

BSP 修改和简化研究

SWD 聂勇

版 本 历 史

版本/状态	责任人	起止日期	备注
V1.0/正式	聂勇	20Oct2010	BCM 平台下 BSP 文档的修改和简化
V1.1/正式	聂勇	12Nov2010	添加调试错误总结

目 录

1. 说明.....	3
2. 控制变量总结.....	3
2.1 CONFIG.H 和 CONFIGALL.H 中的控制变量.....	3
2.2 MAKEFILE 中的控制变量.....	3
2.3 未确定定义位置变量.....	3
3. POST-PROCESSED OUTPUT.....	4
4. 问题总结.....	4
5. 编译调试错误总结.....	4
5.1 MIPS 汇编标签问题.....	4
5.2 加载 vxWORKS 时异常错误.....	5
5.3 无法发出数据包.....	7
5.3.1 前期调试.....	8

1. 说明

本文档为在 BSP 开发调试过程中常用到的调试思路和方法的研究报告。

主要参考资料：

《Understanding the bootrom image》 WIND RIVER 内部资料 AppNote-237

《JTAG 使用研究_MIPS_NY》 调试器使用的研究报告

2. 控制变量总结

控制变量主要在两个部分定义，第一是 BSP 的源文件，主要是 config.h 和 configAll.h 文件中，第二就是在 Makefile 文件中定义。

下面就以在简化 BSP 的过程中，按照遇到变量的顺序，将其陈列总结如下。

2.1 config.h 和 configAll.h 中的控制变量

BUILD_BOOTROM 这个宏是否有定义呢？在 romInit.s 文件和 et_vx.c 文件的 et_send() 函数中使用到了，而且好像确实使其中的代码有效了？

最重要的就是 et_send() 函数中，涉及到发送包的 tag 头的添加与否。

et_vx.c

```
.....  
#ifndef BUILD_BOOTROM  
memcpy(mTag->pCIBlk->clNode.pCIBuf+sizeof(brcmTag),vlanTag,sizeof(vlanTag));  
        mTag->mBlkHdr.mLen += sizeof(vlanTag);  
#endif  
.....
```

2.2 Makefile 中的控制变量

VX_VERSION 定义为 55，表示 vxWorks 的版本号位 5.5

2.3 未确定定义位置变量

ECCMODE 在 romReboot 处，有关这个变量的判断。

romInit.s

```
.....  
#ifdef ECCMODE  
#error "ECCMODE defined"
```

```
    bal    romClearEdac
    nop
    b      6f
    nop
    #endif
.....
```

在 configNet.h 文件中, 有一个宏 INCLUDE_ET_END 的存在控制着 ET_LOAD_FUNC 这个注册函数指针。但是没有发现 INCLUDE_ET_END 的定义, 而相反, 倒有 INCLUDE_ET0_END 和 INCLUDE_ET1_END 的定义。这两个定义, 将 ET_LOAD_FUNC 指向了 sysEtEndLoad()函数而不是 et_load()函数。

这里有什么区别呢?

3. post-processed output

4. 问题总结

在 EPI 下面, 如果让代码一直运行 (go), 最后机器为什么会重启呢? 总会再一次跳转到所设置的断点处。

5. 编译调试错误总结

5.1 MIPS 汇编标签问题

编译命令:

文件: romInit.s(modified)

```
.....

ccmips -c -g -G 0 -mno-branch-likely -mips2 -EB -ansi -fno-builtin -P -xassembler-with-cpp
-DMIPSEB -DSOFT_FLOAT -I.. -IE:\Tornado2.2.1-mips\target\config\bcm4704
-IE:\Tornado2.2.1-mips\target\h -IE:\Tornado2.2.1-mips\target\h\wrn\coreip
-IE:\Tornado2.2.1-mips\target\config\comps\src -IE:\Tornado2.2.1-mips\target\src\drv
-DCPU=MIPS32 -DTOOL_FAMILY=gnu -DTOOL=sfgnu -DPRJ_BUILD
-D_WRS_KERNEL E:\Tornado2.2.1-mips\target\config\bcm4704\romInit.s -o
romInit.o

.....
```

