COMMON MANAGEMENT INTERFACE SPECIFICATION (CMIS)

Rev 5.0 May 8, 2021 (From QSFP-DD MSA)

OIF LEGAL NOTICES

THIRD PARTY SPECIFICATION

THIS [STANDARD/SPECIFICATION] (THE “SPECIFICATION”) WAS DEVELOPED, APPROVED, AND ORIGINALLY PUBLISHED BY **QSFP-DD MSA** (WWW.QSFP-DD.COM/), WHICH HAS CONCLUDED ITS WORK ON THIS SPECIFICATION. THE SPEFICIATION IS BEING PROVIDED BY OPTICAL INTERNETWORKING FORUM (OIF), WITH PERMISSION FROM QSFP-DD MSA, AS A CONVENIENCE TO OIF MEMBERS AND VISITORS. OTHER THAN THE AFOREMENTIONED PERMISSION, OIF IS NOT AFFILIATED WITH AND DOES NOT ENDORSE QSFP-DD MSA OR THIS SPECIFICATION, AND PUBLICATION OF THIS SPECIFICATION SHOULD NOT BE CONSTRUED AS AN ENDORSEMENT. FOR INFORMATION ABOUT QSFP-DD MSA, PLEASE CONTACT INFO@QSFP-DD.COM.

BECAUSE THIS IS A THIRD PARTY SPECIFICATION, THE SPECIFICATION WAS NOT REVIEWED OR APPROVED UNDER THE OIF DEVELOPMENT PROCESS, AND IS NOT GOVERNED BY THE OIF INTELLECTUAL PROPERTY RIGHTS (“IPR”) POLICY. ACCORDINGLY, NEITHER OIF NOR ANY OF ITS MEMBERS ACTING IN THAT CAPACITY HAVE COMMITTED TO GRANTING ANY LICENSES OR MADE ANY OTHER COMMITMENTS OR DISCLOSURES REGARDING THIS SPECIFICATION OR ANY PATENT CLAIMS OR OTHER IPR THAT MIGHT BE INFRINGED BY IMPLEMENTATION OF THE SPECIFICATION.

NO WARRANTIES

THIS SPECIFICATION IS BEING OFFERED AND PROVIDED “AS IS” AND WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. THE AUTHORS OF THIS SPECIFICATION HAVE DISCLAIMED ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY PROPRIETARY RIGHTS, RELATING TO USE OF INFORMATION IN THIS SPECIFICATION. TO THE GREATEST EXTENT PERMITTED BY LAW, OIF HEREBY DISCLAIMS ALL EXPRESS, IMPLIED AND STATUTORY WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF TITLE, NONINFRINGEMENT, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. WITHOUT LIMITING THE FOREGOING, OIF DOES NOT WARRANT THAT THIS DOCUMENT SPECIFICATION IS COMPLETE OR WITHOUT ERROR AND DISCLAIMS ANY WARRANTIES TO THE CONTRARY. NOTHING IN THIS SPECIFICATION CREATES ANY WARRANTIES WHATSOEVER REGARDING THE SUITABILITY OR NON-SUITABILITY OF A PRODUCT OR A SERVICE FOR CERTIFICATION UNDER ANY CERTIFICATION PROGRAM OF OIF OR ANY THIRD PARTY. ANY USE OF THIS SPECIFICATION SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER OIF, NOR ANY OF ITS MEMBERS OR SUBMITTERS OR THEIR RELATED PARTIES, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER, DIRECTLY OR INDIRECTLY, ARISING FROM THE USE OF THIS SPECIFICATION.

PERMISSIONS

ACCORDING TO THE SPECIFICATION, YOU ARE AUTHORIZED TO DOWNLOAD, REPRODUCE, AND DISTRIBUTE THIS SPECIFICATION. ALL OTHER RIGHTS ARE RESERVED. THE PROVISION OF THIS SPECIFICATION SHOULD NOT BE CONSTRUED AS THE GRANTING OF ANY RIGHT TO PRACTICE, MAKE, USE, OR OTHERWISE DEVELOP PRODUCTS THAT ARE BASED ON THE SPECIFICATION. ANY AND ALL IPR RELATED TO THIS SPECIFICATION AND THE DESIGNS DISCLOSED WITHIN, EXCEPT FOR THE RIGHTS EXPRESSLY MENTIONED ABOVE, ARE RESERVED BY THE RESPECTIVE OWNERS OF THE RELEVANT IPR.

PATENTS

OIF DRAWS ATTENTION TO THE FACT THAT THE SPECIFICATION OR ELEMENTS THEREOF MAY BE THE SUBJECT OF IPR (OF THE DEVELOPING ORGANIZAZION, ITS MEMBERS OR CONTRIBUTORS, OR THIRD PARTIES), WHICH MAY BE INFRINGED BY IMPLEMENTATION OF THE SPECIFICATION OR ELEMENTS THEREOF. OIF IS NOT AND SHALL NOT BE RESPONSIBLE FOR IDENTIFYING ANY OR ALL SUCH IPR.

THIRD PARTY RIGHTS

NOTWITHSTANDING AND WITHOUT LIMITING THE GENERALITY OF THE FOREGOING, OIF ASSUMES NO RESPONSIBILITY TO COMPILE, CONFIRM, UPDATE OR MAKE PUBLIC ANY THIRD PARTY ASSERTIONS OF PATENT OR OTHER IPR THAT MIGHT NOW OR IN THE FUTURE BE INFRINGED BY AN IMPLEMENTATION OF THIS SPECIFICATION IN ITS CURRENT, OR IN ANY FUTURE FORM. IF ANY SUCH RIGHTS ARE ASSERTED OR ARE DESCRIBED IN THE SPECIFICATION OR BY OIF, OIF TAKES NO POSITION AS TO THE VALIDITY OR INVALIDITY OF SUCH ASSERTIONS, OR THAT ALL SUCH ASSERTIONS THAT HAVE OR MAY BE MADE ARE SO DESCRIBED.

BY REVIEWING THIS SPECIFICATION, YOU ACKNOWLEDGE AND AGREE TO ALL OF THE FOREGOING, AND AGREE TO INDEMNIFY AND HOLD HARMLESS OIF AND ITS EMPLOYEES, OFFICERS, DIRECTORS AND MEMBERS FROM ANY CLAIMS, DAMAGES, LOSSES, COSTS, AND EXPENSES ARISING FROM OR IN CONNECTION WITH YOUR USE OF THE SPECIFICATION.

Common Management Interface Specification (CMIS)

Rev 5.0 May 8, 2021

Abstract: This document defines the Common Management Interface Specification (CMIS), which may be used by pluggable or on-board modules, such as e.g. QSFP Double Density (QSFP-DD), OSFP, COBO, QSFP, as well as by future module developments with host to module management communication based on a two-wire interface. This specification is targeted for systems manufacturers, system integrators, and suppliers of CMIS compliant modules.

POINTS OF CONTACT:

Stefan Langenbach	Tom Palkert	Mark Nowell	Scott Sommers
Technical Editor	Technical Editor	Co-Chair	Co-Chair
Cisco	Molex	Cisco	Molex
slangenb at cisco.com	tom.palkert at molex.com	mnowell at cisco.com	scott.sommers at molex.com

Limitation on use of Information

This specification is provided "**AS IS**" with **NO WARRANTIES** whatsoever and therefore the provision of this specification does not include any warranty of merchantability, noninfringement, fitness for a particular purpose, or any other warranty otherwise arising out of any proposal, specification, or sample. The authors further disclaim all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

Permissions

Foreword

This CMIS 5.0 specification was developed by the QSFP-DD, OSFP and COBO advisory group.

Further CMIS revisions with the same major revision number will be backwards specification-compatible to CMIS 5.0, allowing interworking in cases when host and module implement different CMIS revisions[1].

Revision History (Published Versions)

Rev 3.0 August 17, 2018

- Initial public release

Rev 4.0 May 8, 2019

WARNING: Implementations compliant to Revision 4.0 of this specification will not interoperate with Revision 3.0 implementations and vice-versa. The following specifications have changed:

- Converted InitMode signal to continuously sampled LPMode
- Associated Module and Data Path State Machine changes
- Added LPMode Override bit
- Data Path initialization control (Page 10h) meaning/polarity has changed
- Added LowPwr bit (Lower memory, Byte 26, bit 6) to switch power mode

Other changes:

- New format to conform to SFF
- Added Scope statement
- Added Conventions
- Added Definitions
- Updated Introduction
- Consolidation of chapter 5
- Rework and Partial Consolidation of chapter 6
- Added Pages 04h and 12h for tunable laser support
- Added diagnostic Pages 13h and 14h
- Added VDM Feature (Versatile Diagnostics Monitoring) (7.1)
- Added CDB Feature (Command/Reply Messaging)(7.2 and 9)
- Added Command Data Block (CDB) section (Pages 9Fh and A0h-AFh)
- Moved Examples to Appendix
- Change Upper Page Lower Page terminology (Upper and Lower Memory plus Pages)
- Removed Interface ID tables and replaced them with references to SFF-8024

Rev 5.0 May 8, 2021

This revision provides a significant number of changes and extensions, as well as comprehensive technical and editorial consolidation of CMIS 4.0. Specification weaknesses (gaps, bugs) have been fixed and some timing specifications have been adjusted, in order to improve host-module interaction performance in the future (see below for a detailed listing of changes). Resulting possibilities of cross-version incompatibilities have been deliberately accepted after careful consideration and broad industry survey.

Note: For instance, the meaning of the Flag Summary registers 00h:4-7 has changed. Careful study of the changes listed below is required to determine if any specific implementation may actually be unaffected.

WARNING: Depending on feature usage and implementation details, module implementations compliant to Revision 5.0 of this specification may or may not interoperate with Revision 4.0 host implementations, and host implementations of Revision 5.0 may need to be module version aware.

HINT: It is strongly recommended that all CMIS 4.0 implementations should be upgraded to CMIS 5.0.

The following list summarizes and classifies the changes made in this revision:

Register Map Changes

-	00h:2.6	SteppedConfigOnly (new advertisement bit)
-	00h:41	Fault Cause register (new, optional)

[1] Note that hosts can learn the CMIS version supported by the module, which allows hosts to adapt to modules supporting earlier CMIS revisions. Working with newer modules may or may not be possible. See Appendix G.

- 10h:153 OutputStatusChangedMaskRx (Mask for new Flag)
- 11h:235 DPInitPending (new status register)
- 11h:202-205 ConfigStatus (now required, new ConfigInProgress status code)
- 11h:132-133 OutputStatusRx, OutputStatusTx (new)
- 11h:153 OutputStatusChangedFlagRx (new)
- 12h:128-135.7-4 33GHz and 75GHz encoding swapped
- 15h New page for Timing Characteristics

CDB Message Changes

- CMD41h, 137.3:New scaling multiplier for maximum duration advertisements (backwards compatible)

Specification Bug Fixes

- TEC Current units of Aux2 Monitor
- LPL Message Length of some CDB command(s)
- Apply_DataPathInit clearing condition was not specified
- Apply_DataPathInit state versus trigger ambiguity resolved (it is a trigger register)
- MediaLaneSupported was mistakenly marked as optional (it is Rqd.)

Extensions (optional)

- Added Page 15h (Timing characteristics for PTP) in section 8.13 (optional)
- Added Fault Cause register in section 8.2.1 (optional)
- Added Firmware readback using CDB in section 7.3.1.3 (optional)
- Intervention-free reconfiguration on Apply* may be advertised as not supported (SteppedConfigOnly)

Technical Requirements Added

- Added OutputStatusRx (with state change flag) and OutputStatusTx in section 8.9.2 (required)
- Added VDM Flag conformance per state in section 6.3.4.3 (new restrictions)
- Added clear specification of Bank and Page Select behavior (section 8.2.13) and timing (section 10.2)

Technical Requirements Changed

- Flag summaries now indicate bank and page (instead on bank and lane) (changed specification)
- Configuration command (Apply*) result status in ConfigStatus register now mandatory
- Configuration command (Apply*) now synchronized with host by ConfigStatus=ConfigInProgress
- ApplyImmediate in DPDeactivated now ignored. ApplyImmediate and ApplyDPInit now have a distinct and non-overlapping purpose; ApplyImmediate only for hot reconfiguration.
- Size matched read of scalar numerical RO data now guarantees data coherency (new restriction)
- Flags on optional pages are now masked by default (changed defaults)
- Generally revised password usage and support (standard CMIS provides only mechanism)
- No password protection allowed for CMIS standardized features
- Timing parameters harmonized with changes defined in QSFP-DD HW Spec revision 5.1.
- Frequency grid encoding in field (12h:128-135.7-4) has swapped the codes for 33GHz and 75 GHz
- Removed ill-defined peak detector from diagnostics features (configuration bits now Reserved)
- Apply* trigger registers allow single byte WRITE only (new restriction)

Technical Content Description

- Management interface timings now normatively defined in CMIS, initialized from QSFP-DD HW spec 5.1
- Refined description of conditionally required fields (section 8.1.3.6)
- TWI renamed as I2C-based MCI (I2CMCI) and described as the first and only instance of an MCI
- Clarification that READ/WRITE retry is an alternative to Acknowledge Polling
- Acknowledge polling is now abstracted as TEST access
- Added descriptive section 7.4: Tunable Lasers, Channel Selection and Center Frequency Setting
- Clarified Wavelength Information fields in Page 01h (section 8.4)
- Updated Appendix C: Example initialization flows
- Added sections 8.2.10.1: Password Type, and 8.2.10.2: Password entry and change
- Added sections on configuration and reconfiguration Command Handling

Changes of document structure

- Some table and figure numbers have changed due to insertion, splitting, or reordering
- Refactored chapter 5: Introduced register ACCESS primitives (READ, WRITE, TEST)
- Register ACCESS primitives implemented by MCI transactions described in new Appendix B.
- Reworked chapter 6 substructure for clarifications and additions
- Added chapter 10 for timing specifications (including MIS specs, migrating from QSFP-DD HW spec)
- Inserted Appendix B for Management Communication Interface (MCI) specifications
- Inserted Appendix E with illustrations of effects after ApplyDPInit
- Added Appendix G for CMIS evolution and maintenance guidelines

Changes of presentation and overall text consolidation

- Expanded Purpose and Scope chapter 1
- Updated document conventions for better text homogeneity in chapter 2
- Updated (extended and enhanced) term definitions in chapter 3
- Introduced arithmetic and other data type definitions in chapter 3
- Expanded description of CMIS modules and CMIS compliance, and document overview in chapter 4
- Updated text to comply to the documented conventions of chapter 2
- Updated text to use documented terminology consistently
- Chapter 6 text consolidated, sub structured, and revised
- Equations are now numbered
- Updated Figures 8-1 and 8-2 describing the management memory map
- Added section 8.1.3 with general specifications across the management memory map
- Added description of management memory access types in the Type column of tables in chapter 8
- Improved description of repetitive field array definitions. Register information in Description column
- Register description now often includes advertisement information
- Reduced verbosity in chapter 8
- Preparations for automated register map extraction
- Removed inverted logic bOOL data type (no longer used)

Renaming and Rewording

- Replaced socially inappropriate technical Master/Slave terminology with **Initiator/Target** terminology
- **Renamed memory map** fields systematically: informal name phrases converted to software-ready single-word field names with instance indexing suffixes (see temporary Appendix H)
- Systematic renaming of controls and signals contributing to a module **Low Power Mode request**:
  ForceLowPwr → LowPwrRequestSW
  LPMode → LowPwrRequestHW
  LowPwr → LowPwrAllowRequestHW
- **Squelching** related terminology updated to better distinguish functionality and control of functionality
- Wording changes due to refactoring of chapter 5 (using WRITE, READ, TEST for register **ACCESS**)

Known Problems

- Unresolved inconsistencies in the description of Diagnostics (multi-bank control)
- Description of CDB in chapters 7, 8, 9 needs further consolidation (structure, dependency, redundancy)

Contents

Figures

Tables

1 Purpose and Scope

This **Common Management Interface Specification** (CMIS) defines a generic management communication interface together with a generic management interaction protocol between managing hosts and managed modules.

The target audience of this specification includes suppliers of modules and transceivers, system manufacturers, and system integrators.

This specification, **CMIS**, has been developed to allow host and module software implementers to utilize a common code base across a variety of form factors and across a variety of module capabilities, and to foster the possibility of vendor agnostic management for standardized module functions.

To this end CMIS specifies a small core of basic functionality that all modules must implement and a larger evolving set of optional features whose implementation is advertised in the so-called management memory map[1] of a module. This advertisement approach allows host software to adapt to optional module capabilities at runtime while ensuring interoperability with all modules at a basic level.

Characteristic and common to all CMIS compliant modules is that a well-defined set of management operations and associated data are transferred over a CMIS defined Management Communication Interface (MCI), e.g. an I2C-based interface. The basic management operations are simple and allow the host to access a 256 byte addressable memory window, with mechanisms to dynamically switch 128 byte sized data pages of a much larger management memory space into the upper half of that host addressable memory window.

Note: This limited set of basic operations and the very small byte-oriented memory window are traced back to earlier SFF specification and allow also simple transducers or transceivers to be CMIS managed. For complex modules, extension mechanisms are implemented on top of these basic elements.

The *physical form factor scope* of CMIS includes pluggable or onboard form factors such as QSFP-DD, OSFP, or COBO. However, CMIS is developed as a generic management interface specification and can be implemented in a variety of existing form factors, such as QSFP, or also in future form factors. Generic advertisement fields in the management memory map inform the host about the particular form factor and whether a module can be managed in a CMIS compliant fashion.

Note: Organizations working on new module developments are invited to use CMIS and contact the CMIS editors or the CMIS publishing organization for adding support and adapting CMIS incrementally, as needed.

The *functional scope* of CMIS includes module types which may range from electrical cable assemblies (hereafter also referred to as modules, unless cable assemblies are specifically mentioned) and active transceiver modules to versatile coherent DWDM modules with integrated framers.

The following classification can be used to distinguish functional module types

1. **data agnostic (basic) system interface modules** that map bit streams from host lanes to media lanes and vice versa without knowledge of data formats or participation in any communication protocol for that bit stream. Examples include cable assemblies and transceivers at not too high lane data rates, e.g. 100GBASE-SR4 modules
2. **data format aware (complex) system interface modules** that perform interface related data processing (such as lane deskewing and FEC codecs), e.g. 400ZR modules
3. **data link terminals**[2] (**client encapsulation modules**), i.e. modules that encapsulate host data into a newly framed media side signal format. Such modules employ framers for an independent media side server data link termination encapsulating host side signals and comprising functionality like framing, mapping, aggregation (multiplexing), switching, or distribution (inverse multiplex).

The *specification scope of this CMIS revision* covers data agnostic and data format aware system interface modules with (multiples of) eight host lanes and management communication based on one of the Management Communication Interface definitions described in Appendix B.

The extensions for modules with more than 32 (4x8) lanes, for more complex system interface modules, or for data link terminals is left to future revisions of this specification, or to external extension specifications.

[1] The management memory map defines registers and memory locations that are accessible to the host.

[2] This term is not in widespread use, but it is used here to indicate that a terminated 'server' data link exists at the media side of a module which can carry one or more 'client' signals attached at the host interface. This 'server' data link can often be operated, managed, and supervised independently of the 'client' signals.

2 References and Conventions

2.1 Industry Documents

The following documents are relevant to this specification

2.1.1 Interdependent Documents

[1] UM10204 I2C-bus specification and user manual, NXP
Rev. 6.0, 2014, https://www.nxp.com/docs/en/user-guide/UM10204.pdf

[2] QSFP-DD MSA QSFP-DD Hardware Specification for QSFP DOUBLE DENSITY 8X PLUGGABLE TRANSCEIVER, Rev. 6.0, QSFP-DD MSA, 2021, http://www.qsfp-dd.com/specification

[3] OSFP MSA Specification for OSFP Octal Small Form Factor Plugable Module, OSFP MSA,
Rev. 3.0, 2020, https://osfpmsa.org/specification.html

[4] COBO 8-Lane & 16-Lane On-Board Optics Specification, Consortium for On-Board Optics (COBO),
Rev 1.1, 2018, https://www.onboardoptics.org/specifications

[5] SFF-8024, SFF Module Management Reference Code Tables,
Rev 4.7, 2020, https://www.snia.org/technology-communities/sff/specifications

2.1.2 General Background

[6] CFP MSA Management Interface Specification Version 2.6, http://www.cfp-msa.org/documents.html

[7] SFF-8074, SFP (Small Formfactor Pluggable) Transceiver, Rev 1.0, 2001

[8] SFF-8636, Management Interface for 4-lane Modules and Cables, Rev. 2.10a, 2019

[9] SFF-8679, QSFP+ 4X Hardware and Electrical Specification, Rev. 1.8, 2018

[10] OIF Common Electrical Interface (CEI) Specifications

[11] IEEE Std 802.3-2018, IEEE Standard for Ethernet

[12] INCITS Fibre Channel Specification, https://standards.incits.org/apps/group_public/download.php

[13] IEEE Organizationally Unique Identifiers (OUI)

2.2 Sources

2.2.1 Standards and Specifications

Copies of IEEE standards may be obtained from the Institute of Electrical and Electronics Engineers (IEEE) (https://www.ieee.org).

Copies of InfiniBand standards may be obtained from the InfiniBand Trade Association (IBTA) (http://www.infinibandta.org).

Copies of OIF Implementation Agreements may be obtained from the Optical Internetworking Forum (OIF) (http://www.oiforum.com).

Copies of small form factor (SFF) specifications may be obtained from the SNIA SFF Technology Affiliate site https://www.snia.org/technology-communities/sff/specifications

2.2.2 Administrative Material

Copies of Common Language Equipment Identification (CLEI) specifications may be obtained from http://www.commonlanguage.com.

Registration information for OUIs may be obtained from the IEEE Registration Authority https://standards.ieee.org/products-services/regauth/index

2.3 Conventions

The conventions defined in this section are used throughout this specification.

2.3.1 General Conventions

Definitions
In this specification, English words or general terms may be used as **technical terms** with well-defined and specific meaning that is either different from or narrower than the normal English meaning in a general context. Technical terms are preferably defined in chapter 3 (Definitions), or otherwise when they first appear in the main text flow.

*Note: A technical term may, depending on context, be printed in **bold** to assist readers in recognizing that special meaning is attached. Typographical conventions are defined in subsection 2.3.3.*

Order of Precedence
If a conflict arises between the interpretations of text, tables, or figures, the order of precedence to resolve the conflicts is text first, then tables, and finally figures. **Exceptions**: Tables that provide detailed and primary specification text (not just overview or illustration) take precedence over text referring to those tables. Diagrams used to convey exact graphical specifications (not just illustrations), such as state transition diagrams, take precedence over text and tables

Figures and Tables
Figures and Tables providing an overview of details specified elsewhere are illustrative. Tables originally specifying data, formats, and values are normative. Not all tables or figures are fully described in the main text.

Lists
Lists sequenced by lowercase or uppercase letters show no ordering relationship between the listed items.
Lists sequenced by numbers show an ordering relationship between the listed items.

Names
Symbolic names assigned to registers, bits, fields, or constants have only local significance within this specification. While it is understood that software implementations may use similar or even identical names, this is neither assumed nor required. Names defined in this specification may change in exceptional cases.

Unnumbered Subheadings
Long text segments may be sub structured for readability by informal, unnumbered subheadings, including paragraph headings placed at the begin of a long paragraph. Such subheadings are set in bold font.

2.3.2 Notational Conventions

Field and Register Names
Field and register names are defined for specification convenience, allowing readers to refer to Memory Map locations by name rather than address.

Field and register names are recognized as one contiguous token (word without spaces) with capitalization of initials to indicate word boundaries. This notation helps to clearly distinguish field names from general text.

An asterisk * as part of a field name is used as a **wildcard** indicating that all fields with names matching the name pattern are referred to. A wildcard matching the instance identification at the end of a field array element name is often suppressed for readability.

The preferred name structure for configuration registers is <noun><verb> representing <object><action>.

Register Structures and Containers
For repeated structures of registers, a **container** name may be prepended to names of these registers to achieve globally unique names. The container name and the register name are separated by a double colon, as in the syntax **<container name>::<register name>**.

Decimal Numbers
The American notation of decimal numbering is used, with a comma as thousands separator and a period for the decimal point.

American	**ISO**
0.6	0.6
1,000.0	1 000
1,323,462.9	1 323 462.9

Non-Decimal Numbers
Hexadecimal numbers are marked with suffix **h** (e.g. 10h), often written with leading zeroes (0010h). Non-numeric hexadecimal digits (ABCDEF) are capitalized (e.g. 0Fh).

Binary numbers are marked with suffix **b** (e.g. 10000b), often written with leading zeroes (00010000b)

Numerals without a base-indicating suffix are understood to be in decimal notation (e.g. 16)

Base-indicating suffixes may be omitted for unambiguous cases like 0=0b=0h and 1=1b=1h.

Spaces may be inserted to make long hexadecimal or binary digit strings readable (e.g. 0001 0000b).

Logical Operators and Expressions
Logical operators are written in uppercase (**OR**, **AND**, **NOT**) and parentheses are used to clarify precedence.

Logical Values
Logical values are TRUE and FALSE.

The default encodings of TRUE and FALSE are 1b and 0b, respectively.

Logical Hardware Signal Values
For generic HW signals, where physical signal levels depend on the signal logic encoding, the symbolic signal levels ASSERTED and DEASSERTED are used, which correspond to TRUE and FALSE (or 1 and 0), respectively.

To allow using the symbolic signal levels ASSERTED and DEASSERTED in logical expressions of this specification, two logical predicates (i.e. mappings into the value set {TRUE, FALSE} or {1,0}) are introduced:

- **ASSERTED**(signal) maps to a logic value of TRUE (or a binary value of 1) when the signal's symbolic level is ASSERTED, and FALSE (or a binary value of 0) otherwise.
- **DEASSERTED**(signal) maps to a logic value of TRUE (or a binary value of 1) when the signal's symbolic level is DEASSERTED, and FALSE (or a binary value of 0) otherwise.

Logical Expressions
Logical expressions can be defined by a mix of bits in the Memory Map, hardware signal levels, and logic values. While Memory Map values are commonly represented as 1b or 0b, hardware signals are expressed as ASSERTED or DEASSERTED, and Boolean logic values are expressed as TRUE or FALSE.

To simplify the truth tables and logic equations in this specification, the following equivalency is used.

CMIS	Bit	Boolean Logic	Hardware Signal
0	0b	FALSE	DEASSERTED
1	1b	TRUE	ASSERTED

A don't care term in a logical expression is denoted by X

Logical Predicate Names (Transition Signal Names)
Logical predicates expressing **exit conditions** for state transitions in state transition diagrams are also referred to as **transition signals**, analog to digital state machine hardware circuit specifications.

When symbolic names are assigned to such logical predicates (a.k.a. transition signals) the names end in **S** (e.g. `ResetS`), as a mnemonic of Transition **S**ignal.

In some cases, a suffix **T** is used to indicate terms in logic expressions, as a mnemonic of **T**erm.

2.3.3 Typographical Conventions

Technical Terms
This specification uses certain English words in a specific sense as technical terms. The specific meaning of these technical terms is defined in section 3.3. To avoid possible confusion with other meanings outside of this specification, words used as specific technical terms have capitalized initials.

Narrowed Meaning
This specification may use certain English words, composites, or general technical terms in a narrower sense than the traditional English meaning. The narrowed meaning of such words or technical term is also described in section 3.3, as a heads-up for the reader. Such words are usually not marked-up typographically.

Emphasized Text
Locally emphasized text (emphasis for better readability) is typeset in **bold font**.

Auxiliary Text
Non-normative, informative, and auxiliary text, such as examples, hints, notes, or explanations is printed in *italic font* to help in visual differentiation of auxiliary text versus normative specification text.

Syntactic Variables
Syntactic variables in register names or field names may be indicated by angle brackets or by italic or bold font, as e.g. <n>, *n*, or **n**. For example, the term OutputStatusRx<n> represents the 8 bits OutputStatusRx1, OutputStatusRx2, …, OutputStatusRx8.

2.3.4 Addressing and Referencing Conventions

Note: The conventions in this subsection require understanding of specific terminology defined in section 3.3.

Referencing Managed Module Resources (by Lanes)
In this specification, a managed internal module resource is identified by reference to a **lane number** (see next paragraph) of a lane that is associated with or connected to the managed resource in question.

In cases where a managed resource status or control aspect is applicable to lanes after multiplexing or demultiplexing has occurred, the status or control aspect is applicable to all lanes of the Data Path containing the lane number given as a reference, unless otherwise indicated.

Whenever the unspecific term 'lane' is used, instead of 'host lane' or 'media lane', and the context does not clearly allow to disambiguate its meaning, a host lane perspective is assumed.

Lane Numbers
Lane numbers are used in CMIS to identify lanes and lane-related resources which are managed via addressable management registers or fields. It is important to understand both how lane numbers relate to register or field addresses and how they relate to physically identifiable hardware reference points.

The association of lane numbers and the relevant registers or fields is defined in the management Memory Map (see chapter 8). Generally, a lane number (minus one) is used as an address offset pointing to the relevant register or field instance in an array of lane related registers or fields.

The association of lane numbers and physically identifiable reference points is defined as follows:

On the host side, the ascending host lane number sequence (e.g. 1 to 8) is always mapped one-to-one to the ascending numbering sequence in the names of the relevant electrical host interface connector contacts, which are the physically identifiable lane reference points as defined in the relevant hardware specification.

On the media side there is no universal mapping of media lane numbers to physically identifiable media side entities like fibers, wavelengths, or media side connector positions. Instead, this mapping is advertised by the module (see section 8.9.8 for details).

Referencing Elements of the Management Memory Map (Bytes, Bits, Fields)
Each host accessible Byte in the structured internal management Memory Map can be identified by an address triple consisting of a Bank Index (0-255), a Page Index (00h-FFh), and a Byte Address (0-255). See chapter 8 for more information on this three-dimensional addressing structure.

Pages without Bank support implicitly have a Bank Index of 0 (which can be omitted).

The so-called Lower Memory implicitly has a Bank Index 0 and a Page Index 00h (both of which can be omitted).

