binho

APPLICATION NOTE

Setting the I2C SCL duty cycle and an exact bus frequency with the Binho Supernova and Pulsar

AN0015

Document information

Keywords	AN0015, Supernova, Pulsar, I2C, SCL, duty cycle, tHIGH, tLOW, timing margin, bus frequency, pyCosmicSDK
Abstract	Some I2C targets, and some qualification plans, care how each SCL period is split between high and low, not only how many periods there are per second. Adapter firmware 4.6.0 and pyCosmicSDK 1.6.0 let a Binho Supernova or Pulsar set that split directly, from 20 to 80 percent high in steps of 10, and setting it also makes the bus frequency exact rather than rounded. This note explains the call, supplies a script that applies a ratio and prints what the adapter reports, and gives the ratios measured on the wire at 100 kHz and 400 kHz with a logic analyzer.

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

REVISION 1.0
6 NOVEMBER 2026

1 Introduction

An I2C controller chooses a high time and a low time for every SCL period. The I2C specification sets minimums for both (tHIGH and tLOW), and most controllers pick a split on their own. That is fine until a target is sensitive to it, or until a test plan asks for a specific split: a 30:70 clock to check that a target samples correctly with a short high phase, or a 40:60 clock to match the timing of the system the target will ship in.

Until now the Binho adapters chose the split themselves. At 100 kHz the default is about 46 percent high; at 400 kHz, about 40 percent. There was no way to ask for another. Adapter firmware 4.6.0 and pyCosmicSDK 1.6.0 add one call that sets it.

1.1 Scope

This document covers:

- what `I2cController.set_duty_cycle` does and what it leaves alone
- the ratios firmware accepts, and what it answers for the ones it refuses
- why setting a ratio also makes the frequency exact
- a script that applies a ratio, generates traffic to measure, and prints what the adapter reports
- the split measured on the wire at 100 kHz and 400 kHz

It does not cover ratios outside 20 to 80 percent, duty-cycle control on the Supernova's I3C port, or SPI. SPI clock timing is a separate firmware change.

1.2 Applicable devices

Table 1. Applicable devices

Adapter	Identifier	Basis
Binho Supernova	hardware revision C, firmware 4.6.0 pre-release build `80cbb484`	measured, this document
Binho Pulsar	hardware revision C, firmware 4.6.0 pre-release build `ea77618b`	measured during firmware validation, within 1 percent of the Supernova; not re-measured for this document
Either adapter	firmware 4.5.1 or earlier	refuses the call with `FW_UNSUPPORTED_COMMAND`

> **Note:** this revision was written against pre-release builds. The firmware reports its version as `4.5.1` with a build suffix because the version string had not yet been moved to 4.6.0. The call and its behavior are those of the 4.6.0 release.

1.3 Related documents

Table 2. Related documents

Document	Title
UM10204	I2C-bus specification and user manual, Rev. 7, NXP Semiconductors
AN0014	Single-port I2C/SMBus/I3C operation for NVMe-MI on Binho Supernova

2 System overview

`set_duty_cycle(high_pct)` sets the share of each SCL period the clock is held high. Three things follow from how the firmware implements it:

- **The frequency stays.** The bus keeps the frequency given to `bring_up` or `configure`. Only the split changes.
- **The frequency becomes exact.** Without a ratio, the adapter picks the nearest divider it can reach, and it may round above the request: 550 kHz runs at 600.6 kHz. With a ratio, the firmware moves the I2C peripheral onto a finer clock, so the frequency is never above the request and at most 0.29 percent below it. The same 550 kHz runs at 549.8 kHz.
- **The ratio persists.** A later `configure` keeps the ratio and applies it at the new frequency. `set_duty_cycle(0)` or a fresh `initialize` returns to the default timing.

