branchAddress = 0x80000000; /* Image 0 branch address after Loading in LE mode
    ← */
1795  ibl.bootModes[0].u.norBoot.blob[0][1].startAddress = 0x80000000; /* Image 1 Load start address in LE mode */
1796  ibl.bootModes[0].u.norBoot.blob[0][1].sizeBytes   = 0x800000; /* Image 1 size (10 MB) in LE mode */
1797  ibl.bootModes[0].u.norBoot.blob[0][1].branchAddress = 0x80000000; /* Image 1 branch address after Loading in LE mode
    ← */
1798  ibl.bootModes[0].u.norBoot.blob[1][0].startAddress = 0x80000000; /* Image 0 Load start address in BE mode */
1799  ibl.bootModes[0].u.norBoot.blob[1][0].sizeBytes   = 0x800000; /* Image 0 size (10 MB) in BE mode */
1800  ibl.bootModes[0].u.norBoot.blob[1][0].branchAddress = 0x80000000; /* Image 0 branch address after Loading in BE mode
    ← */
1801  ibl.bootModes[0].u.norBoot.blob[1][1].startAddress = 0x80000000; /* Image 1 Load start address in BE mode */
1802  ibl.bootModes[0].u.norBoot.blob[1][1].sizeBytes   = 0x800000; /* Image 1 size (10 MB) in BE mode */
1803  ibl.bootModes[0].u.norBoot.blob[1][1].branchAddress = 0x80000000; /* Image 1 branch address after Loading in BE mode
    ← */
1804
1805  ibl.bootModes[1].bootMode = ibl_BOOT_MODE_NONE;
1806
1807  ibl.bootModes[2].bootMode = ibl_BOOT_MODE_TFTP;
1808  ibl.bootModes[2].priority = ibl_HIGHEST_PRIORITY+1;
1809  ibl.bootModes[2].port     = ibl_PORT_SWITCH_ALL;
1810
1811  ibl.bootModes[2].u.ethBoot.doBootp      = TRUE;
1812  ibl.bootModes[2].u.ethBoot.useBootpServerIp = TRUE;
1813  ibl.bootModes[2].u.ethBoot.useBootpFileName = TRUE;
1814  ibl.bootModes[2].u.ethBoot.bootFormat    = ibl_BOOT_FORMAT_NAME;
1815
1816  SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.ipAddr, 192,168,1,3);
1817  SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.serverIp, 192,168,1,2);
1818  SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.gatewayIp, 192,168,1,1);
1819  SETIP(ibl.bootModes[2].u.ethBoot.ethInfo.netmask, 255,255,255,0);
1820
