애플리케이션 노트

Binho Supernova에서 NVMe-MI를 위한 단일 포트 I2C/SMBus/I3C 동작

AN0014

문서 정보

키워드	AN0014, Supernova, I3C, I2C, SMBus, MCTP, 클럭 스트레칭(clock stretching), 레거시 I2C, tTIMEOUT, NVMe-MI, ARP, 핫 조인(Hot-Join), PIC18F16Q20, pyCosmicSDK
요약	많은 SMBus 및 MCTP 디바이스는 응답을 준비하는 동안 SCL을 low로 유지합니다. Supernova의 I3C 엔진은 이를 기다릴 수 없으므로, 기존에는 I3C 포트에서 이러한 타깃의 비트가 손실되었습니다. 어댑터 펌웨어 4.6.0과 pyCosmicSDK 1.6.0은 I3C 포트의 레거시 I2C 전송이 SMBus 35 ms 타임아웃까지 타깃을 기다리는 모드를 추가합니다. 이 애플리케이션 노트는 이 모드를 설명하고, 이 모드로 스트레칭 타깃을 읽고 쓰는 스크립트와, 측정에 사용한 레퍼런스 타깃의 펌웨어를 제공합니다. 타깃이 0 ms에서 30 ms까지 스트레칭하는 조건에서 100 kHz와 400 kHz로 세 번 실행하여 3600회 중 3600회의 읽기가 정확했으며, 모드의 상한인 약 750 kHz까지 스트레칭된 모든 읽기가 정확했습니다. 이 스크립트는 또한 PCI-SIG 대역 외 관리 사양의 SMBus에서 I3C Basic으로의 검출 흐름을 어댑터가 가능한 범위까지 실행합니다.

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

개정 1.0
6 NOVEMBER 2026

1 소개

I2C 타깃은 주소에 ACK를 보낸 후 응답하기 전에 시간을 벌기 위해 SCL을 low로 유지할 수 있습니다. 컨트롤러는 SCL이 해제될 때까지 기다려야 합니다. I2C 사양은 이를 클럭 스트레칭(clock stretching)이라고 부르며, SMBus는 최대 35 ms(tTIMEOUT)까지 이를 허용합니다. SMBus 및 MCTP 디바이스는 이를 일상적으로 사용합니다. 값을 가져오는 중인 관리 컨트롤러는 값이 준비될 때까지 클럭을 유지합니다.

Supernova의 I3C 포트는 I3C 디바이스와 같은 버스에 있는 레거시 I2C 타깃을 주소 지정할 수 있습니다. 그러나 I3C 엔진은 타깃이 SCL을 low로 유지하는 동안에도 클럭을 계속 출력하므로, 스트레칭 타깃은 비트를 놓치고 전송은 잘못된 데이터를 반환하거나 ACK가 없습니다. 어댑터 펌웨어 4.6.0과 pyCosmicSDK 1.6.0은 I3C 포트의 레거시 I2C 전송에 대해 이 문제를 해결하는 모드를 추가합니다.

1.1 범위

이 문서는 다음을 다룹니다.

- 클럭 스트레칭 모드가 무엇을 바꾸는지, 그리고 모드가 켜져 있는 동안 무엇을 거부하는지
- 모드가 동작하는 SCL 속도와 요청 값에 얼마나 가까운지
- 이 모드를 통해 스트레칭 타깃을 읽고 쓰는 스크립트
- 설정 가능한 시간 동안 스트레칭하는 레퍼런스 타깃과 그 펌웨어
- 해당 타깃에 대한 측정 결과
- NVMe-MI에 사용되는 SMBus에서 I3C Basic으로의 검출 흐름과, 그 단계 중 어댑터가 수행하는 단계

ARP Get UDID의 PEC 검사를 넘어서는 SMBus 프레이밍(블록 읽기 및 쓰기, quick command), MCTP 또는 NVMe-MI 메시지, Supernova의 별도 I2C 포트는 다루지 않습니다.

