《嵌入式系统原理与应用技术》

袁志勇 王景存
章登义 刘树波

北京: 北京航空航天大学出版社, 2009.11

PPT教学课件

第十章 嵌入式网络接口技术

9.1 网络接口技术概述

- 基于网络的分布式嵌入式系统应用
  - 处理的任务是分布式的，需要把计算源置于靠近事件的发生地，并通过网络进行协调工作
  - 网络环境下信息共享程度的提高
  - 实际应用中，根据需要选取所需的协议来构造嵌入式系统的网络
  - 可用于构造嵌入式网络的协议有多种，常见的有：**RS-485**，**I2C**总线，**CAN**总线，以太网协议等

9.1.1 分布嵌入式系统结构

构造基于网络的分布嵌入式系统应用主要基于如下几点考虑：

- 所处理的任务是分布式的，因此需要将嵌入式装置或计算源置于靠近事件的发生地，并通过网络进行协调工作；
- 提高网络环境下的信息共享程度；
- 嵌入式系统需要较高的容错性能，采用分布嵌入式结构可以提高系统的容错性。

分布式嵌入式系统结构

- 基本组成要素：处理单元/元素PE(Processing Element)和通信网络
- 处理单元可以是一个完整的控制器(内部有CPU、存储器及相关I/O等)，也可以是支持网络协议的不可编程单元(如具有网络接口的传感器和其他执行机构等)

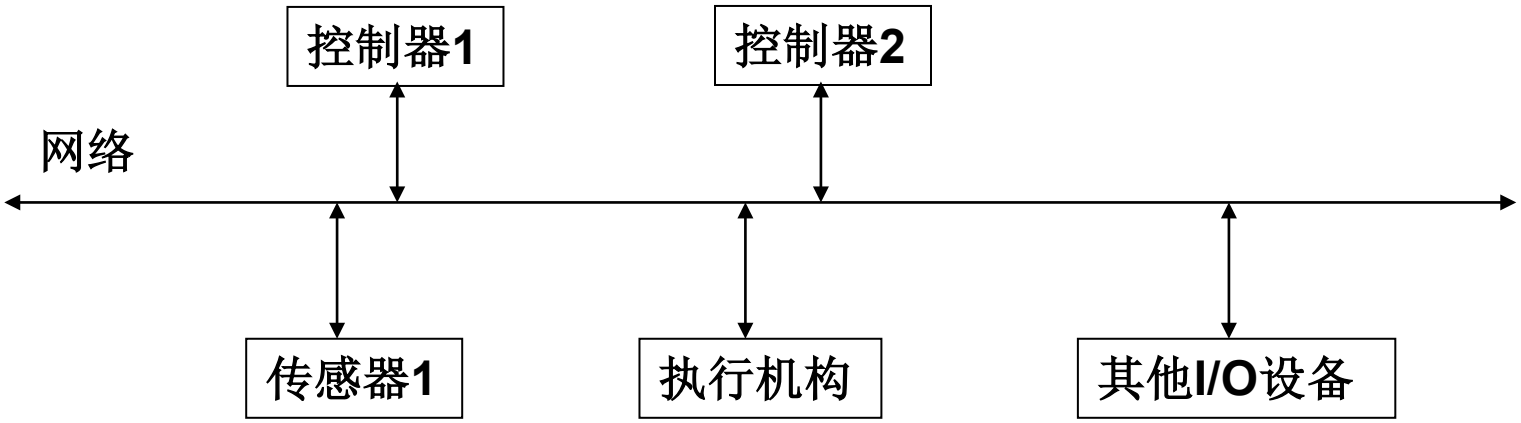


图9.1 分布嵌入式系统基本结构

- 说明：
- 分布嵌入式系统中的网络链路有时也被称为“总线”，但这里的“总线”与微处理器总线是不同的(无存储系统，不支持网络总线上的取指操作)
- 分布嵌入式系统中网络链路通常采用“分层”的体系结构模型来实现各部件之间的通信(对分布嵌入式系统中的网络而言，一般只实现OSI七层协议中的几层)

9.1.2 分布嵌入式网络通信方式

在分布嵌入式系统应用领域，通常是按成本预算和实际需求来构造不同方式的分布嵌入式网络。可用于分布嵌入式系统的网络有许多，分布嵌入式网络通信方式主要有点对点通信方式和总线通信方式。

所谓**点对点通信方式**是指两台或两台以上嵌入式系统之间相互交换信息的方式，它具有安全、快捷、直观、同步和经济等优点。如两台嵌入式系统采用RS-232通信即是一种典型的点对点通信方式。

在分布式嵌入式系统网络应用中，通常是多个设备相互连接，这时，需要采用总线通信方式。**在总线通信方式中，连接到总线上的所有处理单元都必须有自己的唯一地址或标识；总线上的通信**一般是以**报文组的形式进行**的，一个分组报文中包含一个目的地址和要被传送的数据以及检错纠错信息，见图9.2所示。

