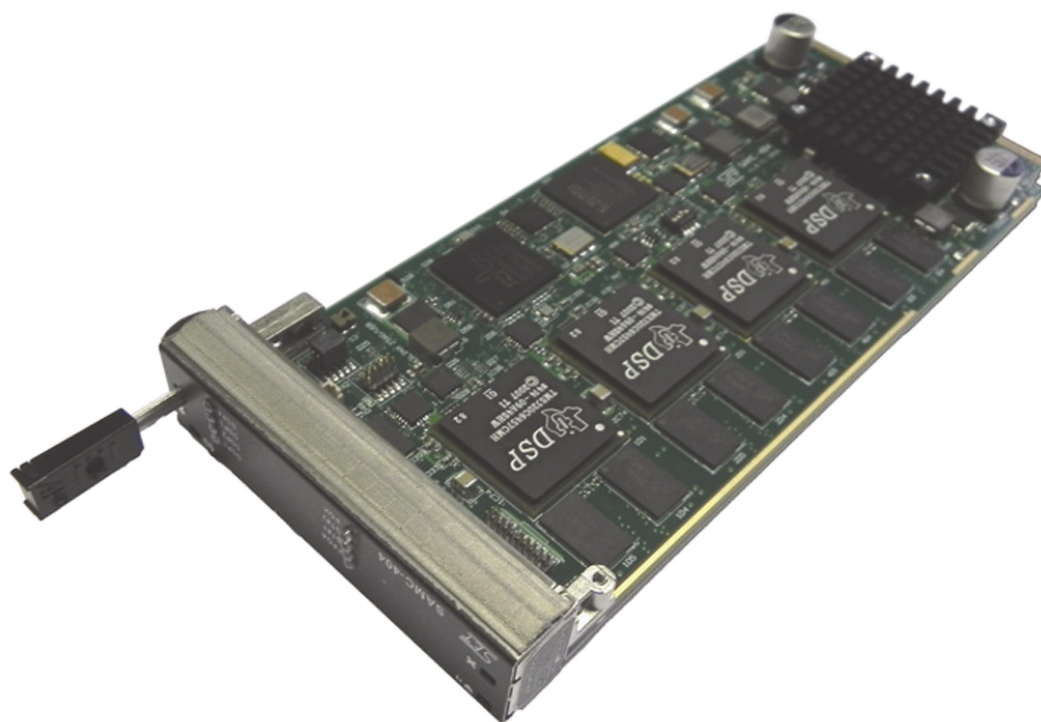




## SAMC-404. Сборка и запуск системы Linux-c6x

Руководство пользователя

Версия 1.0



Код документа: UG-SAMC-404-LC6X

Дата сборки: 27 мая 2015 г.

Листов в документе: 25

© 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 .....	4
<b>1 Общие сведения .....</b>	<b>5</b>
<b>2 Сборка Linux-c6x .....</b>	<b>6</b>
2.1 Настройка системы .....	6
2.2 Получение исходных кодов Linux-c6x .....	6
2.3 Установка зависимостей и конфигурация .....	7
2.4 Сборка .....	8
<b>3 Создание образов Linux-c6x .....</b>	<b>9</b>
3.1 Образ системы для NFS корневой файловой системы .....	9
3.2 Образ системы для ROMFS файловой системы .....	13
<b>4 Запись образов Linux-c6x на модуль .....</b>	<b>14</b>
<b>5 Загрузка Linux-c6x .....</b>	<b>15</b>
<b>6 Подключение к Linux-c6x .....</b>	<b>17</b>
6.1 Подключение по Telnet .....	17
6.2 Использование Netconsole .....	19
<b>7 Передача файлов в Linux-c6x .....</b>	<b>22</b>
7.1 Передача файлов через TFTP .....	22
7.2 Передача файлов через FTP .....	22
<b>Приложение А: Листинг файла конфигурации «setenv» .....</b>	<b>23</b>
<b>Список литературы .....</b>	<b>25</b>

## Список рисунков

3-1	Редактирование файла «/etc/exports»	10
3-2	Вывод команды exportfs	10
3-3	Список файлов NFS корневой файловой системы	11
6-1	Пример TELNET сессии в Windows системе	17
6-2	Программа PuTTY, настройка кодировки	18
6-3	Программа PuTTY, управление соединениями	19
6-4	Программа netcat с выводом лога загрузки модуля SAMC-404	21
7-1	FTP-сессия в Windows системе	22

## Список таблиц

2-1	Назначение устанавливаемых пакетов	6
2-2	Зависимости для сборки Linux-сбх	7
2-3	Назначение сгенерированных файлов Linux-сбх	8
3-1	Соответствие вида имени образа и формата файловой системы	12
6-1	Описание формата параметра ядра «netconsole»	19

## Список листингов

5-1	Вывод загрузки Linux-сбх с TFTP сервера	15
A-1	Листинг файла конфигурации «setenv»	23

## Список процедур

3-1	Создание NFS файловой системы	11
3-2	Создание образа ROMFS файловой системы	13
3-3	Создание образа ROMFS файловой системы	13

## Перечень сокращений и условных обозначений

<b>BOOTP</b>	Bootstrap Protocol	14
<b>CCS</b>	Code Composer Studio	7, 8
<b>CGT</b>	Code Generation Tools	7
<b>DHCP</b>	Dynamic Host Configuration Protocol	12, 14, 17
<b>DSP</b>	Digital Signal Processor	19
<b>EEPROM</b>	Electrically Erasable Programmable Read-Only Memory	12
<b>ELF</b>	Executable and Linkable Format	8
<b>FTP</b>	File Transfer Protocol	22
<b>I<sup>2</sup>C</b>	Inter-Integrated Circuit	14
<b>IBL</b>	Intermediate Boot Loader	12, 14
<b>IP</b>	Internet Protocol	9, 12, 17–20, 22
<b>JFFS2</b>	Journaling Flash File System version 2	12
<b>JTAG</b>	Joint Test Action Group	7
<b>MAC</b>	Media Access Control	20
<b>NFS</b>	Network File System	3, 9–12, 15
<b>ROMFS</b>	Read-Only Memory File System	6, 9, 13
<b>TELNET</b>	TErминаl NETwork	3, 16, 17, 19
<b>TFTP</b>	Trivial File Transfer Protocol	8, 9, 12–15, 22
<b>TI</b>	Texas Instruments	7
<b>UART</b>	Universal Asynchronous Receiver-Transmitter	19
<b>UDP</b>	User Datagram Protocol	19, 20
<b>YACC</b>	Yet Another Compiler Compiler	6
<b>ОС</b>	Операционная Система	5

## 1 Общие сведения

В данном документе рассмотрен процесс сборки системы Linux-сбх из исходных кодов и запуск её на модуле SAMC-404. Все, что описано в данном документе, было выполнено и проверено на 32-х разрядной ОС Ubuntu 10.04.

Использование других 32-х разрядных Linux ОС для сборки системы Linux-сбх возможно, но процесс сборки может отличаться от описанного в данном документе. Поэтому, для сборки, рекомендуется использовать 32-х разрядную ОС Ubuntu 10.04.

Возможность использования 64-х разрядных Linux ОС для сборки системы Linux-сбх на данный момент не проверялась. Сборка Linux-сбх под ОС Windows не возможна.

## 2 Сборка Linux-c6x

### 2.1 Настройка системы

Перед выполнением сборки Linux-c6x необходимо установить требуемые для сборки пакеты. Для установки необходимых пакетов, выполните следующую команду:

```
sudo apt-get install git-core build-essential bison flex genromfs
```

Краткое назначение устанавливаемых пакетов приведено в таблице 2-1.

Таблица 2-1: Назначение устанавливаемых пакетов

Пакет	Назначение
git-core	Система управления версиями Git.
build-essential	Сборный пакет, включающий в себя пакеты необходимые для сборки ядра Linux (компилятор, библиотеки, сборщик и т. п.).
bison	Генератор парсеров совместимый с YACC (Yet Another Compiler Compiler).
flex	Быстрый генератор лексических анализаторов.
genromfs	Программа создания файлов образов с файловой системой ROMFS.

### 2.2 Получение исходных кодов Linux-c6x

Архив с исходными кодами системы Linux-c6x находится на сопроводительном диске к модулю SAMC-404 в папке «linux-c6x/src». Для выполнения сборки системы Linux-c6x необходимо переписать архив с исходными кодами с сопроводительного диска на жесткий диск компьютера в папку «~/my-linux-c6x» и распаковать.

#### Примечание

В исходные коды системы Linux-c6x на сопроводительном диске к модулю SAMC-404 внесены изменения, позволяющие запускать систему на модуле SAMC-404. В связи с этим, использование исходных кодов из официального репозитория проекта Linux-c6x для запуска системы на модуле SAMC-404 не возможно.

