文章编号:1007－757X(2005)10－0016－04

VxWorks在S3C2410上的BSP设计

张　忠，　樊留群

摘　要:VxWorks是美国WindRiver公司设计开发的一种嵌入式实时操作系统,可移植性是对嵌入式操作系统的一个重要要求,VxWorks BSP(板级支持包)正是实现可移植性的中间层软件,使操作系统的应用代码独立于具体硬件。本文介绍了VxWorks BSP的开发,分析了VxWorks的启动过程,给出了基于三星S3C2410处理器开发板的BSP设计以及VxWorks的映像编译下载,其中重点介绍了BSP中几个重要文件和函数的设计,最后提出了开发调试过程中的几个注意点。

关键词:VxWorks;BSP(板级支持包);ARM;S3C2410

中图分类号:TP316　　**文献标识码**:A

1　引言

VxWorks是美国WindRiver公司设计开发的一款嵌入式实时操作系统。能支持多种微处理器:PowerPC,x86,MIPS,ARM,SPARC等。它采用微内核结构,具有高可靠性、实时性、丰富的网络协议、良好的兼容性以及可裁减性等特点,同时具有友好的用户开发环境。其中VxWorks很好的移植性通过BSP来实现,BSP使操作系统能够独立于具体硬件,对上层屏蔽具体的硬件,为系统组件以及应用程序提供统一的接口。

本文主要研究如何将VxWorks操作系统移植到基于三星S3C2410处理器硬件平台上,即BSP的设计。主要内容包括下面两点:熟悉并掌握VxWorks映像种类,系统的启动顺序和过程,以及BSP软件包中各文件的组成和作用以及相应的设置文件的修改;二是熟悉系统底层驱动,也就是要对开发板的硬件环境有一个初步了解,这样才可以结合具体硬件开发板设计出相应的底层驱动程序。

2　BSP组成

2.1　BSP概念

对于嵌入式系统而言,并没有像PC那样的标准,存在着各种不同的嵌入式硬件平台,这就使得无法完全由操作系统来实现上层软件与底层硬件的无关性。BSP正是采用当今嵌入式分层设计的介于操作系统和底层硬件的中间层。BSP通常是指针对具体的硬件平台,用户所编写的启动代码和部分设备驱动程序的集合。它所实现的功能包括初始化,驱动部分设备。最基本的BSP仅需要支持处理机复位,初始化,驱动串口和必要的时钟处理。BSP的概念只是针对嵌入式操作系统而言的,而像DOS Windows UNIX等BIOS操作系统是无BSP可言的。不过到目前为止,嵌入式系统中也没有对BSP有明确统一的定义,不同的嵌入式系统BSP实现的功能有所差别。在VxWorks系统中,对BSP描述为介于底层硬件环境和VxWorks之间的软件接口,它的主要功能是系统加电后初始化目标机硬件,初始化操作系统及提供部分硬件的驱动程序。

2.2　BSP组成

其主要的两部分组成为:初始化代码,硬件驱动程序。

BSP的初始化部分是指从上电复位开始直到WIND内核和usrRoot()函数启动的这段时间内系统的执行过程。具体包括:CPU初始化,设备初始化以及系统初始化。CPU初始化CPU内部寄存器;设备初始化智能I/O的寄存器,将板上设备打通;系统初始化为系统的运行准备数据结构,进行数据初始化。其初始化过程如图1所示:

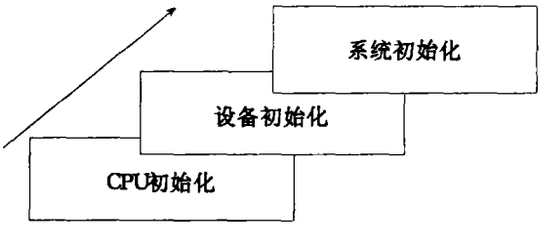

图1　初始化过程

驱动程序就是一些包含I/O操作的子函数。VxWorks系统的驱动程序也可以抽象为三个层次:常规操作,与I/O的接口,以及与组件的接口。其抽象逻辑如图2所示。驱动程序与I/O系统的接口使驱动具有更好的层次性,驱动程序与组件的接口使驱动具有更好的抽象性。

