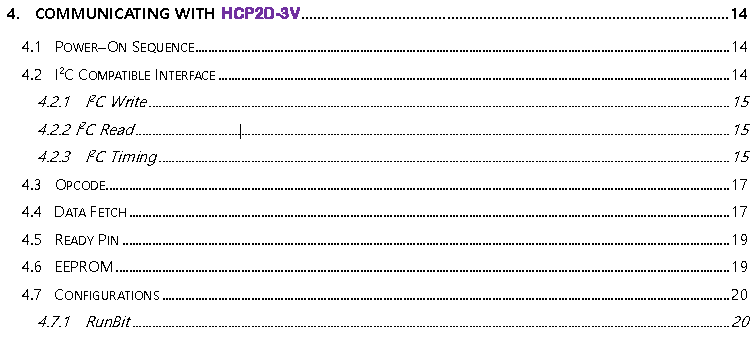
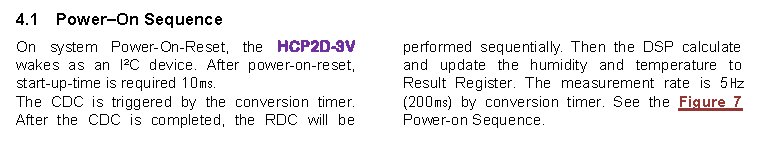
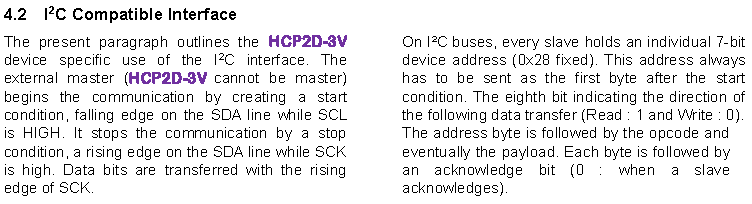


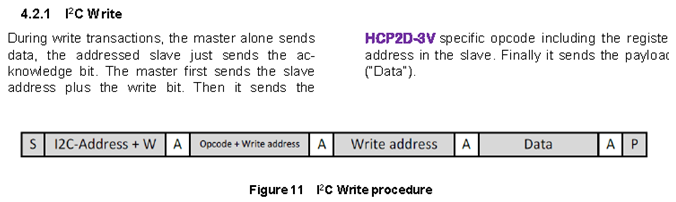
2.4 ,准备脚  
准备脚的信号表示输出数据是否有效.它可以用来连接到MCU的中断脚或者GPIO.假如这个脚没有用的话,断开不连即可

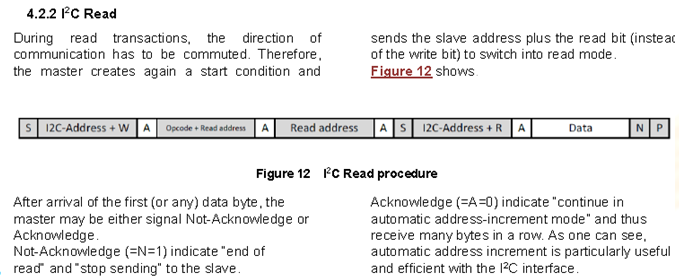




4.1 上电顺序  
在系统，芯片醒来作为一个I2C器件.在电源-开启-复位之后,启动时间需要10ms. CDC通过转换计时器来触发.CDC完成之后,RDC会继续地执行.然后DSP计算和更新温度和湿度到结果寄存器.测试率是5hz (200ms),通过转换计时器,见图7上电顺序  


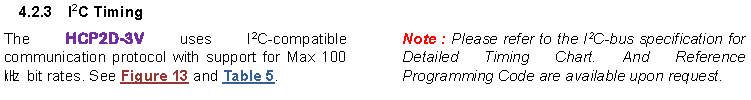
4.2 I2C兼容的接口  
目前的图片展示了芯片在I2C情况下的特定应用.外部主器件(此芯片不能作为主器件)通过产生一个启动条件来开始通信,SDA线的下降沿,在SCL是高的时候.通过一个停止条件来停止通信,SDA上面的上升沿,在SCK是高的时候.数据比特是在SCK的上升沿传送的.在I2C总线上面,每个从设备保留一个单独的7比特器件地址(0x28固定的).这个地址一直必须发送作为第一个字节,在启动条件之后.第8个字节表示下面数据发送的方向(read: 1和写:0).地址字节是Opcode跟随的最终载重.每个字节是跟随一个知悉字节(0:当一个从器件的知悉)

