Встроенный начальный загрузчик СнК «DragonFly»

Текон Микропроцессорные Технологии[1]

версия 0.2.03 (09.11.2023)

[1]Дунаев Д.М. <dunaev@tecon.ru>, Гурин К.Л. <gurin@tecon.ru>

Оглавление

1 Принятые сокращения и определения 2

2 Общие сведения 4

3 Назначение начального загрузчика 5

4 Режим интерактивного монитора начального загрузчика 6
- 4.1 Подготовка системы к работе . . . 6
- 4.2 Основные команды монитора . . . 6
  - 4.2.1 help . . . 6
  - 4.2.2 version . . . 7
  - 4.2.3 status . . . 7
  - 4.2.4 info . . . 7
  - 4.2.5 core . . . 10
  - 4.2.6 mfcsr . . . 11
  - 4.2.7 mtcsr . . . 12
  - 4.2.8 efuse . . . 13
  - 4.2.9 gpio . . . 14
  - 4.2.10 pad . . . 16
  - 4.2.11 mdw, mdb, mmw, mmb . . . 18
  - 4.2.12 mpu . . . 19
  - 4.2.13 uart . . . 21
  - 4.2.14 spi . . . 21
  - 4.2.15 xload . . . 23
  - 4.2.16 reset . . . 23
  - 4.2.17 go . . . 23
  - 4.2.18 call . . . 24
  - 4.2.19 hash . . . 25
  - 4.2.20 ptrace . . . 25
  - 4.2.21 sleep . . . 26
  - 4.2.22 halt . . . 26

5 Диагностическая система PinTrace 27

6 Перечень кодов и сообщений системы PinTrace 28

Глава 1

Принятые сокращения и определения

Консоль (англ. console - пульт управления) Устройство ввода-вывода, обеспечивающее связь оператора с микропроцессором.

ОЗУ Оперативное запоминающее устройство.

ПЗУ Постоянное запоминающее устройство.

ПО Программное обеспечение. Совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ СнК Система на кристалле (англ. System-on-a-Chip, SoC).

CPU Центральный процессор (англ. Central Processing Unit). Процессор, выполняющий в данной вычислительной машине или системе обработки информации основные функции по обработке информации и управлению работой других частей вычислительной машины или системы.

CRC Циклический избыточный код (англ. Cyclic Redundancy Check) — алгоритм нахождения контрольной суммы.

CRC32 Алгоритм CRC с 32-битной контрольной суммой.

CRC16 Алгоритм CRC с 16-битной контрольной суммой.

CSR Регистры контроля и управления (англ. Control and Status Registers).

EFUSE Однократно программируемая электрически независимая память.

FLASH Тип полупроводниковой электрически перепрограммируемой памяти.

GPIO Интерфейс ввода/вывода общего назначения (англ. General Purpose Input/Output).

GRU Блок общего сброса (англ. Global Reset Unit)

ISA Архитектура системы команд (англ. Instruction Set Architecture).

JTAG Отладочный тестовый интерфейс (от англ. Joint Test Action Group - рабочая группа по разработке стандарта IEEE 1149).

LCRU Блок логического контроля и сброса (англ. Logical Control and Reset Unit). MD5 Алгоритм подсчета цифровой подписи.

NSC Контроллер параллельной памяти NOR/SRAM (англ. NOR/SRAM controller).

OCD Отладочный интерфейс СнК (англ. On Chip Debugger).

PAD Управляемый мультиплексированный вывод СнК.

RAM Запоминающее устройство с произвольной выборкой (анг. Random Access Memories).

RISC Компьютер с сокращённым набором команд (англ. Restricted Instruction Set Computer).

ROM Постоянное запоминающее устройство,

ПЗУ (англ. Read-Only Memory).

RTC Часы реального времени (англ. Real Time Clock).

RTL Уровень регистровых передач (англ. Register Transfer Level).

SoC Система на кристалле (англ. System on Chip).

SPI Последовательный периферийный интерфейс (англ. Serial Peripheral Interface).

SRAM Статическая оперативная память с произвольным доступом (англ. Static Random Access Memory).

SREC Формат текстового ASCII файла для хранения двоичных данных (S-RECORD).

SSI Синхронный последовательный интерфейс (англ. Synchronous Serial Interface).

UART Универсальный асинхронный приемопередатчик (англ. Universal Asynchronous Receiver-Transmitter).

X-MODEM Протокол пакетной передачи данных по последовательному интерфейсу с контролем ошибок.

Глава 2

Общие сведения

Название продукта: “Начальный загрузчик DragonBoot”.

Наименование: файл образа ПЗУ интегральной схемы drboot.bin.

Внутреннее имя: "drboot".

Архитектура: RISC-V, спецификация RV32IMFDC, ISA v.1.10.

Разрядность: 32 бита. Версия продукта: 1.06.

Язык интерфейса: English (Unated States).

Язык исходного кода: Си (стандарт C99), Ассемблер RISC-V.

Версия и тип компилятора: riscv32-unknown-elf-gcc (GCC) 10.1.0.

Версия и тип программы линковки: GNU ld (GNU Binutils) 2.35.

Размер бинарного образа: 56 КБ.

Минимальное количество оперативной памяти: 16 КБ.

Наличие диагностических интерфейсов:

- Последовательная консоль (Boot UART).
- Система PinTrace.

Глава 3

Назначение начального загрузчика

Начальный загрузчик DragonBoot (SoC "DragonFly" Bootloader) представляет собой загрузчик первого уровня, расположенный в read-only ПЗУ микроконтроллера (ROM). Основными функциями начального загрузчика являются:

- Начальная инициализация микроконтроллера.
- Базовое тестирование интерфейсов.
- Взаимодействие с процессорными ядрами СнК. Межпроцессорное взаимодействие.
- Базовые функции отладочного монитора (чтение и запись системных регистров, регистров периферийных устройств и области памяти).
- Взаимодействие с рядом периферийных устройств (SPI, SSI, GPIO, PAD и др.) на уровне команд.
- Загрузка отладочного ПО через последовательный интерфейс, передача управления стороннему ПО.
- Обработка ошибок и исключений в режиме информирования оператора.
- Перезагрузка системы через блок общего сброса.

Глава 4

Режим интерактивного монитора начального загрузчика

После начальной инициализации системы начальный загрузчик переходит в режим интерактивного монитора и ожидает команд оператора.

До момента переопределения адреса векторов обработки исключений, начальный загрузчик осуществляет их захват и обработку в режиме информирования оператора. В случае возникновения исключения, начальный загрузчик выводит информацию об исключении, состояние контекста вычислительного ядра на момент возникновения исключения. После этого, загрузчик возвращается в режим интерактивного монитора.

4.1 Подготовка системы к работе

1. Перевести СнК в режим загрузки с внутреннего ROM.
2. Подключить компьютер оператора к последовательному порту BOOT.
3. Запустить на компьютер оператора терминальную программу с подключением к последовательному порту СнК в режиме 9600 8N1.
4. Подать питание на СнК.
5. По окончанию перечисленных действий, система выведет сообщение с названием начального загрузчика, его версию и приглашение командной строки ('tmt >'):

```
DragonBoot ROM monitor. Version 1.05.DragonFly (Nov 10 2020 - 17:55:37 +0300)
Copyright (C) 2016-2020 Tecon MT LLC. All rights reserved.
Type 'help' for help.
tmt >
```

6. Если после выполения действий пп.1-4 не выполнился п.5, то необходимо ввести пароль для доступа в интерактивный монитор. Пароль вводится последовательным набором символов с сохранением регистра, при этом вводимые символы на экране не отображаются. Нажатие клавиш Enter, <Ctrl+C> сбрасывают уже набранные символы. В текущей версии начального загрузчика режим ограничения доступа в интерактивный монитор отсутствует.

4.2 Основные команды монитора

Список всех реализуемых команд монитора выводится по команде “help”.

4.2.1 help

Команда «help» выводит информацию о доступных командах.

