

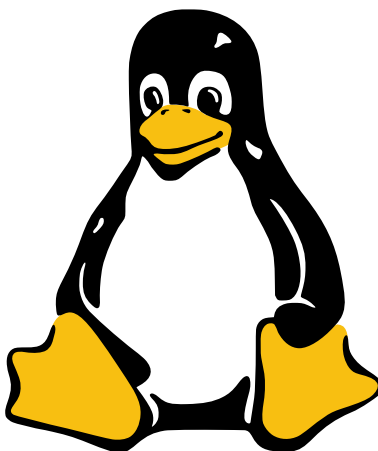


Scan Engineering Telecom SPb

SAMC-403. Сборка и запуск системы Linux-c6x

Руководство пользователя

Версия 1.1



Код документа: UG-SAMC-403-LC6X
Дата сборки: 27 мая 2015 г.
Листов в документе: 32

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

История ревизий

Ревизия	Дата	Изменения
1.1	—	Подправлена вёрстка разделов. 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
1 Общие сведения	6
2 Сборка Linux-c6x	7
2.1 Настройка системы	7
2.2 Получение исходных кодов Linux-c6x	7
2.3 Установка зависимостей и конфигурация	7
2.4 Сборка	11
3 Создание образов Linux-c6x	13
4 Запись образов Linux-c6x на модуль	14
4.1 Запись загрузчика IBL в EEPROM	14
4.2 Запись образов в NAND-флеш память	16
4.3 Запись образов в NOR-флеш память	18
5 Загрузка Linux-c6x	19
5.1 Настройка фиксированного IP-адреса	21
6 Подключение к Linux-c6x	22
6.1 Подключение по последовательному порту	22
6.2 Подключение по TELNET протоколу	26
7 Передача файлов в Linux-c6x	28
7.1 Передача файлов через TFTP	28
7.2 Передача файлов через FTP	28
Приложение А: Аппаратная конфигурация модуля SAMC-403	29
Приложение Б: Листинг файла конфигурации «setenv»	30
Список литературы	32

Список рисунков

2-1	Редактирование файла «setenv». Параметр AUTO_INSTALL	9
2-2	Редактирование файла «setenv». Параметр ROOTFS	9
2-3	Редактирование файла «setenv». Параметры BUILD_BOOTLOADERS и BUILD_SYSLINK.....	10
2-4	Файл «setenv», параметр BOOTBLOBS	12
6-1	Программа minicom	22
6-2	Программа minicom, настройки последовательного порта	23
6-3	Программа minicom, настройки модема	23
6-4	Программа minicom, подключение	24
6-5	Программа PuTTY, настройки последовательного порта	24
6-6	Программа PuTTY, настройка кодировки	25
6-7	Программа PuTTY, управление соединениями (Serial)	25
6-8	TELNET-сессия в Windows системе	26
6-9	Программа PuTTY, управление соединениями (TELNET)	27
7-1	FTP-сессия в Windows системе	28

Список таблиц

2-1	Назначение устанавливаемых пакетов	7
2-2	Зависимости для сборки Linux-сбх	7
2-3	Назначение сгенерированных файлов Linux-сбх	11
A-1	Положение переключателей для программирования модуля SAMC-403	29
A-2	Положение переключателей для загрузки модуля SAMC-403 с NAND-флеш памяти	29
A-3	Положение переключателей для загрузки модуля SAMC-403 с TFTP-сервера	29

Список листингов

5-1	Вывод в UART загрузки системы Linux-сбх с NAND-флеш памяти	19
Б-1	Листинг файла конфигурации «setenv»	30

Список процедур

4-1	Обновление GEL файла конфигурации CCS	14
4-2	Запись загрузчика на модуль SAMC-403 в Linux системе	14
4-3	Запись загрузчика на модуль SAMC-403 в Windows системе	15
4-4	Запись образа системы в NAND-флеш память	18

Перечень сокращений и условных обозначений

CCS	Code Composer Studio	7, 14, 15, 18
CGT	Code Generation Tools	7
DHCP	Dynamic Host Configuration Protocol	21, 22, 26
EEPROM	Electrically Erasable Programmable Read-Only Memory	14–16
FTP	File Transfer Protocol	28
GEL	General Extension Language	14
I²C	Inter-Integrated Circuit	14, 15
IBL	Intermediate Boot Loader	8, 11, 14–16, 18, 29
IPC	Inter Process Communication	8
IP	Internet Protocol	21, 22, 26, 28
JTAG	Joint Test Action Group	7, 29
NAND	Not AND	7, 14, 16–19, 29
NOR	Not OR	18
RFC	Request For Comments	28
SPI	Serial Peripheral Interface	18
TELNET	TErminAl NETwork	26, 27
TFTP	Trivial File Transfer Protocol	14, 19, 28, 29
TI	Texas Instruments	7
UART	Universal Asynchronous Receiver-Transmitter	19
USB	Universal Serial Bus	14, 15, 18
YACC	Yet Another Compiler Compiler	7
ОС	Операционная Система	6

1 Общие сведения

В данном документе рассмотрен процесс сборки системы Linux-сбх из исходных кодов и запуск её на модуле SAMC-403. Все, что описано в данном документе, было выполнено и проверено на 32-х разрядной ОС Ubuntu 10.04.

Использование других 32-х разрядных Linux операционных систем 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
```

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.

2.2 Получение исходных кодов Linux-c6x

Загрузка исходных кодов Linux-c6x выполняется специальным скриптом bootstrap. Для загрузки скрипта bootstrap в папку «~/my-linux-c6x» (в эту же папку будут загружены исходные коды Linux-c6x) выполните в терминале следующие команды:

```
mkdir ~/my-linux-c6x
cd ~/my-linux-c6x
wget http://linux-c6x.org/bootstrap
```

Для запуска скрипта, необходимо установить ему атрибут запускаемого файла:

```
chmod +x bootstrap
```

Для запуска скрипта bootstrap выполните команду:

```
./bootstrap
```

После запуска скрипта bootstrap, будет запущен процесс клонирования git-репозитория исходных кодов Linux-c6x. Данный процесс может занять от нескольких минут до нескольких часов, в зависимости от скорости соединения с сетью интернет.

Примечание

Для успешного выполнения указанных команд необходимо подключение к сети интернет.

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
CCS (Code Composer Studio) 5.0.3.00028	JTAG отладка, JTAG запись загрузчика, JTAG запись образа в NAND-флеш память
TI CGT (Code Generation Tools) 7.2.2	Сборка SysLink, rio-utils, bootloader

Продолжение таблицы на следующей странице

Продолжение таблицы 2-2

Зависимость	Необходимо для
SYS/BIOS 6.32.01.38	Сборка SysLink, примеры SYS/BIOS
IPC (Inter Process Communication) 1.23.01.26	Сборка SysLink
XDC Tools 3.22.01.21	Сборка SysLink

Дистрибутивы указанных зависимостей имеются на сопроводительном диске к модулю SAMC-403 в папке «linux-сбх/deps». Для их установки, необходимо переписать файлы дистрибутивов в папку «~/my-linux-сбх/download». Для этого, выполните следующие команду:

```
cp /mnt/cdrom/linux-с6х/deps/* ~/my-linux-с6х/download
```

В данном случае, предполагается, что сопроводительный диск к модулю SAMC-403 смонтирован в папку «/mnt/cdrom».

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

```
cd linux-с6х-project
```

Выполните скрипт конфигурации:

```
./prj config
```

В терминал будет выведено сообщение:

```
scripts/setup
Edit setenv for your configuration and rerun setup
```

В сообщении сказано, что для конфигурации, необходимо изменить файл «setenv» и перезапустить скрипт конфигурации.

Для редактирования файла «setenv» можно воспользоваться текстовым редактором nano. Запуск редактора nano для редактирования файла «setenv» выполняется следующей командой:

```
nano setenv
```

Примечание

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

В файл «setenv» необходимо внести следующие изменения:

- 1) Добавить параметр AUTO_INSTALL со значением «yes» (см. рисунок 2-1). Параметр AUTO_INSTALL отвечает за автоматическую установку необходимых зависимостей при конфигурации.
- 2) Изменить значение параметра ROOTFS на «min-root mcsdk-demo-root full-root» (см. рисунок 2-2).
- 3) Изменить значение параметра BUILD_BOOTLOADERS на «yes» (см. рисунок 2-3). Параметр BUILD_BOOTLOADERS указывает на необходимость сборки загрузчика IBL (Intermediate Boot Loader).
- 4) Если требуется сборка библиотеки SysLink, изменить значение параметра BUILD_SYSLINK на «yes» (см. рисунок 2-3).

После выполнения изменений в файле «setenv», запустите скрипт конфигурации повторно:

```
./prj config
```

Скрипт автоматически выполнит установку всех необходимых зависимостей из папки «~/my-linux-сбх/download». Зависимости, дистрибутивов которых нет в папке «~/my-linux-сбх/download», будут загружены из сети интернет и установлены.


