SE18B20

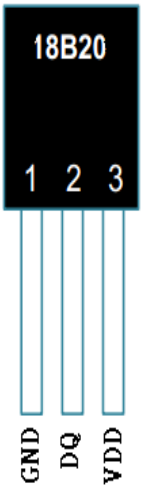
温度测量芯片

应用手册

版本：v1.2

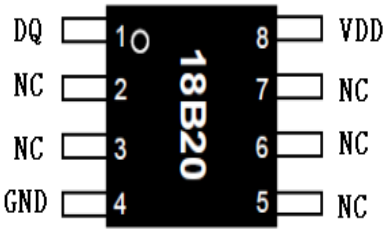
特点：

单线传输；
无需外接电源，寄生供电；
多点应用方案简单；
无需外部器件；
-10°C～+85°C 范围内；
精度为±0.5°C；
测量温度范围-55°C～+125°C；
分辨率从 9 位到 12 位可调；
温度测量一次最大转换时间 30ms；
可设置温度报警，并且可保存；
报警器件可通过主机查找识别；
自带应用软件；
特别适用于无电源供给的环境。



(BOTTOM VIEW)

TO-92
(SE18B20)



MSOP8L
(SE18B20)

描述

SE18B20（以下简称B20）是一款单线传输数字温度芯片，可以通过外部程序控制来设定9到12位的分辨率，检测温度误差在±0.5°C内。封装小，工作电源范围宽。用户设定的报警温度存储在EEPROM中，掉电后信息可依然保存。B20支持"一线总线"接口，测量温度范围-55°C～+125°C，在-10°C～+85°C 范围内，精度为±0.5°C。现场温度直接以"一线总线"的数字方式传输，大大提高了系统的抗干扰性。适合于恶劣环境的现场温度测量，如：环境控制、设备或过程控制、测温类消费电子产品等。产品支持3V～5.5V 的电压。

管脚		符号	描述
MSOP8L	TO-92		
2,3,5,6,7	——	NC	悬空
4	1	GND	地
1	2	DQ	数据输入/输出端口，同时也是电源端口
8	3	VDD	在寄生模式下，VDD 必须与地接在一起

表 1

概述

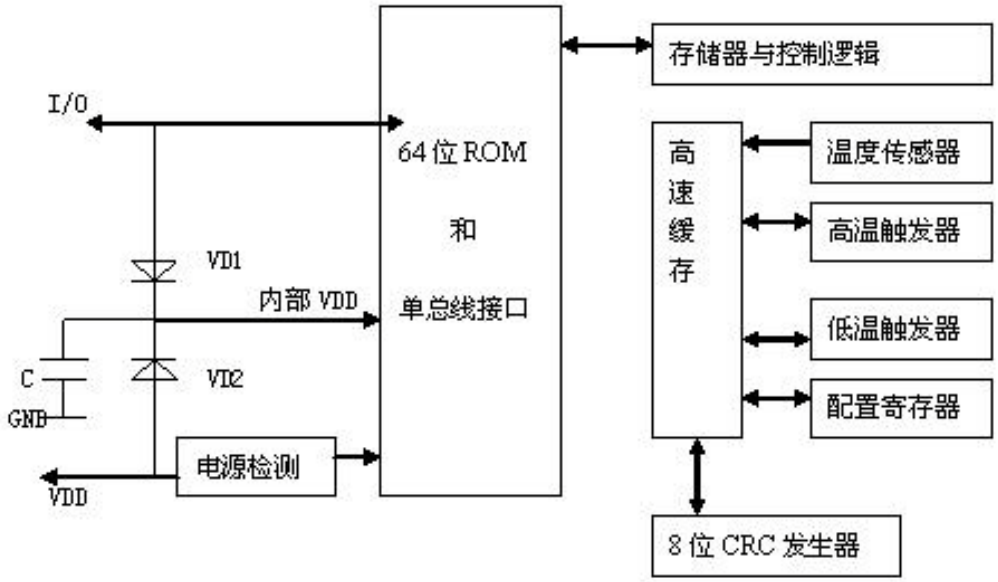
B20采用单线传输协议，也就是采用一根总线进行通信，所以总线在主机和从机传输数据是双向的。所有器件不是处于三态就是OD态，总线需要一个上拉电阻，在任何空闲时间总线必须保持为高。

在总线系统中，主机可以准确的找到每一个器件，因为每一个器件有唯一的64位编码，主机就是靠64位编码去查找的。

不需要外接电源就可以工作是B20的最大特点，而电源实际就是靠上拉电阻将DQ 端拉高提供，在传输1时（也就是DQ为高）同样会给B20提供电能，B20内

部的电容CPP就是储能器，当总线为低，CPP就可以为芯片提供电能。这种供电方法称之为寄生供电。

64 位 ROM 用来存储器件的专属信息，两位暂存器存储由温度传感器输出的数字温度信息，此外，暂存器还可以调用 1 个字节的高温和 1 个字节的低温报警信息，报警信息是被存在 EEPROM 里的，所以会一直保存。



芯片系统结构　图 1

寄生供电

B20 依靠寄生电源供电，这种方式可以适用不同的环境，特别适用控制器（主机）和传感器（B20）分离的情况；

寄生供电方式是通过当 DQ 为高时向芯片内部电容 CPP 充电，将能量存储在 CPP 里；当 DQ 为低时，通过 CPP 放电为芯片提供电能。

由于 CPP 储能有限，而芯片进行某些命令时比较耗能，所以在某些操作发生时仍需要给芯片提供额外的电源供给，否则会因为 CPP 过度放电而导致芯片数据的错误，比如温度转换命令，复制 ROM 命令都需要额外供电。

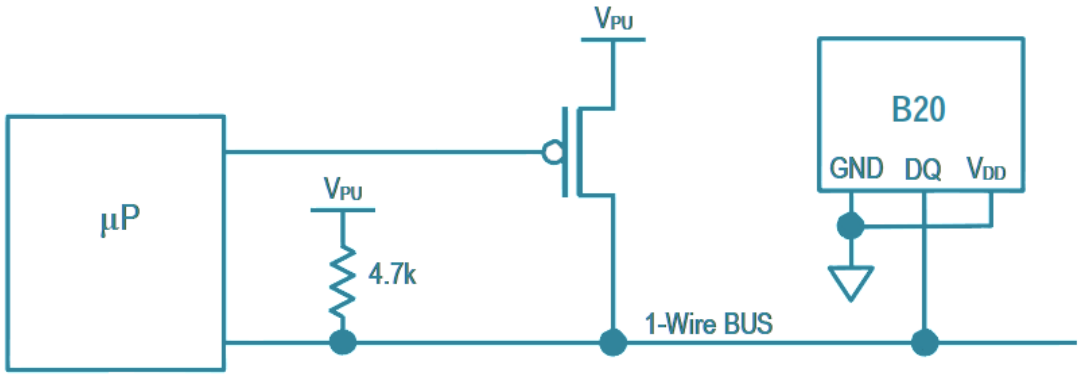


图 2

如上图，在某些比较耗电的命令时将 MOS 管打开。在寄生供电下，芯片的 VDD

端必须要与 GND 接在一起。

温度测量

B20的核心就是数字温度转换功能。它可以将模拟温度信息直接转换成数字信号。用户可通过命令去配置温度转化器，就可以得到不同的分辨率。9位温度数据对应分辨率是0.5°C，10位对应0.25°C，11位对应0.125°C，12位对应0.0625°C；芯片上电后默认为0.0625°C。

