第 15卷 第 2期
2005年 6月

天 津 工 程 师 范 学 院 学 报
JOURNAL OF TIANJIN UNIVERSITY OF TECHNOLOGY AND EDUCATION

Vol 15 No. 2
Jun. 2005

嵌入式实时操作系统 VxWorks下 BSP分析及 VxWorks裁减*

褚 哲, 孟小锁
(西安微电子技术研究所, 陕西 西安 710075)

摘 要:以 VxWorks操作系统为例,阐述了 BSP的概念、原理和系统启动流程,并在此基础上以某目标机为原型,着重叙述了 VxWorks的裁减方法。

关键词: VxWorks; 板级支持包; 映像
中图分类号: TP316. 2 **文献标识码**: A **文章编号**: 1673-1018 (2005) 02-0027-04

BSP analysis under embedded real-time operating system VxWorks and VxWorks 'tailor

CHU Zhe, MENG Xiao-suo
(Xi'an Microelectronics Technology Institute, Xi'an 710054, China)

Abstract: This article takes the VxWorks operating system as an example to illustrate the concept, the princirle and the start-up procedure of BSP. Under this foundation it uses certain target machine as original form to explain the tailor method of VxWorks.

Key words: VxWorks; board suppot package; image

VxWorks操作系统是美国 WindRiver公司推出的一种嵌入式强实时操作系统,自 20世纪 80年代问世以来,以其不断推出的升级版本、高性能内核以及友好的用户开发环境,在嵌入式实时操作系统领域逐渐占据一席之地,尤其以成功应用于火星探测车和爱国者导弹等高科技产品而声名鹊起,拥有较多的用户。BSP(board suppot package)为板级支持包的缩写。因为 VxWorks标准的 BSP是针对 PC的,对于 PC机用户,主要按照标准配置,不需要做太大的改动就可以建立主机与目标机的调试环境。而对于使用非标准 (不是 PC机)加固机的用户,则需要对 BSP进行修改以满足特定目标机的要求,对 VxWorks的裁减就是通过对 BSP的修改来完成的。要进行以上的工作,就需要了解 VxWorks的启动过程及 BSP的概念和工作原理,本文在阐述了 BSP概念和工作原理的基础上,以某目标机为原型,详细叙述 VxWorks的裁减方法。

1 BSP的概念及原理

BSP介于主板硬件和操作系统之间,属于操作系统的一部分,目的是支持操作系统,使之能够更好地运行于硬件主板。BSP是相对于操作系统而言的,不同的操作系统对应于不同定义形式的 BSP,例如 VxWorks的 BSP和 Linux的 BSP相对于某一 CPU来说尽管实现的功能一样,可是写法和接口定义是完全不同的,所以写 BSP一定要按照该系统 BSP的定义形式来写 (BSP的编写过程大多数是在某一个成型的 BSP模板上进行修改),这样才能与上层 OS保持正确的接口,良好地支持上层 OS。BSP在 VxWorks中的地位如图 1所示。

BSP文件在 VxWorks/target/config/all和 VxWorks/target/config/bapname文件夹里。其中, all文

* 收稿日期: 2004 - 12 - 08.
作者简介:褚 哲 (1975—),男,硕士研究生,研究方向为嵌入式软件.

件夹里的文件是所有 BSP的通用文件, bspname文件夹的文件是用户自己定制的 BSP文件。经过编译、链接,并在 makefile和 bspname等文件的控制下,原程序最后将生成 VxWorks映像。VxWorks的映像分为可下载映像和可引导映像。

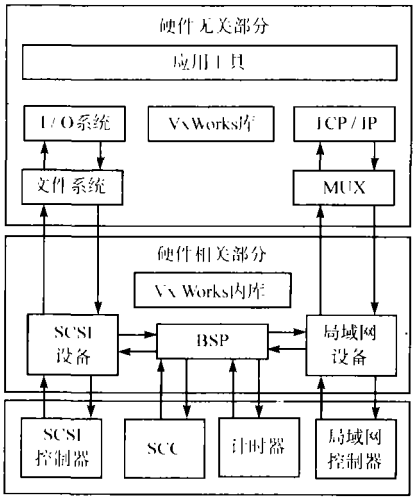

图 1 VxWorks与 BSP的层次关系

可下载映像 (loadable image)包括两部分:一是 VxWorks; 二是 bootRom。两部分是独立创建的。其中 bootRom 包括压缩的 bootRom 映像、非压缩的 bootRom 映像和驻留 ROM 的 bootRom映像 3种类型。

可引导映像 (bootable image)是将引导程序和 VxWorks融为一体的映像,它常常是最终产品,包括不驻留 ROM的映像和驻留 ROM的映像 2种类型。

