binho

应用笔记

使用 Binho Supernova 测量 CMIS 固件更新时间

AN0011

文档信息

关键词	AN0011, Supernova, Pulsar, CMIS, OIF, CDB, 固件更新, LPL, EPL, tWRITE, tCDBF, tBPC, Nmax, 光模块, 等待期（hold-off）, I2CMCI, I3C, SPIMCI, 每块事务数
摘要	在光模块上更新固件很慢，而显而易见的补救办法是换用更快的管理总线。按照 CMIS 的访问限制和 Table 10-4 的时序参数逐项推算后可以看出，时间并没有花在线路上：对于 1 MiB 的映像，总线时钟 只占总时间的一小部分，而每个块的等待期（hold-off）占了绝大部分。本应用笔记从规范推导出时间预算， 明确说明哪些数字是算术推导、哪些必须实测，并提供一条命令，对照规范上限为真实模块的各访问阶段计时， 使主机能够查明其模块实际需要哪些等待期。

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

修订版 1.0
6 NOVEMBER 2026

测试范围。第 4 节中的预算是由规范推导出的算术结果，现已涵盖规范和本系列所涉及的 全部三种管理接口：I2CMCI、SPIMCI，以及作为 Binho 推导的 I3C。第 5 节中的 `timing` 输出是 针对固件目标器件（而非光模块）的真实运行结果，而且事实证明它测量的是主机路径而不是目标器件。 第 5.2 节在两种主机传输方式上测量了两个适配器固件修订版，发现两者都没有改变这一下限；第 5.3 和 5.4 节将该下限追溯到适配器的 USB 描述符，并说明主机能够隐藏其中多少；第 7 节说明尚未完成的工作。

1 简介

在线路上，I3C 单倍数据速率（single data rate）大约比 I2C 快速模式（Fast-mode）快 31 倍。面对模块固件 更新缓慢问题的主机工程师，自然会预期更快的管理总线能让更新快大约 31 倍。

按照 CMIS 的访问限制推算，会得到不同的答案，而这一差异是可以付诸行动的：选择哪条通用数据块 （Common Data Block）命令来承载有效载荷，比选择哪种总线更重要。

本笔记推导出时间预算，并提供一种方法，用于查明特定模块实际把时间花在哪里，而不是规范允许它花在哪里。

本文档不给出实测吞吐量。 第 7 节说明原因，以及要给出该数值需要运行哪些测试。

1.1 范围

本文档涵盖 CMIS 固件下载的算术：访问长度限制、两条 Write Firmware Block 命令的块大小，以及 Table 10-4 中的等待期参数。本文档提供一条命令，对照这些上限为模块的读取、页切换和通用数据块 访问阶段计时。

本文档**不会**向模块写入固件，所提供的实用程序也没有执行此操作的命令。除记录其时序外，本文档不涉及 固件更新状态机、映像签名或 Run 与 Commit 序列。AN0008 涵盖寄存器模型，是阅读本文档的前提。

1.2 适用器件

表 1. 适用器件

项目	角色	依据
Binho Supernova	主机适配器	已测试，第 7 节
Binho Pulsar	主机适配器	未测试
任何 CMIS 受管模块	目标器件	**未测试。** 测量使用的是处于目标器件模式的适配器

第 4 节中的预算是算术推导，不是测量结果。 它由 OIF-CMIS-05.3 的 5.2.2 节和第 9 章以及 Table 10-4 推导而来，可用 `python cmis_mci.py budget` 复现。其中没有任何一行经过模块验证。

1.3 相关文档

表 2. 相关文档

文档	主题
OIF-CMIS-05.3	5.2.2.1 和 5.2.2.2 节，第 9 章（CDB），Table 10-4（时序）
AN0008	使用 Binho Supernova 通过 I2C 启动 CMIS 光模块
AN0009	使用 Binho Supernova 通过 I3C 管理 CMIS 光模块
AN0010	使用 Binho Pulsar 通过 SPI 启动 CMIS 光模块

2 所需设备和软件

表 3. 所需设备

项目
Binho Supernova 或 Binho Pulsar USB 主机适配器，用于 `timing` 命令
一个 CMIS 受管光模块，用于 `timing` 命令
`budget` 命令无需任何设备

```
pip install binhosupernova
python cmis_mci.py budget
```

