



Scan Engineering Telecom SPb

Загрузка кода на SAMC-403 по PCI-Express

Руководство пользователя

Версия 1.2



Код документа: UG-SAMC-403-PCIEBOOT
Дата сборки: 27 мая 2015 г.
Листов в документе: 33

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

История ревизий

Ревизия	Дата	Изменения
1.2	—	Правка вёрстки разделов. Исправлены опечатки
1.1	—	Исправлены опечатки
1.0	—	Начальная версия

Содержание

Список рисунков	4
Список таблиц	4
Список листингов	4
Список процедур	4
Перечень сокращений и условных обозначений	5
Введение	6
1 Общие сведения	7
1.1 Подготовка рабочего пространства	7
2 Примеры Texas Instruments	8
2.1 Пример «DDR Init»	8
2.2 Пример «Hello World»	9
2.3 Пример «POST»	11
2.4 Пример «EDMA-Interrupt»	12
2.5 Пример «DSP Local Reset»	14
2.6 Загрузчик кода на SAMC-403	15
3 Примеры Scan Engineering Telecom	18
3.1 Пример «Driver Example»	18
4 Роль IBL в загрузке кода на SAMC-403	22
4.1 Изменения в коде IBL	22
5 Тестовая установка и результаты	24
Приложение А: Вывод в UART и системную консоль примеров	28
A.1 Пример «Hello World»	28
A.2 Пример «POST»	29
A.3 Пример «EDMA-Interrupt»	30
A.4 Пример «DSP Local Reset»	31
A.5 Пример «Driver Example»	32
Список литературы	33

Список рисунков

1-1	Выбор расположения рабочего пространства в CCS	7
1-2	Окно «Project Explorer» со списком проектов, после запуска CCS	7
2-1	Окно «Project Explorer» с выбранным проектом примера «DDR Init»	8
2-2	Выбор активной конфигурации сборки	8
2-3	Вывод в консоль командного файла «pcieboot_ddrinit_elf2HBin.bat»	9
2-4	Окно «Project Explorer» с выбранным проектом примера «Hello World»	9
2-5	Окно «Project Explorer» с выбранным проектом примера «POST»	11
2-6	Окно «Project Explorer» с выбранным проектом примера «EDMA-Interrupt»	12
2-7	Схема работы примера «EDMA-Interrupt»	13
2-8	Окно «Project Explorer» с выбранным проектом примера «DSP Local Reset»	14
3-1	Окно «Project Explorer» с выбранным проектом примера «Driver Example»	18
3-2	Схема работы примера «Driver Example»	20
4-1	Схема процесса загрузки кода на модуль SAMC-403 с модуля SAMC-514	23
5-1	Схема тестовой установки с прямым соединением (шасси ELMA Blu!One 3000)	24
5-2	Расположение АМС-слотов в шасси ELMA Blu!One 3000	24
5-3	Схема тестовой установки с коммутатором SMCH-100-PE (шасси MicroBlade 2U)	24
5-4	Расположение АМС-слотов в шасси MicroBlade 2U	25
5-5	Проверка значений PCIe регистров BARn в CCS	27

Список таблиц

2-1	Значения PCIe регистров входящей трансляции адресов	15
5-1	Положение переключателей модуля SAMC-403 для включения режима загрузки по PCIe	25

Список листингов

2-1	Код мониторинга состояния соединения PCIe	10
2-2	Код для выполнения сброса C6678	11
A-1	Вывод в UART примера «Hello World»	28
A-2	Вывод в системную консоль примера «Hello World»	28
A-3	Вывод в UART примера «POST»	29
A-4	Вывод в системную консоль примера «POST»	29
A-5	Вывод в UART примера «EDMA-Interrupt»	30
A-6	Вывод в системную консоль примера «EDMA-Interrupt»	30
A-7	Вывод в системную консоль примера «DSP Local Reset»	31
A-8	Вывод в UART примера «Driver Example»	32
A-9	Вывод в системную консоль примера «Driver Example»	32

Список процедур

5-1	Запуск примеров	25
-----	-----------------	----

Перечень сокращений и условных обозначений

AMC	Advanced Mezzanine Card	24, 25
API	Application Programming Interface	15
ASCII	American Standard Code for Information Interchange	9, 18
BAR	Base Address Register	4, 15, 26, 27
CCS	Code Composer Studio	4, 6, 7, 15, 18, 26, 27
CGT	Code Generation Tools	9, 18
CPU	Central Processing Unit	14
CSL	Chip Support Library	12, 19
DBS	Data Burst Size	13
DDR	Double Data Rate	8–10, 12, 15, 19, 21, 22
DSP	Digital Signal Processor	9–12, 14, 15, 18, 19, 21, 22
EDMA	Enhanced Direct Memory Access	12, 13, 15, 21
EEPROM	Electrically Erasable Programmable Read-Only Memory	22, 24
ELF	Executable and Linkable Format	9, 18
EMAC	Ethernet Media Access Controller	22
EP	End Point	12
FPGA	Field-Programmable Gate Array	22
I²C	Inter-Integrated Circuit	22, 24
IBL	Intermediate Boot Loader	10–12, 19, 22, 24, 25
INTC	INTerrupt Controller	12
IPC	Inter Process Communication	10
JTAG	Joint Test Action Group	26
LTSSM	Link Training and Status State Machine	10
MCSDK	MultiCore Software Development Kit	6, 8, 9, 11
MMR	Memory Mapped Register	14
MSI	Message Signaled Interrupts	19
MTRR	Memory Type Range Registers	19
PCIe	PCI Express	4, 6, 10, 12–15, 19, 21, 22, 24–27
PCI	Peripheral Component Interconnect	5, 12, 19, 26
PDK	Platform Development Kit	8, 9, 11
PLL	Phase Locked Loop controller	14, 22
PSC	Power Sleep Controller	14
RC	Root Complex	12
ROM	Read-Only Memory	22
SRIO	Serial RapidIO	22
SSD	Solid State Drive	24
TI	Texas Instruments	6, 10, 18, 22
TLP	Transaction Layer Packet	13
UART	Universal Asynchronous Receiver-Transmitter	9–11, 19, 21, 25–27, 31
ОС	Операционная Система	6

Введение

В данном документе описан процесс сборки и запуска примеров для демонстрации загрузки кода на модуль SAMC-403 с хост-системы с ОС Linux по шине PCIe.

В документе приведено описание по сборке и запуску стандартных примеров TI из состава MCSDK (MultiCore Software Development Kit) и демонстрационного примера для тестирования скорости передачи данных по шине PCIe между модулем SAMC-403 и хост-системой.

В данном документе, в качестве хост-системы выступает модуль SAMC-514 с установленной на него операционной системой Ubuntu 10.04 x64.

Для сборки всех примеров используется CCS (Code Composer Studio) версии 5.2.0.00069. CCS рекомендуется устанавливать в папку «C:/ti». В случае, если CCS установлена в другую папку, некоторые из описанных в данном документе действий могут не работать.

1 Общие сведения

1.1 Подготовка рабочего пространства

Все описанные в данном документе проекты примеров, расположены в папке «Workspace_SAMC-403» сопроводительного диска к модулю SAMC-403. Данная папка является папкой рабочего пространства CCS.

Перед началом работы, необходимо переписать папку «Workspace_SAMC-403» с сопроводительного диска на жесткий диск компьютера, например, в папку «D:\Workspace_SAMC-403».

Далее, при запуске CCS необходимо указать эту папку в качестве расположения рабочего пространства, как показано на рисунке 1-1.

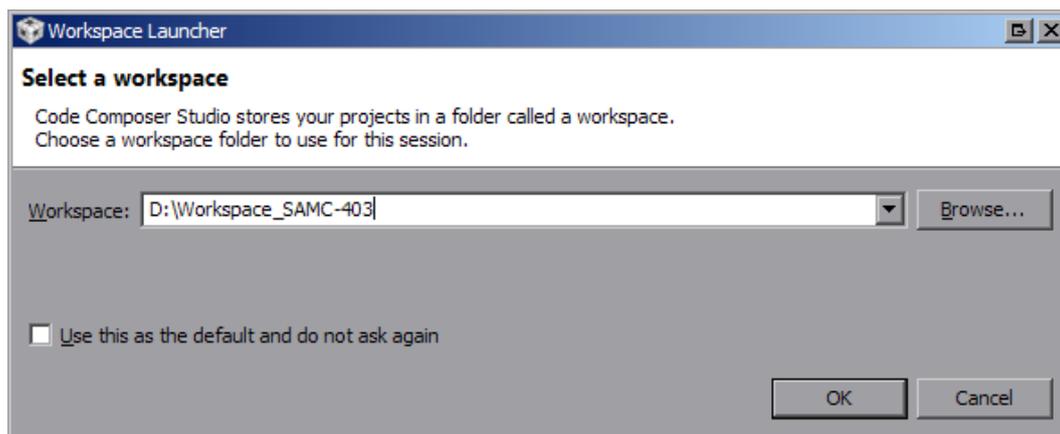


Рисунок 1-1: Выбор расположения рабочего пространства в CCS

После запуска CCS, в окне «Project Explorer» должны присутствовать необходимые проекты. На рисунке 1-2 показан пример окна «Project Explorer», в котором выделены необходимые проекты.

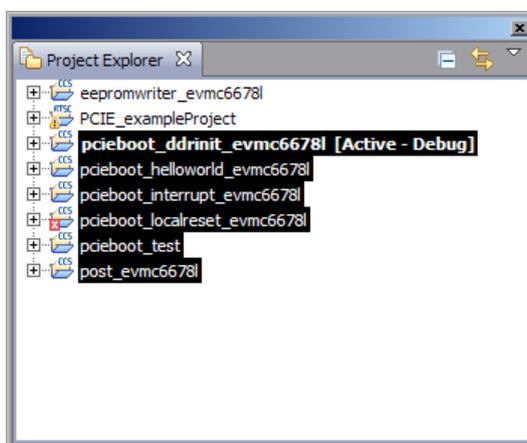


Рисунок 1-2: Окно «Project Explorer» со списком проектов, после запуска CCS

2 Примеры Texas Instruments

2.1 Пример «DDR Init»

Данный пример находится в проекте «pcieboot_ddrinit_evmc6678l» (см. рисунок 2-1). Пример «DDR Init» использует библиотеку MCSDK Platform Library из состава PDK для инициализации DDR памяти.

В примере «DDR Init» часть локальной L2 памяти используется «.out» файлом. Не допускается её использование в пользовательском приложении. Адрес 0x0087FFFC (Magic Address) также нельзя использовать, так как он используется в процессе загрузки. Для подробной информации об использовании памяти приложением смотрите «.mar» файл.

2.1.1 Процедура сборки

Перед сборкой проекта, необходимо убедиться, что установлен режим «little endian». Для этого, активной конфигурацией сборки должна быть установлена конфигурация «Debug». Выбор активной конфигурации осуществляется через меню «Project > Build Configurations > Set Active» (см. рисунок 2-2).