BSP是所有与硬件相关的代码体的集合,主要包括: (1) 在一个系统被引导时,目标系统硬件初始化程序; (2) 目标系统上设备的驱动程序,这些设备包括定时器、以太网控制器、一组串行口和 SCSI控制器等,控制这些设备的函数成为设备驱动程序。

VxWorks系统的 BSP功能模块关系如图 2所示。

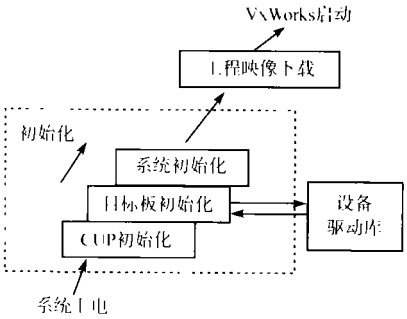

图 2 BSP各功能模块关系

2 VxWorks启动过程

考虑到 VxWorks映像分为可下载映像和可引导映像,不同的映像启动顺序略有不同,但大体的顺序是相同的。因此,此处以固化在 ROM中的可引导映像 (VxWorks_ROM)来讲解 VxWorks的启动过程,如图 3所示。

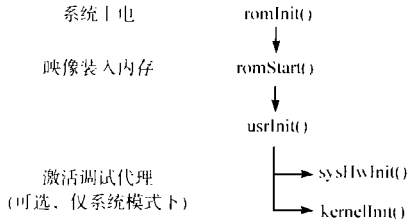

图 3 固化在 ROM中的 VxWorks引导顺序

(1) 机器加电以后,处理器"跳"到 ROM或 Flash中,用汇编语言编写的初始化程序 rominit(),它在文件 rominit s中,主要完成:

禁止中断;

保存启动类型 (冷/热启动);

硬件相关的初始化;

转到函数 romstart()。

(2) romstart(),它在文件 bootinit c中,

从 ROM中将数据段拷贝到 RAM,将内存清 0;

从 ROM中将代码段拷贝到 RAM;

根据启动类型调用函数 usrInit()。

(3) usrInit(),它在文件 usrConfig c中,

将 BSS未初始化的段清 0;

将引导类型保存到 sysStartType;

调用异常向量初始化函数 excVecInit(),初始化所有系统和默认的中断向量;

调用系统的硬件设备初始化函数 sysHwInit();

调用内核库初始化函数 usrKernelInit();

调用内核初始化函数 kernelInit()。

(4) usrKernelInit(),它在文件 usrKernel c中,主要完成库的初始化工作。

classLibInit();

taskLibInit();

taskHookLibInit();

semBLibInit();

semMLibInit();

semCLibInit();

wdLibInit();

qInit();

workQInit()。

(5) kernelInit(),初始化多任务环境并启动内核,它在文件 kernelLib. c中。

(6) usrRoot(),它在文件 usrConfig c中。

初始化 I/O系统,安装设备的驱动,根据配置文件 ConfigAll h和 Config h中的配置创建指定的设备。

(7)转入用户创建的应用程序的入口。

3 VxWorks的裁减方法及步骤

在对 BSP的概念和 VxWorks的启动顺序有了一定的了解之后,此处以某目标机为原型阐述 VxWorks的裁减方法。

3.1 目标机配置

CPU :80486DX2-66,浮点协处理器与 80387兼容。存储区: SDRAM, 00000H-3FFFFH ,256 KB,其中 00000-007FFH为某固定程序使用; EPROM, F8000H-FFFFFH, 32 KB。中断控制器 8259A一片。定时器 8254。3个定时计数器的输入时钟频率为 1 250 000 Hz。通讯控制器 82C52,输入时钟频率为 14 745 600 Hz。

除此之外,目标机无软驱,无网卡,有两个串口。

3.2 建立所需目录

在 C: \Tornado \target \config \中建立子目录 allxxx, bspxxx,其中 xxx用户可根据需要将其改为所用目标机的名称。

拷贝 C: \Tornado \target \config \all中的全部文件到子目录 allxxx中,拷贝 C: \Tornado \target \config \pc486中的全部文件到子目录 bspxxx中以便修改且不影响其他人的使用。

3.3 修改 Makefile

修改或添加以下语句:

(1) TARGET_DIR = bspxxx

TARGET_DIR为 bsp目录名,将其指向要修改的 bsp目录。

(2) USRCONFIG = $(TGT_DIR) \config \ $(TARGET_DIR) \usrConfig c

CONFIG_ALL_H = $(TGT_DIR) \config \ $(TARGET_DIR) \usrConfig c

TGT_DIR为 target路径,默认为 $(WIND_BASE) \target,其中 WIND_BASE为 VxWorks的开发环境 Tornado的安装目录,此处为 C: \Tornado。

