ПАКЕТ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ РАСШИРЕННОЙ ПОДДЕРЖКИ СБИС 1888ТХ018. АДАПТИРОВАННЫЙ ЗАГРУЗЧИК U-BOOT ДЛЯ ПЛАТЫ МВ115.01 НА КРИСТАЛЛЕ СБИС 1888ТХ018

Руководство оператора

ЮФКВ.30168-02 34 01

Листов 25

Инв. № подл.		Подп. и дата		Взам. инв. №		Инв. № дубл.		Подп. и дата	

2020

АННОТАЦИЯ

Настоящее руководство распространяется на адаптированный загрузчик U-Boot для платы MB115.01 на кристалле СБИС 1888ТХ018 (далее – Загрузчик U-Boot, программа).

Документ содержит общие сведения о программе, условия её выполнения, описание действий оператора при выполнении программы, описание сообщений оператору.

Документ составлен с учётом требований ГОСТ 19.101-77, ГОСТ 19.105-78, ГОСТ 19.106-78.

СОДЕРЖАНИЕ

1. НАЗНАЧЕНИЕ ПРОГРАММЫ

1.1 Общие сведения

1.1.1. Адаптированный загрузчик U-Boot для платы MB115.01 на кристалле СБИС 1888ТХ018 входит в состав пакета программного обеспечения (далее по тексту ППО) расширенной поддержки СБИС 1888ТХ018, разработки НТЦ «Модуль», предназначенного для выполнения на аппаратных ресурсах платы MB115.01 на кристалле СБИС 1888ТХ018.

1.1.2. Загрузчик U-Boot предназначен для загрузки U-Boot, образа ядра и device tree с:

– SD карты;

– SPI Flash;

– Ethernet (TFTP, EDCL).

1.1.3. Загрузчик U-Boot обеспечивает следующие основные возможности:

– Конфигурация DDR памяти на базе данных в EEPROM;

– Поддержка драйверов:

- UART;
- Ethernet;
- SDIO;
- GPIO;
- SPI;
- PINMUX;
- USB.

– Поддержка протокола TFTP на канале Ethernet;

– Управление через командную строку на порту UART;

– Тестирование фиксированного блока памяти DDR встроенным тестом;

– Запись и чтение SD-карты;

– Запись и чтение файлов на томе FAT16/FAT32/EXT3/EXT4 на SD-карте;

– Сохранение рабочих переменных на SD карте или SPI flash памяти (в зависимости от параметров, указанных при компиляции);

– Выполнение команды `ums` (с использованием штатного драйвера inventra musb).

1.1.4. Используемая версия исходных кодов загрузчика U-Boot – 2020.10.

2. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

2.1 Требования к среде функционирования

2.1.1. Загрузчик U-Boot должен исполняться на аппаратных ресурсах платы MB115.01 на СБИС 1888ТХ018.

2.1.2. Условия эксплуатации Загрузчик U-Boot должны соответствовать условиям эксплуатации платы MB115.01.

2.2 Установка программы

2.2.1. Для корректной работы с SPI Flash носителем необходимо выполнить прошивку загрузчика U-Boot на носитель. Подробнее см. документ «Пакет программного обеспечения расширенной поддержки СБИС 1888ТХ018. Адаптированный загрузчик U-Boot для платы MB115.01 на кристалле СБИС 1888ТХ018. Инструкция по применению исходных кодов».

2.2.2. Для корректной работы с SD-картой необходимо выполнить прошивку загрузчика U-Boot на карту. Подробнее см. документ «Пакет программного обеспечения расширенной поддержки СБИС 1888ТХ018. Адаптированный загрузчик U-Boot для платы MB115.01 на кристалле СБИС 1888ТХ018. Инструкция по применению исходных кодов».

2.3 Подготовка к запуску программы

2.3.1. Для корректной работы загрузки по сети по протоколу TFTP оператор должен настроить перед работой с программой сервер TFTP.