1.2 적용 디바이스

표 1. 적용 디바이스

역할	부품	식별자	근거
컨트롤러	Binho Supernova	하드웨어 리비전 C, 펌웨어 4.6.1, I3C HV 포트	측정, 이 문서
컨트롤러	Binho Supernova	펌웨어 4.5.1 이하	`FW_UNSUPPORTED_COMMAND`로 모드를 거부
컨트롤러	Binho Pulsar	모두	해당 없음: Pulsar에는 I3C 포트가 없음
레퍼런스 타깃	Microchip PIC18F16Q20 Curiosity Nano	펌웨어 `stretch_target_pic18q20.c`	측정, 이 문서

1.3 관련 문서

표 2. 관련 문서

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

2 시스템 개요

모드가 꺼져 있으면 I3C 포트의 `legacy_i2c_read` 와 `legacy_i2c_write` 는 I3C 엔진이 수행하며, I3C 엔진은 자체 일정에 따라 SCL을 구동합니다. 모드가 켜져 있으면 어댑터의 펌웨어가 이러한 전송을 소프트웨어로 클럭하고, 해제하는 모든 단계 후에 SCL을 확인합니다. 타깃이 SCL을 low로 유지하면 펌웨어는 기다립니다.

이 모드의 사용 방식은 다음 네 가지 사실로 정해집니다.

- **전송별 옵션이 아니라 모드입니다.** `enable_i2c_clock_stretching(frequency_hz)` 는 이후의 모든 레거시 I2C 전송에 대해 모드를 켜고, `disable_i2c_clock_stretching()` 은 모드를 끕니다. `initialize` 와 `configure` 도 모드를 끕니다.
- **I3C 트래픽을 배제합니다.** 모드가 켜져 있는 동안 SDR, HDR, CCC 호출과 `init_bus` , `reset_bus` 는 상태 `FW_I3C_I2C_CLOCK_STRETCHING_ACTIVE` 와 함께 `I3cError` 를 발생시킵니다. I3C 디바이스와 통신하기 전에 모드를 끄십시오.
- **SMBus 타임아웃을 적용합니다.** 35 ms보다 긴 스트레칭은 `FW_BUS_TIMEOUT` 으로 전송을 종료하며, 버스는 해제됩니다.
- **속도는 소프트웨어 클럭이 정합니다.** 이 호출은 10 kHz에서 1 MHz까지를 받아들이고 계산한 속도를 반환합니다. 100 kHz에 대해 97975 Hz, 400 kHz에 대해 391644 Hz이며, 버스상의 실제 값과 1 % 이내입니다. 400 kHz를 넘으면 반환된 속도와 실제 값은 최대 7 % 차이가 나며, 600 kHz 요청은 615 kHz로 동작합니다 (9.2절). 클럭은 약 750 kHz가 상한이므로 800 kHz와 1 MHz 요청은 모두 그 속도로 동작합니다.

3 하드웨어 설정

3.1 필요 장비

표 3. 필요 장비

항목
Binho Supernova, 펌웨어 4.6.0 이상
USB-C 케이블
SCL을 스트레칭하는 I2C 또는 SMBus 타깃, 또는 3.4절의 레퍼런스 타깃
타깃 보드에 없는 경우 SCL과 SDA의 풀업 저항

3.2 연결

표 4. Supernova I3C HV 포트와 타깃 간 연결

Supernova I3C HV 포트	타깃	필수 여부
SCL	SCL	필수
SDA	SDA	필수
GND	GND	필수
VTARG	타깃 I/O 전원	선택, 타깃이 어댑터에서 전원을 받는 경우

3.3 버스 고려 사항

I3C 포트에는 SCL 풀업이 없습니다. I2C 버스에는 두 라인 모두에 풀업이 필요합니다. 저희 벤치에서는 1 kOhm으로 설정하고 유휴 상태로 둔 Binho Pulsar의 I2C 포트에서 풀업을 공급했습니다. 버스 전압에 연결한 1 kOhm에서 2.2 kOhm 저항 한 쌍도 같은 역할을 합니다. SCL에 풀업이 없으면 펌웨어가 모든 상승 에지를 기다리므로 보고되는 속도가 크게 떨어집니다.