上电后B20处在低功耗静态模式下，只有主机发送转换命令，B20才开始工作。转换完成后，温度量被转换成2个字节的编码，存放在温度暂存器里，B20又返回静态模式。如果要显示温度，那需要主机读取温度暂存器的值。B20输出数据以度计算，通过下表可得。

温度数据被存放在有标志扩展位的二进制补码的16位温度暂存器里。S是符号位，S = 0表示温度是正的，S=1表示温度是负的。如果B20被配置成12位的分辨率，那么温度寄存器所有位的数据都是有效的，如果设置11位的分辨率，bit0位无效， 设置成10位分辨率，bit1位和bit0位无效；设置成9位，bit2位，bit1位，bit0位无效；

SE18B20 中的温度传感器完成对温度的测量，用 16 位二进制形式提供，形式表达，其中 S 为符号位。

LSB	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
MSB	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
	S	S	S	S	S	2^6	2^5	2^4

表 2

例如＋125°C 的数字输出为 07D0H（正温度 ，直接把 16 进制数转成 10 进制即得到温度值 ）

-55°C 的数字输出为 FC90H。（负温度，把得到的 16 进制数取反后加 1，再转成 10 进制数）

温度	数字信号（二进制）	数字信号（十六进制）
+85°C	0000 0101 0101 0000	0550H
+25.0625°C	0000 0001 1001 0001	0191H
+10.125°C	0000 0000 1010 0010	00A2H
+0.5°C	0000 0000 0000 1000	0008H
0°C	0000 0000 0000 0000	0000H
-0.5°C	1111 1111 1111 1000	FFF8H
-10.125°C	1111 1111 0101 1110	FF5EH
-25.0625°C	1111 1110 0110 1111	FE6FH
-55°C	1111 1100 1001 0000	FC90H

表 3

报警设置

当 B20 温度转换完之后，温度转换的值将会和用户事先存放在报警寄存器里的值进行比较。T_H 是高温报警寄存器 ，T_L 是低温报警寄存器；标志位 S 指示温度是正还是负，0 表示正，1 表示负。T_H 和 T_L 的值同时被保存在 EEPROM 里，即使掉电也不会丢失。芯片工作时，一上电内部逻辑信号会将 EEPROM 的值调出，存放在暂存器 T_H 和 T_L 里。

EEPROM 的值是通过命令写入的。

T_H T_L 格式

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
S	2^6	2^5	2^4	2^3	2^2	2^1	2^0

表 4

置高温报警，那么在写寄存器时 S 位必须为 0；

置低温报警，那么在写寄存器时 S 位必须为 1；

11 位温度寄存器的值从第 4 位开始同 T_H, T_L 进行比较，前 0 到 3 位不参与比较。如果结果高于 T_H 或低于 T_L，那么报警标志将会在 B20 里产生。在每次温度测量完之后，标志位会被更新，如果下次温度转换超温情况没有出现，标志位将会被关闭。

主机通过报警查找命令可检查位于总线上的有报警状态的 B20，任何设置温度报警的 B20 都会去响应总线命令，因此主机能够精确的定位响应的从机。如果报警标志位存在，而 T_H，T_L 又被改变了，那么需要再一次的温度转换才能够得到正确的报警情况（因为报警标志位会一直存在，直到再一次温度转换命令，它才根据时实情况进行相应的改变）。

64位ROM

每一个 B20 都有唯一的 64 位码子存放在 ROM 里，最低 8 位是产品代码，接下来 48 位是芯片自身序列号，最后 8 位是前 56 位的 CRC 码子。

8 位校验码		48 位序列号		8 位产品代码	
MSB	LSB	MSB	LSB	MSB	LSB

表 5

存储

B20的存储形式如下图，存储包括SRAM和3字节的EEPROM，三个字节的EEPROM分别存放T_H，T_L，配置的数据。

如果B20的报警功能不使用的话，那么T_H，T_L寄存器就相当于通用寄存器了。暂存器的第0和第1字节分别对应温度的低位和高位，这两个字节只能被读取。第2，第3字节用来存放T_H，T_L，第4字节用来存放配置数据，第5，第6，第7是默认的不能被修改。第8字节是只读的，它包含从0到7的CRC码字。

2 3 4字节的数据是需要通过写入命令4EH写入的，数据必须从第2字节的最低有效位开始。为了保证数据的完整性，在对暂存器写入之后，必须能保证将写入

的数据读出。当读暂存器时，数据从暂存器的0字节的最低有效位开始到结束。当然可以通过命令48H，将T_H，T_L配置寄存器的值复制到EEPROM里，这样即使掉电这几组数据也不会丢失。

当上电后，EEPROM 里的数据会被自动加载到相应的暂存器里，当然这种加载也可以通过命令 B8H 随时进行，主机必须确保在发送 B8H 后有读的时隙，因为 B20 把 EEPROM 里的数据加载到相应的暂存器需要时间，当 B20 传输 0 表示未加载完，传输 1 表示加载完。

暂存器上电状态

byte0	温度暂存器 LSB
byte1	温度暂存器 MSB
byte2	T_H 暂存器或用户字节 1*
byte3	T_L 暂存器或用户字节 2*
byte4	配置暂存器*
byte5	预留
byte6	预留
byte7	预留
byte8	CRC *

表 6

字节 2 3 4 的值在上电后，芯片会从 ROM 里调用，所以它们的值取决于 ROM 存放的值。当然如果想要修改可以通过写入命令 4EH。如果 T_H，T_L 不作温度比较用，那么可以作为一般寄存器使用。

配置寄存器

第 4 个字节是配置数据，如下表。

R0 R1 用来设置温度转换分辨率， bit7 是模式位，设置为 0 表示 B20 处于工模式，设置为 1 表示 B20 处于测试模式。

配置寄存器

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0	R1	R0	1	1	1	1	1

表 7

转换分辨率配置

R1	R0	分辨率	最大转换时间	
0	0	9 位	3.75ms	$T_{CONV}/8$
0	1	10 位	7.5ms	$T_{CONV}/4$
1	0	11 位	15ms	$T_{CONV}/2$
1	1	12 位	30ms	$T_{CONV}/1$

