binho

애플리케이션 노트

Microchip dsPIC33AK에서 CMIS I3C 타깃 구축 및 Binho Supernova를 이용한 테스트

AN0019

문서 정보

키워드	AN0019, Supernova, CMIS, OIF, I3C, MIPI I3C Basic, SDR, I3C 타깃, dsPIC33AK, dsPIC33AK256MPS306, MCC Melody, DMA, ENTDAA, IBI, GETSTATUS, 광 모듈
요약	CMIS 광 모듈은 I3C 타깃으로서 호스트에 응답하며, 호스트는 I2C 호스트와 같은 방식으로 이를 읽습니다. 즉, 바이트 주소를 쓰고, 반복 START를 보낸 다음, 읽습니다. dsPIC33AK256MPS306 I3C 주변 장치에서는 그 쓰기에 대한 인터럽트가 읽기 헤더에 이미 응답한 후에 도착하므로, 이 인터럽트에서 응답을 준비(arm)하는 타깃은 그러한 읽기를 모두 NACK합니다. 이 노트는 바이트 주소를 제때 얻을 수 있는 지점, 펌웨어가 대신 수신 DMA에서 준비하는 방법, 그리고 Binho Supernova에서 측정한 수치(4~128바이트 읽기 60회 중 60회, 인밴드 인터럽트를 켠 상태에서 300회 중 300회, 리셋 후에도 동일)를 보여 줍니다. 호스트 측 테스트 스크립트를 제공하고, 포팅 시 미리 고려해야 하는 주변 장치와 드라이버의 동작을 나열합니다.

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

개정 1.0
6 NOVEMBER 2026

> **검증 범위**. 섹션 8의 모든 결과는 섹션 3에서 설명하는 벤치에서 측정했습니다. 타깃 펌웨어는 자산 아카이브에 포함되어 있지 않습니다. techsupport@binho.io로 문의하십시오.

1 소개

CMIS 모듈은 256바이트 윈도를 노출합니다. 이 윈도는 Lower Memory와 Upper Memory의 128바이트 페이지 하나로 구성되며, 페이지는 `00h:127`의 페이지 바이트로 선택됩니다. I2C에서 호스트는 바이트 주소를 쓴 다음 반복 START로 읽습니다. CMIS용으로 정의 중인 I3C 관리 인터페이스는 이 액세스 패턴을 유지하므로, I3C 타깃 주변 장치로 구축한 모듈은 같은 전송 안에서 불과 몇 비트 시간 전에 도착한 바이트 주소에 대한 읽기에 응답해야 합니다.

주변 장치가 각 바이트를 도착하는 대로 펌웨어에 넘겨준다면 이는 쉬운 일입니다. 그러나 dsPIC33AK256MPS306 I3C 주변 장치는 커맨드와 응답 기반으로 동작합니다. 수신된 전송은 응답으로 한 번만 보고되고, 읽기에는 송신 커맨드로 응답하는데, 이 커맨드는 읽기 헤더가 도착할 때 이미 유효해야 합니다. 이 노트는 이 두 가지를 맞물리게 하는 방법을 다룹니다.

1.1 범위

- dsPIC33AK의 CMIS 레지스터 모델: 페이징, 섹션 8.2.15의 미지원 페이지 대체, Appendix B.1.2의 128바이트 세그먼트 내 랩어라운드, 그리고 Flags Summary 바이트 `00h:4`를 전달하는 IBI.
- 2.5 MHz 푸시풀에서 쓰기 + Sr + 읽기(결합)와 쓰기, STOP, 읽기(분할)에 대한 응답.
- 동적 주소 할당(DAA), RSTDAA 및 Direct GET CCC.
- 이 노트의 모든 검사를 Supernova에서 실행하는 호스트 스크립트.

다루지 않는 내용: CDB, 펌웨어 다운로드, HDR 모드, 그리고 CMIS 제어 레지스터의 모듈 측 의미. 레지스터 이미지는 페이지 선택과 호스트가 쓰는 바이트를 제외하면 정적입니다.

