文章编号:1001 - 9081(2006)03 - 0746 - 03

一种 DSP 嵌入式多媒体应用系统板级支持包的研究

曹 荣,刘 峰
(南京邮电大学 信息工程系, 江苏 南京 210003)
(vincent_cr@126.com)

摘 要:在嵌入式系统中,硬件抽象层作为嵌入式操作系统和硬件之间的软件层次是嵌入式应用的一个关键问题。从硬件抽象层的原理切入,介绍了基于 Nexperia 系列数字信号处理器嵌入式多媒体应用系统中板级支持包的实现。

关键词:硬件抽象层;板级支持包;嵌入式系统;pSOS
中图分类号: TN919.82 **文献标识码**:A

DSP-based BSP in embedded multimedia application systems

CAO Rong, LIU Feng
(*Department of Information Engineering, Nanjing University of Posts & Telecommunications, Nanjing Jiangsu* 210003, *China*)

Abstract: As a software layer between embedded operation system and hardware in embedded system, HAL(Hardware Abstraction Layer) is a key problem of embedded application. With the introduction of HAL theory, the BSP(Board Support Package) which was realized in the embedded multimedia application system based on Nexperia DSP(Digital Signal Processor) was proposed.

Key words: HAL; BSP; embedded system; pSOS

由于嵌入式系统具有简洁、高效等特点,近年来在网络通信、生产过程控制、交通控制、仪器、仪表、汽车、船舶、航空、航天、军事装备、消费类产品等方面得到广泛的应用。由于嵌入式系统应用设计的特殊性,因此在设计实现过程中存在着许多特殊问题,其中操作系统及其他系统软件模块与硬件之间的接口形式是嵌入式系统的主要特征和系统设计过程中的必需环节,也是影响嵌入式系统应用前景的关键问题。经过近些年的发展,随着通用嵌入式操作系统技术的日趋成熟和应用的不断扩大,一种统一的接口形式——硬件抽象层(Hardware Abstraction Layer,HAL),得到广泛的认可和应用,这就是通常所说的板级支持包(Board Support Package,BSP)。

1 嵌入式系统硬件抽象层的原理

1.1 硬件抽象层的引入

嵌入式系统自上向下包含三个部分:

嵌入式应用程序 运行于嵌入式操作系统之上,根据不同的系统需要,利用操作系统提供的实时机制完成特定功能的应用程序。

嵌入式操作系统 完成嵌入式应用系统的任务调度和控制以及消息传递等核心功能,具有内核较精简、可配置、与高层应用紧密关联等特点。嵌入式操作系统具有相对不变性和可移植性。

硬件环境 整个嵌入式实时操作系统和实时应用程序运行的硬件平台;不同的应用通常有不同的硬件环境;硬件平台的多样性是嵌入式系统的一个主要特点。

由于嵌入式系统应用的硬件环境差异较大,简洁有效地使嵌入式系统能应用于各种不同的应用环境是嵌入式系统发展中的关键问题。经过不断的发展,原先嵌入式系统的三层结构逐步演化成为一种四层结构。这个新增加的中间层次位于操作系统和硬件之间,包含了系统中与硬件相关的大部分功能,通过特定的上层接口与操作系统进行交互,向操作系统提供底层的硬件信息,并根据操作系统的要求完成对硬件的直接操作。由于引入了一个中间层次,屏蔽了底层硬件的多样性,操作系统不再直接面对具体的硬件环境,而是面向由这个中间层次所代表的、逻辑上的硬件环境。因此,把这个中间层次叫作硬件抽象层(HAL),通常也叫作板级支持包(BSP)。图 1 显示了引入 HAL 以后的嵌入式系统结构。BSP 的引入推动了嵌入式操作系统的通用化,为嵌入式系统的广泛应用提供了可能。