The following **colon separated** Byte addressing syntax is used (*italics* denoting grammatical variables):

Bank:*Page*:*Byte*	(general Byte in the register Memory Map)
Page:*Byte*	(Byte in a page without Banking support)
0:*Page*:*Byte*	(fully specified alternate format)
Byte	(Byte in Lower Memory)
00h:*Byte*	(alternate format)
0:00h:*Byte*	(fully specified alternate format)

At each addressing level (i.e. at Bank, Page, or Byte addressing level), a **range** may be indicated by **dash separated** addresses *from-to*

Examples:
Banks 0-1
Pages A0h-AFh
Pages 0:9Fh-AFh

Bytes 1:9Fh:128-129
Bytes 0-3:9Fh:128

With Bits in a Byte indexed from 0 to 7, where index values denote arithmetic significance, the following **dot separated** notation may be used to identify a single **Bit** or a Bit **Field**:

Bank:*Page*:*Byte*.*Bit*	(single Bit)
Bank:*Page*:*Byte*.*Bit*-*Bit*	(Bit Field ranging from Bit to Bit)

Examples:

Field 2:10h:128.7-4	4-bit field of bits 7-4 of byte 128 in Upper Memory Bank 2 of page 10h
Bit 04h:128.0	least significant bit 0 of byte 128 in Page 04h (of unspecified Bank)
Byte 126	Byte at Byte address 126 in Lower Memory
Word 04h:128-129	A 16-bit word in Page 04h (Endianness conventions are specified in chapter 8)

Referencing Field Values

When clear from context, the address of a field may also denote the value stored in the field. Otherwise square brackets around a field address denote the value stored in the field.

Address = *Value*	An assignment
[*Address*] = *Value*	An expression, e.g. in a comparison

Examples:

When 10h:132 = 0 the module ….

3 Definitions

For the purposes of this specification, the following keywords, acronyms, and term definitions apply.

3.1 Keywords

This section lists verbs and adjectives intended to guide the interpretation of text in this specification in a formalized manner. Note, however, that this specification largely consists of detailed technical specifications which are presented in indicative and factual language, not in formalized requirements language.

Custom: Fields and formats described as **custom** are under control of each individual module vendor. The same custom resource may be used differently by different vendors or groups of vendors.

May: The verb **may** indicates flexibility of choice with no implied preference, both for positive and for negative statements.

Obsolete: The adjective **obsolete** indicates that an item was defined in prior specifications but has been removed from this specification. The adjective obsolescent warns that an item is about to become obsolete.

Optional: The adjective **optional** describes features which are not required by the specification. However, if an optional feature is supported, the specifications in this specification do apply. Describing a feature as optional in the text is done to assist the reader.

Note: As specified in 8.1.3.1, when an optional feature is not supported the default values for bits and bytes associated with this feature shall be zero unless otherwise stated.

Note: Not supported optional registers or fields in a supported Page are accessible. Not supported optional Pages or Banks are not accessible.

Prohibited: The adjective **prohibited** describes a feature, function, or coded value that is defined in a referenced specification to which this specification makes a reference, where the use of said feature, function, or coded value is not allowed for implementations of this specification.

Reserved: The adjective **reserved** describes elements or resources set aside for future standardization. Such resources include Bits, Bytes, Fields, and coded values. A reserved element is not available for vendor specific use.

Note: As specified in 8.1.3.1 the default value of a reserved storage element on a supported Page shall be 0. The module is required to define a Reserved field or bit as 0, but, defensively, the host should not check Reserved fields or bits for 0.

Restricted: The adjective r**estricted** describes features, bits, bytes, words, and fields that are set aside for other standardization purposes defined elsewhere. In contexts where these other standard specifications do not apply, the restricted bit, byte, word, or field is treated like a reserved bit, byte, word, or field (i.e. a **restricted** byte uses the same value as defined for a **reserved** byte).

Shall: The verb **shall** indicates a mandatory requirement of particular importance or of general nature. As is common in detailed technical specifications, simple descriptive statements of fact also express mandatory, or conditionally optional, technical requirements. Implementation of mandatory requirements is necessary to ensure interoperability with other products that conform to this specification. Note, however, that at this time no formally defined specification conformance criteria exist.

Should: The verb **should** indicates flexibility of choice with a strongly preferred alternative.

Vendor specific: Indicates that something (e.g., bit, field, code value) is not defined by this specification. Specifications of vendor specific items are provided by the relevant vendor.

3.2 Abbreviations

This section lists abbreviations, some of which are acronyms, together with their full text expansion, without explanation of meaning (with the possible exception of short hints). When abbreviated terms or acronyms need to be explained or used in a specialized sense, then the meaning of such abbreviations is defined in section 3.3.

ACK	Acknowledge
Adv.	Advertised (a register classification tag)
AOC	Active Optical Cable
ASCII	American Standard Code for Information Interchange (the numerical representation of a character)
BER	Bit Error Ratio (dimensionless) or Bit Error Rate (per unit time)
BERT	BER Testing
BOL	Begin of Life
CDB	Command Data Block
CDM	Code Division Multiplex
CDR	Clock and Data Recovery
CLEI	Common Language Equipment Identification
Cnd.	Conditionally required (a register classification tag)
COR	Clear on Read (side effect)
MSL	Management Signal Layer
CTLE	Continuous Time Linear Equalizer
CWDM	Coarse Wavelength Division Multiplexing
DFB	Distributed Feedback Laser
DPSM	Data Path State Machine
DWDM	Dense Wavelength Division Multiplexing
EC	Explicit Control (name of a control bit)
EML	Externally Modulated Laser
EPL	Extended Payload
ePPS	enhanced Pulse Per Second (a timing signal)
EOL	End of Life
FC	Fibre Channel
FEC	Forward Error Correction
FERC	Frame Error Count
FP	Fabry-Perot (a laser type)
FSM	Finite State Machine
IB	InfiniBand
IC	Integrated Circuit
ISI	Inter-Symbol Interference
I2C	Inter IC (a two-wire bus specification)
I2CMCI	I2C-based MCI (a management communication interface specification defined here)
LOL	Loss of Lock
LOS	Loss of Signal
LPL	Local Payload
LSB	Least Significant Bit (in a multiple bits context), Least Significant Byte (in a multi-byte context)
LTP	Level Transition Parameters
MBR	Module Boot Record
MCI	Management Communication Interface
MIS	Management Interface Specification
MSB	Most Significant Bit (in a multiple bits context), Most Significant Byte (in a multi-byte context)
MSM	Module State Machine
N/A	Not Applicable or Not Available
NACK	Not Acknowledged
NV	Non-Volatile
OIF	Optical Internetworking Forum
OMA	Optical Modulation Amplitude
OOR	Out of Range
Opt.	Optional (register classification)
OUI	Organizationally Unique Identifier (unique vendor code assigned by the IEEE)
PM	Performance Monitoring

PRBS	Pseudo Random Binary Sequence
PSL	Pattern Synchronization Loss
RO	Read-Only
RAL	Register Access Layer
Rqd.	Required (a register classification tag)
RW	Readable and Writeable
Rx	Receiver function of module, receive direction (media to host)
SC	Self-Clearing (side effect)
SI	Signal Integrity
SCL	unidirectional Serial Clock
SDA	bidirectional Serial Data
SM	State Machine
SFF	Small Form Factor
STD	State Transition Diagram
STT	State Transition Table
TC	Threshold Crossing
TDM	Time Division Multiplex
TEC	Thermoelectric Cooler
TWI	Two Wire Interface (an obsolescent alias for I2C)
Tx	Transmitter function of module, transmit direction (host to media)
VCSEL	Vertical Cavity Surface Emitting Laser
VDM	Versatile Diagnostics Monitoring
WDM	Wavelength Division Multiplexing
WO	Write-Only

3.3 Glossary

This Glossary defines the technical vocabulary used in this specification. The Glossary can be skimmed or skipped on first reading but should always be consulted when normative specification text is examined.

This specification uses certain English words as technical terms in a very specific sense. The specific meaning of these technical terms is defined here in this Glossary section. To avoid possible confusion with other meanings outside of this specification, words used as a technical term are usually capitalized (initials).

This specification also uses certain English words or general technical terms in a narrower sense than their general usage. The narrower meaning of such words or technical term is described in this section.

ACCESS: This all-capitals spelling denotes a READ or WRITE access via management communication interface.

ACCESS Hold-Off Period: is a period of time during which a module rejects READ or WRITE accesses. There are a number of different reasons why a module might exercise ACCESS hold-off, which are specified in the relevant section of this specification, with maximum durations specified in chapter 10.

Alarm: An **alarm** informs the host about operationally undesired situations or about critical threshold crossings of monitored observables. The module raises an alarm by setting an associated **Flag** that represents the alarm.

Application: An **Application** is a well-defined transmission function, usually defined by a specific combination of an industry standard host interface and an industry standard media interface. *See Section 6.2 for more information.*

Array: Repeated data structures in Pages of the Management Memory Map are referred to as arrays. For example, the Applications supported by a module are described in arrays of Application Descriptor data structures.

Bank: In the management memory map, a Bank (capitalized) contains several Pages having the same Page Index distinguished by a Bank Index. In the context of firmware management, the term bank refers to a non-volatile storage location for a firmware image.

Banking: A Memory Map architectural feature that allows modules to implement multiple Memory Map Pages that have the same Page address, effectively providing additional memory depth. For example, banking is used to provide additional lane related registers for more than eight lanes. *See Section 8.1 for more information.*

Bit (storage) versus **bit** (value): in common technical parlance the term bit may refer either to a binary value or it may refer to a binary storage element (a binary variable) that is usually part of a larger addressable storage element (e.g. part of a Byte). For clear and context-free differentiation in this specification, the spelling **Bit** (capitalized) is used to refer to a binary storage element, and the spelling **bit** (lowercase) refers to a binary value, which may be stored in a Bit.

Byte (storage) versus **byte** (value): in common technical parlance the term byte may refer either to a value or it may refer to a storage element (a variable). For clear and context-free differentiation in this specification, the spelling **Byte** (capitalized) is used to refer to an addressable storage element, whereas the spelling **byte** (lower case) refers to a value, which may be stored in a Byte or used in a computational expression.

Channel: A (physical) channel is characterized by the resources (e.g., frequencies) within signal transmission media that are used to carry the modulated analog signal representing the data transmitted over lanes. Examples include baseband electrical and passband optical channels.

Checksum: A value derived from a block of digital data for the purpose of detecting errors in that block.

Configuration (data): The term configuration is used in a broad sense to denote data which influence module behavior that can be modified by the host, i.e. excluding administrative data. See also Status and Trigger.

Conformance (Flagging with Interrupt generation rules): In this specification, the wording conformance, Interrupt conformance, Flag conformance, and **Flagging conformance** all relate to requirements, under which conditions the module may set Flags which potentially raise Interrupts.

Controls: Writeable fields that affect module operation are sometimes referred to as **controls**, in contrast to writeable fields which just store passive data. Typical examples of controls are **configuration** data fields and **trigger** fields where writing to the field triggers some processing in the module.

Data Path: Host lanes and media lanes grouped together into a logical concept. A Data Path represents a group of lanes that will be initialized, used, and deinitialized together. The Host Interface of a Data Path can carry a multi-lane signal (e.g. CAUI-4) or a single-lane signal (e.g. 25GAUI). Likewise, the Media Interface can consist of a single lane or of multiple lanes.

Default: The Default value of a management attribute is the value that a host would retrieve immediately after the management memory has become accessible (at some stage after power-on, or post-reset, when the management communication interface has been brought up)

Dynamic (status): A dynamic status register displays status information that can change during operation.

Fault: In this specification the term fault is used exclusively for module detected conditions that incur a risk of physical damage to the module or its environment or that incur a safety risk. A partial or full loss of functionality or a deviation from desired state or behavior is called failure or error.

Field: In this specification, a **field** (lower case) is an informal notion referring to some scalar data element that is stored or accessed in management memory. It could be a portion of a Byte or a value larger than a Byte. When used with precise meaning, a **Field** (capitalized) is simply a part of a **Register** that is sub structured.

Firmware: The firmware of a module may be an aggregate or bundle consisting of multiple components such as executable code or data resources for multiple processors or storage locations. The individual components of the module firmware are not exposed to or visible via CMIS, albeit the firmware update mechanism allows vendors to support access to individual firmware components.

Firmware Update: This term refers to operations on firmware stores and to the function to pass control to another image. Operations on image stores include **download**, **upload**, and **copy** firmware images (to or from or between a module's internal firmware store). The operation to pass control is **activate**. Firmware update can serve both purposes of **upgrade, repair,** and **downgrade** and may occur either immediately (**switch**) or on the next restart (**commit**).

Firmware Upgrade: This term is sometimes used as a feature name, but it may be misleading. This specification prefers the neutral term Firmware Update or Firmware Management.

Flag: In this specification, a **Flag** (capitalized spelling) is a clear-on-read **Latch** with associated maskable **Interrupt** generation. Each Flag comes with an associated **Mask** bit. Interrupt generation from the Flag is suppressed when the associated Mask bit is set. Flags are used to reliably and (optionally) immediately inform the host about the onset of conditions or the occurrence of events. *Note: Using the word flag (lowercase) in a general sense to denote a simple binary indicator bit is discouraged in this specification. Such bits will be called status bits or indicator bits instead (see Status).*

Flat Memory Module: A Flat Memory module provides 256 Bytes of immediately addressable management memory. Unlike a Paged Memory module, a Flat Memory module does not support dynamic Paging into the Upper Memory. The simplest form of a Flat Memory module supports only constant read-only data.

Note: In this version of CMIS, flat memory modules are assumed to support only read-only static data as provided by an EEPROM. See sections 6.3.2.3. and 8.2.

Freezing (Holding): Temporarily freezing dynamically updated registers for consistent readout. This is an alternative to sampling the results into a separate result register.

Note: In CMIS, freezing is used in the VDM feature, whereas sampling is used in Diagnostics Monitoring.

Gating: A gated measurement or monitor runs for a specified amount of time, often triggered by an event. In on-demand measurements, the gating start event is writing a trigger bit; there is then an unobserved time between the end of the measurement and the start of the next one. In periodic measurements, the result of a gating period is sampled in result registers while the next gating period is started without a gap. Ungated measurements are started and stopped on host-command.

Gray Coding: used with PAM4 modulation. Defines the mapping of 2 binary bits into 4 levels.
(0,0) maps to level 0
(0,1) maps to level 1
(1,1) maps to level 2
(1,0) maps to level 3

Host Interface: The host interface is the high-speed electrical interface for connecting the module to a host system via electrical high-speed lanes. The requirements of a specific host interface are defined in the associated industry standard for that interface. See Section 6.1.

Interface: The general term **interface** is ambiguous. In this specification three interpretations prevail:

- Interface as a **device to device** interconnection (e.g. Host Interface, Management Interface)
- Interface as a **device to media** attachment (e.g. Media Interface)
- Interface as an **adapter device function** processing signals carried on an interconnection (e.g. when functions or registers of a module's interface are described).

Interrupt: The generic hardware signal that, when asserted, represents a pending interrupt request from module to host is called **Interrupt** . In this specification, the Interrupt is asserted as long as any **Flag** is set with its associated **Mask** cleared.

Interrupt Flag Bit (obsolete): The term interrupt flag or interrupt flag bit was used side by side with flag or flag bit, in previous versions of this specification. These terms are now obsolete and replaced by the term **Flag**.

Interval Statistics: A raw performance monitor for an observable in the module provides a real-time value, i.e. a current value **sample**, on request. Enhanced performance monitoring adds computation of statistics over a periodic or on-demand **monitoring interval**. The **Interval Statistics** collected over a monitoring interval typically include minimum value, maximum value, and sample mean (average), also known as min, max, avg.

Lane: The term **Lane** (capitalized) denotes the **module resources** or facilities associated with processing or transport of one serial high-speed signal in one of the module interfaces (ports).

A secondary meaning of **lane** (lowercase) denotes the connector contact points for input or output, or the connected channels in interconnection media carrying, a serial digital data stream (e.g. a single-ended electrical wire, a differential electrical wire pair, or a wavelength channel in optical fiber media).

Note: Lanes should not be confused with signals: lanes carry signals like roads carry traffic.

Latch: A latched Bit or Latch, for short, is a read-only Bit in management memory with a sticky value (1b): Once the module has written the sticky value to the Bit, the value is held until the Bit is eventually cleared as an implicit side effect of reading by the host (clear on read). Latches are used to reliably inform the host about events or possibly transient states. *Note: In this specification, Latches with maskable interrupt generation are called Flags.*

Latching (obsolete): This term has been used in CMIS 4.0 to describe storing a real-time value into a result register. It is now replaced by the term 'sampling'. Latches for counters are now referred to as 'results'.

Lower Memory: The 128 bytes of host addressable memory addressed by byte addresses 00h through 7Fh.

Mask: In this specification a Mask bit or **Mask** is a host writable Bit in management memory which, when set, suppresses (or ceases) Interrupt generation by its associated **Flag** bit.

Management Communication Interface (MCI): The data communication interface and interconnection between host and module that is used to transport and implement READ and WRITE register access primitives across the interconnection.

Management Memory: The term **management memory** refers to the internal storage of all host **accessible** management data (**registers**, **memory**). Note that host accessible is not the same as host addressable because a host may need to switch accessible content into Upper Memory such that it becomes host addressable.

Management Memory, Addressable: The **addressable management memory** which is the memory that can be directly accessed by READ and WRITE primitives, via the MCI. In CMIS this a 256 Byte area consisting of Lower Memory and Upper Memory.

Media Interface: A media interface is defined as the high-speed interface for attaching the module to an interconnect medium, such as wires or optical fibers. The requirements of a specific media interface are defined in the associated industry standard for that interface. See Section 6.1.

Memory: In this specification the term memory is often used interchangeably with the term register. Moving forward, **memory** will preferably be used for a host accessible storage location when that storage location is used for temporary or permanent data storage and when there is no module functionality associated with the data stored in that location (cf. **register**, when functionality is associated).

Memory Map: In this specification the term management **Memory Map** refers to the conceptual organization of all host accessible management data, independent of whether the module stores these management data fully or partially in classical hardware **registers** (evaluated by module hardware) or in memory **locations** (virtual registers evaluated by module firmware).

Module: Pluggable transceivers and active or passive cable assembly terminations that plug into a host receptacle, or onboard transceivers, such as, but not limited to, those of QSFP-DD, OSFP, COBO, and QSFP form factors are hereafter referred to as modules unless cable assemblies are specifically mentioned. In CMIS, one classification of modules is based on the management memory map, where modules are distinguished as paged memory modules or as flat memory modules (unpaged).

Monitor: A monitor is the function to measure an observable or to obtain a value from an estimator and to make the results of the measurement or estimation available in a register, for consumption by the host. Many monitors do not just sample an observable or process but require measurement or estimation time until the monitor value can be updated. Many monitors in CMIS also provide additional functionality such as threshold crossing detection (for low/high warning/alarm indications) or statistics (cumulative or interval).

Multi-Wavelength Modules (Multi-Wavelength Applications): Modules running Applications using a fixed or tunable WDM grid or supporting parallel Applications using different wavelengths.

Muting: This generic term refers to any action with the intended effect that the output of a signal source facility becomes **quiescent**: i.e. a muted output is quiescent. A host mutes an output by management command or configuration, either by **disabling an output** or by **forcing** an output to be **squelched**. The module can mute an output when an auto-squelch controller is enabled. Note that unmuting an output is not the same as enabling an output: the inverse of output disable/squelch is therefore to un-disable/un-squelch an output, respectively.

NV: Non-Volatile (NV) memory is persistent and maintains stored information permanently between writes, also when the module is not powered

Observable: An observable is something that can be measured, monitored, estimated, or otherwise observed. In this specification, observables are the inputs to performance monitors which in turn provide real time samples, threshold crossing supervision, and sometimes sample statistics. Previous versions of this specification have also used terms like parameter, attribute, or monitor.

OM2: Cabled optical fiber containing 50/125 um multimode fiber with a minimum overfilled launch bandwidth of 500 MHz-km at 850 nm and 500 MHz-km at 1300 nm as IEC 60793-2-10 Type A1a.1 fiber.

OM3: Cabled optical fiber containing 50/125 um laser optimized multimode fiber with a minimum overfilled launch bandwidth of 1500 MHz-km at 850 nm and 500 MHz-km at 1300 nm as well as an effective laser launch bandwidth of 2000 MHz-km at 850 nm in accordance with IEC 60793-2-10 Type A1a.2 fiber.

OM4: Cabled optical fiber containing 50/125 um laser optimized multimode fiber with a minimum overfilled launch bandwidth of 3500 MHz-km at 850 nm and 500 MHz-km at 1300 nm as well as an effective laser launch bandwidth of 4700 MHz-km at 850 nm in accordance with IEC 60793-2-10 Type A1a.3 fiber.

OM5: Cabled optical fiber containing 50/125 um laser optimized multimode fiber with a minimum overfilled launch bandwidth of 3500 MHz-km at 850 nm, 1850 MHz-km at 953 nm and 500 MHz-km at 1300 nm as well as an effective laser launch bandwidth of 4700 MHz-km at 850 nm and 2470 MHz-km at 953 nm in accordance with IEC 60793-2-10 Type A1a.4 fiber.

OMA: Optical Modulation Amplitude: The difference between two optical power levels, of a digital signal generated by an optical source, e.g., a laser diode.

Operational: A functional resource is **operational** if it has been fully configured and initialized so that it is able to perform its function according to specification. Note that this term represents only a configuration and initialization view of the resource; the actual transmission behavior of an operational resource may depend on other conditions, such as valid input signals being available.

OSNR: Optical Signal to Noise Ratio: The ratio between the optical signal power in a given signal bandwidth and the noise power in a given noise reference bandwidth.

Page: A management memory segment of 128 bytes that can be mapped (paged) into the host addressable Upper Memory, which is host addressable by byte addresses 128 to 255 (80h to FFh)

Paged Memory Module: A module supporting a paging mechanism where parts of a module's Management Memory are dynamically mapped into a smaller host-addressable memory window. Usually a Paged Memory module runs firmware on a module internal controller. See also Flat Memory Module.

Parameter (controlled variable versus uncontrolled observable): The term **parameter** denotes a controlled or controllable property of an object or function, i.e. an attribute that is under control of the object's environment. Note: The term parameter is also used loosely to refer to an observable, i.e. to an emergent rather than controlled characteristic of an object.

P_{av}: Average Power: The average optical power P_{av}

Performance Monitoring (PM): This term is informally used to refer to the functionality of Interval Statistics provided for an intensive observable and for associated threshold crossing Flags.

Polling: The repeated host-driven retrieval of module status data by the host. See also Reporting.

Port: A point of signal entry or signal exit at the module boundary. Sometimes short for Port Function.

Port Function: The functions and resources within the module that process a signal before leaving and after entering the module through a module port. Colloquially this is often called an interface.

Post-cursor equalization: Rx output means used to reduce post-cursor ISI.

Power Mode: The Power Mode of a module is either **High Power** or **Low Power**. The maximum module power consumption in Low Power Mode is defined in the relevant hardware specification. The maximum module power consumption in High Power Mode is module implementation dependent and is advertised. The purpose of the Low Power Mode is to ensure that a plugged-in module can be examined by the host and rejected (not operated) if, e.g., the operational power consumption would exceed the host's capabilities. Power Mode, the power level the module is permitted to consume, should not be confused with the Power Class which represents an advertised range of power consumption limits defined in the relevant hardware specification.

Pre-cursor equalization: Rx output means used to reduce pre-cursor ISI.

Pulse Amplitude Modulation, four levels (PAM4): A modulation scheme where two bits are mapped into four signal amplitude levels to enable transmission of two bits per symbol.

Quiescent: An output is **quiescent** when the signal being output **cannot be detected as a useful input** signal by a remote input. An input receiving the signal from a quiescent output must detect a loss-of-signal (LOS) condition, as specified in the relevant interface standard. Note that there may be multiple methods to achieve quiescence (e.g. for modulated signals, either the carrier or the modulation or both can be removed).

READ: This all-capitals spelling denotes a management register READ operation of one or more bytes, which is executed via a management communication interface.

Real Time Value (Sample): A **real time value**, **current value**, or **sample** (preferred) is a value that is sampled or evaluated (conceptually) at the time of a read request and returned with the read request. Depending on the observable, sampling a new value may include some estimation or measurement time, usually short, such that read operations faster than the sample measurement or estimation interval will yield the same value. See also Statistics.

Register: In this specification a **register** denotes a host accessible management memory element with associated management functionality (displaying status or exercising control) which can be accessed by read or write primitives. In CMIS there are **single-Byte registers** (also referred to as Bytes) and **multi-Byte registers**. Registers may be sub-structured into **fields**. Sometimes **location** is used as a synonym.

Note: Effective addressability refers to the fact that memory may be mapped dynamically into a host addressable memory window. Note that addressability has no implication of how a register is implemented, be it a true register interpreted by module hardware or just a memory location interpreted by module firmware. Apart from registers, there are also addressable memory elements that serve only for (temporary) storage without fixed meaning aside (e.g. memory used for message transfer).

Note: CMIS Multi-Byte registers require a series of Byte accesses, which raises questions of data coherency for the register value during a register modification (see section 5.2.3).

Receiver, Rx: An electronic or opto-electronic component (Rx) that converts an input signal (optical or electrical) to an electrical (non-retimed, retimed, or processed and reformatted) output signal. In this specification **Receiver** (**Rx**) refers to the module function as a whole, not to interface adapter circuity. A module's Rx converts media interface signals to host interface signals.

Reporting: In this specification, setting a binary Flag (potentially) causing a host Interrupt has the purpose to actively **report** some condition to the host. Unlike most other management interactions, reporting originates in the module (or target). The term **indication** is also used, especially when conditions are reported.

Reporting Register: A reporting register is updated by the module to provide information to the host.

Result Register: A result register contains the results to be consumed by the host. For monitoring processes with repeated result generation, the result updates occur either time-based or event-based (with handshake).

Sample: See Real Time Value and Result Register.

Sampling: Sampling data from a permanently running process copies an instantaneous current value (real time value) into a stable sampling **result register** (sample register). The stored sample can be consumed by the host while the internal process continues. In many cases a side effect of sampling is that some internal process is reset and restarted.

Note: Colloquially this sampling process is also called 'latching' of results. This "latching" terminology, however, is discouraged because latched bits (used to record some binary state) often have clear-on-read access mode, which is often not the case for numerical results.

Separable (interface): In this specification, a physical interface or interconnection is called **separable** (disconnectable) when the physically interconnected parts (e.g. module and optical fiber) can be disconnected and detached from each other. Separable interfaces are connectorized and **disconnectable**.

Signal: The term signal is ambiguous. **Physical** signals are carried over physical interface **Lanes**. These signals are sometimes called **single-lane** signals. Differential pairs of physical signals carrying one serial data stream are usually considered to belong to the same Lane. **Logical** (data) signals may be carried over multiple physical lanes and are sometimes called **multi-lane** signals.

Signal Integrity (SI): host side electrical layer equalization (or pre-equalization) **parameters** and **functionality** to counteract signal distortions that come with high-speed electrical signals or long and connectorized electrical signal paths. The SI related functionality ranges from fixed pre-defined equalization using programmable equalizers configured at link startup time to run-time adaptive equalizers.

SNR (Signal to Noise Ratio): Here, the ratio of signal power to the noise power, always (implicitly) expressed in decibels

Squelching: The act or function of intentionally muting an enabled module output to become quiescent such that no signal is detectable at the remote end. Squelching is often an automatic module reaction, e.g. when no module input is available for the output in question. This automatic control reaction (which can be disabled) is also called squelching colloquially, but should better be called **auto-squelching**. It is also possible to squelch an output on host request. This is called **forced squelching**.

State: The term state is used in the context of conceptual "state machines" that are described by state transition diagrams and indicate modal (i.e. state-dependent) behavior.

State Machine: In specifications, a (finite) **state machine** (FSM, SM) is a conceptual object obeying the state change behavior specified in a state transition diagram or, equivalently, a state transition table.

State Machine Modeling: The syntactical and semantical rules for the elements allowed or required in a state transition diagram establish a well-defined modeling language, which can be translated into an executable state machine model. In CMIS, two semi-formal Moore state machine types are specified, while the actual behavior in each state is specified is specified in natural language.

State Transition Diagram: The states and state transitions of the most basic state machine (Moore SM) can be described with a formal diagram of named nodes (representing states) and arrows (representing state transitions) where the arrows are tagged with a predicate expressing a (possibly compound) logical condition that must be true for the transition to occur. The semantics of what the states represent or the general behavior of an object with state-dependent behavior described by the state transition diagram are not formalized in a Moore SM and are therefore added as informal text.

Static: A field is **static** when the values in the field are not changed by the module except at power-up, or as a result of a reset. Quasi static fields that are not changed by the module as long as a programmed Application persists are also called static fields, for short.

Statistics: A **statistic** or **statistics** represent aggregated results computed over a set of **samples** that have been collected in a currently running or past monitoring or **statistics interval**. Typical statistics include minimum and maximum sample value observed in the monitoring interval, and the average over all sample values. Taking samples for the statistics may require its own (shorter) estimation or **measurement interval**.

Status (data): The term status is used in a broad sense for any host readable (but usually not writeable) information. It includes variable binary indications and sample values as well as constant data. See also Controls, Configuration and Trigger.

TEST: This all-capitals spelling denotes a test to determine if the module is ready for READ or WRITE access.

Transmitter, Tx: an electronic or optoelectronic circuit (Tx) that converts an electrical input signal from a host to a signal suitable for the communications media (optical or electrical). In this specification **Transmitter** (**Tx**) refers to the module function as a whole, not to interface adapter circuity. A module's Tx converts host interface signals to media interface signals.

Trigger: The effect of writing a value of 1b to a (usually self-clearing) trigger bit in a register is that some process is started (triggered). See also Controls, Configuration and Status.

Unit (defined for a register): A specified **unit** of a numerical register defines what quantity is associated with a numerical register value of 1. A unit is often specified as the product of a conventional measurement unit (1m, 1kg, 1s, etc.) and a dimensionless scaling factor. Colloquially, this quantity is sometimes loosely referred to as the "LSB". In other formulations the unit of a register value is expressed as "in increments of".

Upper Memory: The 128 Bytes of host addressable memory addressed by Byte addresses 80h through FFh. The actual content of Upper Memory depends on the host-controlled selection of Bank and Page to be mapped by the module into Upper Memory.

User: The word user refers to any party who has access to a CMIS module, including module makers, module OEM users, host systems.

