Application Note

BQ769x2 软件开发指南

Matt Sunna

摘要

本应用报告提供了电池监控器 BQ769x2 器件系列（包括 BQ76952、BQ76942 和 BQ769142）的通信数据包和序列示例。示例包括直接命令、子命令以及读取和写入 RAM 寄存器的位事务处理详细信息。示例还包括有关使用 *BQStudio 命令序列*面板执行这些读写事务的说明。还提供了简单的代码示例。本文档可与器件专用技术参考手册和数据表一起使用。节 7 中列出了这些文档。BQSTUDIO 软件还用于许多示例，并提供了一种查看所有器件寄存器的便捷方式。对于 BQ769x2 器件系列，要求 BQStudio 1.3.102 版本或更高版本。

BQ769x2 器件系列集成了三种不同的通信接口 - I²C、SPI 和单线 HDQ。I²C 和 SPI 接口包含一个可选的 CRC 校验。有关完整选项列表，请参阅器件专用数据表。本文档介绍了很多使用 I²C 接口的示例，还介绍了一些使用 SPI 接口且带 CRC 的相同示例。

内容

插图清单

表格清单

商标

所有商标均为其各自所有者的财产。

1 直接命令

完整的直接命令列表可在器件专用技术参考手册中找到。以下示例示出了直接命令的格式。

1.1 警报启用 - 0x66

表 1-1 展示了使用命令 0x66 的 警报启用 命令。默认情况下，将用于警报启用的寄存器设置为 0xF800。在本示例中，设置更改为 0xF082。数据采用小端格式。BQ769x2 的器件地址是 0x10（8 位），其中 LSB 是 R/W 位。直接命令遵循 *I2C_Write(I2C_ADDR, Command, DataBlock)* 格式，因此对于该示例，该命令为 *I2C_Write(0x10, 0x66, [0x82, 0xF0])*。

表 1-1. 警报启用命令说明

命令	名称	单位	类型	说明
0x66	警报启用	十六进制	H2	警报状态掩码()。可以在操作期间写入更改，以更改启用的警报源。

图 1-1. 为将警报启用设为 0xF082 捕捉的 I2C 波形

1.2 电池 1 电压 - 0x14

表 1-2 示出了如何读取 Cell 1 的电压。Cell 1 电压命令为 0x14，是一个只读命令。写入 I2C 命令 0x14 ，然后读取 2 字节，从而读取 Cell 1 的电压。数据以小端格式返回。在以下示例中，读取 16 位 Cell 1 电压为 0x0E74，对应于 3700mV。

表 1-2. Cell 1 电压命令说明

命令	名称	单位	类型	说明
0x14	Cell 1 电压	mV	I2	电池 1 上的 16 位电压

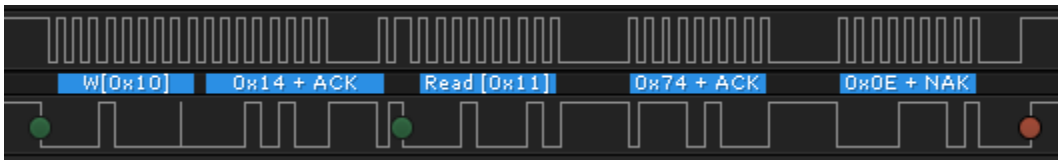

图 1-2. 为读取 Cell 1 电压捕捉的 I2C 波形

1.3 内部温度 - 0x68

表 1-3 示出了如何读取内部温度传感器。16 位温度传感器的读数单位为 0.1K。在以下示例中，0x0BA6 的读数表示十进制值 2982，即 298.2K，转换后约为 25°C。

表 1-3. 内部温度命令说明

命令	名称	单位	类型	说明
0x68	内部温度	0.1K	I2	这是最近测量的内部芯片温度。

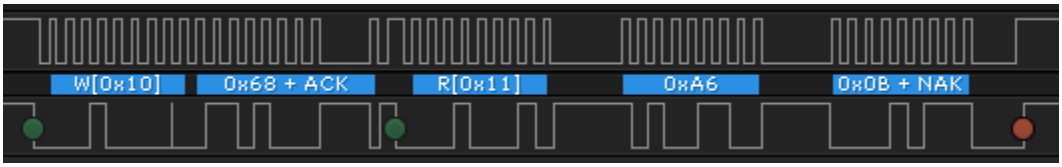

图 1-3. 为读取内部温度捕捉的 I2C 波形

1.4 CC2 电流- 0x3A

表 1-4 示出了如何从 CC2 读取 16 位电流测量值。以下示例中的电流读数为 7mA。

表 1-4. CC2 电流命令说明

命令	名称	单位	类型	说明
0x3A	CC2 电流	userA	I2	16 位 CC2 电流

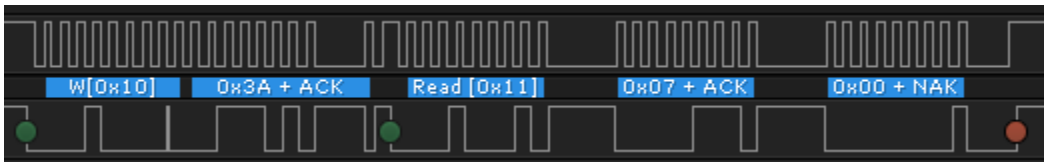

图 1-4. 为读取 CC2 电流捕捉的 I2C 波形

BQStudio 软件中的“命令序列”模块允许您试用命令，该工具还可用于创建和保存命令序列。此示例中的 事务日志示出了目前为止所涵盖的所有命令。

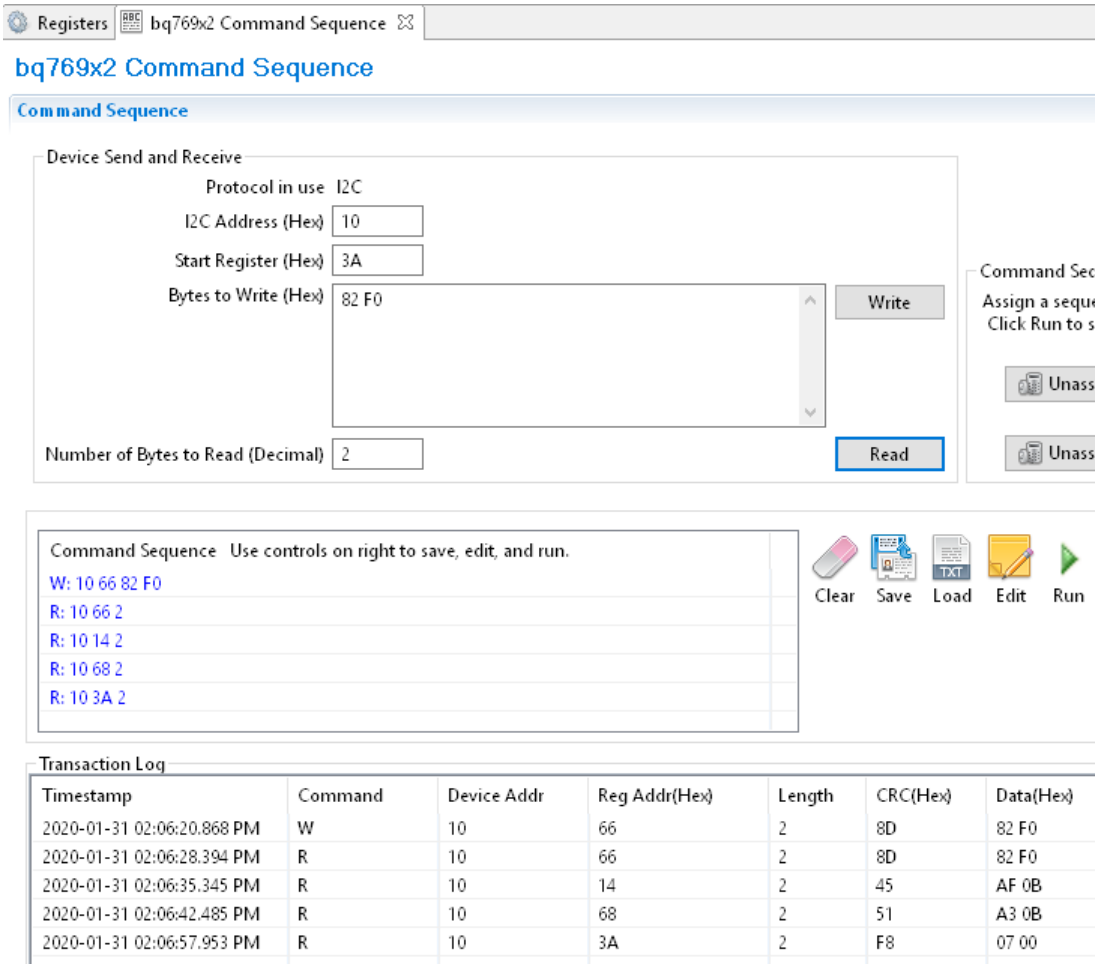

图 1-5. 显示执行多个直接命令的 BQStudio 示例

BQStudio 在控制面板有一个 自动刷新选项，该选项定期读取器件的寄存器来刷新显示的测量值。当使用 命令序列模块时，建议单击绿色横条来禁用 自动刷新功能。该横条将变为红色，表示已禁用 自动刷新功能（参见图 1-6）。

