User's Guide

BQ7690x 软件开发指南

Andria McIntyre

摘要

本应用手册提供了 BQ7690x 器件系列电池监测器（包括 BQ76905 和 BQ76907）的通信数据包和序列示例，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。本文档可与器件特定技术参考手册和数据表一起使用。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 BQ7690x 器件系列，需要 BQStudio 1.3.115 或更高版本。

BQ7690x 器件系列集成了 I^2C 通信接口。I^2C 接口包含一个可选的 CRC 校验。有关完整选项列表，请参阅器件特定数据表。本文档涵盖了许多使用 I^2C 接口的示例。

内容

商标

1 直接命令

直接命令的完整列表可在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，Alarm Enable 的寄存器设置设为 0xC200。在图 1-1 中，设置更改为 0x0060。数据采用小端格式。BQ7690x 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，该命令可以为 *I2C_Write(0x10, 0x66, [0x60, 0x00])*。

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。

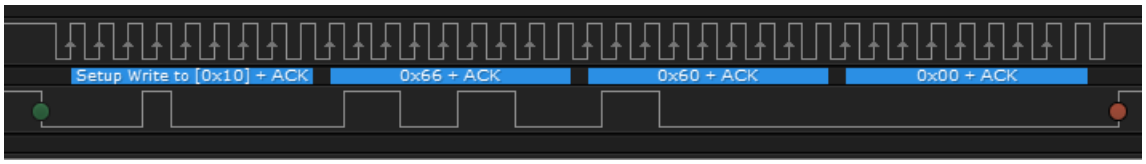


图 1-1. 将 Alarm Enable 设为 0x0060 后捕获到的 I2C 波形

1.2 电芯 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。在图 1-2 中，16 位电芯 1 电压读取 0x0C0C，对应于 3,084mV。

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	电芯 1 上的 16 位电压

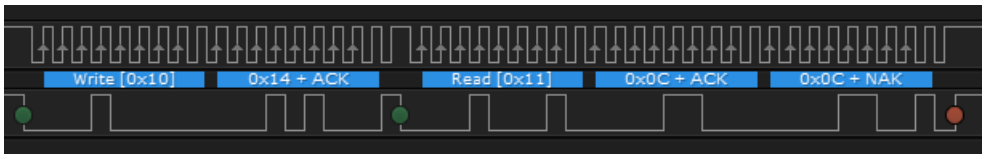


图 1-2. 为读取 Cell 1 电压捕捉的 I2C 波形

1.3 内部温度 - 0x28

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 0.1°C。在图 1-3 中，0x001D 的读数表示十进制值 29，即 29°C。

表 1-3. 内部温度命令说明

命令	名称	单位	类型	说明
0x28	Int Temperature	0.1 °C	I2	这是最近测量的内部芯片温度。

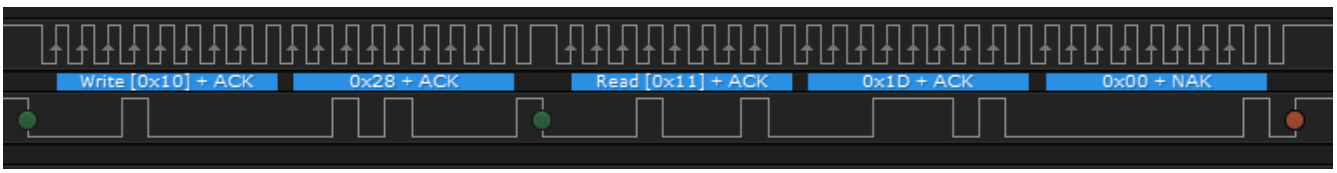


图 1-3. 为读取内部温度捕捉的 I2C 波形

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。在图 1-4 中，当前读数 0x022D 表示十进制值 557，即 557mA。

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

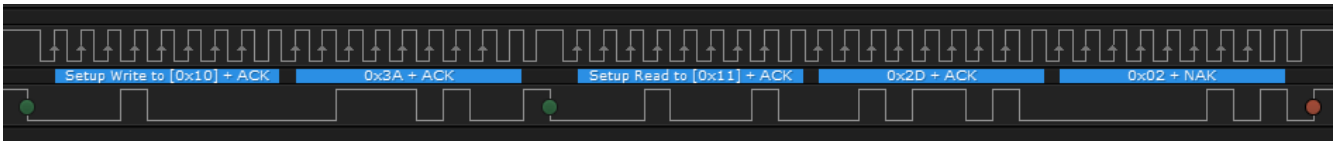


图 1-4. 为读取 CC2 电流捕捉的 I2C 波形

1.5 直接命令总结

BQStudio 软件中的“Command Sequence”模块允许您写入自定义命令帧。该工具还可用于创建和保存命令序列。图 1-5 中的 *Transaction Log* 展示了目前为止已阐述过的所有命令。

