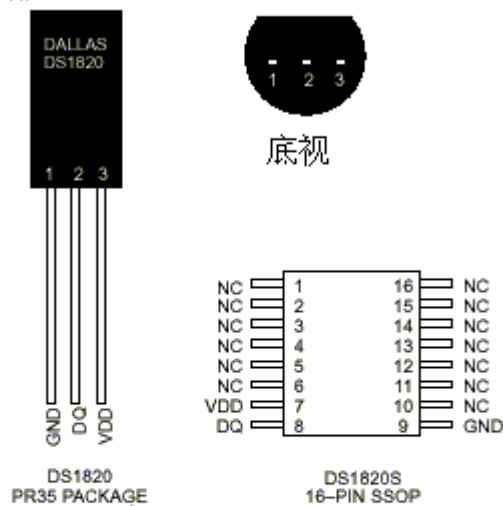


DS1820 单线数字温度计

特性

- 独特的单线接口仅需一个端口引脚进行通讯
- 简单的多点分布应用
- 无需外部器件
- 可通过数据线供电
- 零待机功耗
- 测温范围-55~+125 ，以 0.5 递增。华氏器件-67~+257°F，以 0.9°F 递增
- 温度以 9 位数字量读出
- 温度数字量转换时间 200ms（典型值）
- 用户可定义的非易失性温度报警设置
- 报警搜索命令识别并标志超过程序限定温度（温度报警条件）的器件
- 应用包括温度控制、工业系统、消费品、温度计或任何热感测系统

引脚排列



引脚说明

GND	- 地
DQ	- 数据I/O
V _{DD}	- 可选VDD
NC	- 空脚

说明

DS1820 数字温度计以 9 位数字量的形式反映器件的温度值。

DS1820 通过一个单线接口发送或接收信息，因此在中央微处理器和 DS1820 之间仅需一条连接线（加上地线）。用于读写和温度转换的电源可以从数据线本身获得，无需外部电源。因为每个 DS1820 都有一个独特的片序列号，所以多只 DS1820 可以同时连在一根单线总线上，这样就可以把温度传感器放在许多不同的地方。这一特性在 HVAC 环境控制、探测建筑物、仪器或机器的温度以及过程监测和控制等方面非常有用。

引脚说明

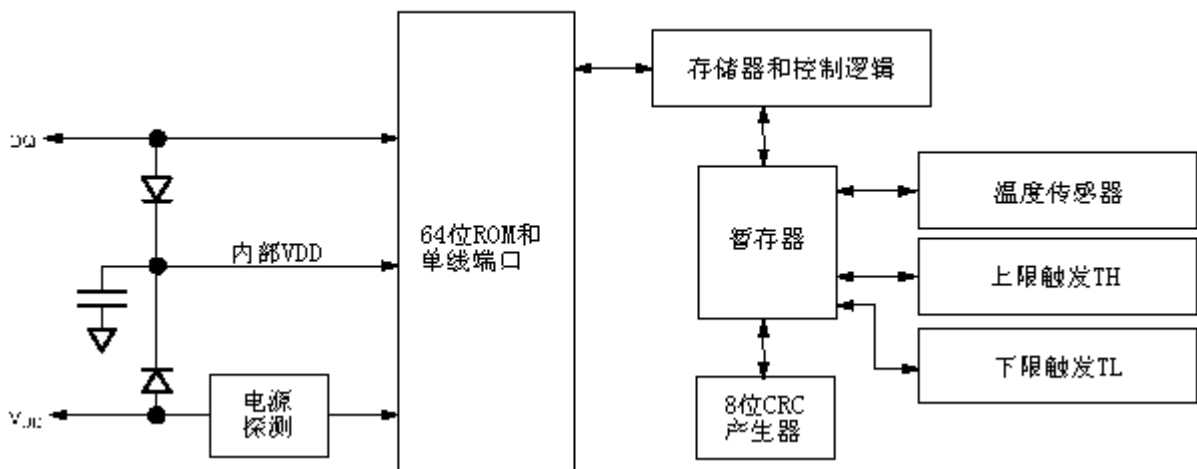
16 脚 SSOP	PR35	符号	说明
9	1	GND	接地
8	2	DQ	数据输入/输出脚。对于单线操作：漏极开路（见“寄生电源”节）
7	3	VDD	可选的 VDD 引脚。具体接法见“寄生电源”节

DS1820S (16 脚 SSOP)：所有上表中未提及的引脚都无连接。

概览

图 1 的方框图示出了 DS1820 的主要部件。DS1820 有三个主要数字部件：1) 64 位激光 ROM，2) 温度传感器，3) 非易失性温度报警触发器 TH 和 TL。器件用如下方式从单线通讯线上汲取能量：在信号线处于高电平期间把能量储存在内部电容里，在信号线处于低电平期间消耗电容上的电能工作，直到高电平到来再给寄生电源（电容）充电。DS1820 也可用外部 5V 电源供电。

DS1820 方框图 (图1)



DS1820 依靠一个单线端口通讯。在单线端口条件下，必须先建立 ROM 操作协议，才能进行存储器和控制操作。因此，控制器必须首先提供下面 5 个 ROM 操作命令之一：1) 读 ROM，2) 匹配 ROM，3) 搜索 ROM，4) 跳过 ROM，5) 报警搜索。这些命令对每个器件的激光 ROM 部分进行操作，在单线总线上挂有多个器件时，可以区分出单个器件，同时可以向总线控制器指明有多少器件或是什么型号的器件。成功执行完一条 ROM 操作序列后，即可进行存储器和控制操作，控制器可以提供 6 条存储器和控制操作指令中的任一条。

一条控制操作命令指示 DS1820 完成一次温度测量。测量结果放在 DS1820 的暂存器里，用一条读暂存器内容的存储器操作命令可以把暂存器中数据读出。温度报警触发器 TH 和 TL 各由一个 EEPROM 字节构成。如果没有对 DS1820 使用报警搜索命令，这些寄存器可以作为一般用途的用户存储器使用。可以用一条存储器操作命令对 TH 和 TL 进行写入，对这些寄存器的读出需要通过暂存器。所有数据都是以最低有效位在前的方式进行读写。

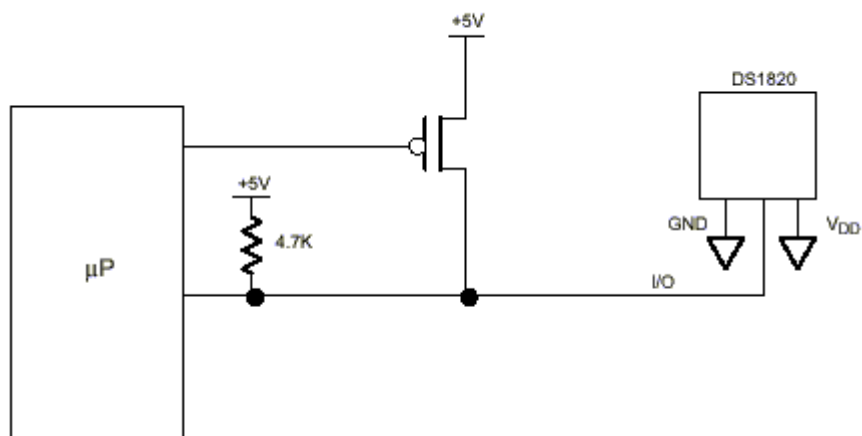
寄生电源

寄生电源的方框图见图 1。这个电路会在 I/O 或 VDD 引脚处于高电平时“偷”能量。当有特定的时间和电压需求时（见节标题“单线总线系统”），I/O 要提供足够的能量。寄生电源有两个好处：1) 进行远距离测温时，无需本地电源，2) 可以在没有常规电源的条件下读 ROM。要想使 DS1820 能够进行精确的温度转换，I/O 线必须在转换期间保证供电。由于 DS1820 的工作电流达到 1mA，所以仅靠 5K 上拉电阻提供电源是不行的，当几只 DS1820 挂在同一根 I/O 线上并同时想进行温度转换时，这个问题变得更加尖锐。

有两种方法能够使 DS1820 在动态转换周期中获得足够的电流供应。第一种方法，当进行温度转换或拷贝到 E² 存储器操作时，给 I/O 线提供一个强上拉。用 MOSFET 把 I/O 线直接拉到电源

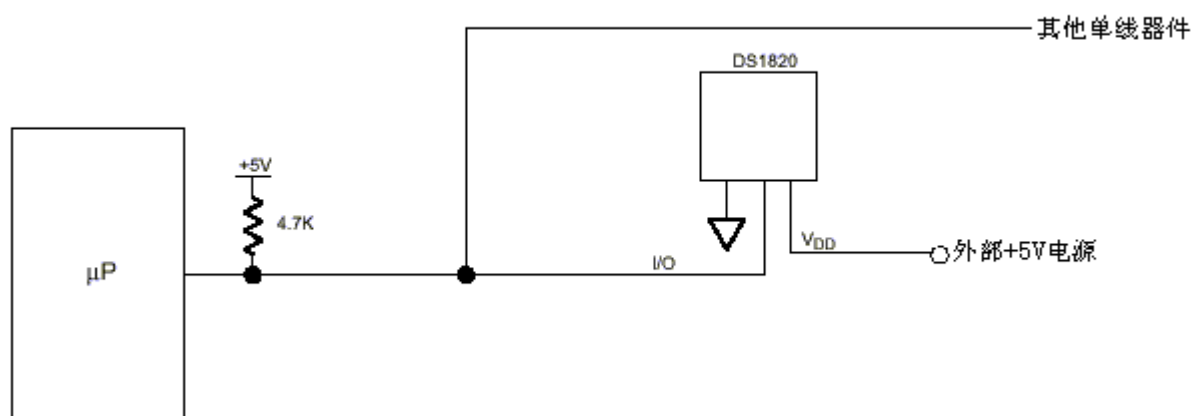
上就可以实现，见图 2。在发出任何涉及拷贝到 E² 存储器或启动温度转换的协议之后，必须在最多 10 μs 之内把 I/O 线转换到强上拉。使用寄生电源方式时，VDD 引脚必须接地。

DS1820 温度转换期间的强上拉供电 (图 2)



另一种给 DS1820 供电的方法是从 VDD 引脚接入一个外部电源，见图 3。这样做的好处是 I/O 线上不需要加强上拉，而且总线控制器不用在温度转换期间总保持高电平。这样在转换期间可以允许在单线总线上进行其他数据往来。另外，在单线总线上可以挂任意多片 DS1820，而且如果它们都使用外部电源的话，就可以先发一个 Skip ROM 命令，再接一个 Convert T 命令，让它们同时进行温度转换。注意当加上外部电源时，GND 引脚不能悬空。

用 VDD 供电 (图 3)



温度高于 100 时，不推荐使用寄生电源，因为 DS1820 在这种温度下表现出的漏电流比较大，通讯可能无法进行。在类似这种温度的情况下，强烈推荐使用 DS1820 的 VDD 引脚。

对于总线控制器不知道总线上的 DS1820 是用寄生电源还是用外部电源的情况，DS1820 预备了一种信号指示电源的使用意图。总线控制器发出一个 Skip ROM 协议，然后发出读电源命令，这条命令发出后，控制器发出读时间隙，如果是寄生电源，DS1820 在单线总线上发回“0”，如果是从 VDD 供电，则发回“1”，这样总线控制器就能够决定总线上是否有 DS1820 需要强上拉。如果控制器接收到一个“0”，它就知道必须在温度转换期间给 I/O 线提供强上拉。这个命令协议详见“存储器操作命令”节。

