ТЕКОН
Микропроцессорные
Технологии

ВСТРОЕННЫЙ НАЧАЛЬНЫЙ ЗАГРУЗЧИК СНК "ДРУЖБА" И СНК "ШЕРШЕНЬ"

Москва, 2022

Юридический адрес:
3-я Хорошевская ул., д. 20,
Москва, 111250, Россия,
ООО «ТЕКОН Микропроцессорные технологии»
+7 (495) 730-41-12
+7 (495) 730-41-12
support@tecon.ru
www.tecon.ru

Почтовый адрес:
3-я Хорошевская ул., д. 20,
Москва, 111250, Россия,
ООО «ТЕКОН Микропроцессорные технологии»
+7 (495) 730-41-12
+7 (495) 730-41-12
support@tecon.ru
www.tecon.ru

СОДЕРЖАНИЕ

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).

API
: Программный интерфейс приложения (англ. Application Programming Interface).

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.

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

FLASH
: Тип полупроводниковой электрически перепрограммируемой памяти.

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

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

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).

TCM
: Статическая оперативная память с быстрым временем доступа (англ. Tightly coupled memory).

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

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

1. ОБЩИЕ СВЕДЕНИЯ

Название продукта: “Начальный загрузчик TEMON”.

Наименование: файла образа ПЗУ интегральной схемы temon.bin.

Внутреннее имя: temon.

Архитектура: RISC-V, спецификация RV32IMFDC, ISA v.1.7.

Разрядность: 32 бита.

Версия продукта: 1.24.

Язык интерфейса: English (Unated States).

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

Версия и тип компилятора: riscv32-unknown-linux-gnu-gcc (GCC) 7.1.1.

Версия и тип программы линковки: GNU ld (GNU Binutils) 2.29.

Размер бинарного образа: 32 КБ.

Минимальное количество оперативной памяти: 16 КБ.

Наличие диагностических интерфейсов:

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

2. НАЗНАЧЕНИЕ НАЧАЛЬНОГО ЗАГРУЗЧИКА

Начальный загрузчик TEMON (TEcon MONitor) представляет собой загрузчик первого уровня, расположенный в read-only ПЗУ микроконтроллера (ROM). Основными функциями начального загрузчика являются:

- Начальная инициализация микроконтроллера.
- Базовое тестирование загрузочных интерфейсов.
- Копирование ПО вторичного загрузчика в область встроенного блока SRAM, а также проверка целостности (CRC) и передача управления загруженному ПО.
- Базовые функции отладочного монитора (чтение и запись системных регистров, регистров периферийных устройств и области памяти).
- Загрузка отладочного ПО через последовательный интерфейс, предоставление базовых функций ввода-вывода стороннему ПО.
- Обработка ошибок и исключений в режиме информирования оператора.

3. РЕЖИМЫ РАБОТЫ НАЧАЛЬНОГО ЗАГРУЗЧИКА

Загрузка исполняемого кода вторичного загрузчика или системного ПО может осуществляться в автоматическом и ручном режимах. Ручной режим загрузки требует вмешательства оператора и является отладочным. Выбор режима и интерфейса загрузки осуществляется посредством декодирования состояния входов bootstrap в Boot Controller.

В автоматическом режиме загрузки в СнК “Дружба” и в СнК “Шершень” внутренняя загрузочная Flash память отображается в загрузочную область адресного пространства СнК. Таким образом осуществляется автоматический старт системы при снятия сигнала сброса. Выбор стартового ядра в СнК “Шершень” осуществляется при помощи входов bootstrap.

При загрузке в ручном режиме, оператор может перевести начальный загрузчик в режим интерактивного монитора. В режиме интерактивного монитора возможна загрузка ПО и его запуск.

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 переходит в режим интерактивного монитора.

3.1. Загрузка программного обеспечения

3.1.1. Подготовка системы к работе

1. Перевести СнК в необходимый режим загрузки посредством выставления `bootstrap` входов (Boot Controller).
2. Подключить к последовательному порту `BOOT UART` рабочий терминал.
3. Запустить на рабочем терминале программу для работы с последовательным портом (Linux: minicom, picocom, screen; Windows: hyperterm, putty).
4. Настроить соответствующий последовательный порт (зависит от операционной системы) на режим работы 9600 8N2 (9600 бод, без контроля четности, 2 стоп-бита).
5. Подать питание на отладочную плату СнК или кратковременно нажать кнопку RESET.
6. После старта системы в зависимости от режима выполнится загрузка ПО со встроенной SPI Flash или появится приглашение терминала интерактивного монитора Temon.

3.1.2. Автоматическая загрузка программного обеспечения со встроенной SPI Flash

В случае установки режима загрузки со встроенной SPI Flash Boot Controller осуществит прямую адресацию области данных встроенной SPI Flash по стартовому адресу системы. Таким образом при подаче питания или после снятия сигнала RESET произойдет исполнение кода, расположенного во встроенной SPI Flash. Переход в интерактивный режим монитора в данном варианте загрузки невозможен.

3.1.3. Загрузка программного при помощи интерактивного монитора

Загрузка программного обеспечения при помощи последовательного порта

1. Дождаться приглашения терминала интерактивного монитора Temon.

```
TEMON ROM monitor. Version 1.18.Friendship2 (Jul 03 2018 - 18:20:22 +0300)
```

```
Copyright (C) 2018 Tecon MT LLC. All rights reserved.
Type 'help' for help.

tmt $
```

1. Запустить загрузку программного кода в оперативную память командой “xload”
2. Осуществить загрузку бинарного программного кода при помощи терминала по протоколу X-Modem.

```
tmt $ xload 0x800200
Ready for X-Modem download to 00800000 at 9600 bps...
CCC
Start Load Addr: 00800200
End Load Addr: 00801200
Total size: 00001000 (4096 bytes)

OK
tmt $
```

Параметры вызова команды “xload” описаны в разделе «xload».

1. При необходимости произвести проверку контрольной суммы загруженного кода при помощи команды “hash”

```
tmt $ hash crc32 0x800200 0x1000

crc32[00800200-00801200]=4cf5a7de

OK
tmt $
```

Параметры вызова команды “hash” описаны в разделе «hash».

1. Передать управление загруженному коду при помощи команд “go” или “call”

```
tmt $ call 0x800200 1 2 3 4
Call fun@00800200 (00000001, 00000002, 00000003, 00000004)

Hello World

arg0=00000001
arg1=00000002
arg2=00000003
arg3=00000004

Return value 00000000(0)

OK
```

Параметры вызова команд “go” и “call” описаны в разделах «go» и «call» соответственно.

Загрузка программного обеспечения со встроенной SPI Flash

1. Дождаться приглашения терминала интерактивного монитора Temon.

```
TEMON ROM monitor. Version 1.18.Friendship2 (Jul 03 2018 - 18:20:22 +0300)
Copyright (C) 2018 Tecon MT LLC. All rights reserved.
Type 'help' for help.

tmt $
```

1. Загрузить бинарный программный код в оперативную память из внутренней SPI Flash командой “spi”

```
tmt $ spi probe
SPI: spi@00415000
  [9f]: 1f 86 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  [ab]: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  [90]: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  [4b]: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

OK
tmt $ spi read 0x0 0x800000 0x1000
Read 4096 bytes from 00000000@flash to 00800000@mem

OK
tmt $
```

Параметры вызова команды “spi” описаны в разделе «spi».

1. Произвести проверку контрольной суммы загруженного кода при помощи команды “hash”

```
tmt $ hash xxh 0x800000 0x1000

xxh[00800000-00801000]=d801b854

OK
tmt $
```

Параметры вызова команды “hash” описаны в разделе «hash».

1. Передать управление загруженному коду при помощи команд “go” или “call”

```
tmt $ go 0x800000
Goto fun@00800000 (00000000, 00000000, 00000000, 00000000)

System Info Example:
--------------------
Total HARTs: 1
Current HART: 0
SYS Freq: 25 MHz
RTC Freq: 100 KHz
Memory: 0x00800000-0x00900000
HARTs info:
 * HART0: SCR4 Friendship2 [16038000] RV32IMFDCX [0090112c] @ 25MHz

halt...
```

Параметры вызова команд “go” и “call” описаны в разделах «go» и «call» соответственно.

3.2. Режим интерактивного монитора

3.2.1. Подготовка системы к работе

1 Выполнить действия п.3.1.1.

2 По окончанию перечисленных действий, система выведет сообщение с названием начального загрузчика, его версию и приглашение командной строки (“tmt $”):

```
TEMON ROM monitor. Version 1.18.Friendship2 (Jul 03 2018 - 18:20:22 +0300)
Copyright (C) 2018 Tecon MT LLC. All rights reserved.
Type 'help' for help.

tmt $
```

3.2.2. Основные команды монитора

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

help

Команда “help” выводит информацию о доступных командах.