输出错误:

ccmips.....

```
.....  
C:\\DOCUME~1\\ADMINI~1\\LOCALS~1\\Temp\\ccwJaaaa.s: Assembler messages:  
C:\\DOCUME~1\\ADMINI~1\\LOCALS~1\\Temp\\ccwJaaaa.s:15: Error: backw. ref to  
unknown label "1:", 0 assumed.  
make: *** [romInit.o] Error 1  
.....
```

解决办法

问题出在这个代码段上，这本来是一个宏定义，如果将标签 **1:** 和后面的指令 **nop** 分开写成两行，则会报上面的错误。出现这种错误的原因，初步猜测是这个汇编器对某些情况下（例如宏中的）的标签的处理不能够完成。

```
romInit.s  
.....  
#define HANG    \  
1:  
    nop        ;\    需要与标签 1:写在同一行  
    b 1b  
.....
```

5.2 加载 vxWorks 时异常错误

在加载 vxWorks 的时候，出现异常错误如下图所示。

在 bootrom 命令行使用命令 **@**手动加载 vxWorks 时出现的错误如下所示：

```

Tera Term - COM1 VT
File Edit Setup Control Help

[UxWorks Boot]: ^C
[UxWorks Boot]: @

boot device      : et
unit number     : 0
processor number : 0
host name       : 192.168.0.79
file name       : vxWorks
inet on ethernet (e) : 192.168.0.1:ffffff00
host inet (h)    : 192.168.0.79
user (u)        : bcm
ftp password (pw) : bcm
flags (f)       : 0x1000

Attaching interface lo0... done
Failed to attach to device <null>@sysToMonitor
sysGetCount3d196418
sysGetCompare2f17db62
intCRGet00008000
ReadEPC81c5a24
ReadBUA00000000
ReadERRPC00000000
vxSpGet81f72580
SR = 0x1000fc00
=====

--- Stack Trace ---
FUNC = 0x81c06400, PC = 0x81c0645c <0x81c0645c>, SP = 0x81f725a0
FUNC = 0x81c02108, PC = 0x81c026bc <0x81c026bc>, SP = 0x81f725c0
FUNC = 0x81c2d0e8, PC = 0x81c2d858 <0x81c2d858>, SP = 0x81f729b8
RA 0x0 out of range

```

启动机器之后，自动加载时出现的异常如下图所示：

产生这个异常的地点还没有找到。从 config.h 文件的角度来看，是因为 bcm5836 的 MAC1 是链接到交换机的 MAC 芯片上，而如果只是使用 bcm5836 的 MAC0 来下载 vxWorks 镜像，那么需要配置号 MAC0。

具体如何配置 MAC0，通过下面的方法解决这个问题。

Config.h

```

#undef BOOT_ET 1

#ifndef BOOT_ET 1
#define INCLUDE_ET 0_END
#else
#define INCLUDE_ET 1_END
#endif