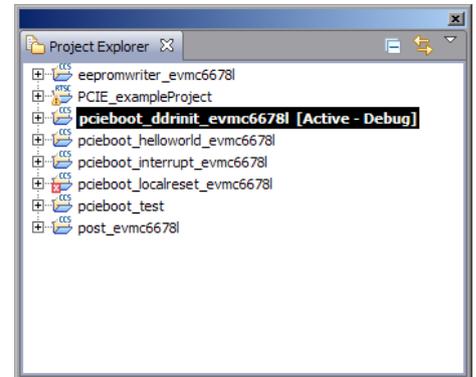


Рисунок 2-1: Окно «Project Explorer» с выбранным проектом примера «DDR Init»

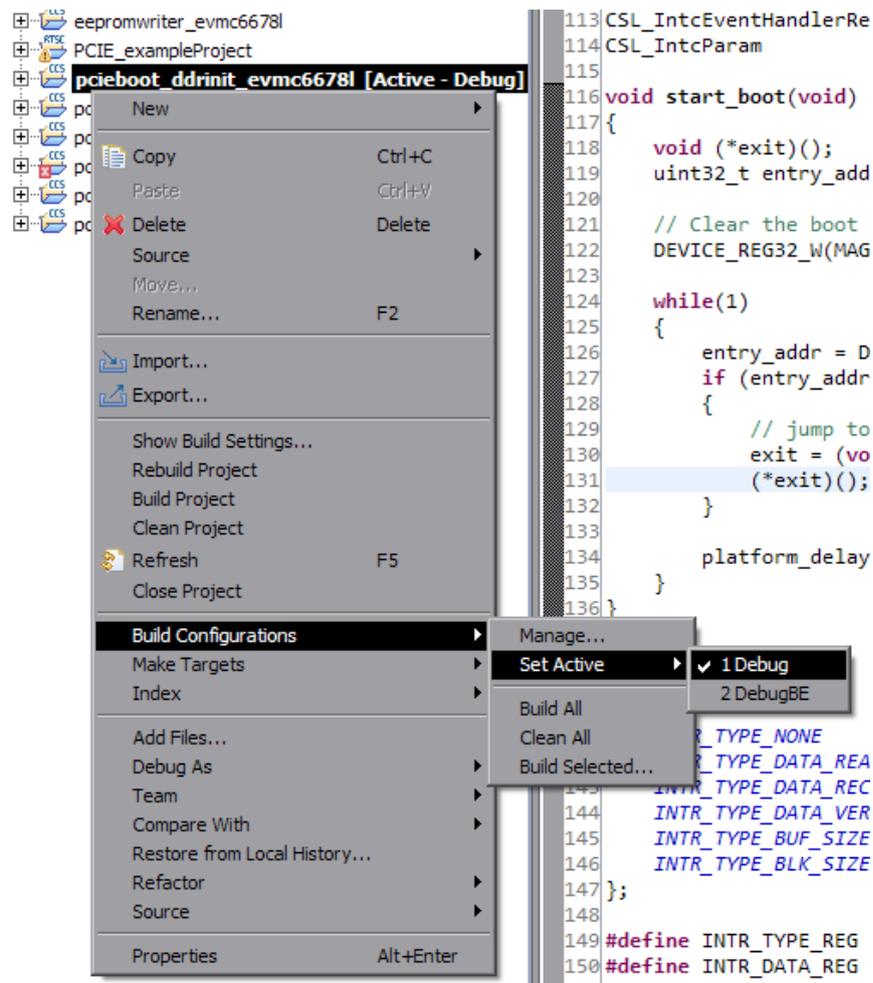


Рисунок 2-2: Выбор активной конфигурации сборки

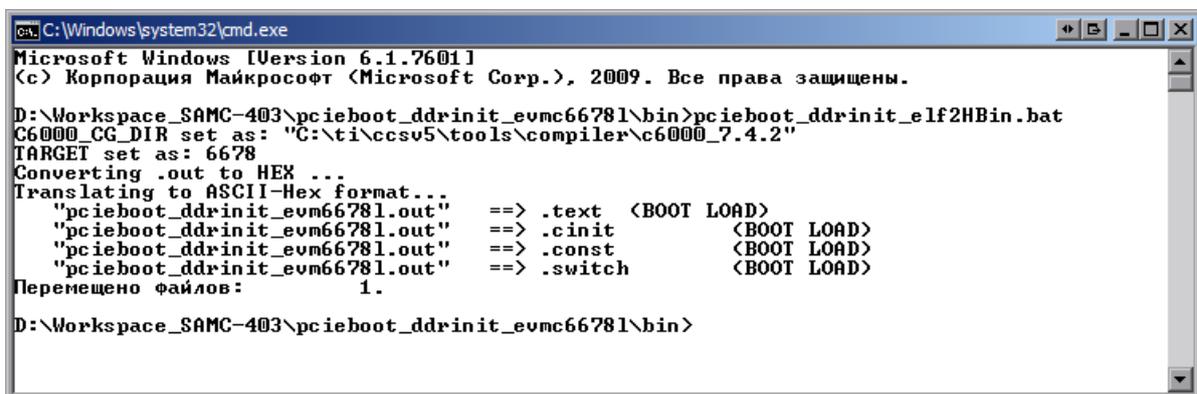
Далее, выполните очистку проекта (пункт меню «Clean Project») и сборку проекта (пункт меню «Build Project») через меню проекта, которое вызывается щелчком правой кнопки мыши на имени проекта в окне «Project Explorer».

В результате сборки проекта, будут сгенерированы файлы «pcieboot_ddrinit_evm6678l.map» и «pcieboot_ddrinit_evm6678l.out». Эти файлы будут расположены в папке «D:\Workspace_SAMC-403\pcieboot_ddrinit_evm6678l\bin».

В этой же папке находится командный файл «pcieboot_ddrinit_elf2HBin.bat». Этот файл необходимо запустить, в результате чего будут выполнены следующие преобразования:

- Используя утилиту «hexbх.exe» из состава CGT (Code Generation Tools) формат ELF файла «.out» будет преобразован в шестнадцатеричный ASCII формат файла таблицы загрузки;
- Используя утилиту «Vttbl2Hfile.exe» файл таблицы загрузки преобразуется в заголовочный текстовый файл;
- Используя утилиту «hfile2array.exe» выполняется преобразование заголовочного текстового файла в C-заголовочный файл «pcieDdrlnit.h» с массивом данных образа;
- Полученный файл «pcieDdrlnit.h» перемещается в папку «D:\Workspace_SAMC-403\linux_host_loader\LE» с именем «pcieDdrlnit_6678.h».

Вывод запуска командного файла «pcieboot_ddrinit_elf2HBin.bat» в системную консоль Windows должен выглядеть как показано на рисунке 2-3.



```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

D:\Workspace_SAMC-403\pcieboot_ddrinit_evm6678l\bin>pcieboot_ddrinit_elf2HBin.bat
C6000_CG_DIR set as: "C:\ti\ccsv5\tools\compiler\c6000_7.4.2"
TARGET set as: 6678
Converting .out to HEX ...
Translating to ASCII-Hex format...
"pcieboot_ddrinit_evm6678l.out" ==> .text <BOOT LOAD>
"pcieboot_ddrinit_evm6678l.out" ==> .cinit <BOOT LOAD>
"pcieboot_ddrinit_evm6678l.out" ==> .const <BOOT LOAD>
"pcieboot_ddrinit_evm6678l.out" ==> .switch <BOOT LOAD>
Перемещено файлов: 1.

D:\Workspace_SAMC-403\pcieboot_ddrinit_evm6678l\bin>

```

Рисунок 2-3: Вывод в консоль командного файла «pcieboot_ddrinit_elf2HBin.bat»

2.2 Пример «Hello World»

Данный пример находится в проекте «pcieboot_helloworld_evm6678l» (см. рисунок 2-4). Пример «DDR Init» использует библиотеку MCSDK Platform Library из состава PDK для инициализации UART, в который будет выведено сообщение «Hello World» и информация о загрузке программы на все ядра DSP.

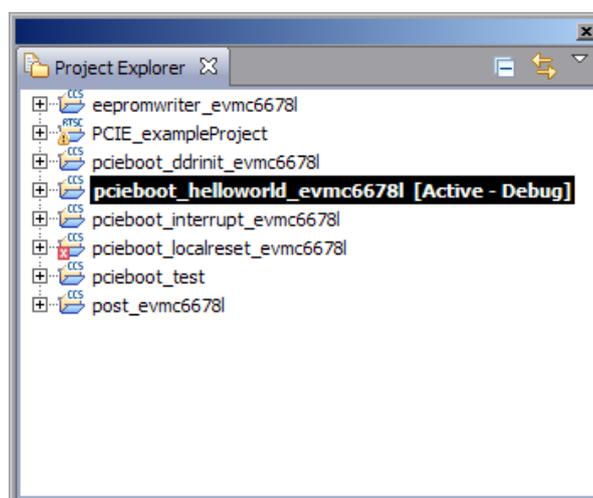


Рисунок 2-4: Окно «Project Explorer» с выбранным проектом примера «Hello World»

В примере «Hello World» часть DDR памяти, которая используется для общего кода для всех ядер DSP не может использоваться пользовательским приложением. Также, те части локальной L2 памяти, которые

используются индивидуально каждым из ядер DSP не могут быть использованы в пользовательском приложении. Адрес 0x0087FFFC (Magic Address) также нельзя использовать, так как он используется в процессе загрузки. Для подробной информации об использовании памяти приложением смотрите «.map» файл.

2.2.1 Процедура сборки

Процедура сборки примера «Hello World» аналогична процедуре сборки примера «DDR Init», которая описана в разделе 2.1.1.

C-заголовочный файл для примера «Hello World» имеет имя «pcieBootCode_6678.h», который будет помещен в папку «D:/Workspace_SAMC-403/linux_host_loader/LE», при запуске соответствующего командного файла из «bin» папки проекта. Командный файл для создания C-заголовочного файла «pcieBootCode_6678.h» имеет имя «helloworld_elf2Hbin.bat» и расположен в папке «bin» проекта примера («D:/Workspace_SAMC-403/pcieboot_helloworld_evmsc6678l»).

2.2.2 Описание процесса работы

Хост-система сначала загружает образ программы DDR инициализации в L2 память ядра 0. Затем записывает адрес точки входа в программу DDR инициализации по адресу 0x0087FFFC (Magic Address). Загрузка кода и запись производится через шину PCIe.

Когда SAMC-403 в режиме загрузки с PCIe, код IBL (Intermediate Boot Loader), работающий на ядре 0, проверят значение по адресу 0x0087FFFC (Magic Address) в бесконечном цикле. Когда это значение будет отличным от нуля, IBL осуществляет переход по этому адресу, тем самым выполняя загрузку кода программы DDR инициализации.

После корректной инициализации DDR памяти, значение по адресу 0x0087FFFC (Magic Address) обнуляется. Программа инициализации DDR памяти проверят значение по адресу 0x0087FFFC (Magic Address) в бесконечном цикле.

Хост-система выполняет загрузку образа примера «Hello World» в DDR память модуля SAMC-403 через PCIe. Затем записывает адрес точки входа в программу примера «Hello World» по адресу 0x0087FFFC (Magic Address).

