面向设备的编程模式概述

嵌入式系统传统编程模式

编程就是让计算机为解决某个问题而使用某种程序设计语言编写程序代码，并最终得到结果的过程。为了使计算机能够理解人的意图，人类就必须要将需解决的问题的思路、方法、和手段通过计算机能够理解的形式告诉计算机，使得计算机能够根据人的指令一步一步去工作，完成某种特定的任务。这种人和计算机之间交流的过程就是编程。从计算机发明至今，随着计算机硬件和软件技术的发展，计算机的编程语言经历了机器语言、汇编语言、面向过程的程序设计语言以及面向对象的程序设计语言阶段。

嵌入式系统与通用计算机系统同源，可是因为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面向寄存器的编程模式、面向 API 的编程模式、面向端口的编程模式等，其中面向寄存器的编程模式仍然占主导地位。

1.1 面向寄存器的编程模式

嵌入式系统是"控制、监视或者辅助装置、机器和设备运行的装置"（devicesusedtocontrol,monitor,orassisttheoperationofequipment,machineryorplants）。从中可以看出嵌入式系统是软件和硬件的综合体，还可以涵盖机械等附属装置。目前国内一个普遍被认同的定义是：以应用为中心、以计算机技术为基础、软件硬件可裁剪、适应应用系统对功能、可靠性、成本、体积、功耗严格要求的专用计算机系统。

所谓"面向寄存器的编程",就是软件直接操作硬件提供的编程接口来编写嵌入式软件的编程模式。目前，本地硬件提供的编程接口大多数为寄存器，它们通常映射到软件能够直接访问的 I/O 空间或存储器空间。

面向寄存器的编程模式的基本步骤如图 1 所示，这是一个蜂鸣器鸣叫的程序。由此可以看出，面向寄存器的编程模式需要对硬件细节非常了解，这是非常繁琐和容易出错的，并且对开发人员的要求较高。

一句话形容：面向寄存器的编程模式就是自己既作将军又作士兵，眉毛胡子一把抓。

1.2 面向 API 的编程模式

API（ApplicationProgrammingInterface,应用程序编程接口）是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件的以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节。API 除了有应用"

