binho

アプリケーションノート

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

AN0015

文書情報

キーワード	AN0015, Supernova, Pulsar, I2C, SCL, デューティ比, tHIGH, tLOW, タイミングマージン, バス周波数, pyCosmicSDK
概要	一部の I2C ターゲットや一部の認定試験計画では、1 秒あたりの周期数だけでなく、各 SCL 周期が High と Low にどのように配分されるかが問題になります。アダプタファームウェア 4.6.0 と pyCosmicSDK 1.6.0 により、 Binho Supernova または Pulsar でこの配分を 20 から 80 パーセント High まで 10 刻みで直接設定できる ようになりました。また、設定するとバス周波数は丸められた値ではなく正確な値になります。本ノートでは、 この呼び出しを説明し、比率を適用してアダプタの報告値を表示するスクリプトを提供し、100 kHz と 400 kHz においてロジックアナライザで実際の波形上で測定した比率を示します。

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

改訂 1.0
6 NOVEMBER 2026

1 はじめに

I2C コントローラは、各 SCL 周期について High 時間と Low 時間を選択します。I2C 仕様は両方の最小値 （tHIGH と tLOW）を定めており、ほとんどのコントローラは配分を独自に選択します。ターゲットがこれに 敏感である場合や、テスト計画が特定の配分を求める場合までは、それで問題ありません。たとえば、High フェーズが短くてもターゲットが正しくサンプリングすることを確認するための 30:70 のクロックや、ターゲット が搭載されるシステムのタイミングに合わせるための 40:60 のクロックです。

これまで Binho のアダプタは配分を自ら選択していました。100 kHz ではデフォルトは約 46 パーセント High、 400 kHz では約 40 パーセントです。別の配分を要求する方法はありませんでした。アダプタファームウェア 4.6.0 と pyCosmicSDK 1.6.0 は、これを設定する呼び出しを 1 つ追加しました。

1.1 適用範囲

本書では次の内容を扱います。

- `I2cController.set_duty_cycle` が行うことと、変更しないこと
- ファームウェアが受け付ける比率と、拒否する比率に対する応答
- 比率を設定すると周波数も正確になる理由
- 比率を適用し、測定用のトラフィックを生成し、アダプタの報告値を表示するスクリプト
- 100 kHz と 400 kHz において実際の波形上で測定した配分

20 から 80 パーセントの範囲外の比率、Supernova の I3C ポートにおけるデューティ比制御、および SPI は 扱いません。SPI のクロックタイミングは別のファームウェア変更です。

1.2 対象デバイス

表 1. 対象デバイス

アダプタ	識別情報	根拠
Binho Supernova	ハードウェアリビジョン C、ファームウェア 4.6.0 プレリリースビルド `80cbb484`	測定済み、本書
Binho Pulsar	ハードウェアリビジョン C、ファームウェア 4.6.0 プレリリースビルド `ea77618b`	ファームウェア検証中に測定、Supernova との差は 1 パーセント以内。本書のための再測定はしていません
いずれかのアダプタ	ファームウェア 4.5.1 以前	`FW_UNSUPPORTED_COMMAND` で呼び出しを拒否

> **注記：** 本リビジョンはプレリリースビルドに基づいて作成しました。バージョン文字列がまだ 4.6.0 に 更新されていなかったため、ファームウェアはバージョンをビルドサフィックス付きの `4.5.1` として報告します。 呼び出しとその動作は 4.6.0 リリースのものです。

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
AN0014	Binho Supernova による NVMe-MI 向けシングルポート I2C/SMBus/I3C 動作

2 システム概要

`set_duty_cycle(high_pct)` は、各 SCL 周期のうちクロックを High に保持する割合を設定します。ファーム ウェアの実装から、次の 3 点が導かれます。

- **周波数は維持されます。** バスは `bring_up` または `configure` に与えた周波数を保ちます。変わるのは 配分だけです。
- **周波数が正確になります。** 比率を設定しない場合、アダプタは到達可能な最も近い分周比を選択し、要求 より高い値に丸めることがあります。550 kHz は 600.6 kHz で動作します。比率を設定すると、ファームウェアは I2C ペリフェラルをより細かいクロックに切り替えるため、周波数は要求を上回ることがなく、下回る場合も 最大 0.29 パーセントです。同じ 550 kHz は 549.8 kHz で動作します。
- **比率は保持されます。** 後の `configure` は比率を保ち、新しい周波数でそれを適用します。`set_duty_cycle(0)` または新たな `initialize` でデフォルトのタイミングに戻ります。