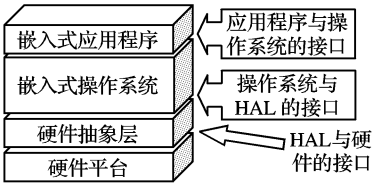

图 1 引入 HAL 后的嵌入式系统结构

1.2 BSP 的特点与功能

HAL/BSP 的提出使通用的嵌入式操作系统及高层的嵌入式应用能够有效地运行于特定的、应用相关的硬件环境之上,使操作系统和应用程序能够控制和操作具体的硬件设备,完成特定的功能。因此,在绝大多数的嵌入式系统中,BSP 是一个必不可少的层次。

由于在系统中的特殊位置,BSP 具有以下主要特点:

1) 硬件关联性

因为嵌入式系统的硬件环境具有应用相关性,所以,作为高层软件与硬件之间的接口,BSP 为操作系统提供操作和控制具体硬件的方法。

收稿日期:2005 - 07 - 14;**修订日期**:2005 - 08 - 29

作者简介:曹荣(1978 -),男,安徽芜湖人,硕士研究生,主要研究方向:多媒体通信、图像处理; 刘峰(1964 -),男,浙江温州人,副教授,博士,主要研究方向:多媒体通信、图像处理.

2）操作系统关联性

不同的操作系统具有各自的软件层次结构,因此,不同的操作系统具有特定的硬件接口形式。在实现上,BSP 是一个介于操作系统和底层硬件之间的软件层次,包括了系统中大部分与硬件相关的软件模块。在功能上包含两部分:系统初始化和与硬件相关的设备初始化。

2　基于 Nexperia PNX13xx DSP 的 BSP 实现

为实现上述两部分功能,设计一个完整的 BSP 需要完成两部分工作:一是设计初始化过程,以完成嵌入式系统的初始化;二是设计硬件相关设备驱动,以完成操作系统及应用程序对具体硬件的操作。

2.1　嵌入式系统 BSP 的设计

嵌入式系统 BSP 的设计过程实际上就是一个包括 CPU 及其相关硬件的初始化和软件(主要是操作系统及系统软件模块)初始化的过程,而操作系统启动以前的初始化操作是 BSP 的主要功能之一。由于嵌入式系统不仅具有硬件环境的多样性,同时具有软件的可置性,因此,不同的嵌入式系统初始化所涉及的内容各不相同,复杂程度也不尽相同。但是初始化过程总是可以抽象为三个主要环节,按照自底向上、从硬件到软件的次序依次为:芯片级初始化、板级初始化和系统级初始化。

1）芯片级初始化:主要完成 CPU 的初始化,包括设置 CPU 的核心寄存器和控制寄存器,CPU 核心工作模式等。芯片级初始化把 CPU 从上电时的缺省状态逐步设置成为系统所要求的工作状态。这是一个纯硬件的初始化过程。

2）板级初始化:完成局部总线的初始化以及 CPU 以外的其他硬件设备,如视频输入/输出芯片、音频输入/输出芯片的初始化。主要设置外围接口芯片的控制寄存器,除此之外,还要设置某些软件的数据结构和参数,为随后的系统级初始化和应用程序的运行建立硬件和软件环境。这是一个同时包含软硬件两部分在内的初始化过程。

3）系统级初始化:这是一个以软件初始化为主的过程,主要进行操作系统初始化。BSP 将控制转交给操作系统,由操作系统进行余下的初始化操作。包括加载和初始化系统设备驱动程序,建立系统内存区,加载并初始化其他系统软件模块,比如网络系统、文件系统等;最后操作系统创建应用程序环境,并将控制转交给应用程序的入口。

经过以上三个层次的操作,嵌入式系统运行所需要的硬件和软件环境已经得到正确设置,接下来便可以运行高层的实时应用程序了。需要指出:系统级初始化设计并不是设计 BSP 的工作。但是,系统级初始化成功与否的关键在于前两个初始化过程中所进行的软件和硬件设置是否正确,而且系统级初始化也是由 BSP 发起的,因此,设计 BSP 主要集中在开始两个环节。