Valid (signal): A signal on a lane is called **valid** if its (electrical or optical) **physical signal characteristics** are **stable** (i.e. not transient) and **proper** (i.e. compliant to the pertinent interface specification for a useful state of transmission), and if the **data** carried on that physical signal is **correctly formatted** at the level of data processing performed in the module. Signal validity plays a crucial role during initialization, because only a valid input signal may allow a module or host to successfully and correctly initialize all resources required to propagate or process the signal. Providing only valid output signals avoids scenarios where the receiving end is misguided in its initialization, which otherwise may lead to undesired link flaps.

Volatile (register or memory location): A volatile register or memory location is not persistent and does not preserve its value over a regular reset or a power cycle.

Warning: A warning is like an alarm. Only the implied severity is different. The reporting mechanism via Flags is the same as for alarms. See Alarm.

WRITE: This all-capitals spelling denotes a management register WRITE operation of one or more bytes, which is executed via a management communication interface.

3.4 Data Types

The following data types allow this specification to define the interpretation of data in multi-byte registers concisely.

The default Endianness (storage order) for numerical data types is defined in section 8.1.3.4 (Big Endian).

In case of non-default storage order (Little Endian), the non-default Endianness must be explicitly specified.

ascii ASCII character

Bool 1-bit Boolean value with encoding TRUE = 1 and FALSE =0
Note: Boolean variables are often named such that the interpretation is self-explanatory (e.g. Bool Initialized instead of Bool InitializationStatus)

F16 A compact 16-bit floating point data type (originally defined in SFF-8636) used to represent non-negative real numbers. Representable values are expressed in the form $m \cdot 10^{s-24}$ where the mantissa m ranges from 0 to 2047 and the scaled exponent s ranges from 0 to 31. Hence, the smallest representable non-zero number is $1.0 \cdot 10^{-24}$ and the largest representable number is $2.047 \cdot 10^{10}$. An F16 value is stored in two bytes (where byte 1 is stored at the lower byte address) with the following representation (which is Big Endian for the 11-bit mantissa):

Byte Number	Bits	Description
1	7-3	Scaled Exponent (s)
	2-0	Mantissa (m), most significant bits 10-8
2	7-0	Mantissa (m), least significant bits 7-0

S8 8-bit signed integer (2's complement) with value range -128 to 127

S16 16-bit signed integer (2’s complement) with value range -32768 to 32767
S32 32-bit signed integer (2’s complement) with value range – 2,147,483,648 to 2,147,483,647
S64 64-bit signed integer (2’s complement) with approximate value range -9.22·10^{18} to 9.22·10^{18}

U8 8-bit unsigned integer with value range 0 to 255, stored in a single byte
U16 16-bit unsigned integer with value range 0 to 65535, stored in two bytes
U32 32-bit unsigned integer with value range 0 to 4 294 967 295, stored in four bytes
U64 64-bit unsigned integer with value range 0 to 18,446,744,073,709,551,615, stored in eight Bytes

U<i> <i> < 8, <i>-bit unsigned integer with value range 0 to $2^{<i>}$-1, stored in a field

X16 Arbitrary 16-bit numerical data type (F16 or S18 or U16)

3.5 Binary Data Operations

A Bit (variable) is said to be **set** when its current value is 1 and it is said to be **cleared** when its current value is 0. In other contexts, however, the same wording can have action semantics, where a Bit (variable) is said to be **set** when a value of 1 is written and **cleared** when a value of 0 is written. This ambiguity between state and action is best avoided by using active grammar (someone sets or clears a Bit) for activities and passive grammar for states (a bit is set or cleared).

To shorten specification text, the following verbs are used with specific meaning when describing management operations on binary data in registers or fields.

Set	Write 1b	(set X means X = 1b)
Clear	Write 0b	(clear X means X = 0b)
Raise	Change value from 0b to 1b	(raise X means X = 1b when X = 0b before the assignment)
Cease	Change value from 1b to 0b	(cease X meansX = 0b when X = 1b before the assignment)

Assert	Assign a numerical value representing the Boolean value TRUE
Deassert	Assign a numerical value representing the Boolean value FALSE

Note that all these verbs except ‘set’ allow the reader to distinguish active operation activity from passive operation result (e.g. ‘raise’ versus ‘raised’).

4 Introduction and General Description

As a management interface specification CMIS defines the management interface and associated protocols for all required and certain optional management interactions between a CMIS aware host and a CMIS compliant module that are relevant for the host using the module in an application.

The remainder of this chapter is intended to provide high level orientation and informal overview. Normative specifications are found in the subsequent sections.

4.1 CMIS Compliant Modules

A CMIS compliant module has two main physical interfaces related to a module's mission of bidirectional signal or data transmission, and one CMIS compliant management interface for the host to manage module operation:

- The **host interface** connects a module and its host system for the purpose of exchanging high-speed electrical signals that the module transmits to and receives from a remote system. The host interface is decomposed into a set of host lanes. The host interface is often viewed as a bidirectional entity but sometimes it is viewed as a pair of two related unidirectional interfaces. This will be clear from context.
- A **host lane** carries a differential electrical high-speed signal over a named pair of electrical wires (or named contacts in a connector). Sometimes the pair of unidirectional host lanes used for bidirectional transmission are also referred to as a host lane. This will be clear from context.
- The **media interface** connects a module and the transmission media that interconnect the module to its far end peer. The media interface carries high-speed electrical or high-speed optical signals. The media interface is decomposed into a set of media lanes.
- A **media lane** carries one media dependent high-speed signal over a pair of copper wires, optical fibers, or optical wavelength channels (in DWDM or CWDM fiber links). The high-speed signal on a media lane may also comprise a multiplex of lower rate signals (TDM or CDM). Sometimes the pair of unidirectional media lanes used for bidirectional transmission are also referred to as a media lane. This will be clear from context.
- The **management interface** allows the host to manage the module's functionality and operation. It carries low-speed signals for management data communication and some discrete low-speed signals for low level control and status.
- Additional interfaces (interconnections) e.g. for power supply, reference clock signals, pulse per second signal, etc. will or may exist but are not considered here.

A CMIS compliant module provides standardized signal or data transmission functions to its host system and each type of transmission function is referred to as an **Application**. In support of an Application the module propagates or processes signals between one or more host lanes and one or more media lanes.

A CMIS compliant module may also provide resources for parallel and mutually independent Applications. The lanes and module internal resources carrying one Application are called a **Data Path**. A Data Path may use only a subset of the host interface lanes or of the media interface lanes.

In CMIS compliant modules, symmetry is expected between the Tx hardware structure and the Rx hardware structure; for example, Tx lanes that are multiplexed in the Tx are likewise demultiplexed in the Rx. Each Application selected by the host is applied to the same Tx and Rx host lanes.

As described in section 8, the data structure of the CMIS management interface favors modules with 8, 16, or 32 lanes, because the core management features are prepared for modules with up to 4 x 8 lanes; future extensions are, however, possible.

4.2 CMIS Management Protocol Layers

The CMIS management interface is best understood in terms of its protocol layers (top to bottom):

- The **management application layer** specifies **meaning** and **effects** of management operations on functions or behaviors of the module as well as any associated behavioral rules and protocols. This layer and its representation as a set of management **hardware signals**, management **register accesses**, or **messages exchanges** is described in chapters 6, 7, 8, and 9.
- An optional **message exchange layer** implemented on top of the essential register access layer provides primitives for management message exchange between host and module. Implementation of this layer is described in sections 7.2 and 8.15.
- The essential **register access layer** provides the primitive management operations in CMIS which are **reading** and **writing** bytes from or to a 256-byte addressable memory window. The mechanisms and protocols pertinent to this layer are defined in chapter 5. The addressing extensions required to

effectively access a larger management memory space are described in chapter 8.

- The **data transfer layer** underlying the register access layer provides the data transmission protocols required to let the host and the module exchange basic management operations (and associated data) via the physical interconnection layer. A set of data transfer stack variants is described under the name Management Communication Interface (MCI) in Appendix B.
- The **physical interconnection layer** is the lowest layer comprising the electrical and mechanical interface elements (that are connected to carry elementary signals) as well as the associated protocols i.e. rules for using those interface elements. Since CMIS is targeted to multiple form factors, the physical layer is defined in the relevant module form factor specification. In CMIS, generic signal names are used instead of form-factor specific signal names. Refer to Appendix A for the association between generic signal names and form-factor specific signal names. Electrical interfaces are also specified in Appendix B by reference to other standards [1] or form factor hardware specifications such as [2][3][4].

The hardware specification of a form factor that supports or requires CMIS based module management contains mechanical and electrical specifications and defines which data transfer stack described in Appendix B is used.

Note: Currently a single MCI variant based on the I2C specifications [1] is described in Appendix B.

4.3 CMIS Compliance and Vendor Specific Extensions

CMIS is designed to enable two important features

- **Vendor-agnostic standard management**: CMIS compliant hosts shall be able to manage optional or required standardized functionality of CMIS compliant modules using only CMIS standardized management functionality
- **Optional custom extensions**: CMIS compliant modules shall be able to provide optional enhanced custom functionality and optional custom management extensions

Note: Usually access to custom extensions may be limited or protected, but this is not technically assumed

Custom extensions require hosts that are entitled, aware, enabled, and willing to use those custom extensions. Any custom extension should therefore offer two important non-obvious properties:

- **Optionality**: A CMIS compliant host shall always be able to use a module with custom extensions as a plain CMIS compliant module, using only CMIS compliant management of standard functionality (as if there were no extensions).
- **Compliant default behavior**: A CMIS compliant module plugged into a CMIS compliant host slot shall initially provide its advertised standard functionality, independent of any vendor specific custom management extensions present. The module must not assume that its current host is capable to use or aware of its custom extensions.

Note: A host may be ignorant of custom management extensions, or may not have access, or may be not willing to use them in a given context, no specific scenario is assumed here.

Note: One subtle consequence is that all custom interrupt sources must remain silent until a host who is aware of and ready to use these extensions explicitly unmasks the custom interrupt sources.

4.4 Document Overview

The remainder of this specification is organized as follows:

- Chapter 5 – Management Interface describes generic hardware signals for module management and the register access layer providing READ and WRITE access to an immediately host addressable space of 256 byte-sized registers or memory locations in the module (the lower layers of the management interface stack are described in Appendix B)
- Chapter 6 – Core Management Features describes the core features and behaviors that are required in each CMIS manageable module, including hardware only modules
- Chapter 7 – Advanced Management Features describes optional (advertised) advanced management features which require more complex module firmware
- Chapter 8 – Module Management Memory Map describes the management application layer of CMIS, which is based on a three-dimensionally addressed byte-organized management memory with register (or memory) read and write access primitives
- Chapter 9 – CDB Command Reference provides the message catalogue of a memory based messaging mechanism called Command Data Block (CDB)
- Chapter 10 – Management Timing Specifications collects specifications related to timing parameters and timing dependencies

Appendices provide further information and examples.

- Appendix A –Form Factor Specific Signal Names describes the mapping of form factor specific signal names to CMIS generic signal names and the encoding of logical values
- Appendix B – Management Communication Interface Definitions describes data transfer and physical layers available to implement the essential register access operations of CMIS
- Appendix C – Examples of Application Advertisements provides examples of how module Applications are advertised
- Appendix D – Examples of Initialization and Deinitialization describes example sequences of events and interactions illustrating how a host may initialize or deinitialize a module
- Appendix E – Illustration of Applying Control Sets provides illustration of the command handling associated with configuration changes.
- Appendix F – Examples of Diagnostic Features Usage provides illustrative examples of how diagnostics features might be used
- Appendix G – Specification Evolution and Maintenance Notes documents guidelines for future evolution of this specification
- Appendix H – Name Mapping CMIS 4.0 to CMIS 5.0 (Temporary) describes the guidelines that have been used to transform the informal register naming of CMIS 4.0 into software-ready names in CMIS 5.0

5 Management Interface

The Management Interface may be understood as the means or mechanisms required by a management application on a CMIS host to manage a CMIS compliant module; it can be partitioned as follows:

- A **Management Signaling Layer** (**MSL**) provides electrical **hardware** control signals with directly assigned management functionality (such as module reset or interrupt request)
- A **Register Access Layer** (**RAL**) provides the elementary mechanisms for **software** to READ from and WRITE to Bytes in the directly addressable management memory window of a module

The side-by-side combination of RAL and MSL provides a set of basic **management mechanisms** available to a management application.

Due to the physical separation between a managing host and a managed module, data communication is needed to implement those basic management mechanisms across the physical interconnection (which itself, confusingly, is also often referred to as the management interface). The data protocol stack used for this purpose is called the **Management Communication Interface** (**MCI**) and is specified in Appendix B.

Note: This partitioning serves a specification purpose and does not constrain implementation.

5.1 Management Control Signals (MSL)

The following discrete management control hardware signals are required by CMIS

- a signal allowing the host to request a module reset (**Reset**)
- a signal allowing the host to control full or partial power up of the module (**LowPwrRequestHW**)
- a signal allowing the module to assert an interrupt request (**Interrupt**) to the host

These generic hardware signal names are mapped to form factor dependent signal names in Appendix A.

In CMIS the values of these generic hardware signals are denoted as either ASSERTED or DEASSERTED, with physical signal value mapping defined in Appendix A.

The usage of these management control signals is described in chapter 8.

5.2 Management Register Access Layer (RAL)

Read and write access to the addressable memory of a module (as described in chapter 8) are the basic module management operations available to a host. These access operations are implemented (emulated) by transactions over the Management Communication Interface (MCI) described in section 10.2 and Appendix B.

5.2.1 Registers in Addressable Memory

The directly host addressable management memory consists of 256 Bytes and is divided in two segments, called **Lower Memory** (Byte addresses 00h through 7Fh) and **Upper Memory** (Byte addresses 80h through FFh).

Note: The (indirectly) accessible management memory of a CMIS module is usually larger but requires a dynamic mapping mechanism for the Upper Memory which is described in chapter 8.

In the remainder of this section, read and write access to the directly addressable management memory of a module is simply referred to as register access (**ACCESS**).

5.2.2 Register Access Methods for Addressable Memory

Note: The following definitions of READ, WRITE, and TEST primitives are for specification purposes only and do not imply that software implementations must use drivers with interfaces of that signature.

For timing specification related to management register access see section 10.2.

5.2.2.1 Read Access (READ)

A read access can be modeled by the following abstract function:

VALUE = **READ**([ByteAddress,] N), $1 \le N \le 8$

A host may read N bytes of addressable module management memory in a READ access.

Note: A READ of more than 8 bytes may be allowed when specified explicitly in chapter 8.

The starting byte address for the read access is optionally specified by the host (ByteAddress). If omitted, the module uses an internally maintained **current byte address** as the starting byte address (see section B.1).

A successful READ access returns N bytes by VALUE.

Side effects of a READ (such as e.g. clear-on-read) may be specified for certain Bytes (see section 8). Such side effects are not guaranteed to be completed at the time when the READ is finished.

A READ access may temporarily be **rejected** by the module, for generic reasons specified in section 5.2.4 and specific reasons specified in chapter 8.

A rejected READ is modeled by returning an abstract REJECTED value that is distinguished from all possible VALUE byte strings returned by a successful READ.

Note: Syntactic or representational details are not important here, but the value REJECTED could be imagined, for example, by an X16 value outside of the X8 subrange representing possible byte values.

A rejected READ access can simply be retried until eventual success.

5.2.2.2 Write Access (WRITE)

A write access can be modeled by the following abstract function:

SUCCESS = **WRITE**(ByteAddress, VALUE, [VALUE, …])

A successful WRITE writes a sequence of up to eight given byte values into the addressable memory of the module, beginning at the given ByteAddress and returns SUCCESS = TRUE.

Note: A WRITE with more than 8 bytes may be allowed when specified explicitly in chapter 8.

A WRITE access may temporarily be rejected by the module, for generic reasons specified in section 5.2.4 and specific reasons specified in chapter 8.

A rejected WRITE access has no effect in the target and is modeled by returning SUCCESS = FALSE.

A rejected WRITE access can simply be retried until eventual success.

The host shall not include a mixture of volatile and non-volatile registers in the same WRITE access.

5.2.2.3 Test Access (TEST)

Apart from certain advertised maximum durations, the host does not know in advance when exactly a module is guaranteed to be ready and to accept the next READ or WRITE without rejection.

Apart from just trying and re-trying the next desired access, the host can explicitly test for module readiness.

This access readiness test can be modeled by the following pseudo-function

ACCESSIBLE = **TEST**( )

A TEST access simply returns if the module is ready to accept a WRITE or READ access.

Note: TEST is the register layer abstraction of the I2CMCI acknowledge polling primitive in section B.2.5.5

5.2.3 Register Access Rejection (Access Hold-Off)

Any READ or WRITE register access may temporarily be rejected by the module, either for generic reasons specified in section 5.2.4 or for specific reasons individually specified in chapter 8.

A host has a number of options to deal with temporary access rejection. A host may schedule access based on worst case timing specifications (see chapter 10), perform retries of a rejected access, or TEST for readiness prior to a READ or WRITE access (a.k.a. acknowledge polling).

5.2.4 Register Access Sequencing and Synchronization Requirements

Note: In this section, specifications are limited to the elementary READ and WRITE accesses to addressable management memory. Sequencing and synchronization requirements for interactions at the management application layer (e.g. Upper Memory Content Switch, CDB, VDM, etc.) are described in chapters 7 and 8.

After a WRITE access the module ensures that the following **read-back conditions** are satisfied for all registers with regular read-write (RW) access type.

When needed to fulfill these read-back conditions, the target temporarily **rejects** further READ or WRITE accesses, for a maximum **ACCESS hold-off** duration specified in chapter 10.

*Note: The modification of a register by a WRITE access often leads to some intended **effect** in the module or in its behavior. Like in most register based interfaces, completion of those **effects** is **not synchronized** to the* *end of a WRITE access, **nor confirmed** using general mechanisms. In those cases, where confirmation or synchronization is required, handshaking or signaling mechanisms must be implemented using dedicated elements of the register map, such as specific status indicators, Flags, handshaking bits, etc.*

Single byte read-back condition: After a properly terminated WRITE to a Byte address with RW access type, an immediately following successful READ from that same Byte address yields the value just written.

Multiple bytes read-back condition: After a properly terminated WRITE to a Byte address range with RW access type, an immediately following successful READ from any part of that Byte address range yields the value of that part just written.

Note: MCI mechanisms (see section B.2.6) can be used by the module to reject or delay a subsequent ACCESS, either by rejecting or by internally delaying (clock stretch) a subsequent transaction while the read back conditions are not yet satisfied.

5.2.5 Register Data Coherency

Data coherency, of READ or WRITE accesses to a given byte range of interest, refers to module functionality ensuring that concurrent accesses to that byte range, externally by the host and internally by the module, are executed atomically (indivisible and under mutual exclusion), such that concurrent accesses to the byte range as a whole are effectively serialized without interference. Neither the module nor the host will ever 'see' a partially updated byte range that supports data coherency.

Both single-Byte READ accesses and single-Byte WRITE accesses guarantee data coherency.

5.2.5.1 Coherency of Size-Matched Multi-Byte READ Results

Size-matched READ access to one **scalar** multi-byte **read-only register** (as specified chapter 8) is atomic and delivers a coherent result, i.e. the module ensures not to update parts of a scalar read-only register during a size-matched READ of that multi-byte register.

This requirement applies in particular to any scalar 2-byte, 4-byte, or 8-byte status or monitoring register when read by the host with a single 2-byte, 4-byte, or 8-byte READ, respectively.

Note: See section 3.4 for a list of defined scalar multi-byte data types (e.g. F16, U16, S16, U64, X16, etc.).

READ access to other multi-byte registers, multiple registers or register arrays does not guarantee coherency.

If data coherency is not guaranteed but required, higher level access synchronization rules are explicitly specified, such as protocols using dedicated producer-consumer handshaking bits in the register map.

Note: Missing access synchronization rules in such cases would be considered a specification bug. Examples of explicit access synchronization protocols are given by the CDB command message send/receive protocol and the CDB reply message send/receive protocol described in sections 7.2 and 8.15.

5.2.5.2 Coherency of Multi-Byte WRITE Operations

Multi-Byte WRITE accesses (see section B.2.5.4) are generally **not** atomic, unless explicitly specified.

Note: Most writeable registers are byte sized. Examples of multi-byte registers with explicitly specified indivisible access include the PageMapping register and CMDID field in CDB messages.

If data coherency is not guaranteed but required, higher level access synchronization rules are explicitly specified, such as protocols using dedicated trigger or handshake fields in the register map.

Note: Missing access synchronization rules in such cases would be considered a specification bug. Examples of access synchronization protocol are the grouping of configuration data into Control Sets (see section 6.2.3) or the message composition phase before trigger to send a CDB command message (see section 8.15).

6 Core Management Features

The features, functions and behaviors described in this chapter are required for all CMIS managed modules, unless otherwise noted.

Note: Many exceptions and simplifications exist for Flat Memory modules supporting only a constant read-only management memory map of 256 Bytes, i.e. providing neither programmability nor dynamic status information.

6.1 Module Management Basics

CMIS managed modules have two physical interfaces that are related to the main purpose of signal transmission, a **Host Interface** and a **Media Interface**.

Note: The Host Interface is viewed as a device to device interconnection, whereas the Media Interface is viewed as a device to media attachment. For interface circuitry within the module, this difference is not important.

6.1.1 Host Interface

The Host Interface is the high-speed electrical interface (interconnection) between the module and a host system.

The Host Interface carries signals travelling from the host into the module, referred to as **transmitter input** signals, and signals travelling from the module into the host, referred to as **receiver output** signals.

All electrical signals carried over the Host Interface are transmitted differentially over wire pairs, each of which is called a **host lane** (element of interconnection).

Note: Depending on context, a host lane may be viewed as unidirectional or as bidirectional.

The term **host lane** is also used to refer to the module internal resources (circuitry) associated with propagating or processing a host lane signal within the module.

6.1.2 Media Interface

The Media Interface is the high-speed electrical or optical interface between the module and the interconnection media (connecting the module to a remote peer). The interconnection media connecting to a remote end may consist of wires, of optical fibers, or of optical channels, a.k.a. wavelengths (in DWDM or CWDM links).

The Media Interface carries signals that travel from the module into the media, referred to as **transmitter output** signals, and signals that travel from the media into the module, referred to as **receiver input** signals.

All Media Interface signals are ultimately carried between the module and a remote peer. From a near-end viewpoint the Media Interface signals are carried either over electrical differential pairs (on a copper cable) or over optical wavelengths on physical fibers, generically called **media lanes**.

Note: Depending on context, a media lane may be viewed as unidirectional or as bidirectional.

The term **media lane** is also used to refer to the module internal resources (optics and circuitry) associated with propagating or processing a media lane signal within the module.

6.1.3 Memory Map Representations

A set of registers is associated with each lane, both of the host interface and of the media interface, to control processing and report status for signals at that interface.

In this specification all lane numbers or other references to host lanes or media lanes refer to the respective registers that control or describe those lanes. When the term 'lane' is used without reference to 'host' or 'media', a host lane perspective is assumed.

Note: In many cases, module resources are associated with either host interface lanes or media interface lanes as a specification convenience, even though the resource may not be physically proximate to the associated interface.

6.2 Module Functional Model

The following subsections define functional aspects common to all CMIS modules, unless otherwise noted.

Note: The functional and behavioral management requirements related to runtime module reconfiguration have grown to some complexity, at least in certain details. Significant simplifications apply, implicitly, to modules with limited or even no support of runtime programmability

Note: CMIS 5.0 introduces an option for functional simplification with subtle behavioral variations for modules that support only step-by-step reconfiguration, but do not support intervention-free reconfiguration. The current text treats this as a restrictive variant of the more general default specification and is not optimized for readers interested in those simplifications.

6.2.1 Functional Module Capabilities – Applications

6.2.1.1 Applications and Application Instances

An **Application** is a type of functional configuration that is characterized by specific signal propagation or signal processing between one or more host lanes and one or more media lanes, overall providing a well-defined signal or data transmission function to the host.

An **Application instance** is one implementation of an Application by a module

An Application is typically characterized and **specified by reference** to a **pair** of **industry standards**, one for the host interface and one for the media interface, each comprising a number of lanes. These interface standards define all necessary attributes for the respective interface, including the signaling rate (Baud rate), the signaling modulation format, the number of lanes, as well as any digital processing (such as lane alignment or FEC encoding and decoding) of the bit stream (if applicable).

A simple module may support only a fixed single Application, or several parallel Application **instances** (for Applications using fewer lanes than the module provides), while a versatile programmable module may support variant Applications or even multiple instances of one or more Applications operated in parallel.

Note: Modules providing parallel Application instances of one homogeneous Application type are sometimes called breakout modules.

6.2.1.2 Data Paths for Application Instances

The specific host and media lanes of a module that are used to implement one instance of an Application, together with all required internal module resources, is called the **Data Path** of that **Application instance**.

The host and media lanes of the Data Path **allocated** to an Application instance implement the host interface instance and the media interface instance of that Application, respectively.

Note: Here it is important to distinguish the host and media interface of a module and those of an Application. Only for modules specifically implementing only a single Application instance there is no difference.

The Data Path allocated to an Application instance always uses host and media lanes that are numbered consecutively on both module interfaces, starting with the lowest lane number, respectively.

Note: Due to this restriction, both the host interface and the media interface of a Data Path allocated to an Application instance are uniquely identified by the lowest lane number of the allocated host interface lanes and of the allocated media interface lanes, respectively.

Given the possible lane groups, i.e. options for host and media interface instances of an Application type, the Data Paths available for allocation to an Application instance are limited by the restriction that the n'th host interface instance option must be connected straight to the n'th media interface instance option.

Note: This rule assumes transceiver Applications with identical information bit rates on host and media side.

Note: This honors the fact that an Application is defined by a pair of interfaces. The restriction above means that the pairing is always 'fixed and straight without an option to switch the host to media side connectivity.

6.2.1.3 Multiple Application Instances and Multiple Applications

A module may support one or more instances of a single Application. Each configured instance of an Application is independent, from a host viewpoint, of all other concurrently configured Application instances (if any).

For example, a module may support an Application with a CAUI-4 host interface and a 100GBASE-SR4 media interface. In this example, a module could advertise support for one or two instances of the Application.

A module may also support multiple Applications.

For example, a module may support one 400Gbps Application that is characterized by a 400GAUI-8 host interface and a 400GBASE-DR4 media interface combination, and a second 100Gbps Application that is characterized by a 100GAUI-2 host interface and a 100GBASE-DR media interface combination. The module may be programmable to work as one instance of the first Application or to work as up to four instances of the second Application.

Each instance of each active Application is independent of all other Application instances, from a host viewpoint.

For each Application where the module supports multiple Application instances, a host needs to be able to determine which host lane group corresponds to which media lane group for each possible Application instance in the module. As specified in the previous section, modules associate the nth host lane group of the Application with the nth media lane group for the same Application.

This rule is best illustrated by means of an example. A module advertising support for four instances of a 100GAUI-2 to 100GBASE-DR Application must identify the lowest lane number for each instance on both interfaces. As shown in Figure 6-1 for this example, the Application can be assigned starting on host lane 1, 3, 5, or 7. These possibilities are the first, second, third, and fourth lane groups. By rule, the first lane group on the media interface in this example, which starts at media lane 1, must correspond to the first lane group on the host interface, which starts on host lane 1. The second lane group on the media interface, which starts on media lane 2, must correspond to the second lane group on the host interface, which starts on host lane 3, and so on. Therefore, in this example, the module would advertise host lanes 1, 3, 5, and 7 are supported as the lowest numbered host lane for each instance of the Application, and media lanes 1, 2, 3, and 4 are supported as the corresponding lowest numbered media lanes.

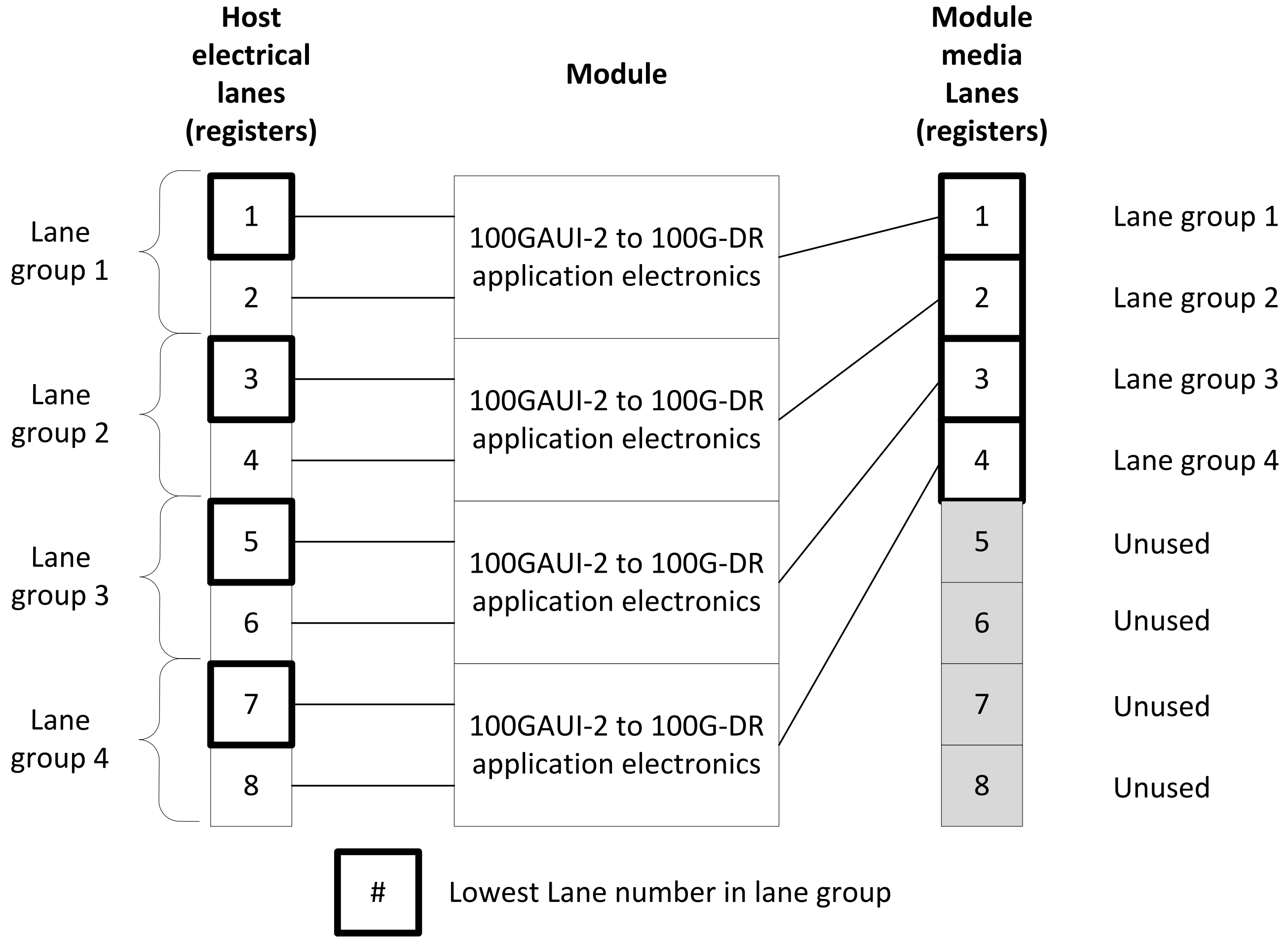

Host electrical lanes (registers)
Module
Module media Lanes (registers)
Lane group 1
Lane group 2
Lane group 3
Lane group 4
1
2
3
4
5
6
7
8
100GAUI-2 to 100G-DR application electronics
100GAUI-2 to 100G-DR application electronics
100GAUI-2 to 100G-DR application electronics
100GAUI-2 to 100G-DR application electronics
1
2
3
4
5
6
7
8
Lane group 1
Lane group 2
Lane group 3
Lane group 4
Unused
Unused
Unused
Unused
#
Lowest Lane number in lane group

Figure 6-1 Lane Assignment Example

6.2.1.4 Application Advertising (Application Descriptors and AppSel Codes)

The module uses **Application Descriptor** data structures to advertise the set of supported instantiations of the set of supported Applications: For each supported Application, a module advertises the lanes that can carry a host interface instance and a media interface instance of a supported instance of that Application, respectively.

