Exynos 4412处理器IIC总线控制器(包括协议)

2016-01-14 17:13 1126人阅读 评论(0) 收藏 举报

分类：

嵌入式linux设备驱动（8）

一、综述

Exynos4412精简指令集微处理器支持4个IIC总线控制器。为了能使连接在总线上的主和从设备之间传输数据，专用的数据线SDA和时钟信号线SCL被使用，他们都是双向的。

如果工作在多主机的IIC总线模式，多个4412处理器将从从机那接收数据或发送数据给从机。在IIC总线上的主机端4412会启动或终止一个数据传输.4412的IIC总线控制器会用一个标准的IIC总线仲裁机制去实现多主机和多从机传输数据。

通过控制如下寄存器以实现IIC总线上的多主机操作：

控制寄存器： I2CCON

状态寄存器： I2CSTAT

Tx/Rx数据偏移寄存器： I2CDS

地址寄存器： I2CADD

如果I2C总线空闲，那么SCL和SDA信号线将都为高电平。在SCL为高电平期间，如果SDA有由高到低电平的跳变，那么将启动一个起始信号，如果SDA有由低到高电平的跳变，将启动一个结束信号。

主机端的设备总是提供起始和停止信号的一端。在起始信号被发出后，一个数据字节的前7位被当作地址通过SDA线被传输。这个地制值决定了总线上的主设备将要选择那个从设备作为传输对象，bit8决定传输数据的方向(是 读还是写)。

I2C总线上的数据(即在SDA上传输的数据)都是以8位字节传输的，在总线上传输操作的过程中，对发送或接收的数据字节数是没有限制的。I2C总线上的主/从设备发送数据总是以一个数据的最高位开始传输(即MSB方式)，传输完一个字节后，应答信号紧接其后。

二、I2C总线接口特性

9个通道多主、从I2C总线接口。其中8个通道作为普通接口(即I2C0、I2C1....)，1个通道作为HDMI的专用接口。

7位地址模式。

串行，8位单向或双向的数据传输。

在标准模式中，每秒最多可以传输100k位，即12.5kB的数据量。

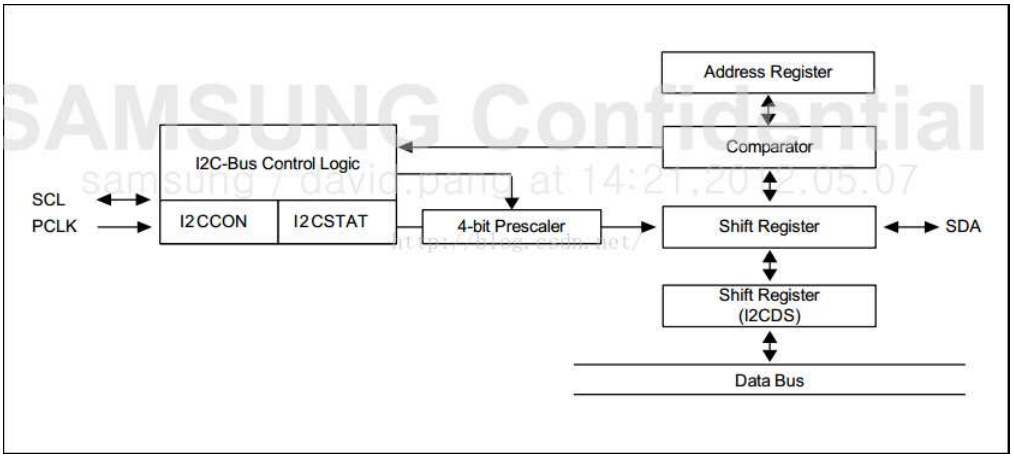
在快速模式中，每秒最多可以传输400k位，即50kB的数据量。

支持主机端发送、接收，从机端发送、接收操作。

支持中断和查询方式。

三、框图

关闭



从上图可以看出，4412提供4个寄存器来完成所有的IIC操作。SDA线上的数据从IICDS寄存器经过移位寄存器发出，或通过移位寄存器传入IICDS寄器；IICADD寄存器中保存4412当做从机时的地址；IICCON、IICSTAT两个寄存器用来控制或标识各种状态，比如选择工作工作模式，发出S信号、P信号，决定是否发出ACK信号，检测是否接收到ACK信号。

