binho

APPLICATION NOTE

# Building a CMIS I3C target on a Microchip dsPIC33AK and testing it with the Binho Supernova

AN0019

## Document information

| | |
|---|---|
| **Keywords** | AN0019, Supernova, CMIS, OIF, I3C, MIPI I3C Basic, SDR, I3C target, dsPIC33AK, dsPIC33AK256MPS306, MCC Melody, DMA, ENTDAA, IBI, GETSTATUS, optical module |
| **Abstract** | A CMIS optical module answers its host as an I3C target, and the host reads it the way an I2C host does: write the byte address, repeated START, read. On the dsPIC33AK256MPS306 I3C peripheral the interrupt for that write arrives after the read header has already been answered, so a target that arms its reply from it NACKs every such read. This note shows where the byte address is available in time, how the firmware arms from the receive DMA instead, and the numbers measured from a Binho Supernova: 60 of 60 reads at 4 to 128 bytes, 300 of 300 with in-band interrupts on, after a reset too. It supplies the host-side test script and lists the behaviors of the peripheral and driver that a port has to plan for. |

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

REVISION 1.0
6 NOVEMBER 2026

> **What was tested.** Every result in Section 8 was measured on the bench described in Section 3. The target firmware is not in the assets archive; ask techsupport@binho.io for it.

# 1 Introduction

A CMIS module exposes a 256-byte window, Lower Memory plus one 128-byte page of Upper Memory, selected by a page byte at `00h:127`. Over I2C the host writes a byte address and then reads with a repeated START. The I3C management interface being defined for CMIS keeps that access pattern, so a module built on an I3C target peripheral has to answer a read whose byte address arrived a few bit times earlier, in the same transfer.

That is easy when the peripheral hands each byte to firmware as it arrives. The dsPIC33AK256MPS306 I3C peripheral is command and response based instead: a received transfer is reported once, as a response, and a read is answered by a transmit command that must already be valid when the read header arrives. This note is about making those two meet.

## 1.1 Scope

- A CMIS register model on the dsPIC33AK: paging, the section 8.2.15 unsupported-page override, the Appendix B.1.2 wrap inside a 128-byte segment, and an IBI that carries the Flags Summary byte `00h:4`.
- Answering write + Sr + read (combined) and write, STOP, read (split) at 2.5 MHz push-pull.
- Dynamic address assignment, RSTDAA, and the Direct GET CCCs.
- A host script that runs every check in this note from a Supernova.

Not covered: CDB, firmware download, HDR modes, and the module-side meaning of the CMIS control registers. The register image is static except for the page select and the bytes the host writes.

## 1.2 Applicable devices

**Table 1.** Applicable devices

| Family | Example parts | Identifier | Basis |
|---|---|---|---|
| dsPIC33AK MPS with I3C | dsPIC33AK256MPS306 (EV02E57A DIM) | PID `02 36 43 4D 49 53` | Tested |
| dsPIC33AK MP without I3C | dsPIC33AK512MPS506 and family | - | Not applicable: no I3C peripheral |
| Other parts with the same I3C peripheral | - | - | Not tested |

### 1.3 Related documents

**Table 2.** Related documents

| Document | Title |
|---|---|
| OIF-CMIS-05.3 | Common Management Interface Specification, revision 5.3 |
| MIPI I3C Basic v1.1.1 | MIPI I3C Basic Specification, version 1.1.1 |
| DS70005539 | dsPIC33AK MPS family data sheet, section 24 (I3C) |
| AN0009 | Managing a CMIS optical module over I3C with the Binho Supernova |

## 2 System overview

The Supernova is the I3C controller and the only one on the bus. The dsPIC33AK is a target with static address `0x50`, the CMIS module address. The controller assigns it a dynamic address with ENTDAA (the bench assigns `0x08`), then reads and writes it in SDR.

The one fact that shapes the firmware is when the byte address of a combined read becomes visible. Measured with the I3C interrupt off, polling the peripheral against a 100 MHz timer and aligning it with a logic analyzer:

**Table 3.** Combined read at 2.5 MHz push-pull, times from START

| Event | Time |
|---|---|
| T-bit of the byte address | 22.3 µs |
| Byte address in the receive FIFO | 23.8 µs |
| Repeated START | 24.1 µs |
| ACK slot of the read header | ~27.4 µs |
| Response interrupt for the write | after the ACK slot |

A transmit command armed from the write's response interrupt is always late, and the target NACKs the read. The byte address itself is available 3.6 µs before the ACK slot, and arming a command takes 1.15 to 1.26 µs. So the firmware arms from the receive DMA: the channel that moves received words into RAM raises an interrupt per word, at a priority above the I3C interrupts, and that interrupt arms the transmit command from the first byte.

