binho

APPLICATION NOTE

Bringing up a CMIS optical module over SPI with the Binho Pulsar

AN0010

Document information

Keywords	AN0010, Pulsar, Supernova, CMIS, OIF, SPIMCI, SPI, co-packaged optics, CPO, flow control, ACK, NACK, chip select, optical module
Abstract	SPIMCI, the SPI management interface added to CMIS in revision 5.3 for co-packaged optics, encodes acknowledgement as 00h. An idle bus, an unpowered module, or a target that was never armed therefore returns a byte perfect acknowledgement followed by zero data, and a bring-up that appears to succeed proves nothing. A second obstacle compounds it: the flow control byte count is not on the wire and is derived from registers that cannot be read until it is already known. This application note brings up a SPIMCI link from a Binho Pulsar or Supernova in a way that a silent bus cannot fool, and supplies a command that establishes both unknowns from the module's own checksum.

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

REVISION 1.0
6 NOVEMBER 2026

> **What was tested.** The specification content in this document is sourced and verified. The procedure in Section 6 has not yet been executed against a SPIMCI module. It has been executed against a board carrying no SPIMCI firmware, which is what Section 8.2 reports and what corrected the checksum guard in Section 6.2. Section 8 states exactly what is outstanding.

1 Introduction

You wire up a SPIMCI module, issue a read, and it works. The transaction completes, the acknowledgement is there, and data comes back.

The module is not powered.

SPIMCI encodes ACK as **00h** (B.3.7.1.2), so all zeros is a valid acknowledgement followed by zero data. Binho reproduced exactly this on the bench: the read completed, the acknowledgement was seen, and the data was `00 00`. A completed SPIMCI transaction is never, by itself, evidence that a module is present.

This note brings up a SPIMCI link in a way that silence cannot fake, and solves the second problem SPIMCI hands a host along the way.

1.1 Scope

By the end of this document a reader with the hardware in front of them will have established the flow control byte count and the R/Wn polarity their module implements, read its identity, and validated its Page 00h checksum, having accepted no intermediate result that an absent module could have produced.

The document covers the SPI management interface, CMIS Appendix B.3, normative and introduced in revision 5.3. AN0008 covers the register model and is a prerequisite. Co-packaged optics system design is out of scope.

1.2 Applicable devices

Table 1. Applicable devices

Item	Role	Basis
Binho Pulsar	Host adapter, SPI controller	Not yet exercised for this note
Binho Supernova	Host adapter, SPI controller	Exercised as the controller, Section 8.1. No SPIMCI target was present
Any CMIS module with a SPIMCI interface	Target	**Not yet exercised for this note**

> **No SPIMCI module has been attached at revision 1.0.** The framing, the flow control derivation and the polarity contradiction are taken from OIF-CMIS-05.3 Appendix B.3, cross-checked against 05.4. What has been exercised is the host side and the failure mode: the forged acknowledgement was observed on a bus with no module attached on 15 August 2026, and again on 2 September 2026 against a present but silent board, where it defeated the checksum guard as well. Everything that depends on a module answering is outstanding.

1.3 Related documents

Table 2. Related documents

Document	Subject
OIF-CMIS-05.3	Appendix B.3, SPI-Based Management Communication Interface, pages 374 to 383
OIF-CMIS-05.4	Appendix B.3, pages 416 to 425. The text is identical to 5.3 apart from footers
AN0008	Bringing up a CMIS optical module over I2C with the Binho Supernova
AN0011	Measuring CMIS firmware update time with the Binho Supernova

2 Required equipment and software

2.1 Hardware

Table 3. Required equipment

Item
Binho Pulsar or Binho Supernova USB host adapter
A CMIS module with a SPIMCI management interface
USB-C cable
Saleae Logic capture hardware, optional

2.2 Connections

CMIS renames the classic SPI signals. Use the specification's names, with the familiar ones alongside so that nobody is misled.