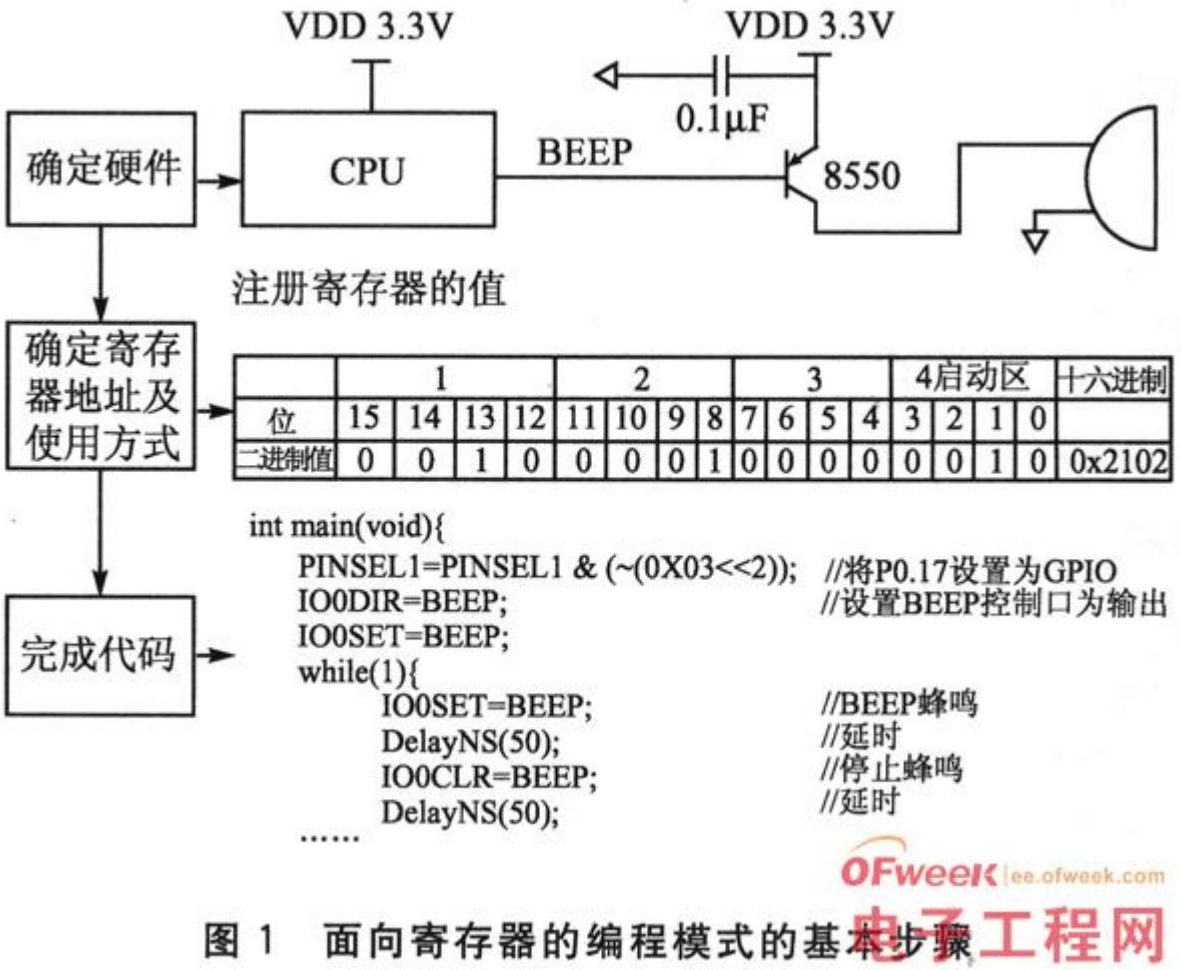
Ofweek电子工程网

应用程序接口"的意思外，还特指 API 的说明文档，也称为帮助文档。另外，也是美国石油协会、空气污染指数、医药、空中位置指示器的英文简称。

面向寄存器的编程模式非常麻烦，效率低下，不是人人都能胜任的。为了方便嵌入式软件的编写，有些公司编写软件把硬件屏蔽起来形成 API,应用软件则通过这些 API 接口访问硬件。这种通过第三方软件提供的接口来访问硬件的编程模式就是面向 API 的编程模式。

即使相同的硬件，不同公司提供的 API 也有很大的出入。有些仅仅提供了一些程序库，对硬件进行简单封装。而有的则提供标准的操作系统接口，如 WinCE、嵌入式 Linux 和 VxWorks 等。

面向 API 编程模式的基本步骤如图 2 所示。可以看出，面向 API 的编程模式只需要对硬件细节有大概的了解即可，但需要对 API 手册进行详细阅读才能开发。不同系统的 API 可能完全不同，换一种系统，开发人员就需要重新熟悉新的 API. 另外，不同系统的 API 功能和性能差异极大，对开发人员的要求也有较大的差别。一句话形容：面向 API 的编程模式就是手把手教别人干活。



	1				2				3				4启动区				十六进制
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
二进制值	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0x2102

