Application Note

BQ79718, BQ78706, BQ79731 使用 I2C 与 EEPROM 通讯应用指南

Harrison Yang
Yang Wu

摘要

TI 用于可堆叠电池检测和均衡芯片 (简称 AFE) BQ797xx 和 BQ787xx 的 GPIO 支持配置成 I2C Controller，可用于 EEPROM、温湿度传感器以及其他 I2C 通讯芯片。AFE 接 EEPROM 可以将电池信息存储在电池包本地，可用于电池护照目的或者电池包梯次利用时更好地了解电池状态。BQ79731(简称 BJB) 接 EEPROM 可以存储 shunt 电阻阻值和温漂等参数，MCU 利用这些参数可以提高电流采样精度。

本应用笔记介绍了 EEPROM 的读操作和写操作协议，提醒用户需要注意 EEPROM 的地址回绕(Roll Over)、自写入等待时间(Self-Timed Write Cycle)以及 AFE 要求 I2C 读指令间隔必须<20ms 等应用问题。同时，本应用笔记也给出了 BQ79718 评估板连接 EEPROM 的硬件改板方案，以及评估板上位机 GUI 的 I2C 通讯软件指令，便于用户快速掌握和调试 AFE 和 BJB 的 I2C 通讯功能。

内容

插图清单

表格清单

1 AFE 和 BJB 的 I2C 功能介绍

图 1-1 是 BQ79718 手册中 I2C 应用框图，该架构同样适用于 78706 以及 79731。菊花链里每个 AFE 可以接多个 I2C 从节点。

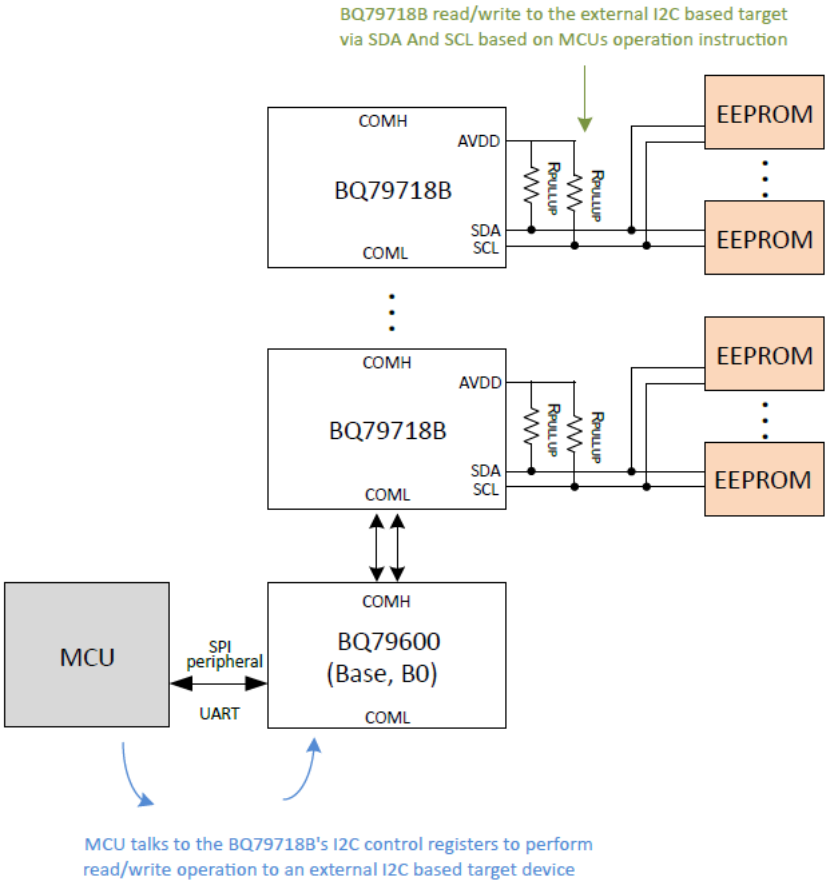

Figure 6-40. I²C Controller Function

图 1-1. BQ797xx I2C 应用示意图

I2C 数据收发主要依赖 I2C_CRTL, I2C_WR_DATA, I2C_RD_DATA 寄存器，在此之前还需要完成 GPIO 的功能配置(I2C 使能)。MCU 通过菊花链将数据写进 AFE 的寄存器 I2C_WR_DATA 里，读回的数据放在 I2C_RD_DATA 里。I2C_CTRL 设置 SEND, RECEIVE, START, STOP, NACK 以及启动 I2C 通讯。