Пример исполнения:

```
tmt > help
Aviable commands:
  help, ?  Show this help
  version  Show version
  status   Show decoded mstatus register
  info     Show system information
  core     Core(s) control and status
  mfcsr    Dump CSR
  mtcsr    Set CSR value
  mdw      Dump memory in words
  mdb      Dump memory in bytes
  mmw      Set memory data in word
  mmb      Set memory data in byte
  gpio     Control GPIO lines
  efuse    Dump EFuse data
  mpu      Dump current MPU entries status
  pad      Control and status of specified PAD
  spi      Load data in SRAM from SPI
  ssi      Load data in SRAM from SSI
  uart     Set console UART
  xload    Load data via X-MODEM
  reset    Perform reset
  go       Goto to address
  call     Call to address
  hash     Calculate hash of memory region
  ptrace   Pintrace operations
  halt     Halt the system
  sleep    Delay for selected count of seconds
OK
```

4.2.2 version

Команда «version» выводит краткую информацию о начальном загрузчике, его версии, версии компилятора и программы линковки:

Пример исполнения:

```
tmt > version
DragonBoot ROM monitor. Version 1.05.DragonFly (Nov 10 2020 - 17:55:37 +0300)
riscv32-unknown-elf-gcc (sdk-mt1 build 20200910 commit: c1c264b) 9.2.0
GNU ld (GNU Binutils) 2.34 Memory Map v0.0.6 / 23.09.2020 / db5fa10
Commit a3b4013 / v1.05 (Gummy Bear)
OK
```

4.2.3 status

Команда «status» выводит дамп текущего состояния регистра `mstatus` для ядра, на котором производится выполнение начального загрузчика:

Пример исполнения:

```
tmt > status
Status: 00003800
  MIE:  0
  MPIE: 0
  MPP:  M
  MPRV: 0
  FS:   INITIAL
  XS:   OFF
  SD:   0
OK
```

4.2.4 info

Команда «info» выводит подробную техническую информацию о вычислительных ядрах системы и об их версии, о статической и быстрой памяти, о состоянии системных регистров ядра (на котором производится выполнение начального загрузчика).

Пример исполнения:

```
tmt > info
Technical information.
SoC name:  DragonFly (rev.301)
Num cores: 2 (max 2)
HART0: Syntacore SCR4.L1[20031141] RV32IMFDCXU [80001603] @ 25MHz
L1i cache: 32K, 4-way, 16-byte line, 512-index, read-only
L1d cache: 32K, 4-way, 16-byte line, 512-index, write-through, no-allocate
L2  cache: n/a
RTL release: 11.08.2020 (rel.41)
HART1: Disabled
CPU freq:   25MHz
RTC freq:   25MHz
BUS freq:   25MHz
Total SRAM: 00800000-00980000 (1536 KB)
Free  SRAM: 00800000-0097c000 (1520 KB)
Area  MMIO: 00000000-20000000 (512 MB)
Peripheral: 00000000-00185000 (1556 KB)
Boot   ROM: 00a00000-00a20000 (128 KB)
Filled ROM: 00a00000-00a0fa3c (62 KB)
Console:
   [0] serial@00100000 inactive
   [1] serial@00101000 inactive
   [2] serial@00102000 inactive
   [3] serial@00103000 inactive
   [4] serial@00104000 inactive
   [5] serial@00105000 inactive
   [6] serial@00106000 inactive
   [7] serial@00107000 inactive
 * [8] serial@00002000 active fifo 64 mode 9600 8N1 (boot)
Current status:
Status: 00003800
  MIE:  0
  MPIE: 0
  MPP:  M
  MPRV: 0
  FS:   INITIAL
  XS:   OFF
  SD:   0
SPI:
 * spi@00001000 flash: 1f8601ff sector:4096 page:256
PAD:
PAD (000) [0004]: 0000  MUX:0 DS:0 --- --- -- --
PAD (001) [0008]: 0000  MUX:0 DS:0 --- --- -- --
PAD (002) [000c]: 0000  MUX:0 DS:0 --- --- -- --
PAD (003) [0010]: 0000  MUX:0 DS:0 --- --- -- --
PAD (004) [0014]: 0000  MUX:0 DS:0 --- --- -- --
PAD (005) [0018]: 0000  MUX:0 DS:0 --- --- -- --
PAD (006) [001c]: 0000  MUX:0 DS:0 --- --- -- --
PAD (007) [0020]: 0000  MUX:0 DS:0 --- --- -- --
PAD (008) [0024]: 0000  MUX:0 DS:0 --- --- -- --
PAD (009) [0028]: 0000  MUX:0 DS:0 --- --- -- --
PAD (010) [002c]: 0000  MUX:0 DS:0 --- --- -- --
PAD (011) [0030]: 0000  MUX:0 DS:0 --- --- -- --
PAD (012) [0034]: 0000  MUX:0 DS:0 --- --- -- --
PAD (013) [0038]: 0000  MUX:0 DS:0 --- --- -- --
PAD (014) [003c]: 0000  MUX:0 DS:0 --- --- -- --
PAD (015) [0040]: 0000  MUX:0 DS:0 --- --- -- --
PAD (016) [0044]: 0000  MUX:0 DS:0 --- --- -- --
PAD (017) [0048]: 0000  MUX:0 DS:0 --- --- -- --
PAD (018) [004c]: 0000  MUX:0 DS:0 --- --- -- --
PAD (019) [0050]: 0000  MUX:0 DS:0 --- --- -- --
PAD (020) [0054]: 0000  MUX:0 DS:0 --- --- -- --
PAD (021) [0058]: 0000  MUX:0 DS:0 --- --- -- --
PAD (022) [005c]: 0000  MUX:0 DS:0 --- --- -- --
PAD (023) [0060]: 0000  MUX:0 DS:0 --- --- -- --
PAD (024) [0064]: 0000  MUX:0 DS:0 --- --- -- --
PAD (025) [0068]: 0000  MUX:0 DS:0 --- --- -- --
PAD (026) [006c]: 0000  MUX:0 DS:0 --- --- -- --
PAD (027) [0070]: 0000  MUX:0 DS:0 --- --- -- --
PAD (028) [0074]: 0000  MUX:0 DS:0 --- --- -- --
PAD (029) [0078]: 0000  MUX:0 DS:0 --- --- -- --
PAD (030) [007c]: 0000  MUX:0 DS:0 --- --- -- --
PAD (031) [0080]: 0000  MUX:0 DS:0 --- --- -- --
PAD (032) [0084]: 0000  MUX:0 DS:0 --- --- -- --
PAD (033) [0088]: 0000  MUX:0 DS:0 --- --- -- --
PAD (034) [008c]: 0000  MUX:0 DS:0 --- --- -- --
```

```
PAD (035) [0090]: 0000  MUX:0 DS:0 --- --- -- --
PAD (036) [0094]: 0000  MUX:0 DS:0 --- --- -- --
PAD (037) [0098]: 0000  MUX:0 DS:0 --- --- -- --
PAD (038) [009c]: 0000  MUX:0 DS:0 --- --- -- --
PAD (039) [00a0]: 0000  MUX:0 DS:0 --- --- -- --
PAD (040) [00a4]: 0000  MUX:0 DS:0 --- --- -- --
PAD (041) [00a8]: 0000  MUX:0 DS:0 --- --- -- --
PAD (042) [00ac]: 0000  MUX:0 DS:0 --- --- -- --
PAD (043) [00b0]: 0000  MUX:0 DS:0 --- --- -- --
PAD (044) [00b4]: 0000  MUX:0 DS:0 --- --- -- --
PAD (045) [00b8]: 0000  MUX:0 DS:0 --- --- -- --
PAD (046) [00bc]: 0000  MUX:0 DS:0 --- --- -- --
PAD (047) [00c0]: 0000  MUX:0 DS:0 --- --- -- --
PAD (048) [00c4]: 0000  MUX:0 DS:0 --- --- -- --
PAD (049) [00c8]: 0000  MUX:0 DS:0 --- --- -- --
PAD (050) [00cc]: 0000  MUX:0 DS:0 --- --- -- --
PAD (051) [00d0]: 0000  MUX:0 DS:0 --- --- -- --
PAD (052) [00d4]: 0000  MUX:0 DS:0 --- --- -- --
PAD (053) [00d8]: 0000  MUX:0 DS:0 --- --- -- --
PAD (054) [00dc]: 0000  MUX:0 DS:0 --- --- -- --
PAD (055) [00e0]: 0000  MUX:0 DS:0 --- --- -- --
PAD (056) [00e4]: 0000  MUX:0 DS:0 --- --- -- --
PAD (057) [00e8]: 0000  MUX:0 DS:0 --- --- -- --
PAD (058) [00ec]: 0000  MUX:0 DS:0 --- --- -- --
PAD (059) [00f0]: 0000  MUX:0 DS:0 --- --- -- --
PAD (060) [00f4]: 0000  MUX:0 DS:0 --- --- -- --
PAD (061) [00f8]: 0000  MUX:0 DS:0 --- --- -- --
PAD (062) [00fc]: 0000  MUX:0 DS:0 --- --- -- --
PAD (063) [0100]: 0000  MUX:0 DS:0 --- --- -- --
PAD (064) [0104]: 0000  MUX:0 DS:0 --- --- -- --
PAD (065) [0108]: 0000  MUX:0 DS:0 --- --- -- --
PAD (066) [010c]: 0000  MUX:0 DS:0 --- --- -- --
GPIO:
   [0] gpio@00122000 inactive
   [1] gpio@00180000 inactive
PinTrace:
Hardware CRC accelerator @00003000
HART0 CSR:
[0001] fflags:       00000000    RW- SCR4 SCR5
[0002] frm:          00000000    RW- SCR4 SCR5
[0003] fcsr:         00000000    RW- SCR4 SCR5
[0c00] cycle:        02174735    R-- SCR4 SCR5
[0c80] cycleh:       00000001    R-- SCR4 SCR5
[0c01] time:         0a52d58b    R-- SCR4 SCR5
[0c81] timeh:        00000000    R-- SCR4 SCR5
[0c02] instret:      02176b00    R-- SCR4 SCR5
[0c82] instreth:     00000001    R-- SCR4 SCR5
[0c03] hpmcnt0:      00000000    R-- SCR4 SCR5
[0c83] hpmcnth0:     00000000    R-- SCR4 SCR5
[0c04] hpmcnt1:      00000000    R-- SCR4 SCR5
[0c84] hpmcnth1:     00000000    R-- SCR4 SCR5
[0c05] hpmcnt2:      00000000    R-- SCR4 SCR5
[0c85] hpmcnth2:     00000000    R-- SCR4 SCR5
[0c06] hpmcnt3:      00000000    R-- SCR4 SCR5
[0c86] hpmcnth3:     00000000    R-- SCR4 SCR5
[0300] mstatus:      00003800    RWX SCR4 SCR5
[0301] misa:         4090112c    R-- SCR4 SCR5
[0304] mie:          00000000    RWX SCR4 SCR5
[0305] mtvec:        00a0034c    RWX SCR4 SCR5
[0340] mscratch:     00000000    RW- SCR4 SCR5
[0341] mepc:         00000000    RW- SCR4 SCR5
[0342] mcause:       00000000    R-- SCR4 SCR5
[0343] mtval:        00000000    RW- SCR4 SCR5
[0344] mip:          00000000    RW- SCR4 SCR5
[0306] mcounteren:   00000000    RWX SCR4 SCR5
[0320] mcountinhib:  00000000    RWX SCR4 SCR5
[0323] mhpmevent0:   00000000    R-- SCR4 SCR5
[0324] mhpmevent1:   00000000    R-- SCR4 SCR5
[0325] mhpmevent2:   00000000    R-- SCR4 SCR5
[0326] mhpmevent3:   00000000    R-- SCR4 SCR5
[0b00] mcycle:       02184506    RW- SCR4 SCR5
[0b80] mcycleh:      00000001    RW- SCR4 SCR5
[0b02] minstret:     0218570f    RW- SCR4 SCR5
[0b82] minstreth:    00000001    RW- SCR4 SCR5
[0f11] mvendorid:    00000000    R-- SCR4 SCR5
[0f12] marchid:      80001603    R-- SCR4 SCR5
```

```
[0f13] mimpid:       20031141    R-- SCR4 SCR5
[0f14] mhartid:      00000000    R-- SCR4 SCR5
[0bf0] mipic_cisv:   00000020    R-X SCR4 SCR5
[0bf1] mipic_cicsr:  00000000    RWX SCR4 SCR5
[0bf2] mipic_ipr:    00000000    RWX SCR4 SCR5
[0bf3] mipic_isvr:   00000000    R-X SCR4 SCR5
[0bf4] mipic_eoi:    00000000    -WX SCR4 SCR5
[0bf5] mipic_soi:    00000000    -WX SCR4 SCR5
[0bf6] mipic_idx:    00000000    RWX SCR4 SCR5
[0bf7] mipic_icsr:   00020300    RWX SCR4 SCR5
[0bf8] mipic_ier:    00000000    RWX SCR4 SCR5
[0bf9] mipic_imap:   ffffffff    R-- SCR4 SCR5
[0bd4] mem_gbl:      00000000    RW- SCR4 SCR5
[0bdc] axi_lpi:      00000000    RW- SCR4 SCR5
[0fc1] mefaddr:      00000000    R-- SCR4 SCR5
[0fc2] mefcmd:       00000000    R-- SCR4 SCR5
[0fc3] cache_l1:     29420942    R-- SCR4 SCR5
OK
```