一个主机适配器可以提供两种传输方式，而您得到哪一种由主机软件决定，且不会有任何提示。 从 4.5.0 起，Binho 适配器固件除了原有的 USB HID 路径外，还通过厂商自定义批量（bulk）端点承载传输。 批量传输需要主动启用：如果主机没有协商启用它，适配器会照常提供 HID 路径，不报错也不警告， 这正是该固件属于非破坏性版本的原因。本笔记及本系列其余笔记使用的 `binhosupernova` 仅支持 HID， 低于 1.5.0 的任何 CosmicSDK 也是如此。请向已打开的设备查询该传输方式是否可用，而不要根据您安装的 版本来推断，因为不支持它的 SDK 会给出旧的数字，并且在此过程中不报告任何异常。第 5.2 节对两者都 进行了测量，这一区别对结论的影响大于对数字的影响。

3 预算所依据的四个事实

3.1 通过本地有效载荷传输的固件块为 116 字节

通用数据块位于 Page 9Fh。其本地有效载荷（Local Payload）为 120 字节，其中 `Write Firmware Block LPL`（命令 `0103h`）用 4 字节表示块偏移量。因此每块剩下 **116 字节** 固件。

扩展有效载荷（Extended Payload）为 A0h 到 AFh 页，共 16 页、每页 128 字节，通过 `Write Firmware Block EPL`（命令 `0104h`）每块可传输 **2048 字节**。

3.2 一次 WRITE 最多携带 8 字节

5.2.2.2 节直接写明："A successful WRITE writes a sequence of up to eight given byte values"（一次成功的 WRITE 写入最多八个给定字节值的序列）。`FullPageReadSupported` 将 **读取** 的 `Nmax` 提高到 128，但对写入没有任何作用。

唯一的例外是紧随其后的注释："A WRITE with more than 8 bytes may be allowed when specified explicitly in chapter 7.8"（当第 7.8 章明确规定时，可以允许超过 8 字节的 WRITE）， 即通用数据块扩展有效载荷区域。

因此，一个 116 字节的本地有效载荷块需要 `ceil(116 / 8)` = **15 次独立的写事务**，每次都有自己的地址阶段。

3.3 随后模块停止响应

表 4. 等待期参数，CMIS Table 10-4 最大值

参数	最大值	适用于以下操作之后
tREAD	500 微秒	寄存器读取
tWRITE	10 ms	写入易失性存储器
tWRITENV	80 ms	写入非易失性存储器
tBPC	10 ms	存储体（bank）或页切换
tCDBF	4960 ms	前台 CDB 命令。默认值为 80 ms
tMgmtInit	2000 ms	复位后，直到进入 ModuleLowPwr

这些都是最大值。模块可以声明不超过这些值的任意数值，而第 5 节的内容就是查明特定模块实际需要多长时间。

3.4 几乎所有 CMIS 存储器都是易失性的

规范明确说明了这一点，因此 **10 ms 的 tWRITE 是常见情况**，而不是乐观情况。

等待期是**每块支付一次**，而不是每次事务支付一次。这就是全部的杠杆所在：对于 1 MiB 的映像，2048 字节的块需要支付 512 次，而 116 字节的块需要支付 9040 次。

4 预算

```
python cmis_mci.py budget --image-kib 1024
```

```
Firmware update budget, 1024 KiB image, tWRITE 10 ms per block
  path                            blocks   per block       bus    hold-off     total
  I2CMCI 400 kHz, LPL 0103h         9040       15 txn   36.61 s     90.40 s   127.01
s
  I3C SDR 12.5 Mb/s, LPL 0103h      9040       15 txn    1.17 s     90.40 s    91.57
s
  SPIMCI 1 MHz, LPL 0103h           9040       15 txn   15.19 s     90.40 s   105.59
s
  I2CMCI 400 kHz, EPL 0104h          512       16 txn   24.33 s      5.12 s    29.45
s
  I3C SDR 12.5 Mb/s, EPL 0104h       512       16 txn    0.78 s      5.12 s     5.90
s
  SPIMCI 1 MHz, EPL 0104h            512        1 txn    8.41 s      5.12 s    13.53
s
  SPIMCI 50 MHz, EPL 0104h           512        1 txn    0.17 s      5.12 s     5.29
s

  Bus clock alone, 400 kHz to 12.5 Mb/s, about 31 times the wire speed: 1.39 times
faster.
  Payload size alone, LPL to EPL at 400 kHz: 4.3 times faster.
  Both together: 21.5 times faster.
  SPIMCI at 50 MHz, 125 times the I2C wire speed and one transaction per block: 24.0
times faster.
```