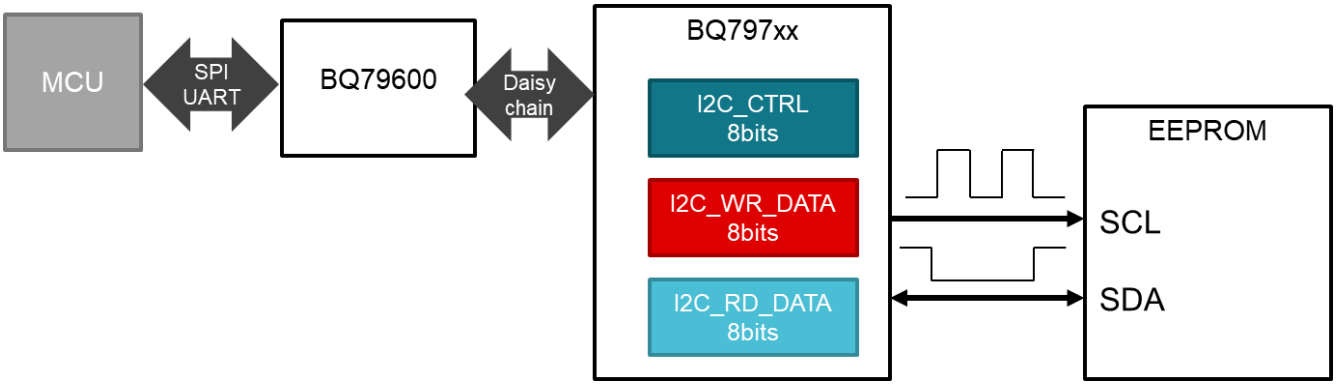

图 1-2. AFE 和 BJB I2C 相关寄存器

I2C_WR_DATA: 存放写进 EEPROM 的数据。

I2C_RD_DATA: 存放从 EEPROM 读出来的数据。

I2C_CTRL: I2C 协议中数据前后的 START, NACK, STOP 通过这个寄存器设置。同时还定义这次通讯是发送还是接收，以及启动 I2C 通讯。

NAME	ADDR	TYPE	RESET VALUE	DATA							
				B7	B6	B5	B4	B3	B2	B1	B0
I2C_CTRL	0x371	R/W-AC	0x00	RSVD	RSVD	SEND	RECEIVE	START	STOP	NACK	I2C_GO

图 1-3. I2C_CTRL 寄存器定义

主节点(AFE 或 BJB)控制 START 和 STOP 来告诉 EEPROM 开始通讯和通讯结束。

从节点(EEPROM)收到 8bit 数据后拉低 SDA 1bit(ACK)告诉 Host 数据接收完成可以发下一个字节。

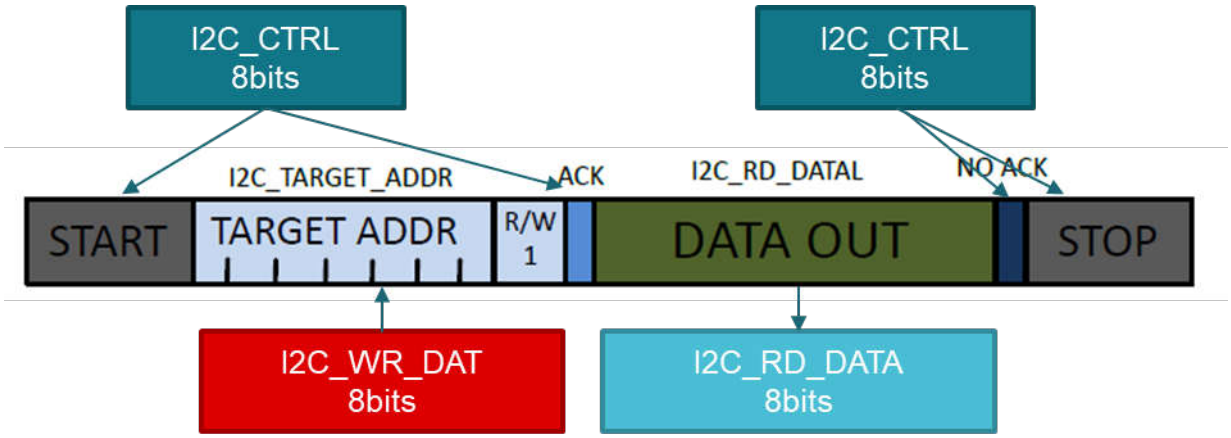

图 1-4. I2C 通讯协议与寄存器对应关系

I2C 详细通讯协议请看应用手册 *Understanding the I2C Bus* 。

读 EEPROM 通讯过程分为以下 7 个步骤:

- Step 1: MCU 通过菊花链将数据放到 AFE 的寄存器 I2C_WR_DATA 中。
- Step 2: MCU 写 I2C_CTRL[I2C_GO] =1 让 AFE 进行 I2C 通讯。
- Step 3: AFE 接收到指令后开始和 EEPROM 进行 I2C 通讯。
- Step 4: 发送 I2C 读指令。
- Step 5: I2C 返回的数据放在 I2C_RD_DATA 中。
- Step 6: MCU 读出 AFE 寄存器数据。
- Step 7: MCU 读取 AFE 的 FAULT_SYS 寄存器，确认本次通讯没有发生异常。