四、I2C总线接口操作

针对4412处理器的I2C总线接口，具备4种操作模式： 主机发送模式、主机接收模式、从机发送模式、从机接收模式。下面将描述这些操作模式之间的功能关系：

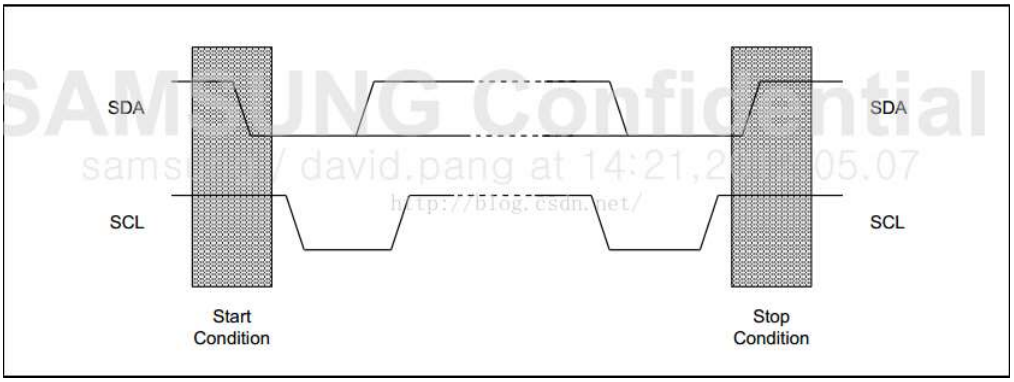
1. 开始和停止条件

当4412的I2C接口空闲时，它往往工作在从机模式。或者说，4412的的i2c接口在SDA线上察觉到一个起始信号之前它应该工作在从机模式。当控制器改变4412的i2c接口的工作模式为主机模式后，SDA线上发起数据传输并且控制器会产生SCL时钟信号。

开始条件通过SDA线进行串行的字节传输，一个停止信号终止数据传输，停止信号是指SCL在高电平器件SDA线有从低到高电平的跳变，主机端产生起始和停止条件。当主、从设备产生一个起始信号后,I2C总线将进入忙状态。这里需要说明的是上述主从设备都有可能作为主机端。

当一个主机发送了一个起始信号后，它也应该发送一个从机地址以通知总线上的从设备。这个地址字节的低7位表示从设备地址，最高位表示传输数据的方向，即主机将要进行读还是写。当最高位是0时，它将发起一个写操作(发送操作)；当最高位是1时，它将发起一个读数据的请求（接收操作）。

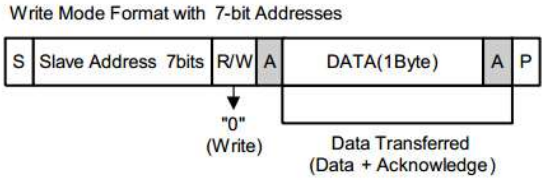
主机端发起一个结束信号以完成传输操作，如果主机端想在总线上继续进行数据的传输，它将发出另外一个起始信号和从设备地址。用这样的方式，它们可以用各种各样的格式进行读写操作。下图为起始和停止信号：

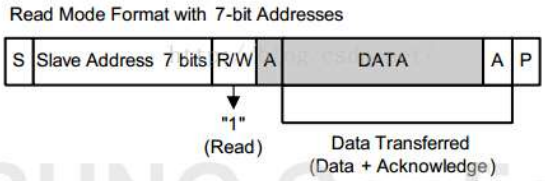


2. 数据传输格式

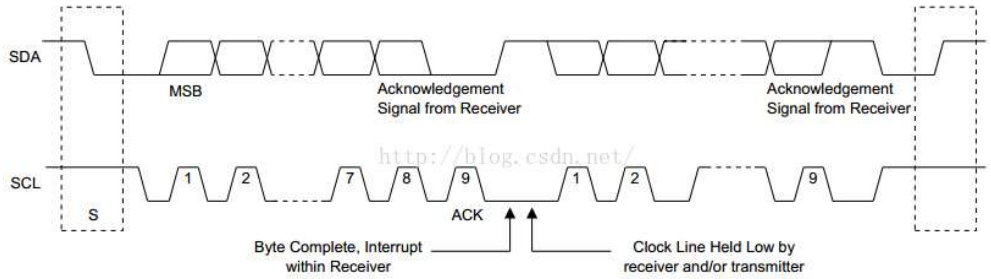
放到SDA线上的所有字节数据的长度应该为8位，在每次传输数据时，对传输数据量没有限制。在起始信号后的第一个数据字节应该包含地址字段，当4412的I2C接口被设置为主模式时，地址字节应该有控制器端发出。在每个字节后，应该有一个应答位。下面的图中将说明数据传输格式：

关闭





NOTES:
1. S: Start, rS: Repeat Start, P: Stop, A: Acknowledge
2. ☐: From Master to Slave, ☐: From Slave to Master



上图中说明，在传输完每个字节数据后，都会有一个应带信号，这个应答信号在第9个时钟周期。具体过程如下(注意下面描述的读写过程都是针对Tiny4412处理器而言，当有具体的I2C设备与4412相连时，数据表示什么需要看具体的I2C设备，4412是不知道数据的含义的)：