2.2　Nexperia 系列 DSP 嵌入式系统 BSP 结构

在 Nexperia 系列 DSP 嵌入式系统中所采用的操作系统为 pSOS,该操作系统是一个模块化、高性能的实时操作系统,提供了基于开放系统的标准多任务环境,可通过串口或 TCP/IP 的网络连接来进行通信。它具有可裁减性,用户可根据应用需要选择使用其提供模块的任意组合,特别适用于内嵌式微处理器。

BSP 是 Nexperia 系列 DSP 嵌入式系统硬件级的 API 库,它分为 Device Library BSP 和 pSOS BSP,图 2 为系统框图。通常我们说的 BSP 指的只是 Device Library BSP。Device Library BSP 提供了 Device Library 与硬件系统之间的接口。开发者使用 Nexperia 系列 DSP 设计不同应用系统时,不需要修改高层应用软件和 Device Library,只需要修改 Device Library BSP。

pSOS 是不假设硬件的,所以 pSOS 和硬件之间有 pSOS BSP。理论上 pSOS BSP 为 pSOS 提供访问硬件的 I/O 接口。但实际应用中,pSOS BSP 已经将 pSOS 融合其中。

pSOS BSP 提供的功能包括:加载初始化系统设备驱动程序;pSOS configuration 代码;加载 pSOS 的系统成分,例如多任务内核、TCP/IP 模块等; pSOS boot 代码;硬件访问库(例如,安装中断句柄和系统定时器)。Bsp. a 提供了实现以上功能的实体函数。

图 2　系统框图

2.3　Nexperia PNX13xx 嵌入式系统 BSP 的实现

1）芯片级初始化

PNX13xx 内部系统启动逻辑单元:执行最低级的初始化工作,在 PNX13xx DSP 系统工作之前,必须配置 PNX13xx 的 MMIO 寄存器、时钟分频寄存器;通过 I^2C 总线从 EEPROM 中读取启动信息,确定是从 JTAG 口还是从 FlashMemory 读取应用程序代码;然后从 JTAG 口或 FlashMemory 将可执行代码送到 SDRAM;顺序执行程序代码。

EEPROM:用于存放启动信息。PNX13xx 有两种工作模式:通过 PCI 总线接入 PC 平台的工作模式和作为独立机型中的主 CPU 的工作模式。

以独立机工作模式为例:

系统上电后,PNX13xx 开始启动。在启动阶段,只有系统时钟、I^2C 接口、PCI 接口允许操作,片内的 DSPCPU 和内部数据总线依然保持在复位状态,直到它们被启动过程所激活。

在 PNX13xx 启动后,它通过 I^2C 总线读取 EEPROM 中前 10 字节的信息,根据该信息来确定 EEPROM 和 SDRAM 的大小、外部时钟速度、EEPROM 的 I^2C 最大时钟速度、测试模式、子系统标识、系统厂商标识、MM-CONFIG 寄存器初始值、PLL-RATION 寄存器初始值、工作模式。外部时钟速度的设置应最接近 PNX13xx 所接入的实际时钟速度,它和随后的 EEPROM 最大时钟速度共同决定 I^2C 时钟的速度。MM-CONFIG 寄存器初始值和 PLL-RATION 寄存器初始值用于控制主存接口和 PNX13xx 内部时钟电路。

当系统确定 PNX13xx 是作为独立机型中的主 CPU 后,PNX13xx 接着从 EEPROM 中读取完成系统启动的其余信息。最后把 EEPROM 中的引导程序调入到 SDRAM 中,PNX13xx 开始顺序执行 EEPROM 中存储的引导程序。引导程序则把 FlashMemory 中的系统程序调入 SDRAM 执行。

2）板级初始化

PNX13xxbsp. a 是与外围芯片驱动程序相关联的部分,完成板级初始化,分为:目标板注册文件、外围芯片配置文件。由编译环境 CodeWarrior 对它们编译、链接生成。

