MDB DEBUGGER

Версия 1.1
24.11.2020

ОГЛАВЛЕНИЕ

1. ОБОЗНАЧЕНИЯ

VER версия программного продукта

prompt приглашение к вводу команд. Выводится как `mdb>`

PC счетчик команд (program counter)

entry point адрес первой инструкции программы, собранной в формате ELF

target_address адрес памяти отлаживаемой платы

SoC System on Chip (Система на кристале)

2. АННОТАЦИЯ

Данный документ является руководством пользователя по применению отладчика MDB. В документе приведено назначение программы, условия её выполнения с точки зрения аппаратной и программной части, установка, режимы работы отладчика, система команд.

3. НАЗНАЧЕНИЕ ПРОГРАММЫ

Отладчик **MDB** – программа, обеспечивающая удаленную отладку программ на процессорах семейства “multicore” через эмулятор MC-USB-JTAG. В это семейство входят:

- NVCom-01;
- NVCom-02T (1892ВМ10Я);
- MC-24R (1892ВМ8Я);
- MC-24M (1892ВМ17Ф);
- LDE-Vega;
- NVCom-02 (1892ВМ11Я);
- MC-0428 (1892ВМ7Я);
- МСТ-03Р (1892ВМ12АТ);
- MC-12M (1892ВМ16Т);
- MC-226M (1892ВМ18Ф);
- MC-30SF6 (1892ВМ15АФ);
- MCom-02 (1892ВМ14Я);
- МСТ-04 (1892ВК016);
- МСТ-05 (1892ВМ196);
- МСТ-06 (1892ВМ206);
- МСТ-07 (1892ВК024).

4. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

MDB распространяется под операционные системы семейства Windows NT и дистрибутива CentOS 7.

4.1 Требования по аппаратной части

Минимальные требования к ПЭВМ для работы:

Процессор:

- Intel Core 2 Duo;
- AMD Phenom.

Оперативная память:

- Требования не предъявляются.

Видео карта:

- Требования не предъявляются.

Жесткий диск:

Количество памяти должно соответствовать требованиям, предъявляемым к операционной системе.

4.2 Требования по программной части

- Multicore JTAG Server (mjtagserver).
- Python 2.7 32bit.

Для ОС семейства Windows NT:

- USB драйвер, поддерживающий работу с libusb-1.0;

Для ОС дистрибутива CentOS 7:

- libusb-1.0.x86_64;
- libftdi-1.1.x86_64;

5. ВЫПОЛНЕНИЕ ПРОГРАММЫ

5.1 Установка

1. Установить `libusb-1.0` и `libftdi-1.1` через системную программу **`yum`** (только для ОС дистрибутива CentOS 7)
2. Установить mjtagserver

   С правами администратора, запустить

   Для ОС семейства Windows NT:

   MJTAG_server_setup_ *VER* **.exe**

   Для ОС дистрибутива CentOS 7:

   mjtagserver-installer-CentOS-7.run

Примечание: При установке в ОС CentOS 7, mjtagserver можно указать адрес, который будет использован для связи с клиентскими приложениями. Подробности в описании, доступном при вызове установщика с аргументом –help.

---

1. Установить Python 2.7 32bit (только для ОС семейства Windows NT).

5.2 Режимы работы

Отладчик MDB может работать в следующих режимах:

- режим чтения команд из файла;
- интерактивный режим;
- режим ввода-вывода.

5.2.1 Режим чтения команд из файла

Отладчик позволяет выполнять команды из файлов, при этом файлы команд могут содержать комментарии, пустые строки и “вызовы” команд из других файлов. Файл должен быть создан в текстовом формате.

Комментарии начинаются с символа ‘#’ и продолжаются до конца строки. Экранирование символа ‘#’ достигается через его дублирование (‘##’).

На одной строке должна находиться только одна команда. Пустую строку отладчик интерпретирует как комментарий.

5.2.2 Интерактивный режим

В интерактивном режиме отладчик способен выполнять одну команду из командной строки. Также для удобства пользователя реализованы следующие возможности:

1. История команд.

   Отладчик сохраняет историю команд и позволяет быстро вернуться к набранной ранее команде с помощью клавиши `Up`.

2. Завершение команды через Tab.

