文章编号:1671-4598(2018)05-0102-05 DOI:10.16526/j.cnki.11—4762/tp.2018.05.026 中图分类号:TP277 文献标识码:A

基于龙芯 1E1F 航天应用平台与 VxWorks 系统的 VxBus 型驱动设计

邹玉龙，刘 彬，田小莉，郭宗芝，牛 崇
（上海微小卫星工程中心，上海 201210）

摘要：分析了 VxWorks 系统中 VxBus 型驱动的组成和一般设计过程，对串口、时钟等 VxWorks 系统自带 VxBus 型驱动的设备进行了配置，对龙芯 1E 芯片的中断控制器驱动进行了分层设计，在 VxWorks 原有的 MIPS 中断控制器驱动基础上，增加了专门针对龙芯 1E 和龙芯 1F 的中断控制器驱动，并且 3 个中断控制器驱动通过设备配置文件相互关联，实现了中断服务程序的嵌套调用；对龙芯 1F 接口芯片上的智能 1553B 功能单独设计了一个 VxBus 型驱动，并通过设计驱动方法的方式向应用层提供了驱动的使用接口，屏蔽了底层硬件细节，简化了应用层的设计；实验结果表明在龙芯 1E1F 航天应用平台上进行 VxBus 型驱动设计使得软件结构更加清晰，系统移植的难度大大降低，并且简化了应用层软件设计，对航天领域基于龙芯和 Vxworks 系统的开发设计具有较高的参考价值。

关键词：龙芯处理器；VxWorks；VxBus

Research on VxBus Driver Design Based on Loongson 1E1F Aerospace Platform and VxWorks System

Zou Yulong, Liu Bin, Tian Xiaoli, Guo Zongzhi, Niu Chong
(Shanghai Engineering Center for Microsatellites , Shanghai 201210, China)

Abstract: Analyzed the structure of VxBus driver and the normal design procedure of VxBus driver on VxWorks system. Configured the VxBus diver which is already supported by VxWorks, such as the serial and the clock driver. The design of interrupt is divided into three layers, the MIPS interrupt, the Loongson 1E interrupt and the Loongson 1F interrupt. The three interrupt layers are nested and connected by board support package (BSP) hardware configuration. Designed a smart 1553B VxBus dirver to use the 1553B function supported by Loongson 1F. Application layer can call the driver methods to perform 1553B function easily and ignore the hardware details. The result shows that VxBus design on Loongson aerospace platform simplify the software design of application layer and makes the software structure much more clear and the system planting easier. The VxBus driver design method has high value for aerospace engineering based on Loongson and VxWorks system.

Keywords: Loongson processor; VxWorks; VxBus

0 引言

龙芯 1E 和 1F 芯片是龙芯中科科技有限公司研制的具有自主知识产权的高性能 32 位抗辐射芯片，打破了国外对我国航天级高性能芯片的封锁。龙芯 1E 是一个高性能应用处理器 SOC，以龙芯 1 号处理器为运算中心，提供了通用的处理器部件和对外接口，包含中断控制器、定时器、RS232 串口控制器、浮点处理器、PCI 和存储器接口（存储器接口支持 SDRAM 和 FLASH ROM）等。龙芯 1E 芯片的外部时钟频率不低于 66 MHz，功耗 3 W。龙芯 1F 芯片是龙芯 1E 芯片的配套 IO 桥芯片，集成了航天领域常用的遥测遥控功能接口和外围接口，时钟 33 MHz，功耗 1 W，支持桥片模式和自主模式。龙芯 1F 芯片的桥片模式分为 3 种，即 PCI 总线设备模式、ISA 总线设备模式和 1553B 总线简易终端设备模式。龙芯 1E 和龙芯 1F 芯片的抗辐射特性为抗总剂量不小于 1×10^3Gy (Si)，单粒子锁定（SEL）阈值不小于 75MeV·cm^2/mg，单粒子翻转（SEU）阈值不小于 37MeV·cm^2/mg 或 IGSO 轨道翻转率小于 10^{-10}次/位天[1-2]。