头	地址	数据	纠错信息

图9.2 分组报文格式

分布嵌入式网络中的总线必须要满足仲裁机制，即网络总线上同时出现传送操作时需要进行必要的选择。

仲裁机制主要有两种类型：(1)固定优先级仲裁机制，即嵌入式网络中采用固定优先级方式给予竞争系统优先级。当嵌入式网络中高优先级和低优先级的嵌入式系统都要同时进行大量的数据传送时，往往是高优先级系统先传送数据包，然后低优先级系统再传送数据包；(2)公平仲裁机制，即嵌入式网络中各系统具有相同的待遇，循环仲裁是一种常见的公平仲裁机制。

10.2 I2C接口

- I2C/IIC（Inter－Integrated Circuit）总线是一种由PHILIPS公司(现更名为NXP)针对MCU需要而研制的两线式串行总线，用于连接MCU及其外围设备。
- I2C总线最主要的优点是其简单性和有效性。
- 由于接口直接在组件之上，因此I2C总线占用的空间非常小，减少了电路板的空间和芯片管脚的数量，降低了互联成本。
- 总线的长度可高达25英尺(约7.6m)，并且能够以100Kbps的最大传输速率支持40个组件。
- 另一个优点是，它支持多主控(multi-mastering)， 其中任何能够进行发送和接收的设备都可以成为主总线。一个主控能够控制信号传输和时钟频率。当然，在任何时间点上只能有一个主控。

◆I2C总线工作原理

- I2C总线系统组成
  - I2C总线是由数据线SDA (Serial DAta line)和时钟SCL (Serial Clock Line)构成的串行总线，可发送和接收数据
  - 在CPU与被控IC之间、IC与IC之间进行双向传送，最高传送速率100kbps
  - 各种从设备均并联在这条总线上，但就像电话机一样只有拨通各自的号码才能工作，因此每个设备都有唯一的地址
  - 在信息的传输过程中，I2C总线上并接的每一设备既是主设备(或从设备)又是发送器(或接收器)，这取决于它所要完成的功能

I2C总线是多主系统：系统可以有多个I2C节点设备组成，并且可以是多主系统，任何一个设备都可以为主I2C；但是任一时刻只能有一个主I2C设备，I2C具有总线仲裁功能，保证系统正确运行。

主I2C设备发出时钟信号、地址信号和控制信号，选择通信的从IIC设备并控制收发。

系统要求：1. 各个节点设备必须具有IIC接口功能；2. 各个节点设备必须共地；3. 两根信号线必须接上拉电阻。如下图所示。

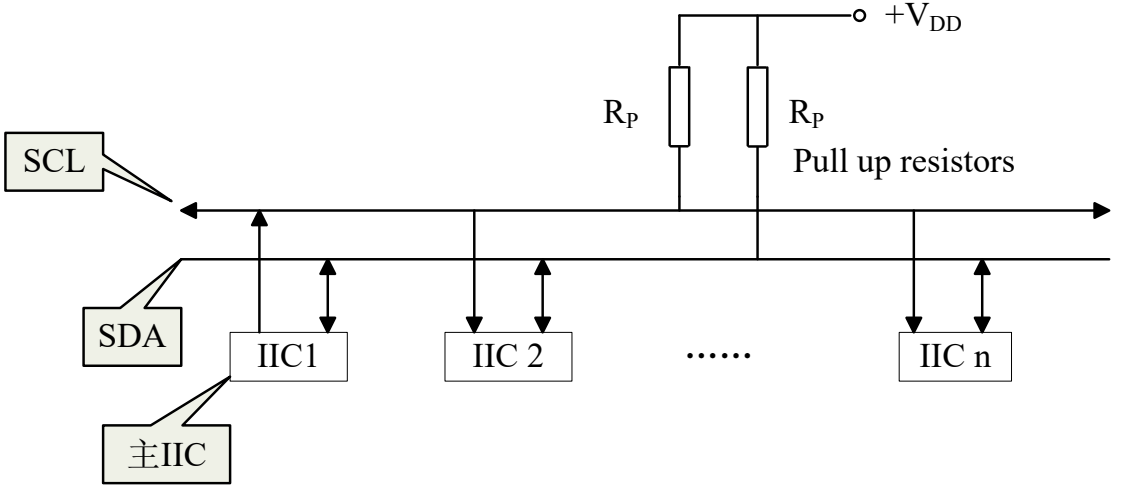


图9.3 多IIC设备接口示意图

I2C总线的状态和信号

1. 空闲状态

SCL和SDA均处于高电平状态，即为总线空闲状态(空闲状态为何是高电平的道理很简单，因为它们都接上拉电阻)。

2. 占有总线和释放总线

器件若想使用总线应当先占有它，**占有总线的主控器向SCL线发出时钟信号**。数据传送完成后应当及时释放总线，即解除对总线的控制(或占有)，使其恢复成空闲状态。

3. 开始/启动信号(S)

