时间：2009-12-28　来源：单片机与嵌入式系统 作者：林凯宏,游林儒,胡庆华 华南理工大学
关键字：EMC2　硬件　抽象层　原理

1 概 述

嵌入式系统具有专用性强、外围设备多样的特性，这决定了其应用的硬件环境差异性较大。系统软件模块与硬件之间的接口是嵌入式实时系统的主要特征，是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之一。硬件抽象层(Hardware Abstraction Layer，HAL)的引入可有效解决这一问题。HAL 是将硬件平台与应用软件隔离开来的软件层次，通过硬件抽象技术实现硬件相关和硬件无关两部分程序代码的隔离，为应用软件提供一个没有硬件特性的接口。硬件抽象层的引入不仅是系统体系结构设计方法的改进，更直接关系到整个系统的开发模式以及嵌入式操作系统的可移植性。硬件抽象层的引入大大推动了嵌入式系统开发的规范化进程。

QQ群：30237713　LinuxCNC/EMC2
论坛：http://www.linuxcnc.cn/

EMC 是一个开放源代码的用于机床或机器人等运动控制系统的计算机控制软件。它能同时驱动 9 轴电机。其运动控制特性包括：刀具半径和长度补偿、轴同步运动、自适应进给速度、恒速度控制等。EMC2 在原有 EMC 软件的基础上加入了许多新的特性和功能，其中包括了 HAL 和软件 PLC 模块 ClassicLadder。ClassicLadder 是一个基于 LGPL 协议的梯形图解释器。它随着 EMC2 一起发布，可以与 EMC2 的 HAL 一起工作。本文中的控制系统利用 EMC2 的 HAL 为软 PLC 中的应用程序提供底层硬件操作支持，提高了应用程序的平台无关性与可移植性。

2 硬件架构

控制器是锂电池卷绕恒张力控制器，采用符合 PC / 104 总线规范的单板计算机(以下简称 PC104)与基于 FPGA 的专用主机板相结合的方法构建系统硬件。PC104 中运行实时 Linux，ClassicLadder 及 HAL 作为实时模块加载到 Linux 系统中。

系统硬件框图如图 1 所示。其中 ADS8361 为 12 位模 / 数转换器，用于采集张力值等模拟量；AD5624 为数 / 模转换器，用于控制直流电机转速及气压阀压力值；FPGA 控制所有外围芯片，并产生电机脉冲方向信号，同时对电机编码器信号进行计数；CPLD 控制 I / O 输入 / 输出点，并与 FPGA 交换信息。利用 EMC2 中 HAL 的实现原理，可编写组件将硬件系统所有设备抽象成引脚和函数的形式，供软 PLC 在需要时加载。

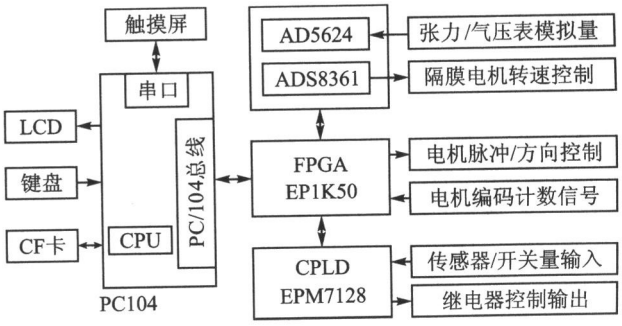

图 1　控制器硬件结构框图

3 EMC2 中 HAL 的基本概念

EMC2 的 HAL 提供了一种简便方法，将一些已有软硬件模块进行加载和组合形成一个复杂的系统，从而使 EMC2 更容易配置，以使用各种硬件设备。硬件资源在 HAL 中被封装成特定组件，随时被控制系统载入使用。EMC2 中的 HAL 有以下基本概念：

Component，组件。是定义好输入、输出及行为的软件模块，可以在需要时安装及连接。

Parameter，参数。许多硬件组件有可调整参数需要进行访问。HAL 有输入及输出两种类型的参数。

Pin，引脚。硬件组件用于互联的连接端子。所有 Pin 都有名称，并在连接时使用。HAL 的 Pin 是只存在于计算机内的软件实体。

Physical_Pin，物理引脚。许多 I / O 设备有真正的物理引脚或终端连接到外部硬件，这些被称为物理引脚。

Signal，信号。现实中硬件组件使用导线互连。在 HAL 中导线相当于“信号”。HAL 的信号将 HAL 的引脚连

接在一起，可以随意断开或重新连接。

Type，类型。引脚和信号都有类型属性，即信号只能连接到相同类型的引脚。目前，HAL 有 4 种类型：BIT、FLOAT、U32、S32。

Function，函数。每个函数是一个执行具体行为的代码块，执行读取输入、计算输出等操作。系统设计者可以使用“线程”对一系列函数加以调度，以使其按照特定的顺序及时间间隔运行。

