binho

애플리케이션 노트

Binho Supernova 및 Pulsar로 I2C SCL 듀티 사이클과 정확한 버스 주파수 설정하기

AN0015

문서 정보

키워드	AN0015, Supernova, Pulsar, I2C, SCL, 듀티 사이클(duty cycle), tHIGH, tLOW, 타이밍 마진, 버스 주파수, pyCosmicSDK
요약	일부 I2C 타깃과 일부 검증 계획은 초당 주기 수뿐만 아니라 각 SCL 주기가 high와 low로 어떻게 나뉘는지도 중요하게 봅니다. 어댑터 펌웨어 4.6.0과 pyCosmicSDK 1.6.0을 사용하면 Binho Supernova 또는 Pulsar에서 이 비율을 20에서 80 퍼센트 high까지 10 단위로 직접 설정할 수 있으며, 이를 설정하면 버스 주파수도 반올림되지 않고 정확해집니다. 이 애플리케이션 노트는 이 호출을 설명하고, 비율을 적용하고 어댑터가 보고하는 값을 출력하는 스크립트를 제공하며, 로직 분석기로 100 kHz와 400 kHz에서 버스상 측정한 비율을 제시합니다.

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

개정 1.0
6 NOVEMBER 2026

1 소개

I2C 컨트롤러는 모든 SCL 주기에 대해 high 시간과 low 시간을 정합니다. I2C 사양은 두 시간 모두에 최솟값(tHIGH 및 tLOW)을 정하며, 대부분의 컨트롤러는 스스로 비율을 정합니다. 이는 타깃이 이 비율에 민감하거나 테스트 계획이 특정 비율을 요구하기 전까지는 문제가 없습니다. 예를 들어 짧은 high 구간에서 타깃이 올바르게 샘플링하는지 확인하기 위한 30:70 클럭이나, 타깃이 탑재될 시스템의 타이밍에 맞추기 위한 40:60 클럭입니다.

지금까지 Binho 어댑터는 비율을 스스로 정했습니다. 100 kHz에서 기본값은 약 46 퍼센트 high이고, 400 kHz에서는 약 40 퍼센트입니다. 다른 비율을 요청할 방법이 없었습니다. 어댑터 펌웨어 4.6.0과 pyCosmicSDK 1.6.0은 이를 설정하는 호출 하나를 추가합니다.

1.1 범위

이 문서는 다음을 다룹니다.

- `I2cController.set_duty_cycle`이 하는 일과 변경하지 않는 것
- 펌웨어가 받아들이는 비율과, 거부하는 비율에 대한 응답
- 비율을 설정하면 주파수도 정확해지는 이유
- 비율을 적용하고, 측정할 트래픽을 생성하며, 어댑터가 보고하는 값을 출력하는 스크립트
- 100 kHz와 400 kHz에서 버스상 측정한 비율

20에서 80 퍼센트 범위 밖의 비율, Supernova I3C 포트의 듀티 사이클 제어, SPI는 다루지 않습니다. SPI 클럭 타이밍은 별도의 펌웨어 변경입니다.

1.2 적용 디바이스

표 1. 적용 디바이스

어댑터	식별자	근거
Binho Supernova	하드웨어 리비전 C, 펌웨어 4.6.0 사전 릴리스 빌드 `80cbb484`	측정, 이 문서
Binho Pulsar	하드웨어 리비전 C, 펌웨어 4.6.0 사전 릴리스 빌드 `ea77618b`	펌웨어 검증 중 측정, Supernova와 1 퍼센트 이내; 이 문서를 위해 재측정하지 않음
두 어댑터 모두	펌웨어 4.5.1 이하	`FW_UNSUPPORTED_COMMAND`로 호출을 거부

> **참고:** 이 리비전은 사전 릴리스 빌드를 기준으로 작성되었습니다. 버전 문자열이 아직 4.6.0으로 변경되지 않았기 때문에 펌웨어는 버전을 빌드 접미사가 붙은 `4.5.1`로 보고합니다. 호출과 그 동작은 4.6.0 릴리스와 같습니다.