1821  /* Use the e-fuse value */
1822  ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[0] = 0;
1823  ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[1] = 0;
1824  ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[2] = 0;
1825  ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[3] = 0;
1826  ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[4] = 0;
1827  ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[5] = 0;
1828
1829  ibl.bootModes[2].u.ethBoot.ethInfo.fileName[0] = 's';
1830  ibl.bootModes[2].u.ethBoot.ethInfo.fileName[1] = 'v';
1831  ibl.bootModes[2].u.ethBoot.ethInfo.fileName[2] = 'p';
1832  ibl.bootModes[2].u.ethBoot.ethInfo.fileName[3] = '7';
1833  ibl.bootModes[2].u.ethBoot.ethInfo.fileName[4] = '1';
1834  ibl.bootModes[2].u.ethBoot.ethInfo.fileName[5] = '6';
1835  ibl.bootModes[2].u.ethBoot.ethInfo.fileName[6] = '.';
1836  ibl.bootModes[2].u.ethBoot.ethInfo.fileName[7] = 'b';
1837  ibl.bootModes[2].u.ethBoot.ethInfo.fileName[8] = 'i';
1838  ibl.bootModes[2].u.ethBoot.ethInfo.fileName[9] = 'n';
1839  ibl.bootModes[2].u.ethBoot.ethInfo.fileName[10] = '\0';
1840  ibl.bootModes[2].u.ethBoot.ethInfo.fileName[11] = '\0';
1841  ibl.bootModes[2].u.ethBoot.ethInfo.fileName[12] = '\0';
1842  ibl.bootModes[2].u.ethBoot.ethInfo.fileName[13] = '\0';
1843  ibl.bootModes[2].u.ethBoot.ethInfo.fileName[14] = '\0';
1844
1845  ibl.bootModes[2].u.ethBoot.blob.startAddress = 0x80000000; /* Load start address */
1846  ibl.bootModes[2].u.ethBoot.blob.sizeBytes   = 0x20000000;
1847  ibl.bootModes[2].u.ethBoot.blob.branchAddress = 0x80000000; /* Branch address after Loading */
1848
1849  ibl.chkSum = 0;
1850 }

```

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-716 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.

Продолжение таблицы на следующей странице

Продолжение таблицы 4-1

Параметр	Описание