The Application Descriptors are stored by the module in three Application Advertising areas of the Memory Map (see Table 8-19, Table 8-51, Table 8-52), but conceptually they form a contiguous array of Application Descriptors, each one structured as shown in Table 6-1.

Each Application Descriptor is identified by a unique **AppSel** code (short for Application Selection code). The AppSel code of an Application Descriptor is simply the **sequence number** of that Application Descriptor in the consecutive array of Application Descriptors advertised by the module. Hence, with a bit of simplification, an **AppSel** code identifies or selects an **Application,** albeit in a **module dependent** way.

Note: The AppSel code identifying a specific type of Application is module dependent because the sequence in which the module advertises its supported Applications in the Application Descriptor array is not specified.

This specification provides space for advertisement of up to fifteen Applications. Applications identified by AppSel codes 1-8 are advertised in registers described in Table 8-19 and Table 8-51, and Applications identified by AppSel codes 9-15 are advertised in registers described in Table 8-52 and Table 8-51.

A module advertises at least one supported Application as its power-up default Application in the first Application Descriptor (identified by AppSel code 1), and it advertises any additionally supported types of Applications consecutively, in a module defined arbitrary order, in a contiguous array of Application Descriptors (starting from AppSel code 2).

A module advertises only supported Applications (which may include custom Applications).

Note: Examples of Application advertising scenarios are shown in Appendix C.

An **Application Descriptor** consists of five bytes and describes all possible instantiations of one Application.

The first byte (**HostInterfaceID**) identifies the industry standard for the host interface. The list of defined HostInterfaceID values can be found in [5] where each Host Interface ID is associated with an industry standard, host interface signaling rate, modulation format, lane count(s), and, implicitly, with signal or data processing characteristics defined in the referenced industry standard. The module defines the number of supported host interface lanes in the HostLaneCount field, described below. The module encodes the first unused entry in the list of Application Descriptors as FFh to indicate the end of the list of supported Applications.

The second byte (**MediaInterfaceID**) identifies the industry standard for the media interface. The list of defined MediaInterfaceID values can be found in [5] in one of the media interface code tables. The module identifies the applicable media interface code table in the **MediaType** advertising field in Byte 00h:85 (see Table 8-17). The media interface code tables implicitly identify the media interface signaling rate, modulation format, and standard-defined lane count(s) for each MediaInterfaceID. The module defines the number of supported media interface lanes in the MediaLaneCount field, described below.

Note: The combination of HostInterfaceID and MediaInterfaceID identifies one type of Application, whereas the following fields describe the possible instantiations of that Application on the lanes of the module.

The third byte (**Lane Counts**) defines the number of lanes for the host interface and for the media interface. These lane counts are derived from the standards identified in the first and second bytes.

The fourth byte (**HostLaneAssignmentOptions**) identifies the lanes where the Application is supported on the module's host interface. The module may support multiple instances of a given Application, so each Lane Assignment Options field identifies the lowest numbered lane in a consecutive group of lanes to which the Application can be assigned.

For example, a module supporting two instances of an Application with a CAUI-4 host interface that can be assigned to host interface lanes 1-4 and 5-8 would advertise a HostLaneAssignmentOptions value of 00010001b, to indicate that the lowest numbered lane for assignment of an instance of the Application can be lane 1 or lane 5.

The fifth byte (**MediaLaneAssignmentOptions**) identifies where the Application instance is supported on the module's media interface. Note that the MediaLaneAssignmentOptions registers are located on Memory Map Page 01h as described in Table 8-51, separated from the first four bytes.

Note: The MediaLaneAssignmentOptions information is not required for flat Memory Map modules.

Table 6-1 Application Descriptor structure

Byte	Bits	Name	Description
First	7-0	HostInterfaceID	ID from **Table "Host Electrical Interface Codes"** in [5]. FFh marks an unused Application Descriptor. *Note: The end of the list of supported Applications is indicated by an unused Application Descriptor with HostInterfaceID FFh.*
Second	7-0	MediaInterfaceID	ID from one of several "**media interface code**" tables in [5], where the relevant table is selected by the MediaType field (see Table 8-19) containing a ModuleTypeEncoding value as described in Table 8-16
Third	7-4	HostLaneCount	Number of host lanes 0000b: lane count defined by interface ID (see [5]) 0001b: 1 lane, 0010b: 2 lanes, ..., 1000b: 8 lanes 1001b-1111b: reserved
	3-0	MediaLaneCount	Number of media lanes. For cable assemblies, this is the number of lanes in the cable. 0000b: lane count defined by interface ID (see [5]) 0001b: 1 lane, 0010b: 2 lanes, ..., 1000b: 8 lanes 1001b-1111b: reserved
Fourth	7-0	HostLaneAssignmentOptions	Bits 0-7 form a bit map corresponding to Host Lanes 1-8. A bit value of 1b indicates that the Application can begin on the corresponding host lane. The host lane numbers of a multi-lane Application are contiguous. If multiple instances of an Application are allowed each supported starting point is identified. If multiple instances are advertised, all possible combination of Application instances are supported concurrently.
Fifth	7-0	MediaLaneAssignmentOptions	Bits 0-7 form a bit map corresponding to Media Lanes 1-8. A bit value 1b indicates that the Application can begin on the corresponding media lane. The media lane numbers of multi-lane Applications are contiguous. If multiple instances of an Application are allowed each supported starting point is identified. If multiple instances are advertised, all possible combination of Application instances are supported concurrently. This field is not required for flat Memory Map modules.

Note: The media interface name identified by the MediaInterfaceID of AppSel code 1 is often used as a common name for the module type.

Note: Since an Application describes a ***combination*** *of a host interface and a media interface, a particular HostInterfaceID or a particular MediaInterfaceID may occur in multiple Application Descriptors.*

Custom Applications

For cases where a module supports a standardized Host Interface with a HostInterfaceID that is included in [5] but uses a media interface that is proprietary or not yet listed in the media interface advertising ID tables in [5], the module can use a null ID (00h=undefined) or a custom ID for the media interface that the module supplier has established, combined with the standardized HostInterfaceID. The HostInterfaceID and the lane counts provide the required information for the host to interoperate with modules that have unfamiliar or vendor-specific media types and future technologies.

Likewise, when a module supports a host interface specification for which no HostInterfaceID has been allocated in [5], a custom HostInterfaceID can be used, which, however, requires the host to be aware of its meaning.

Note: Timely allocation of a standard MediaInterfaceID or HostInterfaceID in [5] is strongly recommended when a new type of interface is targeted for standardized use.

6.2.1.5 Application Selection and Instantiation

A host refers to an Application described in one of the Application Descriptors, located within the contiguous array of advertised valid Application Descriptors, by means of the **AppSel** code of that Application Descriptor.

The actual configuration mechanisms and procedures to instantiate ("provisioning") and implement ("commissioning") one or more Applications is described in the following subsections 6.2.3 and 6.2.4.

Note: The module behavior when a host mistakenly selects an unused Application Descriptor is not specified.

6.2.2 Application Instances on Data Paths – Data Path Lane Assignment

As introduced in section 6.2.1.2, the lanes that are associated with one Application instance, together with all associated module internal resources, are collectively referred to as the **Data Path** carrying that Application instance.

In module configurations with multiple Application instances operating independently, separately, and in parallel, the Data Paths assigned to these instances do not share any lanes.

The host configures the module for its intended use by selecting one or more non-conflicting Applications, from the list of supported Applications in the Application Advertising registers, and by assigning a unique set of lanes as a Data Path to each of the intended non-conflicting Application instances.

The assigned Data Path must be advertised as being supported for this Application, and the lanes used by the Data Path must not create resource conflicts with the lanes of other currently configured Data Paths.

Note: Most Data Path related attributes are duplicated as lane attributes on the lanes of the Data Path. The host configures these attributes on all lanes of the Data Path identically, and likewise the module reports Data Path status attributes on all lanes of a Data Path.

All module internal resources associated with a Data Path are initialized and deinitialized as a group.

Note: In case of resources internally shared between Data Paths, this statement is a simplification.

Separate Data Paths operating in parallel on disjoint (non-overlapping) lane groups are initialized, operated, and deinitialized independently, from a host viewpoint. In particular, the host can issue management operations on separate Data Paths concurrently and asynchronously.

Note: This means that the module must provide functionally independent control for each data path to the host. However, module implementations are permitted to utilize shared physical resources for items such as lasers, TECs, processors, etc., as long as the control mechanisms presented to the host for individual data paths and their associated behaviors remain independent of other data paths.

In one example, a module advertises an Application that consists of a CAUI-4 host interface and a 100GBASE-SR4 media interface combination. In this example, an eight lane module could advertise support for one or two instances of the Application. Each instance of the Application would use a separate Data Path, where each Data Path includes four host lanes and four media lanes and is independent of the other Data Path.

In another example, a module advertises an Application that consists of a 400GAUI-8 host interface and a 400GBASE-DR4 media interface combination, and a second Application that consists of a 100GAUI-2 host interface and a 100GBASE-DR media interface combination. The host may select one instance of the first Application or up to four instances of the second Application. Each instance of each selected Application becomes a separate, independent Data Path.

6.2.3 Data Path Configuration – Control Sets

The fundamental configuration of a module's Data Paths is defined in so called Control Sets.

6.2.3.1 Control Sets Concept

A **Control Set** is a group of registers containing per lane configuration settings that are fundamental for the transmission functions to be provided by Data Paths.

The contents of Control Sets are eventually used by the module during actual Data Path initialization or reinitialization, i.e. when a provisioned Data Path configuration is actually commissioned.

There are two types of control sets. The **Active Control Set** (see section 8.9.6) **reports** the current configuration settings that are used (or to be used) by the module to control its hardware, whereas a **Staged Control Set** (see sections 8.8.3 and 8.8.4) is used by the host to **define** new configuration settings for future use, without immediate effect in the module.

Note: This staging mechanism decouples the timing and sequence of host writes to the Staged Control Set from the module actions to configure the module hardware once the Active Control Set is evaluated.

Each programmable module implements the Active Control Set and at least one Staged Control Set.

The Active Control Set is read-only, and it normally reports settings representing the current hardware configuration of a Data Path, except transiently between a nominal **provisioning** update of the Active Control Set and the completion of a subsequent configuration **commissioning** procedure, in which the Data Path hardware is actually initialized or reinitialized from the Active Control Set.

To **realize** a desired configuration that has been **defined** in a Staged Control Set, the host **requests** the module to replace the current configuration in the Active Control Set with the content of that Staged Control Set (provisioning), and afterwards the module commits the new settings into hardware (commissioning).

Requesting a configuration provisioning procedure (or a combined provisioning and commissioning procedure) is also called **Applying** the Staged Control Set, which is described in more detail, including validation and error checking, in sections 6.2.3.3. and 6.2.4, and in sections 8.8.3.1 and 8.9.5.

6.2.3.2 Control Set Content

Each Control Set contains a **Data Path Configuration** register array **DPConfig** (see Table 8-64, Table 8-68, and Table 8-83) defining two types of configuration settings for each host lane

- **Application Instance** related settings (described below)
- **Signal Integrity** (SI) related settings (described mainly in section 6.2.5)

The host defines an instance of an Application by assigning the relevant Application Descriptor "number" (AppSel code, see below) and its Data Path "location" (i.e. the Data Path lanes allocated to the Application instance, identified by the first host lane) to all lanes carrying that Application instance.

These Application and Data Path related settings must be **identical on all lanes** of a Data Path, while the Signal Integrity related settings may be chosen individually per lane.

For this purpose, each **DPConfigLane**<i> register in the DPConfig array includes an **AppSelCode** field and a **DataPathID** field (together defining the Application and the Data Path which lane <i> is part of), and a signal integrity related **ExplicitControl** bit (allowing host-defined SI settings to be used for lane<i>, instead of SI settings derived by the module for the Application).

AppSel Code Field (Application Descriptor Number)

As described in section 6.2.1.4, each supported Application is advertised by an Application Descriptor and each Application Descriptor is identified by a so called **AppSel** code (an Application Descriptor sequence number).

The host uses the AppSel code of the relevant Application Descriptor to assign the Application identified in that Application Descriptor to one or more host lanes, in the DPConfig register array.

For a multi-lane Data Path, i.e. when an Application requires multiple lanes, the host writes the same AppSel code into the AppSelCode field of each lane of that Data Path.

The special AppSel code value **0000b** in the Data Path Configuration register of a host lane indicates that the lane (together with its associated resources) is **unused** and not part of a Data Path. The DataPathID and ExplicitControl fields of unused host lanes are irrelevant and may be ignored by the module.

Note: This zero valued AppSel code corresponds to a NIL or NULL pointer in programming languages.

In per-lane state reporting, the module always reports a DPDeactivated state for unused lanes (see Table 8-74).

DataPathID Field

The value of the DataPathID field in a DPConfig register localizes and identifies a specific Data Path. The DataPathID of a Data Path is defined as the **lowest lane number** of all host lanes in that Data Path.

Note: See section 6.2.1.2 for rules to determine the media lanes of the Data Path.

Note: The host lane numbers of a Data Path are ultimately referenced to physical lane reference points. However, this does not limit the scope of internal module resources associated with the Data Path: The module resources associated with the Data Path may be located anywhere in the module and may be shared with other host lanes or associated with a media interface in the Memory Map. Any sharing of resources in the module is hidden in the sense that it does not affect independent operation of separate Data Paths in normal operation.

For a multi-lane Data Path, the host writes the same DataPathID in all lanes of that Data Path.

The host must assign lanes to Data Paths in accordance with the Lane Assignment Options field advertised by the module for that Application.

ExplicitControl Field

The **ExplicitControl** bit (EC) determines if the **signal integrity** (**SI**) **settings** for the functional resources associated with a lane are programmed by the host (EC=1) or by the module (EC=0).

The **effective** signal integrity settings resulting from host input (EC=1) or determined by the module (EC=0) are applied by the module during Data Path initialization.

With **host controlled** settings (EC=1) the host programs a signal integrity configuration using the controls listed in Table 6-2 (which are described in section 6.2.5).

With **module controlled** settings (EC=0) the module internally programs a signal integrity configuration that is determined to comply with the (targets of the) interface specifications of the relevant Application.

Note: In the EC=0 case, the signal integrity settings to be used may not be fully and uniquely derivable from the pertinent interface specification of an Application. In that case the module implements settings that accomplish the signal integrity targets of the Application in a best effort manner.

Table 6-2 lists all SI controls together with the values of the ExplicitControl bit required to enable the control. Otherwise the SI control field value is ignored, and the module instead uses SI settings that are compliant with the interface industry standard associated with the selected Application.

Table 6-2 Control fields activated by ExplicitControl bit

Signal Integrity Control Field (see e.g. section 8.8.3.3)	**ExplicitControl** (value to enable)
AdaptiveInputEqEnableTx	1
AdaptiveInputEqRecallTx	X
FixedInputEqTargetTx	1
CDREnableTx	1
CDREnableRx	1
OutputEqPreCursorTargetRx	1
OutputEqPostCursorTargetRx	1
OutputAmplitudeTargetlRx	1

Note: The intent of the AdaptiveInputEqRecallTx feature is to quickly restore previously stored settings of adaptive equalization. This function is always available and may be used for that purpose also when EC=0.

The usage of these signal integrity control fields is defined in section 6.2.5.

6.2.3.3 Control Set Usage (Applying a Staged Control Set)

Initially, the module populates both Staged Control Set 0 and the Active Control Set registers with the module-defined default Application and signal integrity settings before exiting the MgmtInit state (see section 6.3.2.7).

The host can request **provisioning** of one or more Data Paths by asking the module to **validate** and then **copy** the settings of the relevant lanes from the Staged Control Set into the Active Control Set, as illustrated in Figure 6-2.

The resulting lane configuration must consist only of completely valid Data Paths and unused lanes (if any).

Note: Recall that an AppSel value of 0000b in the Data Path Configuration register of a lane denotes an unused lane (not part of a Data Path) for which the module reports the state DPDeactivated (see section 6.3.3).

When the host requests **commissioning** of a Data Path provisioned into the Active Control Set, the module evaluates the settings in the Active Control Set to initialize all resources associated with that Data Path.

Note: Depending on past host provisioning activities, current settings in the Active Control Set may be module-initialized, host-provisioned, or host-commissioned values.

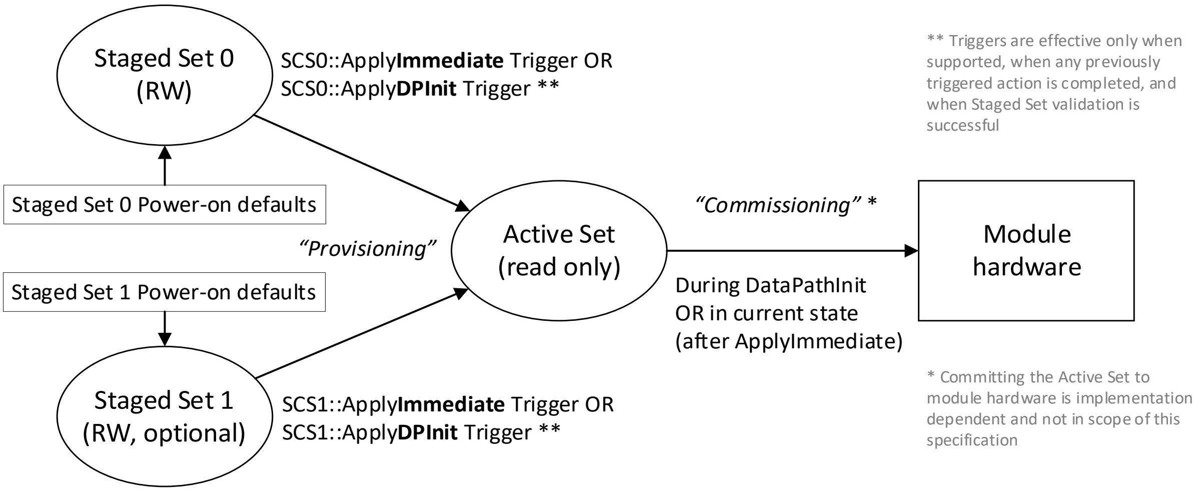

Staged Set 0
(RW)
SCS0::ApplyImmediate Trigger OR
SCS0::ApplyDPInit Trigger **
** Triggers are effective only when supported, when any previously triggered action is completed, and when Staged Set validation is successful
Staged Set 0 Power-on defaults
"Provisioning"
Active Set
(read only)
"Commissioning" *
Module
hardware
During DataPathInit
OR in current state
(after ApplyImmediate)
Staged Set 1 Power-on defaults
Staged Set 1
(RW, optional)
SCS1::ApplyImmediate Trigger OR
SCS1::ApplyDPInit Trigger **
* Committing the Active Set to module hardware is implementation dependent and not in scope of this specification

Figure 6-2 Control Set Data Flow Diagram

Apply Trigger Registers

For each Staged Control Set there are two **Apply** controls (trigger registers) available for the host to request copying **Data Path** settings from a Staged Control Set to the Active Control Set (using a bit mask to indicate the lanes of selected Data Paths): SCS<k>::**ApplyDPInit** and SCS<k>::**ApplyImmediate**.

Note: Typical use cases for these Apply triggers are reconfiguration of the Application to be carried over a Data Path, modification of Data Path lane widths, modification of signal integrity settings on lanes of a Data Path, or changing interface speed by selecting another Application associated with that speed.

Note: ApplyImmediate may be unsupported, depending on module advertisement (see section 6.2.4).

The operations requested by Apply triggers are **Data Path operations**: A host must trigger **Apply** on **all** lanes of a Data Path simultaneously (even if only lane specific attributes of individual lanes are changed), with **one exception**: ApplyImmediate, if supported, can be used on a per lane basis, but only when only lane specific signal integrity settings are changed and only on lanes of Data Paths in state DPActivated or DPInitialized.

A host can trigger **Apply** on multiple Data Paths simultaneously in one WRITE access.

The effects of the host writing a lane selection bit mask to one of those trigger registers depend both on the trigger register instance used and on the states of the **Data Path State Machine** (**DPSM**) instances associated with the lanes selected in the bit mask.

Note: See section 6.3.3 for information on DPSM instances and states, and see section 6.2.4 and Table 6-3 for a command handling oriented description of the module procedures associated with Apply triggers.

Stepwise Provisioning and Commissioning

The host requests **provisioning** of prepared staged settings by invoking any Apply trigger on lanes indicating the **DPDeactivated** state, which causes the module to (1) **validate** and (2) **copy** the staged settings into the Active Control Set, without effect on module hardware.

The subsequent **commissioning** of the Active Control Set occurs separately, after a host-initiated transition to the **DPInit** state, where the module will eventually (3) **initialize** the associated Data Path resources based on the configuration then found in the Active Control Set, regardless of which sequence or type of Apply trigger commands had been used to define the current content of the Active Control Set.

Note: The following intervention-free reconfiguration procedures are supported by default, but a module may advertise that only stepwise configuration is supported.

Combined Provisioning and Commissioning (intervention-free)

When invoked on lanes indicating an **initialized** Data Path that is currently in use, i.e. on lanes indicating the **DPInitialized** or **DPActivated** state, both Apply triggers cause the module to perform the provisioning and commissioning steps sequentially, albeit in two different ways, with or without DPSM state changes, described as **Regular Reconfiguration** or **Hot Reconfiguration** as follows:

Regular Reconfiguration (Provisioning Procedure with Commissioning in DPSM State Cycle)

After an **ApplyDPInit** trigger in an **initialized** state the module will first (1) **validate** and (2) **copy** the staged settings into the Active Control Set , then (3) **deinitialize** the Data Path by **auto-transitioning** the DPSM to the DPDeactivated state, and finally (4) **initialize** the Data Path to **realize** the new Active Control Set content.

This **reinitialization** (3+4) occurs **automatically** as a sequence of **DPSM state transitions** (by internal module procedures executed in the states), eventually moving back to the state where the ApplyDPInit was triggered (unless DPSM state transition relevant configurations have meanwhile changed).

Note: An intervention-free procedure executed internally on host command does not require READ or WRITE access for each step and hence can execute faster, as is required in Applications like InfiniBand or Fibre Channel.

Note: This intervention-free reconfiguration function is provided for cases where new settings need to be applied faster than possible with the stepwise management interactions that are necessary when the host fully controls the reconfiguration or reinitialization loop.

Hot Reconfiguration (Provisioning and Commissioning Procedure without DPSM State Change)

After an **ApplyImmediate** trigger in an **initialized** state the module performs the provisioning and commissioning actions, i.e. (1) **validate**, (2) **copy**, and (3) **re-initialize**, in one optimized internal procedure, **without changing** the DPSM **state**.

Note: This "hot reconfiguration" function is provided for cases where new settings need to be applied quickly without disabling the media lane transmitters, such as in Fibre Channel Link Speed Negotiation (LSN).

Note: Using ApplyImmediate in the DPInitialized or DPActivated state allows still faster completion of a desired configuration change in cases where this is essential. The intent of this specification is for the module to perform the actions associated with ApplyImmediate as quickly and efficiently as possible in a minimally disruptive manner. However, in some cases such as changing speeds, disruption is unavoidable.

Silent Rejection

When invoked on lanes indicating a transient state (**DPInit**, **DPDeinit**, **DPTxTurnOn**, **DPTxTurnOff**) the module may ignore Apply triggers; the result is then undetermined in general.

Note: The detailed behavior depends on the support of intervention-free reconfiguration, as described below.

Note: Modules should preferably provide a rejection feedback (ConfigRejected), when possible, see below.

Error Handling and Result Feedback

In all cases, if a requested Data Path configuration is not accepted during validation, the module skips the remaining steps of the procedure without copying any settings for that Data Path into the Active Control Set.

When terminating the procedure the module always reports a result status code (see Table 8-81) informing about successful execution (ConfigSuccess) or about rejection (ConfigRejected*) for each affected lane, using the Configuration Command Execution and Result Status fields (**ConfigStatus**, see Table 8-80).

Realization (Commissioning)

The translation of the Memory Map settings in the Active Control Set to module hardware is implementation-specific and outside the scope of this specification. Modules should implement a best effort approach when trying to translate Memory Map settings to module hardware settings.

As a visual summary, Figure 6-2 illustrates the data flow of configuration settings from a Staged Control Set into the Active Control Set and further into module hardware.

6.2.4 Data Path Configuration – Procedures and Commands

This section describes the process of changing the Data Path configuration of a module at the level of procedures and commands.

6.2.4.1 Configuration Concept

Bringing a Data Path into service, such that it provides the Application-defined transmission function, requires three steps, herein called definition, provisioning, and commissioning.

Note: It is customary to call the combination of all three steps a configuration activity, but sometimes configuration refers only to the first two steps. For accuracy one would have to distinguish these cases as actual configuration and as nominal configuration, respectively, but we hope that the meaning is clear from context.

Note: Recall that configuration changes are requested at Data Path granularity even if only settings affecting individual lanes are changed (e.g. lane-specific SI settings).

Configuration Steps

The first step, **definition**, is a host-executed procedure defining a desired Data Path configuration in a selected Staged Control Set, using an arbitrary sequence of register WRITE accesses, without module reaction. Definition is a pure **host procedure**.

The second step, **provisioning**, is a host-triggered module procedure that **validates** the settings of a Data Path defined in a given Staged Control Set and, if successful, **copies** them into the Active Control Set, for all lanes of a Data Path. This step is a **module procedure**, triggered by the host.

Commissioning

The third step, **commissioning**, is either a host-initiated procedure or an automatic follow-up of a previously host-triggered provisioning procedure: in both cases the module commits the nominal configuration of selected lanes in the Active Control Set to hardware such that the Active Control Set eventually matches the actual hardware configuration again. This step is a **module procedure** that is either directly **initiated** or indirectly **triggered** by the host.

*Note: **Triggering** and **initiating** are deliberately distinguished here: triggering has an event nature and is implemented as a host writing to a stateless trigger register while initiating is a causal consequence of changing a setting in a configuration register. For triggered procedures, the module provides feedback, while initiated procedures provide feedback only indirectly, by their observable effects.*

The host selects all host lanes of a Data Path by writing to lane-associated registers or fields (in definition or for initiating a module procedure), or by using a lane selection bit mask (when triggering a module procedure).

Host Procedures

This specification describes two types of configuration procedures.

In a **stepwise** procedure, the host performs all configuration actions step-by-step, according to the **Data Path State Machine** (**DPSM**) initialization and configuration model described in section 6.3.3.

Characteristic features of this procedure are that the host provisions only lanes that are currently either **unused** or part of a deactivated Data Path (lanes in **DPDeactivated** state), and that the commissioning procedure is first explicitly initiated by the host and then performed by the module while initializing a Data Path (in **DPInit** state).

Note: For Applications where the stepwise procedure is fast enough, for example Ethernet or Transport, it is recommended that the host uses only a stepwise procedure and avoids the intervention-free procedures described in the following.

In **intervention-free** (or **automatic**) procedures, the host triggers only the provisioning step on initialized lanes that are currently in use (in **DPInitialized** or **DPActivated** state) and the module subsequently performs the commissioning step automatically, without further host intervention.

However, a module may advertise that it does not support intervention-free reconfiguration (see Table 8-4).

If supported, there are two intervention-free reconfiguration procedures:

In the **regular reconfiguration** procedure (triggered via an **ApplyDPInit** register) the module automatically cycles through the states of a Data Path (via DPDeactivated and DPInit) to perform the commissioning step when the triggered provisioning step was successfully executed.

Note: This method is targeted for Applications where the stepwise procedure is not fast enough, for example Fibre Channel and InfiniBand.

In the **hot reconfiguration** procedure (triggered via an **ApplyImmediate** register), the module performs all configuration and commissioning actions automatically while staying in the current Data Path state.

Note: This method is targeted for Applications where the regular reconfiguration procedure is not fast enough, for example Fibre Channel.

Note: This option has been introduced in CMIS 5.0 to simplify implementation and test for modules supporting only Applications that can always be reconfigured step-by-step.

Module Procedures

Unlike configuration definition, the provisioning and commissioning steps are **module procedures** executed by the module after a host action (see the following section 6.2.3.3). See also Table 6-3 below.

Module procedures may take time to complete and may fail. Hence the module provides **feedback** to the host, both about an execution-in-progress status and, eventually, about success or failure of the triggered procedure.

The host **triggers** the begin of a module procedure by writing a lane selection bit mask to an **Apply** register in the relevant Staged Control Set (see Table 8-62 and Table 8-67) and the module provides feedback about the execution and about the result status of the procedure via per-lane **ConfigStatus** registers (see Table 8-80 and Table 8-81).

