binho

应用笔记

在 Binho Supernova 上为 NVMe-MI 实现单端口 I2C/SMBus/I3C 操作

AN0014

文档信息

关键词	AN0014, Supernova, I3C, I2C, SMBus, MCTP, 时钟延展（clock stretching）, 传统 I2C（legacy I2C）, tTIMEOUT, NVMe-MI, ARP, 热加入（Hot-Join）, PIC18F16Q20, pyCosmicSDK
摘要	许多 SMBus 和 MCTP 器件在准备响应时会将 SCL 保持为低电平。Supernova 的 I3C 引擎无法等待这一过程， 因此在其 I3C 端口上，这类目标器件过去会丢失数据位。主机适配器固件 4.6.0 和 pyCosmicSDK 1.6.0 新增了一种模式，在该模式下，I3C 端口上的传统 I2C 传输会等待目标器件，最长可达 SMBus 的 35 ms 超时。本应用笔记介绍该模式，提供一个利用该模式读写时钟延展目标器件的脚本，并提供用于测量的参考目标器件的固件： 在 100 kHz 和 400 kHz 下进行三轮运行，目标器件的延展时间为 0 至 30 ms，3600 次读取中 3600 次正确； 直至该模式约 750 kHz 的上限，每次延展读取均正确。该脚本还会在适配器能力范围内，执行 PCI-SIG 带外管理规范中的 SMBus 到 I3C Basic 检测流程。

Binho Inc.
42 Dore Street, Unit A, San Francisco, CA 94103

修订版 1.0
6 NOVEMBER 2026

1 简介

I2C 目标器件可以在应答其地址后将 SCL 保持为低电平，以便在应答前争取时间。控制器必须等待 SCL 被释放。 I2C 规范将此称为时钟延展（clock stretching），SMBus 允许其最长持续 35 ms（tTIMEOUT）。SMBus 和 MCTP 器件经常使用这一机制：正在获取数值的管理控制器会保持时钟，直到数值准备就绪。

Supernova 的 I3C 端口可以在与 I3C 器件相同的总线上访问传统 I2C 目标器件。然而，其 I3C 引擎在目标器件将 SCL 保持为低电平时仍会继续输出时钟，因此进行时钟延展的目标器件会漏掉数据位，传输返回错误数据或无应答。 主机适配器固件 4.6.0 和 pyCosmicSDK 1.6.0 新增了一种模式，为 I3C 端口上的传统 I2C 传输解决了这一问题。

1.1 范围

本文档涵盖：

- 时钟延展模式改变了什么，以及该模式开启时拒绝哪些操作
- 该模式运行的 SCL 速率，以及与请求值的接近程度
- 一个通过该模式读写时钟延展目标器件的脚本
- 一个延展时间可配置的参考目标器件及其固件
- 针对该目标器件的测量结果
- 用于 NVMe-MI 的 SMBus 到 I3C Basic 检测流程，以及适配器执行其中的哪些步骤

本文档不涉及 ARP Get UDID 的 PEC 校验以外的 SMBus 帧格式（块读写、快速命令），也不涉及 MCTP 或 NVMe-MI 消息，以及 Supernova 独立的 I2C 端口。

1.2 适用器件

表 1. 适用器件

角色	部件	标识	依据
控制器	Binho Supernova	硬件版本 C，固件 4.6.1，I3C HV 端口	已测量，本文档
控制器	Binho Supernova	固件 4.5.1 或更早版本	以 `FW_UNSUPPORTED_COMMAND` 拒绝该模式
控制器	Binho Pulsar	任意	不适用：Pulsar 没有 I3C 端口
参考目标器件	Microchip PIC18F16Q20 Curiosity Nano	固件 `stretch_target_pic18q20.c`	已测量，本文档

1.3 相关文档

表 2. 相关文档

文档	标题
UM10204	I2C-bus specification and user manual, Rev. 7, NXP Semiconductors
SMBus 3.2	System Management Bus Specification, version 3.2
PCI-SIG ECN	PCI-SIG Engineering Change Notice: Chapter 12. Architectural Out-of-Band Management
AN0015	使用 Binho Supernova 和 Pulsar 设置 I2C SCL 占空比和精确的总线频率