   При нажатии на клавишу `Tab`, отладчик завершает набор частично набранной команды или параметра. При двойном нажатии на Tab, отображаются возможные варианты продолжения.

3. Исполнение не до конца набранной команды.

   Если команда набрана не полностью, то отладчик пытается дополнить команду до полного имени и потом запустить ее. Эта возможность позволяет выполнять команды, не набирая их полностью, ускоряя работу в интерактивном режиме и повышая её комфортность.

4. Повторное исполнение команды.

   Если осуществляется перевод строки, но команда не задана (пустая линия), то отладчик повторяет выполнение предыдущей команды. Эта особенность очень удобна при исполнении отлаживаемой программы по шагам и для вывода дампа памяти.

5. Исполнение команды в интерпретаторе Shell.

   Отладчик позволяет выполнять внешние программы. Для этого нужно набрать ‘!’ перед именем команды. Например: ‘!ls -l’ распечатает содержимое текущей директории.

5.2.3 Режим потока ввода-вывода

Отладчик также способен выполнять команды из стандартного потока ввода-вывода. Такие возможности отладчика позволяют использовать его как универсальный инструмент из других программ и надстроек. Такими программами могут быть интегрированная среда тестирования или интегрированная среда разработки.

5.3 Опции командной строки при запуске отладчика

Отладчик распознает следующие опции командной строки:

`-h`
: Помощь по опциям командной строки.

`-n`
: Не подключаться к mjtagserver и не начинать работу с первым попавшимся устройстовом.

`-v`
: Подробный режим ведения лога.

-w
: Проверка загруженной программы в память из ELF файла.

-x <cmd>
: Выполнить одну команду. Эта опция может использоваться несколько раз для последовательного исполнения команд. Если команда содержит параметры, то необходимо использовать кавычки.

-c <cmd>
: Выполнить одну команду и выйти из отладчика. Если команда содержит параметры, то необходимо использовать кавычки.

-f <file>
: Выполнить команды из указанного файла, выводя результат на экран, и перейти в интерактивный режим. Если дополнительно использовать опцию *-q*, то на экране кроме ошибок ничего выводиться не будет.

-q
: Quiet mode. В этом режиме отладчик ничего не выводит на экран кроме ошибок. Эта опция может быть полезена при использовании совместно с *-f*.

-r
: Не сбрасывать процессор в первоначальное состояние при подключении.

-l <file>
: Логировать вывод в <file>.

При использовании опций *-h* и *-c* отладчик не переходит в интерактивный режим.

После запуска отладчик выводит свою версию, дату сборки, список подключённых устройств, затем выводит *prompt* .

Например:

```
mdblib version: 6.3.9<e1c2761> (Jun 11 2019)
Connecting to the default JTAG server.
Successfully connected to @/tmp/mdb.sock
List of suitable devices:
0. NVCom-02T on EZ-USB0
Opening device: EZ-USB0
mdb>
```

Далее пользователь вводит команды и отладчик выполняет их. После выполнения команды отладчик снова выводит *prompt* и ожидает ввода следующей команды.

5.4 Базовые сценарии работы

5.4.1 Подключение к первому устройству на локальной машине

```
[user@local-host mdb_tools]$ ../mdb
mdblib version: 6.5.0<28f5ce26> (Oct 16 2020)
Connecting to the default JTAG server.
Successfully connected to @/tmp/mdb.sock
List of suitable devices:
```

```
 0. MC-24R on EZ-USB0
 1. NVCom-02T on EZ-USB1
 2. MCom-02 on EZ-USB2
Opening device: EZ-USB0
mdb>
```

Примечание:

При запуске отладчика без аргументов, автоматически исполняются команды в следующем порядке:

1. *connect*
2. *listdevs*
3. *opendev* 0.