Module Procedure Execution Protocol (Command Handling)

The execution of a host triggered Data Path module procedure consists of four steps, which may be understood, conceptually, as a **command handling** sequence.

First, in **command acceptance**, the module determines if the requested procedure can currently be accepted at all: here, a new trigger cannot be accepted when a previously triggered configuration procedure is still executing. When accepted, the module immediately sets the execution status for this lane (**ConfigInProgress**); else the command is not executed and silently **ignored**, **without feedback** to the host.

*Note: It is therefore host responsibility to check the **ConfigStatus** of the lanes of a Data Path in advance, or to otherwise ensure that no procedure is still executing, prior to triggering a configuration procedure.*

Second, in the **command validation** step, the module validates the parameters of the command (content of Staged Control Set), and skips the next step on those Data Paths for which parameter validation has failed.

Third, in the actual **command execution** step, the desired procedure is executed.

Last, the module reports the **command result** status in the **ConfigStatus** registers of the lanes of the Data Path. The result status reports either successful execution (**ConfigSuccess**) or rejection without execution due to validation failure prior to command execution (**ConfigRejected***).

Note: Unexpected failures during command execution are not reported.

The host can always query the **ConfigStatus** of a lane to see if a configuration command is currently still being processed or otherwise what the result of the previously executed command was (see Table 8-81).

6.2.4.2 Configuration Commands

The overall response of the module to a host WRITE of a lane selection bit mask to a trigger register is best understood as per Data Path **command handling**, which consists of (1) command handling readiness check, (2) input parameter validation, (3) actual command execution, and (4) positive or negative result status feedback. This is now described in more detail.

Note: Recall that configuration commands are applied to entire Data Paths (with one exception), while the trigger and result reporting mechanism is defined per lane (all lanes reporting the same value). Also note that the desired reaction to a trigger possibly depends on several factors: DPSM state, Apply trigger register type and instance used, nature of changes, and on restriction advertisement.

Configuration Command Handling

Writing a lane selection bit mask to an Apply register triggers execution of one or more parallel **configuration commands**, depending both on the trigger register used and on the current DPSM state(s) associated with each affected lane, as indicated in Table 6-3 and Table 6-4.

Writing a set lane selection bit to an Apply register is only allowed when the selected lane indicates a stable and steady DPSM state and when no previously triggered configuration command is still being executed on the selected lane.

Note: This precondition can be checked by the host by reading the Configuration Command Execution and Result Status registers described below.

Note: Writing to an apply register in transient states or in transitory steady states may have unpredictable effects.

When a previously triggered Provision or Provision-and-Commission command is still being processed for lanes of a Data Path, the module **ignores** new triggers for those lanes, and does not execute the command on the

Data Paths those lanes belong to.

*Note: Ignoring lane selection or trigger bits due to configuration being in progress is **not** indicated to the host.*

Table 6-3 Configuration Commands (default)

Control (Trigger Reg.)	**Data Path State ***	**Host Intent**	**Triggered Module Procedure **** (when trigger is accepted)
ApplyDPInit	DPActivated or DPInitialized	**Intervention-free Reconfiguration** ("copy and cycle") (provisioning + commissioning)	**Provision** 0 ConfigStatus = ConfigInProgress 1 Validate content of Staged Control Set 2 **Copy content to Active Control Set** 3 **Set DPInitPending** 4 Report result status in ConfigStatus *Note: commissioning is done by DPSM* *Note: DPInitPending is cleared in DPInit*
	DPDeactivated	**Stepwise** (host-controlled) **Configuration** ("copy only") (provisioning only)	
ApplyImmediate	DPInitialized or DPActivated	**Intervention-free Hot Reconfiguration** ("copy and commit") (provisioning + commissioning)	**Provision-and-Commission** 0 ConfigStatus = ConfigInProgress 1 Validate content of Staged Control Set 2 **Copy content to Active Control Set** 3 **Commit Active Control Set to HW** 4 Report result status in ConfigStatus

** Triggers are ignored in Data Path states not explicitly mentioned*
*** Steps 2 and 3 are conditional on successful validation in step 1*

Table 6-4 Configuration Commands (SteppedConfigOnly=1)

Control (Trigger Reg.)	**DataPath State ***	**Host Intent**	**Triggered Module Procedure ****
ApplyDPInit	any DPSM state	**Stepwise Configuration**	**Provision** 0 ConfigStatus = ConfigInProgress 1 Validate content of Staged Control Set 2 Copy content to Active Control Set 3 Set DPInitPending 4 Report result status in ConfigStatus *Note: DPInitPending is cleared in DPInit*

** Triggering recommended only in DPDeactivated (avoid difference of Hardware and Active Control Set)*
*** Steps 2 and 3 are conditional on successful validation in step 1*

Configuration Command Execution Status and Result Status

The module reports both the transitory execution-in-progress status and the final result status of the configuration command using the Configuration Command Execution and Result Status (**ConfigStatus**) fields described in Table 8-80. The status reporting for Data Path configurations is per lane (as always in CMIS).

When accepting an Apply trigger, the module **immediately** updates the ConfigStatus field of all affected lanes to indicate **ConfigInProgress**.

When a Module Procedure triggered on a Data Path has been completed, the module updates the ConfigStatus field of all triggered lanes of that Data Path with the appropriate result: **ConfigSuccess** or **ConfigRejected***. Only one Data Path result status is reported on all triggered lanes of a Data Path.

6.2.4.3 Host Rules and Recommendations

This subsection provides a host oriented compilation of important rules and recommendations that follow from module oriented specifications elsewhere, without a claim of completeness:

- The host must assign a valid **AppSel** code to all host lanes in the Active Control Set
- All lanes of the application identified by an AppSel code must be assigned the same AppSel code
- The host must assign **AppSel** = 0000b to each unused host lane (which is not part of a Data Path)
- When a lane is used in a Data Path, the host can reconfigure it to become unused only when the Data Path is in the DPDeactivated state, i.e. the host must not invoke an Apply trigger on lanes marked as unused in the Staged Control unless the Data Path currently associated with those lanes is in the **DPDeactivated** state

- Each used lane in the Active Control Set must be part of a completely valid Application instance (i.e. a valid Application completely allocated on lanes supported for that Application).
- The host can change the width of a Data Path only while in the **DPDeactivated** state, i.e. the host must always transition an existing Data Path to **DPDeactivated** before selecting an Application with a different lane count. Any lane that becomes unused must be marked as such (AppSel = 0000b) or it must be assigned to a new valid Data Path (remaining in DPDeactivated state until eventually used)
- It is recommended that hosts **use only** the fully host-controlled **stepwise reconfiguration** procedure and use ApplyDPInit only for Data Paths in DPDeactivated state – this is the cleanest way to perform a Data Path configuration change, more likely to behave exactly like initial power up configuration
- It is recommended that hosts use the **intervention-free** reconfiguration procedures using ApplyDPInit or ApplyImmediate on Data Paths in steady states DPInitialized or DPActivated **only when required** to meet protocol performance requirements of the Application
- When **intervention-free** reconfiguration procedures are supported, hosts are advised not to invoke an Apply trigger on the lanes of a Data Path in a transient state (**DPInit**, **DPDeinit**, **DPTxTurnOn**, or **DPTxTurnOff**), as the module silently ignores requests received while still being in a transient state.
- Host are advised to use ApplyImmediate only when intervention-free reconfiguration is supported, and only in initialized states (DPInitialized and DPActivated), because the module silently ignores ApplyImmediate in all other cases.
- Before involving any Apply trigger, hosts are advised to ensure that the **ConfigStatus** of all affected lanes is different from **ConfigInProgress**, as the module may otherwise silently ignore the trigger

6.2.4.4 Initialization Sequence Examples

The Data Path architecture described above is intentionally designed to support a broad array of implementations while ensuring compatibility across hosts and modules. Some Applications may not use all of the features provided in the architecture.

Appendix D contains some example host-module initialization flows that can be used for popular Applications.

6.2.5 Signal Integrity Related Controls

Signal integrity control fields in the Memory Map provide a per lane mechanism for the host to program the signal integrity settings that are otherwise controlled by the module to comply to the interface standard of an Application.

Note: Recall that the module decides, in a best effort manner, which signal integrity configuration accomplishes the targets specified in the pertinent interface standard of an Application

If the **ExplicitControl** bit is set in the applicable Staged Control Set (see Table 8-64 and Table 8-68), the module copies the host provisioned signal integrity control values to the Active Control Set (subsequently committed to module hardware, see Figure 6-2) when an **ApplyDPInit** or an **ApplyImmediate** trigger is used.

If the **ExplicitControl** bit is cleared, the module writes suitable Application-dependent default signal integrity control values into the Active Control Set when an **Apply** trigger is used.

Note: Adaptive or Non-Adaptive equalization control in only available when the ExplicitControl bit is set. While the AdaptiveInputEqRecallTx recall works also when the ExplicitControl bit is not set.

For all signal integrity controls, the host configures the desired behavior in terms of reference receiver or transmitter metrics and the module makes a best effort to achieve the desired effect in its implementation.

6.2.5.1 Tx Input Equalization Control

The controls for Tx input equalization can be grouped by equalization type, as shown in Table 6-5.

Table 6-5 Tx Input Eq control relationship to AdaptiveInputEqEnableTx

Equalization Type	Control	AdaptiveInputEqEnableTx
Adaptive	AdaptiveInputEqFreezeTx	1
	AdaptiveInputEqStoreTx	
	AdaptiveInputEqRecallTx	
Non-Adaptive	FixedInputEqTargetTx	0

The controls relevant for **adaptive** Tx input equalization are described in section 6.2.5.4.

The controls relevant for **non-adaptive** Tx input equalization, when Tx input equalization settings are pre-determined or host provisioned, are described below.

The module ignores control field values that are not relevant for the current AdaptiveInputEqEnableTx setting.

Host Controlled Equalization

Tx input equalization values in dB are based on a reference CTLE and may not directly apply to the equalizer implemented in the module.

SCS<k>::**FixedInputEqTargetTx**<i> is a four-bit control field for lane <i> and encoded as shown in Table 6-6. This field allows the host to specify a fixed (non-adaptive) Tx input equalization target and is ignored by the module if AdaptiveInputEqEnableTx<i> is set for that lane.

The module advertises support of non-adaptive Tx input equalization control as described in Table 8-47.

The module advertises the maximum supported Tx input equalization values as described in Table 8-43.

Table 6-6 Fixed Tx Input Equalization Codes

Code Value	Bit pattern	Input Equalization
0	0000b	No Equalization
1	0001b	1 dB
2	0010b	2 dB
3 - 8	0011b … 1000b	3 dB … 8 dB
9	1001b	9 dB
10	1010b	10 dB
11	1011b	11 dB
12	1100b	12 dB
13-15		Custom

6.2.5.2 Rx Output Equalization Control

Rx output equalization is defined at an appropriate test point defined by the relevant standard.

SCS<k>::**OutputEqPreCursorTargetRx**<i> and SCS<k>::**OutputEqPostCursorTargetRx**<i> are four-bit control fields for lane <i> and encoded as shown in Table 6-7.

The module advertises support of Rx output equalization control as described in Table 8-47.

The module advertises the maximum supported Rx output equalization values as described in Table 8-43.

Modules that require only output emphasis utilize the SCS<k>::**OutputEqPostCursorTargetRx**<i> fields and set the SCS<k>::**OutputEqPreCursorTargetRx**<i> fields to zero.

Table 6-7 Rx Output Equalization Codes

Code Value	Bit pattern	Post-Cursor Equalization	Pre-Cursor Equalization
0	0000b	0dB (No Equalization)	0dB (No Equalization)
1	0001b	1 dB	0.5 dB
2	0010b	2 dB	1.0 dB
3	0011b	3 dB	1.5 dB
4	0100b	4 dB	2.0 dB
5	0101b	5 dB	2.5 dB
6	0110b	6 dB	3.0 dB
7	0111b	7 dB	3.5 dB
8-10	1000b-1010b	**Reserved**	**Reserved**
11-15	1011b-1111b	Custom	Custom

Note: The pre-cursor equalizer settings in dB approximates to

$$\text{Pre EQ (dB)} = -20\cdot\log10(\,1 - C_{-1} / (C_{-1}+C_0+C_1)\,) \qquad \text{(Eq. 6-1)}$$

The post-cursor equalizer settings in dB approximates to

$$\text{Post EQ (dB)} = -20\cdot\log10(\,1 - C_1 / (C_{-1}+C_0+C_1)\,) \qquad \text{(Eq. 6-2)}$$

Equalizer coefficients C_n are pre-cursor for $n<0$ and post-cursor when $n>0$.

6.2.5.3 Rx Output Amplitude Control

The Rx output amplitude is measured without Rx output equalization and defined at an appropriate test point defined by the relevant standard.

SCS<k>::**OutputAmplitudeTargetRx**<i> is a four-bit field for lane <i> to specify the Rx output signal level being in a particular amplitude range, with amplitude range encoding as described in Table 6-8.

The module advertises support of Rx output amplitude control as described in Table 8-47.

The module advertises the maximum supported Rx output amplitudes as described in Table 8-43.

Table 6-8 Rx Output Amplitude Codes

Code Value	Bit pattern	Output Amplitude
0	0000b	100-400 mV (P-P)
1	0001b	300-600 mV (P-P)
2	0010b	400-800 mV (P-P)
3	0011b	600-1200 mV (P-P)
4-14	0100b-1110b	**Reserved**
15	1111b	Custom

6.2.5.4 Adaptive Tx Input Equalizer Store and Recall Mechanism

Note: Equalizer adaptation can be time-consuming. In some applications, the available time for a speed change (which incurs selecting a new Application and hence a Data Path reconfiguration) does not include equalizer adaptation time. To better support such applications, an optional host controlled Store and Recall mechanism is specified for storing adapted equalizer settings for later recall and use.

Note: Only modules supporting applications that require fast application change (e.g. for speed negotiation) with critical time budgets are expected to support this mechanism.

The **TxInputEqRecallBuffersSupported** field (see Table 8-47) advertises support of the adaptive Tx input equalizer store and recall mechanism by advertising the number of recall buffers supported by the module.

Recall buffers are numbered and used independently of Staged Control Set instances.

The module provides enough storage in each recall buffer to store adapted equalizer settings for each lane of the module. The storage mechanism is implementation specific and not defined in this specification.

The **AdaptiveInputEqStoreTx** trigger field is described in Table 8-61. To store the most recent adapted Tx input equalizer settings for lane <i>, the host writes the desired target recall buffer number into AdaptiveInputEqStoreTx<i>. Equalizer adaptation is then stopped until the settings have been stored and continues afterwards, unless adaptation is frozen (i.e. unless AdaptiveInputEqFreezeTx<i> is set).

Host requests to store equalizer settings while adaptation is disabled (i.e. AdaptiveInputEqEnableTx<i> is cleared for lane <i>) are ignored by the module.

The host may trigger storage by writing to AdaptiveInputEqStoreTx<i> at any time while the Data Path state is DPInitialized or DPActivated, and a requested storage occurs then immediately .

The **AdaptiveInputEqRecallTx** control field is described in Table 8-65 for Staged Control Set 0 and in Table 8-69 for Staged Control Set 1.

The Active Control Set provides a read-only indication of the current **AdaptiveInputEqRecalledTx** status for each lane (see Table 8-84).

To recall a stored Tx input equalizer adaptation setting into a Staged Control Set, the host writes the storage buffer number to be recalled to the applicable AdaptiveInputEqRecallTx lane controls.

Note: These settings are not recalled into the active equalizer until the host has triggered ApplyDPInit or ApplyImmediate for that Staged Control Set.

The AdaptiveInputEqRecallTx field is used independent of the ExplicitControl field settings for that lane.

If **AdaptiveInputEqFreezeTx** is cleared, the recalled Tx input Eq adaptation setting is used as the starting point for continuous adaptation for the applicable lanes. Otherwise, if **AdaptiveInputEqFreezeTx** is set, the recalled Tx Input Eq Adaptation is used as the frozen Tx Input Eq value for the applicable lanes.

6.3 Module Behavioral Model

6.3.1 State Machine Concept

Fundamental power up, initialization, and reinitialization interactions between host and module are governed and described by state machine based behavioral models. Conceptually, these state machines are considered parts of the module.

Note: These state machines are purely conceptual, precise models to specify required host and module interactions and behaviors associated with those interactions; they do not constrain software implementation.

State machines describe both autonomous behavior (i.e. what the module does) and reactive behavior (i.e. how the module reacts to events caused by the host). In certain situations, the states of the state machines in the module cannot be observed by the host, but they still govern behavior and reactions of the module in these situations.

Module and Data Path State Machines

Two types of state machine are distinguished: Module State Machine and Data Path State Machine.

A **Module State Machine (MSM)** defines host-module interactions and behavioral characteristics of the module as a whole, such as the initialization of the management interface and the Module Power Mode.

A **Data Path State Machine (DPSM)** defines host-module interactions and behavioral characteristics needed for the initialization of one particular **Data Path**, which represents signal flow and signal processing of one instance of one type of **Application**.

Note: For information on Applications and Data Paths see sections 6.2.1and 6.2.2, respectively. For more information on Module State Machine and Data Path State Machines see sections 6.3.2. and 6.3.3.

A module has a single Module State Machine, but the number of Data Path State Machines depends on module configuration: A simple module with a single Data Path for a single Application, has one DPSM. A complex module supporting parallel Data Paths for multiple instances of one Application or instances of multiple Applications has several DPSM instances, one for each provisioned Application instance.

Note: As defined elsewhere in this specification, parallel Data Path State Machines are controlled independently, and they operate concurrently and mutually independent. Parallel Data Paths behave largely like independent virtual modules within a physical module.

Transient and Steady States

Two kinds of states are distinguished (in both types of state machines), steady states and transient states.

In **steady states** the module is essentially waiting for the host to initiate action.

Note: some exceptional module internal events may also cause exit from steady states.

In **transient states** the module performs dedicated activities and automatically performs a state transition upon completion (unless events cause other state transitions before completion).

In the state machine illustrations, such as Figure 6-3, Figure 6-4, and Figure 6-5, steady states are displayed with a rectangular outline and transient states are displayed with an oval outline.

The durations of steady states are unbounded, from a module's viewpoint, as the exit conditions depend on events not controlled by the module.

The durations of transient states are bounded because they are associated with module-controlled activities. The duration bounds may be implementation dependent. For most transient states a maximum duration is either advertised or bounded by a specified limit (see chapter 10).

Form Factor Independent Specification

CMIS is designed to be applied to multiple form factors. Therefore, actions and properties are described using generic signal names instead of using form factor specific signal names. Appendix A describes the association of generic and form factor specific signal names, and the implementation of abstract signal levels.

6.3.2 Module State Machine (MSM)

The **Module State Machine (MSM)** defines host-module interactions and behavioral characteristics of the module as a whole. The MSM state represents a module's readiness to be managed and used by its host.

The Module State Machine describes module-wide behaviors and characteristics.

Note: For Data Path-specific behaviors and characteristics, refer to the Data Path State Machine in section 6.3.3.

Host View
By observing the MSM state, the host can determine when the management interface has completed initialization after power on or on exit from reset. The host can also determine from the Module State Machine state when the high-speed circuitry in **paged memory modules** may be fully powered such that the host can begin to initialize Data Paths through the Data Path State Machine (described in section 6.3.3).

Module View
The MSM is engaged immediately after module insertion and power on.

The MSM is applicable both to modules and to cable assemblies, whether passive or active.

- The MSM for paged memory modules is described in section 6.3.2.2.
- The simplified MSM for flat memory modules is described in section 6.3.2.3.

6.3.2.1 General Module Behavior

This section describes general module behavior and requirements associated with the Module State Machine.

Management Data Relevance, Validity and Accuracy

Prior to proper initialization, i.e. before the state of basic manageability (ModuleLowPwr) has been reached, the contents of the management Memory Map are neither evaluated nor maintained by the module.

During transient states, the contents of reporting registers that are usually dynamically changed by the module may generally be invalid or inaccurate until a fully managed operational state is reached again.

Note: Dynamic reporting registers are read-only updated by the module to report status or performance data.

Flag Setting and Clearing

The set of Flags (possibly causing Interrupts) that a module may set is defined specifically for each state. The state dependent Flag setting rules are defined in section 6.3.4.

Unless specified otherwise, the module shall not clear any Flags on a state transition, and Interrupts are only cleared when the host has read or masked all set Flags.

State Change Indication

The **ModuleStateChangedFlag** is set after entering a new state, but only for the subset of module-initiated state transitions that are defined in Table 6-9.

In addition, setting the **ModuleStateChangedFlag** is generally suppressed when the new state is exited immediately because its exit conditions are already fulfilled on entry.

Note: The rationale for this suppression rule is that intermediate state change notifications are not interesting and should not cause Interrupt; only the last transition into a new stable state should be notified.

Table 6-9 ModuleStateChangedFlag behaviors

New State	**ModuleStateChangedFlag** flagged on state entry? *
Resetting	No
Reset	No
MgmtInit	No
ModuleFault	**Yes**
ModuleLowPwr	**Yes**
ModulePwrUp	No
ModuleReady	**Yes**
ModulePwrDn	No

*unless the exit condition was already satisfied on entry into the entered state

6.3.2.2 Module State Machine for Paged Memory Modules

The behavior of Paged Memory Modules is described by the Module State Machine shown in Figure 6-3

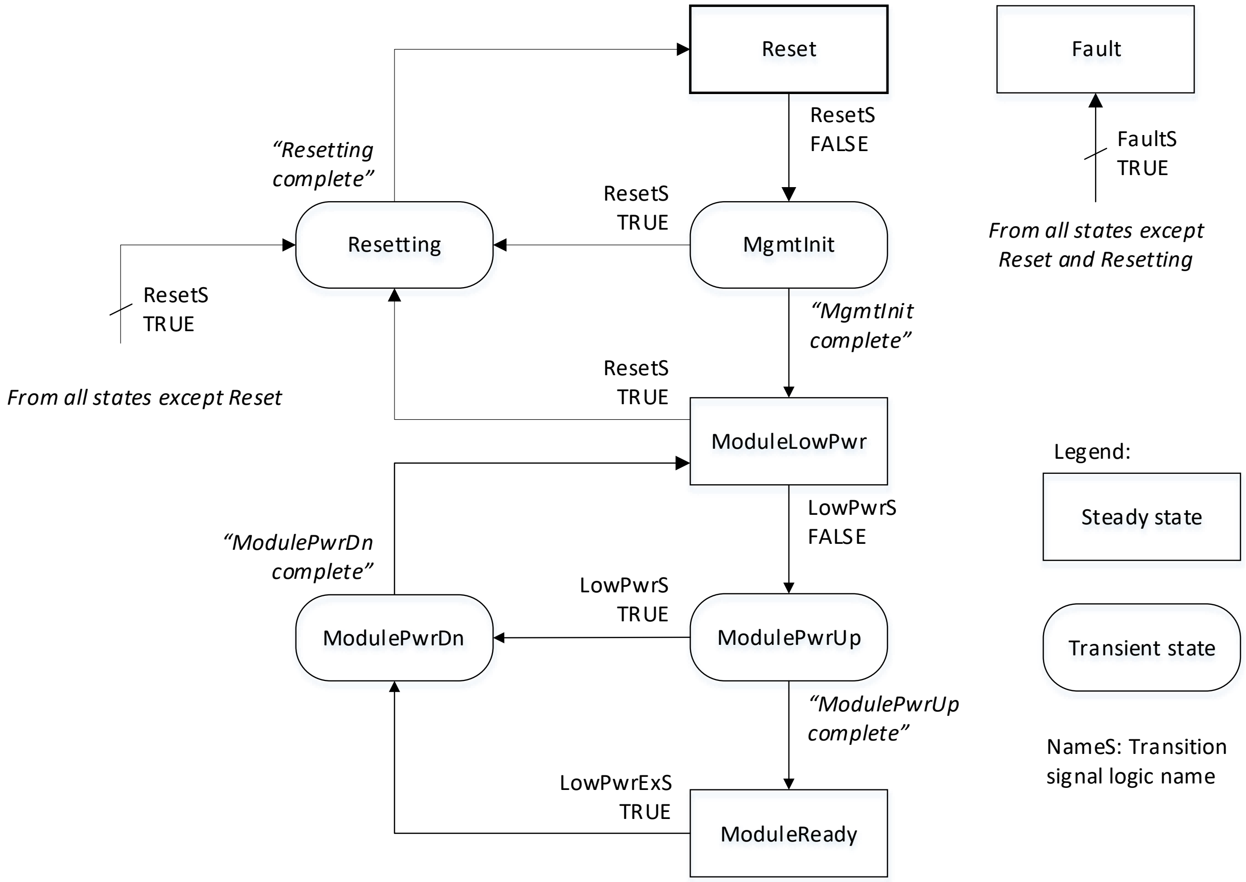


Figure 6-3 Paged Memory Module State Machine (MSM) State Transition Diagram

On module power-up, the initial Module State Machine state is the Reset state, and the State Machine remains in this state while ResetS is TRUE. Otherwise, the Module State Machine transitions to the MgmtInit state.

The state machine exits a given state when specific exit conditions are satisfied. So called **transition signals** (names ending in **S**) are used to describe the logic conditions governing a state transition. Other compound logical **t**erms may be symbolically named with names ending in **T**.

The following table describes the priority of exit conditions, if more than one exit condition is satisfied at the same time. Note that not all exit conditions are applicable to all states.

Table 6-10 Module State Machine exit condition priority

Priority	Exit Condition
1	ResetS
2	FaultS
3	All other exit conditions

Signals and Conditions

The **ResetS** transition signal is described using the truth table shown in Table 6-11, below.

Table 6-11 ResetS transition signal truth table

VccReset (due to low Vcc)	**Reset** (hardware signal)	**SoftwareReset** (see Table 8-10)	**ResetS** (transition signal)
ASSERTED	X	X	1
DEASSERTED	ASSERTED	X	1
DEASSERTED	DEASSERTED	1	1
DEASSERTED	DEASSERTED	0	0

The ResetS transition signal can also be represented by the logic equation

ResetS = ASSERTED(VccReset) OR ASSERTED(Reset) OR SoftwareReset (Eq. 6-3)

The internal **VccReset** generic hardware signal is asserted within the module when the voltage of one or more of the Vcc power rails as observed at the module input drops below an implementation-defined value. Implementation of VccReset is optional.

The **Reset** generic hardware signal must be asserted by the host for longer than the minimum reset pulse duration to trigger a module reset. Refer to form factor-specific documentation for the minimum reset pulse duration. See Appendix A for a mapping of the generic signal name to form factor specific signals.

The **FaultS** transition signal truth table and logic equation are module implementation specific.

The **LowPwrS** transition signal is described by the truth table shown in Table 6-12, below.

Table 6-12 LowPwrS transition signal truth table

LowPwrRequestSW see Table 8-10	**LowPwrAllowRequestHW** see Table 8-10	**LowPwrRequestHW** hardware signal	**LowPwrS** transition signal
1	X	X	1
0	1	ASSERTED	1
0	1	DEASSERTED	0
0	0	X	0

The **LowPwrS** transition signal can also be represented by the logic equation

LowPwrS = LowPwrRequestSW OR (LowPwrAllowRequestHW AND ASSERTED(LowPwrRequestHW))

The **LowPwrExS** transition signal is described by the truth table shown in Table 6-13, below. It represents the exit conditions from the ModuleReady state, when the LowPwrS transition signal is TRUE AND all Data Paths have reached the DPDeactivated state (represented by the logical term **ModuleDeactivatedT**).

Table 6-13 LowPwrExS transition signal truth table

LowPwrS transition signal	**ModuleDeactivatedT**	**LowPwrExS** transition signal
1	1	1
1	0	0
0	1	0
0	0	0

The **LowPwrExS** transition signal can also be represented by the logic equation

LowPwrExS = LowPwrS AND ModuleDeactivatedT (Eq. 6-4)

where

ModuleDeactivatedT = (DPStateHostLane1 = DPDeactivated) AND
(DPStateHostLane2 = DPDeactivated) AND …
… AND …
(DPStateHostLaneN = DPDeactivated) (Eq. 6-5)

N = number of host lanes in the module

State Transition Table

Table 6-14 provides a summary of the high-level behaviors and properties of each module state for paged memory module implementations. Refer to sections 6.3.2.5-6.3.2.12 for detailed requirements for each state.

Table 6-14 Module state behaviors, paged memory modules

State	Power Mode	Behavior in state	Exit condition	Next state	Req./ Opt.
Resetting	High/ Low Power	Management interface and all module electronics transition to reset	Resetting completed	Reset	Rqd.
Reset	Low Power	Management interface and all module electronics in reset	ResetS transition signal is FALSE	MgmtInit	Rqd.
MgmtInit	Low Power	Management interface powering up and initializing	ResetS transition signal is TRUE	Resetting	Rqd.
			FaultS transition signal is TRUE	ModuleFault	
			Module management interface ready AND Interrupt by ModuleStateChangedFlag set. Note: must occur within a duration of tMgmtInit (see section 10.1)	ModuleLowPwr	
ModuleLowPwr	Low Power	Management interface available, entire paged management memory accessible, host may configure module	ResetS transition signal is TRUE	Resetting	Rqd.
			FaultS transition signal is TRUE	ModuleFault	
			LowPwrS transition signal is FALSE	ModulePwrUp	
ModulePwrUp	High Power	Module transitioning to high power mode	ResetS transition signal is TRUE	Resetting	Rqd.
			FaultS transition signal is TRUE	ModuleFault	
			LowPwrS transition signal is TRUE	ModulePwrDn	
			Power up activities are complete	ModuleReady	
ModuleReady	High Power	Module may be consuming power up to the level defined in the fields in Table 8-27	ResetS transition signal is TRUE	Resetting	Rqd.
			FaultS transition signal is TRUE	ModuleFault	
			LowPwrExS transition signal is TRUE	ModulePwrDn	
ModulePwrDn	High Power	Module transitioning to Low Power mode	ResetS transition signal is TRUE	Resetting	Rqd.
			FaultS transition signal is TRUE	ModuleFault	
			Module has returned to Low Power mode	ModuleLowPwr	
ModuleFault	*Low Power	Module is waiting for host action	Module power down	N/A	Opt.
			ResetS transition signal is TRUE	Resetting	

*It is suggested, if possible, that the ModuleFault state be in Low Power mode.

