致远电子

传递价值 成就你我

Anywhere—面向设备的编程模式

广州致远电子有限公司
周立功

嵌入式系统联谊会
htttp://www.esbf.org.cn

http://www.embedcontrol.com

传递价值 成就你我

面向寄存器的开发流程

研发要求 开发步骤

C语言

面向寄存器的软件开发

```
int main (void)
{
  PINSEL0 = PINSEL0 & (~0x0F);
  PINSEL0 = PINSEL0 | 0x05;          // 设置I/O连接到UART
  UARTInit ();                       // 串口初始化
  U0FCR = 0x81;                      // 使能FIFO，设置8字节触发点
  U0IER  = 0x01;                     //使能接收中断
  IRQEnable ();
  ……
}
```

Registers

LPC24xx

致远电子

传递价值 成就你我

嵌入式系统传统编程模式

- 嵌入式系统与通用计算机系统同源，由于应用领域和研发人员的不同，嵌入式系统很早就走向相对独立的发展道路，其编程模式与通用计算机系统有较大的区别。
- 一般来说，嵌入式系统传统编程模式有以下三种:
  - 面向寄存器的编程模式；
  - 面向API的编程模式；
  - 面向端口的的编程。

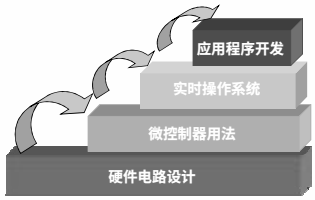
其中面向寄存器的编程模式仍然占主导地位。

致远电子

传递价值 成就你我

面向寄存器的编程模式分析

- 模式分析：面向寄存器的编程就是自己即作将军又做士兵，眉毛胡子一把抓。

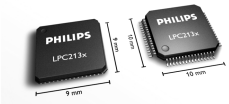


致远电子

传递价值 成就你我

面向寄存器的编程模式

- 面向寄存器的编程是直接操作硬件提供的编程接口来编写嵌入式软件的编程方式；
- “本地硬件”提供的编程接口大多数为寄存器，映射到软件能够直接访问的是I/O空间或Memory空间；
- “远程硬件”一般需要通过本地的通讯硬件接口来访问，而本地的通讯硬件接口也是在寄存器模式下工作的。



致远电子

传递价值 成就你我

面向API的编程模式

- 面向寄存器的模式无论是学习还是编程都非常麻烦且效率低下，所以开发板好卖。为了方便嵌入式软件的编写，一些公司编写了软件把底层硬件屏蔽起来，用户即通过调用API函数来访问硬件。这种通过第三方软件提供的接口来访问硬件的编程方式就是面向API的编程模式。
- 面向API模式的编程要求研发人员对API手册非常地熟悉才能够完成开发，对硬件能力要求比较低。

致远电子

面向API的开发流程

面向API的软件开发

```
int main (void)
{
   uint32 uiNum;
   uartInit(UART0,cUartArg,NULL);                  //  初始化串口
   irqVicSet(UART0_VIC,5,(uint32)uart0Isr);        //  设置UART0中断
   IRQEnable( );
   /*  提示UART0连接成功 */
   uartWrite(UART0, uiConnectOk, sizeof(uiConnectOk), NULL);
   ......
}
```

硬件平台

传统开发模式分析

嵌入式软件设计

Linux

μC/ The Real-Time Kernel

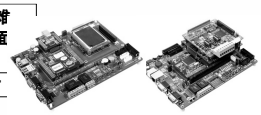
Windows CE.net

系列微控制器

ARM7	ARM9	ARM10E	ARM11	Cortex	Xscale

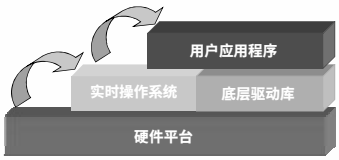
传统开发模式研发周期长，研发难度大，要求技术人员对软硬件全面精通，能力要求非常高

硬件可靠性设计



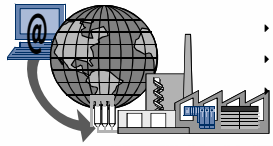
面向API的开发分析

- 面向API编程的开发模式，每换一种系统，研发人员就需要去熟悉新的API手册；不同系统的API的功能和性能差异极大，对研发人员的能力有一些要求，但比面向寄存器的编程模式有了很大的进步；
- 一句话形容：面向API的编程模式就是手把手地教别人干活。



嵌入式应用技术传统开发方式的困境

- 嵌入式系统由独立工作走向了网络互联控制，典型的就是集散控制系统。要将嵌入式系统组成网络，需要为所有控制器增加通讯接口硬件，设计兼容的通讯协议。并要对每个系统的通讯接口硬件、通讯协议进行编程后才能组成网络，这些设计无疑是很复杂的。



