Stellaris® ARM® Cortex™-M4F 培训

USB

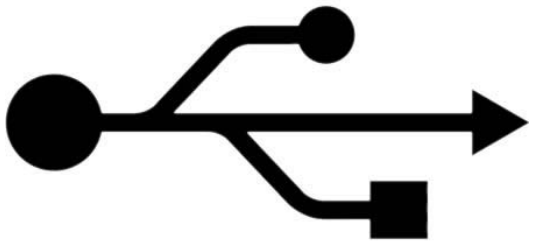
目的

- Stellaris LM4F 的 USB 功能概述
- StellarisWare USB 库概述
- 实验例程演示：
  - 如何采用一个现有的 USB 例程启动开发
  - StellarisWare 中与USB相关的最重要文件位置与用途
  - 如何变更其中的某些文件以适合某种新型应用的需要
  - 到哪里去查找更多的信息

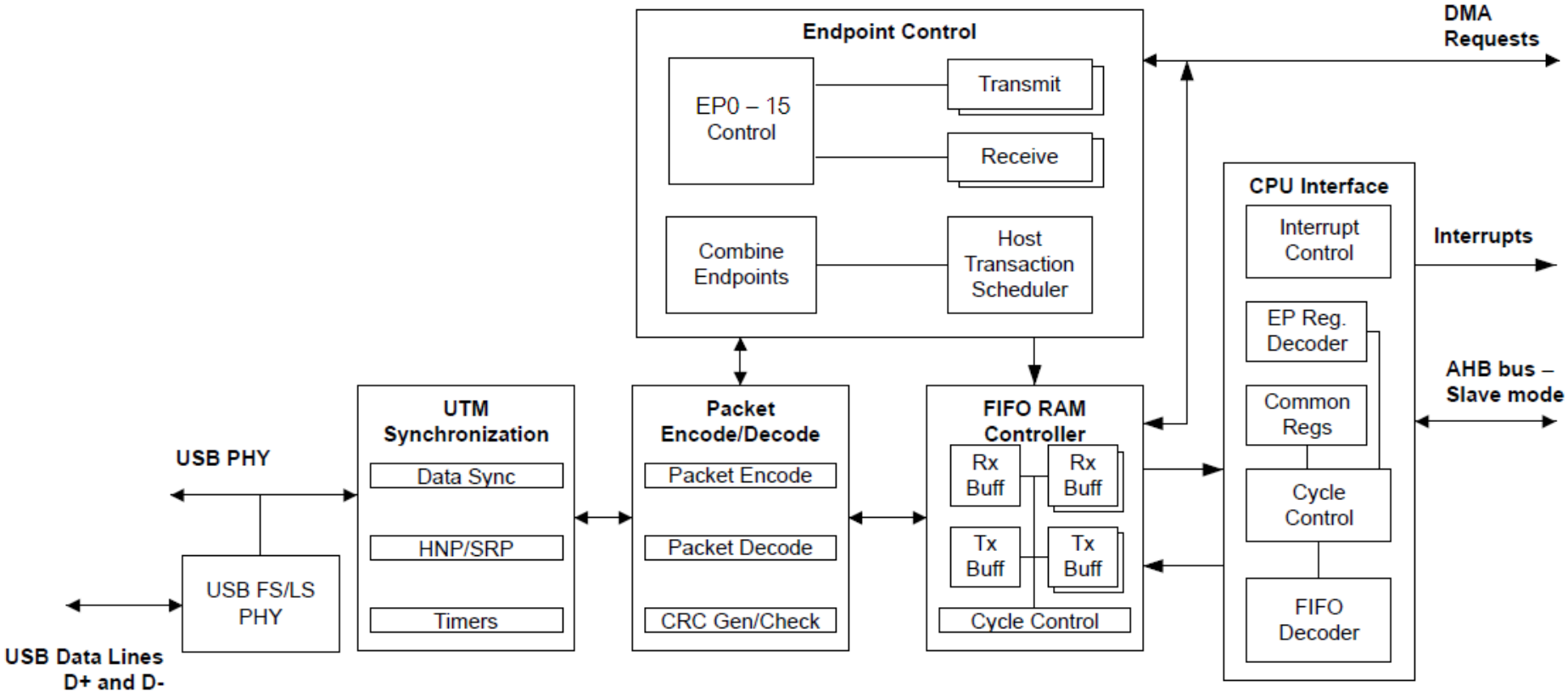
Stellaris® USB 硬件

Stellaris LM4F 器件上的 USB 控制器

- 支持 USB 主机、设备和 OTG (On-The-Go) 模式
- 集成控制器与 PHY
- 符合 USB 2.0 标准的全速规范 (12.0 Mbps)
- 16 个端点
  - 1 个专用的控制 IN 端点，1 个专用的控制 OUT 端点
  - 7 个可配置的 IN 端点，7 个可配置的 OUT 端点
- 可支持 µDMA 的 2KB 专用 USB 端点存储器



USB 模块方框图



外部 USB 信号

- 所需的外部信号
  - USB D+
  - USB D-
  - VBUS
  - USBID（Micro-A、Micro-B、Micro-A/B 和 Mini-B 连接器）
- 其他的外部信号
  - USBPFLT：电源故障引脚，连接电源管理芯片管脚来接收电源错误指示
  - USBEPEN：使能外部电源，在主机模式下用来控制为USB总线供电的外部电源

所支持的 USB 模式

- 主机模式
  - 能与 USB 闪存驱动器和其他 USB 设备实现互动
  - 可利用闪存驱动器替代 PC 完成固件升级
  - 可读写闪存设备文件，以实现数据传输与存储
  - 可用鼠标或键盘作为 Stellaris MCU 的输入端
- 设备模式
  - 可实现与 PC 的简便连接
  - 通过 USB总线给 MCU 供电
  - 采用 PC 端的应用程序通过 USB 完成设备的固件升级
- OTG (On-The-Go) 模式
  - 可简化主机与设备作用之间的动态切换
  - 作为主机执行一组功能，并作为设备执行一组不同的功能
  - 一个 USB 端口可以兼有主机和设备功能

