产品特点

- 包括秒、分、小时、星期、日、月和年的计数，考虑闰年（2100 年前）；
- 为日期存储提供 56 字节的自主管理能力的随机存储器；
- I2C 接口，标准 100KHz，高速 400KHz；
- 可编程矩形输出信号；
- 主供电和备用供电监控和自动切换；
- 电池备用供电模式下，供电电流小于 500nA；
- 工业应用中温度范围：-40℃ 至 +85℃；

产品描述

CBM1307 是基于二进制-十进制的带有日期的数字时钟，具有额外的 56 字节能够自主管理的静态随机存储器并且功耗低。存储空间通过 I2C 读写操作。计时电路用于计算以小时、分钟、秒表示的实时时间，也用于计算周、日、月、年，每月的最后一天自动调整该月为 31 天或少于 31 天，包括修正闰年。时钟可以以 24 小时格式或具有上、下午指示的 12 小时格式运行。CBM1307 具有内置的电源控制电路，该电路决定供电中断并且将设备自动切换到电池供电模式。

目录

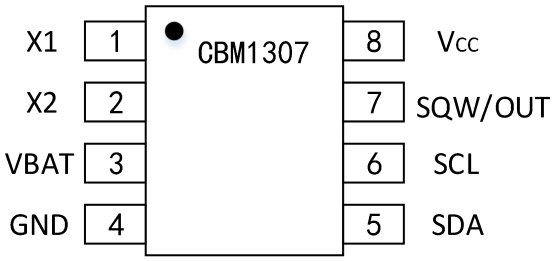
产品特点.....1
产品描述.....1
目录.....2
修订日志.....3
引脚配置.....4
引脚描述.....4
功能框图.....5
工作温度范围.....5
直流电气特性.....6
交流电气特性.....7
典型工作特性.....8
时序图.....9
运行使用.....9
时钟芯片和随机存储器地址表.....9
振荡电路.....10
晶体规格.....10
时钟精度.....10
振荡器电路显示内部偏置网络.....10
晶体的推荐布局.....11
时间和日历.....11
寄存器实时时钟模块（RTC）CBM1307.....12
控制寄存器.....12
双线串行数据总线.....13
串行双线总线数据传输.....14
CBM1307 两种工作模式.....15
应用说明.....17
典型应用电路.....26
封装尺寸及结构.....27
包装/订购信息.....28
附录.....29
1.测试部件概述.....29
2.实时时钟程序.....31
3.RTC 源代码实例.....36

4.RTC 测试过程.....41

修订日志

版本	修订日期	变更内容	变更原因	制作	审核	备注
V1.0	2025.3.14	更新产品特点，产品描述错误	错误更新	WW	LYL	

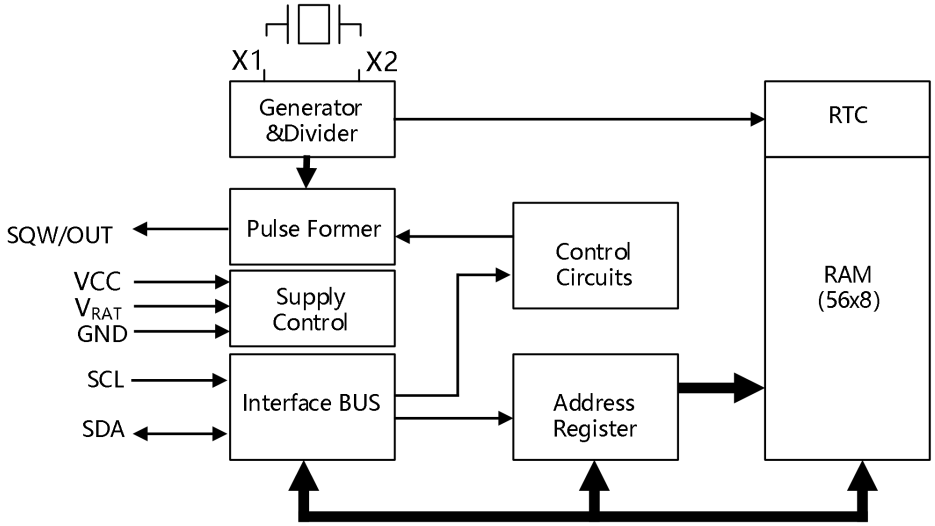
引脚配置



引脚描述

引脚编号	符号	输入/输出	引脚描述
1	X1	In	石英谐振器连接
2	X2	In	石英谐振器连接
3	VBAT	In	电池
4	GND	In	接地
5	SDA	In/Out	串行数据输入/输出
6	SCL	In	连续周期信号输入
7	SQW/OUT	Out	矩形波信号输出
8	V_{CC}	In	电源供应

功能框图



工作温度范围

CBM1307 微型电路工作温度范围：TA = - 40 至 + 85℃

推荐直流工作条件和绝对额定参数

参数	符号	推荐工作条件		绝对额定参数		单位
		最小值	最大值	最小值	最大值	
供电电压	V_{CC}	4.5	5.5	-0.5	7.0	V
电池电压	V_{BAT}	2.0	3.5	-0.5	7.0	V
低电平输入电压	V_{IL}	-0.3	0.8	-0.5	7.0	V
高电平输入电压	V_{IH}	2.2	V_{CC}+0.3	-0.5	7.0	V
储存温度	T_S	-	-	-55	+125	℃

* 超出列表中的绝对额定参数设定会导致设备永久损坏。列表中列举的额定参数是产品使用中允许的强度检测条件下的测定值，本文中没有提及的不在此范围内。长时间在超过绝对额定参数的条件下使用可能会影响产品可靠性。

直流电气特性

(T_A=–40 至+85℃, V_{CC}= 4.5 至 5.5V)

参数	符号	模式	绝对额定参数		单位
			最小值	最大值	
输入漏电流（只有时钟线SCL）	I_{LI}		-	1	uA
输入输出漏电流（只有时钟线SCL） (双向数据线SDA和方波输出SQW/OUT)	I_{LO}		-	1	uA
低电平输出电压	V_{LO} [1]	V_{CC} = 4.5 V	-	0.4	V
数据传输模式的电流损耗	I_{CCA}	f_{SCL} = 100 kHz	-	1500	µA
静态模式的电流损耗	I_{CCS}	V_{CC} = 5 V and SDA, SCL = 5 V	-	200	µA
电池供电模式的电流损耗 (方波输出关闭，32 kHz打开)	I_{BAT1}	V_{CC} = 0 V, V_{BAT} = 3 V	-	0.5	µA
电池供电模式的电流损耗 (方波输出打开，32 kHz打开)	I_{BAT2}	V_{CC} = 0 V, V_{BAT} = 3 V	-	0.8	µA

低电平电压在 5mA负载电流测定；输出低电平电压在电容负载下接地（V_{OL} = GND）。

交流电气特性

(T_A=−40 至+85℃,V_{CC}=4.5 至 5.5V)

参数	符号	模式	绝对额定参数		单位
			最小值	最大值	
循环频率时钟线SCL	f_{SCL}	-	0	100	kHz
从关闭到启动之间的总线空闲状态时间	t_{BUF}	-	4.7	-	μs
启动状态保持（重复）时间	$t_{HD:STA}$ [1]	-	4.0	-	μs
循环脉冲时间线的低电平状态持续时间	t_{LOW}	-	4.7	-	μs
循环脉冲时间线的高电平状态持续时间	t_{HIGH}	-	4.0	-	μs
重启预设时间	$t_{SU:STA}$	-	4.7	-	μs
数据保持时间	$t_{HD:DAT}$ [2]	-	0	-	μs
数据预设时间	$t_{SU:DAT}$	-	250	-	ns
双向数据线和时钟线信号上升时间	t_R	-	-	1000	ns
双向数据线和时钟线信号下降时间	t_F	-	-	300	ns
停止准备设置时间	$t_{SU:STO}$	-	4.7	-	ns
每条总线全部电容加载电压	C_B	-	-	400	pF
电容输入、输出电压	$C_{I/O}$	-	10	10	pF
石英谐振器负载电容电压	C_{LX}	-	12.5	12.5	pF

首次循环信号形成的间隙后，设备应该在电路内部为双向数据线信号（相对时钟信号高电平的最小值）确保保持至少 300 纳秒的持续时间，以便覆盖时钟信号下降沿的不确定区域，在这种情况下，如果不能提高时钟信号高电平的持续时间（$t_{HD:DAT}$），应确保最大化数据保持时间。

典型工作特性

(除非另外说明，V_{CC}=5.0V,T_A=+25℃)

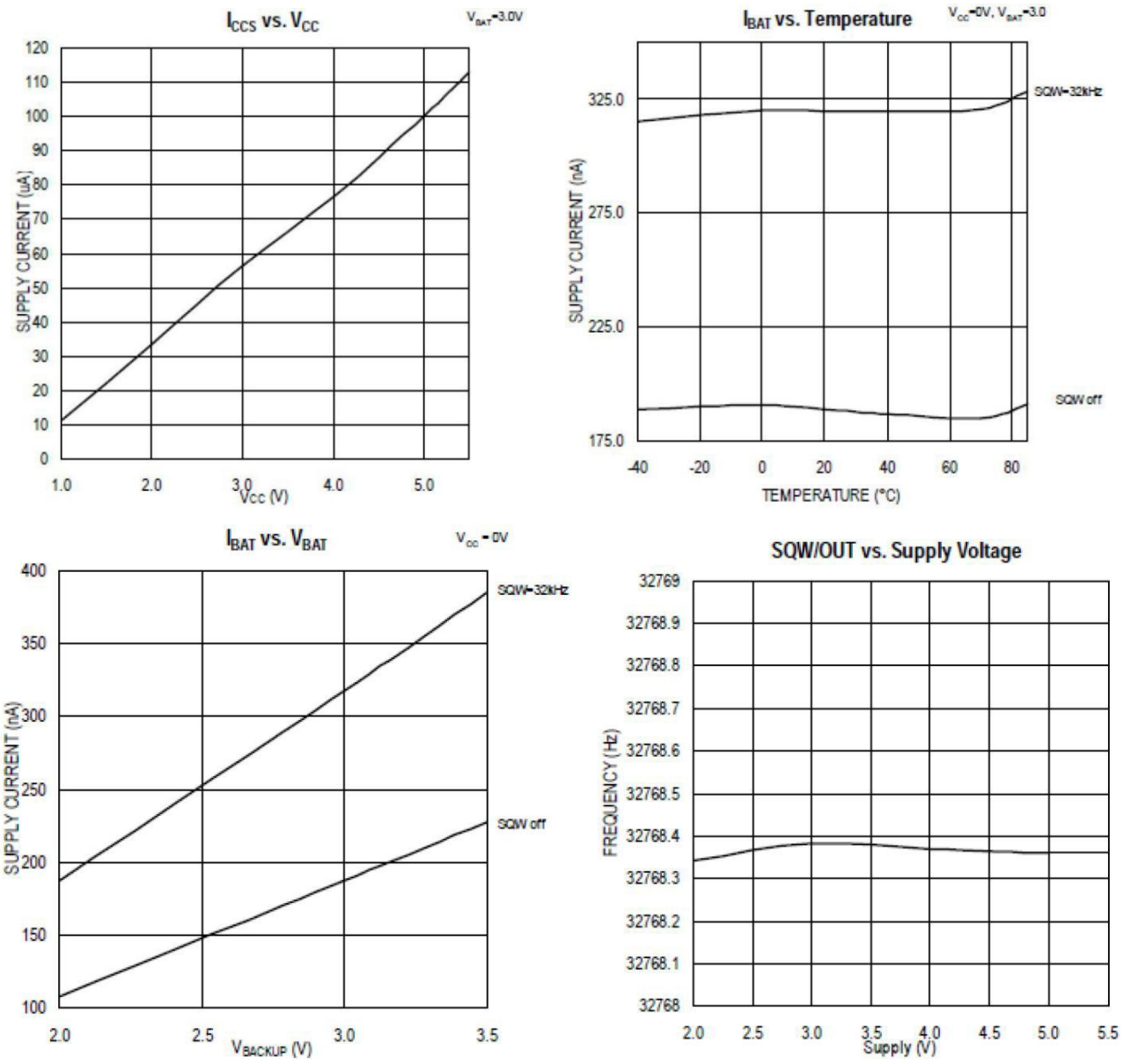
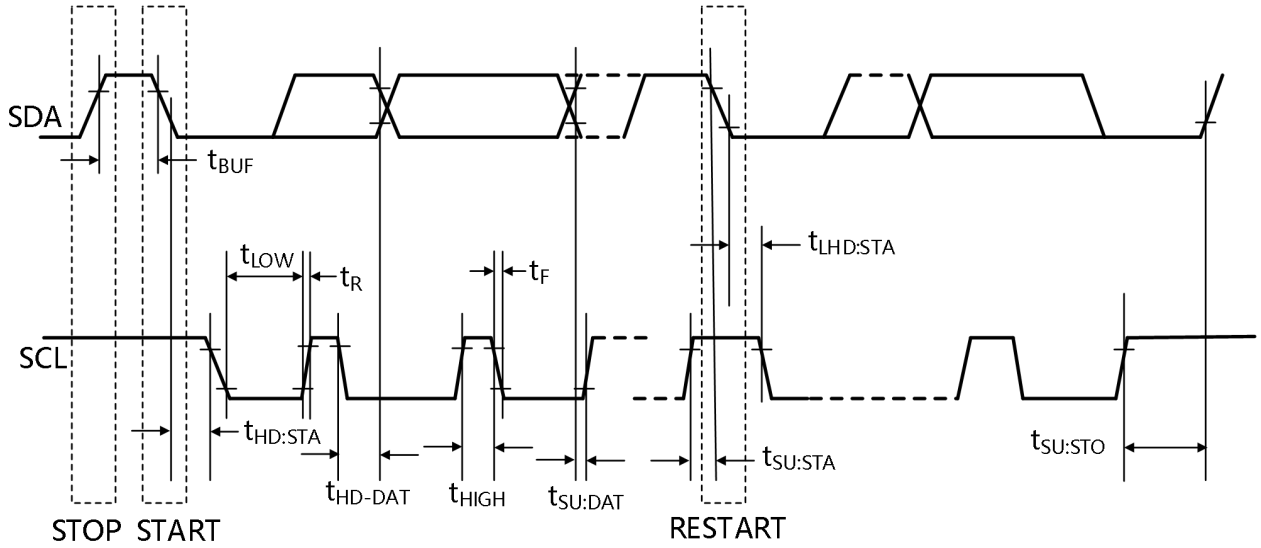