> **주의:** I3C HV 포트는 `set_voltage` 에 따라 1200 mV에서 3300 mV 사이의 전압을 선택합니다. 3.3 V를 견디지 못하는 타깃을 연결하기 전에 타깃의 I/O 전압으로 설정하십시오.
>
> **주의:** 버스에는 컨트롤러가 하나만 있어야 합니다. 두 번째 어댑터가 풀업을 공급하는 경우 그 어댑터에서 전송을 시작하지 마십시오.

3.4 레퍼런스 타깃

모드를 측정하려면 스트레칭 시간을 설정할 수 있는 타깃이 필요했습니다. 에셋 아카이브에 포함된 `stretch_target_pic18q20.c` 는 PIC18F16Q20 Curiosity Nano를 그러한 타깃으로 만듭니다. 이 타깃의 I2C 페리페럴은 주소가 일치할 때마다 프로그래밍한 시간 동안 SCL을 유지하며, 이는 바쁜 SMBus 디바이스의 동작과 같습니다.

표 5. 레퍼런스 타깃 프로토콜, 주소 0x2A

전송	효과
`0xF0`, 하위 바이트, 상위 바이트 쓰기	이후 주소가 일치할 때마다 그 마이크로초만큼 스트레칭; 0은 스트레칭을 끔
읽기	모든 바이트에 대해 `0xA5`를 반환
그 밖의 쓰기	받아들이고 버림

SCL과 SDA는 RB6과 RB5에 있습니다. MPLAB XC8로 빌드하고 MPLAB X 또는 Curiosity Nano의 드래그 앤 드롭 프로그래머로 프로그래밍하십시오.

4 소프트웨어 설정

pyCosmicSDK(최신 릴리스)를 설치합니다.

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

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

5 클럭 스트레칭 모드

5.1 사용 방법

호출만 따로 보면 다음과 같습니다.

```
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()
```

레지스터 읽기, 즉 SMBus Read Byte 또는 Read Word 패턴은 레지스터를 `subaddress`로 전달합니다. `i3c.legacy_i2c_read(0x0B, 2, subaddress=bytes([0x09]))`.

5.2 상태 코드

표 6. 모드에 고유한 상태 코드

상태	발생 위치	의미
`FW_UNSUPPORTED_COMMAND`	enable, disable	펌웨어 4.5.1 이하
`FW_INVALID_PARAMETER`	enable	속도가 10 kHz에서 1 MHz 범위 밖
`FW_INTERFACE_NOT_INITIALIZED`	enable, disable	I3C 컨트롤러가 기동되지 않음
`FW_I3C_I2C_CLOCK_STRETCHING_ACTIVE`	SDR, HDR, CCC, `init_bus`, `reset_bus`	모드가 켜져 있음; 먼저 끌 것
`FW_BUS_TIMEOUT`	레거시 읽기 또는 쓰기	타깃이 SCL을 35 ms보다 길게 유지함

5.3 SMBus 타임아웃

펌웨어 검증 중 레퍼런스 타깃을 40 ms 스트레칭하도록 설정했을 때, 100회의 읽기는 `FW_BUS_TIMEOUT` 50회와 `FW_I3C_NACK_ADDRESS` 50회를 반환했고, 가장 긴 것은 46 ms였으며, 어댑터는 계속 응답했습니다. 주소 NACK는 타깃에서 발생하며, 타깃이 아직 40 ms를 세는 동안 다음 주소를 놓치기 때문입니다.

6 절차

1. 3.2절과 같이 타깃과 풀업을 연결합니다. 타깃에 전원을 공급합니다.
2. 타깃에서 읽습니다. 첫 번째 줄에 펌웨어와 적용 중인 속도가 표시됩니다.

```
$ 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. 타깃에 씁니다.

```
$ 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. 레퍼런스 타깃으로 스트레칭 시간을 스윕합니다. 각 시간마다 스크립트는 타깃의 스트레칭을 설정하고, 16바이트를 100회 읽으며, 모든 바이트를 확인합니다.

```
$ 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 kHz로 반복합니다.