Работающая программа DDR инициализации определяет, что значение по адресу 0x0087FFFC изменилось и выполняет переход по записанному адресу. Загружается код примера «Hello World», который выводит сообщение «Hello World» в UART и загружает код на всех остальных ядрах DSP путем записи адреса функции _c_int00 по адресу 0x0087FFFC (Magic Address) на всех остальных ядрах и отправкой IPC (Inter Process Communication) прерывания другим ядрам. Код на других ядрах переходит по адресу функции _c_int00 и загружается. На каждом ядре по адресу 0x0087FFFC (Magic Address) будет записано значение 0xBABEFACE путем вызова функции write_boot_magic_number().

2.2.3 Изменения в коде примера

В код примера «Hello World» внесены небольшие изменения, связанные с обработкой сигнала «hot reset» при перезагрузке модуля SAMC-514 (более подробно о проблеме написано в разделе 4.1).

Изменения заключаются в том, что после отработки оригинального кода примера TI, работа которого описана в разделе 2.2.2, выполняется код мониторинга состояния соединения PCIe, и в случае его обрыва выполняется сброс модуля SAMC-403. Фрагмент кода мониторинга регистра состояния LTSSM представлен в листинге 2-1.

Листинг 2-1: Код мониторинга состояния соединения PCIe

```

196 // PCIe link down polling (only on Core 0)
197 if (DNUM == 0)
198 {
199     write_uart("\r\n\r\nStarted PCIe link down polling on Core 0...");
200
201     while(1)
202     {
203         platform_delay(50);
204         debug0 = DEVICE_REG32_R(PCIE_DEBUG0);
205
206         if ((debug0 & 0x11) != 0x11) // LTSSM != 0x11 --- link is down

```

```

207     {
208         write_uart("\n\r-- PCIE link is down. Performing C6678 hard reset...\n\r");
209         c6678_hard_reset();           // performing C6679 hard reset
210     }
211 }
212 }
213 else
214 {
215     while(1);
216 }

```

Код функции выполняющей сброс C6678 на модуле SAMC-403 приведен в листинге 2-2.

Листинг 2-2: Код для выполнения сброса C6678

```

128 void c6678_hard_reset(void)
129 {
130     volatile unsigned int *pRSTCTRL = (volatile unsigned int *) (0x023100E8);
131     volatile unsigned int *pRSTCFG = (volatile unsigned int *) (0x023100EC);
132     volatile unsigned int key = 0x5A69;
133
134     *pRSTCTRL = key;           // enable writing to RSTCTRL and RSTCFG
135     *pRSTCFG = 0x00;          // setup reset type to hard reset
136     *pRSTCTRL = key;           // enable writing to RSTCTRL and RSTCFG
137     *pRSTCTRL = 0x00;          // perform reset
138 }

```

2.3 Пример «POST»

Данный пример находится в проекте «post_evmc6678l» (см. рисунок 2-5). Пример «POST» использует библиотеку MCSDK Platform Library из состава PDK для тестирования модуля SAMC-403. Результат тестирования выводится в UART модуля SAMC-403.

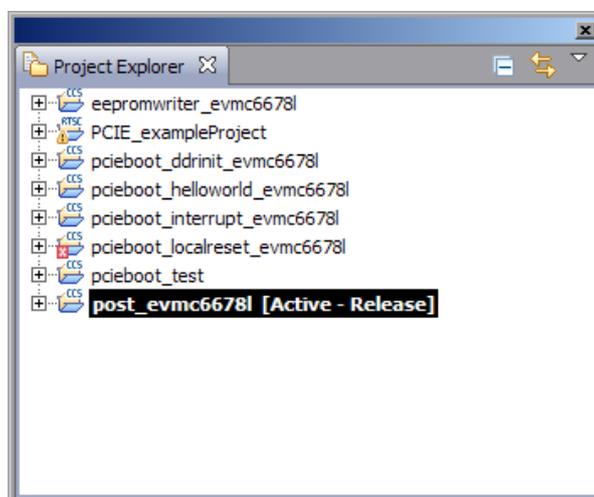


Рисунок 2-5: Окно «Project Explorer» с выбранным проектом примера «POST»

2.3.1 Процедура сборки

Процедура сборки примера «POST» аналогична процедурам сборки примеров «DDR Init» и «Hello World» (см. разделы 2.1.1 и 2.2).

2.3.2 Описание процесса работы

Пример «POST» использует только L2 память. Хост-система записывает код примера в L2 память ядра 0, затем записывает по адресу 0x0087FFFC (Magic Address) точку входа в пример «POST». Код IBL начинает загрузку примера «POST», загруженного в L2 память DSP.

2.4 Пример «EDMA-Interrupt»

Код данного примера находится в проекте «pcieboot_interrupt_evmc6678l» (см. рисунок 2-6). Пример демонстрирует передачу данных между памятью хост-системы и памятью DSP используя механизм EDMA через PCIe. При работе примера производится расчет пропускной способности PCIe. Пример также демонстрирует использование прерываний между хост-системой и DSP.

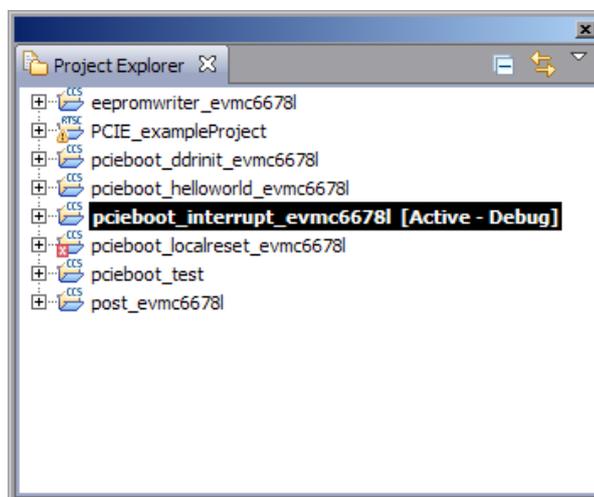


Рисунок 2-6: Окно «Project Explorer» с выбранным проектом примера «EDMA-Interrupt»

В примере «EDMA-Interrupt» DDR память с адреса 0x80000000 по 0x80400000 (4 Мбайт) используется для передачи данных через EDMA. Часть L2 памяти используется для «.out» файла. Использование данных областей памяти пользовательским приложением не допускается. Для подробной информации об использовании памяти приложением смотрите «.map» файл.

2.4.1 Процедура сборки

Процедура сборки примера «EDMA-Interrupt» аналогична процедурам сборки примеров «DDR Init» и «Hello World» (см. разделы 2.1.1 и 2.2).

2.4.2 Описание процесса работы

Пример «EDMA-Interrupt» использует только L2 память DSP. На стороне хост-системы код выполняет поиск PCI-устройства на шине, его инициализацию и регистрацию обработчика прерываний. Далее, на стороне хост-системы выполняется запись кода приложения в L2 память ядра 0 DSP через PCIe. Затем, по адресу 0x0087FFFC (Magic Address) DSP записывается адрес точки входа в код приложения. Код IBL начинает загрузку примера «EDMA-Interrupt», загруженного в L2 память DSP. Код примера выполняет инициализацию DDR памяти, конфигурирует прерывания используя CSL.

Далее, хост-система выполняет запись 4 Мбайт в DDR память DSP используя EDMA. По завершению записи, выполняется отправка прерывания DSP. После получения прерывания от хост-системы, обработчик прерываний на DSP выполняет простую манипуляцию с полученными данными в DDR памяти, после чего отправляет прерывание обратно хост-системе.

После получения прерывания от DSP, хост-система читает обратно 4 Мбайта данных из DDR памяти DSP, выполняет обратное преобразование данных и выполняет проверку правильности данных. Во время работы программы, выполняется подсчет пропускной способности операций EDMA чтения и записи.

Стоит отметить, что использование EDMA реализовано при помощи настройки регистров с хост-системы через PCIe соединение. С точки зрения DSP, это исходящие операции, которые инициированы локальным устройством (DSP) для записи или чтения с внешнего устройства (хост-системы). Функция HAL_readDMA() передает данные с DSP на хост-систему, это операция исходящей записи с точки зрения DSP.

Также, согласно спецификации PCIe, лагасу-прерывания не могут быть сгенерированы RC (только EP устройства). Генерация прерываний реализована путем вызова стандартного прерывания на стороне DSP путем использования одного из каналов событий (event input) контроллера прерываний INTC (Interrupt Controller). Это не является реальным сигналом прерывания, посланным через PCIe соединение.

Схема работы примера «EDMA-Interrupt» приведена на рисунке 2-7.

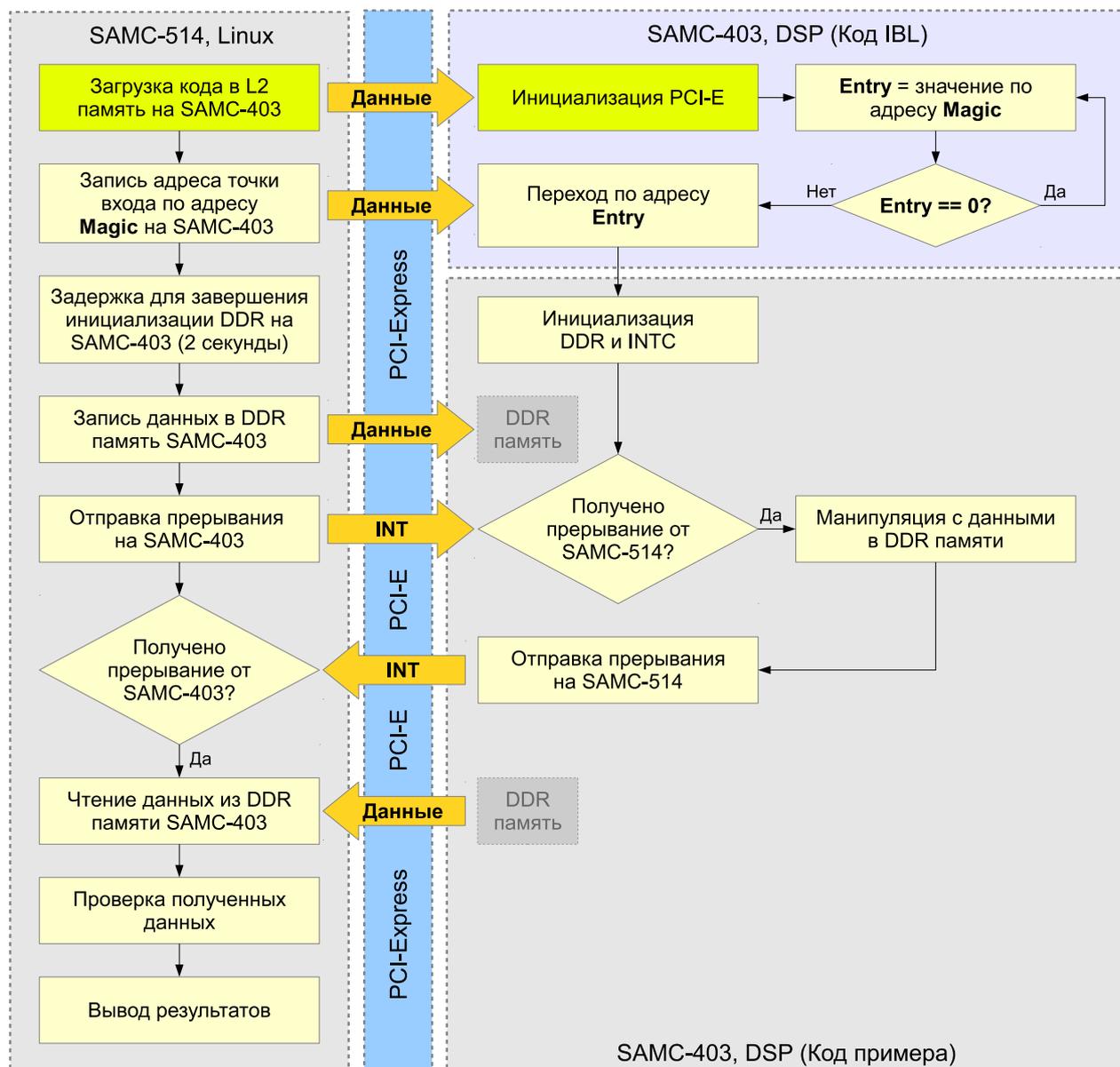


Рисунок 2-7: Схема работы примера «EDMA-Interrupt»

