得程序维护变得容易。另外，由于立即数存储的减少，编译出的程序代码也会变小。

若进一步处理可以将外设操作封装成函数的形式，多个外设实体将其基指针作为参数进行函数调用。例如，利用串口发送一个字节数据，可以封装成如下的函数：

```
void USART_SendData(USART_TypeDef * USARTx, uint16_t Data)
{
    /* Check the parameters */
    assert_param(IS_USART_ALL_PERIPH(USARTx));
    assert_param(IS_USART_DATA(Data));

    /* Transmit Data */
    USARTx->DR = (Data & (uint16_t)0x01FF);
}
```

后面要介绍的 STM32 固件库函数就是基于这种思路设计的。

应该注意的是，在定义 USART_TypeDef 结构时，使用了__IO 变量类型，该类型实质上是 volatile 的宏定义(该宏定义包含在 core_m3.h 文件中)。外设访问定义指针时，需要使用 volatile 关键字。volatile 用于防止相关变量被优化。例如对外部寄存器的读写。对有些外部设备的寄存器来说，读写操作可能都会引发一定硬件操作，但是如果不加 volatile，编译器会把这些寄存器作为普通变量处理，例如连续多次对同一地址写入，会被优化为只有最后一次写入。另一个使用场合是中断。如果一个全局变量，在中断函数和普通函数中都用到过，那么最好对这个变量加 volatile 修饰。否则在普通函数中，可能会仅从寄存器里读取这个变量以便加快速度，而不去实际地址读取该变量。

3. Cortex 微控制器软件接口标准(CMSIS)

1) CMSIS 简介

随着嵌入式系统软件复杂性的增加，软件代码的兼容性和可重用性变得更加重要。可重用的程序能够减少后续项目的开发时间，也就加快了产品推向市场的速度；而软件的兼容性则有助于第三方软件组件的使用。

为了使软件产品具有高度的兼容性和可移植性，ARM 公司与许多微控制器供应商和软件方案供应商共同努力，开发了一个通用的软件框架 CMSIS，该框架适用于大多数的 Cortex-M 处理器以及 Cortex-M 微控制器产品。CMSIS 针对处理器特性提供了标准化的操作函数。

CMSIS 一般是作为微控制器厂商提供的设备驱动库的一部分来使用的。为了使用诸如 NVIC 和系统控制功能等处理器特性，CMSIS 提供了一种标准化的软件接口。CMSIS 对多种微控制器厂商都是统一的，并已被多种 C 编译器和开发工具(包括 Keil MDK)所支持。

2) CMSIS 的标准化

CMSIS 为嵌入式软件提供了以下标准化的内容：

- 标准化的操作函数，用于访问 NVIC、系统控制块(SCB)，SysTick 的中断控制和初始化。
- NVIC、SCB 和 SysTick 寄存器的标准化定义，为了达到最佳的可移植性，应该使用这些标准化的操作函数；不过，有些情况下，需要直接操作 NVIC、SCB 和 SysTick