V8MON 使用手册

版本：V 4.1

珠海欧比特宇航科技股份有限公司

广东省珠海市港湾大道白沙路 1 号(邮编: 519080)

电话: 0756-3391979　　传真: 0756-3391980

http://www.myorbita.net

目　录

1 介绍

1.1 概述

V8MON是一种针对SPARC V8架构系列处理器的PC端监视调试软件，其原理是：在PC端建立与处理器内部DSU单元的直接通讯，通过人机交互的方式实现远端代理调试。

V8MON主要包括下列功能：

- 可以直接对系统的所有寄存器和内存进行读/写操作；
- 内建反汇编器和trace buffer管理单元；
- 支持程序的下载和运行；
- 断点(breakpoint)和监控点(watchpoint)管理；
- 支持与GNU调试器(gdb)的远程连接；
- 可选择RS232、以太网等多种连接方式；
- 支持与集成开发环境Orion3.0/4.0/5.0的连接。

1.2 支持平台及系统要求

V8MON目前提供如下两个操作系统平台的支持：

- Linux桌面版；
- WINDOWS (2000/NT/XP)：必须要安装Cygwin软件包，此软件包可以在安装集成开发环境Orion4.0/5.0/6.0后获得。

1.3 获得 V8MON

最新的V8MON可以从欧比特公司网站 http://www.myorbita.net 获得，V8MON可自由下载。

1.4 安装

V8MON为免安装软件，将V8MON可执行文件（v8mon.exe及其库文件）拷贝到PC机硬盘C:\orbita\cygwin\bin之后，就可以直接使用了。如果您已经安装Orion6.0集成开发环境，建议您将V8MON可执行文件拷贝到Orion的执行目录下（Windows平台为c:\orbita\cygwin\bin），否则您需要将拷贝路径添加到系统环境变量之中，这样使用时系统就能够自动找到V8MON了。

1.5 问题报告

您在使用中如果发现什么问题或者有什么建议，可以将问题报告或描述信息发送到 support@myorbita.net，我们会第一时间跟踪、解决您的问题。

2 调试原理

2.1 概述

V8MON的调试主要依赖于处理器内部的DSU调试单元，DSU调试单元可以独立于IU处理单元在AHB总线上产生读写时序，所以可以直接到访问处理器上的各种资源，V8MON通过串口、以太网、JTAG等方式与硬件系统相连，通过发送指令的方式与DSU调试单元进行交互，从而完成对硬件系统的调试。可以看到这种调试模式无需在硬件系统上烧写监控调试软件，也不需要借助硬件仿真器，甚至不需要硬件处理单元的参与。

V8MON支持LEON2核、LEON3核、LEON4核处理器类型。V8MON默认连接LEON3/4核处理器，如果选择LEON2核处理器，在连接命令添加 -leon2 选项。 V8MON与目标硬件连接结构图如下：

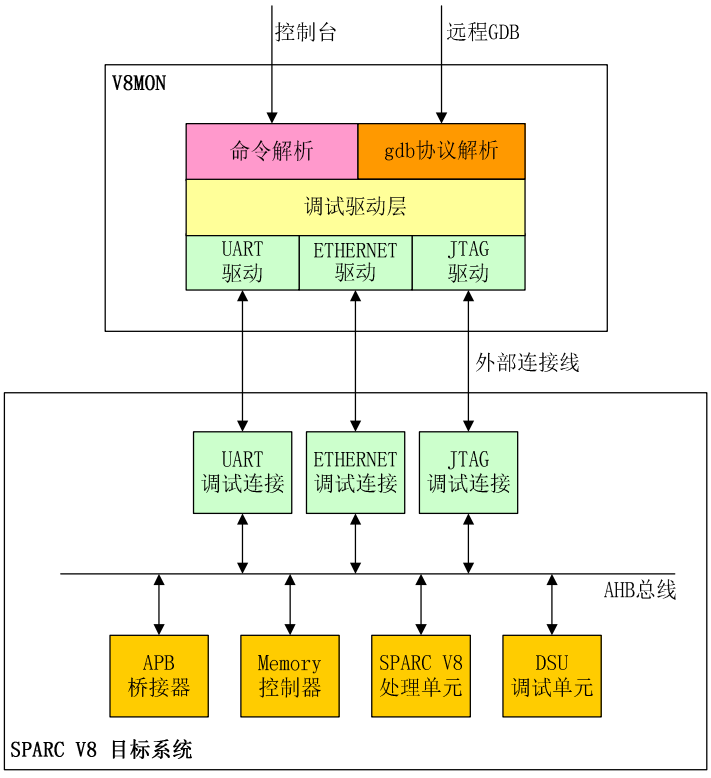

图1：V8MON连接结构图

V8MON有两种操作模式：命令行模式和GDB模式。在命令行模式下，V8MON命令通过窗口终端控制台手工输入，V8MON通过DSU协议把指令传输给目标系统DSU调试单元；在GDB模式下，V8MON作为一个网关，V8MON首先把客户端通过远程GDB协议发过来的GDB命令翻译为调试指令，再通过DSU协议把调试指令传输给目标系统DSU调试单元。

V8MON可以分为3层，包括：命令层，调试驱动层和调试接口层。

- **命令层**:由命令解析器组成，可以解析一般通用的用户命令，例如：下载、运行等。
- **调试驱动层**：实现特殊的命令，这些命令与目标处理器的构造相关。V8MON在启动的时候会扫描目标硬件，检测当前处理器类型及其配置。如果是支持的处理器，调试驱动层激活，可以执行特殊的调试命令。对于特定的处理器，这些特殊命令只包括对存储控制器的检测命令和对CPU的特殊程序调试命令。
- **接口驱动层**：实现调试连接协议（DSU protocol），通过协议与目标系统的DSU单元通讯。