图 1-6. 自动刷新已禁用

2 子命令

子命令使用与直接命令不同的格式，并且使用 7 位命令地址空间进行间接访问。子命令还支持块传输。要发出子命令，将命令地址写入 0x3E/0x3F。如果要读回数据，数据将被填充到 32 字节传输缓冲区中，该缓冲区使用地址 0x40 - 0x5F 。下文列举了多个示例。

器件获取数据所需的时间取决于特定的子命令和器件内正在进行的任何其他处理，因此在运行期间会有所不同。技术参考手册中介绍了每个子命令的大致时间。从子命令读取数据时，有两种方法可以解决此计时问题：

最简单的方法是在写入 0x3E/0x3F 之后，等待 2ms，再从传输缓冲区读取结果。

第二种方法在技术参考手册的第 3 章中进行了描述。这种方法是从 0x3E/0x3F 读取，直到子命令完成操作。如果返回的值为 0xFF，则表示子命令尚未完成操作。子命令完成后，返回的值将与写入的命令匹配。此响应仅适用于返回要读回的数据的子命令。

某些子命令将数据写入寄存器，然后必须写入带有校验和及长度的 0x60/0x61。这仅适用于 *FET_Control()*、*REG12_Control()*、*CB_Active_Cells()* 和 *CB_SET_LVL()* 子命令。下一节将提供计算校验和及长度的示例，因为这在写入 RAM 寄存器时也是必需的。

2.1 DEVICE_NUMBER - 0x0001

可以通过将子命令编号 0x0001（小端格式）写入命令地址 0x3E 中来读取器件型号。然后，从地址 0x40 的数据缓冲区中读取数据。在该示例中，返回的器件型号为 0x7694（代表 BQ76942）。

表 2-1. DEVICE_NUMBER 子命令说明

命令	名称	数据	单位	类型	说明
0x0001	DEVICE_NUMBER	器件型号	十六进制	U2	报告用于识别产品的器件型号。数据以小端格式返回

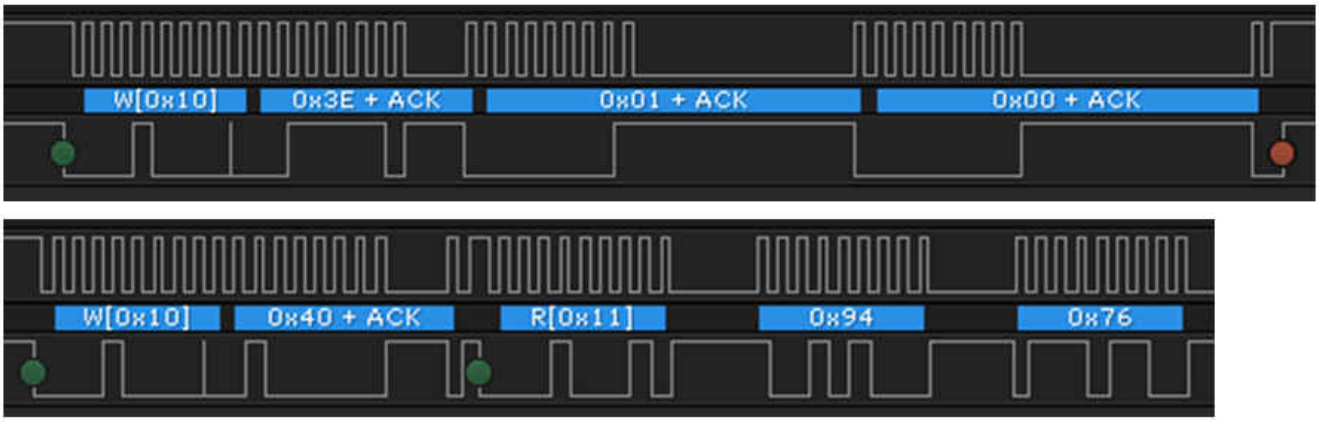

图 2-1. 为 DEVICE_NUMBER 子命令捕捉的 I2C 波形

2.2 生产状态 - 0x0057

生产状态 子命令从 生产状态寄存器中读取两个字节。首先，将 0x0057 命令写入 0x3E，然后从 0x40 读取两个字节。

表 2-2. 生产状态子命令说明

命令	名称	数据	单位	类型	说明
0x0057	生产状态	生产状态	十六进制	H2	提供在生产期间使用的标志。

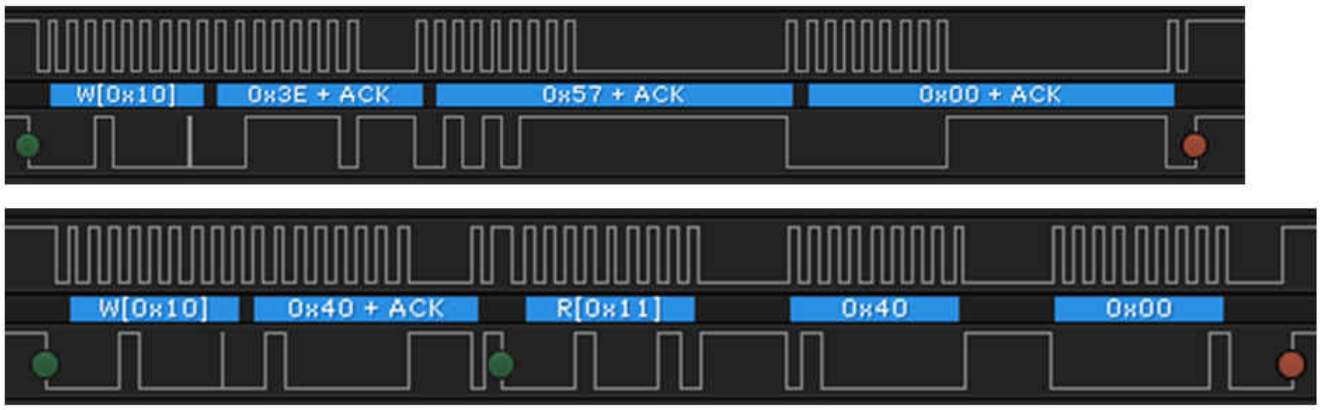

图 2-2. 为 MANUFACTURING_STATUS 子命令捕捉的 I2C 波形

2.3 FET_ENABLE - 0x0022

有些子命令不需要从数据缓冲区读取数据，因为它们只提供指令。FET_ENABLE 子命令就是一个示例。该命令是通过将 0x0022 写入 0x3E 来发出的。

表 2-3. FET_ENABLE 子命令说明

命令	名称	说明
0x0022	FET_ENABLE	在 生产状态中切换 FET_EN。FET_EN = 0 表示 FET 测试模式。FET_EN = 1 表示固件 FET 控制。