1.2 적용 디바이스

표 1. 적용 디바이스

제품군	부품 예	식별자	근거
I3C 탑재 dsPIC33AK MPS	dsPIC33AK256MPS306 (EV02E57A DIM)	PID `02 36 43 4D 49 53`	테스트됨
I3C 미탑재 dsPIC33AK MP	dsPIC33AK512MPS506 및 제품군	-	해당 없음: I3C 주변 장치 없음
동일한 I3C 주변 장치를 가진 기타 부품	-	-	테스트되지 않음

1.3 관련 문서

표 2. 관련 문서

문서	제목
OIF-CMIS-05.3	Common Management Interface Specification, 개정 5.3
MIPI I3C Basic v1.1.1	MIPI I3C Basic Specification, 버전 1.1.1
DS70005539	dsPIC33AK MPS 제품군 데이터시트, 섹션 24 (I3C)
AN0009	Binho Supernova를 사용한 I3C를 통한 CMIS 광 모듈 관리

2 시스템 개요

Supernova는 I3C 컨트롤러이자 버스상의 유일한 컨트롤러입니다. dsPIC33AK는 정적 주소 `0x50` (CMIS 모듈 주소)을 갖는 타깃입니다. 컨트롤러는 ENTDAA로 동적 주소를 할당한 다음(벤치에서는 `0x08`을 할당) SDR로 읽고 씁니다.

펌웨어의 구조를 결정하는 단 하나의 사실은 결합 읽기의 바이트 주소가 언제 보이게 되는가입니다. I3C 인터럽트를 끈 상태에서 100 MHz 타이머를 기준으로 주변 장치를 폴링하고 로직 분석기와 시점을 맞추어 측정한 결과는 다음과 같습니다.

표 3. 2.5 MHz 푸시풀에서의 결합 읽기, START 기준 시간

이벤트	시간
바이트 주소의 T 비트	22.3 µs
수신 FIFO에 들어온 바이트 주소	23.8 µs
반복 START	24.1 µs
읽기 헤더의 ACK 슬롯	~27.4 µs
쓰기에 대한 응답 인터럽트	ACK 슬롯 이후

쓰기의 응답 인터럽트에서 준비한 송신 커맨드는 항상 늦으며, 타깃은 읽기를 NACK합니다. 바이트 주소 자체는 ACK 슬롯보다 3.6 µs 앞서 얻을 수 있고, 커맨드를 준비하는 데는 1.15~1.26 µs가 걸립니다. 그래서 펌웨어는 수신 DMA에서 준비합니다. 수신된 워드를 RAM으로 옮기는 채널이 I3C 인터럽트보다 높은 우선순위로 워드마다 인터럽트를 발생시키고, 그 인터럽트가 첫 번째 바이트로부터 송신 커맨드를 준비합니다.

나머지는 모두 이 주변 장치와 MCC Melody 드라이버의 세 가지 특성에서 비롯됩니다.

- 송신 커맨드 하나는 읽기 한 번에 응답합니다. 펌웨어는 바이트 주소부터 세그먼트 끝까지 128바이트 런을 준비하며, 컨트롤러가 읽기를 일찍 끝냅니다.
- 송신 버퍼는 8워드(32바이트)를 보관합니다. 런의 나머지는 메인 루프에서 공급합니다. 32바이트를 보내는 데 걸리는 시간(약 115 µs)보다 오래 메인 루프를 붙잡아 두는 작업이 있으면 읽기가 32바이트에서 잘립니다.

- 수신을 다시 준비하면 수신 FIFO가 리셋되며, 이때 준비된 송신 커맨드도 버려집니다. 수신은 읽기 후에만 다시 준비합니다.

3 하드웨어 설정

3.1 필요 장비

- Binho Supernova, 펌웨어 4.6.2, I3C HV 포트.
- EV02E57A dsPIC33AK256MPS306 DIM을 장착한 Microchip EV74H48A Curiosity 플랫폼. 이 보드의 PKoB4가 부품을 프로그래밍하고 UART를 전달합니다.
- 선택 사항: SCL과 SDA에 연결한 로직 분석기. 이 노트의 캡처는 Saleae Logic으로 12 MS/s에서 수집했습니다.