表 8

CRC

CRC码子由两处提供，一个是64位ROM提供，另一处在暂存器的第9字节里。

闪存CRC是通过存储在闪存内的数据计算而来，当数据改变，它也会随之改变。当从B20读取数据时，CRC通过一定的方法为总线提供有效的数据。为了核实已经读取的数据的正确性，总线必须对已接收的数据重新计算CRC，然后将计

算的结果同ROM的CRC或暂存器的CRC进行比较（读ROM的数据就和ROM里的CRC码子比较，读暂存器的数据就和暂存器里的CRC比较）。

如果计算的CRC和读取的CRC匹配，那么接收的数据就是正确的。CRC比较的值以及如何进行下一步操作都靠总线来决定；

如果B20里的CRC和总线计算的CRC不匹配，B20里没有电路可以阻止程序继续运行。CRC $=X^8+X^5+X^4+1$

总线重新计算CRC，将结果与B20里的数据通过多项式比较，如图3所示，电路包含一个移位寄存器和异或门，移位寄存器初始化为0。从ROM最低有效位开始或者从暂存器的最低有效0数据位开始，一次移一位到寄存器。当移完第56位或暂存器的最高有效位后，生成器包含重新计算的CRC码字。接下来，B20里8比特的ROM数据或暂存器的CRC必须移位到电路里。此时，如果重新计算的CRC是正确的，移位寄存器将全成0。

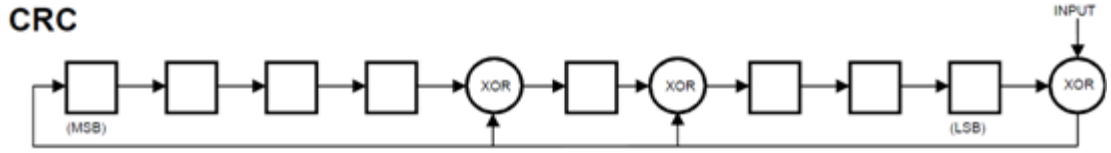


图 3

单线系统

单线系统采用一根总线控制一个或多个从器件。B20是一个从器件，如果只有一个位于总线，则称为单路系统，如果有多个器件称为多路系统。所有的数据和命令都是通过单总线从最低有效位开始传输的，单线系统可分为三个部分进行讨论： 硬件配置，传输顺序和单线信号性质。

硬件配置

单总线只有一个数据线，每一个器件(从或主)通过OD门或三态门与总线连接。这样当器件不传送数据时以便释放总线供其它器件使用总线，B20是一个OD门与等效电路的结构，如图4所示，B20需要一个大约5K的上拉电阻，闲置时总线为高。

不管什么原因传输中止，总线必须为高，只要总线处于高，那么位之间的恢复时间可以无限长。如果总线被拉低的时间大于480 μ s，所有位于总线上的器件将被复位。此外，为了保证B20在温度转换期间有足够的电流供应，很有必要提供一个强有力的上拉位于总线。

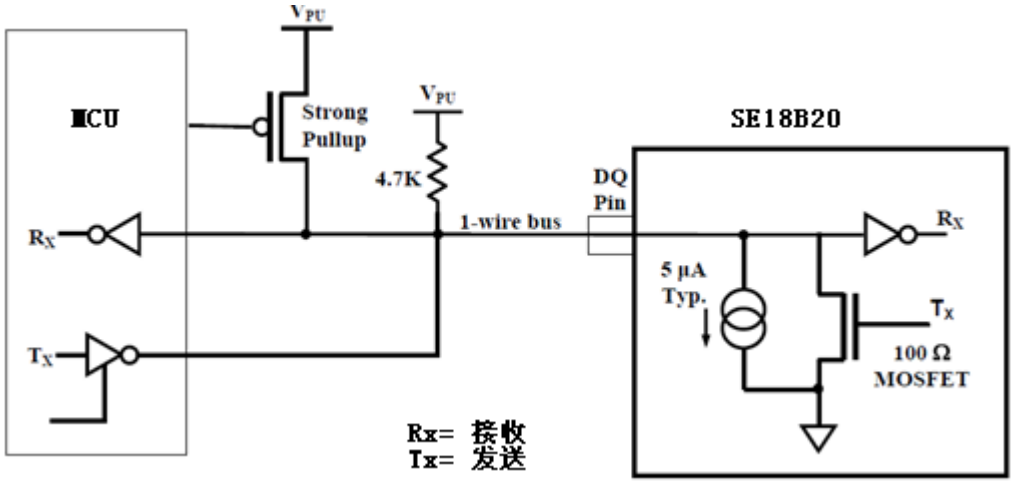


图 4

传输顺序

Step 1： 初始化
Step 2： rom命令
Step 3：B20功能命令

严格按照以上三步才能对B20进行操作，如果缺失某步或顺序不对，B20都不会响应。查找ROM和报警查询不需要遵循以上三步；

初始化

所有传输都是从复位开始，复位过程包括一个主机发送的**复位脉冲**，和继主机脉冲之后从机发送的**存在脉冲**，存在脉冲让主机知道从机处在总线上准备动作，复位脉冲和存在脉冲的时长有严格要求，否则不响应。

ROM 命令

主机侦测到存在脉冲后就可以发送ROM命令了ROM命令对每一个器件的64位ROM数据进行操作，如果总线有多个从器件，那么通过此命令可针对某一器件操作。这些命令允许主机决定处在总线上器件的类型和个数，或有报警情况的器件。有5个ROM命令， 每个命令是8位。在发出功能命令前，主机必须发出合适的ROM命令。

命令 名称	代码	描述
读ROM	33H	读取B20里的编码（64位）
核对ROM	55H	发出此命令后，接着发出64位ROM编码，访问总线上与之匹配的B20。
搜索ROM	0F0H	确定总线上B20的个数和64位ROM编码
跳过ROM	0CCH	忽略64位编码，可直接发送温度转换命令，单片操作
警报搜索	0ECH	搜索超过温度设定值的B20

表9

查找ROM [F0h]

当系统初始化上电之后，主机必须识别总线上所有从器件的ROM码，这样主机就可以知道总线上的器件号码和器件类型。

主机通过排除法，尽可能多次的循环去查找从机的ROM，如果只有一个从机，那么直接就可以用READ ROM命令了。每次查找命令结束后，总线都要恢复到初始化状态。

读ROM [33h]

当总线上只有一个从机时才可以用这个命令，它允许主机在没有用 S ROM 的前提下直接读取从机的64位ROM码字。如果总线上不止一个从机，当采用此命令时，数据将发生冲突，因为不同的从机都在响应。

匹配ROM [55h]

MROM命令后跟随64位ROM码字，这样主机就可以访问指定的从机，无论总线上有单个或多个从机。仅仅匹配64位码字的从机才能响应主机发送的功能命令；而其它从机将等待复位脉冲。

跳过ROM [CCh]

此命令可以不用发送任何ROM码字，同时访问位于总线上的所有器件。此命令后如果再跟随功能命令44H，主机就可以让位于总线上所有器件进行温度转换。

当总线上只有一个从机时，那么跳过命名后可以直接跟随读暂存器命令。如果总线不止一个从机，那么跳过命令后跟随读暂存器命令，数据将会发生冲突的。

报警查找 [ECh]