Stellaris USB 软件

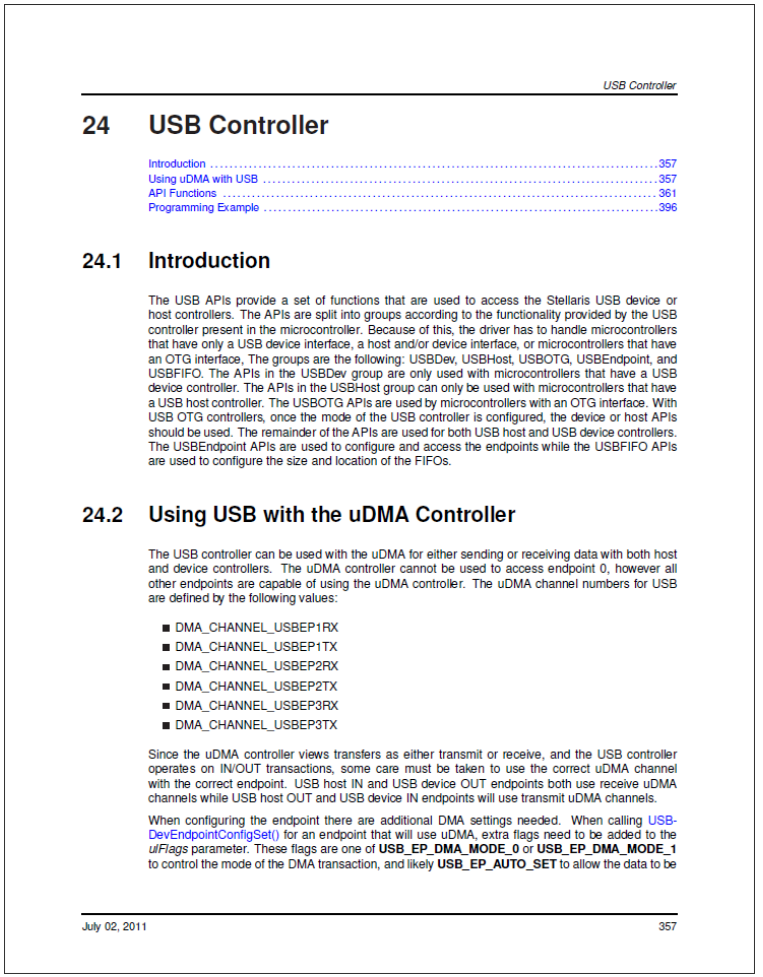
介绍 StellarisWare®

- StellarisWare 软件包括源代码和免版税的程序库
- 可使您的所有程序设计全部采用 C/C++ 语言完成，包括 ISR 与启动代码
- 主要的功能为：
  - Stellaris 外设驱动程序库
  - Stellaris 图形库
  - Stellaris USB 库
  - Stellaris IEC 60730 库

- 包括参考应用软件
- 支持ISP功能
- StellarisWare® 软件得到了所有最常用工具供应商的支持
- 稳健：StellarisWare API 被预先烧写到大多数 Stellaris MCU 上的 ROM 中

外设驱动程序库 (DriverLib)

- API 接口用于外设的配置，包括 USB 控制器的配置
- *免使用许可费且免版税*
- 简化并加快了应用的开发
- 可用于应用开发或作为编程范例
- 提供目标库文件以及源代码
- 采用 Code Composer Studio、ARM/Keil、IAR、Code Red 和 Code Bench 工具进行编译
- **DriverLib 被预先烧写到大多数 Stellaris MCU 上的 ROM 中**



USB Controller

24 USB Controller

24.1 Introduction

The USB APIs provide a set of functions that are used to access the Stellaris USB device or host controllers. The APIs are split into groups according to the functionality provided by the USB controller present in the microcontroller. Because of this, the driver has to handle microcontrollers that have only a USB device interface, a host and/or device interface, or microcontrollers that have an OTG interface, The groups are the following: USBDev, USBHost, USBOTG, USBEndpoint, and USBFIFO. The APIs in the USBDev group are only used with microcontrollers that have a USB device controller. The APIs in the USBHost group can only be used with microcontrollers that have a USB host controller. The USBOTG APIs are used by microcontrollers with an OTG interface. With USB OTG controllers, once the mode of the USB controller is configured, the device or host APIs should be used. The remainder of the APIs are used for both USB host and USB device controllers. The USBEndpoint APIs are used to configure and access the endpoints while the USBFIFO APIs are used to configure the size and location of the FIFOs.

24.2 Using USB with the uDMA Controller

The USB controller can be used with the uDMA for either sending or receiving data with both host and device controllers. The uDMA controller cannot be used to access endpoint 0, however all other endpoints are capable of using the uDMA controller. The uDMA channel numbers for USB are defined by the following values:

- DMA_CHANNEL_USBEP1RX
- DMA_CHANNEL_USBEP1TX
- DMA_CHANNEL_USBEP2RX
- DMA_CHANNEL_USBEP2TX
- DMA_CHANNEL_USBEP3RX
- DMA_CHANNEL_USBEP3TX

Since the uDMA controller views transfers as either transmit or receive, and the USB controller operates on IN/OUT transactions, some care must be taken to use the correct uDMA channel with the correct endpoint. USB host IN and USB device OUT endpoints both use receive uDMA channels while USB host OUT and USB device IN endpoints will use transmit uDMA channels.

When configuring the endpoint there are additional DMA settings needed. When calling USBDevEndpointConfigSet() for an endpoint that will use uDMA, extra flags need to be added to the *ulFlags* parameter. These flags are one of **USB_EP_DMA_MODE_0** or **USB_EP_DMA_MODE_1** to control the mode of the DMA transaction, and likely **USB_EP_AUTO_SET** to allow the data to be

July 02, 2011 357

StellarisWare USB 库 (USBLib)