Для выполнения копирования исходных кодов с сопроводительного диска на жесткий диск выполните команды:

```
mkdir ~/my-linux-c6x
cp /mnt/cdrom/linux-c6x.tar.bz2 ~/my-linux-c6x
cd ~/my-linux-c6x
tar xvfj linux-c6x.tar.bz2
```

В данном случае, предполагается, что сопроводительный диск к модулю SAMC-404 смонтирован в папку «/mnt/cdrom».

#### Примечание

Размер распакованных исходных кодов составляет около 0.9 Гбайт. Для сборки системы Linux-c6x может потребоваться около 4 Гбайт свободного дискового пространства.

## 2.3 Установка зависимостей и конфигурация

Для сборки системы Linux-c6x необходимо установить зависимости указанные в таблице 2-2.

Таблица 2-2: Зависимости для сборки Linux-c6x

Зависимость	Необходимо для
GCC Tool Chain 4.5-124	Сборка ядра Linux-c6x
CCS (Code Composer Studio) 5.0.3.00028	JTAG отладка, JTAG запись загрузчика
TI CGT (Code Generation Tools) 7.2.2	Сборка Rio-utils

Дистрибутивы указанных зависимостей имеются на сопроводительном диске к модулю SAMC-404 в папке «linux-c6x/deps». Для их установки, необходимо переписать файлы дистрибутивов в папку «~/my-linux-c6x/download». Для этого, выполните следующие команду:

```
cp /mnt/cdrom/linux-c6x/deps/* ~/my-linux-c6x/download
```

В данном случае, предполагается, что сопроводительный диск к модулю SAMC-404 смонтирован в папку «/mnt/cdrom».

Перейдите в папку «linux-c6x-project», выполнив команду:

```
cd linux-c6x-project
```

Выполните скрипт конфигурации:

```
./prj config
```

Скрипт автоматически выполнит установку всех необходимых зависимостей из папки «~/my-linux-c6x/download».

В случае успешной конфигурации и наличия всех зависимостей, в терминал будет выведено сообщение:

```
setup is complete
```

Управление процессом сборки системы Linux-c6x выполняется путем изменения значений параметров в файле «setenv» (папка «~/my-linux-c6x/linux-c6x-project»). Файл «setenv» имеющийся на сопроводительном диске к модулю SAMC-404 уже сконфигурирован для корректной сборки системы Linux-c6x. Поэтому, в общем случае, редактирование данного файла не требуется. В случае необходимости изменения каких либо параметров в файле «setenv», руководствуйтесь информацией приведенной в приложении A.

### Примечание

В приложении A приведен полный листинг файла «setenv» с комментариями, переведенными на русский язык. В комментариях листинга описано назначение каждого параметра для управления сборкой проекта Linux-c6x.

После выполнения любых изменений в файле «setenv», необходимо повторно запустить скрипт конфигурации:

```
./prj config
```

## 2.4 Сборка

Для запуска процесса сборки системы Linux-c6x выполните в терминале команду:

```
./prj build
```

После успешной сборки Linux-c6x, сгенерированные файлы образов будут помещены в папку «~/my-linuxc6x/product»:

```
user@ubuntu:~/my-linux-c6x/linux-c6x-project$ cd ~/my-linux-c6x/product
user@ubuntu:~/my-linux-c6x/product$ ls -o
total 30604
-rwxr-xr-x 1 user 11018      2012-06-06 20:40 bootblob
drwxr-xr-x 4 user 4096      2012-06-06 20:40 bootblob-templates
-rw-r--r-- 1 user 17351766 2012-11-14 18:00 full-root-c6x.cpio.gz
-rw-r--r-- 1 user 146692    2012-11-14 18:00 gplv3-devtools-c6x.cpio.gz
-rwxr-xr-x 1 user 7792      2012-06-06 20:40 make-filesystem
-rw-r--r-- 1 user 2202551 2012-11-14 18:00 mcsdk-demo-root-c6x.cpio.gz
-rw-r--r-- 1 user 2123443 2012-11-14 18:00 min-root-c6x.cpio.gz
-rw-r--r-- 1 user 542856    2012-11-14 18:00 modules-2.6.34-evmc6457.el-dev-user-20121114.tar.gz
-rwxr-xr-x 1 user 4815538 2012-11-14 17:49 vmlinux-2.6.34-evmc6457.el-dev-user-20121114
-rwxr-xr-x 1 user 4116992 2012-11-14 17:49 vmlinux-2.6.34-evmc6457.el-dev-user-20121114.bin
```

Краткая информация о сгенерированных файлах представлена в таблице 2-3.

Таблица 2-3: Назначение сгенерированных файлов Linux-c6x

Файл	Назначение
vmlinux-2.6.34-evmc6457.el-dev-user-<дата_сборки>.bin	Собранное ядро Linux-c6x в формате BBLOB для загрузки на модуль SAMC-404 с TFTP сервера.
vmlinux-2.6.34-evmc6457.el-dev-user-<дата_сборки>	Собранное ядро Linux-c6x в формате ELF для загрузки на модуль SAMC-404 через CCS.
min-root-c6x-hf.cpio.gz	Архив минимальной корневой файловой системы.
mcsdk-demo-root-c6x-hf.cpio.gz	Архив корневой файловой системы, включающий в себя демонстрационное веб-приложение запускаемое на Linux-c6x.
full-root-c6x-hf.cpio.gz	Архив полной корневой файловой системы, включающий в себя демонстрационное веб приложение и дополнительные утилиты.
gplv3-devtools-c6x-hf.cpio.gz	Архив части файловой системы, включающий в себя запускаемый файл отладчика.
modules-2.6.34-evmc6457.el-dev-user-<дата_сборки>.tar.gz	Архив части файловой системы, включающий в себя дополнительные модули ядра.

Файл «bootblob» является shell-скриптом, предназначенным для сборки готовых для загрузки на модуле SAMC-404 образов, которые состоят из ядра системы и образа файловой системы. Также, скрипт «bootblob» применяется для изменения строки параметров ядра системы. Работа со скриптом «bootblob» подробно рассмотрена в разделе 3.

Файл «make-filesystem» является shell-скриптом, предназначенным для сборки образов файловой системы. Работа со скриптом «make-filesystem» подробно рассмотрена в разделе 3.



## 3 Создание образов Linux-сбх

Для создания образов Linux-сбх для загрузки, включающих в себя ядро и файловую систему, служат специальные утилиты «bootblob» и «make-filessystem», которые находятся в папке «product» дерева каталогов исходных кодов Linux-сбх.

Утилита bootblob является shell-скриптом, предназначенным для сборки готовых к записи (загрузке) на модуль SAMC-404 образов, которые состоят из более мелких частей (ядро системы, дополнительные модули ядра, файловая система и т. п.). Ядро системы и составные части файловой системы получают в результате сборки системы (см. раздел 2).

Скрипт bootblob позволяет выполнять следующие действия:

- создание готовых к загрузке образов системы;
- чтение и сохранение строки параметров ядра из существующего образа системы;
- запись строки параметров ядра в существующий образ системы;
- создание образов файловых систем и образов ядра из шаблонов с заданной строкой параметров ядра.

При создании образов системы для загрузки Linux-сбх с TFTP сервера на модуле SAMC-404 возможно использовать два вида корневой файловой системы:

- 1) NFS корневая файловая система. С TFTP сервера загружается только ядро системы, корневая файловая система монтируется по NFS протоколу. В этом случае, корневая файловая система доступна для чтения и записи, но требуется настройка NFS сервера;
- 2) ROMFS корневая файловая система. Создается образ корневой файловой системы в формате ROMFS. Образ файловой системы внедряется в образ системы для загрузки с TFTP сервера. После загрузки ядра системы на модуле SAMC-404, корневая файловая система монтируется из оперативной памяти модуля и доступна только для чтения.

В разделах 3.1 и 3.2 подробно рассмотрена конфигурация и создание обоих видов корневой файловой системы с применением утилит «bootblob» и «make-filessystem».

### 3.1 Образ системы для NFS корневой файловой системы

Для запуска системы Linux-сбх с корневой файловой системой, подключаемой по NFS, необходимо настроить NFS сервер и скопировать на него необходимые данные корневой файловой системы из собранных образов. Ядру Linux-сбх адрес NFS сервера передается в строке параметров ядра.

В разделе 3.1.1 рассмотрена процедура настройки NFS сервера в операционной системе Ubuntu 10.04 на том же компьютере на котором производилась сборка системы Linux-сбх (см. раздел 2). Будем считать, что IP адрес этого компьютера 192.168.2.3 и маска подсети 255.255.255.0.