告警命令和查找ROM命令基本是相同的操作，只不过只有设置报警的从机才会响应此命令。这个命令就可以让主机知道哪个从机发生了温度警报。在每个周期结束，总线必须复位。

B20功能命令

当主机发送了ROM命令告后，主机就可以发送功能命令了。功能命令允许主机对从机进行读或写暂存器操作，初始化温度转化，确定电源供电方式等。

命令 名称	代码	描述
温度变换	44H	启动温度转换，
读暂存器	BEH	读暂存器的内容
写暂存器	4EH	向暂存器写入数据，分别位于2 3 4 字节位
复制暂存器	48H	将暂存器2 3 4 的码字复制到EEPROM里
调用EEPROM	B8H	将EEPROM内容重置到暂存器里

表10

温度转换 T [44h]

此命令初始化温度转换功能，温度转换后，结果存在2个字节的温度暂存器

里，B20返回低功耗的静止态。在温度转换命令发送后10us内，主机必须为B20提供强有力的上拉，为转化期间提供电源。

写暂存器 [4Eh]

写暂存器命令允许主机写入3个字节数据到B20的暂存器里。第一组数据写入 T_H 机存器（暂存器的第2字节），第二组数据写入 T_L（暂存器3），第3组数据写入配置寄存器（暂存器4），数据必须从最低有效位开始。所有的三组数据必须在主机复位之前写入，否则会混乱。

读暂存器 [BEh]

允许主机读取暂存器的内容，数据传输从最低有效位0开始，直到读取第9个字节的CRC码字。如果只需要读取某一部分的暂存器，那么只需要主机发送复位信号就可以终止。

复制暂存器 [48h]

可以将暂存器T_H，T_L 配置寄存器的内容复制到EEPROM里，命令发送完10us内主机必须为B20提供不小于10ms 强拉电源。

调用 E2 [B8h]

可以将存在EEPROM的数据重新调出，配置在相应的暂存器里。在调用命令后主机需要分配一定的读时间，B20将会指示调用的状态，B20发送0表示正在进行调用，发送1表示调用结束。

同时调用操作也会在上电后自动进行一次，所以当给B20供电后，B20也会将EEPROM的数据调出的。

ROM命令时序图

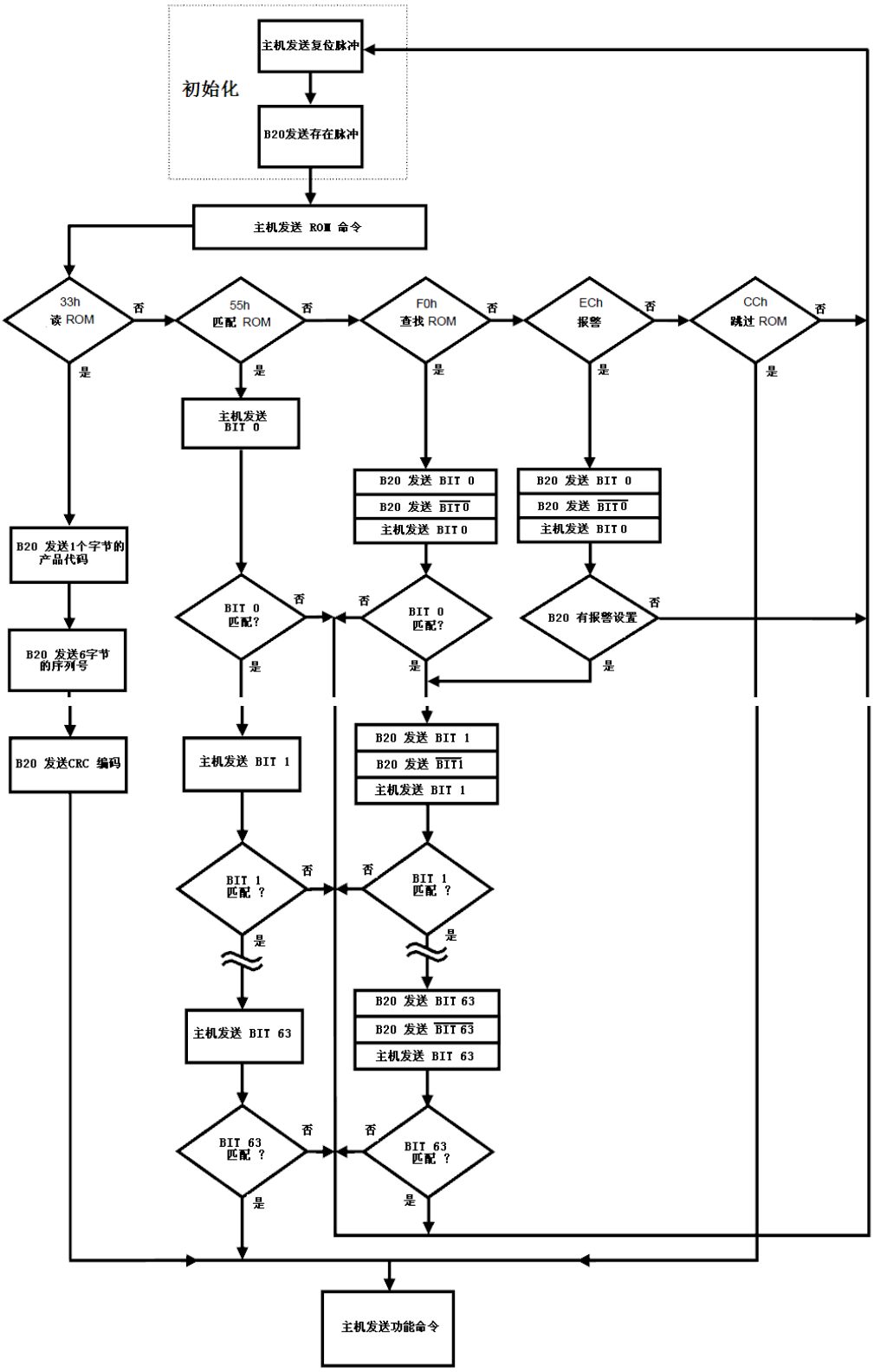

初始化
主机发送复位脉冲
B20发送存在脉冲
主机发送 ROM 命令
33h
读 ROM
55h
匹配 ROM
F0h
查找 ROM
ECh
报警
CCh
跳过 ROM
否
是
主机发送
BIT 0
B20 发送 BIT 0
B20 发送 BIT0
主机发送 BIT 0
B20 发送1个字节的
产品代码
BIT 0
匹配?
B20 有报警设置
B20 发送6字节
的序列号
B20 发送CRC 编码
主机发送 BIT 1
B20 发送 BIT 1
B20 发送 BIT1
主机发送 BIT 1
BIT 1
匹配 ?
主机发送 BIT 63
B20 发送 BIT 63
B20 发送 BIT 63
主机发送 BIT 63
BIT 63
匹配 ?
主机发送功能命令