图 9.逆变调节器关闭状态下的接地参考

时序图



运行使用

CBM1307 作为驱动设备在串行总线下运行。为使用它，它被要求设置启动状态，并且在寄存器地址后面发送设备识别码，有可能因此寻址下一个寄存器，直到设置为停止状态。当供电电压 V_{CC} 降至低于 1.25 x V_{BAT}（电源电压）时，访问设备的进程终止，地址计算器被重新设置。此时，设备不能识别除正在写入的错误信息外的其它输入数据。供电电压 V_{CC} 降至低于 1.25 x V_{BAT}（电源电压）时，设备切换至电池供电、低功耗模式。当供电电压 V_{CC} 高于电源电压 V_{BAT} + 0.2 V 时，设备由电池供电切换至电路供电，并且当供电电压高于 1.25 x V_{BAT} 时，识别输入数据。

时钟芯片和随机存储器地址表

00H	SECONDS
	MINUTES
	HOURS
	DAY
	DATE
	MONTH
	YEAR
07H	CONTROL
08H	RAM
3FH	56X8

时钟芯片和随机存储器的寄存器地址表在图中红列出。实时报时器地址为 00h – 07h，随机存储器寄存器地址为 08h – 3Fh，多字节存储模式下，当指针指向地址 3Fh 时，随机存储器地址末尾，正好转换为寄存器地址 00h，即报时器起始地址。

振荡电路

CBM1307 使用外部 32.768kHz 晶振，振荡电路运行时不需要外部电路或电容，表中列出了外部晶振的部分晶振参数。

晶体规格*

参数	符号	最小值	典型值	最大值	单位
标称频率	f_o		32.768		kHz
串联电阻	ESR			45	kΩ
负载电容	C_L		12.5		pF

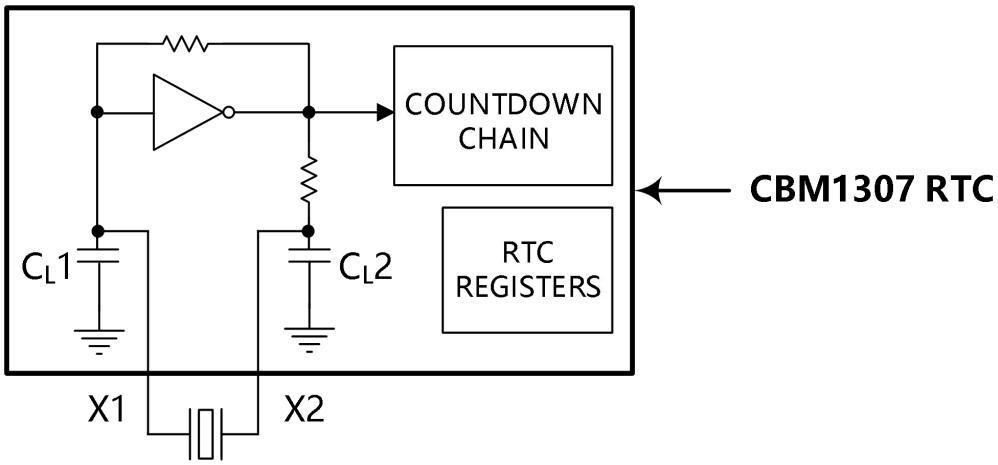
* 晶振，传输和晶振输入针脚应该与射频产生信号绝缘。

应用提示：晶振考虑实时时钟的额外规范，见 14 页。

时钟精度

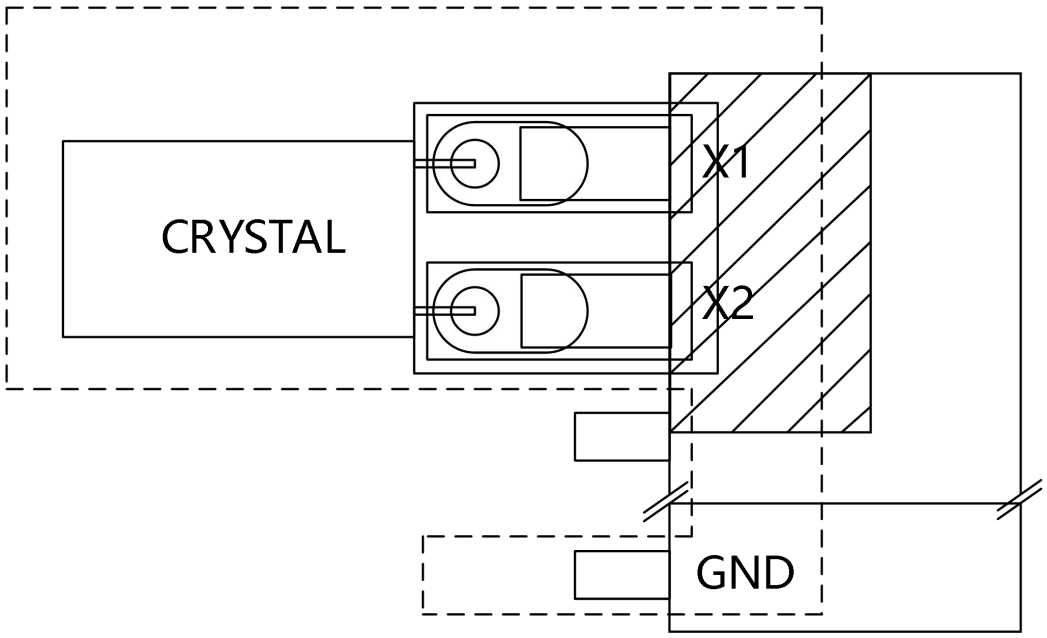
时钟精度依靠晶振精度、振荡电路的电容负载与电容负载对晶振调谐的适配，由温度变化导致的晶振频率漂移会增加额外错误，额外的电路噪声会耦合到振荡器上。

振荡器电路显示内部偏置网络



图中显示了一个振荡电路的功能图。如果在特点性能上使用晶振能够使启动时间通常少于一秒。

晶体的推荐布局



时间和日历

通过读取合适的字节寄存器得到时间和日期信息。实时小时寄存器如图中所示。通过写入合适的字节将预设值与时间以及日历进行初始化。时间和日历寄存器包含的信息以二到十进制编码代表。寄存器 0 的第 7 位代表时间停止位（CH），当这一位设置为 1 时，发生器设置完毕。

当供电电源切换时，不能决定寄存器的初始状态。有必要在设置初始配置时使用发生器（位 CH = 0）。

CBM1307 以 12 小时制或 24 小时制工作。监视寄存器第 6 位决定工作模式 12 小时制模式与高电平一致，在 12 小时制模式下，位 5 是上午/下午（AM/PM）位，高电平代表下午（PM）。在 24 小时制模式下，位 5 代表小时十位的第二位(20 -23 时)。

当信号“START”向两线总线发送时，寄存器辅助设置会发生实时转换。当钟表工作时，实时数据从辅助寄存器中读取，这样可以消除由于在访问进程中更新基本寄存器而进行的不必要的重复读取操作。

寄存器实时时钟模块（RTC）CBM1307

	BIT7						BIT0	
00H	CH	2nd DIGIT of SECONDS			1st DIGIT of SECONDS			00-59
	X	2nd DIGIT of MINUTES			1st DIGIT of MINUTES			00-59
	X				1st DIGIT of HOURS			01-12 00-23
	X	X	X	X	X	DAY of WEEK		1-7
	X	X			1st DIGIT of DATE			01-28/29 01-30 01-31
	X	X	X		1st DIGIT of MONTH			01-12
					1st DIGIT of YEARS			00-99
07H	OUT	X	X	SQWE	X	X	RS1	RS0	

控制寄存器

控制寄存器用来控制引脚 SQW/OUT 。

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUT	**X**	**X**	**SQWE**	**X**	**X**	**RS1**	**RS0**

OUT（输出控制）：输出控制位表示当输出矩形信号锁定时，引脚 SQW/OUT 的输出逻辑电平。

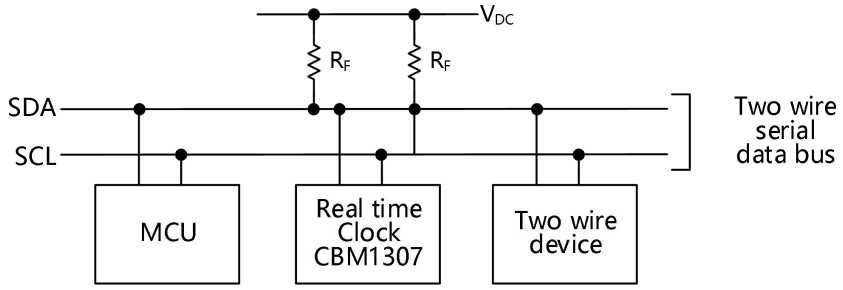
SQWE (矩形信号启动): 矩形信号启动会预先讲逻辑值设为“1”，激活发生器输出。输出矩形信号的频率由位 RS0 和 RS1 决定。