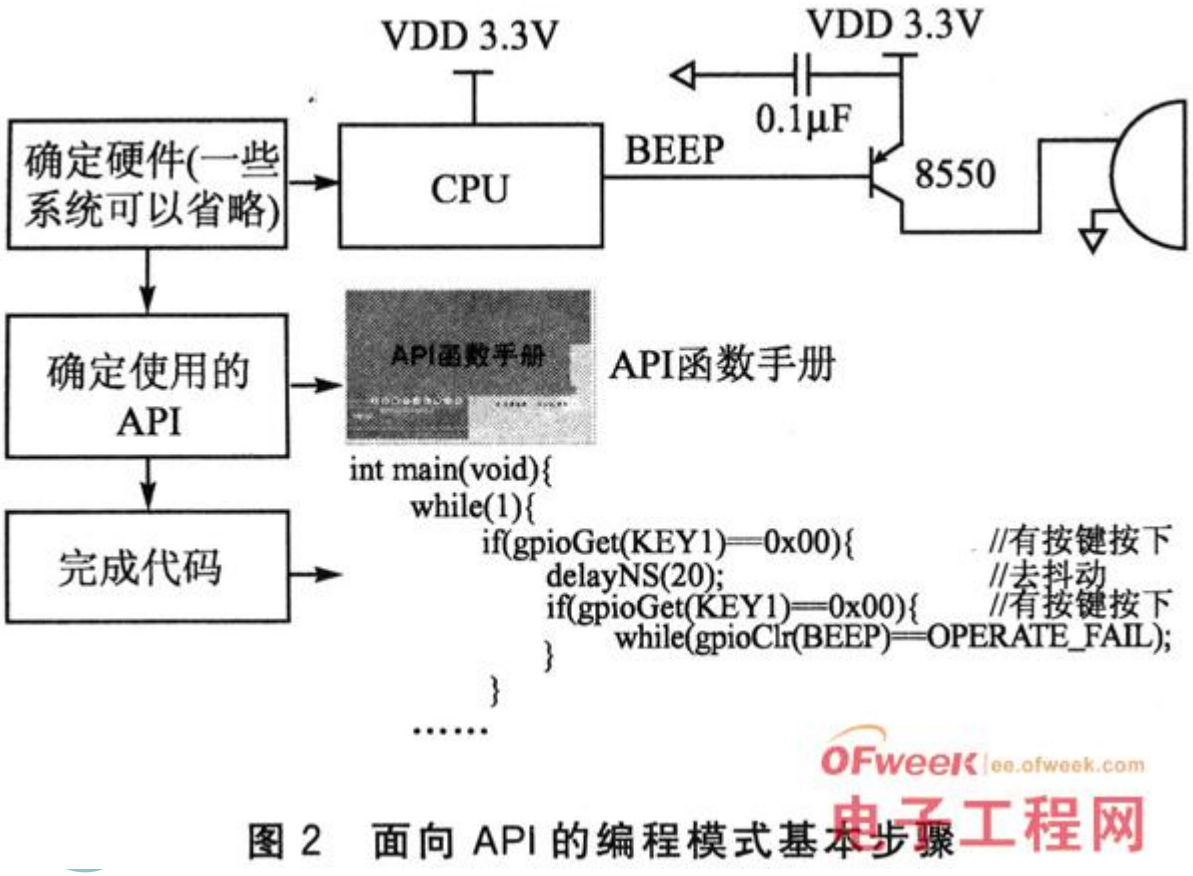
OFweek ee.ofweek.com 电子工程网

图 1　面向寄存器的编程模式的基本步骤

1.3 面向端口的编程

可编程控制器简称 PC（英文全称：ProgrammableController），它经历了可编程序矩阵控制器 PMC、可编程序顺序控制器 PSC、可编程序逻辑控制器 PLC（英文全称：ProgrammableLogicController）和可编程序控制器 PC 几个不同时期。PLC 是一种专门为在工业环境下应用而设计的数字运算操作的电子装置。它采用可以编制程序的存储器，用来在其内部存储执行逻辑运算、顺序运算、计时、计数和算术运算等操作的指令，并能通过数字式或模拟式的输入和输出，控制各种类型的机械或生产过程。PLC 及其有关的外围设备都应该按易于与工业控制系统形成一个整体，易于扩展其功能的原则而设计。

面向端口编程是 PLC 的编程模式。PLC 把所有硬件都虚拟成端口，通过对端口的读写完成对硬件的控制。PLC 最初是为了替代继电器编程，对复杂程序的支持比较弱，对远程硬件的支持也比较弱（主要支持 PLC 厂商自己的配件）。



OFweek ee.ofweek.com 电子工程网

图 2　面向 API 的编程模式基本步骤

2 嵌入式系统传统编程方法的困境

2.1 对 比

各种传统的嵌入式系统编程模式有各自特点，如表 1 所列。

表 1　传统编程模式对照表

研发模式	研发周期	代码量	维护工作	移　植	研发难度	功能与灵活性	对研发人员的要求
基于寄存器	长	大	复杂	难度大	高	高	高
基于 API	较长	较小	比较简单	比较方便	较高	较高	较高
基于端口	较短	较小	简单	仅限于 PLC	低	低	低

OFweek 电子工程网

2.2 困 境

最初，嵌入式系统都是独立工作的。传统的编程模式都是面向独立的微控制器（微处理器），操作的硬件都是本地硬件。

微控制器是将微型计算机的主要部分集成在一个芯片上的单芯片微型计算机。微控制器诞生于 20 世纪 70 年代中期，经过 20 多年的发展，其成本越来越低，而性能越来越强大，这使其应用已经无处不在，遍及各个领域。例如电机控制、条码阅读器/扫描器、消费类电子、游戏设备、电话、HVAC、楼宇安全与门禁控制、工业控制与自动化和白色家电（洗衣机、微波炉）等。