图 5

功能命令

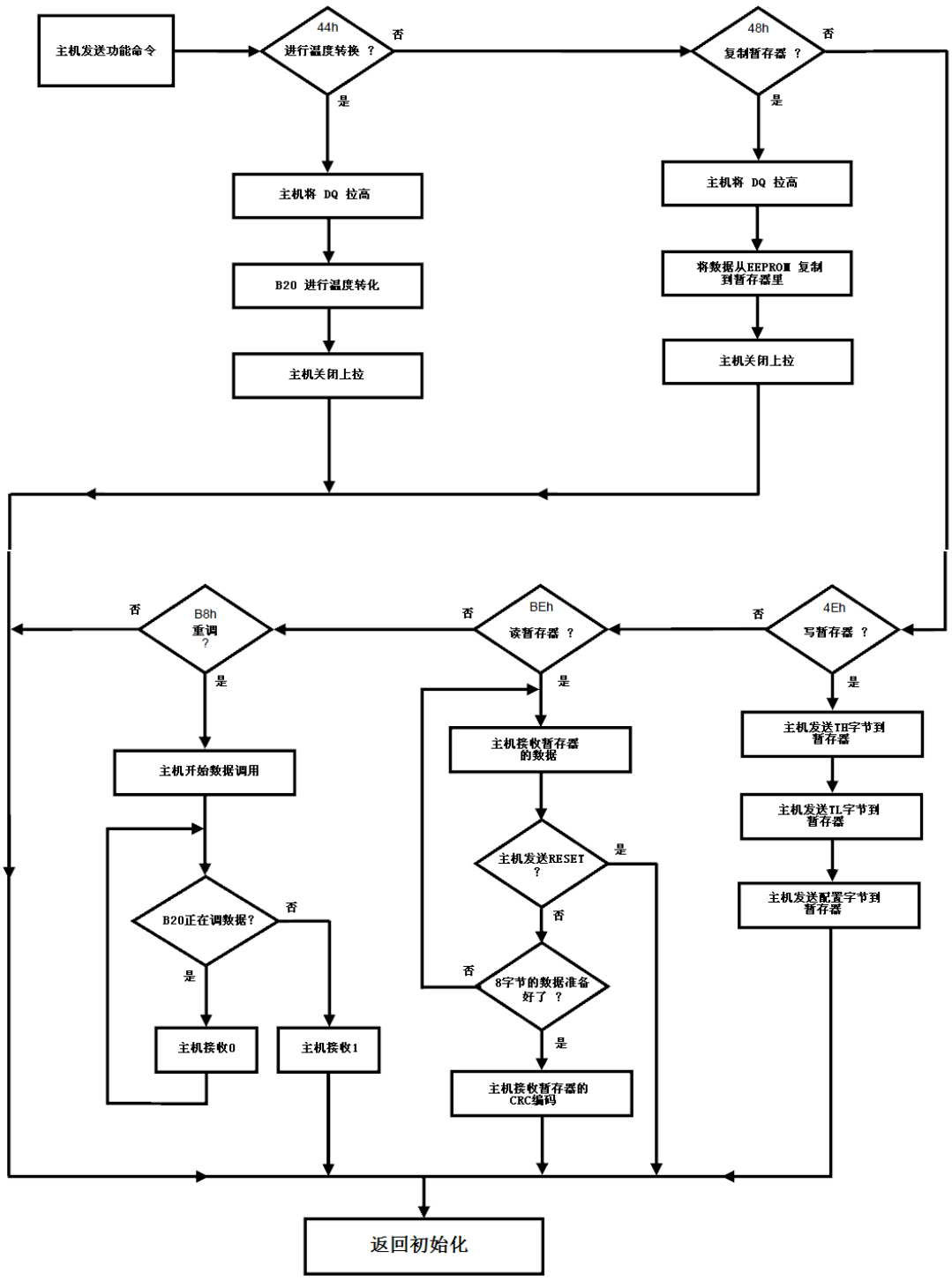


图 6

单线信号

B20采用单线传输协议确保数据的完整性。此协议定义了几个信号：复位脉冲，存在脉冲，写0，写1，读0，读1。所有信号除了存在脉冲都需要主机初始化后才能进行。

初始化

B20所有的通信都是从完成初始化过程开始的，初始化过程包含主机发送的复位脉冲，和之后B20反馈一个存在脉冲。当B20响应主机发送的复位脉冲时会发出发出存在脉冲，这样就表明B20处在总线上准备工作。

初始化时，主机先传输一个复位脉冲，此脉冲将总线拉低最少480μs，然后主机释放总线，处于接受信号模式。当总线被主机释放，5K的上拉电阻将总线拉高，B20检测到总线的上升沿，大约等待15–60μs，B20将总线拉低60–240 μs，这个低脉冲称为存在脉冲。参照下图时序。

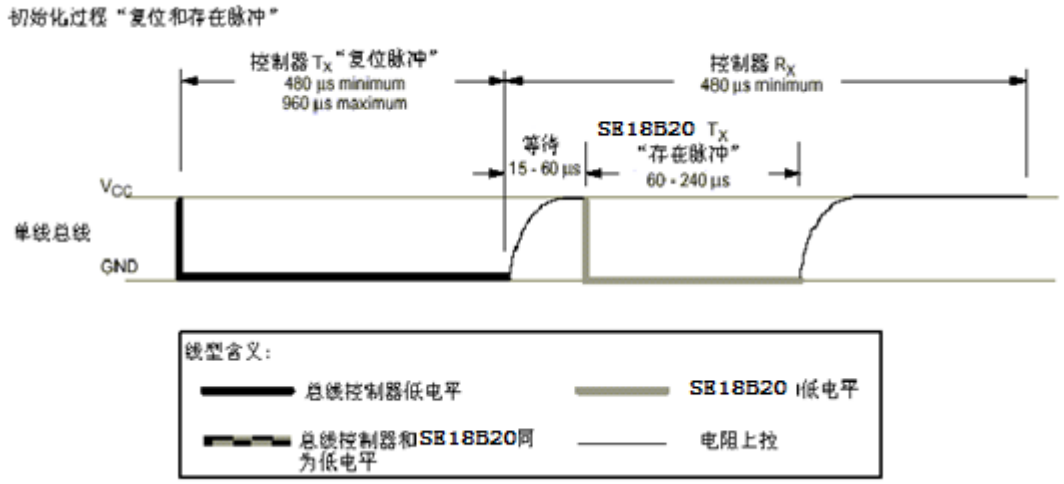


图 7

读写时隙

主机向B20写数据或从B20读数据都需要时间。操作一位所需要的时间和信号的规范称为一个时隙。

写入时隙

有两种写入指令周期，写1与写0。主机用写1时间向B20写入逻辑1，同理用写0时间向B20写入0。所有的写入时间必须包含不小于1us的恢复时间和60us的持续时间。写1时间，在将总线拉低后，主机必须在15μs内释放总线，当总线被主机释放，总线将被拉高。写0时间，在将总线拉低后，主机必须继续保持拉低最少60μs。在主机指示写入时间后，从15us到60us，B20采样总线数据。总线是高，就采为1，总线是低，就采作数据0。