2.2 目标设定初值

当V8MON启动连接目标系统时，它首先扫描系统的指定IP核（Leon2/Leon3/Leon4）是否存在，扫描过程主要是通过读写相关寄存器来完成，扫描不但可以确认硬件单元是否存在，还可以确认目标系统的某些指标参数，如系统时钟频率、SRAM大小、SRAM位宽等。扫描失败V8MON将自动退出；扫描成功，V8MON会根据用户参数对系统进行一定程度的初始化，如串口波特率、堆栈位置等。初始化完成后，V8MON将目标系统硬件配置信息打印输出到控制台。

需要注意的是：V8MON启动连接时对目标系统硬件参数的检测值和设定值不一定准确，检测值仅当参考，设定值如果存在误差用户可以通过命令直接修改。

```
$ v8mon.exe -i -u -uart /dev/ttyS1

 SPARC V8 debug monitor, version v5.3.0, build 2017/11/14

 Copyright (C)  Orbita Inc - all rights reserved.
 Comments or bug-reports to support@myorbita.net

 port /dev/ttyS1 @ 115200 baud

 initialising ..........
 LEON4 SPARC V8 processor
 LEON4 SPARC V8 processor
 LEON4 SPARC V8 processor
 LEON4 SPARC V8 processor
 processor frequency     :  99.1 MHz
 register windows        :  8
 hardware mul/div        :  yes
 floating-point unit     :  GRFPU
 instruction cache       :  4 * 8  kbytes,  4*8 bytes/line
 data cache              :  4 * 4  kbytes,  4*8 bytes/line
 hardware breakpoints    :  2
 sram width              :  32 bits
 sram banks              :  1
 sram bank size          :  8192 kbytes
 ddr2 width              :  32 bits
 ddr2 size               :  256 Mbytes
 ddr2 hz                 :  800MHz
 stack pointer           :  0x407ffff0
 target system link success!
v8mon>
```

比较详细的系统数据用户可以通过‘info sys’指令获得：

```
v8mon>info sys
1   LEON4 SPARC V8 Processor (ver 0x0)
        cpu#0:
          win 8, hwbp 2, V8 mul/div, lddel 1, GRFPU
          icache 4 * 8  kbytes,  4*8 bytes/line
          dcache 4 * 4  kbytes,  4*8 bytes/line
        cpu#1:
          win 8, hwbp 2, V8 mul/div, lddel 1, GRFPU
          icache 4 * 8  kbytes,  4*8 bytes/line
          dcache 4 * 4  kbytes,  4*8 bytes/line
        cpu#2:
          win 8, hwbp 2, V8 mul/div, lddel 1, GRFPU
          icache 4 * 8  kbytes,  4*8 bytes/line
          dcache 4 * 4  kbytes,  4*8 bytes/line
```

```
            cpu#3:
              win 8, hwbp 2, V8 mul/div, lddel 1, GRFPU
              icache 4 * 8  kbytes,  4*8 bytes/line
              dcache 4 * 4  kbytes,  4*8 bytes/line
      2   FT Memory Controller (ver 0x1)
            ahb: 0x00000000 - 0x20000000
            ahb: 0x20000000 - 0x40000000
            ahb: 0x40000000 - 0x80000000
            ahb: 0x80000000 - 0x80000100
            32-bit prom @ 0x00000000
            32-bit static ram: 1 * 8192 kbyte @ 0x40000000
      3   DDR2 Controller (ver 0x1)
            ahb: 0x60000000 - 0x80000000
            ahb: 0xFFE00000 - 0xFFE00100
            32-bit FTDDR2: 1 * 256 Mbyte @ 0x60000000  800 MHz
            stack pointer 0x407ffff0
      4   AHB/APB Bridge (ver 0x0)
            apb: 80000000 - 80100000
      5   LEON4 Debug Support Unit (ver 0x1)
            ahb: 90000000 - a0000000
      6   Generic APB UART1 (ver 0x1)
            apb: 80000100 - 80000200
            baud rate 38400
      7   Multi-processor Interrupt ctrl (ver 0x3)
            apb: 80000200 - 80000300
      8   Modular Timer Unit (ver 0x0)
            apb: 80000300 - 80000400
            16-bit scaler ,2 * 32 bit timers
      9   General purpose I/O port (ver 0x0)
            apb: 80000600 - 80000700
      10  AHB Debug UART (ver 0x0)
            apb: 80000700 - 80000800
            baud rate 115200,ahb frequency 99.1
      11  GR Ethernet MAC (ver 0x0)
            apb: 80000e00 - 80001000
            edcl ip 192.168.0.95, buffer 2 kbyte
```

3 操作说明

3.1 概述

V8MON的执行操作主要由以下三个部分组成：

- 连接目标系统并检测系统参数；
- 加载应用程序及执行用户命令；
- 与GDB工具连接，并通过GDB协议进行调试。

下面将分别描述这些过程如何被执行。

3.2 启动 V8MON

V8MON通过控制台命令行启动，启动命令为：

$ v8mon [option]

[option]为可选启动参数。在未指定启动参数的情况下，V8MON将通过串口与目标系统进行连接，默认使用调试主机的串口1（ttyS0），连接波特率为115200。

启动参数说明如下：

-baud *baudrate*	指定DSU连接串口的波特率，默认情况下为115200。可选的波特率有：9600,19200,38400,57600,115200等；波特率超过115200时目标系统和调试主机都需要使用特殊串口；
-c *batch_file*	在启动时运行批处理文件；
-ddrinit	在扫描检测ddr2之前初始化ddr2 phy；
-edac	使能Memory控制器的EDAC功能（在硬件支持的情况下）；
-eth	使用以太网连接目标系统（仅LEON3/4核支持）；
-freq *sysclk*	忽略扫描结果直接指定目标系统的总线频率，单位为MHZ；
-gdb	在启动时直接打开gdb监听服务功能；
-i	启动时初始化目标系统；
-ip *ip*	指定目标系统ip；