5.4.2 Подключение к MJTAG Server на удалённой машине

```
[user@local-host mdb_tools]$ ../mdb -n -x "connect remote-host:8090"
mdblib version: 6.5.0<28f5ce26> (Oct 16 2020)
Connecting to the JTAG server: remote-host:8090.
Successfully connected to remote-host:8090
mdb>
```

5.4.3 Подключение к заданному устройству

```
[user@local-host mdb_tools]$ ../mdb -n -x "connect" -x "listdevs"
mdblib version: 6.5.0<28f5ce26> (Oct 16 2020)
Connecting to the default JTAG server.
Successfully connected to @/tmp/mdb.sock
List of suitable devices:
 0. MC-24R on EZ-USB0
 1. NVCom-02T on EZ-USB1 SN: 1412047
 2. MCom-02 on EZ-USB2 SN: 1510022
mdb> opendev 2
Opening device: EZ-USB2
mdb> closedev
mdb> opendev MC-24R
Opening device: EZ-USB0
mdb> closedev
mdb> opendev SN: 1412047
Opening device: EZ-USB1
mdb>
```

5.4.4 Настройка DDR с помощью отдельно собранной программы

```
[user@local-host mdb_tools]$ ../mdb
mdblib version: 6.5.0<28f5ce26> (Oct 16 2020)
Connecting to the default JTAG server.
Successfully connected to @/tmp/mdb.sock
List of suitable devices:
 0. MCom-02 on EZ-USB0
Opening device: EZ-USB0
mdb> execute ddr_init_d.o
mdb>
```

Примечание: Для примера был взят отладочный модуль с MCom-02 *SoC*. Его DDR настраивается одной из программ: *ddr_init_d.o* и *ddr_init_pm*, которые идут в пакете MDB Tools. При вызове программы через *execute* без указания пути к ней, происходит её поиск в директориях, указанных через команду *modulepath*.

5.4.5 Запуск выбранного ядра на исполнение отлаживаемой программы до программной точки останова

```
[user@local-host mdb_tools]$ ../mdb
mdblib version: 6.5.0<28f5ce26> (Oct 16 2020)
Connecting to the default JTAG server.
Successfully connected to @/tmp/mdb.sock
List of suitable devices:
 0. MCom-02 on EZ-USB0
Opening device: EZ-USB0
mdb> execute ddr_init_d.o
mdb> loadelf ./arm-infinitive-loop.elf
Loading executable: ./arm-infinitive-loop.elf
mdb> core
 RISC0
 RISC1
 DSP0
 DSP1
*ALL
mdb> core RISC0
mdb> sbp set main
Software breakpoint number 0 is set.
mdb> run
Debug mode was requested by software.
mdb> show risc.pc
CPU.pc : 0x40000120 (0000000F)
mdb>
```

5.5 Команды отладчика

В этом разделе описываются команды отладчика, их формат и операции, которые они выполняют.

5.5.1 help

Синтаксис:

help [command]

Описание:

Выводит краткую справку по заданной команде или по всем командам, если команда не указана.

Пример:

```
mdb> help hbp
```

5.5.2 ?

Синтаксис:

? [command]

Описание:

Аналогично команде *help*.

Пример:

```
mdb> ? hbp
```

5.5.3 run

Синтаксис:

run [<addr>]

Описание:

Переводит процессор в состояние исполнения отлаживаемой программы, после чего ожидает появления отладочного события. По умолчанию, команда работает со всеми ядра. Для того чтобы задействовать только одно ядро, необходимо воспользоваться командой *core*. Если адрес запуска **addr** не задан, то исполнение программы выполняется без изменения *PC*. Если адрес задан, то в *PC* записывается значение *entry point* и процессор запускается на исполнение. В случае, когда задействованы все ядра - используется *PC* ядра с индексом 0.

Исполнение программы может быть прервано по `Ctrl-C`.

Пример:

```
mdb> run 0xbfc01000
```

5.5.4 step

Синтаксис:

step [<N>]

Описание:

Переводит ядро процессора в состояние исполнения одной или **N** инструкций и ожидает, со стороны ядра, переход в режим отладки. По умолчанию, команда работает со всеми ядра. Для того чтобы задействовать только одно ядро, необходимо воспользоваться командой *core*. При достаточно большом **N**, команда может выполняться достаточно долго. Если **N** не указан, то выполняется одна инструкция.

5.5.5 sleep

Синтаксис:

sleep [sec]

Описание:

Пауза. Эта команда может быть полезна для использования в командных файлах. Пауза может быть прервана по `Ctrl-C`.

5.5.6 execute

Синтаксис:

execute [with-timeout <timeout>] <filename>

Описание:

Загрузка и исполнение программы с проверкой exit code. В основном используется для запуска программ по настройке DDR.

timeout - задержка в ms, при окончании, которой, произойдёт прерывание исполнения программы и переход в режим отладки.