3. ВЫПОЛНЕНИЕ ПРОГРАММЫ

3.1 Запуск программы

3.1.1. Программа запускается автоматически при подключении платы.

3.1.2. Управление программой выполняется через командную строку. Список всех доступных команд приведён в приложении А (см. Приложение А. Список доступных команд загрузчика U-Boot).

3.1.3. Для выведения полного списка доступных команд используется команда `help`.

3.1.4. Для получения справки по отдельной команде после команды `help` следует указать название интересующей команды. Например, при вводе `help base` оператору будет выведена следующая информация:

```
base - print or set address offset

Usage:
base
    - print address offset for memory commands
base off
    - set address offset for memory commands to 'off'=>
```

3.2 Загрузка образа ядра

3.2.1. Загрузка образа ядра может выполняться с SPI Flash, SD-карты или по сети автоматически при запуске работы загрузчика, если это задано в переменных окружения (подробнее см. п. 3.3), или по соответствующей команде.

3.2.1.1. К основным командам загрузки относятся:

– `boot` – выполнение команды загрузки по умолчанию, то есть выполнение команды из переменной окружения `bootcmd`;

– `bootd` – выполнение команды загрузки по умолчанию, то есть выполнение команды из переменной окружения `bootcmd`;

– `bootelf` –загрузка образа ELF из памяти;

– `bootm` – загрузка образа приложения из памяти;

```
bootm [addr [arg ...]]
```

– `bootp` – загрузка образа по сети при помощи BOOTP протокола;

```
bootp [loadAddress] [[hostIPaddr:]bootfilename]
```

где `[loadAddress]` – адрес, откуда выполняется загрузка;

`[hostIPaddr:]` – IP-адрес хост-сервера;

`[bootfilename]` – имя загружаемого файла.

– `bootvx` – загрузка образа приложения VxWorks;

– `tftp` – загрузка образа по сети при помощи протокола TFTP;

– `dhcp` – загрузка образа по сети при помощи протокола DHCP;

```
bootp [loadAddress] [[hostIPaddr:]bootfilename]
```

где `[loadAddress]` – адрес, откуда выполняется загрузка;

`[hostIPaddr:]` – IP-адрес хост-сервера;

`[bootfilename]` – имя загружаемого файла.

– `loadb` – загрузка бинарного файла через терминал по протоколу Kermit;

```
loadb [ off ] [ baud ]
```

где `[off]` – смещение;

`[baud:]` – скорость передачи.

– `loads` – загрузка файла в формате S-Record через терминал;

```
loads [ off ] [ baud ]
```

где `[off]` – смещение;

`[baud:]` – скорость передачи.

3.2.2. Загрузка образа по протоколу TFTP

3.2.2.1. Для корректной загрузки по протоколу TFTP должен быть настроен сервер TFTP и переменные окружения `ipaddr`, `serverip` и `bootfile`.

3.2.2.2. Для загрузки образа ядра через сеть по протоколу TFTP используется команда: `tftpboot`.

```
tftpboot [loadAddress] [[hostIPaddr:]bootfilename]
```

где `[loadAddress]` – адрес, откуда выполняется загрузка;

`[hostIPaddr:]` –IP-адрес хост-сервера;

`[bootfilename]` – имя загружаемого файла.

3.2.2.3. Если в строке не указан `[loadAddress]`, то значение берётся из переменной окружения `loadaddr`.

3.3 Настройка переменных окружения загрузчика

3.3.1. Переменные окружения используются для настройки поведения загрузчика по желанию оператора.

3.3.2. Во время запуска загрузчик выполняет поиск переменных окружения, сохраненных в SPI Flash-памяти:

3.3.2.1. При успешном обнаружении устанавливаются переменные окружения, сохраненные в SPI Flash-памяти;

3.3.2.2. В случае ошибки устанавливаются переменные окружения, заданные по умолчанию.

3.3.3. Для настройки значений переменных окружения используется команда `setenv`.

Новое значение задаётся по следующей схеме:

```
setenv name value
```

где `name` – имя переменной окружения, а `value` – значение переменной.