2 系统概述

模式关闭时，I3C 端口上的 `legacy_i2c_read` 和 `legacy_i2c_write` 由 I3C 引擎执行，引擎按自己的时序驱动 SCL。模式开启时，适配器固件以软件方式为这些传输产生时钟，并在其释放的每个阶段之后检查 SCL。当目标器件将 SCL 保持为低电平时，固件会等待。

以下四点决定了该模式的使用方式：

- **它是一种模式，而不是单次传输的选项。** `enable_i2c_clock_stretching(frequency_hz)` 为之后的所有传统 I2C 传输开启该模式；`disable_i2c_clock_stretching()` 将其关闭。`initialize` 和 `configure` 也会将其关闭。
- **它排斥 I3C 通信。** 模式开启期间，SDR、HDR 和 CCC 调用以及 `init_bus` 和 `reset_bus` 会引发 `I3cError`，状态为 `FW_I3C_I2C_CLOCK_STRETCHING_ACTIVE`。与 I3C 器件通信之前请先关闭该模式。
- **它强制执行 SMBus 超时。** 超过 35 ms 的延展会以 `FW_BUS_TIMEOUT` 结束传输，并释放总线。
- **速率由软件时钟设定。** 该调用接受 10 kHz 至 1 MHz，并返回其计算出的速率：100 kHz 时为 97975 Hz， 400 kHz 时为 391644 Hz，与总线上的实际值相差在 1 % 以内。高于 400 kHz 时，返回的速率与总线上的实际值相差可达 7 %，600 kHz 的请求实际运行在 615 kHz（第 9.2 节）。时钟最高约为 750 kHz，因此 800 kHz 和 1 MHz 的请求都运行在该速率。

3 硬件设置

3.1 所需设备

表 3. 所需设备

项目
Binho Supernova，固件 4.6.0 或更高版本
USB-C 线缆
一个进行 SCL 时钟延展的 I2C 或 SMBus 目标器件，或第 3.4 节的参考目标器件
SCL 和 SDA 上的上拉电阻，如果目标板上没有

3.2 连接

表 4. Supernova I3C HV 端口与目标器件之间的连接

Supernova I3C HV 端口	目标器件	是否必需
SCL	SCL	必需
SDA	SDA	必需
GND	GND	必需
VTARG	目标器件 I/O 电源	可选，若目标器件由适配器供电

3.3 总线注意事项

I3C 端口的 SCL 上没有上拉电阻。I2C 总线的两条线上都需要上拉电阻。在我们的测试台上，上拉来自一台 Binho Pulsar 的 I2C 端口，设置为 1 kOhm 并保持空闲；一对接到总线电压的 1 kOhm 至 2.2 kOhm 电阻也能起到相同作用。 SCL 上没有上拉电阻时，报告的速率会急剧下降，因为固件要等待每一个上升沿。

> **注意：** I3C HV 端口的电压由 `set_voltage` 选择：1200 mV 至 3300 mV。连接不耐受 3.3 V 的目标器件之前， 请将其设置为目标器件的 I/O 电压。
>
> **注意：** 总线上只能有一个控制器。如果由第二台适配器提供上拉电阻，请不要从该适配器发起传输。

3.4 参考目标器件

为了测量该模式，我们需要一个延展时间可设置的目标器件。资源压缩包中的 `stretch_target_pic18q20.c` 可将 PIC18F16Q20 Curiosity Nano 变成这样的目标器件。其 I2C 外设在每次地址匹配后将 SCL 保持一段可编程的时间， 这正是忙碌的 SMBus 器件的行为。