- 基于 DriverLib API 并为 USB上层协议提供支持
  - 增添了用于通用主机和设备功能的框架
  - 包括常见 USB 类的实现
- 分层结构意味着开发人员可以根据自己的需求调用不同层的函数。
- 在某些例程中包含了驱动程序与“.inf”文件，这将有助于开发人员为新设备定制驱动程序和“.inf”文件
- 在使用 stellaris 器件时，USBLib **免版税**
- 为支持的 USB 类提供了以下应用范例：
  - 主机类范例
    - HID 键盘
    - HID 鼠标
    - 海量存储器主机
  - 设备类范例
    - HID 键盘
    - HID 鼠标
    - CDC 串行
    - 通用大容量传输
    - 设备固件升级
    - 海量存储设备
  - OTG 类范例
    - 会话请求协议
    - 主机协商协议

人机接口设备类 (HID)

- USB 人机接口设备类
  - 不需要开发PC驱动而完成 USB 设备的开发
    （驱动由Windows 操作系统支持）
  - 可提供多种预定义的通信类型
  - 具有高度灵活性的类：不单单是鼠标与键盘
- 采用 StellarisWare 的优势
  - HID 专用范例可帮助开发
  - 专为处理 HID 功能而设计的USB API 函数
- 应用范例：
  - USB HID 键盘
  - USB HID 鼠标

设备固件升级类 (DFU)

- USB 设备固件升级类
  - 允许设备通过与USB主机的连接来完成其固件的升级
  - 允许设备在其设备描述符中通告其升级能力
- 优势：
  - 在组合式设备中，DFU 主要作为一个辅助设备类，可为另一个设备类增添升级功能
  - 提供一种更加通用的 USB 启动加载程序
  - StellarisWare 包含一个 **Windows PC 应用程序**，以充当应用范例的主机
- 范例：
  - USB 启动加载程序范例

通信设备类 (CDC)

- USB 通信设备类
  - 提供简单的串行通信功能
  - 范例可充当一个起点，借此逐步实现更加复杂的串行通信及文件传输
  - 应用范例：USB CDC Serial
- 大容量 USB 设备
  - 可提供批量传输功能
  - 范例程序是一种简单的数据回送应用程序，可作为更复杂数据传输应用程序的基础
  - 提供了 Windows PC 应用程序，以与范例应用程序进行通信
  - 应用范例：大容量 USB 设备

人机接口设备类主机驱动器

- USB HID 鼠标和键盘主机
  - Stellaris 器件可以对 HID 鼠标和键盘进行枚举和通信
- 优势
  - 采用标准的人机接口设备与 MCU 进行通信
  - 范例应用可作为一个起点，用于支持其他的 HID 设备
- 范例：
  - USB 主机键盘
  - USB 主机鼠标