```

kikin@vm-ubuntu-10-04: ~/my-linux-c6x/linux-c6x-project
File Edit View Terminal Help
GNU nano 2.2.2 File: setenv

# This file should be sourced (not run) into your shell to put the linux-c6x pr$
# example:
# /home/user/my-prj/linux-c6x-project source setenv
AUTO_INSTALL=yes

# this is infrastructure boilerplate; don't edit
SETENV_SOURCED=1
SETENV_FILE=${BASH_SOURCE[0]}
if [ -z "$SETENV_LOCAL_SOURCED" ]; then
    if [ -z "$SETENV_FILE" ]; then echo "Your shell is not BASH, you must$
    SETENV_LOCAL_FILE=$(dirname $SETENV_FILE)/.setenv.local
    if [ ! -r $SETENV_LOCAL_FILE ]; then echo "You have not run ./prj con$
    source $SETENV_LOCAL_FILE
fi

# Choose kernels to build
# This is a space-separated list of kernels to build. The names used are
# used to include a kbuilds/<name>.mk makefile fragment from the top-level
# Makefile.
[ Read 125 lines ]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

```

Рисунок 2-1: Редактирование файла «setenv». Параметр AUTO_INSTALL

```

kikin@vm-ubuntu-10-04: ~/my-linux-c6x/linux-c6x-project
File Edit View Terminal Help
GNU nano 2.2.2 File: setenv

export CGT_LINUX_VERSION=none

# set to yes to install extra kernel modules and scripts used for testing
#export BUILD_TESTS=yes

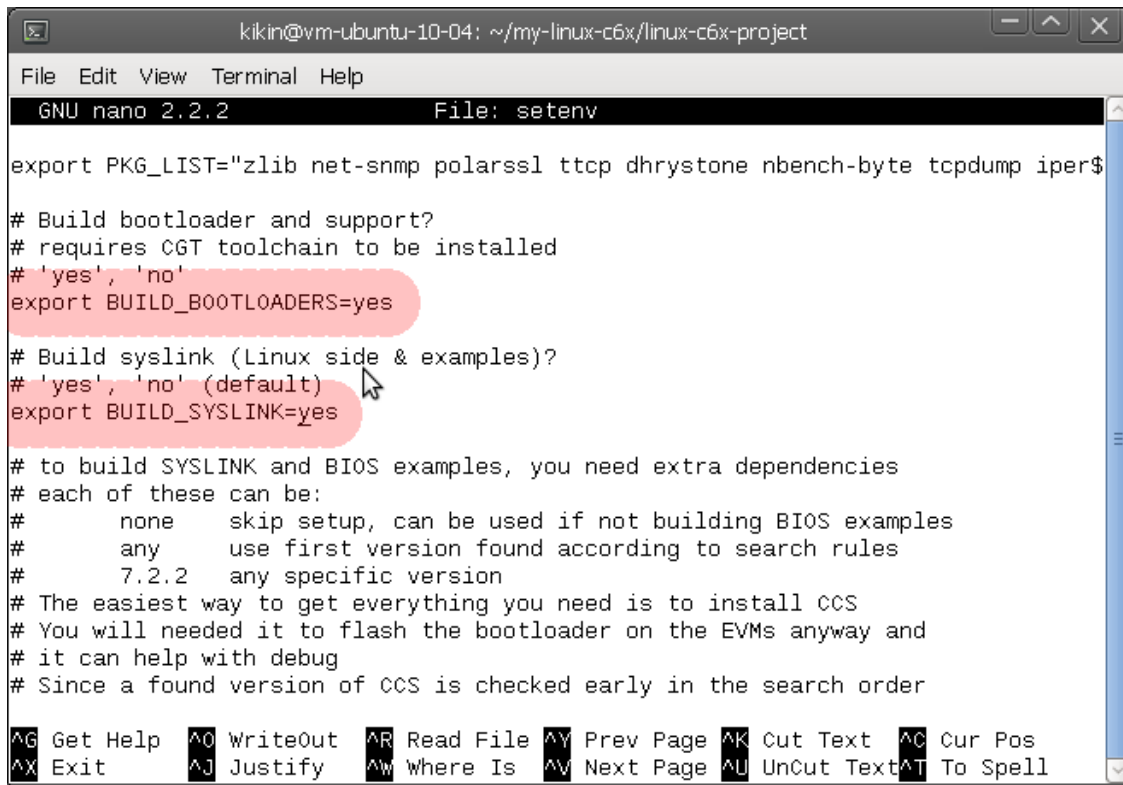
# Choices for ROOTFS, one or more of
# min-root - minimum file system
# full-root - min-root + additional packages such as nbench, polar ssl, $
# ltp-root - min-root + ltp test executables
# mcsdk-demo-root - min-root + mcsdk web control panel demo
export ROOTFS="min-root mcsdk-demo-root full-root"

# Choices for bootblobs to create
# can list zero, one, or more
# see bootblob-templates/* for a list of choices
# can also be "all" to build all combinations for the kernels in ../product/
# you need to ensure that kernels and rootfs'es needed are specified above,
# this does not do dependencies
export BOOTBLOBS="all"

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

```

Рисунок 2-2: Редактирование файла «setenv». Параметр ROOTFS



```
kikin@vm-ubuntu-10-04: ~/my-linux-c6x/linux-c6x-project
File Edit View Terminal Help
GNU nano 2.2.2 File: setenv

export PKG_LIST="zlib net-snmp polarssl ttcp dhrystone nbench-byte tcpdump iper$
# Build bootloader and support?
# requires CGT toolchain to be installed
# 'yes', 'no'
export BUILD_BOOTLOADERS=yes
# Build syslink (Linux side & examples)?
# 'yes', 'no' (default)
export BUILD_SYSLINK=yes
# to build SYSLINK and BIOS examples, you need extra dependencies
# each of these can be:
# none skip setup, can be used if not building BIOS examples
# any use first version found according to search rules
# 7.2.2 any specific version
# The easiest way to get everything you need is to install CCS
# You will needed it to flash the bootloader on the EVMs anyway and
# it can help with debug
# Since a found version of CCS is checked early in the search order

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

Рисунок 2-3: Редактирование файла «setenv». Параметры BUILD_BOOTLOADERS и BUILD_SYSLINK