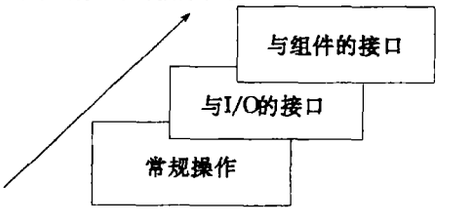

图2　驱动程序抽象逻辑

作者简介:张　忠,同济大学CIMS中心,硕士研究生,上海　200092
樊留群,同济大学CIMS中心,上海　200092

3 VxWorks 映像分类及启动过程

VxWorks 映像可以分为两大类：可下载映像和可引导映像。可加载型映像的执行首先由固化在目标机上的引导代码，通过串口或者网口将系统映像从宿主机上装载到目标机 RAM 中，然后继续执行；可引导型映像与可加载映像不同，它是将引导程序和 VxWorks 融为一体的映像，往往是最终产品，常常直接烧入 ROM 或者 Flash。

两种类型的映像都可以分成两步，第一步为系统加电后对必要的硬件进行初始化，这一步目的就是为了加载系统映像，所以在这一步不需要做太多的工作；第二步为系统加载后首先就是对系统硬件作完整的初始化过程，然后完成系统内核初始化，这一切处理完之后跳入用户应用程序。两种类型的启动过程如图 3 所示。

可加载映像包括两部分：boot ROM 以及 VxWorks。这两部分是独立创建的。加载型映像引导主要是用于开发调试阶段。

可加载型映像引导过程如下：

1)引导代码的装入。系统加电后执行首段引导代码，把引导代码段和数据段从ROM 或者Flash 中装入RAM 中的RAM—HIGH—ADRS。Boot ROM 可分为如下三种类型：被压缩的 boot ROM 映像，非压缩的 boot ROM 映像以及驻留 ROM 的 boot ROM 映像。对于压缩型引导代码，这种类型的引导代码在拷贝时对它进行解压操作；对于非压缩型引导代码，这种类型的引导代码直接进行拷贝操作；驻留型引导代码仅仅拷贝它的数据段到 RAM 中。

2)VxWorks 映像的载入。引导代码执行后，把 VxWorks 映像装入到 RAM 中 RAM—LOW—ADRS，然后跳转到 VxWorks 映像装入点。

3)系统初始化。由 VxWorks 映像中的系统初始化代码来执行，完成系统初始化操作。

可引导型映像引导包括不驻留 ROM 的映像和驻留 ROM 的映像两种类型。对于驻留 ROM 的 VxWorks 映像，引导程序只把 VxWorks 映像的数据段和堆栈段复制到 RAM 的 RAM—LOW—ADRS，代码段则驻留在 ROM 中。驻留型 VxWorks 映像主要是为了节省 RAM 空间，以便应用程序有更大的空间运行，但缺点是运行速度慢。随着片外扩展 RAM 空间越来越大，不需要采用不驻留 ROM 映像。其引导过程与可加载型类似，主要不同在系统加电时候，首先执行的是 romInit.s 中的函数 romInit()，完成最基本的初始化后跳入到 bootInit.c 中的romStart()函数，来完成装载 VxWorks 映像的工作。接下来的启动过程和上面所描述的可加载型一样。

4 VxWorks 在S3C2410 上的BSP 设计

4.1 S3C2410 开发板硬件设计

开发板采用三星 S3C2410 处理器，S3C2410 处理器为 ARM920T 内核16/32 位RISC CPU，拥有独立的16KB 指令和 16KB 数据CACHE，MMU 内存管理单元，NAND Flash boot loader，系统管理单元(SDRAM 控制器等)，3 通道 UART，4 通道DMA，4 通道具备 PWM 功能的定时器，I/O 口，RTC(实时时钟)，8 通道 10 位精度 ADC 和触摸屏控制器，IIC 总线接口，IIS 数字音频总线接口，USB 主机，USB 设备，SD/MMC 卡控制器，集成LCD 控制器(支持STN 和TFT)，2 通道SPI 和 PLL 数字锁相环，主频最高可达 203MHz。