-leon2	强制为LEON2工作模式；
-leon3	强制为LEON3工作模式；
-leon4	强制为LEON4工作模式；
-nosram	目标系统没有SRAM；
-nddr	目标系统没有DDR2；
-port *gdbport*	指定gdb监听服务的端口号，默认值是 2222；
-stack *val*	忽略扫描结果直接指定应用软件运行时的堆栈地址；
-u	设置目标系统的UART1为回送模式，并将其输出显示在V8MON控制台上；
-uart *device*	默认情况下，V8MON使用主机端的串口1与目标系统连接，这个串口可以被重新指定。device名称依赖于主机端的操作系统，如：在Linux操作系统(或类UNIX系统)下，ttyS0表示串口1，ttyS1表示串口2，ttyS2表示串口3，依次类推；
-mp	使能多处理器模式(所有处理器使能，仅LEON3/4多核处理器支持)。

3.3 V8MON 命令说明

V8MON正常启动后，用户就可以通过V8MON命令行进行命令输入了，下面是V8MON的一些通用命令说明。

ahb *[length]*	打印指定长度的trace buffer中ahb数据信息，长度默认为10条；
ahb force *[0/1]*	使能或禁止AHB trace buffer；
ahb timer *[0/1]*	使能timetag计数；
ahb status	打印AHB trace buffer设置；
ahb filter *[addresses/reads/writes] [0/1]*	使能或禁止filter选项（仅当dsu核支持）。当使能addresses filter，第二个ahb断点寄存器将用作filter range；

ahb filter range *〈addr〉〈mask〉*

当filter addresses使能，设置需要进行filter的AHB trace buffer基地址和屏蔽码

ahb filter mmask *〈mask〉* 设置需要排除在外的AHB masters;

ahb filter smask *〈mask〉* 设置需要排除在外的AHB slaves;

ahb filter performance *[0/1]*

使能或禁止连接到performance计数器的信号过滤器;

batch *〈file_name〉* 运行一个命令批处理文件；file_name为文件名;

break *[addr]* 无参数时为打印断点信息，接参数为增加软断点；addr为断点地址;

bwatch *[addr] [mask] [read/write]*

显示或添加AHB监控点；*addr* 为监控地址，mask为地址屏蔽码，read和write为访问方向，缺省则为双向。

cont 程序在断点处往下继续运行;

cpu *[enable/disable/active] cpuid*

切换处理器状态（针对多核处理器）；没有参数时将打印各处理器的状态。

dcache *[0/1]* 无参数为显示数据cache所有内容；参数为0时，禁止数据cache，参数为1时，使能数据cache;

delete *[bpnum]* 删除断点，无参数则删除所有断点；带参数时，删除指定断点；bpnum为断点号;

disas *〈addr〉[length]* 将一段内存地址的数据反汇编后显示;

echo 在控制台上输出字符串信息;

float 显示FPU寄存器的内容;

flash 打印显示v8mon支持的flash类型;

flash *[-type] [addr]* 检测指定地址和类型的flash配置，地址默认0;

flash erase *[range | all]*

擦除指定数量的flash 分区;

flash load *<file> [addr]*	把指定的bin文件写入指定的地址中，地址默认0；
gdb *[port]*	在指定端口上启动gdb监听服务，port为端口，未指定端口则默认使用端口2222；
go *[addr]*	不初始化，从指定地址执行程序；addr为开始地址，缺省则为0x40000000；
hbreak *[addr] [mask]*	无参数时为打印断点信息，接参数为增加硬断点；addr为断点地址，mask是地址屏蔽，缺省则为0xfffffffc；
help	显示命令的使用帮助信息；
hist *[length]*	打印指定长度的trace buffer中信息，指令和AHB数据都将被打印，长度默认各为10条；
icache *[0/1]*	无参数为显示指令cache内容；参数为0时，禁止指令cache，参数为1时，使能指令cache；
info *[drivers/libs/reg/sys]*	显示目标系统中的设备驱动、硬件库、寄存器值、系统配置等信息；
inst *[length]*	打印指定长度的指令trace buffer中信息，长度默认各为10条；
inst filter*[filt]*	打印或设置指令trace buffer filter；
load *file_name*	加载文件到目标系统的内存中(文件必须为elf32格式)；
leon	打印常用leon寄存器,相当于info reg;
l2cache *[en/dis]*	使能或者关闭L2CACHE;
mem *[addr][length]*	显示指定的内存区域的内容；
mcfg1 *[value]*	设置memory寄存器1的缺省值,当执行run或者load命令时memory寄存器将被设置为这些缺省值重新;无参数则打印当前值；
mcfg2 *[value]*	设置memory寄存器2的缺省值，其它与Mcfg1命令相同；

mcfg3 *[value]*	设置memory寄存器3的缺省值，其它与Mcfg1命令相同；
mmu	打印mmu寄存器；
mmu *[mctrl/ctxp/ctx] val*	设置相关mmu寄存器的值；
quit	退出v8mon（也可以使用Ctrl+c强行退出）；
register *[win]*	显示IU寄存器；win:win0 到 win7。缺省时默认为win0；
run *[addr]*	开始执行程序，开始地址未指定则默认从程序入口地址开始执行；
shell *<cmd>*	执行shell命令；
symbols	显示符号表信息或者从文件中加载符号表；
stack *[val]*	显示或设置堆栈指针；
step *[n]*	执行单步，或执行[n]次；
tmode *[proc/ahb/both/none]*	查看或设置调试追踪功能模式,可选仅处理器、ahb数据、处理器及ahb数据、不追踪四种。
verify *<filename>*	比较文件与内存数据是否一样；
wash	清除所有SRAM地址中的内容，不支持ddr2；
watch *[addr] [mask] [read/write]*	显示或添加数据监控点；*addr* 为监控地址，mask为地址屏蔽码，read和write为访问方向，缺省则为双向。
wmem *<addr> <data>*	写指定的数据到指定的内存地址中；