<code>ibl.bootModes[2].u.ethBoot.bootFormat</code>	<p>Задаёт формат загружаемого образа. Может принимать одно из следующих значений:</p> <ul style="list-style-type: none"> <code>ibl_BOOT_FORMAT_COFF</code> — объектный формат <code>COFF</code>. Загружается через встроенный в IBL загрузчик <code>COFF</code> файлов; <code>ibl_BOOT_FORMAT_ELF</code> — объектный формат <code>ELF</code>. Загружается через встроенный в IBL загрузчик <code>ELF</code> файлов; <code>ibl_BOOT_FORMAT_BBLOB</code> — бинарный формат готовый к загрузке на модуле (не требующий соответствующего загрузчика); <code>ibl_BOOT_FORMAT_AUTO</code> — автоматическое определение формата по сигнатуре файла; <code>ibl_BOOT_FORMAT_NAME</code> — автоматическое определение формата по расширению загружаемого файла («.out» — <code>COFF</code>, «.elf» — <code>ELF</code>, «.bin» — <code>BBLOB</code>).
<code>ibl.bootModes[2].u.ethBoot.ethInfo.hwAddress[0...5]</code>	Задаёт значение аппаратного MAC адреса сетевого интерфейса. Если все значения равны 0, используется встроенный производителем в процессор MAC-адрес.
<code>ibl.bootModes[2].u.ethBoot.ethInfo.fileName[0...63]</code>	Задаёт имя файла для загрузки. Максимальная длина имени файла составляет 64 символа. Последним символом имени файла загрузки обязательно должен быть символ «\0».
<code>ibl.bootModes[2].u.ethBoot.ethInfo.ipAddr</code>	Определяет значение фиксированного IP адреса. Октеты IP адреса в файле «i2cConfig.gel» записываются через запятую. Значение параметр используется в случае, если <code>ibl.bootModes[2].u.ethBoot.doBootp = FALSE</code> .
<code>ibl.bootModes[2].u.ethBoot.ethInfo.serverIp</code>	Задаёт IP адрес сервера загрузки в случае, если <code>ibl.bootModes[2].u.ethBoot.doBootp = FALSE</code> .
<code>ibl.bootModes[2].u.ethBoot.ethInfo.gatewayIp</code>	Задаёт IP адрес основного шлюза в случае, если <code>ibl.bootModes[2].u.ethBoot.doBootp = FALSE</code> .
<code>ibl.bootModes[2].u.ethBoot.ethInfo.netmask</code>	Задаёт маску подсети в том случае, если <code>ibl.bootModes[2].u.ethBoot.doBootp = FALSE</code> .