海量存储设备类主机驱动器

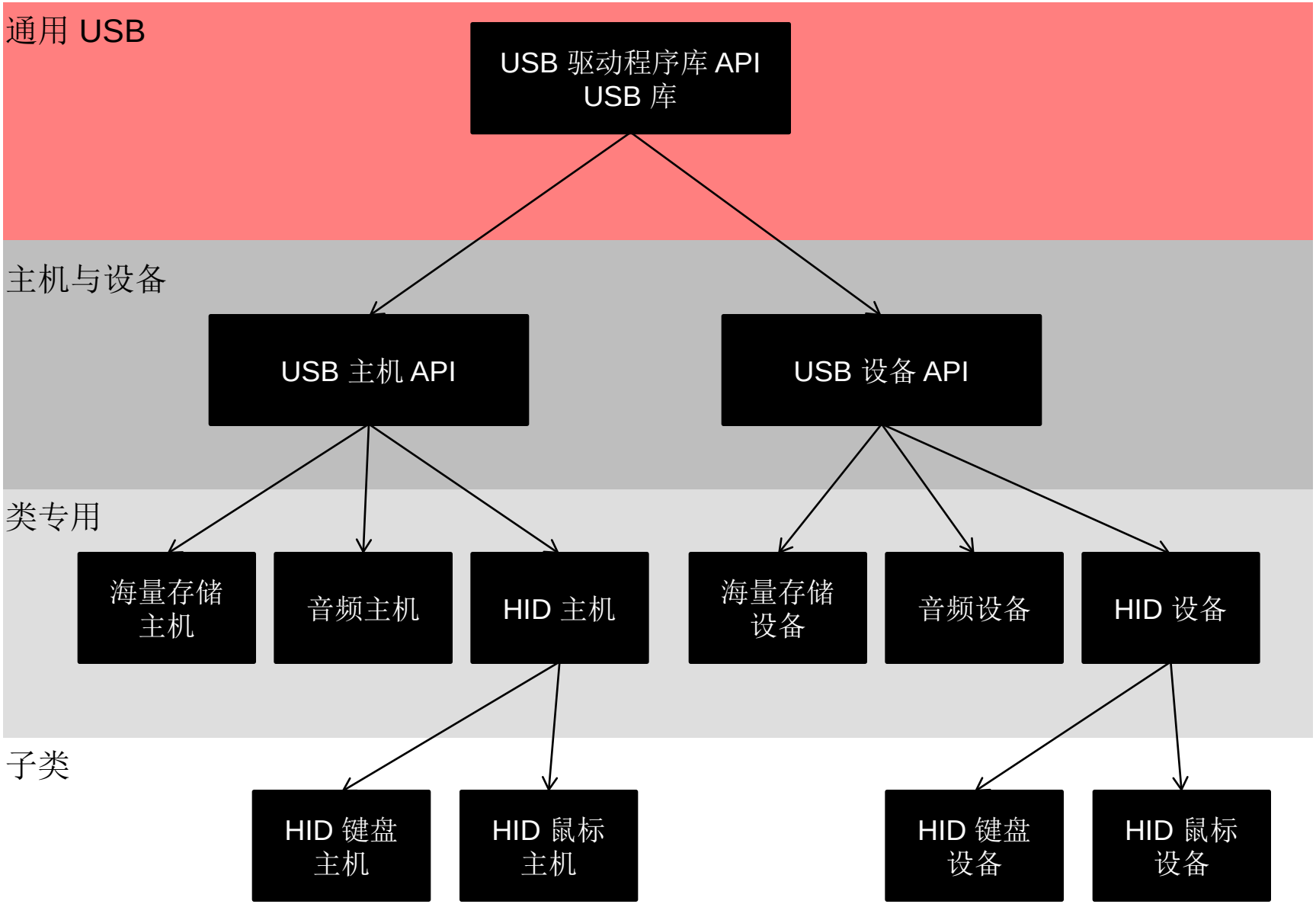
- USB 海量存储主机
  - 允许 Stellaris 器件对 USB 海量存储媒体（如：闪存驱动器）进行文件的读和写
  - 文件系统由开源程序 FatFs 提供
- 优势：
  - 可采用一个闪存驱动器实现与 MCU 之间的数据文件往来传输
  - 可通过 U 盘（而不是采用一个 PC 应用程序）完成固件升级
- 范例：
  - USB 海量存储类主机
  - USB 盘升级

