binho

アプリケーションノート

Binho Supernova による NVMe-MI 向けシングルポート I2C/SMBus/I3C 動作

AN0014

文書情報

キーワード	AN0014, Supernova, I3C, I2C, SMBus, MCTP, クロックストレッチ, レガシー I2C, tTIMEOUT, NVMe-MI, ARP, ホットジョイン（Hot-Join）, PIC18F16Q20, pyCosmicSDK
概要	多くの SMBus デバイスおよび MCTP デバイスは、応答を準備する間 SCL を Low に保持します。Supernova の I3C エンジンはこれを待つことができないため、I3C ポート上ではこのようなターゲットはビットを取りこぼして いました。アダプタファームウェア 4.6.0 と pyCosmicSDK 1.6.0 は、I3C ポート上のレガシー I2C 転送が SMBus の 35 ms タイムアウトまでターゲットを待つモードを追加しました。本ノートでは、このモードを説明し、 このモードでクロックストレッチを行うターゲットを読み書きするスクリプトと、測定に使用したリファレンス ターゲットのファームウェアを提供します。測定結果は、ターゲットが 0 ms から 30 ms のストレッチを行う 条件で、100 kHz と 400 kHz における 3 回の実行で 3600 回中 3600 回の読み出しが正しく、モードの上限で ある約 750 kHz まで、ストレッチを伴うすべての読み出しが正しいというものでした。スクリプトは、PCI-SIG のアウトオブバンド管理仕様にある SMBus から I3C Basic への検出フローも、アダプタで可能な範囲まで実行します。

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

改訂 1.0
6 NOVEMBER 2026

1 はじめに

I2C ターゲットは、アドレスを ACK した後、応答までの時間を稼ぐために SCL を Low に保持することがあります。 コントローラは SCL が解放されるまで待たなければなりません。I2C 仕様ではこれをクロックストレッチと 呼び、SMBus では最大 35 ms（tTIMEOUT）まで許容されています。SMBus デバイスおよび MCTP デバイスは これを日常的に使用します。値を取得中の管理コントローラは、値が準備できるまでクロックを保持します。

Supernova の I3C ポートは、I3C デバイスと同じバス上のレガシー I2C ターゲットにアクセスできます。 しかし、その I3C エンジンはターゲットが SCL を Low に保持している間もクロックを出し続けるため、 クロックストレッチを行うターゲットはビットを取りこぼし、転送は誤ったデータを返すか、ACK が得られません。 アダプタファームウェア 4.6.0 と pyCosmicSDK 1.6.0 は、I3C ポート上のレガシー I2C 転送についてこれを 解決するモードを追加しました。

1.1 適用範囲

本書では次の内容を扱います。

- クロックストレッチモードが変更する内容と、モードがオンの間に拒否する内容
- モードが動作する SCL レートと、それが要求値にどれだけ近いか
- このモードを介してクロックストレッチを行うターゲットを読み書きするスクリプト
- 設定可能な時間だけクロックストレッチを行うリファレンスターゲットとそのファームウェア
- そのターゲットに対する測定結果
- NVMe-MI で使用される SMBus から I3C Basic への検出フローと、そのうちアダプタが実行するステップ

ARP Get UDID の PEC チェックを超える SMBus フレーミング（ブロックリードとブロックライト、クイック コマンド）、MCTP メッセージや NVMe-MI メッセージ、および Supernova の独立した I2C ポートは扱いません。

1.2 対象デバイス

表 1. 対象デバイス

役割	品名	識別情報	根拠
コントローラ	Binho Supernova	ハードウェアリビジョン C、ファームウェア 4.6.1、I3C HV ポート	測定済み、本書
コントローラ	Binho Supernova	ファームウェア 4.5.1 以前	`FW_UNSUPPORTED_COMMAND` でモードを拒否
コントローラ	Binho Pulsar	すべて	対象外：Pulsar には I3C ポートがありません
リファレンスターゲット	Microchip PIC18F16Q20 Curiosity Nano	ファームウェア `stretch_target_pic18q20.c`	測定済み、本書

1.3 関連資料

表 2. 関連資料

資料	タイトル
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	Binho Supernova および Pulsar による I2C SCL デューティ比と正確なバス周波数の設定