4.2.5 core

Команда «core» вызов команды без параметров выводит краткий список вычислительных ядер системы с обозначением активного ядра (символ '*').

Ядра, находящиеся в неактивном состоянии обозначаются как `Disabled`.

Пример исполнения:

```
tmt > core
 * HART0: Syntacore SCR4.L1 RV32IMFDCXU
 - HART1: Disabled
```

Активации неактивного ядра осуществляется командой:

```
core X start
```

где X - номер ядра, которое будет активировано. Если ядра с заданным номером в системе нет, будет выдано сообщение об ошибке.

Пример исполнения:

```
tmt > core 1 start
OK
tmt > core
 * HART0: Syntacore SCR4.L1 RV32IMFDCXU
   HART1: Syntacore SCR4.L1 RV32IMFDCXU
```

Деактивация активного ядра осуществляется командой:

```
core X stop
```

где X - номер ядра, которое будет деактивировано. Если ядра с заданным номером в системе нет, будет выдано сообщение об ошибке.

Пример исполнения:

```
tmt > core 1 stop
OK
tmt > core
 * HART0: Syntacore SCR4.L1 RV32IMFDCXU
 - HART1: Disabled
```

Деактивировать ядро, на котором происходит текущее исполнение кода, невозможно. Для СнК "Стрекоза" возможно отключение только ядра 1. Ядро 0 не имеет механизмов отключения.

Перенос исполнения кода интерактивного монитора на одно из доступных ядер системы осуществляется командой:

```
core X
```

где X - номер ядра, которому предписывается исполнять код интерактивного монитора. Если ядра с заданным номером в системе нет, будет выдано сообщение об ошибке.

Пример исполнения:

```
tmt > core 1 start
OK
tmt > core 1
Now running on HART1 (Syntacore SCR4.L1)
Type 'help' for help.
tmt > core 4
HART4 doesn't exist in system
ERROR: Command 'core' failed (-22)
```

Ядро, ранее исполнявшее код интерактивного монитора, переводится в режим ожидания (idle). Указатель стека при этом сбрасывается в исходное состояние.

4.2.6 mfcsr

Команда «mfcsr» осуществляет дамп системных регистров CSR ядра, на котором происходит исполнение начального загрузчика.

Вызов определенного регистра осуществляется в соответствии с его семантическим именем по RISC-V ISA. Для просмотра дампа регистров CSR другого ядра необходимо перенести исполнение начального загрузчика на данное ядро командой "core".

Формат вызова команды:

```
mfcsr <название регистра CSR>
```

Пример исполнения:

```
tmt > mfcsr mscratch
[0340] mscratch:     00000000    RW- SCR4 SCR5
```

Формат вывода команды:

```
[id] name: value param
```

где

id — идентификатор системного регистра.

name — семантическое имя согласно RISC-V ISA.

value — текущее значение.

param — параметры регистра, состоящие из 3 групп.

Параметры регистра:

Группа 1

- RWX — параметры чтения/записи регистра.
- R — доступно чтение.
- W — доступна запись.
- X — значение регистра влияет на текущее поведение системы.

Группа 2 SCR4 — регистр доступен для ядер RISC-V SCR4.

Группа 3 SCR5 — регистр доступен для ядер RISC-V SCR5.

При попытке вывести дамп несуществующего регистра система выдаст сообщение об ошибке:

```
tmt > mfcsr stvec
Unknown CSR name 'stvec'
ERROR: Command 'mfcsr' failed (-22)
```

Вызов команды без параметров выводит список доступных регистров системы с текущими значениями и параметрами.