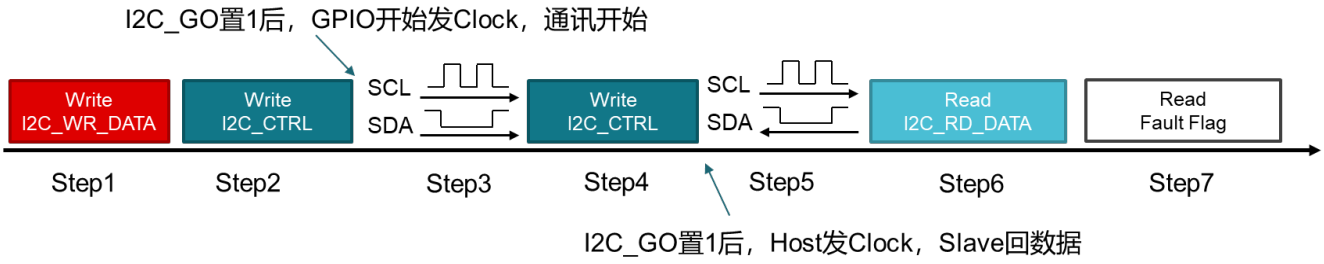

图 1-5. I2C 读 EEPROM 通讯 7 步流程示意图

2 AFE 和 BJB 的 I2C 硬件介绍

AFE 的 GPIO0(SCL)和 GPIO1(SDA) 用于 I2C 通讯, BQ79731 的 GPIO9(SCL)和 GPIO10 (SDA) 用于 I2C 通讯。这两个引脚默认是其他用途，需要写 CONTROL2[I2C_EN] =1 来使能 I2C 功能。SDA 和 SCL 是开漏输出，上拉电阻越小，高电平上升越快，通讯频率越高。上拉电阻小的缺点是功耗大，需要用户在速率和功耗之间做权衡。I2C 不通讯时 GPIO 为高电平，不通讯时没有功耗。AFE 和 BJB 的 I2C 最快通讯频率 400kHz，上拉电阻 RPULLUP 允许的最小值是 2kohm。

上拉电阻的计算方法参考应用手册 *I2C Bus Pullup Resistor Calculation*。

使用 BQ79718 评估板和 TI 上位机调试 I2C 功能需要做如下改动:

焊接 U14 EEPROM，需确认引脚定义是否和原理图一致。

焊接 U14 电源电容 C37。

焊接 R94 和 R184 0R 跳线。

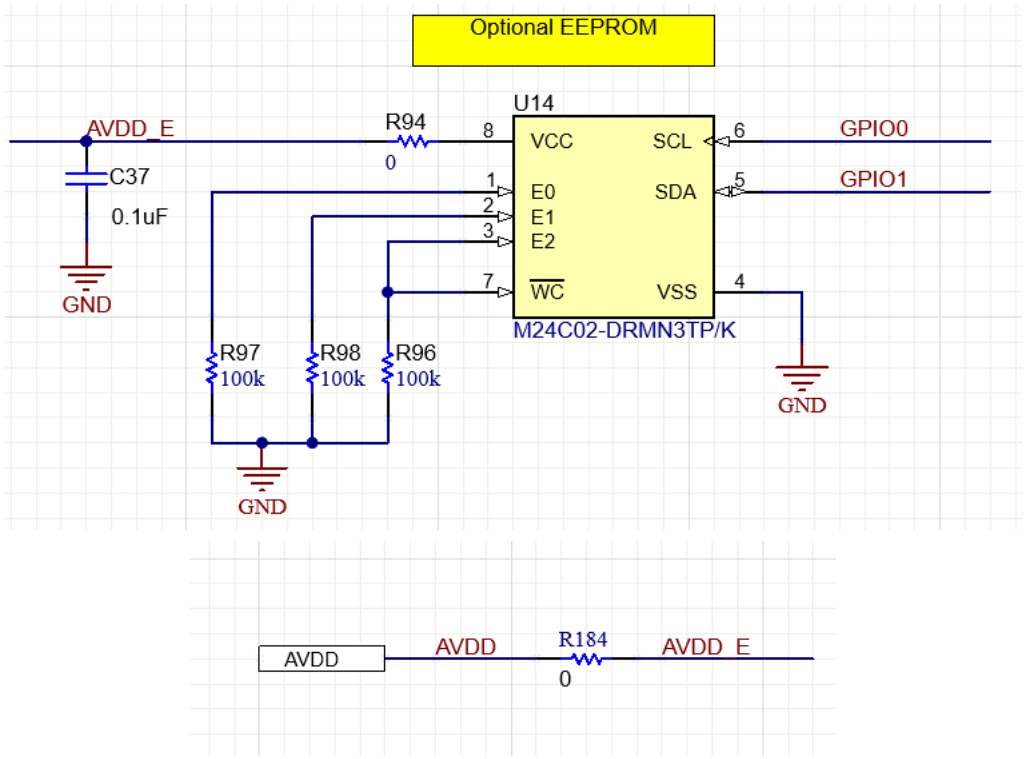

图 2-1. BQ79718 评估板 EEPROM 部分原理图

GPIO0/1 飞线焊接上拉电阻 2K 接到 AVDD。评估板上没有预留上拉电阻封装。

去掉跳线 R62, R64, 焊接跳线 R61, R63。

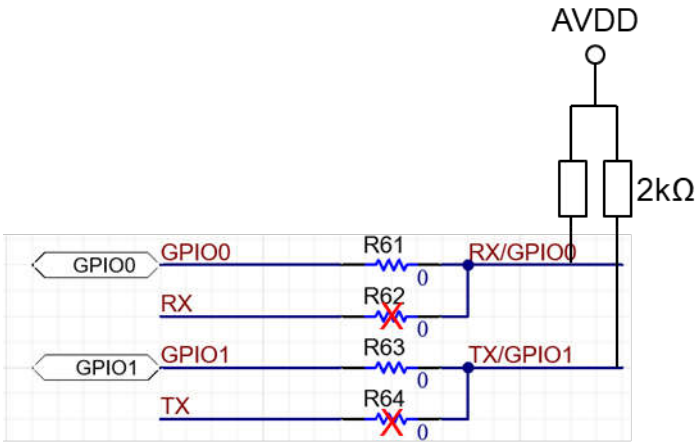

图 2-2. BQ79718 评估板 EEPROM 原理图

3 I2C 通讯指令实操

3.1 EEPROM 写操作

案例 1: 将设备 ID 为 000 的 EEPROM 地址范围 0x0000~0x0004,分别写入 0x00，0x01，0x02，0x03，0x04，0x05.

写指令有三部分组成：设备 ID，寄存器起始地址，写入的数据。

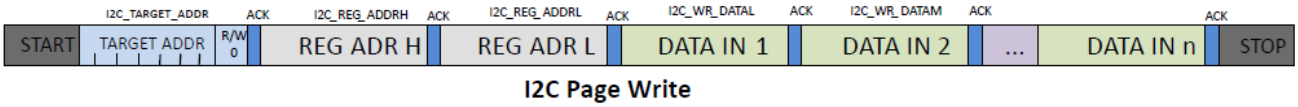

图 3-1. EEPROM 写指令协议

设备 ID: TARGET ADDR

阅读 EEPROM 手册地址定义，绝大部分 EEPROM 地址定义如下:

设备类型固定为 0b1010。 设备地址由 A2~A0 的引脚状态决定。 Bit0 在写操作是 0，读操作时 1。

表 3-1. EEPROM 设备 ID 定义

Device Type Identifier				Device Address Bits			R/W
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	1	0	A2	A1	A0	R/W

以下图为例：

写操作时，ROM0 地址 0xA0 (0b1010 000 0)， ROM1 地址 0xA2 (0b1010 001 0)

读操作时，ROM0 地址 0xA1 (0b1010 000 1)， ROM1 地址 0xA3 (0b1010 001 1)