Пример для СнК “Дружба”:

```
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
  pad      Control & Status of specified PAD
  mdw      Dump memory in words
  mdb      Dump memory in bytes
  mmw      Set memory data in word
  mmb      Set memory data in byte
  uart     Set console UART
  spi      Load data in SRAM from SPI
  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

OK
```

Пример для СнК “Шершень”:

```
tmt $ help

Aviable commands:

  help  version  status  info  core  mfcsr  mtcsr  pad  mdw  mdb
  mmw  mmb  uart  spi  dspi  xload  reset  go  call  hash  ptrace

OK
```

version

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

Пример для СнК “Дружба”:

```
tmt $ version

TEMON ROM monitor. Version 1.18.Friendship2 (Jul 03 2018 - 18:20:22 +0300)
riscv32-unknown-linux-gnu-gcc (GCC) 5.3.0
GNU ld (GNU Binutils) 2.25.1

OK
```

Пример для СнК “Шершень”:

```
tmt $ version

TEMON ROM monitor. Version 1.24-rc4.Hornet2 (Feb 21 2019 - 11:05:20 +0300)
riscv32-unknown-elf-gcc (sdk-mt1: build 20190207 commit: 5009846) 7.1.1 20170509
GNU ld (GNU Binutils) 2.29
Memory Map v0.0.3 / 11.09.2018
Local 3c76a18

OK
```

status

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

```
tmt $ status
 * IE: 0 PRV: M
   IE1:0 PRV1:M
   IE2:0 PRV2:U
   IE3:0 PRV3:U
   FS: 1 MPRV:0
   XS: 0
   VM: Mbare

OK
```

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 для СнК “Дружба”:

```
tmt $ info

Technical information.

SoC Name: Friendship2 (rev.00000001)
Num HARTS: 1 (max 1)

HART0: Syntacore SCR4[16038000] RV32IMFDCXU [0090112c] @ 25MHz
L1i cache: n/a
L1d cache: n/a
L2  cache: n/a
RTL release: 26.01.17 (build 01)

CPU freq: 25MHz
RTC freq: 100KHz

Total SRAM: 00800000-00900000 (1024 KB)
Free  SRAM: 00800000-00900000 (1024 KB)
Total FRAM: 00480000-004a0000 (128 KB)
Free  FRAM: 00480000-0049d000 (116 KB)
Boot   ROM: 00000000-00008000 (32 KB)

Console:
   [0] serial@00440000
   [1] serial@00441000
   [2] serial@00442000
   [3] serial@00443000
 * [4] serial@00414000

Current status:
 * IE: 0 PRV: M
   IE1:0 PRV1:M
   IE2:0 PRV2:U
   IE3:0 PRV3:U
   FS: 1 MPRV:0
   XS: 0
   VM: Mbare

SPI:
 * spi@00415000 flash: 001f8601 sector:4096 page:256

PAD:
PAD0(1) [2004]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(2) [2008]: 0000  MUX:0 DS:0 --- --- -- --
```

```
PAD0(3) [200c]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(4) [2010]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(5) [2014]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(6) [2018]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(7) [201c]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(8) [2020]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(9) [2024]: 0000  MUX:0 DS:0 --- --- -- --
PAD(10) [2028]: 0000  MUX:0 DS:0 --- --- -- --
PAD(11) [202c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(12) [2030]: 0000  MUX:0 DS:0 --- --- -- --
PAD(13) [2034]: 0000  MUX:0 DS:0 --- --- -- --
PAD(14) [2038]: 0000  MUX:0 DS:0 --- --- -- --
PAD(15) [203c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(16) [2040]: 0000  MUX:0 DS:0 --- --- -- --
PAD(17) [2044]: 0000  MUX:0 DS:0 --- --- -- --
PAD(18) [2048]: 0000  MUX:0 DS:0 --- --- -- --
PAD(19) [204c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(20) [2050]: 0000  MUX:0 DS:0 --- --- -- --
PAD(21) [2054]: 0000  MUX:0 DS:0 --- --- -- --
PAD(22) [2058]: 0000  MUX:0 DS:0 --- --- -- --
PAD(23) [205c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(24) [2060]: 0000  MUX:0 DS:0 --- --- -- --
PAD(25) [2064]: 0000  MUX:0 DS:0 --- --- -- --
PAD(26) [2068]: 0000  MUX:0 DS:0 --- --- -- --
PAD(27) [206c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(28) [2070]: 0000  MUX:0 DS:0 --- --- -- --
PAD(29) [2074]: 0000  MUX:0 DS:0 --- --- -- --
PAD(30) [2078]: 0000  MUX:0 DS:0 --- --- -- --
PAD(31) [207c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(32) [2080]: 0000  MUX:0 DS:0 --- --- -- --
PAD(33) [2084]: 0000  MUX:0 DS:0 --- --- -- --
PAD(34) [2088]: 0000  MUX:0 DS:0 --- --- -- --
PAD(35) [208c]: 0000  MUX:0 DS:0 --- --- -- --

PinTrace:
 * ptrace@lcru

HART0 CSR:
[0001] fflags:       00000000    RW- SCR4 SCR5
[0002] frm:          00000000    RW- SCR4 SCR5
[0003] fcsr:         00000000    RW- SCR4 SCR5
[0c00] cycle:        0119ac6b    R-- SCR4 SCR5
[0c01] time:         0002d12b    R-- SCR4 SCR5
[0c02] instret:      0119bd18    R-- SCR4 SCR5
[0c80] cycleh:       00000000    R-- SCR4 SCR5
[0c81] timeh:        00000000    R-- SCR4 SCR5
[0c82] instreth:     00000000    R-- SCR4 SCR5
[0a01] stimew:       n/a         RW- ---- SCR5
[0a81] stimehw:      n/a         RW- ---- SCR5
[0300] mstatus:      00001036    RWX SCR4 SCR5
[0301] mtvec:        00000100    RWX SCR4 SCR5
```

```
[0302] mtdeleg:      00000000    R-- SCR4 SCR5
[0304] mie:          00000000    RWX SCR4 SCR5
[0321] mtimecmp:     00000000    RW- SCR4 SCR5
[0340] mscratch:     00000000    RW- SCR4 SCR5
[0341] mepc:         00000000    RW- SCR4 SCR5
[0342] mcause:       00000000    R-- SCR4 SCR5
[0343] mbadaddr:     00000000    R-- SCR4 SCR5
[0344] mip:          00000000    RW- SCR4 SCR5
[0701] mtime:        0002d291    RW- SCR4 SCR5
[0741] mtimeh:       00000000    RW- SCR4 SCR5
[0790] mipic_cisv:   00000020    R-X SCR4 SCR5
[0791] mipic_cicsr:  00000000    RWX SCR4 SCR5
[0792] mipic_ipr:    00000000    RWX SCR4 SCR5
[0793] mipic_isvr:   00000000    R-X SCR4 SCR5
[0794] mipic_eoi:    00000000    -WX SCR4 SCR5
[0795] mipic_soi:    00000000    -WX SCR4 SCR5
[0796] mipic_idx:    00000000    RWX SCR4 SCR5
[0797] mipic_icsr:   00020300    RWX SCR4 SCR5
[0798] mipic_ier:    00000000    RWX SCR4 SCR5
[0799] mipic_imap:   ffffffff    R-- SCR4 SCR5
[07b0] mrc_gbl:      000000f0    RW- SCR4 SCR5
[07b1] mrc_mask:     00000000    RW- SCR4 SCR5
[07b2] mrc_pattern:  00000000    RW- SCR4 SCR5
[07b3] mrc_upd:      00000000    RW- SCR4 SCR5
[07b4] clk_setup:    00010064    RWX SCR4 SCR5
[07b8] miccm_addr:   n/a         RW- ---- SCR5
[07b9] miccm_status: n/a         R-- ---- SCR5
[07ba] miccm_rdata:  n/a         RW- ---- SCR5
[07bb] miccm_wdata:  n/a         -W- ---- SCR5
[0bfb] pccnt:        n/a         RW- ---- SCR5
[0bfc] pwr_ctrl:     n/a         R-- ---- SCR5
[0bfd] pwr_state:    n/a         R-- ---- SCR5
[0bfe] cg_clkc:      n/a         RW- ---- SCR5
[0bff] fg_clkc:      n/a         RW- ---- SCR5
[0f00] mcpuid:       0090112c    R-- SCR4 SCR5
[0f01] mimpid:       16038000    R-- SCR4 SCR5
[0f10] mhartid:      00000000    R-- SCR4 SCR5
[0fb0] cache_l1:     n/a         R-- ---- SCR5
[0fb1] cache_l2:     n/a         R-- ---- SCR5
[0fc0] mrtlid:       17012601    R-X SCR4 SCR5
[0fc1] mefaddr:      00000000    R-- SCR4 SCR5
[0fc2] mefcmd:       00000000    R-- SCR4 SCR5

OK
```