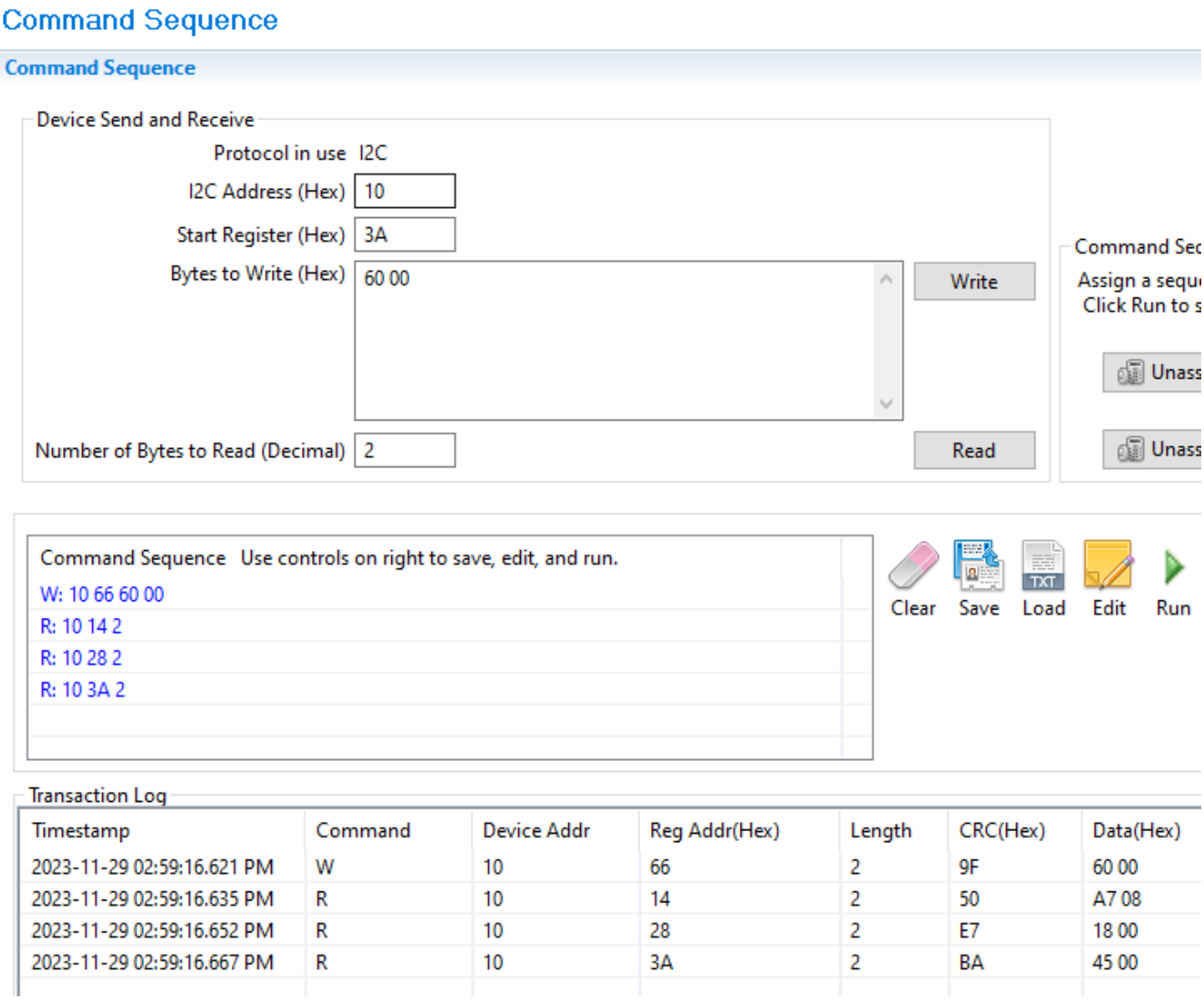


图 1-5. 显示执行多个直接命令的 BQStudio

1.5.1 禁用自动刷新

BQStudio 的 *Dashboard* 上有一个 *Auto Refresh* 选项，它会定期读取器件的寄存器以刷新显示的测量值。使用 *Command Sequence* 模块时，建议点击绿条来禁用 *Auto Refresh*。该条变为红色时，指示已禁用 *Auto Refresh*（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 。

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 CB_ACTIVE_CELLS、PROG_TIMER、PROT_RECOVERY 和 SECURITY_KEYS 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

表 2-1 显示了如何读取 BQ7690x 器件型号。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。在图 2-1 中，返回的器件型号为 0x7605（代表 BQ76905）。