请将前两行对照来看。线路速度提高 31 倍，节省 35 秒。在原来的低速时钟下，改变承载有效载荷的通用数据块命令，节省 97 秒。

4.1 三种管理接口，给出同一个答案

SPIMCI 行承载了整个系列旨在进行的比较，因为 SPIMCI 是 CMIS 实际规定的唯一另一种管理接口。Appendix B.3.7.1 将每个事务组帧为一个四字节控制字、`N` 个流控字节和最多 **2048 字节** 的 有效载荷，而 `00h:27.3-0` 处的 `MciSpeedConfiguration` 范围为 1 MHz 到 **50 MHz**。这是 I2C 时钟的 125 倍，也是分页接口所能进行的最大访问的 16 倍。

它带来 24.0 倍的提速，而 I2CMCI 为 21.5 倍。其中有两个事实值得单独指出。

在模块实际启动时的速度下，SPIMCI 在线路上比 I3C 慢。 退出 MgmtInit 时，`MciSpeedConfiguration` 的初始值始终为 1 MHz，而在 1 MHz 下，扩展有效载荷路径在线路上花费 8.41 s，I3C 则为 0.78 s。 时钟必须由主机主动提高：主机需读取模块的能力声明并写入该字段。

无论速度如何，SPIMCI 改变的是事务数。 在 I2CMCI 和 I3C 上（两者都将一次访问限制为一页），一个扩展有效载荷块需要 16 次事务，而在 SPIMCI 上只需**一次**。这一列从不出现在带宽数据中， 而第 5 节表明，它正是主机能感受到的那一列：每次事务都要经过主机往返一次，而在测试台上，无论总线在做什么，这一往返大约都是两毫秒。

因此，这三种接口在线路上相差 125 倍，在总时间上相差 24 倍，但在时间花在哪里这一点上完全一致。 固件更新不是带宽问题。

表 5. 预算模型包含和不包含的内容

包含	不包含
每块事务数，由访问长度限制得出	主机 USB 延迟，它不是总线属性
每次事务的总线时间，按 4 个组帧字节加有效载荷、每字节 9 位计算	tCDBF，因模块和命令而异
每块一次 tWRITE	Run Image 和 Commit Image 步骤
	访问被拒绝后的重试

> **总线时间列是对线路占用的估计，不是实测吞吐量。** 它有意做得很粗略，因为它唯一的作用是表明 它与等待期列相比很小。更精细的模型不会改变结论，而实测值所表征的主机 USB 路径与总线一样多。

5 测量真实模块

```
python cmis_mci.py timing
```

该命令不写入任何固件。它为构成更新的各访问阶段计时：单字节读取、达到写入限制的八字节读取、模块声明支持时的整页读取、双向页切换，以及切换到通用数据块页，并将每项与其 Table 10-4 上限并列输出。

```
$ python cmis_mci.py timing
phase                                 median        max    CMIS max
single byte read                    2.058 ms    3.400 ms      0.5 ms
8 byte read, the WRITE limit        1.995 ms    2.064 ms           -
page change 00h to 01h to 00h       7.988 ms    9.019 ms     20.0 ms
select CDB page 9Fh                 4.043 ms    5.913 ms     10.0 ms

One 116 byte LPL firmware block is 15 write accesses, about 29.9 ms of bus time,
against a tWRITE hold-off of up to 10 ms that the module pays once per block.
```

应将其理解为对主机的测量，而不是对目标器件的测量。 每一行大约都是每次 USB 往返两毫秒： 一次访问约 2 ms，CDB 页选择的两次寄存器写入约 4 ms，双向页切换的四次写入约 8 ms。单字节读取 是 tREAD 的四倍，而八字节读取并不比单字节读取慢，只有在计时对象不是线路上的字节时才可能如此。