测温操作

DS1820 通过一种片上温度测量技术来测量温度。图 4 示出了温度测量电路的方框图。

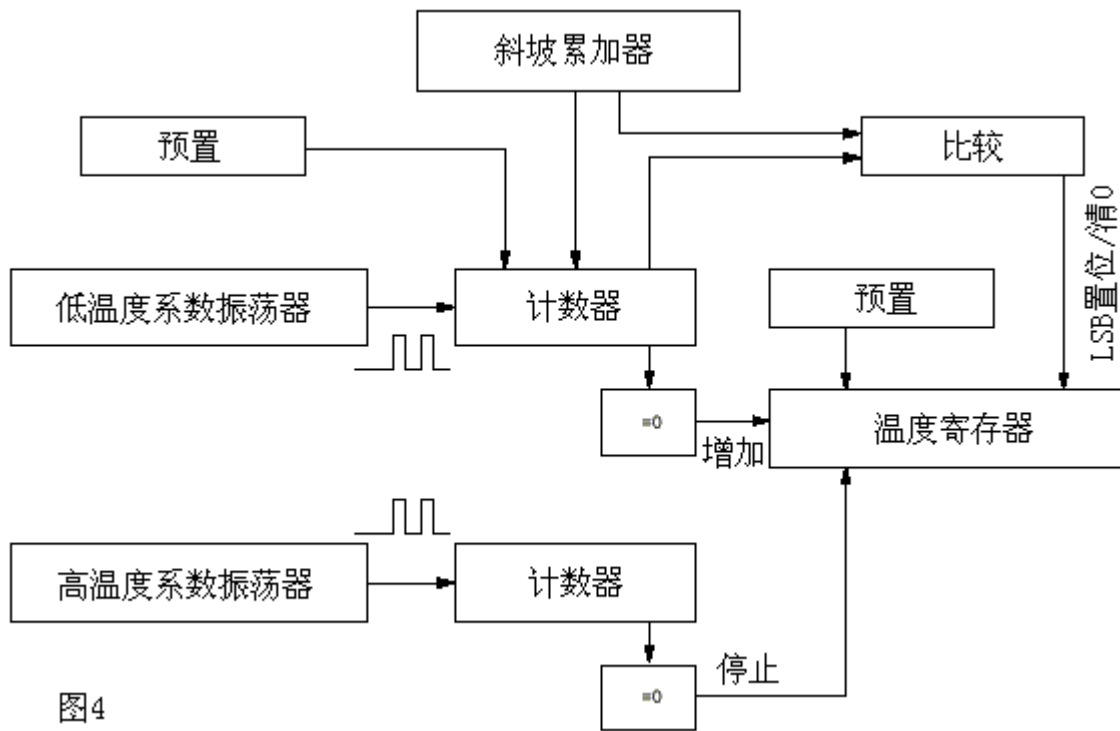


图4

温度/数据关系 (表 1)

温度	数据输出 (二进制)	数据输出 (十六进制)
+125	00000000 11111010	00FA
+25	00000000 00110010	0032
+1/2	00000000 00000001	0001
0	00000000 00000000	0000
-1/2	11111111 11111111	FFFF
-25	11111111 11001110	FFCE
-55	11111111 10010010	FF92

DS18B20 是这样测温的：用一个高温系数的振荡器确定一个门周期，内部计数器在这个门周期内对一个低温系数的振荡器的脉冲进行计数来得到温度值。计数器被预置到对应于-55 的一个值。如果计数器在门周期结束前到达 0，则温度寄存器（同样被预置到-55 ）的值增加，表明所测温度大于-55 。

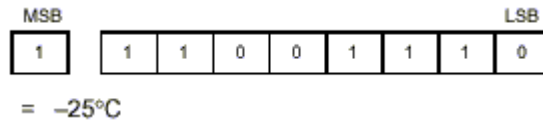
同时，计数器被复位到一个值，这个值由斜坡式累加器电路确定，斜坡式累加器电路用来补偿感温振荡器的抛物线特性。然后计数器又开始计数直到 0，如果门周期仍未结束，将重复这一过程。

斜坡式累加器用来补偿感温振荡器的非线性，以期在测温时获得比较高的分辨力。这是通过改变计数器对温度每增加一度所需计数的值来实现的。因此，要想获得所需的分辨力，必须同时知道在给定温度下计数器的值和每一度的计数值。

DS18B20 内部对此计算的结果可提供 0.5 的分辨力。温度以 16bit 带符号位扩展的二进制补码形式读出，表 1 给出了温度值和输出数据的关系。数据通过单线接口以串行方式传输。

DS18B20 测温范围-55 ~+125 ，以 0.5 递增。如用于华氏温度，必须要用一个转换因子查找表。

注意 DS18B20 内温度表示值为 1/2 LSB，如下所示 9bit 格式：



最高有效（符号）位被复制充满存储器中两字节温度寄存器的高 MSB 位，由这种“符号位扩展”产生出了示于表 1 的 16bit 温度读数。

可用下述方法获得更高的分辨力。首先，读取温度值，将 0.5 位（LSB）从读取的值中截去，这个值叫做 TEMP_READ。然后读取计数器中剩余的值，这个值是门周期结束后保留下来的值（COUNT_REMAIN）。最后，我们用到在这个温度下每度的计数值（COUNT_PER_C）。用户可以用下面的公式计算实际温度值：

$$\text{TEMPERATURE} = \text{TEMP_READ} - 0.25 \frac{(\text{COUNT_PER_C} - \text{COUNT_REMAIN})}{\text{COUNT_PER_C}}$$

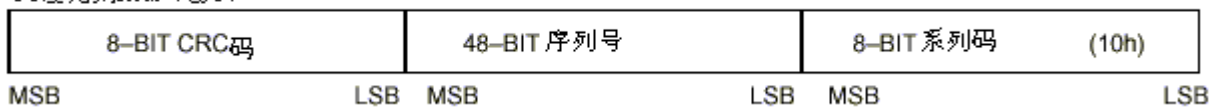
报警搜索操作

DS1820 完成一次温度转换后，就拿温度值和存储在 TH 和 TL 中的值进行比较。因为这些寄存器是 8 位的，所以 0.5 位被忽略不计。TH 或 TL 的最高有效位直接对应 16 位温度寄存器的符号位。如果测得的温度高于 TH 或低于 TL，器件内部就会置位一个报警标识。每进行一次测温就对这个标识进行一次更新。当报警标识置位时，DS1820 会对报警搜索命令有反应。这样就允许许多 DS1820 并联在一起同时测温，如果某个地方的温度超过了限定值，报警的器件就会被立即识别出来并读取，而不用读未报警的器件。

64 位（激）光刻 ROM

每只 DS1820 都有一个唯一的长达 64 位的编码。最前面 8 位是单线系列编码（DS1820 的编码是 19h）。下面 48 位是一个唯一的序列号。最后 8 位是以上 56 位的 CRC 码。（见图 5）64 位 ROM 和 ROM 操作控制区允许 DS1820 做为单线制器件并按照详述于“单线总线系统”一节的单线协议工作。只有建立了 ROM 操作协议，才能对 DS1820 进行控制操作。这个协议用 ROM 操作流程图来描述（图 6）。单线总线控制器必须得天独厚提供 5 个 ROM 操作命令其中之一：1) Read ROM, 2) Match ROM, 3) Search Rom, 4) Skip ROM, 5) Alarm Search。成功进行一次 ROM 操作后，就可以对 DS1820 进行特定的操作，总线控制器可以发出六个存储器和控制操作命令中的任一个。

64位光刻ROM（图5）



CRC 发生器

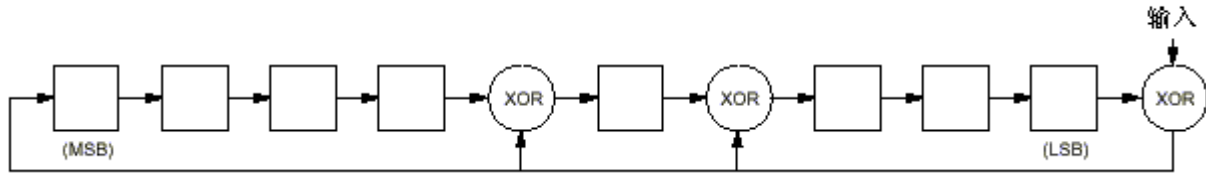
DS1820 中有 8 位 CRC 存储在 64 位 ROM 的最高有效字节中。总线控制器可以用 64 位 ROM 中的前 56 位计算出一个 CRC 值，再用这个和存储在 DS1820 中的值进行比较，以确定 ROM 数据是否被总线控制器接收无误。CRC 计算等式如下：

$$\text{CRC} = X^8 + X^5 + X^4 + 1$$

DS1820 同样用上面的公式产生一个 8 位 CRC 值，把这个值提供给总线控制器用来校验传输的数据。在任何使用 CRC 进行数据传输校验的情况下，总线控制器必须用上面的公式计算出一个 CRC 值，和存储在 DS1820 的 64 位 ROM 中的值或 DS1820 内部计算出的 8 位 CRC 值（当读暂存器时，做为第 9 个字节读出来）进行比较。CRC 值的比较以及是否进行下一步操作完全由总线控制器决定。当在 DS1820 中存储的或由其计算的 CRC 值和总线控制器计算的值不相符时，DS1820 内部并没有一个能阻止命令序列进行的电路。

单线 CRC 可以用一个由移位寄存器和 XOR 门构成的多项式发生器来产生，见图 7。

单线CRC码 (图7)

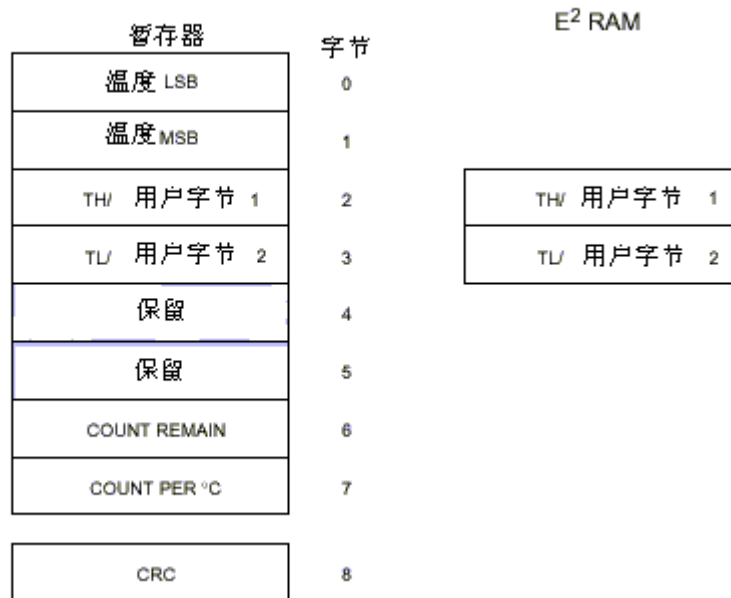


移位寄存器的各位都被初始化为 0。然后从系列编号的最低有效位开始，一次一位移入寄存器，8 位系列编码都进入以后，序列号再进入，48 位序列号都进入后，移位寄存器中就存储了 CRC 值。移入 8 位 CRC 会使移位寄存器复 0。

存储器

DS1820 的存储器结构示于图 8。存储器由一个暂存 RAM 和一个存储高低温报警触发值 TH 和 TL 的非易失性电可擦除 (E²) RAM 组成。当在单线总线上通讯时，暂存器帮助确保数据的完整性。数据先被写入暂存器，这里的数据可被读回。数据经过校验后，用一个拷贝暂存器命令会把数据传到非易性 (E²) RAM 中。这一过程确保更改存储器时数据的完整性。

DS1820存储器图 (图8)