随着时间的推移，嵌入式系统由独立工作走向了网络控制（典型的系统就是集散控制系统），此时嵌入式系统的编程模式依然是面向独立的微控制器（微处理器）。要把这些嵌入式系统组成网络，需要为所有控制器增加兼容的通信接口硬件，并设计兼容的通信协议。而且，每个系统需要对硬件通信接口编程及对通信协议编程后才可能组成网络。这个设计无疑是复杂的。

用面向寄存器的编程模式编写联网控制系统的步骤如图 3 所示，面向 API 的编程模式的步骤如图 4 所示。图 3、图 4 的右边是编写控制远程蜂鸣器鸣叫程序的步骤。至于面向端口的编程模式，目前主要是 PLC，它的开发步骤比较简单，读者可以参考 PLC 的开发手册。不过，PLC 一般支持有限的远程设备，并且成本高昂，很多时候并不适用。通过图 3 和图 4 可以看出，对于联网的控制系统，这两种编程模式的步骤基本相同。面向寄存器的编程模式开发难度很大，而面向 API 的编程模式相对小一些，不过任务依然艰巨。

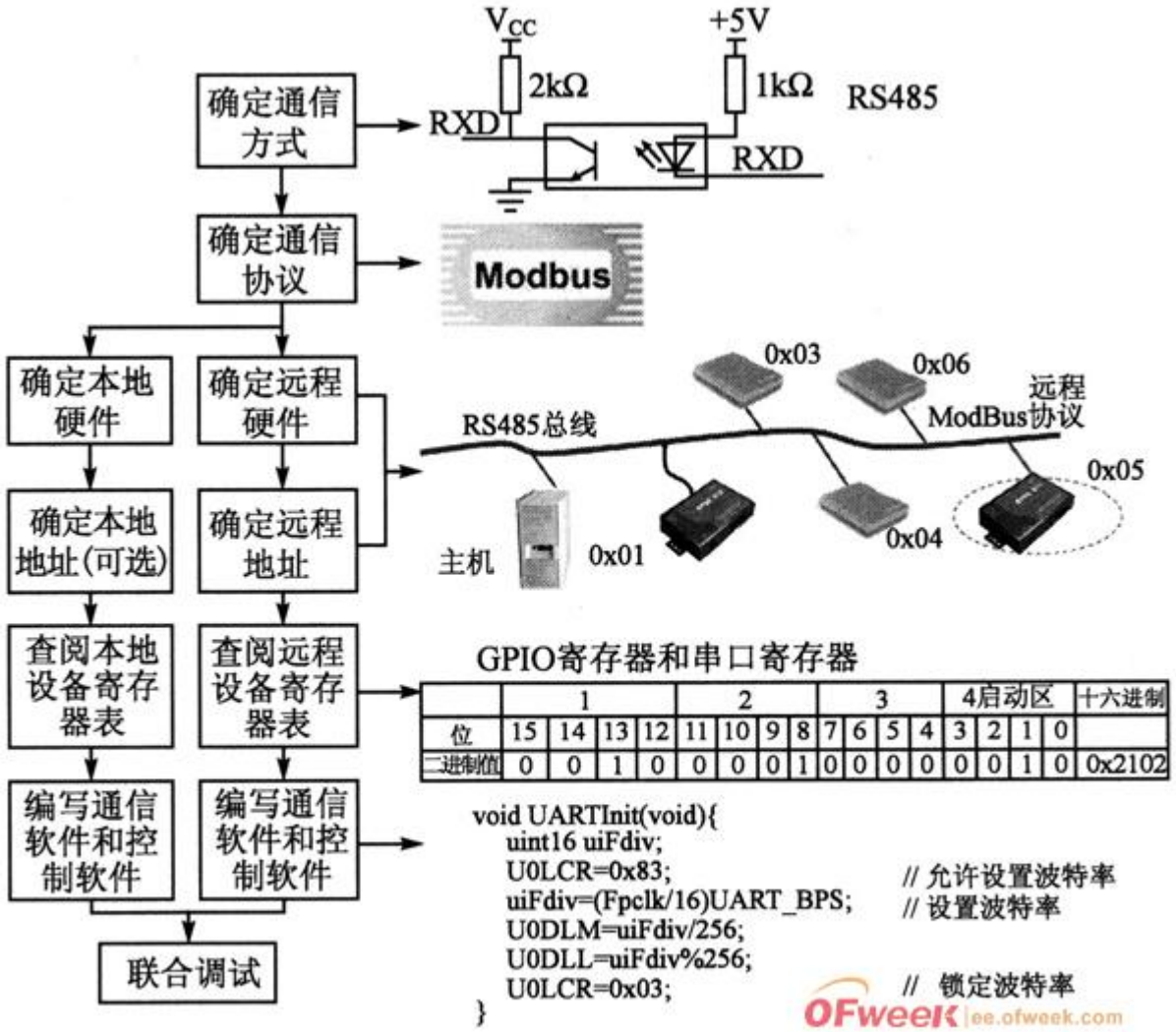


图 3　面向寄存器的网络设备开发步骤

OFweek ee.ofweek.com 电子工程网

各种编程模式的对比如表 2 所列。

表 2　传统编程模式开发联网控制系统对照表

研发模式	研发周期	代码量	维护工作	移　植	研发难度	功能与灵活性	对研发人员的要求
基于寄存器	很长	巨大	很复杂	难度大	很高	高	很高
基于 API	长	大	复杂	比较方便	高	较高	高
基于端口	较短	小	简单	仅限于 PLC	低	低	低