图 2-3. 为 FET_ENABLE 子命令捕捉的 I2C 波形

2.4 重置 - 0x0012

重置子命令对器件进行重置，并将 RAM 寄存器恢复为默认（或 OTP 编程）值。该命令是通过将 0x0012 写入 0x3E 来发出的。

表 2-4. 重置子命令说明

命令	名称	说明
0x0012	重置	重置器件。

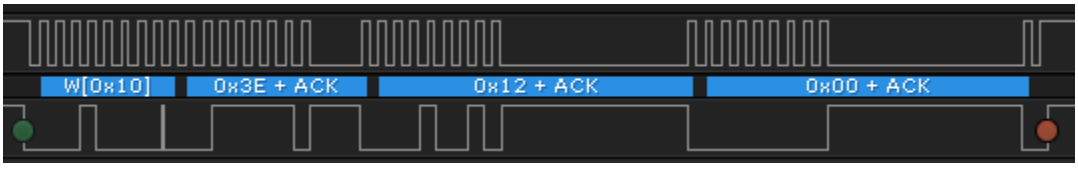

图 2-4. 重置子命令捕捉的 I2C 波形

该示例中的*事务日志*显示了用于执行子命令所涵盖的所有命令。

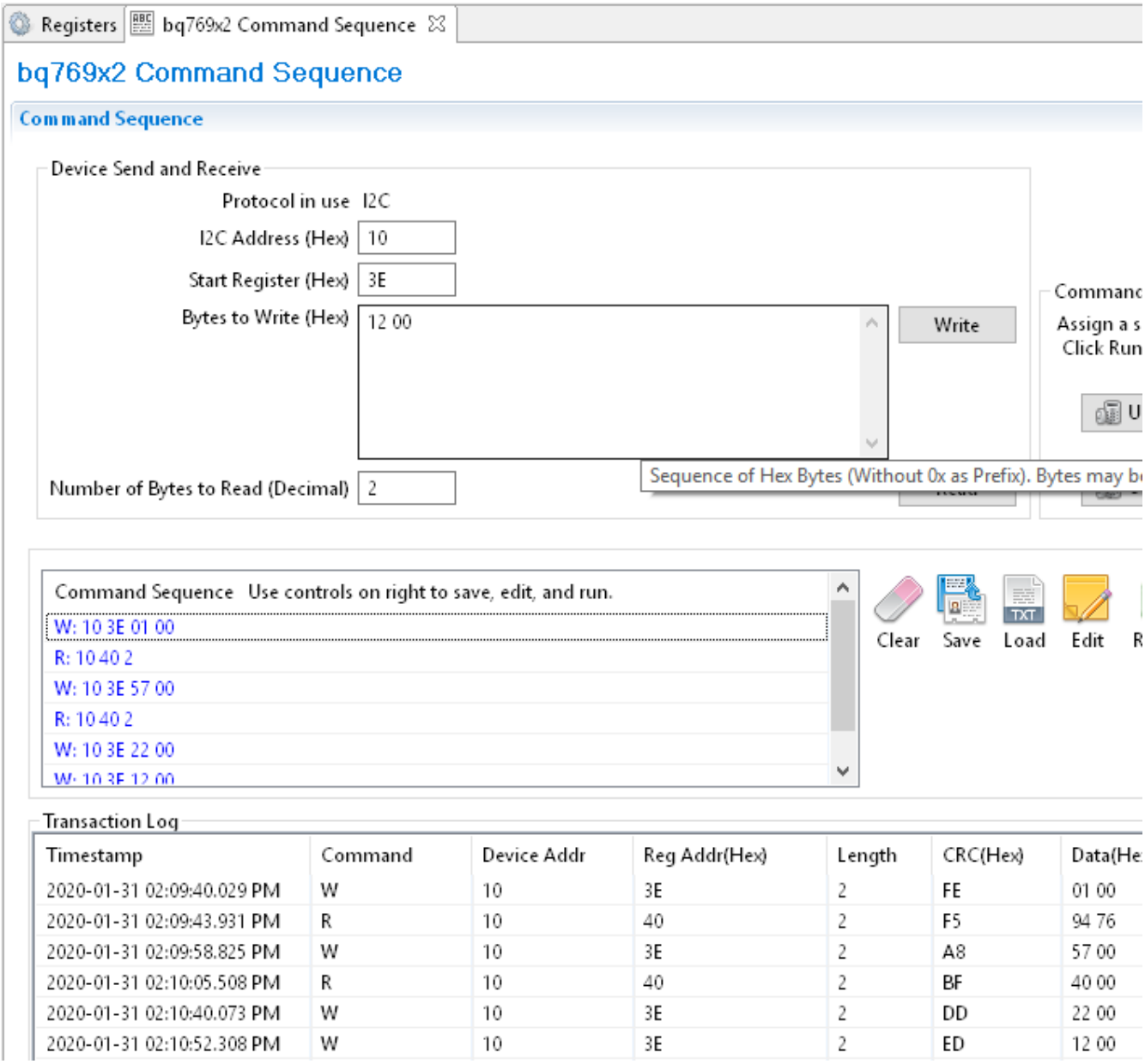

图 2-5. 显示执行多个子命令的 BQStudio 示例

3 读取和写入 RAM 寄存器

RAM 中寄存器的完整视图可以在器件专用技术参考手册中找到，也可以在 BQStudio 的数据内存屏幕中找到。要在 BQStudio 中查看 RAM 寄存器地址，进入“Window”->“Preferences”菜单，然后选择“Show Advanced Views”。将寄存器地址写入 0x3E，然后从 0x40 的数据缓冲区开始读取，以完成 RAM 寄存器的读取向 RAM 寄存器写入时，首先将寄存器地址写入 0x3E，然后写入数据，再将校验和长度写入 0x60/0x61 中。校验和及长度计算在器件专用数据表中有更多描述，但在以下示例中略做说明。

备注

写入 RAM 寄存器时，强烈建议先进入 CONFIG_UPDATE 模式，完成后执行退出 CONFIG_UPDATE 模式的命令。如此，可以确保在修改设置时仍然稳定运行。

3.1 读取启用保护功能 A

BQ769x2 器件的默认设置启用了 COV（过压）和 SCD（短路）保护功能。下面通过读取***启用保护功能***寄存器来验证这一点，其中， RAM 地址 0x9261 返回的值为 0x88。

表 3-1. 启用保护功能 A 说明

类别	子类别	名称	类型	最小值	最大值	默认值	单位
设置	保护	启用保护功能 A	U1	0x00	0xFF	0x88	十六进制

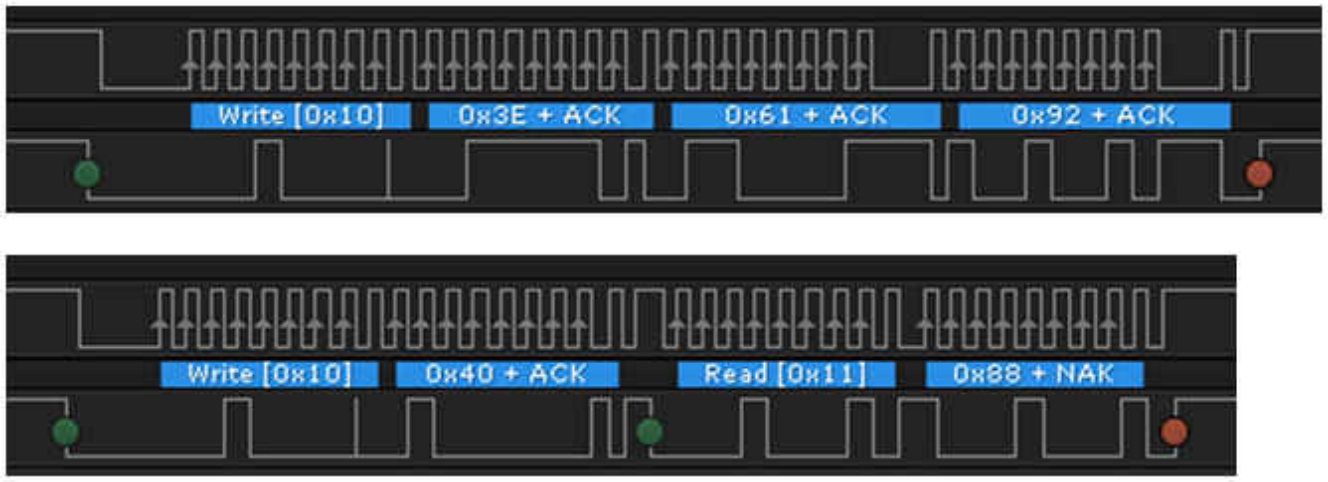

图 3-1. 为读取启用保护功能 A 寄存器捕捉的 I2C 波形

3.2 进入 CONFIG_UPDATE 模式

在写入 RAM 寄存器之前，建议进入 CONFIG_UPDATE 模式，以防止所有更改完成之前设置生效。SET_CFGUPDATE 遵循子命令格式。

表 3-2. SET_CFGUPDATE 和 EXIT_CFGUPDATE 说明

命令	名称	说明
0x0090	SET_CFGUPDATE	进入 CONFIG_UPDATE 模式。
0x0092	EXIT_CFGUPDATE	退出 CONFIG_UPDATE 模式。这也会清除 电池状态()[POR] 和 电池状态()[WD] 位。