Пример для СнК “Шершень”:

```
tmt $ info

Technical information.

SoC Name: Hornet2 (rev.00000002)
```

```
Num HARTS: 3 (max 3)

HART0: Syntacore SCR5[16038000] RV32IMAFDCXSU [0094112d] @ 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: 512K, 8-way, 32-byte line, 512-index, 4 banks, write-back, allocate,
 inclusive
RTL release: 25.10.17 (build 00)

HART1: Syntacore SCR5[16038000] RV32IMAFDCXSU [0094112d] @ 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: 512K, 8-way, 32-byte line, 512-index, 4 banks, write-back, allocate,
 inclusive
RTL release: 25.10.17 (build 00)

HART2: Syntacore SCR4[16038000] RV32IMFDCXU [0090112c] @ 25MHz
L1i cache: n/a
L1d cache: n/a
L2  cache: n/a
RTL release: 26.01.17 (build 01)

CPU freq: 25MHz
RTC freq: 100KHz

Total SRAM: 00800000-00900000 (1024 KB)
Free  SRAM: 00800000-008fc800 (1010 KB)
Boot   ROM: 00000000-00008000 (32 KB)

Console:
 * [0] serial@00603000

Current status:
 * IE: 0 PRV: M
   IE1:0 PRV1:M
   IE2:0 PRV2:U
   IE3:0 PRV3:U
   FS: 1 MPRV:0
   XS: 0
   VM: Mbare

SPI:
 * spi@0060c000 flash: 001f8601 sector:4096 page:256

PAD:
PAD0(1) [0004]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(2) [0008]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(3) [000c]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(4) [0010]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(5) [0014]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(6) [0018]: 0000  MUX:0 DS:0 --- --- -- --
```

```
PAD0(7) [001c]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(8) [0020]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(9) [0024]: 0000  MUX:0 DS:0 --- --- -- --
PAD(10) [0028]: 0000  MUX:0 DS:0 --- --- -- --
PAD(11) [002c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(12) [0030]: 0000  MUX:0 DS:0 --- --- -- --
PAD(13) [0034]: 0000  MUX:0 DS:0 --- --- -- --
PAD(14) [0038]: 0000  MUX:0 DS:0 --- --- -- --
PAD(15) [003c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(16) [0040]: 0000  MUX:0 DS:0 --- --- -- --
PAD(17) [0044]: 0000  MUX:0 DS:0 --- --- -- --
PAD(18) [0048]: 0000  MUX:0 DS:0 --- --- -- --
PAD(19) [004c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(20) [0050]: 0000  MUX:0 DS:0 --- --- -- --
PAD(21) [0054]: 0000  MUX:0 DS:0 --- --- -- --
PAD(22) [0058]: 0000  MUX:0 DS:0 --- --- -- --
PAD(23) [005c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(24) [0060]: 0000  MUX:0 DS:0 --- --- -- --
PAD(25) [0064]: 0000  MUX:0 DS:0 --- --- -- --
PAD(26) [0068]: 0000  MUX:0 DS:0 --- --- -- --
PAD(27) [006c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(28) [0070]: 0000  MUX:0 DS:0 --- --- -- --
PAD(29) [0074]: 0000  MUX:0 DS:0 --- --- -- --
PAD(30) [0078]: 0000  MUX:0 DS:0 --- --- -- --
PAD(31) [007c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(32) [0080]: 0000  MUX:0 DS:0 --- --- -- --
PAD(33) [0084]: 0000  MUX:0 DS:0 --- --- -- --
PAD(34) [0088]: 0000  MUX:0 DS:0 --- --- -- --
PAD(35) [008c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(36) [0090]: 0000  MUX:0 DS:0 --- --- -- --
PAD(37) [0094]: 0000  MUX:0 DS:0 --- --- -- --
PAD(38) [0098]: 0000  MUX:0 DS:0 --- --- -- --
PAD(39) [009c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(40) [00a0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(41) [00a4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(42) [00a8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(43) [00ac]: 0000  MUX:0 DS:0 --- --- -- --
PAD(44) [00b0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(45) [00b4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(46) [00b8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(47) [00bc]: 0000  MUX:0 DS:0 --- --- -- --
PAD(48) [00c0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(49) [00c4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(50) [00c8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(51) [00cc]: 0000  MUX:0 DS:0 --- --- -- --
PAD(52) [00d0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(53) [00d4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(54) [00d8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(55) [00dc]: 0000  MUX:0 DS:0 --- --- -- --
PAD(56) [00e0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(57) [00e4]: 0000  MUX:0 DS:0 --- --- -- --
```

```
PAD(58) [00e8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(59) [00ec]: 0000  MUX:0 DS:0 --- --- -- --
PAD(60) [00f0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(61) [00f4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(62) [00f8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(63) [00fc]: 0000  MUX:0 DS:0 --- --- -- --
PAD(64) [0100]: 0000  MUX:0 DS:0 --- --- -- --
PAD(65) [0104]: 0000  MUX:0 DS:0 --- --- -- --
PAD(66) [0108]: 0000  MUX:0 DS:0 --- --- -- --
PAD(67) [010c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(68) [0110]: 0000  MUX:0 DS:0 --- --- -- --
PAD(69) [0114]: 0000  MUX:0 DS:0 --- --- -- --
PAD(70) [0118]: 0000  MUX:0 DS:0 --- --- -- --
PAD(71) [011c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(72) [0120]: 0000  MUX:0 DS:0 --- --- -- --
PAD(73) [0124]: 0000  MUX:0 DS:0 --- --- -- --
PAD(74) [0128]: 0000  MUX:0 DS:0 --- --- -- --
PAD(75) [012c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(76) [0130]: 0000  MUX:0 DS:0 --- --- -- --
PAD(77) [0134]: 0000  MUX:0 DS:0 --- --- -- --
PAD(78) [0138]: 0000  MUX:0 DS:0 --- --- -- --
PAD(79) [013c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(80) [0140]: 0000  MUX:0 DS:0 --- --- -- --
PAD(81) [0144]: 0000  MUX:0 DS:0 --- --- -- --
PAD(82) [0148]: 0000  MUX:0 DS:0 --- --- -- --
PAD(83) [014c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(84) [0150]: 0000  MUX:0 DS:0 --- --- -- --
PAD(85) [0154]: 0000  MUX:0 DS:0 --- --- -- --
PAD(86) [0158]: 0000  MUX:0 DS:0 --- --- -- --
PAD(87) [015c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(88) [0160]: 0000  MUX:0 DS:0 --- --- -- --
PAD(89) [0164]: 0000  MUX:0 DS:0 --- --- -- --
PAD(90) [0168]: 0000  MUX:0 DS:0 --- --- -- --
PAD(91) [016c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(92) [0170]: 0000  MUX:0 DS:0 --- --- -- --
PAD(93) [0174]: 0000  MUX:0 DS:0 --- --- -- --
PAD(94) [0178]: 0000  MUX:0 DS:0 --- --- -- --
PAD(95) [017c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(96) [0180]: 0000  MUX:0 DS:0 --- --- -- --
PAD(97) [0184]: 0000  MUX:0 DS:0 --- --- -- --
PAD(98) [0188]: 0000  MUX:0 DS:0 --- --- -- --
PAD(99) [018c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(100) [0190]: 0000  MUX:0 DS:0 --- --- -- --
PAD(101) [0194]: 0000  MUX:0 DS:0 --- --- -- --
PAD(102) [0198]: 0000  MUX:0 DS:0 --- --- -- --
PAD(103) [019c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(104) [01a0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(105) [01a4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(106) [01a8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(107) [01ac]: 0000  MUX:0 DS:0 --- --- -- --
PAD(108) [01b0]: 0000  MUX:0 DS:0 --- --- -- --
```

```
PAD(109) [01b4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(110) [01b8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(111) [01bc]: 0000  MUX:0 DS:0 --- --- -- --
PAD(112) [01c0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(113) [01c4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(114) [01c8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(115) [01cc]: 0000  MUX:0 DS:0 --- --- -- --
PAD(116) [01d0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(117) [01d4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(118) [01d8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(119) [01dc]: 0000  MUX:0 DS:0 --- --- -- --
PAD(120) [01e0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(121) [01e4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(122) [01e8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(123) [01ec]: 0000  MUX:0 DS:0 --- --- -- --
PAD(124) [01f0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(125) [01f4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(126) [01f8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(127) [01fc]: 0000  MUX:0 DS:0 --- --- -- --
PAD(128) [0200]: 0000  MUX:0 DS:0 --- --- -- --
PAD(129) [0204]: 0000  MUX:0 DS:0 --- --- -- --
PAD(130) [0208]: 0000  MUX:0 DS:0 --- --- -- --
PAD(131) [020c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(132) [0210]: 0000  MUX:0 DS:0 --- --- -- --
PAD(133) [0214]: 0000  MUX:0 DS:0 --- --- -- --

GPIO:
 * gpio@00601000

PinTrace:
 * ptrace@du

HART0 CSR:
[0001] fflags:        00000000    RW- SCR4 SCR5
[0002] frm:           00000000    RW- SCR4 SCR5
[0003] fcsr:          00000000    RW- SCR4 SCR5
[0c00] cycle:         16f574bd    R-- SCR4 SCR5
[0c01] time:          003ac65e    R-- SCR4 SCR5
[0c02] instret:       16f58501    R-- SCR4 SCR5
[0c80] cycleh:        00000000    R-- SCR4 SCR5
[0c81] timeh:         00000000    R-- SCR4 SCR5
[0c82] instreth:      00000000    R-- SCR4 SCR5
[0100] sstatus:       80003010    RWX ---- SCR5
[0101] stvec:         00000000    RWX ---- SCR5
[0104] sie:           00000000    RWX ---- SCR5
[0121] stimecmp:      00000000    RW- ---- SCR5
[0140] sscratch:      00000000    RW- ---- SCR5
[0141] sepc:          00000000    RW- ---- SCR5
[0144] sip:           00000000    RW- ---- SCR5
[0180] sptbr:         00000000    RWX ---- SCR5
[0181] sasid:         00000000    RWX ---- SCR5
```

```
[0590] ipic_cisv:     00000020    RWX ---- SCR5
[0591] ipic_scicsr:   00000000    RWX ---- SCR5
[0592] ipic_ipr:      00000000    RWX ---- SCR5
[0593] ipic_isvr:     00000000    RWX ---- SCR5
[0594] ipic_seoi:     00000000    RWX ---- SCR5
[0595] ipic_soi:      00000000    RWX ---- SCR5
[0596] ipic_idx:      00000000    RWX ---- SCR5
[0597] ipic_icsr:     00000000    RWX ---- SCR5
[0900] cyclew:        16f63097    RW- SCR4 SCR5
[0901] timew:         003ac83f    RW- SCR4 SCR5
[0902] instretw:      16f640e9    RW- SCR4 SCR5
[0980] cyclehw:       00000000    RW- SCR4 SCR5
[0981] timehw:        00000000    RW- SCR4 SCR5
[0982] instrethw:     ffffffff    RW- SCR4 SCR5
[0d01] stime:         003ac8a8    RW- ---- SCR5
[0d81] stimeh:        00000000    RW- ---- SCR5
[0d42] scause:        00000000    RW- ---- SCR5
[0d43] sbadaddr:      00000000    RW- ---- SCR5
[0300] mstatus:       80003216    RWX SCR4 SCR5
[0301] mtvec:         00000100    RWX SCR4 SCR5
[0304] mie:           00000000    RWX SCR4 SCR5
[0321] mtimecmp:      00000000    RW- SCR4 SCR5
[0340] mscratch:      008fdffc    RW- SCR4 SCR5
[0341] mepc:          00000000    RW- SCR4 SCR5
[0342] mcause:        00000000    RW- SCR4 SCR5
[0343] mbadaddr:      00000000    RW- SCR4 SCR5
[0344] mip:           00000000    RW- SCR4 SCR5
[0380] mbase:         00000000    RWX ---- SCR5
[0381] mbound:        00000000    RWX ---- SCR5
[0382] mibase:        00000000    RWX ---- SCR5
[0383] mibound:       00000000    RWX ---- SCR5
[0384] mdbase:        00000000    RWX ---- SCR5
[0385] mdbound:       00000000    RWX ---- SCR5
[0701] mtime:         003aca34    RW- SCR4 SCR5
[0741] mtimeh:        00000000    RW- SCR4 SCR5
[0790] ipic_cisv:     00000020    RWX SCR4 SCR5
[0791] ipic_cicsr:    00000000    RWX SCR4 SCR5
[0792] ipic_ipr:      00000000    RWX SCR4 SCR5
[0793] ipic_isvr:     00000000    RWX SCR4 SCR5
[0794] ipic_eoi:      00000000    RWX SCR4 SCR5
[0795] ipic_soi:      00000000    RWX SCR4 SCR5
[0796] ipic_idx:      00000000    RWX SCR4 SCR5
[0797] ipic_icsr:     00020300    RWX SCR4 SCR5
[07a0] tlbi_attr:     00000000    -WX ---- SCR5
[07a1] tlbi_va:       00000000    RWX ---- SCR5
[07a2] tlbd_attr:     00000000    -WX ---- SCR5
[07a3] tlbd_va:       00000000    RWX ---- SCR5
[07a4] tlbi_scan:     00000000    -WX ---- SCR5
[07a5] tlbd_scan:     00000000    -WX ---- SCR5
[07a6] tlbie_attr:    00000000    R-X ---- SCR5
[07a7] tlbie_va:      00000000    R-X ---- SCR5
```

```
[07a8] tlbde_attr:    00000000    R-X ---- SCR5
[07a9] tlbde_va:      00000000    R-X ---- SCR5
[07b0] cache_ctrl:    000000f0    RWX ---- SCR5
[07b0] mem_ctrl:      n/a         RWX SCR4
[07b1] mem_mask:      n/a         RWX SCR4
[07b2] mem_pattern:   n/a         RWX SCR4
[07b3] mem_update:    n/a         RWX SCR4
[07b4] clk_setup:     00010064    RWX SCR4 SCR5
[0f00] mcpuid:        0094112d    R-- SCR4 SCR5
[0f01] mimpid:        16038000    R-- SCR4 SCR5
[0f10] mhartid:       00000000    R-- SCR4 SCR5
[0fb0] cache_l1:      29420942    R-- ---- SCR5
[0fb1] cache_l2:      2012e953    R-- ---- SCR5
[0fc0] mrtlid:        17011301    R-X SCR4 SCR5
[0fc1] mefaddr:       n/a         R-- SCR4
[0fc2] mefcmd:        n/a         R-- SCR4
```