1.3 관련 문서

표 2. 관련 문서

문서	제목
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 시스템 개요

`set_duty_cycle(high_pct)`는 각 SCL 주기 중 클럭이 high로 유지되는 비율을 설정합니다. 펌웨어의 구현 방식에서 다음 세 가지가 따릅니다.

- **주파수는 유지됩니다.** 버스는 `bring_up` 또는 `configure`에 지정한 주파수를 유지합니다. 비율만 바뀝니다.
- **주파수가 정확해집니다.** 비율이 없으면 어댑터는 도달할 수 있는 가장 가까운 분주비를 고르며, 요청보다 높게 반올림될 수 있습니다. 550 kHz는 600.6 kHz로 동작합니다. 비율이 있으면 펌웨어는 I2C 페리페럴을 더 세밀한 클럭으로 옮기므로, 주파수는 요청보다 높지 않으며 최대 0.29 퍼센트 낮습니다. 같은 550 kHz는 549.8 kHz로 동작합니다.
- **비율은 유지됩니다.** 이후의 `configure`는 비율을 유지하고 새 주파수에 적용합니다. `set_duty_cycle(0)` 또는 새로운 `initialize`는 기본 타이밍으로 되돌립니다.

이 호출은 적용 중인 값을 두 필드를 가진 `I2cDutyCycle`로 반환합니다.

표 3. I2cDutyCycle의 필드

필드	의미
`frequency_hz`	비율이 설정된 경우 버스상의 주파수. `high_pct`가 0이면 요청한 주파수이며, 기본 타이밍이 이를 반올림할 수 있음
`high_pct`	적용 중인 비율(퍼센트); 0은 기본 타이밍

펌웨어는 비율을 I2C 사양에 맞게 제한하지 않습니다. 100 kHz에서 30 퍼센트 클럭의 tHIGH는 3.0 us로, Standard-mode의 최솟값 4.0 us보다 짧습니다. 이는 의도된 것입니다. 사양을 벗어난 타이밍은 마진 테스트에 필요합니다. 제품에 사용하기 전에 비율을 사양과 대조하여 확인하십시오.

3 하드웨어 설정

3.1 필요 장비

표 4. 필요 장비

항목
Binho Supernova 또는 Pulsar, 펌웨어 4.6.0 이상
USB-C 케이블
I2C 타깃, 또는 SCL 측정만이 목적이라면 없어도 됨
결과를 측정하기 위한 SCL상의 오실로스코프 또는 로직 분석기

3.2 연결

표 5. I2C 포트 연결

어댑터	타깃 또는 계측기	필수 여부
SCL	타깃 SCL, 계측기 프로브	필수
SDA	타깃 SDA	타깃 사용 시 필수
GND	타깃 및 계측기 접지	필수
3.3 V	타깃 전원	선택

3.3 버스 고려 사항

어댑터는 SCL이 high로 보이는 순간부터 tHIGH를 측정합니다. 따라서 느린 상승 에지는 주기를 늘립니다. 2.2 kOhm 풀업에서 1 MHz 클럭은 약 905 kHz로 동작합니다. 1 MHz에서는 470 Ohm 이하의 더 강한 풀업을 사용하십시오. 100 kHz와 400 kHz에서는 1 kOhm에서 2.2 kOhm이면 충분합니다.

4 소프트웨어 설정

pyCosmicSDK(최신 릴리스)를 설치합니다.

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

스크립트 `i2c_duty_cycle.py` 는 에셋 아카이브에 있습니다. 그 밖에 필요한 것은 없습니다.

5 듀티 사이클 호출

호출만 따로 보면 다음과 같습니다.

```
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)
```

표 6. set_duty_cycle이 받아들이는 값

`high_pct`	결과
0	기본 타이밍, 펌웨어 4.5.1과 동일
20, 30, 40, 50, 60, 70, 80	각 주기 중 그 비율만큼 high; 주파수 정확
그 밖의 값	`FirmwareError`, 상태 `FW_INVALID_PARAMETER`; 적용 중인 비율은 유지됨

펌웨어 4.5.1 이하에서는 이 호출이 상태 `FW_UNSUPPORTED_COMMAND`와 함께 `FirmwareError`를 발생시키며, 버스는 계속 동작합니다.