Пример исполнения:

```
tmt > mfcsr
[0001] fflags:       00000000    RW- SCR4 SCR5
[0002] frm:          00000000    RW- SCR4 SCR5
[0003] fcsr:         00000000    RW- SCR4 SCR5
[0c00] cycle:        5f7aed80    R-- SCR4 SCR5
[0c80] cycleh:       00000001    R-- SCR4 SCR5
[0c01] time:         0e0f23e1    R-- SCR4 SCR5
[0c81] timeh:        00000000    R-- SCR4 SCR5
[0c02] instret:      5f7b1163    R-- SCR4 SCR5
[0c82] instreth:     00000001    R-- SCR4 SCR5
[0c03] hpmcnt0:      00000000    R-- SCR4 SCR5
[0c83] hpmcnth0:     00000000    R-- SCR4 SCR5
[0c04] hpmcnt1:      00000000    R-- SCR4 SCR5
[0c84] hpmcnth1:     00000000    R-- SCR4 SCR5
[0c05] hpmcnt2:      00000000    R-- SCR4 SCR5
[0c85] hpmcnth2:     00000000    R-- SCR4 SCR5
[0c06] hpmcnt3:      00000000    R-- SCR4 SCR5
[0c86] hpmcnth3:     00000000    R-- SCR4 SCR5
[0300] mstatus:      00003800    RWX SCR4 SCR5
[0301] misa:         4090112c    R-- SCR4 SCR5
[0304] mie:          00000000    RWX SCR4 SCR5
[0305] mtvec:        00a0034c    RWX SCR4 SCR5
[0340] mscratch:     00000000    RW- SCR4 SCR5
[0341] mepc:         00000000    RW- SCR4 SCR5
[0342] mcause:       00000000    R-- SCR4 SCR5
[0343] mtval:        00000000    RW- SCR4 SCR5
[0344] mip:          00000000    RW- SCR4 SCR5
[0306] mcounteren:   00000000    RWX SCR4 SCR5
[0320] mcountinhib:  00000000    RWX SCR4 SCR5
[0323] mhpmevent0:   00000000    R-- SCR4 SCR5
[0324] mhpmevent1:   00000000    R-- SCR4 SCR5
[0325] mhpmevent2:   00000000    R-- SCR4 SCR5
[0326] mhpmevent3:   00000000    R-- SCR4 SCR5
[0b00] mcycle:       5f7beb81    RW- SCR4 SCR5
[0b80] mcycleh:      00000001    RW- SCR4 SCR5
[0b02] minstret:     5f7bfda2    RW- SCR4 SCR5
[0b82] minstreth:    00000001    RW- SCR4 SCR5
[0f11] mvendorid:    00000000    R-- SCR4 SCR5
[0f12] marchid:      80001603    R-- SCR4 SCR5
[0f13] mimpid:       20031141    R-- SCR4 SCR5
[0f14] mhartid:      00000000    R-- SCR4 SCR5
[0bf0] mipic_cisv:   00000020    R-X SCR4 SCR5
[0bf1] mipic_cicsr:  00000000    RWX SCR4 SCR5
[0bf2] mipic_ipr:    00000000    RWX SCR4 SCR5
[0bf3] mipic_isvr:   00000000    R-X SCR4 SCR5
[0bf4] mipic_eoi:    00000000    -WX SCR4 SCR5
[0bf5] mipic_soi:    00000000    -WX SCR4 SCR5
[0bf6] mipic_idx:    00000000    RWX SCR4 SCR5
[0bf7] mipic_icsr:   00020300    RWX SCR4 SCR5
[0bf8] mipic_ier:    00000000    RWX SCR4 SCR5
[0bf9] mipic_imap:   ffffffff    R-- SCR4 SCR5
[0bd4] mem_gbl:      00000000    RW- SCR4 SCR5
[0bdc] axi_lpi:      00000000    RW- SCR4 SCR5
[0fc1] mefaddr:      00000000    R-- SCR4 SCR5
[0fc2] mefcmd:       00000000    R-- SCR4 SCR5
[0fc3] cache_l1:     29420942    R-- SCR4 SCR5
OK
```

4.2.7 mtcsr

Команда «mtcsr» осуществляет запись в системные регистры CSR ядра, на котором происходит исполнение начального загрузчика.

Формат вызова команды:

```
mtcsr <название регистра CSR> <значение(hex)>
```

Пример исполнения:

```
tmt > mtcsr mscratch 0x100
[0340] mscratch:     00000100    RW- SCR4 SCR5
```

Формат вывода команды аналогичен формату вывода команды “mfcsr”.

При попытке изменить значение несуществующего регистра система выдаст сообщение об ошибке:

```
tmt > mtcsr stvec 0x100
Unknown CSR name 'stvec'
ERROR: Command 'mtcsr' failed (-2)
```

4.2.8 efuse

Команда «efuse» позволяет выводить информацию однократно программируемой энергонезависимой памяти EFUSE.

При вызове команды без параметров на консоль выводится содержимое банков EFUSE в виде байтового дампа.

Пример исполнения:

```
tmt > efuse
efuse@00006000:
0000:  fe  01  00  00  0f  01  00  a2  dc  e3  f0  00  01  02  03  00
0010:  02  02  03  00  03  02  03  00  04  02  03  00  05  02  03  00
0020:  06  02  03  00  00  00  00  00  00  00  00  00  00  00  00  00
0030:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0040:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0050:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0060:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0070:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
efuse@00007000:
0000:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0010:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0020:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0030:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0040:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0050:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0060:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0070:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
OK
```

Для получения расшифрованной информации о чипе, записанной в банк 0 EFUSE, используется команда:

```
efuse dump
```

Пример исполнения:

```
tmt > efuse dump
Chip: 10
Rev:  2
S/N:  271
OID:  f0e3dc
MAC00 f0:e3:dc:03:02:01
MAC01 f0:e3:dc:03:02:02
MAC02 f0:e3:dc:03:02:03
MAC03 f0:e3:dc:03:02:04
MAC04 f0:e3:dc:03:02:05
MAC05 f0:e3:dc:03:02:06
OK
```

Для получения байтового дампа отдельного банка EFUSE используется команда:

```
efuse bank X
```

где X - номер банка EFuse

Пример исполнения:

```
tmt > efuse bank 0
efuse@00006000:
0000:  fe  01  00  00  0f  01  00  a2  dc  e3  f0  00  01  02  03  00
0010:  02  02  03  00  03  02  03  00  04  02  03  00  05  02  03  00
0020:  06  02  03  00  00  00  00  00  00  00  00  00  00  00  00  00
0030:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0040:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0050:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0060:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0070:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
OK
```

Вызов команды с несуществующем в системе банком EFUSE вернет ошибку:

```
tmt > efuse bank 3
ERROR: Command 'efuse' failed (-19)
```

Для получения информации о состоянии банка EFUSE используется команда:

```
efuse status X
```

где X - номер банка EFuse
Пример исполнения:

```
tmt > efuse status 0
efuse@00006000:
      00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
00:      ++ ++ ++ ++ ++ ++ ++ ++ -- -- -- -- -- -- -- --
10:   -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
OK
```

Отметка '++' соответствует записанным ячейкам, '--' — свободным ячейкам. Ячейка со смещением 0 зарезервирована и содержит текущий статус банка EFUSE.

4.2.9 gpio

Команда «gpio» позволяет управлять контроллерами GPIO.

При вызове команды без параметров на консоль выводится базовые адреса контроллеров GPIO.

```
tmt > gpio
 * [0] gpio@00122000 active
   [1] gpio@00180000 inactive
```

Для получения полной информации о текущем состоянии всех линий контроллеров GPIO используется команда:

```
gpio dump
```

Пример исполнения:

```
tmt > gpio dump
gpio@00122000:
   [00] pin:00 [0] IN
   [01] pin:01 [0] IN
   [02] pin:02 [0] IN
   [03] pin:03 [0] IN
   [04] pin:04 [0] IN
   [05] pin:05 [0] IN
   [06] pin:06 [0] IN
   [07] pin:07 [0] IN
   [08] pin:08 [0] IN
   [09] pin:09 [0] IN
   [10] pin:10 [0] IN
   [11] pin:11 [0] IN
   [12] pin:12 [0] IN
   [13] pin:13 [0] IN
   [14] pin:14 [0] IN
   [15] pin:15 [0] IN
   [16] pin:16 [0] IN
   [17] pin:17 [0] IN
   [18] pin:18 [0] IN
   [19] pin:19 [0] IN
   [20] pin:20 [0] IN
   [21] pin:21 [0] IN
   [22] pin:22 [0] IN
   [23] pin:23 [0] IN
   [24] pin:24 [0] IN
   [25] pin:25 [0] IN
 * [26] pin:26 [1] OUT
   [27] pin:27 [0] IN
   [28] pin:28 [0] IN
   [29] pin:29 [0] IN
   [30] pin:30 [0] IN
   [31] pin:31 [0] IN
gpio@00180000:
```

```
   [32] pin:00 [0] IN
   [33] pin:01 [0] IN
   [34] pin:02 [0] IN
   [35] pin:03 [0] IN
   [36] pin:04 [0] IN
   [37] pin:05 [0] IN
   [38] pin:06 [0] IN
   [39] pin:07 [0] IN
   [40] pin:08 [0] IN
   [41] pin:09 [0] IN
   [42] pin:10 [0] IN
   [43] pin:11 [0] IN
   [44] pin:12 [0] IN
   [45] pin:13 [0] IN
   [46] pin:14 [0] IN
   [47] pin:15 [0] IN
   [48] pin:16 [0] IN
   [49] pin:17 [0] IN
   [50] pin:18 [0] IN
   [51] pin:19 [0] IN
   [52] pin:20 [0] IN
   [53] pin:21 [0] IN
   [54] pin:22 [0] IN
   [55] pin:23 [0] IN
   [56] pin:24 [0] IN
   [57] pin:25 [0] IN
   [58] pin:26 [0] IN
   [59] pin:27 [0] IN
   [60] pin:28 [0] IN
   [61] pin:29 [0] IN
   [62] pin:30 [0] IN
   [63] pin:31 [0] IN
OK
```

Здесь и далее, формат вывода информации о текущем состоянии линии GPIO:

```
LINE PIN VALUE DIRECTION
```

где

LINE — десятичное число, порядковый номер линии в системе, нумерация начинается с 0.

PIN — номер вывода соответствующего контроллера GPIO.

