

## 第 17 章

# I<sup>2</sup>C、SPI、USB 驱动架构类比

### 本章导读

本章类比 I<sup>2</sup>C、SPI、USB 的结构，从而进一步帮助读者理解本书第 2 章的内容，也进一步实证 Linux 驱动万变不离其宗的道理。

## 17.1 I<sup>2</sup>C、SPI、USB 驱动架构

根据图 12.4，Linux 倾向于将主机端的驱动与外设端的驱动分离，而通过一个核心层将某种总线的协议进行抽象，外设端的驱动调用核心层 API 间接过渡到对主机驱动传输函数的调用。对于 I<sup>2</sup>C、SPI 这类不具备热插拔能力的总线而言，一般在 arch/arm/mach-xxx 或者 arch/arm/boot/dts 中会有相应的板级描述信息，描述外设与主机的连接情况。

Linux 的各个子系统都呈现为相同的特点，表 17.1 类比了 I<sup>2</sup>C、SPI、USB 驱动架构，其他的 PCI 等都是类似的。

表 17.1 I<sup>2</sup>C、SPI、USB 驱动架构的类比

项目		I <sup>2</sup> C	SPI	USB
主 机 侧	描述主机的数据结构	i2c_adapter	spi_master	usb_hcd
	主机驱动传输成员函数	master_xfer()	transfer()	urb_enqueue()
	主机的枚举方法	由 I <sup>2</sup> C 控制器挂接的总线决定（一般是 platform）	由 SPI 控制器挂接的总线决定（一般是 platform）	由 USB 控制器挂接的总线决定（一般是 platform）
核 心 层	描述传输协议的数据结构	i2c_msg	spi_message	URB
	传输 API	i2c_transfer()	spi_sync() spi_async()	usb_submit_urb()
外 设 端	外设的枚举方法	i2c_driver	spi_driver	usb_driver
	描述外设的数据结构	i2c_client	spi_device	usb_device
板 级 逻 辑	非设备树模式	i2c_board_info	spi_board_info	总线具备热插拔能力
	设备树模式	在 I <sup>2</sup> C 控制器节点下添加子节点	在 SPI 控制器节点下添加子节点	总线具备热插拔能力

对于 USB、PCI 等总线而言，由于它们具备热插拔能力，所以实际上不存在类似 I<sup>2</sup>C、SPI 这样的板级描述信息。换句话说，即便是有这类信息，其实也没有什么用，因为如果写

了板子上有个 U 盘，但实际上没有，其实反而是制造了麻烦；相反，如果没有写，U 盘一旦插入，Linux USB 子系统会自动探测到一个 U 盘。

同时我们注意到，I<sup>2</sup>C、SPI、USB 控制器虽然给别人提供了总线，但是其实自己也是由它自身依附的总线枚举出来的。比如，对于 SoC 而言，这些控制器一般是直接集成在芯片内部，通过内存访问指令来访问的，因此它们自身是通过 platform\_driver、platform\_device 这种模型枚举进来的。

## 17.2 I<sup>2</sup>C 主机和外设眼里的 Linux 世界

I<sup>2</sup>C 控制器所在驱动的 platform\_driver 与 arch/arm/mach-xxx 中的 platform\_device（或者设备树中的节点）通过 platform 总线的 match() 函数匹配导致 platform\_driver.probe() 执行，从而完成 I<sup>2</sup>C 控制器的注册；而 I<sup>2</sup>C 上面挂的触摸屏依附的 i2c\_driver 与 arch/arm/mach-xxx 中的 i2c\_board\_info 指向的设备（或者设备树中的节点）通过 I<sup>2</sup>C 总线的 match() 函数匹配导致 i2c\_driver.probe() 执行，从而使触摸屏展开。

图 17.1 虚线上方部分是 i2c\_adapter 眼里的 Linux 世界；下方部分是 i2c\_client 眼里的 Linux 世界。其实，Linux 中的每一个设备通过它依附的总线被枚举出来，尽管它自身可能给别人提供总线。