Table 4. Connections between the adapter and the module

Adapter	Spec name	Classic name	Required
CS	`CSn`	CS, SS or ModSeln	Yes
SCK	`CLK`	SCLK	Yes
MOSI	`IOTI`, Initiator Out Target In	MOSI, PICO	Yes
MISO	`IITO`, Initiator In Target Out	MISO, POCI	Yes
3V3	Supply		Yes, unless the board supplies its own
GND	Return		Yes

> **Caution:** `CSn` is mandatory, not optional. Active low is normative (B.3.3), and one dedicated `CSn` serves each target with at most one asserted at a time (B.3.2). Without it there is no framing reference, no error detection and no resynchronization: one extra clock edge misaligns the remainder of the session permanently. A generic SPI decoder tolerates a missing chip select and returns plausible garbage.

2.3 Software

```
pip install binhopulsar          # Pulsar
pip install binhosupernova       # Supernova
python cmis_mci.py selfcheck
```

`--transport spi` selects the SPI binding. The utility is the same file supplied with AN0008, AN0009 and AN0011.

3 What SPIMCI is, and why it is worth understanding

SPIMCI was introduced in CMIS 5.3, driven by co-packaged optics; the revision history cites "SPI-based SPIMCI in support of CPO". The string `SPI` does not appear anywhere in CMIS 5.2.

It is also the precedent that matters for any future binding. It was added without touching the register access layer or the Common Data Block at all, and while it was in there it fixed the addressing. Where I2CMCI carries only a page local byte address, and therefore needs the page select registers at 00h:126 and 00h:127, a SPIMCI transaction carries a full three dimensional bank, page and byte address with implicit page switching, and transfers up to 2048 bytes.

The practical consequence for a host is that **there is no page select register to write and no cross-transaction address state to get wrong.**

Table 5. Parameters fixed by the specification

Parameter	Value	Source
Word size	8 bits, most significant bit first	B.3.7.2.3
Sampling edge	Rising edge of CLK	B.3.7.1
Chip select polarity	Active low	B.3.3
Maximum transfer	2048 bytes	B.3.7.1

> **Note:** the specification pins the sampling edge but never states the CLK idle level, so "SPIMCI is SPI mode 0" is an inference rather than a quotation. The utility defaults to mode 0 and leaves the mode selectable.

4 Transaction structure

Every SPIMCI transaction is **4 + N + M** bytes in three phases.

Table 6. Transaction Control phase, 4 bytes, Initiator to Target only

Bits	Field	Notes
0	Command Type, R/Wn	See Section 5.2
1:4	Bank Index	4 bits. The specification notes this "does not allow for arbitrary bank index values greater than 15"
5:15	EncodedSize	11 bits. Bytes transferred = EncodedSize + 1, maximum 2048
16:23	Page Index	8 bits
24:31	Byte Index	8 bits, the start address. Sequential thereafter

`IITO` is undefined throughout the control phase.

Flow Control phase, N bytes, N at least 2. The Initiator sends zero valued dummy bytes, which the Target ignores. The Target returns **ACK = 00h** or **NACK = FFh** in the **last** byte of the phase, on `IITO`.

Data Transfer phase, M = EncodedSize + 1 bytes. On a read, data is on `IITO` and 00h is sent on `IOTI`. On a write, data is on `IOTI` and `IITO` is undefined.

5 The two unknowns, and how to resolve them

5.1 N is not on the wire, and it has a bootstrap hole

`N` is derived from `MciFlowControlDuration` at `00h:213` bits 6-0 and `MciFlowControlDurationEncoding` at `00h:213` bit 7:

Table 7. Flow control byte count

Encoding bit	Derivation
0, static	N = 2 + the advertised value
1, speed dependent	N = max(2, ceil(duration x SpiSpeedInMHz / 8)), speed from `MciSpeedConfiguration` at `00h:27.3-0`

B.3.7.1.2 requires both sides to derive `N` identically. But `00h:213` cannot be read until `N` is already known, and the specification names no safe bootstrap value. A passive bus observer cannot know it either.