VALUE — текущее значение уровня вывода GPIO.

DIRECTION — расшифровка текущего направления линии:

IN — линия сконфигурирована как вход.

OUT — линия сконфигурирована как выход.

OPEN — линия сконфигурирована в режиме "открытый коллектор".

Для индивидуального управления отдельной линией GPIO используются команды

```
gpio X [dir|set|get|free]
```

где

X — десятичный номер линии GPIO.

dir — выбор направления работы линии (in, out). При выборе направления разрешается работа соответствующего PAD.

set — установить выходной уровень на линии GPIO (0,1). Команда актуальна только при конфигурации линии на выход.

get — получить текущий уровень линии GPIO.

free — освободить линию GPIO, ранее сконфигурированную командой dir. PAD, соответствующий текущей линии GPIO переводится в неактивное состояние.

После исполнения команды на консоль выводится информация о новом состоянии линии GPIO в формате, описанном выше.

Пример исполнения:

```
tmt > gpio 26
   [26] pin:26 [0] IN
OK
tmt > gpio 26 dir out
 * [26] pin:26 [0] OUT
OK
tmt > gpio 26 set 1
 * [26] pin:26 [1] OUT
OK
tmt > gpio 26 get
 * [26] pin:26 [1] OUT
OK
tmt > gpio 26 free
   [26] pin:26 [1] IN
OK
```

4.2.10 pad

Команда «pad» позволяет управлять контроллером PAD кластера периферийных устройств.

При вызове команды без параметров на консоль выводится базовый адрес контроллера PAD.

```
tmt > pad
 * pad@0012e000
```

⚠ В текущей реализации начального загрузчика, из 3-х существующих в СнК "Стрекоза" контроллеров PAD, поддерживается работа только с контроллером кластера периферийных устройств с базовым адресом 0x0012E000.

Для получения полной информации о текущем состоянии всех регистров контроллера PAD используется команда:

```
pad dump
```

Пример исполнения:

```
tmt > pad dump
PAD (000) [0004]: 0000  MUX:0 DS:0 --- --- -- --
PAD (001) [0008]: 0000  MUX:0 DS:0 --- --- -- --
PAD (002) [000c]: 0000  MUX:0 DS:0 --- --- -- --
PAD (003) [0010]: 0000  MUX:0 DS:0 --- --- -- --
PAD (004) [0014]: 0000  MUX:0 DS:0 --- --- -- --
PAD (005) [0018]: 0000  MUX:0 DS:0 --- --- -- --
PAD (006) [001c]: 0000  MUX:0 DS:0 --- --- -- --
PAD (007) [0020]: 0000  MUX:0 DS:0 --- --- -- --
PAD (008) [0024]: 0000  MUX:0 DS:0 --- --- -- --
PAD (009) [0028]: 0000  MUX:0 DS:0 --- --- -- --
PAD (010) [002c]: 0000  MUX:0 DS:0 --- --- -- --
PAD (011) [0030]: 0000  MUX:0 DS:0 --- --- -- --
PAD (012) [0034]: 0000  MUX:0 DS:0 --- --- -- --
PAD (013) [0038]: 0000  MUX:0 DS:0 --- --- -- --
PAD (014) [003c]: 0000  MUX:0 DS:0 --- --- -- --
PAD (015) [0040]: 0000  MUX:0 DS:0 --- --- -- --
PAD (016) [0044]: 0000  MUX:0 DS:0 --- --- -- --
PAD (017) [0048]: 0000  MUX:0 DS:0 --- --- -- --
PAD (018) [004c]: 0000  MUX:0 DS:0 --- --- -- --
PAD (019) [0050]: 0000  MUX:0 DS:0 --- --- -- --
PAD (020) [0054]: 0000  MUX:0 DS:0 --- --- -- --
PAD (021) [0058]: 0000  MUX:0 DS:0 --- --- -- --
PAD (022) [005c]: 0000  MUX:0 DS:0 --- --- -- --
PAD (023) [0060]: 0000  MUX:0 DS:0 --- --- -- --
PAD (024) [0064]: 0000  MUX:0 DS:0 --- --- -- --
PAD (025) [0068]: 0000  MUX:0 DS:0 --- --- -- --
PAD (026) [006c]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (027) [0070]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (028) [0074]: 010f  MUX:2 DS:7 --- --- -- EN
```

```
PAD (029) [0078]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (030) [007c]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (031) [0080]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (032) [0084]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (033) [0088]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (034) [008c]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (035) [0090]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (036) [0094]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (037) [0098]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (038) [009c]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (039) [00a0]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (040) [00a4]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (041) [00a8]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (042) [00ac]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (043) [00b0]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (044) [00b4]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (045) [00b8]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (046) [00bc]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (047) [00c0]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (048) [00c4]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (049) [00c8]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (050) [00cc]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (051) [00d0]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (052) [00d4]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (053) [00d8]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (054) [00dc]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (055) [00e0]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (056) [00e4]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (057) [00e8]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (058) [00ec]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (059) [00f0]: 010f  MUX:2 DS:7 --- --- -- EN
PAD (060) [00f4]: 0000  MUX:0 DS:0 --- --- -- --
PAD (061) [00f8]: 0000  MUX:0 DS:0 --- --- -- --
PAD (062) [00fc]: 0000  MUX:0 DS:0 --- --- -- --
PAD (063) [0100]: 0000  MUX:0 DS:0 --- --- -- --
PAD (064) [0104]: 0000  MUX:0 DS:0 --- --- -- --
PAD (065) [0108]: 0000  MUX:0 DS:0 --- --- -- --
PAD (066) [010c]: 0000  MUX:0 DS:0 --- --- -- --
```

Здесь и далее, формат вывода информации о текущем состоянии PAD:

```
PAD REGISTER VALUE DECODE
```

где

PAD — десятичное число, порядковый номер PAD в контроллере, нумерация начинается с 0.

REGISTER — адрес регистра соответствующего PAD.

VALUE — текущее значение регистра.

DECODE — расшифровка текущего значения регистра:

MUX — номер функции мультиплексирования.

DS — конфигурация управления фронтом сигнала.

PUD — флаг направления работы подтягивающего регистра.

PUE — флаг разрешения работы подтягивающего регистра.

ST — флаг разрешения работы триггера Шмитта.

EN — флаг разрешения работы данного PAD.

Для индивидуального управления отдельным PAD используются команды

```
pad X [enable|disable|mux|ds|st|pud]
pad X <value>
```

где

X — десятичный номер PAD.

enable — разрешает работу заданного PAD.

disable — запрещает работу заданного PAD. Сбрасывает регистр PAD;

mux — задает функцию мультиплексирования для заданного PAD (0..3).

ds — задает значение силы тока (Driver Strength) для заданного PAD (0..7).

st — задает флаг использования триггера Шмидта (Schmitt trigger) для заданного PAD (0,1).

pud — задает флаг использования подтяжки (Pull Up/Down) для заданного PAD. Команда принимает 2 параметра: разрешение подтяжки (PUE: 0,1) и уровень подтяжки (PUD: 0,1).

value — непосредственное значение, записываемое в регистр настройки выбранного PAD

После исполнения команды на консоль выводится информация о новом состоянии PAD в формате, описанном выше.

Пример исполнения:

```
tmt > pad 26
PAD (026) [006c]: 010f  MUX:2 DS:7 --- --- -- EN
OK
tmt > pad 26 disable
PAD (026) [006c]: 008e  MUX:1 DS:7 --- --- -- --
OK
tmt > pad 26 enable
PAD (026) [006c]: 008f  MUX:1 DS:7 --- --- -- EN
OK
tmt > pad 26 mux 1
PAD (026) [006c]: 008f  MUX:1 DS:7 --- --- -- EN
OK
tmt > pad 26 ds 4
PAD (026) [006c]: 0089  MUX:1 DS:4 --- --- -- EN
OK
tmt > pad 26 st 1
PAD (026) [006c]: 0099  MUX:1 DS:4 --- --- ST EN
OK
tmt > pad 26 pud 1 1
PAD (026) [006c]: 00f9  MUX:1 DS:4 PUD PUE ST EN
OK
tmt > pad 26 0x00f1
PAD (026) [006c]: 00f1  MUX:1 DS:0 PUD PUE ST EN
OK
```

4.2.11 mdw, mdb, mmw, mmb

Команда «mdw», «mdb», «mmw», «mmb» осуществляют операции чтения/записи данных в оперативной памяти, и в области MMIO в виде байтов (8-бит) или слов (32-бита).

Данные команды не имеют ограничения по адресному пространству и могут читать или изменять данные в любой ячейке доступного адресного пространства 0x0000_0000 — 0xFFFF_FFFF.

⚠ При обращении к адресам вне оперативной памяти и области периферийных блоков, системой будет вызвано исключение (exception).