この呼び出しは、有効な設定を 2 つのフィールドを持つ `I2cDutyCycle` として返します。

表 3. I2cDutyCycle のフィールド

フィールド	意味
`frequency_hz`	比率を設定している場合は実際の波形上の周波数。`high_pct` が 0 の場合は要求した周波数で、デフォルトのタイミングでは丸められることがあります
`high_pct`	有効な比率（パーセント）。0 はデフォルトのタイミング

ファームウェアは比率を I2C 仕様に合わせて制限しません。100 kHz で 30 パーセントのクロックは tHIGH が 3.0 us となり、Standard-mode の最小値 4.0 us を下回ります。これは意図的なものです。仕様外のタイミングは マージン試験に必要なものです。製品で使用する前に、比率を仕様と照合してください。

3 ハードウェアのセットアップ

3.1 必要な機器

表 4. 必要な機器

品目
Binho Supernova または Pulsar、ファームウェア 4.6.0 以降
USB-C ケーブル
I2C ターゲット。SCL の測定だけが目的であれば不要
結果を測定するための SCL 上のオシロスコープまたはロジックアナライザ

3.2 接続

表 5. I2C ポートへの接続

アダプタ	ターゲットまたは測定器	要否
SCL	ターゲットの SCL、測定器のプローブ	必須
SDA	ターゲットの SDA	ターゲットを使用する場合は必須
GND	ターゲットと測定器のグランド	必須
3.3 V	ターゲットの電源	任意

3.3 バスに関する考慮事項

アダプタは、SCL が High であることを検出した時点から tHIGH を計測します。そのため、立ち上がりエッジが 遅いと周期が長くなります。2.2 kOhm のプルアップでは、1 MHz のクロックは約 905 kHz で動作します。1 MHz では 470 Ohm またはそれより強いプルアップを使用してください。100 kHz と 400 kHz では 1 kOhm から 2.2 kOhm で十分です。

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 `i2c_duty_cycle.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 デューティ比の呼び出し

呼び出しのみを示します。

```
import pycosmicsdk as p

dev = p.Device.open(model=p.DeviceModel.SUPERNOVA)
bus = dev.i2c()
bus.set_voltage(voltage_mv=3300)
bus.bring_up(frequency_hz=400_000, pull_up=p.I2cPullUp.KOHM_1_0)
print(bus.set_duty_cycle(high_pct=40))
# I2cDutyCycle(frequency_hz=400000, high_pct=40)
```

表 6. set_duty_cycle が受け付ける値

`high_pct`	結果
0	デフォルトのタイミング。ファームウェア 4.5.1 と同じ
20, 30, 40, 50, 60, 70, 80	各周期のその割合が High。周波数は正確
その他の値	`FirmwareError`、ステータス `FW_INVALID_PARAMETER`。有効な比率は維持される

ファームウェア 4.5.1 以前では、この呼び出しはステータス `FW_UNSUPPORTED_COMMAND` の `FirmwareError` を 送出し、バスは動作し続けます。

6 手順

1. アダプタと、使用する場合は SCL 上の測定器を接続します。
2. 対象の周波数で、すべての比率についてアダプタが報告する値を問い合わせます。各周波数を `configure` で 適用し、続いて各比率を順に適用し、最後にデフォルトのタイミングを復元します。

```
$ python i2c_duty_cycle.py --model supernova --pull-up KOHM_1_0 table --khz 100 400
1000
SUPERNOVA FW-4.5.1-80cbb484  I2C 3300 mV  pull-up KOHM_1_0
request   100 kHz  ->    100000 Hz  default
request   100 kHz  ->    100000 Hz  20 % high
request   100 kHz  ->    100000 Hz  30 % high
request   100 kHz  ->    100000 Hz  40 % high
request   100 kHz  ->    100000 Hz  50 % high
request   100 kHz  ->    100000 Hz  60 % high
request   100 kHz  ->    100000 Hz  70 % high
request   100 kHz  ->    100000 Hz  80 % high
request   400 kHz  ->    400000 Hz  default
request   400 kHz  ->    400000 Hz  20 % high
request   400 kHz  ->    400000 Hz  30 % high
request   400 kHz  ->    400000 Hz  40 % high
request   400 kHz  ->    400000 Hz  50 % high
request   400 kHz  ->    400000 Hz  60 % high
request   400 kHz  ->    400000 Hz  70 % high
request   400 kHz  ->    400000 Hz  80 % high
request  1000 kHz  ->   1000000 Hz  default
request  1000 kHz  ->    999024 Hz  20 % high
request  1000 kHz  ->    999024 Hz  30 % high
request  1000 kHz  ->    999024 Hz  40 % high
request  1000 kHz  ->    999024 Hz  50 % high
request  1000 kHz  ->    999024 Hz  60 % high
request  1000 kHz  ->    999024 Hz  70 % high
request  1000 kHz  ->    999024 Hz  80 % high
```