Thread，线程。作为一个实时任务的组成部分，线程是一个以特定时间间隔运行的函数序列。函数可以添加到线程并在每次线程运行时调用。

4 HAL 架构

QQ 群：302377173 LinuxCNC/EMC2
教程：http://www.linuxcnc.cn/

系统软件架构如图 2 所示。用 HAL 将各 I / O 通道、ADC 通道、DAC 通道、脉冲通道、编码器通道抽象成 Pin，将对硬件各模块的操作抽象成各个 Function，将 Pin 和 Function 封装在命名为 hal_CNC 的 Component 中。

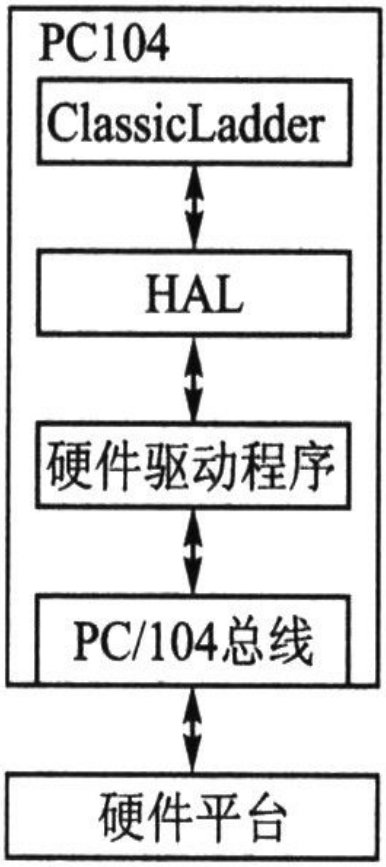

图 2　系统软件架构

```
typedef struct {
    hal_u32_t *  width;              //脉宽
    hal_u32_t *  total;              //脉冲个数
}pulse_struct;
typedef struct{
    hal_float_t  * dac_value[8];     //DAC 输出
    hal_float_t  * adc_value[12];    //ADC 读入
    hal_bit_t  * IO_in[128];         //I/O 输入
    hal_bit_t  * IO_out[120];        //I/O 输出
    hal_u32_t  * EncoderCount[8];    //编码器读数
    pulse_struct pulse[8];           //脉冲输出
} CNC_struct;
static CNC_struct  * CNC_driver;
```

uxCNC/EMC2
xcnc.cn/

由上述结构可以看出，每个 Pin 对应一个相应类型的指针，该指针指向的内存区便存放该引脚的值。

5 基于 HAL 的驱动程序编写

hal_CNC 由源文件 hal_CNC. c 和 hal_CNC. h 构成。hal_CNC. c 定义了对 hal_CNC 的初始化、底层硬件驱动函数、hal_CNC 退出时的操作等。rtapi_app_main()函数是载入 Component 时的程序入口。

1 首先申请当前 Component 的 ID 号，对 Component 的每个操作都由该 ID 号索引。

comp_id=hal_init(“hal_CNC”);

/ / hal_CNC 为 Component 的名字

2 其次，分配组件运行时所需内存，代表 Pin 的指针指向的内存区域便是在此分配：

CNC_driver=hal_malloc(sizeof(CNC_struct));

3 接着将所定义 Pin 导出到 HAL。该操作通过调用自定义的 export()函数来完成。以 DAC 模块为例，操作如下：

```
for(n=0; n<8; n++)
export_dac(n, CNC_driver);   / / 导出 Pin
```

每次调用 export_dac()时，都会调用如下语句注册一个 Pin。

```
rtapi_snprintf(buf, HAL_NAME_LEN,“CNC. DAC. %d. value”, num);
hal_pin_float_new(buf, HAL_IN, &(addr->dac_value[num]), comp_id);
```

其中“CNC. DAC. XXX. value”是 Pin 的名称。软 PLC 通过该名称对此 Pin 进行引用。hal_pin_float_new()是 HAL 提供的函数，在新建 Type 为 FLOAT 型的 Pin 时使用。该函数一共有 4 个参数，依次是 Pin 名、Pin 方向、Pin 内存指针地址、Component ID。引脚方向 HAL_IN 表示该值是从软件层“输入”到 HAL 中的，该方向针对软件层与 HAL 层而言。

对 Pin 进行操作的 Function 也要导出到 HAL：

hal_export_[unct(“CNC.DAC.write”,CNC_dac_write,CNC_driver,1,0,comp_id);其中“CNC.DAC.write”为软件层使用该 Function 时引用的名字；CNC_dac_write 为函数在 C 源文件中实际对应的 C 函数名称；CNC_driver 为 Component 的内存指针；1 表示函数用到了浮点数；0 表示该函数不可重入；comp_id 为 Component ID。