При отсутствии пути к файлу в **filename** , происходит поиск этого файла по директориям, указанным через команду *modulepath*.

5.5.7 modulepath

Синтаксис:

modulepath (add | clear) <directory>

Описание:

Изменение списка директорий, который используется для поиска модулей, исполняемых командой *execute*. Исполнение команды без аргументов - вывод текущего списка директорий.

5.5.8 reset

Синтаксис:

reset

Описание:

Сброс процессора в начальное состояние.

5.5.9 set

Синтаксис:

set (<regname> | <address> | <symbol>) <value>

Описание:

Устанавливает регистр ‘regname’ или память в заданное значение. Адрес памяти может быть задан как через отладочный глобальный символ symbol, так и явно через виртуальный 32-х битовый адрес. Формат аргументов описан в разделе *Формат регистров для команд set и show*.

Пример:

```
mdb> set 0xbfc00010 0x12344321
mdb> set RISC1.pc 0x80000000
mdb> set var1 0x4
```

5.5.10 show

Синтаксис:

show (<regname1> | <address1> | <symbol1>) (<regname2> | <address2> | <symbol2>) ...

Описание:

Отображает содержимое перечисленных регистров и памяти. Адрес памяти может быть представлен так же, как и в команде *set*. При некорректно заданном адресе выдается ошибка. Формат аргументов описан в разделе *Формат регистров для команд set и show*.

Пример:

```
mdb> show 0xbfc00010
0xbfc00010      : 0xffffffff
mdb> show DSP0.pc
PCU.PC : 0x0000 (B8480120)
mdb> show var1
var0    : 0xf
```

```
mdb> show regAndVariableThatDoesNotExist
"regAndVariableThatDoesNotExist" is neither a register nor a symbol, wrong name?
```

5.5.11 tlb

Синтаксис:

tlb

Описание:

Отображает содержимое TLB для процессоров семейства “multicore”.

Примечание: Эта команда не предназначена для работы с MCom-02.

Пример:

```
mdb> tlb
No G ASID    PM    VPN2    PFN0 C0 D0 V0    PFN1 C1 D1 V1
0 0 0x35 0x30c 0x6d5c8 0x392f0  1  0  0 0x19958  0  0  1
1 1 0x36 0x3cf 0x367d2 0x4bb2b  1  1  0 0xd0b96  1  0  0
2 1 0x2a 0xff3 0x041fd 0x30deb  1  1  1 0x448e2  1  0  0
3 1 0xf1 0xfc3 0x07b69 0xb045e  0  1  0 0x7eb15  0  0  1
4 0 0x11 0xfc0 0x35d23 0x2260a  0  0  1 0x98f06  0  0  0
5 1 0xb2 0xc0f 0x489c1 0x6ef74  0  1  0 0x7bdea  0  1  0
6 1 0xaf 0xcc3 0x35237 0x71636  0  1  0 0x3b9e1  0  0  1
7 0 0x91 0xc33 0x099ad 0xaba7e  1  0  0 0xc8fd9  0  1  1
8 0 0x75 0x0cf 0x6d76b 0xdf62a  0  0  0 0xc6166  1  0  1
9 0 0xd0 0xc00 0x224d2 0x81b2a  0  0  0 0x8da23  1  1  0
10 1 0x21 0x03c 0x3d5d7 0xb0f06  1  1  1 0x6c7fc  0  1  1
11 0 0x58 0x3ff 0x76c25 0x53f66  0  0  0 0xae2a9  0  1  0
12 1 0x67 0xf0c 0x004fd 0xe2caf  0  1  0 0x8fe5a  1  1  1
13 0 0x5a 0xc0f 0x5156c 0x40fb9  0  1  0 0xf85a6  0  0  0
14 0 0x2b 0xf33 0x62c9b 0x43678  0  0  0 0x3a175  1  0  0
15 0 0x96 0x003 0x3462a 0xbe436  1  1  1 0x61d56  0  1  1
```

5.5.12 source

Синтаксис:

source <filename>

Описание:

Исполняет команды из указанного файла (отладчик переходит в режим чтения команд из файла).

