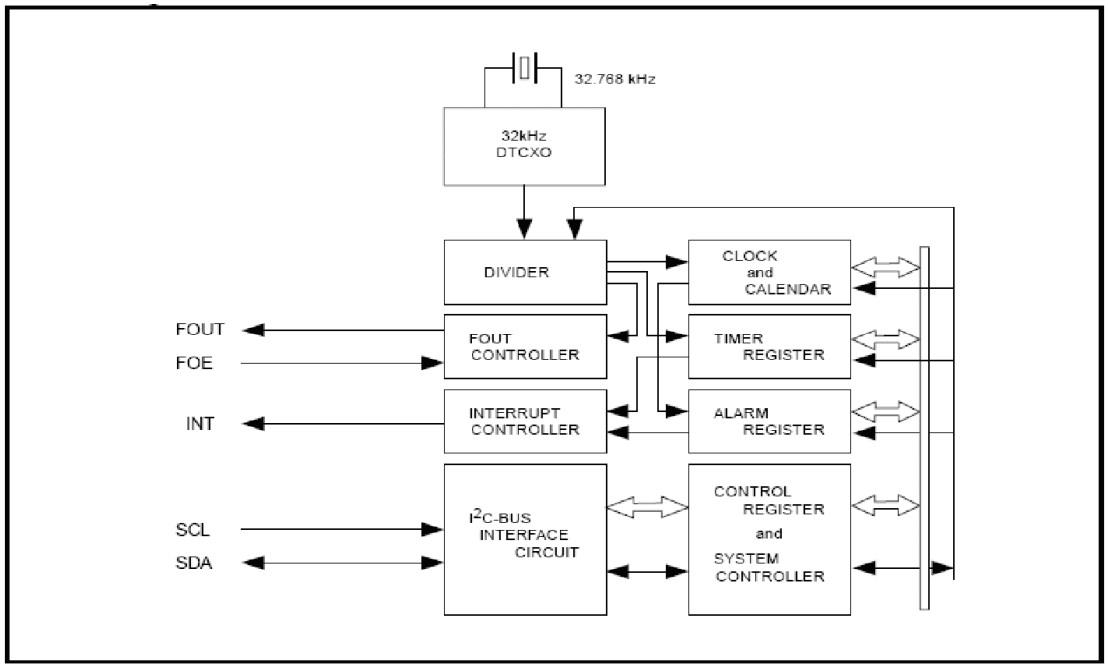
XCRSE8025TIAD
使用说明

XCRSE8025TIAD 使用说明概要

1 XCRSE8025TIAD 的特点：

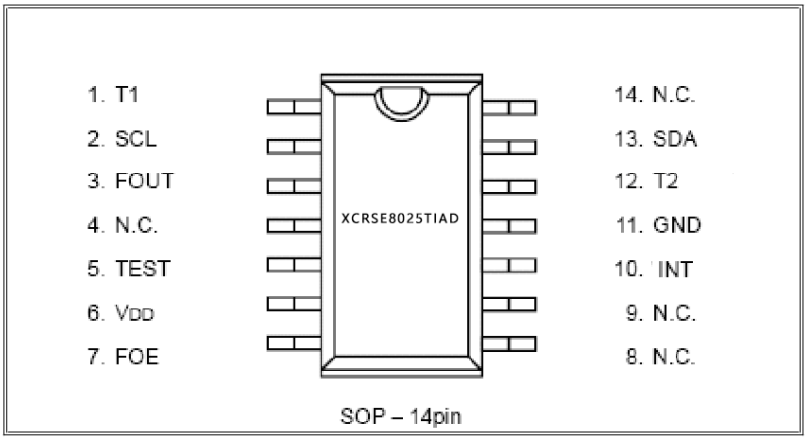
1. 内置高稳定度的32.768KHz 的DTCXO (数字温度补偿晶体振荡器)
2. 支持I2C 总线的高速模式（400K）。
3. 定时报警功能（可设定：天，日期，小时，分钟）
4. 固定周期定时中断功能。
5. 时间更新中断功能。
6. 32．768KHz频率输出（具有使能OE功能）
7. 闰年自动调整功能。（2000 到2099）
8. 宽范围接口电压：2.0V 到 5.5V
9. 宽范围的时间保持电压：1.8V 到 5.5V
10.低电流功耗：1uA/3V (Typ.)

2 XCRSE8025TIAD 结构框图：



3 XCRSE8025TIAD 管脚定义：

3.1 管脚封装：



3.1 管脚功能定义：

管脚名称	I/O	功能
1：T1	In	*工厂测试用（不用额外连接）
2：SCL	In	I2C总线通讯的串行时钟输入端
3：FOUT	Out	这是个C-MOS输出引脚，可通过FOE进行控制。 当FOE='H'，该引脚输出一个32.768KHz 信号 当输出停止时，FOUT引脚="H-Z"（高阻状态）
4/8/9/14：NC	-	这些引脚没有连接内部IC
5：TEST	In	*工厂测试用（不用额外连接）
6：VDD	-	电源正端
7：FOE	In	该引脚用来控制FOUT的输出模式，当为高电平时 FOUT输出使能。
10：INT	Out	该引脚用于输出：报警信号，时钟信号，时间更新信号， 以及其它信号。该引脚为开漏输出引脚。
11：GND	-	电源接地端
12：T2	-	*工厂测试用，应用时可以悬空或接VDD
13：SDA	I/O	I2C 总线通讯，数据传输端。该引脚为N-ch 开漏输出， 所以一定要连接到一个有上拉电阻的相关信号线上。

注意：确认在VDD 和GND 之间连接一个至少0.1uF 的旁路电容。

4 绝对电气指标：

项目	符号	条件	数值	单位
电源电压	VDD	VDD 和 GND之间	-0.3 to +7.0	V
输入电压（1）	Vin1	FOE引脚	* GND-0.3 to VDD+0.3	V
输入电压（2）	Vin2	SCL，SDA 引脚	GND-0.3 to +7.0	V
输入电压（3）	Vout1	FOUT引脚	GND-0.3 to VDD+0.3	V
输入电压（4）	Vout2	SDA，INT引脚	* GND-0.3 to +7.0	V
存储温度	T-STG	分散存放，无包装	-55 to +125	℃