暂存器的结构为 8 个字节的存储器。头两个字节包含测得的温度信息。第三和第四字节是 TH 和 TL 的拷贝，是易失性的，每次上电复位时被刷新。下面两个字节没有使用，但是在读回数据时，它们全部表现为逻辑 1。第七和第八字节是计数寄存器，它们可以被用来获得更高的温度分辨力 (见“测温操作”一节)。

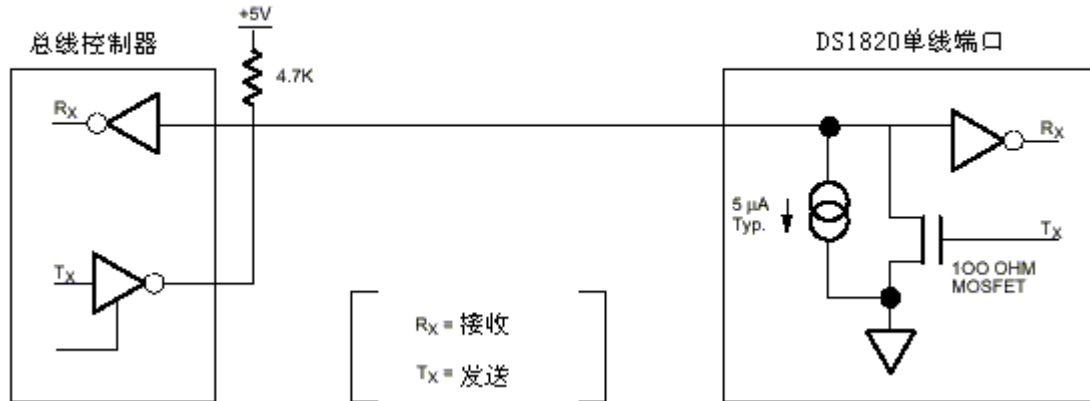
还有一个第九字节，可以用读暂存器命令读出。这个字节是以上八个字节的 CRC 码。CRC 的执行方式如第二个标题“CRC 发生器”所述。

单线总线系统

单线总线系统包括一个总线控制器和一个或多个从机。DS1820 是从机。关于这种总线分三个题目讨论：硬件结构、执行序列和单线信号 (信号类型和时序)。

单线总线只有一条定义的信号线；重要的是每一个挂在总线上的器件都能在适当的时间驱动它。为此每一个总线上的器件都必须是漏极开路或三态输出。DS1820 的单总线端口 (I/O 引脚) 是漏极开路式的，内部等效电路见图 9。一个多点总线由一个单线总线和多个挂于其上的从机构成。单线总线需要一个约 5K 的上拉电阻。

硬件结构 (图9)



单线总线的空闲状态是高电平。无论任何理由需要暂停某一执行过程时，如果还想恢复执行的话，总线必须停留在空闲状态。在恢复期间，如果单线总线处于非活动（高电平）状态，位与位间的恢复时间可以无限长。如果总线停留在低电平超过 $480\ \mu\text{s}$ ，总线上的所有器件都将被复位。

执行序列

通过单线总线端口访问 DS1820 的协议如下：

- 初始化
- ROM 操作命令
- 存储器操作命令
- 执行/数据

初始化

通过单线总线的所有执行（处理）都从一个初始化序列开始。初始化序列包括一个由总线控制器发出的复位脉冲和跟有其后由从机发出的存在脉冲。

存在脉冲让总线控制器知道 DS1820 在总线上且已准备好操作。详见“单线信号”节。

ROM 操作命令

一旦总线控制器探测到一个存在脉冲，它就可以发出 5 个 ROM 命令中的任一个。所有 ROM 操作命令都 8 位长度。下面是这些命令（参见图 6 流程图）：

[ROM 操作流程图 \(图 6\)](#)

Read ROM [33h]

这个命令允许总线控制器读到 DS1820 的 8 位系列编码、唯一的序列号和 8 位 CRC 码。只有在总线上存在单只 DS1820 的时候才能使用这个命令。如果总线上不止一个从机，当所有从机试图同时传送信号时就会发生数据冲突（漏极开路连在一起开成相与的效果）。

Match ROM [55h]

匹配 ROM 命令，后跟 64 位 ROM 序列，让总线控制器在多点总线上定位一只特定的 DS1820。只有和 64 位 ROM 序列完全匹配的 DS1820 才能响应随后的存储器操作命令。所有和 64 位 ROM 序列不匹配的从机都将等待复位脉冲。这条命令在总线上有单个或多个器件时都可以使用。

Skip ROM [CCh]

这条命令允许总线控制器不用提供 64 位 ROM 编码就使用存储器操作命令，在单点总线情况下可以节省时间。如果总线上不止一个从机，在 Skip ROM 命令之后跟着发一条读命令，由于多个从机同时传送信号，总线上就会发生数据冲突（漏极开路下拉效果相当于相与）。

Search ROM [F0h]

当一个系统初次启动时，总线控制器可能并不知道单线总线上有多少器件或它们的 64 位 ROM 编码。搜索 ROM 命令允许总线控制器用排除法识别总线上的所有从机的 64 位编码。

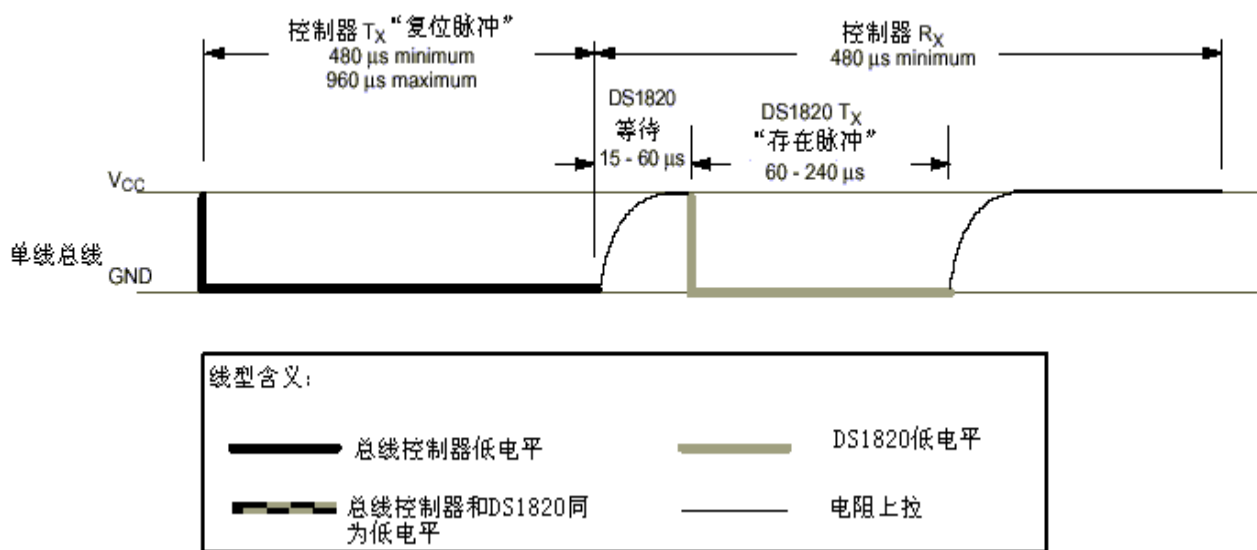
Alarm Search [ECh]

这条命令的流程图和 Search ROM 相同。然而，只有在最近一次测温后遇到符合报警条件的情况，DS1820 才会响应这条命令。报警条件定义为温度高于 TH 或低于 TL。只要 DS1820 不掉电，报警状态将一直保持，直到再一次测得的温度值达不到报警条件。

I/O 信号

DS1820 需要严格的协议以确保数据的完整性。协议包括几种单线信号类型：复位脉冲、存在脉冲、写 0、写 1、读 0 和读 1。所有这些信号，除存在脉冲外，都是由总线控制器发出的。和 DS1820 间的任何通讯都需要以初始化序列开始，初始化序列见图 11。一个复位脉冲跟着一个存在脉冲表明 DS1820 已经准备好发送和接收数据(适当的 ROM 命令和存储器操作命令)。

初始化过程“复位和存在脉冲”(图11)



DS1820 命令设置 (表 2)

命令	说明	协议	单线总线发出协议后	备注
温度转换命令				
Convert T	开始温度转换	44h	<读温度忙状态>	1
存储器命令				
Read Scratchpad	读取暂存器和 CRC 字节	BEh	<读数据直到 9 字节>	
Write Scratchpad	把字节写入暂存器的地址 2 和 3 (TH 和 TL 温度报警触发)	4Eh	<写两个的字节到地址 2 和 3>	
Copy Scratchpad	把暂存器内容拷贝到非易失性存储器中 (仅指地址 2 和 3)	48h	<读拷贝状态>	2
Recall E ²	把非易失性存储器中的值召回暂存器 (温度报警触发)	B8h	<读温度忙状态>	
Read Power Supply	标识 DS1820 的供电模式	B4h	<读供电状态>	

备注：

1、温度转换时间可长达 500ms。接到温度转换的协议后，如果器件不是从 VDD 供电的话，I/O 线就必须至少保持 500ms 高电平。这样，发出一个 Convert T 命令之后，单线总线上在这段时间内就不能有其他活动。

2、接到 Copy Scratchpad 协议后，如果器件不是从 VDD 供电的话，I/O 线必须至少保持 10ms 高电平。这样，在发出一个 Copy Scratchpad 命令后，这段时间内单线总线上就不能有其他活动。

总线控制器发出 (TX) 一个复位脉冲 (一个最少保持 480 μ s 的低电平信号)，然后释放总线，进入接收状态 (RX)。单线总线由 5K 上拉电阻拉到高电平。探测到 I/O 引脚上的上升沿后，DS1820 等待 15~60 μ s，然后发出存在脉冲 (一个 60~240 μ s 的低电平信号)。

存储器操作命令

下述命令协议概括于表 2，并用流程图示于图 10。

[存储器操作流程图 \(图 10\)](#)

[续图 10](#)

[续图 10](#)

Write Scratchpad [4E]

这个命令向 DS1820 的暂存器中写入数据，开始位置在地址 2。接下来写入的两个字节将被存到暂存器中的地址位置 2 和 3。可以在任何时刻发出复位命令来中止写入。

Read Scratchpad [BEh]

这个命令读取暂存器的内容。读取将从字节 0 开始，一直进行下去，直到第 9 (字节 8, CRC) 字节读完。如果不想读完所有字节，控制器可以在任何时间发出复位命令来中止读取。

Copy Scratchpad [48h]

这条命令把暂存器的内容拷贝到 DS1820 的 E² 存储器里，即把温度报警触发字节存入非易失性存储器里。如果总线控制器在这条命令之后跟着发出读时间隙，而 DS1820 又正在忙于把暂存器拷贝到 E² 存储器，DS1820 就会输出一个“0”，如果拷贝结束的话，DS1820 则输出“1”。如果使用寄生电源，总线控制器必须在这条命令发出后立即起动强上拉并最少保持 10ms。

Convert T [44h]

这条命令启动一次温度转换而无需其他数据。温度转换命令被执行，而后 DS1820 保持等待状态。如果总线控制器在这条命令之后跟着发出读时间隙，而 DS1820 又忙于做时间转换的话，DS1820 将在总线上输出“0”，若温度转换完成，则输出“1”。如果使用寄生电源，总线控制器必须在发出这条命令后立即起动强上拉，并保持 500ms。

Recall E² [B8h]

这条命令把报警触发器里的值拷回暂存器。这种拷回操作在 DS1820 上电时自动执行，这样器件一上电暂存器里马上就存在有效的数据了。若在这条命令发出之后发出读时间隙，器件会输出温度转换忙的标识：“0”=忙，“1”=完成。