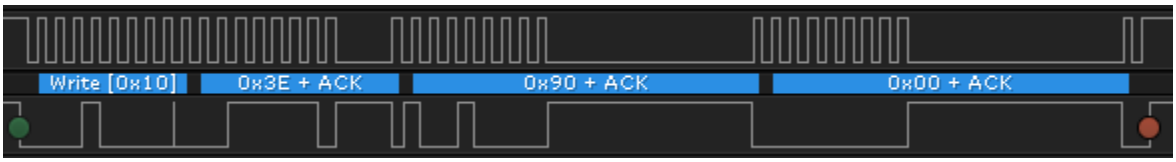

图 3-2. 为 SET_CFGUPATE 捕捉的 I2C 波形

3.3 写入启用保护功能 A

在本例中，启用默认保护功能时，一并启用 CUV（欠压）保护特性。这需要将 0x8C 写入 RAM 地址 0x9261。校验和是根据地址和数据（0x61、0x92、0x8C）计算的，并且是这些字节总和的补充。长度还包括器件地址和命令地址这两个字节，总长度为 5。

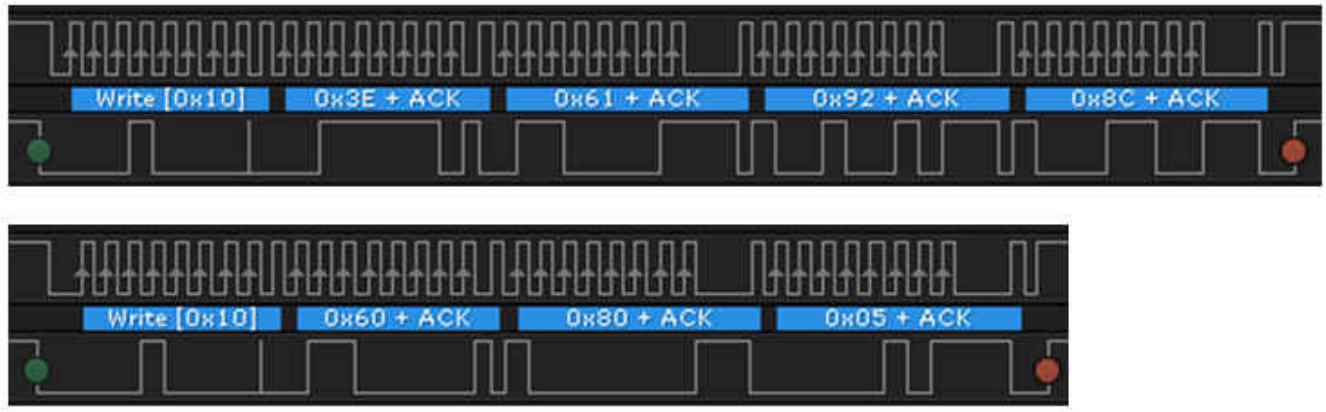

图 3-3. 为写入启用保护功能 A 捕捉的波形

3.4 写入“VCell 模式”

接下来，向 ***VCell 模式***寄存器写入，以便为器件 BQ76942 配置 9 节电池。下述示例将 0x037F 写入 0x9304，然后将新的校验和及长度写入 0x60/0x61。

表 3-3. VCell 模式说明

类别	子类别	名称	类型	最小值	最大值	默认值	单位
设置	配置	VCell 模式	H2	0x0000	0xFFFF	0x0000	十六进制

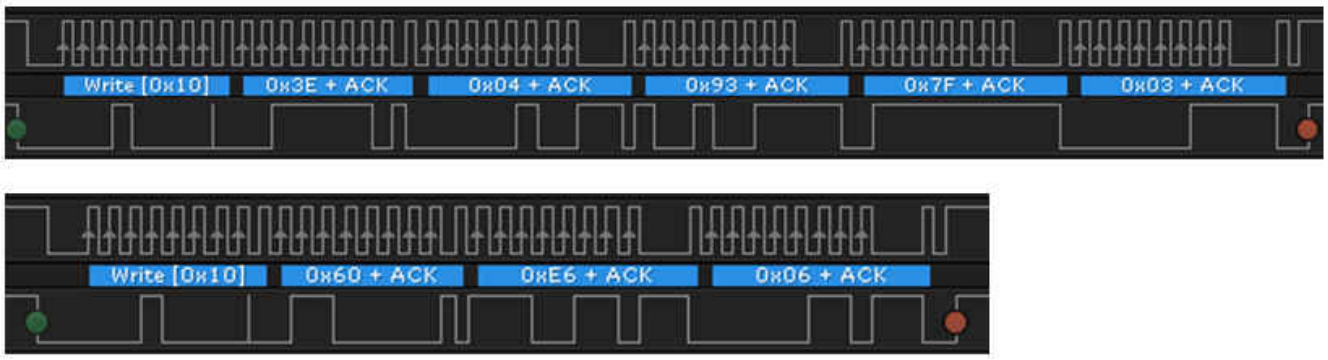

图 3-4. 为写入“VCell 模式”捕捉的 I2C 波形

3.5 退出 CONFIG_UPDATE 模式

写入 RAM 寄存器后，退出 CONFIG_UPDATE 模式，此时新设置将生效。

图 3-5. 为 EXIT_CFGUPDATE 捕捉的 I2C 波形

本示例中的 事务日志示出了所有已涵盖的用于读取和写入 RAM 寄存器的命令。

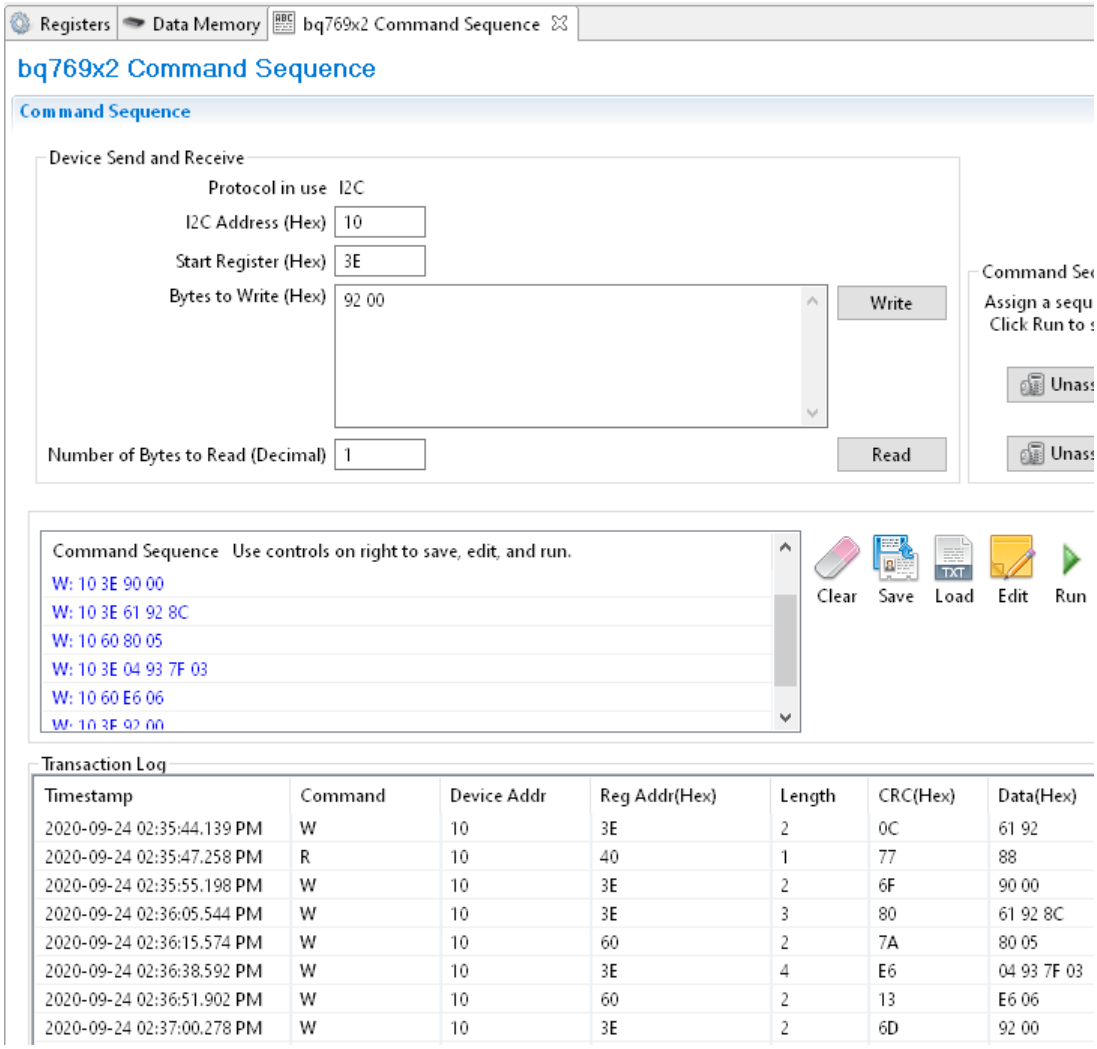

图 3-6. 显示执行 RAM 寄存器的读取和写入的 BQStudio 示例

4 具有 CRC 的 I2C

BQ769x2 系列上的 I2C 接口包含一个可选的 CRC 校验。可以在 ***Settings:Configuration:Comm Type*** 寄存器中启用 CRC 特性。如果在使用 BQStudio 时更改了该寄存器，则应执行 *SWAP_COMM_MODE()* 子命令，然后重新启动 BQStudio，以便其能够检测新的通信模式。下述为 CRC 校验启用的 I2C 波形捕获图的两个示例。