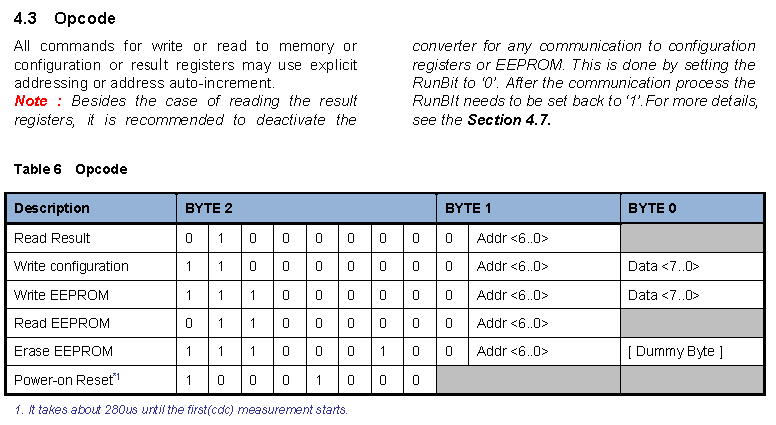
  
4.2.1 I2C写  
在写的过程中,主设备单独发送数据,寻址的从设备就只发送知悉比特.主设备首先发送从地址加上写比特.然后发送芯片特定的Opcode包括在从地址的寄存器地址，最后发送数据。

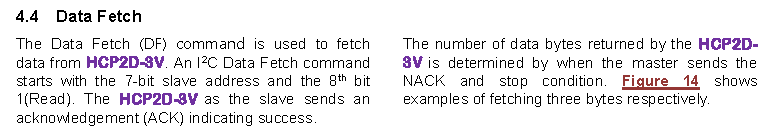


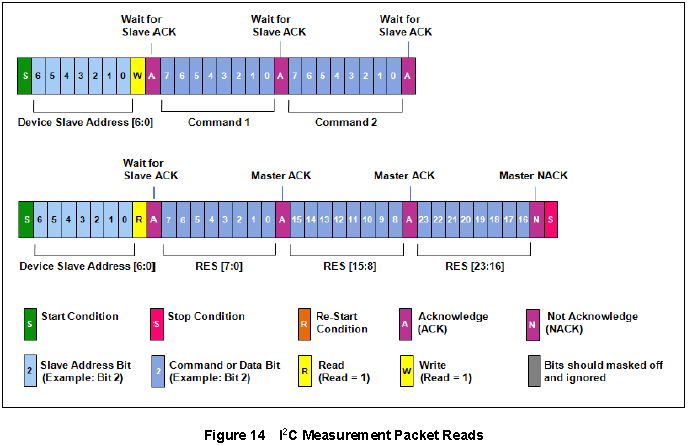
**4.2.2 I2C读**

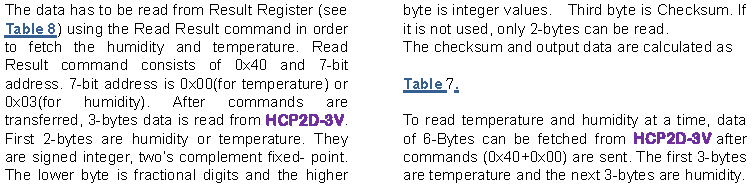
在读取过程中，通信的方向必须是交换。那么，主设备再次产生一个启动条件然后发送从设备地址加上读取比特（不再是写的比特）到开关进入读取模式，图12显示了在第一个(或者一些)数据比特达到之后，主设备要么信号没有知悉(=N=1)要么知悉了。没有知悉表示“读取结束”和”停止发送”到从设备。知悉(=A=0)表示”连续自动寻址增量模式”继而连续接受很多的比特。如我们看到的，I2C接口下, 自动寻址增量是特别有用的和有效的.

  
**4.2.3** I2C时序  
此芯片使用I2C兼容的通讯协议,支持最大100Khz的比特率,如图13和表5.**注意**:参考I2C总线规格关于详细的时序图.参考编程代码请联系我们来获得

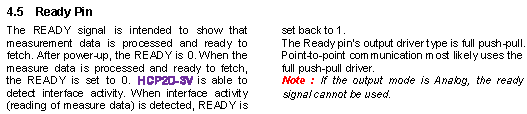
  
**4.3 OPCODE**

可以使用显示寻址或者寻址自动增量将所有的用于读或者写的命令到存储器或者配置寄存器或者结果寄存器。**注意**：除了读取结果寄存器这一种情况之外,我们也推荐解除转换器来用于任何的通信到配置寄存器或者EEPROM.这个是通过设置Runbit到”0”来实现的.通信过程完成之后”runbit”需要设置回”1”,欲知详细请参考4.7章  
  