此次运行中的目标器件是处于 I2C 目标器件模式的适配器，它会立即应答，从不让总线等待。因此这些 数字显示的是在模块尚未贡献任何时间之前，主机路径施加的下限，在此配置下该下限约为每次访问 2 ms。 第 7.2 节给出结论。

结果中需要关注的内容：

中位数远低于规范最大值，意味着模块比规范允许的更快，而每次写入后都等待最大值的主机，会让大部分 更新时间白白浪费。这是该测量所能告诉您的最有价值的信息。

页切换的最大值接近规范上限，意味着模块有时确实需要 tBPC。假设中位数的主机会间歇性失败， 这是最难调试的故障模式。

单字节读取高于 tREAD，是主机侧或适配器侧的开销，而不是模块的等待期。tREAD 是对 I2C 时钟延展 （clock stretching）的抽象，时钟延展在总线捕获中是可见的，而主机侧计时器无法将其与其他因素区分开来。

5.1 针对 I3C CMIS 目标器件的相同测量

上述运行是通过 I2CMCI 针对处于目标器件模式的适配器进行的。通过 I3C 针对 AN0009 的 CMIS 目标器件固件（它实现的是寄存器模型，而不是平面存储器）重复测量，没有任何重要变化：

```
$ python cmis_mci.py timing --transport i3c --address 0x09 --push-pull
PUSH_PULL_2_5_MHZ_25_DC
phase                                  median        max    CMIS max
single byte read                     2.059 ms    3.008 ms      0.5 ms
8 byte read, the WRITE limit         2.062 ms    2.760 ms           -
page change 00h to 01h to 00h        7.997 ms    9.170 ms     20.0 ms
select CDB page 9Fh                  3.998 ms    4.120 ms     10.0 ms

One 116 byte LPL firmware block is 15 write accesses, about 30.9 ms of bus time,
against a tWRITE hold-off of up to 10 ms that the module pays once per block.
```

快 31 倍的总线，加上一个由固件而非平面存储区应答的目标器件，所有行都保持不变。这与预算的预测结果 相同，只是从另一个方向得出。

5.2 下限属于主机软件栈，而当前软件栈无法摆脱它

Supernova 固件 4.5.0 将适配器传输移到厂商自定义批量端点上，提高了传输上限并重构了 USB 路径。 通过本笔记提供的实用程序测量，它没有带来任何变化。

表 6. 两个适配器固件修订版上的相同四个阶段，三次运行的中位数

阶段	4.4.0	4.5.0
单字节读取	2.006 ms	2.036 ms
8 字节读取，即 WRITE 限制	2.003 ms	2.036 ms
页切换 00h 到 01h 再到 00h	7.993 ms	8.061 ms
选择 CDB 页 9Fh	3.987 ms	4.035 ms

该板从 4.4.0 升到 4.5.0，再回到 4.4.0，然后再次升级，并在每种状态下都进行了测量，因为跨一次重新 烧录的单次比较无法区分固件变化和其他发生变化的因素。

> **上表中的两者都是兼容路径。** 较新的传输方式需要主机主动启用：如果主机没有协商启用它，适配器会照常提供旧路径，这正是该固件属于非破坏性版本的原因。本笔记及整个系列所运行的 SDK `binhosupernova` 4.2.0 通过 USB HID 传输，完全不含厂商自定义批量端点的代码。因此这组数据说明 兼容路径没有退化，但单凭它无法说明 4.5.0 新增的传输方式的任何情况。

通过确实会协商启用它的主机软件栈重复相同的四个阶段，就能回答上表无法回答的问题。各阶段没有变化。

表 7. 兼容传输方式和批量传输方式上的相同四个阶段，均在固件 4.5.0 上

阶段	USB HID	批量传输方式
单字节读取	2.036 ms	2.009 ms
8 字节读取，即 WRITE 限制	2.036 ms	2.006 ms
页切换 00h 到 01h 再到 00h	8.061 ms	8.019 ms
选择 CDB 页 9Fh	4.035 ms	3.993 ms