<code>ibl.bootModes[2].u.ethBoot.useBootpServerIp</code>	Если равен <code>FALSE</code> , то в качестве IP адреса сервера загрузки будет использовано значение параметра <code>ibl.bootModes[2].u.ethBoot.ethInfo.serverIp</code> . Если равен <code>TRUE</code> , то будет использован IP адрес сервера загрузки, указанный в <code>BOOTP</code> ответе от <code>BOOTP</code> сервера. Данный параметр имеет значение только в том случае, когда <code>ibl.bootModes[2].u.ethBoot.doBootp = TRUE</code> .
<code>ibl.bootModes[2].u.ethBoot.useBootpFileName</code>	Если равен <code>FALSE</code> , то в качестве имени файла для загрузки будет использовано значение параметра <code>ibl.bootModes[2].u.ethBoot.ethInfo.fileName[0...63]</code> . Если равен <code>TRUE</code> , то будет использовано имя файла загрузки, указанное в <code>BOOTP</code> ответе от <code>BOOTP</code> сервера. Данный параметр имеет значение только в том случае, когда <code>ibl.bootModes[2].u.ethBoot.doBootp = TRUE</code> .
<code>ibl.bootModes[2].u.ethBoot.blob.startAddress</code>	Адрес области памяти куда будет осуществляться загрузка образа.
<code>ibl.bootModes[2].u.ethBoot.blob.sizeBytes</code>	Максимальный допустимый объем образа, который допускается загрузить.
<code>ibl.bootModes[2].u.ethBoot.blob.branchAddress</code>	Адрес точки входа, куда будет осуществлен переход после завершения загрузки образа приложения.

5 Импорт и запуск целевой конфигурации модуля

Для загрузки кода приложений на модуль SVP-716, в первую очередь, необходимо запустить целевую конфигурацию модуля SVP-716. В папке «TargetConfigurations» сопроводительного диска к модулю SVP-716 находятся файлы целевых конфигураций для различных отладчиков. В данном документе рассматривается загрузка кода через отладчик Blackhawk LAN560 v2 System Trace. Данному отладчику соответствует файл целевой конфигурации «SVP-716-LAN560v2.ccxml», который необходимо импортировать в рабочее пространство.

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

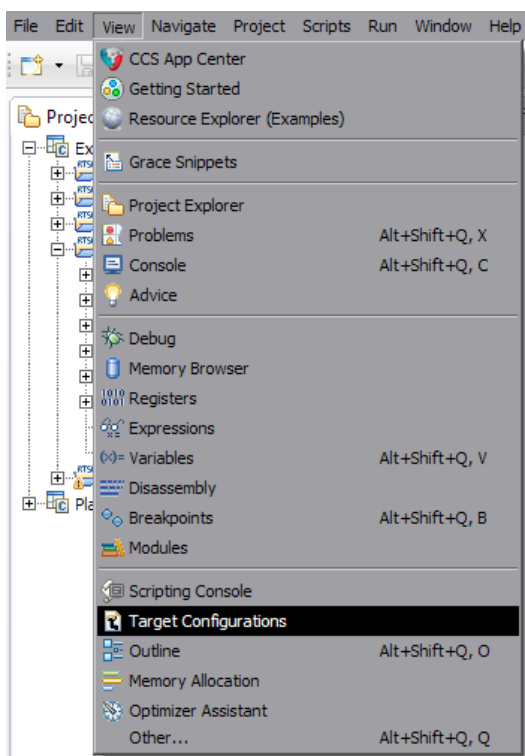