```
$ 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
```

읽기당 중앙값 시간은 스트레칭 시간에 따라, 대략 스트레칭 시간만큼 늘어납니다. 어댑터가 매번 기다렸다는 뜻입니다. 스크립트는 종료하기 전에 타깃의 스트레칭을 0으로 되돌립니다.

7 SMBus에서 I3C Basic으로의 검출 흐름

SMBus에서 I3C Basic으로 전환하는 NVMe-MI 디바이스는 한 쌍의 핀을 공유하므로, 호스트는 버스를 변경하기 전에 각 디바이스가 어떤 프로토콜을 사용하는지 알아내야 합니다. 호스트가 따르는 흐름은 PCI-SIG Engineering Change Notice, Chapter 12, Architectural Out-of-Band Management에 정의되어 있습니다. `smbus_stretch.py` 의 `handoff` 명령은 이 흐름을 Supernova I3C 포트에서 실행합니다. 표 7은 단계와 각 단계의 수행 주체를 보여 줍니다.

표 7. 검출 흐름 단계

단계	흐름	수행 주체
1	SMBus 전압에서 7Eh로 DISHJ를 포함한 DISEC 브로드캐스트; ACK는 I3C Basic 디바이스가 있다는 뜻	스크립트. 7Eh 헤더만 오픈 드레인이며, CCC 바이트와 데이터는 I3C 속도의 푸시풀입니다. NACK는 `FW_I3C_NACK_ADDRESS` 로 보고됩니다
2	정적 주소와 61h(ARP Get UDID)에서 SMBus 디바이스 탐색	스크립트, 클럭 스트레칭 모드를 켠 상태. 스크립트가 PEC를 확인합니다
3	SMBus 전용 디바이스를 찾음: SMBus 인터페이스를 리셋하고 SMBus 모드를 유지	스크립트: `hold_scl_low` 가 T2wrst(`--t2wrst-us`) 동안 SCL을 low로 유지합니다. SMRST#가 있는 디바이스는 대신 GPIO로 리셋할 수 있습니다
4	DISHJ를 포함한 DISEC를 다시 보낸 후, SMBus 풀업 끄기 및 I3C Basic 풀업 켜기	DISEC는 스크립트. 어댑터의 SDA 풀업은 SMBus 단계 동안 `set_sda_pull_up` (`--sda-pull-up`)으로 설정되며, 그 후에는 I3C 엔진을 따릅니다. 어댑터에는 SCL 풀업이 없으므로 외부 풀업은 수동으로 전환합니다
5	Tsmb2i3c를 기다리고, I3C Basic 전압으로 전환한 후, ENHJ를 포함한 ENEC 브로드캐스트	스크립트. `--t-smb2i3c-ms` 와 `--i3c-voltage-mv` 는 사양과 디바이스에서 정합니다
6	디바이스가 핫 조인(Hot-Join)함	펌웨어: 펌웨어가 동적 주소를 직접 할당하고 스크립트가 이를 출력합니다

클럭 스트레칭 모드는 모든 CCC를 거부하므로(5.2절), 스크립트는 2단계에서만 모드를 켜고 4단계 전에 다시 끕니다.

Saleae 분석기에서 흐름을 보려면 `capture_flow.py` 를 사용합니다. 이 스크립트는 Logic 2의 자동화 서버를 통해 캡처를 시작하고, `--` 뒤의 인수로 `handoff` 를 실행한 다음, Logic 2의 I2C 분석기와 각 단계의 이름을 별도의 행에 표시하는 `flow_labels` 확장과 함께 캡처를 저장합니다. `--step-delay-s` 는 각 단계를 타임라인에서 찾을 수 있도록 단계 사이에 간격을 둡니다.

> **참고:** SMBus 전용 분기만 실행했으며, 9.1절의 벤치에서 펌웨어 4.6.1로 6회 중 6회 동일한 결과를 얻었고, 그중 한 번은 어댑터 리셋 후였습니다. 1단계는 NACK, 2단계는 0x2A에서 레퍼런스 타깃을 찾았고 ARP 디바이스는 없었으며, SMBus 리셋은 35000 us를 요청했을 때 SCL을 35000에서 35777 us 동안 low로 유지했습니다. 3단계에서 6단계까지는 7Eh에 ACK를 보내고 핫 조인하는 디바이스가 필요하며, ARP 및 PEC 경로에는 ARP를 지원하는 디바이스가 필요합니다. 둘 다 벤치에 없었습니다. PEC 루틴은 CRC-8/SMBUS check value로만 검증했습니다.