表 5. 参考目标器件协议，地址 0x2A

传输	效果
写入 `0xF0`、低字节、高字节	在之后的每次地址匹配后延展该微秒数；0 关闭延展
读取	每个字节返回 `0xA5`
其他任何写入	接受并丢弃

SCL 和 SDA 位于 RB6 和 RB5。使用 MPLAB XC8 构建，并通过 MPLAB X 或 Curiosity Nano 的拖放式编程器烧录。

4 软件设置

安装 pyCosmicSDK（最新版本）：

```
pip install --upgrade pycosmicsdk
```

脚本 `smbus_stretch.py` 位于资源压缩包中。它不需要其他任何依赖。

5 时钟延展模式

5.1 使用方法

单独列出的调用如下：

```
import pycosmicsdk as p

dev = p.Device.open(model=p.DeviceModel.SUPERNOVA)
i3c = dev.i3c()
i3c.set_voltage(voltage_mv=3300)
i3c.bring_up(
    push_pull_rate=p.I3cPushPullRate.PUSH_PULL_1_MHZ_DC_40,
    open_drain_rate=p.I3cOpenDrainRate.OPEN_DRAIN_100_KHZ,
    i2c_open_drain_rate=p.I2cOpenDrainRate.STANDARD_MODE,
)
print(i3c.enable_i2c_clock_stretching(100_000))
# I2cClockStretching(enabled=True, frequency_hz=97975)
data = i3c.legacy_i2c_read(0x2A, 16)
i3c.disable_i2c_clock_stretching()
```

寄存器读取，即 SMBus Read Byte 或 Read Word 模式，将寄存器作为 `subaddress` 传入：`i3c.legacy_i2c_read(0x0B, 2, subaddress=bytes([0x09]))` 。

5.2 状态码

表 6. 该模式特有的状态码

状态	引发者	含义
`FW_UNSUPPORTED_COMMAND`	enable、disable	固件 4.5.1 或更早版本
`FW_INVALID_PARAMETER`	enable	速率超出 10 kHz 至 1 MHz
`FW_INTERFACE_NOT_INITIALIZED`	enable、disable	I3C 控制器未启动
`FW_I3C_I2C_CLOCK_STRETCHING_ACTIVE`	SDR、HDR、CCC、`init_bus` 、`reset_bus`	模式已开启；请先将其关闭
`FW_BUS_TIMEOUT`	传统读取或写入	目标器件保持 SCL 超过 35 ms

5.3 SMBus 超时

在固件验证期间，将参考目标器件设置为延展 40 ms，100 次读取中返回 50 次 `FW_BUS_TIMEOUT` 和 50 次 `FW_I3C_NACK_ADDRESS`，最长耗时 46 ms，适配器始终保持响应。地址 NACK 来自目标器件：它在仍在计满 40 ms 时漏掉了下一个地址。

6 操作步骤

1. 按第 3.2 节连接目标器件和上拉电阻。为目标器件上电。
2. 从目标器件读取。第一行显示固件和生效的速率：

```
$ python smbus_stretch.py read 0x2A 16
FW-4.6.1  I3C port 3300 mV  stretching on, SCL 97975 Hz
read  0x2A: a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
```

3. 向其写入：

```
$ python smbus_stretch.py write 0x2A 0x01 0x02 0x03
FW-4.6.1  I3C port 3300 mV  stretching on, SCL 97975 Hz
write 0x2A: 01 02 03
```

4. 使用参考目标器件扫描延展时间。对于每个时长，脚本设置目标器件的延展时间，读取 16 字节 100 次，并检查每个字节：

```
$ python smbus_stretch.py sweep
FW-4.6.1  I3C port 3300 mV  stretching on, SCL 97975 Hz
  stretch  reads   ok  median ms  result
      0 us    100  100       6.02  all bytes correct
     50 us    100  100       6.02  all bytes correct
    500 us    100  100       7.20  all bytes correct
   2000 us    100  100       8.13  all bytes correct
  10000 us    100  100      18.01  all bytes correct
  30000 us    100  100      38.99  all bytes correct
```

5. 在 400 kHz 下重复：

```
$ python smbus_stretch.py --khz 400 sweep
FW-4.6.1  I3C port 3300 mV  stretching on, SCL 391644 Hz
  stretch  reads   ok  median ms  result
      0 us    100  100       5.10  all bytes correct
     50 us    100  100       5.04  all bytes correct
    500 us    100  100       6.11  all bytes correct
   2000 us    100  100       8.25  all bytes correct
  10000 us    100  100      16.90  all bytes correct
  30000 us    100  100      37.77  all bytes correct
```

每次读取的中位时间随延展时间增加，且增加量约等于延展时间：适配器每次都在等待。脚本在退出前会将目标器件的延展时间恢复为 0。

7 SMBus 到 I3C Basic 检测流程

从 SMBus 迁移到 I3C Basic 的 NVMe-MI 器件共用一对引脚，因此主机必须在改变总线之前查明每个器件使用哪种协议。 主机遵循的流程定义于 PCI-SIG Engineering Change Notice 第 12 章 Architectural Out-of-Band Management。 `smbus_stretch.py` 的 `handoff` 命令在 Supernova I3C 端口上执行该流程。表 7 列出各步骤及其执行者。