2 システム概要

モードがオフのとき、I3C ポート上の `legacy_i2c_read` と `legacy_i2c_write` は I3C エンジンによって 実行され、I3C エンジンは独自のスケジュールで SCL を駆動します。モードがオンのとき、アダプタの ファームウェアはこれらの転送をソフトウェアでクロッキングし、SCL を解放する各フェーズの後に SCL を 確認します。ターゲットが SCL を Low に保持していれば、ファームウェアは待機します。

モードの使い方は、次の 4 つの事実によって決まります。

- **これは転送ごとのオプションではなく、モードです。** `enable_i2c_clock_stretching(frequency_hz)` は 以降のすべてのレガシー I2C 転送に対してモードをオンにし、`disable_i2c_clock_stretching()` はオフに します。`initialize` と `configure` もモードをオフにします。
- **I3C トラフィックを排除します。** モードがオンの間、SDR、HDR、CCC の各呼び出し、`init_bus` および `reset_bus` は、ステータス `FW_I3C_I2C_CLOCK_STRETCHING_ACTIVE` の `I3cError` を送出します。I3C デバイスと通信する前にモードをオフにしてください。
- **SMBus タイムアウトを適用します。** 35 ms を超えるストレッチは `FW_BUS_TIMEOUT` で転送を終了し、 バスは解放されます。
- **レートはソフトウェアクロックで設定されます。** この呼び出しは 10 kHz から 1 MHz を受け付け、計算した レートを返します。100 kHz では 97975 Hz、400 kHz では 391644 Hz で、実際の波形との差は 1 % 以内です。 400 kHz を超えると、返されるレートと実際の波形の差は最大 7 % になり、600 kHz の要求は 615 kHz で動作 します（9.2 節）。クロックの上限は約 750 kHz のため、800 kHz と 1 MHz の要求はどちらもその値で動作 します。

3 ハードウェアのセットアップ

3.1 必要な機器

表 3. 必要な機器

品目
Binho Supernova、ファームウェア 4.6.0 以降
USB-C ケーブル
SCL のクロックストレッチを行う I2C または SMBus ターゲット、あるいは 3.4 節のリファレンスターゲット
ターゲットボードにプルアップがない場合、SCL と SDA のプルアップ抵抗

3.2 接続

表 4. Supernova の I3C HV ポートとターゲットの接続

Supernova I3C HV ポート	ターゲット	要否
SCL	SCL	必須
SDA	SDA	必須
GND	GND	必須
VTARG	ターゲットの I/O 電源	任意、ターゲットをアダプタから給電する場合

3.3 バスに関する考慮事項

I3C ポートには SCL のプルアップがありません。I2C バスでは両方のラインにプルアップが必要です。私たちの ベンチでは、1 kOhm に設定してアイドル状態にした Binho Pulsar の I2C ポートからプルアップを得ました。 バス電圧への 1 kOhm から 2.2 kOhm の抵抗のペアでも同じ役割を果たします。SCL にプルアップがないと、 ファームウェアが立ち上がりエッジごとに待機するため、報告されるレートは大きく低下します。

> **注意：** I3C HV ポートは `set_voltage` で電圧を選択します（1200 mV から 3300 mV）。3.3 V トレラントで ないターゲットを接続する前に、ターゲットの I/O 電圧に設定してください。
>
> **注意：** バス上のコントローラは 1 つだけにしてください。2 台目のアダプタがプルアップを供給する場合、 そのアダプタから転送を開始しないでください。

3.4 リファレンスターゲット

このモードを測定するには、ストレッチ時間を設定できるターゲットが必要でした。アセットアーカイブに含まれる `stretch_target_pic18q20.c` は、PIC18F16Q20 Curiosity Nano をそのようなターゲットにします。その I2C ペリフェラルは、アドレス一致のたびにプログラム可能な時間だけ SCL を保持します。これはビジー状態の SMBus デバイスの振る舞いと同じです。

表 5. リファレンスターゲットのプロトコル、アドレス 0x2A

転送	効果
`0xF0`、下位バイト、上位バイトを書き込み	以降のアドレス一致のたびに、その値のマイクロ秒数だけストレッチ。0 でストレッチをオフ
読み出し	すべてのバイトで `0xA5` を返す
その他の書き込み	受け付けて破棄

SCL と SDA は RB6 と RB5 です。MPLAB XC8 でビルドし、MPLAB X または Curiosity Nano のドラッグアンド ドロップ書き込み機能で書き込んでください。