3.2 연결

표 4. Supernova와 타깃 간 연결

Supernova I3C 포트	dsPIC33AK DIM	필수 여부
SCL	I3CSCL1 (RC5)	예
SDA	I3CSDA1 (RC4)	예
GND	GND	예
I3C 레일, 3.3 V	연결하지 않음, DIM은 Curiosity 보드에서 전원 공급	-
-	UART1 TX (RD0), PKoB4 가상 COM 포트로 출력	선택

3.3 버스 고려 사항

버스는 3.3 V로 동작합니다. 푸시풀 2.5 MHz는 Supernova가 제공하는 가장 낮은 속도이며, 여기에서 측정한 유일한 속도입니다. 속도가 빨라지면 표 3의 3.6 µs 윈도가 짧아지며, 12.5 MHz에서는 커맨드를 준비하는 데 걸리는 시간보다 짧아집니다.

4 소프트웨어 설정

타깃 펌웨어:

- MPLAB X v6.35 및 MPLAB IPE, XC-DSC v4.00, dsPIC33AK-MP DFP 1.4.260.
- 섹션 5.6의 로컬 변경을 적용한 MCC Melody I3C 타깃 드라이버.

호스트:

- Python 3 및 pycosmicsdk(최신 릴리스), 그리고 이 노트의 에셋에 포함된 `cmis_target_check.py`.

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

5 타깃 펌웨어 설계

5.1 레지스터 모델

Lower Memory와 Upper Memory 페이지 `00h`는 CMIS 5.3 Appendix C의 예제 이미지에서 가져왔습니다. `00h:127`의 페이지 선택은 `00h`와 `01h`를 받아들이며, 그 밖의 값은 `00h`로 대체됩니다(섹션 8.2.15). 각 세그먼트는 연속으로 두 번 저장되고, 여기에 1~3바이트씩 밀린 사본도 저장되므로, 어떤 바이트 주소에서 시작하든 128바이트 런은 정렬된 버퍼 하나가 되며, 세그먼트 내의 B.1.2 랩어라운드에는 코드가 필요하지 않습니다.

5.2 식별 정보

표 5. 컨트롤러가 보는 식별 정보

항목	값
정적 주소	`0x50`
PID	`02 36 43 4D 49 53` (MIPI 제조사 `0x11B`, 부품 `"CMIS"`)
BCR	`0x17`: IBI 지원, IBI가 1바이트를 전달, 최대 데이터 속도 제한 있음
DCR	`0x00`

5.3 읽기 준비

1. 수신 DMA 채널은 수신된 각 워드를 `rxbuf`로 옮깁니다. 완료 인터럽트가 활성화되어 있으며 우선순위는 5로, 우선순위 4인 I3C 인터럽트보다 높습니다.
2. 쓰기의 첫 번째 워드에서 DMA 인터럽트는 `rxbuf[0]`을 바이트 주소로 취하고, 그 주소부터의 런에 대한 송신 커맨드를 준비합니다.
3. 쓰기의 응답이 도착했을 때 주소만 전달한 경우, 커맨드는 준비된 상태로 유지되고 수신은 읽기가 완료된 후 다시 준비됩니다.
4. 쓰기가 데이터를 전달한 경우, 확장 송신 버퍼를 리셋하여 커맨드를 취소하고 수신을 즉시 다시 준비합니다.
5. 읽기 헤더 시점에 여전히 준비된 것이 없으면 주변 장치는 읽기 요청을 발생시킵니다. 펌웨어는 늦게 준비된 것을 모두 버리고 `I3C1_Target_ReceiveCancel()`로 수신을 다시 준비합니다. 이 경우에는 쓰기에 대한 응답이 결코 오지 않기 때문입니다.

5.4 긴 읽기 공급

송신 버퍼는 미리 8워드로 채워집니다. `I3C1_Target_TxRefill()`이 메인 루프에서 나머지를 공급하며, 이를 실행하는 몇 개의 명령어 동안 I3C 및 DMA 인터럽트를 마스크합니다. 브링업 펌웨어의 UART 상태 줄은

기본적으로 꺼져 있습니다. 9600 baud에서의 블로킹 `printf`가 초당 약 110 ms 동안 메인 루프를 붙잡아 128바이트 읽기를 32바이트에서 잘랐기 때문입니다.