启动信号由主控器产生；在SCL信号为高时，SDA产生一个由高变低的电平变化，产生启动信号。

- **4. 结束/停止信号(P)**

  当SCL线高电平时，主控器在SDA线上产生一个由低电平向高电平跳变，产生停止信号。启动信号和停止信号的产生见下图如所示。

- **5. 应答/响应信号(A/ACK)**

  应答信号是对字节数据传输的确认。占1位，数据接收者接收1字节数据后，应向数据发出者发送一个应答信号。对应于SCL第9个应答时钟脉冲，若SDA线仍保持高电平，则为非应答信号($\overline{ACK}$/NA)。低电平为应答，继续发送；高电平为非应答，结束发送。

- **6. 控制位信号**

  占1位，I2C主机发出的读写控制信号，高为读、低为写（对I2C主机而言）。控制位(方向位)在寻址字节中。

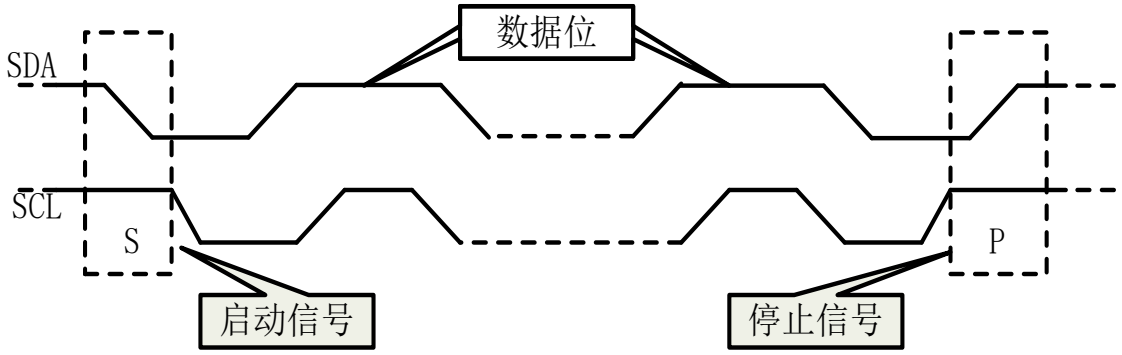


图9.4 启动信号和停止信号的产生

7. 地址信号

为从机地址，占7位，如下表所示，称之为“寻址字节”，各字段含义如下：

D7	D6	D5	D4	D3	D2	D1	D0
DA3	DA2	DA1	DA0	A2	A1	A0	R/ W

器件地址（DA3---DA0）：是I2C总线接口器件固有的地址编码，由器件生产厂家给定。如I2C总线EEPROM AT24C××器件的地址为1010等。

引脚地址（A2、A1、A0）：由I2C总线接口器件的地址引脚A2、A1、A0的高低来确定，接电源者为1，接地者为0。

读写控制位/方向位（R/ W）：1表示主机读，0表示主机写。

7位地址和读写控制位组成1个字节(寻址字节)。

8. 等待状态(*)

在I2S总线中，赋予接收数据的器件有使系统进行等待状态的权力，但等待状态只能在一个数据字节完整接收之后进行。

例如，当进行主机发送从机接收的数据传送操作时，若从机在接收到一个数据字节后，由于中断处理等原因而不能按时接收下一个字节；从机可以通过把SCL下拉为低电平，强行使主机进入等待状态。在等待状态下，主机不能发送数据，直到从机认为自己能继续接收数据时，再释放SCL线，使系统退出等待状态，主机才可以继续进行后续的数据传送。

IIC总线基本操作

- 串行数据SDA和串行时钟SCL线在连接到总线的器件间传递信息。
- 每个器件都有一个唯一的地址标识，无论是MCU、LCD驱动器、存储器或键盘接口。
- 都可以作为一个发送器或接收器，由器件的功能决定。很明显，LCD驱动器只是一个接收器，而存储器则既可以接收又可以发送数据。
- 器件除了能看作发送器和接收器外，在执行数据传输时它也可以被看作是主机或从机。
- 主机是初始化总线的数据传输并产生允许传输时钟信号的器件，此时任何被寻址的器件都被认为是从机。

术　语	描　　述
发送器	发送数据到总线的器件
接收器	从总线接收数据的器件
主机	初始化发送产生时钟信号和终止发送的器件
从机	被主机寻址的器件
多主机	同时有多于一个主机尝试控制总线但不破坏报文
仲裁	是一个在有多个主机同时尝试控制总线但只允许其中一个控制总线并使报文不被破坏的过程
同步	两个或多个器件同步时钟信号的过程

启动和停止条件

- 在SCL 线是高电平时，SDA 线从高电平向低电平切换，这个情况表示启动条件
- 当SCL是高电平时，SDA 线由低电平向高电平切换，表示停止条件
- 启动和停止条件一般由主机产生。总线在起始条件后被认为处于忙的状态，在停止条件的某段时间后，总线被认为再次处于空闲状态
- **如果产生重复启动条件Sr而不产生停止条件，总线会一直处于忙状态**