RS (频率选择): 这些频率选择位决定当矩形信号输出被激活时，输出矩形信号的频率。下列表中列出了由频率选择位（RS）所决定的频率。

RS1	RS0	SQW/OUT频率
0	0	1 Hz
0	1	4,096 kHz
1	0	8,192 kHz
1	1	32,768 кГц

双线串行数据总线

CBM1307 支持双向两线总线和数据交换协议。总线由“master”设备控制，这个设备产生周期信号（SCL），控制总线访问，可以产生 START 和 STOP 两个状态信号。图中所示为两线协议总线的典型配置。



只有总线空闲时，才能初始化数据传输。数据传输进程中，当循环信号线处于高电平状态时，数据传输线应保持线路稳定。当循环信号线处于高电平状态时，数据传输线的状态改变被当做控制信号。

遵从上述过程，下列条件受到影响：

总线空闲：数据传输线和循环信号处于高电平状态。

数据传输启动：当循环信号线处于高电平状态时，数据传输线由高电平向低电平变迁时，由状态“START”所决定的状态转换。

数据传输停止：当循环信号线处于低电平状态时，数据传输线由低电平向高电平变迁时，由状态“STOP”所决定的状态转换。

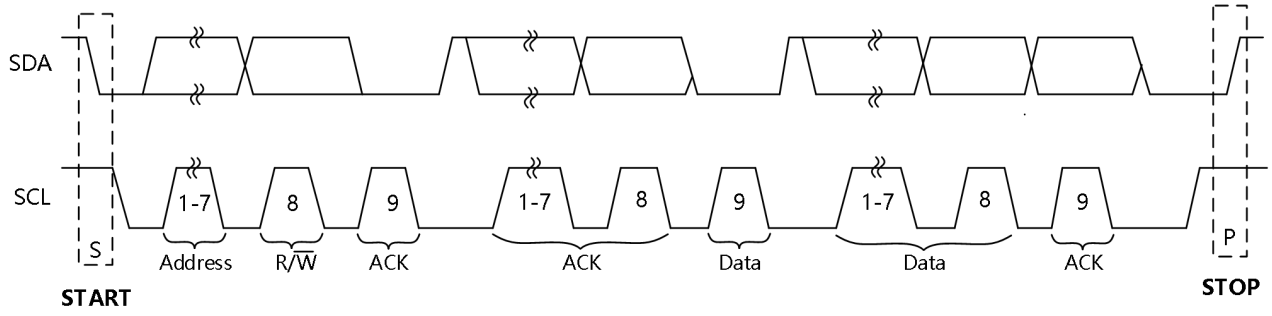
有效数据：数据线状态依据有效数据确定，当进入状态“START”后，循环信号处于高电平状态时，数据线保持稳定。在线数据应该在循环信号处于低电平状态时发生改变，一个循环脉冲会影响每个数据位。

每一次数据传输状态在状态 START 开始时启动，在状态 STOP 开始时终止。状态 START 和状态 STOP 之间传输数据字节的大小没有限制，其大小是由«master»设备决定的。信息以字节形式传输，是否成功接收每个字节是由第九个字节确认。CBM1307 在正常模式下的工作频率只有 100 kHz。

接收确认：每个确认接收数据的接收设备接收完所有字节后，必须产生一个接收确认信息。《Master》设备应该产生一个循环脉冲，这个循环脉冲信号由确认位分配。

如果接收确认信号处于高电平状态，确认周期脉冲到达时要确认数据接收，此时设备应将串行数据线（SDA）切换为低电平状态。当然，也应当认为考虑预设时间和保持时间。《master》设备应向“slave”设备发送数据传输完成信号，从“slave”循环脉冲收到接收确认信号时，终止确认位生成。在这个过程中，“slave”设备应该将数据传输线状态切换为低电平状态，这样能够使《master》设备电平状态达到产生 STOP 信号的条件。

串行双线总线数据传输



根据位 RF 的状态，有两种可能的传输类型：

1. 从«master»发射器到«slave»接收器的数据传输

第一个字节由一个«master»设备发送给一个«slave»设备，紧接着一系列数据字节。《slave》设备接收到每个字节后返回一个接收确认字节。数据传输顺序为：首先传送最高位数字（MSB）。

2. 从« slave»发射器到«master»接收器的数据传输

第一个字节（«slave»发送的）施加到«master»设备上的数据，然后«master»设备返回一个确认位，其后传输由«slave»设备发出的数据序列，《master》设备接收到每个字节后返回一个接收确认字节，除此之外还包括最后一位字节，接收到最后一位字节候，接收确认位不再返回。

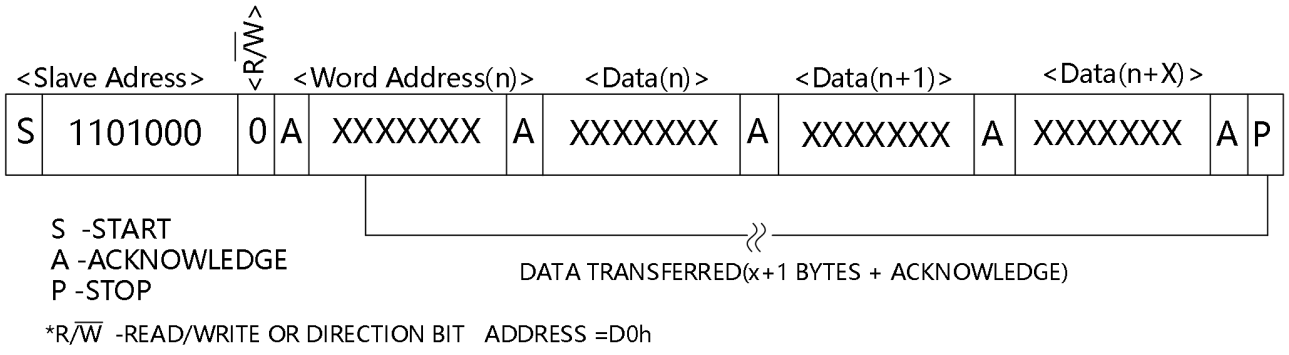
«master»设备产生所有的周期脉冲、设置状态“START”和状态“STOP”。数据传输是在状态“STOP”出现时或者状态“START”重复出现时完成。重复出现的状态“START”是下一个串行数据传输的开始标志，此时总线空闲。据传输顺序为：首先传送最高位数字（MSB）。

CBM1307 两种工作模式

1. «slave»接收器模式（CBM1307 写入模式）：

串行数据和周期脉冲由对应的数据线（SDA）和控制线（SCL）接收，每个字节传输后，确认位发送接收信号，状态“START”和“STOP”串行数据传输的开始和结束。在“slave”地址确认接收和发送接收位设置完后，地址识别通过硬件工作，地址字节是首字节，在状态“START”发生后接收，由“master”产生。地址字节包含 7 位 CBM1307 地址，即 1101000，伴随着方向位（R_F）状态变化按照指示的方向进行数据传送，该模式时方向位显示为 0。地址字节被接收和解码后，CBM1307 将确认信息发送到数据线（SDA）上，CBM1307 “slave”地址和写入为确认后，«master»发送 CBM1307 寄存器地址，因此 CBM1307 寄存器指示器会预留出来，然后«slave»开始发送每个数据字节，每个数据字节后都会有接收确认信息。数据写入“master”完成后，由“master”产生状态“STOP”表示数据传输结束。

数据写入-《SLAVE》数据接收模式

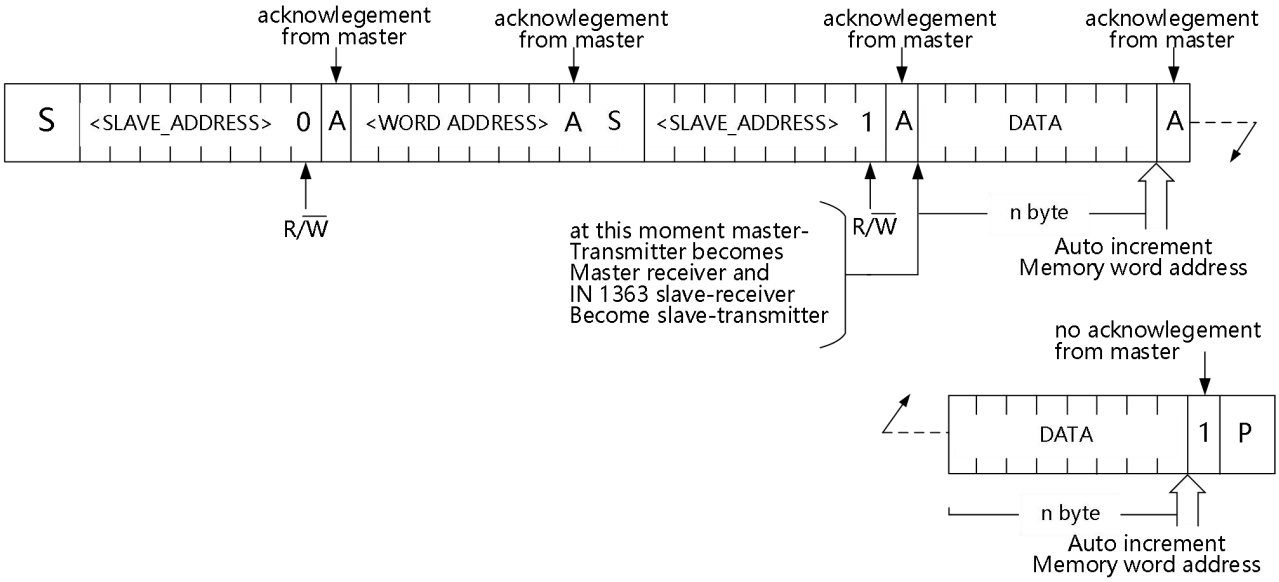


数据写入-《SLAVE》数据接收模式

2.《slave》收发模式（CBM1307 实时输出模式):

«slave»处于接收模式时，接收和处理第一个字节，在这个模式下，传送方向位会表示数据传输方向的改变。CBM1307 通过数据线 (SDA) 发送串行数据，控制线 (SCL) 的循环脉冲状态 “START” 和 “STOP” 作为数据连续传输的开始和结束。地址字节是第一个字节，由«master»产生的状态 “START” 出现时，接收该字节。地址字节包含 7 位 CBM1307 地址，即 1101000，伴随着方向位（R_F）状态变化按照指示的方向进行数据传送，该模式时方向位显示为 1。地址字节被接收和解码后，CBM1307 收到数据线（SDA）确认信息，然后，CBM1307 开始从寄存器指示器所指示的地址发送数据，如果寄存器指示器还没写入时初始化读取模式，则第一个读取的地址就是保存在寄存器指示器中的最后地址。为完成读取，CBM1307 应该发送《非确认》的位信息。

数据读取 – 《SLAVE》数据传输模式



数据读取 –数据传输模式

应用说明

实时时钟系统(RTCS)使用晶体的考虑事项

应用说明描述了将 32,768Hz 的晶体连接到实时时钟（RTC）上时如何选择晶体和有关布局技术。同时也提供里有关晶振电路设计标准、系统设计和制造相关问题讨论的信息。

晶振基础说明

实时时钟系统中（RTCs）使用的晶振是满足 CMOS 逆变器变化的皮尔斯石英振荡器。图 1 所示为一般配置，实时时钟系统（RTCs）包括集成负载电容（C_{L1} 和 C_{L2}）和偏置电阻。皮尔斯石英振荡器利用晶体以并联谐振模式工作。使用并联谐振模式的晶体与专门的负载电容指定产生某一确定频率。为了使晶振能够以正确的频率工作，晶振电路的负载晶体必须选择合适的负载电容。