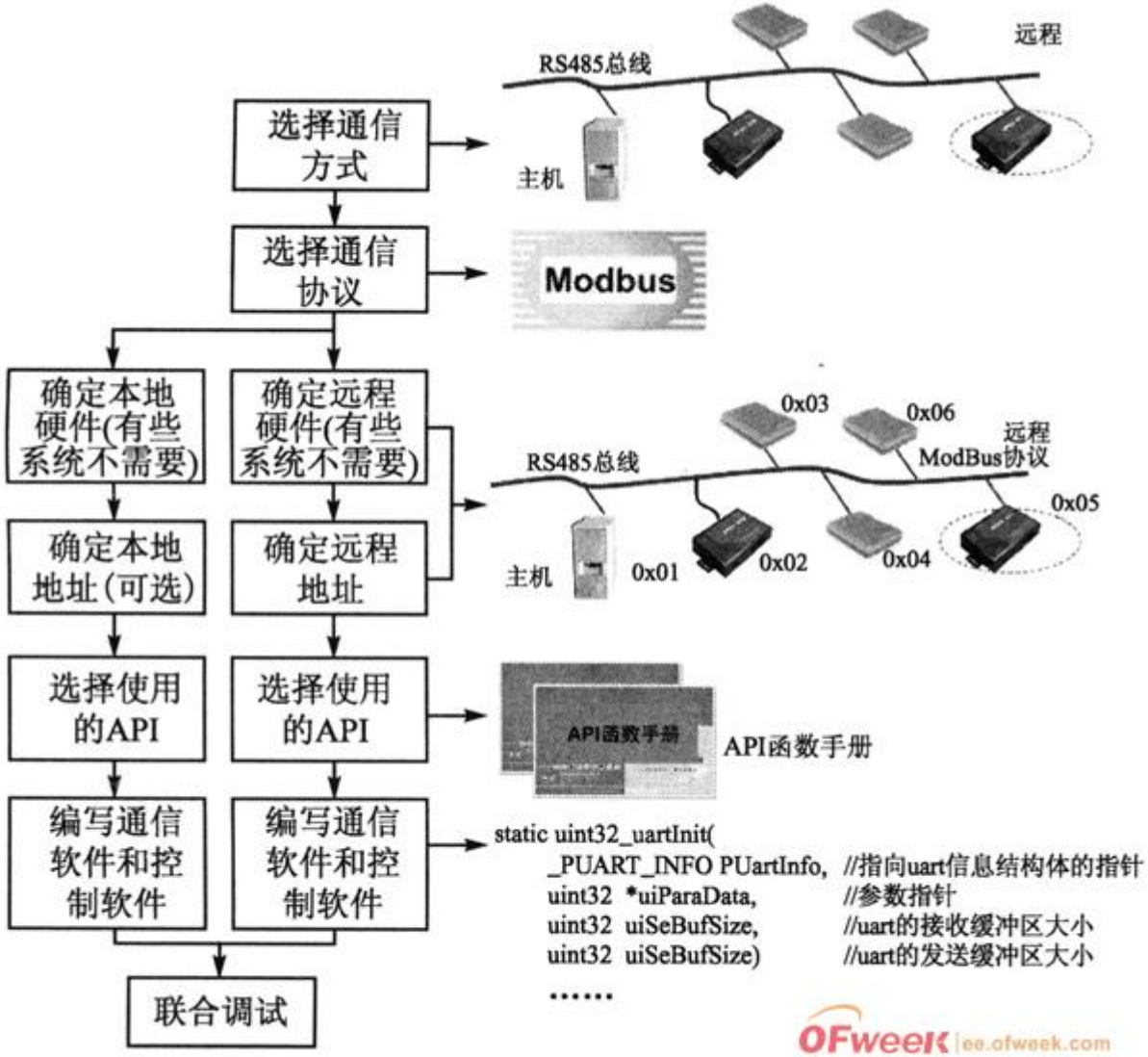
OFweek ee.ofweek.com 电子工程网

现在，组网的范围更加广泛：不但需要本地组网，还需要远程组网；不但控制设备之间需要互连，控制设备与普通计算机之间还需要互连，以及不同厂商的设备之间也要互连。这些要求无疑加剧了系统编程的复杂性。