5 推荐操作条件：

项目	符号	条件	Min.	Typ.	Max.	单位
运行电压	VDD	接口电压	2.0	3.0	5.5	V
温度补偿电压	V-TEM	温度补偿电压	2.4	3.0	5.5	V
时钟供电电压	V-CLK		1.6	3.0	5.5	V
操作温度	T-OPR		-40	+25	+85	℃

6 频率特性：

频率稳定度：

△f/f= ±3.8ppm @ Ta= 0 to +50℃, VDD=3.0V

相当于：60*60*24*3.8ppm = 0.328 （s/day）

△f/f= ±5.0ppm @ Ta= -40 to +85℃, VDD=3.0V

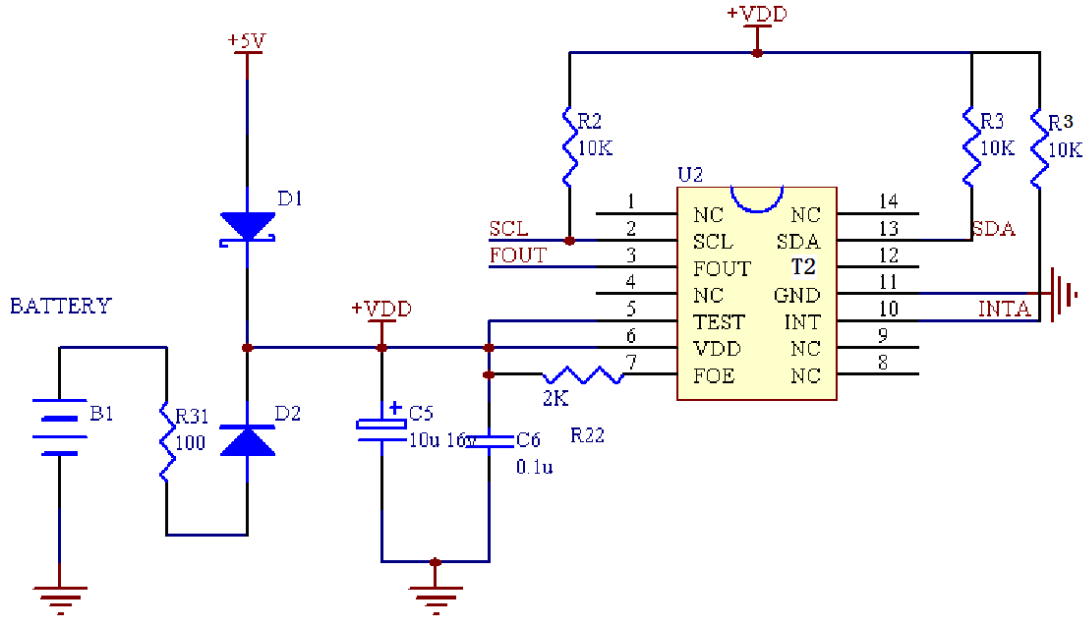
相当于：60*60*24*5.0ppm = 0.432 （s/day）

7 电气特性：

项目	符号	条件		Min	Typ	Max	单位
电流功耗1	IDD1	Fscl=0 Hz INT=VDD;FOE=GND	VDD=5V		1.8	3.4	uA
电流功耗2	IDD2		VDD=3V		1	2.8	
电流功耗3	IDD3	Fscl=0 Hz INT,FOE=VDD FOUT:32.768K, CL=0pF	VDD=5V		4.3	7.5	uA
电流功耗4	IDD4		VDD=3V		2.6	5.0	
电流功耗5	IDD5	Fscl=0 Hz INT,FOE=VDD FOUT:32.768K, CL=30pF	VDD=5V		10	22.0	uA
电流功耗6	IDD6		VDD=3V		7	14.0	
高电平输入电压	V_IH1	FOE 引脚		0.7*VDD		VDD+0.3	V
	V_IH2	SCL & SDA 引脚		0.7*VDD		5.5	V
低电平输入电压	VIL	输入引脚		GND-0.3		0.3*VDD	V
高电平输出电压	VOH1	FOUT 引脚	VDD=5V，IOH=-1 mA	4.5		5.0	V
	VOH2		VDD=3V，IOH=-1 mA	2.2		3.0	
	VOH3		VDD=3V，IOH=-100 uA	2.9		3.0	
低电平输出电压	VOL1	FOUT 引脚	VDD=5V，IOL= 1 mA	GND		GND+0.5	V
	VOL2		VDD=3V，IOL= 1 mA	GND		GND+0.8	
	VOL3		VDD=3V，IOL= 100 uA	GND		GND+0.1	
	VOL4	/INT 引脚	VDD=5V，IOL= 1 mA	GND		GND+0.25	V
	VOL5		VDD=3V，IOL= 1 mA	GND		GND+0.4	
	VOL6		VDD≥2V，IOL= 3 mA	GND		GND+0.4	
输入漏电流	ILK	输入引脚，VIN = VDD 或 GND		-0.5		0.5	uA
输出漏电流	IOZ	/INT，SDA，FOUT， VIN=VDD 或 GND		-0.5		0.5	uA

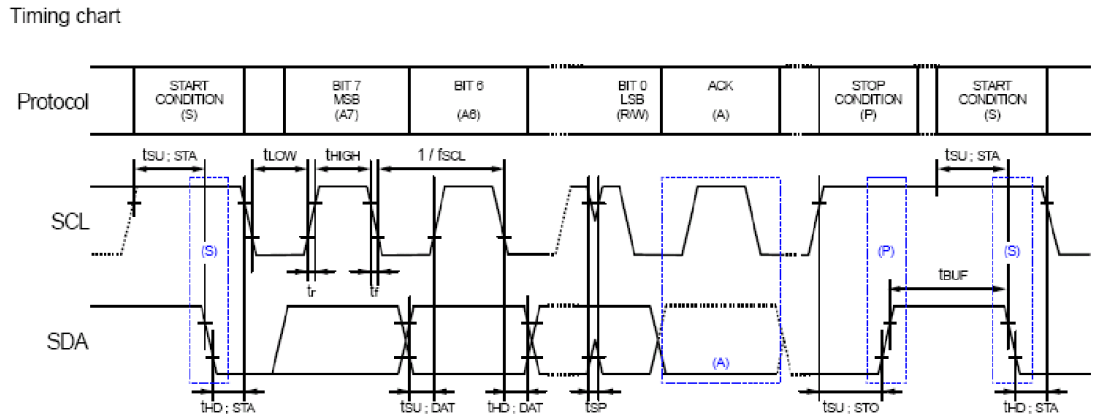
* 除非特别指定，GND=0V，VDD=2.0 V to 5.5V， Ta=-40 to +85 ℃