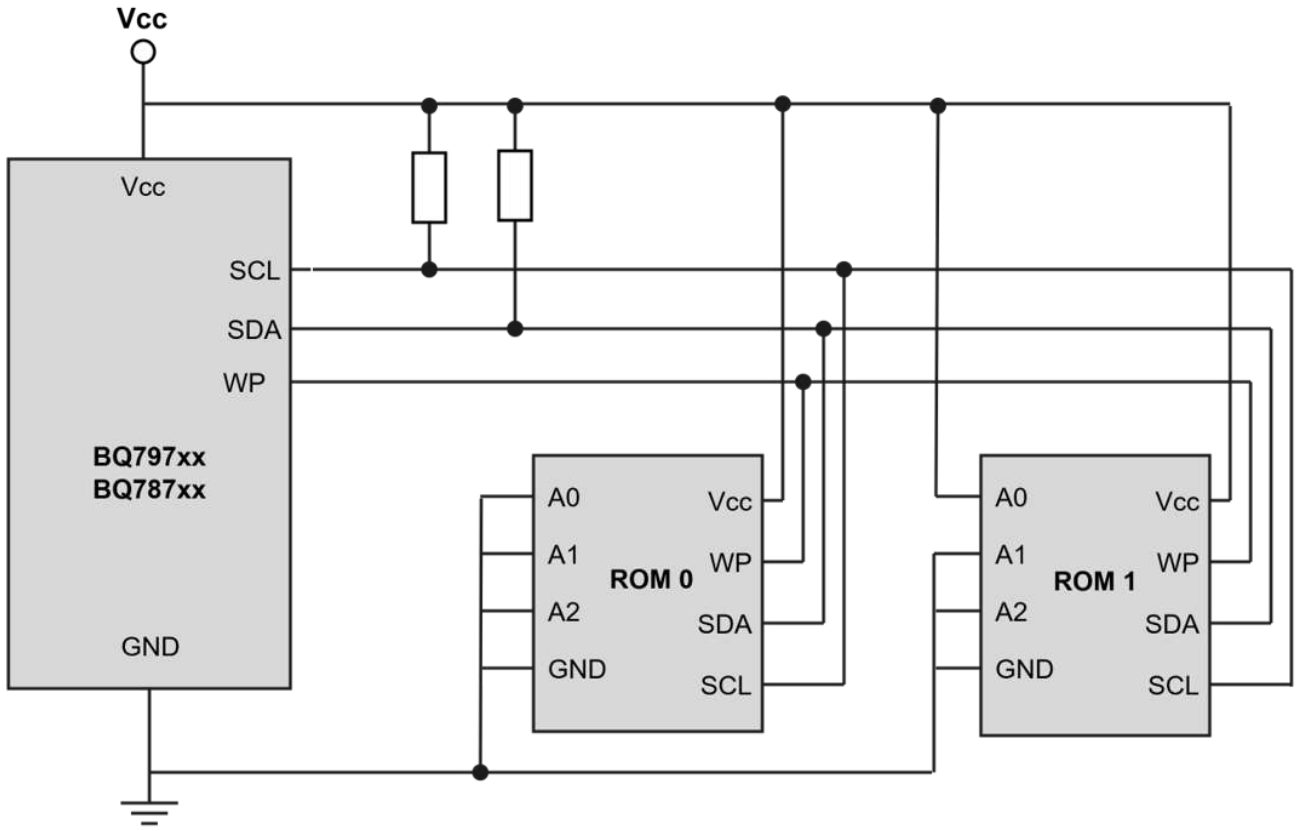

图 3-2. EEPROM 地址示意图

寄存器地址的起始地址 REG ADR H/L:

可以从任意地址开始写，**连续写入字节不能超过一个 Pape 的长度**。EEPROM 知道起始地址后，每写入一个字节，地址会自动+1, 再写入下一个字节。但地址不会跨页累加，否则会产生地址回绕问题（Roll Over），写入的字节数超过了当前 Page 的剩余空间，地址计数器可能会自动 "回绕" 到该 Page 的开头，导致之前的数据被意外覆盖。

不同芯片的 Page 大小不一样，查看手册搜"byte page"可以找到 Page 大小。比如是 32byte-page，那么从 Page 起始位置 0x00 开始写，最多可以连续写 32 字节。如果在 Page 结束位置 0x1F 开始写, 只能写一个字节。

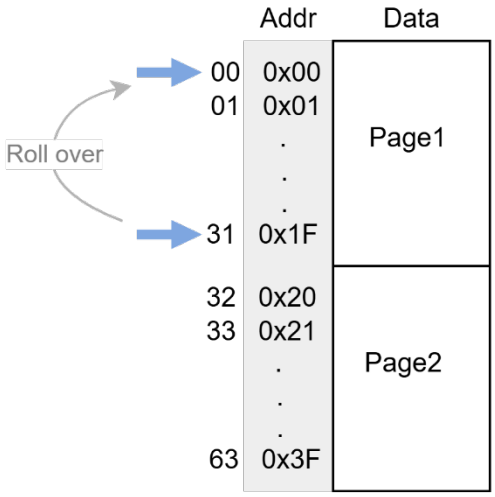

图 3-3. 寄存器地址自动累加到 Page 底部自动回绕 Page 起始位置

大量数据跨页写入时，需拆分成多条写指令。**每条写指令需间隔 5ms 以上**，这是 EEPROM 数据手册的要求: Self-Timed Write Cycle within 5 ms Maximum。这个时间用于 EEPROM 擦除、电荷泵加压，数据写入和验证的一套复杂内部流程。

下面是连续写入 6 个字节的指令

```
# 使能 I2C，将 GPIO0 和 GPIO1 设置为 SCL 和 SDA.
B0 03 0A 05 10 47
# I2C_WR_DATA 写设备 ID 0xA0 。I2C_CTRL 写 0x29，SEND-START-I2C_GO
B1 03 70 A0 29 E0 5B
# ADDH=00.  SEND-I2C_GO
B1 03 70 00 21 99 9D
# ADDL=00. SEND-I2C_GO
B1 03 70 00 21 99 9D
#写 1~5 个字节
B1 03 70 00 21 99 9D
B1 03 70 01 21 98 0D
B1 03 70 02 21 98 FD
B1 03 70 03 21 99 6D
B1 03 70 04 21 9B 5D
# 写最后一个字节，  SEND-STOP-I2C_GO
B1 03 70 05 25 9B 0E
# 读故障寄存器，检查 I2C_LOW 和 I2C_NACK 是否有故障。
A0 05 3D 00 22 B5
```