右侧一列是在传输方式报告自身可用并声明 32768 字节上限的情况下测得的，因此它不是换了名字的第二次 兼容路径运行。该传输方式从 CosmicSDK 1.5.0 起提供；较旧的 SDK 会给出左侧一列的结果，同时不报告 任何异常，这正是本次测量必须排除而不能假定的故障。检查方式是向已打开的设备进行能力查询，而不是看 版本号。不过，这并不是关于该传输方式的负面结果，因为在同一会话中，该传输方式完全实现了其设计目的：

表 8. 从 SPI 输出字节，同一适配器和会话，按传输大小

传输大小	请求路径	批量路径
8 字节	2.05 ms	2.00 ms
1024 字节	35.0 ms	5.0 ms
32768 字节	不可用	114.8 ms，285 kB/s

更宽的传输方式是有效载荷的杠杆，而不是延迟的杠杆。 在一千字节时，它快七倍。在八字节时速度相同， 因为八字节传输的开销就是一次往返，而管道变宽并不会使往返变短。

这正是第 4 节在主机侧的重述，也是这两节结论一致的原因。`5.2.2.2` 将一次 WRITE 限制为八字节、 一次访问限制为一页，因此 **在分页管理接口上，CMIS 从不发出足够大的传输，让这些改进派上用场。** 确实会发出这种传输的接口是 SPIMCI，可达 2048 字节，这对应第 4 节中将每块十六次事务压缩为一次的 那一行。本测试台上没有 SPIMCI 目标器件，因此该行仍是算术推导。

5.3 两毫秒从何而来

两个独立的主机软件栈在一个数值上三位有效数字都一致，这不是巧合，原因在于适配器的 USB 描述符， 而不在于其中任何一个软件栈。

表 9. 适配器的端点，取自其配置描述符

接口	端点	传输类型	最大包长	间隔
人机接口设备（Human Interface Device）	0x01 OUT, 0x81 IN	中断	64 字节	1 ms
厂商自定义	0x02 OUT, 0x82 IN	批量	64 字节	无

命令平面是那对中断端点，而中断端点是按计划轮询的。 这里的间隔是一毫秒，因此请求不能在下一帧 之前发出，其应答也不能在再下一帧之前到达。这解释了两毫秒中的大部分，而且这并不是任何 SDK 在做的 工作：它是总线允许传输发生的时刻。剩下的是适配器固件的周转时间和主机软件栈，本测量没有将两者分开。

批量端点对承载有效载荷，而不是命令。这就是前面结果的全部原因：加宽有效载荷路径后，命令往返保持 原样，因此一个八字节访问（只有一条命令和一个应答，别无其他）无法变得更快。

同一模型还能预测另一列，这才是值得做的验证。请求路径上的 1024 字节传输是在命令本身中携带的， 每次 64 字节，因此它是十六帧有效载荷加上命令往返：约 34 ms，实测为 35.0 ms。在批量路径上， 有效载荷完全脱离调度，只剩下命令往返：约 2 ms 加上传输时间，实测为 5.0 ms。

因此这个下限有了名字，它也告诉主机该怎么应对。 能改变它的是命令的数量，以及同时有多少条命令 在等待。两者都与命令所经过的管道大小无关。

5.4 下限的一半是应答，主机可以隐藏它

如果一条命令和它的应答各占一帧，那么等到每个应答返回才发送下一条命令的主机要为两者付出代价，而让 命令保持在途（in flight）的主机应该只为前者付出代价。这是可以验证的。通过非阻塞接口发出同样的 1 字节读取 200 次，一次等待每个应答，一次不等待：

表 10. 同样的 200 次读取，等待每个应答与让它们保持在途的对比

	总计	每条命令
逐条执行	400.0 ms	2.000 ms
全部在途	200.9 ms	1.005 ms

流水线化使其减半，被消除的那一半就是应答。 剩下的是每帧一条命令，这是出站轮询间隔，主机除了 少发命令之外无法低于这一值。200 条命令全部得到应答，因此这不是对命令被丢弃的测量。