5.5.13 load

Синтаксис:

load <filename> <address>

Описание:

Загружает программу в память по адресу **address** из текстового файла **filename**.

Содержимое одной строки, в текстовом файле, имеет следующий формат:

```
32bit_word 32bit_word 32bit_word 32bit_word //address
```

где

32bit_word - 32-ух битное слово по основанию 16.

address - 32-ух битовый адрес.

Пример содержимого файла:

```
bfc06538 bfc00074 bfc00080 bfc00ff0  //bfc00060
1000ffec 00000000                    //bfc00070
```

Пример:

```
mdb> load program.txt
```

5.5.14 save

Синтаксис:

save <filename> (<address> | <symbol>) <size>

Описание:

Сохраняет содержимое памяти размером **size** в формате, описанном в команде *load*. Начало блока памяти может быть установлено как адресом **address**, так и отладочным глобальным символом **symbol**.

Пример:

```
mdb> save file.txt 0xbfc00000 0x1000
mdb> save file.txt main 0x1000
```

5.5.15 loadbin

Синтаксис:

loadbin <filename> <address>

Описание:

Загружает содержимое файла **filename** в память по адресу **address**.

5.5.16 savebin

Синтаксис:

savebin <filename> (<address> | <symbol>) <size>

Описание:

Сохраняет **size** байт памяти с адреса **address** или отладочного глобально символа **symbol** в файл **filename**.

5.5.17 loadelf

Синтаксис:

loadelf <filename>

Описание:

Загружает содержимое ELF файла **filename** в память. Записывает значение *entry point* в *PC* текущего ядра. В случае установки режима all командой *core* текущим ядром является нулевое ядро. После успешного выполнения этой команды отладчик позволяет использовать имена глобальных символов из ELF файла в командах *show* | *set* | *sbp* | *hbp* | *wp* | *save* | *savebin* | *dump*.

5.5.18 dump

Синтаксис:

dump [(<address> | <symbol>) <size>]

Описание:

Выводит шестнадцатеричный дамп памяти с адреса **address** или отладочного глобального символа **symbol**. Если **address** или **symbol** не задан, то дамп выводится с адреса следующего за последним распечатанным словом при предыдущем выполнении этой команды. **size** - размер в байтах, должен быть кратен 4. Если параметр **size** не задан, то используется значение **size** от предыдущей команды. По умолчанию значение **size** равно 256.

Пример:

```
mdb> dump 0xbfc00000 0x200
```

5.5.19 sbp

Синтаксис:

sbp (set | unset) (<vm_addr> | <symbol>) [length]

Описание:

Команда управления программными точками останова (software breakpoints). При вызове без параметров выводит информацию об установленных программных точках останова.

set - устанавливает точку останова по адресу **vm_addr** или по отладочному глобальному символу **symbol** и выводит её номер.

unset - удаляет точку останова по адресу **vm_addr** или по отладочному глобальному символу **symbol**.

length - длина инструкции на которую будет установлена точка останова. Значение длины инструкции, по умолчанию, составляет 4 байта.

Пример:

```
mdb> sbp set 0x80000000
Software breakpoint number 0 is set.
mdb> sbt unset 0x80000000
Software breakpoint number 0 is deleted.
```

5.5.20 hbp

Синтаксис:

hbp (set | unset) (<vm_addr> | <symbol>) [length]

Описание:

Команда управления аппаратными точками останова (hardware breakpoints). Все действия по манипуляции аппаратных точек останова направлены на выбранное ядро. Выбор ядра осуществляется с помощью команды *core*. Описание аргументов аналогично команде *sbp*.

5.5.21 wp

Синтаксис:

wp (set | unset) (r | w| rw) (<vm_addr1> | <symbol1>) [(<vm_addr2> | <symbol2>)]

wp clear num <N>

Описание:

Команда управления точками останова по обращению к памяти (watchpoints).

set - установить watchpoint

unset - удалить watchpoint

r - read (чтение)

w - write (запись)

rw - read/write (чтение и запись)

clear num - удаление watchpoint по номеру **N**.