① 目标板注册文件:RemoteVideo_PNX13xx_Board. c,由如下的输出宏链接到设备库:

```
TSA_COMP_DEF_O_COMPONENT( RemoteVideo_PNX13xx,
                                        /* 目标板名称 */
  TSA_COMP_BUILD_ARG_LIST_1( "bsp/boardID"),
                                        /* 目标板输出符号 */
  RemoteVideo_PNX13xx_board_activate) /* 目标板激活函数 */
```

目标板激活函数 RemoteVideo_PNX13xx_board_activate 完成目标板的检测、初始化及注册。包括如下三个函数：

```
TRY( RemoteVideo_PNX13xx_board_detect( ) );
TRY( RemoteVideo_PNX13xx_board_init( ) );
TRY( RemoteVideo_PNX13xx_board_register( ) );
```

其中注册函数 RemoteVideo_PNX13xx_board_register( ) 完成所支持外围设备的注册：

```
TRY( tsaBoardRegisterAO (0, & RemoteVideo_PNX13xx _vo) );
                                        /* 音频输出 */
TRY( tsaBoardRegisterAI (0, & RemoteVideo_PNX13xx _vi) );
                                        /* 音频输入 */
TRY( tsaBoardRegisterSPDO(0,
  &RemoteVideo_PNX13xx_SPDO) );      /* SPDIF 输出单元 */
TRY( tsaBoardRegisterVO (0, & RemoteVideo_PNX13xx _vo) );
                                        /* 视频输出 */
TRY( tsaBoardRegisterVI (0, & RemoteVideo_PNX13xx _vi) );
                                        /* 视频输入 */
TRY( tsaBoardRegisterBoard ( ID, "RemotVideo PNX13xx Board") );
                                        /* 目标板 ID */
```

以视频输出为例，由于系统中视频输出的 D/A 芯片是 SAA7121H，所以 VO 的注册数据结构 RemoteVideo_PNX13xx_vo 如下：

```
boardVOConfig_t RemoteVideo_PNX13xx_vo = {
  "SAA 7121",                           /* 芯片名称 */
  RemoteVideo_PNX13xx_VO_Init,          /* 初始化函数 */
  RemoteVideo_PNX13xx_VO_Term,          /* 关闭函数 */
  saa7121SetBrightness,                 /* 设置亮度 */
  saa7121Configure,                     /* 配置函数 */
  SAA7121_SUPPORTED_STANDARDS,
                                        /* 支持的视频信号制式 */
  SAA7121_SUPPORTED_ADAPTERS,  /* 支持的信号方式 */
  intVIDEOOUT,                          /* 中断号 */
  VO_STATUS,                            /* mmio 基址 */
};
```

② 外围芯片配置文件：包括多个 C 文件，通常每个 C 文件完成对一个外围芯片的所有操作细节，如 SAA7121. c，SAA7113. c 等。以 SAA7121. c 为例，完成视频处理芯片所有操作细节，包含如下几个函数：saa7121Config( )，saa7121Init( )，saa7121Start( )，saa7121Stop( )，saa7121SetBrightness( ) 等。这些函数通过 I^2C 总线对 SAA7121 的内部控制寄存器进行设置，真正完成对外围芯片的设置。

3）系统级初始化

对操作系统初始化过程主要由 sysinit. c，drv_conf. c，bsp. c三个 C 文件完成。

sysinit. c：这个文件的代码初始化操作系统的配置。大部分配置基于两个头文件：sys_conf. h 中的选项和 bsp. h 所定义使用的板级支持包。

drv_conf. c：对 pSOS 系统设备配置和初始化，主要为 SetUpDrivers（配置网络设备以外的设备驱动）和 SetUpNI（配置网络设备驱动）两个函数。

bsp. c：包含一个主要函数 RtcInit( )，安装实时时钟（real time clock）句柄到中断表。

系统级初始化（图 3）：由 main( ) 函数进入对 pSOS 系统的配置和初始化，由 SysInit( ) 完成并屏蔽所有的中断。其中 BuildConfigTable( ) 使用系统变量信息建立系统成分配置表，根据应用需要调用成分配置函数初始化 pSOS，同时配置安装网络设备和其他非网络设备并进行初始化。最后，由 _psos_start( ) 函数进入操作系统运行并执行用户应用程序。