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

```
setup is complete
```

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 130688
-rwxr-xr-x 1 user 11018 2012-10-08 14:38 bootblob
drwxr-xr-x 4 user 4096 2012-10-08 14:38 bootblob-templates
-rw-r--r-- 1 user 10485760 2012-10-10 14:31 evmc6678-initramfs-demo.el-hf-dev-user-20121009.bin
-rw-r--r-- 1 user 4547559 2012-10-10 14:31 evmc6678-initramfs-demo.el-hf-dev-user-20121009.cpio.gz
-rw-r--r-- 1 user 7340032 2012-10-10 14:31 evmc6678-initramfs-min.el-hf-dev-user-20121009.bin
-rw-r--r-- 1 user 2079306 2012-10-10 14:31 evmc6678-initramfs-min.el-hf-dev-user-20121009.cpio.gz
-rwxr-xr-x 1 user 4624896 2012-10-10 14:31 evmc6678-jffs2.el-hf-dev-user-20121009.bin
-rw-r--r-- 1 user 43805988 2012-10-10 14:31 evmc6678-jffs2.el-hf-dev-user-20121009.jffs2
-rwxr-xr-x 1 user 4624896 2012-10-10 14:31 evmc6678-nfs.el-hf-dev-user-20121009.bin
-rw-r--r-- 1 user 20062409 2012-10-10 14:31 evmc6678-nfs.el-hf-dev-user-20121009.cpio.gz
-rw-r--r-- 1 user 17131338 2012-10-10 18:22 full-root-c6x-hf.cpio.gz
-rw-r--r-- 1 user 145294 2012-10-10 18:13 gplv3-devtools-c6x-hf.cpio.gz
-rw-r--r-- 1 user 51912 2012-10-10 18:23 i2crom_0x51_c6678_le.bin
-rwxr-xr-x 1 user 7792 2012-10-08 14:38 make-filestystem
-rw-r--r-- 1 user 2037194 2012-10-10 18:14 mcsdk-demo-root-c6x-hf.cpio.gz
-rw-r--r-- 1 user 1958760 2012-10-10 18:13 min-root-c6x-hf.cpio.gz
-rw-r--r-- 1 user 540431 2012-10-09 12:02 modules-2.6.34-evmc6678.el-dev-user-20121009.tar.gz
-rw-r--r-- 1 user 2390122 2012-10-09 11:53 syslink-demo-evmc6678.el-hf-dev-user-20121009.tar.gz
-rwxr-xr-x 1 user 5417772 2012-10-09 11:40 vmlinux-2.6.34-evmc6678.el-dev-user-20121009
-rwxr-xr-x 1 user 4624896 2012-10-09 11:40 vmlinux-2.6.34-evmc6678.el-dev-user-20121009.bin
```

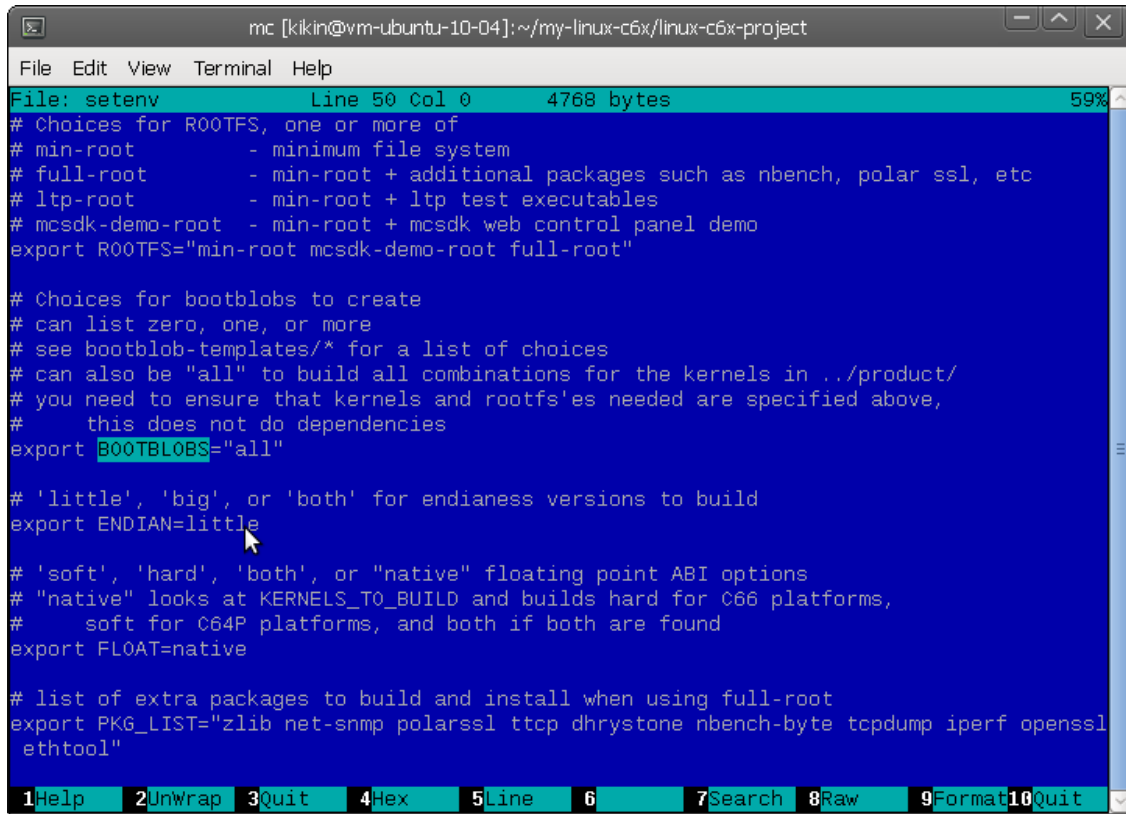
Краткая информация о сгенерированных файлах представлена в таблице 2-3.

Таблица 2-3: Назначение сгенерированных файлов Linux-c6x

Файл	Назначение
i2crom_0x51_c6678_le.bin	Бинарный образ загрузчика IBL (см. раздел 4.1).
vmlinux-2.6.34-evmc6678.el-dev-user-<дата_сборки>.bin	Собранное ядро Linux-c6x
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-evmc6678.el-dev-user-<дата_сборки>.tar.gz	Архив части файловой системы, включающий в себя дополнительные модули ядра.
syslink-demo-evmc6678.el-dev-user-<дата_сборки>.tar.gz	Архив части файловой системы, включающий в себя демонстрационные программы SysLink и SYS/BIOS, запускаемые на системе Linux-c6x.

Файлы, названия которых начинается на «evmc6678-», это готовые к загрузке на модуль SAMC-403 образы системы с различными параметрами и файловыми системами. Эти файлы образов сгенерированы автоматически утилитой bootblob (см. раздел 3) при сборке системы.

Утилита bootblob предназначена для создания образов на основе шаблонов (см. раздел 3). Названия шаблонов, по которым будут сгенерированы образы при сборке системы, указываются в параметре BOOTBLOBS файла setenv. По умолчанию, параметру BOOTBLOBS присвоено значение «all», то есть генерация всех возможных образов (см. рисунок 2-4).



```
mc [kikin@vm-ubuntu-10-04]:~/my-linux-c6x/linux-c6x-project
File Edit View Terminal Help
File: setenv Line 50 Col 0 4768 bytes 59%
# Choices for ROOTFS, one or more of
# min-root - minimum file system
# full-root - min-root + additional packages such as nbench, polar ssl, etc
# ltp-root - min-root + ltp test executables
# mcsdk-demo-root - min-root + mcsdk web control panel demo
export ROOTFS="min-root mcsdk-demo-root full-root"

# Choices for bootblobs to create
# can list zero, one, or more
# see bootblob-templates/* for a list of choices
# can also be "all" to build all combinations for the kernels in ../product/
# you need to ensure that kernels and rootfs'es needed are specified above,
# this does not do dependencies
export BOOTBLOBS="all"

# 'little', 'big', or 'both' for endianness versions to build
export ENDIAN=little

# 'soft', 'hard', 'both', or "native" floating point ABI options
# "native" looks at KERNELS_TO_BUILD and builds hard for C66 platforms,
# soft for C64P platforms, and both if both are found
export FLOAT=native

# list of extra packages to build and install when using full-root
export PKG_LIST="zlib net-snmp polarssl ttpc dhrystone nbench-byte tcpdump iperf openssl
ethtool"

1Help 2Unwrap 3Quit 4Hex 5Line 6 7Search 8Raw 9Format10Quit
```

Рисунок 2-4: Файл «setenv», параметр BOOTBLOBS

3 Создание образов Linux-сбх

Для создания образов Linux-сбх для загрузки, включающих в себя ядро и файловую систему, служит специальная утилита `bootblob`, которая находится в папке «product» дерева каталогов исходных кодов Linux-сбх.

Утилита `bootblob` является shell-скриптом, предназначенным для сборки готовых к записи (загрузке) на модуль SAMC-403 образов, которые состоят из более мелких частей (ядро системы, дополнительные модули ядра, файловая система и т. п.). Части, из которых собирается конечный образ системы, получаются 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.

Скрипт `bootblob` может использоваться для создания образов двумя способами:

- 1) на основе шаблонов, которые хранятся в папке «product/bootblob-templates»;
- 2) без использования шаблонов, путем указания, в качестве параметров командной строки, файла ядра и файла образа файловой системы, из которых должен состоять образ системы.

Внимание



Скрипт `bootblob` должен запускаться из каталога «product» дерева каталогов исходных кодов Linux-сбх, т. е. из папки «~/my-linux-сбх/product».

Для сборки образа по определенному шаблону, используется вызов `bootblob` в следующем виде:

```
./bootblob <имя_шаблона>
```

Например, для сборки образа по шаблону `evmc6678-initramfs-demo`, необходимо выполнить команду:

```
./bootblob evmc6678-initramfs-demo
```

Для повторного создания полного набора образов, которые были получены в результате сборки (раздел 2.4), необходимо выполнить команду:

```
./bootblob all
```

Для подробного ознакомления с возможностями и параметрами командной строки скрипта `bootblob` выполните в терминале команду:

В папке «product», также, имеется утилита `make-filesystem`, предназначенная для создания образов файловой системы из отдельных компонентов. Компонентами файловой системы могут быть архивы с модулями ядра, бинарные запускаемые файлы, отдельная папка на диске и т. п. Утилита `make-filesystem` также является shell-скриптом.

Скрипт `make-filesystem` вызывается скриптом `bootblob` при создании образов файловых систем.

Для более подробной информации о возможностях утилиты `make-filesystem`, выполните в терминале:

```
./make-filesystem -h
```

4 Запись образов Linux-с6х на модуль

Для загрузки образов системы Linux-с6х используется загрузчик IBL, который должен быть записан в EEPROM память на I²C шине по адресу 0x51.

Загрузчик IBL может обеспечить загрузку образа системы Linux-с6х различными способами:

- Загрузка с NAND-флеш памяти. Для загрузки Linux-с6х с NAND-флеш памяти, положение переключателей на модуле SAMC-403 необходимо установить в соответствии с таблицей A-2 (см. приложение A).
- Загрузка по сети с TFTP-сервера. Для загрузки Linux-с6х с TFTP-сервера, положение переключателей на модуле SAMC-403 должно быть установлено в соответствии с таблицей A-3 (см. приложение A).

Примечание

Подробную информацию по настройке TFTP-сервера в Linux и Windows системах для сетевой загрузки можно найти в документе [1].

4.1 Запись загрузчика IBL в EEPROM

Для выполнения записи загрузчика, в системе должна быть установлена интегрированная среда разработки CCS. Запись загрузчика на модуль SAMC-403 можно выполнять как в Linux системе, так и в Windows системе.

Перед выполнением записи, необходимо обновить GEL файл конфигурации «evmc6678l.gel» модуля SAMC-403 для CCS, который имеется на сопроводительном диске к модулю в папке «linux-с6х/program_evm/gel».

Для обновления GEL файла конфигурации, выполните следующие действия, описанные в процедуре 4-1.

Процедура 4-1. Обновление GEL файла конфигурации CCS

1. В последующих шагах предполагается, что CCS установлена в папку «<CCS_INSTALL_DIR>» (обычно это «C:/Program Files/Texas Instruments/ccsv5»).
2. Закройте Code Composer Studio, если она открыта.
3. Замените существующий файл «evmc6678l.gel» в папке «<CCS_INSTALL_DIR>/ccs_base/emulation/boards/evmc6678l/gel» на файл «evmc6678l.gel» с сопроводительного диска к модулю SAMC-403.

Для записи загрузчика IBL на модуль SAMC-403 в Linux системе выполните шаги описанные в процедуре 4-2.

Процедура 4-2. Запись загрузчика на модуль SAMC-403 в Linux системе

1. На выключенном модуле SAMC-403 установите переключатели в соответствии с таблицей A-1 (см. приложение A).
2. Установите правильное значение переменной окружения DSS_SCRIPT_DIR. Если CCS установлена в папку «/opt/ti/ccsv5», то переменной DSS_SCRIPT_DIR должно быть присвоено значение «/opt/ti/ccsv5/ccs_base/scripting/bin». Сделать это можно выполнив команду:

```
./make-filesystem -h
```

3. Включите модуль SAMC-403 и дождитесь инициализации эмулятора и завершения USB инициализации (около 10 секунд для XDS100 эмулятора и около 45 секунд для XDS560 эмулятора).
4. Запустите программу program_evm для записи образа загрузчика IBL в EEPROM:

```
cd ~/my-linux-c6x/program_evm
$DSS_SCRIPT_DIR/dss.sh program_evm.js TMDSEVMC6678L-1e "eeprom51"
```

В случае успешной записи загрузчика в EEPROM, в терминал будут выведены следующие сообщения:

```
I2C Bus address is = 81
I2C writing has started
Please be Patient
I2C write complete, reading data
I2C read complete, comparing data
Data compare passed
```

Шаги, необходимые для выполнения записи загрузчика IBL в Windows системе, описаны в процедуре 4-3.

Процедура 4-3. Запись загрузчика на модуль SAMC-403 в Windows системе

1. На выключенном модуле SAMC-403 установите переключатели в соответствии с таблицей A.1 (см. приложение A).
2. Установите правильное значение переменной окружения DSS_SCRIPT_DIR. Если CCS установлена в папку «C:/Program Files (x86)/Texas Instruments/ccsv5», то переменной DSS_SCRIPT_DIR должно быть присвоено значение «C:/Program Files (x86)/Texas Instruments/ccsv5/ccs_base/scripting/bin». Сделать это можно выполнив команду:

```
set DSS_SCRIPT_DIR="C:\Program Files (x86)\Texas Instruments\ccsv5\ccs_base\scripting\bin"
```

3. Включите модуль SAMC-403 и дождитесь инициализации эмулятора и завершения USB инициализации (около 10 секунд для XDS100 эмулятора и около 45 секунд для XDS560 эмулятора).
4. Запустите программу program_evm для записи образа загрузчика IBL в EEPROM:

```
cd <путь к программе program_evm>
%DSS_SCRIPT_DIR%\dss.bat program_evm.js TMDSEVMC6678L-1e "eeprom51"
```

В случае успешной записи загрузчика в EEPROM, в терминал будут выведены следующие сообщения:

```
I2C Bus address is = 81
I2C writing has started
Please be Patient
I2C write complete, reading data
I2C read complete, comparing data
Data compare passed
```

Примечание

Процедуры 4-2 и 4-3 записывают данные только в EEPROM на I²C шине с адресом 0x51. Поэтому, при выполнении процедур 4-2 и 4-3, все предупреждения о невозможности записи других типов памяти следует игнорировать.

При запуске программы program_evm для записи содержимого EEPROM (на это указывает параметр командной строки «eeprom51»), будет записываться файл «~/my-linux-c6x/program_evm/binaries/evm6678l/eeprom51.bin».

Имя данного файла считывается программой program_evm из файла «~/my-linux-c6x/program_evm/binaries/eepromwriter_input.txt». По умолчанию, содержимое этого файла имеет следующий вид:

```
file_name = eeprom51.bin
bus_addr = 81
start_addr = 0
swap_data = 0
```

По умолчанию, файл «~/my-linux-c6x/program_evm/binaries/evm6678l/eeprom51.bin» является символической ссылкой на файл «i2crom_0x51_c6678_le.bin» в папке «product» дерева каталогов проекта Linux-c6x.

В свою очередь, файл «i2crom_0x51_c6678_le.bin» является бинарным образом загрузчика IBL, который был получен в результате последней сборки проекта (см. раздел 2.4).

Примечание

Подробную информацию по настройке и сборке загрузчика IBL на модуле SAMC-403 можно найти в документе [2].

В этом можно убедиться, выполнив последовательно следующие команды:

```
cd ~/my-linux-c6x/program_evm/binaries/evm6678l
ls -l eeprom51.bin
```

Вывод должен выглядеть следующим образом:

```
lrwxrwxrwx 1 user user 41 2012-10-10 18:23 eeprom51.bin -> ../../../../product/i2crom_0x51_c6678_le.
< bin
```

Таким образом, в случае, если потребуется записывать какой либо другой файл образа в EEPROM память модуля SAMC-403, это можно сделать одним из следующих способов:

- изменить значение параметра «file_name» в файле «eepromwrite_input.txt» в папке «~/my-linux-c6x/program_evm/binaries».
- изменить символическую ссылку или заменить файл «eeprom51.bin» в папке «~/my-linux-c6x/program_evm/binaries/evm6678l» требуемым файлом.

Примечание

При необходимости использовать Windows систему для записи загрузчика IBL в EEPROM память, необходимо переписать все содержимое папки «~/my-linux-c6x/program_evm» с Linux машины на Windows машину вместе с бинарным файлом eeprom51.bin и запускать программу program_evm согласно процедуре 4-3.

4.2 Запись образов в NAND-флеш память

Для записи образов в NAND-флеш память, может использоваться программа program_evm, работа с которой описана в разделе 4.1 для записи загрузчика в EEPROM.

Отличие заключается в параметрах командной строки программы program_evm при запуске.

При запуске программы program_evm для записи содержимого NAND-флеш памяти (на это указывает параметр командной строки «nand»), будет записываться файл «~/my-linux-c6x/program_evm/binaries/evm6678l/nand.bin».

Имя данного файла считывается программой program_evm из файла «~/my-linux-c6x/program_evm/binaries/nand_writer_input.txt». По умолчанию, содержимое этого файла имеет следующий вид:

```
file_name = nand.bin
start_addr = 16384
```

По умолчанию, файл «~/my-linux-c6x/program_evm/binaries/evm6678l/nand.bin» является символической ссылкой на файл «evmc6678-initramfs-demo.el-hf-dev-kikin-<дата_сборки>.bin» в папке «product» дерева каталогов проекта Linux-c6x.

Файл «evmc6678-initramfs-demo.el-hf-dev-kikin-<дата_сборки>.bin» является одним из образов системы Linux-c6x, включающим в себя ядро системы и файловую систему, который был получен в результате последней сборки проекта (см. раздел 2.4).

Примечание

Информация по сборке образов для записи в NAND-флеш память дана в разделе [3](#).

Процедура записи образа в NAND-флеш память выполняется следующим образом:

Процедура 4-4. Запись образа системы в NAND-флеш память

1. На выключенном модуле SAMC-403 установите переключатели в соответствии с таблицей [A-1](#) (см. приложение [A](#)).
2. Установите правильное значение переменной окружения `DSS_SCRIPT_DIR`. Если `CCS` установлена в папку «`/opt/ti/ccsv5`», то переменной `DSS_SCRIPT_DIR` должно быть присвоено значение «`/opt/ti/ccsv5/ccs_base/scripting/bin`». Сделать это можно выполнив команду:

```
export DSS_SCRIPT_DIR=/opt/ti/ccsv5/ccs_base/scripting/bin
```

3. Включите модуль SAMC-403 и дождитесь инициализации эмулятора и завершения USB инициализации (около 10 секунд для XDS100 эмулятора и около 45 секунд для XDS560 эмулятора).
4. Запустите программу `program_evm` для записи образа системы в NAND-флеш память:

```
cd ~/my-linux-c6x/program_evm  
$DSS_SCRIPT_DIR/dss.sh program_evm.js TMDSEVMC6678L-1e "nand"
```

В случае успешной записи образа в NAND-флеш память, в терминал будут выведены следующие сообщения:

```
TODO: add program output
```

5. Выключите модуль SAMC-403 и установите переключатели в положение соответствующее загрузке модуля с NAND-флеш памяти (таблица [A-2](#), приложение [A](#)).

4.3 Запись образов в NOR-флеш память

На модуле SAMC-403, помимо NAND-флеш памяти, имеется SPI NOR-флеш память объемом 16 Мбайт, доступная для записи пользовательских данных (приложений).

Загрузчик IBL позволяет загружать приложения с SPI NOR-флеш памяти, но система Linux-с6x не поддерживает работу с данным типом памяти. Однако, информацию по записи данных в NOR флеш память модуля SAMC-403 можно найти в документе [\[2\]](#).

5 Загрузка Linux-c6x

При включении модуля SAMC-403 будет выполнена загрузка системы Linux-c6x, в зависимости от положения переключателей на модуле (см. приложение A), с NAND-флеш памяти или с TFTP-сервера.

Ниже приведен вывод в UART загрузки Linux-c6x с NAND-флеш памяти:

Листинг 5-1: Вывод в UART загрузки системы Linux-c6x с NAND-флеш памяти

```

1 IBL: PLL and DDR Initialization Complete
2 IBL Result code 00
3 IBL: Booting from NAND
4 Linux version 2.6.34-evmc6678.el-linux-c6x-2.0.0.63
5 (ubuntu@cm-build.linux-c6x.org) (gcc version 4.5.1 (Sourcery CodeBench
6 Lite 4.5-124) ) #1 Sat Dec 10 07:09:40 UTC 2011
7 Designed for the EVMC6678 board, Texas Instruments.
8 CPU0: C66x rev 0x0, 1.2 volts, 1000MHz
9 Initializing kernel
10 physical RAM map changed by user
11 Built 1 zonelists in Zone order, mobility grouping on. Total pages:
12 65024
13 Kernel command line: console=ttyS0,115200 rw mem=256M ip=dhcp
14 initrd=0x80400000,0x500000
15 PID hash table entries: 1024 (order: 0, 4096 bytes)
16 Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)
17 Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)
18 Memory available: 250776k/258156k RAM, 0k/0k ROM (793k kernel code,
19 201k data)
20 SLUB: Genslabs=7, HWalign=128, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
21 Hierarchical RCU implementation.
22 RCU-based detection of stalled CPUs is enabled.
23 NR_IRQS:288
24 Console: colour dummy device 80x25
25 Calibrating delay loop... 999.42 BogoMIPS (lpj=1998848)
26 Mount-cache hash table entries: 512
27 C64x: 9 gpio irqs
28 NET: Registered protocol family 16
29 SGMII init complete
30 bio: create slab <bio-0> at 0
31 Switching to clocksource TSC64
32 NET: Registered protocol family 2
33 IP route cache hash table entries: 2048 (order: 1, 8192 bytes)
34 TCP established hash table entries: 8192 (order: 4, 65536 bytes)
35 TCP bind hash table entries: 8192 (order: 3, 32768 bytes)
36 TCP: Hash tables configured (established 8192 bind 8192)
37 TCP reno registered
38 UDP hash table entries: 256 (order: 0, 4096 bytes)
39 UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
40 NET: Registered protocol family 1
41 RPC: Registered udp transport module.
42 RPC: Registered tcp transport module.
43 RPC: Registered tcp NFSv4.1 backchannel transport module.
44 Trying to unpack rootfs image as initramfs...
45 Freeing initrd memory: 5120k freed
46 JFFS2 version 2.2. (NAND) (SUMMARY) © 2001-2006 Red Hat, Inc.
47 ROMFS MTD (C) 2007 Red Hat, Inc.
48 msgmni has been set to 499
49 Block layer SCSI generic (bsg) driver version 0.4 loaded (major 254)
50 io scheduler noop registered
51 io scheduler deadline registered
52 io scheduler cfq registered (default)
53 Serial: 8250/16550 driver, 1 ports, IRQ sharing disabled
54 serial8250.0: ttyS0 at MMIO 0x2540000 (irq = 276) is a 16550A
55 console [ttyS0] enabled
56 brd: module loaded
57 loop: module loaded
58 at24 1-0050: 131072 byte 24c1024 EEPROM (writable)
59 uclinux[mtd]: RAM probe address=0x803e4760 size=0x0
60 Creating 1 MTD partitions on "RAM":
61 0x000000000000-0x000000000000 : "ROMfs"

```

```
62 mtd: partition "ROMfs" is out of reach~--- disabled
63 Generic platform RAM MTD, (c) 2004 Simtec Electronics
64 NAND device: Manufacturer ID: 0x20, Chip ID: 0x36 (ST Micro NAND 64MiB
65 1,8V 8-bit)
66 Scanning device for bad blocks
67 Creating 3 MTD partitions on "davinci_nand.0":
68 0x000000000000-0x000000004000 : "bootconfig"
69 0x000000004000-0x000001000000 : "kernel"
70 0x000001000000-0x000004000000 : "filesystem"
71 davinci_nand davinci_nand.0: controller rev. 2.5
72 m25p80 spi0.0: n25q128 (16384 Kbytes)
73 Creating 1 MTD partitions on "spi_flash":
74 0x000000000000-0x000001000000 : "test"
75 spi_davinci spi_davinci.0: Controller at 0x20bf0000
76 spi_davinci spi_davinci.0: Operating in interrupt mode using IRQ 182
77 keystone_netcp keystone_netcp.0: firmware: using built-in firmware
78 keystone-pdsp/qmss_pdsp_acc48_le.fw
79 keystone_netcp keystone_netcp.0: firmware: using built-in firmware
80 keystone-pdsp/pa_pdsp_default.fw
81 pktgen 2.72: Packet Generator for packet performance testing.
82 TCP cubic registered
83 NET: Registered protocol family 17
84 Sending DHCP requests ., OK
85 IP-Config: Got DHCP answer from 192.168.1.102, my address is 192.168.1.87
86 IP-Config: Complete:
87     device=eth0, addr=192.168.1.87, mask=255.255.255.0, gw=255.255.255.255,
88     host=tic6678-0d1206, domain=, nis-domain=(none),
89     bootserver=192.168.1.102, rootserver=192.168.1.102, rootpath=
90 Freeing unused kernel memory: 140K freed
91 starting pid 17, tty '': '/etc/rc.sysinit'
92
93 Starting system...
94
95 Mounting proc filesystem: done.
96 Mounting other filesystems: done.
97 Starting mdev
98 Setting hostname tic6678-0d1206: done.
99 Bringing up loopback interface: done.
100 Starting inetd: done.
101
102
103 eth0 Link encap:Ethernet HWaddr 90:D7:EB:0D:12:06
104     inet addr:192.168.1.87 Bcast:192.168.1.255
105     Mask:255.255.255.0
106     UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
107     RX packets:2 errors:0 dropped:0 overruns:0 frame:0
108     TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
109     collisions:0 txqueuelen:1000
110     RX bytes:692 (692.0 B) TX bytes:1180 (1.1 KiB)
111     Interrupt:48
112
113 System started.
114
115 starting pid 66, tty '/dev/console': '/bin/sh'
116 / #
```

Данный лог получен путем подключения к последовательному порту модуля терминальным клиентом. Информация по настройке терминального клиента для подключения к модулю SAMC-403 приведена в разделе 6.1.

5.1 Настройка фиксированного IP-адреса

По умолчанию, в строке параметров ядра Linux-сбх указана конфигурация IP-адреса через DHCP-сервер. Если DHCP-сервер отсутствует в сети, к которой подключен модуль SAMC-403, во время загрузки Linux-сбх после двух попыток отправки DHCP-запросов будет выведено приглашение на последовательном порту, подключившись к которому, можно задать системе фиксированный IP-адрес.

Примечание

Информация по настройке терминального клиента, для подключения к последовательному порту модуля SAMC-403 дана в разделе 6.1.

Например, для установки фиксированного IP-адреса 192.168.20.10 с маской подсети 255.255.255.0 на интерфейсе eth0, необходимо ввести команду:

```
ifconfig eth0 192.168.20.10 netmask 255.255.255.0 up
```

6 Подключение к Linux-c6x

6.1 Подключение по последовательному порту

Подключение к системе по последовательному может быть удобно в случае необходимости увидеть лог загрузки системы или в случае отсутствия в сети DHCP-сервера для настройки в системе фиксированного IP-адреса для сетевого интерфейса (см. раздел 5.1).

6.1.1 Подключение в Linux системе

В качестве терминального клиента для подключения к последовательному порту в Linux системе рекомендуется использовать программу minicom. Для установки minicom в системе Ubuntu 10.04 выполните команду:

```
sudo apt-get install minicom
```

После установки программы, запустите её с правами суперпользователя для конфигурации, выполнив команду:

```
sudo minicom -s
```

Выполните конфигурацию последовательного порта, выбрав пункт меню «Serial port setup» (см. рисунок 6-1). Установите параметры порта в соответствии с рисунком 6-2. Убедитесь, что имя устройства последовательного порта указано верно (параметр «Serial Device»).

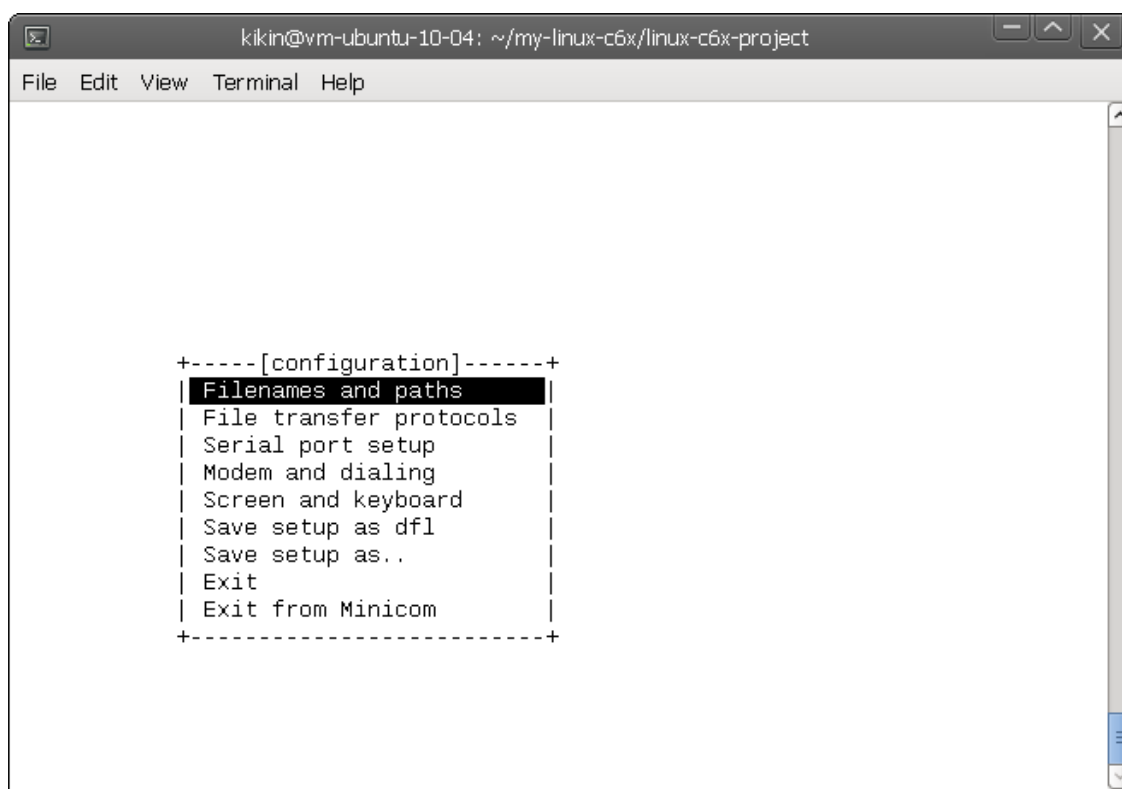


Рисунок 6-1: Программа minicom

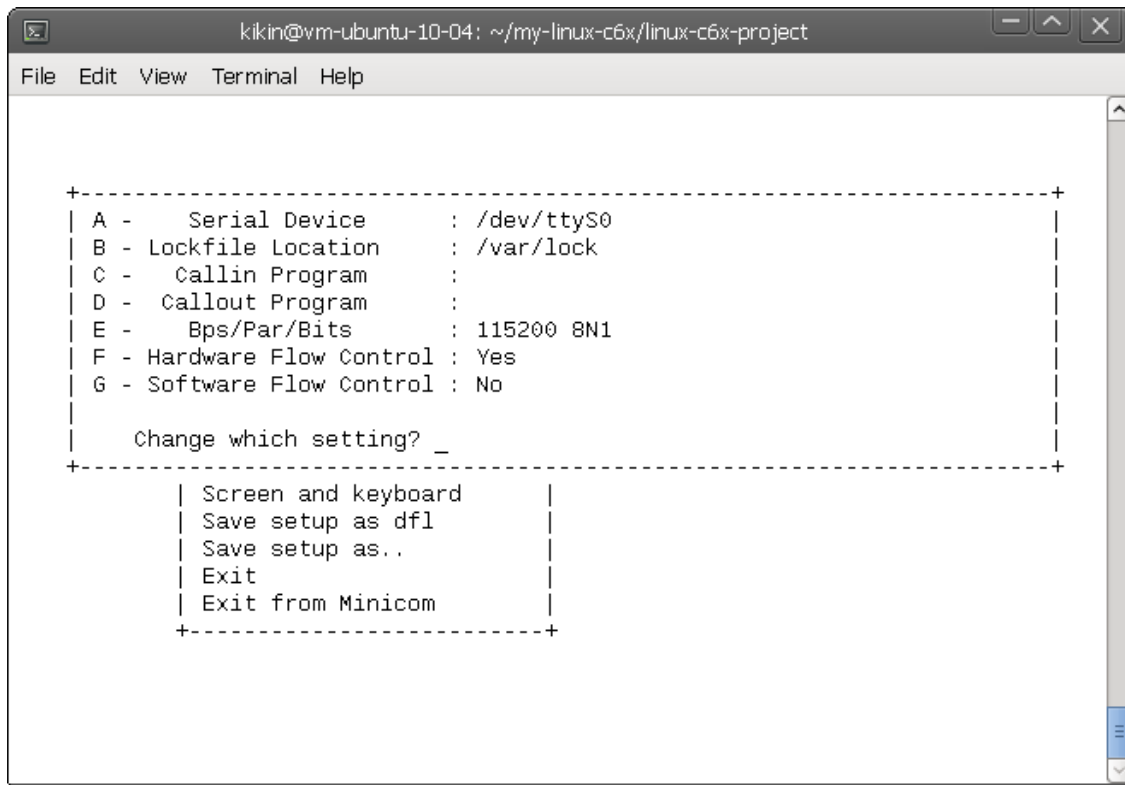


Рисунок 6-2: Программа minicom, настройки последовательного порта

Выберите пункт главного меню «Modem and dialing». В открывшемся окне настроек необходимо удалить значение параметра «Init string» (см. рисунок 6-3).

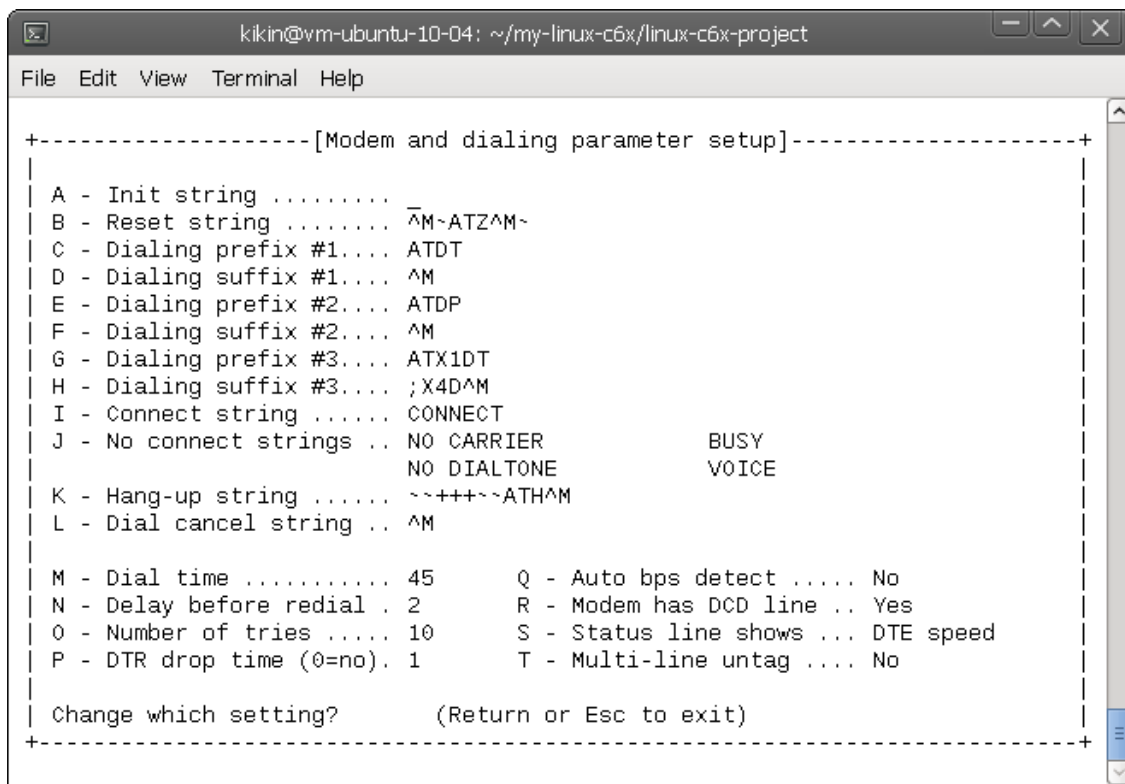


Рисунок 6-3: Программа minicom, настройки модема

Выберите пункт главного меню «Save setup as dfl». Произойдет сохранение сделанных настроек. Далее, выберите пункт меню «Exit». Произойдет подключение по последовательному порту к модулю SAMC-403 (см. рисунок 6-4).

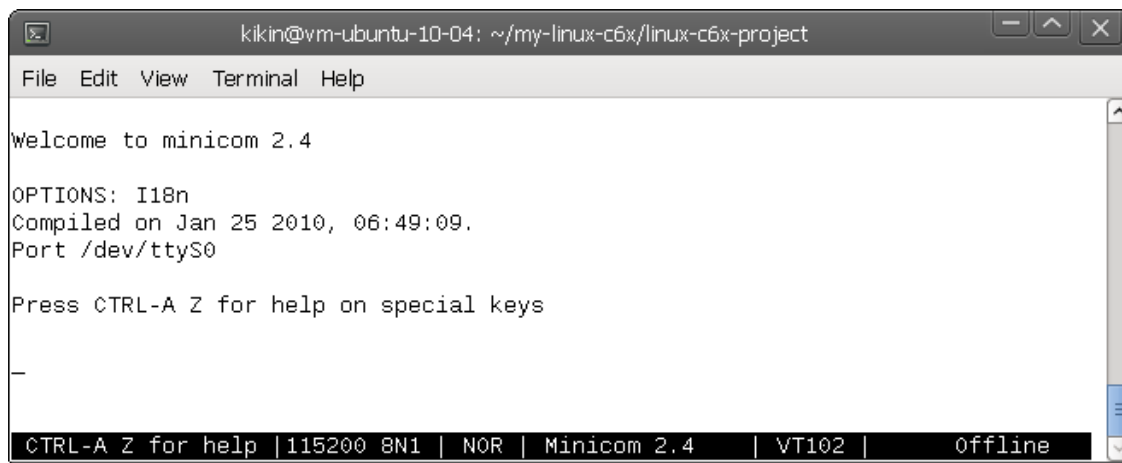


Рисунок 6-4: Программа minicom, подключение

После подключения, если система Linux-c6x уже загружена, нажмите несколько раз клавишу Enter на клавиатуре. Будет выведено приглашение системы Linux-c6x, в котором можно вводить команды.

Если на момент подключения, система Linux-c6x еще не была загружена, в терминал будет выводиться лог процесса загрузки системы. Пример такого лога приведен в разделе 5.