依照上述做法将所有硬件功能模块全部导出到 HAL 后，在 rtapi_app_main()的最后调用 hal_ready(comp_id)，表明该 Component 已经初始化完毕，可以开始使用了。

在关闭 Component 退出时，系统会自动调用 hal_CNC. C 中编写的 rtapi_app_exit()。其实现如下：

```
void rtapi_app_exit(void){hal_exit(comp_id); }
```

hal_exit()关闭并释放 HAL 及 RTAPI 使用到的系统资源，使这些资源可被重新使用。

编译工具 comp

用 EMC2 自带的工具 comp 对源文件 hal_CNC.c 和 hal_CNC.h 进行编译，即可得到名为 hal_CNC 的 Compo-nent。该组件自动放入 EMC2 的模块库中，随时可被其他软件模块调用。

编译居然有专用工具，而不是将该版本的内核+驱动重新编译

6 HAL 的使用

以 DAC 为例，在 Linux 下的命令行输入“halrun”进入 EMC2 的 HAL 运行界面，输入：

```
loadrt threads name1=thread period1=1000000
```

创建名为“thread”的线程，该线程执行周期为 1 ms。

执行：

```
loadrt hal_CNC
```

将所编写的硬件系统组件调入，执行：

```
addf CNC. DAC. write thread
```

将 DAC 的写函数加入到前面创建的线程 thread，使之以与 thread 相同的执行周期被调用。然后使可通过控制 DAC 的引脚来输出相应的电压。如：

```
setp CNC. DAC. 0. value 1
```

该语句将使电路板上的 DAC 输出端子输出 1 V 的电压。

用类似的方法将其他软件模块通过与 HAL 的引脚连接，便实现了其他软件对 HAL 的调用。

7 HAL 在 ClassicLadder 中的调用

以从 DAC 输出 5 V 为例，将 classicladder 的一个名为“classicladder. 0. s320ut-00”的有符号 32 位整型 Pin 赋值为 5。该值经过 HAL 中的一个类型转换 Component“s32tofloat”变为浮点数，再连接到 hal_CNC 中的 DAC 单元的引脚“CNC. DAC. 0. value”，便在实际硬件电路板的 DAC 输出端输出 5 V 的电压。引脚连接如表 1 所列。

表 1　软 PLC 控制 DAC 输出所用的引脚连接

comp_id	类型	方向	数值	映射关系
3	s32	OUT	5	classicladder. 0. s32out-00→ W10s32
4	s32	IN	5	s32tofloat. 0. in0←W10s32
4	float	OUT	5	s32tofloat. 0. out0→W10float
2	float	IN	5	CNC. DAC. 0. value←W10float

其中“→”和“←”表示引脚之间的连接，用 HAL 中的 Sig-nal 实现。

在软 PLC 中设置变量 W10 的值为 5，则在 DA 输出端子引脚上用万用表测到 5 V 的电压。软 PLC 中的操作输出如图 3 所示。

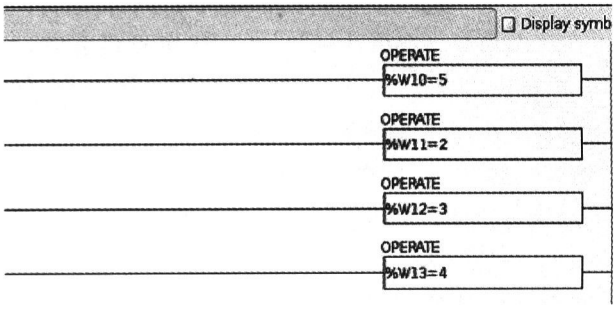

图 3　软 PLC 中 DAC 控制界面

QQ 群：302377173　LinuxCNC/EMC2

其中 4 个窗口表示 DAC 的 4 个通道，分别令 DAC 输出 5 V、2 V、3 V、4 V 的电压。

8 结 论

实践证明，HAL 的引入可极大提高嵌入式软件实现的硬件无关性。从软件的角度来看，其面向的硬件具有同质的接口，对硬件的操作具有相似的方法与架构，极大地简化了软件对硬件的控制，方便了同类软件在不同硬件平台间的移植。这就为软硬件同步设计、分工协作奠定了良好的基础。该架构已成功应用在文中所述的锂电池卷绕恒张力控制器中，取得了良好效果。

QQ 群：302377173　LinuxCNC/EMC2
论坛：http://www.linuxcnc.cn/

QQ 群：302377173　LinuxCNC/EMC2
论坛：http://www.linuxcnc.cn/