VxWorks 系统由于其优秀的实时性和可靠性，在航天领域得到了广泛的应用。VxWorks 6.2 之后引入了新的基于 VxBus 的设备驱动开发模式。与传统的设备驱动程序高度耦合在板级支持包（BSP）中相比，基于 VxBus 的设备驱动开发可以使 BSP 的开发与设备驱动的开发相对分离，使得这两部分的开发工作可以并行开展。提高了设备驱动开发效率，使得设备驱动程序开发更加规范化，结构化，进一步提高了软件开发的质量和可靠性，并且降低了开发和测试成本[3]。

由于龙芯芯片在商用领域推出较多，在航天领域推出较少，目前对龙芯应用技术的研究大多集中在商用领域，在航天领域的应用技术的研究尚处于初始阶段[4]。在航天领域龙芯的应用介绍较少，并且研究内容主要着眼于系统引导，软件更新[5]和传统的 VxWorks BSP 移植设计等方面[6-7]，对如何针对 VxWorks 新推出的 VxBus 型驱动进行龙芯应用的设计介绍较少。

1 VxBus 设备驱动分析

VxWorks 系统下 VxBus 型驱动具有设备可配置以及驱动使用界面化操作的特点。如图 1 所示，进行 VxBus 驱动开发一般要提供驱动头文件，驱动源代码文件，驱动说明文件，

收稿日期:2017-09-15； **修回日期**:2017-10-23。

作者简介:邹玉龙(1984-),男,江苏东台人,工程师,硕士研究生,主要从事卫星软件技术方向的研究。

Makefile 编译文件，cdf 驱动配置文件，dr 驱动注册文件以及 dc 驱动注册文件。这几个文件存放的位置各不相同，驱动说明文件，驱动源代码文件和 Makefile 编译配置文件存放在“C：\ WindRiver \ vxworks－6.8 \ target \ 3rdparty \ vendorName \ driveName”路径下。其中“C：\ WindRiver”为 VxWorks 开发环境的安装路径，vendorName 是自定义的驱动供应商的名称，driveName 是自定义的驱动的名称。一个驱动供应商可以对应多个驱动，多个驱动都以驱动文件夹的方式存放在供应商文件夹下。cdf 配置文件存放在“C：\ WindRiver \ vxworks－6.8 \ target \ config \ comps \ vxWorks”路径下，这个目录下是专门用来存放驱动配置文件。dc 和 dr 配置文件存放在 C：\ WindRiver \ vxworks－6.8 \ target \ config \ comps \ src \ hwif 路径下，这个目录下是专门用来存放 VxBus 型驱动的注册文件。

VxWorks 系统初始化时，首先在 sysHwInit（）函数中调用 hardWareInterFaceInit（）函数，该函数完成 VxBus 型驱动的注册，并调用各个 VxBus 型驱动的 xxxInit（）函数，执行各个驱动的第一次初始化。在第一次初始化过程中，VxWorks 系统为每个 VxBus 型驱动设备分配了内存空间，并且连接了供应用层调用的驱动方法，从 BSP 的硬件设备定义文件 hwconf.c 中获取每个硬件设备的驱动参数。然后在 sysHwInit2（）函数中调用 vxbDevInit（）函数，在 vxbDevInit（）函数中调用各个 VxBus 型驱动的 xxxInit2（）函数，执行各个驱动的第二次初始化，进行了 VxBus 型驱动的中断服务函数的连接和使能。接着建立一个任务 vxbDevConnect，该任务执行了每个 VxBus 型驱动的 xxxConnect（）函数，在该函数中将执行一些额外的比较费时，但是不影响整个系统初始化的初始化操作[8-9]。整个 VxBus 驱动的初始化流程如下图 2 所示。

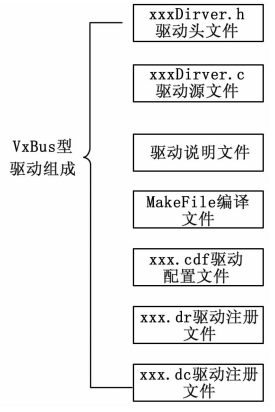

图 1　VxBus 驱动组成图

2　VxBus 设备驱动程序设计

对于 VxWorks 系统的基本运行，一般需要提供 3 种必需的驱动：时钟驱动，串口驱动和中断控制驱动。时钟驱动用来驱动 VxWorks 系统运行，如任务调度，定时控制等，串口驱动一般用作调试时输出日志信息，输入控制命令等，中断控制驱动用作在系统发生中断时进行中断响应。