8 推荐电路：



注意：若不需要FOUT 输出，FOE 引脚可悬空或接地。

9 I2C 总线协议（时序图）



注意：当访问该器件的时候，所有的通讯从传输开始条件到传输结束条件为止，所有的操作必须在0.95秒内完成。 如果这样的通讯需要0.95s 或更长时间，那么I2C 总线接口将由内部总线时间溢出功能复位。

10 XCRSE8025TIAD 操作模式：

1）实时时钟模式

该功能被用来设定和读取年，月，日，星期，时，分，秒 时间信息。年份为后两位数字表示，任何可以被 4 整除的年份被当成闰年处理。（2000 年到 2099 年）

2）固定周期的中断发生功能：

固定周期定时中断发生功能可以产生一个固定周期的中断事件，固定周期可在244.14uS 到4095 分钟之间的 任意时间设定。

3)定时更新中断功能：

该功能可以根据内部时钟的定时设定，每秒或每分钟产生一个中断事件。当中断事件产生，UF 标志位的值变成 1 同时/INT 引脚变成低电平表示一个中断事件的产生。

4）闹钟中断功能：

该功能可以根据报警设定来产生一个中断。

5）32.768K Hz 时钟输出：

可以通过FOUT 引脚来输出一个32.768kHz 频率的时钟信号，该功能可以通过FOE 引脚控制。

6）和CPU的接口功能：

数据的读写都是通过I2C 总线接口的方式来完成。

11 寄存器简介：

地址	功能	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	秒	0	40	20	10	8	4	2	1
1	分	0	40	20	10	8	4	2	1
2	时	0	0	20	10	8	4	2	1
3	周	0	6	5	4	3	2	1	0
4	天	0	0	20	10	8	4	2	1
5	月	0	0	0	10	8	4	2	1
6	年	80	40	20	10	8	4	2	1
7	RAM	.	.	.	.	.	.	.	.
8	Min alarm	AE	40	20	10	8	4	2	1
9	Hour alarm	AE	.	20	10	8	4	2	1
A	Week alarm	AE	6	5	4	3	2	1	0
	Day alarm		.	20	10	8	4	2	1
B	Timer couner0	128	64	32	16	8	4	2	1
C	Timer couner1	.	.	.	.	2048	1024	512	256
D	Ext register	TEST	WADA	USEL	TE	FSEL1	FSEL0	TEL1	TEL0
E	Flag register	0	0	UF	TF	AF	0	VLF	VDET
F	Contr registor	CSEL1	CSEL0	UIE	TIE	AIE	0	0	RESET

注意：当内部上电复位或当读到 VLF 位的值为1 的时候，需要对所有的寄存器重新初始化。
确保输入正确的数据，如果数据或时间不正确，那么时钟操作的结果将不能得到保证。
*1）在内部上电期间，TEST 位复位为‘0’VLF 位复位为‘1’
此时所有寄存器的值是不确定的。
*2）只有‘0’能被写入到UF，TF，AF，VLF，VDET 这些寄存器的位里面。
*3）任何标有‘o’的位在初始化以后应该被当作‘0’来使用。
*4）任何标有‘· ’的位可以读写任意值。
*5）TEST 位被用作工厂测试用，该位在写操作的时候一定确保是‘0’。

11　寄存器详解：

11.1 控制寄存器 F

地址	功能	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
F	Contr registor	CSEL1 默认	CSEL0	UIE	TIE	AIE	O	O	RESET

该寄存器用来控制/INT 引脚的中断输出以及时钟的启动/停止的状态和日历的操作。

1）CSEL0，1（补偿间隔选择 0，1）

这两个控制位被用来设定温度补偿的时间间隔。

SCEL0,1	SCEL1	SCEL0	补偿间隔
写/读	0	0	0.5s
	0	1	2.0s(默认)
	1	0	10s
	1	1	30s

2）UIE（更新中断使能位）

写入一个‘1’到该位，当一个中断事件产生时，就会有一个中断信号产生（/INT 的状态
会从H-Z 高阻状态变为低电平。）
写入一个‘0’到该位，当一个中断事件发生时，不会有中断信号产生。

UIE	DATE	描述
W/R	0	当中断周期到达，不能输出中断信号。INT 保持高阻。
	1	当时间更新事件产生，中断信号产生。INT 从高阻变为低，7.813ms 后恢复为高阻。

注意：在中断发生以后，/INT 的状态在7.8ms 或500ms 后自动清除（通过USEL 位选择）

3）TIE（定时中断使能位）

写入一个‘1’到该位，当一个中断事件产生时，就会有一个中断信号产生（/INT 的

状态会从H-Z 高阻状态变为低电平。）
写入一个‘0’到该位，当一个中断事件发生时，不会有中断信号产生。

TIE（中断使能信号）	DATE	
W/R	0	当一个固定周期定时中断发生时，不会有中断信号产生。（/INT 的状态从低变成 H-Z）
	1	当一个固定周期定时中断产生时，就会有一个中断信号产生（/INT 的状态会从H-Z 高阻状态变为低电平。）

4）RESET 位

写入一个‘1’到该位，并且该值维持1 秒以上，可以停止计数器操作以及对RTC模块内部计数器值进行复位。
在0.95 秒总线时间溢出功能运行时，如果接收到一个停止STOP 条件或重复开始RE-START 条件，那么停止状态自动取消（RESET 位的值从‘1’变成‘0’）。

11.2 标志寄存器 E

地址	功能	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
E	Flag register	0	0	UF	TF	AF	0	VLF	VDET