Рисунок 5-1: Пункт меню для отображения окна целевых конфигураций

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

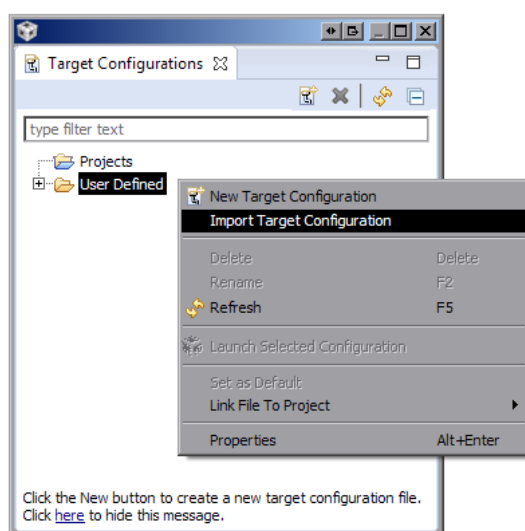


Рисунок 5-2: Меню импорта целевой конфигурации

В появившемся окне выбора файла (см. рисунок 5-3) необходимо выбрать файл «SVP-716-USB560v2.ccxml» и нажать на кнопку «Открыть». В данном документе предполагается, что папка «TargetConfigurations» с сопроводительного диска к модулю SVP-716, где расположен файл «SVP-716-USB560v2.ccxml», скопирована в папку «D:/Dev/Modules/SVP-716/TargetConfigurations».

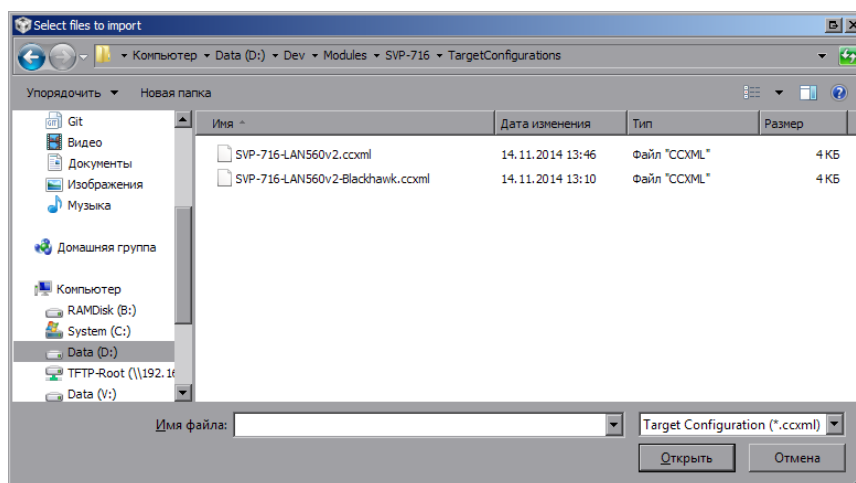


Рисунок 5-3: Окно выбора файла для импорта целевой конфигурации

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».

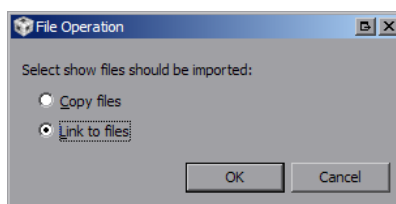


Рисунок 5-4: Окно выбора способа импорта файла целевой конфигурации

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

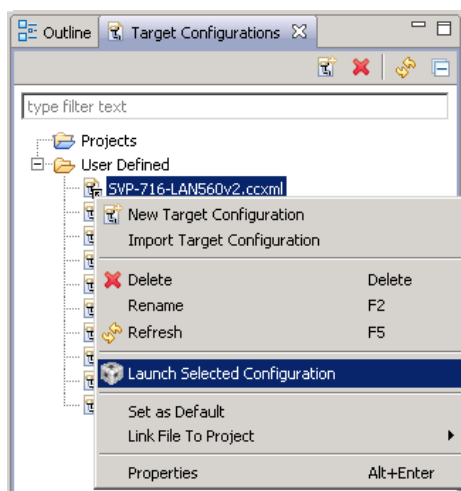


Рисунок 5-5: Запуск целевой конфигурации

После запуска целевой конфигурации модуля SVP-716, среда разработки CCS перейдет в режим отладки и в окне «Debug» будет выведен список ядер процессора TMS320C6670 модуля SVP-716, как показано на рисунке 5-6.

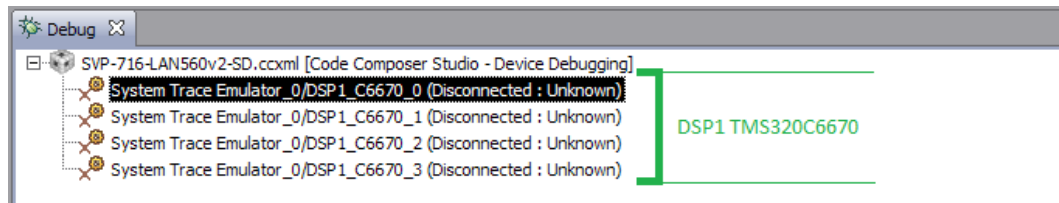


Рисунок 5-6: Список ядер процессора модуля SVP-716

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 сегмента 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 атрибутов 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 утилита 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 прединициализационных 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 подходит 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 подходит 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».

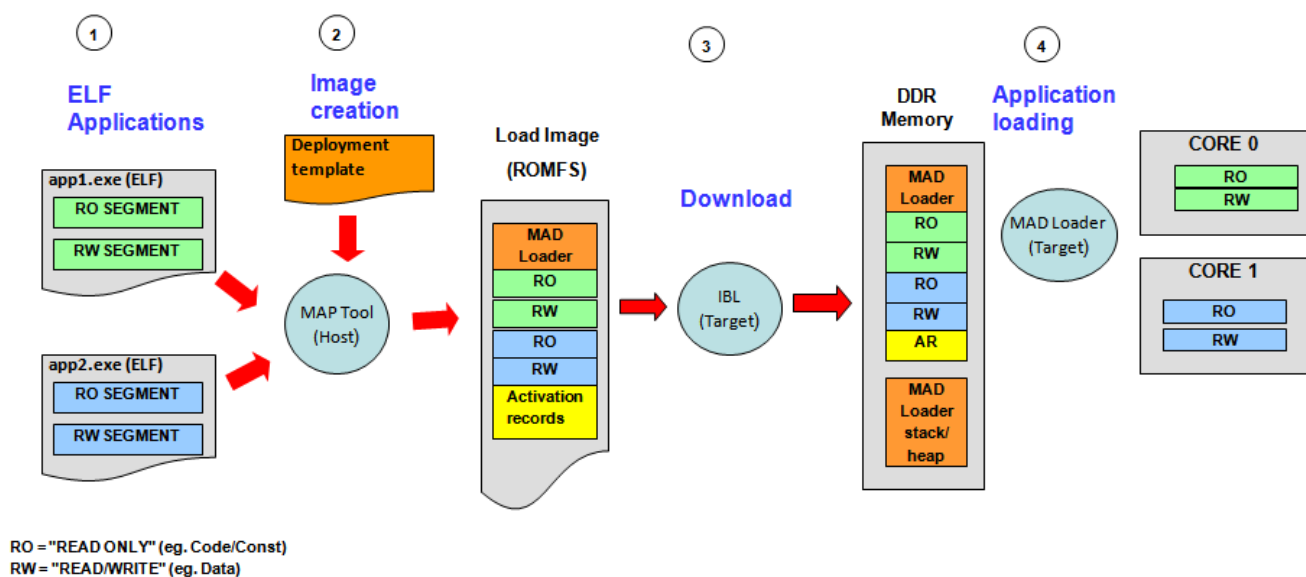


Рисунок 6-1: Схема работы MAD в режиме «Prelinker bypass mode»

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 I²C 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