读时隙

当主机发出读的时隙时，B20才能传送数据到主机。所以主机发出读暂存器命令后必须立刻生成读的时隙，以便B20穿数据。除此之外，当主机发送重调EEPROM的命令后，必须生成读的时隙，因为B20会反馈给主机信号来指示内部调用是否结束。所有读的时隙必须不小于60μs，其中包含不小于1μs的复位。一个读的时隙首先是主机将总线拉低1μs，然后释放。在读的时隙初始化后，B20就可以传送数据了，如果传1，保持总线高，传0，拉低总线。

当传0时，B20在读的时隙结束释放总线，以便总线靠上拉电阻重新回到高。

B20发送的数据有效采样时间必须在从主机发送读时隙的开始时刻算起15μs内。

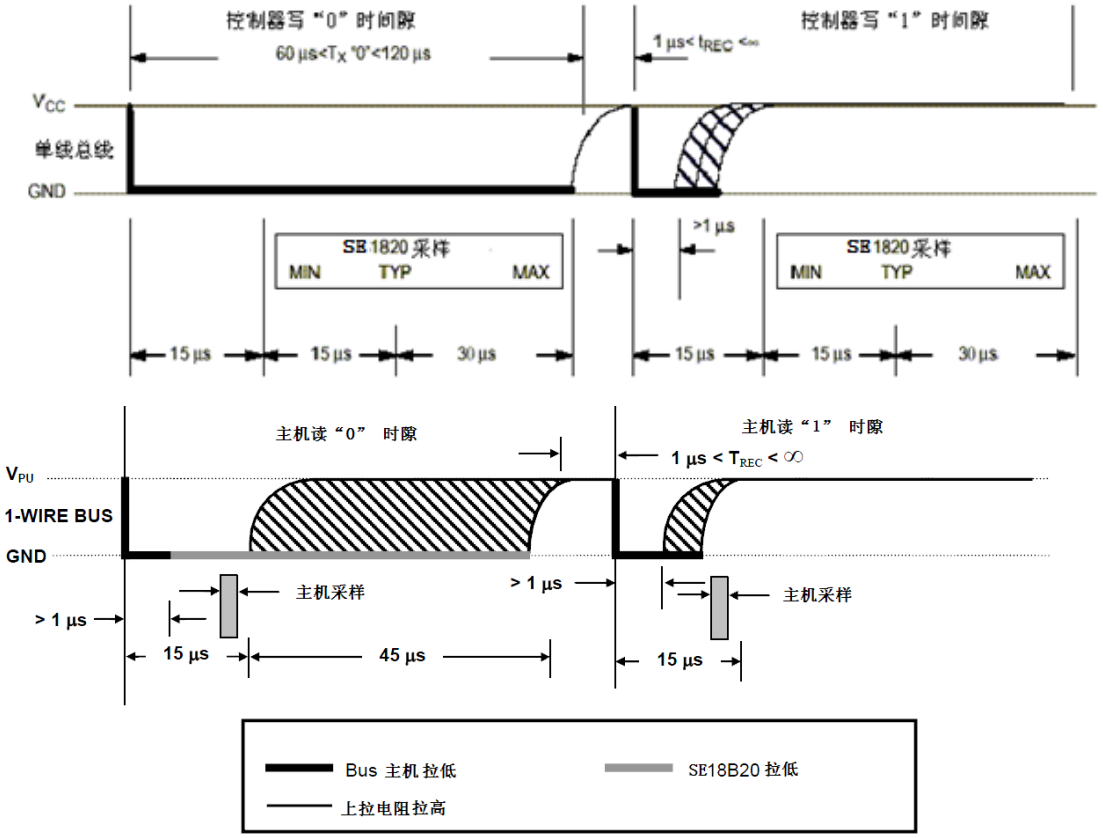


图 8

最大安全操作值

任何管脚到地的电压　–0.5V to +6.0V
工作温度　　–55°C to +100°C
存储温度　　–55°C to +125°C

直流特性　　（-55 °C到100°C　V_{PU}=3.0V到5.5V）

参数	符号	条件	最小	典型	最大	单位
上拉电源	V_{PU}		3.0		5.5	V
温度误差	t_{ERR}	-10 °C到+85°C			±0.5°C	°C
		-55°C到+100°C			±2°C	°C
逻辑低	V_{IL}		-0.3		+0.8	V
逻辑低	V_{IH}		3.0		5.5	V
拉电流	I_L	$V_{I/O}$ =0.4V	4.0			mA
工作电流	I_{DQA}			1	1.5	mA
DQ输入电流	I_{DQ}			5		uA

表11

交流特性　　（-55 °C到100°C　V_{PU}=3.0V 到5.5V）

参数	符号	条件	最小	典型	最大	单位
温度转换时间	t_{CONV}	9位			3.75	ms
		10位			7.5	ms
		11位			15	ms
		12位			30	ms
上拉延迟	t_{SPON}	温度转换开始或复制暂存器			10	us
时隙	t_{SLOT}		60		120	us
恢复时间	t_{REC}		1			us
写0低时间	r_{LOW0}		60		120	us
写1低时间	r_{LOW1}		1		15	us
有效读时间	t_{RDV}				15	us
复位高时间	t_{RSTH}		480			us
复位低时间	t_{RSTL}		480		960	us
存在检测高	t_{PDHIGH}		15		60	us
存在检测低	t_{PDLOW}		60		240	us
电容	$C_{IN/OUT}$				25	pF