4 ソフトウェアのセットアップ

pyCosmicSDK（最新リリース）をインストールします。

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

スクリプト `smbus_stretch.py` はアセットアーカイブに含まれています。他に必要なものはありません。

5 クロックストレッチモード

5.1 使い方

呼び出しのみを示します。

```
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()
```

レジスタ読み出し（SMBus の Read Byte または Read Word のパターン）では、レジスタを `subaddress` として 渡します：`i3c.legacy_i2c_read(0x0B, 2, subaddress=bytes([0x09]))`。

5.2 ステータスコード

表 6. モード固有のステータスコード

ステータス	送出元	意味
`FW_UNSUPPORTED_COMMAND`	enable、disable	ファームウェア 4.5.1 以前
`FW_INVALID_PARAMETER`	enable	レートが 10 kHz から 1 MHz の範囲外
`FW_INTERFACE_NOT_INITIALIZED`	enable、disable	I3C コントローラが起動されていない
`FW_I3C_I2C_CLOCK_STRETCHING_ACTIVE`	SDR、HDR、CCC、`init_bus`、`reset_bus`	モードがオン。先にオフにする
`FW_BUS_TIMEOUT`	レガシーの読み出しまたは書き込み	ターゲットが SCL を 35 ms より長く保持した

5.3 SMBus タイムアウト

ファームウェア検証中、リファレンスターゲットを 40 ms ストレッチに設定した条件で、100 回の読み出しは 50 回の `FW_BUS_TIMEOUT` と 50 回の `FW_I3C_NACK_ADDRESS` を返し、最長は 46 ms でした。アダプタは 応答可能な状態を保ちました。アドレス NACK はターゲットに起因します。ターゲットは 40 ms をカウントして いる間に次のアドレスを取りこぼします。

6 手順

1. 3.2 節のとおりにターゲットとプルアップを接続します。ターゲットに電源を投入します。
2. ターゲットから読み出します。1 行目にはファームウェアと有効なレートが表示されます。

```
$ 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. ターゲットに書き込みます。

```
$ 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. リファレンスターゲットでは、ストレッチ時間をスイープします。各時間について、スクリプトはターゲットの ストレッチを設定し、16 バイトの読み出しを 100 回行い、すべてのバイトを確認します。

```
$ 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. 400 kHz で繰り返します。

```
$ 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
```

1 回の読み出しあたりの時間の中央値はストレッチとともに、ほぼストレッチ分だけ増加します。つまり アダプタは毎回待機しました。スクリプトは終了前にターゲットのストレッチを 0 に戻します。

7 SMBus から I3C Basic への検出フロー

SMBus から I3C Basic へ移行する NVMe-MI デバイスは 1 組のピンを共有するため、ホストはバスを変更する 前に、各デバイスがどちらのプロトコルを使用するかを調べる必要があります。ホストが従うフローは、 PCI-SIG Engineering Change Notice の Chapter 12, Architectural Out-of-Band Management で定義されて います。`smbus_stretch.py` の `handoff` コマンドは、これを Supernova の I3C ポート上で実行します。 表 7 に各ステップと、それぞれを誰が実行するかを示します。

表 7. 検出フローのステップ

ステップ	フロー	実行者
1	SMBus 電圧で 7Eh に DISHJ 付きの DISEC をブロードキャスト。ACK であれば I3C Basic デバイスが存在する	スクリプト。オープンドレインは 7Eh ヘッダのみで、CCC バイトとデータは I3C レートのプッシュプルです。NACK は `FW_I3C_NACK_ADDRESS` として報告されます
2	スタティックアドレスおよび 61h（ARP Get UDID）での SMBus デバイス検出	スクリプト。クロックストレッチモードをオンにして実行します。スクリプトは PEC を確認します
3	SMBus 専用デバイスが見つかった場合：SMBus インタフェースをリセットし、SMBus モードにとどまる	スクリプト：`hold_scl_low` が T2wrst（`--t2wrst-us`）の間 SCL を Low に保持します。SMRST# を持つデバイスは GPIO からリセットすることもできます
4	再度 DISHJ 付きの DISEC、その後 SMBus プルアップをオフ、I3C Basic プルアップをオン	DISEC はスクリプト。アダプタの SDA プルアップは SMBus フェーズ用に `set_sda_pull_up`（`--sda-pull-up`）で設定し、その後は I3C エンジンに従います。アダプタには SCL のプルアップがないため、外部プルアップは手動で切り替えます
5	Tsmb2i3c 待機し、I3C Basic 電圧に移行し、ENHJ 付きの ENEC をブロードキャスト	スクリプト。`--t-smb2i3c-ms` と `--i3c-voltage-mv` は仕様とデバイスから決まります
6	デバイスがホットジョイン（Hot-Join）する	ファームウェア：ダイナミックアドレスを自ら割り当て、スクリプトがそれを表示します

クロックストレッチモードはすべての CCC を拒否するため（5.2 節）、スクリプトはステップ 2 の間だけモードを オンにし、ステップ 4 の前に再びオフにします。

フローを Saleae アナライザで確認するには、`capture_flow.py` を使用します。これは Logic 2 の自動化サーバ を介してキャプチャを開始し、`--` に続く引数で `handoff` を実行し、Logic 2 の I2C アナライザと、各ステップ を独自の行に表示する `flow_labels` 拡張機能とともにキャプチャを保存します。`--step-delay-s` はステップの 間隔を空け、タイムライン上で各ステップを見つけられるようにします。

> **注記：** 実行したのは SMBus 専用の分岐のみです。9.1 節のベンチにおいてファームウェア 4.6.1 で 6 回中 6 回同一の結果となり、うち 1 回はアダプタのリセット後でした。ステップ 1 は NACK となり、ステップ 2 は 0x2A のリファレンスターゲットを検出し ARP デバイスは検出せず、SMBus リセットは 35000 us を要求した ときに SCL を 35000 us から 35777 us の間 Low に保持しました。ステップ 3 から 6 には 7Eh を ACK しホットジョインするデバイスが必要で、ARP と PEC の経路には ARP 対応デバイスが必要です。 どちらもベンチにはありませんでした。PEC ルーチンは CRC-8/SMBUS のチェック値に対してのみ確認されています。