1）串口驱动设计。

龙芯 1E 芯片上提供了 2 个 RS232 串口，串口控制器与 NS16550 串口控制器在寄存器接口上兼容，并且 VxWorks 开发环境中已经提供了基于 VxBus 型 NS16550 串口控制驱动的

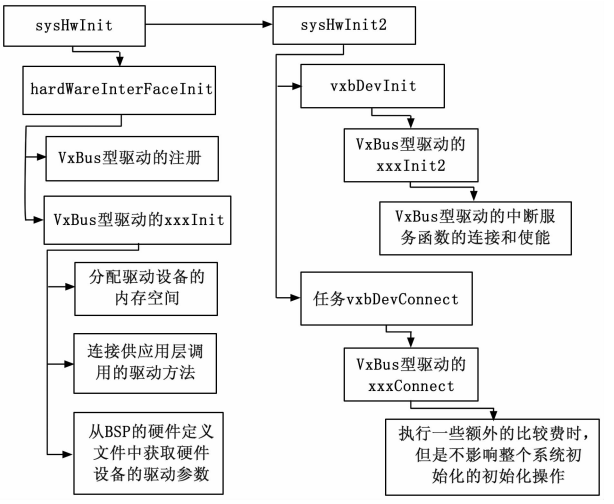

图 2　VxBus 驱动初始化流程图

参考源代码，源代码文件为 vxbNs16550Sio.c。源代码文件不需要修改，只要在 BSP 的硬件配置文件 hwconf.c 中配置各个串口的寄存器的起始地址，波特率，时钟等即可。关键代码如下。

首先定义两个串口设备的相关驱动参数：

```
define LSN_UART_DEFAULT_BAUD 115200
struct hcfResource ns16550Dev0Resources[] = {
{ "regBase", HCF_RES_INT, {(void *) LS1E_UART0_BASE} },
{ "baudRate", HCF_RES_INT, { (void *)LSN_UART_DEFAULT_BAUD }},
{ "clkFreq", HCF_RES_INT, {(void *)BAUD_CLK_FREQ} },
{ "regInterval", HCF_RES_INT, {(void *)UART_DELTA} },
};
define ns16550Dev0Num NELEMENTS ( ns16550Dev0Resources)

struct hcfResource ns16550Dev1Resources[] = {
{ "regBase", HCF_RES_INT, {(void *) LS1E_UART1_BASE} },
{ "baudRate", HCF_RES_INT, { (void *)LSN_UART_DEFAULT_BAUD }},
{ "clkFreq", HCF_RES_INT, {(void *)BAUD_CLK_FREQ} },
{ "regInterval", HCF_RES_INT, {(void *)UART_DELTA} },
};
define ns16550Dev1Num NELEMENTS ( ns16550Dev1Resources)
```

然后在设备列表中定义 2 个串口设备：

```
{ "ns16550", 0, VXB_BUSID_PLB, 0, ns16550Dev0Num, ns16550Dev0Resources },
{ "ns16550", 1, VXB_BUSID_PLB, 0, ns16550Dev1Num, ns16550Dev1Resources },
```

2）时钟驱动设计。

龙芯 1E1F 开发板上使用的时钟频率为 33 MHz，时钟驱动可以使用 VxWorks 开发环境中提供的 MIPS R4K 时钟驱动，源文件为 vxbMipsR4KTimer.c。需要在 BSP 的硬件配置文件 hwconf.c 中定义最小时钟，最大时钟，cpu 频率等时钟驱动参

数。关键代码如下。

首先定义时钟驱动的相关驱动参数：

```
struct hcfResource r4KTimerDevResources[] =
{
{"regBase", HCF_RES_INT, {(void *)0} },
{"minClkRate", HCF_RES_INT, {(void *)SYS_CLK_RATE_MIN} },
{"maxClkRate", HCF_RES_INT, {(void *)SYS_CLK_RATE_MAX} },
{"cpuClkRate", HCF_RES_INT, {(void *) 33000000} }
};

  define r4TimerDevNum NELEMENTS(r4KTimerDevResources)
```

然后在设备列表中定义时钟设备：

```
{" r4KTimerDev", 0, VXB _ BUSID _ PLB, 0, r4TimerDevNum, r4KTimerDevResources},
```