Read Power Supply [B4h]

若把这条命令发给 DS1820 后发出读时间隙，器件会返回它的电源模式：“0”=寄生电源，“1”=外部电源。

读/写时间隙

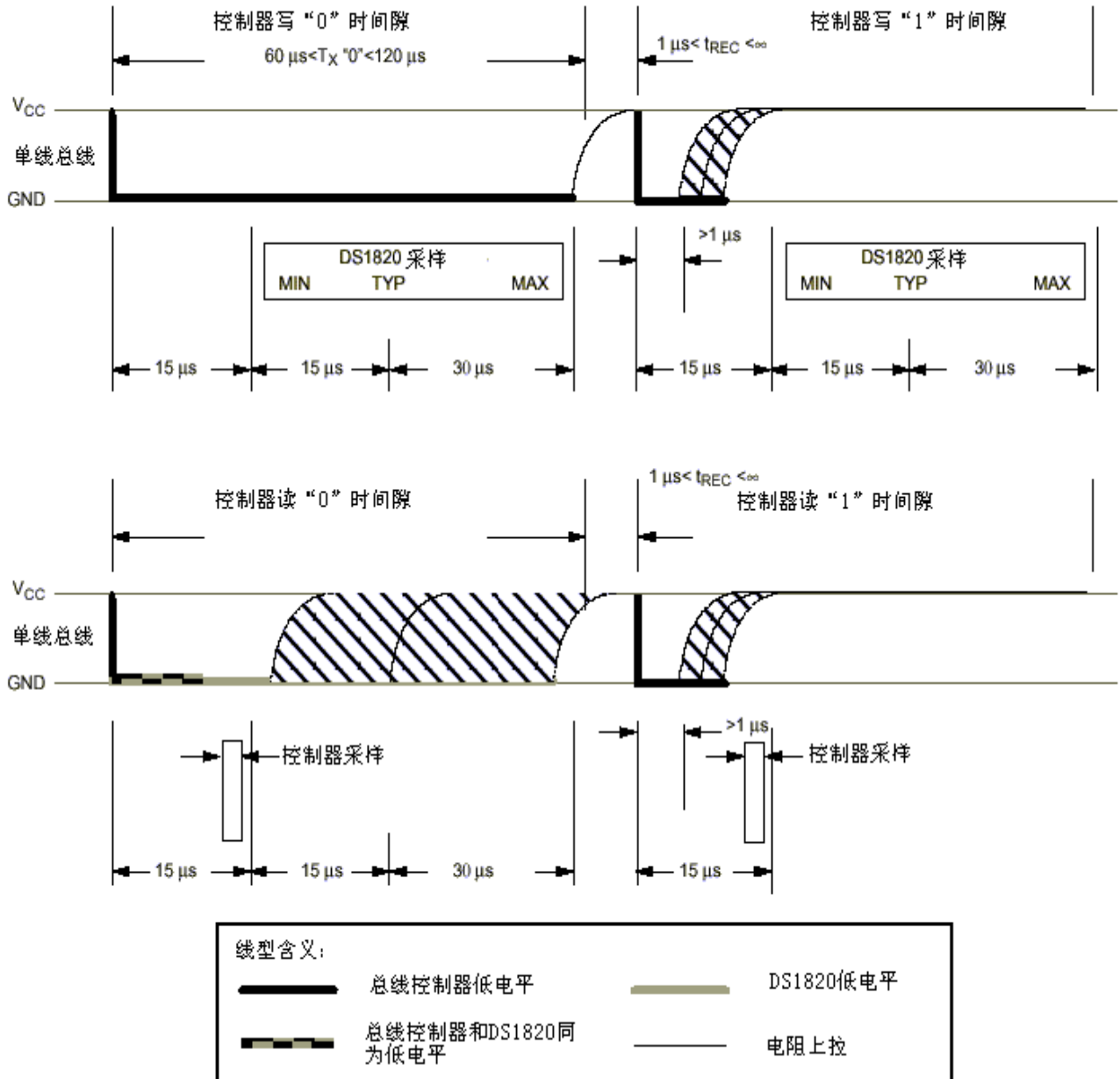
DS1820 的数据读写是通过时间隙处理位和命令字来确认信息交换。

写时间隙

当主机把数据线从逻辑高电平拉到逻辑低电平的时候，写时间隙开始。有两种写时间隙：写 1 时间隙和写 0 时间隙。所有写时间隙必须最少持续 60 μ s，包括两个写周期间至少 1 μ s 的恢复时间。

I/O 线电平变低后，DS1820 在一个 15 μ s 到 60 μ s 的窗口内对 I/O 线采样。如果线上是高电平，就是写 1，如果线上是低电平，就是写 0 (见图 12)

读/写时序图 (图12)



主机要生成一个写时间隙，必须把数据线拉到低电平然后释放，在写时间隙开始后的 $15 \mu s$ 内允许数据线拉到高电平。

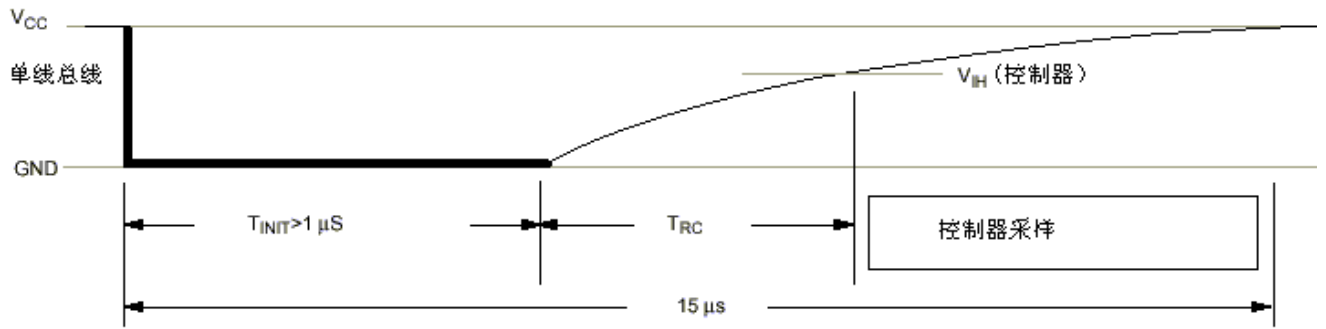
主机要生成一个写 0 时间隙，必须把数据线拉到低电平并保持 $60 \mu s$ 。

读时间隙

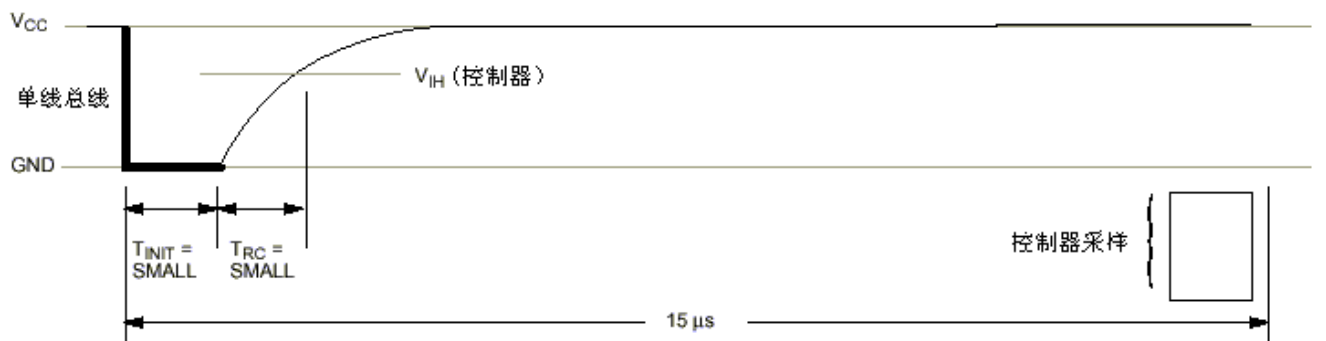
当从 DS1820 读取数据时，主机生成读时间隙。当主机把数据线从高高平拉到低电平时，写时间隙开始。数据线必须保持至少 $1 \mu s$ ；从 DS1820 输出的数据在读时间隙的下降沿出现后 $15 \mu s$ 内有效。因此，主机在读时间隙开始后必须停止把 I/O 脚驱动为低电平 $15 \mu s$ ，以读取 I/O 脚状态（见图 12）。在读时间隙的结尾，I/O 引脚将被外部上拉电阻拉到高电平。所有读时间隙必须最少 $60 \mu s$ ，包括两个读周期期间至少 $1 \mu s$ 的恢复时间。





图 13 表示 T_{INIT} 、 T_{RC} 和 T_{SAMPLE} 之和必须小于 $15 \mu s$ 。图 14 示出，系统时间可以用下面方法达到最大： T_{INIT} 和 T_{RC} 保持时间尽可能小；把控制器采样时间放到 $15 \mu s$ 周期的最后。