I2C总线数据传输格式

(1) 一般格式：

S	从I2C地址(7位)	R/$\overline{W}$	A	传输数据 ……	A	P

(2) 主控制器写操作格式：

S	从I2C地址	$\overline{W}$	A	数据1	A	数据2	A	……	数据n	A/$\overline{A}$	P

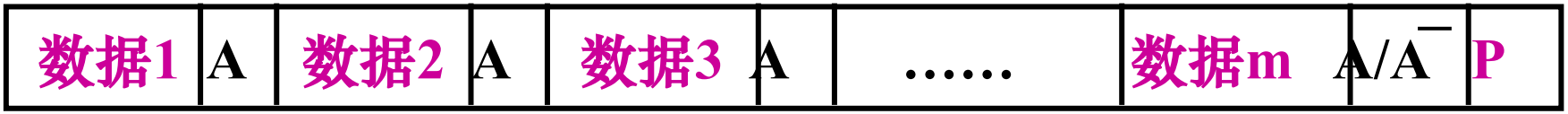
红色起始信号S、地址信号、控制信号$\overline{W}$、各个数据、结束信号P，均为主I2C设备发送、从I2C设备接收；黑色的应答信号A/$\overline{A}$由从I2C设备发送、主I2C设备接收。

(3) 主控制器读操作格式：

S	从I2C地址	R	A	数据1	A	数据2	A	……	数据n	$\overline{A}$	P

红色的信号均为主I2C设备发送、从I2C设备接收；黑色的信号均为从I2C设备发送、主I2C设备接收。

(4)主控制器读/写操作格式：



由于在一次传输过程中要改变数据的传输方向，因此**起始信号和寻址字节都要重复一次**，而中间可以不要结束信号。

在一次传输中，可以有多次启动信号。

标准模式I2C 总线规范的扩展(*:了解)

- 标准模式I2C总线规范在80年代初期已经存在。它规定数据传输速率可高达100kbit/s ，而且7 位寻址。这个概念在普及中迅速成长。今天它已经作为一个标准被全世界接受，而且Philips Semiconductors和其他供应商提供了几百种不同的兼容IC。
- 为了符合更高速度的要求以及制造更多可使用的从机地址给数量不断增长的新器件，标准模式I2C总线规范不断升级。到今天它提供了以下的扩展:
  - 快速模式位速率高达400kbit/s
  - 高速模式(Hs模式)位速率高达3.4Mbit/s
  - 10位寻址允许使用高达1024个额外的从机地址(*)

9.2.2 S3C2410 I2C接口

- S3C2410的I2C主要有5部分构成：数据收发寄存器、数据移位寄存器、地址寄存器、时钟发生器、控制逻辑等部分。如下图所示。

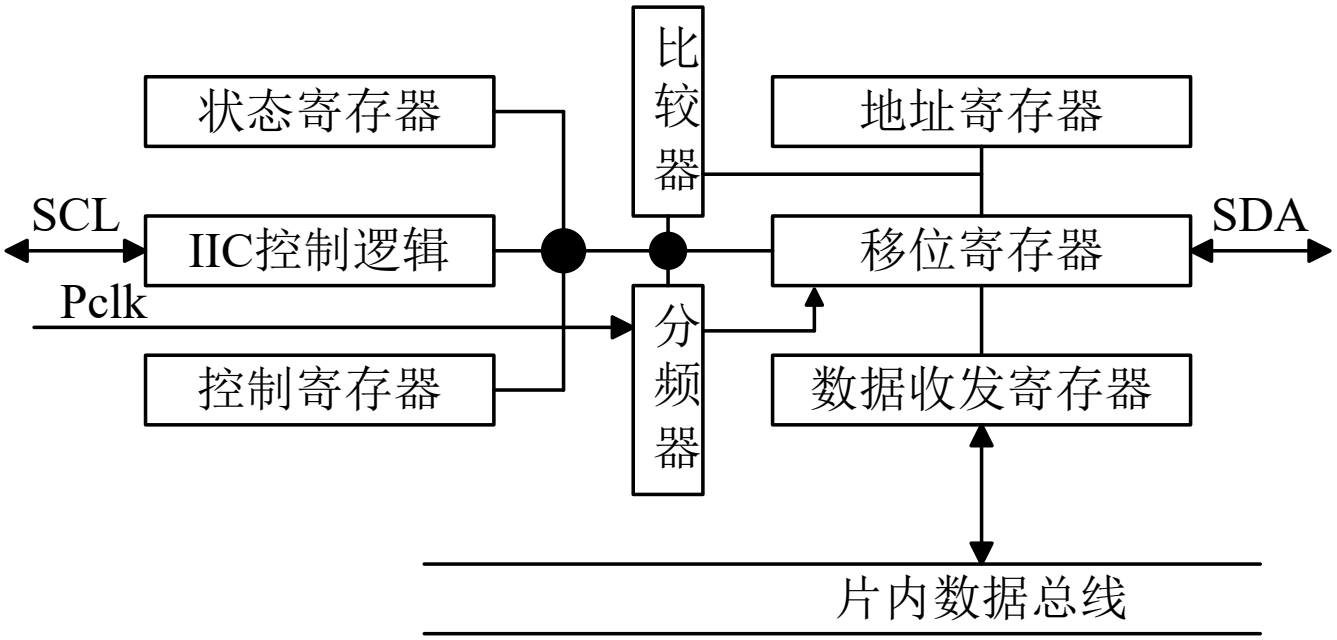


图9.9 S3C2410 IIC串行总线控制器框图

- S3C2410 I2C总线接口具有一个专门的串行数据线和串行时钟线。它有主机发送模式、主机接收模式、从机发送模式和从机接收模式4种操作模式。
- 控制S3C2410 I2C总线操作，需要写数据到IICCON (I²C总线控制寄存器)、IICSTAT (I²C总线控制/状态寄存器)、IICDS (I²C总线Tx/Rx数据移位寄存器)和IICADD (I²C总线地址寄存器)。