Формат вызова команды:

```
mdw <addr> <length>
mdb <addr> <length>
mmw <addr> <value> [count]
mmb <addr> <value> [count]
```

где

addr — стартовый адрес блока данных.

length — длина читаемого блока данных (по умолчанию 1).

value — записываемое значение (hex).

count — опциональное значение количества записываемых элементов.

Команда «mdw» выводит дамп блока данных, начиная с адреса `addr` длиной `length` слов (32-бит).

Команда «mdb» выводит дамп блока данных, начиная с адреса `addr` длиной `length` байт (8-бит).

Команда «mmw» производит модификацию значения слова (32-бит) по адресу `addr` значением `value`.

Команда «mmb» производит модификацию значения байта (8-бит) по адресу `addr` значением `value`.

Пример исполнения:

```
tmt > mdw 0x880000 20
00880000:  00000000  00000000  00000000  00000000
00880010:  00806558  00001004  0080556c  00000000
00880020:  00806558  00000004  008004dc  0080e000
00880030:  00003800  0000000d  0088012c  00000004
00880040:  00806558  008800e4  008003e0  00000000
tmt > mdb 0xa00000 48
00a00000:  6f  00  00  20  13  00  00  00  13  00  00  00  13  00  00  00
00a00010:  13  00  00  00  13  00  00  00  13  00  00  00  13  00  00  00
00a00020:  13  00  00  00  13  00  00  00  13  00  00  00  13  00  00  00
tmt > mmw 0x880000 0xdeadbeef
OK
tmt > mdw 0x880000
00880000:  deadbeef
OK
tmt > mmb 0x880003 fe
OK
tmt > mdw 0x880000
00880000:  feadbeef
OK
```

Команда чтения/записи слов использует алгоритм для невыровненных обращений. Таким образом, она не имеет ограничений на выравнивание.

```
tmt > mdw 0x880003
00880003:  000000fe
OK
```

4.2.12 mpu

Команда «mpu» выводит информацию о текущем состоянии записей Memory Protection Unit (MPU).

При вызове команды без параметров на консоль выводится содержимое записей MPU. Аналогичная информация выводится командой:

```
mpu dump
```

Пример исполнения:

```
tmt > mpu
MPU:
Entry:   0
Flags:   disabled
Entry:   1
Address: 00a00000
Mask:    fffe0000
MemType: Non-cacheable, strong-ordered
M-Mode:  RX
U-Mode:
Flags:   enabled
Entry:   2
Address: 00800000
Mask:    ffe00000
MemType: Cacheable, weakly-ordered
M-Mode:  RWX
U-Mode:  RWX
Flags:   enabled
Entry:   3
Address: 00000000
Mask:    ffe00000
MemType: Non-cacheable, strong-ordered
M-Mode:  RW
U-Mode:  RW
Flags:   enabled
```

```
Entry:   4
Flags:   disabled
Entry:   5
Flags:   disabled
Entry:   6
Flags:   disabled
Entry:   7
Address: 00084000
Mask:    fffff000
MemType: Memory-mapped I/O
M-Mode:  RW
U-Mode:
Flags:   enabled
Entry:   8
Flags:   disabled
Entry:   9
Flags:   disabled
Entry:   10
Flags:   disabled
Entry:   11
Flags:   disabled
Entry:   12
Flags:   disabled
Entry:   13
Flags:   disabled
Entry:   14
Flags:   disabled
Entry:   15
Address: 00084000
Mask:    fffff000
MemType: Memory-mapped I/O
M-Mode:  RW
U-Mode:
Flags:   enabled
OK
```

Здесь и далее, формат вывода информации о текущем состоянии записи блока MPU

```
Entry:   <Entry>
Address: <Address>
Mask:    <Mask>
MemType: <MemType>
M-Mode:  <M-Mode>
U-Mode:  <U-Mode>
Flags:   <Flags>
```

где

Entry — десятичное число, порядковый номер записи в блоке MPU, нумерация начинается с 0.

Address — базовый адрес описываемой области памяти.

Mask — маска адреса описываемой области памяти.

MemType — тип доступа в описываемую область памяти.

M-Mode — права доступа в описываемую область памяти в Machine Mode

(R,W,X).

U-Mode — права доступа в описываемую область памяти в User Mode

(R,W,X).

Flags — флаги текущего состояния записи (enabled,disabled,locked).

При настройке блока MPU начальным загрузчиком принимается следующая карта записей MPU для доступных областей памяти:

0 INIT Область, ответственная за настройку начального состояния MPU. После настройки всех областей отключается.

1 ROM Область памяти ПЗУ. Non-cacheable, strong-ordered. Read, Execute in Machine Mode.

2 RAM Область оперативной памяти SRAM. Cacheable, weakly-ordered. Read, Write, Execute in Machine Mode.

3 MMIO Область памяти Memory IO. Non-cacheable, strong-ordered. Read, Write in Machine Mode.

15 CPU MMIO Область памяти CPU Memory IO. CPU MMIO, non-cacheable, strong-ordered. Read, Write in Machine Mode.

4.2.13 uart

Команда «uart» осуществляет управление текущими параметрами серийной консоли.

Команда позволяет сменить текущий серийный порт и его скорость. При смене порта команда не выполняет модификацию настроек PAD контроллера, поэтому для использования серийных портов с мультиплексированными выводами необходимо произвести дополнительные настройки.

Формат вызова команды:

```
uart <port> <baud>
```

где

port — номер порта в системе.

baud — скорость порта в бод из стандартного ряда скоростей.

Пример исполнения:

```
tmt > uart 0 115200
tmt > uart 2 9600
```

При вызове команды для текущего порта (по умолчанию система стартует с серийным портом «boot») изменяется текущая скорость обмена. В остальных случаях, кроме инициализации и/или изменении скорости обмена происходит перенаправление ввода/вывода серийной консоли на выбранный к команде порт.

Вызов команды без параметров выводит список всех серийных портов, определенных в загрузчике, с обозначением текущего порта (символ “*”):

```
tmt > uart
   [0] serial@00100000 inactive
   [1] serial@00101000 inactive
   [2] serial@00102000 inactive
   [3] serial@00103000 inactive
   [4] serial@00104000 inactive
   [5] serial@00105000 inactive
   [6] serial@00106000 inactive
   [7] serial@00107000 inactive
 * [8] serial@00002000 active fifo 64 mode 9600 8N1 (boot)
```

4.2.14 spi

Команда «spi» осуществляет управление SPI NOR Flash на линии BOOT SPI.

```
spi
spi probe
spi direct [on|off]
spi read  <offset> <addr> <length>
spi write <offset> <addr> <length>
spi erase <offset>
spi erase page <offset>
spi erase sector <offset>
spi erase all
```

где

offset — смещение от начала SPI Flash.

addr — стартовый адрес блока данных в оперативной памяти.

length — длина блока данных.

Команда «spi» выводит базовую информацию о блоке SPI

Команда «spi probe» выполняет запросы к SPI NOR Flash при помощи JEDEC команд идентификации 0x9F, 0xAB, 0x90, 0x4B. Команда «spi direct» выполняет переключение контроллера SPI из режима прямой адресации в режим SPI и обратно.

Команда «spi direct» выполняет переключение контроллера SPI из режима прямой адресации в режим SPI и обратно.

Команда «spi read» считывает блок данных длиной length из SPI Flash со смещением offset от начала, в оперативную память по адресу addr (JEDEC команда 0x03).

Команда «spi write» стирает регион в SPI Flash, начиная со смещения offset, длиной length (блоками по 4KB) (JEDEC команда 0x20) и записывает в этот регион блоки данных по 256 байт (JEDEC команда 0x02) из оперативной памяти, начиная с адреса addr и длиной length.
Команда «spi erase» осуществляет стирание 4KB сектора SPI Flash в пределах расположения смещения offset (JEDEC команда 0x20).

Команда «spi erase page» осуществляет стирание страницы 4KB SPI Flash выбранного блока SSI в пределах расположения смещения offset (JEDEC команда 0x20).

Команда «spi erase sector» осуществляет стирание сектора 64KB SPI Flash выбранного блока SSI в пределах расположения смещения offset (JEDEC команда 0xD8).

Команда «spi erase all» осуществляет стирание всей SPI Flash (JEDEC команда 0xC7).