3.3.4. К переменным окружения загрузчика, которые может задать оператор, относятся:

– `ipaddr` - IP-адрес, необходимый для выполнения команды `tftp`. Подробнее см. п. 3.4;

– `serverip` - IP-адрес TFTP-сервера, необходимый для выполнения команды `tftp`. Подробнее см. п. 3.2.2;

– `netmask` – маска подсети;

– `loadaddr` – адрес буфера для копирования образа с SD/MMC-карты;

– `bootdelay` – время в секундах до выполнения команды, заданной в переменной `bootcmd` после перезагрузки загрузчика U-Boot. При этом на экране отображается обратный отсчёт, оператор может прервать выполнение команды, нажав любую кнопку на клавиатуре. Если значение переменной – «0», то загрузка выполняется сразу же;

– `bootfile` - имя файла, содержащего образ для загрузки образа по протоколу TFTP;

– `bootcmd` – команда, выполняемая при перезагрузке загрузчика после обратного отсчёта, заданного в переменной `bootdelay`;

– `bootm_low` – нижняя граница области памяти, используемой для загрузки образа;

– `bootm_size` – размер области памяти, используемой для загрузки образа;

– `fdt_addr_r` – адрес, где хранится device tree.

3.3.5. Команда `setenv` используется также для удаления заданного значения у переменной. Уже заданное значение удаляется по следующей схеме:

```
setenv name
```

где `name` – имя переменной окружения.

3.3.6. Для изменения уже заданных значений переменных окружения используется команда `editenv`.

Новое значение задаётся по следующей схеме:

```
edit name value
```

где `name` – имя переменной окружения, а `value` – новое значение переменной.

3.3.7. Все описанные выше изменения, выполняемые с переменными окружения, сохраняются только в RAM. При перезагрузке все настройки будут утеряны. Для того чтобы сохранить изменения, используется команда `saveenv`.

3.4 Работа с памятью

3.4.1. При частом обращении к определенному участку памяти устанавливают смещение для команд обращения при помощи команды `base`. Пример:

```
=> base 0x100000
Base Address: 0x00100000
```

3.4.1.1. При вводе команды без указания смещения выводится текущее значение смещения. Значение по умолчанию: 0.

```
=> base
Base Address: 0x00000000
```

3.4.1.2. Для отключения смещения используется команда `base off`.

3.4.2. Для подсчёта контрольных сумм CRC32 используется команда `crc32` с указанием диапазона памяти в качестве аргумента:

```
=>  crc 0x100004 0x3FC
CRC32 for 00100004 ... 001003ff ==> 8083764e
```

3.4.2.1. При вводе трёх аргументов произведённый расчёт сохраняется в указанном участке памяти:

```
=>  crc 0x100004 0x3FC 0x100000
CRC32 for 00100004 ... 001003ff ==> 8083764e
=> md 0x100000 4
00100000: 8083764e bd86200a 60a19054 2c12c402    ..vN.. .`..T,...
=>
```

3.4.3. Для сравнения содержимого памяти двух участков памяти используется команда `cmp`:

```
cmp [.b, .w, .l] addr1 addr2 count
```

3.4.3.1. Команда выполнит либо полную проверку участка, заданного в третьем аргументе (длина), либо остановится при первом обнаруженном различии:

```
=> cmp 0x100000 0x200000 0x400
word at 0x00100000 (0x8083764e) != word at 0x00200000 (0x27051956)
Total of 0 words were the same
```

3.4.3.2. Команда может получить доступ к участкам памяти с разными размерами информации:

`cmp.l` - 32 бита (longword). По умолчанию сравнение выполняет для 32-битной памяти;

`cmp.w` - 16 бита (word),

`cmp.b` - 8 байт (byte).

3.4.3.3. Аргумент `count` задаёт количество обрабатываемых единиц данных, т.е. размер машинных слов и символов (32 бита, 16 бит или 8 байт).