以上这些命令支持缩写格式，缩写定义参见附录2。同时命令行中支持Tab功能键，在输入符号表和文件名时，用户输入部分字符后按Tab键，系统将自动匹配到全部字符。

3.4 常规调试操作

3.4.1 文件下载

通过load命令可以将一个SPARC V8体系的应用程序下载到目标系统中:

```
v8mon>lo hello.exe
[Nr]      Name      Type          Addr      Off       Size
[1]     .text       PROGBITS    40000000 00010000 00009d80
[2]     .data       PROGBITS    40009d80 00019d80 00000770
total size:42224 bytes
entry point: 0x40000000
v8mon>
```

V8MON支持的文件格式为elf32-sparc。V8MON内建有ELF解析器，可以将程序的每个功能段正确的下载到对应的链接地址上,下载完成后程序运行时的%pc指针指向程序的入口地址。

下载过程中，控制台将输出段地址、段字节大小、程序入口地址等信息。

3.4.2 程序运行

运行程序之前，必须先下载程序，而下载的程序必须确保是经过sparc-rtems或sparc-elf工具链编译得到；使用run命令可以运行已下载的程序。

```
v8mon>lo stanford.out
[Nr]      Name      Type          Addr      Off       Size
[1]     .text       PROGBITS    40000000 00010000 0000c540
[2]     .data       PROGBITS    4000c540 0001c540 00000770
total size:52400 bytes
entry point: 0x40000000
v8mon>ru
Starting
Perm  Towers  Queens  Intmm  Mm  Puzzle  Quick Bubble  Tree  FFT
0      0       0      16      0   34      0      16      0   17

Nonfloating point composite is      7
Floating point composite is         14
Program exited normally.
v8mon>
```

目标系统的UART1用来输出应用程序的打印信息，这些信息并不能从V8MON的控

制台显示，而如果使用-u参数启动V8MON，V8MON将初始化目标系统的UART1为回采模式，这样DSU单元可以读到从UART1输出的信息，进而可以在V8MON控制台上把这些信息显示出来。

程序运行过程中可以使用Ctrl+c快捷键中止运行；运行程序后如果要再次运行程序，必须再次下载程序，原因是程序的.data段数据已经被更改，需要还原为初始值。

3.4.3 插入断点和监测点

程序下载完后，可以通过break或hbreak命令设置断点，其中break命令用来添加一个软断点（ta 1），而hbreak命令是通过IU单元的watchpoint寄存器添加插入一个硬断点，在调试只读代码（如程序在ROM中运行）时只能使用硬断点。

一个断点只能作用于一个字的地址，当程序运行到此地址时，系统切换到调试模式。软断点可以设置在基于RAM的代码段的任何位置，包括禁止trap的区域（如trap处理函数中）；系统提供两个硬断点，设置硬断点将冻结系统的trace buffer。

以下示例演示软断点的相关操作。

添加断点：

```
v8mon> bre 0x40001000
v8mon> bre main
v8mon>
```

显示已有断点：

```
v8mon> bre
 num    address        type
  1 : 0x40001000    (soft)
  2 : 0x400012ec    (soft)
v8mon>
```

删除断点：

```
v8mon> del
v8mon> bre
num    address      type
v8mon>
```

3.4.4 查看寄存器

在命令行下可以随时输入reg命令查看窗口寄存器的当前值：

```
v8mon> reg

        INS       LOCALS      OUTS      GLOBALS
  0:  00000000  00000000  0000000C  00000000
  1:  00000000  40001024  40007400  00000001
  2:  00000000  40001028  00000000  00000006
  3:  00000000  00000000  00000000  00000074
  4:  00000000  00000000  00000000  00000001
  5:  00000000  00000000  00000000  00000770
  6:  403FFE00  00000000  403FFD98  00000001
  7:  40001208  00000000  400012F8  00000000

 psr: F24010C6   wim: 00000001   tbr: 40000800   y: 00000000

 pc:  40000800  91d02000  ta    0x0
 npc: 40000804  01000000  nop
```

也可以使用 reg wn命令查看指定窗口寄存器的值,其中n表示窗口号。使用float指令可以查看FPU寄存器的值（存在FPU运算单元的前提下）。

3.4.5 查看内存

可以通过mem命令查看任何内存地址上的内容，mem命令的参数包括目标地址和长度（单位为字节），这两个参数都可以缺省，但不允许存在没有目标地址而有长度的情况，允许存在没有长度而有目标地址的情况。如果符号表被加载，可以使用文本符号代替地址。

```
v8mon> mem 0x0 20
0   00000000  00000000  00000000  00000000    ................
10  00000000  00000000  00000000  00000000    ................
v8mon>
```