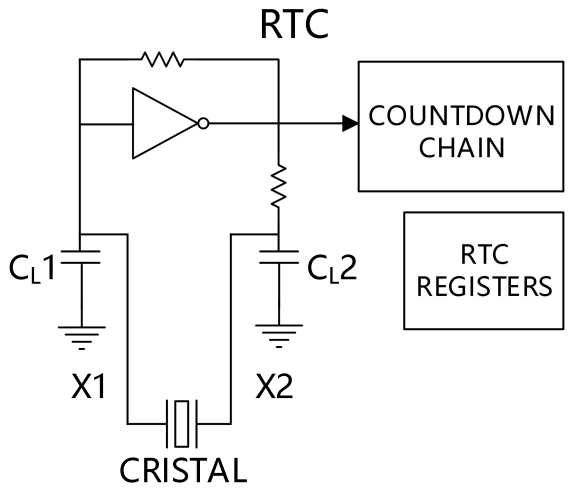


图 1. 具有内部负载电容和偏置电阻的实时时钟振荡器

精度

基于晶体的振荡器电路的频率精度主要由晶体的精度与晶体和振荡器之间电容负载的匹配度来决定。如果负载电容值小于晶体所需电容设置值，振荡器会频率比标称值大，如果负载电容值大于晶体所需电容设置值，振荡器会频率比标称值小。

另外除了晶体与负载匹配产生误差外，当环境温度变化时，晶体固有频率也会发生变化。实时时钟系统（RTCs）使用音叉晶体，这种晶体会在超温时产生误差，如图 2 所示。20ppm 的误差相当于每月 1 分钟的误差。

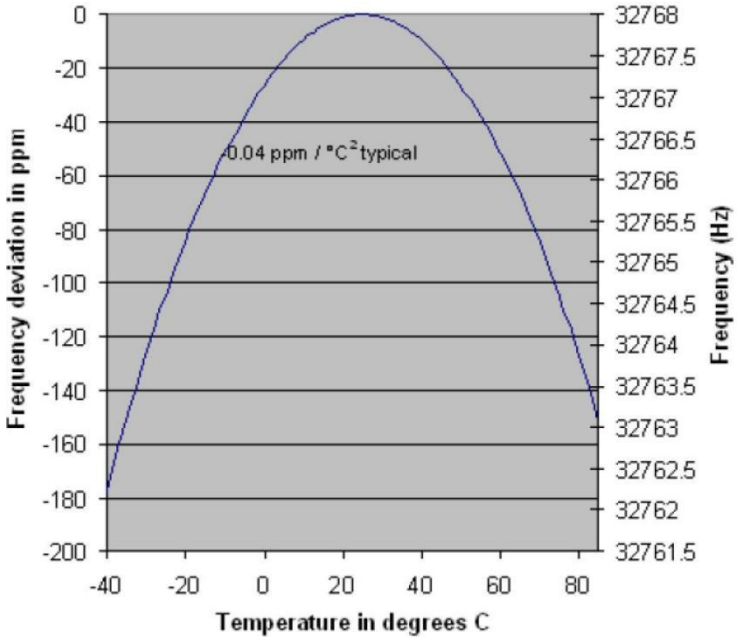


图 2. 晶体频率与温度的关系

提示：如果需要更好的精确度，可以使用温度补偿晶体振荡器，如 DS32kHz 的振荡器

晶体参数

图 3 所示为晶体频率等效电路，与晶体频率接近的谐振频率电路包括一个串联电路，串联电路由动态电感 L_1，动态电阻 R_1 和动感电容 C_1 串联组成，还有一个并联元件 C_O，该元件是晶体寄生电容。

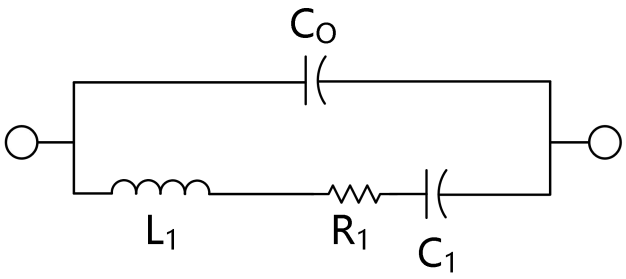


图 3. 晶体频率等效电路

由晶体引脚可以看出，负载电容 C_L 是振荡电路的负载电容，图 4 所示 C_L 是与晶体并联的电容，振荡电路中使用负载电容 C_{L1} 和 C_{L2}，与电路中的任何寄生电容一起构成总负载电容。所有的实时时钟系统(RTCs)集成了 C_{L1} 和 C_{L2}，注意在 PC 电路板布局中应最小化寄生电容的占用空间。下列公式描述的是 C_L 与负载电容值的关系：

$$CL = [\frac{(C_L1 \times C_2 2)}{(C_L1 + C_2 2)} + C_{STRAY}]$$

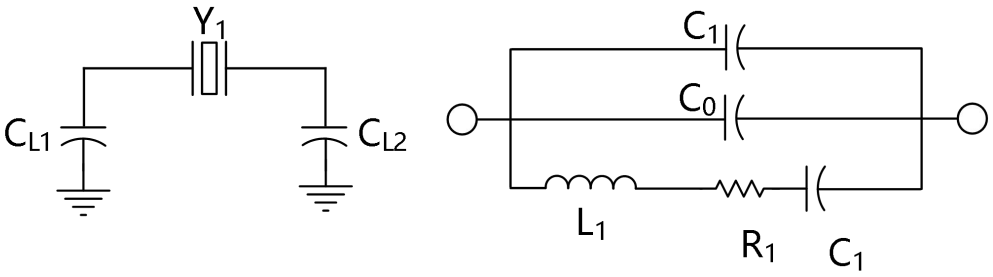


图 4. 晶体负载电容与等效兵龄负载

大多数晶体最大可达 1μW 的驱动功率，所有的实时时钟系统在 1μW 的功率下运行，驱动功率的大小由下列公式决定：

$$P = 2R_1 \times [\pi \times 32768(C_0 + C_L)VRMS]2$$

式中 VRMS 是通过振荡器电压的均方根（RMS）。

振荡器起振时间

振荡器起振时间高度依赖晶体特性，PC 板漏电和布局，高电平等效串联电阻和过载电容负载是起振时间过长的主要原因，使用推荐特点和合理布局晶体的电路通常会在一秒钟内启动。

表 1. 振荡器特点

参数	符号	最小值	典型值	最大值	单位
额定频率	FO		32.768		kHz
调整频差	delta F/ FO		±20		ppm
负载电容	C_L		6		pF
拐点温度	T0	20	25	30	℃
频率温度系数	k			0.042	ppm/℃
品质因数	Q	40,000	70,000		
等效阻抗	ESR			45	kΩ
寄生电容	C0		1.1	1.8	pF
电容比率	C0/C1		430	600	
激励功率	DL			1	μW

注释 1：有些设备允许更高的等效阻抗值，具体要求请看参数说明。

表 2. 晶体供应商，圆柱形 (ESR=45kΩ)

制造商	部件	频率容差 (ppm)	等效串行电阻 ESR(KΩ)	激励功率最大值(μW)	晶振-pF	调整电容	温度范围(℃)	表面安装或穿孔安装	封装尺寸 (mm)	制造商订购代码
Citizen	CFS-1 45	±20	40	1.0	8.0	yes	-10至+60	TH	1.5×5.1	
Citizen	CFS-206	±20	35	1.0	12.5	yes	-10至+60	TH	2.1×6.2	
Citizen	CMR-200T	±20	35	1.0	12.5或6.0	yes	-40至+85	SMT	2.0×6.0	CMR200TB32.768KDZFTR 或 CMR200TB32.768KDZBTR
ECS, Inc.	ECS-3×8	±20	35	1.0	12.5	?	-40至+60	TH	3.1×8.2	
ECS, Inc.	ECS-2×6	±20	35	1.0	12.5	?	-10至+60	TH	2.1×6.2	
ECS, Inc.	ECS-1×5	±20	35	1.0	8	?	-10至+60	TH	1.5×5.1	
KDS/Daiwa	DT-26	±20或±30	40	1.0	12.5	yes	-10至+60	TH	2.0×6.0	1TB602G00
KDS/Daiwa	DT-38	±20或±30	30	1.0	12.5	yes	-10至+60	TH	3.0×8.0	
Pletronics	W×15	±20	40	1.0	8.0	yes	-10至+60	TH	1.5×5.1	WX15-32.768k-6pF
Pletronics	WX26	±20	40	1.0	12.5	6.0	-10至+60	TH	2.1×6.2	WX26-32.768k-6pF
Fox	NC-38		35	1.0	12.5	6.0	-20至+60	TH	3.0×8.3	
Seiko	C-001R	±20	45	1.0	12.5	6	-10至+60	TH	3.1×8.0	
Seiko	C-2	±20	35	1.0	12.5	6	-10至+60	TH	2.0×6.0	

注: 圆柱尺寸为柱体直径和长度，不包括引脚，所有的尺寸为近似值。

表 3. 晶体供应商，表面贴装

制造商	部件	频率容差(ppm)	等效串行电阻 ESR(KΩ)	激励功率最大值(μW)	晶振-pF	调整电容	温度范围(℃)	尺寸(mm) 近似值 包括引脚
Seiko	SP-T3	±10, ±20	55	1.0	12.5	yes	-40至+85	7.3×4.3×1.8
Seiko	SP-T2	±20	50	1.0	12.5	yes	-40至+85	8.7×3.7×2.5
EPSON	MC-3.6	±20	50	1.0	12.5	yes	-40至+85	8.0×3.8×2.54
Citizen	CM200S	±20	50	1.0	12.5	yes	-40至+85	8.0×3.8×2.5
KDS	DMX-26S	±30	50	1.0	12.5	yes	-40至+85	8.0×3.8×2.4

电源功耗

很多实时控制系统（RTCs）工作时由电池供电，在典型应用中，主电源关闭时，小锂电池用来使振荡器和时钟电路运行。为最大延长电池寿命，振荡器必须尽可能小的使用电源。为达到上述要求，需要对一些设计问题进行折中和优化。

负阻抗

对典型的高频振荡器电路，通常电路的等效串联电阻（ESR）设计 5 或者 10 倍裕度，低频晶体会使用更高的等效串联电阻（ESRs）。实时时钟（RTC）振荡器对负电阻少于 2 倍裕度。低裕度的振荡电路消耗更少的电流量，因此，实时时钟（RTC）振荡器对相对较少的杂散电流泄露、噪声或等效串联电阻（ESR）的阻值增加比较敏感。振荡器电容 C_L 影响功率消耗，内部负载 12.5pF 电容的实时时钟（RTC）比负载 6pF 电容功率消耗更大，但负载 12.5pF 电容的振荡器噪声敏感度较低。

晶体布局指南

因为实时时钟电路系统（RTCs）的晶体输入阻抗很高，所以晶体引脚可用作非常好的引线，耦合系统其他部分的高频信号。如果信号耦合到晶体引脚，能够抵消或者增加脉冲。因为多数板载信号工作频率在比 32.768kHz 的晶体频率更高，所以更有可能在不需要的情况下增加脉冲，这些噪声脉冲会产生额外的时钟节拍（ticks），使时钟加快。

下列步骤描述如何判断噪声是否导致实时时钟（RTC）加快：

1. 启动系统，同步实时时钟（RTC）与已知准确的时钟同步；
2. 关闭系统；
3. 关闭一段时间（2 小时，24 小时等），时间越长，更容易判断时钟是否准确；
4. 重启系统，读取时钟并与已知准确始终比较；
5. 将实时时钟（RTC）与已知准确时钟重新同步；
6. 保持系统打开，等待一段时间，该时间与步骤三的等待时间相同；
7. 上述时间过后读取时钟，与已知准确时钟相比较。

按照上述步骤，能够在系统启动和系统关闭时判断时钟的准确性。当系统启动时，判断出时钟是不准确的，但系统关闭时却是准确的，则问题极有可能是系统其它信号的噪声造成的。

然而，当系统启动和关闭时都不准确，则问题不是由系统噪声造成的。因为可能噪声耦合到晶体引脚上，所以应该关注额外的晶体在电路板上的布局，遵从电路板基本布局指南非常重要，晶体安放到电路板时要确保额外的时钟节拍不会耦合到晶体引脚上。