1 MHz では正確な周波数は 999024 Hz で、要求より 0.1 パーセント低い値です。`default` の行は実際の波形 ではなく要求値を報告します。その違いは 8.2 節に示します。

3. 1 つの比率を適用します。`--traffic` を指定すると、スクリプトは測定器がキャプチャできるように、その 秒数の間 SCL を動作させ続けます。

```
$ python i2c_duty_cycle.py --model supernova --pull-up KOHM_1_0 set --khz 100 --high
30
SUPERNOVA FW-4.5.1-80cbb484  I2C 3300 mV  pull-up KOHM_1_0
request   100 kHz  ->    100000 Hz  30 % high
```

4. 受け付けられる値以外の比率は拒否され、有効な比率は維持されます。

```
$ python i2c_duty_cycle.py --model supernova --pull-up KOHM_1_0 set --khz 100 --high
35
error: i2c_set_duty_cycle: FW_INVALID_PARAMETER: At least one of the parameters of
the command is not valid.
SUPERNOVA FW-4.5.1-80cbb484  I2C 3300 mV  pull-up KOHM_1_0
```

5. デフォルトのタイミングで丸められる周波数において、比率とデフォルトを比較します。

```
$ python i2c_duty_cycle.py --model supernova --pull-up KOHM_1_0 set --khz 550 --high
50
SUPERNOVA FW-4.5.1-80cbb484  I2C 3300 mV  pull-up KOHM_1_0
request   550 kHz  ->    549798 Hz  50 % high
$ python i2c_duty_cycle.py --model supernova --pull-up KOHM_1_0 set --khz 550 --high
0
SUPERNOVA FW-4.5.1-80cbb484  I2C 3300 mV  pull-up KOHM_1_0
request   550 kHz  ->    550000 Hz  default
```

2 回目の結果は要求値をそのまま返すため 550000 Hz と報告されます。実際の波形では、デフォルトは 600.6 kHz で動作します（8.2 節）。

7 リファレンス

表 7. i2c_duty_cycle.py のオプション

オプション	デフォルト	意味
`--model`	任意	両方が接続されている場合の `supernova` または `pulsar`
`--serial`	任意	アダプタのシリアル番号
`--voltage-mv`	3300	バス電圧
`--pull-up`	`KOHM_2_2`	オンボードのプルアップ、任意の `I2cPullUp` 名

表 8. i2c_duty_cycle.py のコマンド

コマンド	オプション	動作
`set`	`--khz`、`--high`、`--traffic`、`--address`	1 つの比率を適用。`--traffic` を指定すると、その秒数の間 `--address` から読み出す
`table`	`--khz`（1 つ以上）	0 および 20 から 80 パーセントについて報告された周波数と比率を表示し、その後デフォルトを復元

トラフィックはデフォルトでアドレス 0x50 から読み出します。そこにターゲットがない場合、各読み出しは アドレスバイトと NACK になりますが、それでも 9 つの SCL 周期をクロッキングしま

す。

8 測定結果

8.1 テスト構成

表 9. 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 `FW-4.5.1-80cbb484`（4.6.0 プレリリース）
ホストソフトウェア	pyCosmicSDK 1.6.0 プレリリース、`integration/1.6.0` の `baacfdba`
バス	I2C ポート、3.3 V、1 kOhm のオンボードプルアップ、ターゲットなし
トラフィック	`set --traffic 10`、1 キャプチャあたり約 4800 転送
測定器	SCL 上の Acute TravelBus ロジックアナライザ、800 MS/s、スレッショルド 1.6 V
キャプチャ	各ポイント 3 回、さらに電源サイクル後に各ポイント 1 回。それぞれ約 15000 の SCL 周期