#### 3.1.1 Настройка NFS сервера

Для установки службы NFS сервера в системе, выполните команду:

```
sudo apt-get install nfs-kernel-server
```

Создайте на сервере папку, в которой будет расположена корневая файловая система Linux-сбх:

```
sudo mkdir -p /srv/nfs/samc404/rootfs
```

Далее, необходимо добавить созданную папку в список экспортируемых NFS сервером ресурсов. Список ресурсов, экспортируемых NFS сервером указывается в файле «/etc/exports».

Для редактирования файла «/etc/exports» можно воспользоваться текстовым редактором nano, выполнив в терминале команду:

```
sudo nano /etc/exports
```

В конец файла «/etc/exports» необходимо добавить строку (см. рисунок 3-1)

```
/srv/nfs/samc404/rootfs *(rw,no_root_squash,no_all_squash, sync, no_subtree_check)
```

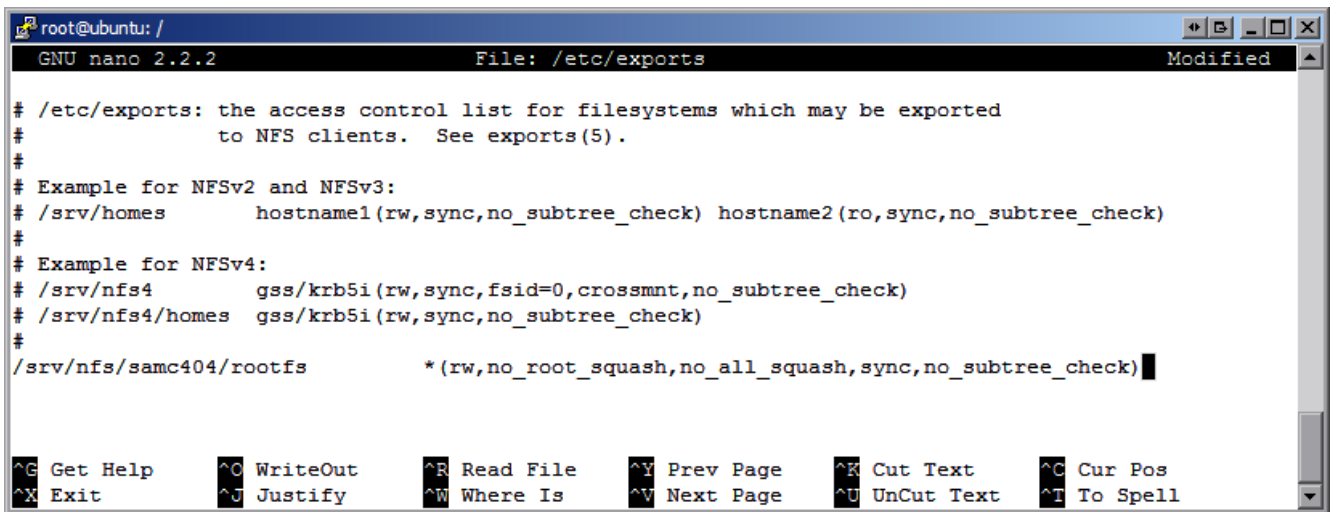


Рисунок 3-1: Редактирование файла «/etc/exports»

После редактирования файла «/etc/exports», для применения сделанных изменений, необходимо перезапустить службу NFS сервера. Для перезапуска NFS сервера выполните команду:

```
sudo service nfs-kernel-server restart
```

Убедиться в правильности сделанных изменений можно выполнив в терминале команду:

```
exportfs
```

В случае, если все было сделано правильно, вывод команды exportfs должен быть как на рисунке 3-2.

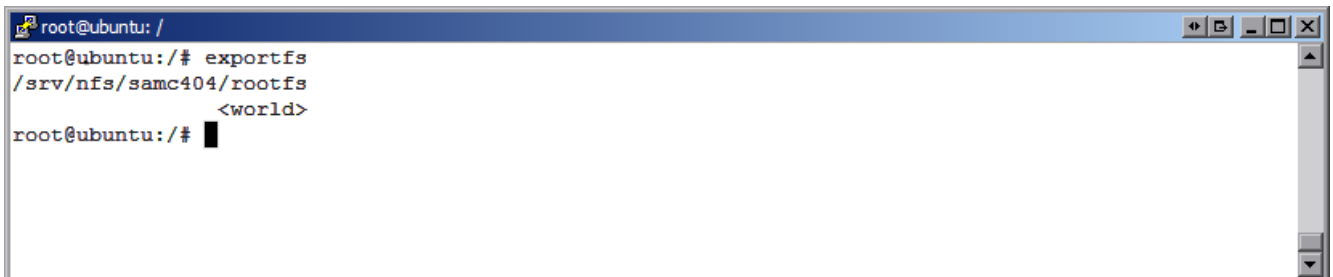


Рисунок 3-2: Вывод команды exportfs

### 3.1.2 Создание файловой системы

Далее, необходимо в папку «/srv/nfs/samc404/rootfs» поместить файлы корневой файловой системы.

Во время сборки системы Linux-сбх (см. раздел 3) создается несколько файлов образов корневой файловой системы и её частей в файлах с расширением «.cpio.gz». Краткая информация о создаваемых файлах во время сборки приведена в таблице 2-3 раздела 2.4.

Для создания NFS корневой файловой системы в папке «/srv/nfs/samc404/rootfs» из файла образа корневой файловой системы «mcsdk-demo-root-сбх-hf.cpio.gz» (файл расположен в папке «~/my-linux-сбх/product» после выполнения сборки системы) нужно выполнить действия изложенные в процедуре 3-1.

## Процедура 3-1. Создание NFS файловой системы

1. Скопируйте файл «mcsdk-demo-root-c6x-hf.cpio.gz» в папку NFS ресурса:

```
cd ~/my-linux-c6x/product
sudo cp mcsdk-demo-root-c6x-hf.cpio.gz /srv/nfs/samc404/rootfs
```

2. Перейдите в папку NFS ресурса:

```
cd /srv/nfs/samc404/rootfs
```

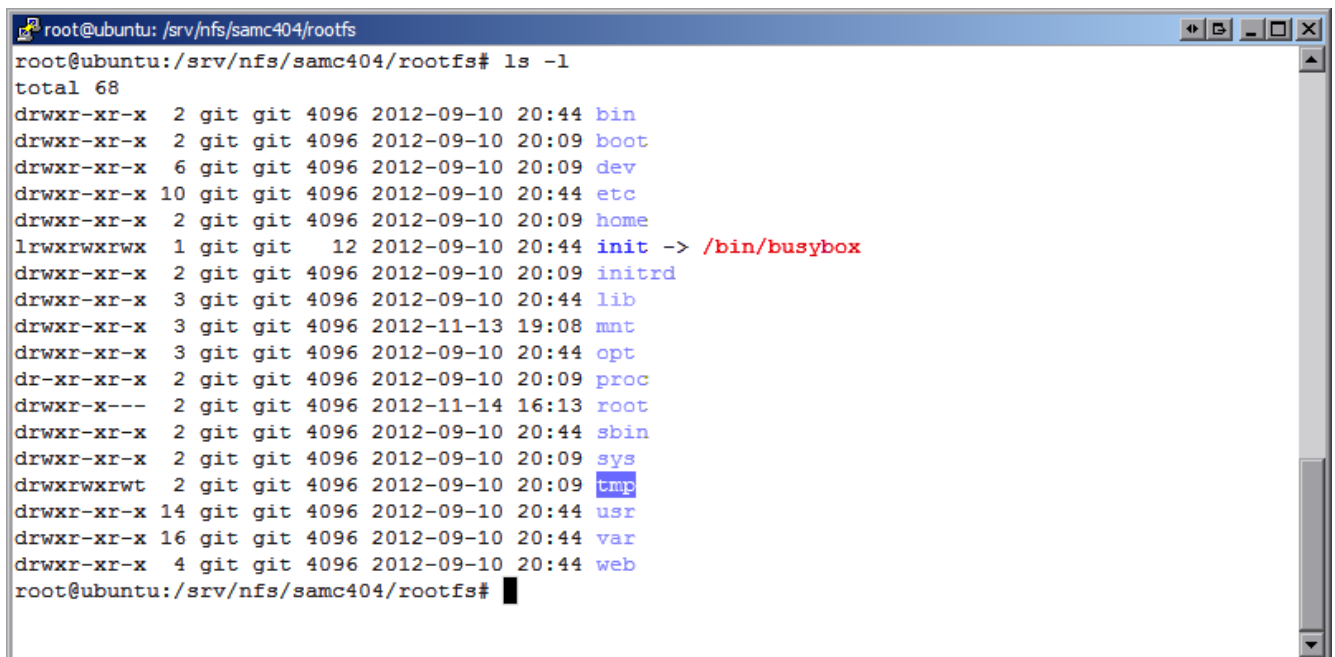
3. Выполните распаковку файла «mcsdk-demo-root-c6x-hf.cpio.gz»:

```
sudo gunzip -d mcsdk-demo-root-c6x-hf.cpio.gz
sudo cpio -idmv < mcsdk-demo-root-c6x-hf.cpio
```