```
$ 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 참조

표 8. smbus_stretch.py 옵션

옵션	기본값	의미
`--serial`	any	Supernova 시리얼 번호
`--voltage-mv`	3300	I3C 포트 전압, HV 커넥터의 경우 1200에서 3300
`--khz`	100	SCL 속도, 10에서 1000; 최대 약 750 kHz로 동작

표 9. smbus_stretch.py 명령

명령	인수	동작
`read`	주소, 길이, 선택적 `--register`	읽기, 선택적으로 레지스터 바이트 하나를 쓴 후
`write`	주소, 바이트	바이트를 씀
`sweep`	`--address`, `--count`, `--length`	레퍼런스 타깃 전용: 스트레칭 0, 50, 500, 2000, 10000, 30000 us에서 각각 `--count` 회 읽기
`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`	7절의 검출 흐름

모든 명령은 전송 전에 모드를 켜고 종료 전에 모드를 끄므로, 이후 I3C 포트를 I3C 트래픽에 사용할 수 있습니다.

9 측정 결과

9.1 테스트 구성

표 10. 테스트 구성

항목	값
어댑터	Binho Supernova, 하드웨어 리비전 C, 펌웨어 4.6.1(`FW-4.6.1`). 9.2절의 소크 테스트, SMBus 리셋 및 SDA 풀업 측정은 같은 기능의 4.6.0 사전 릴리스 빌드에서 얻은 것입니다
호스트 소프트웨어	PyPI의 pyCosmicSDK 1.6.0, Linux
버스	I3C HV 포트, 3.3 V, 유휴 상태의 Binho Pulsar(펌웨어 4.6.1)에서 SCL과 SDA에 1 kOhm 풀업
분석기	Acute TravelBus 로직 분석기, 800 MS/s, 임계값 1.6 V
타깃	PIC18F16Q20 Curiosity Nano, `stretch_target_pic18q20.c` , 주소 0x2A

9.2 결과

6절의 스윕은 두 속도와 여섯 가지 스트레칭 시간에 걸쳐 각 16바이트씩 1200회 읽었습니다. 세 번 실행했으며, 세 번째는 어댑터 리셋 후였습니다. 3600회 정확, 오류 없음.

같은 기능의 펌웨어 검증에서 사전 릴리스 빌드로 얻은 추가 결과:

- 1에서 64바이트의 무작위 읽기와 쓰기로 소크 테스트를 수행하고, 100회 전송마다 스트레칭을 0, 50 us, 500 us, 2 ms, 10 ms 중에서 변경했습니다. 100 kHz와 400 kHz에서 실행당 약 3400회의 스트레칭된 전송 중 잘못된 읽기 0회였으며, 전원을 껐다 켠 후의 실행도 포함됩니다.

펌웨어 4.6.1에서 요청별 SCL 속도. 400 kHz를 넘는 경우 50 us에서 30 ms까지 스트레칭마다 16바이트 읽기 20회:

표 11. 요청별 SCL 속도

요청	보고값	버스상 실측	tLOW / tHIGH	스트레칭된 읽기
100 kHz	97975 Hz	98.2 kHz	5105 / 5075 ns	1500회 중 1500회
400 kHz	391644 Hz	395.3 kHz	1600 / 935 ns	1500회 중 1500회
600 kHz	583657 Hz	615.4 kHz	855 / 755 ns	100회 중 100회
800 kHz	746268 Hz	749.1 kHz	750 / 585 ns	100회 중 100회
1 MHz	746268 Hz	749.1 kHz	750 / 585 ns	100회 중 100회