3 面向设备的编程模式

3.1 范 例

面向设备的编程模式是由面向 API 的编程模式和面向端口的编程模式继承发展而来的，具有两者的优点，避免了各自的缺点，同时极大地增强了组网能力。



OFweek ee.ofweek.com 电子工程网

图 4 面向 API 的网络开发步骤

这里依然以开发控制远程蜂鸣器的嵌入式系统为例，其开发流程如图 5 所示。图 5 的左边是面向设备的编程模式，右边是开发步骤。

通过查看远程设备图，得知蜂鸣器的端口地址为 0x1111,写 1 为鸣叫，写 0 为停止鸣叫。

这种编程模式非常简单。事实上延时功能已经定义成本地端口，真实的程序将更简单。面向设备编程模式与传统编程模式的对比如表 3 所列。

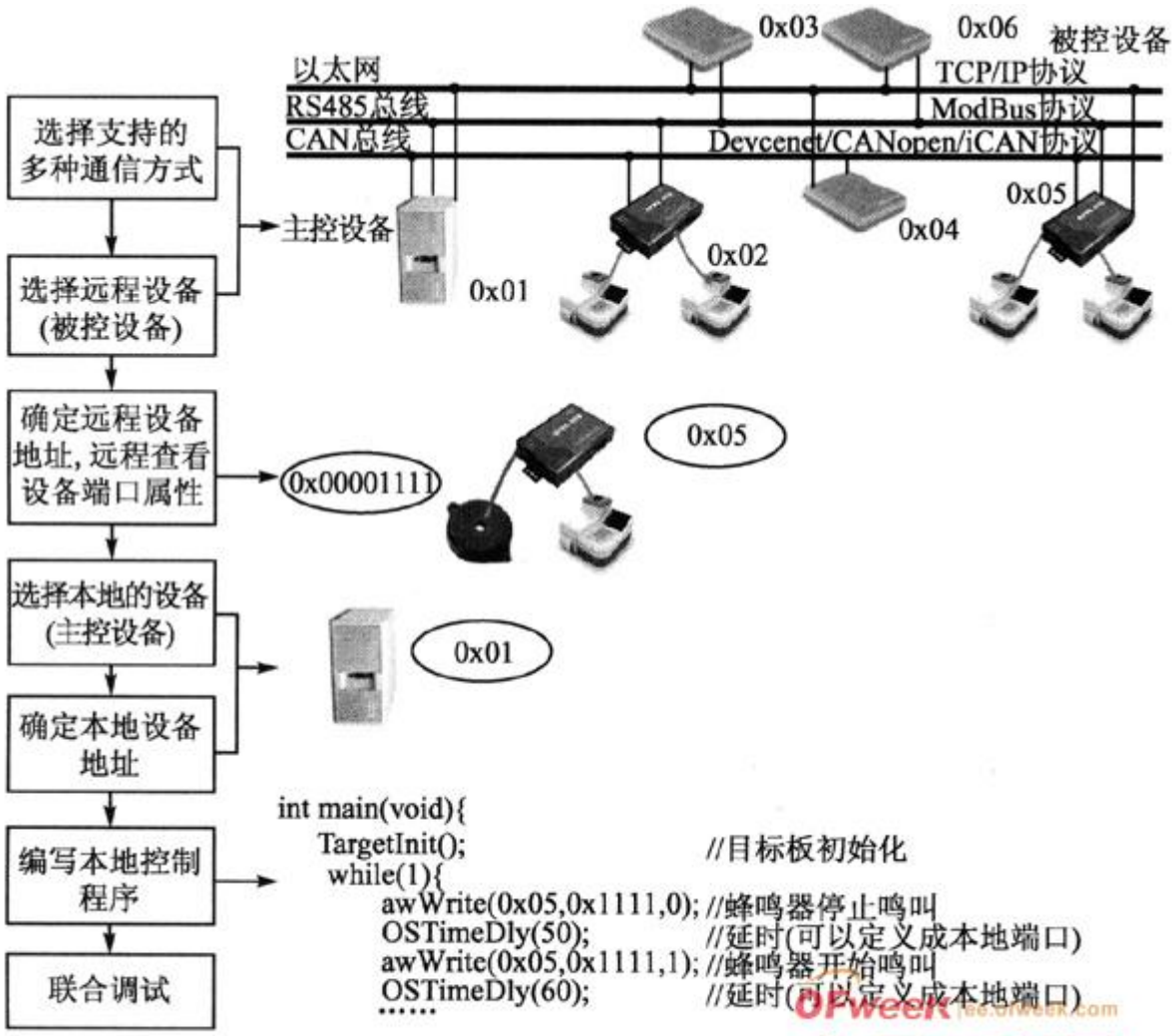


图 5　AnyWhere 开发步骤

表 3　AnyWhere 与传统编程模式的对照表

研发模式	研发周期	代码量	维护工作	移　植	研发难度	功能与灵活性	对研发人员的要求
基于寄存器	很长	巨大	很复杂	难度大	很高	高	很高
基于 API	长	大	复杂	比较方便	高	较高	高
基于端口	较短	小	简单	仅限于 PLC	低	低	低
AnyWhere	较短	小	简单	方便	较低	高	较低

3.2 设计目标

针对目前嵌入式系统设计的困境，本文提出"面向设备的编程"这一概念。研发人员不需要考虑硬件细节和网络细节，使用同一种方式操作本地硬件和远程硬件。