数据显示是将以16进制32位数据对齐方式显示，16个字节内容显示一行，每行前面为首字节地址，中间为数据内容，后面为数据对应的ASCII码。

地址参数和长度参数如果不被4整除，系统将自动修正为4的倍数。

Mem命令未接参数时，执行的长度缺省值为64，地址缺省值分两种情况：V8MON

启动后首次使用mem时为0，再次使用时为上次已显示内存段的下一个地址；此功能在连续查看时将极大的方便用户的操作，如下例：

```
v8mon> mem
0   81882fe0  81980000  1080001a  81900000    ../.............
10  91d02000  01000000  01000000  01000000    .. .............
20  91d02000  01000000  01000000  01000000    .. .............
30  91d02000  01000000  01000000  01000000    .. .............

v8mon> mem
40  a1480000  29000000  81c52204  01000000    .H..).....".....
50  a1480000  29000000  81c52300  01000000    .H..).....#.....
60  a1480000  29000000  81c52378  01000000    .H..).....#x....
70  81d82000  82100000  84100000  86100000    .. .............

v8mon> mem 0x40000000
40000000  01344578  29100004  81c52000  01000000    .4Ex)..... .....
40000010  91d02000  01000000  01000000  01000000    .. .............
40000020  91d02000  01000000  01000000  01000000    .. .............
40000030  91d02000  01000000  01000000  01000000    .. .............

v8mon> mem
40000040  a1480000  29100004  81c520dc  01000000    .H..)..... .....
40000050  a1480000  29100004  81c5202c  01000000    .H..)..... ,....
40000060  a1480000  29100004  81c52084  01000000    .H..)..... .....
40000070  91d02000  01000000  01000000  01000000    .. .............
```

可以通过disas命令将内存中的内容反汇编成SPARC机器指令进行显示，disas命令的参数形式和mem命令相同。

```
v8mon> dis 0x40000000
 40000000  a0100000   clr      %l0
 40000004  29100004   sethi    %hi(0x40001000), %l4
 40000008  81c52000   jmp      %l4
 4000000c  01000000   nop
 40000010  91d02000   ta       0x0
 40000014  01000000   nop
 40000018  01000000   nop
 4000001c  01000000   nop
 40000020  91d02000   ta       0x0
 40000024  01000000   nop
 40000028  01000000   nop
 4000002c  01000000   nop
 40000030  91d02000   ta       0x0
 40000034  01000000   nop
 40000038  01000000   nop
```

```
4000003c  01000000    nop
```

3.4.6 使用 trace buffer

处理器中的trace buffer用来保存最近的执行指令以及最近的AHB总线上的传输数据，V8MON启动时自动使能trace buffer，启动后用户也可以通过tmode命令使能或禁止trace buffer。ahb/inst/hist分别用来显示trace buffer中的内容。trace buffer在调试时可以起到辅助作用，下面是一个典型的调试实例。

```
v8mon> lo stanford.out
[Nr]      Name      Type          Addr      Off       Size
[1]     .text       PROGBITS    40000000 00010000 0000c540
[2]     .data       PROGBITS    4000c540 0001c540 00000770
total size:52400 bytes
entry point: 0x40000000
v8mon> tm both
combined processor/AHB tracing
v8mon> bre main
v8mon> watch 0x4000a500
watch 4000a500
v8mon> bre
num       address          type
  0       0x4000513c        <soft>
  1       0x4000a500        <watch>
v8mon> ru
 soft breakpoint  0  0x4000513c
v8mon> ahh
time address type data trans size burst mst lock resp tt pil irl
239371467 400042d8 read 38800002 3 2 1 0 0 0 06 0 0
239371469 400042dc read d222a100 3 2 1 0 0 0 06 0 0
239371472 4000a4fc read 00000000 2 2 0 0 0 0 06 0 0
239371480 4000a4fc write 000005d0 2 2 0 0 0 0 06 0 0
239371481 90000000 read 000055f9 2 2 0 3 0 0 06 0 0
v8mon> inst
time address instruction result
239371473 400042bc ld [%o2 + 0xfc], %o0 [00000000]
239371475 400042c0 cmp %o1, %o0 [000005d0]
239371476 400042c4 bgu,a 0x400042cc [00000000]
239371478 400042c8 st %o1, [%o2 + 0xfc] [4000a4fc 000005d0]
239371479 400042cc sethi %hi(0x4000a400), %o2 [4000a400]
```

```
v8mon> hist
254992755 40003870 sethi %hi(0x4001f800), %l0 [4001f800]
254992759 ahb read, mst=0, size=2 [40003880 94146198]
254992760 40003874 mov 19, %i0 [00000013]
254992761 ahb read, mst=0, size=2 [40003884 961423cc]
254992762 40003878 mov 256, %o0 [00000100]
254992763 ahb read, mst=0, size=2 [40003888 190fec00]
254992764 4000387c or %l2, 0x28c, %o1 [40014e8c]
254992765 ahb read, mst=0, size=2 [4000388c 7fffff5f]
254992766 40003880 or %l1, 0x198, %o2 [40014598]
254992767 ahb read, mst=0, size=2 [40003890 9a102000]
254992769 ahb read, mst=0, size=2 [40003894 b0863fff]
254992771 40003884 or %l0, 0x3cc, %o3 [4001fbcc]
254992772 40003888 sethi %hi(0x3fb00000), %o4 [3fb00000]
254992773 4000388c call 0x40003608 [4000388c]
254992774 40003890 mov 0, %o5 [00000000]
```

当打印执行指令时，左边第一列是相对时间（指令完成的时间），第二列为指令地址，第三列为指令内容，最右边括号内的表示指令执行的数据结果，如果是存储指令则括号内的为存储地址以及存储数据。在进行Trace模式切换时，不能保障trace buffer中的记录有效，除非重新执行程序或者trace buffer自动溢出。

3.4.7 应用程序打印输出

如果V8MON带-u参数启动，目标系统的串口1将被设置成回采模式，并使能流控制。在应用程序运行时，串口1的接收缓冲将被DSU接管，所有应用程序运行过程中的的打印信息得以在V8MON控制台上显示,这个时候无需再在主机上开一个串口接收器接收目标系统串口1的数据观察运行输出。