4. Удалите файл образа:

```
sudo rm mcsdk-demo-root-c6x-hf.cpio
```

После выполнения действий, описанных в процедуре 3-1, в папке «/srv/nfs/samc404/rootfs» будут находиться файлы корневой файловой системы Linux-c6x (рисунок 3-3).



```
root@ubuntu: /srv/nfs/samc404/rootfs
root@ubuntu:/srv/nfs/samc404/rootfs# ls -l
total 68
drwxr-xr-x  2 git git 4096 2012-09-10 20:44 bin
drwxr-xr-x  2 git git 4096 2012-09-10 20:09 boot
drwxr-xr-x  6 git git 4096 2012-09-10 20:09 dev
drwxr-xr-x 10 git git 4096 2012-09-10 20:44 etc
drwxr-xr-x  2 git git 4096 2012-09-10 20:09 home
lrwxrwxrwx  1 git git  12 2012-09-10 20:44 init -> /bin/busybox
drwxr-xr-x  2 git git 4096 2012-09-10 20:09 initrd
drwxr-xr-x  3 git git 4096 2012-09-10 20:44 lib
drwxr-xr-x  3 git git 4096 2012-11-13 19:08 mnt
drwxr-xr-x  3 git git 4096 2012-09-10 20:44 opt
dr-xr-xr-x  2 git git 4096 2012-09-10 20:09 proc
drwxr-xr-x  2 git git 4096 2012-11-14 16:13 root
drwxr-xr-x  2 git git 4096 2012-09-10 20:44 sbin
drwxr-xr-x  2 git git 4096 2012-09-10 20:09 sys
drwxrwxrwt  2 git git 4096 2012-09-10 20:09 tmp
drwxr-xr-x 14 git git 4096 2012-09-10 20:44 usr
drwxr-xr-x 16 git git 4096 2012-09-10 20:44 var
drwxr-xr-x  4 git git 4096 2012-09-10 20:44 web
root@ubuntu:/srv/nfs/samc404/rootfs#
```

Рисунок 3-3: Список файлов NFS корневой файловой системы

Процедура 3-1 может применяться для создания NFS файловой системы из любого файла образа «.cpio.gz» путем подстановки в нее имени файла образа вместо «mcsdk-demo-root-c6x-hf».

Для создания произвольного образа файловой системы применяется утилита «make-filesystem». Если вызвать «make-filesystem» без аргументов командной строки, будет выведена подсказка:

```
user@ubuntu:~/my-linux-c6x/product$ ./make-filesystem
expected 2 arguments and found 0
arg1    output name
arg2    list of filesystem components
```

В подсказке указано, что утилита «make-filesystem» принимает в качестве параметров командной строки два аргумента — «arg1» и «arg2».

В аргументе «arg1» задается имя выходного образа файловой системы. Формат выходного образа файловой системы будет зависеть от вида заданного в аргументе «arg1» имени образа. Соответствие между видом задаваемого имени образа и форматом результирующей файловой системы указано в таблице 3-1.

В аргументе «arg2» передается список компонентов результирующей файловой системы. Все компоненты могут быть в тех же форматах, что описаны в таблице 3-1.

Таблица 3-1: Соответствие вида имени образа и формата файловой системы

Имя	Формат файловой системы
*.cpio.gz	Файловая система в файле образа «*.cpio.gz». Данный формат используется в образах initramfs.
*.tar.gz	Файловая система в архивном файле «*.tar.gz». Стандартный формат архива.
*.tar.bz2	Файловая система в архивном файле «*.tar.bz2». Стандартный формат архива.
*.jffs2	Файловая система в файле образа «*.jffs2». Файловая система JFFS2 (Journaling Flash File System version 2) для флеш памяти.
«другое»	Файловая система в распакованном виде в указанном каталоге.

Например, для создания результирующего образа «filesystem.cpio.gz», который должен состоять из файла образа «min-root-c6x-hf.cpio.gz», архивного файла «modules-2.6.34-evmc6678.el-dev-user-20121115.tar.gz» и содержимого папки «~/samc404-fs» нужно выполнить команду:

```
./make-filsystem filesystem.cpio.gz min-root-c6x-hf.cpio.gzmodules-2.6.34-evmc6678.el-dev-user-
← 20121115.tar.gz ~/samc404-fs
```

### 3.1.3 Конфигурация параметров ядра Linux-с6x

Для конфигурации параметров ядра системы Linux-с6x применяется утилита «bootblob». Получить справку по использованию утилиты «bootblob» можно запустив её с параметром «help»:

```
./bootblob help
```

Для того, что бы при загрузке систем Linux-с6x на модуле SAMC-404 корневая файловая система было смонтирована по NFS протоколу, ядру необходимо передать следующие параметры (при условии, что IP-адрес NFS сервера 192.168.2.3 и на нем экспортируется папке «/srv/nfs/samc404/rootfs»):

```
console=cio ip=dhcp root=/dev/nfs nfsroot=192.168.2.3:/srv/nfs/samc404/rootfs,v3,tcp rw
```

В данном случае, указывается на способ получения IP-адреса через DHCP сервер (параметр «ip=dhcp»). Загрузка системы Linux-с6x на модуле SAMC-404 возможна только с использованием связки DHCP и TFTP серверов. Поэтому использование фиксированного IP-адреса для модуля SAMC-404 маловероятно и нецелесообразно, но в случае такой необходимости, его необходимо указать в параметре «ip=». Например, для IP-адреса 192.168.2.98 строка параметров ядра будет выглядеть следующим образом:

```
console=cio ip=192.168.2.98 root=/dev/nfs nfsroot=192.168.2.3:/srv/nfs/samc404/rootfs,v3,tcp rw
```

Параметры ядра задаются в файле «vmlinux-2.6.34-evmc6678.el-dev-user-<дата\_сборки>.bin» (см. раздел 2), который в последствии загружается на TFTP сервер, с которого уже посредством загрузчика IBL (Intermediate Boot Loader), который записан в EEPROM память модуля SAMC-404, происходит загрузка ядра Linux-с6x в память модуля.

#### Примечание

Настройка DHCP и TFTP серверов для загрузки образов на модуль SAMC-404 рассматривается в документе [1]. Информация по загрузчику IBL и записи его в EEPROM память имеется в документе [2].

## 3.2 Образ системы для ROMFS файловой системы

Для запуска системы Linux-сбх с корневой файловой системой ROMFS необходимо создать образ файловой системы в формате ROMFS, после чего, при помощи утилиты «bootblob» создать конечный образ для загрузки на модуле SAMC-404, включающий в себя ядро системы и файловую систему.

### 3.2.1 Создание образа файловой системы ROMFS

Для создания образа файловой системы в формате ROMFS необходимо выполнить шаги, описанные в процедуре 3-3.

Процедура 3-2. Создание образа ROMFS файловой системы

1. Создайте произвольную файловую систему, распакованную в папку (например, в папку «~/mylinux-сбх/product/romfs») при помощи утилиты «make-filessystem». Использование утилиты «makefilessystem» описано в разделе 3.1.2.

Например, для создания файловой системы из образа файловой системы «mcsdk-demo-root-сбхhf.cpio.gz», распакованной в папку «~/my-linux-сбх/product/rootfs», нужно выполнить команду:

```
./make-filessystem rootfs mcsdk-demo-root-с6х-hf.cpio.gz
```

После, можно добавить все необходимые файлы на файловой системе в эту папку.

2. Создайте файл образа файловой системы в формате ROMFS из содержимого папки, полученной на предыдущем шаге. Для создания файла образа «rootfs.romfs» из содержимого папки «rootfs» выполните команду:

```
genromfs -d rootfs -f rootfs.romfs
```

Здесь, в параметре «-d» задается имя папки, в которой расположено содержимое файловой системы, в параметре «-f» задается имя файла образа в формате ROMFS.

### 3.2.2 Создание образа для загрузки

После получения файла образа файловой системы в формате ROMFS необходимо выполнить сборку конечного образа системы Linux-сбх для загрузки на модуле SAMC-404 с TFTP сервера. Данный образ должен включать в себя ядро системы Linux-сбх и образ файловой системы в формате ROMFS. Действия, необходимые для создания такого образа, описаны в процедуре 3-3.