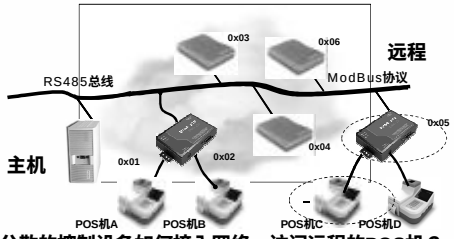
- 面向寄存器的网络设备开发；
- 面向API的网络设备开发；
- 面向端口的网络设备开发。

面向端口的编程模式

- 面向端口的编程模式是PLC(可编程逻辑控制器)的编程模式。我们知道PLC是将所有的硬件都虚拟成端口，其实质就是通过对端口的读写来完成对硬件的控制；
- PLC最初是为了替代继电器电气控制线路而提出的一种编程思想，无论是对于复杂的控制功能还是支持复杂程序的能力都很弱，而且对于远程硬件的支持也比较弱，即便PLC有一些远程硬件的支持能力，但那也主要是支持PLC厂商自己的配件；
- 关于PLC的编程不做详细讨论。

plc.com

面向寄存器的网络设备开发



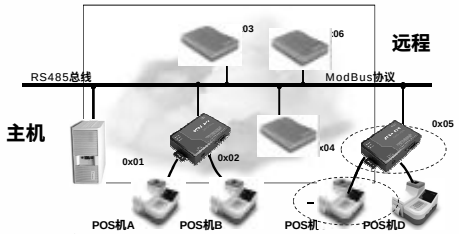
分散的控制设备如何接入网络，访问远程的POS机？

寄存器模式开发难点：用户需要对所有控制器增加硬件通讯接口，设计兼容的通讯协议；并要对通讯接口硬件、通讯协议编程后才能组成网络。要求用户精通接入网络的硬件通讯*接口和通讯协议，开发难度大！*

1、确定通信方式及协议：增加通讯接口并设计兼容的协议；

2、编写控制软件，对通讯接口、通讯协议编程。实现主机对远程设备的查找访问和

面向API的网络设备开发



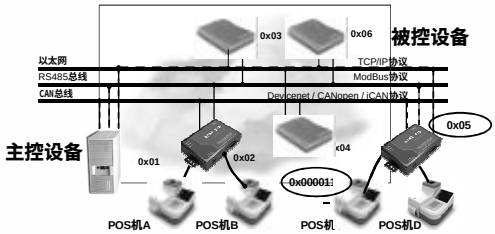
面向API的网络设备开发，本地主机如何访问远程的POS机？

API模式开发特点：硬件平台已经搭建好，用户只需选择控制器支持的硬件通讯接口和通讯协议；调用API函数对通讯接口和通讯协议编程后组成组成网络。要求用户大体了解接入网络的通信接口和通讯协议！

1、选择通信方式及协议：选择共同支持的RS485总线和ModBus协议；

2、根据API手册对通讯接口和通讯协议进行编程。实现主机对远程设备的直接访问和控制。

面向设备的网络设备开发



面向设备模式开发特点：研发人员不需要了解设备如何连接到系统，不需要了解硬件接口和通讯协议，只需要知道设备地址和设备内地址分配，就可以通过有限的几个函数操作设备。

1、 选择多种通信方式及协议：如选择双方都支持的RS485、CAN总线和以太网，ModBus、Devicenet和TCP/IP等协议。

2、AnyWhere编程，只需要知道设备地址0x05和设备内地址分配，用有限的几个函数编写控制程序。

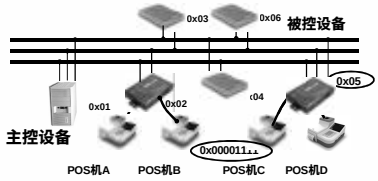
传统网络设备开发模式分析

- 目前，组网的范围更加广泛，不但需要本地组网，还需要远程组网；不但需要控制设备之间互连，还需要控制设备与普通计算机之间互连，以及不同厂商的设备之间的互联等，这些要求都极大的加剧了系统编程的复杂性。

研发方式	基于寄存器	基于API	基于端口
研发周期	很长	长	较短
代码量	巨大	大	小
维护工作	很复杂	复杂	简单
移植难度	难度大	比较方便	仅限于PLC
研发难度	很高	高	低
功能和灵活性	高	较高	低
对研发人员要求	很高	高	低

面向设备编程基本概念

- 设备
- 主控设备
- 被控设备
- 混合设备
- 设备地址
- 端口
- 端口地址
- 端口组



端口是虚拟内存，对某一个特定端口的读写可实现设备的特定功能。只有被控设备具有端口。

Anywhere—面向设备的编程

- 面向设备的编程模式是传统编程方式集合发展而来的。继承三者的优点，避免了各自的缺点。将所有接入网络的嵌入式系统和计算机作为一个整体考虑，研发人员只需知道设备地址和设备内地址分配，不需要知道设备如何连接到系统，就可以通过有限的几个函数操作设备。
- “无论您在哪里，我都用同样的方式操作您的设备，不管您如何不停的改变位置”
- “不需要知道您在哪里，我都可以访问您”