6.3.2.3 Module State Machine for Flat Memory Modules (Simplified)

Flat Memory Modules shall adhere to the behaviors described by the simplified Module State Machine shown in Figure 6-4.

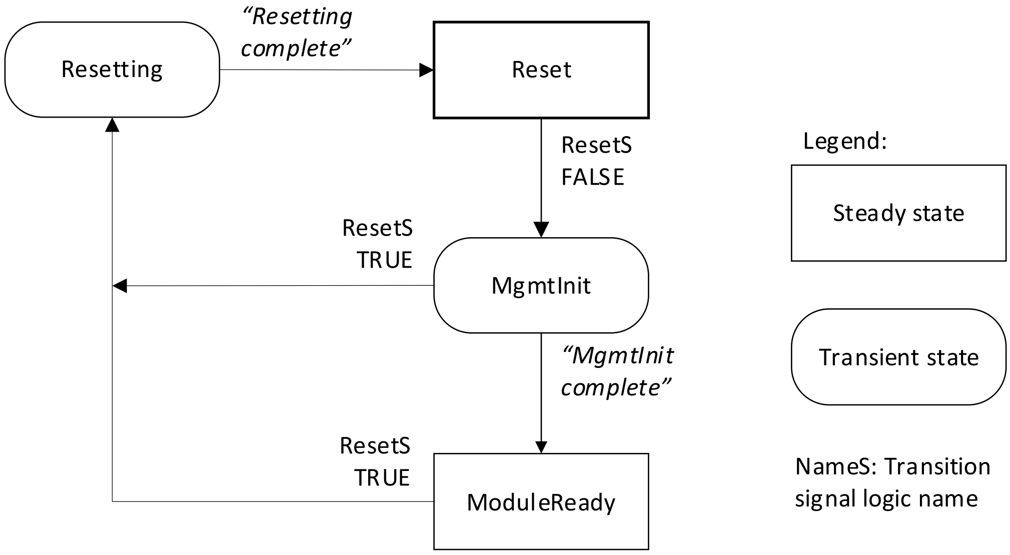


Figure 6-4 Flat Memory Module State Machine (MSM) State Transition Diagram

As shown in Figure 6-4, flat memory modules transition to the ModuleReady state without host interaction.

Module state transitions for flat memory modules are just a **specification formalism**, since the contents of the static EEPROM-based Memory Map does not change. Although the behaviors of the Resetting, Reset, and MgmtInit states apply to such modules, those **states are not reported** to the host through the Memory Map.

Flat memory modules shall statically report a module state of ModuleReady (see Table 8-4).

The ResetS transition signal is described in Table 6-11. For flat memory modules, assertion of the Reset signal may optionally hold the EEPROM in reset, however the Data Path remains active.

Flat memory modules are not required to support SoftwareReset or VccResetL.

At initial module insertion or application of power, the (conceptual) Module State Machine initializes to the Reset state. Table 6-15 provides a summary of the high-level behaviors and properties of each module state for flat memory modules. Refer to sections 6.3.2.5-6.3.2.12 for detailed requirements for each state.

Table 6-15 Module state behaviors, flat memory modules

State	Power Mode	Behavior in state	Exit condition	Next state	Required/ Optional
Resetting (not reported)	Low Power	Management interface transitions to reset	Resetting completed	Reset	Optional
Reset (not reported)	Low Power	Management interface in reset	ResetS transition signal becomes TRUE	MgmtInit	Optional
MgmtInit (not reported)	Low Power	Management interface powering up and initializing until host can access the flat management memory	ResetS transition signal becomes TRUE	Resetting	Required
			Module Management Interface ready is simply assumed after a duration tMgmtInit (see section 10.1), so hosts should just wait for that period.	ModuleReady	
ModuleReady (reported)	Low Power	Management interface available for host to access the flat management memory	ResetS transition signal becomes TRUE	Resetting	Required

6.3.2.4 Module Power Mode

The **Module Power Mode** dictates the maximum electrical power that the module is permitted to consume while operating in that Module Power Mode.

The Module Power Mode is a function of the state of the Module State Machine.

Two Module Power Modes are defined:

- In **Low Power Mode** (characteristic of all MSM steady states except ModuleReady) the maximum module power consumption is defined in the form factor-specific hardware specification.
- In **High Power Mode** (characteristic of the MSM state ModuleReady) the implementation dependent maximum module power consumption is advertised in the **MaxPower** Byte 00h:201 (see Table 8-27).

All modules initially boot in Low Power Mode. After the management interface has been initialized and the MSM has reached the steady ModuleLowPwr state the host may transition paged memory modules to High Power Mode using the conditions defined by the LowPwrS transition signal (see Table 6-12) provided that the advertised MaxPower value is supported in the host system.

If LowPwrS is FALSE when the module is in the ModuleLowPwr state, the module begins to enable High Power Mode operation, using the power up procedures defined for the ModulePwrUp state (see section 6.3.2.9).

Conversely, whenever LowPwrS or LowPwrExS (as applicable) is TRUE while the module is in or moving towards High Power Mode, the module begins to return to the ModuleLowPwr state and hence to Low Power Mode operation, using the power down procedures defined for the ModulePwrDn state (see section 6.3.2.11).

Note: The LowPwrS transition signal only controls the Module Power Mode but it does not control Data Path initialization. Refer to section 6.3.3 for more information on Data Path initialization.

6.3.2.5 Resetting State (Shutting Down)

The **Resetting** state is a transient state.

The Resetting state is entered from any state except the Reset state when the ResetS transition signal is TRUE. The ResetS transition signal is defined in Table 6-11.

On Entry
When a paged memory module enters the Resetting state, all Data Path State Machines cease to exist. Refer to section 6.3.3 for Data Path State Machine behaviors.

Autonomous Behavior
The module tries to gracefully power down module optics and electronics before entering the Reset state. The Resetting state initiates a complete module reset. The shutdown procedure used by the module for a reset event is implementation dependent.

Note: The module may be in High Power Mode during portions of the Resetting state.

Reactive Behavior
Management interactions initiated by the host during the Resetting state may be ignored by the module. Transactions in progress may be aborted when entering the Resetting state. Note: While the ResetS transition signal is TRUE, the management communication interface may be held in reset and may not respond (NACK).

Exit
When all module electronics have been powered down and are in reset, the module state transitions automatically to the Reset state.

6.3.2.6 Reset State (Ground State)

The **Reset** state is a steady state.

The Reset state is the initial Module State Machine state on module insertion or power-up.

On Entry
The self-clearing SoftwareReset bit (see Table 8-10) effectively returns to its default value. All other bits are not affected.

Note that this means the module must have some mechanism to clear this bit when in the Reset State or upon exiting the Reset State. Other bits will be set to their power-up default values later, in the MgmtInit State, regardless of their value when exiting the Reset State.

Autonomous Behavior

All internal module electronics are held in reset.

The module remains in Low Power mode.

All Interrupts to the host are suppressed.

Reactive Behavior

The module may abort any MCI transactions in progress on entry to the Reset state.

The module may ignore all MCI transactions while in the Reset state.

Note: While the ResetS transition signal is TRUE, the management interface may be held in reset and may not respond (NACK); when the ResetS transition signal has become FALSE, the module may still not respond until the subsequent MgmtInit state is exited.

Exit

The module exits the Reset state when the ResetS transition signal is found FALSE (see Table 6-11).

The Reset state can only be exited when the ResetS transition signal is FALSE and power is applied.

On exit from the Reset state, the Module State Machine enters the MgmtInit state.

6.3.2.7 MgmtInit State (Initializing Management Interface)

The **MgmtInit** state is a transient state.

The MgmtInit state is applicable to both paged memory modules and flat memory modules.

Entry

The MgmtInit state is entered any time the module comes out of the Reset state.

Autonomous Behavior

During the MgmtInit state, the module initializes the Memory Map to default values and initializes the management communication interface allowing the host to eventually manage the module.

The module may perform limited power-up of Data Path circuitry provided the module remains in Low Power Mode throughout this state.

Note that, for paged memory modules, all Data Path States are DPDeactivated in MgmtInit.

Reactive Behavior

The module may ignore all MCI transactions while in the MgmtInit state.

Exit

The regular exit occurs automatically when the specified autonomous behavior has run to completion.

The next state after regular exit is ModuleLowPwr.

On regular exit to ModuleLowPwr state the module has ensured that all Memory Map register locations have been set to their power-on default values.

6.3.2.8 ModuleLowPwr State (Basic Manageability)

The **ModuleLowPwr** state is a steady state.

It represents the situation where the management interface is fully initialized and operational while the device is still in Low Power Mode.

During this state, the host may configure the module using the management interface to read from and write to the management Memory Map. Details of host-module interactions in the ModuleLowPwr state are outside the scope of this specification.

Note: The Data Path State of all lanes is still DPDeactivated in the ModuleLowPwr state.

Note: Some examples of configuration activities include reading the ID and device property fields, setting CDR and other lane attributes and configuration of Interrupt related Masks.

On Entry

On entry into the ModuleLowPwr state, the module sets the Module State register (Table 8-4) to the ModuleLowPwr state and it sets the ModuleStateChangedFlag (Table 8-8) unless the ModuleLowPwr exit criteria are already met upon entry into the state.

Autonomous Module Behavior
The module waits for exit conditions to become true.

Reactive Behavior
The module reacts to all management operations but is allowed to ignore or reject commands that are not consistent with the Low Power condition.

Exit
The module state transitions to ModulePwrUp when the LowPwrS transition signal is FALSE (see Table 6-12).

Note: This transition can occur at any time during ModuleLowPwr, and so modules shall sense LowPwrS throughout the ModuleLowPwr state. This behavior was defined differently in CMIS 3.0

Note: The LowPwrS transition signal may evaluate to 0 the first time it is sampled in ModuleLowPwr. In such circumstances, the transition to ModulePwrUp may be too fast for the host to detect that the module was transiently in the ModuleLowPwr state.

6.3.2.9 ModulePwrUp state (Powering Up)

The **ModulePwrUp** state is a transient state. In this state the host is informed that the module is in the process of powering up to High Power Mode.

Note: The module is expected to power up module components as needed to expedite later module and Data Path reconfigurations for operational use.

Note that the Data Path State of all lanes is DPDeactivated in the ModulePwrUp state.

On Entry
On entry the module shall set the Module State register (Table 8-4) to ModulePwrUp.

Autonomous Behavior
The module may be in High Power mode at any time during the ModulePwrUp state.

Reactive Behavior
The module reacts to all management operations.

Exit
When the **LowPwrS** transition signal is TRUE at any time during the **ModulePwrUp** state, the module state immediately transitions to **ModulePwrDn**.

When the module power up sequence has completed, the module state transitions to the **ModuleReady** state.

6.3.2.10 ModuleReady State (Fully Operational)

The **ModuleReady** state is a steady state.

In this state the module is in High Power mode and the host may initialize or deinitialize Data Paths.

On Entry
On entry into the **ModuleReady** state, the module shall set the Module State register (Table 8-4) to the ModuleReady state and set the ModuleStateChangedFlag (Table 8-8).

Autonomous Behavior
The module operates as configured by the host.

Reactive Behavior
The module reacts to all management operations and to all relevant input signal changes.

Exit
Except for Reset or Fault, the action that results in an exit from ModuleReady is if the **LowPwrExS** transition signal is TRUE (see Table 6-13), which causes the module state to transition to **ModulePwrDn**.

6.3.2.11 ModulePwrDn State (Powering Down)

The **ModulePwrDn** state is a transient state.

In this state the host is informed that the module is in the process of returning to Low Power mode.

Note that the Data Path State for all lanes is DPDeactivated in the ModulePwrDn state.

On Entry
On entry into the ModulePwrDn state, the module shall set the Module State register (Table 8-4) to the ModulePwrDn state.

Autonomous Behavior
The module reduces the power consumption of module electronics such that the module power consumption is less than the Low Power Mode threshold. The electronics associated with the management interface remains powered and available.

The module may still be in High Power mode at any time during the ModulePwrDn state.

Reactive Behavior
Modules in ModulePwrDn shall ignore the LowPwrS transition signal, so if this signal is FALSE during ModulePwrDn, the module will complete the power-down sequence and transition to ModuleLowPwr before sampling LowPwrS again.

Exit
When the module has arrived in Low Power Mode, the module state transitions to the ModuleLowPwr state.

6.3.2.12 ModuleFault State (Module Fault)

The **ModuleFault** state is a steady state.

The ModuleFault state is provided for notification to the host that a module **fault** has occurred after which the module aims to prevent physical damage and to avoid safety risks for the module or for its environment (e.g. host).

The exact ModuleFault state entry conditions are implementation dependent. The ModuleFault state is only entered when module detects a condition (e.g. TEC runaway, memory corruption) that could compromise safety or cause damage. The specification intent of the ModuleFault state is to put the module in a condition that does not compromise safety or create further equipment failures.

On Entry
On entry the module shall set the ModuleState register (see Table 8-5) to ModuleFault.

The module may write information about the fault cause into the ModuleFaultCause register (see Table 8-15)

Autonomous Behavior
It is strongly recommended that the module enters Low Power mode during the ModuleFault state but the response to a Fault condition is implementation specific.

Reactive Behavior
The module reacts only to events that cause the ResetS transition signal to become TRUE.

The module may respond to READ access if that is still possible.

Exit
Except for a power cycle, the only exit path from the ModuleFault state is to perform a module reset by taking an action that causes the ResetS transition signal to become TRUE (see Table 6-11).

6.3.3 Data Path State Machines (DPSM)

A **Data Path State Machine** (**DPSM**) instance describes Data Path-specific behaviors and properties that are related to the configuration of the Data Path, as managed by the host.

Note: For configuration dependent behaviors and properties of the module as a whole, refer to the Module State Machine described in section 6.3.2.

Note: As described in section 6.2.2, a Data Path is a bidirectional combination of one or more host lanes, one or more media lanes, and a set of internal module resources implementing the Application that is described in an associated Application Descriptor.

*Note: The DPSM state represents a **management** or **configuration realization status** of a Data Path, representing the effects of certain host configuration commands and of module reactions to those commands. It **does not** necessarily represent other behavioral or operational aspects of a bidirectional Data Path, e.g. in terms of current input or output signal conditions or in terms of transmission service being provided.*

*Note: The DPSM state should neither be confused with the **operational status** of the functional resources of a Data Path in Tx direction or in Rx direction, nor with the resulting signal **output status** of Rx host lane outputs or of Tx media lane outputs, which are reported independently in separate Output Status registers (see section 8.9.2).*

Data Paths (and Data Path State Machines) are only applicable to **paged memory modules**.

Module State and DPSM Life Cycle

All **DPSM** instances required to represent the power-up default Application defined in the Data Path Configuration field values of the Active Control Set are initially created and set-up during the **MgmtInit** state.

After its creation, a DPSM remains in the DPDeactivated State until the Module State Machine is in the **ModuleReady** state and an exit condition from the DPDeactivated state is met.

When the host updates the Data Path Configuration fields in the Active Control Set, in either the **ModuleLowPwr** or **ModuleReady** states, the module tears down any previous DPSM that is no longer defined and then creates and sets up any newly defined DPSM.

All Data Path State Machines are torn down in the **Resetting** state.

Note: Refer to section 6.3.2 for an overview of the Module State Machine, section 6.2.1 for an overview of Applications, section 6.2.2 for an overview of Data Paths, and section 6.2.3 for an overview of Control Sets.

DPSM Purpose

A Data Path State Machine is used by the module to represent the initialization status of the resources associated with a Data Path in response to certain host configuration settings or commands. Although individual resources within a Data Path may complete initialization activities at different times, the module waits to report the updated DPSM state until all resources associated with the Data Path have completed the requested configuration or reconfiguration action. This synchronized status reporting across all lanes and resources in a Data Path reflects the fact that there is only one Data Path State Machine per Data Path.

Note: The DPSM specification model does not imply any specific way of implementation.

Note: Some example Data Path initialization sequences are provided in Appendix D.

Note: Modules identify supported Data Path configurations through the Application advertisement fields.

DPSMs for parallel Data Paths of Multiple Application Instances

Each Data Path in a module is required to operate independently of other Data Paths: if the host changes the Data Path State of one Data Path, the other Data Paths in the module shall be unaffected and uninterrupted.

Note: Independent operation of Data Paths may require independent clocking per Data Path, from a recovered clock within that Data Path. See the applicable hardware specification for further information.

Note: The module shall only advertise Applications and lane configurations that are supported by the implemented clocking scheme.

6.3.3.1 DPSM State Transition Diagram and DPSM Specification

Figure 6-5 shows the state transition diagram (STD) of a DPSM representing the Data Path configuration related state of one Data Path instance[1].

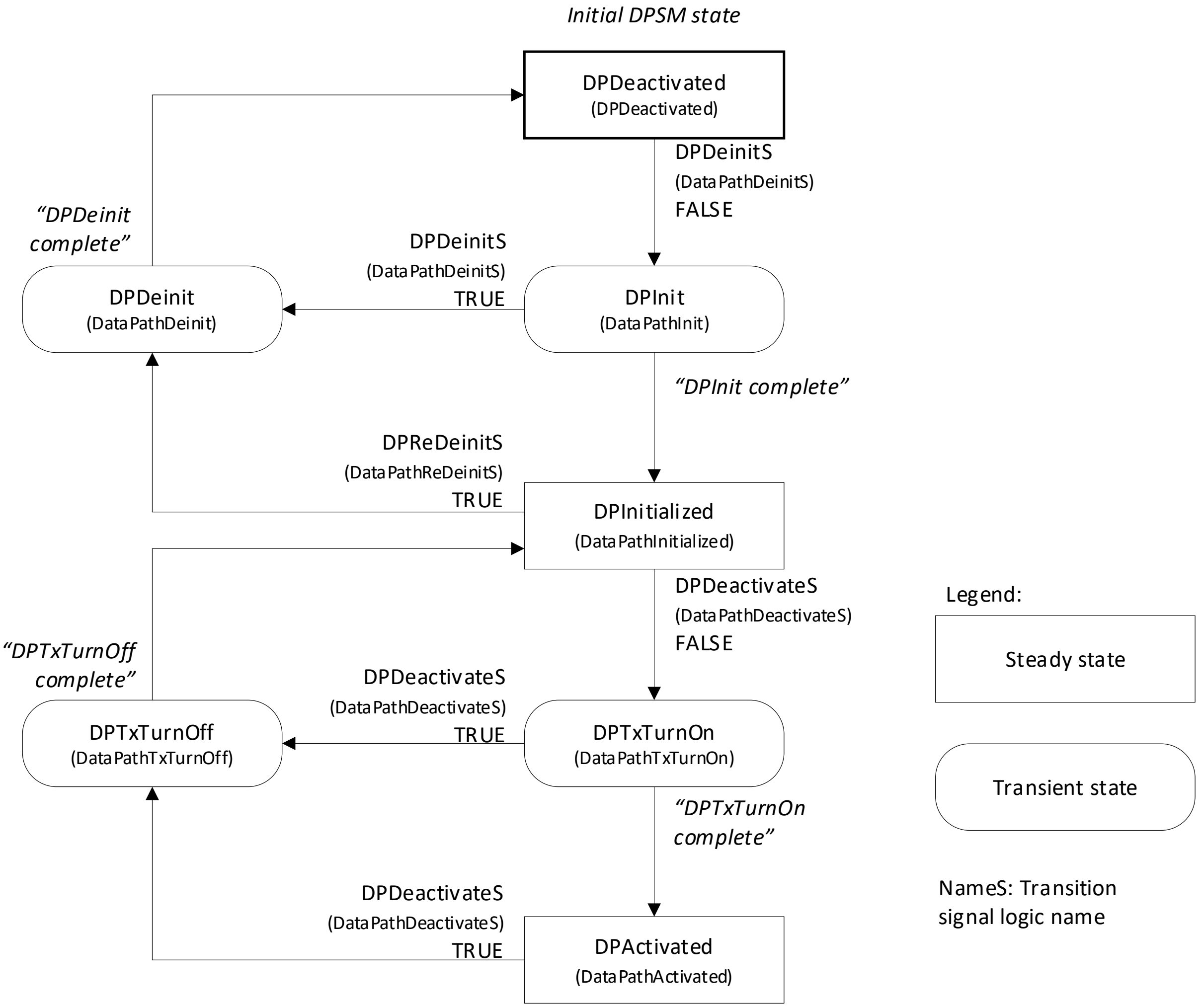


Figure 6-5 Data Path State Machine (DPSM) State Transition Diagram

Note: Prior to exit from the MgmtInit module state, all Data Paths initialize to the DPDeactivated state.

State Exit Conditions and Transition Signals

The state machine exits a given state when specific exit conditions are satisfied. So called **Transition Signals** (recognized by name suffix S) represent these exit conditions for steady states.

DPDeinitS (DataPathDeinitS)

The DPDeinitS (DataPathDeinitS) transition signal is defined using the truth table shown in Table 6-16.

Table 6-16 DPDeinitS transition signal truth table

ModuleReadyT (MSM term)	**LowPwrS** (MSM, Table 6-12)	**DPDeinitT** (term)	**DPDeinitS** (transition signal)
0	X	X	1
1	1	X	1
1	0	1	1
1	0	0	0

[1] For brevity, a 'DataPath' prefix in the name of a state, a transition signal, or a named logical predicate may always be replaced by 'DP'. The shortened names are fully equivalent synonyms to the full names.

The DPDeinitS transition signal can also be represented by the logic equation

DPDeinitS = (NOT ModuleReadyT) OR LowPwrS OR DPDeinitT (Eq. 6-6)

where

ModuleReadyT = (ModuleState = ModuleReady) (Eq. 6-7)

DPDeinitT = DPDeinitLane<N> (Eq. 6-8)
OR DPDeinitLane<N+1>
...
OR DPDeinitLane<N+M-1>

N = first host lane in the Data Path
M = number of host lanes in the Data Path

DPReDeinitS (DataPathReDeinitS) when SteppedConfigOnly = 0

The DPReDeinitS (DataPathReDeinitS) transition signal is defined using the truth table shown in Table 6-17 when the module supports intervention-free reconfiguration, which is advertised by SteppedConfigOnly = 0 (see Table 8-4).

Table 6-17 DPReDeinitS transition signal truth table (default)

DPDeinitS (transition signal)	**DPReinitT** (term)	**DPReDeinitS** (transition signal)
1	1	1
1	0	1
0	1	1
0	0	0

The DPReDeinitS transition signal can also be represented by the logic equation

DPReDeinitS = DPDeinitS OR DPReinitT (Eq. 6-9)

where

DPReinitT = DPInitPendingLane<N> (Eq. 6-10)
OR DPInitPendingLane<N+1>
...
OR DPInitPendingLane<N+M-1>

N = first host lane in the Data Path
M = number of host lanes in the Data Path

The **DPInitPending** status register is described in section 8.9.7. Each **DPInitPendingLane**<i> bit represents the condition following a successful **ApplyDPInit** trigger that a transit through state **DPInit** is pending.

The module simultaneously **sets** the DPInitPendingLane<i> bits for all triggered lanes <i> of a Data Path after it has successfully copied all settings for all those triggered lanes <i> from a selected Staged Control Set to the Active Control Set during a **Provision** procedure that was triggered by a host WRITE to one of the two SCS<k>::**ApplyDPInit** trigger registers (see section 8.8.3.1, 8.8.4.1).

*Note: DPInitPending bits are not set in response to **ApplyImmediate** triggers.*

The module **clears** all DPInitPendingLane<i> bits of a Data Path while in DPSM state **DPInit**.

Note: The desired effect of the DPInitPending condition variable is to report a discrepancy between Active Control Set and hardware until it is resolved in the DPInit state. When intervention-free reconfiguration is supported (by default: SteppedConfigOnly = 0), this discrepancy causes an intervention-free DPSM state cycle through DPDeactivated and DPInit after ApplyDPInit was successfully triggered in DPInitialized or DPActivated.

DPReDeinitS (DataPathReDeinitS) when SteppedConfigOnly = 1

When the module advertises lack of support for intervention-free reconfiguration, by SteppedConfigOnly = 1 (see Table 8-4), then the DPReDeinitS transition signal is simplified as follows

DPReDeinitS = DPDeinitS (Eq. 6-11)

Note: The intentional change in behavior, when only stepwise configuration is supported, is that ApplyDPInit has no impact on the DPSM dynamics at all, and is therefore fully independent of the DPSM.

DPDeactivateS (DataPathDeactivateS)

The DPDeactivateS (DataPathDeactivateS) transition signal is defined by the logic equation

DPDeactivateS = DPReDeinitS OR DPTxDisableT OR DPTxForceSquelchT (Eq. 6-12)

where

DPTxDisableT = OutputDisableTx<N> (Eq. 6-13)
OR OutputDisableTx<N+1>
...
OR OutputDisableTx<N+M-1>

DPTxForceSquelchT = OutputSquelchForceTx<N> (Eq. 6-14)
OR OutputSquelchForceTx<N+1>
...
OR OutputSquelchForceTx<N+M-1>

N = first media lane in the Data Path
M = number of media lanes in the Data Path

*Note: When in **DPActivated** or **DPTxTurnOn** states, setting OutputDisableTx or OutputSquelchForceTx on one media lane of a Data Path intentionally causes that entire Data Path to transition to **DPInitialized** via DPTxTurnOff. Although some media lanes of the Data Path may continue to be operational (i.e. media lanes with output neither disabled nor squelched), as long as some media lanes are not operational, the Data Path as a whole is considered not activated.*

Reaction to Module Reset

When the MSM **ResetS** transition signal (see Table 6-11) becomes TRUE, any Data Path related power down activities are performed in the **Resetting** module state. The DPSM state machines cease to exist in this case.

Note: Module dependent pre-reset clean up and power down activities may be implemented, possibly depending also on the reset trigger in either hardware or software.

Reaction to Module Fault

When Module State Machine transitions to **ModuleFault** state, the DPSM behavior is not defined formally, but governed by the behavioral requirements of the ModuleFault state.

DPSM State and Tx Output Status

Table 6-18 provides a high-level summary of the Tx output behaviors and characteristics of each Data Path state and outlines the meaning of exit conditions of transient states that are not captured in formal transition signal definitions. Refer to sections 6.3.3.4-6.3.3.10 for detailed requirements for each state.

Table 6-18 Data Path state behaviors and Exit Conditions

State	Output Status Tx	Exit condition	Next State
DPDeactivated	Quiescent	DPDeinitS is FALSE	DPInit
DPInit	Quiescent	DPDeinitS is TRUE	DPDeinit
		DPInit complete: Data Path resource initialization completed	DPInitialized
DPInitialized	Depends on per-lane OutputDisableTx and OutputSquelchForceTx	DPReDeinitS is TRUE	DPDeinit
		DPDeactivateS is FALSE	DPTxTurnOn
DPDeinit	In transition	**DPDeinit complete**: Module dependent deinitialization completed	DPDeactivated
DPTxTurnOn	In transition	DPDeactivateS is TRUE	DPTxTurnOff
		DPTxTurnOn complete: All Data Path Tx outputs are operational	DPActivated
DPActivated	Operational	DPDeactivateS is TRUE	DPTxTurnOff
DPTxTurnOff	In transition	**DPTxTurnOff complete**: Physical status of some Tx outputs reflects the nominal configuration of being squelched or disabled on host command	DPInitialized

DPSM State and Rx Output Status

Except for an implicit dependency on the initialization of internal functional resources of the Rx Data Path on host and media side, which occurs in the first DPInit state traversal, the Rx output state is **not further controlled** by the Data Path State Machine.

After Rx Data Path resource initialization in the DPInit state, a module always forwards a **valid** Rx input signal to a valid Rx output signal in the DPInitialized, DPTxTurnOn, DPTxTurnOff, DPActivated states (unless overridden by Rx output muting host commands).

When **returning** to the other DPSM states (DPDeinit, DPDeactivated, DPInit), a module may or may not mute the Rx output signal; especially it may transmit a **valid** Rx output signal when a valid Rx input signal is available.

*Note: See section 3.3 for the definition of a **valid** signal.*

The host can always ensure a muted Rx output using the OutputDisableRx control (Table 8-61).

The module mutes the Rx output when supported automatic squelching functionality is enabled and when the relevant squelch conditions are present.

The module always reports the **resulting actual status** of the Rx output signal in the OutputStatusRx register, independent of the Data Path State Machine state.

6.3.3.2 Data Path Control (Host)

A single main configuration register is provided for the host to control initialization and deinitialization of all Data Paths represented in a given Bank. This **DPDeinit** register (see Table 8-60) defines per host lane if lane or the associated Data Path resources are determined to be unused for functional operation (and hence can be deinitialized) or if they are determined for functional operation (and hence need to be initialized).

Note: Per-lane configuration has been chosen to allow a variety of Data Path configurations from a single Memory Map specification.

*Note: Initialization status and behavior of Tx media lane outputs are further controlled by the host using the media lane specific control bits **OutputDisableTx**<i> and **OutputSquelchForceTx**<i>.*

A host requesting initialization or deinitialization of a Data Path ensures that the Active Control set contains the desired configuration settings and then writes the value 0 or 1, respectively, to the DPDeinit bits associated with the lanes of that Data Path.

The host may request initialization or deinitialization of multiple Data Paths with one register access.

Some informative Data Path initialization flow examples are provided in Appendix D to facilitate understanding of the relationship between the initialization of physical structures in the module and Data Path-level reporting in the Memory Map.

6.3.3.3 Data Path Status (Module)

The module provides information on the current state of the Data Path (DPSM **current state reporting**) and on entry to certain DPSM states (DPSM **state change indication**).

DPSM Current State Reporting

On entry to a DPSM state the module reports the DPSM state entered as the current DPSM state in the **DPState** status register (see Table 8-73), on all lanes of the Data Path, with optional exceptions specified below.

Note: Due to the identical behavior of all lanes of a Data Path the host needs to read only the first lane of the Data Path to determine the Data Path state.

Note: The DPSM model in this specification describes a single state machine per Data Path, with Data Path attributes replicated on all lanes of the Data Path. This does not limit software implementations.

DPSM Current State Reporting Exceptions

The module **may** suppress reporting the current DPSM state in the DPStateHostLane<i> registers when that state is known to be transitional, i.e. when it is exited immediately because its exit conditions are fulfilled on entry, or when the duration of staying in that state is known to be in the order of 1 ms or less.

Note: The duration specification is intentionally vague. The intention for allowing exceptions in state reporting is to avoid reporting short-lived status data which the host is unlikely to read and react upon.