对需要它的主机而言有两个推论。节省是真实的但有限：它将一个扩展有效载荷块的主机时间开销从约 32 ms 降到约 16 ms，而 tWRITE 为 10 ms，因此主机不再主导该块，但也没有变得可以忽略。而且只有在各访问 彼此不相互依赖时才能使用，而在一个固件块内部它们确实不相互依赖，因为它们是写入同一扩展有效载荷 区域的连续写操作。

需要明确指出的是，**本笔记提供的实用程序并不这样做。** 它使用阻塞接口，一次一条命令，这就是第 5 节 中每个数字都是两毫秒那一列的原因。上面的测量是针对非阻塞接口的单独测试工具，其目的是评估这一选项 的价值，而不是描述该实用程序。

这一点值得与第 4 节放在一起看。一个扩展有效载荷块是十六次访问，约 32 ms 的主机时间，而 tWRITE 为 10 ms，因此在本测试台上，主导该块的是主机而不是模块。流水线化将其降到约 16 ms，

而 **SPIMCI 将其 变为一次事务而不是十六次**，这是本笔记中能影响第 5 节所测量项的最大变化。一个假设自身往返开销 可以忽略的主机在这里是错误的，而在分页管理接口上，它最多只能将这些开销减半。

这就是第 4 节和第 5 节从相反方向得出同一结论的原因。

6 如何利用这一发现

对于主机软件。 在下载开始时查询一次 `Firmware Management Features`（CDB 命令 `0041h`），并在模块支持时优先使用 `Write Firmware Block EPL`（`0104h`）而不是 `0103h`。不要对任何一方作出假设。 然后在一个块内发出各访问时不要等待每个应答：第 5.4 节测得这样做的主机开销减半，之所以可行是因为 一个块内的访问彼此不相互依赖，而它需要的是非阻塞接口，而不是更快的接口。

对于模块固件。 声明您能够遵守的等待期时间。tWRITE 是最大值，不是目标值，而第 5 节中的测量就是 主机发现您留下多少未用余量的方法。

对于制定新 MCI 绑定规范的任何人。 SPIMCI 已经能在一次事务中承载最多 2048 字节，第 4.1 节对此 做了估算，而 I3C 通过 SETMWL、SETMRL 和 GETMXDS 原生协商同样的能力。继承了 I2CMCI 8 字节写入 限制的绑定，也会随之继承这一整个问题，而且无论运行在什么时钟下都会继承。

> **单独为 Run Image 步骤编制预算。** CDB 命令 `0109h` 所需时间与一次完整的模块重启相同，"mainly governed by the tMgmtInit timing parameter"（主要由 tMgmtInit 时序参数决定），因此应为它 预留大约两秒，而不是一个 CDB 超时时间。

7 测量结果

7.1 测试配置

表 11. 测试配置，第 5 节的 I2CMCI 运行

项目	值
控制器	Binho Supernova，序列号 `1912DEB5…79FA`，固件 4.4.0，硬件修订版 C
目标器件	Binho Supernova，序列号 `ED7BF8F3…F7BB`，固件 4.4.0，配置为地址 0x50 的 I2C 目标器件
目标器件内容	CMIS 5.3 映像，Appendix C 中的 400G QSFP-DD 示例
SDK	`binhosupernova` 4.2.0，Python 3.10.12
主机	Ubuntu 22.04，x86-64
总线	3.3 V，400 kHz
重复次数	每个阶段 20 次，报告中位数和最大值
日期	2026 年 9 月 1 日

表 12. 测试配置，第 5.1 和 5.2 节的 I3C 运行

项目	值
控制器	Binho Supernova，序列号 `4F970418…1111`，固件 4.4.0 和 4.5.0
主机传输方式	通过 USB HID 的 `binhosupernova` 4.2.0，以及通过厂商自定义批量端点的 CosmicSDK 1.5.0，后者经能力查询确认可用，并声明 32768 字节上限
目标器件	运行 AN0009 CMIS 目标器件固件的 NXP FRDM-MCXN947，动态地址 0x09
目标器件内容	CMIS 5.3 映像，Appendix C 中的 400G QSFP-DD 示例
SDK	`binhosupernova` 4.2.0，Python 3.14
主机	Arch Linux，x86-64
总线	3.3 V，I3C SDR，推挽 2.5 Mbit/s、25% 占空比，开漏 1 Mbit/s
重复次数	每个阶段 20 次，报告中位数和最大值，每个固件运行三次
第 5.4 节	针对非阻塞接口的单独测试工具，每组 200 次读取，运行三次
日期	2026 年 9 月 3 日