控制器读“1”的详细时序（图13）

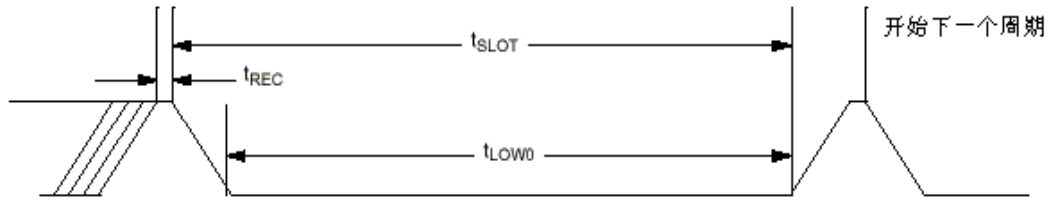


推荐控制器读“1”时序（图14）

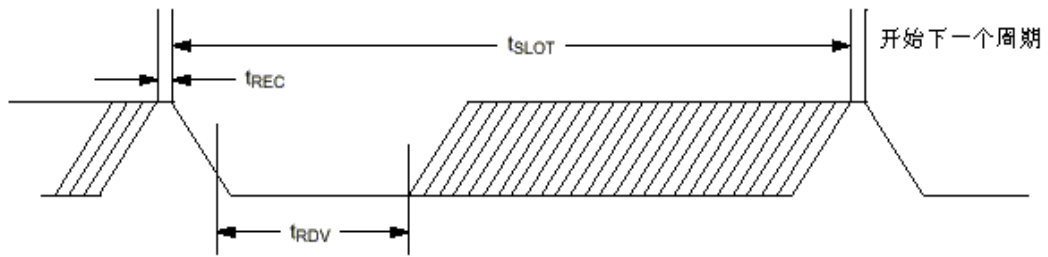


线型含义:			
	总线控制器低电平		DS1820低电平
	总线控制器和DS1820同为低电平		电阻上拉

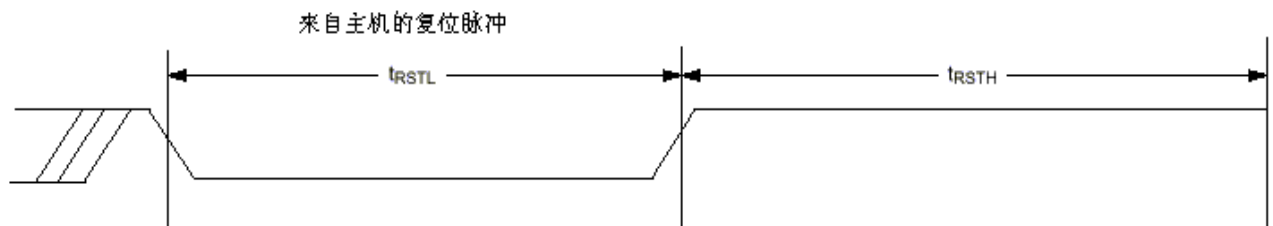
单线写0时间隙



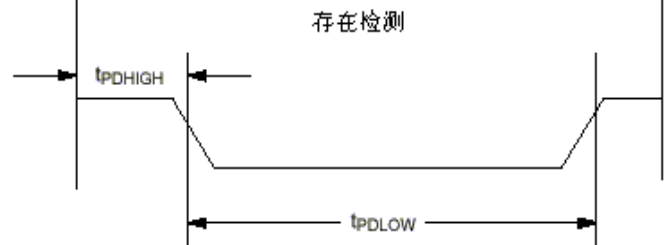
单线读0时间隙



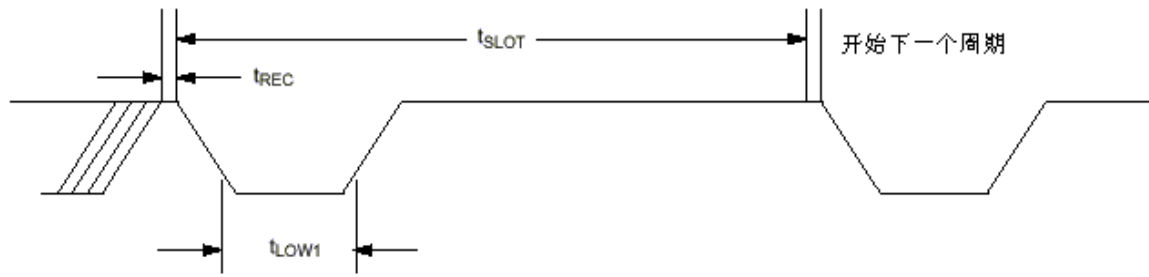
单线复位脉冲



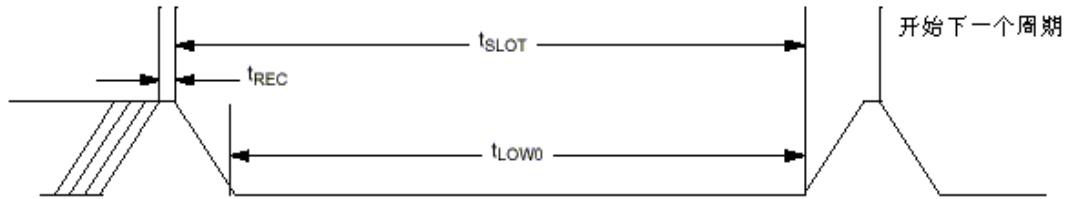
单线存在检测



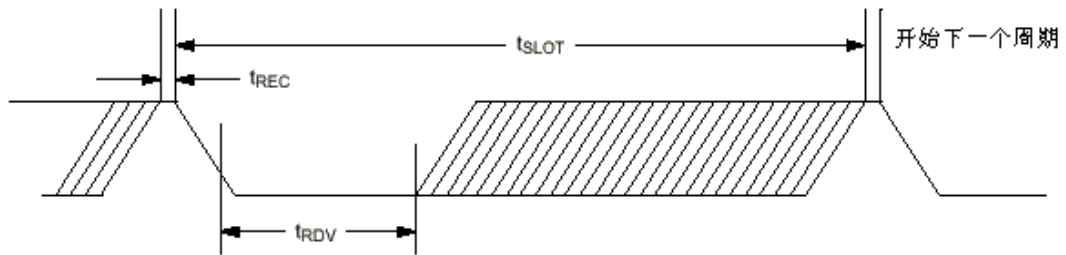
单线写1时间隙



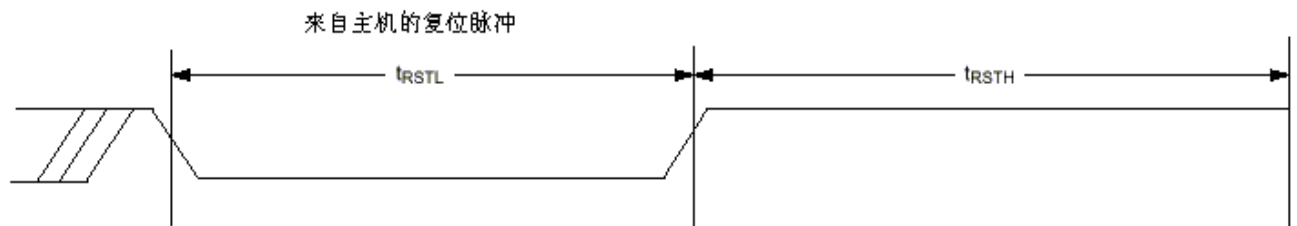
单线写0时间隙



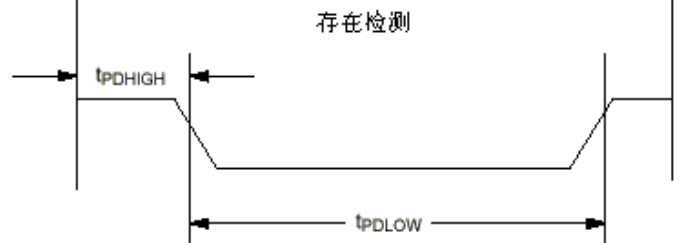
单线读0时间隙



单线复位脉冲



单线存在检测



存储器操作举例（表3）例：总线控制器启动温度转换，然后读取温度（寄生电源）

控 制 器 状 态	数据（LSB 在前）	内容
-----------------------	---------------	----

TX	复位	复位脉冲 (480-960 μs)
RX	存在	存在脉冲
TX	55h	发“Match ROM”命令
TX	<64 位 ROM 编码>	发 DS1820 地址
TX	44h	发“Convert T”命令
TX	<I/O 线高电平>	I/O 线保持至少 500ms 高电平，以完成温度转换
TX	复位	复位脉冲
RX	存在	存在脉冲
TX	55h	发“Match ROM”命令
TX	<64 位 ROM 编码>	发 DS1820 地址
TX	BEh	发“Read Scratchpad”命令
RX	<9 个数据字节>	读整个暂存器加上 CRC；控制器重新计算从暂存读到的 8 个数据字节的 CRC，把计算的 CRC 和读取的 CRC 进行比较，如果相同，控制器向下进行，如果不同，就重复读操作。
TX	复位	复位脉冲
RX	存在	存在脉冲，结束。

存储器操作举例 (表 4)

例：总线控制器写存储器 (寄生电源且只有一只 DS1820)

控制状态	数据 (LSB 在前)	内容
TX	复位	复位脉冲
RX	存在	存在脉冲
TX	CCh	Skip ROM 命令
TX	4Eh	Write Scratchpad 命令
TX	<两个数据字节>	写两个字节到暂存器 (TH 和 TL)
TX	复位	复位脉冲
RX	存在	存在脉冲
TX	CCh	Skip ROM 命令
TX	BEh	Read Scratchpad 命令
RX	<9 个数据字节>	读整个暂存器加上 CRC；控制器重新计算从暂存读到的 8 个数据字节的 CRC，把计算的 CRC 和读取的 CRC 进行比较，如果相同，控制器向下进行，如果不同，就重复读操作。
TX	复位	复位脉冲
RX	存在	存在脉冲
TX	CCh	Skip ROM 命令

TX	48h	Copy Scratchpad 命令；发出这条命令后，控制器必须等待 6ms 到拷贝操作完成。
TX	复位	复位脉冲
RX	存在	存在脉冲，完成。

存储器操作举例（表 5）

例：温度转换和插补（外部电源供电且只有一只 DS1820）

控 制 器 状 态	数据(LSB 在前)	内容
TX	复位	复位脉冲
TR	存在	存在脉冲
TX	CCh	Skip ROM 命令
TX	44h	Convert T 命令
RX	<1 个字 节的数据 >	读 8 次忙标志。控制器一字节（或位）接一个字节读下去，直到数据为 FFh（所有位都为 1）
TX	复位	复位脉冲
RX	存在	存在脉冲
TX	CCh	Skip ROM 命令
TX	BEh	Read Scratchpad 命令
RX	<9 个数据字节>	读整个暂存器加上 CRC；控制器重新计算从暂存读到的 8 个数据字节的 CRC，把计算的 CRC 和读取的 CRC 进行比较，如果相同，数据就是有效的。控制器存储温度值并分别存储计数寄存器的内容和每度计数值寄存器的内容，做为 COUNT_REMAIN 和 COUNT_PER_C。
TX	复位	复位脉冲
RX	存在	存在脉冲
—	—	CPU 按手册中的方法计算温度值以得到更高的分辨力。

极限使用条件：

各引脚对地电压：-0.5 到+7.0V

工作温度：-55 到+125

储存温度：-55 到+125

焊接温度：260 10 秒

推荐直流工作条件

参数	符号	条件	最小	典型	最大	单位	备注
电源电压	VDD	I/O 操作	2.8	5.0	5.5	V	1, 2
		± 1/2 温度转换精度	4.3		5.5		
数据引脚	I/O		-0.5		+5.5	V	2
逻辑 1	V _{IH}		2.0		VCC+0.3	V	2, 3

逻辑 0	V_{IL}		-0.3		+0.8	V	2, 4
------	----------	--	------	--	------	---	------

直流电特性

参数	符号	条件	最小	典型	最大	单位	备注
温度误差	t_{ERR}	-0 到+70 -55 到 0 和+70 到+125		$\pm 1/2$ 见曲线图			1, 9, 10
输入逻辑高电平	V_{IH}		2.2		5.5	V	2, 3
输入逻辑低电平	V_{IL}		-0.3		+0.8	V	2, 4
吸收电流	I_L	$V_{I/O}=0.4V$	-4.0			mA	2
待机电流	I_o			200	350	nA	8
动态电流	I_{DD}			1	1.5	mA	5, 6
输入负载电流	I_L			5		μA	7