表 7. 检测流程步骤

步骤	流程	执行者
1	在 SMBus 电压下向 7Eh 广播带 DISHJ 的 DISEC；ACK 表示存在 I3C Basic 器件	脚本。只有 7Eh 头为开漏；CCC 字节和数据以 I3C 速率推挽传输。NACK 报告为 `FW_I3C_NACK_ADDRESS`
2	在静态地址和 61h（ARP Get UDID）上进行 SMBus 器件发现	脚本，开启时钟延展模式。脚本校验 PEC
3	发现仅支持 SMBus 的器件：复位 SMBus 接口并保持在 SMBus 模式	脚本：`hold_scl_low` 将 SCL 保持为低电平 T2wrst（`--t2wrst-us`）。带 SMRST# 的器件也可以通过 GPIO 复位
4	再次发送带 DISHJ 的 DISEC，然后关闭 SMBus 上拉电阻并开启 I3C Basic 上拉电阻	DISEC 由脚本执行。适配器的 SDA 上拉电阻在 SMBus 阶段由 `set_sda_pull_up`（`--sda-pull-up`）设置，之后跟随 I3C 引擎。适配器的 SCL 上没有上拉电阻：外部上拉电阻需手动切换
5	等待 Tsmb2i3c，切换到 I3C Basic 电压，广播带 ENHJ 的 ENEC	脚本。`--t-smb2i3c-ms` 和 `--i3c-voltage-mv` 取自规范和器件
6	器件执行热加入（Hot-Join）	固件：由其自行分配动态地址，脚本打印该地址

时钟延展模式拒绝所有 CCC（第 5.2 节），因此脚本仅在步骤 2 开启该模式，并在步骤 4 之前再次关闭。

如需在 Saleae 分析仪上查看该流程，`capture_flow.py` 会通过 Logic 2 的自动化服务器启动采集，以 `--` 之后的参数运行 `handoff`，并保存附带 Logic 2 I2C 分析器和 `flow_labels` 扩展的采集文件，该扩展在单独一行中标注每个步骤。`--step-delay-s` 使各步骤之间留出间隔，以便在时间轴上找到每个步骤。

> **说明：** 仅运行过仅 SMBus 的分支，在第 9.1 节的测试台上使用固件 4.6.1 运行 6 次，6 次结果一致，其中一次在适配器复位之后：步骤 1 收到 NACK，步骤 2 在 0x2A 找到参考目标器件且未发现 ARP 器件， SMBus 复位在请求 35000 us 时将 SCL 保持为低电平 35000 至 35777 us。步骤 3 至 6 需要一个应答 7Eh 并执行热加入的器件，ARP 和 PEC 路径需要一个支持 ARP 的器件；测试台上两者均没有。PEC 例程仅以 CRC-8/SMBUS 校验值进行了检验。

```
$ python smbus_stretch.py --khz 400 handoff --address 0x2A 0x2B --sda-pull-up ON --
t2wrst-us 35000
FW-4.6.1  I3C port 3300 mV (SMBus voltage)
1. DISEC(DISHJ) to 7Eh: NACK, SMBus only
2. SMBus discovery, stretching on, SCL 391644 Hz
  static 0x2A: ACK
  static 0x2B: FW_I3C_NACK_ADDRESS
  ARP 0x61 Get UDID: FW_I3C_NACK_ADDRESS (no ARP-capable device)
   SMBus mode stays.
   SMBus reset: SCL held low 35059 us (asked 35000 us)
```

8 参考

表 8. smbus_stretch.py 选项

选项	默认值	含义
`--serial`	任意	Supernova 序列号
`--voltage-mv`	3300	I3C 端口电压，HV 连接器为 1200 至 3300
`--khz`	100	SCL 速率，10 至 1000；最高约运行在 750 kHz