Процедура 3-3. Создание образа ROMFS файловой системы

1. Создайте образ для загрузки на TFTP сервер, включающий в себя ядро системы Linux-сбх и файл образа файловой системы в формате ROMFS. Для создания образа «samc404.bin», включающего в себя ядро Linux-сбх «vmlinux-2.6.34-еvmc6678.el-dev-user-<дата\_сборки>.bin» и файл образа файловой системы в формате ROMFS «rootfs.romfs», выполните команду:

```
./bootblob make-image --round=0x1000 --abs-base=0xE0000000 samc404.bin vmlinux-2.6.34-  
← evmc6678.el-dev-user-<дата_сборки>.bin rootfs.romfs
```

2. Установите требуемые параметры ядра в получившемся файле образа «samc404.bin». Для этого выполните команду:

```
./bootblob set-cmdline samc404.bin "console=cio ip=dhcp root=/dev/mtdblock0 rootfstype=  
← romfs"
```

3. Перепешите файл образа «samc404.bin» в корневую папку TFTP сервера и выполните загрузку модуля SAMC-404. Информация по настройке TFTP сервера имеется в документе [1].

## 4 Запись образов Linux-c6x на модуль

Для загрузки образов системы Linux-c6x на модуле SAMC-404 используется загрузчик IBL, который должен быть записан в EEPROM память на I<sup>2</sup>C шине по адресу 0x50. Загрузчик IBL загружает образ ядра системы Linux-c6x с TFTP сервера, адрес которого и имя файла для загрузки выдается BOOTP/DHCP сервером в сети.

Подробная информация по записи и конфигурации загрузчика IBL на модуле SAMC-404 приведена в документе [2]. Информация по настройке BOOTP/DHCP и TFTP серверов для загрузки системы Linux-c6x на модуле SAMC-404 приведена в документе [1].

## 5 Загрузка Linux-c6x

При включении модуля SAMC-404 будет выполнена загрузка системы Linux-c6x с TFTP сервера.

Ниже приведен лог загрузки Linux-c6x с TFTP сервера. Корневая файловая система монтируется по NFS.

Листинг 5-1: Вывод загрузки Linux-c6x с TFTP сервера

```

1 samc404 login: root
2 /root # dmesg
3 Linux version 2.6.34-evmc6457.e1-dev-user-20120910 (user@vm-ubuntu-10-04) (gcc version 4.5.1
4 (Sourcery CodeBench Lite 4.5-124) ) #2 Mon Sep 10 20:57:51 MSK 2012
5 Designed for the EVM6457 board, Texas Instruments.
6 CPU0: C64x+ rev 0x0, 1.2 volts, 1200MHz
7 Initializing kernel
8 no initrd specified
9 On node 0 totalpages: 65536
10 free_area_init_node: node 0, pgdat e03b39bc, node_mem_map e0410000
11   DMA zone: 512 pages used for memmap
12   DMA zone: 0 pages reserved
13   DMA zone: 65024 pages, LIFO batch:0
14 Built 1 zonelists in Zone order, mobility grouping on. Total pages: 65024
15 Kernel command line: console=cio ip=dhcp root=/dev/nfs nfsroot=192.168.2.3:/srv/nfs/samc404/
16 rootfs,v3,tcp rw
17 PID hash table entries: 1024 (order: 0, 4096 bytes)
18 Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)
19 Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)
20 Memory available: 255744k/258004k RAM, 0k/0k ROM (840k kernel code, 192k data)
21 SLUB: Genslabs=7, HWalign=128, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
22 Hierarchical RCU implementation.
23 RCU-based detection of stalled CPUs is enabled.
24 NR_IRQS:128
25 console [cio0] enabled
26 Console: colour dummy device 80x25
27 Calibrating delay loop... 1196.03 BogoMIPS (lpj=2392064)
28 Mount-cache hash table entries: 512
29 C64x: 16 gpio irqs
30 NET: Registered protocol family 16
31 bio: create slab <bio-0> at 0
32 Switching to clocksource TSC64
33 NET: Registered protocol family 2
34 IP route cache hash table entries: 2048 (order: 1, 8192 bytes)
35 TCP established hash table entries: 8192 (order: 4, 65536 bytes)
36 TCP bind hash table entries: 8192 (order: 3, 32768 bytes)
37 TCP: Hash tables configured (established 8192 bind 8192)
38 TCP reno registered
39 UDP hash table entries: 256 (order: 0, 4096 bytes)
40 UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
41 NET: Registered protocol family 1
42 RPC: Registered udp transport module.
43 RPC: Registered tcp transport module.
44 RPC: Registered tcp NFSv4.1 backchannel transport module.
45 eth0: EMAC(0) driver version 2.1 IRQ=25 queue=0
46 eth0: MAC address=7c:8e:e4:bb:32:27 PHY=SGMII
47 JFFS2 version 2.2. (NAND) (SUMMARY) © 2001-2006 Red Hat, Inc.
48 ROMFS MTD (C) 2007 Red Hat, Inc.
49 msgmni has been set to 499
50 Block layer SCSI generic (bsg) driver version 0.4 loaded (major 254)
51 io scheduler noop registered
52 io scheduler deadline registered
53 io scheduler cfq registered (default)
54 brd: module loaded
55 loop: module loaded
56 at24 1-0050: 131072 byte 24c1024 EEPROM (writable)
57 uclinux[mtd]: RAM probe address=0xe040a560 size=0x98001000
58 Creating 1 MTD partitions on "RAM":
59 0x000000000000-0x000098001000 : "ROMfs"
60 Generic platform RAM MTD, (c) 2004 Simtec Electronics
61 console [netcon0] enabled
62 netconsole: network logging started

```

```
63 TCP cubic registered
64 NET: Registered protocol family 17
65 Sending DHCP requests ., OK
66 IP-Config: Got DHCP answer from 192.168.2.3, my address is 192.168.2.98
67 IP-Config: Complete:
68     device=eth0, addr=192.168.2.98, mask=255.255.255.0, gw=192.168.2.2,
69     host=samc404, domain=, nis-domain=(none),
70     bootserver=192.168.2.3, rootserver=192.168.2.3, rootpath=
71 Looking up port of RPC 100003/3 on 192.168.2.3
72 Looking up port of RPC 100005/3 on 192.168.2.3
73 VFS: Mounted root (nfs filesystem) on device 0:11.
74 Freeing unused kernel memory: 156K freed
75 /root #
```

Данный лог получен путем выполнения команды `dmesg` после загрузки системы при подключении к системе по протоколу [TELNET](#).

Информация по настройке клиента для подключения к модулю SAMC-404 по протоколу [TELNET](#) приведена в разделе [6.1](#).



## 6 Подключение к Linux-c6x

### 6.1 Подключение по Telnet

Подключение по TELNET протоколу может быть выполнено с использованием стандартного клиента как в Windows системе, так и в Linux системе.

Подключение по TELNET протоколу выполняется по IP адресу системы. IP адрес на системе Linux-c6x может быть сконфигурирован посредством DHCP сервера в сети или вручную.