  
**4.4** **数据推送**  
DF命令,是用来从芯片推送数据.一个I2C数据推送命令是从7bit从模式地址和第8个位1(读)开始的。此芯片作为从设备发送一个知悉就表示成功了.芯片返回的数据比特的号码是通过当主设备发送NACK和停止条件来确定的。图14显示了推送三个字节的各自的例子

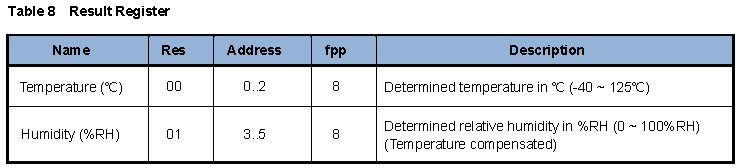
. 

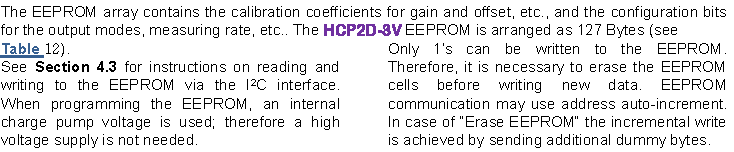


为了推送温度和湿度数据, 数据必须用’读取结果命令’从结果寄存器读取(见表8).读取结果命令由0x40组成和7位地址.7位地址是0x00(用于温度)或者)0x03(用于湿度).命令发送之后,从芯片读取到3比特的数据,开始的2比特是湿度或者温度.他们是带符号整数, 两个的补充固定点.低位是小数位,高位是整数位.第3字节是校验和。假如没有使用的话,只有2个字节能够读取到。校验和和输出数据如表格7来计算的。为了读取温度和湿度, 在命令(0x40+0x00)发送之后, 6比特的数据可从芯片推送.开始的3个字节是温度,下面3个字节是湿度.



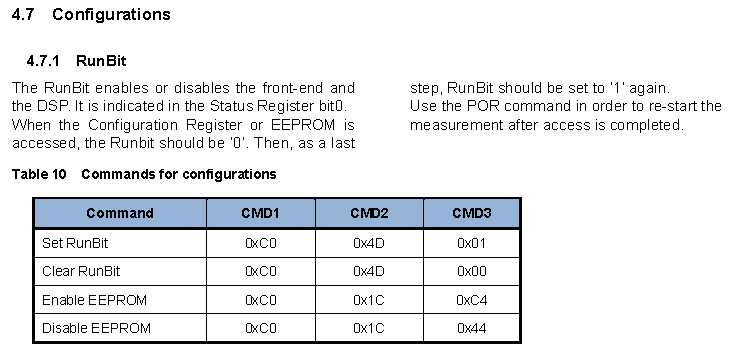
4.5**准备**信号是用来显示测试数据是处理了的,且准备推送.一旦上电之后,**准备脚**是”0”.当测试数据处理了且准备推送时,**准备脚**设置成”0”.此芯片能够检测**接口的活动**.当**接口活动**(读取测试数据)检测到了,准备脚就设置回“1”.**准备脚**输出驱动字节是满推挽式.点到点通信大多数类似使用满推挽驱动器.注意:假如输出模式是**模拟**的话,不能使用这个**准备信号**.