写过程：主机发送一个起始信号S→发送从机7位地址和1位方向，方向位表示写→主机释放SDA线方便从机给回应→有从机匹配到地址，拉低SDA线作为ACK→主机重新获得SDA传输8位数据→主机释放SDA线方便从机给回应→从机收到数据拉低SDA线作为ACK告诉主机数据接收成功→主机发出停止信号。

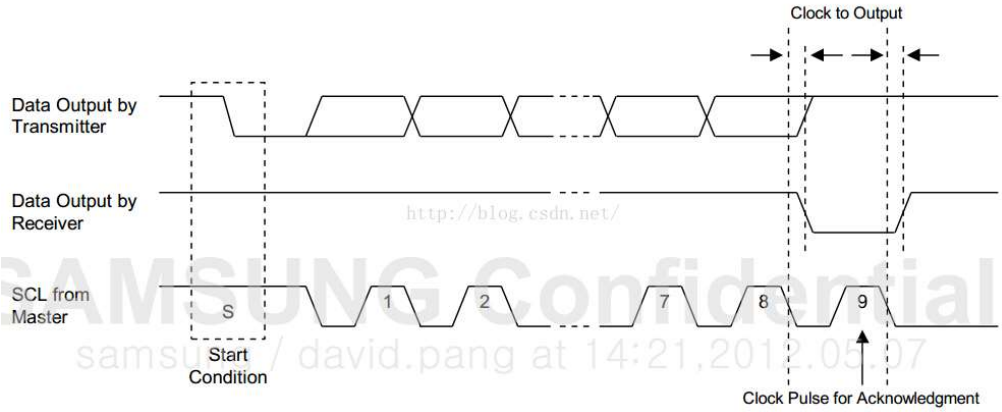
读过程：主机发送一个起始信号S→发送从机7位地址和1位方向，方向位表示读→主机释放SDA线方便从机给回应→有从机匹配到地址，拉低SDA线作为ACK→从机继续占用SDA线，用SDA传输8位数据给主机→从机释放SDA线(拉高)方便主机给回应→主机接收到数据→主机获得SDA线控制并拉低SDA线作为ACK告诉从机数据接收成功→主机发出停止信号。

注意：在具体的I2C通信时，要看I2C设备才能确定读写时序，比如下面即将描述的第七大点中的示例，读写EEPROM中就会说道具体的数据含义，读写过程。

3. 应答信号的传输

为了完成一个字节数据的传输，接收方将发送一个应答位给发送方。应答信号出现在SCL线上的时钟周期中的第九个时钟周期，为了发送或接收1个字节的数据，主机端会产生8个时钟周期，为了传输一个ACK位，主机端需要产生一个时钟脉冲。

ACK时钟脉冲到来之际，发送方会在SDA线上设置高电平以释放SDA线。在ACK时钟脉冲之间，接收方会驱动和保持SDA线为低电平，这发生在第9个时钟脉冲为高电平期间。 应答信号为低电平时，规定为有效应答位（ACK简称应答位），表示接收器已经成功地接收了该字节；应答信号为高电平时，规定为非应答位（NACK），一般表示接收器接收该字节没有成功。 对于反馈有效应答位ACK的要求是，接收器在第9个时钟脉冲之前的低电平期间将SDA线拉低，并且确保在该时钟的高电平期间为稳定的低电平。 如果接收器是主控器，则在它收到最后一个字节后，发送一个NACK信号，以通知被控发送器结束数据发送，并释放SDA线，以便主控接收器发送一个停止信号P。



4. 读写操作

当I2C控制器在发送模式下发送数据后，I2C总线接口将等待直到移位寄存器(I2CDS)接收到一个数据。在往此寄存器写入一个新数据前，SCL线应该保持为低电平，写完数据后，I2C控制器将释放SCL线。当前正在传输的数据传输完成后，4412会捕捉到一个中断，然后cpu将开始往I2CDS寄存器中写入一个新的数据。

当I2C控制器在接收模式下接收到数据后，I2C总线接口将等待知道I2CDS寄存器被读。在读到新数据之前，SCL线会被保持为低电平，读到数据后I2C控制器将释放掉SCL线。一个新数据接收完成后，4412将收到一个中断，cpu收到这个中断请求后，它将从I2CDS寄存器中读取数据。

5. 总线仲裁机制

总线上可能挂接有多个器件，有时会发生两个或多个主器件同时想占用总线的情况，这种情况叫做总线竞争。I2C总线具有多主控能力，可以对发生在SDA线上的总线竞争进行仲裁，其仲裁原则是这样的：当多个主器件同时想占用总线时，如果某个主器件发送高电平，而另一个主器件发送低电平，则发送电平与此时SDA总线电平不符的那个器件将自动关闭其输出级。总线竞争的仲裁是在两个层次上进行的。首先是地址位的比较，如果主器件寻址同一个从器件，则进入数据位的比较，从而确保了竞争仲裁的可靠性。由于是利用I2C总线上的信息进行仲裁，因此不会造成信息的丢失。