与传统编程模式不同，面向设备的编程模式把所有通过网络连接的嵌入式系统和计算机作为一个整体来考虑。研发人员只需要知道设备地址和设备内端口地址的分配即可，不需要知道设备如何连接到系统，可以通过有限的几个函数操作设备。

3.3 特 点

AnyWhere 最大的特点是着眼于系统，是系统级解决方案。一个系统中的所有嵌入式设备都使用 AnyWhere 兼容设备，整体效果最佳。除了这个特点外，AnyWhere 还有以下特点：

①使用 ANSIC 编程。将来可能增加编程语言支持。

②编程接口统一。无论操作设备的什么功能，都使用有限的几个函数操作。

③编程不区分远程设备和本地硬件。系统保留 1 个系统地址（符号为 AW_LOCAL_ADDR,值为 0x00000000）用于识别本地设备，用这个地址操作的就是本地设备。大多数情况下，设备也可以使用设备的真实地址来访问本地硬件。这样，设备可以使用同样的接口访问本地硬件和远程设备。

④多协议多网络支持。AnyWhere 默认协议计划支持 RS232、RS485、RS422、以太网、CAN、USB 等网络。AnyWhere 还计划支持 ModeBus、iCAN、CANOpen、DeviceNet、J1939、DMX512、MVB 等协议。用户还可以通过多协议接口增加特定的协议。

⑤协议及链路自动动态匹配。研发人员只需要知道设备的地址就可以编程，而不需关心主控设备与被控设备之间的网络与协议匹配问题。系统会自动选择两者均支持（并且当前网络结构支持）的协议。如果网络结构发生变化，系统会再次主动选择协议。这些过程都是透明的，研发人员无需关心。