这两种配置使用不同的测试台、不同的主机和不同的 Python 版本。本笔记中没有任何地方将它们相互比较， 并且只有第二种配置用于第 5.2 节中的固件比较，在该比较中主机和目标器件保持不变，适配

器固件是唯一 变化的因素。

7.2 结果

推导结果。 第 4 节中的预算可用所提供的实用程序复现，并且仅依据第 3 节中引用的规范条款。它是算术推导，本文档不将其表述为其他任何东西。

实测结果，而它测量的是主机。 第 5 节的运行确定了在此主机、适配器和 SDK 上，每次寄存器访问约 2 ms 的下限，而所用目标器件本身不施加任何等待期。这个数字属于 USB 路径，而不属于总线，也不属于 模块：单字节读取超过 tREAD 四倍，而八字节读取并不比单字节读取慢，只有在线路时间不是主导项时， 这两点才可能成立。

这就是 **本文档中任何地方都没有出现端到端吞吐量数据** 的原因。以这种方式测得的数值表征的是主机， 若将其作为总线或模块的属性引用，就会造成一种日后难以察觉的错误。

不过，它反而强化了本笔记自身的论点，而不是削弱它。第 4 节论证固件更新的时间不是花在线路上。 测量进一步表明，时间也不是花在主机往返开销上。两者与每块的等待期相比都很小，并且当块变大时，两者 都会因同一原因而缩小：**更少的访问和更少的块。** 每次访问 2 ms 的下限，使每个本地有效载荷块开销 30 ms、每个扩展有效载荷块开销 32 ms，但扩展有效载荷块承载的固件多 17 倍。

在更新的适配器固件和更新的传输方式上再次测量，结果没有变化。 Supernova 固件 4.5.0 使全部四个 阶段保持在 4.4.0 时的水平，该固件新增的批量传输方式也是如此，测量时该传输方式报告自身可用。在同一 会话中，该传输方式在一千字节的传输上快七倍，因此这是一个有界的结果，而不是配置失败：更宽的传输方式 是有效载荷的杠杆而不是延迟的杠杆，而一个八字节访问无论哪种方式都只需一次往返。

这是从另一端重述第 4 节。`5.2.2.2` 将一次 WRITE 限制为八字节、一次访问限制为一页，因此在分页管理 接口上，CMIS 从不发出足够大的传输，让更宽的管道派上用场。能影响此表的杠杆是往返次数（即每块事务数 一列所统计的、也是 SPIMCI 所改变的），以及同时有多少往返处于未完成状态。

其中第二个杠杆值一半，而且现在就可以用。 第 5.4 节在不等待每个应答的情况下发出同一访问 200 次， 测得 1.005 ms 对比 2.000 ms，每条命令都得到了应答。剩下的是每个 USB 帧一条命令。因此，写入固件块 的主机可以在块内（各访问相互独立之处）不等待应答，从而将自身的开销减半，而在此之后，在分页接口上 它就没有其他杠杆可用了。

这也是第 4 节中每块事务数这一列值得关注的原因。SPIMCI 在单次事务中承载一个扩展有效载荷块，而分页 接口需要十六次，按每次事务约两毫秒的主机开销计算，这就是在总线做任何事情之前，每块 2 ms 对比 32 ms。

在硬件上，针对模块。 此处没有任何结果。截至修订版 1.0 尚未完成的工作：针对真实模块运行 `timing` 命令，这是确定模块实际需要哪些等待期的唯一方法；通过本地有效载荷和扩展有效载荷两条路径完成一次完整的 固件下载，这是能够支持端到端数据的唯一方法；以及在 SPIMCI 上进行任何测量，本测试台上不存在 SPIMCI 目标器件。第 4 节中的每个 SPIMCI 数字都是根据 Appendix B.3 的算术推导，并已如此标注。