S3C2410 开发板在此基础上外扩：两片 32MB 的三星 SDRAM；共同组成 64MB 内存，一片 Intel strata flash (16MB)，用以存储系统映像；一片 Atmel 的 AT49LV1614A，2MB NOR flash，可以用来存放系统上电引导程序，即相当于 PC 的 BIOS；通过以太网控制器芯片 DM9000E 扩展了一个网口。其硬件功能模块结构图如下：

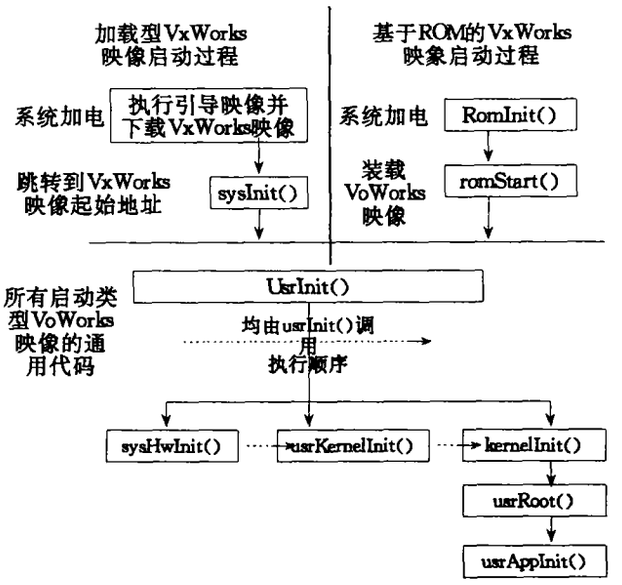

图 3　VxWorks 映像启动过程

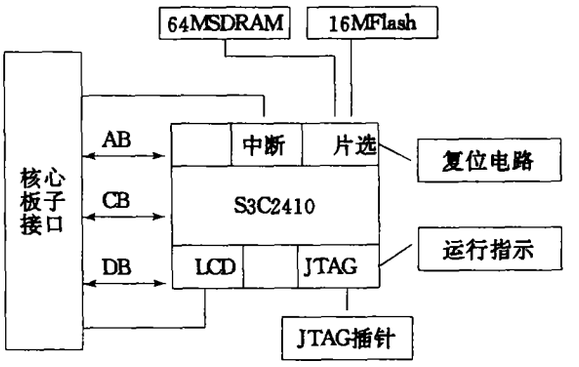

图 4　S3C2410 开发板功能结构图

4.2 BSP 中几个重要文件的修改

BSP 文件主要包含在 Target/config 下的目录中：其中 target/config/all 中包含如下文件 configAll.h，bootConfig.c，

bootInit.c，usrConfig.c 等文件，这些都是BSP 通用文件，我们一般不对这些通用文件作修改；target/config/integrator920t 是专为基于ARM920T 内核的处理器设计的BSP 模板文件，我们所选择的S3C2410 处理器是基于ARM920T 内核的，因此我们主要的工作就是对这个文件夹下面的文件作修改。下面主要给出对该文件夹下面的几个重要文件所作的修改。

1)修改config.h

Config.h 文件包含了所有头文件和与CPU 板子相关的特殊定义。config.h 中主要修改的内容有：

定义引导行

```
#define DEFAULT—BOOT—LINE \
"fei(0,0) host:/tor2/target/config/integrator920t/vxWorks" \
"h=90.0.0.3 e=90.0.0.50:ffffff00 u=target tn=targetname"
```

Fei 为传输协议；host 后面为主机中存放VxWorks 映像的路径，h 为主机IP，e 为目标板IP。

目标板内存的配置，这一部分必须根据实际CPU 以及外扩存储器的大小来确定：

```
#define LOCAL—MEM—LOCAL—ADRS 0x00000000
/* 固定在零点 */
#define LOCAL—MEM—BUS—ADRS 0x00000000
/* 固定在零点 */
#define LOCAL—MEM—SIZE 0x04000000
/* 64MB 由两片32M 的外扩SDRAM 组成 */
#define ROM—BASE—ADRS 0x24000000
/* ROM 基地址 */
#define ROM—TEXT—ADRS ROM—BASE—ADRS
/* ROM 中代码起始地址 */
#define ROM—SIZE 0x00100000
#define RAM—LOW—ADRS 0x00001000
/* VxWorks 映像入口点，对所有ARM 处理器存储布局都一样 */
#define RAM—HIGH—ADRS 0x00600000
/* boot ROM 在RAM 中的入口 */
```