6. 终止条件

当一个从接收者不能识别从地址时，它将保持SDA线为高电平。在这样的情况下，主机会产生一个停止信号并且取消数据的传输。当终止传输产生后，主机端接收器会通过取消ACK的产生以告诉从机端发送器结束发送操作。这将在主机端接收器接收到从机端发送器发送的最后一个字节之后发生，为了让主机端产生一个停止条件，从机端发送者将释放SDA线。

7. 配置I2C总线

如果要设置I2C总线中SCL时钟信号的频率，可以在I2CCON寄存器中设置4位分频器的值。I2C总线接口地址值存放在I2C总线地址寄存器(I2CADD)中，默认值未知。

8. 每种模式下的操作流程图

在I2C总线上执行任何的收发Tx/Rx操作前，应该做如下配置：

（1）在I2CADD寄存器中写入从设备地址

（2）设置I2CCON控制寄存器

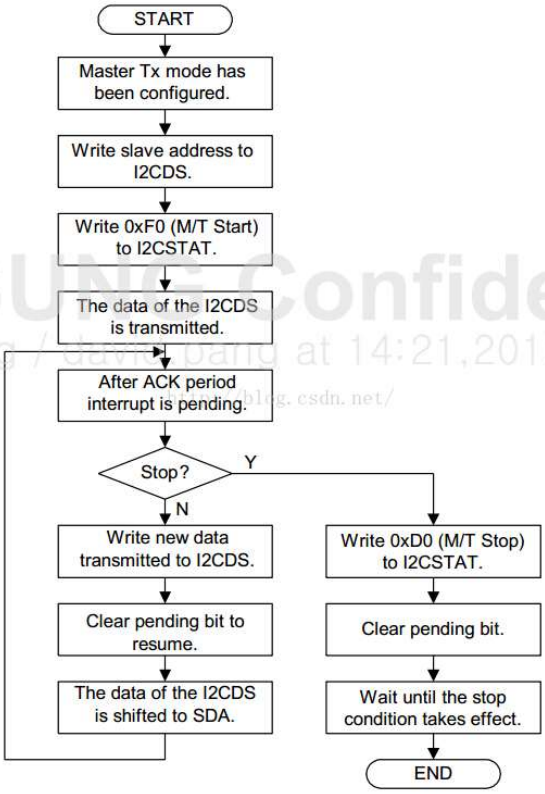
a. 使能中断

b. 定义SCL频率

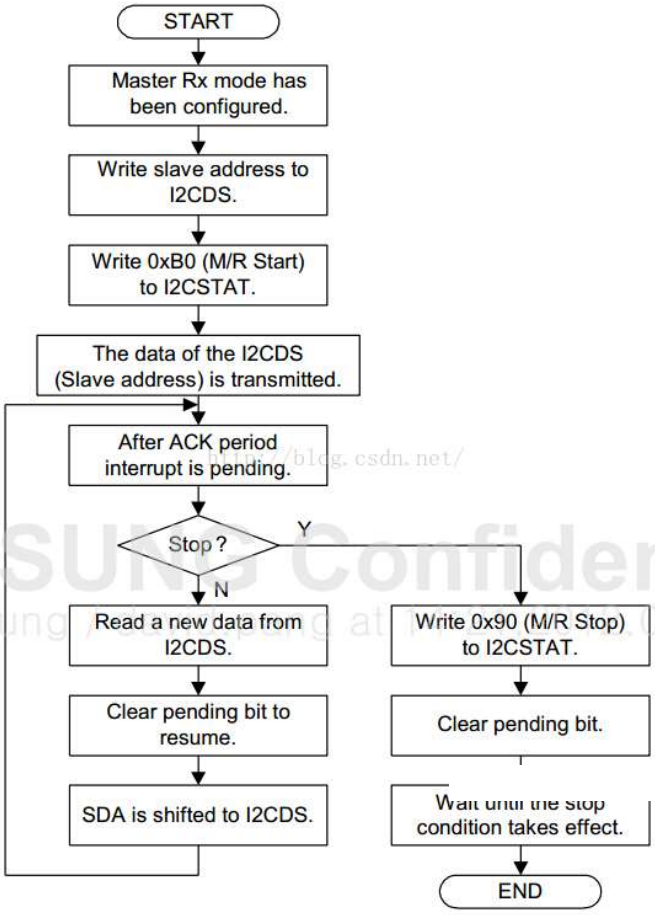
（3）设置I2CSTAT寄存器以使能串行输出

下图为主设备发送模式

关闭

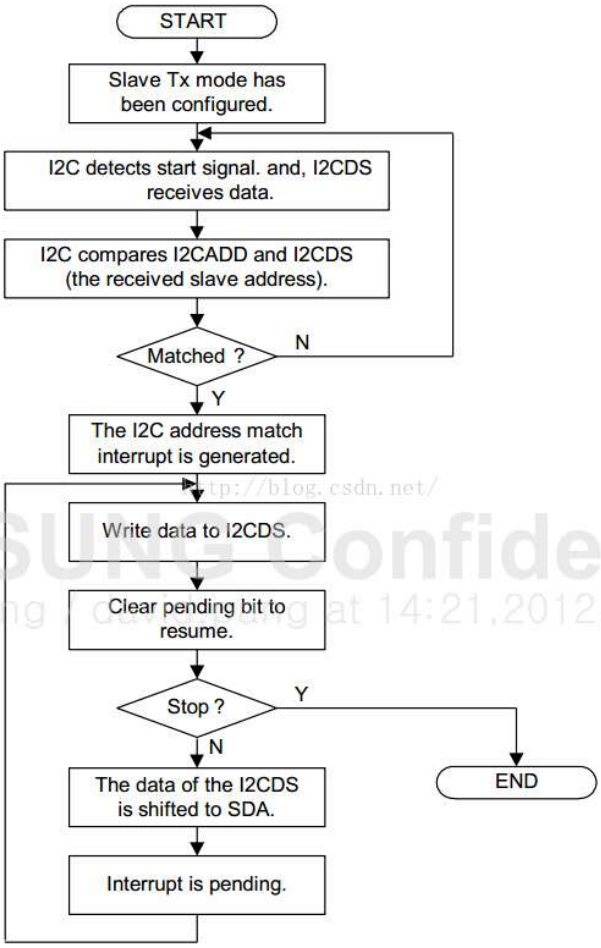


下图为主设备接收模式：

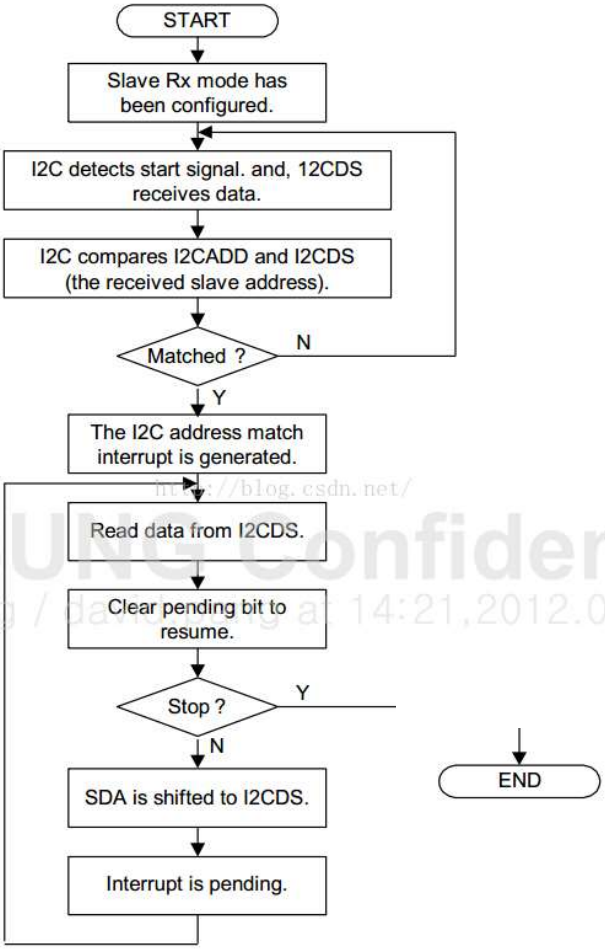


下图为从设备发送模式：

关闭



下图为从设备接收模式：



五、I/O描述

Signal	I/O	Description	Pad	Type
I2C0_SCL	Input/Output	I2C-bus interface0 serial clock line	XI2C0SCL	muxed
I2C0_SDA	Input/Output	I2C-bus interface0 serial data line	XI2C0SDA	muxed
I2C1_SCL	Input/Output	I2C-bus interface1 serial clock line	XI2C1SCL	muxed