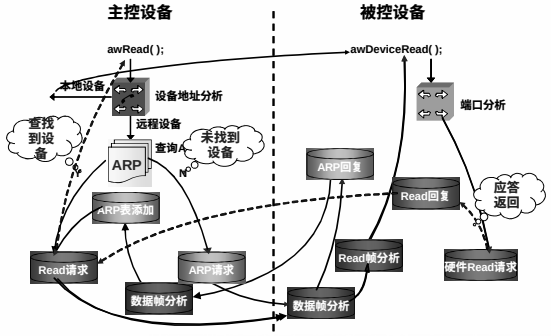
面向设备编程的特点

- 使用ANSI C编程
- 编程接口统一
- 编程不区分远程设备和本地硬件
- 多协议多网络支持
- 协议及链路自动动态匹配
- 提供被控设备编程接口

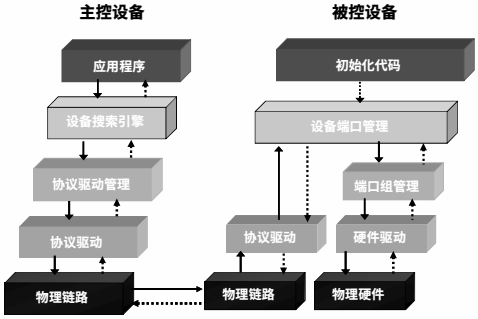
AnyWhere的总体设计思想

- Anywhere将所有接入网络的嵌入式系统作为一个整体来考虑，依据其在网络中的作用，把嵌入式系统分为主控设备和被控设备两大类；
- 主控设备是系统的核心，用于系统的逻辑控制和人机接口，并不直接控制被控对象，实现“协议及链路的自动动态匹配”；
- 被控设备是直接控制对象的，数目众多。在Anywhere中被控设备以“被动应答”方式工作，没有“协议及链路自动动态匹配”过程，也不会主动联系主控设备。

Anywhere函数处理流程



Anywhere的基本框图



主机接口函数一

端口读函数——awRead

INT32S awRead (INT32U ulDevice, INT32U uiAddr, INT32U *pulData)

功能描述：端口读
输入参数：ulDevice 设备地址
ulAddr 端口地址
输出参数：pulData 读到的数据
返回值： AW_OK 成功
负数 错误，绝对值参考anywhere.h

Anywhere设备调用关系

- 主控设备通过远程调用来控制被控设备。当主控设备调用Anywhere的主机接口核心函数时，被控设备的对应函数开始执行，执行完毕后，把返回值和执行结果反馈给主控设备，主控设备获得执行结果，函数返回。

主控设备与被控设备函数关系表

功能	主控设备函数	被控设备函数
端口读	awRead	awDeviceRead
端口写	awWrite	awDeviceWrite
扩展端口读	awReadEx	awDeviceReadEx
扩展端口写	awWriteEx	awDeviceWriteEx

主机接口函数二

面向设备写函数——awWrite

INT32S awWrite (INT32U ulDevice, INT32U uiAddr, INT32U ulData)

功能描述：端口写
输入参数：ulDevice 设备地址
ulAddr 端口地址
ulData 数据
输出参数：无
返回值： AW_OK 成功
负数 错误，绝对值参考anywhere.h

主机接口函数三

扩展端口读函数——awReadEx

```
INT32S awReadEx (INT32U ulDevice, INT32U uiAddr,
                 INT16U usLen, void *pvData, INT8U ucMod, )
```

输入参数：	ulDevice	设备地址
	ulAddr	端口地址
	usLen	数据数目
	ucMod	读写模式与端口位宽的或
输出参数：	pvData	读到的数据
返回值：	>=0	完成的数据数目
	负数	错误，绝对值参考aw_device.h
特殊说明：	ucMod	必须与端口的实际属性一致

主机接口函数四

扩展端口写函数——awWriteEx

```
INT32S awWriteEx (INT32U ulDevice, INT32U uiAddr,
                  INT16U usLen, void *pvData, INT8U ucMod, )
```

输入参数：	ulDevice	设备地址
	ulAddr	端口地址
	usLen	数据数目
	pvData	数据
	ucMod	读写模式与端口位宽的或
输出参数：	无	
返回值：	>=0	完成的数据数目
	负数	错误，绝对值参考aw_device.h
特殊说明：	ucMod	必须与端口的实际属性一致

谢 谢 ！