The call returns what is in effect, as an `I2cDutyCycle` with two fields:

Table 3. Fields of I2cDutyCycle

Field	Meaning
`frequency_hz`	With a ratio set, the frequency on the wire. With `high_pct` 0, the requested frequency, which the default timing may round
`high_pct`	The ratio in effect, in percent; 0 is the default timing

The firmware does not clamp the ratio to the I2C specification. A 30 percent clock at 100 kHz has a tHIGH of 3.0 us, below Standard-mode's 4.0 us minimum. That is deliberate: out-of-spec timing is what margin testing needs. Check a ratio against the specification before using it in a product.

3 Hardware setup

3.1 Required equipment

Table 4. Required equipment

Item
Binho Supernova or Pulsar, firmware 4.6.0 or later
USB-C cable
An I2C target, or none if the goal is only to measure SCL
Oscilloscope or logic analyzer on SCL, to measure the result

3.2 Connections

Table 5. Connections to the I2C port

Adapter	Target or instrument	Required
SCL	target SCL, instrument probe	required
SDA	target SDA	required with a target
GND	target and instrument ground	required
3.3 V	target supply	optional

3.3 Bus considerations

The adapter measures tHIGH from the moment it sees SCL high. A slow rising edge therefore lengthens the period: with a 2.2 kOhm pull-up, a 1 MHz clock runs at about 905 kHz. At 1 MHz use 470 Ohm or a stronger pull-up. At 100 kHz and 400 kHz, 1 kOhm to 2.2 kOhm is enough.

4 Software setup

Install pyCosmicSDK (latest release):

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

The script `i2c_duty_cycle.py` is in the assets archive. It needs nothing else.

5 The duty-cycle call

The call in isolation:

```
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)
```

Table 6. Values accepted by set_duty_cycle

`high_pct`	Result
0	default timing, the same as firmware 4.5.1
20, 30, 40, 50, 60, 70, 80	that share of each period high; frequency exact
anything else	`FirmwareError`, status `FW_INVALID_PARAMETER`; the ratio in effect is kept

On firmware 4.5.1 and earlier, the call raises `FirmwareError` with status `FW_UNSUPPORTED_COMMAND`, and the bus keeps working.

6 Procedure

1. Connect the adapter and, if used, the instrument on SCL.
2. Ask the adapter what it reports for every ratio at the frequencies of interest. Each frequency is applied with `configure`, then every ratio in turn, then the default timing is restored:

```
$ 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
```

At 1 MHz the exact frequency is 999024 Hz, 0.1 percent under the request. The `default` rows report the request, not the wire; section 8.2 shows the difference.

3. Apply one ratio. With `--traffic`, the script keeps SCL running for that many seconds so the instrument can capture it:

```
$ 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. A ratio outside the accepted set is refused, and the one in effect stays:

```
$ 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. Compare a ratio with the default at a frequency the default rounds:

```
$ 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
```

The second line reports 550000 Hz because it echoes the request. On the wire the default runs at 600.6 kHz (section 8.2).

7 Reference

Table 7. i2c_duty_cycle.py options

Option	Default	Meaning
`--model`	any	`supernova` or `pulsar`, when both are connected
`--serial`	any	adapter serial number
`--voltage-mv`	3300	bus voltage
`--pull-up`	`KOHM_2_2`	on-board pull-up, any `I2cPullUp` name

Table 8. i2c_duty_cycle.py commands

Command	Options	Action
`set`	`--khz`, `--high`, `--traffic`, `--address`	apply one ratio; with `--traffic`, read from `--address` for that many seconds
`table`	`--khz` (one or more)	print the reported frequency and ratio for 0 and 20 to 80 percent, then restore the default

The traffic reads from address 0x50 by default. With no target there, each read is an address byte and a NACK, which still clocks nine SCL periods.

8 Measured results

8.1 Test configuration

Table 9. Test configuration

Item	Value
Adapter	Binho Supernova, hardware revision C, firmware `FW-4.5.1-80cbb484` (4.6.0 pre-release)
Host software	pyCosmicSDK 1.6.0 pre-release, `integration/1.6.0` at `baacfdba`
Bus	I2C port, 3.3 V, 1 kOhm on-board pull-up, no target
Traffic	`set --traffic 10`, about 4800 transfers per capture
Instrument	Acute TravelBus logic analyzer on SCL, 800 MS/s, threshold 1.6 V
Captures	3 per point, then 1 per point after a power cycle; about 15000 SCL periods each

8.2 Results

Median values. The ratio column is the median of tHIGH over the period across one capture; the p1 to p99 column is its spread. All four captures of each point agreed to 0.1 percent.