The resolution is to sweep `N` and let the module's own checksum decide which value is right, which is what `spi-probe` does.

5.2 The R/Wn polarity contradicts itself

Table 8. R/Wn polarity in CMIS 5.3

Source	States
B.3.7.1.1	"When this bit is 1, a read is being requested"
B.3.8.1 to B.3.8.5, Figures B-14 to B-17	0 for a read

Both are in the same document. There is a tiebreaker worth knowing: B.3.8.5 defines **TEST** as a one byte read from byte address 0 with **all 32 control bits zero**, which is only self consistent if 0 means read. The utility defaults to the figures, speaks either polarity on request, and probes both when not told.

6 Procedure

6.1 Confirm the tool offline

```
python cmis_mci.py selfcheck
```

6.2 Establish N and the polarity

```
python cmis_mci.py spi-probe --transport spi
```

The command sweeps `N` from 2 to 16 against each polarity and accepts a combination **only** when the Page 00h checksum validates: 94 bytes of module specific content agreeing with a byte the module itself wrote. A completed transaction, an acknowledgement, and even plausible looking data can all be produced by an unpowered board.

> **Caution:** a matching checksum can be produced by an unpowered board too, in exactly one way, and it is the way that occurs in practice. A silent SPI bus returns `00h` for every byte. Ninety four bytes of `00h` sum to `00h`, and `00h` is what the same bus reports as the stored checksum, so the two agree. The sweep therefore accepts a combination only when the checksum validates **and** Page 00h carries more than one distinct value. Section 8.2 records what the command reported before that condition was added.

Where the values are already known, pass them and skip the sweep:

```
python cmis_mci.py spi-probe --transport spi --n 4 --read-bit 0
```

On failure the command reports what to check and deliberately does not offer "the module answered" as consolation, because on SPIMCI that is not information.

6.3 Read the module

```
python cmis_mci.py bringup  --transport spi --n 4 --read-bit 0
python cmis_mci.py identify --transport spi --n 4 --read-bit 0
python cmis_mci.py read --transport spi --page 01h --byte 251 --n 4 --read-bit 0
```

Note that the page read in the last command needs no page select write. The page travels in the transaction header.

7 What the analyzer adds

A capture of `CSn`, `CLK`, `IOTI` and `IITO` decoded with the CMIS SPIMCI analyzer shows the 4 + N + M framing, the acknowledgement byte, and the register the transaction addressed.

Beyond that it reports the one error condition the specification actually defines, which a generic decoder structurally cannot represent.

ABORT. A regular STOP falls on an 8 bit boundary at the expected time. An irregular STOP, on a non 8 bit boundary or earlier than expected, terminates the transaction, and "any incompletely received byte must be discarded or ignored by the receiving side". A stock SPI

decoder silently discards a word interrupted by a chip select deassert: no frame, no flag, no marker. That case is precisely the SPIMCI ABORT condition.

The analyzer also surfaces `N` rather than hiding it. It can be set explicitly, given as duration plus speed, or auto-detected by scanning forward from byte 4 for the first 00h or FFh on `IITO`, with a confidence indicator, because an all zero data byte can alias.

> **The CMIS SPIMCI analyzer is in development and has not shipped.** This section describes the decode it produces, not a product a reader can buy today.

8 Measured results

8.1 Test configuration

Table 9. Test configuration

Item	Value
Adapter	Binho Supernova, serial `4F970418…1111`, firmware 4.4.0, hardware revision C
SDK	`binhosupernova`, Python 3.13
Module	None. An NXP FRDM-MCXN947 was wired in the module's place, carrying no SPIMCI firmware
Bus	3.3 V, 1 MHz, mode 0, chip select S0 on the Binho adapter board

8.2 Results

Observed, 15 August 2026. A SPIMCI read issued against a bus with no module attached completed, returned a valid acknowledgement, and returned `00 00` as data. That is the observation Section 1 rests on, and it is a property of the encoding rather than of any particular module.