5.5 이벤트

IBI가 활성화되어 있는 동안 IBI는 1초에 한 번 `00h:4`를 필수 데이터 바이트로 전달합니다. 주변 장치는 I3C Basic의 규정대로 IBI가 활성화된 상태로 리셋에서 빠져나오며, 해당 활성화 비트에 쓴 0을 유지하지 않습니다. IBI를 끄려면 DISEC를 사용해야 합니다.

5.6 Melody 드라이버 변경 사항

Table: `mcc/`의 로컬 변경 사항, `binho:`로 표시

변경	이유
`BufferTransmit`가 8워드를 미리 채우고, `I3C1_Target_TxRefill()`이 나머지를 공급	8워드를 넘으면 DMA 경로가 워드 7을 두 번 쓰고 더 진행하지 않아 커맨드가 거부되었음
`I3C1_Target_ReceiveCancel()`	쓰기 + Sr + NACK된 읽기 헤더 이후에는 쓰기에 대한 응답이 오지 않음. 드라이버가 수신을 진행 중으로 표시한 채로 두어 이후의 모든 수신을 거부했음
완료 콜백 전에 `rxInProgress` 클리어	콜백이 수신을 다시 준비할 수 있도록 하기 위함
정적 주소 `0x50`	CMIS 모듈 주소

6 절차

단계 1: 펌웨어 폴더에서 타깃을 빌드하고 프로그래밍합니다.

```
make flash
```

빌드는 MPLAB IPE의 `Program Succeeded.`로 끝납니다.

단계 2: 표 4와 같이 Supernova I3C 포트를 DIM에 연결합니다.

단계 3: 쓰기 + Sr + 읽기로 128바이트를 60회 읽습니다.

```
python cmis_target_check.py reads --length 128 --count 60
```

```
init bus: FW_I3C_BUS_INIT_NACK_RSTDAA
target DA 0x08, PID 02 36 43 4d 49 53
............................................................
reads 60/60  (combined, 128 B, IBI off)
```

`init bus` 줄은 리셋 후 첫 번째 RSTDAA가 NACK되었음을 나타냅니다(섹션 8.3). 스크립트는 다시 초기화합니다.

단계 4: IBI를 활성화한 상태로 같은 작업을 수행합니다(읽기 전에 ENEC).

```
python cmis_target_check.py reads --length 128 --count 300 --ibi
```

```
reads 299/300  (combined, 128 B, IBI on)
```

점으로 된 줄에서 `F`는 오류가 발생한 읽기를, `B`는 잘못된 데이터를 반환한 읽기를 나타냅니다.

단계 5: 데이터 쓰기와 읽기 혼합: 라운드마다 `7Fh=00h`와 `7Eh=00h`를 쓴 다음 `7Fh`(1바이트)와 `80h`(16바이트)를 읽고 이미지와 대조합니다.

```
python cmis_target_check.py mixed --count 60
```

```
rounds 60/60
```

단계 6: 주소 리셋 및 재할당:

```
python cmis_target_check.py daa --cycles 10
```

```
0: RSTDAA SUCCESS, GETSTATUS at old DA FW_I3C_NACK_ADDRESS, read after ENTDAA ok
...
cycles 10/10
```

단계 7: 식별 정보 및 Direct GET CCC:

```
python cmis_target_check.py ccc --getstatus 100
```

```
GETBCR   SUCCESS               17
GETDCR   SUCCESS               00
GETPID   SUCCESS               02 36 43 4d 49 53
GETMXDS  SUCCESS               00 00 00 00 01
GETMRL   FW_I3C_NACK_ADDRESS
GETMWL   SUCCESS               ff
GETSTATUS 100/100  {'SUCCESS': 100}
```

Direct GET은 NACK된 `7Eh` 헤더로 인해 실패할 수 있습니다(섹션 8.3). 여기서는 GETMRL이 실패했습니다. 다시 실행하십시오.