在 TI 评估板上位机输入上面指令执行一次 I2C 通讯，下图是逻辑分析仪记录的通讯数据。

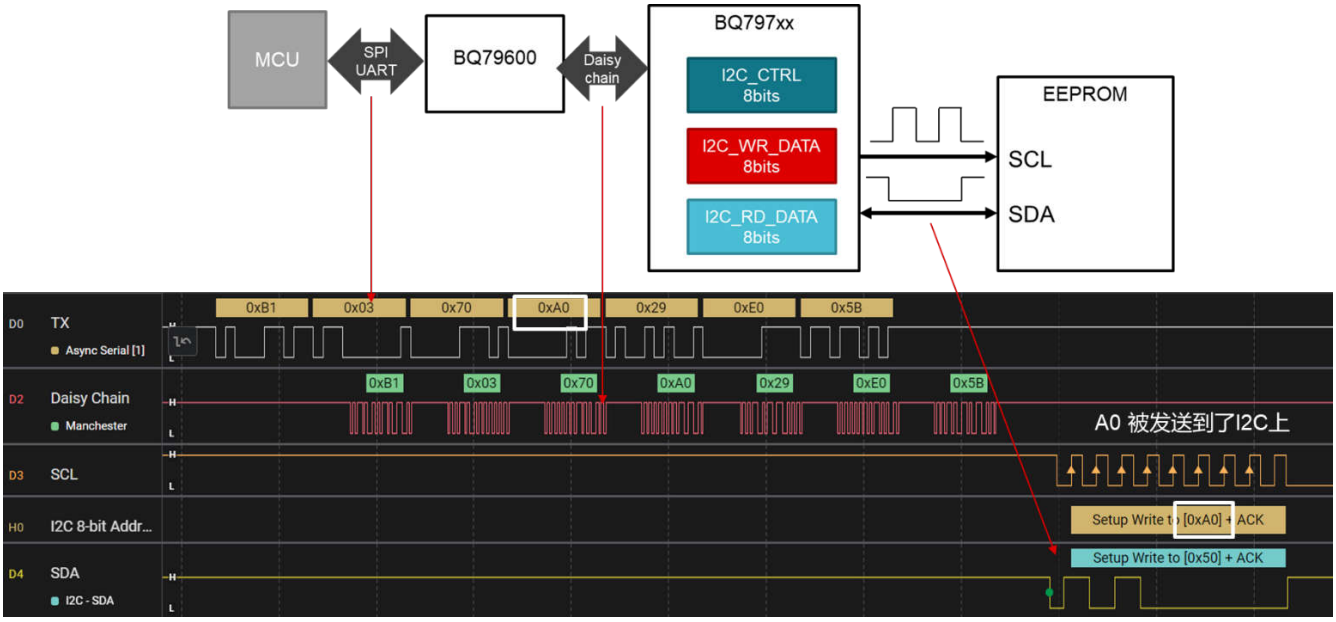

图 3-4. I2C 发送 A0 的逻辑分析仪数据解析

3.2 EEPROM 读操作

3.2.1 读取任意寄存器地址的一个字节

案例 2: 读取设备 ID 为 000 的 EEPROM 地址 0x0000 的数据

写指令流程: 先配置设备 ID 0xA0 和寄存器地址 0x0000，再写设备 ID 0xA1, 读取数据。

图 3-5. 读取一个字节协议定义

下面是读取一个字节的指令

```
# 使能 I2C，将 GPIO0 和 GPIO1 设置为 SCL 和 SDA.
B0 03 0A 05 10 47
# I2C_WR_DATA 写设备 ID 0xA0 。I2C_CTRL 写 0x29，SEND-START-I2C_GO
B1 03 70 A0 29 E0 5B
# ADDH=00.  SEND-I2C_GO
B1 03 70 00 21 99 9D
# ADDL=00. SEND-I2C_GO
B1 03 70 00 21 99 9D
#设备 ID A1; SEND-START-GO
B1 03 70 A1 29 E1 CB
# RECEIVE-STOP-NACK-I2C_GO
B0 03 71 17 B2 BA
# I2C 通讯已经结束，读取 I2C 传出来的数据。
A0 06 10 00 CF E5
# 读故障寄存器，检查 I2C_LOW 和 I2C_NACK 是否有故障。
A0 05 3D 00 22 B5
```

下图是 GUI 操作结果，读取结果 0x610 寄存器是 0x00。0x00 是案例 1 写进去的值，EEPROM 存储数据的默认值是 0xFF，说明案例 1 的写操作和案例 2 的读操作都成功了。