OK

core

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

Пример для СнК “Дружба”:

```
tmt $ core
 * HART0: Syntacore SCR4 RV32IMFDCXU

OK
```

Пример для СнК “Шершень”:

```
tmt $ core
 * HART0: Syntacore SCR5 RV32IMAFDCXSU
   HART1: Syntacore SCR5 RV32IMAFDCXSU
   HART2: Syntacore SCR4 RV32IMFDCXU

OK
```

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

```
core <ядро>
```

где <ядро> - номер ядра, которому предписывается исполнять код интерактивного монитора. Если ядра с заданным номером в системе нет, будет выдано сообщение об ошибке. Команда `core` актуальна только для многоядерной системы (СнК “Шерешень”).

Пример для СнК “Шерешень”:

```
tmt $ core 2
Now running on HART2 (Syntacore SCR4)
Type 'help' for help.

tmt $ core 4
HART4 doesnt exist in the 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.

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:  006fdffc  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 hvec
Unknown CSR name 'hvec'
```

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

Пример для СнК “Дружба”:

```
tmt $ mfcsr

[0001] fflags:     00000000    RW- SCR4 SCR5
[0002] frm:        00000000    RW- SCR4 SCR5
[0003] fcsr:       00000000    RW- SCR4 SCR5
[0c00] cycle:      00000000    R-- SCR4 SCR5
```

```
[0c01] time:       00000000    R-- SCR4 SCR5
[0c02] instret:    00000000    R-- SCR4 SCR5
[0c80] cycleh:     00000000    R-- SCR4 SCR5
[0c81] timeh:      00000000    R-- SCR4 SCR5
[0c82] instreth:   00000000    R-- SCR4 SCR5
[0300] mstatus:    00000000    RWX SCR4 SCR5
[0301] mtvec:      00000000    RWX SCR4 SCR5
[0304] mie:        00000000    RWX SCR4 SCR5
[0321] mtimecmp:   00000000    RW- SCR4 SCR5
[0340] mscratch:   00000000    RW- SCR4 SCR5
[0341] mepc:       00000000    RW- SCR4 SCR5
[0342] mcause:     00000000    RW- SCR4 SCR5
[0343] mbadaddr:   00000000    RW- SCR4 SCR5
[0344] mip:        00000000    RW- SCR4 SCR5
[0701] mtime:      00000000    RW- SCR4 SCR5
[0741] mtimeh:     00000000    RW- SCR4 SCR5
[0790] ipic_cisv:  00000000    RWX SCR4 SCR5
[0791] ipic_cicsr: 00000000    RWX SCR4 SCR5
[0792] ipic_ipr:   00000000    RWX SCR4 SCR5
[0793] ipic_isvr:  00000000    RWX SCR4 SCR5
[0794] ipic_eoi:   00000000    RWX SCR4 SCR5
[0795] ipic_soi:   00000000    RWX SCR4 SCR5
[0796] ipic_idx:   00000000    RWX SCR4 SCR5
[0797] ipic_icsr:  00000000    RWX SCR4 SCR5
[07b0] cache_ctrl: n/a         RWX ---- SCR5
[07b0] mem_ctrl:   00000000    RWX SCR4
[07b1] mem_mask:   00000000    RWX SCR4
[07b2] mem_pattern:00000000    RWX SCR4
[07b3] mem_update: 00000000    RWX SCR4
[07b4] clk_setup:  00000000    RWX SCR4 SCR5
[0f00] mcpuid:     00000000    R-- SCR4 SCR5
[0f01] mimpid:     00000000    R-- SCR4 SCR5
[0f10] mhartid:    00000000    R-- SCR4 SCR5
[0fb0] cache_l1:   n/a         R-- ---- SCR5
[0fb1] cache_l2:   n/a         R-- ---- SCR5
[0fc0] mrtlid:     00000000    R-X SCR4 SCR5
[0fc1] mefaddr:    00000000    R-- SCR4
[0fc2] mefcmd:     00000000    R-- SCR4
```