表12

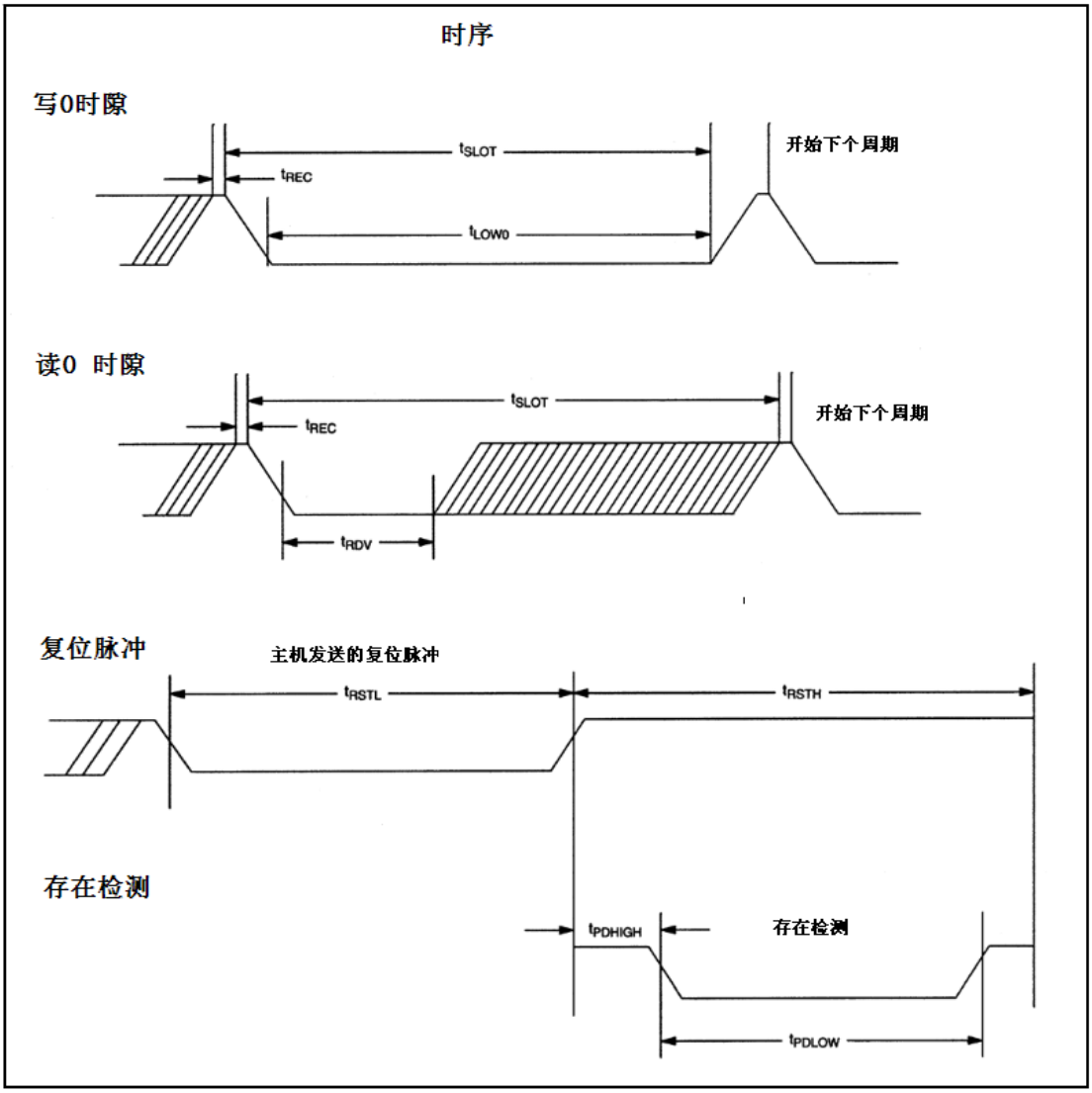
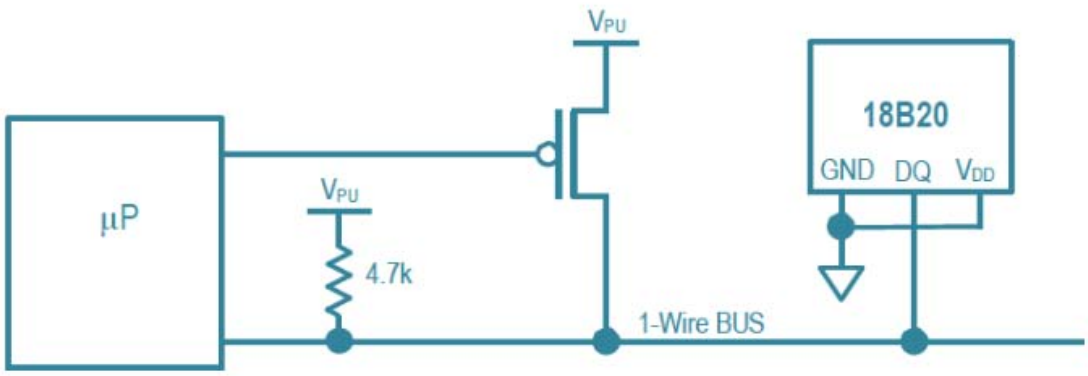
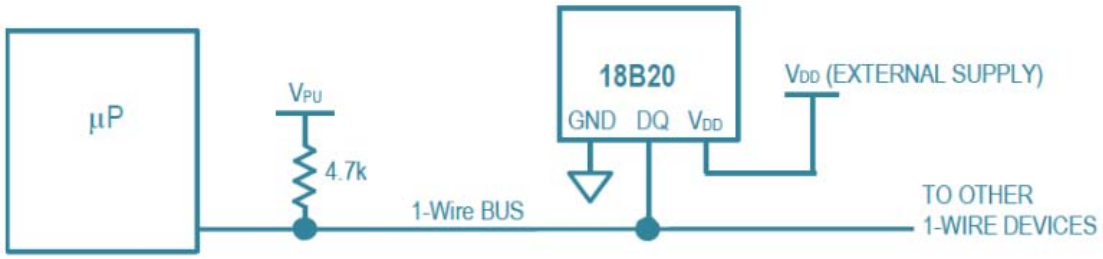

时序
写0时隙
tSLOT
tREC
tLOW0
开始下个周期
读0 时隙
tSLOT
tREC
tRDV
开始下个周期
复位脉冲
主机发送的复位脉冲
tRSTL
tRSTH
存在检测
tPDHIGH
存在检测
tPDLOW