表 9. smbus_stretch.py 命令

命令	参数	操作
`read`	地址、长度、可选 `--register`	读取，可选择先写入一个寄存器字节
`write`	地址、字节	写入这些字节
`sweep`	`--address`、`--count`、`--length`	仅限参考目标器件：延展 0、50、500、2000、10000 和 30000 us，每档读取 `--count` 次
`handoff`	`--address`、`--i3c-voltage-mv`、`--t-smb2i3c-ms`、`--hj-timeout-s`、`--no-smbus-only`、`--yes`、`--t2wrst-us`、`--sda-pull-up`、`--step-delay-s`、`--pulsar-pull-up`	第 7 节的检测流程

每个命令都会在传输前开启该模式，并在退出前关闭，因此之后 I3C 端口可用于 I3C 通信。

9 测量结果

9.1 测试配置

表 10. 测试配置

项目	值
适配器	Binho Supernova，硬件版本 C，固件 4.6.1（`FW-4.6.1`）。第 9.2 节中的长时间测试、SMBus 复位和 SDA 上拉电阻测量来自同一功能的 4.6.0 预发布版本
主机软件	来自 PyPI 的 pyCosmicSDK 1.6.0，Linux
总线	I3C HV 端口，3.3 V，SCL 和 SDA 上的 1 kOhm 上拉电阻来自一台空闲的 Binho Pulsar，固件 4.6.1
分析仪	Acute TravelBus 逻辑分析仪，800 MS/s，阈值 1.6 V
目标器件	PIC18F16Q20 Curiosity Nano，`stretch_target_pic18q20.c`，地址 0x2A

9.2 结果

第 6 节的扫描在两种速率和六种延展时长下共读取 1200 次，每次 16 字节。共运行三次，第三次在适配器复位之后：3600 次正确，无错误。

同一功能在预发布版本上进行固件验证得到的其他结果：

- 长时间测试，随机进行 1 至 64 字节的读写，延展时间每 100 次传输在 0、50 us、500 us、2 ms 和 10 ms 之间变化：在 100 kHz 和 400 kHz 下，每轮约 3400 次延展传输中错误读取为 0，其中包括一轮在断电重启之后的运行。

固件 4.6.1 上按请求值列出的 SCL 速率。高于 400 kHz 时，每种延展时间（50 us 至 30 ms）读取 16 字节 20 次：

表 11. 按请求值列出的 SCL 速率

请求	报告值	总线实测	tLOW / tHIGH	延展读取
100 kHz	97975 Hz	98.2 kHz	5105 / 5075 ns	1500 次中 1500 次
400 kHz	391644 Hz	395.3 kHz	1600 / 935 ns	1500 次中 1500 次
600 kHz	583657 Hz	615.4 kHz	855 / 755 ns	100 次中 100 次
800 kHz	746268 Hz	749.1 kHz	750 / 585 ns	100 次中 100 次
1 MHz	746268 Hz	749.1 kHz	750 / 585 ns	100 次中 100 次

总线实测值为每种速率约 9000 个 SCL 周期的中位数。每种速率的 tLOW 和 tHIGH 都高于 UM10204 对其模式规定的最小值。 高于 400 kHz 时，返回的速率与总线实际值不一致：500 kHz 的请求报告

477707 Hz，实际运行在 467.3 kHz；700 kHz 报告 652173 Hz，实际运行在 696.9 kHz；600 kHz 实际运行比请求值高 2.6 %。如果 400 kHz 以上的精确速率很重要，请在总线上实测。

当参考目标器件不进行延展时，高于 400 kHz 的读取会在第二个字节返回 0xFF，每种速率 100 次中 0 次正确，而每次延展读取都正确。 两种情况下适配器的 SCL 相同；只是目标器件自身的时钟延展关闭了。我们将其归因于参考目标器件的固件在不延展时未能及时装载第二个字节，而不是适配器。

SMBus 复位（`hold_scl_low`），每个测量点保持 5 次，预发布固件 `FW-4.5.1-b11a067e`，SCL 低电平时间在 Acute 上测量。跨越分析仪某个缓冲区边界的保持读数约短 1.5 ms，已排除：

表 12. SMBus 复位的 SCL 保持

请求值	总线实测	适配器报告	保持后的读取
1 ms	999.7 至 1678 us	1000 至 1677 us	5 次中 5 次
10 ms	9987 至 11006 us	10000 至 10998 us	5 次中 5 次
35 ms	34996 至 35941 us	35000 至 35937 us	每种模式 5 次中 5 次，复位后亦然
100 ms	100013 至 100114 us	100000 us	5 次中 5 次