Пример для СнК “Шершень”:

```
tmt $ mfcsr

[0001] fflags:        00000000    RW- SCR4 SCR5
[0002] frm:           00000000    RW- SCR4 SCR5
[0003] fcsr:          00000000    RW- SCR4 SCR5
[0c00] cycle:         1ce63522    R-- SCR4 SCR5
[0c01] time:          0049fb7e    R-- SCR4 SCR5
[0c02] instret:       1ce64567    R-- SCR4 SCR5
[0c80] cycleh:        fffffffe    R-- SCR4 SCR5
[0c81] timeh:         ffffffff    R-- SCR4 SCR5
[0c82] instreth:      fffffffe    R-- SCR4 SCR5
```

```
[0100] sstatus:       80003010    RWX ---- SCR5
[0101] stvec:         00000000    RWX ---- SCR5
[0104] sie:           00000000    RWX ---- SCR5
[0121] stimecmp:      00000000    RW- ---- SCR5
[0140] sscratch:      00000000    RW- ---- SCR5
[0141] sepc:          00000000    RW- ---- SCR5
[0144] sip:           00000000    RW- ---- SCR5
[0180] sptbr:         00000000    RWX ---- SCR5
[0181] sasid:         00000000    RWX ---- SCR5
[0590] ipic_cisv:     00000020    RWX ---- SCR5
[0591] ipic_scicsr:   00000000    RWX ---- SCR5
[0592] ipic_ipr:      00000000    RWX ---- SCR5
[0593] ipic_isvr:     00000000    RWX ---- SCR5
[0594] ipic_seoi:     00000000    RWX ---- SCR5
[0595] ipic_soi:      00000000    RWX ---- SCR5
[0596] ipic_idx:      00000000    RWX ---- SCR5
[0597] ipic_icsr:     00000000    RWX ---- SCR5
[0900] cyclew:        1ce6f0e4    RW- SCR4 SCR5
[0901] timew:         0049fd5f    RW- SCR4 SCR5
[0902] instretw:      1ce70133    RW- SCR4 SCR5
[0980] cyclehw:       fffffffe    RW- SCR4 SCR5
[0981] timehw:        ffffffff    RW- SCR4 SCR5
[0982] instrethw:     fffffffd    RW- SCR4 SCR5
[0d01] stime:         0049fdc7    RW- ---- SCR5
[0d81] stimeh:        00000000    RW- ---- SCR5
[0d42] scause:        00000000    RW- ---- SCR5
[0d43] sbadaddr:      00000000    RW- ---- SCR5
[0300] mstatus:       80003216    RWX SCR4 SCR5
[0301] mtvec:         00000100    RWX SCR4 SCR5
[0304] mie:           00000000    RWX SCR4 SCR5
[0321] mtimecmp:      00000000    RW- SCR4 SCR5
[0340] mscratch:      008fdffc    RW- SCR4 SCR5
[0341] mepc:          00000000    RW- SCR4 SCR5
[0342] mcause:        00000000    RW- SCR4 SCR5
[0343] mbadaddr:      00000000    RW- SCR4 SCR5
[0344] mip:           00000000    RW- SCR4 SCR5
[0380] mbase:         00000000    RWX ---- SCR5
[0381] mbound:        00000000    RWX ---- SCR5
[0382] mibase:        00000000    RWX ---- SCR5
[0383] mibound:       00000000    RWX ---- SCR5
[0384] mdbase:        00000000    RWX ---- SCR5
[0385] mdbound:       00000000    RWX ---- SCR5
[0701] mtime:         0049ff54    RW- SCR4 SCR5
[0741] mtimeh:        00000000    RW- SCR4 SCR5
[0790] ipic_cisv:     00000020    RWX SCR4 SCR5
[0791] ipic_cicsr:    00000000    RWX SCR4 SCR5
[0792] ipic_ipr:      00000000    RWX SCR4 SCR5
[0793] ipic_isvr:     00000000    RWX SCR4 SCR5
[0794] ipic_eoi:      00000000    RWX SCR4 SCR5
[0795] ipic_soi:      00000000    RWX SCR4 SCR5
[0796] ipic_idx:      00000000    RWX SCR4 SCR5
```

```
[0797] ipic_icsr:     00020300    RWX SCR4 SCR5
[07a0] tlbi_attr:     00000000    -WX ---- SCR5
[07a1] tlbi_va:       00000000    RWX ---- SCR5
[07a2] tlbd_attr:     00000000    -WX ---- SCR5
[07a3] tlbd_va:       00000000    RWX ---- SCR5
[07a4] tlbi_scan:     00000000    -WX ---- SCR5
[07a5] tlbd_scan:     00000000    -WX ---- SCR5
[07a6] tlbie_attr:    00000000    R-X ---- SCR5
[07a7] tlbie_va:      00000000    R-X ---- SCR5
[07a8] tlbde_attr:    00000000    R-X ---- SCR5
[07a9] tlbde_va:      00000000    R-X ---- SCR5
[07b0] cache_ctrl:    000000f0    RWX ---- SCR5
[07b0] mem_ctrl:      n/a         RWX SCR4
[07b1] mem_mask:      n/a         RWX SCR4
[07b2] mem_pattern:   n/a         RWX SCR4
[07b3] mem_update:    n/a         RWX SCR4
[07b4] clk_setup:     00010064    RWX SCR4 SCR5
[0f00] mcpuid:        0094112d    R-- SCR4 SCR5
[0f01] mimpid:        16038000    R-- SCR4 SCR5
[0f10] mhartid:       00000000    R-- SCR4 SCR5
[0fb0] cache_l1:      29420942    R-- ---- SCR5
[0fb1] cache_l2:      2012e953    R-- ---- SCR5
[0fc0] mrtlid:        17011301    R-X SCR4 SCR5
[0fc1] mefaddr:       n/a         R-- SCR4
[0fc2] mefcmd:        n/a         R-- SCR4
```

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 hvec 0x100100
Unknown CSR name 'hvec'
```

pad

Команда “pad” позволяет управлять контроллером PAD.

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

Для СнК “Дружба”:

```
tmt $ pad
 * pad@00420000

OK
```

Для СнК “Шершень”:

```
tmt $ pad
 * pad@0061c000

OK
```

В текущей реализации начального загрузчика из 3-х существующих в СнК “Шершень” контроллеров PAD поддерживается работа только с контроллером низкоскоростной периферии (LSI_CLUSTER) с базовым адресом 0x0061C000.

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

```
pad dump
```

Пример для СнК “Дружба”:

```
tmt $ pad dump
PAD0(1) [2004]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(2) [2008]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(3) [200c]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(4) [2010]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(5) [2014]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(6) [2018]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(7) [201c]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(8) [2020]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(9) [2024]: 0000  MUX:0 DS:0 --- --- -- --
PAD(10) [2028]: 0000  MUX:0 DS:0 --- --- -- --
PAD(11) [202c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(12) [2030]: 0000  MUX:0 DS:0 --- --- -- --
PAD(13) [2034]: 0000  MUX:0 DS:0 --- --- -- --
PAD(14) [2038]: 0000  MUX:0 DS:0 --- --- -- --
PAD(15) [203c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(16) [2040]: 0000  MUX:0 DS:0 --- --- -- --
PAD(17) [2044]: 0000  MUX:0 DS:0 --- --- -- --
PAD(18) [2048]: 0000  MUX:0 DS:0 --- --- -- --
PAD(19) [204c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(20) [2050]: 0000  MUX:0 DS:0 --- --- -- --
PAD(21) [2054]: 0000  MUX:0 DS:0 --- --- -- --
```

```
PAD(22) [2058]: 0000  MUX:0 DS:0 --- --- -- --
PAD(23) [205c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(24) [2060]: 0000  MUX:0 DS:0 --- --- -- --
PAD(25) [2064]: 0000  MUX:0 DS:0 --- --- -- --
PAD(26) [2068]: 0000  MUX:0 DS:0 --- --- -- --
PAD(27) [206c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(28) [2070]: 0000  MUX:0 DS:0 --- --- -- --
PAD(29) [2074]: 0000  MUX:0 DS:0 --- --- -- --
PAD(30) [2078]: 0000  MUX:0 DS:0 --- --- -- --
PAD(31) [207c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(32) [2080]: 0000  MUX:0 DS:0 --- --- -- --
PAD(33) [2084]: 0000  MUX:0 DS:0 --- --- -- --
PAD(34) [2088]: 0000  MUX:0 DS:0 --- --- -- --
PAD(35) [208c]: 0000  MUX:0 DS:0 --- --- -- --

OK
```

Пример для СнК “Шершень”:

```
tmt $ pad dump
PAD0(1) [0004]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(2) [0008]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(3) [000c]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(4) [0010]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(5) [0014]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(6) [0018]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(7) [001c]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(8) [0020]: 0000  MUX:0 DS:0 --- --- -- --
PAD0(9) [0024]: 0000  MUX:0 DS:0 --- --- -- --
PAD(10) [0028]: 0000  MUX:0 DS:0 --- --- -- --
PAD(11) [002c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(12) [0030]: 0000  MUX:0 DS:0 --- --- -- --
PAD(13) [0034]: 0000  MUX:0 DS:0 --- --- -- --
PAD(14) [0038]: 0000  MUX:0 DS:0 --- --- -- --
PAD(15) [003c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(16) [0040]: 0000  MUX:0 DS:0 --- --- -- --
PAD(17) [0044]: 0000  MUX:0 DS:0 --- --- -- --
PAD(18) [0048]: 0000  MUX:0 DS:0 --- --- -- --
PAD(19) [004c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(20) [0050]: 0000  MUX:0 DS:0 --- --- -- --
PAD(21) [0054]: 0000  MUX:0 DS:0 --- --- -- --
PAD(22) [0058]: 0000  MUX:0 DS:0 --- --- -- --
PAD(23) [005c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(24) [0060]: 0000  MUX:0 DS:0 --- --- -- --
PAD(25) [0064]: 0000  MUX:0 DS:0 --- --- -- --
PAD(26) [0068]: 0000  MUX:0 DS:0 --- --- -- --
PAD(27) [006c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(28) [0070]: 0000  MUX:0 DS:0 --- --- -- --
PAD(29) [0074]: 0000  MUX:0 DS:0 --- --- -- --
PAD(30) [0078]: 0000  MUX:0 DS:0 --- --- -- --
PAD(31) [007c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(32) [0080]: 0000  MUX:0 DS:0 --- --- -- --
```

```
PAD(33) [0084]: 0000  MUX:0 DS:0 --- --- -- --
PAD(34) [0088]: 0000  MUX:0 DS:0 --- --- -- --
PAD(35) [008c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(36) [0090]: 0000  MUX:0 DS:0 --- --- -- --
PAD(37) [0094]: 0000  MUX:0 DS:0 --- --- -- --
PAD(38) [0098]: 0000  MUX:0 DS:0 --- --- -- --
PAD(39) [009c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(40) [00a0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(41) [00a4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(42) [00a8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(43) [00ac]: 0000  MUX:0 DS:0 --- --- -- --
PAD(44) [00b0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(45) [00b4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(46) [00b8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(47) [00bc]: 0000  MUX:0 DS:0 --- --- -- --
PAD(48) [00c0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(49) [00c4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(50) [00c8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(51) [00cc]: 0000  MUX:0 DS:0 --- --- -- --
PAD(52) [00d0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(53) [00d4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(54) [00d8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(55) [00dc]: 0000  MUX:0 DS:0 --- --- -- --
PAD(56) [00e0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(57) [00e4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(58) [00e8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(59) [00ec]: 0000  MUX:0 DS:0 --- --- -- --
PAD(60) [00f0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(61) [00f4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(62) [00f8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(63) [00fc]: 0000  MUX:0 DS:0 --- --- -- --
PAD(64) [0100]: 0000  MUX:0 DS:0 --- --- -- --
PAD(65) [0104]: 0000  MUX:0 DS:0 --- --- -- --
PAD(66) [0108]: 0000  MUX:0 DS:0 --- --- -- --
PAD(67) [010c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(68) [0110]: 0000  MUX:0 DS:0 --- --- -- --
PAD(69) [0114]: 0000  MUX:0 DS:0 --- --- -- --
PAD(70) [0118]: 0000  MUX:0 DS:0 --- --- -- --
PAD(71) [011c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(72) [0120]: 0000  MUX:0 DS:0 --- --- -- --
PAD(73) [0124]: 0000  MUX:0 DS:0 --- --- -- --
PAD(74) [0128]: 0000  MUX:0 DS:0 --- --- -- --
PAD(75) [012c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(76) [0130]: 0000  MUX:0 DS:0 --- --- -- --
PAD(77) [0134]: 0000  MUX:0 DS:0 --- --- -- --
PAD(78) [0138]: 0000  MUX:0 DS:0 --- --- -- --
PAD(79) [013c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(80) [0140]: 0000  MUX:0 DS:0 --- --- -- --
PAD(81) [0144]: 0000  MUX:0 DS:0 --- --- -- --
PAD(82) [0148]: 0000  MUX:0 DS:0 --- --- -- --
PAD(83) [014c]: 0000  MUX:0 DS:0 --- --- -- --
```