При просмотре пропускной способности PCIe через EDMA необходимо учитывать следующие факторы:

- Избыточность на физическом уровне составляет 8b/10b (восемь бит в десяти);
- На транспортном уровне, в режиме 32-х битной адресации, для служебной информации (SEQ, TLP, заголовков, ECRC, LCRC, и т. д.) используется 24-байта в дополнении к полезной нагрузке;
- Максимальный размер полезной нагрузки подсистемы PCIe архитектуры KeyStone составляет 128 байт для исходящих передач (outbound transfer) и 256 байт для входящих (inbound transfer). При использовании EDMA для исходящей передачи (outbound transfer) размер полезной нагрузки в TLP равен значению DBS (Data Burst Size) контроллера передачи EDMA если DBS меньше или равен максимальному значению максимальной полезной нагрузки PCIe. Здесь, DBS равен 128 байт, когда используются CC0 и TC0 EDMA.
- Скорость подключения PCIe. Узнать текущую скорость подключения PCIe можно выполнив на хост-системе команду `sudo lspci -vvv`. Например, ниже показана часть вывода этой команды со скоростью подключения $2.5 \times 2 = 5.0$ GT/s:

```
LnkSta: Speed 2.5GT/s, Width x2, TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
```

Таким образом, теоретическая максимальная пропускная способность PCIe на скорости подключения 5.0 GT/s будет равна $5.0 \text{ GT/s} \times 8/10 \times 128 / (128 + 24) = 3368 \text{ Гбит/с} = 421 \text{ Мбайт/с}$.

2.5 Пример «DSP Local Reset»

Исходный код примера «DSP Local Reset» расположен в проекте «pcieboot_localreset_evmc6678l» (см. рисунок 2-8). Данный пример проверяет адрес 0x0087FFFC (Magic Address) для повторной загрузки. Код загружается в локальную L2 память.

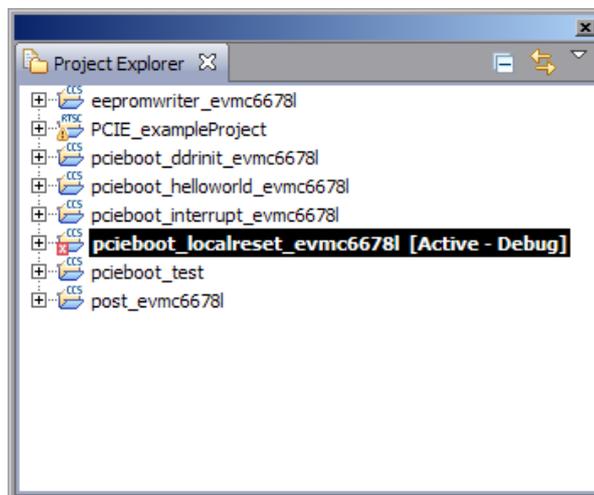


Рисунок 2-8: Окно «Project Explorer» с выбранным проектом примера «DSP Local Reset»

2.5.1 Процедура сборки

Процедура сборки примера «DSP Local Reset» аналогична процедурам сборки примеров «DDR Init» и «Hello World» (см. разделы 2.1.1 и 2.2).

2.5.2 Описание процесса работы

Пользователь может захотеть перезапустить демонстрационные примеры без перезапуска хост-системы. Для этого необходимо иметь возможность сброса DSP с хост системы. Существует несколько типов сброса: hard reset, soft reset и CPU local reset. Hard reset выполняет сброс всех устройств на DSP кроме PLL (Phase Locked Loop controller), тестов, эмуляции логики и модулей изолированных от сброса. PCIe не поддерживает изоляцию сброса, поэтому hard reset выполнит сброс PCIe модуля, при котором значения всех настроенных PCIe регистров будут потеряны. Soft reset ведет себя аналогично hard reset за исключением того, что значения PCIe регистров с установленным MMR битом будут сохранены. В обоих случаях, при hard reset и soft reset, хост-система после сброса не сможет выполнить соединение по PCIe с модулем SAMC-403.

Для выполнения сброса DSP, при сохранении не тронутой PCIe подсистему, пример «DSP Local Reset» выполняет следующие действия:

- Переводит все ядра в состояние сброса через PSC (Power Sleep Controller);
- Отключает питание всех модулей кроме PCIe через PSC;
- Выполняет конфигурацию регистров DSP_BOOT_ADDRn и IPCGRn. В этот момент код примера «DSP Local Reset» загружается на каждое ядро DSP через PCIe. После загрузки, значение адреса точки входа (адрес функции `_c_int00()`) записывается в регистры DSP_BOOT_ADDRn для каждого ядра. Регистр IPCGRn конфигурируется на переход по адресу точки входа в программу «DSP Local Reset», которая просто выполняет мониторинг значение по адресу 0x0087FFFC (Magic Address) для последующей загрузки;
- Включает все модули, которые ранее были отключены;
- Выводит все ядра из состояния сброса через PSC.

2.6 Загрузчик кода на SAMC-403

Код загрузчика расположен в папке «linux_host_loader» в папке рабочего пространства CCS.

Основные функции, которые выполняет код загрузчика:

- Выполнение отображения памяти между памятью хост-системы и памятью DSP. DSP запрашивает 4 блока памяти через маски PCIe регистров BAR0, BAR1, BAR2 и BAR3:
 - 1) Блок размером 4 Кбайт отображает область PCIe регистров DSP (BAR0);
 - 2) Блок размером 512 Кбайт отображает область локальной L2 памяти DSP (BAR1);
 - 3) Блок размером 4 Мбайт отображает область общей L2 памяти DSP (BAR2);
 - 4) Блок размером 16 Мбайт отображает область DDR памяти DSP (BAR3).

Маски BAR регистров конфигурируются внутри PCIe инициализационного кода, когда выбран режим PCIe загрузки на модуле SAMC-403 при помощи переключателей на плате.

- Конфигурация PCIe входящей трансляции адресов (inbound address translation) через PCIe регистры IB_BARn, IB_STARTn_LO, IB_STARTn_HI и IB_OFFSETn (n = 0, 1, 2, 3) на DSP. Значение данных регистров приведено в таблице 2-1.
- Предоставление API для чтения/записи DSP памяти:
 - Функция для чтения DSP памяти:

```
Uint32 ReadDSPMemory(Uint32 coreNum, Uint32 DSPMemAddr, Uint32 *buffer, Uint32 length)
```

- Функция для записи gldsp памяти:

```
Uint32 WriteDSPMemory(Uint32 coreNum, Uint32 DSPMemAddr, Uint32 *buffer, Uint32 length)
```

- Разбор массива данных загрузочного образа из заголовочного C-файла на предмет определения адреса загрузки, адреса точки входа, рамера секций, начального адреса секций. Осуществление загрузки образа в память DSP через API.
- Запись адреса загрузки по адресу 0x0087FFFC (Magic Address) на ядре 0 через API.
- Представление API для чтения/записи DSP памяти с использованием EDMA механизма:
 - Функция для чтения DSP памяти:

```
void HAL_readDMA(uint32_t srcAddr, uint32_t dstAddr, uint32_t size, uint32_t flag)
```

- Функция для записи gldsp памяти:

```
void HAL_writeDMA(uint32_t srcAddr, uint32_t dstAddr, uint32_t size, uint32_t flag)
```

Таблица 2-1: Значения PCIe регистров входящей трансляции адресов

n	IB_BARn	IB_STARTn_LO	IB_STARTn_HI	IB_OFFSETn
0	0x00000000	0xFE800000	0x00000000	0x21800000
1	0x00000001	0xFE800000	0x00000000	0x10800000
2	0x00000002	0xE5000000	0x00000000	0x0C000000
3	0x00000003	0xE4000000	0x00000000	0x80000000

2.6.1 Процедура сборки и запуска

Сборка кода загрузчика должна производиться на Linux системе, так как загрузчик является модулем ядра (драйвером).

Создайте папку «linux_host_loader» в домашней папке на Linux машине. Скопируйте в эту папку файлы «pciedemo.c», «Makefile», «pcieDdrInit_6678.h», «pcieBootCode_6678.h», «pcieInterrupt_6678.h» и «post_6678.h» из папки «D:/Workspace_SAMC-403/linux_host_loader» с Windows машины.

На Linux машине перейдите в созданную папку («~/linux_host_loader»):

```
cd ~/linux_host_loader
```

Находясь в папке «~/linux_host_loader» выполните команду:

```
make
```

В результате выполнения команды `make` будет собран файл модуля ядра «`pciedemo.ko`».

По-умолчанию, командой `make` будет собран пример «Hello World». Для выбора примера для сборки служат следующие макросы в файле «`pciedemo.c`»:

```
49 /* Must select which demo to run */
50 #define HELLO_WORLD_DEMO    1
51 #define POST_DEMO          0
52 #define EDMA_INTC_DEMO     0
53 #define LOCAL_RESET        0
```

Для сборки другого примера необходимо установить значение соответствующего макроса в 1. При сборке, необходимо, что-бы только один из этих макросов имел значение 1. В противном случае, сборка завершится с ошибками. После того, как требуемый макрос установлен в 1, выполните команду `make clean` для очистки, затем команду `make` для сборки «`pciedemo.ko`» с требуемым примером:

```
make clean
make
```

Для запуска собранного модуля ядра «`pciedemo.ko`» необходимо выполнить команду:

```
insmod pciedemo.ko
```

Для просмотра сообщений ядра, полученных при запуске модуля ядра, выполните команду

```
dmesg
```

Сообщения ядра, полученные при запуске всех описанных примеров приведены в приложении [A](#) данного документа.