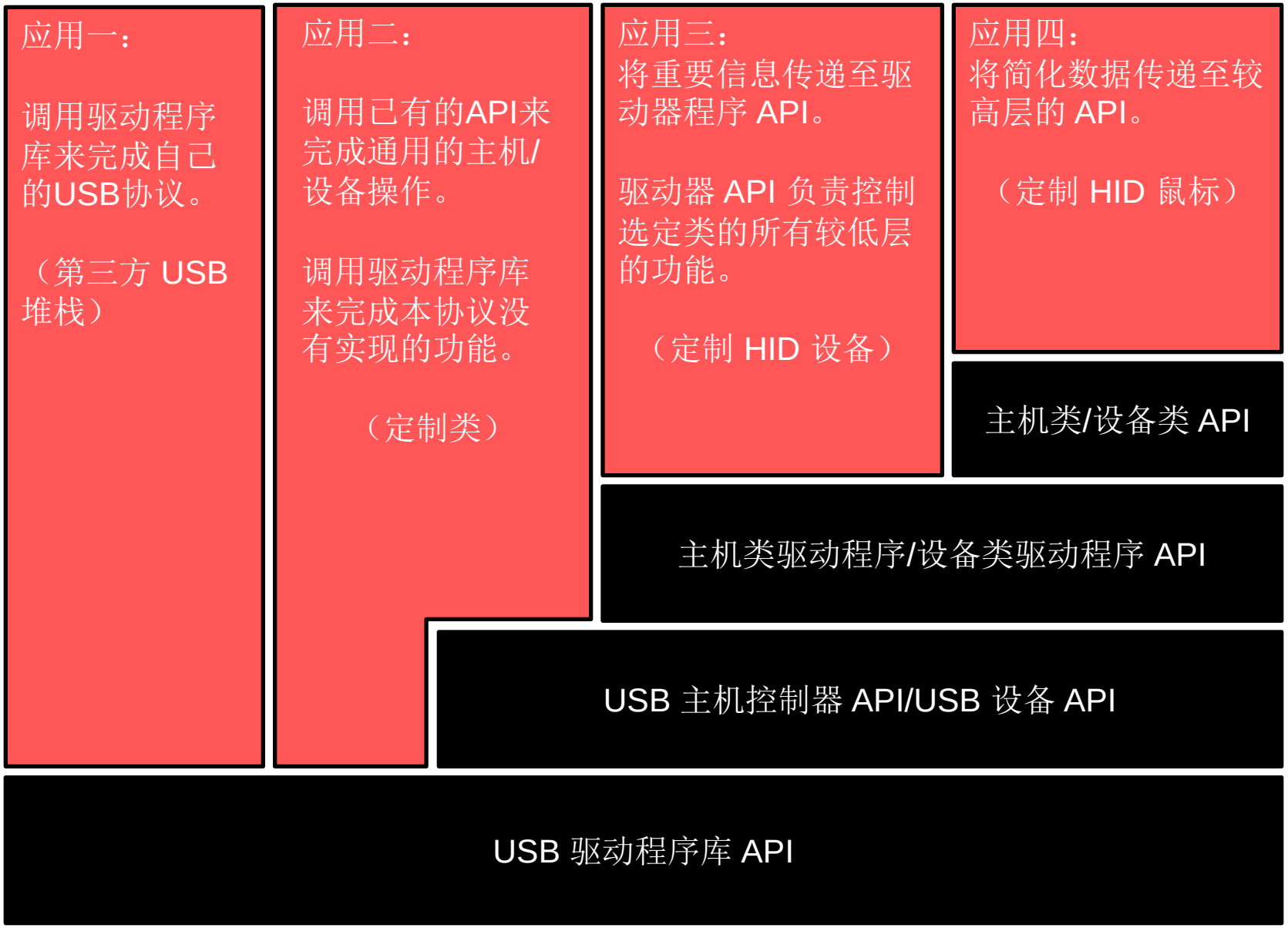
USB 海量存储文件系统：FatFs

- 海量存储主机范例中的文件操作由 FatFs 控制
  - 为嵌入式平台编写的第三方开源文件系统
  - 作为一种附加的代码资源与 StellarisWare 一道提供。
- 特性：
  - 支持 8.3 文件名格式（以及近期版本中的长文件名）
  - 支持多卷
  - 小代码量
- 欲知更多有关 FatFs 的信息，敬请访问下面的网址：
  - http://elm-chan.org/fsw/ff/00index_e.html

StellarisWare USB 库的结构

- 通用 – 为主机与设备应用所使用，横跨所有的类（描述符解析、设定操作模式等）
- 设备专用 – 所有设备都需要的功能（用于响应主机请求、主机信令等的功能）
- 主机专用 – 所有主机应用都需要的功能（与枚举相关的功能、端点管理等）
- 类专用 – 用于帮助开发符合常用 USB 类标准之设备的功能（人机接口设备功能、设备固件升级等）


通用 USB
USB 驱动程序库 API
USB 库
主机与设备
USB 主机 API
USB 设备 API
类专用
海量存储
主机
音频主机
HID 主机
海量存储
设备
音频设备
HID 设备
子类
HID 键盘
主机
HID 鼠标
主机
HID 键盘
设备
HID 鼠标
设备



结论

- Stellaris LM4F 器件将继续提供可靠的 USB 功能
- 针对 USB 主机、设备和 OTG 模式的硬件支持仍然是一个卖点
- 通过软件持续不断地提供 USB 支持
  - 外设驱动程序库
  - USB 库
  - 更多的设备类的软件范例