Everything else follows from three properties of the peripheral and its MCC Melody driver:

- A transmit command answers one read. The firmware arms a 128-byte run from the byte address to the end of the segment; the controller ends the read early.
- The transmit buffer holds 8 words (32 bytes). The rest of a run is fed from the main loop. Anything that keeps the main loop away for longer than 32 bytes take to send (about 115 µs) cuts the read at 32 bytes.

- Re-arming receive resets the receive FIFO, and that also drops an armed transmit command. Receive is re-armed only after the read.

# 3 Hardware setup

## 3.1 Required equipment

- Binho Supernova, firmware 4.6.2, I3C HV port.
- Microchip EV74H48A Curiosity platform with the EV02E57A dsPIC33AK256MPS306 DIM. Its PKoB4 programs the part and carries the UART.
- Optional: a logic analyzer on SCL and SDA. The captures in this note were taken with a Saleae Logic at 12 MS/s.

## 3.2 Connections

**Table 4.** Connections between the Supernova and the target

| Supernova I3C port | dsPIC33AK DIM | Required |
|---|---|---|
| SCL | I3CSCL1 (RC5) | Yes |
| SDA | I3CSDA1 (RC4) | Yes |
| GND | GND | Yes |
| I3C rail, 3.3 V | not connected, the DIM is powered from the Curiosity board | - |
| - | UART1 TX (RD0), out on the PKoB4 virtual COM port | Optional |

## 3.3 Bus considerations

The bus runs at 3.3 V. Push-pull 2.5 MHz is the lowest rate the Supernova offers, and the only rate measured here. A faster rate shortens the 3.6 µs window in Table 3: at 12.5 MHz it is shorter than the time to arm a command.

# 4 Software setup

Target firmware:

- MPLAB X v6.35 with MPLAB IPE, XC-DSC v4.00, dsPIC33AK-MP DFP 1.4.260.
- The MCC Melody I3C target driver, with the local changes in Section 5.6.

Host:

- Python 3 with pycosmicsdk (latest release), and `cmis_target_check.py` from the assets of this note.

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

# 5 Target firmware design

## 5.1 Register model

Lower Memory and Upper Memory page `00h` come from the CMIS 5.3 Appendix C example image. Page select at `00h:127` accepts `00h` and `01h`; any other value is replaced by `00h` (section 8.2.15). Each segment is stored twice back to back, plus copies shifted by one to three bytes, so a 128-byte run from any byte address is one aligned buffer and the B.1.2 wrap inside the segment needs no code.

## 5.2 Identity

**Table 5.** Identity seen by the controller

| Item | Value |
|---|---|
| Static address | `0x50` |
| PID | `02 36 43 4D 49 53` (MIPI manufacturer `0x11B`, part `"CMIS"`) |
| BCR | `0x17`: IBI capable, IBI carries a byte, max data speed limitation |
| DCR | `0x00` |

## 5.3 Arming a read

1. The receive DMA channel moves each received word into `rxbuf`, with its done interrupt enabled and priority 5, above the I3C interrupts at 4.
2. On the first word of a write, the DMA interrupt takes `rxbuf[0]` as the byte address and arms the transmit command for the run from that address.
3. When the write's response arrives and it carried only the address, the command stays armed and receive is re-armed after the read completes.
4. When the write carried data, the command is cancelled by resetting the extended transmit buffer, and receive is re-armed at once.
5. If the read header still finds nothing armed, the peripheral raises a read request. The firmware drops anything armed late and re-arms receive with `I3C1_Target_ReceiveCancel()`, because in that case no response for the write ever comes.

## 5.4 Feeding long reads

The transmit buffer is pre-filled with 8 words. `I3C1_Target_TxRefill()` feeds the rest from the main loop, with the I3C and DMA interrupts masked for the few instructions it takes. The UART

status line of the bring-up firmware is off by default: its blocking `printf` at 9600 baud held the main loop for about 110 ms a second and cut 128-byte reads at 32 bytes.