此时，有以下两点约束（即针对应用程序编程,也针对用户命令行操作）：

1. 串口1的控制寄存器不允许被更改，比如禁止回采模式；
2. 串口1的接收寄存器不允许读操作；

这就意味者-u参数不适用于由mkprom生成的prom映像以及linux操作系统映像，因为它们在启动代码里对UART1进行了专门设置。

3.4.8 连接目标系统

使用-i参数启动V8MON时，将首先探测系统配置并初始化内存和寄存器。有些情况下并不需要初始化启动，如在定位系统故障，或者在断开后重新连接时，此时必须去掉-i参数，启动后最好手工初始化内存相关寄存器以及程序运行堆栈。

3.4.9 多处理器支持

在多核处理器系统中，可以使用cpu命令来切换各处理器的调试状态。在对称多核处理器中，处理器的编号为0～n-1，n表示处理器个数。每个处理器可以设置为使能、禁止两种状态，处理器使能后，将响应run、cont、go等命令。当处理器禁止时，将处于挂起状态，不响应V8MON的任何指令。使用cpu enable n 命令将使能对应处理器，cpu disable n 命令将禁止对应处理器。系统允许同时有多个处理器被使能，但同时只能有一个处理器被激活。所有的调试命令如查看寄存器、增加断点等只对被激活的处理器有效，这就允许非并行处理程序在多核系统中执行。用户可以使用cpu active n 命令来选择对应处理器激活；V8MON启动后，系统默认为：处理器0被使能并激活，其它处理器被禁止；使用-mp启动V8MON时，系统将默认使能所有处理器。

```
v8mon> cpu
cpu 0: enabled active
cpu 1: disabled
cpu 2: disabled
cpu 3: disabled
v8mon> cpu en 1
cpu 0: enabled active
cpu 1: enabled
cpu 2: disabled
cpu 3: disabled
v8mon> cpu act 1
cpu 0: enabled
cpu 1: enabled active
cpu 2: disabled
cpu 3: disabled
```

可以使用gdb来调试多处理器系统，gdb连接后，当前被激活的处理器负责接收

gdb命令。选择其它处理器激活时要断开gdb连接，**注意**：断开并重连后，gdb仍然会保存之前的断点设置。

3.5 符号表信息

应用文件下载后，V8MON将自动提取文件中的符号表信息，这些符号可以在一些用户命令中使用，用来表示对应内存地址，如下：

```
v8mon> break main
breakpoint 1 main (0x40001ac8)
v8mon> dis strlen 3
40001e4c 808a2003 andcc %o0, 0x3, %g0
40001e50 12800016 bne 0x40001ea8
40001e54 96100008 mov %o0, %o3
```

命令symbols用来从备选文件中提取符号表或者显示当前的符号表信息。

```
v8mon> symbols samples/hello
read 71 symbols
v8mon> symbols
0x40000000 L _trap_table
0x40000000 L start
0x4000102c L _window_overflow
0x40001084 L _window_underflow
0x400010dc L _fpdis
0x400011a4 T _flush_windows
0x400011a4 T _start
0x40001218 L fstat
...
```

从用来从备选文件中提取符号表的功能适用在执行文件本身并不包含符号表的情况，例如bootproms程序以及linux/uClinux应用程序都不包含符号表。

3.6 GDB 调试

3.6.1 原理及作用

对gdb的支持是V8MON的重要功能，它为使用V8MON进行调试提供了另外一种模式。

Gdb是GNU软件序列中的调试工具，Gdb可以调试各种程序，包括C、C++、JAVA、PASCAL、FORAN和一些其它的语言，支持现有的大部分处理器平台，即可用于上位机软件调试，也可用于嵌入式软件交叉调试。gdb运用广泛，使用简捷，目前已经作为许多集成开发环境内嵌调试器，Orion系列集成开发环境就是采用gdb＋V8MON的模式实现了图形化交叉调试。与V8MON配合使用的gdb版本为sparc-rtems-gdb。

V8MON通过标准socket接口与gdb连接，V8MON为socket服务器端，gdb为客户端。V8MON和gdb可以在同一个PC机上运行，也可以在两台不同的PC上运行，这两台机器之间通过以太网相连；连接后用户可以在主机上用gdb对目标系统进行调试，V8MON在目标系统与gdb之间进行命令转换，起到桥梁纽带作用。

连接原理框图如下:

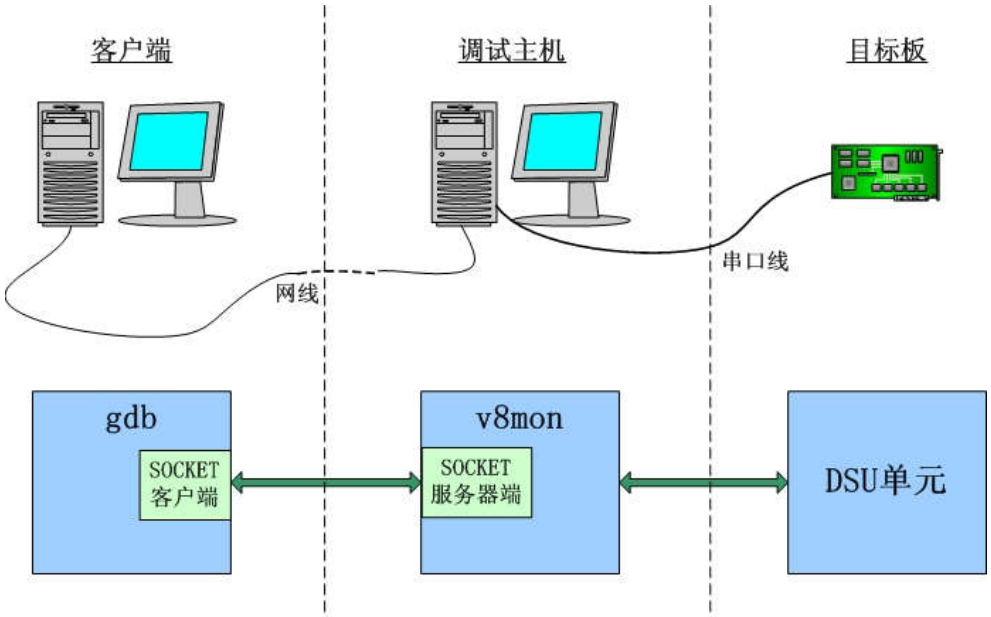

图2：V8MON GDB模式调试原理框图

借助gdb来调试比用V8MON命令行直接调试有以下好处：

1. 用户可以在高级语言层面调试应用程序,如查看变量,源代码级单步执行等；而命令行方式只能进行汇编指令级调试，能够观察的是寄存器的值；
2. 可以进行远程调试,理论上通过以太网用户可以在任何一个地方对目标系统进行调试。同时允许多个使用不同PC机的用户通过远程连接分时使用的方式

调试同一个目标系统，此功能适用于目标系统不方便移动的情况。

因此可以看出，采用gdb调试更关注于软件本身，较适合对大型应用软件的调试，而采用命令行方式方便实现对硬件资源的直接访问，便于硬件除错。

3.6.2 连接 gdb

V8MON可以被当作服务端与gdb连接，V8MON启动gdb服务监听后，将不响应用户的命令行输入。添加-gdb参数启动V8MON或者在命令行中输入gdb命令都可以将V8MON切换到gdb服务监听模式，默认的服务端口为2222。

```
$ v8mon.exe -u -gdb

 SPARC V8 debug monitor, version v1.6.0, build 09/11/2009

 Copyright (C) 2009,2012 Orbita Inc - all rights reserved.
 Comments or bug-reports to support@myorbita.net

 port /dev/ttyS0 @ 115200 baud

 initialising ..........
 processor frequency     :  19.8 MHz
 register windows        :  8
 hardware mul/div        :  yes
 floating-point unit     :  unkown FPU
 instruction cache       :  2 * 16  kbytes,  32 bytes/line
 data cache              :  2 * 8  kbytes,  16 bytes/line
 hardware breakpoints    :  4
 sram width              :  32 bits
 sram banks              :  2
 sram bank size          :  2048 kbytes
 sdram                   :  none
 stack pointer           :  0x403ffff0
 ŏ target system link success!
gdb port 2222
Listening on port 2222
```

另外打开一个命令行终端，启动sparc-rtems-gdb并通过2222端口连接v8mon。

```
$ sparc-rtems-gdb6.4
GNU gdb 6.4
Copyright 2005 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "--host=i686-pc-cygwin --target=sparc-elf".
(gdb) file stanford.out
Reading symbols from /cygdrive/d/hardware/samples/stanford.out...done.
(gdb) tar extended-remote :2222
Remote debugging using :2222
0x00000000 in ?? ()
(gdb)
```

3.6.3 调试应用程序

通过gdb调试应用程序，同样也是要分为下载、设置断点、运行等几步操作,；以下是典型的gdb调试实例:

```
(gdb) lo
Loading section .text, size 0xc540 lma 0x40000000
Loading section .data, size 0x770 lma 0x4000c540
Start address 0x40000000, load size 52400
Transfer rate: 28850 bits/sec, 508 bytes/write.
(gdb) bre main
Breakpoint 1 at 0x40005140: file stanford.c, line 1033.
(gdb) ru
The program being debugged has been started already.
Start it from the beginning? (y or n) y

Starting program: /cygdrive/d/hardware/samples/stanford.out

Breakpoint 1, main () at stanford.c:1033
1033        fixed = 0.0;
(gdb) cont
Continuing.
Starting
Perm  Towers  Queens  Intmm  Mm  Puzzle  Quick Bubble  Tree  FFT
0      0       0      16      0   34      0      16      0   17

Nonfloating point composite is      7
Floating point composite is         14
```

```
Program exited normally.

Program received signal SIGSEGV, Segmentation fault.
0x40000800 in text_start ()
(gdb)
```

调试过程中可以使用Ctrl+c快捷键中止调试，**注意**再次运行程序时必须要重新下载应用程序。在gdb模式下，所有断点以软断点的方式进行处理。详细的gdb命令说明请参照gdb操作手册。

3.6.4 断开 gdb

如果在gdb中使用detach命令，gdb将断开与V8MON的连接，V8MON将切换到命令行接收模式，这时可以在V8MON中对应用程序继续调试，也可以重新切换到gdb服务监听模式等待gdb连接。

当应用程序运行过程中出错产生trap时，程序停止运行，trap信息传递给V8MON，V8MON再将SPARC trap转换成操作系统信号传递给gdb，此时用户可以通过gdb来查看目标系统的内存和寄存器进而判断trap产生的原因。

3.6.5 限制条件

Gdb客户端建议使用sparc-rtems-gdb6.4版本，否则可以会存在版本不兼容的问题导致调试异常。

另外使用gdb时不要在程序的中断处理程序中设置断点，因为这里trap被禁止，堆栈指针无效，gdb会一直停留在错误的堆栈窗口内反复操作。

3.7 FLASH 操作

V8MON支持LEON2/LEON3系统下兼容CFI接口的flash PROM操作，包括flash检测，擦除和程序烧写。PROM总线宽度可以为8bit/16bit/32bit，由MCFG1寄存器控制。