- S3C2410芯片支持I2C总线序列接口，其端口E的GPE15用作数据线 (SDA)，GPE14用作连续时钟线(SCL)。这2根信号线用于在S3C2410芯片内部的总线主控器和连接到I2C总线上的外围设备之间传输信息，此数据线和连续时钟线均是双向的。
- 当I2C总线空闲时，GPE15引脚（SDA信号线）和GPE14引脚（SCL信号线）都应设置成高电平。GPE15引脚从高电平转换到低电平时，启动一次传输。当GPE14保持在高电平时，GPE15引脚从低电平转换到高电平则表示传输结束。其传输协议遵循上节所介绍的I2C总线协议。

读写操作

在发送器模式下，数据被发送之后，I2C 总线接口会等待直到 IICDS（I2C 数据移位寄存 器）被程序写入新的数据。在新的数据被写入之前，SCL 线都被拉低。新的数据写入之后， SCL 线被释放。

S3C2410 可以利用中断来判断当前数据字节是否已经完全送出。在 CPU 接收到中断请求后，在中断处理中再次将下一个新的数据写入 IICDS，如此循环。

在接收模式下，数据被接收到后，I2C 总线接口将等待直到 IICDS 寄存器被程序读出。 在数据被读出之前，SCL 线保持低电平。新的数据从读出之后，SCL 线才释放。

S3C2410 也利用中断来判别是否接收到了新的数据。CPU收到中断请求之后，处理程序将从IICDS 读取数据。

总线仲裁

总线仲裁发生在两个主I2C设备中。如果一个主设备欲使用总线，而测得SDA为低电平，则该主设备仲裁不能够使用总线启动传输。这个仲裁过程会延长，直到信号线SDA变为高电平。

每次操作都要进行仲裁。

S3C2410 I2C专用寄存器

- S3C2410有4个专用寄存器

Register	Address	R/W	Description	Reset Value
IICCON	0x54000000	R/W	IIC总线控制寄存器	0x0X
IICSTAT	0x54000004	R/W	IIC总线控制/状态寄存器	0x0
IICADD	0x54000008	R/W	IIC总线地址寄存器	0xXX
IICDS	0x5400000C	R/W	IIC数据发送/接收寄存器	0xXX

1. I2C控制寄存器（IICCON）

字段名	位	意　义	初值
Acknowledge generation	7	应答使能。0：禁止；1：使能 应答电平：Tx时为高；Rx时为低	00
Tx clock source selection	6	发送时钟分频选择。 0：IICCLK = fPCLK/16; 1：IICCLK = fPCLK/512	0
Tx/Rx Interrupt	5	收发中断控制位。0：禁止；1：允许	0
Interrupt Pending flag	4	中断标志位。读：0无，1：有中断请求 写：写0清除中断标志，写1不操作	0
Transmit clock value	3:0	发送时钟预分频值。 Tx clock = IICCLK/(IICCON[3:0]+1)	0

1. I2C控制寄存器（IICCON）(Cont.)

说明：

（1）应答使能问题：一般情况下为使能；在对I2C EEPROM读最后1个数据前可以禁止应答，便于产生结束信号。

（2）中断事件：1）完成收发；2）地址匹配；3）总线仲裁失败。

（3）中断控制位问题：设为0时，中断标志位不能正确操作，故总设为1。

（4）时钟预分频问题：当分频位选择为0时，预分频值必须大于1。

2. I2C控制状态寄存器(IICSTAT)

字段名	位	意　义	初值
Mode selection	7:6	工作模式选择。 00：从收； 01：从发 10：主收； 11：主发	00
Busy / START STOP condition	5	忙状态/启、停控制。读：1示忙；0示闲 写：0产生结束信号，1产生启动信号	0
Serial output	4	数据输出控制。0：禁止；1：允许收发	0
Arbitration Status flag	3	仲裁状态标志。0：仲裁成功； 1：仲裁失败（因为在连续I/O中）	0
Address-as-slave status flag	2	从地址匹配状态。 0：与IICADD不匹配 1：匹配。在收到SART/STOP时清0	0
Address zero status flag	1	0地址状态标志。 0：收到的为非0地址 1：收到0地址。在收到SART/STOP时清0	0
Last-received bit status flag	0	最后收到位状态。0：最后位为0，收到ACK；　1：最后位为1，未收到ACK。	0