3.4.4. Команда `cp` используется для копирования участков памяти:

```
cp [.b, .w, .l] source target count
```

где `[.b, .w, .l]` – разрядность копируемой информации в битах или байтах,

`source` – участок, откуда копируются данные;

`target` – участок, куда копируются данные;

`count` – количество копируемых данных в указанной величине.

Например:

```
=> cp.l 0x200000 0x100000 0x10000
=> cp.w 0x200000 0x100000 0x20000
=> cp.b 0x200000 0x100000 0x40000
```

3.4.5. Команда `md` используется для отображения содержимого памяти:

```
md [.b, .w, .l] address [# of objects]
```

где `[.b, .w, .l]` – разрядность отображаемой информации в битах или байтах,

`address` – адрес участка памяти;

`[# of objects]` – количество объектов.

3.4.5.1. Отображение содержимого памяти выполняется и в шестнадцатеричной системе счисления, и в ASCII:

```
=> md 0x100000
00100000: 8083764e bd86200a 60a19054 2c12c402    ..vN.. .`..T,...
00100010: c101d028 00438198 7ab01239 62406128    ...(.C..z..9b@a(
00100020: 0c900d05 320b4581 1ca3d0a2 c498293a    ....2.E.......):
```

3.4.5.2. Загрузчик запоминает последний указанный адрес и аргумент количества, поэтому при вводе команды без аргументов, будет автоматически подставлен следующий адрес и прежний аргумент количества.

3.4.6. Команда `mm` используется для изменения содержимого памяти:

```
mm [.b, .w, .l] address
```

где `[.b, .w, .l]` – разрядность изменяемой информации в битах или байтах,

`address` – адрес участка памяти;

3.4.6.1. После ввода команды на экран выводится текущее содержимое участка памяти и знак вопроса для ввода новых данных в шестнадцатеричной системе:

```
=> mm 0x100000
00100000: 8083764e ? 0
00100004: bd86200a ? 0xaabbccdd
00100008: 60a19054 ? 0x01234567
0010000c: 2c12c402 ? .
```

3.4.7. Для проведения теста RAM используется команда `mtest`:

```
mtest [start [end [pattern [iterations]]]]
```

3.4.8. Для записи разных данных несколько раз на один и тот же адрес используется команда `nm`:

```
nm [.b, .w, .l] address
```

где `[.b, .w, .l]` – разрядность изменяемой информации в битах или байтах,

`address` – адрес участка памяти.

3.4.9. Для запуска бесконечного цикла по диапазону адресов используется команда `loop`

```
loop [.b, .w, .l] address number_of_objects
```

где `[.b, .w, .l]` – разрядность информации в битах или байтах,

`address` – адрес участка памяти;

`number_of_objects` – количество объектов.

3.5 Работа с Flash памятью

3.5.1. Команда `cp` используется для копирования участков памяти. Подробнее см. п. 3.4.4.

3.5.2. Команда `flinfo` используется для вывода информации по flash-памяти.

3.5.3. Команда `erase` используется для вывода информации по flash.

3.5.4. Для удаления от одного участка памяти до конца сектора используется команда `erase start end`.

3.5.5. Для удаления всех участков памяти используется команда `erase all`.

3.5.6. Команда `protect` используется для включения или отключения защиты Flash носителя от записи. Защиту можно поставить на отдельные сектора памяти или на всю память. Подробнее см. справку загрузчика U-boot в программе.

3.6 Тестирование фиксированного блока памяти DDR встроенным тестом

3.6.1. Для тестирования произвольного блока памяти необходимо воспользоваться механизмом отображения адресов физической памяти на адресное пространство процессора. Это позволяет сделать команда `mmap`:

```
mmap set  cpu_adr { 4k | 16k | 64k | 1m | 16m | 256m | 1g } phys_adr
```

3.6.2. Команда устанавливает отображение блока физической памяти указанного размера с адресом `phys_adr` на память процессора, начиная с адреса `cpu_adr`.