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

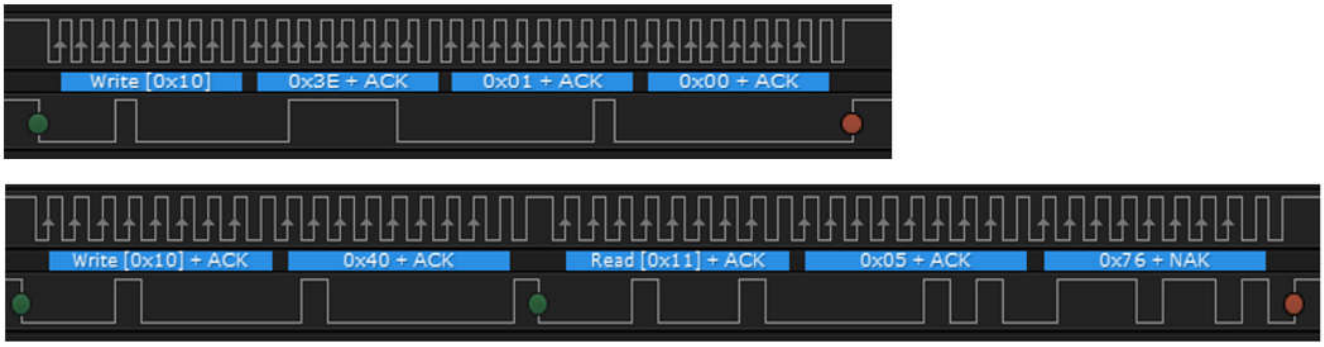


图 2-1. 为 DEVICE_NUMBER 子命令捕捉的 I2C 波形

2.2 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。表 2-2 中显示的 FET_ENABLE 子命令就是一个示例。在图 2-2 中，该命令是通过将 0x0022 写入 0x3E 来发出的。

表 2-2. 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 2-2. 为 FET_ENABLE 子命令捕捉的 I2C 波形

2.3 重置 - 0x0012

表 2-3 中显示的 RESET 子命令在器件上执行复位并将 RAM 寄存器设置恢复为默认值。在图 2-3 中，该命令是通过将 0x0012 写入 0x3E 来发出的。

表 2-3. 重置子命令说明

命令	名称	说明
0x0012	重置	重置器件。

图 2-3. 重置子命令捕捉的 I2C 波形

2.4 CB_ACTIVE_CELLS - 0x0083

表 2-4 中显示的 CB_ACTIVE_CELLS 子命令是一个可以将电芯均衡数据读取或写入寄存器的子命令示例。执行写入操作后，开始对指定的电芯进行均衡。在图 2-4 中，通过将 0x0083 命令和 0x02 数据写入 0x3E，然后向 0x60/0x61 写入校验和及长度，对电芯 1 执行电芯均衡。当使用子命令写入数据时，校验和和长度是接受数据所必需的。校验和是根据地址和数据（0x83、0x00、0x02）计算的，并且是这些字节总和的补码。在本例中，校验和为 0x7A。长度包括器件地址和命令地址占用的两个字节，总长度为 0x05。