⑥提供被控设备编程接口。用户可以通过这个接口设计特殊的被控设备。

4 基本设计思想

（1）总体设计思想

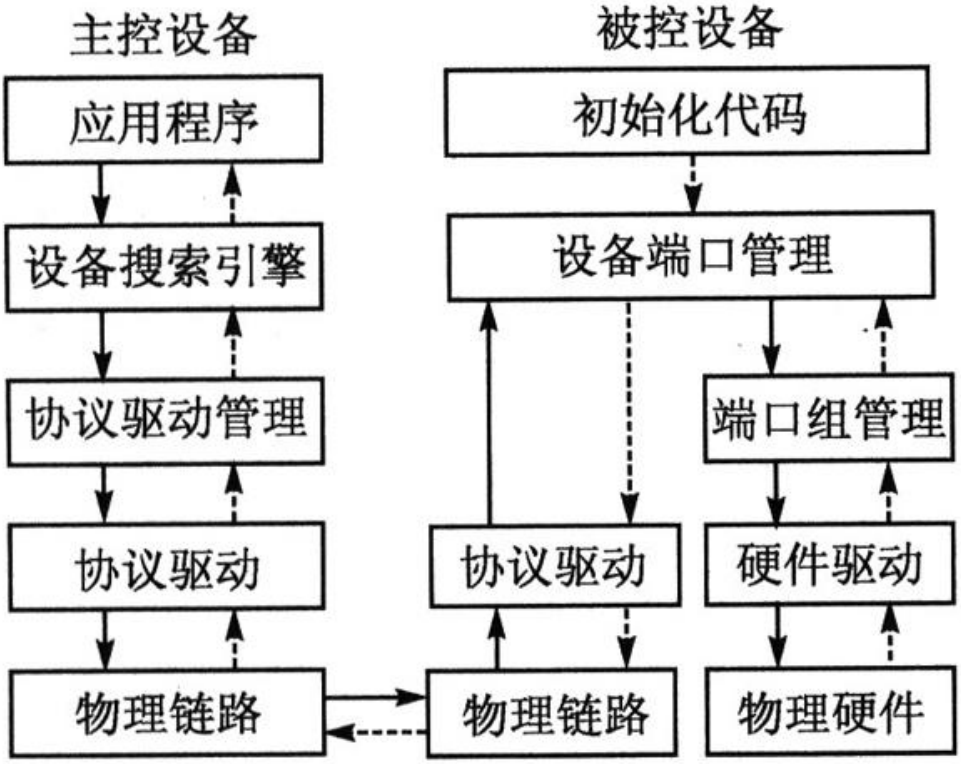
AnyWhere 把所有用网络连接起来的嵌入式系统作为一个整体来考虑。依据其在系统中的作用，把嵌入式系统分为主控设备和被控设备两类。

主控设备通过远程调用来控制被控设备。每当主控设备调用 AnyWhere 的主机接口核心函数时，对应的被控设备执行相应的函数。被控设备的函数执行完毕后，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。

（2）基本框图

AnyWhere 的基本框图如图 6 所示。

OFweek 电子工程网



OFweek ee.ofweek.com 电子工程网

图 6　AnyWhere 基本框图

（3）一般处理流程

为主控设备访问远程设备中 awRead（）函数的一般处理流程是：主控设备首先查找 ARP 表，如果 ARP 表中存有被控设备信息，调用被控设备函数开始执行；如果 ARP 表中不存在此远程设备的情况，则需要请求添加此设备；在远程设备添加成功后，调用被控设备函数开始执行；程序处理完成后应答返回。

5 主要的用户编程接口

5. 1 主控设备编程接口（核心编程接口）

这是一般用户使用的接口，也是最常用的 API.这部分有 4 个函数，分别是 awRead（）、awWrite（）、awReadEx（）和 awWriteEx（）。其中函数 awRead（）和 awWrite（）是对指定设备的指定端口用默认的模式读写，读写的数据都会转化为 32 位无符号数。而 awReadEx（）和 awWriteEx（）用于对端口一次读写多个数据，需要指定读写模式，这个模式还必须与端口的模式一致。

5.2 被控设备编程接口

在设计一个控制系统时，被控设备一般会选择标准设备，不需要用户编程。如果使用非标准的被控设备，就需要进行产品研发。从图 6 可以看出，被控设备

的应用程序仅仅是初始化而已。如果用户选择广州致远电子有限公司的半成品模块，大多数情况下也无需开发，只需通过向导（PC 机程序）配置需要的功能就可以生成需要的代码。如果这些半成品模块不能完全满足系统需求，就要进行研发。

OFweek电子工程网