该寄存器用来监测各种中断时间以及内部数据的相关问题。
UF，TF，AF，分别是时间更新中断，固定周期定时中断，闹钟中断的中断标志位。

1）VLF（电压低标志位）

该标志位用来指示时钟运行或内部数据的保持状态。当数据丢失的情况发生，该位的值由‘0’变成‘1’。一旦该位变成‘1’，该值将维持到一个‘0’被写入该位。

VLF	数值	功能
写	0	VLF 位被清零,准备下次的状态检测
	1	写入‘1’后，该位无效
读	0	没有产生数据丢失的情况
	1	检测到数据丢失，所有寄存器必须重新配置。（该位必须软件清零）

2）VDET（电压检测标志位）

该位用来检测温度补偿的工作状态，当温度补偿停止工作时该位从‘0’变成‘1’。（该位必须软件清零）

11.3 扩展寄存器 D

地址	功能	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
E	EXT register	TEST	WADA	USEL	TE	FSEL1	FSEL0	TSEL1	TSEL0

该寄存器用来说明闹钟功能或定时更新中断功能以及用来选择或设定等操作。
例如
FSEL0，1 用来选择FOUT 的输出频率。
TSEL0，1 用来设定固定周期的内部时钟源。

控制秒脉冲输出的控制位：

USEL	数值	更新中断	自动复位时间tRTN
读/写	0	秒更新	500ms （秒脉冲设定）
	1	分更新	7.813ms

FSEL0，1	FSEL1	FSEL0	FOUT 输出频率
W/R	0	0	32768 Hz
	0	1	1024 Hz
	1	0	1 Hz
	1	1	32768 Hz

TSEL0，1	TSEL1	TSEL0	时钟源
W/R	0	0	4096 Hz
	0	1	64 Hz
	1	0	1s
	1	1	1 min

11.4 时钟计数器（寄存器 0 到 2）

地址	功能	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	秒	0	40	20	10	8	4	2	1
1	分	0	40	20	10	8	4	2	1
2	时	0	0	20	10	8	4	2	1

分别记录时钟的-时，分，秒
所有的数据格式都为BCD 码，例如秒寄存器的值为 ‘0101 1001’ 实际表示为5秒。
小时计数器从‘00’‘01’一直到‘23’，然后重新从‘00’开始，为24 小时进制。

11.5 星期寄存器 REG-3

地址	功能	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3	WEEK	0	6	5	4	3	2	1	0

该寄存器用来记录星期的信息：第0 位到第6 位用来表示星期日，星期一....到星期六。
数据格式不再是BCD 编码，而是分别用一位来表示不同的日期。

具体见下表：

WEEK	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	DAY	DATA
Write/Read	0	0	0	0	0	0	0	1	Sunday	01h
	0	0	0	0	0	0	1	0	Monday	02h
	0	0	0	0	0	1	0	0	Tuesday	04h
	0	0	0	0	1	0	0	0	Wedday	08h
	0	0	0	1	0	0	0	0	Thursday	10h
	0	0	1	0	0	0	0	0	Friday	20h
	0	1	0	0	0	0	0	0	Saturday	40h

特别注意：不要同时设定多位为‘1’的情况，因为任何错误的设定都会导致正常操作的混乱。

11.6 日历寄存器（4 到 6）

WEEK	功能	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
4	DAY	0	0	20	10	8	4	2	1

5	MONTH	0	0	0	10	8	4	2	1
6	YEAR	80	40	20	10	8	4	2	1

具有自动日历调节的功能，作用范围2001 年1 月1 日到2099 年12 月31 日。数据格式为BCD 编码。

注意：设定不存在的日期数据将导致计数器不能正常操作。

另外，日历对应的星期系统不能自动调整，可以通过一定的算法来实现，下面介绍一种常用的公式：

A：最常见的公式：

W = [Y-1] + [(Y-1)/4] - [(Y-1)/100] + [(Y-1)/400] + D

Y 是年份数，D 是这一天在这一年中的累积天数，也就是这一天在这一年中是第几天。

B：最好用的是蔡勒公式：

W = [C/4] - 2C + y + [y/4] + [13 * (M+1) / 5] + d - 1

C 是世纪数减一，y 是年份后两位，M 是月份，d 是日数。1 月和2 月要按上一年的13 月和

14 月来算，这时C 和y 均按上一年取值。

两个公式中的[...]均指只取计算结果的整数部分。算出来的W 除以7，余数是几就是星期

几。如果余数是0，则为星期日。

12 占空比 50% 的秒脉冲输出：

1、设定XCRSE8025TIAD工作在时间更新中断模式.

2、使用/INT引脚输出，INT 输出状态：1：高阻态，0：低电平。

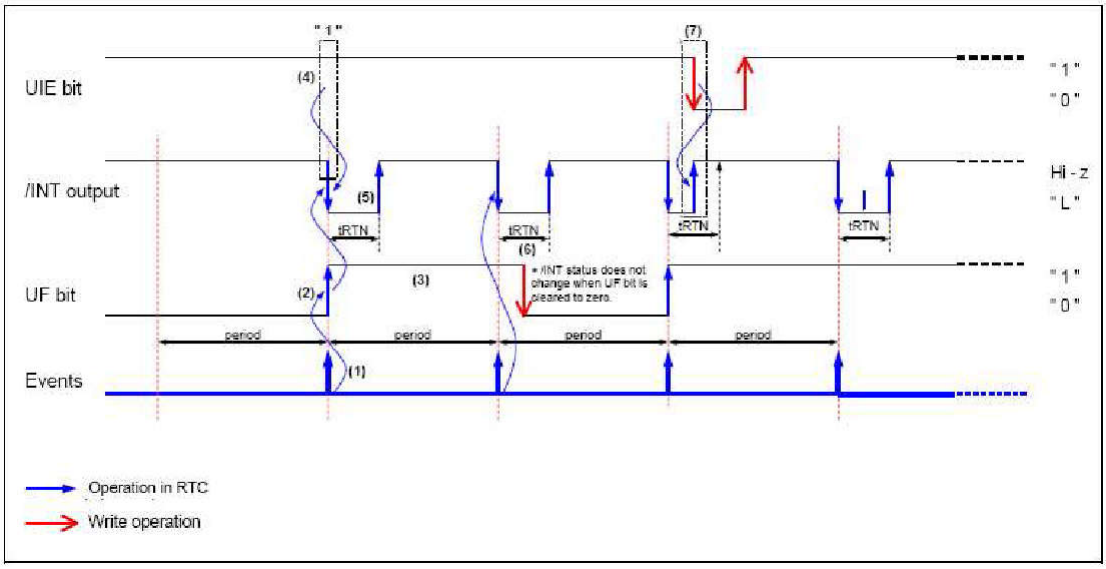
3、相关寄存器

addres	功能	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
D	ext	TEST	WADA	USEL	TE	FSEL1	FSEL0	TEL1	TEL0
E	flag	O	O	UF	TF	AF	O	VLF	VDET
F	con	CSEL1	CSEL0	UIE	TIE	AIE	O	O	RESET