7 참조

Table: `cmis_target_check.py` 명령

명령	동작
`reads --length N --count N [--split] [--ibi]`	`00h`, `10h`, `7Ch`, `80h`, `A0h`, `FCh`를 차례로 읽고 이미지와 비교합니다. `--split` 이 없으면 결합 읽기이며, `--ibi`는 먼저 ENEC를 보내고, 그렇지 않으면 DISEC를 보냅니다
`mixed --count N`	`7Fh`와 `7Eh`에 데이터를 쓴 다음, `7Fh`와 `80h`를 결합 읽기로 읽어 검사합니다
`ccc [--getstatus N]`	GETBCR, GETDCR, GETPID, GETMXDS, GETMRL, GETMWL, 이어서 GETSTATUS를 N회
`daa --cycles N`	RSTDAA, 이전 주소가 NACK하는지 확인, ENTDAA, 읽기 확인
`i2c`	주소 할당 전에 I2C 모드로 `0x50`에서 `00h:0-3`과 `00h:46-48`을 읽습니다

공통 옵션: `--serial`, `--push-pull` (기본값 `PUSH_PULL_2_5_MHZ_DC_25`), `--voltage-mv` (기본값 3300), `--image` (기본값은 스크립트와 같은 위치의 `cmis_image.h`). 각 명령은 RSTDAA와 ENTDAA로 시작하며, PID의 `"CMIS"`로 타깃을 찾습니다.

8 측정 결과

8.1 테스트 구성

표 6. 테스트 구성

항목	버전
Supernova	시리얼 4F970418, 펌웨어 4.6.2
타깃	EV74H48A에 장착한 EV02E57A DIM
툴체인	XC-DSC v4.00, DFP 1.4.260, MPLAB X v6.35
호스트	Ubuntu 26.04; 세션 1 및 2: Python 3.13, binhosupernova 4.2.0; 세션 3: Python 3.14.4, pycosmicsdk 1.9.1
버스	3.3 V, 푸시풀 2.5 MHz, 오픈 드레인 1 MHz

8.2 결과

세 번의 세션을 진행했습니다. 세션 2는 프로그래밍(리셋) 후에 시작했고, 세션 3(2026년 11월 6일)은 모든 테스트 전에 타깃을 프로그래밍했으며 호스트에서는 pycosmicsdk를 사용했습니다.

표 7. 결과

테스트	세션 1	세션 2	세션 3
결합 읽기, 4 / 16 / 32 / 128 B, IBI 끔	각 60/60	60/60 (4 및 128 B)	각 60/60
결합 읽기, 128 B, IBI 켬	300/300	300/300	299/300
분할 읽기, 4 / 128 B	각 60/60	60/60 (128 B)	60/60 (128 B)
데이터 쓰기 및 결합 읽기, 데이터 검사	60/60 라운드	60/60 라운드	60/60 라운드
RSTDAA, 이전 주소 NACK, ENTDAA, 읽기	10/10	10/10	10/10
GETSTATUS	97/100	94/100	100/100

수신 DMA 준비를 도입하기 전에는 같은 결합 읽기가 60회 중 0회 성공했습니다.

8.3 호스트가 예상해야 하는 동작

- **타깃 리셋 후 첫 번째 전송은 NACK됩니다**. 5회의 리셋 각각 이후, 첫 번째 `7Eh` 헤더는 NACK되었고 두 번째는 ACK되었습니다. DS70005539 섹션 24.4.5에는 모듈이 SCL 클록을 감지한 후에야 리셋에서 빠져나온다고 기술되어 있습니다. 한 번만 프로브하고 NACK를 "I3C 타깃 없음"으로 간주하는 호스트는 이 모듈을 놓치게 됩니다. 프로브를 재시도하십시오.
- **일부 Direct GET CCC는 NACK된 `7Eh` 로 인해 실패합니다**. 한 세션에서는 Direct GET 뒤에 오는 `7Eh` 헤더 약 7개 중 1개가 NACK되었습니다(GET 1200회 중 173회 실패, 173회 모두 와이어상에서 `7E/W NACK` 로 확인됨). 다른 세션에서는 500회 중 500회가 통과했습니다. 프라이빗 읽기는 영향을 받지 않았습니다. NACK로 실패한 GET은 재시도하십시오.
- **GETMRL은 255를 보고하며 SETMRL은 적용되지 않습니다**. 펌웨어는 128을 설정하지만, 주변 장치가 자체적으로 GETMRL에 255로 응답합니다. SETMRL 16 이후에도 32바이트 읽기는 여전히 32바이트를 반환합니다.