0x53D 结果只报了 DRST 故障，表示芯片发生过一次复位，这是初始化没有清除，导致芯片唤醒后一定会出现的故障。所以本次 I2C 通讯成功，没有任何异常。

COMMAND DIALOG

Run command sequence in loop

1. B0 03 0A 05 10 47
2. B1 03 70 A0 29 E0 5B
3. B1 03 70 00 21 99 9D
4. B1 03 70 00 21 99 9D
5. B1 03 70 A1 29 E1 CB
6. B0 03 71 17 B2 BA
7. A0 06 10 00 CF E5
8. A0 05 3D 00 22 B5

Command	Dev Addr (hex)	Reg Add (hex)	Length	CRC	Data (hex)
Stack Write	-	03 0A	1	10 47	05
Stack Write	-	03 70	2	E0 5B	A0 29
Stack Write	-	03 70	2	99 9D	00 21
Stack Write	-	03 70	2	99 9D	00 21
Stack Write	-	03 70	2	E1 CB	A1 29
Stack Write	-	03 71	1	B2 BA	17
Stack Read	-	06 10	1	CF E5	00
-	01	06 10	1	C8 3D	00
Stack Read	-	05 3D	1	22 B5	00
-	01	05 3D	1	24 AE	04

图 3-6. 左边为 TI 上位机指令编辑区域，右边为实际发送和接收的数据

3.2.2 连续读取任意地址多个字节

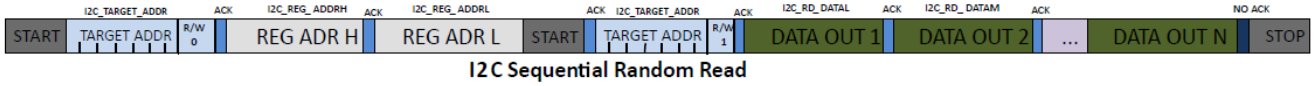

图 3-7. 读取多个字节协议

多字节读取和单字节读取的区别是读完一个字节没有 Stop，继续读下一个字节直到读最后一个字节设置 Stop。下面是连续写 5 个字节和连续读 16 个字节的例程。

```
# I2C Enable
B0 03 0A 05 10 47
######### 写操作指定 ID 和寄存器起始地址#########
# 写设备 ID A0
B1 03 70 A0 29 E0 5B
# Send, ADDH=00, ADDL=00, I2C GO
B1 03 70 00 21 99 9D
B1 03 70 00 21 99 9D
######### 连续写 5 个字节 #########
B1 03 70 00 21 99 9D
B1 03 70 01 21 98 0D
B1 03 70 02 21 98 FD
B1 03 70 03 21 99 6D
B1 03 70 04 21 9B 5D
# Write Data SEND-STOP-I2C GO
B1 03 70 05 25 9B 0E
# Read Fault
A0 05 3D 00 22 B5
######### 读操作指定 ID 和寄存器起始地址##########
# 设备 ID A0; SEND-START-GO
B1 03 70 A0 29 E0 5B
# Send, ADDH=00, ADDL=00, SEND-GO
B1 03 70 00 21 99 9D
B1 03 70 00 21 99 9D
## 读设备 ID A1; SEND-START-GO
B1 03 70 A1 29 E1 CB
######### 开始读 16 个字节 RECEIVE-GO #########
B0 03 71 11 32 B8 A0 06 10 00 CF E5
B0 03 71 11 32 B8 A0 06 10 00 CF E5
B0 03 71 11 32 B8 A0 06 10 00 CF E5
B0 03 71 11 32 B8 A0 06 10 00 CF E5
B0 03 71 11 32 B8 A0 06 10 00 CF E5
B0 03 71 11 32 B8 A0 06 10 00 CF E5
B0 03 71 11 32 B8 A0 06 10 00 CF E5
B0 03 71 11 32 B8 A0 06 10 00 CF E5
B0 03 71 11 32 B8 A0 06 10 00 CF E5
B0 03 71 11 32 B8 A0 06 10 00 CF E5
B0 03 71 11 32 B8 A0 06 10 00 CF E5
B0 03 71 11 32 B8 A0 06 10 00 CF E5
B0 03 71 11 32 B8 A0 06 10 00 CF E5
B0 03 71 11 32 B8 A0 06 10 00 CF E5
B0 03 71 11 32 B8 A0 06 10 00 CF E5
```

```
B0 03 71 11 32 B8 A0 06 10 00 CF E5
## I2C 读最后一个字节
B0 03 71 17 B2 BA
A0 06 10 00 CF E5
A0 05 3D 00 22 B5
```