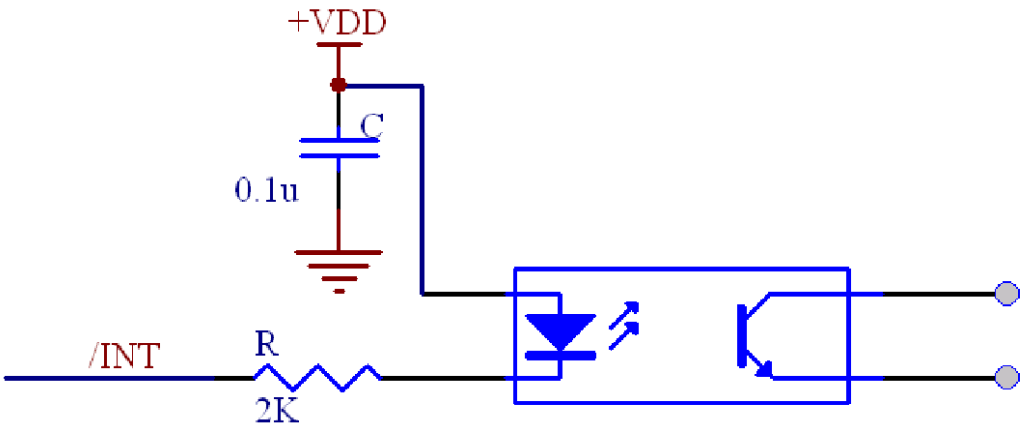
4、寄存器配置数值

addres	值	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
D	0X00	0	0	0	0	0	0	0	0
E	0X00	0	0	0	0	0	0	0	0
F	0X60	0	1	1	0	0	0	0	0

时序图：



秒脉冲输出外接光藕隔离，共检测时间精度用。

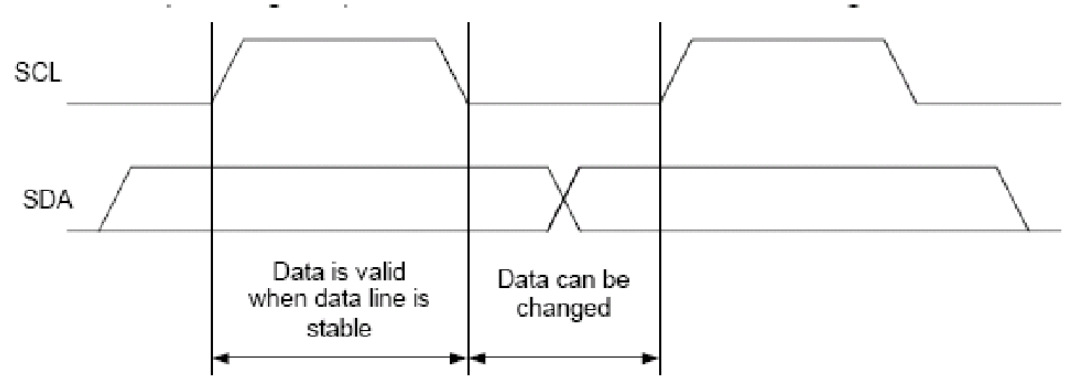


13 I2C-Bus 操作

数据传输注意事项：

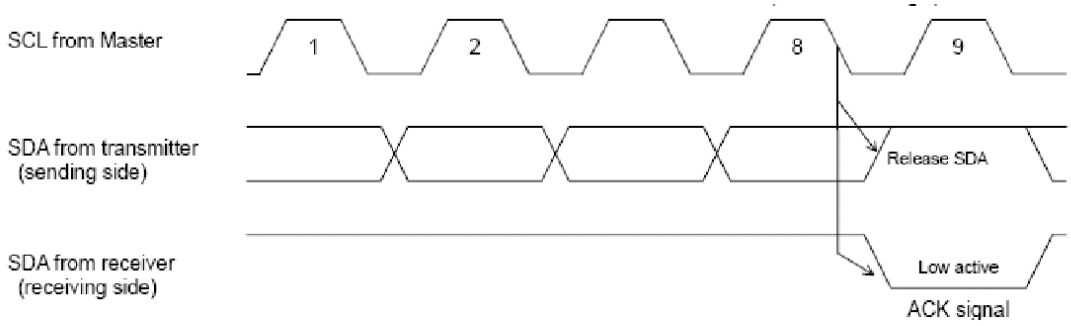
当开始START条件出现，数据传输就以8-bit（一个字节）为单位进行。在开始和停止条件之间的数据个数没有严格限制，但是整个传输时间一定不能大于0.95s。

无论读操作或写操作寄存器的地址都会自动增加，在地址0FH 之后地址自动变成地址00H。数据传输的发送端在时钟线SCL为低的情况进行改变，而接收端在SCL为高的情况下读取稳定有效的数据（ SDA 状态）



数据确认响应（ACK信号）

当传输数据的时候，接收器会在接收到一个8-bit数据段时产生一个确认响应信号（ACK信号，低有效）。如果接收器没有ACK信号产生，表示该通讯过程没有实现。（这种情况不包括主机故意不产生ACK信号的情况）,读取ACK信号需要释放SDA.