```
$ 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 リファレンス

表 8. smbus_stretch.py のオプション

オプション	デフォルト	意味
`--serial`	任意	Supernova のシリアル番号
`--voltage-mv`	3300	I3C ポート電圧、HV コネクタでは 1200 から 3300
`--khz`	100	SCL レート、10 から 1000。最大で約 750 kHz で動作

表 9. smbus_stretch.py のコマンド

コマンド	引数	動作
`read`	アドレス、長さ、オプションの `--register`	読み出し。オプションで 1 バイトのレジスタを書き込んだ後に読み出す
`write`	アドレス、バイト列	バイト列を書き込む
`sweep`	`--address`、`--count`、`--length`	リファレンスターゲット専用：ストレッチ 0、50、500、2000、10000、30000 us で、それぞれ `--count` 回読み出す
`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`	7 節の検出フロー

すべてのコマンドは転送の前にモードを有効にし、終了前に無効にするため、その後 I3C ポートを I3C トラフィックに使用できます。

9 測定結果

9.1 テスト構成

表 10. テスト構成

項目	値
アダプタ	Binho Supernova、ハードウェアリビジョン C、ファームウェア 4.6.1（`FW-4.6.1`）。9.2 節のソーク試験、SMBus リセット、SDA プルアップの測定は、同じ機能の 4.6.0 プレリリースビルドによるものです
ホストソフトウェア	PyPI の pyCosmicSDK 1.6.0、Linux
バス	I3C HV ポート、3.3 V、アイドル状態の Binho Pulsar（ファームウェア 4.6.1）から SCL と SDA に 1 kOhm のプルアップ
アナライザ	Acute TravelBus ロジックアナライザ、800 MS/s、スレッショルド 1.6 V
ターゲット	PIC18F16Q20 Curiosity Nano、`stretch_target_pic18q20.c`、アドレス 0x2A

9.2 結果

6 節のスイープは、2 つのレートと 6 つのストレッチ時間にわたって、16 バイトずつ 1200 回読み出しました。 これを 3 回実行し、3 回目はアダプタのリセット後に行いました。3600 回が正しく、エラーはありませんでした。

同じ機能のファームウェア検証において、プレリリースビルドで得られたその他の結果は次のとおりです。

- 1 から 64 バイトのランダムな読み出しと書き込みを行い、100 転送ごとにストレッチを 0、50 us、500 us、 2 ms、10 ms の間で変更したソーク試験：100 kHz と 400 kHz で、1 回の実行あたり約 3400 回のストレッチを 伴う転送で不正な読み出しは 0 回でした。電源サイクル後の実行を含みます。