6 절차

1. 어댑터와, 사용하는 경우 SCL에 계측기를 연결합니다.
2. 관심 있는 주파수에서 모든 비율에 대해 어댑터가 무엇을 보고하는지 확인합니다. 각 주파수는 `configure`로 적용한 후 모든 비율을 차례로 적용하고, 마지막으로 기본 타이밍을 복원합니다.

```
$ 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 MHz에서 정확한 주파수는 999024 Hz로, 요청보다 0.1 퍼센트 낮습니다. `default` 행은 버스상의 값이 아니라 요청값을 보고합니다. 그 차이는 8.2절에 나와 있습니다.

3. 비율 하나를 적용합니다. `--traffic` 을 지정하면 스크립트는 계측기가 캡처할 수 있도록 그 초만큼 SCL 을 계속 동작시킵니다.

```
$ 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. 허용 범위 밖의 비율은 거부되며, 적용 중인 비율이 유지됩니다.

```
$ 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. 기본 타이밍이 반올림하는 주파수에서 비율과 기본값을 비교합니다.

```
$ 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
```

두 번째 줄은 요청값을 그대로 돌려주기 때문에 550000 Hz를 보고합니다. 버스상에서 기본 타이밍은 600.6 kHz로 동작합니다(8.2절).

7 참조

표 7. i2c_duty_cycle.py 옵션

옵션	기본값	의미
`--model`	any	둘 다 연결된 경우 `supernova` 또는 `pulsar`
`--serial`	any	어댑터 시리얼 번호
`--voltage-mv`	3300	버스 전압
`--pull-up`	`KOHM_2_2`	온보드 풀업, 임의의 `I2cPullUp` 이름

표 8. i2c_duty_cycle.py 명령

명령	옵션	동작
`set`	`--khz`, `--high`, `--traffic`, `--address`	비율 하나를 적용; `--traffic`을 지정하면 그 초만큼 `--address`에서 읽음
`table`	`--khz` (하나 이상)	0 및 20에서 80 퍼센트에 대해 보고된 주파수와 비율을 출력한 후 기본값을 복원

트래픽은 기본적으로 주소 0x50에서 읽습니다. 그 주소에 타깃이 없으면 각 읽기는 주소 바이트와 NACK가 되며, 이것만으로도 SCL 주기 9개가 클럭됩니다.

8 측정 결과

8.1 테스트 구성

표 9. 테스트 구성

항목	값
어댑터	Binho Supernova, 하드웨어 리비전 C, 펌웨어 `FW-4.5.1-80cbb484` (4.6.0 사전 릴리스)
호스트 소프트웨어	pyCosmicSDK 1.6.0 사전 릴리스, `integration/1.6.0` 의 `baacfdba`
버스	I2C 포트, 3.3 V, 1 kOhm 온보드 풀업, 타깃 없음
트래픽	`set --traffic 10` , 캡처당 약 4800회 전송
계측기	SCL상의 Acute TravelBus 로직 분석기, 800 MS/s, 임계값 1.6 V
캡처	지점당 3회, 이후 전원을 껐다 켠 후 지점당 1회; 각각 약 15000개 SCL 주기

8.2 결과

중앙값입니다. 비율 열은 하나의 캡처에서 주기 대비 tHIGH의 중앙값이며, p1에서 p99 열은 그 분포 범위입니다. 각 지점의 네 캡처는 모두 0.1 퍼센트 이내로 일치했습니다.

표 10. 버스상 측정한 SCL 타이밍

요청	비율	주파수	High	tHIGH	tLOW	High, p1에서 p99
100 kHz	기본값	100.0 kHz	46.4 %	4645 ns	5360 ns	46.1에서 46.9 %
100 kHz	30	100.0 kHz	29.7 %	2975 ns	7030 ns	29.4에서 30.1 %
100 kHz	40	100.0 kHz	39.9 %	3990 ns	6015 ns	39.2에서 40.1 %