。

14 操作说明

14.1 器件地址（Device Address/Slave Address）

所有的通讯操作都是以 [START 条件] + [从设备地址 + （R/W 读写选择）开始的。
从设备地址如下：

	Transter data	SLAVE addrres							R/W bit
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
READ	65h	0	1	1	0	0	1	0	1(r)
write	64h								0(w)

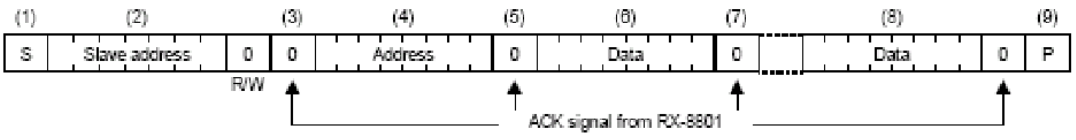
14.2 寄存器地址（Address/Register Address）

地址	功能	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	秒	0	40	20	10	8	4	2	1
1	分	0	40	20	10	8	4	2	1
2	时	0	0	20	10	8	4	2	1
3	周	0	6	5	4	3	2	1	0
4	天	0	0	20	10	8	4	2	1
5	月	0	0	0	10	8	4	2	1
6	年	80	40	20	10	8	4	2	1
7	RAM	.	.	.	.	.	.	.	.
8	Min alarm	AE	40	20	10	8	4	2	1
9	Hour alarm	AE	.	20	10	8	4	2	1
A	Week alarm	AE	6	5	4	3	2	1	0
	Day alarm		.	20	10	8	4	2	1
B	Timer couner0	128	64	32	16	8	4	2	1
C	Timer couner1	.	.	.	.	2048	1024	512	256
D	Ext register	TEST	WADA	USEL	TE	FSEL1	FSEL0	TEL1	TEL0
E	Flag register	0	0	UF	TF	AF	0	VLF	VDET
F	Contr registor	CSEL1	CSEL0	UIE	TIE	AIE	0	0	RESET

14.3 寄存器写操作时序图

指定地址写操作：

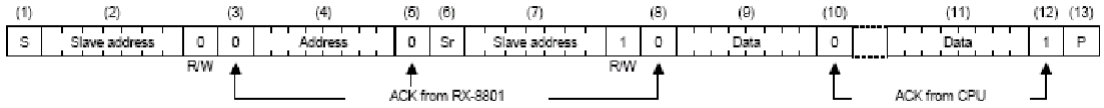
1） CPU发送开始条件[S]
2） CPU传输SE8025TC的从地址，用R/W位设定写模式。
3） 检测从SE8025TC发出的ACK信号。
4） CPU传输写入寄存器的地址到SE8025TC
5） 检测从SE8025TC发出的ACK信号
6） CPU将要写入的数据写道指定的寄存器
7） 检测从SE8025TC发出的ACK信号
8） 如果有需要可重复（6）和（7）步骤，写入的地址自动增加
9） CPU发送停止位[P]



14.4 寄存器读操作时序图

指定地址读操作：

1） CPU发送开始条件[S]
2） CPU传输XCRSE8025TIAD的从地址，用R/W位设定写模式
3） 检测从XCRSE8025TIAD发出的ACK信号
4） CPU传输读寄存器的地址到XCRSE8025TIAD
5） 检测从XCRSE8025TIAD发出的ACK信号
6） CPU发送RESTART条件[Sr]
7） CPU传输XCRSE8025TIAD的从地址，用R/W位设定读模式
8） 检测从XCRSE8025TIAD发出的ACK信号
9） 从XCRSE8025TIAD中读取步骤（4）指定的寄存器内容
10） CPU发送ACK信号给XCRSE8025TIAD
11） 如果有需要可重复（9）和（10）步骤，读取的地址自动增加
12） CPU发送一个‘1’作为ACK信号
13） CPU发送停止信号[P]