备注

总长度必须为十六进制格式。例如，一个包含 10 个寄存器的块的长度可以是 0x0A，而不是 0x10。

表 2-4. CB_ACTIVE_CELLS 子命令说明

命令	名称	说明
0x0083	CB_ACTIVE_CELLS	运行电芯均衡的电芯：写入时，开始对指定的电芯进行均衡。

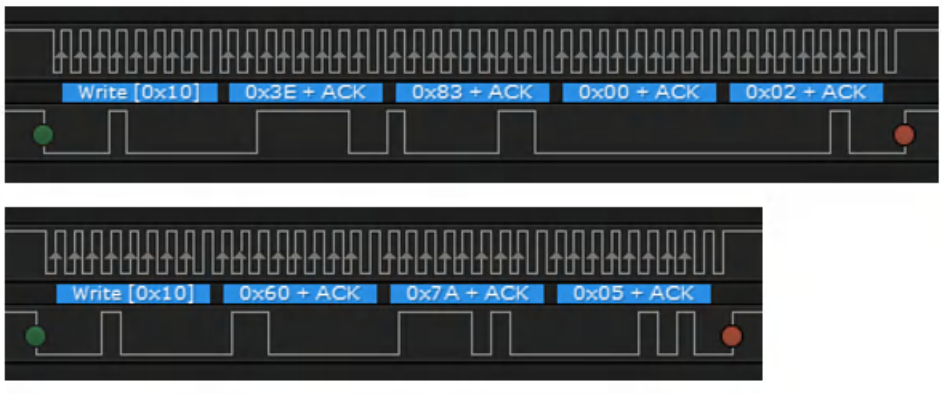
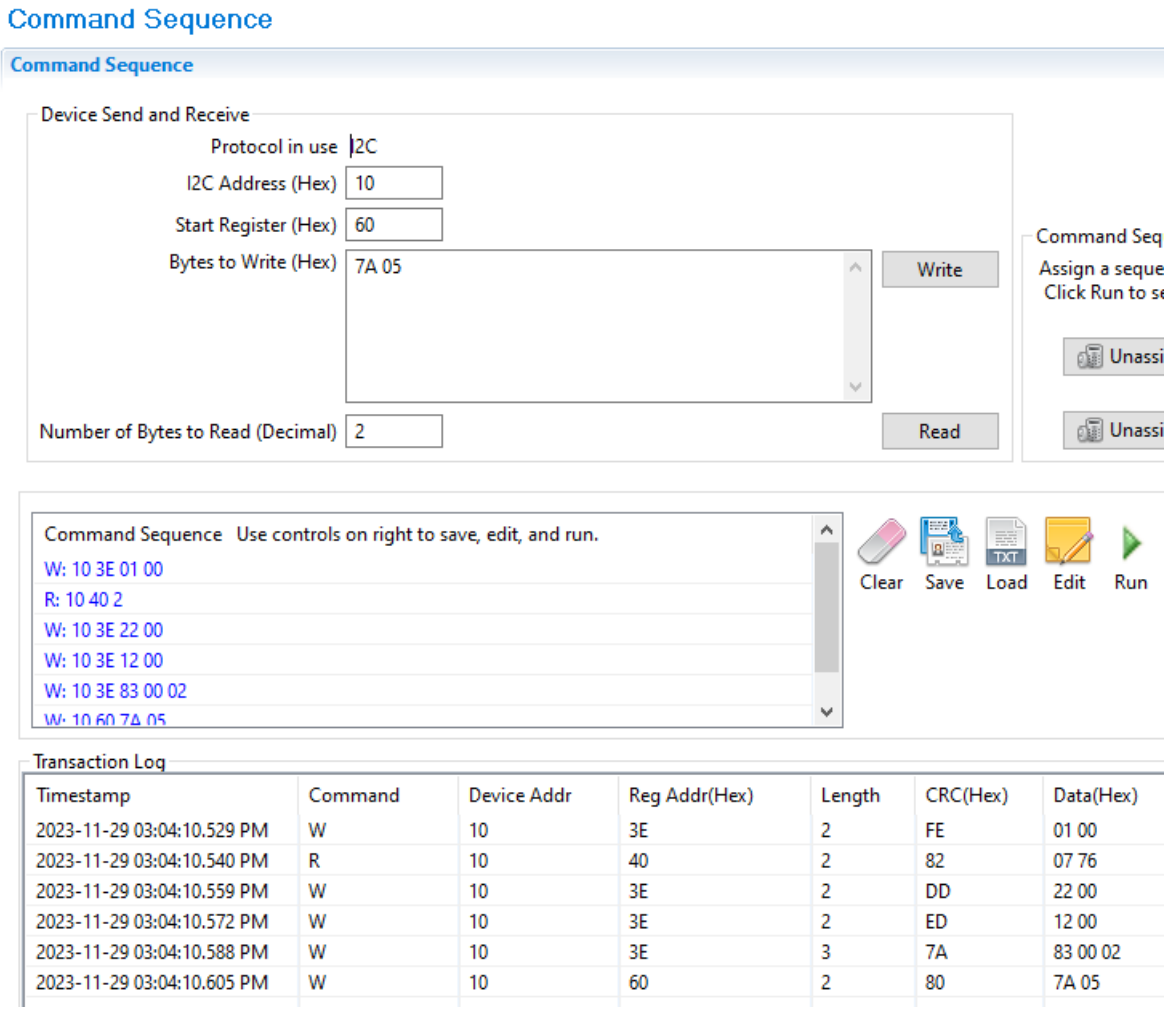


图 2-4. 为 CB_ACTIVE_CELLS 子命令捕获的 I2C 波形

2.5 子命令摘要

图 2-5 中的 *Transaction Log* 展示了已阐述过用于执行子命令的所有命令。



Timestamp	Command	Device Addr	Reg Addr(Hex)	Length	CRC(Hex)	Data(Hex)
2023-11-29 03:04:10.529 PM	W	10	3E	2	FE	01 00
2023-11-29 03:04:10.540 PM	R	10	40	2	82	07 76
2023-11-29 03:04:10.559 PM	W	10	3E	2	DD	22 00
2023-11-29 03:04:10.572 PM	W	10	3E	2	ED	12 00
2023-11-29 03:04:10.588 PM	W	10	3E	3	7A	83 00 02
2023-11-29 03:04:10.605 PM	W	10	60	2	80	7A 05

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，请转到“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数据表中有更多描述，但在以下示例中进行了说明。

备注

写入 RAM 寄存器时，建议先进入 CONFIG_UPDATE 模式，然后在完成后执行退出 CONFIG_UPDATE 模式的命令。这样可确保在修改设置时稳定运行。

3.1 读取启用保护功能 A

BQ7690x 器件的默认设置启用了 COV（过压）、SCD（短路）和 REGOUT（LDO 输出）保护。这在下面进行验证，方法是对 ***Enabled Protections A*** 寄存器进行读取，如表 3-1 所示。在图 3-1 中，从 RAM 地址 0x9024 返回的值为 0xA1。