3.6.3. Имеются следующие ограничения:

– максимальный размер блока – 256m (мегабайт);

– для отображения должны использоваться адреса процессора в диапазоне 0x50000000-0x5FFFFFFF;

– оба адреса должны быть выровнены по размеру блока;

– в начале физической памяти расположен код и данные самой программы, так что запуск разрушающего теста приведет к зависанию.

3.6.4. Для того чтобы снять отображение, используется команда

```
mmap drop cpu_adr
```

3.6.4.1. Например, для тестирования второго гигабайта физической памяти задаётся следующая команда:

```
mmap  set  50000000  256m 40000000
mtest  50000000  60000000  55555555  4
mmap  drop  50000000
mmap  set  50000000  256m 50000000
mtest  50000000  60000000  55555555  4
```

```
mmap  drop  50000000
mmap  set  50000000  256m 60000000
mtest  50000000  60000000  55555555  4
mmap  drop  50000000
mmap  set  50000000  256m 70000000
mtest  50000000  60000000  55555555  4
mmap  drop  50000000
```

3.6.4.2. В данном примере в пространство процессора последовательно отображаются 4 блока по 256 мегабайт из второго гигабайта физической памяти, после чего каждый раз запускается стандартный тест.

3.7 Подключение устройства памяти как USB Mass Storage

3.7.1. Команда `ums` используется для чтения устройства памяти как USB Mass Storage. Например,

```
ums 0 mmc 0
```

где `ums 0` – выбор контроллера USB, а `mmc 0` – выбор устройства памяти

3.7.2. После запуска команды `ums`, можно использовать функции USB Mass Storage для загрузки и записи новых образов.

4. СООБЩЕНИЯ ОПЕРАТОРУ

4.1 В процессе своей работы Загрузчик U-Boot формирует лишь один типа сообщений – информационные сообщения.

4.2 Информационные сообщения выводятся в строке после запуска команды, оповещая о статусе выполнения команды или полученных результатах.

ПРИЛОЖЕНИЕ А. СПИСОК ДОСТУПНЫХ КОМАНД ЗАГРУЗЧИКА U-BOOT

Команда	Описание
`base`	установка смещения для команд обращения к памяти
`bdinfo`	печать информации о модуле
`boot`	выполнение команды загрузки
`bootd`	выполнение команды загрузки по умолчанию
`bootelf`	загрузка образа ELF из памяти
`bootm`	загрузка образа приложения из памяти
`bootp`	загрузка образа по сети при помощи BOOTP-протокола
`bootvx`	загрузка образа приложения VxWorks
`cmp`	сравнение содержимого памяти
`coninfo`	печать информации о консольных устройствах
`cp`	копирование содержимого памяти
`crc32`	вычисление контрольной суммы
`dhcp`	загрузка образа по сети при помощи DHCP-протокола
`dm`	печать информации о драйверах устройств
`echo`	печать аргументов
`editenv`	редактирование переменных окружения
`env`	управление переменными окружения

Команда	Описание
`erase`	удаление данных с Flash-памяти
`exit`	завершить выполнение сценария
`ext4load`	загрузка бинарного файла из файловой системы EXT4
`ext4ls`	вывести список всех файлов в папке из файловой системы EXT4
`ext4size`	задать размер файла из файловой системы EXT4
`false`	ничего не выполняет, неуспешное выполнение
`fatinfo`	печать информации о файловой системе
`fatload`	загрузка бинарного файла из файловой системы DOS
`fatls`	вывести список всех файлов в папке из файловой системы DOS
`fatsize`	задать размер файла из файловой системы DOS
`fdt`	управление Flattened Device Tree (FDT)
`flinfo`	печать информации по Flash памяти
`fstype`	просмотреть тип файловой системы
`go`	запуск приложения по указанному адресу
`gpio`	запрос и управление интерфейса GPIO
`help`	печать справки и полного списка команд монитора
`iminfo`	печать информации об образе приложения