I2C1_SDA	Input/Output	I2C-bus interface1 serial data line	XI2C1SDA	muxed
I2C2_SCL	Input/Output	I2C-bus interface2 serial clock line	XuRTSn_1	muxed
I2C2_SDA	Input/Output	I2C-bus interface2 serial data line	XuCTSn_1	muxed
I2C3_SCL	Input/Output	I2C-bus interface3 serial clock line	XuRTSn_2	muxed
I2C3_SDA	Input/Output	I2C-bus interface3 serial data line	XuCTSn_2	muxed
I2C4_SCL	Input/Output	I2C-bus interface4 serial clock line	XspiMOSI_0	muxed
I2C4_SDA	Input/Output	I2C-bus interface4 serial data line	XspiMISO_0	muxed
I2C5_SCL	Input/Output	I2C-bus interface5 serial clock line	XspiMOSI_1	muxed
I2C5_SDA	Input/Output	I2C-bus interface5 serial data line	XspiMISO_1	muxed
I2C6_SCL	Input/Output	I2C-bus interface6 serial clock line	Xi2s2SDO	muxed
I2C6_SDA	Input/Output	I2C-bus interface6 serial data line	Xi2s2SDI	muxed
I2C7_SCL	Input/Output	I2C-bus interface7 serial clock line	XpwmTOUT_3	muxed
I2C7_SDA	Input/Output	I2C-bus interface7 serial data line	XpwmTOUT_2	muxed