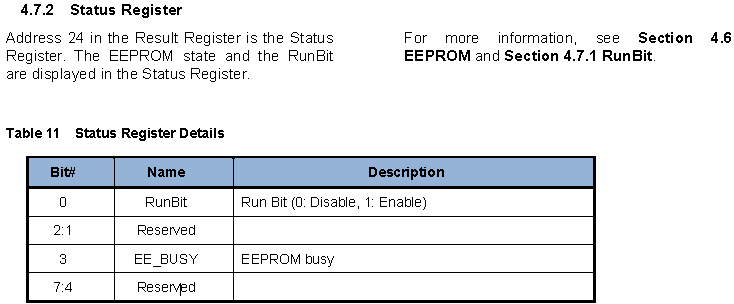




4.6 EEPROM

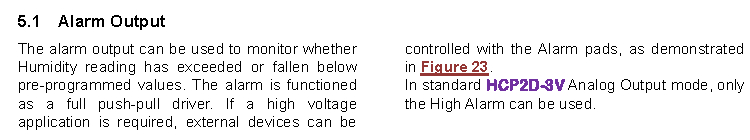
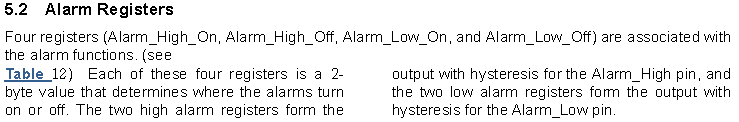
EEPROM阵列包含了校正因子用于增益和补偿等, 配置比特给输出模式, 测量速率等.此EEPROM是127字节配置的(表12), 见4.3章节关于通过I2C接口读和写到EEPROM的指令。当在编程EEPROM的时候,内部充电泵就开始使用了,继而就不需要一个高电压供应了.只有一次可以写入EEPROM,所以,有必要写入新数据之前擦除EEPROM.EEPROM的通信使用地址自动增量.假如”擦除EEPROM”增量写是通过发送额外的哑字节.  
  
在每次写的时候,有必要唤醒EEPROM.所以,在清除”run bit”之后将EEPROM使能.EEPROM唤醒可以自动或者明确地完成。对于每个单独的字节必须小心设置时序。

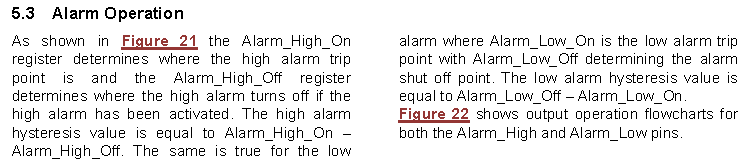


4.7 配置  
4.7.1 runbit  
这个命令使能或者关闭前端和DSP.表示状态寄存器 bit 0.当配置寄存器或者EEPROM被寻址时, runbit应该是”0”.然后作为最后一步, runBit应该再次设置到”1”,在寻址完成之后, 为了重启测量需要使用PoR命令.  
  


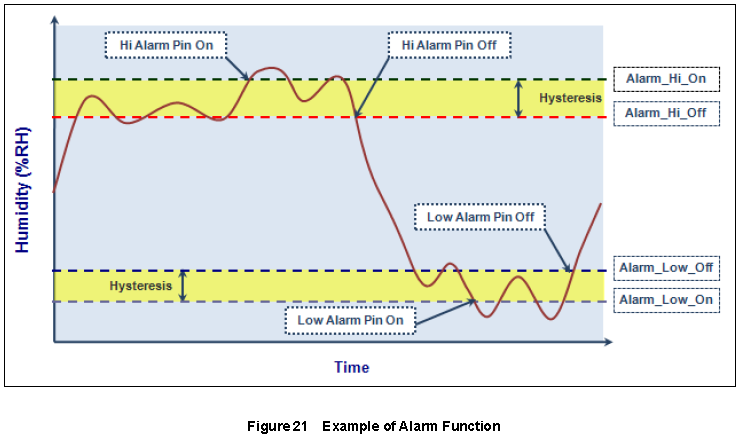
4.7.2 状态寄存器  
在结果寄存器里面的地址24是状态寄存器.EEPROM状态和Runbit显示在状态寄存器里面.要获得更多的信息,见4.6和4.7.1章节

**5. Alarm Function (Optional) 警报功能( 可选)**

****5.1警报输出  
警报输出可以用来监控湿度读取是否超过或者下跌到预编程的值.报警时作为一个圈推拉驱动器来工作的.假如一个高压应用北要求,外部设备可以用报警脚来控制,如图23展示的那样.在芯片的模拟输出模式下面,只有高报警能够使用  
****  
5.2报警寄存器  
四个寄存器(报警高-开启;报警高-关闭；报警低-开启；报警低-关闭)时和报警函数相关的（如表12）.这4个寄存器都是2个字节来确定报警是开启还是关闭。两个高报警寄存器用报警-高脚组成了滞后输出，两个低报警寄存器用报警-低脚组成了滞后输出。

****5.3 报警操作

如图21所示，假如高报警被激活的话，报警-高-开启寄存器确定高报警跳变点是在哪里, 报警-高-关闭寄存器确定高报警关闭的位置。高报警滞后值等于报警-高-开启减去报警-高关闭。同样的也适合于低报警-开启，报警-低-关闭确定关闭点。低报警滞后值=报警-低-关闭减去报警-低-开启。图22显示了报警-高和报警低脚的输出操作的流程图。



****此芯片也可以直接安装在没有MCU的场合,只有一个开关存在,达到想要的湿度水平（比如浴室排气扇，除湿机，加湿机）