```

1 {
2   "deploymentCfgFile" : ".\\deploy.json",
3   "LoadImageName"    : "mad_test.bin",
4   "prelinkExe"       : "C:\\ti\\ccsv5\\tools\\compiler\\c6000_7.4.2\\bin\\prelink6x",
5   "stripExe"         : "C:\\ti\\ccsv5\\tools\\compiler\\c6000_7.4.2\\bin\\strip6x",
6   "ofdTool"          : "C:\\ti\\ccsv5\\tools\\compiler\\c6000_7.4.2\\bin\\ofd6x.exe",
7   "malApp"           : "C:\\ti\\mcsdk_2_01_02_06\\tools\\boot_loader\\mad-utils\\mad-loader\\bin\\C6670\\le\\mal_app.exe",
8   "nmlLoader"        : "C:\\ti\\mcsdk_2_01_02_06\\tools\\boot_loader\\mad-utils\\mad-loader\\bin\\C6670\\le\\nml.exe"
9 }

```

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: <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • name — имя раздела. Используется для идентификации раздела памяти в отладочном выводе MAP Tool; • vaddr — 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: <ul style="list-style-type: none"> • 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

```

1  {
2      "deviceName" : "C6670",
3      "partitions" : [
4          {
5              "name" : "load-partition",
6              "vaddr" : "0x8e000000",
7              "size" : "0x2000000",
8              "loadPartition" : true
9          }
10     ],
11     "applications" : [
12         {
13             "name" : "app1",
14             "fileName" : "../build/app_1.out",
15             "allowedCores" : [0,1,2,3]
16         },
17         {
18             "name" : "app2",
19             "fileName" : "../build/app_2.out",
20             "allowedCores" : [0,1,2,3]
21         }
22     ],
23     "appDeployment" : [
24         "app1",
25         "app1",
26         "",
27         "app2"
28     ]
29 }

```

6.6 Запуск MAP Tool

Для запуска MAP Tool потребуется установленный Python интерпретатор, который можно бесплатно скачать с официального сайта¹.

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 устройстве, 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_BLOB;
ibl.bootModes[2].u.ethBoot.blob.startAddress = 0x8e000000;
ibl.bootModes[2].u.ethBoot.blob.sizeBytes   = 0x20000000;
ibl.bootModes[2].u.ethBoot.blob.branchAddress = 0x8e001040;
```

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

```
ibl.bootModes[0].u.norBoot.bootFormat      = ibl_BOOT_FORMAT_BLOB;
ibl.bootModes[0].u.norBoot.blob[0][0].startAddress = 0x8e000000;
ibl.bootModes[0].u.norBoot.blob[0][0].sizeBytes   = 0x800000;
ibl.bootModes[0].u.norBoot.blob[0][0].branchAddress = 0x8e001040;
```

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

¹ <https://www.python.org/download/>

Приложение А Примеры работы скрипта записи EEPROM и NOR флеш памяти

В листинге А-1 приведен пример вывода в терминал запуска скрипта «svp716_program.bat» для записи в NOR флеш память образов по умолчанию на все четыре процессора модуля SVP-716.

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 «svp716_program.bat NOR all»

```
1 D:\Dev\Modules\SVP-716\Program>svp716_program NOR
2
3 SVP-716 program script started
4 Copyright (c) 2014, Scan Engineering Telecom SPb
5
6 Writing to DSP1 ENABLED
7
8 Target configuration file:
9   "../TargetConfigurations/SVP-716-LAN560v2.ccxml"
10
11 Images for DSP's:
12   DSP1: "bin/nor_c6670.bin" (572741 bytes)
13
14 Configuring debug server for SVP-716 board...
15 Done!
16 Opening a debug session for 1 DSP(s)...
17 Done!
18 Connecting to DSP(s)...
19 DSP1_C6670_0: GEL Output: No initialization performed since bootmode = 0x0000000
20 5
21
22 DSP1_C6670_0: GEL Output: You can manually initialize with GlobalDefaultSetup
23
24 Done!
25 Loading program to DSP(s)...
26 DSP1_C6670_0: GEL Output: Invalidate All Cache...
27
28 DSP1_C6670_0: GEL Output: Invalidate All Cache... Done.
29
30 DSP1_C6670_0: GEL Output: GEL Reset...
31
32 DSP1_C6670_0: GEL Output: GEL Reset... Done.
33
34 Done!
35 Loading binary images to DSP(s) memory...
36   Loading file "bin/nor_c6670.bin" (572741 bytes)...
37 Done!
38 Configuring writers on DSP(s)...
39 Done!
40 Executing writers on DSP(s)...
41 NOR Writer Utility Version 02.00.00.00
42 Copyright (c) 2014, Scan Engineering Telecom SPb
43
44 Write size:   572741 bytes
45 Start address: 0x0
46
47 Flashing sector 0 (0 bytes of 572741)
48 Flashing sector 1 (65536 bytes of 572741)
49 Flashing sector 2 (131072 bytes of 572741)
50 Flashing sector 3 (196608 bytes of 572741)
51 Flashing sector 4 (262144 bytes of 572741)
52 Flashing sector 5 (327680 bytes of 572741)
53 Flashing sector 6 (393216 bytes of 572741)
54 Flashing sector 7 (458752 bytes of 572741)
55 Flashing sector 8 (524288 bytes of 572741)
56 Reading and verifying sector 0 (0 bytes of 572741)
57 Reading and verifying sector 1 (65536 bytes of 572741)
58 Reading and verifying sector 2 (131072 bytes of 572741)
59 Reading and verifying sector 3 (196608 bytes of 572741)
60 Reading and verifying sector 4 (262144 bytes of 572741)
```

```
61 Reading and verifying sector 5 (327680 bytes of 572741)
62 Reading and verifying sector 6 (393216 bytes of 572741)
63 Reading and verifying sector 7 (458752 bytes of 572741)
64 Reading and verifying sector 8 (524288 bytes of 572741)
65 NOR programming completed successfully
66 Done!
67 Terminating debug sessions on DSP(s)...
68 Done!
69 Stopping debug server...
70 Done!
71
72
73 D:\Dev\Modules\SVP-716\Program>
```