1. 晶体位置尽可能临近 X1 和 X2 非常重要，晶体和实时时钟（RTC）的连接线路导线要尽可能细，以减少因为缩短天线带来的噪声耦合，还能减少寄生电容数量。

2. 保持 X1 和 X2 引脚的晶体焊接线盘和线路宽度尽可能小，焊接线盘和线路越宽，越容易耦合相邻信号的噪声。

3. 有条件的话，在晶体周围放置保护环（接地），这能让晶体与相邻信号的噪声耦合绝缘，如图 2 所示显示了在晶体周围放置保护环。

4. 尽量确保没有其他板载信号线直接紧挨晶体或线路下方进入 X1 和 X2 引脚，晶体与电路板上的其他信号绝缘越好，耦合到晶体的噪声就会越少，任何数字信号线与任何连接到 X1 和 X2 的线路之间的距离最小为 0.200 英寸。实时时钟（RTC）应该与任何产生电磁辐射的器件绝缘(EMR)，离散式和模块式实时时钟系统（RTCs）遵从上述要求。

5. 在 PC 电路板上紧挨晶体放置局部接地层也会起到将晶体与电路板其它层信号的噪声耦合进行绝缘的作用。这里要注意接地层只能在晶体附近，不能覆盖整个电路板，图 5 所示为局部接地层示意。接地层外缘不能大于保护环境外缘。

因为寄生电容的原因，应该关注局部接地层的使用，线路/焊盘和接地层之间的电容连接到内部负载电容上（C_L1 和 C_L2），因此，当考虑增加局部接地层时必须考虑一些因素，例如，受接地层影响的电容，由下列方程式估算：

C=εA/t，式中

ε=电路板介电常数

A=线路/焊盘区域面积

t=PC 电路板层厚

而且，为确定接地层是否符合设计要求，上述参数必须考虑局部接地层电容不会大到使时钟变慢。

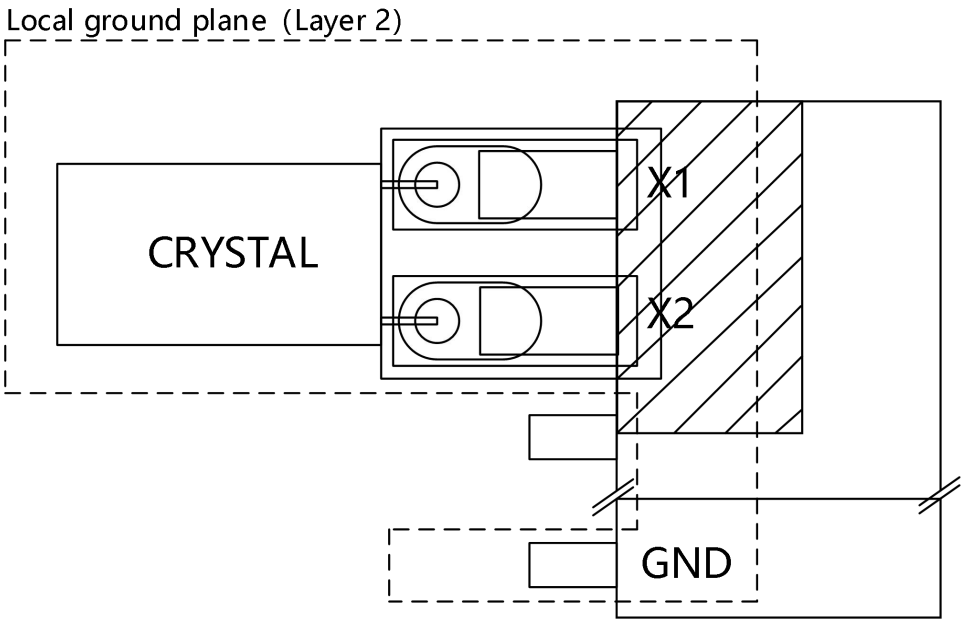


图 5. 推荐的晶体局部接地层

振荡器检查

设计者检查振荡器工作时需要测定一阶脉冲，检查时通常将接示波器探头连接到振荡器输入（X1）和输出（X2）引脚上。当时用实时时钟时，一般不推荐这种检查方法。因为振荡器设计在低功耗下工作（通过电池延长工作时间），将示波器探针连接到振荡器时可能会使振荡器停止工作。如果振荡器没有停止，

探针作为额外的负载会降低信号放大，可能造成振荡器不稳定，如改变振幅，应该通过间接的方法检查振荡器工作。

振幅可以用多种方法检测。其中一种方法就是多次读取秒寄存器，计算日期增加量。在具有振荡器停止标志（OSF）的实时时钟系统（RTCs）中，清除和监测停止位可以确认振荡器是否已经开始工作和是否连续工作。

如果设计者使用该方法检测遇到设计难题或不能与实时时钟进行通信的情况下，不能使用这些方法检测振荡器工作。检测实时时钟系统（RTCs）方波输出是一个可替代的检查方法，对比手册中的数据，检查实时时钟是否（RTC）首先处于写入状态，能够使振荡器和方波输出。注意，多数实时时钟（RTC）方波输出为开漏级输出，需要上拉电阻才能正常工作。方波输出可用来检测实时电路（RTC）准确度，但需要使用一个足够准确的频率计数器。

时钟变快

下列情况是多数常见的导致基于晶体的实时时钟运行加快的情况。

1. 相邻信号的晶体噪声耦合，这个问题在手册中多次提及，噪声耦合通常会导致实时时钟（RTC）极不准确。

2. 晶体与电容匹配不合适。如果与晶体匹配使用的负载电容(C_L)容量大于实时时钟（RTC）所需要的负载电容容量，在实时时钟（RTC）会显著加快。实时时钟不准确的重要原因是所依赖的电容 C_L 的电容量，例如，实时时钟上应使用 12pF 电容与晶体匹配，却设计使用了 6pF 电容，就会使实时时钟每个月快 3 到 4 分钟。

时钟变慢

下列情况是多数常见的导致基于晶体的实时时钟运行变慢的情况。

1. 实时时钟（RTC）输入引脚过冲。这可能会因为周期性振荡停止导致实时时钟（RTC）变慢，这可能是由输入到实时时钟(RTC)的噪声信号随机引发的。如果输入信号引起的电压升高到大于工作电压(V_{DD})的二极管压降，输入引脚的静电释放（ESD）保护二极管将正向偏置，允许电流涌入电路板基材，继而晶体停止震荡，直到输入信号电压降至低于二极管压降，二极管压降高于工作电压（V_{DD}）。

如果输入信号为噪声信号，这种工作模式会导致振荡器频繁停止振荡，而且应该注意确保输入信号上没有过冲。

当实时时钟（RTC）供电电源为备用电池时，相同过冲问题的其它情况是有一个 5V 电压输入到实时时钟（RTC），这是系统中正常切断电路某些部分但保持电路其他部分为开启状态时所引起的问题。非常重要的是要确保没有信号电压输入到实时时钟（RTC），当设备使用备用电池提供电源时，这个信号电压大于电池电压（除非另外说明）。

2. 晶体与电容匹配不合适。如果与晶体匹配使用的负载电容(C_L)容量小于实时时钟（RTC）所需要的负载电容容量，在实时时钟（RTC）会显著变慢。实时时钟不准确的重要原因是所依赖的电容 C_L 的电容量。

3. 寄生电容。晶体引脚和接地之间的寄生电容会导致实时时钟（RTC）变慢，而且应该注意当设计 PC 电路板布局时，确保寄生电容占用空间足够小。

4. 温度。工作温度特点表现为晶体转换温度，晶体振荡频率下降，则实时时钟越慢，见图 3、图 4 所示。

时钟不运行

以下是大多数情况是导致实时时钟不能运行的的原因：

1. 时钟不能运行的一个最常见的原因是不能设置 CH (时钟停止)或 EOSC (启用振荡器)位或者不能清除其错误。很多实时时钟系统（RTCs）包括其应用电路，在首次通电工作时保持振荡器正常工作，这样可以让系统等待用户加载设置，无需使用备用电源。当系统首次加电使用时，软件/固件必须使振荡器正常工作，提示用户设置正确的时间和日期。

2. 表面贴装晶体可能有一些 N.C. (无连接)引脚。务必要保证晶体引脚正确连接到 X1 和 X2 引脚。

晶体制造问题

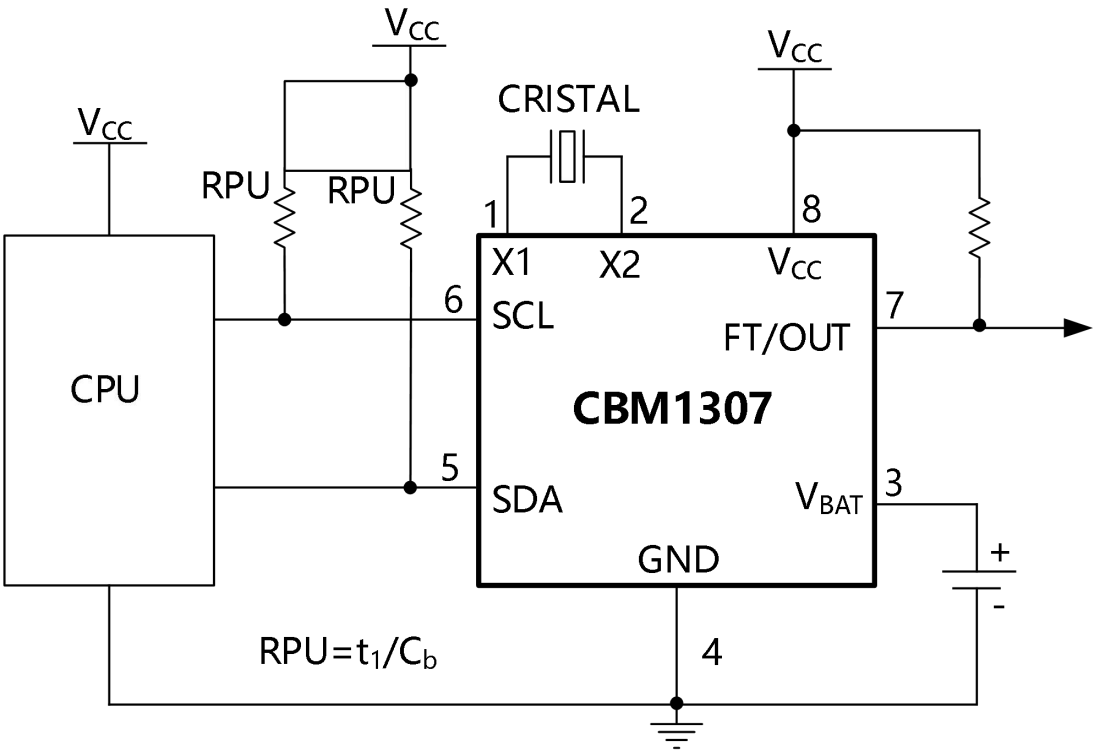
音叉晶振不能使用超声波清洗，很容易因为共振而损坏。

晶体不能暴露在温度高于最大额定参数的环境中，否则会损坏晶体，通常等效串联电阻（ESR）值会增加。晶体容器不能焊接到 PC 电路板上，否则有时会使晶体接地。直接焊接晶体外壳会让晶体单元超温。

实时时钟系统通常应该在无冷凝环境中使用，振荡器导体周围的水气环境会导致漏电，这会是振荡器停止工作。保形涂层用来保护电路，然后，图层会因为本身出现问题。有时候，保形涂层，尤其是使用环氧树脂会造成不能接受的离子污染。另外，如果 PC 电路板表面不能充分清洁尤其是保形涂层，保形涂层就会受到漏电与离子残留等污染。