2. I2C控制状态寄存器（Cont.）

IICSTAT控制字：

启动主设备发送：0xF0；结束主设备发送：0xD0

启动主设备接收：0xB0；结束主设备接收：0x90

3. I2C地址寄存器（IICADD）

Register	Address	R/W	Description	Reset Value
IICADD	0x54000008	R/W	地址寄存器	0xXX

字段名	位	意　义	初值
Slave address	7:1	7位从地址。	0xXX
Not mapped	0	不用	-

说明：

（1）对从设备，该地址有意义，对主设备其值无意义。

（2）只有在不发送数据时（数据传送控制位IICSTAT[4] =0）才能对其写；任何时间都可以读。

4. I2C数据发送/接收寄存器（IICDS）

Register	Address	R/W	Description	Reset Value
IICDS	0x5400000C	R/W	数据发送/接收移位寄存器	0xXX

字段名	位	意　义	初值
Data shift	7:0	8位移位接收或移位发送的数据	0xXX

说明：

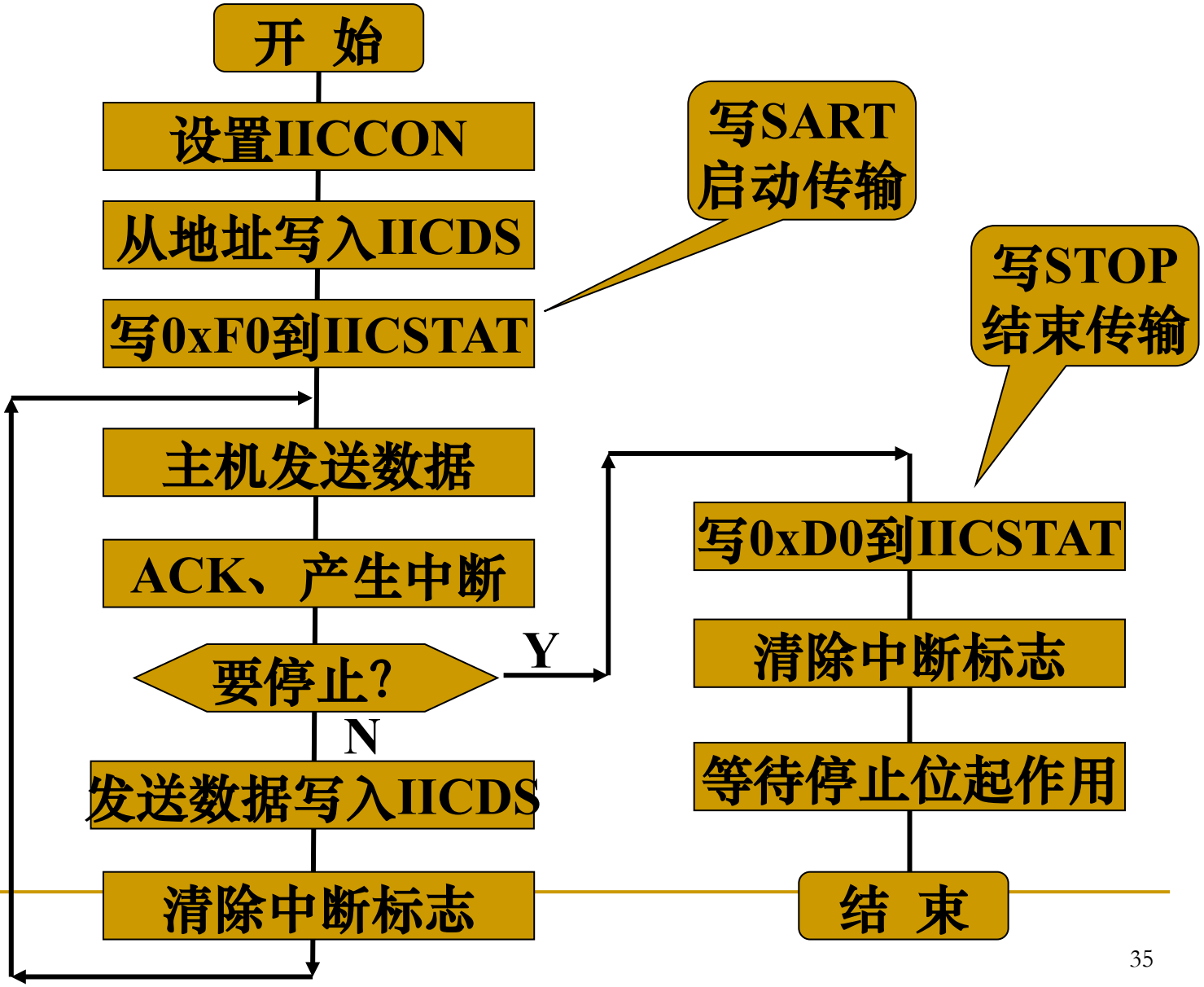