DPSM State Change Indication (Flag)

A DPSM State Change Indication consist of the module setting a **DPStateChangedFlag** for each lane of the Data Path associated with the relevant DPSM instance.

Note: The intention of the following specification is that the module indicates a state change only on entry to a lasting steady state and only when the transition time since the previous lasting steady state was significant.

The maximum duration of a transient state is advertised in a MaxDuration* field (see Table 8-41, Table 8-50) and is considered insignificant when the coded MaxDuration* field value is 0000b (see Table 8-42).

The module performs a DPSM State Change Indication on entry to a **steady** DPSM state when the following two Flag setting conditions are fulfilled:

- No exit condition of the entered **steady** state is fulfilled on state entry (state is not visited transitorily)
- The advertised maximum state **duration** is **significant** for at least one **transient** state passed through since the previous State Change Indication for this Data Path, or since module reset if there was no such State Change Indication yet since reset.

The module does **not** perform a DPSM State Change Indication on entry to a **transient** DPSM state.

Table 6-19 defines the Flag behavior for each DPSM state entry.

Table 6-19 Data Path State Changed Flag behaviors

Entered state	DPStateChangedFlag may be set *
DPInit	No
DPInitialized	**Yes**
DPDeinit	No
DPDeinit	No
DPDeactivated	**Yes**
DPTxTurnOn	No
DPActivated	**Yes**
DPTxTurnOff	No
DPTxTurnOff	No
DPInitialized	**Yes**

** Note: The Flag setting conditions are described in the main text.*

Note: Steady state exit conditions may already be met upon entry into the steady state and lead to immediate transition to the next state (after state entry or state exit activities, if defined).

Flag Related Behavior

The module does not clear any Flag due to a state change of a Data Path State Machine.

The module raises Flags only according the DPSM state-specific conformance rules defined in section 6.3.4.

6.3.3.4 DPDeactivated State (Ground State)

The **DPDeactivated** (DataPathDeactivated) state is a steady state.

This per-lane state indicates that no Data Path is initialized on the indicated lane(s).

The host may configure or reconfigure Data Paths on lanes reporting DPDeactivated.

Note: Reconfiguration begins with update of the Active Control Set.

Note: When SteppedConfigOnly=1, the host may reconfigure Data Paths in other states as well, but this is not recommended, as described in section 6.2.4.3.

Autonomous Behavior

On entry to this state, the module updates the Data Path state register (see Table 8-73) and the Data Path State Changed Flag (Table 8-77) for all lanes in the Data Path according to the rules described in section 6.3.3.3.

While in DPDeactivated state, all Tx media lane outputs of the Data Path shall be quiescent while the status of all Rx host lane outputs of the Data Path are undefined.

Note: The Rx output status of the Rx host lanes of a Data Path depends on host controlled configuration history and on unspecified module controlled deinitialization behaviors (see sections 6.3.3.7 and 6.3.3.10).

Reactive Behavior (on Host Actions)

Changes to the OutputDisableTx or OutputSquelchForceTx register values for Data Paths in DPDeactivated shall have no impact on the output quiescence of those Data Paths.

Exit

The Data Path remains in DPDeactivated as long as ResetS is TRUE (see Table 6-11).

Otherwise, the DPSM transitions to DPInit when the DPDeinitS transition signal is FALSE (see Table 6-16).

The Host shall provide a **valid** high-speed Tx input signal at the required signaling rate and encoding type prior to causing a DPSM to exit the DPDeactivated state.

Note: The module must receive a valid input signal while performing initialization activities in the subsequent DPInit state, such as adaptation of signal integrity equalizer settings. Otherwise, if no valid Tx input signal is available, the resulting initialization behavior is not predictable (the module may or may not wait for a valid signal and may or may not initialize to a possibly inadequate condition).

6.3.3.5 DPInit State (Initializing)

The **DPInit** (DataPathInit) state is a transient state.

In this state the module performs all initialization activities on the internal resources of the Data Path that are necessary to make the Data Path operational.

Note: Initialization activities include the realization of the selected Application properties and/or adaptation of signal integrity settings, and possibly power up and initialization of Tx and Rx Data Path electronics if opportunistic power savings were employed by the module in DPDeactivated.

Note: The module assumes a valid high-speed Tx input signal when entering the DPInit state.

Autonomous Behavior

On entry to this state, the module updates the Data Path state register (see Table 8-73) for all lanes in the Data Path according to the rules described in section 6.3.3.3.

Within the DPInit state, the module performs any necessary power-up and initialization activities for module electronics associated with the Data Path.

Note: In some cases, these electronics may be shared between multiple Data Paths. Depending on prior power up and power down actions, some or all of these electronics may already be powered; in such cases, the power up sequence is bypassed. The details of the power up sequence are implementation-dependent and outside the scope of this specification.

During DPInit, the module realizes the Application properties and signal integrity settings as defined in the Active Control Set (see section 6.2.3) by configuring the relevant internal resources of the Data Path.

Note: The details of how the module applies Application settings is implementation-dependent and outside the scope of this specification.

Note: Attributes that require adaptation, such as CTLE settings, are adapted at the appropriate time during DPInit. The order in which signal integrity settings are applied and adapted is implementation-dependent and outside the scope of this specification.

The module clears the DPInitPending bits of all lanes in the Data Path.

While in DPInit, all Tx media lane outputs of the Data Path shall be quiescent, while the status of all Rx host lane outputs associated with the Data Path are undefined.

Note: The actual Rx output status of the host lanes of the Data Path depends on configuration history and on unspecified (i.e. module or vendor specific) deinitialization behaviors (see sections 6.3.3.7 and 6.3.3.10).

When no Rx input signal is present at the module Rx input at the time of initialization, the module configures its electronics such that any required adaptation or CDR locking occurs automatically at a later point in time when an input signal is provided, without host intervention.

Reactive Behavior (on Host Actions)

Changes to OutputDisableTx<i> or to OutputSquelchForceTx<i> controls shall have no impact on the media lane output quiescence of the Data Path.

Note: It is recommended that hosts minimize management operations while in this state. Dynamic Memory Map content may be unreliable while in this state and should not be read or written.

Exit

If the DPDeinitS logic signal is TRUE at any time during DPInit, the Data Path State Machine transitions to DPDeinit.

Otherwise, once the module has completed power-up and initialization of all Tx and Rx resources associated with the Data Path, and all associated Tx and Rx Flags and status registers are valid, the Data Path State Machine state transitions to DPInitialized.

Note: If the module fails to complete DPInit, the host may determine this by observing that the maximum advertised duration of DPInit has been exceeded (see MaxDurationDPInit in Table 8-41). Actions taken by the host in response to such a failure are outside the scope of this specification.

6.3.3.6 DPInitialized State (Initialized)

The **DPInitialized** (DataPathInitialized) state is a steady state.

In this state all functional resources of the Data Path are fully initialized. However, the output of one or more Tx media lanes whose Data Path stays in DPInitialized is either squelched by the host or disabled, and so the Data Path is not ready to transmit traffic.

Autonomous Behavior

On entry to this state, the module updates the Data Path state register (see Table 8-73) and the Data Path State Changed Flag (Table 8-77) for all lanes in the Data Path according to the rules described in section 6.3.3.3.

Reactive Behavior (on Host Actions)

Transmitter output quiescence for Data Paths in DPInitialized is configured per media lane by the setting in the OutputDisableTx<i> and OutputSquelchForceTx<i> controls.

Note: A media lane output may also be quiescent when a module internal auto-squelching controller reacts to a Tx LOS condition on a host lane input and decides to squelch the media lane output.

Exit

If the DPReDeinitS transition signal is TRUE at any time during DPInitialized, the Data Path State Machine transitions to DPDeinit.

Otherwise, if the DPDeactivateS signal is FALSE at any time during DPInitialized, the Data Path State Machine transitions to DPTxTurnOn. Either of these conditions may be met upon entry into DPInitialized.

6.3.3.7 DPDeinit State (Deinitializing)

The **DPDeinit** (DataPathDeinit) state is a transient state.

In this state the module may deinitialize the module internal resources associated with a Data Path.

Note: Deinitialization tasks are implementation dependent but can include tasks such as opportunistic power savings, hardware reconfiguration, or module software variable clean-up.

Autonomous Behavior

On entry to this state, the module updates the Data Path state register (see Table 8-73) for all lanes in the Data Path according to the rules described in section 6.3.3.3.

During DPDeinit, the module may power down applicable Data Path electronics for opportunistic power savings. In some cases, electronics may be shared with other Data Paths that are not in DPDeinit or DPDeactivated. In such cases, these electronics remain powered. Similarly, modules may keep electronics that require significant power up times powered even when the host requests Data Path deinitialization.

Note: If a host wants to ensure maximum power savings, the host should initiate a module transition to Low Power Mode by causing LowPwrS to become TRUE, but this will deactivate all Data Paths in the module.

In DPDeinit, all Tx media lane outputs of the Data Path shall eventually become quiescent while the status of all Rx host lane outputs associated with the Data Path are undefined.

Note: The actual Rx output status of the Rx host lanes of the Data Path depends on configuration history and on unspecified (i.e. module or vendor specific) deinitialization behaviors (see also section 6.3.3.10).

Reactive Behavior (on Host Actions)

Changes to OutputDisableTx or OutputSquelchForceTx for Data Paths in DPDeinit shall have no impact on the output quiescence of those Data Path outputs.

Note: It is recommended the host minimize management operations while in this state. Dynamic Memory Map content may be unreliable for lanes in this state and should not be read or written.

Exit

When the module has completed deinitialization activities on <u>all</u> resources associated with the Data Path, the Data Path State Machine transitions to DPDeactivated.

6.3.3.8 DPTxTurnOn State (Turning On)

The **DPTxTurnOn** (DataPathTxTurnOn) state is a transient state.

In this state the module unmutes the Tx output for all media lanes associated with the Data Path.

Note: In somewhat abnormal operational conditions, when auto-squelched by the module, a lane output may actually remain or become quiescent, overriding the host configured output status.

Autonomous Behavior

On entry to this state, the module updates the Data Path state register (see Table 8-73) for all lanes in the Data Path according to the rules described in section 6.3.3.3.

While in DPTxTurnOn, all Tx outputs associated with the Data Path shall be in transition and the status of all Rx outputs associated with the Data Path are as reported in the Rx Output Status indicator (see Table 8-75).

Reactive Behavior (on Host Actions)

Note: It is recommended the host minimize management operations while in this state.

Exit

If the DPDeactivateS transition signal becomes TRUE at any time during DPTxTurnOn, the Data Path State Machine transitions to DPTxTurnOff.

The Data Path state advances to DPActivated once all Tx outputs associated with the Data Path are enabled, have stabilized, and are ready to transmit live traffic.

6.3.3.9 DPActivated State (Operational)

The **DPActivated** (DataPathActivated) state is a steady state.

In this state Data Paths are fully operational (initialized and ready to transmit traffic).

Autonomous Behavior

On entry to this state, the module updates the Data Path state register (see Table 8-73) and the Data Path State Changed Flag (Table 8-77) for all lanes in the Data Path according to the rules described in section 6.3.3.3.

All Tx outputs associated with the Data Path in DPActivated are unmuted and operational throughout the state.

Note: While operational, Tx outputs may still be auto squelched, overriding the host configured output status.

Reactive Behavior (on Host Actions)

The module reacts to all events at all interfaces in this DPSM state of providing regular transmission service for the fully operational Data Path

Exit

The Data Path state transitions to DPTxTurnOff if the host causes the DPDeactivateS transition signal to become TRUE for that Data Path.

One way for the DPDeactivateS transition signal to become TRUE is if the host triggers the ApplyDPInit bits associated with the Data Path.

The host may reconfigure one or more Data Paths while in DPActivated by defining a new Application in one of the Staged Control Sets and then using ApplyDPInit.

The host shall set ApplyDPInit to the same value for all lanes in the Data Path being reinitialized.

Note: When ApplyDPInit bits are triggered, the Data Path State Machine will transition through the DPTxTurnOff → DPInitialized → DPDeinit → DPDeactivated → DPInit → DPInitialized → DPTxTurnOn → DPActivated state sequence, reinitializing the new Data Path configuration in DPInit.

Prior to triggering the ApplyDPInit bits for applicable lanes, the host shall provide a valid high-speed input signal at the required signaling rate and encoding type.

The ApplyDPInit bits for all lanes in the Data Path shall be triggered with one register access.

The host may request reinitialization of multiple Data Paths in the same register access.

Data Paths excluded from the ApplyDPInit selector are not affected.

Note: This selective control allows host reconfiguration of individual Data Paths without affecting the operation of other Data Paths in the module.

The DPDeactivateS transition signal will also become TRUE if the host sets Tx Output Disable or Tx Force Output Squelch for any lane of the Data Path.

6.3.3.10 DPTxTurnOff State (Turning Off)

The **DPTxTurnOff** (DataPathTxTurnOff) state is a transient state.

In this state the module performs the programmed OutputDisableTx and/or OutputSquelchForceTx action on applicable media lanes in the Data Path.

Autonomous Behavior

On entry to this state, the module updates the Data Path state register (see Table 8-73) for all lanes in the Data Path according to the rules described in section 6.3.3.3.

Tx media lane output quiescence for Data Paths in DPTxTurnOff is eventually determined per media lane by the programmed setting in the OutputDisableTx and OutputSquelchForceTx controls.

Note: An externally caused Tx LOS condition of the lane also mutes the Tx output.

Reactive Behavior (on Host Actions)

Note: It is recommended that a host minimizes register accesses while in this state.

Exit

The Data Path state advances to DPInitialized after the OutputDisableTx or OutputSquelchForceTx configuration actions (causing entry to the DPTxTurnOff state) have been realized and the lanes are quiescent and stable.

6.3.4 Flagging Conformance Rules per State

Some Flags (causing a host Interrupt unless masked) are raised by virtue of the Module State Machine (MSM) or by virtue of an active Data Path State Machine (DPSM), but the majority of Flags are raised by other sources.

In some states, certain Flags are not applicable and shall not be raised by the module.

The following sections define the conformance rules for all Flags, for each state of MSM and of DPSM.

Note: The presence of a flagging conformance rule for a particular Flag does not imply that that Flag is required. A conformance rule for a Flag is applicable only when the (required or optional) Flag is actually supported.

6.3.4.1 Module-Level Flagging Conformance Rules per Module State

Table 6-20 describes the Flagging conformance rules for all module-level Flags, per MSM state.

Flag Setting Restrictions

While in an MSM state where a Flag is indicated as **N/A** (Not Allowed), the module shall not set that Flag.

All module-level Flags are generally N/A throughout the **Resetting**, **Reset**, and **MgmtInit** MSM states.

Module-level Flagging conformance is independent of Data Path State.

The meaning of some module-level Flags is configuration dependent; Table 6-20 defines the Flag conformance for each configuration option for those Flags.

Note: The host can suppress undesirable Interrupts caused by known Flags by setting the corresponding known Mask bit at any time after the management interface is initialized.

Table 6-20 Module Flag Conformance Rules

Flag / Flag Group	Page	Byte	Resetting Reset MgmtInit	ModuleLowPwr ModuleFault	ModulePwrUp ModulePwrDn ModuleReady
ModuleStateChangedFlag	00h	8	N/A	allowed	allowed
ModuleFirmwareErrorFlag	00h	8	N/A	allowed	allowed
DataPathFirmwareErrorFlag	00h	8	N/A	allowed	allowed
CdbCmdCompleteFlag*	00h	8	N/A	allowed	allowed
TempMon*Flag	00h	9	N/A	allowed	allowed
VccMon*Flag	00h	9	N/A	allowed	allowed
Aux1Mon*Flag (see Table 8-9)	00h	10	N/A	N/A	allowed
Aux2Mon*Flag (see Table 8-9)	00h	10	N/A	N/A	allowed
Aux2Mon*Flag (see Table 8-9)	00h	10	N/A	N/A	allowed
Aux3Mon*Flag (see Table 8-9)	00h	11	N/A	N/A	allowed
Aux3Mon*Flag (see Table 8-9)	00h	11	N/A	allowed	allowed
CustomMon*Flag	00h	11	N/A	[1]	allowed

Note 1: The CustomMon*Flag of a vendor-defined monitor is allowed in the ModuleLowPwr and ModuleFault states only if the monitor relates to functionality available in Low Power Mode.

Flag Specification Conformance

Setting allowed alarm and warning Flags of module-level monitors including associated Interrupt generation as per specified Flag semantics is only assured in the **ModuleReady** MSM state.

6.3.4.2 Lane-Specific Flagging Conformance per Data Path State

Table 6-21 and Table 6-22 describe the Flagging conformance for all lane-specific Flags, per DPSM state.

Flag Setting Restrictions

While in a DPSM state where a Flag is indicated as **N/A** (not allowed), the module shall not set that Flag.

All lane-specific Flags are N/A throughout the **Reset** and **MgmtInit** MSM states.

For all other MSM states, the DPSM State determines lane-specific Flagging conformance.

Note: For Flags allowed in a DPSM state, additional and more specific rules may exist

Note: The host can suppress undesirable Interrupts by setting the corresponding Mask bit at any time after the management interface is initialized.

Flag Specification Conformance

Setting permitted ('allowed') alarm and warning Flags of Data Path related monitors, including associated Interrupt generation, is only assured in the DPInitialized and DPActivated states.

Table 6-21 Lane-Specific Flagging Conformance Rules

Flag / Flag Group [1]	Page	Byte	DPDeactivated	DPInit	DPDeinit	DPInitialized	DPTxTurnOn	DPTxTurnOff	DPActivated
Data Path Related Flags									
DPStateChangedFlag*	11h	134	allowed	N/A	N/A	allowed	N/A	N/A	allowed
Tx Related Flags									
FailureFlagTx*	11h	135	allowed	allowed	allowed	allowed	allowed	allowed	allowed
LOSFlagTx*	11h	136	N/A	N/A	N/A	allowed	allowed	allowed	allowed
CDRLOLFlagTx*	11h	137	N/A	N/A	N/A	allowed	allowed	allowed	allowed
AdaptiveInputEqFailFlagTx*	11h	138	N/A	allowed	N/A	allowed	allowed	allowed	allowed
OpticalPowerHighAlarmFlagTx*	11h	139	allowed	allowed	allowed	allowed	allowed	allowed	allowed
OpticalPowerLowAlarmFlagTx*	11h	140	N/A	N/A	N/A	allowed[1]	allowed	allowed	allowed
OpticalPowerHighWarningFlagTx*	11h	141	allowed	allowed	allowed	allowed	allowed	allowed	allowed
OpticalPowerLowWarningFlagTx*	11h	142	N/A	N/A	N/A	allowed[1]	allowed	allowed	allowed
LaserBiasHighAlarmFlagTx*	11h	143	allowed	allowed	allowed	allowed[1]	allowed	allowed	allowed
LaserBiasLowAlarmFlagTx*	11h	144	N/A	N/A	N/A	allowed[1]	allowed	allowed	allowed
LaserBiasHighWarningFlagTx*	11h	145	allowed	allowed	allowed	allowed[1]	allowed	allowed	allowed
LaserBiasLowWarningFlagTx*	11h	146	N/A	N/A	N/A	allowed[1]	allowed	allowed	allowed
Rx Related Flags									
LOSFlagRx*	11h	147	allowed	allowed	allowed	allowed	allowed	allowed	allowed
CDRLOLFlagRx*	11h	148	N/A	N/A	N/A	allowed	allowed	allowed	allowed
OpticalPowerHighAlarmFlagRx*	11h	149	allowed	allowed	allowed	allowed	allowed	allowed	allowed
OpticalPowerLowAlarmFlagRx*	11h	150	N/A	N/A	N/A	allowed	allowed	allowed	allowed
OpticalPowerHighWarningFlagRx*	11h	151	allowed	allowed	allowed	allowed	allowed	allowed	allowed
OpticalPowerLowWarningFlagRx*	11h	152	N/A	N/A	N/A	allowed	allowed	allowed	allowed
OutputStatusChangedFlagRx* [2]	11h	153	N/A	N/A	N/A	allowed	allowed	allowed	allowed

Note 1: For instance, Tx output power and Tx bias Flags are not allowed (N/A) in the DPInitialized state for media lanes where the Tx output is squelched or disabled by the host.

Note 2: While the OutputStatusChangedFlagRx is N/A in certain states, the associated OutputStatusRx* status fields are valid in all DPSM states. An OutputStatusChangedFlagTx is deliberately not available.

6.3.4.3 VDM Flagging Conformance per State

Table 6-22 describes the Flag conformance for all Flags related to the (optional) Versatile Diagnostic Monitoring Observables, per Data Path State. See section 7.1 and section 8.14 for information on the optional VDM feature.

In Data Path States where a Flag is indicated as 'Not Allowed', the module shall not set the associated Flag bit while the Data Path is in that state.

All VDM Flags are 'Not Allowed' throughout the Reset and MgmtInit module states. For all other module states, implementers should refer to the Data Path State to determine lane-specific Flag conformance.

[1] An asterisk '*' in a name is a wildcard: All Flags matching the name pattern are referred to.

Note: The host can suppress undesirable Interrupts by setting the corresponding Mask bit at any time after the management interface is initialized.

Table 6-22 VDM Flag Conformance Rules

VDM Observable Type	DPDeactivated	DPInitialized	DPInit	DPDeinit	DPTxTurnOn DPTxTurnOff DPActivated
Laser Age	N/A	allowed	allowed	allowed	allowed
TEC Current	allowed	allowed	allowed	allowed	allowed
Laser Frequency Error	N/A	allowed	allowed	allowed	allowed
Laser Temperature	N/A	allowed	allowed	allowed	allowed
eSNR Media Input	allowed	allowed	allowed	allowed	allowed
eSNR Host Input	N/A	allowed	allowed	N/A	allowed
PAM4 Level Transition Parameter (LTP) Media Input	N/A	allowed	N/A	allowed	allowed
PAM4 Level Transition Parameter (LTP) Host Input	N/A	allowed	allowed	N/A	allowed
Pre-FEC BER Minimum Media Input (data-path)	N/A	allowed	N/A	allowed	allowed
Pre-FEC BER Minimum Host Input	N/A	allowed	N/A	N/A	allowed
Pre-FEC BER Maximum Media Input	N/A	allowed	N/A	allowed	allowed
Pre-FEC BER Maximum Host Input	N/A	allowed	N/A	N/A	allowed
Pre-FEC BER Average Media Input	N/A	allowed	N/A	allowed	allowed
Pre-FEC BER Average Host Input	N/A	allowed	N/A	N/A	allowed
Pre-FEC BER Current Value Media Input	N/A	allowed	N/A	allowed	allowed
Pre-FEC BER Current Value Host Input	N/A	allowed	N/A	N/A	allowed
FERC Minimum Media Input	N/A	allowed	N/A	allowed	allowed
FERC Minimum Host Input	N/A	allowed	N/A	N/A	allowed
FERC Maximum Media Input	N/A	allowed	N/A	allowed	allowed
FERC Maximum Host Input	N/A	allowed	N/A	N/A	allowed
FERC Average Media Input	N/A	allowed	N/A	allowed	allowed
FERC Average Host Input	N/A	allowed	N/A	N/A	allowed
FERC Current Value Media Input	N/A	allowed	N/A	allowed	allowed
FERC Current Value Host Input	N/A	allowed	N/A	N/A	allowed

7 Advanced Management Features

This chapter describes advanced management features and capabilities, which all are optional.

A module advertises the actually supported advanced features in the module management Memory Map.

7.1 Versatile Diagnostics Monitoring (VDM)

Versatile diagnostics monitoring (VDM) is an optional feature, extending performance and diagnostics monitoring capabilities both in terms of new functionality (statistics) and in terms of new observables that can possibly be monitored (when supported by a module).

The module advertises general VDM support in Bit 01h:142.6. When supported in general, VDM Page 2Fh provides more details as well as descriptor arrays for the set of VDM observables actually supported.

VDM functionality is represented in Pages 20h-2Fh as described in detail in section 8.14, where also a summary of these VDM Pages is shown in Table 8-119.

7.1.1 Purpose and Background

Advanced modules come with new types of observables. For all observables, regular performance monitoring functionality is desired: real-time **samples** and **threshold crossing notifications** for warning and alarm thresholds.

For instance, for modules with tunable lasers, implementing a Dense Wavelength Division Multiplexing (DWDM) optical interface, laser wavelength (or frequency) monitoring is important because it allows estimating the health of the laser. When direct measurement of the laser frequency error is not available, laser temperature deviation from target is often used instead. In addition, DWDM modules typically use a thermo-electric cooler (TEC) to control the laser temperature, and the current flowing through the TEC is a module health indicator. Threshold crossing warnings or alarms for such observables are early indications of pending module failure.

For modules using PAM4 signaling several additional observables are useful to determine the health of the module and the line signal quality. These include bit error ratio (BER) and frame error ratio or frame error count, signal-to-noise ratio estimation, and a level transition (decision threshold) measurement that characterizes the received PAM eye.

Modules of all kinds may contain forward error correction (FEC), which allows estimating digital transmission quality and link margins: Counters of corrections performed and counters of uncorrectable frames are used to derive frame, symbol, or bit error counts, and associated error rates (per unit time) or ratios (per received data volume). The short-term fluctuations of the error rates or ratios over time are best observed by **statistics**, keeping track of minimum, maximum, and average values. The statistics interval is under host control.

7.1.2 Technical Overview

VDM first extends the list of observables that can be monitored and alarmed beyond those defined as part of core management functionality in Lower Memory and in Banked Page 11h. The list of additional observables that a module may offer for monitoring is provided in Table 8-122.

VDM also extends the monitoring functionality of modules by adding support for **interval statistics** for basic observables with fast internal sampling rates.

The basic functionality of any VDM monitor is the same as the functionality of the basic monitors:

- Providing a **current value** (a.k.a. **sample**, **real time value**) of the monitored observable
- **Flags** indicating high and low **threshold crossings**, both for warning and alarm threshold levels
- **Masks** to suppress Interrupt generation associated with the threshold crossing Flags
- Threshold crossing level information (advertised Flag setting criteria)

The functional extensions of VDM provide host-controlled **interval statistics** for a subset of observables, i.e.

- Minimum
- Maximum
- Average

Note: Statistics are currently supported for error performance observables only, because a fine sampling time resolution is used, intentionally, to capture fluctuations for these performance metrics. Other fluctuating observables of more general interest are usually sampled or updated at much slower time scales, which enables hosts to build any desired statistics themselves.

Note: It is important to distinguish two types of intervals for observables with statistics support: the sampling interval and the statistics collection interval (sometimes just called a statistics interval).

- *A **sampling interval** or measurement interval is the module internal interval used to estimate or measure one sample of an observable. The durations of the sampling intervals are not specified (**vendor defined**) but typically short (sub-second) because one is interested in performance relevant fluctuations at a short time scale. The series of samples are used to feed the statistics for the observable, such as min/max/average values of the observable samples within a statistics interval.*
- *A **statistic collection interval** is the duration between two "VDM Freeze" requests issued by the host to get the current statistics results and simultaneously restart statistics collection for the next statistics interval. The typical duration of a statistics collection interval is in the order of one or a few seconds. Note that the statistics interval is entirely **host defined** and may vary intentionally or unintentionally. This is a difference compared to a module generated periodic "PM tick" that is known from other systems. Otherwise the VDM Freeze command acts like a host controlled "PM tick".*

See also section 8.14 for more detailed description of the VDM Freeze/Unfreeze mechanism

7.1.3 Technical Details

See also section 8.14 for more detailed descriptions

VDM functionality is represented in the following Banked Pages:

- Pages 20h-23h: Descriptors of supported VDM observable (64 per Page).
- Pages 24h-27h: Real-time samples (current values) of the supported VDM observables (64 per Page).
- Pages 28h-2Bh: Quad Thresholds used to generate TC Alarms or Warnings (16 Quads per Page).
- Page 2Ch: Up to 256 Flag registers, 4 Flags per observable (alarm/warning-high/low).
- Page 2Dh: Up to 256 Mask Bits associated with the Flags.
- Page 2Fh: Advertisement registers and handshake controls.

VDM allows a module to offer up to 256 observables for 8 lanes in Bank 0. Up to 3 additional Banks can provide extensions for modules with more than 8 lanes, where each Bank provides observables for 8 lanes.

VDM observables may be related to the module as a whole, to a Data Path, or they may be lane specific.

VDM observables are identified by a numerical Observable ID.

It is expected that not all possible observables will be supported by all modules or on all lanes. To address this situation, modules advertise which observables are being monitored.

Note: This allows adding new observables without major specification changes.

A module advertises supported observables by a 2-byte descriptor register per observable instance (slots).

Unused slots are indicated by a value 00h in both bytes of the slot configuration register.

Up to 64 threshold value sets of four thresholds (quads) each are provided, and each of the 256 observables is associated (by the module) with one of the threshold value sets from a threshold group.

Note: Some performance related observables may not have four thresholds implemented. For example, BER does not need low alarm or warning thresholds and SNR does not need high alarm or warning thresholds implemented.

To indicate a missing high (low) threshold, the module shall advertise the maximum (minimum) representable threshold value, because such a threshold cannot be crossed outwards.

Note: It is not specified if crossing (preferred) or reaching a threshold causes Flag setting. For full accuracy, it is left for vendors to specify in data sheets which observables have less than all four thresholds implemented.

The observable descriptor registers also indicate the threshold set used for each observable.

Note: The VDM mechanism assumes that several observables may share the same threshold set, e.g. when the same observable is measured on multiple lanes.

The 256 observable slots are split into 4 groups. Each group consists of one Page of descriptors, one Page of real-time values (samples), one Page of threshold sets and ¼ Page each of Flags and Masks.

7.1.4 PAM4 Observables

Figure 7-4 below shows a general block diagram of the optical ingress path of a module showing the location where SNR and level transition (decision threshold) observables are measured.