Table 10. SCL timing measured on the wire

Request	Ratio	Frequency	High	tHIGH	tLOW	High, p1 to p99
100 kHz	default	100.0 kHz	46.4 %	4645 ns	5360 ns	46.1 to 46.9 %
100 kHz	30	100.0 kHz	29.7 %	2975 ns	7030 ns	29.4 to 30.1 %
100 kHz	40	100.0 kHz	39.9 %	3990 ns	6015 ns	39.2 to 40.1 %
100 kHz	50	100.0 kHz	49.6 %	4965 ns	5035 ns	49.2 to 50.2 %
400 kHz	default	400.0 kHz	39.6 %	990 ns	1510 ns	38.6 to 41.1 %
400 kHz	30	400.0 kHz	28.0 %	700 ns	1800 ns	27.2 to 30.7 %
400 kHz	40	400.0 kHz	39.5 %	990 ns	1515 ns	38.8 to 40.7 %
400 kHz	50	400.0 kHz	48.8 %	1220 ns	1280 ns	48.2 to 50.7 %

At 400 kHz the measured high share sits 0.5 to 2 points under the ratio. The rising edge accounts for it: the analyzer reads SCL as high only once it crosses 1.6 V, and at 400 kHz that delay is a larger share of the period.

Two further points were measured during firmware validation on earlier builds of the same change, not with the configuration above:

- 1 MHz with a 470 Ohm pull-up, Supernova: 1000.0 kHz at 29.5, 37.0 and 50.5 percent for ratios 30, 40 and 50.
- 550 kHz: the default timing runs at 600.6 kHz at 50.2 percent high, on both adapters and on the 4.5.1 release. A ratio of 50 runs at 551.0 kHz.

The 20, 60, 70 and 80 percent ratios were checked through the adapter's report (section 6, step 2) and not on the wire.

9 Troubleshooting

Table 11. Troubleshooting

Symptom	Probable cause and action
`FW_UNSUPPORTED_COMMAND`	Firmware 4.5.1 or earlier. Update the adapter to 4.6.0 or later.
`FW_INVALID_PARAMETER`	The ratio is not 0 or 20 to 80 in steps of 10. The previous ratio is still in effect.
`FW_INTERFACE_NOT_INITIALIZED`	The I2C bus was not brought up as a controller. Call `bring_up` first.
Measured frequency below the request at 1 MHz	The rising edge is slow. Use a 470 Ohm or stronger pull-up (section 3.3).
Ratio back to default after a script restarts	`initialize` restores the default timing. Call `set_duty_cycle` after every `bring_up` that initializes the bus.
`CapabilityError` from `table`	`configure` needs a Supernova of hardware revision C. Use `set` once per frequency instead.

10 Acronyms

Table 12. Acronyms

Term	Meaning
I2C	Inter-Integrated Circuit
SCL	Serial clock line
SDA	Serial data line
tHIGH	High period of the SCL clock
tLOW	Low period of the SCL clock
NACK	Not acknowledge

11 Note about the source code in the document

Example code shown in this document and supplied in the accompanying archive is provided for evaluation and reference. It is released under the MIT license, the text of which is included in the archive.

The archive `AN0015-assets.zip` contains `i2c_duty_cycle.py` and `LICENSE`.

12 Revision history

Table 13. Revision history

Revision	Date	Description
0.1	25 September 2026	Draft for internal review, measured on firmware 4.6.0 and pyCosmicSDK 1.6.0 pre-release builds.
1.0	6 November 2026	Published.

Legal information

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** and **TravelBus** are trademarks or registered trademarks of Acute Technology Inc.