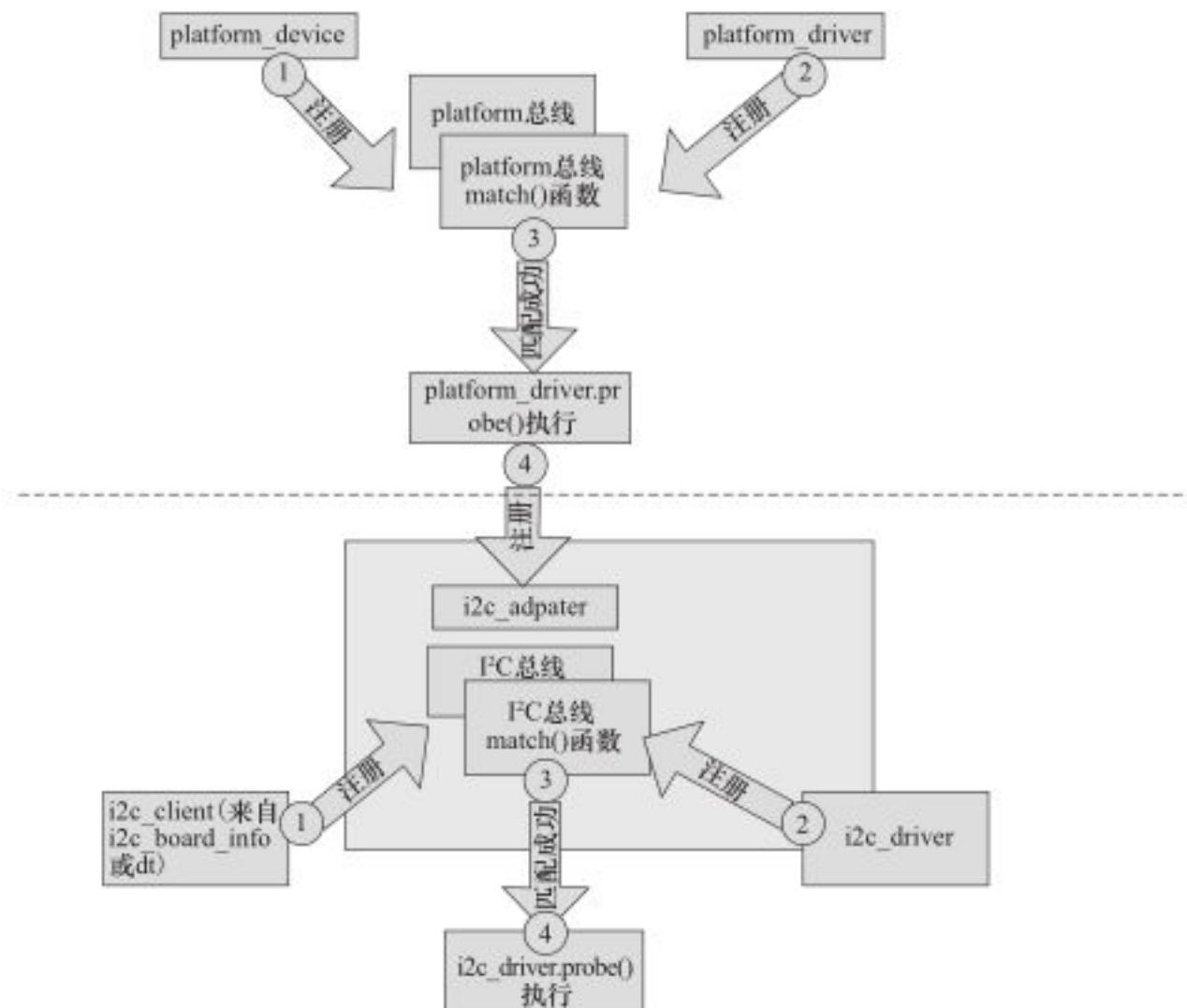


图 17.1 I<sup>2</sup>C 主机和外设眼里的 Linux 世界