Для выгрузки загруженного модуля ядра «`pciedemo.ko`» необходимо выполнить команду:

```
rmmmod pciedemo.ko
```

2.6.2 Изменения в коде загрузчика

В оригинальном коде загрузчика, номер прерывания, который назначается PCI устройству определяется в коде функции `PCI_FindPciDevices()` и записывается в переменную `irqNo`:

```
411 void PCI_FindPciDevices(void)
412 {
413     struct pci_dev *dev = NULL;
414
415     while ((dev = pci_get_device(PCI_ANY_ID, PCI_ANY_ID, dev)) != NULL)
416     {
417         if ((dev->vendor == PCIE_TI_VENDOR) && (dev->device == PCIE_TI_DEVICE)) {
418             printk("Found TI device\n");
419             irqNo = dev->irq;
420             PCIE_DEV = dev;
421             printk("TI device: vendor=0x%04x, dev=0x%04x, irq=0x%08x\n", dev->vendor, dev->device,
422                 <- dev->irq);
423             break;
424         }
425     }
```

Далее, после регистрации PCI устройства, для запроса прерывания для устройства, в вызов системной функции `request_irq()` передается номер прерывания из переменной `irqNo`:

```
1596 printk("Registering the irq %d ...\n", irqNo);
1597 request_irq(irqNo, ISR_handler, IRQF_SHARED, "TI 667x PCIE", &dummy);
```

При этом, вызов функции `PCI_FindPciDevices()`, в которой происходит определения номера прерывания, выполняется до вызова системных функций `pci_enable_device()`. Значение поля `irq` структуры `studct pci_dev` можно считать корректным только после вызова системной функции `pci_enable_device()`. Таким образом, на некоторых системах, значение запрашиваемого номера прерывания передаваемое в системную функцию `request_irq()` может быть не корректным, что и происходит при использовании модуля SAMC-514 в качестве хост-системы. В результате чего, при запуске примера «EDMA-Interrupt» (см. раздел 2.4) загрузчик не сможет зарегистрировать прерывания, и пример окажется не работоспособным.

Для устранения данной проблемы, из кода загрузчика была убрана переменная `irqNo` и изменен код функции `PCI_FindPciDevices()` и код запроса прерывания для устройства.

Измененная функция `PCI_FindPciDevices()` приведена к следующему виду:

```
410 void PCI_FindPciDevices(void)
411 {
412     struct pci_dev *dev = NULL;
413
414     while ((dev = pci_get_device(PCI_ANY_ID, PCI_ANY_ID, dev)) != NULL)
415     {
416         if ((dev->vendor == PCIE_TI_VENDOR) && (dev->device == PCIE_TI_DEVICE)) {
417             printk("Found TI device\n");
418             PCIE_DEV = dev;
419             printk("TI device: vendor=0x%04x, dev=0x%04x\n", dev->vendor, dev->device);
420             break;
421         }
422     }
423 }
```

Код запроса прерывания для устройства приведен к следующему виду:

```
1594 printk("Registering the irq %d ... \n", PCIE_DEV->irq);
1595 request_irq(PCIE_DEV->irq, ISR_handler, IRQF_SHARED, "TI 667x PCIE", &dummy);
```

3 Примеры Scan Engineering Telecom

3.1 Пример «Driver Example»

Пример «Driver Example» основан на примере TI «EDMA-Interrupt» (см. раздел 2.4). Пример состоит из двух частей — кода работающего на DSP и кода работающего на хост-системе, который выполняет загрузку DSP кода на SAMC-403.

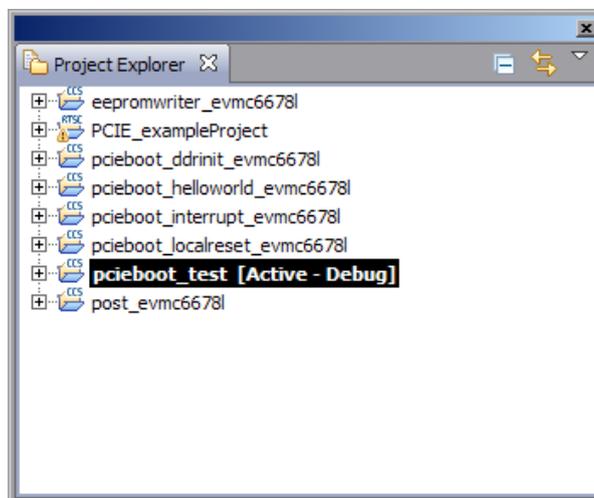


Рисунок 3-1: Окно «Project Explorer» с выбранным проектом примера «Driver Example»

Код примера «Driver Example», предназначенный для работы на DSP, расположен в проекте «pcieboot_test» (см. рисунок 3-1). Код примера «Driver Example», предназначенный для работы на хост-системе, расположен в папке «samc-403-linux-driver» в папке рабочего пространства CCS.

3.1.1 Процедура сборки

Сборка DSP части проекта осуществляется аналогично сборке примера TI «DDR Init» (см. раздел 2.1.1).

В результате сборки проекта, будут сгенерированы файлы «pcieboot_interrupt_evm6678l.map» и «pcieboot_interrupt_evm6678l.out». Эти файлы будут расположены в папке «D:/Workspace_SAMC-403/pcieboot_test/bin».

В этой же папке находится командный файл «interrupt_elf2HBin.bat». Этот файл необходимо запустить, в результате чего будут выполнены следующие преобразования:

- Используя утилиту «hex2h.exe» из состава CGT формат ELF файла «.out» будет преобразован в шестнадцатеричный ASCII формат файла таблицы загрузки;
- Используя утилиту «Btbl2Hfile.exe» файл таблицы загрузки преобразуется в заголовочный текстовый файл.
- Используя утилиту «hfile2array.exe» выполняется преобразование заголовочного текстового файла в C-заголовочный файл «pcieboot_interrupt.h» с массивом данных образа;
- Полученный файл «pcieboot_interrupt.h» перемещается в папку «D:/Workspace_SAMC-403/samc-403-linux-driver» с именем «pcieInterrupt_6678.h», который будет использован при сборке части примера для хост-системы.

Код примера для хост-системы расположен в папке «samc-403-linux-driver» в папке рабочего пространства CCS.

Сборка кода примера должна производиться на Linux системе. Создайте папку (например, «~/samc-403-linux-driver») на Linux машине. Скопируйте в эту папку все файлы из папки «samc-403-linux-driver» рабочего пространства CCS с Windows машины.

На Linux машине перейдите в папку « /samc-403-linux-driver» и выполните команду make:

```
cd ~/samc-403-linux-driver
make
```

В результате выполнения команды `make` будет получен файл модуля ядра «samc-403.ko».

Для загрузки собранного модуля ядра «samc-403.ko» необходимо выполнить команду:

```
insmod samc-403.ko
```

Для просмотра сообщений ядра, полученных при запуске модуля ядра, выполните команду

```
dmesg
```

Примеры вывода сообщений ядра хост-системы и UART модуля SAMC-403, полученные при запуске «Driver Example», приведены в приложении A (листинги A-8 и A-9).

Для выгрузки загруженного модуля ядра «samc-403.ko» необходимо выполнить команду:

```
rmmmod samc-403.ko
```

3.1.2 Описание процесса работы

Схема работы примера «Driver Example» приведена на рисунке 3-2.

На стороне хост-системы код выполняет поиск PCI-устройства на шине, его инициализацию и регистрацию обработчика прерываний. Диапазон памяти, который будет использован для передачи данных, добавляется в MTRR (Memory Type Range Registers).

Далее, на стороне хост-системы выполняется запись кода приложения в L2 память ядра 0 DSP через PCIe. Затем, по адресу 0x0087FFFC (Magic Address) DSP записывается адрес точки входа в код приложения. Код IBL на SAMC-403 начинает загрузку примера «Driver Example», загруженного в L2 память DSP. Код примера на DSP выполняет инициализацию DDR памяти, конфигурирует legacy-прерывания используя CSL, и инициализацию MSI прерываний. Далее, на стороне DSP происходит ожидание прерываний от хост-системы с информацией о размере пакета и общем объеме передаваемых данных.

Хост-система формирует два прерывания на DSP с информацией о размере пакета и общем объеме передаваемых данных. Для передачи дополнительной информации в прерывании используются PCIe регистры DSP общего назначения GPRn (n = 0, 1, ..., 3). По-умолчанию размер пакета установлен равным 16 Кбайт, общий объем передаваемых данных установлен в 512 Мбайт (полный объем DDR памяти на SAMC-403). Изменить данные значения можно отредактировав значения макросов MEM_BLOCK_SIZE (размер пакета) и MEM_BUFFER_SIZE (общий объем передаваемых данных) в файле «samc-403.h» проекта хост-системы:

```
92 #define MEM_BLOCK_SIZE          0x00004000    // 16KB
93 #define MEM_BUFFER_SIZE        0x20000000    // 512MB
94 #define MEM_BLOCKS_COUNT      (MEM_BUFFER_SIZE / MEM_BLOCK_SIZE)
```

Далее, на хост-системе запускается тасклет (tasklet), который выполняет запись данных в DDR память SAMC-403 через PCIe. Запись выполняется блоками, размер которых равен значению макроса MEM_BLOCK_SIZE. Общий объем записываемых данных равен значению макроса MEM_BUFFER_SIZE. После выполнения записи всех данных формируется прерывание «данные готовы» на DSP. Вычисляется скорость передачи данных, исходя из разницы временных меток «t1» и «t0» (см. рисунок 3-2).

На DSP при получении прерывания от хост-системы «данные готовы» формируется MSI прерывание на хост-систему «данные получены».

При получении прерывания «данные получены» от DSP выполняется вычисление скорости передачи данных, исходя из разницы временных меток «t2» и «t0» (см. рисунок 3-2).

При выводе работы примера в системную консоль, скорость передачи, вычисленная исходя из временных меток «t2» и «t0», отображается в скобках как значение «real»:

```
GPP transfered 512 MB of data to DSP with PCIe-E transfer speed 291.57 (real = 290.26) MB/s
```

На DSP выполняется верификация полученных данных в DDR памяти. По завершению верификации формируется MSI прерывание на хост-систему о завершении верификации с информацией о результате верификации. После верификации, DDR память DSP заполняется новыми данными и формируется MSI прерывание на хост-систему «данные готовы».