100 kHz	50	100.0 kHz	49.6 %	4965 ns	5035 ns	49.2에서 50.2 %
400 kHz	기본값	400.0 kHz	39.6 %	990 ns	1510 ns	38.6에서 41.1 %
400 kHz	30	400.0 kHz	28.0 %	700 ns	1800 ns	27.2에서 30.7 %
400 kHz	40	400.0 kHz	39.5 %	990 ns	1515 ns	38.8에서 40.7 %
400 kHz	50	400.0 kHz	48.8 %	1220 ns	1280 ns	48.2에서 50.7 %

400 kHz에서 측정된 high 비율은 설정 비율보다 0.5에서 2 포인트 낮습니다. 이는 상승 에지 때문입니다. 분석기는 SCL이 1.6 V를 넘은 후에야 high로 읽으며, 400 kHz에서는 이 지연이 주기에서 차지하는 비중이 더 큽니다.

다음 두 지점은 위 구성이 아니라, 같은 변경의 이전 빌드로 펌웨어 검증 중에 측정했습니다.

- 470 Ohm 풀업에서 1 MHz, Supernova: 비율 30, 40, 50에 대해 1000.0 kHz에서 29.5, 37.0, 50.5 퍼센트.

- 550 kHz: 기본 타이밍은 두 어댑터 모두와 4.5.1 릴리스에서 50.2 퍼센트 high로 600.6 kHz에서 동작합니다. 비율 50은 551.0 kHz로 동작합니다.

20, 60, 70, 80 퍼센트 비율은 어댑터의 보고(6절, 2단계)로 확인했으며 버스상에서는 측정하지 않았습니다.

9 문제 해결

표 11. 문제 해결

증상	가능한 원인과 조치
`FW_UNSUPPORTED_COMMAND`	펌웨어 4.5.1 이하. 어댑터를 4.6.0 이상으로 업데이트하십시오.
`FW_INVALID_PARAMETER`	비율이 0 또는 20에서 80까지 10 단위 값이 아닙니다. 이전 비율이 계속 적용됩니다.
`FW_INTERFACE_NOT_INITIALIZED`	I2C 버스가 컨트롤러로 기동되지 않았습니다. 먼저 `bring_up` 을 호출하십시오.
1 MHz에서 측정 주파수가 요청보다 낮음	상승 에지가 느립니다. 470 Ohm 이하의 더 강한 풀업을 사용하십시오(3.3절).
스크립트를 다시 시작한 후 비율이 기본값으로 돌아감	`initialize` 는 기본 타이밍을 복원합니다. 버스를 초기화하는 모든 `bring_up` 후에 `set_duty_cycle` 을 호출하십시오.
`table` 에서 `CapabilityError`	`configure` 에는 하드웨어 리비전 C의 Supernova가 필요합니다. 대신 주파수마다 `set` 을 한 번씩 사용하십시오.

10 약어

표 12. 약어

용어	의미
I2C	Inter-Integrated Circuit
SCL	직렬 클럭 라인(Serial clock line)
SDA	직렬 데이터 라인(Serial data line)
tHIGH	SCL 클럭의 high 구간
tLOW	SCL 클럭의 low 구간
NACK	Not acknowledge

11 문서 내 소스 코드에 관한 참고 사항

이 문서에 표시되고 함께 제공되는 아카이브에 포함된 예제 코드는 평가 및 참조용으로 제공됩니다. MIT 라이선스로 배포되며, 라이선스 전문은 아카이브에 포함되어 있습니다.

아카이브 `AN0015-assets.zip`에는 `i2c_duty_cycle.py`와 `LICENSE`가 포함되어 있습니다.

12 개정 이력

표 13. 개정 이력

리비전	날짜	설명
0.1	25 September 2026	내부 검토용 초안, 펌웨어 4.6.0 및 pyCosmicSDK 1.6.0 사전 릴리스 빌드에서 측정.
1.0	6 November 2026	공개.

법적 고지(영문)

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** 및 **TravelBus**는 Acute Technology Inc.의 상표 또는 등록 상표입니다.