I2C 启用后，根据所有的字节（包括第一数据字节）来计算第一个数据字节的 CRC。对于第一字节之后的每个数据字节，仅计算该字节的 CRC 字节。在图 4-1，使用 *FET_ENABLE* 子命令为 [0x10 0x3E 0x22] 计算第一个字节的 CRC - CRC 计算结果为 0x63。第二字节 [0x00] 的 CRC 为 0x00。

图 4-1. 使用 CRC 为 FET_ENABLE 子命令捕捉的 I2C 波形

在图 4-2，使用 *VCell 1* 命令为 [0x10 0x14 0x11 0x68] 计算第一字节的 CRC - CRC 计算结果为 0x33。第二字节 [0x0B] 的 CRC 为 0x31。

图 4-2. 使用 CRC 为 VCell 1 命令捕捉的 I2C 波形

5 具有 CRC 的 SPI 示例

可以在 ***Settings:Configuration:Comm Type*** 寄存器中启用 BQ769x2 系列上的 SPI 接口。更改为 SPI 模式时，默认的 SPI 输出逻辑电压电平为 1.8V，这是因为它引用了器件的内部稳压器。若要更改逻辑电平，应启用 REG1 LDO 并将其编程到所需的电压电平，再将 ***SPI 配置***寄存器编程到 0x60 以启用 MISO_REG1 位。接着，应执行 *SWAP_COMM_MODE()* 子命令。如果使用 BQStudio，则应重新启动 BQStudio，以便其能够检测新的通信模式。

一些器件版本可先预配置为 SPI 模式。有关可用的不同器件型号的信息，请参阅器件专用数据表。

以下示例涵盖了 I2C 示例中包含的一些相同命令。关于 SPI 接口协议（具有 CRC 功能）的一些重要事项：

- SPI_CS 为低电平有效。
- 第一个 SPI 数据包为 8 位。第一个位是 R/W 位，后跟一个 7 位地址。
- 第二个数据包是 8 位数据。
- 第三个数据包是根据第一个和第二个字节计算的 8 位 CRC。

所有示例都包含每个事务的多次写入。EV2400 和 BQStudio 使用此方法来验证命令已成功写入。这是因为，如果内部振荡器未运行（如果器件处于睡眠模式）或处理器繁忙，则器件会忽略某些事务。一旦 MISO 引脚上的数据（应反映先前写在 MOSI 上 的数据）显示为正确数据，则确认数据包已成功写入。有关 SPI 接口的更多详细说明，请参阅器件专用技术参考手册。

在 SPI 模式和 I2C 模式下使用 BQ769x2 系列时，需要注意一些差异。I2C 模式支持块写入和读取，而 SPI 模式仅支持单字节事务。 I2C 模式支持直接命令时钟延展，然而， SPI 模式没有此特性，因此除了子命令时序之外，还需了解直接命令时序。