NOTE: The I2C bus interface for the HDMI has no external I/O.

六、寄存器描述

1. I2CCONn(n=0-7)寄存器

- Base Address: 0x1386_0000, 0x1387_0000, 0x1388_0000, 0x1389_0000, 0x138A_0000, 0x138B_0000, 0x138C_0000, 0x138D_0000, 0x138E_0000
- Address = Base Address + 0x0000, Reset Value = 0x0X

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	–	Reserved	0
Acknowledge generation (1)	[7]	RW	I2C-bus acknowledge enable bit 0 = Disables 1 = Enables In Tx mode, the I2CSDA is idle in the ACK time. In Rx mode, the I2CSDA is low in the ACK time.	0
Tx clock source selection	[6]	RW	Source clock of I2C-bus transmit clock prescaler selection bit 预分频器 0 = I2CCLK = fPCLK/16 1 = I2CCLK = fPCLK/512	0
Tx/Rx Interrupt (5)	[5]	RW	I2C-bus Tx/Rx interrupt enable/ disable bit 0 = Disables 1 = Enables	0
Interrupt pending flag (2), (3)	[4]	S	I2C-bus Tx/Rx interrupt pending flag You cannot write this bit to 1. If you read this bit as 1, the I2CSCL is tied to Low and the I2C is stopped. To resume the operation, write this bit as 0. 0 = 1) No interrupt is pending (If Read). 2) Clears pending condition and resumes the operation (If Write). 发生 1 = 1) Interrupt is pending (If Read) 2) N/A (If Write)	0
Transmit clock value (4)	[3:0]	RW	I2C-bus transmit clock prescaler 4-bit prescaler value determines the I2C-bus transmit clock frequency according to the formula given here: 公式 Tx clock = I2CCLK/(I2CCON[3:0] + 1).	–

注意：

（1）当EEPROM连接到I2C总线上时，为了在接收模式中产生停止信号，在读最后一个数据之前ACK将被禁止产生。

（2）一个I2C中断发生在以下三种情况：

当发出地址信息或接收到一个从机地址并吻合时，当总线总裁失败时，当发送/接收完一个字节的数据(包括ACK响应位)时。当发出地址信息或接收到一个从机地址并吻合时产生中断，在中断处理函数中要准备发送或者接收数据，即读取或设备IICDS寄存器，或者发出P信号。当总线总裁失败时产生中断，在中断处理函数中决定时候延时后再次竞争总线等。当发送/接收完一个字节的数据(包括ACK响应位)时产生中断，在中断处理函数中要准备下次要发送或者接收数据，即读取或设置IICDS寄存器，或者发出P信号。

（3）基于SDA、SCL线上事件特性的考虑，要发送数据时，先将数据写入I2CDS寄存器，然后再清除中断。清除中断，即向I2CCON[4]写入0，也就是将SCL线拉高，此时产生一个上升沿，将移位寄存器中的数据发送到SDA线。至于先将数据写入I2CDS寄存器，然后再清除中断，可能是数据稳定需要一个时间。

（4）I2CCON[6] determines I2CCLK. Tx clock can vary by SCL transition time.When I2CCON[6] = 0, I2CCON[3:0] = 0x0 or 0x1 is not available.

（5）如果IICCON[5]==0,IICCON[4]将不能正常工作。所以，即使不使用IIC中断，也要将IICCON[5]设为1。

2. I2CSTATn(n=0-7)寄存器

关闭

- Base Address: 0x1386_0000, 0x1387_0000, 0x1388_0000, 0x1389_0000, 0x138A_0000, 0x138B_0000, 0x138C_0000, 0x138D_0000, 0x138E_0000
- Address = Base Address + 0x0004, Reset Value = 0x00