Если задан только адрес **vm_addr1** или отладочный глобальный символ **symbol1**, то точка останова ставится только на этот адрес. Если указан **vm_addr2**

или **symbol2**, точка останова ставится на диапазон адресов [(**vm_addr1** | **symbol1**); (**vm_addr2** | **symbol2**)]. Все действия по манипуляции аппаратных точек останова по обращению к памяти направлены на выбранное ядро. Выбор ядра осуществляется с помощью команды *core*.

Пример:

```
mdb> wp set rw 0x80000000
Watchpoint breakpoint number 0 is set.
mdb> wp clear num 0
Watchpoint at range 0x80000000-0x80000000 is deleted.
```

Команды *hbp* и *wp* используют одни и те же аппаратные ресурсы на всех процессорах семейства “multicore”, кроме MCom-02.

5.5.22 conf

Синтаксис:

conf [(set | unset) option [value]]

Описание:

Команда управления настройками MDB. При вызове без параметров выводит список и состояние настроек.

set - установить настройку в значение “истина”

unset - сбросить настройку в значение “ложь”

value - значение настройки с типом отличным от булевого

option - название настройки

Для пользователя доступны следующие **option**:

- *blkio* - блочный режим чтения/записи (значительно ускоряет работу, если поддерживается эмулятором USB-JTAG).
- *checkwrites* - проверка успешности записи данных из ELF файла.
- *expert* - отключение, в MDB, вспомогательных функций обработки отладочных событий.
- *gdb* - включение совместимости с gdb. Опция является устаревшей и будет удалена в следующей версии MDB.
- *intdis* - отключение прерываний во время пошагового исполнения инструкций ARM процессра.
- *monitor* - включение обработки системных вызовов отлаживаемой программы.
- *noreset* - не сбрасывать процессор в изначальное состояние при подключении к плате.
- *output* - вывод в stdout.
- *pipeline* - вывод состояния конвейера текущего ядра.
- *verbose* - расширенный режим ведения лога.

- *nrst_delay* - время в миллисекундах удержания сигнала сброса в активном состоянии.

5.5.23 exit

Синтаксис:

exit

Описание:

Выход из MDB.

5.5.24 quit

Синтаксис:

quit

Описание: Аналогично команде *exit*.

5.5.25 listdevs

Синтаксис:

listdevs [<format>]

Описание:

Перечислить все доступные для отладки устройства.

Опциональный аргумент **format** определяет формат вывода. Может принимать следующие значения:

- **json** - вывод в формате json
- **default** - стандартный вывод

При указании формата json выводятся следующие поля для каждого устройства:

- *index* - номер устройства, который начинается с нуля;
- *adapterName* - имя адаптера, например, ELVEES-USB;
- *serialNumber* - серийный номер устройства (тип - строка);
- *status* - текущий статус устройства, это поле может принимать следующие значения:
  - *READY* - устройство готово для работы;
  - *BUSY* - устройство занято в другой сессии отладки;
  - *NOT_POWERED* - устройство не подключенно к источнику питания;
  - *NOT_CONNECTED* - к адаптеру не подключен отладочный модуль;
  - *FAILURE* - процесс автоидентификации завершился ошибкой;
- *name* - название отладочного модуля;

- *error* - сообщение об ошибке в случае неудачи процесса автоидентификации, иначе это поле отсутствует;
- *idcode* - JTAG IDCODE отладочного модуля (тип - шестнадцатиричное число в виде строки); отсутствует при статусе *BUSY*;

Пример:

```
mdb> listdevs
List of suitable devices:
0. NVCom-02T on EZ-USB0
```

Пример печати списка устройств в формате json:

```
mdb> listdevs json
 [
  { "index": 0, "adapterName": "EZ-USB35", "serialNumber": "1412049", "status":
↪"READY", "name": "MC-0428", "idcode": "0x40777001" },
  { "index": 1, "adapterName": "EZ-USB33", "serialNumber": "1401001", "status":
↪"BUSY", "name": "MCom-02", "error": "server-side: EZ-USB33 is already opened" }
 ]
```

5.5.26 opendev

Синтаксис:

opendev [<num>|<pattern>]

Описание:

Выбрать устройство, отладка которого будет производится.

num - номер устройства в списке, выведенным listdevs.

pattern - часть строки, состоящей из следующих разделенных пробелами элементов: номер устройства, завершенной точкой, имя адаптера, строка “SN: ”, серийный номер адаптера, имя устройства. Регистр не важен.