Команда	Описание
imxtract	извлечь образ из файла с несколькими образами
itest	вернуть значение true/false по целому числу
load	загрузка файла из файловой системы
loadb	загрузка файла через терминал по протоколу Kermit
loads	загрузка файла в формате S-Record через терминал
loadx	загрузка файла через терминал по протоколу XMODEM
loady	загрузка файла через терминал по протоколу YMODEM
loop	бесконечный цикл по диапазону адресов
ls	вывести список файлов в папке
md	отображение содержимого памяти
mm	изменение содержимого памяти с автоматическим увеличением адреса
mmap	маппинг физической памяти
mmc	функции для работы с подсистемой MMC
mmcinfo	отображение информации о MMC
mtest	простой тест RAM памяти на чтение/запись
mw	заполнение памяти
nfs	загрузка образа по сети при помощи NFS-протокола
nm	изменение памяти (постоянный адрес)

Команда	Описание
part	команды для разбиения диска;
ping	отправка запроса ICMP ECHO_REQUEST на хост-сервер в сети
printenv	печать переменных окружения
protect	включить/отключить защиту от записи для Flash-памяти
reginfo	печать регистрационных данных
reset	перезагрузка процессора
run	выполнение команд из указанной переменной окружения
save	сохранение файла в файловой системе
saveenv	сохранение переменных окружения
setenv	установка переменных окружения
setexpr	установка переменных окружения как результат оценочного выражения
showvar	печать локальных переменных hushshell
size	задать размер файла
sleep	отложить выполнение команды на какое-то время
source	Выполнить сценарий из памяти
test	выполнение минимального теста
tftpboot	загрузка образа по сети при помощи протокола TFTP

Команда	Описание
`true`	ничего не выполняет, успешное выполнение
`ums`	подключение функций USB Mass Storage
`version`	печать версий монитора, компилятора и компоновщика

ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ

ASCII – American Standard Code for Information Interchange (Американский стандартный код для обмена информацией)

BOOTP – Bootstrap protocol (Сетевой протокол, используемый для автоматического получения клиентом IP-адреса)

CRC – Cyclic Redundancy Check (Циклический избыточный код)

DDR – Double Data Rate (Удвоенная скорость передачи данных)

DHCP – Dynamic Host Configuration Protocol (Протокол динамической настройки узла)

DOS – Disk Operating System (Дисковая операционная система)

EDCL – Ethernet Debug Communication Link (Отладочная коммуникационная ссылка Ethernet)

ELF – Executable and Linkable Format (Формат исполнимых и компонуемых файлов)

EEPROM – Electrically Erasable Programmable Read-Only Memory (Электрически стираемое перепрограммируемое ПЗУ)

EXT – Extended File System (Расширенная файловая система)

FAT – File Allocation Table (Таблица размещения файлов)

FDT – Flattened Device Tree (Формат файла с упрощенным описание дерева устройств)

GPIO – General-Purpose Input/Output (Интерфейс ввода/вывода общего назначения)

IP – Internet Protocol (Межсетевой протокол)

MMC – MultiMedia Card (Портативная флеш-карта памяти, использующаяся для многократной записи и хранения информации в портативных электронных устройствах)

NFS – Network File System (ПUротокол сетевого доступа к файловым системам)

RAM – Random Access Memory (Память с произвольной выборкой)

SD – Secure Digital (Формат карт памяти для использования в портативных устройствах)

SDIO – Secure Digital Input Output (Стандарт, поддержка которого позволяет использовать со слотом расширения формата SD/MMS соответствующую периферию)

SPI – Serial Peripheral Interface (Последовательный периферийный интерфейс)

TFTP – Trivial File Transfer Protocol (Простой протокол передачи файлов)

UART – Universal Asynchronous Receiver-Transmitter (Универсальный асинхронный приёмопередатчик)

USB – Universal Serial Bus (Универсальная последовательная шина)

НТЦ – научно-технический центр

ППО – пакет программного обеспечения

Лист регистрации изменений

Номера листов (страниц)					Всего листов (страниц) в докум.	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
Изм.	измененных	замененных	новых	аннулированных					