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	–	Reserved	0
Mode selection	[7:6]	RWX	I2C-bus Master/Slave Tx/Rx mode select bits 00 = Slave receive mode 01 = Slave transmit mode 10 = Master receive mode 11 = Master transmit mode	00
Busy signal status/START STOP condition	[5]	S	I2C-bus busy signal status bit 0 = (Read) Not busy (If Read) (write) STOP signal generation 1 = (Read) Busy (If Read) (write) START signal generation. Transfers the data in I2CDS automatically just after the start signal.	0
Serial output	[4]	S	I2C-bus data output enable/ disable bit 0 = Disables Rx/Tx 1 = Enables Rx/Tx	0
Arbitration status flag	[3]	RO	I2C-bus arbitration procedure status flag bit 0 = Bus arbitration successful 1 = Bus arbitration fails during serial I/O	0
Address-as-slave status flag	[2]	RO	I2C-bus address-as-slave status flag bit 0 = Clears when it detects START/STOP condition 1 = Receives slave address that matches the address value in the I2CADD	0
Address zero status flag	[1]	RO	I2C-bus address zero status flag bit 0 = Clears when it detects START/ STOP condition 1 = Received slave address is 00000000b	0
Last-received bit status flag	[0]	RO	I2C-bus last-received bit status flag bit 0 = Last-received bit is set to 0 (receives ACK). 1 = Last-received bit is set to 1 (does not receive ACK).	0

3. I2CADDn(n=0-7)寄存器

- Base Address: 0x1386_0000, 0x1387_0000, 0x1388_0000, 0x1389_0000, 0x138A_0000, 0x138B_0000, 0x138C_0000, 0x138D_0000, 0x138E_0000
- Address = Base Address + 0x0008, Reset Value = 0xXX

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	–	Reserved	0
Slave address	[7:0]	RWX	7-bit slave address, latched from the I2C-bus. When serial output enable = 0 in the I2CSTAT, I2CADD is write-enabled. The I2CADD value is Read any time, regardless of the current serial output enable bit (I2CSTAT) setting. Slave address: [7:1] Not mapped: [0]	–

用到IICADD寄存器的位[7:1]，表示从机地址。IICADD寄存器在串行输出使能位IICSTAT[4]为0时，才可以写入；在任何时候都可以读出。IICSTAT[4]为0时，禁止接收/发送功能，即此时SCL线被拉低。

4. I2CDSn(n=0-7)寄存器

- Base Address: 0x1386_0000, 0x1387_0000, 0x1388_0000, 0x1389_0000, 0x138A_0000, 0x138B_0000, 0x138C_0000, 0x138D_0000, 0x138E_0000
- Address = Base Address + 0x000C, Reset Value = 0xXX

Name	Bit	Type	Description	Reset Value
RSVD	[31:8]	–	Reserved	0
Data shift	[7:0]	RWX	8-bit data shift register for I2C-bus Tx/Rx operation. When serial output enable = 1 in the I2CSTAT, I2CDS is write-enabled. The I2CDS value is Read any time, regardless of the current serial output enable bit (I2CSTAT) setting.	–

用到IICDS寄存器的位[7:0]，其中保存的是要发送或已经接收到的数据。IICDS寄存器在串行输出使能位IICSTAT[4]为1时才可以写入；在任何时候都可以读出。IICSTAT[4]为1时，使能接收/发送功能，也就是

关闭

5. I2CLCn(n=0-7)寄存器

- Base Address: 0x1386_0000, 0x1387_0000, 0x1388_0000, 0x1389_0000, 0x138A_0000, 0x138B_0000, 0x138C_0000, 0x138D_0000, 0x138E_0000
- Address = Base Address + 0x0010, Reset Value = 0x00

Name	Bit	Type	Description	Reset Value
RSVD	[31:27]	–	Reserved	0
Filter enable	[2]	RW	I2C-bus filter enable bit When SDA port is operating as input, set this bit to High. This filter prevents error caused by glitch between two PCLK clocks. 0 = Disables Filter 1 = Enables Filter	0
SDA output delay	[1:0]	RW	I2C-bus SDA line delay length selection bits The I2C controller delays the SDA line by following clock cycle: 00 = 0 clock 01 = 5 clocks 10 = 10 clocks 11 = 15 clocks	00

七、实例(EEPROM)

1、I2C原理：

（1）IIC只有2条线，SDA(数据线)/SCL(时钟线)，分为主机(IIC控制器)和从机(EEPROM)，即主从结构，所有的数据传输都是由主机发起，从机只能接收，2条线上可以挂很多 从机设备。

（2）主机通过向从机发地址(每个I2C设备都有一个嵌入到芯片里的设备地址，并且每种I2C设备地址格式还不一样，比如EEPROM的设备地址就由前面地址和后面格式构成， 后面格式由根据硬件上的连线决定，Tiny4412上接的EEPROM的A0-2都被拉低，即地址为1010000的七位地址),哪个从机响应了,就与哪个从机通信，具体是读还是写根据 第8位去区别。