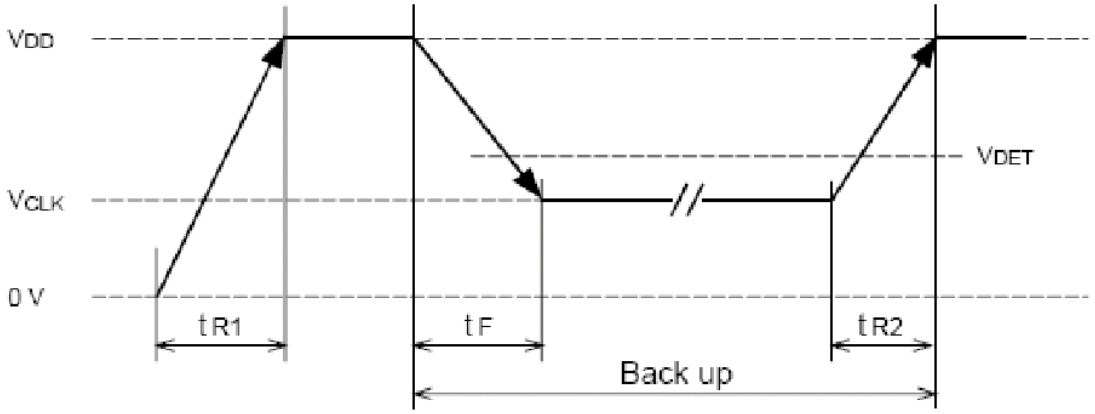


5、寄存器读写操作备注

读写操作中，寄存器的地址能自动加1，因此可以连续读写多个寄存器。（与EEPROM操作
相同）

读写时序图中（4），Address，即为寄存器的地址。（与XCRSE8025TIAD中Address不同）

15 备份以及恢复

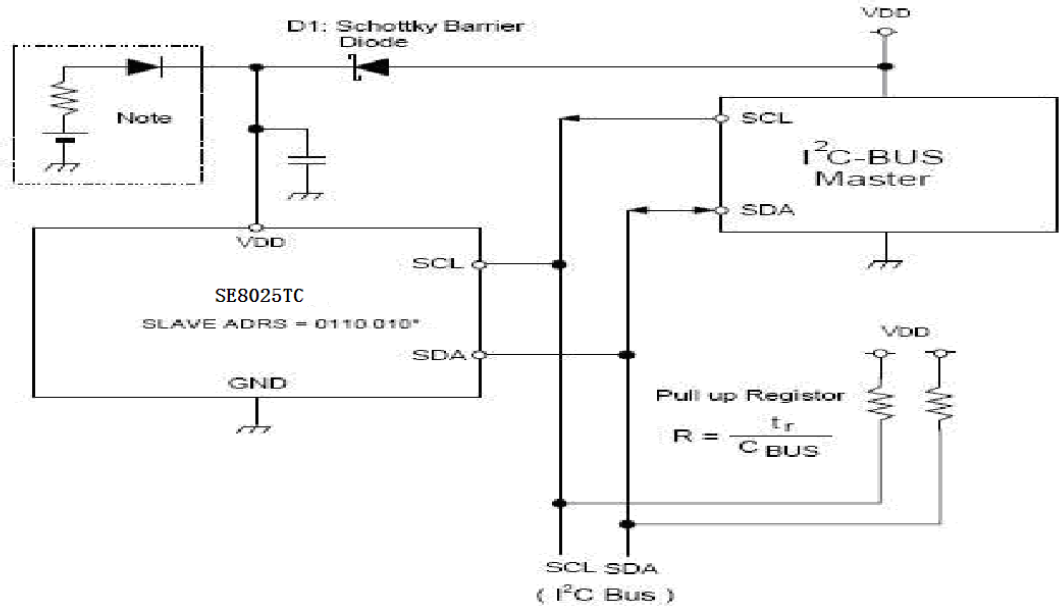


在应用电池作后备电源时特别注意：

ITEM	symble	condition	min	typ	max	unit
Power supply detection voltage 1	V_{DET}				2.2	V
Power supply detection voltage 2	V_{LOW}				1.6	V
Power supply drop	tf		2			us/v
Initial power-up time	Tr1				10	ms/v
Clock maintenance power-up time	Tr2	1.6v VDD≤3.6V	5			us/v
		1.6v VDD＞3.6V	15			us/v

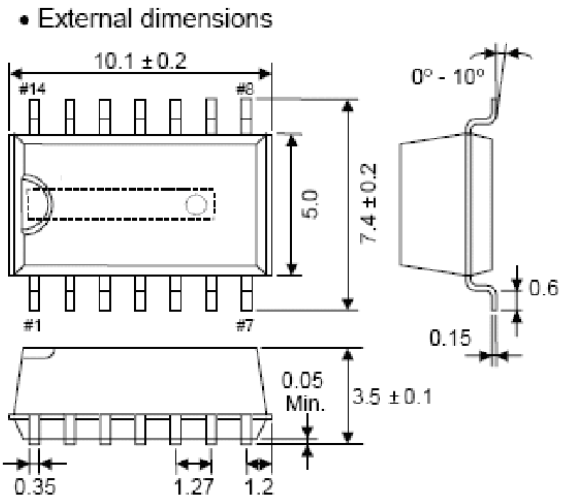
注意电平的切换，防止数据丢失

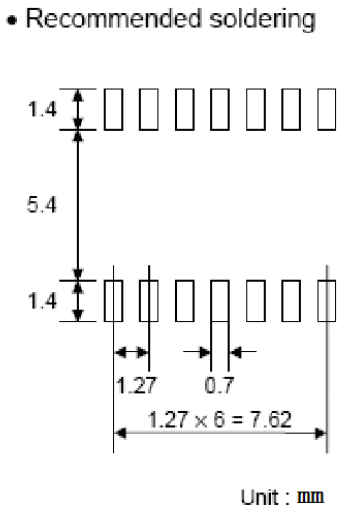
参考电路：



16 XCRSE8025TIAD 封装以及 PCB 设计：

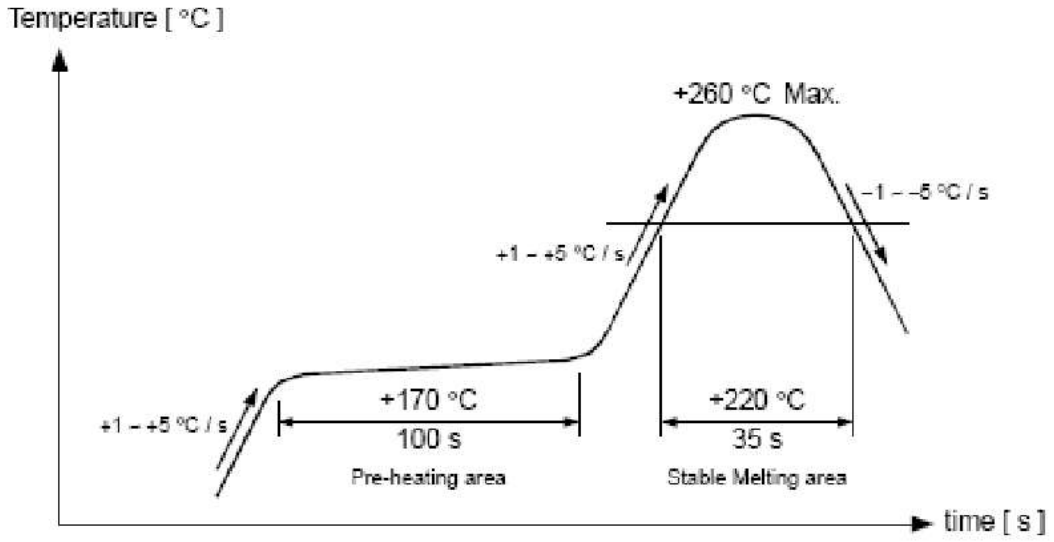
封装以及 PCB 设计：





焊接的参考温度曲线：

焊接的参考温度曲线：



※ 设计时应注意事项：

A：XCRSE8025TIAD 的I2C 总线与CPU 之间的接口电路设计，注意电压匹配。由于 一般设计会需要电池作备用电源，因此设计特别注意XCRSE8025TIAD 的电压要求，
参考（4 电气指标）给出的数据。

B：系统设计时，在I2C 总线上有多个器件的时候，应注意软件的设计，防止对XCRSE8025TIAD 的误操作。

C：时间设定的时候，一定要确保数据的正确性，一般应增加校验的子程序，例如验证日期对应的星期数就可以利用上面第8 页提到的两个公式。 D：在PCB设计的时候注意XCRSE8025TIAD 与CPU 的通讯走线应该越短越好，并且远离高频、高电流的信号线。同时旁路电容也应该靠近XCRSE8025TIAD 的电源端。 同时增加地线敷铜的面积，以防止干扰的产生。