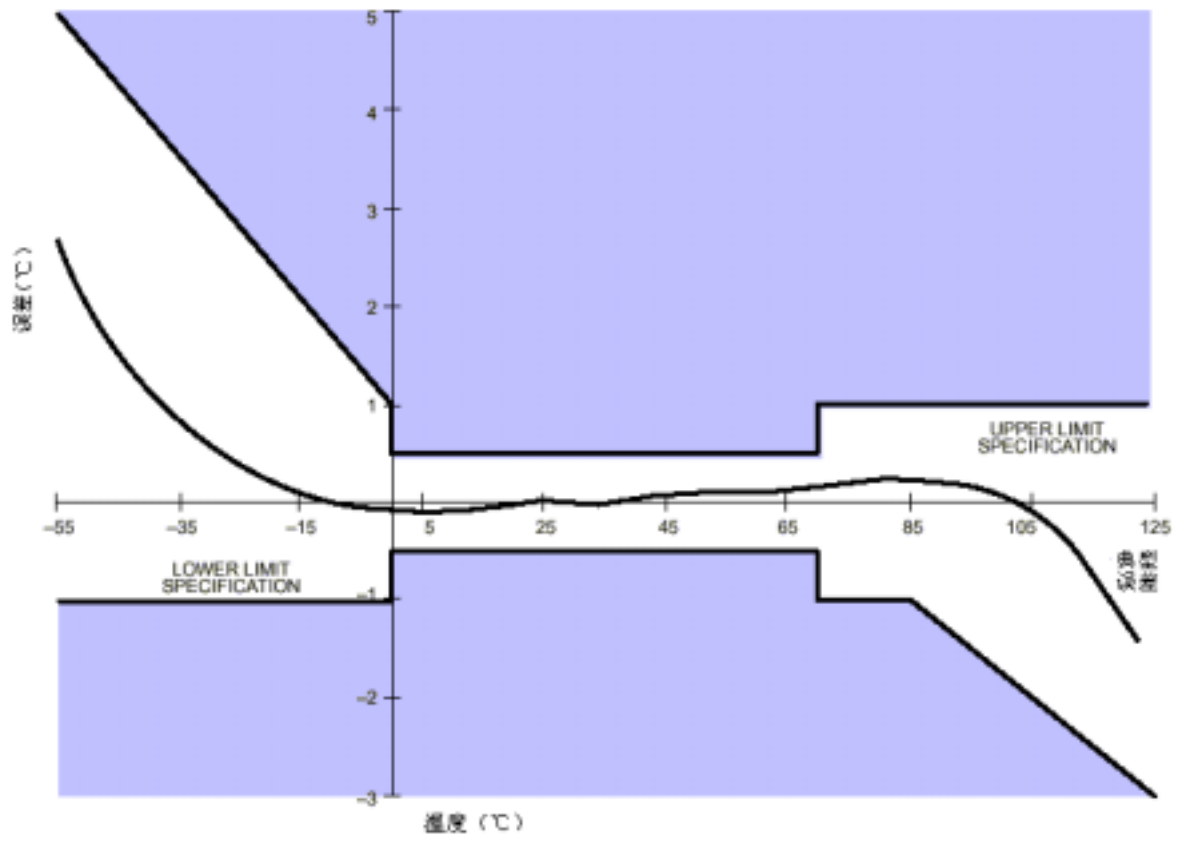
交流电特性：（-55 到+125 ；VDD=3.6V 到 5.5V）

参数	符号	最小	典型	最大	单位	备注
温度转换时间	t_{CONV}		200	500	ms	
时间隙	t_{SLOT}	60		120	μs	
恢复时间	t_{REC}	1			μs	
写 0 低电平时间	t_{LOW0}	60		120	μs	
写 1 低电平时间	t_{LOW1}	1		15	μs	
读数据有效时间	t_{RDV}			15	μs	
复位高电平时间	t_{RSTH}	480			μs	
复位低电平时间	t_{RSTL}	480		4800	μs	
存在检测高电平时间	$t_{PDH1GHL}$	15		60	μs	
存在检测低电平时间	t_{PDL0W}	60		240	μs	
电容	$C_{I/N/OUT}$			25	pF	

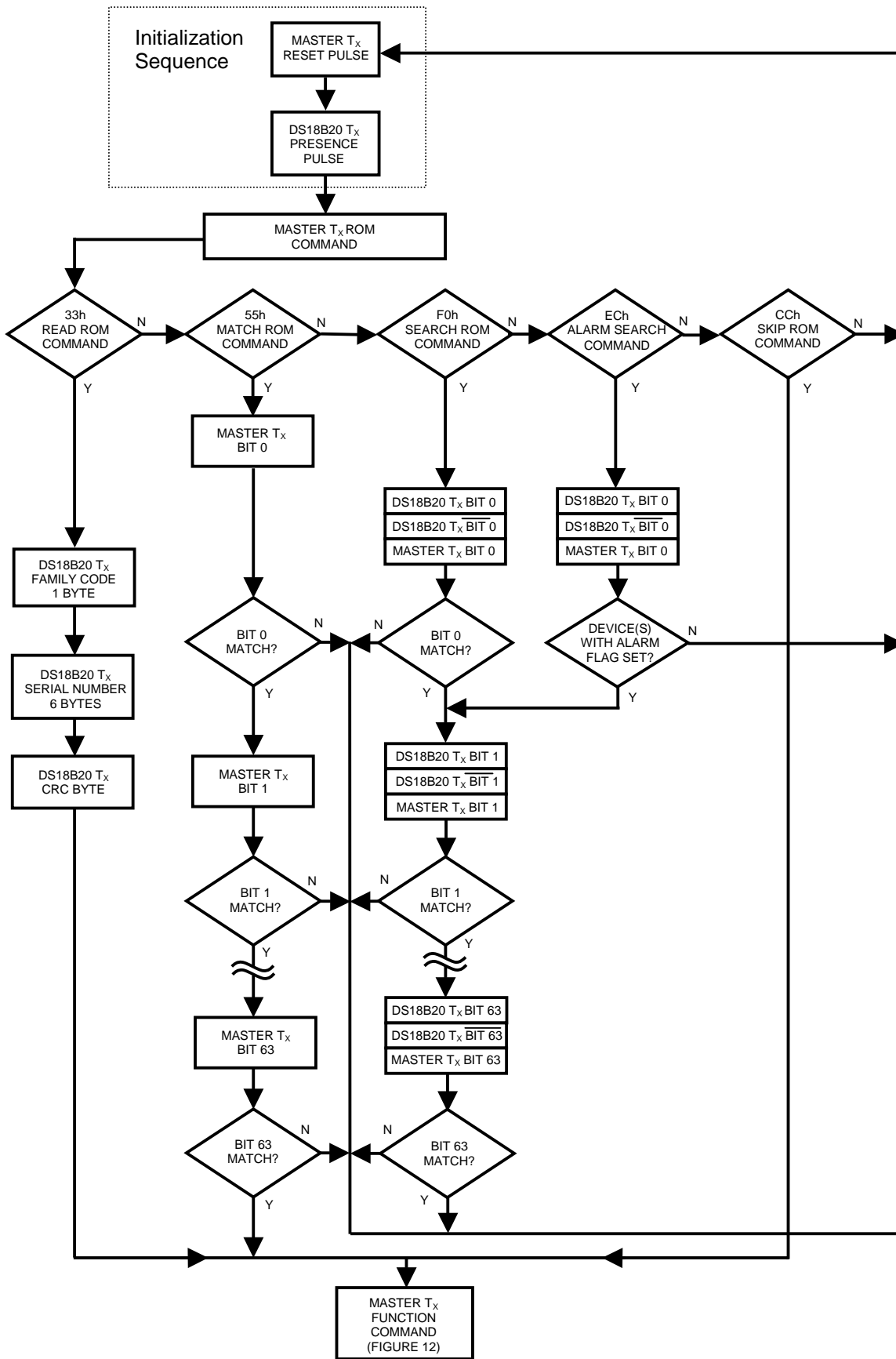
备注：

- 1、VDD 低至 3.4V 时，温度转换精度 ± 2 。
- 2、所有电压参考点都是接地点。
- 3、逻辑 1 电压在源电流为 1mA 时得到。
- 4、逻辑 0 电压在吸收电流为 4mA 时得到。
- 5、 I_{DD} 在 VCC 为 5.0V 时得到。
- 6、动态电流涉及温度转换和写 E^2 存储器。写 E^2 存储器最大用 10ms，消耗将近 100 μA 电流。
- 7、输入负载接地。
- 8、待机电流最大定义到 70 。125 时典型待机电流为 5 μA 。
- 9、见典型曲线图中超出 0 到 70 的部分。温度误差反映了传感器在校准时测试的精度。
- 10、典型精度曲线在 4.3V VDD 5.5V 时有效。

DS18B20读数误差



ROM COMMANDS FLOW CHART Figure 11



数字温度传感器 DS1820(DS18B20)的应用

一、单线数字温度计 DS1820 介绍

DS1820 数字温度计提供 9 位(二进制)温度读数, 指示器件的温度。信息经过单线接口送入 DS1820 或从 DS1820 送出, 因此从主机 CPU 到 DS1820 仅需一条线(和地线)。DS1820 的电源可以由数据线本身提供而不需要外部电源。因为每一个 DS1820 在出厂时已经给定了唯一的序号, 因此任意多个 DS1820 可以存放在同一条单线总线上。这允许在许多不同的地方放置温度敏感器件。DS1820 的测量范围从 -55°C 到 $+125^{\circ}\text{C}$, 增量值为 0.5°C , 可在 1s (典型值)内把温度转换成数字。

每一个 DS1820 包括一个唯一的 64 位长的序号, 该序号值存放在 DS1820 内部的 ROM(只读存储器)中。开始 8 位是产品类型编码(DS1820 编码均为 10H)。接着的 48 位是每个器件唯一的序号。最后 8 位是前面 56 位的 CRC(循环冗余校验)码。DS1820 中还有用于贮存测得的温度值的两个 8 位存储器 RAM, 编号为 0 号和 1 号。1 号存储器存放温度值的符号, 如果温度为负($^{\circ}\text{C}$), 则 1 号存储器 8 位全为 1, 否则全为 0。0 号存储器用于存放温度值的补码, LSB(最低位)的“1”表示 0.5°C 。**将存储器中的二进制数求补再转换成十进制数并除以 2 就得到被测温度值($-550^{\circ}\text{C}-125^{\circ}\text{C}$)**。DS1820 的引脚如图 2. 26—1 所示。每只 DS1820 都可以设置成两种供电方式, 即数据总线供电方式和外部供电方式。采取数据总线供电方式可以节省一根导线, 但完成温度测量的时间较长; 采取外部供电方式则多用一根导线, 但测量速度较快。

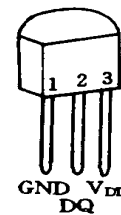


图 2. 26-1 DS1820 的引脚

1. GND: 地;
2. DQ: 数字输入/输出
3. V_{DD}: 可选的 +5V 电源

温度计算

1、Ds1820 用 9 位存储温度值, 最高位为符号位, 下图为 DS1820 的温度存储方式, 负温度 S=1, 正温度 S=0。如:

00AAH 为 $+85^{\circ}\text{C}$, 0032H 为 25°C , FF92H 为 -55°C

TEMPERATURE REGISTER FORMAT Figure 2

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
LS Byte	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}
	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
MS Byte	S	S	S	S	S	S	S	S

2、DS18B20 用 12 位存储温度值, 最高位为符号位, 下图为 DS18B20 的温度存储方式, 负温度 S=1, 正温度 S=0。如:

0550H 为 $+85^{\circ}\text{C}$, 0191H 为 $+25.0625^{\circ}\text{C}$, FC90H 为 -55°C

TEMPERATURE REGISTER FORMAT Figure 2

LS Byte	bit 7 2 ⁷	bit 6 2 ⁶	bit 5 2 ⁵	bit 4 2 ⁴	bit 3 2 ³	bit 2 2 ²	bit 1 2 ¹	bit 0 2 ⁰
MS Byte	bit 15 S	bit 14 S	bit 13 S	bit 12 S	bit 11 S	bit 10 2 ⁶	bit 9 2 ⁵	bit 8 2 ⁴

二、DS1820 工作过程及时序

DS1820 工作过程中的协议如下：

初始化：RoM 操作命令；存储器操作命令；处理数据。

1. 初始化

单总线上的所有处理均从初始化开始。

2. ROM 操作命令

总线主机检测到 DS1820 的存在，便可以发出 ROM 操作命令之一，这些命令如

指令	代码
Read ROM(读 ROM)	[33H]
Match ROM(匹配 ROM)	[55H]
Skip ROM(跳过 ROM)	[CCH]
Search ROM(搜索 ROM)	[F0H]
Alarm search(告警搜索)	[ECH]

3. 存储器操作命令

指令	代码
Write Scratchpad(写暂存存储器)	[4EH]
Read Scratchpad(读暂存存储器)	[BEH]
Copy Scratchpad(复制暂存存储器)	[48H]
Convert Temperature(温度变换)	[44H]
Recall EPROM(重新调出)	[B8H]
Read Power supply(读电源)	[B4H]