ファームウェア 4.6.1 における要求ごとの SCL レートです。400 kHz を超える条件では、50 us から 30 ms の 各ストレッチについて 16 バイトの読み出しを 20 回行いました。

表 11. 要求ごとの SCL レート

要求	報告値	実際の波形	tLOW / tHIGH	ストレッチを伴う読み出し
100 kHz	97975 Hz	98.2 kHz	5105 / 5075 ns	1500 回中 1500 回
400 kHz	391644 Hz	395.3 kHz	1600 / 935 ns	1500 回中 1500 回
600 kHz	583657 Hz	615.4 kHz	855 / 755 ns	100 回中 100 回
800 kHz	746268 Hz	749.1 kHz	750 / 585 ns	100 回中 100 回
1 MHz	746268 Hz	749.1 kHz	750 / 585 ns	100 回中 100 回

実際の波形の値は、各レートで約 9000 の SCL 周期にわたる中央値です。すべてのレートで、tLOW と tHIGH は 各モードの UM10204 の最小値を上回っています。400 kHz を超えると、返されるレートは実際の波形に追従 しません。500 kHz の要求は 477707 Hz を報告して 467.3 kHz で動作し、700 kHz は 652173 Hz を報告して 696.9 kHz で動作し、600 kHz は要求より 2.6 % 高く動作します。400 kHz を超える正確なレートが重要な 場合は、実際の波形を測定してください。

リファレンスターゲットがストレッチを行わない場合、400 kHz を超える読み出しは 2 バイト目に 0xFF を返し、 各レートで 100 回中 0 回が正しい結果でした。一方、ストレッチを伴う読み出しはすべて正しい結果でした。 アダプタの SCL はどちらの場合も同じで、ターゲット自身のクロックストレッチがオフになっている点だけが 異なります。私たちは、これをアダプタではなく、ストレッチなしでは 2 バイト目を間に合うようにロードできない リファレンスターゲットのファームウェアに起因するものと考えています。

SMBus リセット（`hold_scl_low`）、各ポイント 5 回の保持、プレリリースファームウェア `FW-4.5.1-b11a067e` 、 Acute で測定した SCL の Low 時間です。アナライザのバッファ境界をまたぐ保持は約 1.5 ms 短く読み取られる ため、除外しています。

表 12. SMBus リセットのための SCL 保持

要求値	実際の波形での測定値	アダプタの報告値	保持後の読み出し
1 ms	999.7 から 1678 us	1000 から 1677 us	5 回中 5 回
10 ms	9987 から 11006 us	10000 から 10998 us	5 回中 5 回
35 ms	34996 から 35941 us	35000 から 35937 us	各モードで 5 回中 5 回、リセット後も同様
100 ms	100013 から 100114 us	100000 us	5 回中 5 回

保持は最大で約 1 ms 長くなることがあり、アナライザのクロックに対して最大 0.13 % 短くなることがあります。 これはアダプタの内部発振器の許容差です。このマージンを見込んで T2wrst を要求してください。0 us と 1 s を 超える値は `FW_INVALID_PARAMETER` で拒否され、ファームウェア 4.5.1 は `FW_UNSUPPORTED_COMMAND` を 返します。

SDA プルアップを、Pulsar のプルアップをオフにした状態で、各 STOP において SCL の立ち上がりから SDA が 1.6 V を横切るまでの遅延として測定しました。`OFF` で 7813 ns、`ON` で 5975 ns、