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	Protection	Enabled Protections A	U1	0x00	0xFF	0xA1	十六进制

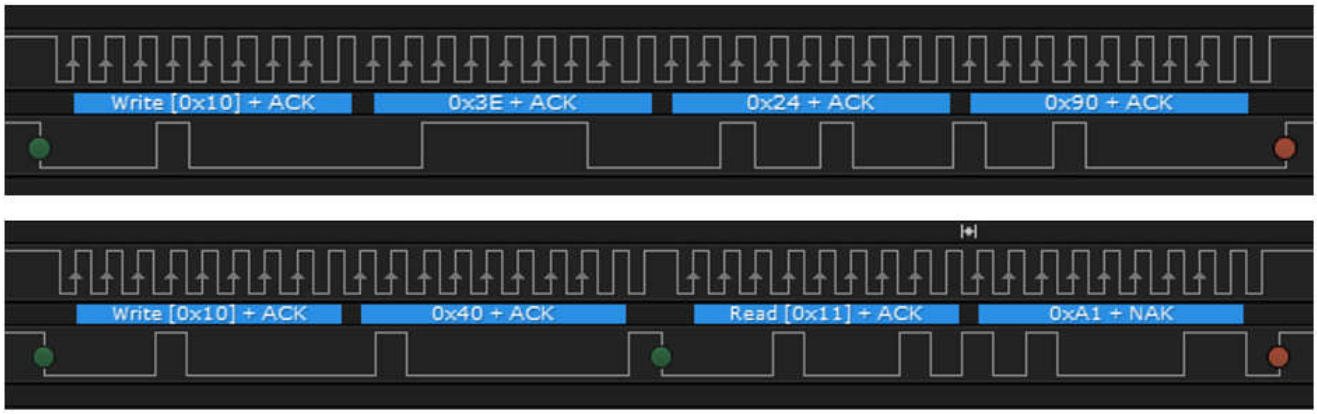


图 3-1. 读取 Enabled Protections A 寄存器后捕获到的 I2C 波形

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，防止设置在完成所有更改之前生效。请参阅表 3-2。SET_CFGUPDATE 和 EXIT_CFGUPDATE 都遵循子命令格式。在图 3-2 中，SET_CFGUPDATE 子命令是通过将 0x0090 写入 0x3E 来给出的。

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 Battery Status()[POR] 和 Battery Status()[WD] 位。

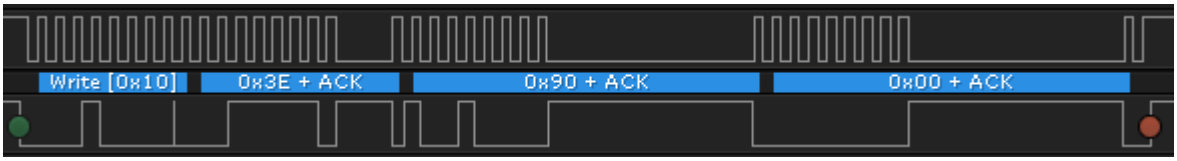


图 3-2. 为 SET_CFGUPATE 捕捉的 I2C 波形

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。这要求将 0xE1 写入 RAM 地址 0x9024，如图 3-3 所示。校验和是根据地址和数据（0x24、0x90、0xE1）计算的，并且是这些字节总和的补码。在本例中，校验和为 0x6A。长度包括器件地址和命令地址占用的两个字节，总长度为 0x05。