（1）在本设备接收时，对其作读操作得到对方发来的数据。任何时间都可以读。

（2）在本设备发送时，对其写操作，将数据发向对方。

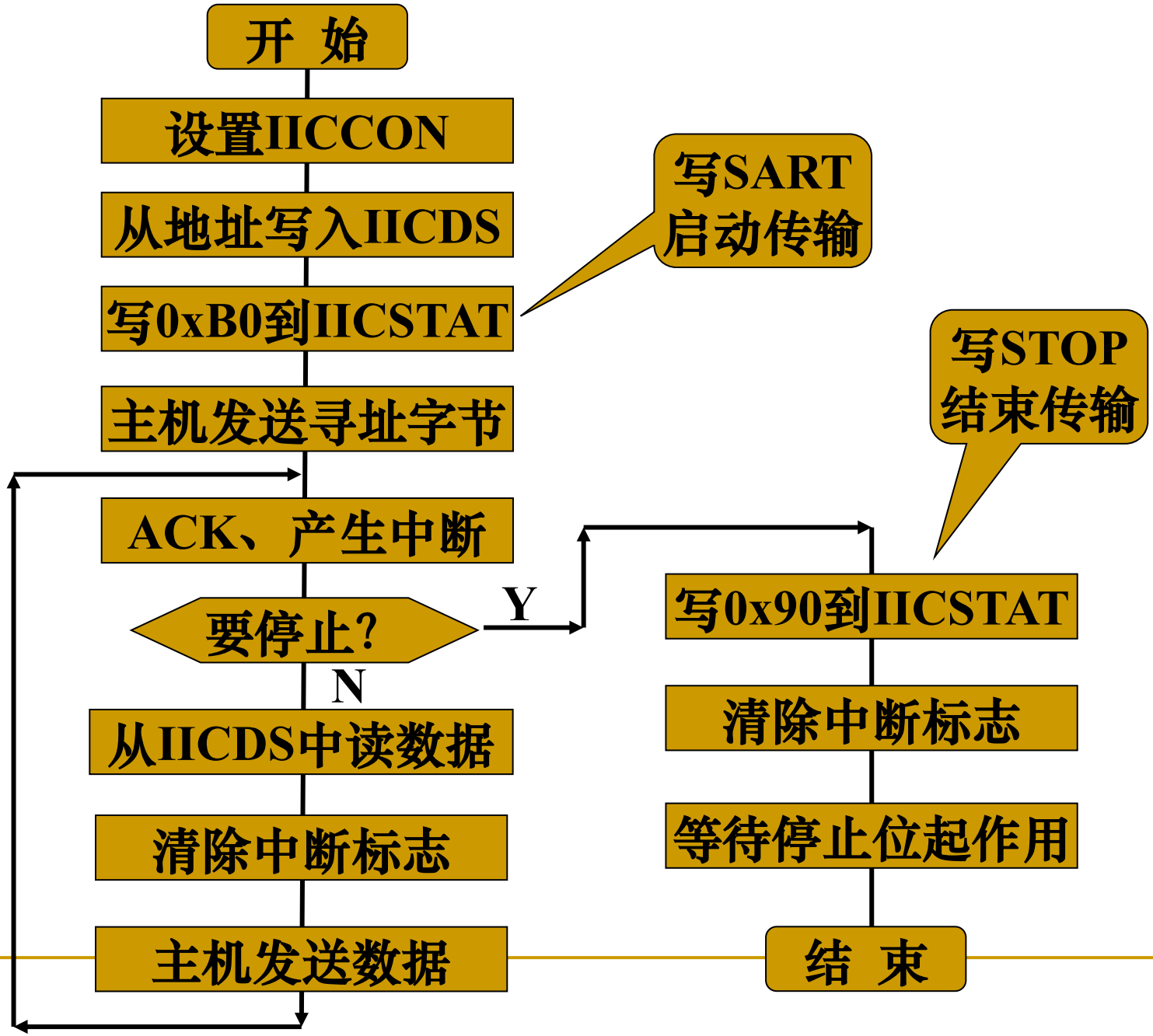
（3）欲发送数据，必须使数据传输控制位IICSTAT[4]=1才能对其写。

I2C操作流程

1. 主发送模式流程



2. 主接收模式流程


开 始
设置IICCON
从地址写入IICDS
写0xB0到IICSTAT
写SART
启动传输
主机发送寻址字节
ACK、产生中断
要停止?
Y
N
写STOP
结束传输
写0x90到IICSTAT
从IICDS中读数据
清除中断标志
清除中断标志
等待停止位起作用
主机发送数据
结 束

I2C总线编程举例

- I2C总线编程除了需要对I2C总线的专用寄存器进行初始化编程外，还需要按照I2C总线的时序要求编写传送程序和接收程序

（1）初始化编程

在任何I2C总线的传送和接收操作之前，必须执行初始化程序。初始化程序的主要功能是：

- 配置S3C2410芯片相关的I/O引脚为I2C总线所需的功能引脚
- 若有必要，在IICADD寄存器中写入本芯片的从地址
- 设置IICCON寄存器，用来使能中断、设定SCL周期等
- 设置 IICSTAT 以使能传送模式等