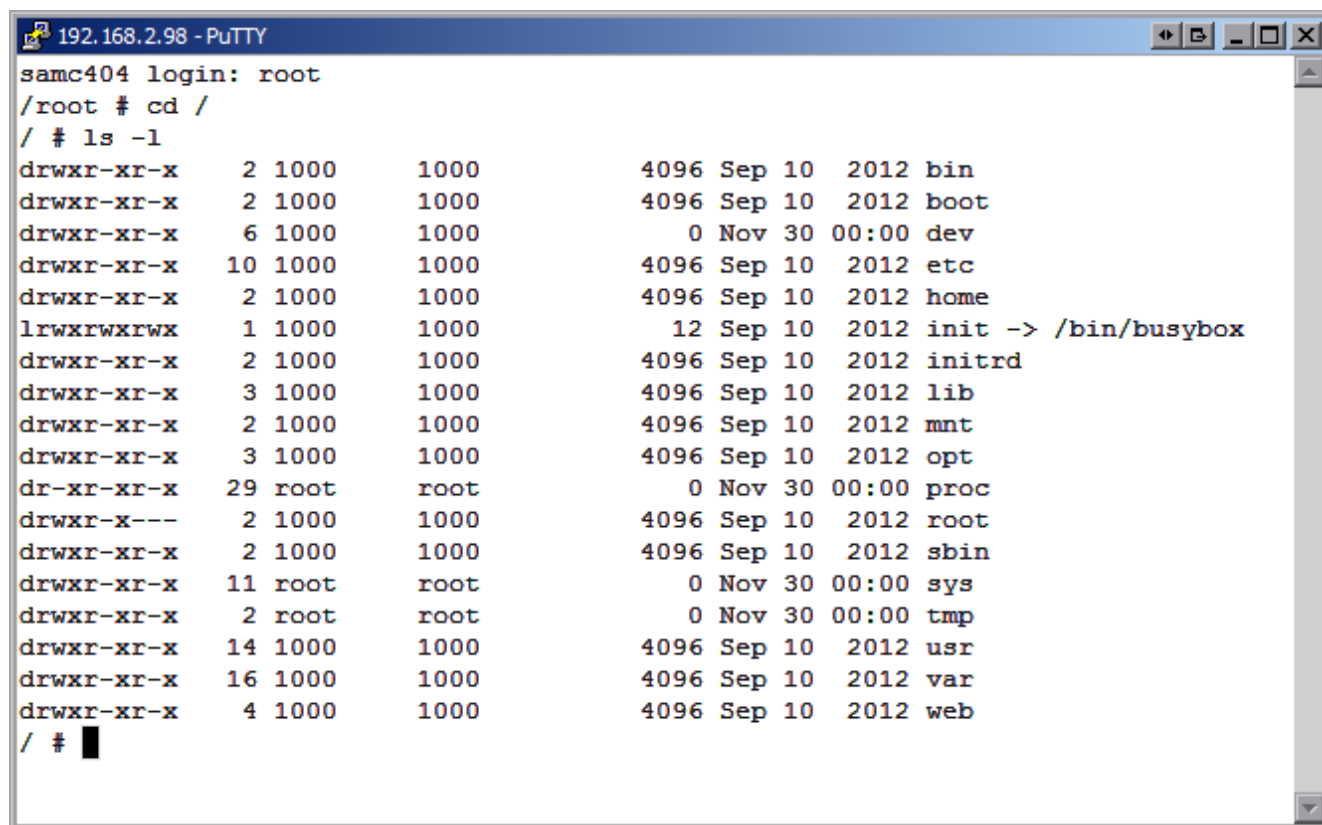
Для подключения к системе Linux-c6x по TELNET протоколу выполните команду:

```
telnet <IP_адрес_системы>
```

Например, если IP адрес системы 192.168.2.98, то необходимо выполнить команду:

```
telnet 192.168.2.98
```

Пример TELNET сессии в Windows системе приведен на рисунке 6-1.



```
192.168.2.98 - PuTTY
samc404 login: root
/root # cd /
/ # ls -l
drwxr-xr-x  2 1000    1000    4096 Sep 10  2012 bin
drwxr-xr-x  2 1000    1000    4096 Sep 10  2012 boot
drwxr-xr-x  6 1000    1000         0 Nov 30  00:00 dev
drwxr-xr-x 10 1000    1000    4096 Sep 10  2012 etc
drwxr-xr-x  2 1000    1000    4096 Sep 10  2012 home
lrwxrwxrwx  1 1000    1000         12 Sep 10  2012 init -> /bin/busybox
drwxr-xr-x  2 1000    1000    4096 Sep 10  2012 initrd
drwxr-xr-x  3 1000    1000    4096 Sep 10  2012 lib
drwxr-xr-x  2 1000    1000    4096 Sep 10  2012 mnt
drwxr-xr-x  3 1000    1000    4096 Sep 10  2012 opt
dr-xr-xr-x 29 root     root         0 Nov 30  00:00 proc
drwxr-x---  2 1000    1000    4096 Sep 10  2012 root
drwxr-xr-x  2 1000    1000    4096 Sep 10  2012 sbin
drwxr-xr-x 11 root     root         0 Nov 30  00:00 sys
drwxr-xr-x  2 root     root         0 Nov 30  00:00 tmp
drwxr-xr-x 14 1000    1000    4096 Sep 10  2012 usr
drwxr-xr-x 16 1000    1000    4096 Sep 10  2012 var
drwxr-xr-x  4 1000    1000    4096 Sep 10  2012 web
/ # █
```

Рисунок 6-1: Пример TELNET сессии в Windows системе

В Windows системе, вместо стандартного TELNET клиента, рекомендуется использовать программу PuTTY для подключения к системе по протоколу TELNET. При использовании программы PuTTY, необходимо указать правильную кодировку для соединения. Для этого, в дереве навигации слева выберите раздел «Window > Translation», и укажите кодировку UTF-8 (см. рисунок 6-2).

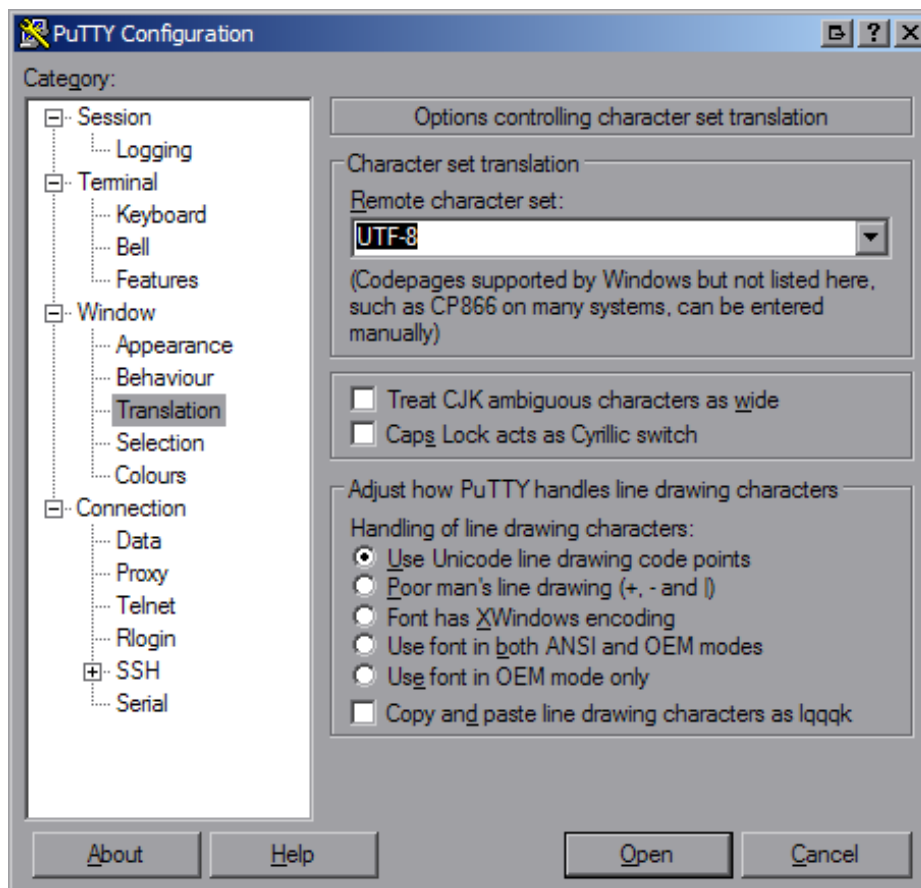


Рисунок 6-2: Программа PuTTY, настройка кодировки

Далее, в дереве навигации слева выберите раздел «Session». Установите параметру «Connection type» значение «Telnet» и впишите в поле «Host Name (or IP address)» IP адрес системы с Linux-c6x, как показано на рисунке 7-1.

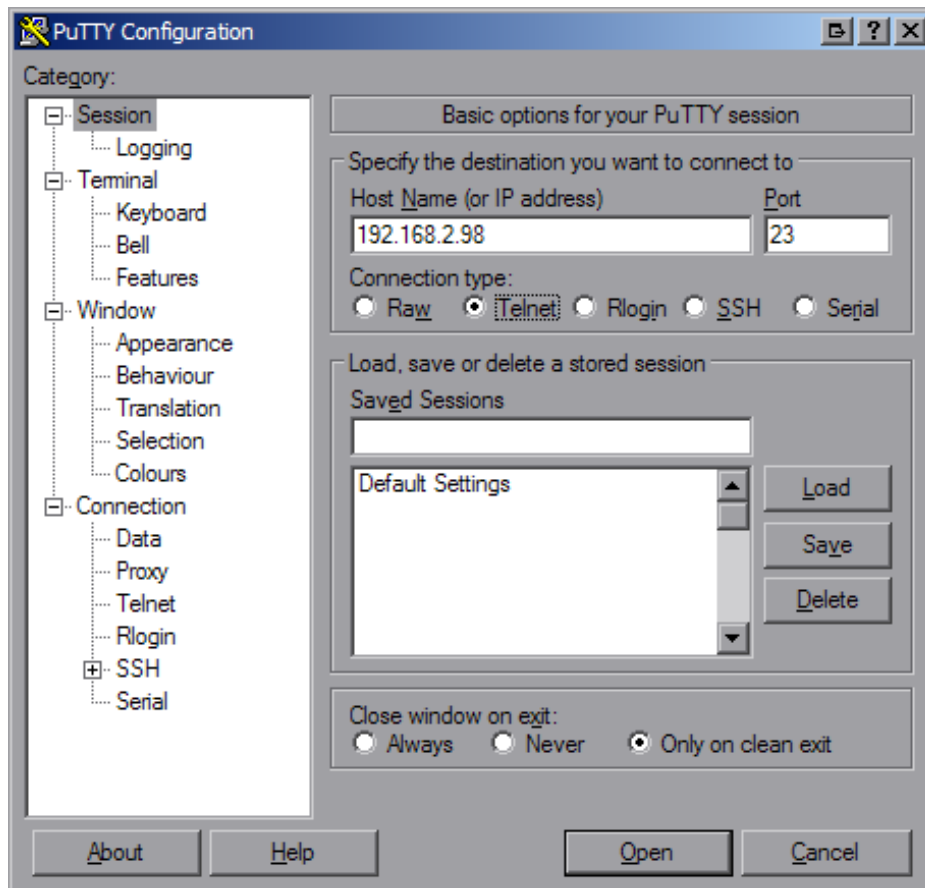


Рисунок 6-3: Программа PuTTY, управление соединениями

Нажмите на кнопку «Open». Произойдет подключение по протоколу TELNET к модулю SAMC-404 и будет выведено приглашение в следующем виде:

```
samc404 login:
```