5.1 直接命令示例：警报启用 - 0x66

图 5-1 将 警报启用寄存器设为值 0xF082 。

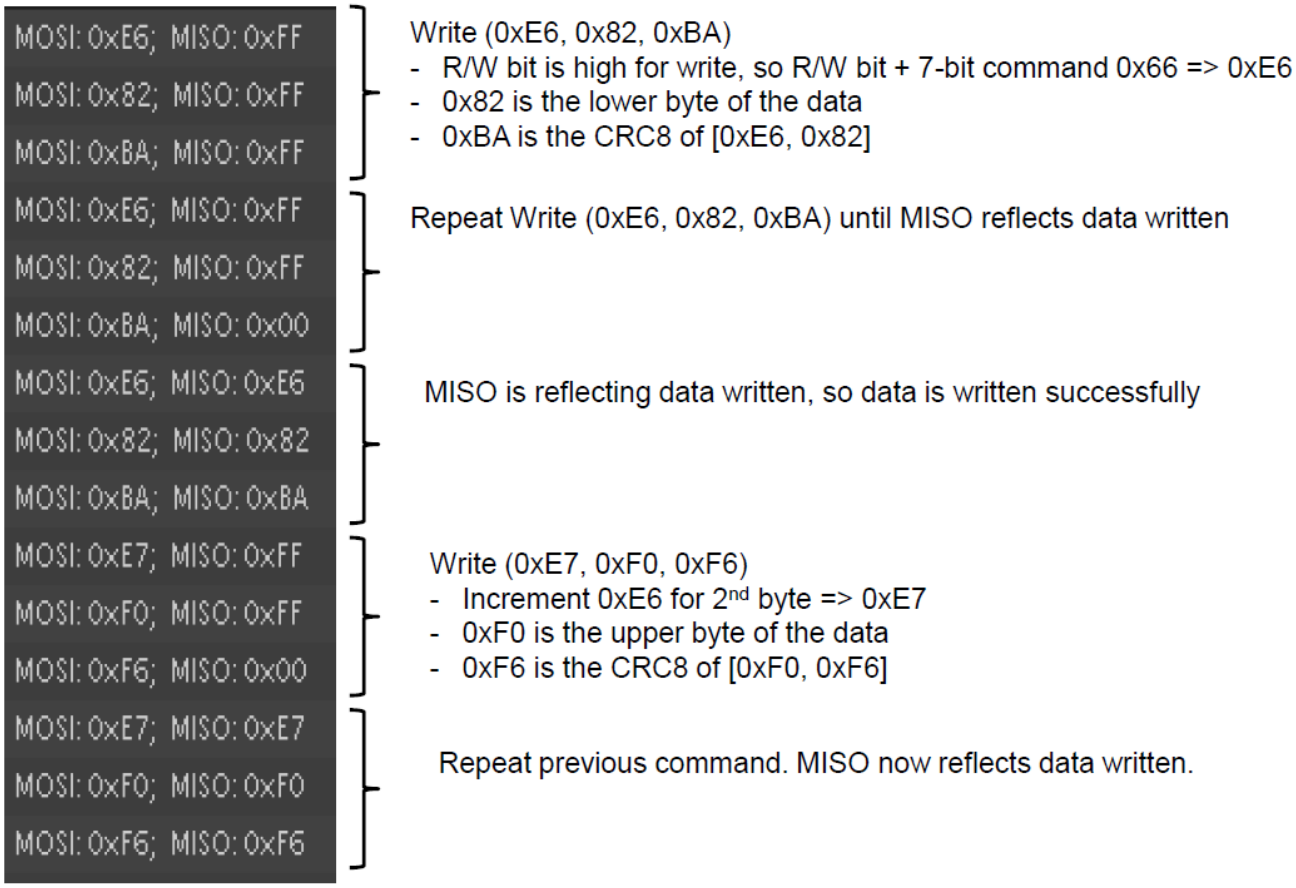

图 5-1. 警报启用直接命令

5.2 直接命令示例：Cell 1 电压- 0x14

图 5-2 示出了如何读取 Cell 1 的电压。

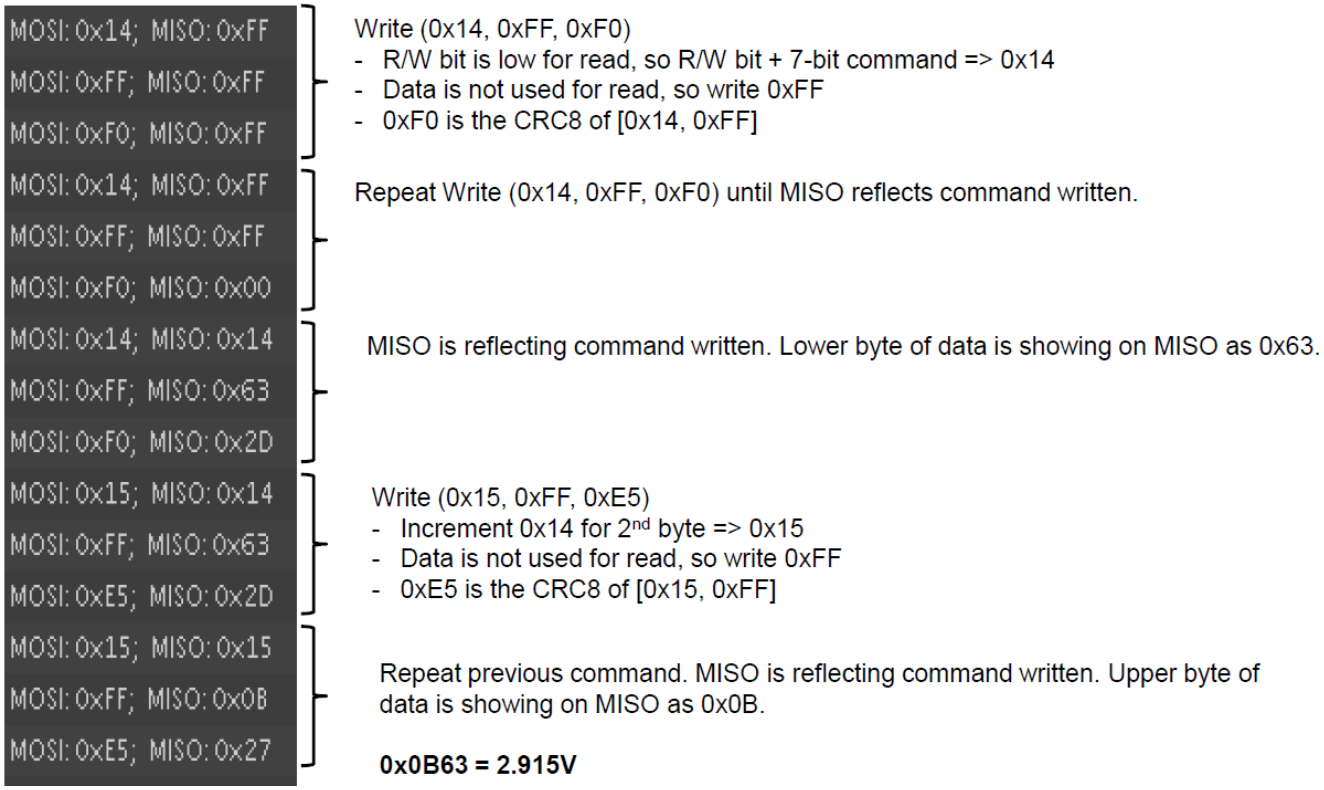

图 5-2. Cell 1 电压读取直接命令

5.3 子命令示例：器件型号 - 0x0001

图 5-3 示出了如何从子命令 0x0001 中读取器件型号。

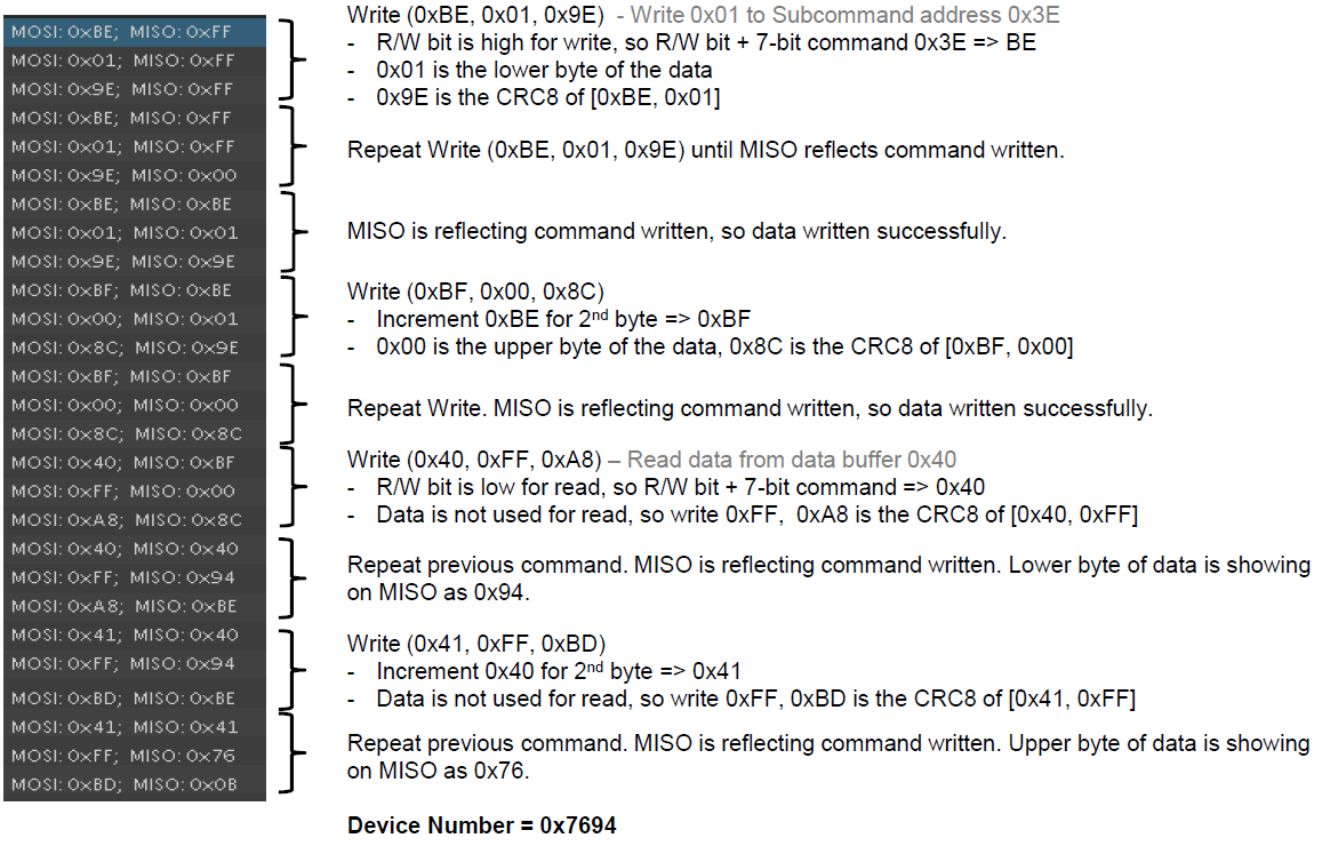

图 5-3. 器件型号子命令

5.4 子命令示例：FET_ENABLE - 0x0022

图 5-4 示出了如何写入 FET_ENABLE 子命令 0x0022。

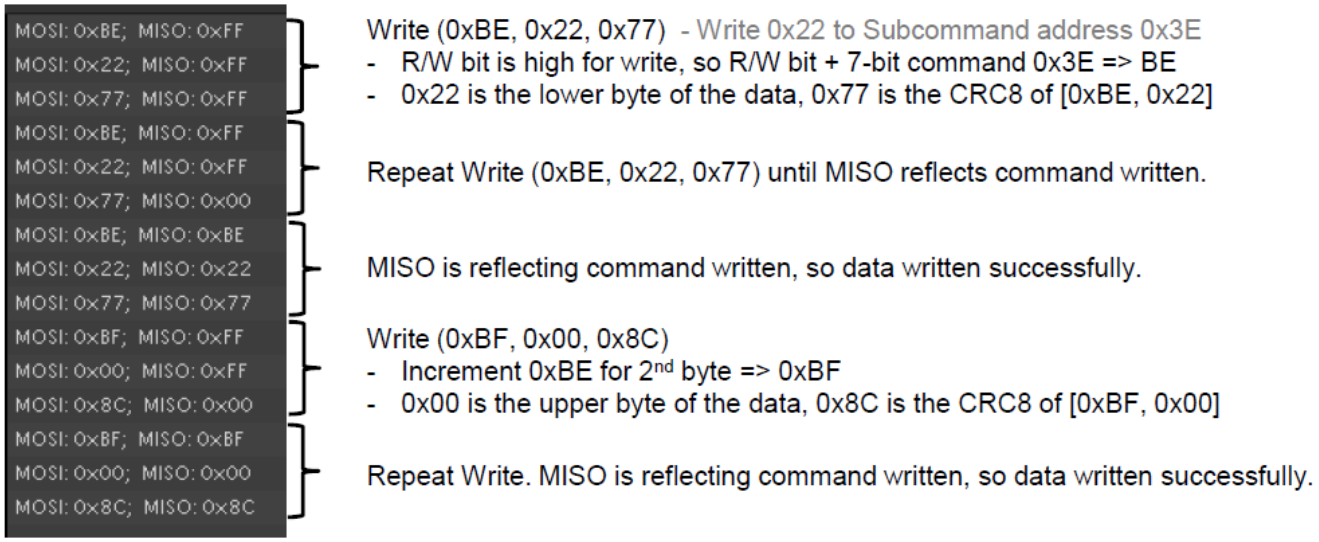

图 5-4. FET_ENABLE 子命令

5.5 子命令示例：重置 - 0x0012

图 5-5 示出了如何写入 重置子命令 0x0012。

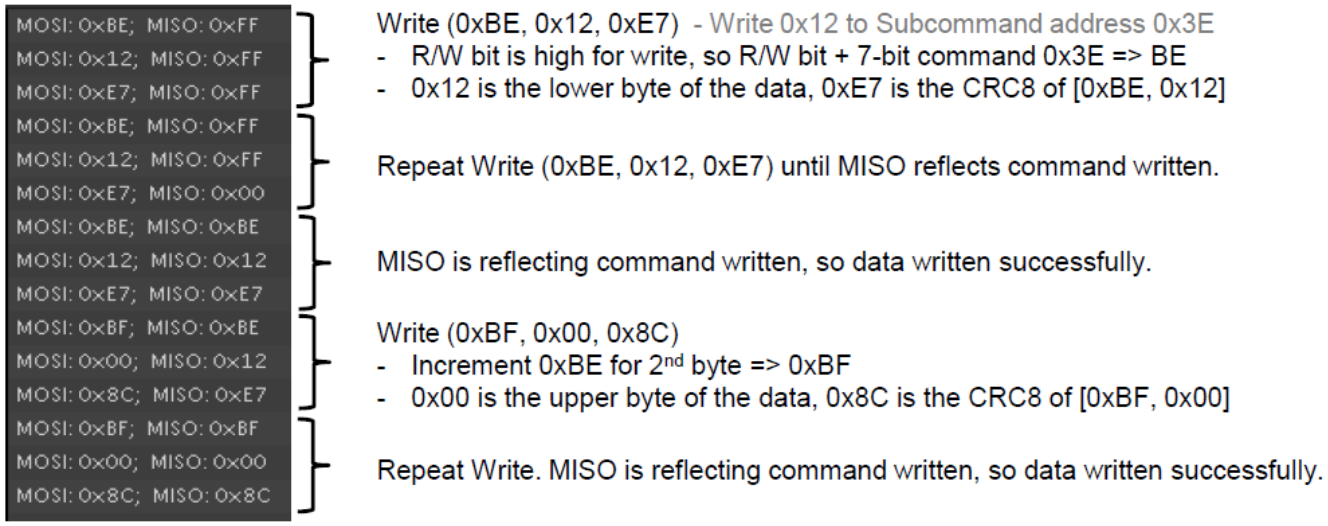

图 5-5. 重置子命令

5.6 RAM 寄存器读取示例：启用保护功能 A

图 5-6 示出了如何读取 RAM 寄存器 **启用保护功能 *A***。地址为 0x9261。

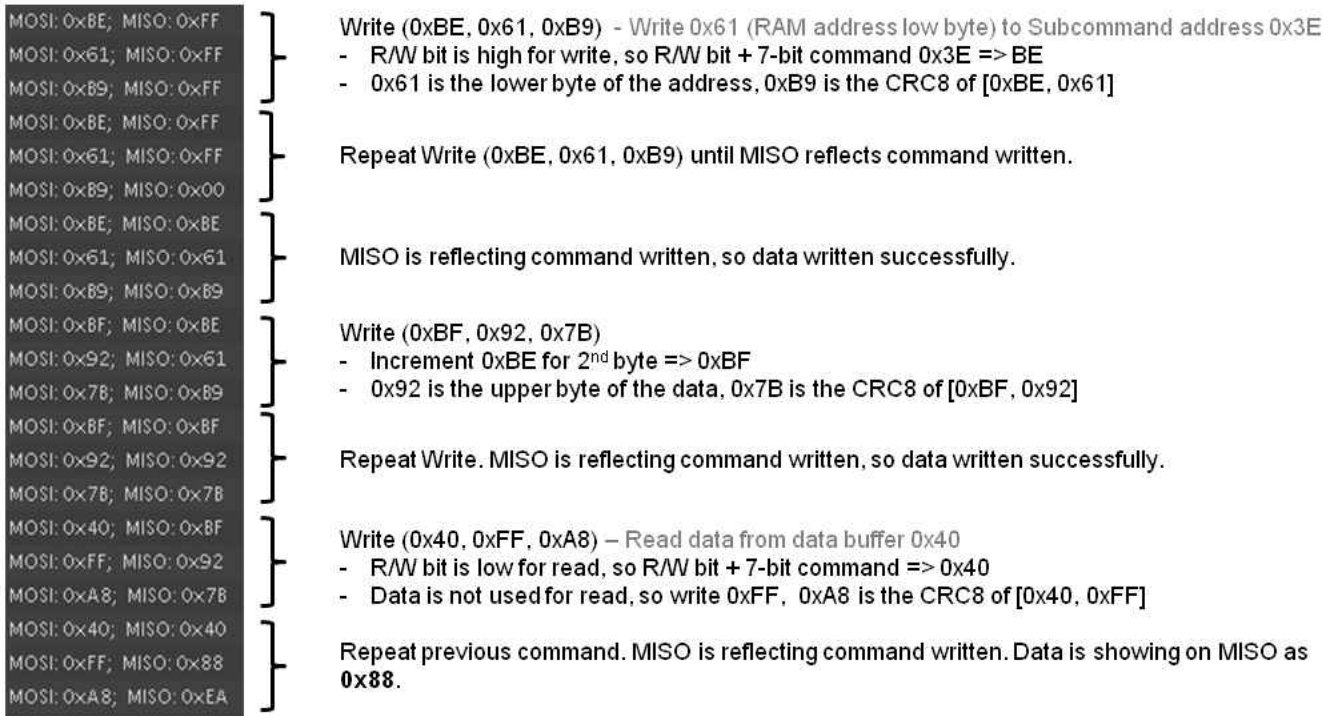

图 5-6. 读取启用保护功能 A

5.7 RAM 寄存器写入示例：启用保护功能 A

图 5-7 示出了如何将值 0x8C 写入 RAM 寄存器**启用保护功能 *A***。

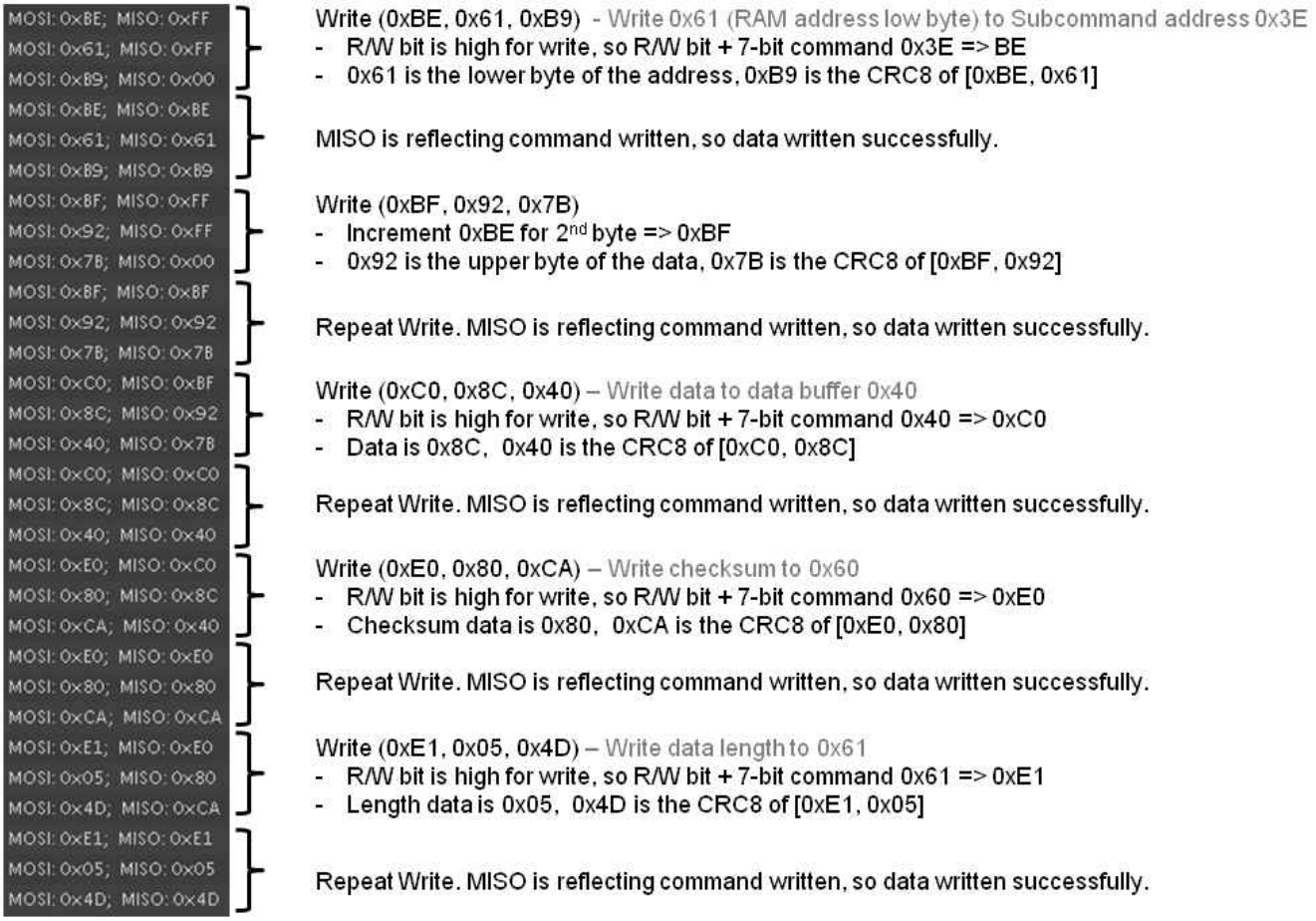

图 5-7. 写入启用保护功能 A

6 简单代码示例

以下示例代码是用 Python 编写的，旨在通过 EV2400 模块或通过 BQ76942 或 BQ76952 评估模块上的 USB 连接器从 PC 与 BQ769x2 器件进行通信。该代码显示了简单 的 I2C 读取和写入函数的创建、 DataRAM_Read 函数（也可用于执行子命令，因其遵循相同的格式）的创建，以及显示校验和及长度计算的 DataRAM_Write 函数的创建。代码的主要部分介绍了本文档前面三个章节中涉及的所有示例。

这个简单的代码示例旨在说明 I2C 命令的基本命令结构。微控制器代码示例也可用于 I2C 和 SPI。节 7 中提供了微控制器代码的链接。

```
'''
/* BQ769x2 示例程序演示了直接命令、子命令以及从器件 RAM 写入/读取 的示例。
'''
import pywinusb
import bqcomm
import sys
import time
from time import sleep
import sets
I2C_ADDR = 0x10  # BQ769x2 slave address
numCells = 10    # Set to 10 for BQ76942
#######################################################
## 检查 EV2400 是否已连接
#######################################################
try:
    a = bqcomm.Adapter()  # 这将使用第一个找到的 Aardvark 或 EV2400
except:
    print "No EV2400 Available"
    sys.exit(1)
#######################################################
## 定义一些命令函数
#######################################################
def I2C_Read(device_addr, reg_addr, length):
    '''
    使用 全局 I2C 地址并返回读取值
    '''
    try:
        value = a.i2c_read_block(device_addr, reg_addr, length)
    except:
        print "Nack received"
        return
    return value