Пример исполнения:

```
tmt > spi
SPI: direct mode is enabled
ERROR: Command 'spi' failed (-5)
tmt > spi direct off
DSPI mode is off
OK
tmt > spi
* spi@00001000 flash: 1f8601ff sector:4096 page:256
OK
tmt > spi probe
SPI: spi@00001000
  [9f]: 20 20 15 10 00 00 00
  [ab]: -- -- -- 14 14 14 14
  [90]: -- -- -- -- -- -- --
  [4b]: -- -- -- -- -- -- --
OK
tmt > spi read 0x0 0x880000 0x400
Read 1024 bytes from 00000000@flash to 00880000@mem
OK
tmt > spi write 0x0 0x880000 0x400
Erase flash ..
Erased flash from 00000000@flash to 00000ffff@flash
Write flash
Write block to 00000000
Written 1024 bytes from 00880000@mem to 00000000@flash
OK
tmt > spi erase 0x10
Erased 4KiB block from 00000000@flash to 00000ffff@flash
OK
tmt > spi erase all
```

Вызов команды без параметров выводит краткую информацию о контроллере SPI и SPI NOR Flash:

```
tmt > spi
 * spi@00001000 flash: 20201510 sector:4096 page:256
```

При выходе адреса считывания за пределы оперативной памяти или превышении смещения SPI Flash 16MB блока (длина адреса 3 байта) выводится сообщение об ошибке:

```
tmt > spi read 0x0 0x990000 0x1000
End address 00991000 is out of memory
ERROR: Command 'spi' failed (-12)
tmt > spi read 0x1000000 0x880000 0x2000
Flash offset 01000000 exceeds flash space
ERROR: Command 'spi' failed (-12)
```

4.2.15 xload

Команда «xload» осуществляет загрузку бинарного программного кода в оперативную память через терминал с использованием протокола X-Modem.

Формат протокола обмена X-Modem [6]. Для загрузки данных можно использовать стандартные функции терминальных программ (передача по протоколу X-Modem).

Прервать исполнение команды можно нажатием клавиш <Ctrl+C>.

Формат вызова команды:

```
xload <адрес начала загрузки>
```

Пример исполнения:

```
tmt > xload 0x880000
Ready for X-Modem download to 0x880000 at 9600 bps...
CCC
Start Load Addr: 00880000
End Load Addr: 00880680
Total size: 00000680 (1664 bytes)
```

Вызов команды без параметров осуществляет загрузку бинарного исполняемого кода в начало SRAM.

Пример исполнения:

```
tmt > xload
Ready for X-Modem download to 00800000 at 9600 bps...
CCCCC
```

4.2.16 reset

Команда «reset» осуществляет сброс системы.

Доступно 2 варианта сброса системы: soft и hard. Режим soft осуществляет программный сброс начального загрузчика, без фактического сброса периферии СнК. Режим hard осуществляет сброс системы с использованием аппаратных систем сброса. При этом последовательно исполняется сброс с использованием блока GRU. При невозможности сброса выводится сообщение “Not implemented”.

Формат вызова команды:

```
reset <hard|soft>
```

Пример исполнения:

```
tmt > reset soft
Perform software reset
DragonBoot ROM monitor. Version {bootloader_build_ver}.DragonFly ({bootloader_build_date})
Copyright (C) 2016-2020 Tecon MT LLC. All rights reserved.
Type 'help' for help.
tmt > reset hard
Perform hardware reset
DragonBoot ROM monitor. Version {bootloader_build_ver}.DragonFly ({bootloader_build_date})
Copyright (C) 2016-2020 Tecon MT LLC. All rights reserved.
Type 'help' for help.
tmt >
```

Вызов команды без параметров осуществляет программный сброс системы:

```
tmt > reset
Perform software reset
DragonBoot ROM monitor. Version {bootloader_build_ver}.DragonFly ({bootloader_build_date})
Copyright (C) 2016-2020 Tecon MT LLC. All rights reserved.
Type 'help' for help.
tmt >
```

4.2.17 go

Команда «go» осуществляет передачу управления программному коду, расположенному в оперативной памяти.

При вызове возможна передача до 4 параметров, передаваемых в регистрах a0-a3. Обратный возврат в начальный загрузчик не предусмотрен, за исключением генерации исключения (exception) при исполнении.

Формат вызова команды:

```
go <addr> <arg0> <arg1> <arg2> <arg3>
```

где

addr — адрес передачи управления.

arg0 — аргументы, передаваемые в регистрах a0-a3.

arg1

arg2

arg3

Аргументы arg0-arg3 могут быть пропущены, в таком случае значения неиспользуемых регистров a0-a3 будут заполнены нулями.

Пример исполнения:

```
tmt > go 0x880200 0x01 0x02 0x03 0x04
Goto fun@00880200 (00000001, 00000002, 00000003, 00000004)
Hello World
arg0=00000001
arg1=00000002
arg2=00000003
arg3=00000004
```

В случае генерации исключения при неизменном значении регистра mtvec начальный загрузчик перехватит данное исключение, выведет отладочную информацию на момент возникновения исключения и перезапустит монитор:

```
tmt > go 0x880000
Goto fun@00880000 (00000000, 00000000, 00000000, 00000000)
HART0: Unhandled exception at 0x00880000 (Illegal instruction)
status 0x00003800   tval 0x00000000
ecause 0x00000002   epc  0x00880000
  00 fp  302e3120   01 ra  00802ec4   02 sp  0097dedc   03 gp  0097f800
  04 tp  0097dffc   05 t0  00880000   06 t1  0097df54   07 t2  00000000
  08 s0  00000000   09 s1  00000000   10 a0  00000000   11 a1  00000000
  12 a2  00000000   13 a3  00000000   14 a4  ffffffff   15 a5  00000000
  16 a6  ffffffff   17 a7  00000021   18 s2  00880000   19 s3  00000000
  20 s4  00000000   21 s5  0080e000   22 s6  0097e008   23 s7  0097e000
  24 s8  0080e000   25 s9  00000000   26 s10 00000000   27 s11 00000000
  28 t3  00000023   29 t4  00000000   30 t5  00000000   31 t6  00000000
Type 'help' for help.
```

4.2.18 call

Команда «call» осуществляет выполнение программного кода, расположенного в оперативной памяти.

При вызове возможна передача до 4 параметров, передаваемых в регистрах a0-a3. По окончании исполнения программного кода, по команде возврата "ret" осуществляется обратная передача управления начальному загрузчику. В качестве результата выполнения возвращается значение регистра a0.

Формат вызова команды:

```
call <addr> <arg0> <arg1> <arg2> <arg3>
```

где

addr — адрес передачи управления.

arg0 — аргументы, передаваемые в регистрах a0-a3.

arg1

arg2

arg3

Аргументы arg0-arg3 могут быть пропущены, в таком случае значения неиспользуемых регистров a0-a3 будут заполнены нулями.

Пример исполнения:

```
tmt > call 0x880200 0x01 0x02 0x03 0x04
Call fun@00880200 (00000001, 00000002, 00000003, 00000004)
Hello World
arg0=00000001
arg1=00000002
arg2=00000003
arg3=00000004
Return value 00000000(0)
```

В случае генерации исключения поведение начального загрузчика аналогично поведению при исполнении команды “go”.

4.2.19 hash

Команда «hash» осуществляет вычисление хеш-функции отдельного блока оперативной памяти по заданному алгоритму.

Формат вызова команды:

```
hash <algo> <start addr> <length>
```

где

algo — алгоритм вычисления хеш-функции.

start

addr — стартовый адрес блока оперативной памяти.

length — размер блока.

Пример исполнения:

```
tmt > hash xxh 0x880000 0x1000
xxh[00880000-00881000]=475546f5
OK
```

Вызов команды без параметров выводит список доступных алгоритмов для вычисления хеш-функции:

```
tmt > hash
Algos: crc16 crc32 md5 xxh
```

При попытке вычисления хеш-функции с блоком, выходящим за пределы оперативной памяти или по неподдерживаемому алгоритму выводится сообщение об ошибке:

```
tmt > hash crc32 0x990000 0x1000
Region 00990000-00991000 is out of RAM
ERROR: Command 'hash' failed (-12)
tmt > hash crc8 0x880000 0x1000
Unknown algo 'crc8'
ERROR: Command 'hash' failed (-22)
```

Для расчета хеш-функций CRC32 и CRC16 используется аппаратный блок CRC.

4.2.20 ptrace

Команда «ptrace» Осуществляет управлением системой программно-аппаратной отладки PINTRACE.

Система осуществляет выдачу контрольных значений при прохождении отдельных этапов работы начального загрузчика в виде последовательного кода на заданном выводе СнК: GPIO или SYS_ERR. Вывод GPIO или SYS_ERR определяется на этапе конфигурирования начального загрузчика. Для СнК "Стрекоза" используется только вывод SYS_ERR (управляется через блок контроля системы). Формат вывода данных и временные характеристики сигналов приведены в разделе 5. Коды контрольных значений системы PINTRACE приведены в приложении 6.

Формат вызова команды:

```
ptrace <on|off|status>
ptrace send <код>
```

где

код — произвольный 16-битный код.

Команда «ptrace on» разрешает работу системы PINTRACE (в случае использовании блока GPIO вывод GPIO системы переводится в режим “output”).

Команда «ptrace off» запрещает работу системы PINTRACE (в случае использовании блока GPIO вывод GPIO * системы переводится в режим “input”).

Команда «ptrace status» выдает текущее состояние системы PINTRACE.