(3) CONFIG_ALL = .. \allxxx

将缺省 All的文件复制到 allxxx目录,在 allxxx目录下作自己的修改。

修改 CONFIG_ALL定义,指向自己的 ALL目录。

(4) 因为用此目标机中的固定程序下载 VxWorks应用的映像,所以,

```
#define ROM_BASE_ADRS    00001000
#define ROM_TEXT_ADRS    00001000
#define ROM_SIZE         00008000
#define RAM_HIGH_ADRS    00001000
```

其中,用规则 VxWorks_rom_low来生成映像,以便映像中包含 romInit o,实模式转换为保护模式。

ROM_BASE_ADRS: rom的起始地址

ROM_TEXT_ADRS: rom映像的起始地址

ROM_SIZE: rom的大小

RAM_HIGH_ADRS: VxWorks应用在存贮器低端定位开始的地址

注意: ROM_TEXT_ADRS, ROM_SIZE, RAM_HIGH_ADRS, 和 RAM_LOW_ADRS在 config h,Makefile和 Makefile. *文件中都要定义,且大小必须要保持一致。Makefile中的上述地址不以 0X开头,与 config h中有所区别。

(5) MACH_EXTRA = i82c52Sio. o lsDemo. o

i82c52Sio. o lsDemo. o这两个模块是为目标机所编写的特定模块。

此处用户可加入与目标机有关的所需模块。

3.4 修改 configAll h

(1) 因 RAM容量只有 256 KB,故排除掉不需要的软件模块。把“INCLUDE SOFTWARE FACILITIES”中不需要的模块移到“EXCLUDE FACILITIES”中。

(2) 在“low memory lauout”中修改

```
#define BOOT_Line_offset    0x828
```

因为不用引导程序,所以修改为

```
#define Exc_MSG_offset    0x828
```

这是因为 VxWorks系统 (x86)低端存储区缺省布局是

中断向量表	0x0000
全局描述符表	0x800
共享内存	0x1100
启动行	0x1200
异常消息	0x1300
软盘直接访问区	0x2000
初始化堆栈	0x5000
系统映像	0x8000

全局描述符表仅用 5项,不用启动行,把 romInit.s 系统映像中第一模块定位在 0x1000处。

3.5 修改 config.h

(1) 因为没有 PC控制台,所以在 PC console definitions处改 #if TRUE为 #if FALSE。

(2) 因为此处的通讯用 COM2口,所以添加

```
#undef      CONSOLE_TTY
#define     CONSOLE_TTY  1
```

(3) 因为用目标机的固定程序下载 VxWorks应用的映像,所以

#define ROM_BASE_ADRS 0X1000 根据片选参数设置 ROM基地址

```
#define ROM_TEXT_ADRS   0X1000
#define ROM_SIZE        0X8000
#define RAM_HIGH_ADRS   0X1000
```

(4) 因为无软驱,所以在 driver and file system option部分用 #if FALSE和 #endif删除 4个 #define。

(5) 因为无网卡,所以在 Network driver option部分取消网络驱动程序的定义。

(6) 因为用一片 8259A,所以 #difine NUMBER_OF_IRQS 8

(7) 根据 SDRAM的大小设置 LOCAL_MEM_SIZE

```
#define  LOCAL_MEM_SIZE  0X40000
```

3.6 修改 PC.h

(1) #define PIT_CLOCK 1250000 定时器输入时钟。

(2) #define ROM_STACK 0X1000 系统初始化时用的初始堆栈指针。

3.7 修改 sysLib.c

#include“i8259Intr.c”

#include“i8253Timers.c”

3.8 修改 c:\tornado\target\config\d5b\usrConfig.c

将 sysclkRateSet(60)改为 sysclkRateSet(50)。

VxWorks默认系统时钟速率为 60tick,在此处改为所需的 50tick。

3.9 DOS下做

\tornado\host\x86-win32\bin\torvars.bat 设置环境变量

```
cd  \tornado\target\config\bspxxx
make  VxWorks_rom_low
```

在低端存储区生成 VxWokrs应用映像

将 VxWorks_rom_low转换为 hex文件,烧入目标机的 ROM中,加电,成功启动。

4 结束语

本文结合某目标机的硬件配置,在介绍 BSP的概念的基础上,给出了一个对 VxWorks进行裁减以适应目标机的详细过程,为建立主机与宿主机之间的开发调试环境以及后续的应用程序的开发打下了坚实的基础。

参考文献:

[1] 王学龙. 嵌入式 VxWorks系统开发与应用 [M]. 北京:人民邮电出版社, 2003.

[2] 陈智育. VxWorks程序开发实践 [M]. 北京:人民邮电出版社, 2004.