VxWorks 的缺省配置是由configAll.h 来确定的，用户可通过config.h 来改变缺省配置，一般来说，缺省配置是按照比较完备的方式进行系统配置，而实际的软件、硬件环境往往不那么完备，用户必须查看configAll.h，并在config.h 中将不要的软、硬件配置和初始化去掉。

2)修改romInit.s

romInit.s 主要工作由函数romInit()完成，该函数是所有固化在ROM 或者Flash 里VxWorks 映像首先执行的代码。主要完成的任务有：

禁止中断，使处理器复位。通过设置CPSR 的I—BIT 和F—BIT 都为1 来实现。还要设置中断寄存器为关模式同时将处理器切换到SVC32 模式下，屏蔽AIC 中断。

代码如下：

```
MRS r1, cpsr
BIC r1, r1, #MASK—MODE
ORR r1, r1, #MODE—SVC32 | I—BIT | F—BIT
MSR cpsr, r1
MOV r2, #IC—BASE
/* R2 的值传递给中断控制器 */
MVN r1, #0 /* &FFFFFFFF */
STR r1, [r2, #FIQ—DISABLE—IC—BASE]
/* 关闭所有FIQ 中断源 */
STR r1, [r2, #IRQ—DISABLE—IC—BASE]
/* 关闭所有IRQ 中断源 */
```

保存启动类型，在跳至romInit()函数的时候，传递启动类型。

初始化缓存并屏蔽缓存。

设定内存系统以及片子的片选，刷新周期，注意在配置刷新周期时一定要与具体板上扩展的SDRAM 相一致，否则片子无法正常工作。关闭缓存建立内存控制器。

在调试这段代码的时候，由于串口网口都没有启动，因此只能通过点灯程序来加跟踪程序的执行情况。

另外，在romStart()跳转到romInit()的C 程序之前尽可能少地初始化硬件。硬件的初始化是sysHwInit()程序完成的任务。

3)修改sysLib.c

sysLib.c 是BSP 初始化的核心代码，在这个文件中必须复位所有的硬件，使其处于初始化状态，以保证在后面中断使能之后，不会产生没有初始化的中断。它提供了板级接口，基于这些接口，VxWorks 和应用程序的构造与系统无关。该文件功能大致包括：初始化功能；存储器/地址映射功能；总线中断功能；时钟计数器功能等。如下是其中两个重要的函数sysHwInit()，sysHwInit2()以及代码注释。

SysHwInit()，它由通用初始化函数usrInit()调用：

```
void sysHwInit (void)
{
func—armIntStackSplit=sysIntStackSplit;
/*初始化IRQ/SVC 中断堆栈程序*/
#if defined(INCLUDE—PCI)
sysV3Init();/*初始化V3 PCI 桥接控制器 */
if(pciIomapLibInit(PCI—MECHANISM—3, CPU—PCI—CNFG—ADRS,CPU—PCI—CNFG—ADRS,0)
! = OK)
sysToMonitor (BOOT—NO—AUTOBOOT);
#endif
#ifdef INCLUDE—SERIAL
```

```
sysSerialHwInit ();  /* 初始化串口设备,该函数在sysSerial.c 中 */
#endif
}
```