def I2C_Write(device_addr, reg_addr, block):
    '''
    使用全局 I2C 地址
    '''
    try:
        a.i2c_write_block(device_addr, reg_addr, block)
    except:
        print "Nack received"
    return
def DataRAM_Read(addr, length):
    '''
    Write address location to 0x3E and read back from 0x40
    Used to read configuration registers and for subcommands
    '''
    addressBlock = [(addr%256), (addr/256)]
    I2C_Write(I2C_ADDR, 0x3E, addressBlock)
    value = I2C_Read(I2C_ADDR, 0x40,length)
    return value
def DataRAM_Write(addr, block):
    '''
    Write address location to 0x3E and Checksum,length to 0x60
    Used to write configuration registers
    '''
    addressBlock = [(addr%256), (addr/256)]
    wholeBlock = addressBlock + block
    I2C_Write(I2C_ADDR, 0x3E, wholeBlock)             # Write Data Block
    # Write Data Checksum and length to 0x60, required for RAM writes
    I2C_Write(I2C_ADDR, 0x60, [~sum(wholeBlock) & 0xff, len(wholeBlock)+2])
    return
def crc8(b,key):
    crc = 0
    ptr = 0
    for j in range(len(b),0,-1):
```

```
        for k in range(8):
            i = 128 / (2**k)
            if ((crc & 0x80) != 0):
                crc = crc * 2
                crc = crc ^ key
            else:
                crc = crc * 2
            if ((b[ptr] & i) != 0):
                crc = crc ^ key
        ptr = ptr + 1
    return crc
########################################
#  主脚本开始
########################################
################ 直接命令示例 ################
#将警报启用写入 0xF082
I2C_Write(I2C_ADDR, 0x66, [0x82, 0xF0])
#读取电池 #1 上的电压
result = I2C_Read(I2C_ADDR, 0x14, 2)
print "Cell 1 = ", (result[1]*256 + result[0]), " mV"
#读取内部温度
result = I2C_Read(I2C_ADDR, 0x68, 2)
print "Internal Temp = ", ((result[1]*256 + result[0])/10 - 273.15), "degrees C"
#读取 CC2 电池测量值
result = I2C_Read(I2C_ADDR, 0x3A, 2)
print "CC2 = ", (result[1]*256 + result[0]), " mA"
################ 子命令示例 ################
#读取器件型号
b = DataRAM_Read(0x0001,6)
print "Device_Number = " '{0:04X}'.format(b[1]*256+b[0])
#读取生产状态