## 6.2 Использование Netconsole

Использование Netconsole позволяет получить вывод сообщений ядра системы Linux-с6x по сети. Так как модуль SAMC-404 не имеет UART выводов со своих DSP, Netconsole может использоваться для получения вывода сообщений ядра во время загрузки модуля SAMC-404.

Конфигурация Netconsole осуществляется через параметры ядра системы Linux-с6x. Для включения Netconsole при загрузке, ядру системы необходимо передать параметр «netconsole», который имеет следующий формат:

```
netconsole=[src-port]@[src-ip]/[dev],[tgt-port]<[tgt-ip]/[tgt-macaddr]
```

Описание параметров параметра ядра «netconsole» приведено в таблице 6-1.

Таблица 6-1: Описание формата параметра ядра «netconsole»

Параметр	Описание
[src-port]	Номер исходящего UDP-порта. Не обязательный параметр, по умолчанию принимается равным 6665.
[src-ip]	IP-адрес исходящего сетевого интерфейса. Не обязательный параметр. В случае, если не указан, используется текущий IP-адрес сетевого интерфейса, который указан в параметре [dev].
[dev]	Имя сетевого интерфейса для исходящих сообщений. Не обязательный параметр, по умолчанию используется интерфейс eth0.

*Продолжение таблицы на следующей странице*

Продолжение таблицы 6-1

Параметр	Описание
[tgt-port]	Номер UDP-порта клиента куда будут отправлены сообщения. Не обязательный параметр, по умолчанию принимается равным 6666.
<tgt-ip>	IP-адрес клиента куда будут отправлены сообщения. Обязательный параметр.
[tgt-macaddr]	MAC-адрес клиента куда будут отправлены сообщения. Не обязательный параметр. В случае, если не задан, будут посылаться широковещательные сообщения.

Например, для отправки сообщений ядра Linux-сбх с модуля SAMC-404 по сети на компьютер с IP-адресом 192.168.2.3 достаточно добавить параметр «netconsole» в следующем виде:

```
netconsole=@/,@192.168.2.3/
```

Перенаправление вывода консоли Linux-сбх в Netconsole осуществляется добавлением еще одного параметра ядра:

```
console=netcon0
```

Таким образом, для включения Netconsole в образ системы, необходимо при помощи утилиты «bootblob» установить параметры ядра «console» и «netconsole». Конфигурация параметров ядра описана в разделе 3.1.3.

Например, строка параметров, приведенная в разделе 3.1.3:

```
console=cio ip=dhcp root=/dev/nfs nfsroot=192.168.2.3:/srv/nfs/samc404/rootfs,v3,tcp rw
```

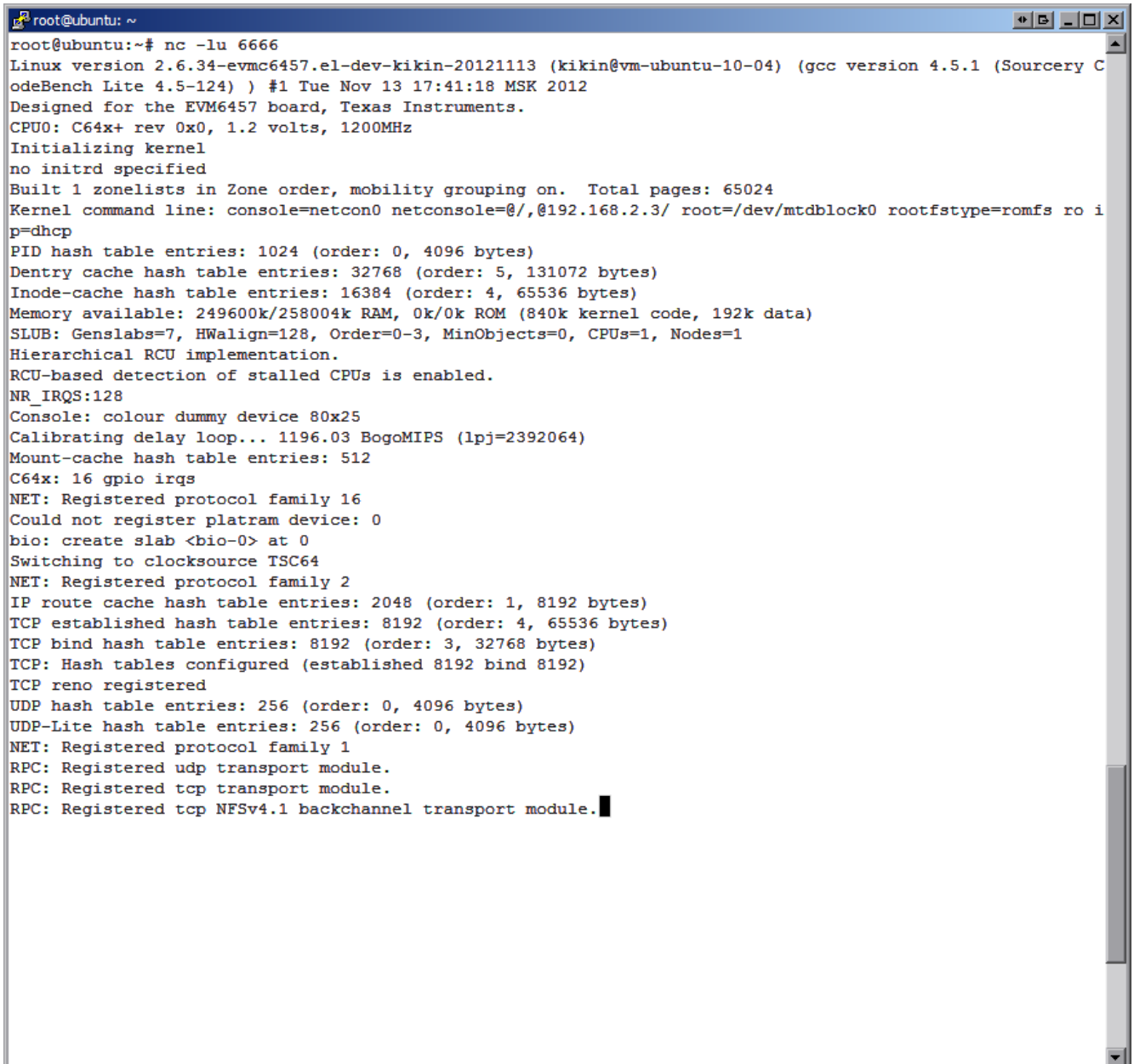
При включения Netconsole для отправки сообщений ядра на компьютер с IP-адресом 192.168.2.3 данная строка параметров ядра будет иметь следующий вид:

```
console=netcon0 netconsole=@/,@192.168.2.3/ ip=dhcp root=/dev/nfs nfsroot=192.168.2.3:/srv/nfs/  
← samc404/rootfs,v3,tcp rw
```

Для приема сообщений на удаленной машине (192.168.2.3) может использоваться программа netcat. Для запуска программы netcat, принимающей сообщения на UDP-порту 6666, выполните команду:

```
nc -lu 6666
```