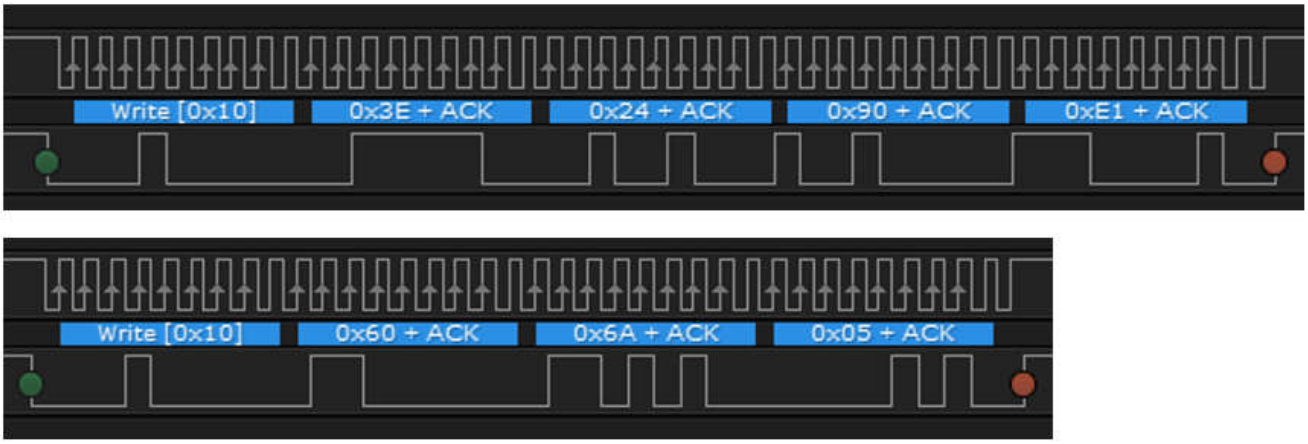


图 3-3. 为写入启用保护功能 A 捕捉的波形

3.4 写入 VCell Mode

接下来，向表 3-3 中所示的 ***VCell Mode*** 寄存器写入，以便为 4 节电芯配置器件 BQ76905。以下示例将 0x04 写入 0x901B，然后将新的校验和及长度写入 0x60/0x61，如图 3-4 所示。

表 3-3. VCell Mode 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 Mode	H2	0x0000	0xFFFF	0x0000	十六进制