8.4 주소 할당 전 I2C 읽기

ENTDAA 이전에 타깃은 `0x50` 에서 I2C에 응답합니다. Supernova의 I2C 포트(3.3 V에서 2.2 kOhm 풀업을 사용하는 일반 I2C 컨트롤러)에서, 매 실행 전에 타깃을 프로그래밍한 상태로 `00h` , `10h` , `7Ch` , `80h` , `A0h` , `FCh` 에서 3, 4, 16, 128바이트를 읽은 결과, 100 kHz와 400 kHz 모두에서 각각 30/30이었고 데이터는 이미지와 일치했습니다. 각 전송을 `7Eh` 헤더로 시작하는 I2C 모드의 I3C 포트에서는 세션 1에서 30회 중 8회가 정확했고, 세션 3에서는 리셋 후 첫 번째 읽기가 NACK되었으며 두 번째 읽기는 `00h:46-48` 에서 `80 00 00` 을 반환했습니다.

9 문제 해결

표 8. 문제 해결

증상	예상 원인 및 조치
모든 결합 읽기가 `FW_I3C_NACK_ADDRESS` 로 실패	커맨드가 너무 늦게 준비됨: DMA 인터럽트 우선순위가 I3C 인터럽트보다 높은지, 그리고 그 안에서 느린 작업이 실행되지 않는지 확인하십시오
128바이트 읽기가 실패하고 분석기에 SCL 클록 308개가 표시됨	읽기가 32바이트에서 잘림: 메인 루프가 송신 버퍼를 공급하지 않고 있습니다. 블로킹 `printf` 및 유사한 작업을 제거하십시오
읽기 실패 후 리셋할 때까지 타깃이 응답하지 않음	수신이 진행 중으로 표시된 채 남아 있음: 펌웨어에 `I3C1_Target_ReceiveCancel()` 이 필요합니다
전원 투입 후 첫 초기화가 `FW_I3C_BUS_INIT_NACK_RSTDAA` 로 실패	리셋 후 첫 번째 전송은 NACK됨: 다시 초기화하십시오
MPLAB IPE가 "Connection Failed" 또는 `Transmission on endpoint 2 failed (err = -150)` 을 보고	PKoB4의 USB를 다시 연결해야 합니다

10 약어

표 9. 약어

용어	의미
BCR	Bus Characteristics Register(버스 특성 레지스터)
CCC	Common Command Code(공통 커맨드 코드)
CMIS	Common Management Interface Specification(공통 관리 인터페이스 사양)
DA	Dynamic Address(동적 주소)
DCR	Device Characteristics Register(디바이스 특성 레지스터)
DMA	Direct Memory Access(직접 메모리 액세스)
ENTDAA	Enter Dynamic Address Assignment(동적 주소 할당 시작)
IBI	In-Band Interrupt(인밴드 인터럽트)
MRL	Max Read Length(최대 읽기 길이)
PID	Provisioned ID(프로비저닝된 ID)
RSTDAA	Reset Dynamic Address Assignment(동적 주소 할당 리셋)
SDR	Single Data Rate(단일 데이터 속도)
Sr	Repeated START(반복 START)

11 문서 내 소스 코드에 관한 참고 사항

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

아카이브 `AN0019-assets.zip`에는 `cmis_target_check.py`와 `cmis_image.h`가 포함되어 있습니다.

12 개정 이력

표 10. 개정 이력

개정	날짜	설명
0.1	29 October 2026	내부 검토용 초안.
0.2	6 November 2026	호스트 스크립트를 pycosmicsdk로 이전, 세션 3 결과, 주소 할당 전 I2C 읽기를 일반 I2C 컨트롤러에서 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. **I3C** 및 **MIPI**는 MIPI Alliance, Inc.의 상표 또는 등록 상표입니다. **CMIS** 및 **OIF**는 Optical Internetworking Forum의 상표 또는 등록 상표입니다. **Microchip**, **dsPIC**, **MPLAB** 및 **Curiosity**는 Microchip Technology Incorporated의 상표 또는 등록 상표입니다. **Saleae** 및 **Logic**은 Saleae, Inc.의 상표입니다.