图 9

应用电路


VPU
18B20
GND DQ VDD
µP
VPU
4.7k
1-Wire BUS

寄生模式



正常模式

```
/************************************************************
*JG1820 驱动程序
*版本:V1.0
************************************************************/
#include <reg52.h>
#define U8 unsigned char
sbit SE1820_DQ= P1^4; //单总线引脚
void SE18B20_Init() ; // SE18B20 初始化
bit SE1820_Reset(); // SE1820 复位
void SE1820_WriteData(U8 wData); //写数据到SE1820
U8 SE1820_ReadData(); //读数据
/************************************************************
* SE18B20 初始化
*函数名称: SE1820_WriteData()
*说明：本初始化程序可以不要，因为SE18B20 在出厂时就被配置为12 位精度了
************************************************************/
void SE18B20_Init()
{
SE1820_Reset();
SE1820_WriteData(0xCC); // 跳过ROM
SE1820_WriteData(0x4E); // 写暂存器
SE1820_WriteData(0x20); // 往暂存器的第三字节中写上限值
SE1820_WriteData(0x00); // 往暂存器的第四字节中写下限值
SE1820_WriteData(0x7F); // 将配置寄存器配置为12 位精度
SE1820_Reset();
}
/************************************************************
*SE1820 复位及存在检测(通过存在脉冲可以判断SE1820 是否损坏)
*函数名称:SE1820_Reset()
*说明:函数返回一个位标量(0 或1)flag=0 存在,反之flag=1 不存在
************************************************************/
bit SE1820_Reset()
{
U8 i;
bit flag;
```

```
SE1820_DQ = 0; //拉低总线
for (i=240;i>0;i--); //延时480 微秒,产生复位脉冲
SE1820_DQ = 1; //释放总线
for (i=40;i>0;i--); //延时80 微秒对总线采样
flag = SE1820_DQ; //对数据脚采样
for (i=200;i>0;i--); //延时400 微秒等待总线恢复
return (flag); //根据flag 的值可知SE1820 是否存在或损坏 ，可加声音告警提示SE1820 故障
}
/*************************************************************
*写数据到SE1820
*函数名称:SE1820_WriteData()
*************************************************************/
void SE1820_WriteData(U8 wData)
{
U8 i,j;
for (i=8;i>0;i--)
{
SE1820_DQ = 0; //拉低总线,产生写信号
for (j=2;j>0;j--); //延时4us
SE1820_DQ = wData&0x01; //发送1 位
for (j=30;j>0;j--); //延时60us,写时序至少要60us
SE1820_DQ = 1; //释放总线,等待总线恢复
wData>>=1; //准备下一位数据的传送
}
}
/*************************************************************
*从SE1820 中读出数据
*函数名称:SE1820_ReadData()
*************************************************************/
U8 SE1820_ReadData()
{
U8 i,j,TmepData;
for (i=8;i>0;i--)
{
TmepData>>=1;
SE1820_DQ = 0; //拉低总线,产生读信号
for (j=2;j>0;j--); //延时4us
SE1820_DQ = 1; //释放总线,准备读数据
for (j=4;j>0;j--); //延时8 微秒读数据
if (SE1820_DQ == 1)
{ TmepData |= 0x80;}
for (j=30;j>0;j--); //延时60us
SE1820_DQ = 1; //拉高总线,准备下一位数据的读取.
}
```

```
return (TmepData);//返回读到的数据
}
/************************************************************
*采用SE1820+1602 精确到4 位小数的温度计
*编程:XXX
*版本:V1.1
*SE1820 设置为12 位精度
************************************************************/
#include <reg52.h>
#include <temp.h>
U8 code string[16]={"Temp Display"};
U8 code DispStr[16]={"www.sxcckj.com"};
U8 temperature[2]; //存放温度数据
void display(); //待显温度值转换
/************************************************************
*主程序
************************************************************/
void main()
{
U8 i;
DelayMs(255);//等待电源稳定,液晶复位完成
LCDInit(); //1602 液晶初始化
SE18B20_Init(); //18B20 初始化，可不用初始化，因为18B20 出厂时默认是12 位精度
SetLight(1); //打开背光灯
DisplayStr(0,0,string);//显示"温度:"
DisplayStr(0,1,DispStr);//显示"系列号"
DelayMs(250);
DelayMs(250);
DelayMs(250);
DelayMs(250);
DelayMs(250);
DelayMs(250);
DelayMs(250);
DelayMs(250);
DelayMs(250);
DelayMs(250);
WriteCMD(0x01,1); //1601 清屏
DelayMs(50);
SetLight(0); //关闭背光灯
while (1)
{
SE1820_Reset(); //复位
SE1820_WriteData(0xcc); //跳过ROM 命令
SE1820_WriteData(0x44); //温度转换命令
```

```
SE1820_Reset(); //复位
SE1820_WriteData(0xcc); //跳过ROM 命令
SE1820_WriteData(0xbe); //读SE1820 温度暂存器命令
for (i=0;i<2;i++)
{
temperature[i]=SE1820_ReadData(); //采集温度
}
SE1820_Reset(); //复位,结束读数据
display(); //显示温度值
DelayMs(50);
}
}
/**********************************************************
*转换显示子程序
**********************************************************/
void display()
{
U8 temp_data,temp_data_2;
U8 temp[7]; //存放分解的7 个ASCII 码温度数据
U16 TempDec; //用来存放4 位小数
temp_data = temperature[1];
temp_data &= 0xf0; //取高4 位
if (temp_data==0xf0) //判断是正温度还是负温度读数
{
//负温度读数求补,取反加1,判断低8 位是否有进位
if (temperature[0]==0)
{ //有进位,高8 位取反加1
temperature[0]=~temperature[0]+1;
temperature[1]=~temperature[1]+1;
}
else
{ //没进位,高8 位不加1
temperature[0]=~temperature[0]+1;
temperature[1]=~temperature[1];
}

}
temp_data = temperature[1]<<4; //取高字节低4 位(温度读数高4 位)，注意此时是12 位精度
temp_data_2 = temperature[0]>>4; //取低字节高4 位(温度读数低4 位)，注意此时是12 位精
度
temp_data = temp_data|temp_data_2; //组合成完整数据
temp[0] = temp_data/100+0x30; //取百位转换为ASCII 码
temp[1] = (temp_data%100)/10+0x30; //取十位转换为ASCII 码
temp[2] = (temp_data%100)%10+0x30; //取个位转换为ASCII 码
```

```
temperature[0]&=0x0f; //取小数位转换为ASCII 码
TempDec = temperature[0]*625; //625=0.0625*10000，表示小数部分,扩大1 万倍 ，方便显示
temp[3] = TempDec/1000+0x30; //取小数十分位转换为ASCII 码
temp[4] = (TempDec%1000)/100+0x30; //取小数百分位转换为ASCII 码
temp[5] = ((TempDec%1000)%100)/10+0x30;//取小数千分位转换为ASCII 码
temp[6] = ((TempDec%1000)%100)%10+0x30;//取小数万分位转换为ASCII 码
if(temp[0]==0x30) DisplayChar(3,0,' ');//如果百位为0，显示空格
else DisplayChar(3,0,temp[0]); //否则正常显示百位
DisplayChar(4,0,temp[1]);//十位
DisplayChar(5,0,temp[2]);//个位
DisplayChar(6,0,0x2e); //小数点 .
DisplayChar(7,0,temp[3]);
DisplayChar(8,0,temp[4]);
DisplayChar(9,0,temp[5]);
DisplayChar(10,0,temp[6]);
DisplayChar(11,0,'\''); //显示'
DisplayChar(12,0,'C'); //显示C
}
```

文件名称 Doc. Title： SE18B20 芯片使用手册			文件编号 Doc. NO.： XSEC_ SE18B20	
修改记录表 Change History				
版次 Version	攥写者 Author	变更原因 Change Cause	叙述 Description	生效日期 Effective Date
V1.1	赵新毅	新定义的说明书	-	2018/10/24
V1.2	赵新毅	根据测试，修改温度转换的时间	$T_{CONV}/8$＝3.75mS $T_{CONV}/4$＝7.5mS $T_{CONV}/2$＝15mS $T_{CONV}/1$＝30mS	2020/06/16