4. 时序

主机使用时间隙(time slots)来读写 DS1820 的数据位和写命令字的位

(1)初始化

时序见图 2.25-2。主机总线 t_0 时刻发送一复位脉冲(最短为 480us 的低电平信号)，接着在 t_1 时刻释放总线并进入接收状态，DS1820 在检测到总线的上升沿之后，等待 15-60us，接着 DS1820 在 t_2 时刻发出存在脉冲(低电平，持续 60-240 us)，如图中虚线所示。

以下子程序在 MCS51 仿真机上通过，其晶振为 12M。初始化子程序：

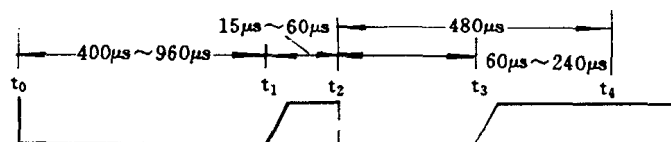


图 2.25-2 初始化时序

RESET:

```

PUSH B      ;保存 B 寄存器
PUSH A      ;保存 A 寄存器
MOV A,#4    ;设置循环次数
CLR P1.0    ;发出复位脉冲
MOV B,#250  ;计数 250 次
DJNZ B,$    ;保持低电平 500us
SETB P1.0   ;释放总线
MOV B,#6    ;设置时间常数
CLR C       ;清存在信号标志
WAITL: JB P1.0,WH ;若总线释放,跳出循环
      DJNZ B,WAITL ;总线低,等待
      DJNZ ACC,WAITL;释放总线等待一段时间
      SJMP SHORT
WH:     MOV B,#111
WH1:    ORL C,P1.0
      DJNZ B,WH1 ;存在时间等待
SHORT:  POP A
      POP B
      RET

```

(2)写时间隙

当主机总线 t_0 时刻从高拉至低电平时,就产生写时间隙,见图 2.25—3、图 2.25—4,从 t_0 时刻开始 $15\mu\text{s}$ 之内应将所需写的位送到总线上,DS1820 在 t_0 后 $15\text{--}60\mu\text{s}$ 间对总线采样。若低电平,写入的位是 0,见图 2.25—3;若高电平,写入的位是 1,见图 2.25—4。连续写 2 位间的间隙应大于 $1\mu\text{s}$ 。

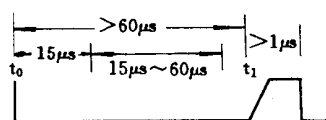


图 2.25-3 写 0 时序

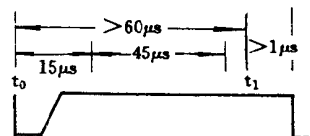


图 2.25-4 写 1 时序

写位子程序(待写位的内容在 C 中):

```

WRBIT:
      PUSH B      ;保存 B
      MOV B,#28   ;设置时间常数
      CLR P1.0    ;写开始
      NOP         ;1us
      NOP         ;1us
      NOP         ;1us
      NOP         ;1us
      NOP         ;1us
      MOV P1.0,C ;C 内容到总线
WDLT:  DJNZ B,WDLT;等待 56Us
      POP B

```



```

SETB P1.0    ;释放总线
RET          ;返回

```

写字节子程序(待写内容在 A 中):

WRBYTB:

```

        PUSH B          ;保存 B
        MOV B, #8H      ;设置写位个数
WLOP:   RRC A           ;把写的位放到 C
        ACALL WRBIT     ;调写 1 位子程序
        DJNZ B, WLOP    ;8 位全写完?
        POP B
        RET

```

(3)读时间隙

见图 2. 25—5, 主机总线 t_0 时刻从高拉至低电平时, 总线只须保持低电平 $17t_s$ 。之后在 t_1 时刻将总线拉高, 产生读时间隙, 读时间隙在 t_1 时刻后 t_2 时刻前有效。 t_z 距 t_0 为 $15\mu s$, 也就是说, t_z 时刻前主机必须完成读位, 并在 t_0 后的 $60\mu s - 120\mu s$ 内释放总线。读位子程序(读得的位到 C 中):

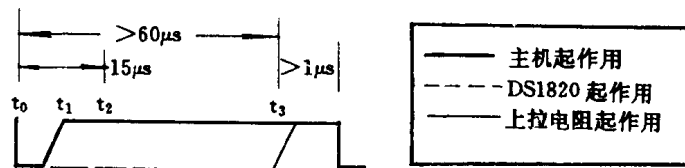


图 2. 25-5 读时序

RDBIT:

```

        PUSH B          ;保存 B
        PUSH A          ;保存 A
        MOV B,#23       ;设置时间常数
        CLR P1.0        ;读开始, 图 2. 25—5 的 t0 时刻
        NOP             ;1us
        NOP             ;1us
        NOP             ;1us
        NOP             ;1us
        SETB P1.0       ;释放总线
        MOV A,P1        ;P1 口读到 A
        MOV C,E0H       ;P1.0 内容 C
        NOP             ;1us
        NOP             ;1us
        NOP             ;1us
        NOP             ;1us
RDDLTL: DJNZ B,RDDLTL  ;等待 46us
        SETB P1.0
        POP A

```

POP B

RET

读字节子程序(读到内容放到 A 中);

RDBYTE:

PUSH B ;保存 B

RLOP: MOV B,#8H ;设置读位数

ACALL RDBIT ;调读 1 位子程序

RRC A ;把读到位在 C 中并依次送给 A

DJNZ B,RLOP ;8 位读完?

POP B ;恢复 B

RET

三、多路测量

每一片 DS1820 在其 ROM 中都存有其唯一的 48 位序列号, 在出厂前已写入片内 ROM 中, 主机在进入操作程序前必须逐一接入 1820 用读 ROM(33H)命令将该 1820 的序列号读出并登录。

当主机需要对众多在线 1820 的某一个进行操作时, 首先要发出匹配 ROM 命令(55H), 紧接着主机提供 64 位序列(包括该 1820 的 48 位序列号), 之后的操作就是针对该 1820 的。而所谓跳过 ROM 命令即为: 之后的操作是对所有 1820 的。框图中先有跳过 ROM, 即是启动所有 1820 进行温度变换, 之后, 通过匹配 ROM, 再逐一地读回每个 1820 的温度数据。

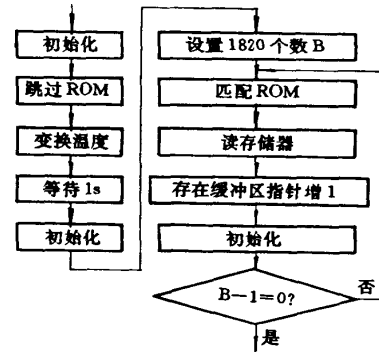


图 2.25-6 多路测温程序框图

在 1820 组成的测温系统中, 主机在发出跳过 ROM 命令之后, 再发出统一的温度转换启动码 44H, 就可以实现所有 1820 的统一转换, 再经过 1s 后, 就可以用很少的时间去逐一读取。这种方式使其 T 值往往小于传统方式 (由于采取公用的放大电路和 A / D 转换器, 只能逐一转换。), 显然通道数越多, 这种省时效应就越明显。

四、实际应用

1、ds1820 序列号获得

```

;|-----|
;|      读出 ds1820 序列号应用程序,P1.6 接 ds1820      |
;|-----|

```

ORG 0000H

AJMP MAIN

ORG 0020H

MAIN:

MOV SP,#60H

CLR EA ;使用 ds1820 一定要禁止任何中断产生

LCALL INT ;初始化 ds1820

MOV A,#33H

```

LCALL WRITE      ;送入读 ds1820 的 ROM 命令
LCALL READ      ;开始读出当前 ds1820 序列号
MOV 40H,A
LCALL READ
MOV 41H,A
LCALL READ
MOV 42H,A
LCALL READ
MOV 43H,A
LCALL READ
MOV 44H,A
LCALL READ
MOV 45H,A
LCALL READ
MOV 46H,A
LCALL READ
MOV 47H,A
SETB EA
SJMP $

```

```

INT:              ;初始化 ds1820 子程序
    CLR EA
L0:CLR P1.6      ;ds1820 总线为低复位电平
    MOV R2,#200
L1:CLR P1.6
    DJNZ R2,L1   ;总线复位电平保持 400us
    SETB P1.6    ;释放 ds1820 总线
    MOV R2,#30
L4:DJNZ R2,L4   ;释放 ds1820 总线保持 60us
    CLR C        ;清存在信号
    ORL C,P1.6
    JC L0        ;存在吗?不存在则重新来
    MOV R6,#80
L5:ORL C,P1.6
    JC L3
    DJNZ R6,L5
    SJMP L0
L3:MOV R2,#240
L2:DJNZ R2,L2
    RET

```

```

WRITE:           ;向 ds1820 写操作命令子程序
    CLR EA
    MOV R3,#8    ;写入 ds1820 的 bit 数,一个字节 8 个 bit

```

```

WR1:SETB P1.6
    MOV R4,#8
    RRC A           ;把一个字节 data(A)分成 8 个 bit 环移给 C
    CLR P1.6       ;开始写入 ds1820 总线要处于复位(低)状态
WR2:DJNZ R4,WR2   ;ds1820 总线复位保持 16us
    MOV P1.6,C     ;写入一个 bit
    MOV R4,#20
WR3:DJNZ R4,WR3   ;等待 40us
    DJNZ R3,WR1    ;写入下一个 bit
    SETB P1.6      ;重新释放 ds1820 总线
    RET

```

READ:

```

    CLR EA
    MOV R6,#8      ;连续读 8 个 bit
RE1:CLR P1.6      ;读前总线保持为低
    MOV R4,#4
    NOP
    SETB P1.6     ;开始读, 总线释放
RE2:DJNZ R4,RE2   ;持续 8us
    MOV C,P1.6    ;从 ds1820 总线读得一个 bit
    RRC A         ;把读得的位值环移给 A
    MOV R5,#30
RE3:DJNZ R5,RE3   ;持续 60us
    DJNZ R6,RE1   ;读下一个 bit
    SETB P1.6     ;重新释放 ds1820 总线
    RET

    END

```

2、温度转换和读取

```

;-----|
; 获取单个 ds1820 转化的温度值的应用程序,P1.6 接 ds1820 |
;-----|

    ORG 0000H
    AJMP MAIN
    ORG 0020H
MAIN:
    MOV SP,#60H
    LCALL GET_TEMP
    SJMP $

GET_TEMP:
    CLR PSW.4

```

```

SETB PSW.3      ;设置工作寄存器当前所在的区域
CLR EA          ;使用 ds1820 一定要禁止任何中断产生
LCALL INT       ;调用初使化子程序
MOV A,#0CCH
LCALL WRITE     ;送入跳过 ROM 命令
MOV A, #44H
LCALL WRITE     ;送入温度转换命令
LCALL INT       ;温度转换完全,再次初使化 ds1820
MOV A,#0CCH
LCALL WRITE     ;送入跳过 ROM 命令
MOV A,#0BEH
LCALL WRITE     ;送入读温度暂存器命令
LCALL READ
MOV R7,A        ;读出温度值低字节存入 R7
LCALL READ
MOV R6,A        ;读出温度值高字节存入 R6
SETB EA
RET

INT:             ;初始化 ds1820 子程序
CLR EA
L0:CLR P1.6     ;ds1820 总线为低复位电平
MOV R2,#200
L1:CLR P1.6
DJNZ R2,L1     ;总线复位电平保持 400us
SETB P1.6     ;释放 ds1820 总线
MOV R2,#30
L4:DJNZ R2,L4  ;释放 ds1820 总线保持 60us
CLR C          ;清存在信号
ORL C,P1.6
JC L0         ;存在吗?不存在则重新来
MOV R6,#80
L5:ORL C,P1.6
JC L3
DJNZ R6,L5
SJMP L0
L3:MOV R2,#240
L2:DJNZ R2,L2
RET

WRITE:          ;向 ds1820 写操作命令子程序
CLR EA
MOV R3,#8     ;写入 ds1820 的 bit 数,一个字节 8 个 bit
WR1:SETB P1.6