以分析仪的时钟为准，一次保持最多可能长约 1 ms，最多短 0.13 %，这是适配器内部振荡器的容差；请求 T2wrst 时请预留该余量。0 us 和超过 1 s 的请求会以 `FW_INVALID_PARAMETER` 拒绝，固件 4.5.1 则返回 `FW_UNSUPPORTED_COMMAND`。

SDA 上拉电阻，在 Pulsar 的上拉电阻关闭时，测量每个 STOP 处从 SCL 上升到 SDA 越过 1.6 V 的延迟：`OFF` 时为 7813 ns，`ON` 时为 5975 ns，`AUTO` 时为 5965 ns。三种设置下，无论是否开启时钟延展模式，读取均为 10 次中 10 次。

以上是我们在一个测试台上测得的结果，并非规格。

10 故障排除

表 13. 故障排除

现象	可能原因及措施
开启时出现 `FW_UNSUPPORTED_COMMAND`	固件 4.5.1 或更早版本。将 Supernova 更新到 4.6.0 或更高版本。
报告的速率远低于请求值	SCL 上没有上拉电阻。添加一个（第 3.3 节）。
`FW_I3C_I2C_CLOCK_STRETCHING_ACTIVE`	在模式开启时进行了 I3C 调用。请先调用 `disable_i2c_clock_stretching`。
`FW_BUS_TIMEOUT`	目标器件保持 SCL 超过 35 ms。检查目标器件；这就是 SMBus tTIMEOUT。
重新配置参考目标器件后立即出现 `FW_I3C_NACK_ADDRESS`	目标器件会重启其 I2C 外设以应用新的延展时间。脚本为此等待 10 ms。
模式关闭时从时钟延展目标器件读到错误数据	预期行为：I3C 引擎不会等待。请开启该模式。
400 kHz 以上开启时出现 `FW_INVALID_PARAMETER`	4.6.0 之前的固件有 1 MHz 限制。请更新 Supernova。
400 kHz 以上 `sweep` 中出现 `FW_I3C_I2C_NACK_BYTE`	参考目标器件在不延展时漏掉了重新配置的写入：在我们的测试台上 24 次扫描中出现 1 次。请重新运行扫描。
`capture_flow.py` 在 Saleae Logic Pro 上解码出错误的地址和数据	数字阈值为设备默认值。对于 3.3 V 总线请传入 `--threshold 3.3`：在我们的 Logic Pro 8 上，在同一次扫描中，SCL 为高电平时出现的 SDA 边沿从 732 个减少到 50 个，且全部是 START 和 STOP。
总线上有 I3C Basic 器件时 `handoff` 步骤 1 收到 NACK	器件在所设电压下未处于 SMBus 模式，或未上电。检查 `--voltage-mv`。

11 缩略语

表 14. 缩略语

术语	含义
I2C	Inter-Integrated Circuit（集成电路间总线）
I3C	MIPI Improved Inter-Integrated Circuit（MIPI 改进型集成电路间总线）
SMBus	System Management Bus（系统管理总线）
MCTP	Management Component Transport Protocol（管理组件传输协议）
SCL	串行时钟线
SDA	串行数据线
SDR	单倍数据速率，I3C 的基本传输模式
HDR	I3C 的高数据速率传输模式
CCC	I3C 通用命令代码
tTIMEOUT	SMBus 时钟低电平超时，35 ms
ARP	SMBus 地址解析协议
PEC	SMBus 数据包错误码，一种 CRC-8
UDID	SMBus 唯一器件标识符
DISEC, ENEC	禁用和启用目标器件事件的 I3C CCC
Hot-Join	目标器件加入总线并获得动态地址的 I3C 事件
NVMe-MI	NVM Express Management Interface（NVM Express 管理接口）
T2wrst, Tsmb2i3c	PCI-SIG 检测流程的时序参数

12 关于文档中源代码的说明

本文档中展示以及随附压缩包中提供的示例代码仅供评估和参考。这些代码以 MIT 许可证发布，许可证文本包含在压缩包中。

压缩包 `AN0014-assets.zip` 包含 `smbus_stretch.py`、`capture_flow.py`、Logic 2 扩展 `flow_labels`、`test_flow_labels.py`、`stretch_target_pic18q20.c` 和 `LICENSE`。