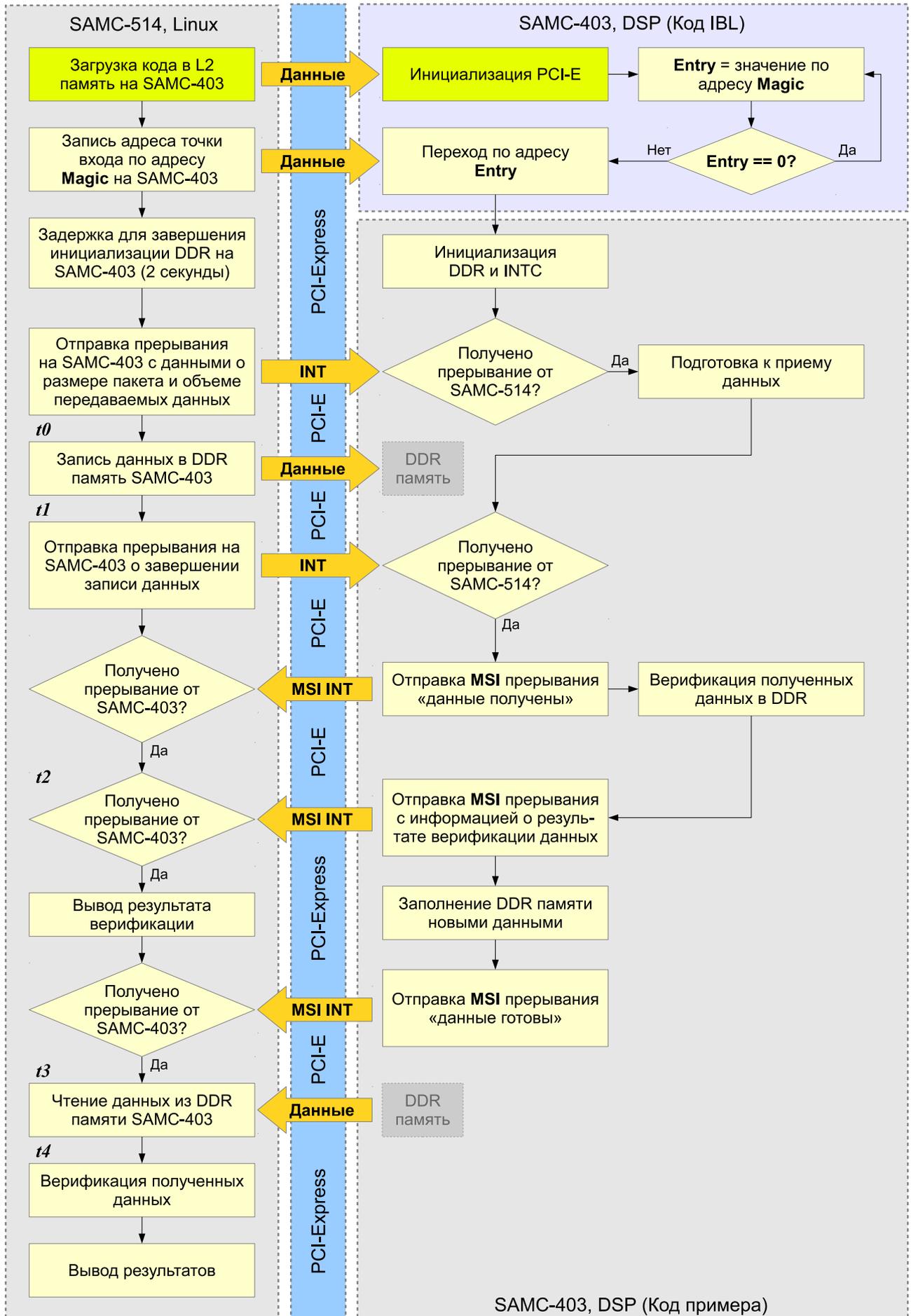


Рисунок 3-2: Схема работы примера «Driver Example»

На хост-системе, при получении прерывания от DSP с информацией о верификации, происходит вывод результатов верификации данных на стороне DSP. Далее, ожидается прерывание «данные готовы» от DSP.

Когда прерывание «данные готовы» от DSP получено, запускается тасклет (tasklet) чтения данных из DDR памяти SAMC-403 через PCIe. Тасклет выполняет чтение данных из DDR памяти SAMC-403 через PCIe. Чтение выполняется блоками, размер которых равен значению макроса MEM_BLOCK_SIZE. Общий объем считываемых данных равен значению макроса MEM_BUFFER_SIZE. После выполнения чтения всего объема данных, вычисляется скорость передачи данных, исходя из разницы временных меток «t4» и «t3» (см. рисунок 3-2). После, выполняется верификация полученных данных и выводится результат верификации.

Чтение и запись в примере «Driver Example» может осуществляться как с использованием механизма EDMA так и без него. По-умолчанию, включен режим передачи данных с EDMA. Изменить режим передачи данных можно отредактировав значение макроса USE_DMA в файле «samc-403.h» проекта хост-системы:

```
8 #define USE_DMA 1 // 0 - no, 1 - yes
```

Для отключения режима передачи данных EDMA необходимо установить значение макроса USE_DMA равным «0». В этом случае, для передачи данных будут использованы системные функции memscrw_toio() и memscrw_fromio().

При выгрузке модуля «samc-403.ko» командой rmmod происходит автоматический запуск кода примера «DSP Local Reset» (см. раздел 2.5). Поэтому, после выгрузки модуля «samc-403.ko», DSP сразу готов к повторному запуску примера.

Примеры вывода сообщений ядра хост-системы и UART модуля SAMC-403, полученные при запуске «Driver Example», приведены в приложении А (листинги А-8 и А-9).

4 Роль IBL в загрузке кода на SAMC-403

Код IBL записан в EEPROM память по адресу I²C шины 0x51. IBL выполняет обход проблемы блокировки PLL (проблема описана в документе TI «sprz334f.pdf»¹).

Проблема заключается в том, что для режимов ROM загрузки (EMAC, SRIO (Serial RapidIO), PCIe, Hyperlink и т. д.) и загрузке в режиме загрузки с EEPROM с адресом I²C шины 0x50, DSP пытается выполнить загрузку с EEPROM с адресом I²C шины 0x51. Таким образом, код IBL выполняет обход данной проблемы, путем записи в регистр DEVSTAT соответствующих значений на основе состояния физических переключателей режима загрузки на плате (SW3, SW4, SW5 и SW6). После чего выполняет повторный вход в ROM код для завершения требуемого режима загрузки.

Однако, повторный вход выполняется для всех режимов, кроме режима PCIe загрузки и режима загрузки с EEPROM с адресом I²C шины 0x51.

Ниже приведены шаги, которые выполняются кодом IBL в режиме PCIe загрузки:

- FPGA считывает данные режима загрузки (положение переключателей на плате);
- FPGA выполняет загрузку DSP с EEPROM с адресом I²C шины 0x51;
- С EEPROM с адресом I²C шины 0x51 загружается IBL;
- IBL выполняет корректную инициализацию PLL;
- IBL читает из регистра FPGA считанные им данные о режиме загрузки;
- IBL проверяет режим загрузки. Если это не режим загрузки с I²C или это режим загрузки с I²C адреса шины 0x50, то IBL записывает данные о режиме загрузки в регистр DEVSTAT.
- IBL проверяет режим загрузки, если это режим загрузки с PCIe, то выполняется код инициализации PCIe подсистемы. Далее, выполняется очистка адреса 0x0087FFFC (Magic Address), после чего происходит ожидание изменения значения адресу 0x0087FFFC (Magic Address) для загрузки.

Для работы примеров, использующих DDR память, описанных в данном документе, необходима правильная конфигурация DDR. Для функционирования IBL этого не требуется. DDR память может быть инициализированная несколькими способами:

- Хост-система инициализирует DDR регистры напрямую через PCIe соединение;
- На модуль SAMC-403 в L2 память загружается приложение инициализации DDR памяти, которое после инициализации DDR памяти обнуляет адрес 0x0087FFFC (Magic Address) и ожидает дальнейшей загрузки.

В примерах, описанных в разделе 2, используется второе решение. Иллюстрация процесса загрузки показана на рисунке 4-1.

4.1 Изменения в коде IBL

При включении модуля SAMC-514, его BIOS посылает по шине PCIe сигнал hot reset и разрывает соединение. Для правильной работы, сигнал hot reset должен быть корректно обработан модулем SAMC-403. Оригинальный код IBL подобной обработки не имеет. В связи с этим, на сопроводительном диске к модулю SAMC-403 в папке «ibl» находится загрузчик IBL, который модифицирован для корректной обработки сигнала hot reset от модуля SAMC-514.

Для корректной работы примеров, описанных в данном документе, необходимо использовать IBL, который находится в папке «ibl» на сопроводительном диске к модулю SAMC-403.

В качестве основы для модификации IBL взят оригинальный код IBL версии 1.0.0.16.

¹ TMS320C6678 Silicon Errata, Advisory 8, «Multiple PLLs May Not Lock After Power-on Reset Issue»

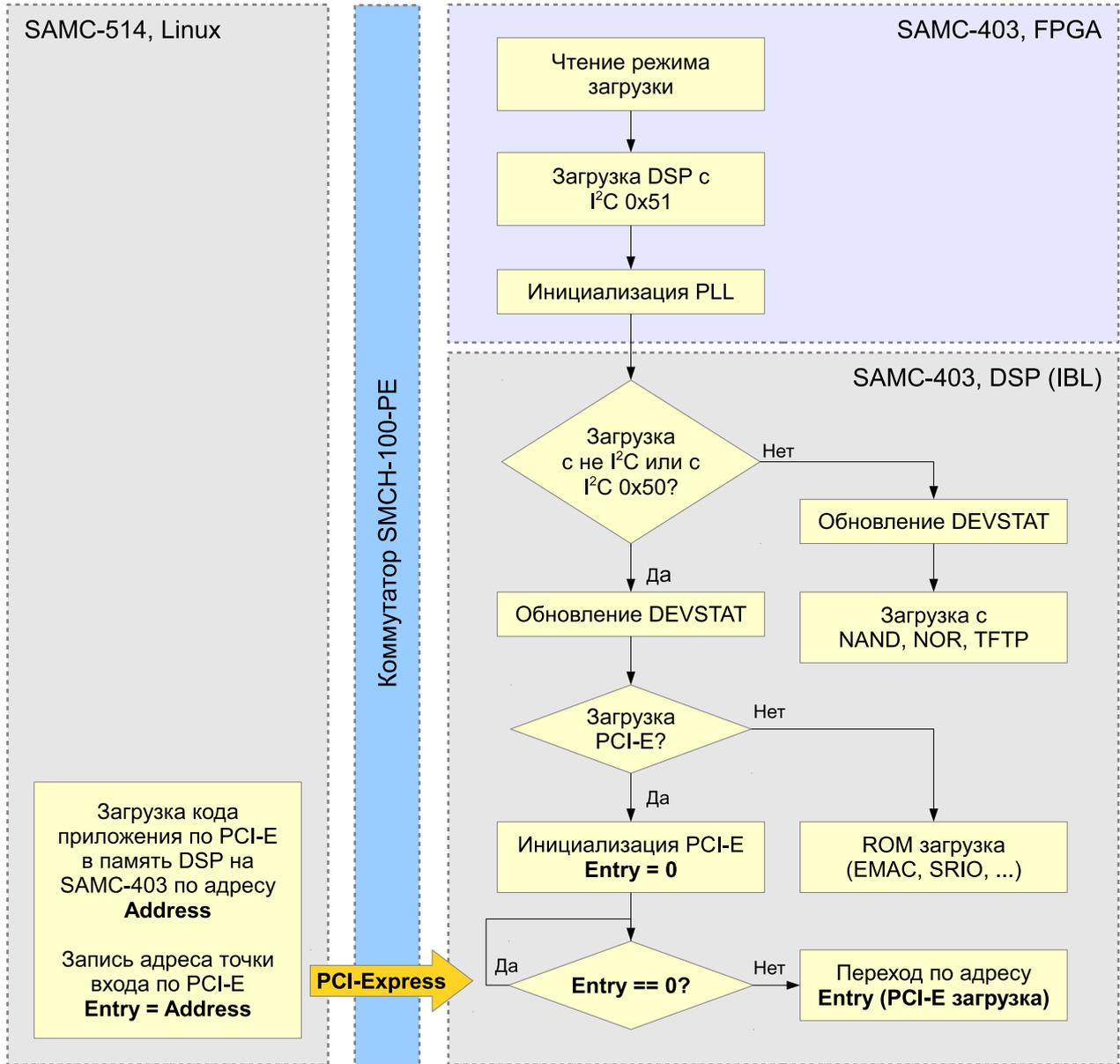


Рисунок 4-1: Схема процесса загрузки кода на модуль SAMC-403 с модуля SAMC-514

5 Тестовая установка и результаты

Тестовая установка без использования PCIe коммутатора была собрана с использованием:

- одного модуля SAMC-403 с прошитым в EEPROM на шине I²C с адресом 0x51 загрузчиком IBL с сопроводительного диска к модулю SAMC-403;
- одного модуля SAMC-514 с установленной на внутренний SSD системой Linux Ubuntu 10.04 x86_64;
- шасси ELMA Blu!One 3000.