上面这段代码有个特殊处理，把两个指令合并在一起发送。

正常指令如下

B0 03 71 11 32 B8 # I2C 开始读数据 1

A0 06 10 00 CF E5 # 菊花链读取返回数据 1

B0 03 71 11 32 B8 # I2C 开始读数据 2

A0 06 10 00 CF E5 # 菊花链读取返回数据 2

特殊处理指令如下

B0 03 71 11 32 B8 A0 06 10 00 CF E5

B0 03 71 11 32 B8 A0 06 10 00 CF E5

这样处理的原因是上位机的指令间隔太长，加上 AFE 的复位机制，如果不做特殊处理，AFE 会进入 Shutdown Mode。

蓝色读数据 1 指令执行完，CLK/GPIO0 会处于低电平状态。直到下一次绿色读数据 2，CKL/GPIO0 才会拉高。GPIO0 低电平时间超过 20ms 会强制 AFE 进入 Shutdown Mode。I2C 通讯和菊花通讯全部中断，需要重新唤醒和初始化才能正常工作。

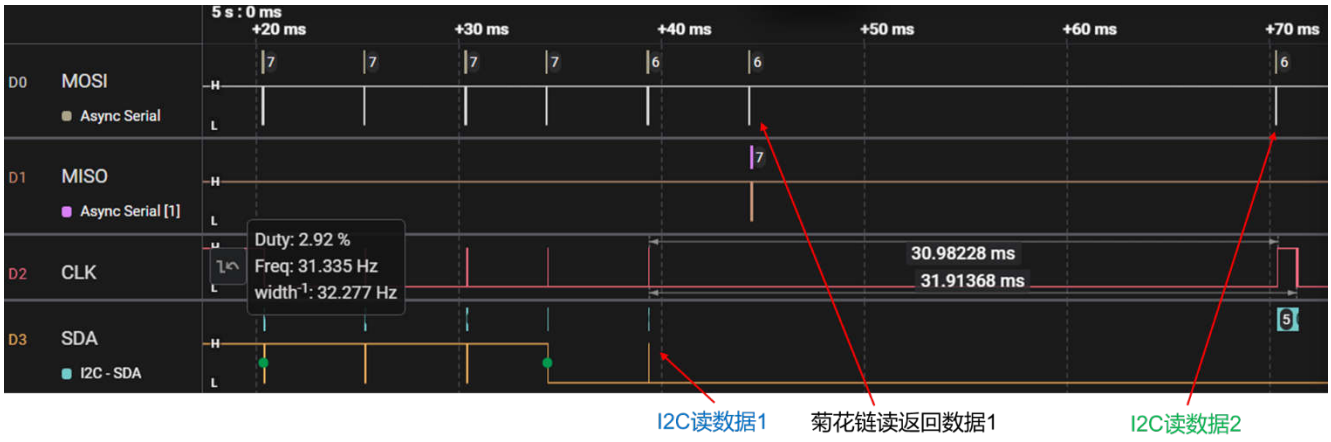

图 3-8. 两次 I2C 读指令间隔 31ms 导致 AFE 误识别成硬件复位

AFE 的 GPIO0 复用的功能是硬件复位 (Hardware Reset) AFE。当 AFE 用 UART 和 MCU 通讯时，GPIO0 复用为 RX 引脚，接收 MCU 发过来的指令。GPIO0 拉低时间超过 20ms，AFE 会认为这是 MCU 发来的硬件复位指令，执行完复位流程后，AFE 进入 Shutdown Mode。

Note

The [I2C_EN] setting is ignored in the base device (device woken up by PING) and device UART communication continues to work even if [I2C_EN=1] in the base device. Stack devices (woken up by TONE) block UART comm clear/ Sleep2Act ping/WAKE ping ONLY if the I^2C feature is enabled ([I2C_EN] =1). When GPIO0 is used as I^2C SCL, SCL must not be low for more than 20 ms. Otherwise, it may mis-trigger SD_HW_RESET ping.

图 3-9. BQ79718 手册中 I2C 通讯注意事项

为了避免这个问题，MCU 在连续读 I2C 数据时，两次读指令间隔要小于 20ms，建议<15ms 多留一些余量。AFE 手册中如上图的 Note 部分，也提醒用户 SCL 拉低时间不能超过 20ms，否则可能会被误识别试硬件复位信号。

GUI 连续读取数据时，需要按照上面实例代码特殊处理的方式，将两个条指令合在一起发送，这样读数据 1 和数据 2 的指令间隔大约是 5ms，不会被识别成硬件复位信号。缺点是上位机不会把特殊指令当成读指令，返回的数据不会在右边的 LOG 框里显示。需要用逻辑分析仪解析 I2C 总线上的返回数据来确认是否操作成功。

4 总结

本文讲解了 I2C 通讯的要点和注意事项。给出 EEPROM 写入和读取上位机指令代码，用户可以在 TI 上位机运行实例直观了解此通讯过程，从而在掌握流程后快速地在自己的软件中实现这个功能。

5 参考文档

1. Understanding the I2C Bus
2. I2C Bus Pullup Resistor Calculation

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月