버스상 실측값은 속도별 약 9000개 SCL 주기의 중앙값입니다. 모든 속도에서 tLOW와 tHIGH는 해당 모드의 UM10204 최솟값보다 높게 유지됩니다. 400 kHz를 넘으면 반환된 속도는 실제 값을 따라가지 않습니다. 500 kHz 요청은 477707 Hz를 보고하고 467.3 kHz로 동작하며, 700 kHz는 652173 Hz를 보고하고

696.9 kHz로 동작하고, 600 kHz는 요청보다 2.6 % 높게 동작합니다. 400 kHz를 넘는 정확한 속도가 중요하다면 버스를 측정하십시오.

레퍼런스 타깃이 스트레칭하지 않을 때 400 kHz를 넘는 읽기는 두 번째 바이트에서 0xFF를 반환하며(각 속도에서 100회 중 0회), 반면 스트레칭된 읽기는 모두 정확합니다. 어댑터의 SCL은 두 경우 모두 같으며, 타깃 자체의 클럭 스트레칭만 꺼져 있습니다. 저희는 이것이 어댑터가 아니라, 스트레칭 없이는 두 번째 바이트를 제때 로드하지 못하는 레퍼런스 타깃의 펌웨어 때문이라고 판단합니다.

SMBus 리셋(`hold_scl_low`), 지점당 5회 유지, 사전 릴리스 펌웨어 `FW-4.5.1-b11a067e`, Acute에서 측정한 SCL low 시간. 분석기의 버퍼 경계에 걸친 유지는 약 1.5 ms 짧게 읽히므로 제외했습니다.

표 12. SMBus 리셋을 위한 SCL 유지

요청값	버스상 측정값	어댑터 보고값	유지 후 읽기
1 ms	999.7에서 1678 us	1000에서 1677 us	5회 중 5회
10 ms	9987에서 11006 us	10000에서 10998 us	5회 중 5회
35 ms	34996에서 35941 us	35000에서 35937 us	모드별 5회 중 5회, 리셋 후에도
100 ms	100013에서 100114 us	100000 us	5회 중 5회

유지 시간은 최대 약 1 ms 길어질 수 있고, 분석기의 클럭 대비 최대 0.13 % 짧아질 수 있으며, 이는 어댑터 내부 발진기의 허용 오차입니다. T2wrst는 이 여유를 두고 요청하십시오. 0 us와 1 s를 넘는 값은 `FW_INVALID_PARAMETER`로 거부되며, 펌웨어 4.5.1은 `FW_UNSUPPORTED_COMMAND`로 응답합니다.

SDA 풀업은 Pulsar의 풀업을 끈 상태에서 각 STOP에서 SCL 상승부터 SDA가 1.6 V를 넘을 때까지의 지연으로 측정했습니다. `OFF`에서 7813 ns, `ON`에서 5975 ns, `AUTO`에서 5965 ns였습니다. 세 설정 모두에서, 클럭 스트레칭 모드 사용 여부와 관계없이 읽기는 10회 중 10회였습니다.

이는 하나의 벤치에서 저희가 측정한 값이며, 사양이 아닙니다.