Схема тестовой установки без коммутатора приведена на рисунке 5-1. При использовании тестовой установки с прямым соединением (без коммутатора), максимальная скорость соединения по PCIe между SAMC-403 и SAMC-514 будет составлять x1, 5.0 GT/s.

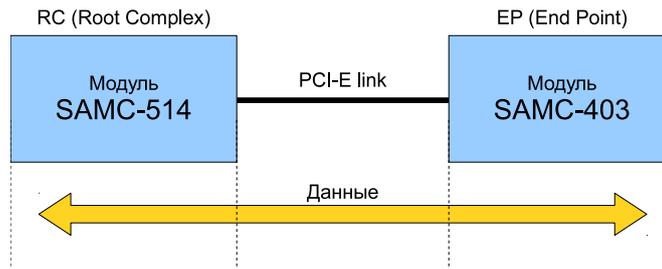


Рисунок 5-1: Схема тестовой установки с прямым соединением (шасси ELMA Blu!One 3000)

При использовании данной схемы тестовой установки, модуль SAMC-403 был установлен в слот AMC-х, модуль SAMC-514 в слот AMC-х шасси ELMA Blu!One 3000. Физическое расположение AMC-слотов в шасси ELMA Blu!One 3000 приведено на рисунке 5-2.

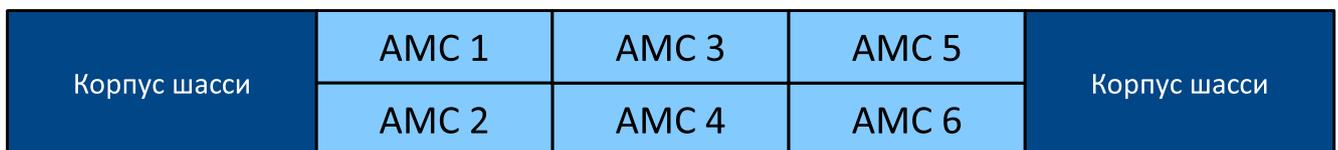


Рисунок 5-2: Расположение AMC-слотов в шасси ELMA Blu!One 3000

Для сборки тестовой установки с PCIe коммутатором, помимо модулей SAMC-403 и SAMC-514, были использованы:

- шасси MicroBlade 2U;
- коммутатор SMCH-100-PE.

Схема тестовой установки с коммутатором SMCH-100-PE приведена на рисунке 5-3. При использовании тестовой установки с коммутатором SMCH-100-PE, максимальная скорость соединения по PCIe между SAMC-403 и SAMC-514 будет составлять x2, 5.0 GT/s.

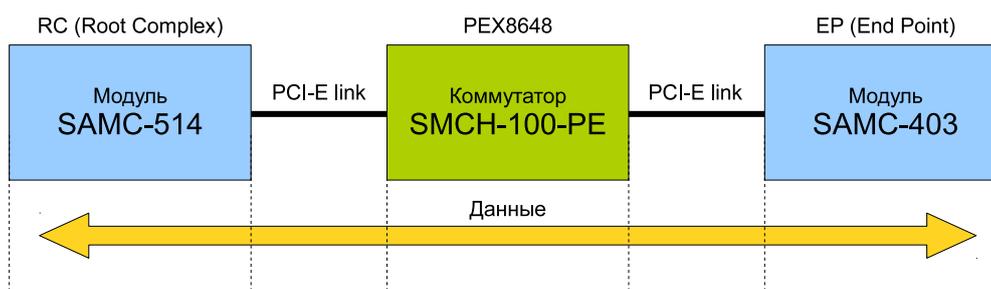


Рисунок 5-3: Схема тестовой установки с коммутатором SMCH-100-PE (шасси MicroBlade 2U)

При использовании данной схемы тестовой установки, коммутатор SMCH-100-PE был установлен в слот MCH-1, модуль SAMC-403 в слот AMC-6 и модуль SAMC-514 в слот AMC-2 шасси MicroBlade 2U. На коммутаторе SMCH-100-PE в качестве PCIe upstream порта был назначен слот AMC-6, в который установлен модуль SAMC-514. Информация по настройке upstream порта на коммутаторе SMCH-100-PE приведена в его документации. Физическое расположение AMC-слотов в шасси MicroBlade 2U приведено на рисунке 5-4.

Блок охлаждения	Блок питания	MCH 1	AMC 4	AMC 8	AMC 12	Блок охлаждения
	Заглушка		Заглушка	AMC 3	AMC 7	
	Блок питания	MCH 2	AMC 2	AMC 6	AMC 10	
	Заглушка		Заглушка	AMC 1	AMC 5	

Рисунок 5-4: Расположение AMC-слотов в шасси MicroBlade 2U

Таблица 5-1: Положение переключателей модуля SAMC-403 для включения режима загрузки по PCIe

Переключатель	1	2	3	4
SW3	OFF	ON	ON	OFF
SW4	ON	ON	ON	ON
SW5	ON	ON	ON	OFF
SW6	OFF	ON	ON	ON
SW9	OFF	ON	—	—

Для запуска примеров выполните действия, описанные в процедуре 5-1.

Процедура 5-1. Запуск примеров

1. Убедитесь в том, что на модуле SAMC-403 используется версия IBL с сопроводительного диска к модулю SAMC-403. Для сборки IBL или прошивки уже собранного образа, который имеется на диске в папке «`ibl/src/make/bin`» руководитесь инструкциями, приведенными в документе [1], который имеется на сопроводительном диске к модулю SAMC-403 в папке «`docs`».
2. Установите переключатели на плате SAMC-403 в соответствии с таблицей 5-1.
3. Установите модули SAMC-403 и SAMC-514 в шасси;
4. Для возможности просмотра UART сообщений на модуле SAMC-403, соедините Mini-USB кабелем компьютер и фронтальный Mini-USB выход на модуле SAMC-403.
5. Включите питание шасси;
6. Включите модуль SAMC-403 (модуль SAMC-514 должен быть выключен);
7. Выполните настройку UART соединения с модулем SAMC-403;
8. Нажмите на модуле SAMC-403 кнопку «Full Reset». В UART выводе должны появиться следующие сообщения:

```
IBL INIT
-- Detected PCIe boot mode
-- PCIe bootloader build date: May 31 2013, time: 16:35:24
-- Device is C6678
-- PCIe boot magic address is 0x0087FFFC
* Register SERDES_STS = 0x0000020D
- PLL locked
- Lane0 loss of signal detected
- Lane1 loss of signal detected
* Register PCS_STATUS = 0x65006465
* Register DEBUG0 = 0x00800934
```

```

* Register DEBUG1      = 0x30303830
-- Provide PLL reference clock to SerDes inside PCIESS...
-- Power down PCIE...
-- Power up PCIE...
-- Program PLL settings and enable PLL
-- Wait for PCIE PLL lock...
-- PLL configured
-- Disable link training...
-- Writing PCIE registers...
-- Configuring link speed (x1, 5.0GT/s)...
-- Configuring BAR registers...
-- Enabling MSI interrupts...
* Register SERDES_STS = 0x000020D
- PLL locked
- Lane0 loss of signal detected
- Lane1 loss of signal detected
* Register PCS_STATUS = 0x0001300
* Register DEBUG0     = 0x00007100
* Register DEBUG1     = 0x08200000
-- Enable link training...
-- Waiting for link up...

```

9. Включите модуль SAMC-514. Продолжение вывода в UART модуля SAMC-403 должно выглядеть подобно следующему:

```

-- Waiting for link up...
-- Link is up
* Register SERDES_STS = 0x0000201
- PLL locked
- Lane0 is ok
- Lane1 loss of signal detected
* Register PCS_STATUS = 0x0001111
* Register DEBUG0     = 0x03F77A11
* Register DEBUG1     = 0x08000010
-- Waiting for boot from host...

```

10. Для того что-бы убедиться, что модуль SAMC-403 корректно определился в системе необходимо после загрузки системы на модуле SAMC-514 выполнить в консоле команду `lspci -n`:

```

root@ubuntu-samc514:~# lspci -n
00:00.0 0600: 8086:0104 (rev 09)
00:01.0 0604: 8086:0101 (rev 09)
00:01.1 0604: 8086:0105 (rev 09)
00:01.2 0604: 8086:0109 (rev 09)
00:16.0 0780: 8086:1c3a (rev 04)
00:1a.0 0c03: 8086:1c2d (rev 05)
00:1c.0 0604: 8086:1c10 (rev b5)
00:1c.4 0604: 8086:1c18 (rev b5)
00:1d.0 0c03: 8086:1c26 (rev 05)
00:1f.0 0601: 8086:1c4f (rev 05)
00:1f.2 0106: 8086:1c03 (rev 05)
00:1f.3 0c05: 8086:1c22 (rev 05)
00:1f.6 1180: 8086:1c24 (rev 05)
01:00.0 0480: 104c:b005 (rev 01)
02:00.0 ff00: 8086:10a6 (rev 01)
02:00.2 0200: 8086:1510 (rev 01)
02:00.3 0200: 8086:1510 (rev 01)

```

В выводе данной команды должна присутствовать строка:

```
01:00.0 0480: 104c:b005 (rev 01)
```

11. PCIe регистры `BARn` ($n = 0, 1, 2, \dots, 5$), записанные Linux системой во время перечисления шины PCI, не должны быть нулевыми. В случае, если к модулю подключен JTAG эмулятор, значения регистров можно проверить подключившись к модулю SAMC-403 из CCS (см. рисунок 5-5).