В листинге A-2 приведен пример вывода в терминал запуска скрипта «svp716_program.bat» для записи в EEPROM память образов по умолчанию на все четыре процессора модуля SVP-716.

Листинг A-2: Вывод в терминал при запуске команды «svp716_program.bat EEPROM all»

```
1 D:\Dev\Modules\SVP-465\Program>svp716_program EEPROM
2
3 SVP-716 program script started
4 Copyright (c) 2014, Scan Engineering Telecom SPb
5
6 Writing to DSP1 ENABLED
7
8 Target configuration file:
9   "../TargetConfigurations/SVP-716-LAN560v2.ccxml"
10
11 Images for DSP's:
12   DSP1: "bin/eeprom_0x50_c6670.bin" (60744 bytes)
13
14 Configuring debug server for SVP-716 board...
15 Done!
16 Opening a debug session for 1 DSP(s)...
17 Loaded FPGA Image: C:\ti\ccsv5\ccs_base\common\uscif\dtc_top.jbc
18 Done!
19 Connecting to DSP(s)...
20
21 DSP1_C6670_0: GEL Output: Global Default Setup...
22
23 DSP1_C6670_0: GEL Output: SVP-716 GEL file Ver is 2.0
24
25 DSP1_C6670_0: GEL Output: Setup Cache...
26
27 DSP1_C6670_0: GEL Output: L1P = 32K
28
29 DSP1_C6670_0: GEL Output: L1D = 32K
30
31 DSP1_C6670_0: GEL Output: L2 = ALL SRAM
32
33 DSP1_C6670_0: GEL Output: Setup Cache... Done.
34
35 DSP1_C6670_0: GEL Output: Main PLL (PLL1) Setup ...
36
37 DSP1_C6670_0: GEL Output: PLL1 Setup for DSP @ 1000.0 MHz.
38
39 DSP1_C6670_0: GEL Output:           SYSCLK2 = 333.3333 MHz, SYSCLK5 = 200.0 MHz.
40
41 DSP1_C6670_0: GEL Output:           SYSCLK8 = 15.625 MHz.
42
43 DSP1_C6670_0: GEL Output: PLL1 Setup... Done.
44
45 DSP1_C6670_0: GEL Output: Power on all PSC modules and DSP domains...
46
47 DSP1_C6670_0: GEL Output: Security Accelerator disabled!
48
49 DSP1_C6670_0: GEL Output: Power on all PSC modules and DSP domains... Done.
50
51 DSP1_C6670_0: GEL Output: DDR3 PLL (PLL2) Setup ...
52
53 DSP1_C6670_0: GEL Output: DDR3 PLL Setup... Done.
54
55 DSP1_C6670_0: GEL Output: DDR begin (1333 auto)
56
57 DSP1_C6670_0: 2: XMC setup complete.
58
59 DSP1_C6670_0: GEL Output:
60 DDR3 initialization is complete.
61
62 DSP1_C6670_0: GEL Output: DDR done
63
64 DSP1_C6670_0: GEL Output: DDR3 memory test... Started
65
66 DSP1_C6670_0: GEL Output: DDR3 memory test... Passed
67
68 DSP1_C6670_0: GEL Output: PLL and DDR Initialization completed(0) ...
69
70 DSP1_C6670_0: GEL Output: Set Board and DSP IO/Timers Pins...
71
72 DSP1_C6670_0: GEL Output: Set Board and DSP IO/Timers Pins... Done.
73
```



```
74 DSP1_C6670_0: GEL Output: Configuring CPSW ...