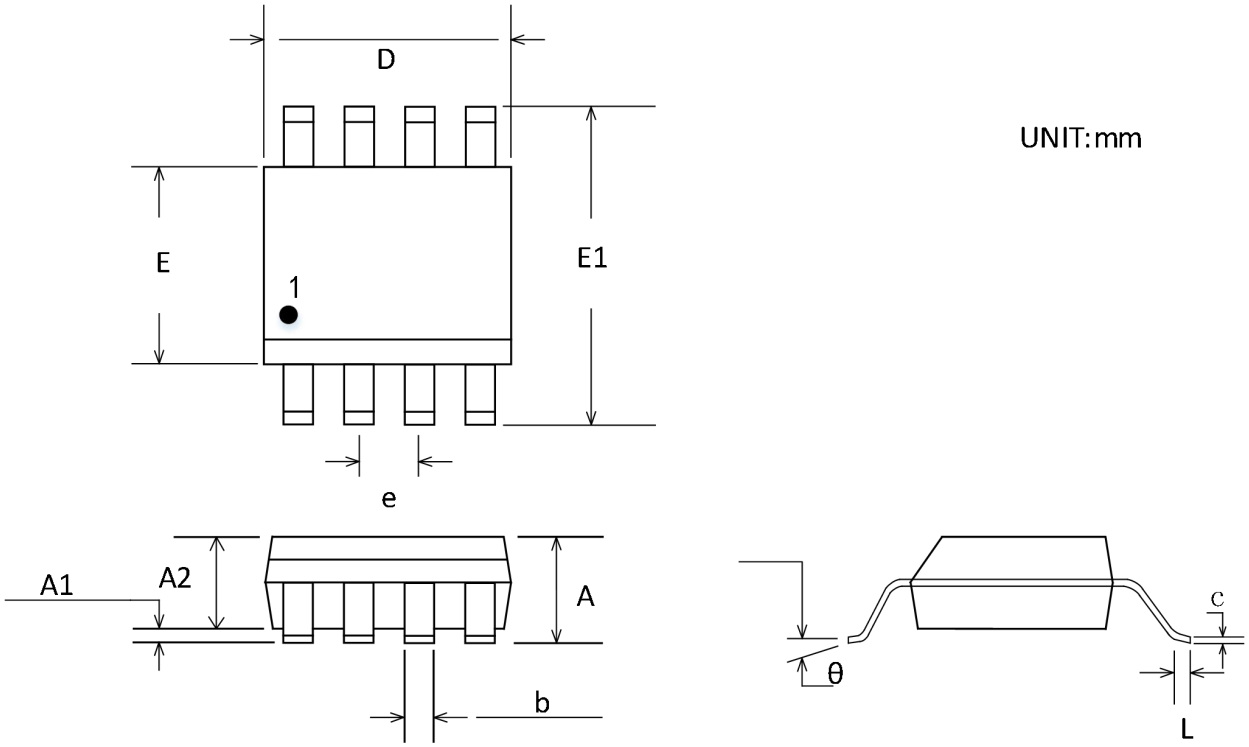
焊剂残留物会导致引脚间电流泄露。实时时钟振荡电路尤其容易漏电，电路在低功耗下运行。振荡器输入和输出的漏电，或漏电接地经常会让晶振停止工作。

典型应用电路


VCC
VCC
VCC
VCC
CRISTAL
RPU
RPU
1
2
8
X1
X2
VCC
6
SCL
7
FT/OUT
CPU
CBM1307
5
SDA
VBAT
3
GND
+
-
RPU=t1/Cb
4

封装尺寸及结构

SOP-8



UNIT:mm

符号	尺寸（mm）		尺寸（Inches）	
	最小值	最大值	最小值	最大值
A	1.350	1.750	0.053	0.069
A1	0.100	0.250	0.004	0.010
A2	1.350	1.550	0.053	0.061
b	0.330	0.510	0.013	0.020
c	0.170	0.250	0.007	0.010
D	4.800	5.000	0.189	0.197
E	3.800	4.000	0.150	0.157
E1	5.800	6.200	0.228	0.244
e	1.270 BSC		0.050 BSC	
L	0.400	1.270	0.016	0.050
θ	0°	8°	0°	8°

包装/订购信息

产品型号	温度范围	产品封装	丝印	包装数量
CBM1307AS8	-45℃ ~ 85℃	SOP-8	CBM1307	编带和卷盘,每卷 2500
CBM1307AS8-RL	-45℃ ~ 85℃	SOP-8	CBM1307	编带和卷盘,每卷 3000
CBM1307AS8-REEL	-45℃ ~ 85℃	SOP-8	CBM1307	编带和卷盘,每卷 4000

附录

1. 测试部件概述

芯佰微公司准备了测试元件，给 RTC 测试提供了便利的方法。测试元件由芯科科技的 MCU C8051F410 控制，测试者可以用这个部件进行实时时钟测试。

1.1. 硬件安装

CBM1363/CBM1307 是兼容实时时钟（RTC）的芯片。

作为一个应用实例，实例中声明了如何设置 RTC。

应用提示说明了如何将寄存器值设置为日期和时间，可以设置为以下格式：[YY]：[MM]：[DD]：[HH]：[MM]：[SS]。

1.2. 主要模块

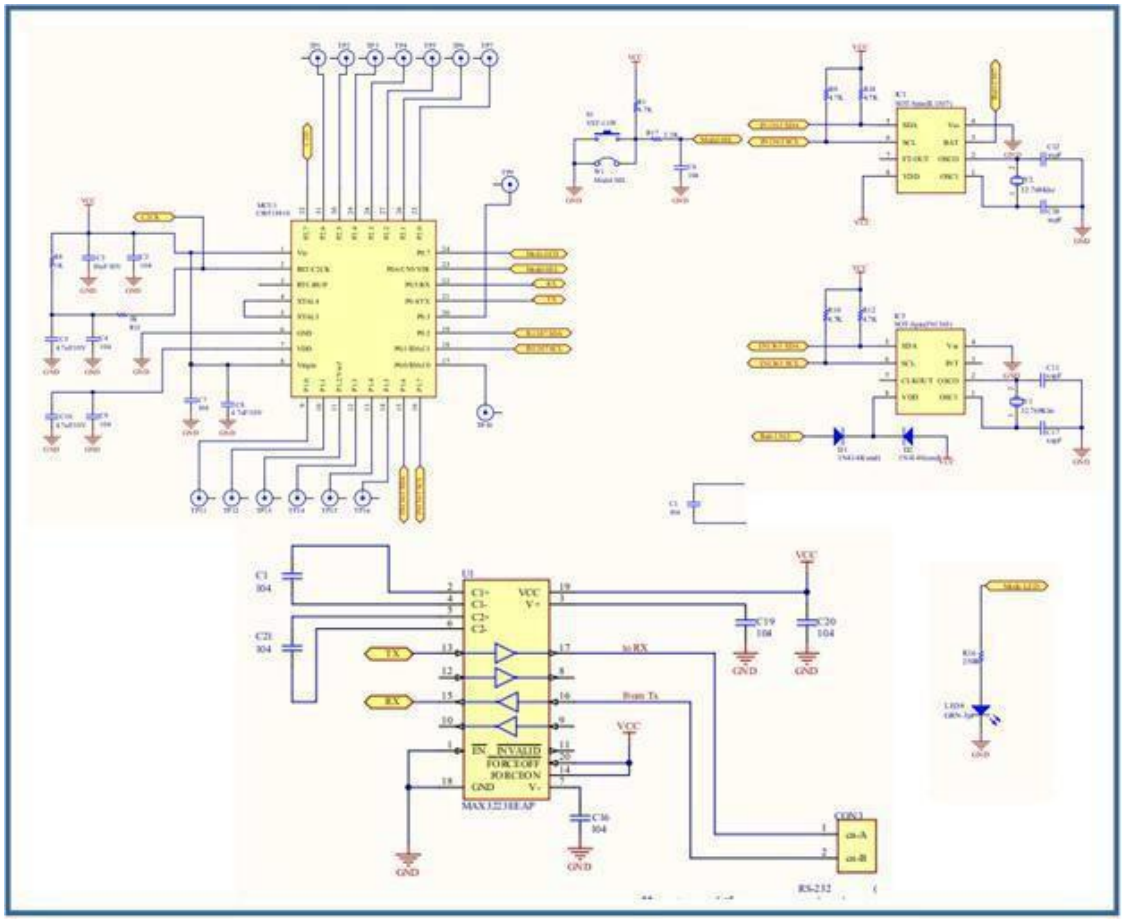


图 1. 测试板示意图

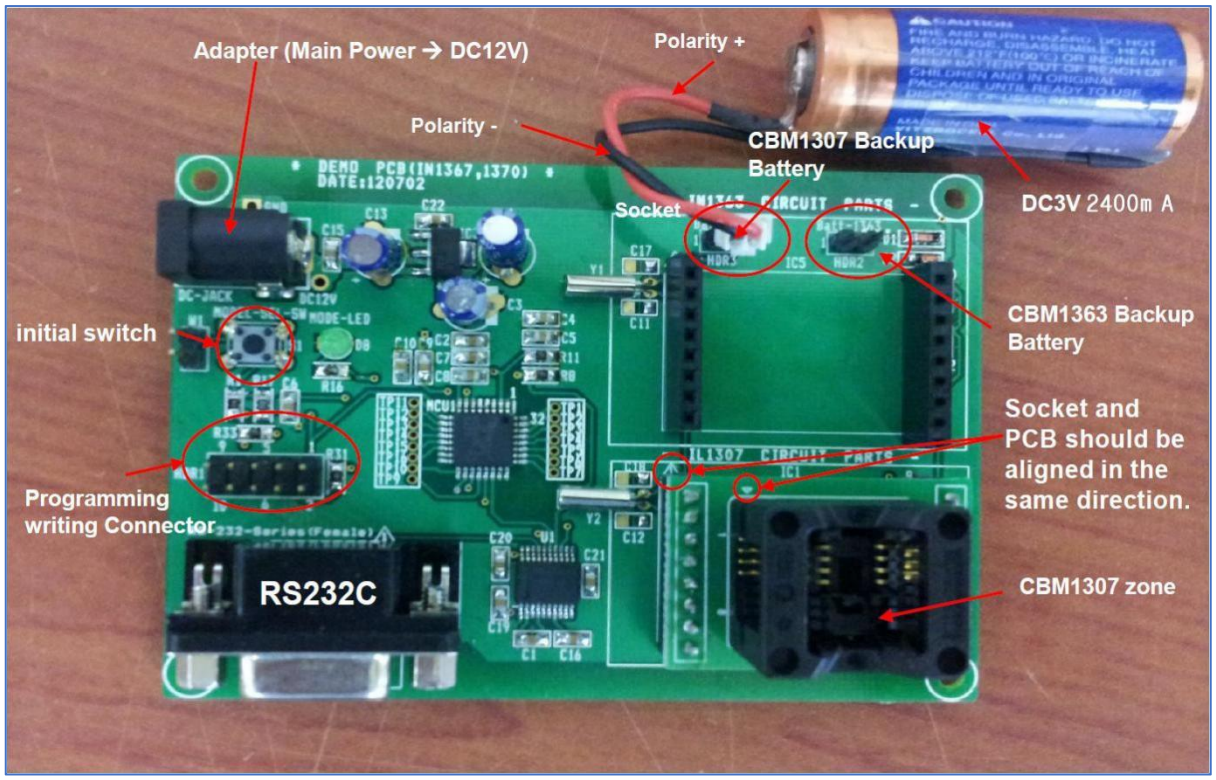

Adapter (Main Power → DC12V)
Polarity +
Polarity -
CBM1307 Backup Battery
DC3V 2400m A
Socket
initial switch
CBM1363 Backup Battery
Socket and PCB should be aligned in the same direction.
Programming writing Connector
RS232C
CBM1307 zone

