第10章 底层驱动构件制作过程

应用层软件不直接干预硬件，若需要干预硬件，则通过底层驱动构件的API接口与硬件打交道，微控制器的底层驱动构件通过对映像寄存器的读写直接干预硬件。同样功能模块，在不同微控制器中的映像寄存器的名称、地址、数量、各位含义、操作顺序等，差异很大，属于个性化技术。但是，应用层编程则是统一的，面向功能的，必须通过构件封装技术将个性化差异屏蔽在构件内部，这是一个比较复杂的过程，本章在讨论底层驱动构件制作的一般方法基础上，给出GPIO、定时器、串口构件的制作过程，希望可以起到举一反三的作用。

10.1 底层驱动构件制作的通用基础知识

底层驱动构件制作属于微型计算机原理中较深层次的内容，但又是不可或缺的。掌握底层驱动构件制作方法，对深入理解计算机基本运行原理有很大帮助。

10.1.1 底层驱动构件概念及作用

举例来说，CH573F微控制器的PTB14引脚接一个发光二极管，如图10-1所示。蓝色发光二极管LED的负极接MCU的引脚，其正极过限流电阻R1（1KΩ）接电源Vcc（3.3V），这样只要编程使得引脚PTB14=0（低电平，相当于地），则发光二极管导通发光，若编程使得引脚PTB14=1（高电平，接近Vcc），则发光二极管不导通。

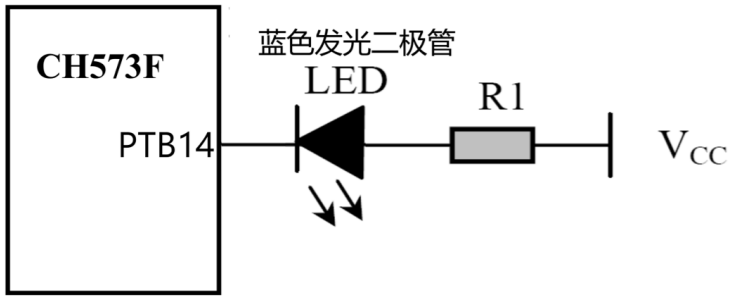


图 10-1　CH573F 芯片的 PTB14 引脚接一个发光二极管

现在的关键问题是，如何使得引脚PTB14=0或者PTB14=1？要按照一定的编程步骤，对CH573F的GPIO模块的映像寄存器进行操作，才能完成对引脚PTB14干预。同时，对其他引脚也需要同样编程，若均使用映像寄存器直接地址，就过于个性化，不利于应用层程序的可复用可移植。为此，需要使用函数传参的方式把对引脚的直接干预封装起来，提供对外**应用程序接口（Application Programming Interface，API）**，方便应用层编程调用，提高应用层程序的可复用性可移植性。把同一模块的若干函数按照一定规则设计好，放在一个文件中（.c文件），在把使用说明放在另一个文件中（.h文件），就形成了一个**底层驱动构件**。

有了这个底层驱动构件，就可以调用其API进行应用层程序的编程，使得软件编程划分成使用底层驱动构件的编程与底层驱动构件制作两个独立层次。底层驱动构件的制作具有一定难度，但是通过合适的方法，也可以比较简单顺利地进行。

10.1.2 底层驱动构件制作的基本知识

1. 模块功能与原理

若要制作一个底层驱动构件，首先必须基本理解模块功能。比如，GPIO模块，要知道GPIO的基本含义，即可以通过编程把MCU的引脚定义成输入还是输出，若是输出可以打出高电平（1）或低电平（0）若是输入，可以通过该引脚获得芯片外部状态为高电平（1）或低电平（0）。

2. 模块寄存器

在芯片的硬件设计中，一个功能模块，对外提供若干映像寄存器 ，每个寄存器有对应地址。要进行底层驱动构件设计，就必须查芯片手册该模块有哪些寄存器，功能是什么？许多寄存器的不同位含义不同，相当一部分寄存器需要按照一定操作顺序，其间还要有一定的延时等。同样功能模块，针对不同芯片，这个编程过程差异很大，属于个性化工作。主要难度就在这里。

可以参考本章后面的例子，突破这个难点。例如，GPIO模块，要使得一个引脚输出低电平，至少涉及引脚方向寄存器与引脚输出控制寄存器，制作构件时在手册里找到寄存器的地址，寄存器各位的含义，编程干预之。不同芯片，对应寄存器名称、地址、含义均有所不同，必须在深入理解模块功能与原理的基础上，参考芯片手册进行底层驱动构件的编程。大部分公司的芯片手册都比较冗长，风格不一，因此制作底层驱动构件难度较大。也许将来的芯片会简化这部分工作，使得芯片的使用者专注于芯片应用程序的设计。

3. 模块的API分析

制作底层驱动构件的目的在于使得芯片应用程序的开发者可以简单方便地通过它干预硬件。API应该是面向功能的，而不是面向芯片的。也就是说，它必须把芯片的操作细节屏蔽在内部，不外露。模块的API分析，就是要从模块功能的角度，分析出需要哪些对外函数，各个函数应该有哪些形参，形参类型是什么等。例如，根据GPIO的功能与原理，GPIO构件的API，应该提供初始化、设置引脚状态、获取引脚状态、设置引脚上下拉、设置引脚中断类型、使能引脚中断、禁止引脚中断等函数；对于初始化函数，应该具备引脚名、引脚方向、引脚初始状态等形参。分析之后，按照构件制作基本规范，给出函数名、类型、形参名等英文全称或简写，为构件封装做好准备。

10.1.3 构件制作的基本要求与一般步骤

1. 构件制作的基本要求

（1）关于可以移植性。使用同一芯片的应用系统，构件不更改，直接使用；同系列芯片的同功能底层驱动移植时，仅改动头文件；不同系列芯片的同功能底层驱动移植时，头文件与源程序文件的改动尽可能少；

（2）关于编码书写规范。要有统一、规范的编码风格与注释，主要涉及文件、函数、变量、宏及结构体类型的命名规范；涉及空格与空行、缩进、断行等的排版规范；涉及文件头、函数头、行及边等的注释规范。

（3）关于宏的使用限制。宏使用具有两面性，有提高可维护性一面，也有降低阅读性一面，不要随意使用宏。

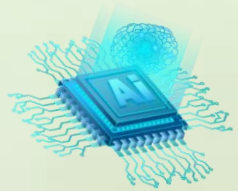
（4）关于全局变量问题。构件封装时，应该禁止使用全局变量。

2. 底层驱动构件制作的一般步骤

（1）拷贝构件制作模版。
（2）进行第一个模块映像寄存器的写与读。
（3）选择最容易看到现象的一条流程进行编程。
（4）进行构件API分析。
（5）进行函数封装与构件测试。

10.2 GPIO构件制作过程

10.2开始，进行练习



本章作业：1~3