例：试编写一程序，用S3C2410的I2C接口对串行EEPROM（I2C接口）进行读/写操作，写入一组数据，然后读出并显示出来，检验是否正确。

分析：S3C2410的I2C为主设备，串行EEPROM的I2C为从设备，进行的操作为主设备写和主设备读。

解：　1. 设置I2C控制寄存器

1）收发传输：IICCON=0b 1 0 1 0 1111 = 0xAF

含义：应答使能、时钟分频为 IICCLK = f PCLK /16 、中断使能、清除中断标志、预分频值取15。

2）接收结束传输：IICCON=0b 0 0 1 0 1111 = 0x2F

含义：禁止应答（非应答）、时钟分频为 IICCLK = f PCLK /16 、中断使能、清除中断标志、预分频值取15。

2. I2C控制状态寄存器

1）主模式发送、启动传输

IICSTAT=0b 11 1 1 0 0 0 0 = 0xF0

含义：主设备发送、启动传输、输出使能、低4位为状态

2）主模式发送、结束传输

IICSTAT=0b 11 0 1 0 0 0 0 = 0xD0

含义：主设备发送、结束传输、输出使能、低4位为状态

3）主模式接收、启动传输

IICSTAT=0b 10 1 1 0 0 0 0 = 0xB0

含义：主设备接收、启动传输、输出使能、低4位为状态

4）主模式接收、结束传输

IICSTAT=0b 10 0 1 0 0 0 0 = 0x90

含义：主设备接收、结束传输、输出使能、低4位为状态

3. 地址寄存器设置

1）S3C2410地址寄存器：

作为从设备地址为0x10（作为主设备无意义）

2）AT24C04 EEPROM芯片地址：

作为从设备地址为 0xA0

注*: AT24C04存储容量为512字节，该器件的编址为1010A2A1P0，P0为页地址，P0=0表示寻址低256字节单元，P0=1表示寻址高256字节单元。

4. 寻址字节值

所寻址的“从设备地址+操作控制命令”($\overline{R}/W$):

1）主设备发送： 0xA0

2）主设备接收： 0xA1

5. 程序

```
#include <string.h>
#include "2410addr.h"
#include "2410lib.h"
#include "def.h "

……
```

```
void Test_Iic(void)
{
    unsigned int i,j;
    static U8 data[256];

    Uart_Printf("[ IIC Test using AT24C02 ]\n");

    rGPEUP  |= 0xc000;
    // GPE15 and GPE14  pull-up resistors disable.
    rGPECON |= 0xa0000000; //GPECON[31:30]=10b: GPE15=IICSDA
                           //GPECON[29:28]=10b: GPE14=IICSCL
    rIICCON  = (1<<7) | (0<<6) | (1<<5) | (0xf); //rIICCON  = 0xAF
    rIICADD  = 0x10;       // S3C2410作为I2C总线从设备的地址
    rIICSTAT = 0x10;                  //I2C总线数据输出允许(Rx/Tx)
```

```
Uart_Printf("Write test data into AT24C02(0-255)\n");

for(i=0;i<256;i++)
    _Wr24C02(0xa0,(U8)i,i);

for(i=0;i<256;i++)
    data[i] = 0;

Uart_Printf("\nRead test data from AT24C02\n");

for(i=0;i<256;i++)

_Rd24C02(0xa1,(U8)i,&(data[i]));
```

存储器地址

所写数据

存储数据

存储器地址

```
for(i=0;i<16;i++)
    {
        for(j=0;j<16;j++)
            Uart_Printf("%2x ",data[i*16+j]);
        Uart_Printf("\n");
    }
}
```

```
void _Wr24C02(U8 slvAddr,U8 addr,U8 data)
{   // slvAddr: 从设备地址，此处为0xa0
    // addr: 待写入数据到芯片的地址
    // data: 待写入的数据
    rIICDS       = slvAddr;              //发送从设备地址
    rIICSTAT     = 0xf0;                 //启动发送

    while(rIICCON & 0x10==0);            //查询Tx中断状态
    rIICDS       = addr;                 //发送存储器地址
    rIICCON  = 0xaf;                     //清除中断状态

    while(rIICCON & 0x10==0);            //查询中断状态
    rIICDS= data ;                       //发送数据
    rIICCON  = 0xaf;                     //清除中断状态
```

```
    while(rIICCON & 0x10==0);     //查询中断状态
    while(rIICSTAT&1);            //等待I2C EEPROM应答ACK

    rIICSTAT = 0xd0;              //Stop(Write)
    Delay(1);                     //等待停止结束生效
}
```

```
void _Rd24C02(U8 slvAddr,U8 addr,U8 *data )
{   // slvAddr: 从设备地址，此处为0xa1
    // addr: 待读入数据的芯片地址
    // data: 待读入的数据
     rIICDS       = slvAddr;              //发送从设备地址
    rIICSTAT     = 0xf0;                 //启动发送

    while(rIICCON & 0x10==0);            //查询Tx中断状态
    rIICDS       = addr;                 //发送存储器地址
    rIICSTAT   = 0xb0;                   //启动接收
    rIICCON     = 0xaf;                 //清除中断状态

    while(rIICCON & 0x10==0);            //查询Rx中断状态
     *data = rIICDS;                     //接收数据
     rIICCON     = 0xaf;                 //清除中断状态
}
```