### 5.5 Events

The IBI carries `00h:4` as its mandatory data byte, once a second while IBIs are enabled. The peripheral comes out of reset with IBIs enabled, as I3C Basic specifies, and does not keep a 0 written to that enable: DISEC is the way to turn them off.

### 5.6 Changes to the Melody driver

Table: Local changes in `mcc/`, marked `binho:`

| Change | Why |
|---|---|
| `BufferTransmit` pre-fills 8 words; `I3C1_Target_TxRefill()` feeds the rest | Over 8 words the DMA path wrote word 7 twice and never continued, so the command was rejected |
| `I3C1_Target_ReceiveCancel()` | After write + Sr + a NACKed read header, no response comes for the write; the driver kept its receive marked as in progress and refused every later one |
| `rxInProgress` cleared before the completion callback | So the callback can re-arm receive |
| Static address `0x50` | CMIS module address |

## 6 Procedure

**Step 1.** Build and program the target from the firmware folder:

```
make flash
```

The build ends with `Program Succeeded.` from MPLAB IPE.

**Step 2.** Wire the Supernova I3C port to the DIM as in Table 4.

**Step 3.** Read with write + Sr + read, 128 bytes, 60 times:

```
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)
```

The `init bus` line is the first RSTDAA after a reset being NACKed (Section 8.3); the script initializes again.

**Step 4.** The same with IBIs enabled (ENEC before the reads):

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

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

`F` in the row of dots marks a read that raised an error, `B` one that returned wrong data.

**Step 5.** Data writes mixed with reads: per round, write `7Fh=00h` and `7Eh=00h`, then read `7Fh` (1 byte) and `80h` (16 bytes), checked against the image:

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

```
rounds 60/60
```

**Step 6.** Address reset and reassignment:

```
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
```

**Step 7.** Identity and Direct GET CCCs:

```
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}
```

A Direct GET can fail on a NACKed `7Eh` header (Section 8.3); here GETMRL did. Run it again.

## 7 Reference

Table: `cmis_target_check.py` commands

| Command | What it does |
|---|---|
| `reads --length N --count N [--split] [--ibi]` | Reads at `00h`, `10h`, `7Ch`, `80h`, `A0h`, `FCh` in turn and compares with the image. Combined unless `--split`; `--ibi` sends ENEC first, else DISEC |
| `mixed --count N` | Data writes to `7Fh` and `7Eh`, then combined reads of `7Fh` and `80h`, checked |
| `ccc [--getstatus N]` | GETBCR, GETDCR, GETPID, GETMXDS, GETMRL, GETMWL, then GETSTATUS N times |
| `daa --cycles N` | RSTDAA, checks the old address NACKs, ENTDAA, checks a read |
| `i2c` | Before any address assignment, reads `00h:0-3` and `00h:46-48` at `0x50` in I2C mode |

Common options: `--serial`, `--push-pull` (default `PUSH_PULL_2_5_MHZ_DC_25`), `--voltage-mv` (default 3300), `--image` (default `cmis_image.h` next to the script). Each command starts with RSTDAA and ENTDAA and finds the target by the `"CMIS"` in its PID.

# 8 Measured results

## 8.1 Test configuration

**Table 6.** Test configuration

| Item | Version |
|---|---|
| Supernova | serial 4F970418, firmware 4.6.2 |
| Target | EV02E57A DIM on EV74H48A |
| Toolchain | XC-DSC v4.00, DFP 1.4.260, MPLAB X v6.35 |
| Host | Ubuntu 26.04; sessions 1 and 2: Python 3.13, binhosupernova 4.2.0; session 3: Python 3.14.4, pycosmicsdk 1.9.1 |
| Bus | 3.3 V, push-pull 2.5 MHz, open-drain 1 MHz |

## 8.2 Results

Three sessions. Session 2 started after programming (a reset); session 3 (6 November 2026) programmed the target before every test, with the host on pycosmicsdk:

**Table 7.** Results

| Test | Session 1 | Session 2 | Session 3 |
|---|---|---|---|
| Combined read, 4 / 16 / 32 / 128 B, IBI off | 60/60 each | 60/60 (4 and 128 B) | 60/60 each |
| Combined read, 128 B, IBI on | 300/300 | 300/300 | 299/300 |
| Split read, 4 / 128 B | 60/60 each | 60/60 (128 B) | 60/60 (128 B) |
| Data writes and combined reads, data checked | 60/60 rounds | 60/60 rounds | 60/60 rounds |
| RSTDAA, old address NACKs, ENTDAA, read | 10/10 | 10/10 | 10/10 |
| GETSTATUS | 97/100 | 94/100 | 100/100 |

Before the receive DMA arm, the same combined reads were 0 of 60.