На рисунке 6-4 приведен пример запуска программы netcat с выводом процесса загрузки модуля SAMC-404 с системой Linux-сбх.



```
root@ubuntu: ~  
root@ubuntu:~# nc -lu 6666  
Linux version 2.6.34-evmc6457.el-dev-kikin-20121113 (kikin@vm-ubuntu-10-04) (gcc version 4.5.1 (Sourcery CodeBench Lite 4.5-124) ) #1 Tue Nov 13 17:41:18 MSK 2012  
Designed for the EVM6457 board, Texas Instruments.  
CPU0: C64x+ rev 0x0, 1.2 volts, 1200MHz  
Initializing kernel  
no initrd specified  
Built 1 zonelists in Zone order, mobility grouping on. Total pages: 65024  
Kernel command line: console=netcon0 netconsole=@/,@192.168.2.3/ root=/dev/mtdblock0 rootfstype=romfs ro ip=dhcp  
PID hash table entries: 1024 (order: 0, 4096 bytes)  
Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)  
Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)  
Memory available: 249600k/258004k RAM, 0k/0k ROM (840k kernel code, 192k data)  
SLUB: Genslabs=7, HWalign=128, Order=0-3, MinObjects=0, CPUs=1, Nodes=1  
Hierarchical RCU implementation.  
RCU-based detection of stalled CPUs is enabled.  
NR_IRQS:128  
Console: colour dummy device 80x25  
Calibrating delay loop... 1196.03 BogoMIPS (lpj=2392064)  
Mount-cache hash table entries: 512  
C64x: 16 gpio irqs  
NET: Registered protocol family 16  
Could not register platram device: 0  
bio: create slab <bio-0> at 0  
Switching to clocksource TSC64  
NET: Registered protocol family 2  
IP route cache hash table entries: 2048 (order: 1, 8192 bytes)  
TCP established hash table entries: 8192 (order: 4, 65536 bytes)  
TCP bind hash table entries: 8192 (order: 3, 32768 bytes)  
TCP: Hash tables configured (established 8192 bind 8192)  
TCP reno registered  
UDP hash table entries: 256 (order: 0, 4096 bytes)  
UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)  
NET: Registered protocol family 1  
RPC: Registered udp transport module.  
RPC: Registered tcp transport module.  
RPC: Registered tcp NFSv4.1 backchannel transport module.█
```

Рисунок 6-4: Программа netcat с выводом лога загрузки модуля SAMC-404



## Приложение А Листинг файла конфигурации «setenv»

Листинг А-1: Листинг файла конфигурации «setenv»

```

1 # Данный файл должен использоваться с командой source (не запускаться)
2 # в вашей командной оболочке для установки контекста проекта linux-сбх.
3 # Например:
4 # /home/user/my-prj/linux-сбх-project$ source setenv
5
6 AUTO_INSTALL=yes
7
8 # Шаблонная инфраструктура. Не изменять.
9 SETENV_SOURCED=1
10 SETENV_FILE=${BASH_SOURCE[0]}
11 if [ -z "$SETENV_LOCAL_SOURCED" ]; then
12     if [ -z "$SETENV_FILE" ]; then echo "Your shell is not BASH, you must source .setenv.local
13     < instead"; return 1; fi
14     SETENV_LOCAL_FILE=$(dirname $SETENV_FILE)/.setenv.local
15     if [ ! -r $SETENV_LOCAL_FILE ]; then echo "You have not run ./prj config yet" ; return 1; fi
16     source $SETENV_LOCAL_FILE
17 fi
18
19 # Выбор ядер для сборки.
20 # Это разделенный пробелами список ядер для сборки. Используемые имена
21 # ядер для сборки будут использованы для включения файлов kbuilds/<имя_ядра>.mk как
22 # фрагментов файла сборки (makefile) из файла сборки (makefile) верхнего уровня.
23 #
24 # Могут использоваться любое или все сразу из перечисленного
25 # dsk6455 evmc6472 evmc6474 evmc6457 evmc6474-lite
26 # evmc6678 evmc6670
27 #
28 # Примечание: для запуска на модуле SAMC-403 подходит только ядро evmc6678
29 #
30 # Например:
31 # export KERNELS_TO_BUILD="dsk6455 evmc6472"
32 #
33 export KERNELS_TO_BUILD="evmc6678"
34
35 # Выбор дополнительных ядер для сборки.
36 # По умолчанию - пусто.
37 # Используется для дополнительных возможностей ядра.
38 export EXTRA_KERNELS_TO_BUILD=""
39
40 # Версия используемого компилятора (GCC) для сборки.
41 # Значение это переменной используется при выполнении "./prj config" для
42 # поиска/настройки ресурсов компилятора.
43 export GCC_VERSION=4.5-124
44
45 # Версия компилятора Texas Instruments используемая для сборки
46 # Linux програм.
47 #
48 # Может принимать значения:
49 # none Пропуск настройки. Для сборки будет использована
50 # первая найденная версия компилятора.
51 # 7.2.2 Любая конкретная версия компилятора.
52 export CGT_LINUX_VERSION=none
53
54 # Установите в "yes" для установки дополнительных
55 # модулей ядра и скриптов для тестирования
56 #export BUILD_TESTS=yes
57
58 # Выбор корневых файловых систем.
59 # min-root - минимальная файловая система
60 # full-root - min-root + доп. пакеты, такие как nbench, polar ssl, и т.н.
61 # ltp-root - min-root + запускаемые файлы ltp теста
62 # mcsdk-demo-root - min-root + демонстрационный mcsdk веб сервер
63 export ROOTFS="min-root mcsdk-demo-root full-root"
64
65 # Список образов для сборки. Может быть пустым (не собирать образы), одно

```

```

65 # или несколько названий шаблонов для сборки разделенных пробелом.
66 #
67 # Список доступных шаблонов смотрите в папке bootblob-templates/*
68 #
69 # Также может принимать значение "all" для сборки всех комбинаций
70 # ядер в папке "./product" и файловых систем.
71 export BOOTBLOBS="all"
72
73 # Установка порядка байт в системе.
74 # Может быть 'little', 'big', или 'both' для обеих версий для сборки
75 export ENDIAN=little
76
77 # Вид реализации операций с плавающей точкой.
78 # Может принимать значения: 'soft', 'hard', 'both', или "native".
79 export FLOAT=native
80
81 # Список дополнительных пакетов для сборки и установки, при использовании версии
82 # файловой системы "full-root".
83 export PKG_LIST="zlib net-snmp polarssl ttcp dhystone nbench-byte tcpdump iperf openssl ethtool"
84
85 # Собирать ли загрузчик IBL и программы для его записи в EEPROM?
86 # "yes" - собирать
87 # "no" - не собирать
88 export BUILD_BOOTLOADERS=yes
89
90 # Собирать ли SysLink и SYS/BIOS примеры?
91 export BUILD_SYSLINK=yes
92
93 # Для сборки SysLink и SYS/BIOS примеров, необходимы дополнительные
94 # зависимости.
95 #
96 # Значение для каждой из зависимостей может принимать значения:
97 # none Пропуск настройки. Для сборки будет использована
98 # первая найденная версия компилятора.
99 # 7.2.2 Любая конкретная версия компилятора.
100 export CCS_VERSION=none
101 export CGT_BIOS_VERSION=7.2.2
102 export IPC_VERSION=1.23.01.26
103 export XDC_VERSION=3.22.01.21
104 export BIOS_VERSION=6.32.01.38
105 export XDAIS_VERSION=none
106
107
108 # ***** Точные пути к ресурсам *****
109 # Если вы не хотите производить настройку с помощью "./prj config" для
110 # поиска/установки утилит для сборки, вы можете указать конкретные пути
111 # для этих утилит в этих параметрах.
112 # export GCC_DIR=~/.opt/c6x-4.5
113 # export CGT_BIOS_DIR=~/.opt/TI/TI_CGT_C6000_X.Y.Z
114 # export CGT_LINUX_DIR=~/.opt/TI/TI_CGT_C6000_X.A.B
115 # export CCS_DIR=~/.opt/ti/ccsv5
116 # export BIOS_DIR=~/.opt/ti/bios_w_xx_yy_zz
117 # export XDC_DIR=~/.opt/ti/xdctools_a_bb_cc_dd
118 # export IPC_DIR=~/.opt/ti/ipc_i_jj_kk_ll
119
120 # Путь для проверки файлов необходимых для скачивания.
121 # По умолчанию - $LINUX_C6X_TOP_DIR/downloads
122 # export DOWNLOAD_PATH=~/.downloads/linux-c6x
123
124 # ***** Определение параметров по умолчанию *****
125 source $LINUX_C6X_TOP_DIR/linux-c6x-project/scripts/setenv.defaults
126 # ***** Переопределение переменных производить здесь *****

```



## Список литературы

1. Установка и настройка сервера сетевой загрузки (BOOTP и TFTP). Руководство пользователя. [UG-CMN-BOOTP-TFTP](#) (цит. на с. 12—14).
2. SAMC-404. Загрузчик IBL. Руководство пользователя. [UG-SAMC-404-IBL](#) (цит. на с. 12, 14).