3）中断控制驱动设计。

根据龙芯 1E1F 平台的中断控制器的特点，中断控制驱动可以分层设计[10]，将中断驱动分为 3 个层次，如图 3 所示。第 1 层为 MIPS 中断控制，共 8 个中断类型，第 2 层为龙芯 1E 中断控制，共 32 个中断类型，第 3 层为龙芯 1F 中断控制，共 32 个中断类型。其中第 2 层龙芯 1E 的中断是作为第 1 层 MIPS 中断的第 2 种中断类型进行连接，第 3 层龙芯 1F 中断是作为第 2 层龙芯 1E 的第 17 种中断类型进行连接。

图 3 中仅列出了本文涉及的串口中断，时钟中断，龙芯 1E 中断，龙芯 1F 中断，SM1553B 中断，其他中断类型可以参考龙芯芯片数据手册[11]。

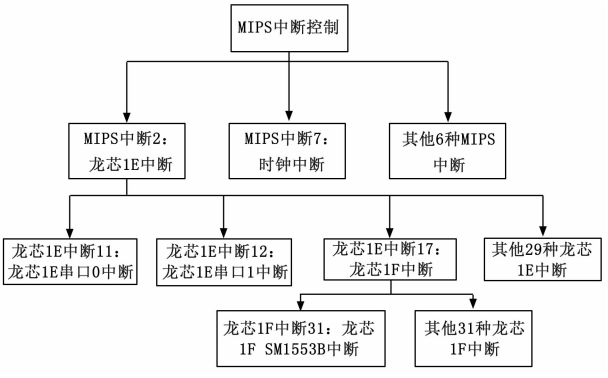

图 3 龙芯 1E1F 中断分层图

a）对 MIPS 中断控制的驱动可以使用 VxWorks 提供的 mipsIntCtlr 驱动，源文件为 vxbMipsIntCtlr. c。在 BSP 的 hwconf. c 文件中配置 MIPS 驱动中各个引脚中断所属的中断服务设备。关键代码如下。

首先定义 MIPS 中断控制设备的相关驱动参数：

```
const struct intrCtlrInputs mipsIntCtlr0Inputs[] = {
/* pin, driver, unit, index */
/* IPI interrupts from IPI controller */
{0, "legacy", 0, 0},
{1, "legacy", 0, 0},
{2, "LS1EIntCtlr", 0, 0},
{7, "r4KTimerDev", 0, 0},
};

const struct hcfResource cpu0Resources[] = {
{ "regBase", HCF_RES_INT, {(void *)TRUE} },
{ "input", HCF_RES_ADDR, {(void *) &mipsIntCtlr0Inputs[0]} },
{ "inputTableSize", HCF_RES_INT, {(void *)NELEMENTS(mipsIntCtlr0Inputs)}}
};
  define cpu0Num NELEMENTS(cpu0Resources)
```

然后在设备列表中定义 MIPS 中断控制器设备：

```
{ " mipsIntCtlr", 0, VXB _ BUSID _ PLB, 0, cpu0Num, cpu0Resources },
```

b）对龙芯 1E 中断控制的驱动需要编写新的 VxBus 类型的驱动。中断程序设计整体结构上参考 MIPS 中断控制程序。对于中断服务程序需要进行更改，在中断服务程序中，需要首先读取龙芯 1E 的中断状态寄存器，然后清除龙芯 1E 的中断状态，防止重复产生中断，接着再根据获取的龙芯 1E 的中断状态依次调用所有发生中断的子服务程序。中断服务程序关键代码如下：

```
/* 获取中断状态 */
vxbMipsLsnIntLevelGet (pInst, &ints);

/* 清除中断 */
MIPS_SW32(LS1E_INT_BASE+ LS1E_INT_ CLR, ints);

/* 调用子服务程序 */
if (ints ! = (UINT32)0)
{
for (i = 0;i < LS1E_INT_INPUTS && ints ! = 0;i++,ints >>= 1)
{
if ((ints & 0x01) ! = 0)
{
VXB_INTCTLR_ISR_CALL(&pDrvCtrl->isrHandle, I +pInst->unitNumber * 32);
}
}
}
```