```

configNet.h

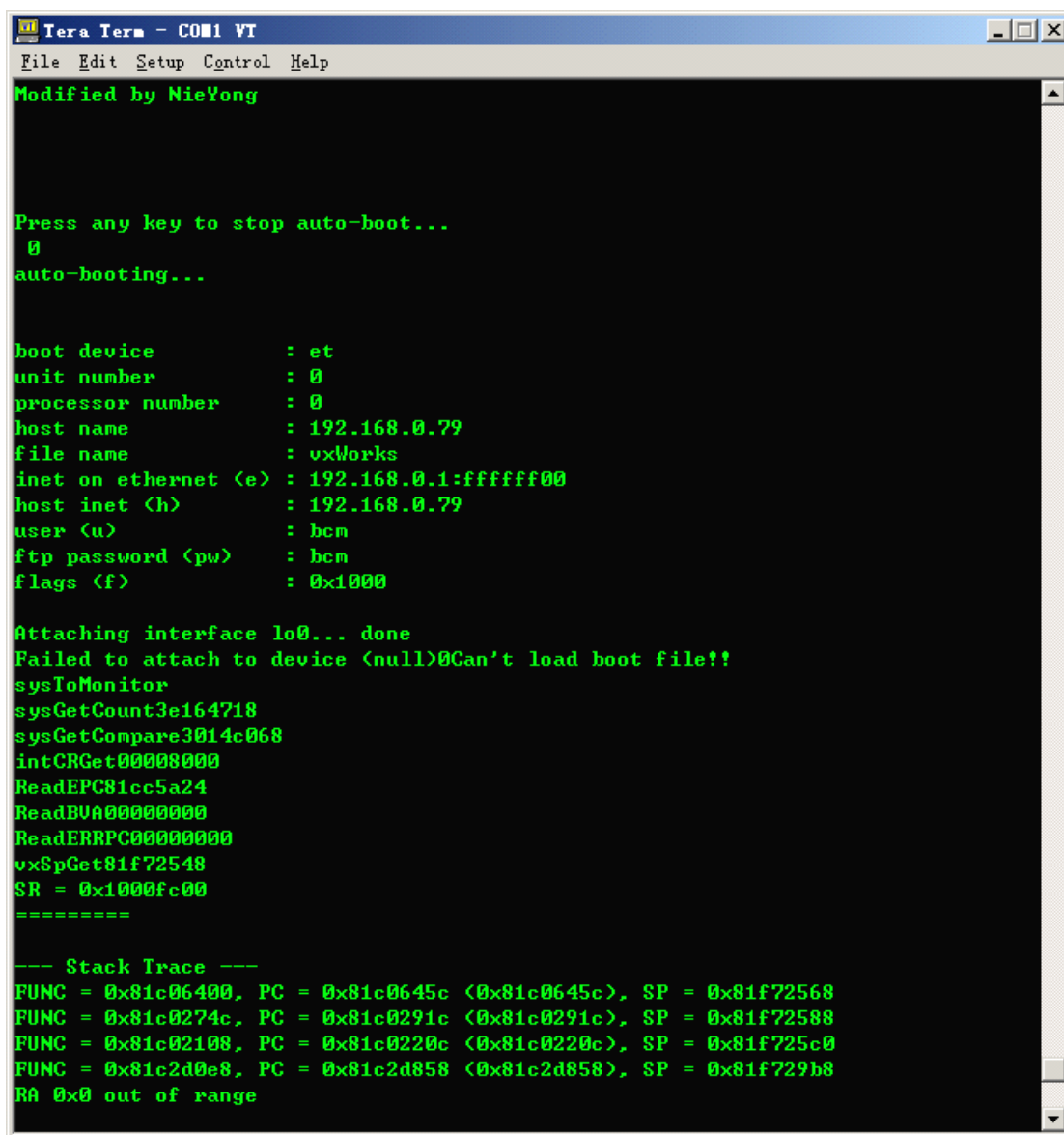
```

END_TBL_ENTRY endDevTbl[] =
{
    #ifndef INCLUDE_ET 0_END
    { 0, ET_LOAD_FUNC,
      ET_LOAD_STRING, 0, NULL, FALSE },
    #endif
}

```

config.h 文件中不定义宏 BOOT_ET1，这样导致 INCLUDE_ET1_END 未定义，而 INCLUDE_ET0_END 定义。在 configNet.h 文件中，就会导致 END_TBL_ENTRY endDevTb1[] 中定义 MAC0 的条目而没有 MAC1 的条目。

END_TBL_ENTRY endDevTb1[] 这个条目，就决定了网络驱动在 vxWorks 中的注册情况。



```
Tera Term - COM1 VT
File Edit Setup Control Help
Modified by NieYong

Press any key to stop auto-boot...
0
auto-booting...

boot device      : et
unit number     : 0
processor number : 0
host name       : 192.168.0.79
file name       : vxWorks
inet on ethernet (e) : 192.168.0.1:ffffff00
host inet (h)    : 192.168.0.79
user (u)        : bcm
ftp password (pw) : bcm
flags (f)       : 0x1000

Attaching interface lo0... done
Failed to attach to device <null>0Can't load boot file!!
sysIoMonitor
sysGetCount3e164718
sysGetCompare3014c068
intCRGet00008000
ReadEPC81cc5a24
ReadBUA00000000
ReadERRPC00000000
vxSpGet81f72548
SR = 0x1000fc00
=====

--- Stack Trace ---
FUNC = 0x81c06400, PC = 0x81c0645c <0x81c0645c>, SP = 0x81f72568
FUNC = 0x81c0274c, PC = 0x81c0291c <0x81c0291c>, SP = 0x81f72588
FUNC = 0x81c02108, PC = 0x81c0220c <0x81c0220c>, SP = 0x81f725c0
FUNC = 0x81c2d0e8, PC = 0x81c2d858 <0x81c2d858>, SP = 0x81f729b8
RA 0x0 out of range
```