`AUTO` で 5965 ns です。 読み出しは、クロックストレッチモードの有無にかかわらず、3 つの設定すべてで 10 回中 10 回でした。

これらは 1 つのベンチで私たちが測定した値であり、仕様値ではありません。

10 トラブルシューティング

表 13. トラブルシューティング

症状	考えられる原因と対処
enable で `FW_UNSUPPORTED_COMMAND`	ファームウェア 4.5.1 以前です。Supernova を 4.6.0 以降に更新してください。
報告されるレートが要求値を大きく下回る	SCL にプルアップがありません。追加してください（3.3 節）。
`FW_I3C_I2C_CLOCK_STRETCHING_ACTIVE`	モードがオンの状態で I3C の呼び出しが行われました。先に `disable_i2c_clock_stretching` を呼び出してください。
`FW_BUS_TIMEOUT`	ターゲットが SCL を 35 ms より長く保持しました。ターゲットを確認してください。これは SMBus の tTIMEOUT です。
リファレンスターゲットの再設定直後の `FW_I3C_NACK_ADDRESS`	ターゲットは新しいストレッチを適用するために I2C ペリフェラルを再起動します。スクリプトはこのために 10 ms 待機します。
モードがオフのとき、クロックストレッチを行うターゲットから誤ったデータ	想定どおりです。I3C エンジンは待機しません。モードをオンにしてください。
400 kHz を超える条件で enable 時に `FW_INVALID_PARAMETER`	1 MHz の上限を導入した 4.6.0 より前のファームウェアです。Supernova を更新してください。
400 kHz を超える条件で `sweep` に `FW_I3C_I2C_NACK_BYTE`	リファレンスターゲットが、ストレッチを行っていない間に再設定の書き込みを取りこぼしました。私たちのベンチでは 24 回のスイープ中 1 回でした。スイープを再実行してください。
Saleae Logic Pro で `capture_flow.py` が誤ったアドレスとデータをデコードする	デジタルスレッショルドがデバイスのデフォルトのままです。3.3 V バスでは `--threshold 3.3` を指定してください。私たちの Logic Pro 8 では、同じスイープにおいて SCL が High の間に検出された SDA エッジが 732 から 50 に減少し、残りはすべて START と STOP でした。
I3C Basic デバイスがバス上にあるのに `handoff` のステップ 1 が NACK	デバイスが設定した電圧で SMBus モードになっていないか、電源が入っていません。`--voltage-mv` を確認してください。

11 略語

表 14. 略語

用語	意味
I2C	Inter-Integrated Circuit
I3C	MIPI Improved Inter-Integrated Circuit
SMBus	System Management Bus
MCTP	Management Component Transport Protocol
SCL	シリアルクロックライン
SDA	シリアルデータライン
SDR	シングルデータレート、I3C の基本転送モード
HDR	I3C の高データレート転送モード
CCC	I3C 共通コマンドコード
tTIMEOUT	SMBus のクロック Low タイムアウト、35 ms
ARP	SMBus Address Resolution Protocol
PEC	SMBus パケットエラーコード、CRC-8
UDID	SMBus 固有デバイス識別子
DISEC, ENEC	ターゲットイベントを無効化および有効化する I3C CCC
Hot-Join	ターゲットがバスに参加してダイナミックアドレスを受け取る I3C イベント
NVMe-MI	NVM Express Management Interface
T2wrst, Tsmb2i3c	PCI-SIG 検出フローのタイミングパラメータ

12 本書のソースコードに関する注記

本書に示し、付属のアーカイブで提供するサンプルコードは、評価および参考用に提供するものです。MIT ライセンスの下で公開されており、その全文はアーカイブに含まれています。

アーカイブ `AN0014-assets.zip` には、`smbus_stretch.py`、`capture_flow.py`、`flow_labels` Logic 2 拡張機能、`test_flow_labels.py`、`stretch_target_pic18q20.c` および `LICENSE` が含まれています。

13 改訂履歴

表 15. 改訂履歴

リビジョン	日付	説明
0.1	25 September 2026	社内レビュー用ドラフト。ファームウェア 4.6.0 と pyCosmicSDK 1.6.0 のプレリリースビルドで測定。
0.2	28 September 2026	NVMe-MI 向けにタイトルを変更。最大 1 MHz のレート（実効 746 kHz）。7 節の SMBus から I3C Basic への検出フローと `handoff` コマンドを追加。
0.3	28 September 2026	検出フローに `hold_scl_low` と `set_sda_pull_up` を追加し、その測定結果を記載。
0.4	30 September 2026	ファームウェア 4.6.1 と PyPI の pyCosmicSDK 1.6.0 で再測定。すべての要求について実際の波形の SCL レートを記載。400 kHz を超えると返されるレートは概算値。`--serial` が pyCosmicSDK 1.6.0 で動作。
1.0	6 November 2026	公開。

法的情報（英語）

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** および **Curiosity** は、Microchip Technology Incorporated の商標または登録商標です。**Acute** および **TravelBus** は、Acute Technology Inc. の商標または登録商標です。**PCI-SIG** は PCI-SIG の登録商標です。**NVMe** および **NVMe-MI** は NVM Express, Inc. の商標です。