当前支持的flash有：

sst(8-BIT)　：39LF/VF512,39LF/VF010,39LF/VF020,39LF/VF040;

intel(16-BIT): E28F128J3A;

amd(16-BIT) : AM29DL32XD;

atmel(8/16-BIT) : AT49BV/LV32X,AT49BV/LV32XT。

下面是V8MON的FLASH操作命令说明，更多的命令列表请参见附录1。

flash 打印显示v8mon支持的flash类型;

flash *<-type> [addr]* 检测指定地址和类型的flash配置，地址默认0;

flash erase *[range|all]* 擦除指定数量的flash 分区;

flash load *<file> [addr]* 把指定的bin文件写入指定的地址中，地址默认0;

对flash(如sst)进行操作的时候，常用命令顺序为:

flash

flash –sst //如果flash是挂载到0x10000000地址，flash –sst 0x10000000

flash erase 10

flash load hello.bin // flash load hello.bin 0x10000000

4 接口形式

默认情况下，V8MON通过主机的串口于目标系统连接，同时V8MON也提供JTAG、以太网等接口与目标系统连接，选择非串口方式需要添加以下的启动参数：

-eth 使用以太网连接：使用直连网线；

-jtag 使用JTAG接口连接：使用Xilinx并行电缆线（III或IV型）(未实现)。

以太网连接，默认连接IP地址为：192.168.0.51。如果需要指定IP地址，则用-ip参数指定。如：v8mon.exe –i –u –eth –ip 192.168.0.XXX， 其中XXX的范围为0~255。

当V8MON通过串口连接目标系统时，目标系统的DSU串口波特率将自动匹配V8MON设置的主机波特率。在部分主机环境下，高波特率通讯将导致通讯不稳定，此时建议降低通讯波特率，用户可以通过启动参数-baud更改波特率为57600、38400等，如使用。

附录： V8MON 命令描述

A1. 通用命令

命令	描述	命令缩写	参数缺省说明
batch <batchfile>	执行一个批处理命令文件<batchfile>	bat	
disassemble [addr] [cnt]	将地址[addr]处的[cnt]条机器指令反汇编显示	dis	addr:0x40000000 count:0x10
help [cmd]	显示有效命令或者指定命令[cmd]的用法		显示所有命令
info [drv|libs|reg |sys]	显示目标系统中的设备驱动、硬件库、寄存器值、系统配置等信息		显示所有信息
init	重新初始化目标系统		
load <file>	下载可执行文件<file>到内存中，文件必须是ELF32格式	lo	
l2cache [en|dis]	使能或者关闭L2CACHE		
mcfg1 [value]	显示或设置memory寄存器1的缺省值		显示寄存器值
mcfg2 [value]	显示或设置memory寄存器2的缺省值		显示寄存器值
mcfg3 [value]	显示或设置memory寄存器3的缺省值		显示寄存器值
mem [addr][count]	显示从地址[addr] 开始的[count]个字节的数据		addr:0x40000000 count:0x10
shell <command>	执行一个shell命令<command>		
symbols [symbol_file]	显示符号表信息或者从文件[symbol_file]中加载符号表		
quit	退出v8mon并返回shell命令行	q	
verify <file>	检查内存中的数据是否和文件<file>一致		
version	显示v8mon的版本号		
wash	清空sram内的所有数据		
wmem <addr> <data>	写数据<data>到内存地址<addr>处		

说明：

1. <>括号表示不可缺省项；
2. []括号表示可缺省项；
3. [|]表示仅括号中的参数可选择；
4. 使用wmem命令修改memory寄存器后还应使用mcfg命令设置缺省值，不然程序运行时寄存器值将被恢复，操作示例如下：

```
v8mon>wmem 0x80000004 0xe4f
v8mon>mcfg2 0xe4f
 mcfg2 = 0x00000e4f
v8mon>
```

A2. 调试命令

命令	描述	命令缩写	参数缺省说明
ahb [trace_length]	显示AHB tracebuffer数据		trace_length：10
break [addr]	显示所有断点信息或者在地址[addr]上插入软断点	bre	显示断点信息
bwatch [addr] [mask] [read|write]	显示所有地址观察点信息或者增加新地址观察点[addr]		显示地址观察点
cpu [enable|disable|active] cpuid	显示或切换处理器状态（使能/禁止/激活）		显示处理器状态
cont	继续执行		
dcache [0|1]	显示数据cache中的内容；设置dcache		
delete [bp]	删除序号为[bp]的断点或者删除所有断点	del	删除所有断点
float	显示FPU寄存器的内容		
gdb [port]	在指定端口[port]上启动gdb监听服务		port：2222
go <addr>	不初始化，从指定地址<addr>开始执行程序		addr：0x40000000
hbreak [addr] [mask]	显示所有断点信息或者在地址[addr]上插入硬断点		显示断点信息
hist [trace_length]	显示[trace_length]条trace历史记录		trace_length：10
icache [0|1]	显示数据cache中的内容,设置icache		
inst [trace_length]	显示[trace_length]条指令trace		trace_length：10
inst filter [filt]	打印或设置trace buffer filter		
leon	显示leon功能寄存器		
mmu	打印mmu寄存器		
register [reg|win]	显示IU寄存器或窗口寄存器	reg	显示所有寄存器
run	运行已下载的应用程序	ru	
stack [addr]	设置下次运行的堆栈指针		
step [n]	单步运行[n]步		n：1
tmode[ahb|proc|both|none]	显示或设置trace模式，有ahb数据、处理器、处理器及ahb数据、不追踪四种模式	tm	显示当前模式
Watch [addr] [mask] [read|write]	显示所有观察点信息或者增加新观察点[addr]		显示所有观察点

说明：

1. <>括号表示不可缺省项；
2. []括号表示可缺省项；
3. [|]表示仅括号中的参数可选择；
4. 使用命令go 0可以执行烧写在flash中的程序。