```
PAD(84) [0150]: 0000  MUX:0 DS:0 --- --- -- --
PAD(85) [0154]: 0000  MUX:0 DS:0 --- --- -- --
PAD(86) [0158]: 0000  MUX:0 DS:0 --- --- -- --
PAD(87) [015c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(88) [0160]: 0000  MUX:0 DS:0 --- --- -- --
PAD(89) [0164]: 0000  MUX:0 DS:0 --- --- -- --
PAD(90) [0168]: 0000  MUX:0 DS:0 --- --- -- --
PAD(91) [016c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(92) [0170]: 0000  MUX:0 DS:0 --- --- -- --
PAD(93) [0174]: 0000  MUX:0 DS:0 --- --- -- --
PAD(94) [0178]: 0000  MUX:0 DS:0 --- --- -- --
PAD(95) [017c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(96) [0180]: 0000  MUX:0 DS:0 --- --- -- --
PAD(97) [0184]: 0000  MUX:0 DS:0 --- --- -- --
PAD(98) [0188]: 0000  MUX:0 DS:0 --- --- -- --
PAD(99) [018c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(100) [0190]: 0000  MUX:0 DS:0 --- --- -- --
PAD(101) [0194]: 0000  MUX:0 DS:0 --- --- -- --
PAD(102) [0198]: 0000  MUX:0 DS:0 --- --- -- --
PAD(103) [019c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(104) [01a0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(105) [01a4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(106) [01a8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(107) [01ac]: 0000  MUX:0 DS:0 --- --- -- --
PAD(108) [01b0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(109) [01b4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(110) [01b8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(111) [01bc]: 0000  MUX:0 DS:0 --- --- -- --
PAD(112) [01c0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(113) [01c4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(114) [01c8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(115) [01cc]: 0000  MUX:0 DS:0 --- --- -- --
PAD(116) [01d0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(117) [01d4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(118) [01d8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(119) [01dc]: 0000  MUX:0 DS:0 --- --- -- --
PAD(120) [01e0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(121) [01e4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(122) [01e8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(123) [01ec]: 0000  MUX:0 DS:0 --- --- -- --
PAD(124) [01f0]: 0000  MUX:0 DS:0 --- --- -- --
PAD(125) [01f4]: 0000  MUX:0 DS:0 --- --- -- --
PAD(126) [01f8]: 0000  MUX:0 DS:0 --- --- -- --
PAD(127) [01fc]: 0000  MUX:0 DS:0 --- --- -- --
PAD(128) [0200]: 0000  MUX:0 DS:0 --- --- -- --
PAD(129) [0204]: 0000  MUX:0 DS:0 --- --- -- --
PAD(130) [0208]: 0000  MUX:0 DS:0 --- --- -- --
PAD(131) [020c]: 0000  MUX:0 DS:0 --- --- -- --
PAD(132) [0210]: 0000  MUX:0 DS:0 --- --- -- --
PAD(133) [0214]: 0000  MUX:0 DS:0 --- --- -- --
```

OK

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 1.

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 <num> [enable|disable|mux]
```

где

<num> — десятичный номер PAD.

enable — разрешает работу заданного PAD.

disable — запрещает работу заданного PAD. Сбрасывает регистр PAD;

mux — задает функцию мультиплексирования для заданного PAD (0..7).

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

Пример исполнения (для СнК “Шершень”):

```
tmt $ pad 119
PAD(119) [01dc]: 0000  MUX:0 DS:0 --- --- -- --

OK
tmt $ pad 119 mux 3
PAD(119) [01dc]: 0180  MUX:3 DS:0 --- --- -- --

OK
tmt $ pad 119 enable
PAD(119) [01dc]: 0181  MUX:3 DS:0 --- --- -- EN

OK
tmt $ pad 119 disable
PAD(119) [01dc]: 0180  MUX:3 DS:0 --- --- -- --
```

OK

mdw, mdb, mmw, mmb

Команды “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>
mmb <addr> <value>
```

где

addr — стартовый адрес блока данных.

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

value — записываемое значение (hex).

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 0x800 20

00000800:  02812403  02412483  03010113  00008067
00000810:  0ff0000f  01810513  f51ff0ef  fcdff06f
00000820:  00000000  3e800793  02f535b3  02f50533
00000830:  f5dff06f  00000000  000f47b7  24078793
00000840:  02f535b3  02f50533  f45ff06f  00000000

OK
tmt $ mdb 0x1000 48

00001000:  13  05  85  cf  ef  40  40  64  93  04  24  00  13  85  04  00
00001010:  ef  20  c0  6d  13  05  a0  03  ef  20  00  69  13  85  04  00
00001020:  ef  30  50  58  93  04  05  00  13  09  c0  00  63  74  99  10

OK
tmt $ mmw 0x800000 0x1000

OK
tmt $ mmb 0x800002 0x14
```

```
OK
```

При попытке обращения командой чтения/записи слов по невыровненому адресу будет выведено сообщение об ошибке:

```
tmt $ mmw 0x480003 0x1000
Unaligned access

ERROR: Command 'mmw' failed (-14)
```

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.

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

```
tmt $ uart
   [0] serial@00440000
   [1] serial@00441000
   [2] serial@00442000
   [3] serial@00443000
 * [4] serial@00414000

OK
```

spi

Команда “spi” осуществляет управление SPI NOR Flash на линии BOOT SPI.

Формат вызова команды:

```
spi probe
spi read  <offset> <addr> <length>
spi write <offset> <addr> <length>
spi erase <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 probe” выполняет запросы к SPI NOR Flash при помощи JEDEC команд идентификации 0x9F, 0xAB, 0x90, 0x4B.
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 all” осуществляет стирание всей SPI Flash (JEDEC команда 0xC7).

Пример исполнения:

```
tmt $ spi probe
SPI: spi@00415000
  [9f]: 1f 86 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  [ab]: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  [90]: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  [4b]: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

OK
tmt $ spi read 0x0 0x808000 0x400
Read 1024 bytes from 00000000@flash to 00808000@mem

OK
tmt $ spi write 0x0 0x808000 0x400
Erase flash Erased flash from 00000000@flash to 00002000@flash

Write flash Write block to 00000100
Write block to 00000200
Write block to 00000300
Written 1024 bytes from 00808000@mem to 00000000@flash

OK
```

```
tmt $ spi erase 0x10
Erased 4KiB block from 00000000@flash to 00000fff@flash

OK
tmt $ spi erase all
Erased 4KiB block from 00000000@flash to 00000fff@flash

OK
```

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

```
tmt $ spi
 * spi@00415000 flash: 001f8601 sector:4096 page:256

OK
```

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

Пример:

```
tmt $ spi read 0x0 0x910000 0x1000
End address 00911000 is out of memory

tmt $ spi read 0x0 0x8ff000 0x2000
End address 00901000 is out of memory

tmt $ spi read 0x1000000 0x880000 0x2000
Flash offset 01000000 exceeds flash space
```

xload

Команда “xload” осуществляет загрузку бинарного программного кода в оперативную память через терминал с использованием протокола X-Modem. Формат протокола обмена X-Modem приведен в источнике в разделе Литература. Для загрузки данных можно использовать стандартные функции терминальных программ (передача по протоколу X-Modem).

Загрузка программного кода с использованием данной функции допустима при скоростях обмена не выше 115200 бод. Прервать исполнение команды можно нажатием клавиш **Ctrl**+**C**.

Формат вызова команды:

```
xload <адрес начала загрузки>
```

Пример:

```
tmt $ xload 0x800200
Ready for X-Modem download to 00800200 at 9600 bps...
CCC
Start Load Addr: 00800200
End Load Addr: 00800880
Total size: 00000680 (1664 bytes)
```

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

Пример:

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

При выходе начального адреса загрузки за пределы оперативной памяти или превышение ее размера в момент загрузки вызовет сообщение об ошибке.

Пример:

```
tmt $ xload 0x900000
Address 00900000 is out of memory
```

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 LCRU, в случае его неудачи осуществляется сброс с использованием блока WDT. При невозможности сброса с использованием этих двух блоков выводится сообщение “Not implemented”.

Формат вызова команды:

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

Пример исполнения:

```
tmt $ reset soft
Perform software reset


TEMON ROM monitor. Version 1.18.Friendship2 (Jul 03 2018 - 18:20:22 +0300)
Copyright (C) 2018 Tecon MT LLC. All rights reserved.
Type 'help' for help.

tmt $
```

Вызов команды без параметров осуществляет программный сброс системы:

```
tmt $ reset
Perform software reset
```

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 0x800200 0x01 0x02 0x03 0x04
Goto fun@00800200 (00000001, 00000002, 00000003, 00000004)

System Info Example:
--------------------
Total HARTs: 1
Current HART: 0
SYS Freq: 25 MHz
RTC Freq: 100 KHz
Memory: 0x00800000-0x00900000
HARTs info:
 * HART0: SCR4 Friendship2 [16038000] RV32IMFDCX [0090112c] @ 25MHz

halt...
```

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 0x480000
Goto fun@00480000 (00000000, 00000000, 00000000, 00000000)
HART0: Unhandled exception at 0x0047ff7c (Instruction access fault)
status 0x000011b6   badaddr 0x0047ff7c
ecause 0x00000001   epc     0x0047ff7c
  00 fp  30303020   01 ra  00001da0   02 sp  0049dd4c   03 gp  0049f800
  04 tp  0049dffc   05 t0  00480000   06 t1  0000003b   07 t2  00000032
  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  00000000   17 a7  00000000   18 s2  00000000   19 s3  00480000
  20 s4  00000000   21 s5  00000000   22 s6  0049e080   23 s7  0049e000
  24 s8  00000000   25 s9  00000000   26 s10 00000000   27 s11 00000000
  28 t3  00000039   29 t4  0000000a   30 t5  00000009   31 t6  00000063
Type 'help' for help.

tmt $
```

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 0x800200 1 2 3 4

Call fun@00800200 (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”.

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:

```
tmt $ hash xxh 0x800000 0x1000

xxh[00800000-00801000]=5cf0e71d

OK
```

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

```
tmt $ hash

Algos: crc32 xxh

OK
```

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

Пример:

```
tmt $ hash crc32 0x800000 0x100000
Region 00800000-00900000 is out of RAM

tmt $ hash md5 0x800000 0x1000
Unknown algo 'md5'
```

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 SYS_ERR (System Error). Для СнК “Дружба” вывод SYS_ERR управляется через LCRU, для СнК «Шершень» вывод SYS_ERR управляется через блок Diagnostic Unit (DU). Формат вывода данных и временные характеристики сигналов приведены в разделе 4. Коды контрольных значений системы PINTRACE приведены в приложении 1.

Формат вызова команды:

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

где

код — произвольный 16 битный код.

Команда “ptrace on” разрешает работу системы PINTRACE, а вывод GPIO системы переводится в режим “output”. **Команда “ptrace off”** запрещает работу системы PINTRACE, а вывод GPIO системы переводится в режим “input”. **Команда “ptrace status”** выдает текущее состояние системы PINTRACE.

Пример исполнения:

```
tmt $ ptrace off
Pintrace disabled

tmt $ ptrace on
Pintrace enabled

tmt $ ptrace status
Pintrace enabled

tmt $ ptrace send 0x10
```

Вызов команды без параметров выводит краткую информацию о системе PINTRACE.

Пример для СнК “Дружба”:

```
tmt $ ptrace
 * ptrace@lcru
```

Пример для СнК “Шершень”:

```
tmt $ ptrace
 * ptrace@du
```

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

```
tmt $ ptrace enable
Invalid operation 'enable'
```

4. ДИАГНОСТИЧЕСКАЯ СИСТЕМА PINTRACE

Отладочный интерфейс PinTrace использует интерфейс регистра SYS_ERR (System Error) блока LCRU и DU для вывода информации о контрольных точках исполнения начального загрузчика и ошибок в виде последовательного кода. Система функционирует только в режиме интерактивного монитора.

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, информационной группы и стоп группы. Стартовая группа посылки обозначается импульсом низкого уровня длительностью 16 мс. Стоп группа обозначается импульсом высокого уровня длительностью 16мс. Каждый бит информационной группы имеет длительность 2мс, размер информационной группы составляет 16 бит. Для уравления линией SYS_ERR в СнК «Дружба-2» используется блок LCRU, в СнК «Шершень» - блок DU.

Примеры временных диаграмм передачи отладочной информации приведены на рисунке 4.1.

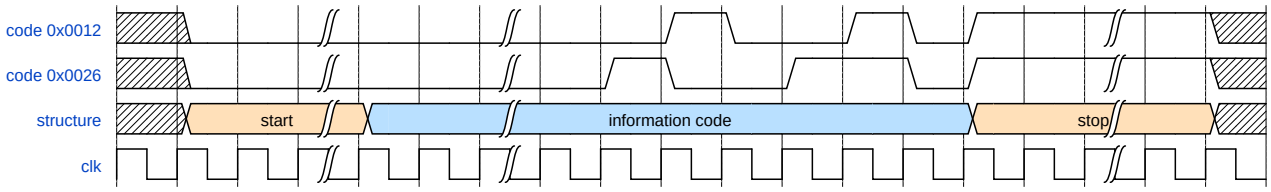

Рисунок 4.1. Временные диаграммы сигналов системы PinTrace

Перечень кодов этапов исполнения начального загрузчика и кодов ошибок представлен в приложении 1.

5. ИНТЕРФЕЙС ПРОГРАММИРОВАНИЯ ПРИЛОЖЕНИЙ (API)

Для обеспечения упрощения разработки и отладки вторичного программного обеспечения начальный загрузчик содержит подсистему обслуживания пользовательского API (Application Programming Interface). Состав представляемого функционала позволяет получить базовую информацию о системе, осуществлять ввод и вывод информации через отладочную системную консоль и использовать функции системного таймера.

Системные вызовы API архитектурно зависимы и базируются на RISC-V вызове mcall (системный вызов в Machine Mode). При этом регистр A7 содержит номер вызываемой функции, а регистры A0-A6 параметры вызываемой функции. Вызываемая функция всегда возвращает код результата выполнения в регистре A0, однако функционал ряда вызовов позволяет игнорировать возвращаемое значение.

Перечень функций API и структур данных приведен в приложении 2. Определение функций на базе системных вызовов описано в заголовочных файлах traps.h и mcall.h в приложении 3.

ЛИТЕРАТУРА

- C99 ISO/IEC 9899:TC3 draft http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1256.pdf
- Linux kernel coding style https://www.kernel.org/doc/html/v4.10/process/coding-style.html
- RISC-V Instruction Set Privileged Architecture v.1.7 https://people.eecs.berkeley.edu/~krste/papers/riscv-priv-spec-1.7.pdf
- X-MODEM serial protocol https://en.wikipedia.org/wiki/XMODEM
- МРПЦ.431291.001.РЭ “Полное описание блока GPIO”
- МРПЦ.431296.008.ИЭ “Полное описание блока PAD”
- МРПЦ.431296.009.РЭ “Полное описание блока SRAM”
- МРПЦ.431291.010.РЭ “Полное описание блока SPI”
- МРПЦ.431296.011.РЭ “Полное описание блока WDT”
- МРПЦ.431291.012.РЭ “Полное описание блока UART”
- МРПЦ.431296.013.РЭ “Полное описание блока BOOTCTRL”
- МРПЦ.431296.018.РЭ “Полное описание блока LCRU”

ПРИЛОЖЕНИЕ 1. ПЕРЕЧЕНЬ КОДОВ И СООБЩЕНИЙ СИСТЕМЫ PINTRACE

Таблица. 1.1. Перечень кодов и сообщений системы PinTrace

Код
Мнемоника
Значение 3+\^
Глобальное состояние системы PinTrace
0x0002
PT_OFF
Система PinTrace отключена.
0x0006
PT_ON
Система PinTrace включена. 3+\^
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. 3+\^
Инициализация драйверов
0x0020
PT_DRV_PAD
Инициализация драйвера PAD.
0x0021
PT_DRV_GPIO
Инициализация драйвера GPIO.
0x0022

Код
PT_DRV_PTRACE
Инициализация линии PinTrace.
0x0023
PT_DRV_BOOT
Инициализация драйвера Boot Controller.
0x0024
PT_DRV_SERIAL
Инициализация драйвера UART и отладочной консоли.
0x0025
PT_DRV_SPI
Инициализация драйвера SPI.
0x0026
PT_DRV_NFC
Инициализация драйвера NFC (не задействовано).
0x0027
PT_DRV_WDT
Инициализация драйвера WDT.
0x0028
PT_DRV_LCRU
Инициализация драйвера LCRU.
0x0029
PT_DRV_I2C
Инициализация драйвера I2C (не задействовано). 3+\^
Запуск системы загрузки
0x0040 ... 0x004F
PT_INIT_START/ PT_INIT_CORE
Старт ядра 0 ... 15.
0x0080
PT_INIT_DATA
Инициализация Core Local Data.
0x0084
PT_INIT_PASSWD
Инициализация системы ограничения доступа в интерактивный монитор (не задействовано).

Код
0x0088
PT_INIT_BOOT
Определение типа загрузки, передача управления выбранной процедуре загрузки. 3+\^
Процесс загрузки
0x0100 … 0x010F
PT_INIT_START
0x0110
PT_BOOT_SPI
Загрузка со внутренней SPI Flash.
0x0111
PT_BOOT_NFC
Загрузка с NAND Flash (не задействовано).
0x0112
PT_BOOT_UART
Загрузка с последовательного интерфейса.
0x0120 … 0x012F
PT_BOOT_START
Старт загрузки сегмента 0 … 15.
0x0130
PT_BOOT_MEDIA
Невозможно обратиться к загрузочному устройству.
0x0140
PT_BOOT_TIMEOUT
Таймаут ожидания загрузочного устройства.
0x0142
PT_BOOT_HCRC
Ошибка контрольной суммы заголовка загрузочного сегмента.
0x0144
PT_BOOT_DCRC
Ошибка контрольной суммы данных загрузочного сегмента.
0x0180
PT_BOOT_ERROR

Код
Общая ошибка загрузки. 3+\^
Системные события
0x0200
PT_MONITOR
Старт интерактивного монитора.
0x0800
PT_HANG
Остановка системы.
0x1000 … 0x100F
PT_EXCEPTION
Аппаратное исключение. Номер исключения — последний байт кода.

ПРИЛОЖЕНИЕ 2. ПЕРЕЧЕНЬ ФУНКЦИЙ API И ПАРАМЕТРОВ ИХ ВЫЗОВОВ

Таблица. 2.1. Перечень функций API

Код
Функция
Описание
1
api_hart_id
Получение номера ядра на котором исполняется код. **Входные параметры:** отсутствуют. **Выходные параметры:** номер ядра на котором исполняется код.
2
api_num_harts
Получение количества ядер в системе. **Входные параметры:** отсутствуют. **Выходные параметры:** количество ядер в системе.
3
api_console_putchar
Вывод символа в отладочную консоль. **Входные параметры:** код символа. **Выходные параметры:** отсутствуют.
4
api_console_getchar
Ввод символа с отладочной консоли. **Входные параметры:** отсутствуют. **Выходные параметры:** код символа, или отрицательное значение в случае, если входной буфер пуст.
5
api_reset
Сброс системы. **Входные параметры:** отсутствуют. **Выходные параметры:** отсутствуют.
6
api_halt
Остановка системы. **Входные параметры:** отсутствуют. **Выходные параметры:** отсутствуют.
7
api_query_mach_info

Код
Получение общей информации о системе. **Входные параметры:** указатель на область памяти, зарезервированной для структуры machine_info. **Выходные параметры:** результат исполнения команды, обновляются данные структуры machine_info.
8
api_query_hart_info
Получение информации об определенном ядре системы. **Входные параметры:** номер ядра, указатель на область памяти, зарезервированной для структуры hart_info. **Выходные параметры:** результат исполнения команды, обновляются данные структуры hart_info.
9
api_get_random
Получение псевдо-случайного числа. **Входные параметры:** отсутствуют. **Выходные параметры:** псевдо-случайное число.
10
api_timer_start
Старт таймера. **Входные параметры:** отсутствуют. **Выходные параметры:** отсутствуют.
11
api_timer_stop
Остановка таймера. **Входные параметры:** отсутствуют. **Выходные параметры:** отсутствуют.
12
api_timer_get
Получение значения таймера. **Входные параметры:** отсутствуют. **Выходные параметры:** интервал работы таймера в микросекундах.
13
api_send_msg
Отправка сообщения определенному ядру системы. **Входные параметры:** номер ядра, идентификатор сообщения. **Выходные параметры:** код результата обработки сообщения.
14
api_mdelay

Код
Пауза в работе системы (мс). **Входные параметры:** время паузы в миллисекундах. **Выходные параметры:** отсутствуют.
15
api_sleep
Пауза в работе системы (сек). **Входные параметры:** время паузы в секундах. **Выходные параметры:** отсутствуют.

Таблица. 2.2. Структура данных machine_info

Тип данных
Название поля
Описание
uint32_t
hartnum
Количество ядер в системе.
uint32_t
sys_freq
Частота системного таймера в Гц.
uint32_t
rtc_freq
Частота таймера RTC в Гц.
uint32_t
flags
Флаги платформы.
uint32_t
mem_base
Базовый адрес начала области памяти.
uint32_t
mem_size
Размер области памяти.

Таблица. 2.3. Структура данных hart_info

Тип данных
Название поля

Тип данных
Описание
uint32_t
hartid
Номер ядра системы.
uint32_t
impid
Implementation ID.
uint32_t
cpuid
CPU ID (RISC-V).
uint32_t
flags
Флаги ядра.
uint32_t
cpu_freq
Тактовая частота ядра.
uint32_t
vm_info
Информация о виртуальной памяти.
uint32_t
l1i_cache_info
Информация о L1 Instruction Cache.
uint32_t
l1d_cache_info
Информация о L1 Data Cache.
uint32_t
l2_cache_info
Информация о L2 Cache.
char
cpu_name[16]
Текстовое название ядра.
char
impl_str[16]

Тип данных
Текстовое название исполнения.

ПРИЛОЖЕНИЕ 3. ЛИСТИНГ ОПРЕДЕЛЕНИЙ ФУНКЦИЙ API НА БАЗЕ СИСТЕМНЫХ ВЫЗОВОВ

3.1. Файл traps.h