Observed, 2 September 2026, and it is worse than the above. The same probe was run against a board that was present, powered and correctly wired, but running no SPIMCI firmware, so every byte it returned was `00h`. The command did not report an absent module. It reported a healthy one:

```
N        : 2 flow control bytes, transaction is 4 + 2 + M
R/Wn     : 0 means READ on this module
00h:222  : checksum 0x00 matches the stored byte

PASS     : the module is present and self consistent.
```

The checksum agreed because both sides of the comparison were zero: 94 bytes of `00h` sum to `00h`, and `00h` is what the same silent bus reported as the stored checksum. The check that

Section 6.2 relies on to reject a silent bus was itself defeated by the silence, and the utility was corrected to require Page 00h to carry more than one distinct value before a checksum agreement counts. Against the same hardware it now reports:

```
FAIL     : no combination of N and R/Wn produced a valid Page 00h checksum.
           Either the module is absent, or CSn, CLK, IOTI or IITO are
           miswired. On SPIMCI a silent bus still returns a perfect ACK.
```

This is the note's own thesis turned on the note: a bring-up that looks successful can be reading nothing at all. A host implementing Appendix B.3 has to guard the same case.

Offline. The register access layer this note drives is the one AN0008 self-checks, 8 of 8 against a simulated module. The eighth case is the one this section added: an all-zero Page 00h must be rejected rather than read as a checksum that agrees.

On hardware, against a module. Nothing. Outstanding at revision 1.0: the `N` sweep, the polarity determination, the identity read and the Page 00h checksum, each against a SPIMCI module. What Section 8.2 establishes is the failure mode and the guard against it, not the success path.

9 Troubleshooting

Table 10. Troubleshooting

Symptom	Probable cause and action
Every transaction succeeds and all data is zero	The classic forged acknowledgement. The module is absent, unpowered or unarmed. The checksum distinguishes it only if you also require Page 00h to carry more than one distinct value, because 94 zeros agree with a stored checksum of zero
`spi-probe` finds no valid combination	Check `CSn`, `CLK`, `IOTI` and `IITO` wiring before suspecting `N`. A silent bus still acknowledges
The flow control byte is neither 00h nor FFh	`N` is wrong, so the byte being read is data or control, not the acknowledgement
Data is offset by one or more bytes	`N` is off by that many. Re-run the sweep rather than adjusting by hand
Reads return data, writes appear ignored	The R/Wn polarity is inverted. Try the other `--read-bit`
A transaction is truncated mid word	An irregular STOP, the ABORT condition. Check that nothing else drives `CSn`
The adapter reports a short transfer	The transfer length and payload length disagree. This is a host bug, not a bus fault

10 Acronyms

Table 11. Acronyms

Term	Meaning
ACK	Acknowledgement, 00h on SPIMCI
CPO	Co-packaged optics
CMIS	Common Management Interface Specification
CSn	Chip select, active low
IITO	Initiator In, Target Out
IOTI	Initiator Out, Target In
MCI	Management Communication Interface
NACK	Negative acknowledgement, FFh on SPIMCI
OIF	Optical Internetworking Forum
RAL	Register Access Layer
SPIMCI	SPI Management Communication Interface

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.

Table 12. Contents of the assets archive

File	Description
`AN0010.md`	Markdown source of this document
`assets/cmis_mci.py`	Reference utility, shared by AN0008 through AN0011
`assets/LICENSE`	MIT license covering the example code

12 Revision history

Table 13. Revision history

Revision	Date	Description
0.1	1 September 2026	Draft for internal review. Specification content verified; procedure not yet run on hardware.
0.2	2 September 2026	Renumbered from AN0009. `spi-probe` run against a board carrying no SPIMCI firmware: the all-zero forged acknowledgement defeated the Page 00h checksum and the command reported PASS. The utility now rejects a page of one repeated value, and Section 8.2 records both outputs. No SPIMCI module has been attached.
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. **CMIS** and **OIF** are trademarks or registered trademarks of the Optical Internetworking Forum. **Saleae** and **Logic** are trademarks of Saleae, Inc.