6.1.2 Подключение в Windows системе

В качестве терминального клиента для подключения к последовательному порту в Windows системе рекомендуется использовать программу PuTTY. Программа является бесплатной и её можно скачать с официального сайта по ссылке <http://www.chiark.greenend.org.uk/~sgtatham/putty/>.

Запустите программу PuTTY. В дереве навигации слева выберите раздел «Serial». Приведите настройки последовательного порта к виду, показанному на рисунке 6-5. Убедитесь, что имя устройства последовательного порта указано верно (параметр «Serial line to connect to»).

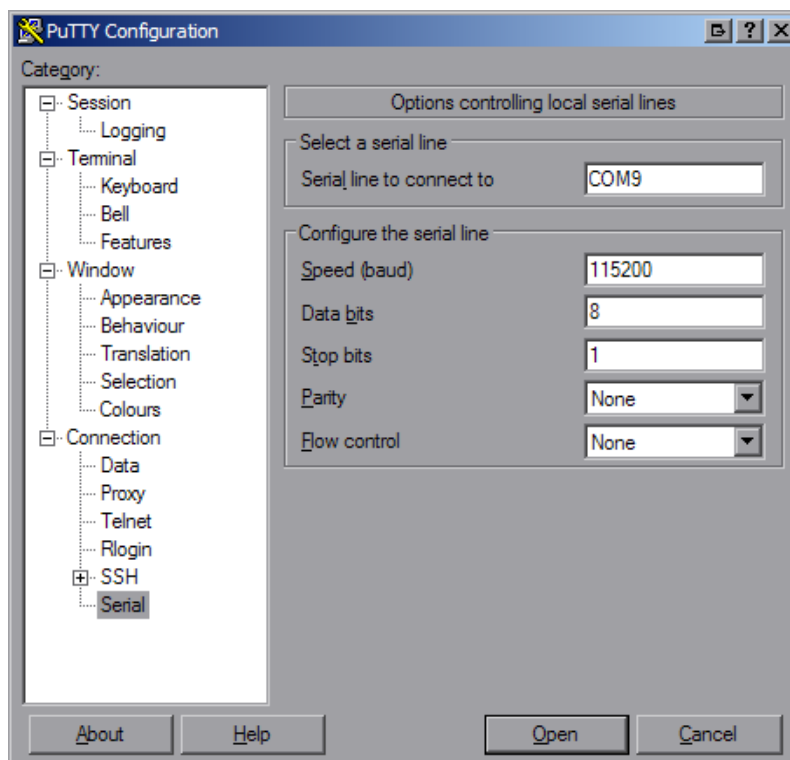


Рисунок 6-5: Программа PuTTY, настройки последовательного порта

Далее, необходимо указать правильную кодировку для соединения. Для этого, в дереве навигации слева выберите раздел «Window > Translation», и укажите кодировку UTF-8 (см. рисунок 6-6).

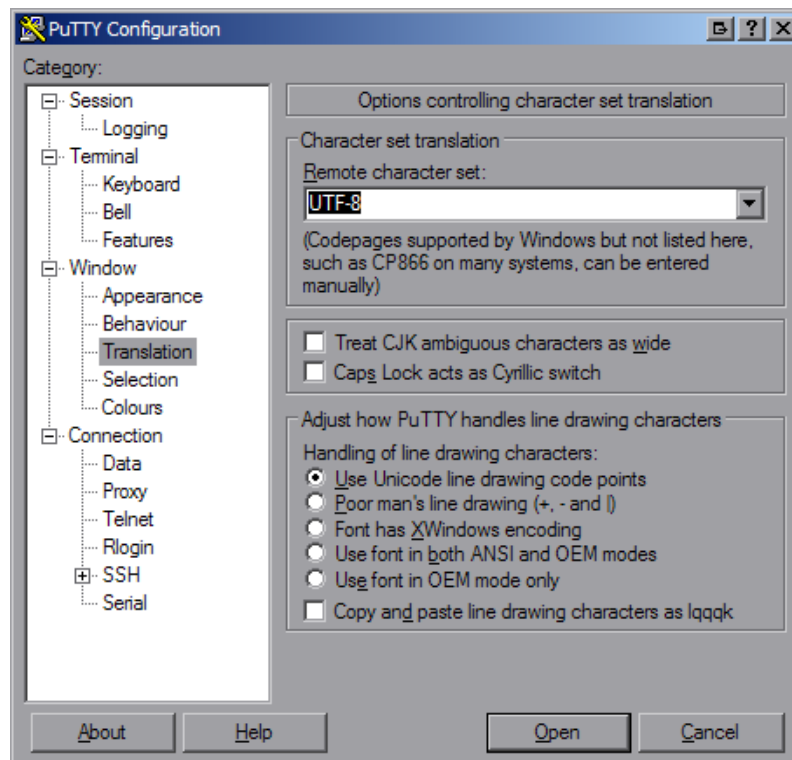


Рисунок 6-6: Программа PuTTY, настройка кодировки

В дереве навигации слева выберите раздел «Session». Установите параметру «Connection type» значение «Serial», как показано на рисунке 6-7.

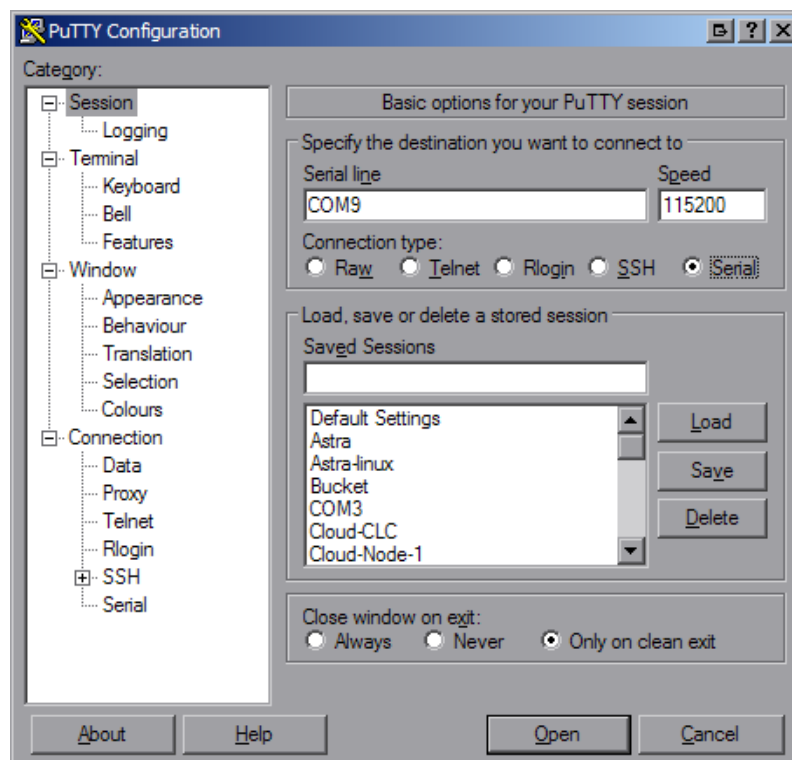


Рисунок 6-7: Программа PuTTY, управление соединениями (Serial)

Нажмите на кнопку «Open». Произойдет подключение по последовательному порту к модулю SAMC-403.

После подключения, если система Linux-сбх уже загружена, нажмите несколько раз клавишу Enter на клавиатуре. Будет выведено приглашение системы Linux-сбх, в котором можно вводить команды.

Если на момент подключения, система Linux-сбх еще не была загружена, в терминал будет выводиться лог процесса загрузки системы. Пример такого лога приведен в разделе 5.

6.2 Подключение по TELNET протоколу

Подключение по TELNET протоколу [3] 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-сбх может быть сконфигурирован посредством DHCP-сервера в сети или вручную (см. раздел 5.1).

Если IP-адрес системы конфигурируется через DHCP-сервер в сети, его можно узнать подключившись к системе через последовательный порт (см. раздел 6.1) и выполнив команду:

```
ifconfig
```

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

```
telnet <IP_адрес_системы>
```

Например, если IP-адрес системы 158.218.100.184, то необходимо выполнить команду:

```
telnet 158.218.100.184
```

Пример TELNET сессии в Windows системе приведен на рисунке 6-8.

```

localhost login: root
/root # ls
/root # cd /
/ # ls -l
drwxr-xr-x  2 13713  1262      0 May  4  2011 bin
drwxr-xr-x  2 13713  1262      0 May  4  2011 boot
drwxr-xr-x  6 13713  1262     0 Nov 30 00:02 dev
drwxr-xr-x  8 13713  1262      0 May  4  2011 etc
drwxr-xr-x  2 13713  1262      0 May  4  2011 home
lrwxrwxrwx  1 13713  1262    12 May  4  2011 init -> /bin/busybox
drwxr-xr-x  2 13713  1262      0 May  4  2011 initrd
drwxr-xr-x  3 13713  1262      0 May  4  2011 lib
drwxr-xr-x  2 13713  1262      0 May  4  2011 mnt
drwxr-xr-x  2 13713  1262      0 May  4  2011 opt
dr-xr-xr-x 30 root    root     0 Nov 30 00:00 proc
drwxr-xr-x  2 13713  1262      0 May  4  2011 root
drwxr-xr-x  2 13713  1262      0 May  4  2011/sbin
drwxr-xr-x 11 root    root     0 Nov 30 00:00 sys
drwxr-xr-x  2 root    root     0 Nov 30 00:08 tmp
drwxr-xr-x 13 13713  1262      0 May  4  2011 usr
drwxr-xr-x 15 13713  1262      0 May  4  2011 var
drwxr-xr-x  4 13713  1262      0 May  4  2011 web
/ #

```

Рисунок 6-8: TELNET-сессия 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 6-6).

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-сбх, как показано на рисунке 6-9.

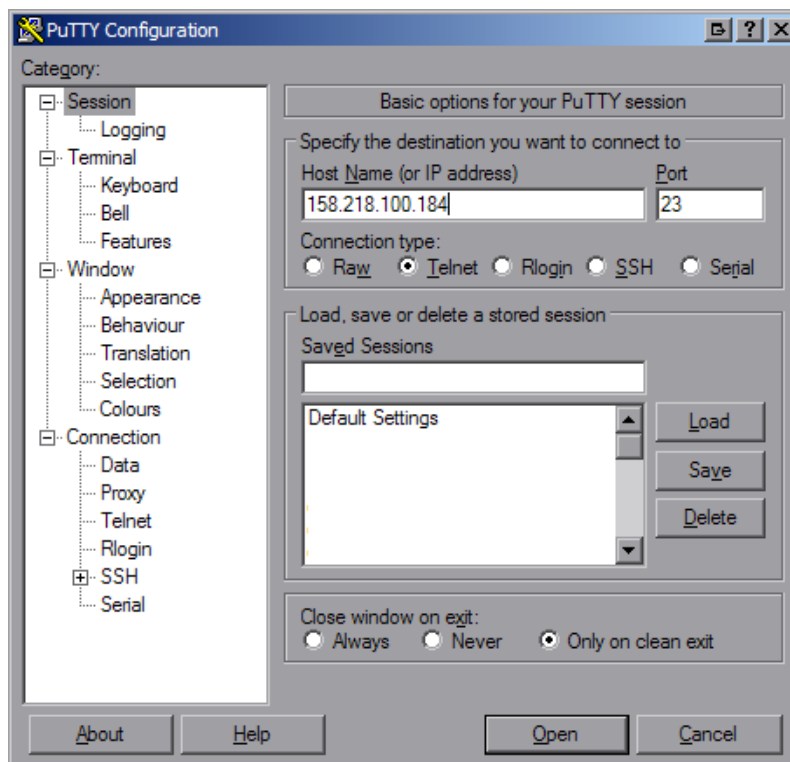


Рисунок 6-9: Программа PuTTY, управление соединениями (TELNET)

Нажмите на кнопку «Open». Произойдет подключение по протоколу TELNET к модулю SAMC-403 и будет выведено приглашение в следующем виде:

```
localhost login:
```

В качестве имени пользователя необходимо ввести «root». Пароль не требуется.

Приложение А Аппаратная конфигурация модуля SAMC-403

При загрузке и отладке программ напрямую через JTAG, без использования загрузчика IBL, положение переключателей на модуле SAMC-403 необходимо установить в соответствии с таблицей А-1.

Таблица А-1: Положение переключателей для программирования модуля SAMC-403

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

Для загрузки образа приложения с NAND-флеш памяти, положение переключателей на модуле SAMC-403 необходимо установить в соответствии с таблицей А-2.

Таблица А-2: Положение переключателей для загрузки модуля SAMC-403 с NAND-флеш памяти

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

Для загрузки образа приложения с TFTP-сервера, положение переключателей на модуле SAMC-403 необходимо установить в соответствии с таблицей А-3.

Таблица А-3: Положение переключателей для загрузки модуля SAMC-403 с TFTP-сервера

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

Приложение Б Листинг файла конфигурации «setenv»

Листинг Б-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. 14).
2. SAMC-403. Загрузчик IBL. Руководство пользователя. [UG-SAMC-403-IBL](#) (цит. на с. 16, 18).
3. J. Postel и J.K. Reynolds. Telnet Protocol Specification. RFC 854 (INTERNET STANDARD). Updated by RFC 5198. Internet Engineering Task Force, май 1983.
URL: <http://www.ietf.org/rfc/rfc854.txt> (цит. на с. 26).
4. K. Sollins. The TFTP Protocol (Revision 2). RFC 1350 (INTERNET STANDARD). Updated by RFCs 1782, 1783, 1784, 1785, 2347, 2348, 2349. Internet Engineering Task Force, июль 1992.
URL: <http://www.ietf.org/rfc/rfc1350.txt> (цит. на с. 28).
5. J. Postel и J. Reynolds. File Transfer Protocol. RFC 959 (INTERNET STANDARD). Updated by RFCs 2228, 2640, 2773, 3659, 5797. Internet Engineering Task Force, окт. 1985.
URL: <http://www.ietf.org/rfc/rfc959.txt> (цит. на с. 28).