串口设备的初始化在sysSerial.c 中完成初始化设备描述符;设置设备参数地址,波特率,以及设备中断级。

sysHwInit2(),连接系统中断,初始化其它必须的配置,该函数由定时器驱动中的sysClkConnect()调用。

```
void sysHwInit2 (void)
{
static BOOL initialised = FALSE;
/* 初始化中断库以及中断驱动 */
intLibInit (AMBA—INT—NUM—LEVELS, AMBA—INT—NUM—LEVELS, INT—MODE);
ambaIntDevInit();
/* 连接系统时钟中断以及辅助时钟中断 */
(void)intConnect (INUM—TO—IVEC (SYS--TIMER—INT—VEC), sysClkInt, 0);
(void)intConnect (INUM—TO—IVEC (AUX--TIMER—INT—VEC), sysAuxClkInt, 0);
#ifdef INCLUDE—SERIAL
/* 连接串口中断 */
sysSerialHwInit2();
/* 该函数也在sysSerial.c 中 */
#endif
initialised = TRUE;
}
```

4.3 VxWorks 的编译生成

对BSP 包里面的文件修改好以后,需要对其进行编译下载调试,装载到目标板中的VxWorks 映像取决于使用的下载方法,主要映像有:

1) VxWorks

这是下载型的VxWorks 映像,从RAM 开始执行,它由目标板上的引导程序通过串口或者网口将它下载到目标板本地的RAM 中运行。在Tornado 开发环境中,这是一个默认的选项,主要在调试阶段使用。使用宿主机上的WindShell 工具和符号表。

2) VxWorks.st

这也是基于RAM 的映像,需要通过引导ROM 将它下载到目标机执行,但该对象文件内置符号表。

3) VxWorks—rom

这是非压缩,基于ROM 的映像。在这个对象文件执行前把自己拷贝到目标机RAM 中。这种类型的映像通常在启动阶段比较慢,但其执行阶段都在RAM 中完成的,比驻留型的映像要快。

4) VxWorks.st—rom

这是ROM 驻留的压缩的VxWorks 映像。它在执行前把自己解压并拷贝到目标机RAM 中执行。

5) VxWorks.res—rom

这是ROM 驻留的非压缩VxWorks.st 的映像。它在执行前把自己数据段拷贝到目标机RAM 中。S3C2410 可以由Strata flash NOR flash 或者smart media card NAND flash 两种flash 卡启动,当系统映像下载到某个flash,通过适当的跳线从指定的flash 启动。

4.4 调试中需要注意的几个问题

在romInit.s 中初始化程序不要太多,它初始化的目的是为了载入VxWorks 映像,实际初始化任务主要由sysHwInit()函数来完成。对于下载型的VxWorks 映像,sysAlib.s()中的sysInit()函数应该重复映像下载前的所有初始化操作,否则在一些情况下可能会忽略存储控制器的设置。

对特定的BSP 驱动程序的修改,只能在特定的BSP 的目录下,在这里为target/config/integrator920t 目录下,不要直接在target/src/drv 以及target/h/drv 目录下修改。

调试阶段不要使用malloc()等缓冲区调用函数。如在前面所提到的函数intConnect()不能够在sysHwInit()中被调用,这是因为在VxWorks 未启动之前,此时整个系统的内存分配库还没有被初试化,调用malloc()等缓冲区调用函数,将会导致系统瘫痪。

结语

本文在介绍了BSP 组成以及VxWorks 映像的启动过程后,以S3C2410 开发板的BSP 为例,介绍了VxWorks 的BSP 设计中的几个重要文件的修改以及在BSP 开发调试中需要注意的几个问题。针对不同的目标板,BSP 的实现不尽相同,需要根据目标板的具体硬件结构进行具体设计,但各种目标板之间有一定的共性,希望本文可以对开发各类型的目标板具有一定的参考价值。

参考文献

[1]安军社,刘艳秋,孙辉先. VxWorks 操作系统板级支持包的设计与实现[J]. 计算机工程, 2003 (1) 87—89

[2]蒋巧文,潘孟春. 基于ARM 体系的嵌入式系统BSP 的程序设计[J]. 电子技术应用 2004 (9) 4—6

[3][美]Wind River. VxWorks BSP User Guide[M]. Tornado2.2 2002

[4][美]Wind River. VxWorks Programmer Guide[M]. Tornado2.2 2002

[5]周启平,张杨. VxWorks 下设备驱动程序及BSP 开发指南[M]. 中国电力出版社, 2004

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

(收稿日期:2005—5—9)