75
76 DSP1_C6670_0: GEL Output: Configuring CPSW ...Done
77
78 DSP1_C6670_0: GEL Output: Global Default Setup... Done.
79
80 Done!
81 Loading program to DSP(s)...
82 DSP1_C6670_0: GEL Output: Invalidate All Cache...
83
84 DSP1_C6670_0: GEL Output: Invalidate All Cache... Done.
85
86 DSP1_C6670_0: GEL Output: GEL Reset...
87
88 DSP1_C6670_0: GEL Output: GEL Reset... Done.
89
90 Done!
91 Loading binary images to DSP(s) memory...
92   Loading file "bin/eprom_0x50_c6670.bin" (60744 bytes)...
93 Done!
94 Configuring writers on DSP(s)...
95 Done!
96 Executing writers on DSP(s)...
97 EEPROM Writer Utility Version 02.00.00.00
98 Copyright (c) 2014, Scan Engineering Telecom SPb
99
100 Write size:      60744 bytes
101 I2C bus address: 0x50
102 I2C start address: 0x0
103 Swap data:      no
104
105 Writing 60744 bytes from DSP memory address 0x0c000100 to EEPROM (0x0050)...
106 Reading 60744 bytes from EEPROM (0x0050) to DSP memory address 0x0c010100...
107 Verifying data read...
108 EEPROM programming completed successfully
109 Done!
110 Terminating debug sessions on DSP(s)...
111 Done!
112 Stopping debug server...
113 Done!
114
115
116
117 D:\Dev\Modules\SVP-716\Program>
```

Приложение Б Выбор режима загрузки IBL

TODO: Переключение режимов загрузки при помощи переключателей на плате не реализовано на стороне FPGA модуля SVP-716.

Выбор режима загрузки IBL осуществляется при помощи переключателей SW1–SW2 на плате модуля SVP-716.

В таблице Б-1 представлены положения переключателей SW1–SW2 для всех возможных режимов загрузки IBL для каждого из двух процессоров модуля SVP-716.

Таблица Б-1: Положение переключателей модуля SVP-716 для установки режимов загрузки IBL

Режим	DSP1_C6670	
	SW1	SW2
NOBOOT (DEBUG)	Разомкнут	Разомкнут
NOR	Разомкнут	Замкнут
TFTP (BOOTP)	Замкнут	Разомкнут
PCI-E	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 процессоре TMS320C6670 32-х битный регистр $DEVSTAT$ доступен по адресу 0x02620020.

Список литературы

1. SVP-716. Сборка и запуск приложения веб-сервера. Руководство пользователя. [UG-SVP-716-WEB](#) (цит. на с. 13, 16).
2. Multicore Application Deployment (MAD) Utilities. User's Guide.
URL: http://processors.wiki.ti.com/index.php/MAD_Utils_User_Guide (цит. на с. 33, 34).
3. Установка и настройка сервера сетевой загрузки (BOOTP и TFTP). Руководство пользователя.
[UG-CMN-BOOTP-TFTP](#) (цит. на с. 37).