图 2. 测试板顶视图

2.实时时钟程序

2.1. 流程图

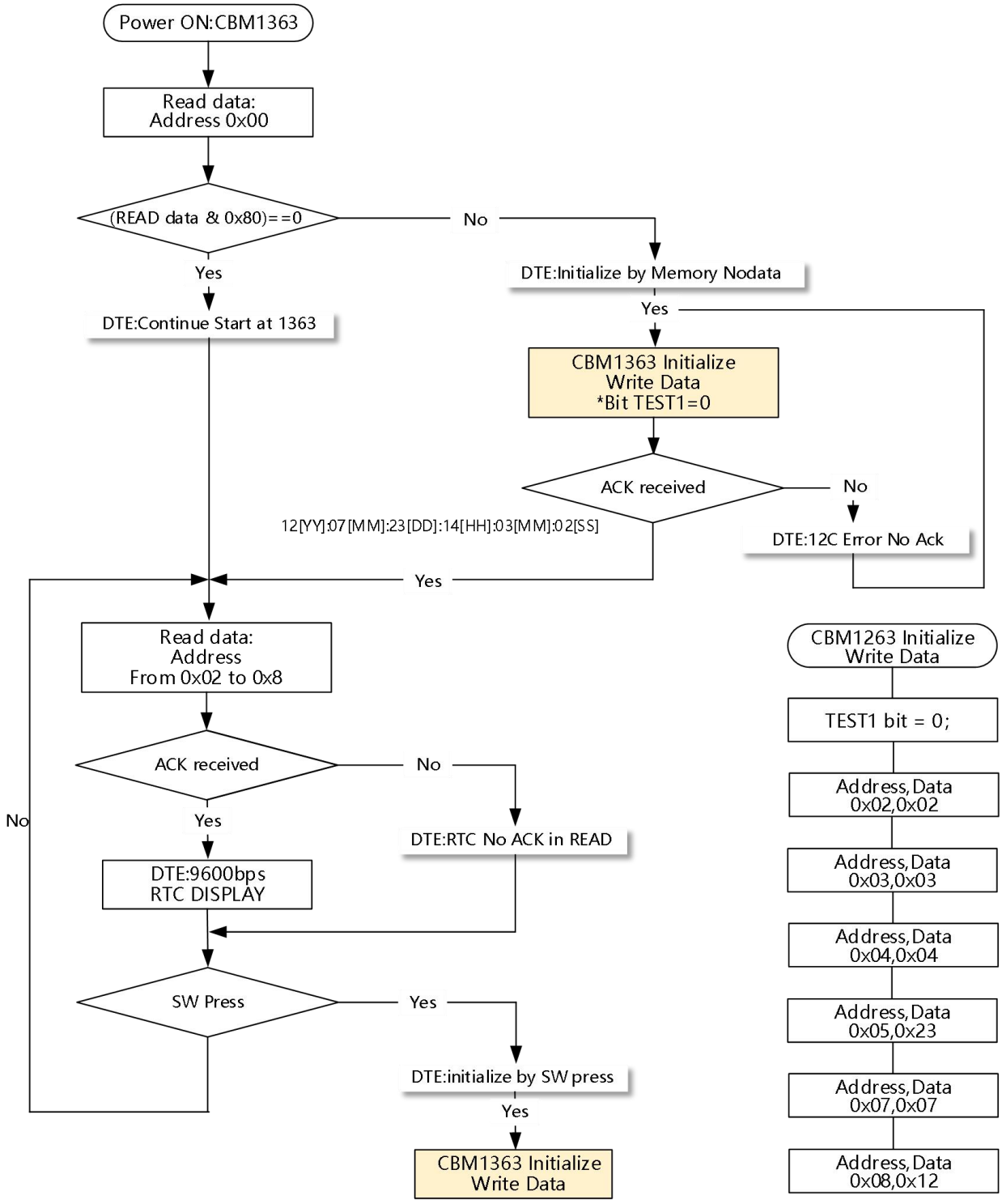


图 3. CBM1363 流程图

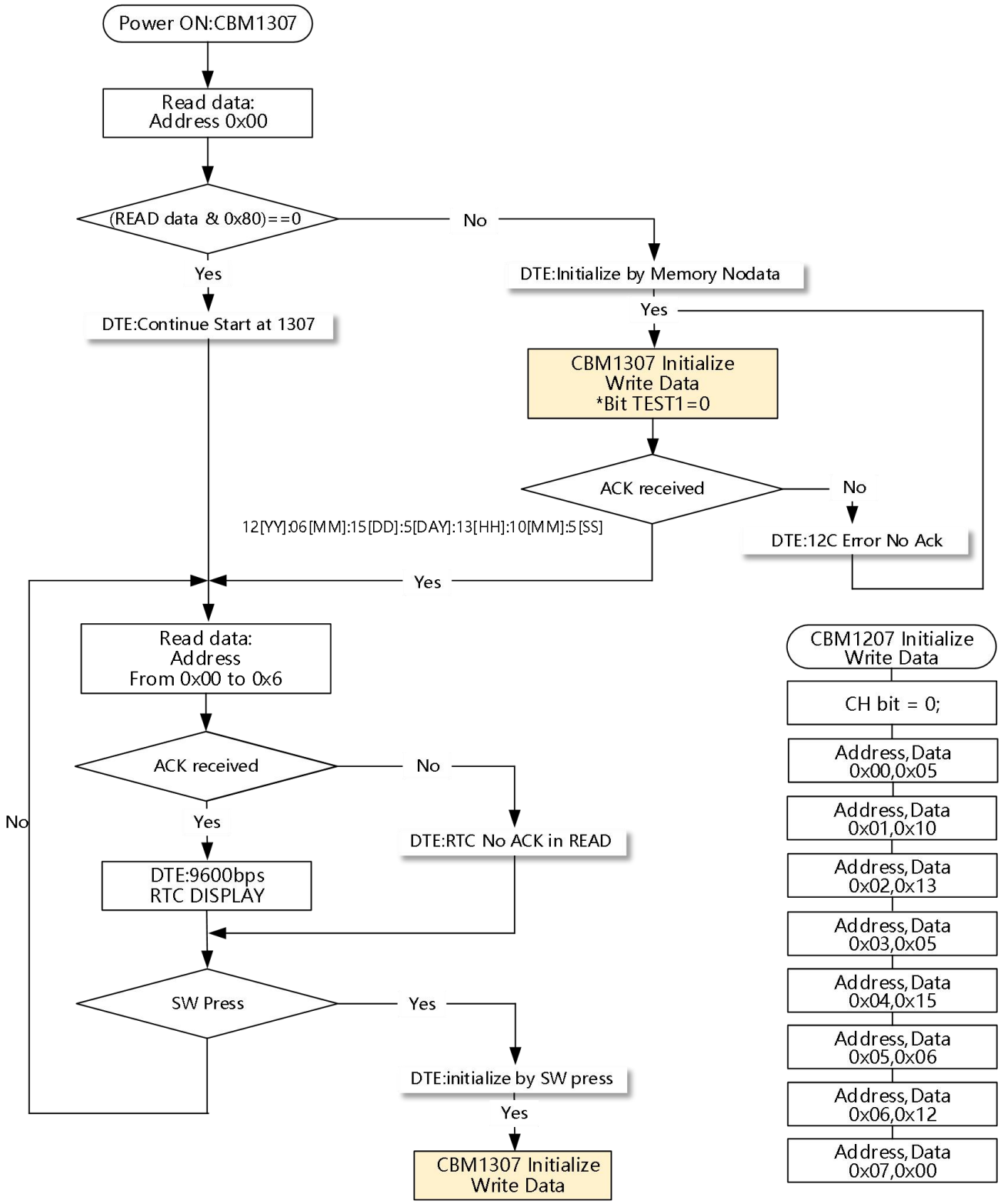


图 4. CBM1307 流程图

2.2. CBM1363 寄存器设置

2.2.1.初始化寄存器设置

进行初始化配置时，必须使发生器有效(bit TEST1 = 0)：

寄存器地址	寄存器名称	初始写入数据	备注
0x00H	控制/状态 1	0x00	TEST1 bit = 0 (正常模式) STOP bit = 0 (实时时钟源) TESTC bit = 0 (正常操作)
0x01H	控制/状态 2	0x00	AIE bit = 0 (时钟寄存器为整数型，0为无效) TIE bit = 0 (定时器寄存器为整数型，0为无效)
0x0DH	时钟设置控制		
0x0EH	时钟控制		
0x0FH	时钟		

2.2.2 时钟寄存器设置

寄存器地址	寄存器名称	初始写入数据	备注
0x02H	秒	0x02	
0x03H	分	0x03	
0x04H	小时	0x14	
0x05H	日期	0x23	
0x06H	星期		Sunday=0,Monday=1,Tuesday=2, Wednesday=3,..., Saturday=6
0x07H	世纪/月	0x07	月份/世纪寄存器为位7，置为“C”时指示为世纪，置为1时，为年（19xx），置为0时，为年（20xx）。
0x08H	年	0x12	
0x09H	分钟		
0x0AH	小时提示		
0x0BH	日期提示		
0x0CH	星期提示		

例如：如果想要设置日期 2012-6-23，当前时间为 13:10:05，然后需要进行下列命令设置

1. 秒设置: rtc 写入(SECONDS, 0x02);
2. 分设置 ：rtc 写入 (MINUTES, 0x03);
3. 小时设置 ：rtc 写入 (HOURS,0x14);
4. 日期设置 ：rtc 写入 (DATES, 0x23);
5. 月设置: rtc 写入 (MONTH, 0x07);
6. 年设置 ：rtc 写入 (YEAR, 0x12)

2.3 CBM1307 寄存器设置

2.3.1.初始化寄存器设置

进行初始化配置时，必须使发生器有效：

寄存器地址	寄存器名称	初始写入数据	备注
0x00H.Bit8	控制/状态 1	0x00	CH bit = 0 (正常模式)

2.3.2 Timer Register Setting Up

寄存器地址	寄存器名称	初始写入数据	备注

0x00H	秒	0x05	
0x01H	分	0x10	
0x02H	小时	0x13	
0x03H	日期	0x05	范围1-7
0x04H	日	0x15	
0x05H	月	0x06	
0x06H	年	0x12	
0x07H	控制		

例如：如果想要设置日期 2012-6-23，当前时间为 13:10:05，然后需要进行下列命令设置

1. 秒设置 : rtc 写入(0x00, 0x05);
2. 分设置 : rtc 写入(0x01, 0x10);
3. 小时设置 : rtc_写入(0x02, 0x13);
4. 日期设置 : rtc 写入(0x03, 0x05);
5. 日设置 : rtc 写入(0x04, 0x15);
6. 月设置 : rtc 写入(0x05, 0x06);
7. 年设置 : rtc 写入 (0x06 , 0x12);

2.4 超级终端配置

1. Bits/second : 9600 BPS
2. 数据位宽 ：8 bit
3. 奇偶校验位 ：None
4. 停止标志位 ：1 bit
5. 数据流控制: None

```
case RX_LOOP:
    if( Timer1000ms_f ){
        Timer1000ms_f = 0;

        if( rtc_read_7_byte(READ_ADDRESS)){
            printf("I2c RTC No ACK in READ\n");
            break;
        }

        format[11] = ((Receive_Data[6] & 0xf0) >> 4) + '0'; //year
        format[12] = (Receive_Data[6] & 0x0F) + '0';

        format[0] = ((Receive_Data[5] & 0x10) >> 4) + '0';  //month
        format[1] = (Receive_Data[5] & 0x0F) + '0';

        format[2] = ((Receive_Data[3] & 0x30) >> 4) + '0';  //Date
        format[3] = (Receive_Data[3] & 0x0F) + '0';

        format[5] = ((Receive_Data[2] & 0x30) >> 4) + '0';  //Hour
        format[6] = (Receive_Data[2] & 0x0F) + '0';

        format[7] = ((Receive_Data[1] & 0x70) >> 4) + '0';  //Min
        format[8] = (Receive_Data[1] & 0x0F) + '0';

        format[9] = ((Receive_Data[0] & 0x70) >> 4) + '0';  //Second
        format[10] = (Receive_Data[0] & 0x0f) + '0';


        printf (" %c%c[YY]:%c%c[MM]:%c%c[DD]:%c%c[HH]:%c%c[MM]:%c%c[SS]\n",
            format[11],format[12],format[0], format[1],format[2],
            format[3],format[5],format[6],format[7],
            format[8],format[9],format[10]);
```