```

```

MOV R4,#8
RRC A           ;把一个字节 data(A)分成 8 个 bit 环移给 C
CLR P1.6       ;开始写入 ds1820 总线要处于复位(低)状态
WR2:DJNZ R4,WR2 ;ds1820 总线复位保持 16us
MOV P1.6,C     ;写入一个 bit
MOV R4,#20
WR3:DJNZ R4,WR3 ;等待 40us
DJNZ R3,WR1    ;写入下一个 bit
SETB P1.6      ;重新释放 ds1820 总线
RET

READ:
CLR EA
MOV R6,#8      ;连续读 8 个 bit
RE1:CLR P1.6   ;读前总线保持为低
MOV R4,#4
NOP
SETB P1.6      ;开始读, 总线释放
RE2:DJNZ R4,RE2 ;持续 8us
MOV C,P1.6     ;从 ds1820 总线读得一个 bit
RRC A          ;把读得的位值环移给 A
MOV R5,#30
RE3:DJNZ R5,RE3 ;持续 60us
DJNZ R6,RE1    ;读下一个 bit
SETB P1.6      ;重新释放 ds1820 总线
RET

END

```

DESIGN SHOWCASE

微控制器和 1-Wire 温度传感器的 软件接口

目前有数种方法,可将1-Wire®器件,如DS18B20, DS18S20或DS1822,与微处理器接口。这些方法包括:从简单的软件方案,到串行接口芯片,如DS2480,以及整合Dallas Semiconductor的VHDL 1-Wire主控制器于定制ASIC。本文陈述了一种最简单的软件解决方案,可实现微处理器和任意个数的DS18x20或DS1822温度传感器之间的1-Wire通信。

DS18B20, DS18S20和DS1822的详细时序和工作信息,由它们对应的数据手册提供,可以从Maxim/Dallas网址www.maxim-ic.com上下载。

硬件配置

图1的框图说明了在采用多个1-Wire温度传感器时,该硬件配置是多么的简单。一线制总线向所有的器件既提供通信连接,又提供工作电源。总线电源经由一个连接于3V至5.5V电源端的4.7kΩ上拉电阻提供。由于每个器件具有唯一的64位ROM识别码,所以挂载在总线上的1-Wire器件数量几乎不受限制。

接口时序

与DS18x20/DS1822的通信,是通过操作间隙,完成1-Wire总线上的数据传输。每个通信

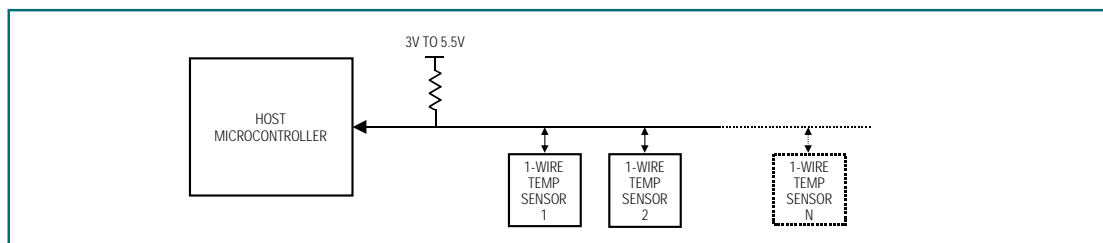


图1. 多个1-Wire温度传感器能够接口至同一条单总线。

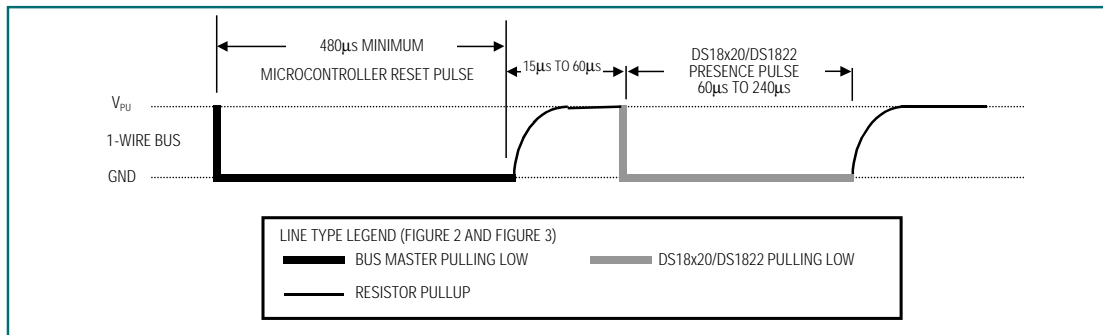


图2. 每个通信周期起始于微处理器发出的复位脉冲,其后紧跟DS18x20/DS1822发出的应答脉冲。

1-Wire是Dallas Semiconductor的一个注册商标。

周期起始于微控制器发出复位脉冲，其后紧跟 DS18x20/DS1822 发出的应答脉冲，如图 2 所示。

当主机将 1-Wire 总线从逻辑高(空闲状态)拉为逻辑低时，即启动一个写时隙。所有的写时隙必须在 60 μ s 至 120 μ s 内完成，且在每个循环之间至少需要 1 μ s 的恢复时间。写 0 和写 1 时隙如图 3 所示。在写 0 时隙期间，微控制器在整个时隙中将总线拉低；而写 1 时隙期间，微处理器将总线拉低，然后在时隙起始后 15 μ s 之内释放总线。

读时隙起始于微处理器将总线拉低 1 μ s，接着释放总线，这样 DS18x20/DS1822 就能够接管总线，输出有效数据(高或低)。所有读时隙在 60 μ s 至 120 μ s 内完成，且在每个循环之间至少需要 1 μ s 的恢复时间(图 3)。

软件控制

为了精确地控制 1-Wire 接口的特殊时序要求，必须先建立几个关键的函数。第一个函数应该是延时函数，它是所有读和写控制的组成部分。这个函数完全依赖于微处理器的速度。为了更好地理解，本文采用 DS5000(兼容 8051) 微处理器(工作时钟 11.059MHz) 为例。图 4 列举了一个用于创建时间延时的 C 原型函数。

由于每个通信周期起始于微处理器发出的复位脉冲，因而复位函数是下一个最为重要的函数。复位时隙为 480 μ s。首先以参数 3，接着以参数 25 分别调用延时函数(见图 5)，将产生所要求的复位脉冲，紧接着复位之后微处理器释放总线，以便 DS18x20/DS1822 通过拉低总线来指示其是否在线。如果多个温度传感器在此总线上，它们将同时发出应答脉冲。

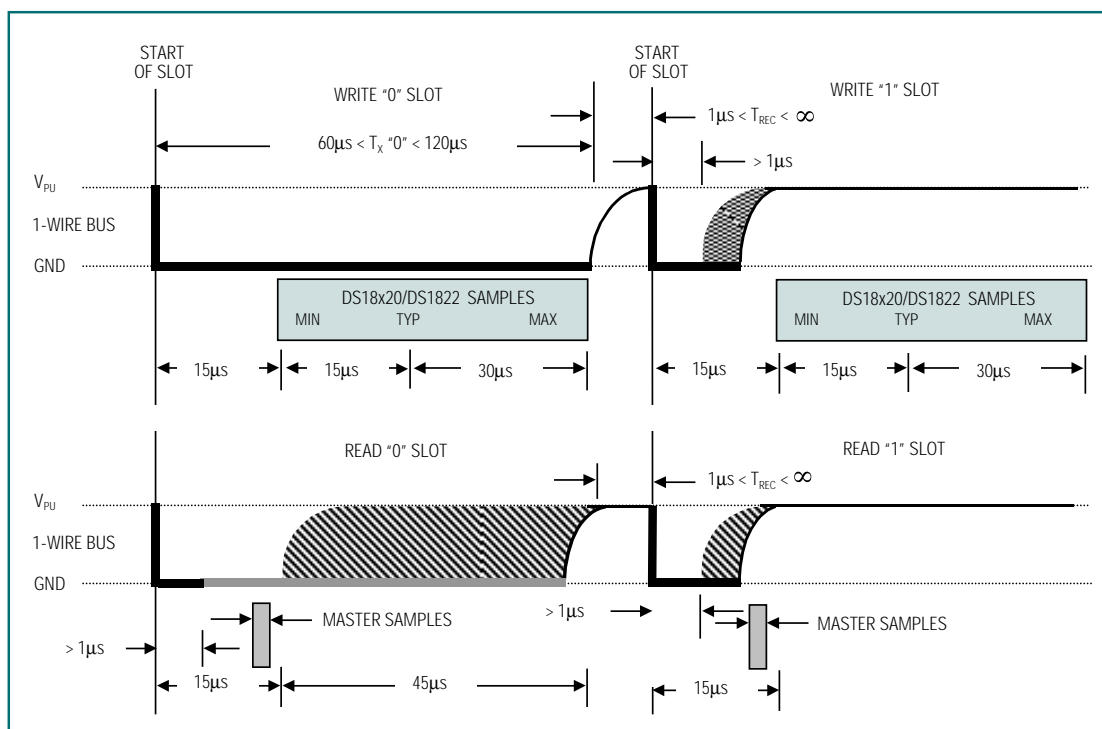


图3. 用于创建时间延时的C 原型函数

读和写函数实例如图6、7、8和9所示，提供了所有读/写数据位和字节操作的基本结构。

```
// DELAY - with an 11.059MHz crystal
// Calling the routine takes about 24µs, and then
// each count takes another 16µs
//
void delay (int µs)
{
    int s;
    for (s = 0; s < µs; s++);
}
```

图4. 延时实例

```
unsigned char ow_reset(void)
{
    unsigned char presence;

    DQ = 0;          //pull DQ line low
    delay(29);       // leave it low for 480µs
    DQ = 1;          // allow line to return high
    delay(3);        // wait for presence
    presence = DQ;   // get presence signal
    delay(25);       // wait for end of timeslot
    return(presence); // presence signal returned
}                  // presence = 0, no part = 1
```

图5. 复位实例

```
unsigned char read_bit(void)
{
    unsigned char i;

    DQ = 0; // pull DQ low to start timeslot
    DQ = 1; // then return high
    for (i = 0; i < 3; i++); // delay 15µs from
    start of timeslot
    return(DQ); // return value of DQ line
}
```

图6. 读位实例

```
void write_bit(char bitval)
{
    DQ = 0; // pull DQ low to start timeslot
    if(bitval==1) DQ =1; // return DQ high if write 1
    delay(5); // hold value for remainder of timeslot
    DQ = 1;

} // Delay provides 16µs per loop, plus 24µs
    Therefore, delay(5) = 104µs
```

图7. 写位实例

```
unsigned char read_byte(void)
{
    unsigned char i;
    unsigned char value = 0;

    for (i = 0; i < 8; i++)
    {
        if(read_bit()) value|= 0 x 01<<i;
        // reads byte in, one byte at a time and then
        // shifts it left
        delay(6); // wait for rest of timeslot
    }
    return(value);
}
```

图8. 读字节实例

```
void write_byte(char val)
{
    unsigned char i;
    unsigned char temp;

    for (i = 0; i < 8; i++) // writes byte, one bit at a time
    {
        temp = val>>i; // shifts val right 'i' spaces
        temp &= 0x01; // copy that bit to temp
        write_bit(temp); // write bit in temp into
    }

    delay(5)
}
```

图9. 写字节实例