E：生产测试过程中，对XCRSE8025TIAD 的焊接应参考焊接温度曲线。同时避免用超 声波对器件进行清洗操作，防止对内部晶体造成损坏。

F：当使用电池供电的时候，为了延长电池的供电时间，可以关闭FOUT 输出， 改变温度补偿的时间间隔（正常供电时用软件恢复设置）以达到降低功耗的 目的。

17 附件（I2C 读写程序 C 语言）

```
///**********************************
//** 函数原型: void IC_start(void); **
//** 功 能: IC 总线起始位. **
//**********************************/
void IC_start(void)
{
SDA_DIR=I2C_OUTPUT;
SCL=1; /** Send Start **/
SDA=1;
nop();
nop();
SDA=0;
nop();
nop();
SCL=0;
}
////**********************************
////** 函数原型: void IC_stop(void); **
////** 功 能: IC 总线停止位. **
////**********************************/
void IC_stop(void)
{
SCL=0; /** Send Stop **/
SDA_DIR=I2C_OUTPUT;
SDA=0;
nop();
nop();
SCL=1;
nop();
nop();
SDA=1;
}
/////*****************************************************
////** 函数原型: bit IC_writebyte(uchar wdata); **
////** 功 能: 向 IC 总线发送 8 位数据,并请求一个应答信 **
////** 号 ACK.如果收到 ACK 应答则返回 1(TRUE), **
////** 否则返回 0(FALSE). **
////*****************************************************/
unsigned char IC_WriteByte( unsigned char wdata)
```

```
{
unsigned char i;
SDA_DIR=I2C_OUTPUT;
for(i=0;i<8;i++)
{
SCL=0;
SCL=0;
nop();
nop();
if(wdata&0x80) SDA=1;
else SDA=0;
nop();
nop();
SCL=1;
wdata<<=1;
}
SCL=0;
SCL=0;
nop();
nop();
SDA_DIR=I2C_INPUT;
SCL=1;
i=0;
while(SDA)
{
if(++i>12){SCL=0;return(0);}
}
SCL=0;
return(1);
}
/////*************************************************
////** 函数原型: uchar IC_readbyte(void); **
////** 功 能: 从 IC 总线上接收 8 位数据,并将接受到 **
////** 8 位数据作为一个字节返回,不回送应 **
////** 答信号 ACK. **
////*************************************************/
unsigned char IC_ReadByte(void)
{ unsigned char i;
unsigned char IC_data=0;
SCL=0;
SDA_DIR=I2C_OUTPUT;
SDA=1;
SDA_DIR=I2C_INPUT;
for(i=0;i<8;i++)
```

```
{
SCL=1;
SCL=1;
nop();
nop();
IC_data<<=1;
IC_data|=SDA;
nop();
SCL=1;
nop();
SCL=0;
nop();
}
nop();
nop();
// SCL=1;
nop();
nop();
nop();
SCL=0;
SDA_DIR=I2C_OUTPUT;
SDA=0;
return(IC_data);
}
/////*****************************************************************
////** 函数原型: bit readEEone(uchar instr,uchar addr,uchar num); **
////** 功 能: 从 S42WD42 中读取 num 个字节的数据,采用序列读操作方 **
////** 式从片内 Address 地址开始连续读取数据.S42WD42 不接 **
////** 受指定的地址则返回 0(FALSE). **
////*****************************************************************/
unsigned char readEEone(unsigned char addr,unsigned char num)
{
unsigned char i;
//di();
IC_start();
if(IC_WriteByte(WRITE)==0)
{
IC_stop(); return(0);
}
if(IC_WriteByte(addr)==0)
{
IC_stop(); return(0);
}
IC_start();
```

```
if(IC_WriteByte(READ)==0)
{
IC_stop(); return(0);
}
for(i=0;i<num-1;i++)
{
EEbuf[i]=IC_ReadByte();
SDA_DIR=I2C_OUTPUT;
SDA=0; /** Send ACK **/
SDA=0;
SCL=1;
}
EEbuf[num-1]=IC_ReadByte();
SDA_DIR=I2C_OUTPUT;
SDA=1; /** Send Read End **/
SDA=1;
nop();
SCL=1;
IC_stop();
//ei();
return(1);
}
/////*********************************************************************
////** 函数原型: bit readEE(uchar instr,uchar addr,uchar num); **
////** 功 能: 从 S42WD42 中读取 num 个字节的数据,采用序列读操作方 **
////** 式从片内 Address 地址开始连续读取数据.S42WD42 不接 **
////** 受指定的地址则返回 0(FALSE). **
////*********************************************************************/
unsigned char readEE(unsigned char addr,unsigned char num)
{
unsigned char i;
for(i=0;i<3;i++)
{
if(readEEone(addr,num)) { return(1);}
}
//ei();
return(0);
}
///*********************************************************************
//** 函数原型: bit writeEEone(uchar instr,uchar addr,uchar num); **
//** 功 能: 将 EEbuf[]单元中的数据写入 S42WD42 的 num 个字节. **
//** 采用页写操作方式,每次写入时都需要指定片内地址. **
//** 如果 S42WD42 不接受指定的地址或某个传送的字节未收到 **
//** 应答信号 ACK,则返回 0(FALSE). **
```

```
//*****************************************************************/
unsigned char writeEEone(unsigned char addr,unsigned char num)
{
unsigned char i;
gie=0;
IC_start();
if(IC_WriteByte(WRITE)==0)
{
IC_stop(); return(0);
}
if(IC_WriteByte(addr)==0)
{
IC_stop(); return(0);
}
for(i=0;i<num;i++)
{
if(IC_WriteByte(EEbuf[i])==0)
{
IC_stop(); return(0);
}
}
IC_stop();
nop();
nop();
SDA=0;
SCL=0;
gie=1;
return(1);
}
//=============================================================================
```

writeEEone(0x0d,3); 写数据到 8025T 0X0D 是地址，3 是写入数据的数量。
要写入的数据存在 EEbuf[i] 中。
readEEone(0x0d,3); 读出数据到 EEbuf[i]
改变 SDA 方向： I2C_OUTPUT -----输出。
I2C_INPUT------输入