在 BSP 的 hwconf. c 文件中龙芯 1E 中断控制器驱动的配置为：

定义龙芯 1E 中断控制器的驱动参数：

```
const struct intrCtlrInputs ls1EIntCtlrInputs0[] = {
/* pin, driver, unit, index */
/* interrupts inputs into cpu */
  {11, "ns16550", 0, 0 },
  {12, "ns16550", 1, 0 },
  {17, "LS1FIntCtlr", 0, 0 },
};

const struct hcfResource ls1EIntCtlrResources0[] = {
{ "regBase", HCF_RES_INT, {(void *)TRUE} },
{ "input", HCF_RES_ADDR, {(void *)&ls1EIntCtlrInputs0[0]} },
{ "inputTableSize", HCF_RES_INT, {(void *)NELEMENTS(ls1EIntCtlrInputs0)}},
};
  define ls1EIntCtlrNum0 NELEMENTS(ls1EIntCtlrResources0)
```

在设备列表中定义龙芯 1E 中断控制器设备：

```
{ " LS1EIntCtlr", 0, VXB _ BUSID _ PLB, 0, ls1EIntCtlrNum0, ls1EIntCtlrResources0 },
```

c）对龙芯 1F 中断控制器的驱动需要编写新的 VxBus 类型的驱动。中断程序整体结构上参考龙芯 1E 中断控制的程序。同样对于中断服务程序需要进行更改。在中断服务程序中，需要首先读取龙芯 1F 的中断状态寄存器，接着再根据获取的龙芯 1F 的中断状态调用所有发生中断的子服务程序。与龙芯 1E 中断服务程序不同，在龙芯 1F 的中断服务程序中不需要进行中断清除，某个引脚的中断状态的清除在被调用的中断子服务程序中单独进行。中断服务程序关键代码如下：

```
/ * 获取中断状态 * /
vxbMipsLsn_1fIntLevelGet (pInst, &ints);

/ * 调用子服务程序 * /
if (ints ! = (UINT32)0)
{
for (i = 0;i < (LS1F_INT_INPUTS) && ints ! = 0;i++,ints >>= 1)
{
if ((ints & 0x01) ! = 0)
{
VXB_INTCTLR_ISR_CALL(&pDrvCtrl->isrHandle,i);
}
}
}
```

在 BSP 的 hwconf. c 文件中龙芯 1F 中断控制器驱动的配置为：

定义龙芯 1F 中断控制器的驱动参数：

```
const struct intrCtlrInputs ls1FIntCtlrInputs0[] =
{
/ * pin, driver, unit, index * /
{31,"LS1F_sm1553b", 0, 0},
};

const struct hcfResource ls1FIntCtlrResources0[] = {
{ "regBase", HCF_RES_INT, {(void * )TRUE} },
{ "input", HCF_RES_ADDR, {(void * ) & ls1FIntCtlrInputs0[0]} },
{ "inputTableSize", HCF_RES_INT, {(void * ) NELEMENTS(ls1FIntCtlrInputs0)}},

};
  define ls1FIntCtlrNum0 NELEMENTS(ls1FIntCtlrResources0)
```

在设备列表中定义龙芯 1F 中断控制器设备：

```
{ " LS1FIntCtlr", 0, VXB _ BUSID _ PLB, 0, ls1FIntCtlrNum0, ls1FIntCtlrResources0},
```

4）龙芯 1F 芯片的智能 1553B 功能驱动设计。

首先在“C：\ WindRiver \ vxworks － 6.8 \ target \ 3rdparty \ vendorName”路径下建立一个新的文件夹用来存放驱动文件，文件夹名称为 LS1FSM1553B，然后在该文件夹下分别建立 vxbLS1F _ sm1553b. c，40vxbLS1F _ sm1553b. cdf，vxbLS1F _ sm1553b. dc，vxbLS1F _ sm1553b. dr，Makefile，README 等几个文件。其中 vxbLS1F _ sm1553b. c 文件中包含了驱动的实现代码，是驱动的主要文件。40vxbLS1F _ sm1553b. cdf 包含了用于 vxWorks 工程进行图形化配置 1553B 驱动的描述信息。vxbLS1F _ sm1553b. dc，vxbLS1F _ sm1553b. dr 文件包含 1553B 驱动的注册信息。Makefile 文件用于将 1553B 驱动编译到目标函数库文件中，README 文件包含了介绍驱动的功能，使用方法等信息。