8.2 結果

中央値です。比率の列は 1 つのキャプチャにおける周期に対する tHIGH の比の中央値で、p1 から p99 の列は そのばらつきです。各ポイントの 4 回のキャプチャはすべて 0.1 パーセント以内で一致しました。

表 10. 実際の波形上で測定した SCL タイミング

要求	比率	周波数	High	tHIGH	tLOW	High、p1 から p99
100 kHz	デフォルト	100.0 kHz	46.4 %	4645 ns	5360 ns	46.1 から 46.9 %
100 kHz	30	100.0 kHz	29.7 %	2975 ns	7030 ns	29.4 から 30.1 %
100 kHz	40	100.0 kHz	39.9 %	3990 ns	6015 ns	39.2 から 40.1 %
100 kHz	50	100.0 kHz	49.6 %	4965 ns	5035 ns	49.2 から 50.2 %
400 kHz	デフォルト	400.0 kHz	39.6 %	990 ns	1510 ns	38.6 から 41.1 %
400 kHz	30	400.0 kHz	28.0 %	700 ns	1800 ns	27.2 から 30.7 %
400 kHz	40	400.0 kHz	39.5 %	990 ns	1515 ns	38.8 から 40.7 %
400 kHz	50	400.0 kHz	48.8 %	1220 ns	1280 ns	48.2 から 50.7 %

400 kHz では、測定された High の割合は比率より 0.5 から 2 ポイント低くなります。これは立ち上がりエッジに よるものです。アナライザは SCL が 1.6 V を横切った時点で初めて High と読み取り、400 kHz ではその遅延が 周期に占める割合が大きくなります。

さらに 2 つのポイントを、同じ変更の以前のビルドでファームウェア検証中に測定しました。上記の構成による ものではありません。

- 470 Ohm のプルアップでの 1 MHz、Supernova：比率 30、40、50 に対して、1000.0 kHz で 29.5、37.0、 50.5 パーセント。
- 550 kHz：デフォルトのタイミングは、両方のアダプタおよび 4.5.1 リリースで、600.6 kHz、50.2 パーセント High で動作します。比率 50 では 551.0 kHz で動作します。

20、60、70、80 パーセントの比率は、アダプタの報告値（6 節のステップ 2）で確認しており、実際の波形上では 確認していません。

9 トラブルシューティング

表 11. トラブルシューティング

症状	考えられる原因と対処
`FW_UNSUPPORTED_COMMAND`	ファームウェア 4.5.1 以前です。アダプタを 4.6.0 以降に更新してください。
`FW_INVALID_PARAMETER`	比率が 0、または 20 から 80 の 10 刻みのいずれでもありません。以前の比率が引き続き有効です。
`FW_INTERFACE_NOT_INITIALIZED`	I2C バスがコントローラとして起動されていません。先に `bring_up` を呼び出してください。
1 MHz で測定した周波数が要求を下回る	立ち上がりエッジが遅いです。470 Ohm またはそれより強いプルアップを使用してください（3.3 節）。
スクリプトの再起動後に比率がデフォルトに戻る	`initialize` はデフォルトのタイミングを復元します。バスを初期化する `bring_up` のたびに、その後で `set_duty_cycle` を呼び出してください。
`table` から `CapabilityError`	`configure` にはハードウェアリビジョン C の Supernova が必要です。代わりに周波数ごとに `set` を 1 回ずつ使用してください。

10 略語

表 12. 略語

用語	意味
I2C	Inter-Integrated Circuit
SCL	シリアルクロックライン
SDA	シリアルデータライン
tHIGH	SCL クロックの High 期間
tLOW	SCL クロックの Low 期間
NACK	Not acknowledge

11 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 `AN0015-assets.zip` には `i2c_duty_cycle.py` と `LICENSE` が含まれています。

12 改訂履歴

表 13. 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。
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. **Acute** および **TravelBus** は、Acute Technology Inc. の商標または登録商標です。