Пример исполнения:

```
tmt > ptrace off
Pintrace disabled
OK
tmt > ptrace on
Pintrace enabled
OK
tmt > ptrace status
Pintrace enabled
OK
tmt > ptrace send 0x30
OK
```

При попытке вызова команды с иными параметрами выводится сообщение об ошибке:

```
tmt > ptrace enable
Invalid operation 'enable'
ERROR: Command 'ptrace' failed (-22)
```

4.2.21 sleep

Команда «sleep» осуществляет паузу в работе системы на заданное количество секунд.

Формат вызова команды:

```
sleep X
где X - количество секунд ожидания
Пример исполнения:
tmt > sleep 10
..........
OK
```

После каждой секунды ожидания система выводит символ “.”.

4.2.22 halt

Команда «halt» осуществляет остановку системы.

Формат вызова команды:

```
halt
```

Пример исполнения:

```
tmt > halt
System halted.
```

Система переходит в состояние непрерывного цикла с выполнением команды “wfi”.

Глава 5

Диагностическая система PinTrace

Отладочный интерфейс PinTrace использует интерфейс System Controller (линия SYS_ERR) для вывода информации о контрольных точках исполнения начального загрузчика и ошибок в виде последовательного кода.

Система использует битовый режим кодирования информации, значению логической единицы соответствует высокий уровень сигнала на линии, значению логического нуля — низкий уровень. Каждая кодовая посылка состоит из стартовой группы (start preamble), информационной группы и стоп группы (stop preamble). Стартовая группа посылки обозначается импульсом низкого уровня длительностью 8 мкс. Стоп группа обозначается импульсом высокого уровня длительностью 8 мкс. Каждый бит информационной группы имеет длительность 1мкс, размер информационной группы составляет 16 бит. В текущей версии начального загрузчика для системы PinTrace СнК "Стрекоза" используется линия SYS_ERR блока System Controller.

Примеры временных диаграмм передачи отладочной информации приведены на рисунке 5.1.

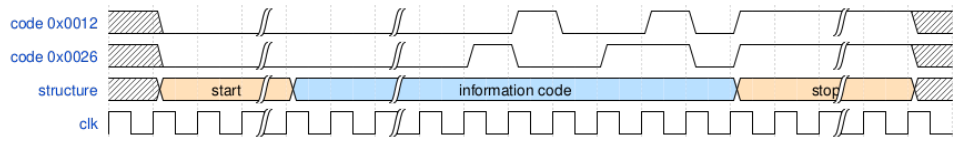

Рис. 5.1: Временные диаграммы сигналов системы PinTrace

Перечень кодов этапов исполнения начального загрузчика и кодов ошибок представлен в приложении 6.

Глава 6

Перечень кодов и сообщений системы PinTrace

Код	Мнемоника	Значение
Глобальное состояние системы PinTrace		
0x0002	PT_OFF	Система PinTrace отключена.
0x0006	PT_ON	Система PinTrace включена.
Низкоуровневая инициализация		
0x0010	PT_MAIN_START	Старт основного кода программы (после архитектурно-зависимой инициализации).
0x0011	PT_MAIN_TIMER	Инициализация системного таймера.
0x0012	PT_MAIN_CACHE	Инициализация weak/strong ordering областей памяти.
0x0013	PT_MAIN_MEM	Очистка области памяти данных.
Инициализация драйверов		
0x0020	PT_DRV_SYSCTRL	Инициализация драйвера System Controller.
0x0021	PT_DRV_PTRACE	Инициализация линии PinTrace.
0x0022	PT_DRV_UART	Инициализация драйвера UART.
0x0023	PT_DRV_SERIAL	Инициализация отладочной консоли.
0x0024	PT_DRV_BOOT	Инициализация драйвера Boot Controller.
0x0025	PT_DRV_LCRU	Инициализация драйвера LCRU.
0x0026	PT_DRV_PAD	Инициализация драйвера PAD.
0x0027	PT_DRV_GPIO	Инициализация драйвера GPIO.
0x0028	PT_DRV_EFUSE	Инициализация драйвера EFUSE.
0x0029	PT_DRV_GRU	Инициализация драйвера GRU.
0x002A	PT_DRV_NOR	Инициализация драйвера NSC (не задействовано).
0x002B	PT_DRV_REMAP	Инициализация драйвера REMAP.
0x002C	PT_DRV_SPI	Инициализация драйвера SPI.
0x002D	PT_DRV_SSI	Инициализация драйвера SSI.
0x002E	PT_DRV_TSENS	Инициализация драйвера Temperature Sensor (не задействовано).
0x002F	PT_DRV_WDT	Инициализация драйвера WDT (не задействовано).
Инициализация системных устройств		
0x0030	PT_INIT_MPU	Инициализация Memory Protection Unit (MPU).
0x0031	PT_INIT_TIMER	Инициализация Machine Mode Timer (MTIMER).
0x0032	PT_INIT_MBOX	Инициализация intercore Mailbox (MBOX).
Запуск системы загрузки		
0x0040 ... 0x004F	PT_INIT_START/ PT_INIT_CORE	Старт ядра 0 ... 15.
0x0060 ... 0x006F	PT_INIT_STOP/ PT_DEINIT_CORE	Остановка ядра 0 ... 15.
0x0080	PT_INIT_DATA	Инициализация Core Local Data.
0x0084	PT_INIT_PASSWD	Инициализация системы ограничения доступа в интерактивный монитор (не задействовано).

0x0088	PT_INIT_BOOT	Определение типа загрузки, передача управления выбранной процедуре загрузки.
Процесс загрузки		
0x0100 ... 0x010F	PT_BOOT_SECTOR	Выбор загрузочного сектора 0 ... 15 (не задействовано).
0x0110	PT_BOOT_SPI	Загрузка со внутренней SPI Flash (не задействовано).
0x0111	PT_BOOT_NOR	Загрузка с NOR/SRAM контроллера (не задействовано).
0x0112	PT_BOOT_UART	Загрузка с последовательного интерфейса (не задействовано).
0x0120 ... 0x012F	PT_BOOT_MEDIA	Невозможно обратиться к загрузочному устройству в сегментах 0 ... 15.
0x0130	PT_BOOT_TIMEOUT	Таймаут ожидания загрузочного устройства.
0x0132	PT_BOOT_HCRC	Ошибка контрольной суммы заголовка загрузочного сегмента.
0x0134	PT_BOOT_DCRC	Ошибка контрольной суммы данных загрузочного сегмента.
0x0140	PT_BOOT_ERROR	Общая ошибка загрузки.
Системные события		
0x0200	PT_MONITOR	Старт интерактивного монитора.
0x0400	PT_RESET_START	Старт программного сброса (Software reset).
0x0800	PT_HANG	Остановка системы.
0x1000 ... 0x100F	PT_EXCEPTION	Аппаратное исключение. Номер исключения — последний байт кода.

Литература и ссылки

[1] C99 ISO/IEC 9899:TC3 draft http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1256.pdf

[2] Linux kernel coding style https://www.kernel.org/doc/Documentation/CodingStyle

[3] The RISC-V Instruction Set Manual. Volume I: User-Level ISA Version 2.2. https://riscv.org//wp-content/uploads/2017/05/riscv-spec-v2.2.pdf

[4] The RISC-V Instruction Set Manual. Volume II: Privileged Architecture Version 1.10. https://riscv.org//wp-content/uploads/2017/05/riscv-privileged-v1.10.pdf

[5] SRECORD (SREC) file format https://en.wikipedia.org/wiki/SREC_(file_format)

[6] X-MODEM serial protocol https://en.wikipedia.org/wiki/XMODEM

[7] Архитектурная спецификация СнК "Стрекоза" (SoC "DragonFly") v0.0.2 / 06/24/2020

[8] Микроархитектурная спецификация СнК "Стрекоза" (SoC "DragonFly") v0.0.6 / 07/31/2020

[9] Микроархитектурная спецификация блока REMAP v0.0.1 / 03.08.2018

[10] Микроархитектурная спецификация блока CRC / 04/06/2020

[11] Инструкция по программированию блока FUSE / 08/11/2020

[12] Микроархитектурная спецификация блока GPIO v1.2.5 / 08/14/2019

[13] Микроархитектурная спецификация блока SCR_1_10_CPI_MAILBOX 05/22/2020

[14] Микроархитектурная спецификация блока SPI v0.0.1 / 03/11/2020

Версии документа

№	Дата	Внесённые изменения	Автор
0.1.01	17.09.2020	Документация перенесена из проекта Temon и адаптирована под название проекта	Дунаев Д.М.
0.1.02	17.09.2020	Документация полностью адаптирована под проект DragonFly и дополнена описанием новых команд	Дунаев Д.М.
0.2.01	05.10.2023	Перевод документа в формат LyX	Гурин К.Л.
0.2.02	20.10.2023	Исправления в тексте согласно регламента технического ревью.	Дунаев Д.М.
0.2.03	09.11.2023	Уточнения версий спецификаций	Гурин К.Л.