在 vxbLS1F _ sm1553b. c 文件中除了包含 vxBus 型驱动所必需的第一阶段初始化函数 xxxinit（），第二阶段初始化函数 xxxinit2（），中断连接与使能函数 xxxConnect（），中断服务函数 xxxISR（）等函数外，还对外提供了一系列的方法供上层应用调用。提供的方法有 BC 功能初始化方法，BC 启动方法，用户自定义的中断服务等。这里为了简便起见，在 BC 启动方法里面也完成了 BC 消息块的设置和 BC 消息数据的设置。实际应用中需要将 BC 消息块的设置和 BC 数据的设置分别设计为可供上层应用调用的驱动方法，以增强驱动的灵活性和适应能力。

源代码文件和配置文件编辑完成后在 VxWorks shell 中使用编译命令 make CPU＝MIPSI32R2 TOOL＝gnule，将最新的智能 1553B 驱动增加到库文件中。其中 MIPSI32R2 表示将驱动编译成适合架构为 MIPS 的 32 位 R2 类型 CPU，gnule 表示驱动使用 gnu 编译器编译，并且编译成小端模式。

接着将 40vxbLS1F _ sm1553b. cdf 拷贝到“C：\ WindRiver \ vxworks－6.8 \ target \ config \ comps \ vxWorks”路径下。然后将 vxbLS1F _ sm1553b. dc，vxbLS1F _ sm1553b. dr 文件拷贝到“C：\ WindRiver \ vxworks－6.8 \ target \ config \ comps \ src \ hwif”路径下。完成驱动在 Wind River Workbench 开发环境中的配置。更新驱动文件后需要重新启动 Workbench 开发环境，并且重新建立工程才能使用新的驱动程序。

3　实验验证

1）实验环境。

目标机：龙芯 1E1F 开发板。

宿主机：研华工控机（i7 处理器，16G 内存，2TB 硬盘），Alta PCI 1553B 双通道 仿真卡（用于模拟 RT 以及总线数据监控），windows 7 x64，vxWorks6.8 开发环境 Workbench 3.2，On Chip Debugger ICE2（用于下载程序），AltaView Bus Analyzer（Alta PCI 1553B 双通道 仿真卡的应用软件）。

设备连接图如图 4。

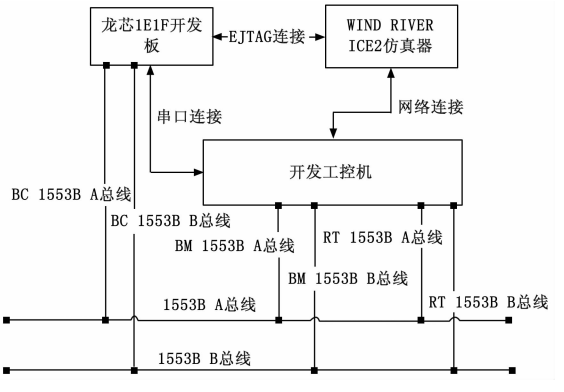

图 4　龙芯 1E1F 实验连接图

2）实验过程。

首先在 workbench 开发环境中创建一个 VxWorks Image

型工程，选择相应的支持龙芯 1E1F 的 BSP，然后在工程的内核模块配置界面上将上文提到的 ns16550 串口，mips r4k 时钟，mips 中断控制器，龙芯 1E 中断控制器，龙芯 1F 中断控制器，智能 1553B 等驱动模块都选择上，然后保存工程。接着在 usrAppInit.c 文件中 usrAppInit（）函数内添加应用层代码。通过调用智能 1553B 驱动提供的一些方法，进行 1553B 的 BC 功能和相关驱动的功能测试。关键代码如下：

a）调用 BC init 方法执行 BC 的初始化。

```
methodLS1FBCInit = vxbDevMethodGet(devID,DEVMETHOD_
CALL(ls1fbcInit));
if(NULL ! = methodLS1FBCInit)
{
  methodLS1FBCInit(devID,0);
}
```

b）添加用户自定义的中断服务函数。

```
methodLS1FBCSetISR = vxbDevMethodGet(devID,DEVMETH-
OD_CALL(ls1fbcSetISR));
if(NULL ! = methodLS1FBCSetISR)
{
methodLS1FBCSetISR(devID,sm1553bISR);
}
```