## 8.3 Behaviors a host should expect

- **The first transfer after a target reset is NACKed.** After each of 5 resets the first `7Eh` header was NACKed and the second ACKed. DS70005539 section 24.4.5 describes the module leaving reset only after it sees SCL clocks. A host that probes once and takes the NACK as "no I3C target" will miss this module; retry the probe.

- **Some Direct GET CCCs fail on a NACKed `7Eh`.** In one session about 1 in 7 `7Eh` headers that followed a Direct GET was NACKed (173 failures in 1200 GETs, all 173 visible on the wire as `7E/W NACK`); in another, 500 of 500 passed. Private reads were not affected. Retry a GET that fails with a NACK.
- **GETMRL reports 255 and SETMRL is not enforced.** The firmware sets 128; the peripheral answers GETMRL itself with 255. After SETMRL 16, a 32-byte read still returns 32 bytes.

### 8.4 I2C reads before address assignment

Before ENTDAA the target answers I2C at `0x50`. From the Supernova's I2C port, a plain I2C controller with 2.2 kOhm pull-ups at 3.3 V, with the target programmed before each run, reads of 3, 4, 16 and 128 bytes at `00h`, `10h`, `7Ch`, `80h`, `A0h` and `FCh` were 30/30 each at 100 kHz and at 400 kHz, data checked against the image. From the I3C port in I2C mode, which starts each transfer with a `7Eh` header, session 1 had 8 of 30 correct; in session 3 the first read after a reset was NACKed and the second returned `80 00 00` from `00h:46-48`.

## 9 Troubleshooting

**Table 8.** Troubleshooting

| Symptom | Probable cause and action |
|---|---|
| Every combined read fails with `FW_I3C_NACK_ADDRESS` | Command armed too late: check the DMA interrupt priority is above the I3C interrupts, and that nothing slow runs in it |
| 128-byte reads fail and the analyzer shows 308 SCL clocks | Read cut at 32 bytes: the main loop is not feeding the transmit buffer. Remove blocking `printf` and similar |
| The target stops answering until reset after a failed read | Receive left marked in progress; the firmware needs `I3C1_Target_ReceiveCancel()` |
| First init after power-up fails with `FW_I3C_BUS_INIT_NACK_RSTDAA` | First transfer after reset is NACKed; init again |
| MPLAB IPE reports "Connection Failed" or `Transmission on endpoint 2 failed (err = -150)` | The PKoB4 needs a USB replug |

## 10 Acronyms

**Table 9.** Acronyms

| Term | Meaning |
|---|---|
| BCR | Bus Characteristics Register |
| CCC | Common Command Code |
| CMIS | Common Management Interface Specification |
| DA | Dynamic Address |
| DCR | Device Characteristics Register |
| DMA | Direct Memory Access |
| ENTDAA | Enter Dynamic Address Assignment |
| IBI | In-Band Interrupt |
| MRL | Max Read Length |
| PID | Provisioned ID |
| RSTDAA | Reset Dynamic Address Assignment |
| SDR | Single Data Rate |
| Sr | Repeated START |

## 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 `AN0019-assets.zip` contains `cmis_target_check.py` and `cmis_image.h`.

## 12 Revision history

**Table 10.** Revision history

| Revision | Date | Description |
|---|---|---|
| 0.1 | 29 October 2026 | Draft for internal review. |
| 0.2 | 6 November 2026 | Host script moved to pycosmicsdk; session 3 results; I2C before address assignment measured from a plain I2C controller. |
| 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. **I3C** and **MIPI** are trademarks or registered trademarks of MIPI Alliance, Inc. **CMIS** and **OIF** are trademarks or registered trademarks of the Optical Internetworking Forum. **Microchip**, **dsPIC**, **MPLAB** and **Curiosity** are trademarks or registered trademarks of Microchip Technology Incorporated. **Saleae** and **Logic** are trademarks of Saleae, Inc.