图 5. CBM1363 测试程序源代码

串行终端设备工作并且确保比特率恰好为 9600bps。

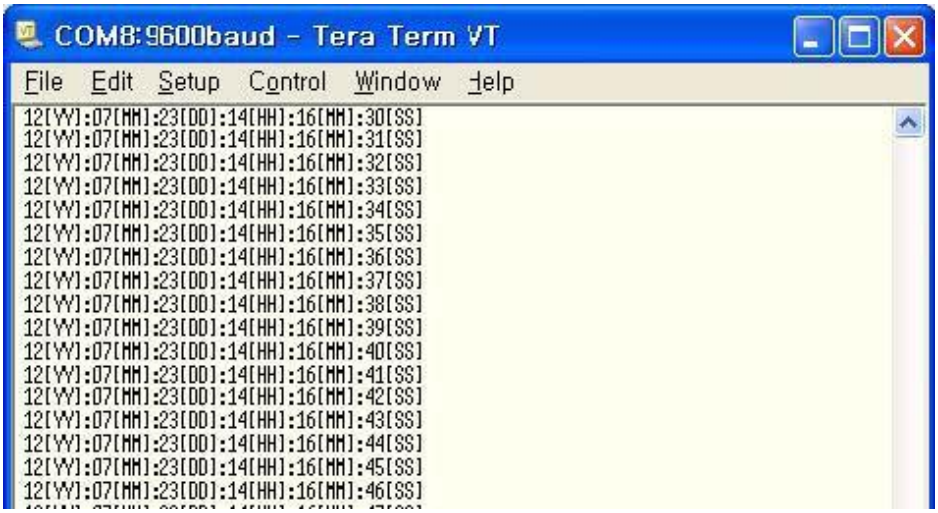


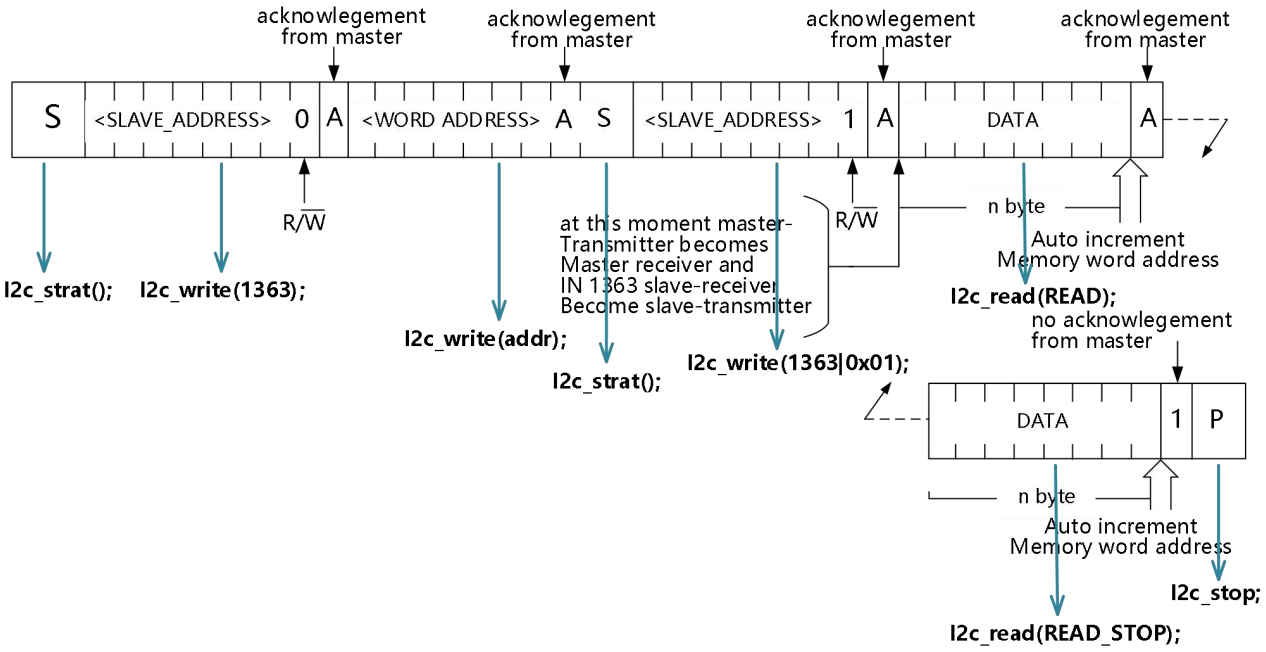
Fig6. CBM1363 RTC 串行终端设备数据

图 7. CBM1307 RTC 串行终端设备数据

3. RTC 源代码实例

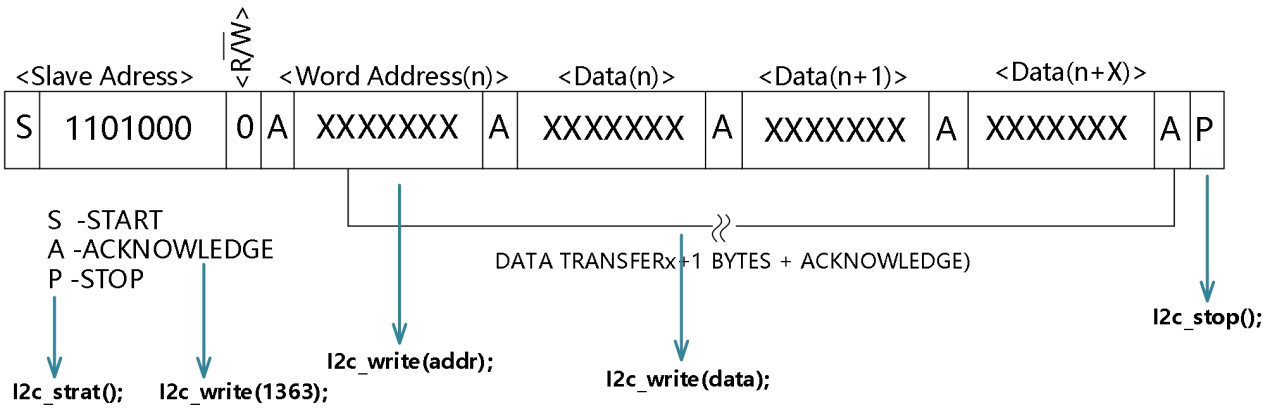
这部分文档内容是设置和读取/写入数据的主要功能实例。如果您需要源代码实例，我们会向您提供这些源代码。

3.1.从 CBM1363 & CBM1307 读取当前时间



Slave 地址：CBM1363 = 0xA2　　Slave 地址：CBM1307 = 0xD0

3.2. CBM1363 & CBM1307 时间设置写入


<Slave Adress>
<R/W>
<Word Address(n)>
<Data(n)>
<Data(n+1)>
<Data(n+X)>
S
1101000
0
A
XXXXXXX
A
XXXXXXX
A
XXXXXXX
A
XXXXXXX
A
P
S -START
A -ACKNOWLEDGE
P -STOP
DATA TRANSFERx+1 BYTES + ACKNOWLEDGE)
I2c_strat();
I2c_write(1363);
I2c_write(addr);
I2c_write(data);
I2c_stop();

3.3. CBM1363 & CBM1307 I2C（串行传输总线）源代码实例

```
#ifdef CBM1363
	sbit SCL = P1^7;
	sbit SDA = P1^6; //
#endif
#ifdef CBM1307
	sbit SDA = P0^2;
	sbit SCL = P0^1;
#endif
void i2c_start(void)		// START condition
{
	SDA = HIGH;
	DelayTimeLoop();
	SCL = HIGH;
	DelayTimeLoop();
	SDA = LOW;
	DelayTimeLoop();
	SCL = LOW;
	DelayTimeLoop();
}
void i2c_stop(void)		// STOP condition
{
	SCL = HIGH;
	DelayTimeLoop();
	SDA = HIGH;
	DelayTimeLoop();
}
/* Clock pulse generation. The function returns data or acknowledgment bit */
unsigned char i2c_clock(void)//bit i2c_clock(void)
{
	bit level;		// state of SDA line
	SCL = 1;
	DelayTimeLoop();
	while (!SCL);	// if a pulse was stretched
	DelayTimeLoop();
	level = SDA;
	DelayTimeLoop();
	SCL = 0;
	return (level);
```

```
}
/* Writing a byte 至 a slave, with most significant bit first. The function returns acknowledgment bit.*/
unsigned char i2c_write(unsigned char byte)
{
	unsigned char mask = 0x80;
	unsigned char aaa;
	while (mask) {
		if (byte & mask)
			SDA = 1;
		else
			SDA = 0;
		i2c_clock();
		mask >>= 1;
}
aaa = i2c_clock();
return (aaa);
}
/* Reading byte from a slave, with most significant bit first. The parameter indicates, whether 至 acknowledge (1) or not (0) */
unsigned char i2c_read(unsigned char acknowledgment)
{
	uchar mask = 0x80, byte = 0x00;
	while (mask) {
		if (i2c_clock())
			byte |= mask;
		mask >>= 1; /* next bit 至 receive */
	}
	if (acknowledgment) {
		SDA = 0;
		i2c_clock();
		SDA = 1;
	}
	else {
		SDA = 1;
		i2c_clock();
	}
	return (byte);
}
```

```
unsigned char rtc_read_7_byte(unsigned char addr)
{
        unsigned char status , i;
        i2c_start();
        if (!i2c_write(0xA2 )){//0xA2 is Slave Address for CBM1363, 0xD2 is for CBM1307
                DelayTimeLoop();
                if (!i2c_write(addr)){
                        i2c_start();
                        if (!i2c_write(0xA2 | 0x01)){ CBM1307’ s slave address is 0xD2
                                for(i=0;i<6;i++){
                                        Receive_Data[i] = i2c_read(1);
                                }
                                Receive_Data[6] = i2c_read(0);
                        }
                        else {
                                status = 1;
                        }
                        else{
                        status = 1;
                        }
                }
                else   status = 1;
                i2c_stop(); return(status);
}
```

4. RTC 测试过程

4.1. CBM1363 的测试过程

1．务必确定测试元件为断电状态。

2．将 CBM1363 接入测试元件 CBM1363 的接口上。

3．将电池连接到 CBM1363 电池接口上。

4．串行终端设备工作并且确保比特率恰好为 9600bps。

5．打开测试元件的开关，确认绿色 LED 状态指示灯是否闪烁。如果每隔 1 秒钟闪烁一次，说明 CBM1307 已经开始工作。编程方法参见第 2 章。

6．如图 6 在串行终端设备窗口显示的数据，参考图 6。

7．然后测试元件关闭。

8．等待时间（2 个小时，10 分钟，等等）。

9．重启测试元件，检查测试时间，即设备关闭时间。

4.2. CBM1307 的测试过程

1．务必确定测试元件为断电状态。

2．将 CBM1363 接入测试元件 CBM1307 的接口上。

3．将电池连接到 CBM1307 电池接口上。

4．串行终端设备工作并且确保比特率恰好为 9600bps。

5．打开测试元件的开关，确认绿色 LED 状态指示灯是否闪烁。如果每隔 1 秒钟闪烁一次，说明 CBM1307 已经开始工作。编程方法参见第 2 章。

6．如图 7 在串行终端设备窗口显示的数据，参考图 7。

7．然后测试元件关闭。

8．等待时间（2 个小时，10 分钟，等等）。

9．重启测试元件，检查测试时间，即设备关闭时间。