其中 sm1553bISR 内部只进行一个打印输出：

```
logMsg("1F 1553b new ISR,\r\n",0,0,0,0,0,0);
```

c）启动一个任务进行每秒启动一次 BC 消息帧。

```
taskSpawn("tBC－Test",100, 0,5000,（FUNCPTR） task_1F_
BC, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
```

任务 task _ 1F _ BC 中首先获取 SM1553b 驱动的 ls1fbcStart 方法，然后设计了一个 while 循环模块，在循环中调用 ls1fbcStart 方法，然后通过 taskDelay 等待 1 秒钟，再进行下一个循环。主要代码如下：

```
VXB_DEVICE_ID devID = NULL;
STATUS ( * methodLS1FBCStart)(VXB_DEVICE_ID devID,void
 * pArg);

devID = vxbInstByNameFind("LS1F_sm1553b",0);
if(NULL ! = devID)
{
methodLS1FBCStart = vxbDevMethodGet ( devID,DEVMETH-
OD_CALL ( ls1fbcStart));
}

while(NULL ! = methodLS1FBCStart)
{
  methodLS1FBCStart(devID,0);
  /* 等待 1 秒,1 秒 60 个 tick, */
  taskDelay(60);
}
```

3）实验结果。

系统启动后在串口终端运行 vxBusShow 命令显示当前系统的驱动和设备。从串口输出可知当前系统注册的驱动有 mipsIntCtlr 中断控制器驱动，LS1F _ sm1553b 驱动，LS1FIntCtlr 中断控制器驱动，LS1EIntCtlr 中断控制器驱动，ns16550 驱动，r4KTimerDev 时钟驱动等，已经与驱动匹配的设备有 LS1F _ sm1553b unit0，mipsIntCtlr0，LS1FIntCtlr unit0，LS1EIntCtlr unit0，ns16550 unit0，ns16550 unit1，r4KTimerDev unit0 等。

BC 运行后，观察串口输出内容（见图 5），每秒执行了一次 BC 帧，并且响应中断 2 次，进入了用户自定义的中断服务函数 2 次。表明中断响应过程正常。

图 5 中断响应串口输出图

通过 Alta 1553B 仿真卡提供的 RT 和 BM 功能可以监控到 1553B 总线上进行了从 BC 到 RT4 的消息传输，分别为子地址 1，子地址 2，子地址 3，数据内容与数据频率与前面的 BC 消息块设置一致。

4 结论

本文基于龙芯 1E1F 平台和 VxWorks 系统，对 VxBus 型驱动设计技术进行了详细介绍，对串口、时钟，中断控制器和智能 1553B 的驱动进行了设计，并通过实验验证，证明了 VxBus 型驱动设计技术结构更加清晰，能够更好的进行底层技术封装，对上层应用软件开发提供了更加友好的接口。对未来航天领域基于龙芯和 Vxworks 系统的开发设计具有较高的参考价值。

参考文献：

[1] 龙芯中科技术有限公司，龙芯 1E 处理器数据手册［Z］. 北京：龙芯中科技术有限公司，2016.

[2] 龙芯中科技术有限公司，龙芯 1F 处理器用户手册［Z］. 北京：龙芯中科技术有限公司，2015.

[3] 向昱丞，周加谊，浅谈 VxBus 的设备驱动开发［J］. 机电产品开发与创新，2016，29（2）：57－58.

[4] 王 雷，樊晓桠，王党辉，龙芯 3A 平台 Vxworks 移植的研究和实现［J］. 微电子学与计算机，2012，29（2）：86－90.

[5] 史毅龙，薛长斌，基于“龙芯”的 VxWorks 系统函数在轨更新研究［J］. 电子设计工程，2015，23（21）：106－109.

[6] 杨 晔，安军社，VxWorks 在龙芯处理器上移植与实现［J］. 微计算机信息，2010，26（12－2）：31－33.

[7] 陈学兵，沈毅南，张振华，VxWorks5. 5 在龙芯 2 号处理器的移植和性能分析［J］. 计算机测量与控制，2012，20（9）：2542－2545.

（下转第 111 页）