```
#ifndef _ASM_RISCV_TRAPS_H
#define _ASM_RISCV_TRAPS_H

#include <asm/mcall.h>

#define MCALL_HART_ID             0
#define MCALL_NUM_HARTS           1
#define MCALL_CONSOLE_PUTCHAR     2
#define MCALL_CONSOLE_GETCHAR     3
#define MCALL_RESET               4
#define MCALL_HALT                5
#define MCALL_GET_MACH_INFO       6
#define MCALL_GET_HART_INFO       7
#define MCALL_GET_RANDOM          8
#define MCALL_TIMER_START         9
#define MCALL_TIMER_STOP         10
#define MCALL_TIMER_GET          11
#define MCALL_SEND_MSG           12
#define MCALL_MDELAY             13
#define MCALL_SLEEP              14

#ifndef __ASSEMBLY__

struct hart_info_s {
 unsigned int hartid;
 unsigned int impid;
 unsigned int cpuid;
 unsigned int flags;
 unsigned int cpu_freq;
 unsigned int vm_info;
 unsigned int l1i_cache_info;
 unsigned int l1d_cache_info;
 unsigned int l2_cache_info;
 char cpu_name[16];
 char impl_str[16];
};
typedef struct hart_info_s hart_info_t;

struct machine_info_s {
 unsigned int hartnum;
 unsigned int sys_freq;
 unsigned int rtc_freq;
 unsigned int flags;
 unsigned int mem_base;
 unsigned int mem_size;
};
```

```
typedef struct machine_info_s machine_info_t;

struct hart_msg_s {
 unsigned int hart;
 unsigned int msg;
};
typedef struct hart_msg_s hart_msg_t;

static inline
int api_hart_id(void)
{
 return do_mcall0(MCALL_HART_ID);
}

static inline
int api_num_harts(void)
{
 return do_mcall0(MCALL_NUM_HARTS);
}

static inline
void api_console_putchar(char ch)
{
 do_mcall1(MCALL_CONSOLE_PUTCHAR, ch);
}

static inline
int api_console_getchar(void)
{
 return do_mcall0(MCALL_CONSOLE_GETCHAR);
}

static inline
void api_reset(void)
{
 do_mcall0(MCALL_RESET);
}
static inline
void api_halt(void)
{
 do_mcall0(MCALL_HALT);
}

static inline
int api_query_mach_info(machine_info_t *p)
{
 return do_mcall1(MCALL_GET_MACH_INFO, (unsigned long)p);
}

static inline
int api_query_hart_info(unsigned int hart, hart_info_t *p)
```

```
{
 return do_mcall2(MCALL_GET_HART_INFO, hart, (unsigned long)p);
}

static inline
unsigned int api_get_random(void)
{
 return do_mcall0(MCALL_GET_RANDOM);
}

static inline
int api_timer_start(void)
{
 return do_mcall0(MCALL_TIMER_START);
}

static inline
int api_timer_stop(void)
{
 return do_mcall0(MCALL_TIMER_STOP);
}

static inline
unsigned int api_timer_get(void)
{
 return do_mcall0(MCALL_TIMER_GET);
}

static inline
int api_send_msg(unsigned int hart, unsigned int msg)
{
 return do_mcall2(MCALL_SEND_MSG, hart, msg);
}

static inline
int api_mdelay(unsigned int ms)
{
 return do_mcall1(MCALL_MDELAY, ms);
}

static inline
int api_sleep(unsigned int sec)
{
 return do_mcall1(MCALL_MDELAY, sec);
}

#endif /* __ASSEMBLY__ */
#endif /* _ASM_RISCV_TRAPS_H */
```

3.2. Файл mcall.h

```
#ifndef _ASM_RISCV_MCALL_H
#define _ASM_RISCV_MCALL_H

#ifndef __ASSEMBLY__

static inline unsigned long do_mcall0(unsigned long num)
{
    register unsigned long a7 asm("a7") = num;
    register unsigned long a0 asm("a0");
    __asm__ __volatile__ ("ecall"
                          : "=r" (a0)
                          : "r" (a7)
                          : "memory");
    return a0;
}

static inline unsigned long do_mcall1(unsigned long num,
                                      unsigned long arg0)
{
    register unsigned long a7 asm("a7") = num;
    register unsigned long a0 asm("a0") = arg0;
    __asm__ __volatile__ ("ecall"
                          : "+r" (a0)
                          : "r" (a7)
                          : "memory");
    return a0;
}

static inline unsigned long do_mcall2(unsigned long num,
                                      unsigned long arg0,
                                      unsigned long arg1)
{
    register unsigned long a7 asm("a7") = num;
    register unsigned long a0 asm("a0") = arg0;
    register unsigned long a1 asm("a1") = arg1;
    __asm__ __volatile__ ("ecall"
                          : "+r" (a0)
                          : "r" (a1),
                            "r" (a7)
                          : "memory");
    return a0;
}

#else /* __ASSEMBLY__ */

.macro do_mcall num
 li a7, \num
 ecall
.endm
```

```
#endif /* __ASSEMBLY__ */
#endif // _ASM_RISCV_MCALL_H
```

ВЕРСИЯ ДОКУМЕНТА

Версия	Дата	Внесённые изменения	Автор
0.1.15	09.11.2017	Изменения из коммита af7d1fd37	Брехов С.Е.
0.1.18	12.07.2018	Добавлено описание загрузчика для СнК "Дружба-2"	Дунаев Д.М. Чикин А.В.
0.1.19	15.12.2022	Описание приведено в соответствие с используемыми загрузчиками	Дунаев Д.М.