Пример:

```
mdb> opendev 0
Opening device: EZ-USB0
```

Пример открытия устройства по имени:

```
mdb> opendev mcom-02
Opening device: EZ-USB0
```

5.5.27 closedev

Синтаксис:

closedev

Описание:

Освободить текущее отлаживаемое устройство.

5.5.28 setflash

Синтаксис:

setflash [address]

Описание:

Указать адрес, с которого начинается флеш-память. Команда необходима для корректной работы записи во флэш-память. Адрес по умолчанию для всех поддерживаемых процессоров, на данный момент, кроме Mcom-02 - 0xbc000000.

Предупреждение: Эта команда является устаревшей и будет убрана в последующих версиях MDB.

5.5.29 clearflash

Синтаксис:

clearflash <base_address> [sector_address]

Описание:

Если задан **sector_address**, стереть сектор флеш-памяти, находящийся по адресу **base_address** + **sector_address**. Иначе стереть всю флеш-память, начинающуюся с адреса **base_address**.

Предупреждение: Эта команда является устаревшей и будет убрана в последующих версиях MDB.

5.5.30 testflash

Синтаксис:

testflash [base_address [data_size]]

Описание:

Проверка флеш-памяти, размера **data_size**, с указанием адреса **base_address**, с которого она начинается.

Предупреждение: Эта команда является устаревшей и будет убрана в последующих версиях MDB.

5.5.31 testmem

Синтаксис:

testmem <vm_addr> <size> [passno]

Описание:

Проверка памяти, размера **size**, начиная с виртуального адреса **vm_address**.

passno – количество итераций проверки.

5.5.32 trace

Синтаксис:

trace

trace <trace_source> <filename>

trace ep <endpoint_index> <new_filename>

Описание:

Управление выводом трассировки. Вызов без параметров отображает текущие потоки вывода трассировки. Команда с названием файла **filename** из источника трассы **trace_source** помещает вывод трассы в этот файл. Источник трассы - программный модуль *mdbmon*. *mdbmon* линкуется при сборке отлаживаемой программы. Вывод осуществляется через функцию `printf`. Формат трассы определяется в первом аргументе `printf`. Команда с индексом потока вывода трассы **endoint_index** и названием файла **new_filename** перенаправляет вывод трассы в этот файл. Индекс потока вывода трассы можно получить при запуске команды без параметров.

Пример:

```
mdb> trace mdbmon traceLog.txt
mdb> trace
Stream Id:      File:
2               traceLog.txt
mdb> trace ep 2 newTraceLog.txt
mdb> trace
Stream Id:      File:
2               newTraceLog.txt
```

5.5.33 core

Синтаксис:

core [<coreName>]

Описание:

Выбрать ядро процессора, отладка, которого, будет осуществляться. Такие команды как *sbp*, *hbp*, *wp*, *step*, *run*, команды обращающиеся к памяти будут работать с текущим ядром. Преобразование виртуального адреса в физический

происходит на основе правил трансляций адресов выбранного ядра. После исполнения таких команд, как *step* и *run*, будет выведено состояние конвейера для текущего ядра. Вызов без параметров выведет список доступных ядер.

coreName - имя ядра.

Пример:

```
mdb> core
RISC0
RISC1
DSP0
DSP1
*ALL
mdb> core RISC0
mdb> core
*RISC0
 RISC1
 DSP0
 DSP1
 ALL
```

5.5.34 connect

Синтаксис:

connect

connect <host>:<port>

connect (<pipe> | <local_socket>)

Описание:

Подключиться к mjtagserver'у по адресу **host:port** по протоколу TCP/IP, либо через именованный канал **pipe** (под Windows), либо через доменный сокет **local_socket** (под Linux). Вызов без параметров - подключение к серверу на локальном компьютере.

5.5.35 loaddesc

Синтаксис:

loaddesc [<file>]

Описание:

Загрузить xml-файл с описанием процессоров. При вызове без параметров, будет использовано описание, вшитое в MDB.

5.5.36 version

Синтаксис:

version

Описание:

Вывести информацию о номере версии и дате сборке MDB.