5.3 无法发出数据包

在解决了 5.2 中的异常错误之后，有出现了新的情况，那就是 bootLoad() 的过程中，在 MAC0 口上，没有数据包发出。串口显示情况如下所示：

```
Loading...
Error loading file: errno = 0x3c.
sysIoMonitor
sysGetCountd9428248
sysGetComparecb40ec36
intCRGet00008000
ReadEPC81cc5954
ReadBUA00000000
ReadERRPC00000000
vxSpGet81f72580
SR = 0x1000fc00
=====

--- Stack Trace ---
FUNC = 0x81c06460, PC = 0x81c064bc <0x81c064bc>, SP = 0x81f725a0
FUNC = 0x81c02108, PC = 0x81c026cc <0x81c026cc>, SP = 0x81f725c0
FUNC = 0x81c2d018, PC = 0x81c2d788 <0x81c2d788>, SP = 0x81f729b8
RA 0x0 out of range
```

使用抓包工具对 host 机器上的网口进行监视，没有发现从来自开发板的 MAC0 的数据包。这样可以断定，虽然显示了 loading……的字样，但是数据包没有从 MAC0 端口发送出来。

我们可以看到有一个错误码 0x3c，通过查找我们也可以知道，这个 vxWorks 的错误码的含义是超时。可以初步推断，因为数据包没有发送出去，所以等待超时。错误码定义如下：

Errno.h

.....

```
#define ETIMEDOUT 60      /* Connection timed out */
```

.....

5.3.1 前期调试

在遇到这个问题之后，前期主要做的工作有下面一些。

第一是分析了函数 netLoad()函数的代码，这个函数被函数 bootLoad()函数调用，主要功能有：初始化 ftp 的相关代码，主要调用函数 ftpXfer()函数。然后是调用了函数 bootLoadModule()函数。

第二是跟踪了发包的整个过程，从整体上来说，主要是下面几个函数。可以肯定的一点是，在跟踪调试的过程中，发现所有的这些函数都依次被调用执行了。


```

File Edit Setup Control Help
2 of usrNetworkInit
3 of usrNetworkInit
Loading...
Error loading file: errno = 0x41.
sysToMonitor
sysGetCountb91242b2
sysGetCompareab10b504
intCRGet00008000
ReadEPC81cc5954
ReadBUA00000000
ReadERRPC00000000
vxSpGet81f72580
SR = 0x1000fc00
=====
--- Stack Trace ---
FUNC = 0x81c06460, PC = 0x81c064bc <0x81c064bc>, SP = 0x81f725a0
FUNC = 0x81c02108, PC = 0x81c026cc <0x81c026cc>, SP = 0x81f725c0
FUNC = 0x81c2d018, PC = 0x81c2d788 <0x81c2d788>, SP = 0x81f729b8
RA 0x0 out of range

[UxWorks Boot]:

```

```

Tera Term - COM1 VT
File Edit Setup Control Help
[UxWorks Boot]: N
Unrecognized command. Type '?' for help.
[UxWorks Boot]: p

boot device      : et
unit number     : 0
processor number : 0
host name       : 192.168.0.79
file name       : vxworks
inet on ethernet <e> : 192.168.0.88:ffffff00
host inet <h>    : 192.168.0.79
user <u>        : ny
ftp password <pw> : ny
flags <f>       : 0x100

[UxWorks Boot]: $

boot device      : et
unit number     : 0
processor number : 0
host name       : 192.168.0.79
file name       : vxworks
inet on ethernet <e> : 192.168.0.88:ffffff00
host inet <h>    : 192.168.0.79
user <u>        :

```