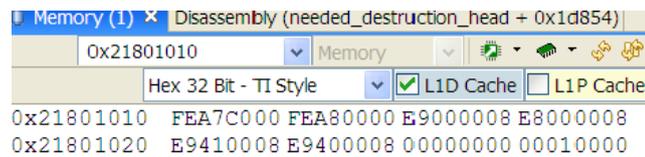


Рисунок 5-5: Проверка значений PCIe регистров BARn в CCS

12. Выполните сборку модуля ядра «pciedemo.ko». Процесс сборки описан в соответствующих подразделах раздела 2.
13. Загрузите модуль ядра «pciedemo.ko» выполнив в системной консоле команду:

```
sudo insmod pciedemo.ko
```

В приложении A приведен вывод в UART модуля SAMC-403 и сообщения ядра на модуле SAMC-514 для каждого из примеров, описанных в данном документе:

14. Для выгрузки модуля ядра, выполните в консоле команду:

```
sudo rmmmod pciedemo.ko
```

Приложение А Вывод в UART и системную консоль примеров

А.1 Пример «Hello World»

Листинг А-1: Вывод в UART примера «Hello World»

```
PCIE Boot Hello World Example Version 01.00.00.00
Booting Hello World image on Core 0 from PCIE ...
Booting Hello World image on Core 1 from Core 0 ...
Booting Hello World image on Core 2 from Core 0 ...
Booting Hello World image on Core 3 from Core 0 ...
Booting Hello World image on Core 4 from Core 0 ...
Booting Hello World image on Core 5 from Core 0 ...
Booting Hello World image on Core 6 from Core 0 ...
Booting Hello World image on Core 7 from Core 0 ...
Started PCIE link down polling on Core 0...
```

Листинг А-2: Вывод в системную консоль примера «Hello World»

```
[ 675.725621] Finding the device....
[ 675.729381] Found TI device
[ 675.732433] TI device: vendor=0x104c, dev=0xb005
[ 675.737480] Reading the BAR areas....
[ 675.743163] Enabling the device....
[ 675.746991] pci 0000:01:00.0: PCI INT A -> GSI 16 (level, low) -> IRQ 16
[ 675.754313] pci 0000:01:00.0: setting latency timer to 64
[ 675.760228] Access PCIE application register ....
[ 675.765370] Registering the irq 16 ...
[ 675.769512] Boot entry address is 0x1082cf00
[ 675.776447] Total 4 sections, 0xd998 bytes of data were written
[ 675.854140] Boot entry address is 0x8000d100
[ 675.860905] Total 4 sections, 0xdd5c bytes of data were written
```

A.2 Пример «POST»

Листинг А-3: Вывод в UART примера «POST»

```
TMDXEVM6678L POST Version 01.00.00.06
-----
SOC Information

FPGA Version: 000B
Board Serial Number: 0DCE5330
EFUSE MAC ID is: 90 D7 EB 0D 75 16
SA is disabled on this board.
PLL Reset Type Status Register: 0x00000004
Platform init return code: 0x00000000
Additional Information:
(0x02350014) :0BEF0000
(0x02350624) :000215FF
(0x02350678) :00831F70
(0x0235063C) :00081800
(0x02350640) :00091800
(0x02350644) :000A1800
(0x02350648) :000B1800
(0x0235064C) :000C1800
(0x02350650) :000D1800
(0x02350654) :000E1800
(0x02350658) :000F1800
(0x0235065C) :00000009
(0x02350660) :00832038
(0x02350668) :0083204C
(0x02350670) :00832060
(0x02620008) :05013009
(0x0262000c) :0401412E
(0x02620010) :00000000
(0x02620014) :43800020
(0x02620018) :0009E02F
(0x02620180) :0602F000
-----

Power On Self Test

POST running in progress ...
POST I2C EEPROM read test started!
POST I2C EEPROM read test passed!
POST SPI NOR read test started!
POST SPI NOR read test passed!
POST EMIF16 NAND read test started!
POST EMIF16 NAND read test passed!
POST EMAC loopback test started!
POST EMAC loopback test passed!
POST external memory test started!
POST external memory test passed!
POST done successfully!

POST result: PASS
```

Листинг А-4: Вывод в системную консоль примера «POST»

```
[ 1666.444535] Finding the device....
[ 1666.448314] Found TI device
[ 1666.451353] TI device: vendor=0x104c, dev=0xb005
[ 1666.456412] Reading the BAR areas....
[ 1666.462247] Enabling the device....
[ 1666.466087] pci 0000:01:00.0: setting latency timer to 64
[ 1666.471974] Access PCIe application register ....
[ 1666.477121] Registering the irq 16 ...
[ 1666.481244] Boot entry address is 0x 83a6e0
[ 1666.487792] Total 3 sections, 0xb2b0 bytes of data were written
```

A.3 Пример «EDMA-Interrupt»

Листинг A-5: Вывод в UART примера «EDMA-Interrupt»

```
Debug: GEM-INTC Configuration...
Debug: GEM-INTC Configuration Completed
Debug: CPINTC-0 Configuration...
Debug: CPINTC-0 Configuration Completed
DSP receives interrupt from host.
DSP generates interrupt to host.
```

Листинг A-6: Вывод в системную консоль примера «EDMA-Interrupt»

```
[ 41.958389] Finding the device...
[ 41.962019] Found TI device
[ 41.964984] TI device: vendor=0x104c, dev=0xb005
[ 41.969870] Reading the BAR areas...
[ 41.974430] Enabling the device...
[ 41.978166] pci 0000:01:00.0: PCI INT A -> GSI 16 (level, low) -> IRQ 16
[ 41.985285] pci 0000:01:00.0: setting latency timer to 64
[ 41.991042] Access PCIE application register ....
[ 41.995998] Registering the irq 16 ...
[ 41.999997] Allocating consistent memory ...
[ 42.011774] Boot entry address is 0x 82e640
[ 42.017448] Total 5 sections, 0xf5e4 bytes of data were written
[ 44.092460] Write DMA to DSP ...
[ 44.108442] Generating interrupt to DSP ...
[ 44.237658] Interrupt 16 received from DSP
[ 44.242038] Read DMA from DSP ...
[ 44.261506] DMA test passed!
[ 45.111683] DMA write throughput is: 318.92 MB/s
[ 45.116550] DMA read throughput is: 339.09 MB/s
[ 45.121333] Freeing consistent memory ...
```

A.4 Пример «DSP Local Reset»

Вывод в UART модуля SAMC-403 в примере «DSP Local Reset» отсутствует. Вывод происходит только в системную консоль на модуле SAMC-514.

Листинг A-7: Вывод в системную консоль примера «DSP Local Reset»

```
[ 1515.234980] Finding the device....
[ 1515.238780] Found TI device
[ 1515.241834] TI device: vendor=0x104c, dev=0xb005
[ 1515.246893] Reading the BAR areas....
[ 1515.252710] Enabling the device....
[ 1515.256547] pci 0000:01:00.0: setting latency timer to 64
[ 1515.262457] Access PCIIE application register ....
[ 1515.267622] Registering the irq 16 ...
[ 1525.260760] Start local reset assert for core (module id): 15 ...
[ 1525.267425] Start local reset assert for core (module id): 16 ...
[ 1525.274107] Start local reset assert for core (module id): 17 ...
[ 1525.280774] Start local reset assert for core (module id): 18 ...
[ 1525.287419] Start local reset assert for core (module id): 19 ...
[ 1525.294075] Start local reset assert for core (module id): 20 ...
[ 1525.300725] Start local reset assert for core (module id): 21 ...
[ 1525.307371] Start local reset assert for core (module id): 22 ...
[ 1525.321098] Boot entry address is 0x 878000
[ 1525.325832] Total 3 sections, 0x7a0 bytes of data were written
[ 1525.333189] Boot entry address is 0x 878000
[ 1525.337933] Total 3 sections, 0x7a0 bytes of data were written
[ 1525.345299] Boot entry address is 0x 878000
[ 1525.350037] Total 3 sections, 0x7a0 bytes of data were written
[ 1525.357403] Boot entry address is 0x 878000
[ 1525.362143] Total 3 sections, 0x7a0 bytes of data were written
[ 1525.369504] Boot entry address is 0x 878000
[ 1525.374226] Total 3 sections, 0x7a0 bytes of data were written
[ 1525.381578] Boot entry address is 0x 878000
[ 1525.386343] Total 3 sections, 0x7a0 bytes of data were written
[ 1525.393728] Boot entry address is 0x 878000
[ 1525.398465] Total 3 sections, 0x7a0 bytes of data were written
[ 1525.405825] Boot entry address is 0x 878000
[ 1525.410574] Total 3 sections, 0x7a0 bytes of data were written
[ 1525.434981] MD stat for pid 2 mid 9 state: 3 timeout
[ 1525.443408] Start local reset de-assert for core (module id): 15 ...
[ 1525.450321] Start local reset de-assert for core (module id): 16 ...
[ 1525.457243] Start local reset de-assert for core (module id): 17 ...
[ 1525.464185] Start local reset de-assert for core (module id): 18 ...
[ 1525.471120] Start local reset de-assert for core (module id): 19 ...
[ 1525.478068] Start local reset de-assert for core (module id): 20 ...
[ 1525.484966] Start local reset de-assert for core (module id): 21 ...
[ 1525.491884] Start local reset de-assert for core (module id): 22 ...
```

A.5 Пример «Driver Example»

Листинг А-8: Вывод в UART примера «Driver Example»

```
Debug: GEM-INTC Configuration...
Debug: GEM-INTC Configuration Completed
Debug: CPINTC-0 Configuration...
Debug: CPINTC-0 Configuration Completed
Received interrupt with buffer size information. Buffer size = 536870912Kb
Received interrupt with block size information. Block size = 16Kb
Sending interrupt to GPP (DATA_RECEIVED)
Sending interrupt to GPP (DATA_VERIFY)
Sending interrupt to GPP (DATA_READY)
```

Листинг А-9: Вывод в системную консоль примера «Driver Example»

```
[ 93.869749] Finding the device...
[ 93.873352] Founded SAMC-403 device
[ 93.877151] TI device: VENDOR=0x104c, DEVICE=0xb005
[ 93.882484] Reading the BAR areas...
[ 93.888024] * BAR0: physical = 0xe0100000, virtual = 0xffffc9000584c000, length = 4096
[ 93.896671] * BAR1: physical = 0xe0c00000, virtual = 0xffffc90005880000, length = 524288
[ 93.905531] * BAR2: physical = 0xe0800000, virtual = 0xffffc90005980000, length = 4194304
[ 93.914485] * BAR3: physical = 0xe1000000, virtual = 0xffffc90005e00000, length = 16777216
[ 93.923504] Enabling the device...
[ 93.927192] pci 0000:01:00.0: PCI INT A -> GSI 16 (level, low) -> IRQ 16
[ 93.934529] pci 0000:01:00.0: setting latency timer to 64
[ 93.940441] Access PCI-E application register...
[ 93.945411] MTRR initialization...
[ 93.957223] Added region at 0xe1000000 of size 0x1000000 to MTRR region
[ 93.964442]   alloc irq_desc for 48 on node -1
[ 93.969290]   alloc kstat_irqs on node -1
[ 93.973683] pci 0000:01:00.0: irq 48 for MSI/MSI-X
[ 93.978936] MSI support is enabled
[ 93.982628] Registering the IRQ 48 for SAMC-403...
[ 93.987858] Boot entry address is 0x 82eb60
[ 93.995146] Total 5 sections, 0xfb78 bytes of data were written
[ 96.070567] Allocating memory for Rx buffer...
[ 96.075527] Allocating memory for Tx buffer...
[ 98.341526] GPP transfered 512 MB of data to DSP with PCIe-E transfer speed 291.57 (real =
  ← 290.26) MB/s
[ 98.357401] DSP verified received data with status [OK]
[ 104.795895] DSP prepared data. Starting data receiver tasklet.
[ 107.163716] GPP transfered 512 MB of data from DSP with speed 339.61 MB/s
[ 107.171118] Data verification status is [OK] (0/32768 blocks)
```

Список литературы

1. SAMC-403. Загрузчик IBL. Руководство пользователя. [UG-SAMC-403-IBL](#) (цит. на с. 25).