5.5.37 dumpcache

Синтаксис:

dumpcache [<filename>]

Описание:

Сохранить информацию о кеше данных и инструкций в файлы с названиями icache_**filename**, itag_**filename**, dcache_**filename**, dtag_**filename**.

Примечание: Эта команда не предназначена для работы с MCom-02.

5.5.38 invdcache

Синтаксис:

invdcache

Описание:

Сбросить локальный кэш регистров процессора в MDB.

5.5.39 jtagspeed

Синтаксис:

jtagspeed <Hz>[K|M] [<delay-cycles>]

Описание:

Установить на эмуляторе USB-JTAG частоту работы JTAG в Гц и, если поддерживается эмулятором, коэффициент задержки обрабатываемых команд. Частота может быть указана в КГц, постфикс **K**, и в МГц, постфикс **M**. Исполнение команды без аргументов - вывод текущего состояния эмулятора USB-JTAG.

Пример:

```
mdb> jtagspeed 1000000
mdb> jtagspeed 12M 200
```

5.5.40 py

Синтаксис:

py <statement>

Описание:

Исполнить строку Python кода.

Пример:

```
mdb> py print "sum result: {}".format(2+3)
sum result: 5
```

5.5.41 pb

Синтаксис:

pb <python_statement>|<keyword>

Описание:

Добавить строку Python кода в исполнительный блок. Исполнительный блок (python block) - конструкция, оборачивающая код Python. python block инициализируется с использованием *python* в качестве **keyword**. Закрытие python block и запуск на исполнения кода, связанного с ним, достигается через передачу *end* как значения **keyword**.

Повторное выполнение команды **pb** *python* до закрытия python block приведет к отмене предыдущих команд **pb**.

Пример:

```
mdb> pb python
mdb> pb def print_hello():
mdb> pb   print 'Hello world.'
mdb> pb print_hello()
mdb> pb end
Hello world.
```

6. ФОРМАТ РЕГИСТРОВ ДЛЯ КОМАНД SET И SHOW

Синтаксис:

<device_name>.<reg_name>

Описание:

device_name - имя устройства, к которому относится регистр.

reg_name - имя регистра.

Пример:

```
DSP0.PC
dsp0.pc
MFBSP0.CSR
```

7. СООБЩЕНИЯ ОПЕРАТОРУ

Таблица 7.1. Описание ошибок и способы их устранения

Контекст	Описание	Решение
`Error: server-side: unknown command`	MJTAG Server принял неизвестный ему пакет.	Обновить MJTAG Server до последней версии.
`Error: failed to enter into debug state`	Ядро находится в невалидном состоянии и не может отвечать на запросы отладчика. В это состояние оно могло попасть из-за исполнения отлаживаемой программы.	Проверьте корректность работы программы по отношению к ядру и переферии. Убедитесь, что присутствуют все обработчики ошибок и прерываний.
`Error: memory operation timeout`	Память на плате, к которой обращается отладчик, не отвечает на обращения по записи или по чтению. Это может быть связано с неподходящими настройками для этого типа памяти. Так же, возможно, что отлаживаемая программа перевела ядро и отладочную логику в невалидное состояние.	- Убедитесь, что память настроена верно. - Проверьте, что отлаживаемая программа не приводит к зависанию ядра.
`Error: verification failure (the mismatched address is` *target_address* `)`	Данные из памяти не соответствуют тому, что было записано при загрузке ELF файла.	Убедитесь, что память настроена верно.
`Error: an overrun detection error, lower the adapter frequency to avoid such errors`	Рассинхронизация между частотами JTAG и APB или AHB шинами на *SoC* .	Понизьте или выставите рекомендованную частоту, необходимую для отладки.

8. ПРИМЕЧАНИЯ

При доступе к памяти, отладчик производит преобразование адресов из виртуальных в физические. Поэтому пользователь должен использовать виртуальные адреса.

Большинство команд отладчика могут быть прерваны по `Ctrl-C`. Если команда в это время ожидает останова CPU, то отладчик запрашивает режим отладки. Если команда была введена с ошибками, то отладчик выведет формат этой команды, которого необходимо придерживаться.

Программный модуль *mdbmon* интегрирован в библиотеку *newlib*, поставляемую вместе с инструментами сборки для процессоров семейства “multicore”.

АЛФАВИТНЫЙ УКАЗАТЕЛЬ