10 문제 해결

표 13. 문제 해결

증상	가능한 원인과 조치
enable 시 `FW_UNSUPPORTED_COMMAND`	펌웨어 4.5.1 이하. Supernova를 4.6.0 이상으로 업데이트하십시오.
보고된 속도가 요청보다 훨씬 낮음	SCL에 풀업이 없음. 풀업을 추가하십시오(3.3절).
`FW_I3C_I2C_CLOCK_STRETCHING_ACTIVE`	모드가 켜진 상태에서 I3C 호출을 했습니다. 먼저 `disable_i2c_clock_stretching`을 호출하십시오.
`FW_BUS_TIMEOUT`	타깃이 SCL을 35 ms보다 길게 유지했습니다. 타깃을 확인하십시오. 이것은 SMBus tTIMEOUT입니다.
레퍼런스 타깃을 재구성한 직후 `FW_I3C_NACK_ADDRESS`	타깃은 새 스트레칭을 적용하기 위해 I2C 페리페럴을 다시 시작합니다. 스크립트는 이를 위해 10 ms를 기다립니다.
모드가 꺼진 상태에서 스트레칭 타깃으로부터 잘못된 데이터	예상된 동작: I3C 엔진은 기다리지 않습니다. 모드를 켜십시오.
400 kHz를 넘는 속도로 enable 시 `FW_INVALID_PARAMETER`	4.6.0의 1 MHz 한도 이전의 펌웨어. Supernova를 업데이트하십시오.
400 kHz를 넘는 속도에서 `sweep` 중 `FW_I3C_I2C_NACK_BYTE`	레퍼런스 타깃이 스트레칭하지 않는 동안 재구성 쓰기를 놓쳤습니다. 저희 벤치에서 24회 스윕 중 1회. 스윕을 다시 실행하십시오.
Saleae Logic Pro에서 `capture_flow.py`가 잘못된 주소와 데이터를 디코딩함	디지털 임계값이 디바이스 기본값입니다. 3.3 V 버스에는 `--threshold 3.3`을 전달하십시오. 저희 Logic Pro 8에서는 같은 스윕에서 SCL이 high일 때 보이는 SDA 에지가 732개에서 50개로 줄었으며, 모두 START와 STOP이었습니다.
버스에 I3C Basic 디바이스가 있는데 `handoff` 1단계에서 NACK	디바이스가 설정한 전압에서 SMBus 모드가 아니거나 전원이 공급되지 않았습니다. `--voltage-mv`를 확인하십시오.

11 약어

표 14. 약어

용어	의미
I2C	Inter-Integrated Circuit
I3C	MIPI Improved Inter-Integrated Circuit
SMBus	System Management Bus
MCTP	Management Component Transport Protocol
SCL	직렬 클럭 라인(Serial clock line)
SDA	직렬 데이터 라인(Serial data line)
SDR	Single data rate, 기본 I3C 전송 모드
HDR	High data rate I3C 전송 모드
CCC	I3C common command code
tTIMEOUT	SMBus 클럭 low 타임아웃, 35 ms
ARP	SMBus Address Resolution Protocol
PEC	SMBus packet error code, CRC-8
UDID	SMBus unique device identifier
DISEC, ENEC	타깃 이벤트를 비활성화 및 활성화하는 I3C CCC
Hot-Join	타깃이 버스에 참여하여 동적 주소를 받는 I3C 이벤트
NVMe-MI	NVM Express Management Interface
T2wrst, Tsmb2i3c	PCI-SIG 검출 흐름의 타이밍 파라미터

12 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 `AN0014-assets.zip`에는 `smbus_stretch.py`, `capture_flow.py`, `flow_labels` Logic 2 확장, `test_flow_labels.py`, `stretch_target_pic18q20.c` 및 `LICENSE`가 포함되어 있습니다.

13 개정 이력

표 15. 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.
0.2	28 September 2026	NVMe-MI용으로 제목 변경. 최대 1 MHz 속도(실제 746 kHz). 7절 SMBus에서 I3C Basic으로의 검출 흐름과 `handoff` 명령 추가.
0.3	28 September 2026	검출 흐름에 `hold_scl_low`와 `set_sda_pull_up` 추가 및 그 측정값.
0.4	30 September 2026	펌웨어 4.6.1 및 PyPI의 pyCosmicSDK 1.6.0에서 다시 측정. 모든 요청에 대한 버스상 SCL 속도; 400 kHz를 넘으면 반환되는 속도는 근삿값. `--serial`이 pyCosmicSDK 1.6.0에서 동작.
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. **PIC**, **MPLAB** 및 **Curiosity**는 Microchip Technology Incorporated의 상표 또는 등록 상표입니다. **Acute** 및 **TravelBus**는 Acute Technology Inc.의 상표 또는 등록 상표입니다. **PCI-SIG**는 PCI-SIG의 등록 상표입니다. **NVMe** 및 **NVMe-MI**는 NVM Express, Inc.의 상표입니다.