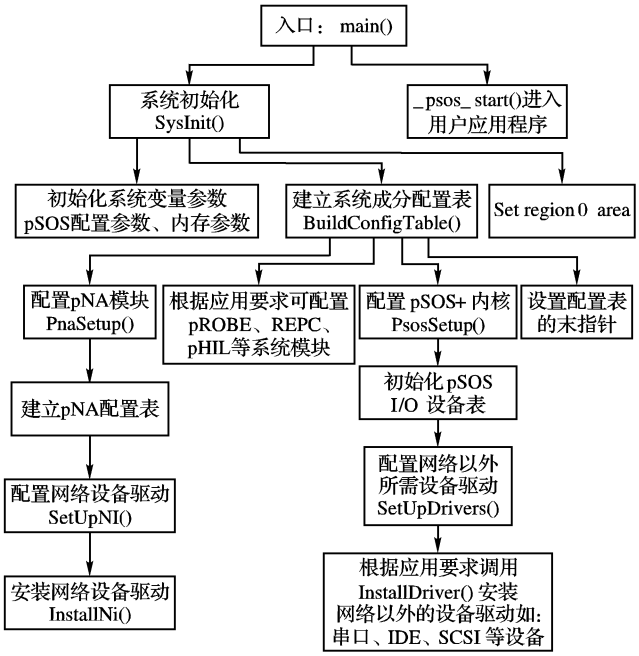

图 3 系统级初始化

2.4 Nexperia 嵌入式系统 BSP 的应用

在应用系统中，Device Library 接口函数向高层软件提供了统一的接口，它通过调用 BSP 实现对硬件的操作，它本身不依赖于硬件系统。开发者无需修改 Device Library 接口函数，只需根据接口函数找到其在 BSP 中的具体实现，针对目标板设计修改相应的 BSP 实现。如原来目标板上所用视频输出的 D/A 变换芯片是 SAA7185 改为 SAA7121H，需做一些修改：首先，修改目标板的注册文件 RemoteVideo_PNX13xx_Board. c，所有与视频输出有关的注册函数、数据结构都改为 SAA7121；其次，剔除 BSP 库中原来的 Saa7185. c 文件，针对 SAA7121H 芯片编写 Saa7121. c 文件，包括所有对芯片的操作细节，用它代替 Saa7185. c 文件。通过以上修改，构成针对新目标板的 BSP。

对 pSOS BSP 的配置仅在于对 pSOS 内核的配置，无需考虑与硬件接口，这通过修改 sysinit. c、sys_conf. h 来实现。

这样就不需要对应用软件进行任何修改，只要把应用软件利用新的 BSP 重新编译一遍就行了，对应用软件的移植就很方便。

3 结语

在嵌入式应用系统开发的过程中，对嵌入式系统基本原理的深刻理解有利于设计优化。本文详细辨析了嵌入式系统硬件抽象层的原理，最后在上述分析的基础上给出了基于 Nexperia 系列 DSP 嵌入式多媒体应用系统 BSP 实现实例，此 BSP 已在基于 IP 网络的远程视频监控系统中成功实现。

参考文献：

[1] 王涛，张伟良，冯重熙. 嵌入式系统硬件抽象层的原理与实现[J]. 电子技术应用，2001，(10)：26－28.

[2] 胡明. 用于远程监控的视频编码盒的设计与实现[D]. 南京邮电大学，2002.

[3] Philips TriMediaTM Documentation Set[M/CD]. Philips Semiconductors，1999.