（3）当SCL/SDA都为高电平时,拉低SDA作为起始信号，SCL为高，拉高SDA做为结束信号，从机在收到8位数据后,在第9个时钟周期拉底SDA作为ACK应答信号。

（4）scl为低时可以传送数据,传送完后SCL会被拉高,在scl上升沿开始传数据。scl为高时要保持sda数据稳定。

（5）发送完数据,读完数据,发送完stop信号，都要delay一会，等设备反应。

（6）传送数据按芯片手册的timing走,先传地址,先传数据,每传一个8位数据,从机要发送一个ACK应答。

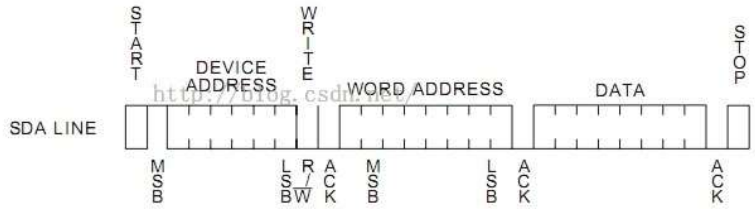
（7）我们编程是以主机的角色,从机自己会拉高拉底scl/sda,主机也不用管怎么拉高拉低SCL/SDA,只需设置IIC控制器的寄存器就可以按timing传数据，如果没有I2C控制器，需 要用GPIO模拟时才需要自行控制scl/sda线。

（8）Tiny4412有一个直接连接CPU IIC0 信号引脚的EEPROM芯片AT24C08,主要是为了测试IIC总线用，我们看下往这个设备读写数据应该怎么操作。

2、写时序

在第四大点中描述了基于4412处理器I2C操作中读写时序，但是具体数据含义还得看具体I2C设备，比如这里的EEPROM。

Figure 2. Byte Write



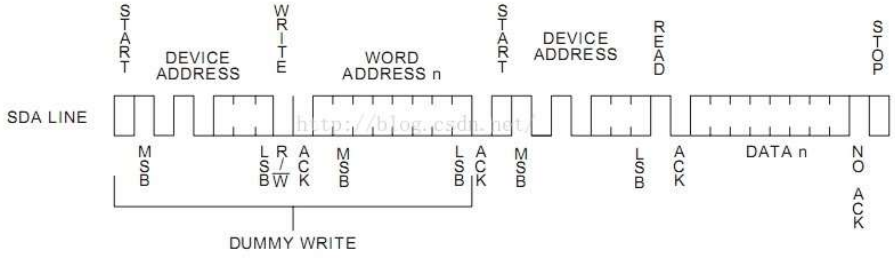
当4412要往EEPROM里写一个字节数据时，首先发start信号，然后写从机的8位设备地址(可以从AT24C08的手册找到),最后1位用来标识接下来是要写还是读从机设备，然后拉高SDA。从机收到地址后如果跟自己地址匹配，就发一个ACK应答给主机，即会拉低SDA线。

从机里的数据是按addr(EEPROM芯片内部的内存地址)来index的，要往哪块地址写数据，4412先发addr给从机，从机准备好后发ACK，然后主机就发8位data给从机，从机应答 然后主机发stop信号结束 主机在发送8位数据的8个clk里，SDA由主机驱动，第9个clk里，SDA由从机驱动。

关闭

3、读时序

Figure 5. Random Read



当4412要在EEPROM中读一个字节时，先发start信号，发从机地址,最后1位标识write(写的这个地址即告诉从机后面将要在EEPROM里哪个地方读取数据)，拉高SDA方便从机回应，有从机匹配到从机地址后拉低SDA线回应主机；主机发要读哪块地址数据的addr数据，从机收到主机发来的数据后给出回应；主机重新发start信号，发从机地址，最后1位标识read，然后4412读数据，读完数据主机(4412)一般不会应答(no ack),主机发stop信号结束。

在主机读数据的那8个clk里，SDA由从机驱动，第9个CLK，SDA由主机驱动。总的来说，谁收数据，谁就要给出应答信号(拉低SDA)，因为要告诉发送方，数据已经收到。

上一篇 二分法与printk()
下一篇 杨毅：不够优秀就不要腆着脸继续占便宜

相关文章推荐

- Exynos4412 裸机开发 —— IIC总线
- 实验十三 I2C总线协议控制器实验
- 51单片机学习笔记,模拟iic总线连续读写24c02存…
- Exynos4412 IIC总线驱动开发（二）—— IIC 驱动…
- IIC总线
- Exynos4412 IIC总线驱动开发（一）—— IIC 基础…
- I2S总线协议
- Exynos4412 IIC 总线驱动开发相关问题总结
- 解决通用串行总线控制器里全是叹号问题
- Linux内核---33.IIC总线控制器驱动分析