在完成上述运行之前，本笔记所支持的结论是关于 **时间被允许花在哪里**，以及在模块贡献任何时间之前 主机的开销是多少。

8 故障排除

表 13. 故障排除

现象	可能原因和处理措施
`timing` 报告单字节读取远高于 tREAD	主机或适配器开销，而非模块所致。请与总线捕获结果进行比较
每个阶段都接近其上限	模块可能确实很慢，或者有另一个控制器在争用总线
输出中没有 94 字节读取	模块未声明支持整页读取，因此 `Nmax` 保持为 8
各次运行之间的中位数差异很大	有其他程序在使用 USB 路径。关闭其他主机工具后重试
在更新的适配器固件上数字看起来相同	这是预期结果，第 5.2 和 5.3 节对此作了解释。同时请确认您使用的是哪种传输方式：如果 SDK 没有协商启用批量传输，适配器会提供 HID 路径而不报错。请向 SDK 查询，不要根据版本号推断
每次访问约 2 ms，无论如何更改都没有改善	其中大部分是 USB 调度：命令平面是以 1 ms 轮询的中断端点，因此一条命令及其应答跨越两帧。第 5.3 节给出了描述符。应减少命令的数量，而不是命令的大小
页切换超过 tBPC 的两倍	模块所需时间超过规范允许的范围。在归咎于主机之前，请先捕获总线

9 缩略语

表 14. 缩略语

术语	含义
CDB	通用数据块（Common Data Block）
CMIS	通用管理接口规范（Common Management Interface Specification）
EPL	扩展有效载荷（Extended Payload），CDB 页 A0h 到 AFh
LPL	本地有效载荷（Local Payload），位于 CDB 页 9Fh
MCI	管理通信接口（Management Communication Interface）
Nmax	一次寄存器访问中的最大字节数
OIF	光互联论坛（Optical Internetworking Forum）
SDR	单倍数据速率（Single data rate）

10 关于文档中源代码的说明

本文档中展示的以及随附归档中提供的示例代码仅供评估和参考。这些代码依据 MIT 许可证发布，许可证文本 包含在归档中。

表 15. 资源归档的内容

文件	说明
`AN0011.md`	本文档的 Markdown 源文件
`assets/cmis_mci.py`	参考实用程序，由 AN0008 至 AN0011 共用
`assets/LICENSE`	涵盖示例代码的 MIT 许可证

11 修订历史

表 16. 修订历史

修订版	日期	说明
0.1	2026 年 9 月 1 日	供内部评审的草稿。预算由规范推导；未进行测量。
0.2	2026 年 9 月 1 日	针对 Supernova I2C 目标器件测试夹具运行 `timing`。粘贴了输出，并将其报告为对主机路径而非模块的测量。
0.3	2026 年 9 月 2 日	由 AN0010 重新编号。测量结果和预算均无变化。
0.4	2026 年 9 月 3 日	预算扩展到 SPIMCI，包括其初始的 1 MHz 和 `MciSpeedConfiguration` 最高的 50 MHz，并指出每块事务数一列是主机能感受到的那一列。第 5 节新增针对 AN0009 CMIS 目标器件通过 I3C 进行的相同测量，以及 Supernova 固件 4.4.0 与 4.5.0 在两种主机传输方式上的比较，两者都没有改变下限；并在同一会话中进行了 SPI 传输大小扫描，表明更宽的传输方式在一千字节时快七倍、在八字节时相同。在该传输方式加入 CosmicSDK 1.5.0 后，针对其重复了两种传输方式的运行，结果没有变化。新增第 5.3 节，将两毫秒下限追溯到命令平面所使用的 1 ms 中断端点，并用传输大小扫描验证了该模型。实用程序版本为 1.1。
0.5	2026 年 9 月 3 日	新增第 5.4 节，评估让命令保持在途的价值，测得每条命令的开销减半且每条命令都得到应答；第 6 节现在建议主机在块内这样做。剩下的下限是每帧一条命令。
1.0	2026 年 11 月 6 日	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. **CMIS** 和 **OIF** 是 Optical Internetworking Forum 的商标或注册商标。 **I3C** 和 **MIPI** 是 MIPI Alliance, Inc. 的商标或注册商标。