13 修订历史

表 15. 修订历史

版本	日期	说明
0.1	25 September 2026	供内部评审的草稿，在固件 4.6.0 和 pyCosmicSDK 1.6.0 预发布版本上测量。
0.2	28 September 2026	为 NVMe-MI 更改标题。速率最高 1 MHz（实际 746 kHz）。新增第 7 节 SMBus 到 I3C Basic 检测流程以及 `handoff` 命令。
0.3	28 September 2026	检测流程中加入 `hold_scl_low` 和 `set_sda_pull_up`，并附其测量结果。
0.4	30 September 2026	在固件 4.6.1 和来自 PyPI 的 pyCosmicSDK 1.6.0 上重新测量。给出每个请求值的总线实测 SCL 速率；高于 400 kHz 时返回的速率为近似值。`--serial` 可配合 pyCosmicSDK 1.6.0 使用。
1.0	6 November 2026	正式发布。

法律信息（英文）

Definitions

Draft

A document marked draft is under internal review and subject to formal approval. Binho Inc. makes no representation or warranty as to its accuracy or completeness and accepts no liability arising from its use.

Disclaimers

Limited warranty and liability

Information in this document is believed to be accurate and reliable. Binho Inc. makes no representation or warranty, express or implied, as to its accuracy or completeness, accepts no liability for the consequences of its use, and takes no responsibility for content originating outside Binho Inc.

In no event shall Binho Inc. be liable for any indirect, incidental, punitive, special or consequential damages, including without limitation lost profits, lost savings, business interruption, or costs of removing or replacing any product, whether or not based on tort, warranty, breach of contract or any other legal theory. Notwithstanding any damages a customer might incur, the aggregate and cumulative liability of Binho Inc. is limited in accordance with its terms and conditions of commercial sale.

Right to make changes

Binho Inc. reserves the right to change the information in this document, including specifications and product descriptions, at any time and without notice. This document supersedes all information supplied prior to its publication.

Suitability for use

Binho Inc. products are not designed, authorized or warranted for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure of a Binho Inc. product can reasonably be expected to result in personal injury, death, or severe property or environmental damage. Any such inclusion or use is at the customer's own risk and Binho Inc. accepts no liability for it.

Applications

Applications described here are illustrative only. Binho Inc. makes no warranty that they are suitable for the specified use without further testing or modification. Customers are responsible for the design and operation of their own applications and products, for appropriate design and operating safeguards, and for determining whether a Binho Inc. product is suitable and fit for their purpose. Binho Inc. accepts no liability for assistance with applications or customer product design.

Third-party products and specifications

This document refers to products, boards, specifications and documentation supplied by third parties, which remain the responsibility of their respective owners. Binho Inc. makes no warranty regarding them and their behavior may change without notice. Register maps, protocol details and device identifiers reproduced here were observed on the specific hardware and firmware versions stated in this document and must be confirmed against the vendor's own documentation for the reader's part.

Example code

Source code supplied with this document is provided as an example, for evaluation and reference. It is not qualified for production use, has not been developed under a functional safety or security process, and is licensed under the terms accompanying it in the distribution archive.

Terms and conditions of commercial sale

Binho Inc. products are sold subject to the general terms and conditions of commercial sale published by Binho Inc., unless otherwise agreed in a valid written individual agreement, in which case only the terms of that agreement apply.

Export control

This document and the items it describes may be subject to export control regulations. Export may require prior authorization from the competent authorities.

Security

All products may be subject to unidentified vulnerabilities or may support security standards with known limitations. The customer is responsible for the design and operation of its applications and products throughout their lifecycle to reduce the effect of such vulnerabilities, and for compliance with all legal, regulatory and security requirements concerning its products.

Trademarks

All referenced brands, product names, service names and trademarks are the property of their respective owners. **Binho** and the Binho logo are trademarks of Binho Inc. **PIC**、**MPLAB** 和 **Curiosity** 是 Microchip Technology Incorporated 的商标或注册商标。 **Acute** 和 **TravelBus** 是 Acute Technology Inc. 的商标或注册商标。**PCI-SIG** 是 PCI-SIG 的注册商标。**NVMe** 和 **NVMe-MI** 是 NVM Express, Inc. 的商标。