b = DataRAM_Read(0x0057,2)
print "Manufacturing Status = " '{0:04X}'.format(b[0]+256*b[1])
## 只包含命令的子命令 ##
#FET_ENABLE
I2C_Write(I2C_ADDR, 0x3E, [0x22, 0x00])
#RESET - 器件恢复为默认设置
I2C_Write(I2C_ADDR, 0x3E, [0x12, 0x00])
sleep(1)
############ 读取和写入 RAM 寄存器 ##########
# 读取"启用保护功能 A RAM 寄存器 0x9261
b = DataRAM_Read(0x9261,1)
print "Enabled Protections A = 0x" '{0:02X}'.format(b[0])
#设置 CONFIG_UPDATE 模式（RAM 寄存器的写入应在 CONFIG_UPDATE 模式时进行，并退出 CONFIG_UPDATE 模式时生效）
I2C_Write(I2C_ADDR, 0x3E, [0x90, 0x00])
#写入启用保护功能 A RAM 寄存器以启用 CUV 保护
DataRAM_Write(0x9261, [0x8C])
#写入"VCell Mode"RAM 寄存器，为 9 节电池进行配置
DataRAM_Write(0x9304, [0x03, 0x7f])
#退出 CONFIG_UPDATE 模式
I2C_Write(I2C_ADDR, 0x3E, [0x92, 0x00])
# CRC8 示例计算
TestValue = [0x10, 0x14, 0x11, 0x68]
crcKey = 0x107
check = 0xff & crc8(TestValue,crcKey)
print "crc8 check = 0x" '{0:02X}'.format(check)
```

下述是 BQ76942 评估模块上运行的示例 Python 脚本的输出。

```
Cell 1 =  3700  mV
Internal Temp =  25.05 degrees C
CC2 = 7  mA
Device_Number = 7694
Manufacturing Status = 0040
Enabled Protections A = 0x88
crc8 check = 0x33
```

7 参考文献

- 德州仪器 (TI)：*BQ76952 3S-16S 电池监控器和保护器数据表*
- 德州仪器 (TI)：*BQ76942 3S-160S 电池监控器和保护器数据表*
- 德州仪器 (TI)：*BQ76952 技术参考手册*
- 德州仪器 (TI)：*BQ76942 技术参考手册*

8 修订历史记录

注：以前版本的页码可能与当前版本的页码不同

Changes from Revision A (October 2020) to Revision B (August 2021) **Page**

重要声明和免责声明

TI 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、某特定用途方面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 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款 (https:www.ti.com/legal/termsofsale.html) 或 ti.com 上其他适用条款/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。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

重要声明和免责声明

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、某特定用途方面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功能安全、信息安全、监管或其他要求。

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 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/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。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265