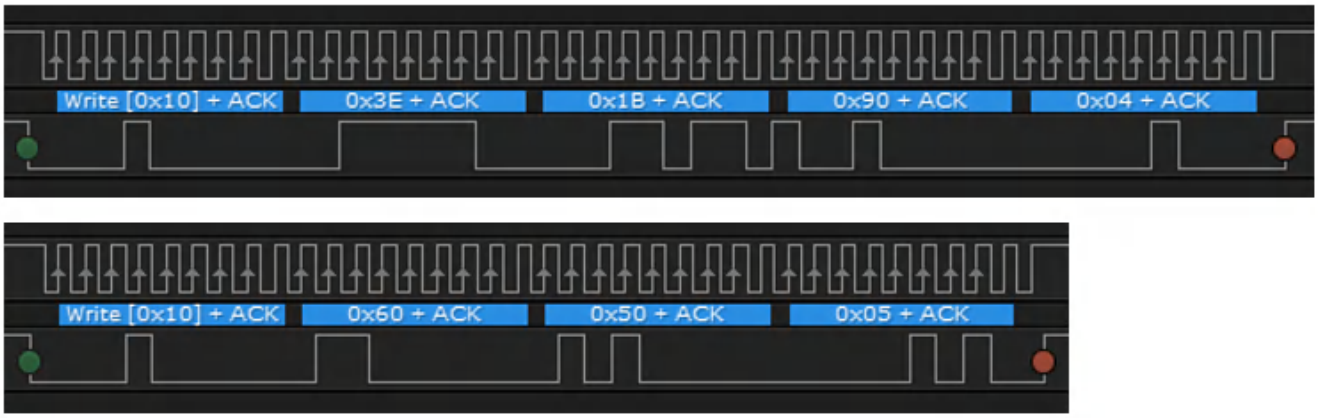


图 3-4. 为写入“VCell Mode”捕捉的 I2C 波形

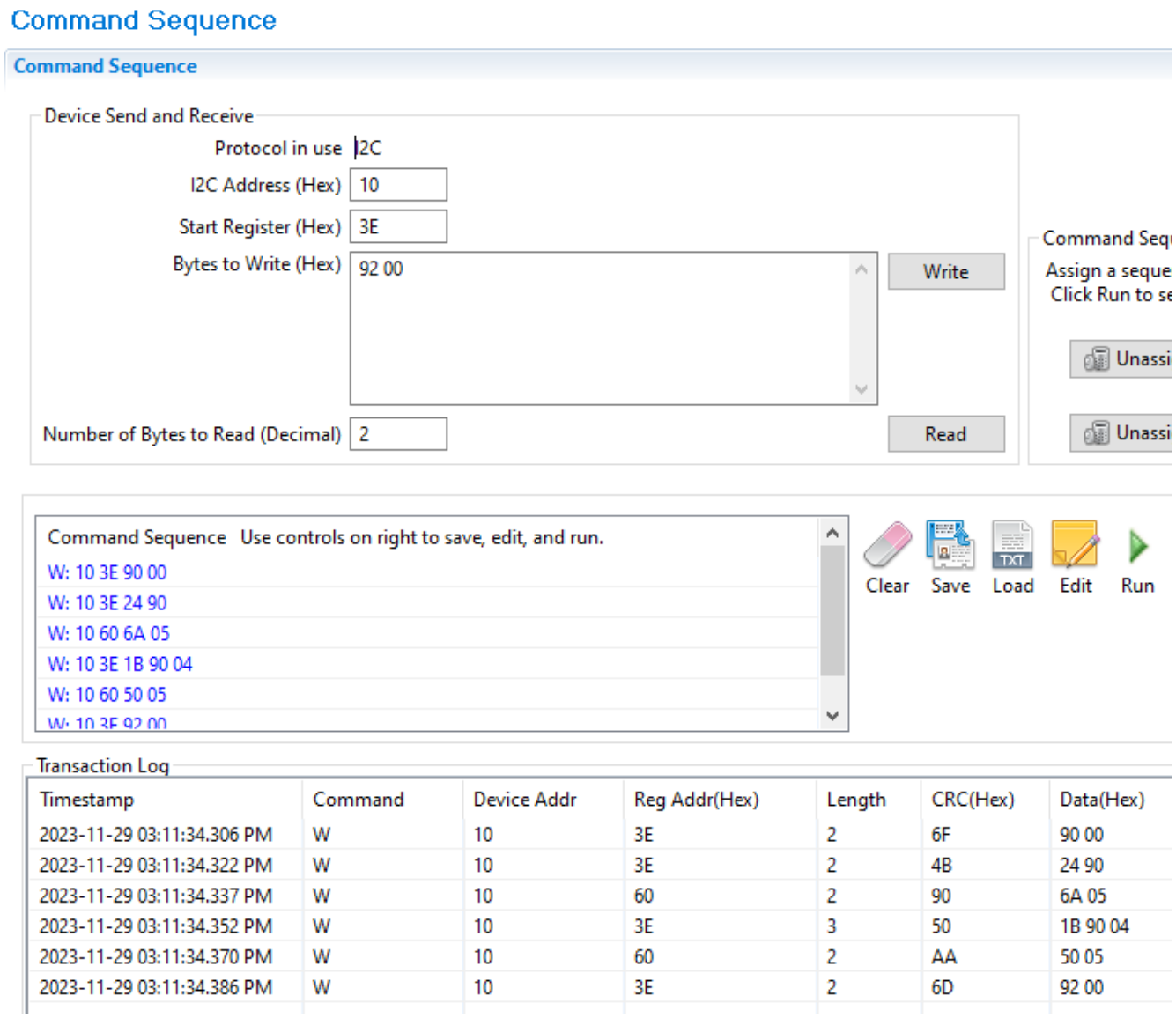
3.5 退出 CONFIG_UPDATE 模式

写入 RAM 寄存器后，使用 EXIT_CFGUPDATE 子命令退出 CONFIG_UPDATE 模式。请参阅图 3-5。此时，新的设置即可生效。

图 3-5. 为 EXIT_CFGUPDATE 捕捉的 I2C 波形

3.6 读取和写入 RAM 寄存器摘要

图 3-6 中的 *Transaction Log* 展示了已阐述过用于读取和写入 RAM 寄存器的所有命令。



Timestamp	Command	Device Addr	Reg Addr(Hex)	Length	CRC(Hex)	Data(Hex)
2023-11-29 03:11:34.306 PM	W	10	3E	2	6F	90 00
2023-11-29 03:11:34.322 PM	W	10	3E	2	4B	24 90
2023-11-29 03:11:34.337 PM	W	10	60	2	90	6A 05
2023-11-29 03:11:34.352 PM	W	10	3E	3	50	1B 90 04
2023-11-29 03:11:34.370 PM	W	10	60	2	AA	50 05
2023-11-29 03:11:34.386 PM	W	10	3E	2	6D	92 00

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

BQ7690x 系列的 I2C 接口包含一个可选的 CRC 校验。可以在 ***Settings:Configuration:I2C_Config[CRC]*** 寄存器中启用 CRC 特性。如果在使用 BQStudio 时更改了该寄存器，则可以重新启动 BQStudio，以便检测新的通信模式。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 中使用了 *VCell 1* 命令，针对 [0x10 0x14 0x11 0xC5] 计算了第一个字节的 CRC。生成的 CRC 为 0x79。第二字节 [0x0B] 的 CRC 为 0x31。

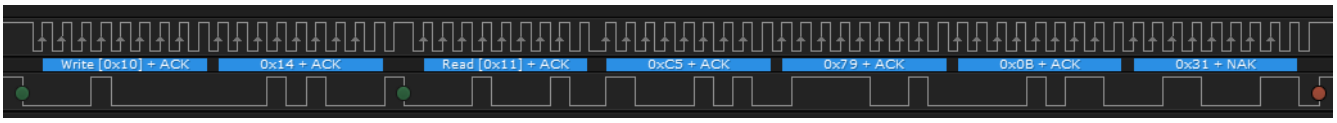


图 4-2. 使用 CRC 为 VCell 1 命令捕捉的 I2C 波形

5 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 BQ76905 或 BQ76907 评估模块上的 USB 连接器从 PC 与 BQ7690x 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适用于 I2C。

```
'''
/* BQ7690x example Program demonstrates examples for direct commands, subcommands, and writing /
reading from device RAM.
'''
I2C_ADDR = 0x10  # BQ7690x default responder address
numCells = 5    # Set to 7 for BQ76907
def DataRAM_Read(addr, length):
    '''
    Write address location to 0x3E and read back from 0x40
    Used to read dataflssh 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 dataflssh
    Add 2 to length for Rev A0 of Octane
    '''
    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 ReadCellVoltage(cell):
    '''
    Reads a specific cell voltage
    '''
    cmd_addr = 0x14 + (cell * 2)
    result = I2C_Read(I2C_ADDR, cmd_addr, 2)
    print ("Cell", cell, " = ", (result[1]*256 + result[0]), " mV")
    return
def ReadAllCellVoltages():
    '''
    Reads all cell voltages, Stack voltage, PACK voltage, and LD voltage
    '''
    cmd_addr = 0x12
    for i in range(0,numCells):
        cmd_addr += 2
        result = I2C_Read(I2C_ADDR, cmd_addr,2)
        print ("Cell", (i+1), " = ", (result[1]*256 + result[0]), " mV")
    result = I2C_Read(I2C_ADDR, 0x26,2)
    print ("Stack Voltage = ", (result[1]*256 + result[0]), " mV")
    result = I2C_Read(I2C_ADDR, 0x22,2)
    print "REG18 Voltage = ", (result[1]*256 + result[0]), " ADC Counts"
    result = I2C_Read(I2C_ADDR, 0x24,2)
    print ("VSS Voltage = ", (result[1]*256 + result[0]), " ADC Counts")
    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
```

```
########################################
#  Start of Main Script
########################################
################ Direct Command Examples ################
#Write Alarm Enable to 0x0060 - FULLSCAN, ADSCAN
I2C_Write(I2C_ADDR, 0x66, [0x60, 0x00])
#Read Voltage on Cell #1
result = I2C_Read(I2C_ADDR, 0x14, 2)
print ("Cell 1 = ", (result[1]*256 + result[0]), " mV")
#Read Internal Temperature
result = I2C_Read(I2C_ADDR, 0x28, 2)
print ("Internal Temp = ", ((result[1]*256 + result[0])), "degrees C")
#Read CC2 Current Measurement
result = I2C_Read(I2C_ADDR, 0x3A, 2)
print ("CC2", (result[1]*256 + result[0]), " mA")
################ Subcommand Examples ################
## Command-only Subcomands ##
#Read Device Number
b = DataRAM_Read(0x0001,6)
print ("Device_Number = " '{0:04X}'.format(b[1]*256+b[0]))
#FET_ENABLE
I2C_Write(I2C_ADDR, 0x3E, [0x22, 0x00])
#Cell Balance Write Command - starts balancing on specified cells when written
DataRAM_Write(0x0083, [0x02])
#Cell Balance Read Command - read which cells are being balanced
b = DataRAM_Read(0x0083,1)
print ("Cell Balancing = 0x" '{0:02X}'.format(b[0]))
#RESET - returns device to default settings
I2C_Write(I2C_ADDR, 0x3E, [0x12, 0x00])
sleep(2)
# Read 'Enabled Protections A' RAM register 0x9024
b = DataRAM_Read(0x9024,1)
print ("Enabled Protections A = 0x" '{0:02X}'.format(b[0]))
#Set CONFIG_UPDATE Mode (RAM registers can be written while in
#CONFIG_UPDATE mode and will take effect after exiting CONFIG_UPDATE mode
I2C_Write(I2C_ADDR, 0x3E, [0x90, 0x00])
#Write to 'Enabled Protections A' RAM register to enable CUV protection
DataRAM_Write(0x9024, [0xE1])
#Write to 'VCell Mode' RAM register to configure for a 9-cell battery
DataRAM_Write(0x901B, [0x04])
#Exit CONFIG_UPDATE Mode
I2C_Write(I2C_ADDR, 0x3E, [0x92, 0x00])
# CRC8 Example Calculation
TestValue = [0x10, 0x14, 0x11, 0x68]
crcKey = 0x107
check = 0xff & crc8(TestValue,crcKey)
print "crc8 check = 0x" '{0:02X}'.format(check)
ReadAllCellVoltages()
```

在 BQ76905 评估模块中运行示例 Python 脚本的输出如下。

```
('Cell 1 = ', 3089, ' mV')
('Internal Temp = ', 29, 'degrees C')
('CC2', 45, ' mA')
Device_Number = 7605
Cell Balancing = 0x02
Enabled Protections A = 0xA1
crc8 check = 0x33
('Cell', 1, ' = ', 3089, ' mV')
('Cell', 2, ' = ', 3082, ' mV')
('Cell', 3, ' = ', 3093, ' mV')
('Cell', 4, ' = ', 3089, ' mV')
('Cell', 5, ' = ', 3089, ' mV')
('Stack Voltage = ', 15471, ' mV')
REG18 Voltage =  19153  ADC Counts
('VSS Voltage = ', 3, ' ADC Counts')
```

6 参考文献

- 德州仪器 (TI)，*2S-7S BQ76907 电池监测器和保护器* 数据表。
- 德州仪器 (TI)，*2S-5S BQ76905 电池监测器和保护器* 数据表。
- 德州仪器 (TI)，*BQ76907 技术参考手册*。
- 德州仪器 (TI)，*BQ76905 技术参考手册*。

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