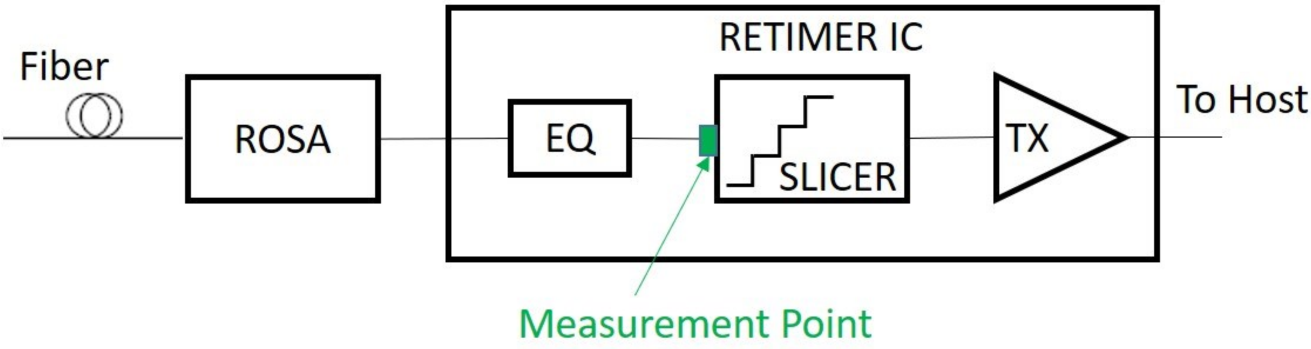


Figure 7-1 Optical ingress path of Module

Figure 7-2 shows a PAM4 amplitude histogram collected by sampling the received signal in the center of the eye, just before deciding the data in the slicer. The VDM SNR and level transition (decision threshold) observables are determined from a PAM4 amplitude histogram.

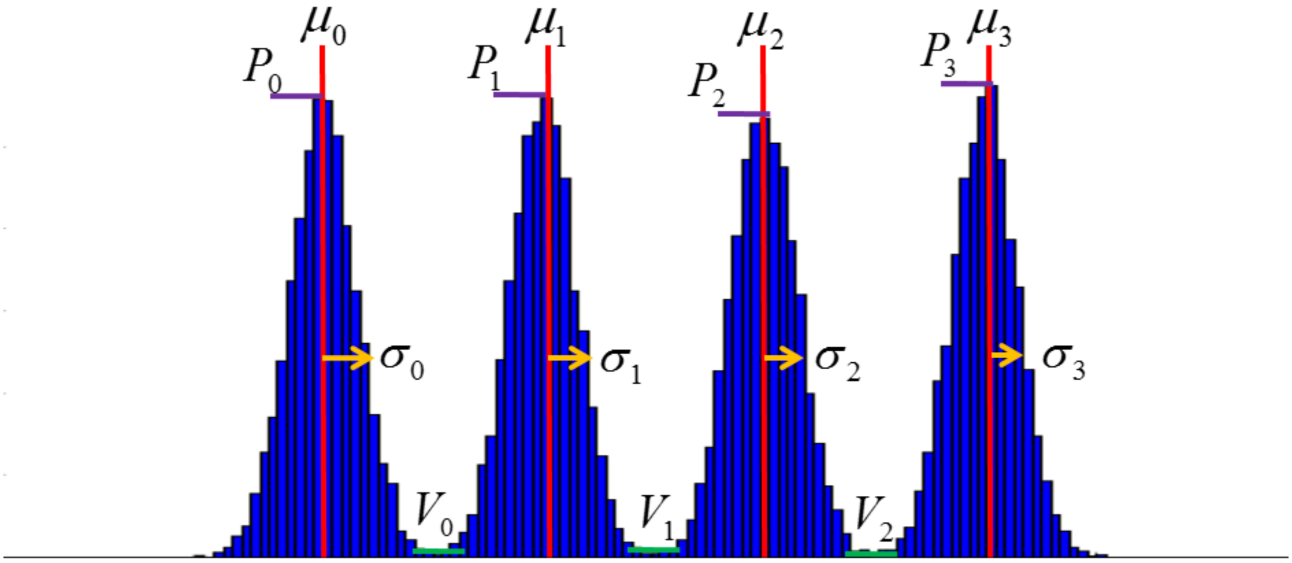


Figure 7-2 PAM4 amplitude histogram

The histogram x-axis is in bins and the y-axis is in number of bin hits. The number of bins and the hit count magnitude is vendor specific. The histogram is taken at the point in where data is converted from analog to digital. The PAM4 slicer determines the best decision thresholds for slicing the data. The peaks are the bins with the largest number of hits between any two valleys (or below valley 1/above valley 3 for the first and last peaks). The valley location corresponds to the decision thresholds determined by the slicer.

The calculations for the reported eye parameters SNR and LTP are:

SNR = 10* $\log_{10}(\min\{SNR_0, SNR_1, SNR_2\})$ where $SNR_i = (\mu_{i+1} - \mu_i)/(\sigma_{i+1} + \sigma_i)$, expressed in 1/256 dB units

LTP = 10* $\log_{10}(\min\{LTP_0, LTP_1, LTP_2\})$ where $LTP_i = (P_{i+1} + P_i)/(2V_i)$, expressed in 1/256 dB units

Where,

μ_i: level of ith peak, optionally averaged over neighboring bins
σ_i: std dev of ith peak, optionally averaged over neighboring bins

P_i: height of ith peak, optionally averaged over neighboring bins

V_i: height of ith valley, optionally averaged over neighboring bins

Note: Both SNR and LTP measure signal-to-noise but LTP is more sensitive to a noise floor.

For the vendor specified wavelength, the accuracy of the reported SNR and LTP observables shall be better than +/-3 dB over specified temperature and voltage.

7.1.4.1 SNR (eSNR)

This observable represents the electrical signal-to-noise ratio on an ingress optical lane and is defined as the minimum of the three individual eye SNR values, where the SNRi for each of the three eyes is defined as the ratio of the difference of the mean voltage between neighboring levels divided by the sum of the standard deviations of the two neighboring levels.

SNR is encoded as U16 in units of 1/256 dB. *For example, a value of 1380h corresponds to an SNR of 19.5 dB.*

7.1.4.2 PAM Level Transition Parameter (LTP)

This observable measures the electrical level slicer noise and is defined as the minimum of three individual PAM level LTP values, where LTP_i for each PAM level is defined as the average of the peak histogram height of neighboring PAM levels divided by the minimum histogram height between them.

LTP is encoded as U16 in units of 1/256 dB. When the minimum histogram height between PAM levels is zero the LTP value is encoded as FFFFh. Finite LTP values greater than 255.996 dB are encoded as FFFEh.

7.1.5 Error Performance Statistics (FEC)

Two detection error metrics may be supported as VDM observables:

- Frame Error Count (**FERC**): This observable represents the number of uncorrectable FEC frames, measured as RS(544,514) equivalent frames, in a short interval. This is a post-FEC decoder error metric.
- Pre-FEC Bit Error Ratio (**BER**): This is an intensive (i.e. fluctuating) observable representing the total number of bits that were corrected by the FEC during an interval divided by the total number of bits received in that interval. This is an estimate of the pre-FEC bit error ratio or bit error probability.

Frame errors are reported in units of RS(544,514) [1] equivalent frames: When the FEC actually used is different, then the measured frame error count is scaled proportional to the binary FEC frame size divided by 5440.

Note: For example, if the FEC frame size is 10% larger than the RS(544,514) FEC frame, then the reported equivalent frame error count will be 10% higher than the actually measured true frame error count.

Note: This is to allow comparing frame error counts regardless of the FEC encoding actually employed.

Note that different FEC schemes have different maximum pre-FEC BER requirements for achieving the same maximum post-FEC BER target.

Current Sample

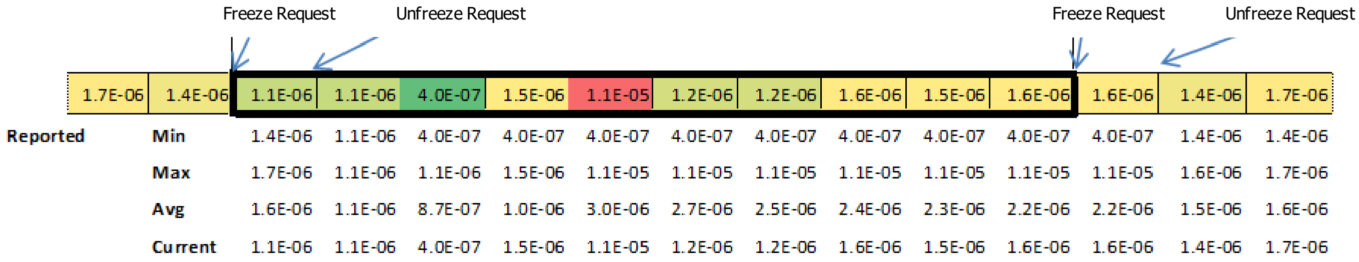
If supported, the host can always read a recent BER or FERC sample ("real-time-value" or "current value").

Statistics (Minimum, Maximum, Average)

A module supporting statistics takes short-term measurements (samples) of BER or FERC over a vendor-specific fine measurement time interval (e.g. 1 ms) and then updates internal statistics variables (min, max, average), thus providing cumulative statistics in a currently running interval until the host eventually terminates the current statistics interval by requesting to freeze the statistics reporting registers and to internally restart the statistics collection for a new interval at the same time. As long as no freeze is active, the host may read out the "live" statistics at a slower pace at any time during the currently running statistics collection interval.

This mechanism of closing a statistics interval and reading stable results is described in section 8.14.

Example



		1.7E-06	1.4E-06	1.1E-06	1.1E-06	4.0E-07	1.5E-06	1.1E-05	1.2E-06	1.2E-06	1.6E-06	1.5E-06	1.6E-06	1.6E-06	1.4E-06	1.7E-06
Reported	Min			1.4E-06	1.1E-06	4.0E-07	4.0E-07	4.0E-07	4.0E-07	4.0E-07	4.0E-07	4.0E-07	4.0E-07	4.0E-07	1.4E-06	1.4E-06
	Max			1.7E-06	1.1E-06	1.1E-06	1.5E-06	1.1E-05	1.1E-05	1.1E-05	1.1E-05	1.1E-05	1.1E-05	1.1E-05	1.6E-06	1.7E-06
	Avg			1.6E-06	1.1E-06	8.7E-07	1.0E-06	3.0E-06	2.7E-06	2.5E-06	2.4E-06	2.3E-06	2.2E-06	2.2E-06	1.5E-06	1.6E-06
	Current			1.1E-06	1.1E-06	4.0E-07	1.5E-06	1.1E-05	1.2E-06	1.2E-06	1.6E-06	1.5E-06	1.6E-06	1.6E-06	1.4E-06	1.7E-06

Figure 7-3 BER short term measurements and interval statistics (example)

Figure 7-3 shows an *example* of a time series of fine interval measurements punctuated by statistics freeze events that demark the host statistics monitoring interval. While no freeze request is pending, the host can read calculated statistics values. The selection of which statistics are available depends on the observable type.

The module calculates a BER/FERC sample for each fine interval, indicated with light borders. The host monitoring (statistics collection) interval is indicated with dark borders. The top row in the diagram shows the samples measured internally at each fine interval. The bottom four rows show the reported data by the module.

The Min, Max, and Avg rows show that after an Unfreeze request, the module continuously updates the statistics reporting values. Then, upon the next Freeze request, the data stops changing until the following Unfreeze request, whereupon it resumes reporting the updating minimum value of the new host monitoring interval. Current sample values are always updated, independent of statistics freezing.

Note: Threshold crossing detection is available also for the BER and FERC observables. Modules should use low thresholds of 0. To alarm post-FEC errors the high threshold for FERC should also be 0 unless other error correcting schemes are present.

[1] see IEEE 802.3 Clause 91.5

7.1.6 Tunable Laser Monitors

7.1.6.1 TEC Current

If supported, this monitor observes the amount of current flowing to the TC of a cooled laser.

A single observable for a whole-module TC is available in the Auxiliary monitoring capabilities. Modules should use the VDM system to provide for a per-lane TEC controller.

See Table 8-122 for encoding, units and scaling information for the monitored observable.

7.1.6.2 Laser Frequency Deviation

If supported, this monitor observes the difference (in frequency units) between the target center frequency and the actual current center frequency. It is a similar measurement to the Laser Temperature except expressed as a frequency difference instead of a temperature difference, and vendors may support one or the other measurement, or both.

See Table 8-122 for encoding, units and scaling information

7.1.6.3 Laser Temperature

If supported, this monitor observes the laser temperature difference between the target laser temperature for a cooled laser, and the actual current temperature.

It is a similar measurement to the Frequency Error except expressed as a temperature difference instead of a frequency difference, and vendors may support one or the other measurement, or both.

A single observable for a whole-module laser frequency is available in the Auxiliary monitoring capabilities. Modules should use the VDM system to provide for a per-lane laser temperature.

See Table 8-122 for encoding, units and scaling information for the monitored observable.

7.2 Command Data Block (CDB) Message Communication

7.2.1 Feature Overview

The optional Command Data Block (CDB) feature allows host-originated **functional interactions** between host and module beyond the basic CMIS core interactions, which allow only to READ or WRITE registers.

CDB is best understood as an asymmetric **message exchange mechanism** for implementing a **command and reply** type of interaction, not much different from invoking a function and waiting for a result being returned. There are **message sending** facilities for the host (initiator) and **message reply** facilities for the module (target) which include status information and completion notification using a Flag.

Two command **processing variants** exist. In both cases the host may send a command and then go off doing other things, coming back later when the reply has returned a result, as determined either by polling or by receiving an Interrupt. With **foreground processing**, the module rejects any register ACCESS during command processing, while with **background processing** register ACCESS is possible.

7.2.2 CDB Technical Basics

The module advertises support for CDB in several fields 01h:163-166 as described below and in section 8.4.11.

The host sends a CDB Command (**CMD**) message which is identified by a **CMD ID** and the module responds with a CDB Reply (**REPLY**) message without changing the CMD ID.

The CMD and REPLY message exchange is performed by reading and writing **message header** fields located on **CDB Message Page** 9Fh (section 8.15) and **message body** data (also called **payload**) either locally on Page 9Fh or on the **Extended Payload Pages** A0h-AFh (section 8.16).

A **CDB Status Register** (00h:37 or 00h:38) informs the host about the status of a CDB facility and the status of the most recent message exchange, as described in section 8.2.8.

A **CDB Command Completion Flag** (00h:8.6 or 00h:8.7) informs the host when the REPLY is available, as described in section 8.2.4.

The **CDB Message Page** 9Fh contains the CMD **message header** of the command to be executed and provides an on-page area for a **message body** of up to 120 bytes, called the **local payload** (**LPL**) area.

The **CDB EPL Pages** A0h-AFh provide an area of up to 2048 bytes of **extended payload** (**EPL**) for messages with a large message body. Support for EPL Pages is advertised as described in section 8.4.11.

The same LPL and EPL areas are used for both CMD parameters and REPLY data, depending on command specifications in chapter 9. While the message REPLY does not overwrite the CMD header, the REPLY message body may overwrite the CMD message body. This implies that at most one message exchange is performed at any one time in a CDB instance (i.e. queuing of messages is not supported).

7.2.3 CDB Message Sending and Reply Receiving (Host)

The host triggers sending a message either when writing to a particular Byte address (9Fh:129) or when a multi-byte write to Page 9Fh that includes Byte 9Fh:129 finishes with the STOP condition (see Table 8-48). The method to be used is advertised by the module.

When the module performs CDB command processing in the background, the host can poll the status of CDB command execution in byte **CdbStatus1** (00h:37) or **CdbStatus2** (00h:38). See section 8.2.8 for details.

When the processing of a CDB command has been completed, the module notifies the host using a **CdbCommandComplete** Flag (with associated maskable interrupt) in Byte 00h:8 (see Table 8-8).

A module not responding to a supported CDB command is considered defective and the host may need to initiate a hardware Reset or issue CMD 0004h (Abort Processing) to recover the module.

7.2.4 CDB Message Receiving and Reply Sending (Module)

If a module receives a CDB command, it may perform the following steps, typically:

1. Capture CMD header and body (into a buffer or queue)
2. Recompute the integrity check code (CdbChkCode) for verification
3. Validate the CMD ID and the command parameters
4. Check if the requested command can be executed in the current state
5. Schedule the command for execution and update the command status register
6. Execute the command to completion

7. Determine if the command was executed successfully or if it has failed
8. Compose the REPLY header and body from execution results
9. Update the command status register with command result codes (indicating success or failure)
10. Send the REPLY by setting the command completion Flag, which causes the Interrupt signal to be asserted (unless the command completion Mask is set).

The steps described above may vary by module vendor but the overall behavior of the CDB message exchange is identical: a CDB command sent will result in a CDB reply received.

7.2.5 CDB Support Levels and Messaging Details

Modules may provide different levels of CDB support, e.g. based on processing capabilities.

Support level options include:

- **CDB foreground** processing (ACCESS hold-off until command completion)
- **CDB background** processing (ACCESS during command processing)
- Number of CDB instances supported for parallel CDB command execution in background mode

Note: Modules supporting CDB background processing typically require a more powerful CPU and more memory resources, while modules supporting CDB foreground processing may be more cost or power effective.

This CMIS revision specifies support of up to two CDB instances (with foreground or background processing).

Each CDB instance uses its own Bank. CDB instance 1 resides in Bank 0 and CDB instance 2 resides in Bank 1.

All supported CDB instances support the same set of advertised CDB commands.

When background processing of two CDB instances is supported, up to two CDB commands may be executed in parallel, while register access to the memory mapped registers is also possible at the same time.

When foreground processing is used, even when two CDB instances are supported, at most one CDB command may be executed at any one time and ACCESS to registers is impossible until the CDB command completes its execution.

On completion of a command the module updates the relevant CdbStatus register (00h:37 or 00h:38 for Bank 0 or 1) and then sets the relevant command completion Flag in Byte 00h:8:7-6.

Note: Setting a Flag causes an Interrupt unless the Flag is masked by its associated Mask in Byte 00h:31:7-6.

Command completion with status update and Flag setting occurs in both background and foreground mode.

7.2.5.1 Foreground Mode CDB Messaging

A module may support CDB command execution in foreground mode only. This is indicated by Bit 01h:163.5 cleared (see Table 8-48).

In foreground mode the module rejects any register ACCESS until a currently executing CDB command execution has completed.

Note: READs of the CdbStatus registers 00h:37 or 00h:38 (see Table 8-12) will also be rejected by the module.

Once a command has eventually completed execution, the module sets the relevant CdbStatus register 00h:37 or 00h:38 (for Bank 0 and 1), then sets the relevant command completion Flag 00h:8:7 or 00h:8.6, respectively, and resumes accepting register ACCESS by the host (e.g. to read the reply).

Note: While a foreground mode CDB command execution is in progress, a host may use either register access with retry or Acknowledge Polling (see section 0).

Note: In the error event that the module does not stop to NACK, the host might need to eventually reset the module via the hardware Reset signal, or toggle the power supply to the module.

7.2.5.2 Background Mode CDB Messaging

One CDB Instances

A module advertising [01h:163.7-5] = 011b implements one CDB instance in background mode.

In background mode the module will generally allow register ACCESS while CDB commands are being processed: it will block ACCESS only for a short time until the command has been captured. Register access is possible as soon as the command is captured.

When only one CDB is supported, the module responds only to CDB commands in Bank 0. Reading the CDB status Bit 00h:37.7 will return 1b (CdbIsBusy=1).

The host shall not use the same CDB instance, until the current CDB command execution is completed or aborted.

Note: If the host attempts to send another CDB command while the CDB instance is busy (indicated by the CdbIsBusy status bit) the command may be ignored. Under race conditions, the command may also be processed if CdbIsBusy is cleared in background during the MCI transaction time. This makes the behavior of the CDB command or response unpredictable.

The only CDB command that is allowed to be sent while a CDB instance is busy (CdbIsBusy), is CMD 0004h (CDB Abort Processing), which is designed to abort a command that has not completed within expected command execution time and to clear the CdbIsBusy status.

The module indicates command completion by setting Flag 00h:8.6 (CdbCmdCompleteFlag1). An Interrupt request will be asserted, unless masked by the associated Mask bit 00h:31.6 (M-Cdb1CommandComplete).

Note: Modules with one CDB instance and background processing capability may e.g. support firmware update using CDB while the module is in service. Depending on host software implementation, the module may also support running Performance Monitors (PM) at the same time.

Two CDB Instances

A module may advertise with [01h:163.7-5] = 101b that it supports two CDB instances in background mode.

A module supporting two CDB instances in background mode can run two CDB commands simultaneously.

Otherwise, the rules described in the previous section apply to each CDB instance, individually.

Note: This type of module allows the host to have separate threads or tasks for dedicated purposes; for example, having one thread specifically for firmware update and another thread for regular performance monitoring, diagnostics, and other features.

Note: When using multiple threads, the host must implement proper interlocks for Bank and Page registers and MCI writes/reads. This is one of the main reasons that the status registers 37 and 38 have been located in Lower Memory where these registers are always available.

7.3 CDB Message Communication Based Features

7.3.1 Firmware Management using CDB

This section describes concepts and procedures related to firmware management, especially firmware update, using the set of CDB commands defined in section 9.7.

Firmware update based on CDB messaging is an advertised optional feature.

7.3.1.1 Firmware Management Concepts

The **generic** CDB based firmware update facilities allow hosts to update the module firmware completely or partially in a **vendor supported** depth and granularity.

Firmware

Implementation dependent examples of firmware items that might be updated include main controller firmware, the parts needed to support another Memory Map revision, modified administrative or operational data resources, microcode in the module, code for internal devices such as converters or DSPs, or a bundle of such items, in any granularity as defined and supported by the module vendor.

The definition of what updateable items exist and if they may be included in a complete or partial firmware update is module implementation dependent and outside the scope of the specification. This is possible because the CDB based firmware update mechanism is not dependent on the content of a firmware update.

Note: Host suppliers are encouraged to coordinate with module suppliers to ensure firmware update packages meet their requirements.

Firmware Revisions

In base CMIS firmware revisions are identified by a **major** version number and a **minor** version number. With CDB firmware management, a finer distinction using a **build number** and an extra **string** is possible.

Note: As the aggregate firmware of a module may consist of multiple components, it is strongly recommended that the CMIS firmware revision numbers refer to that complete aggregate (bundle) of all firmware components, not to a particular firmware component.

Note: It is also strongly recommended that revision numbers, at least when complemented with build number and extra string, uniquely specify exactly one aggregate firmware configuration.

Note: When piecewise or incremental firmware updates are used or supported by a vendor, it is strongly recommended that transiently invalid or unsupported combinations of firmware components in the module are considered invalid and not declared runnable until eventually a supported and well defined aggregate firmware configuration status is achieved, which is identified by a unique aggregate version identification.

Firmware Update

Firmware update includes operations on module-internal firmware stores (called firmware banks), and the function of passing control to firmware in a firmware store.

- Operations on internal firmware stores include **query**, **download**, **upload**, and **copy** of stored firmware (to, or from, or between firmware stores), which can serve the purposes of **upgrade, repair,** and **downgrade**.
- The operation to eventually pass control to downloaded firmware is called **activation** and occurs on demand either immediately (**run**) or on the next restart (**commit**)

Note: If a module contains firmware but does not support firmware update, the module may still allow to query firmware information using CMD 0100h (Get Firmware Info). This is especially recommended when vendor specific firmware update mechanisms exist.

Firmware Storage and Administration

Modules may have up to two host updateable images stored in up to two storage banks, called A and B; and modules may also have an internal factory image.

Each Bank (or its content) is classified by its current content as follows:

- The **Administrative Status** indicates if the firmware in the bank is run upon resets (**committed**) or not (**uncommitted**). Only one bank is committed at a time. The administrative status can be changed with CMD 010Ah (Commit Firmware Image).

- The **Operational Status** of a bank indicates if the image in the bank is currently **running** or **not running**. Only one image can be running at a time. In special conditions, the module may indicate that it is running a factory image which is neither A nor B. The Operational Status may be changed with CMD 0109h (Run Firmware Image), or with a module reset (triggered by software or hardware signal), or with power cycling. Upon a reset or power cycle the module will run the "committed" image as indicated by the Administrative Status.
- The **Bank Validity Status** indicates if the firmware in the bank is runnable (**valid**) or not (**invalid**). A valid firmware is runnable and persistently stored completely undamaged. The Bank validity of images A and B may be affected by the CDB firmware **update** commands with IDs 0101h (Start), 0102h (Abort), 0103h and 0104h (Write), 0107h (Complete) and 0108h (Copy). Only a proper sequence will be able to restore the Bank Validity Status from "Invalid" to "Valid".

Note: Firmware Update using a single storage bank is possible as well. In this case, there is no option to fall back to a previous firmware during trial phase for a downloaded but not yet committed firmware image.

Firmware Update Media

For any specific firmware update purpose, the module vendor can provide a specific **firmware download file** which includes vendor specific instructions for updating firmware in a module, completely or partially.

A firmware download file consists of a vendor specific and host transparent **header** section and a **firmware** section containing a representation of the actual executable firmware image bundle. This representation may be coded (ASCII) or uncoded (binary).

If firmware readback is supported, the module vendor may also provide a **firmware readback file** which serves to control reading back the most recently downloaded firmware from the module. A delivered firmware readback file consists of a header and of dummy data with the expected length of the firmware to be read back.

Note: The purpose of firmware readback is vendor or host specific. CMIS makes no assumptions here.

The vendor specific **header** section in both types of firmware files is initially downloaded to and interpreted by the module. Since size, content, and interpretation of these headers are vendor specific, the module only advertises the number of header bytes at the beginning of a firmware file in the field **StartCmdPayloadSize** (9Fh:138) returned by CMD 0041h (Firmware Management Features).

The file format of a firmware file may be either binary or ASCII (for example Intel hex file format). If the file is ASCII encoded, the host first decodes the file contents into binary format before downloading or after uploading.

Note: The method for a host to determine the firmware file format are not specified in CMIS.

7.3.1.2 Reference Firmware Download Procedure using CDB

The host begins by reading module capabilities using CMD 0041h (Firmware Management Features).

The host then reads (and possibly converts to binary format) the vendor specific and vendor provided firmware download file into a contiguous addressable byte array defined as the binary **download image**.

To start the firmware download, the host sends the download image header consisting of the first **StartCmdPayloadSize** bytes from the download image using CMD 0101h (Start Firmware Download). This header instructs the module in a vendor specific way about the full or partial download content to be expected.

Before the module updates an image bank in a download procedure the module ensures that the bank is marked as empty or corrupt until the download has eventually finished successfully.

When CMD 0101h was successful, the module is ready to accept data from the host using the advertised method, either CMD 0103h (Write Firmware Block LPL) or CMD 0104h (Write Firmware Block EPL). These command differ only in the allowed size of a download image block, which is advertised for the EPL variant.

The host zero-initializes a variable containing the address of the next data block to be sent or received in subsequent Write Firmware Block or Read Firmware Block commands, respectively.

In a loop the host then reads subsequent bytes of the download image in blocks not exceeding the allowed block site and sends it using CMD 0103h or 0104h.

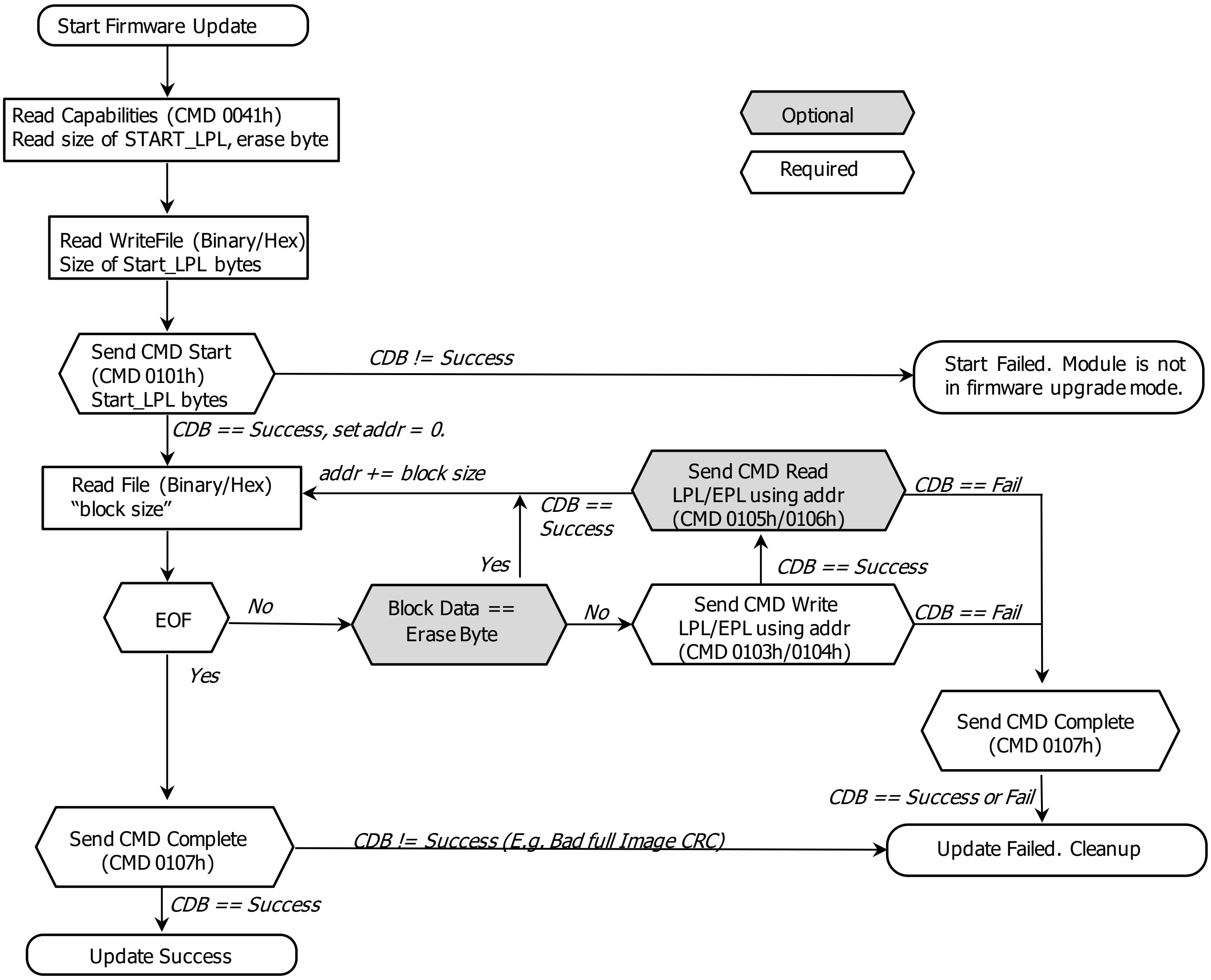

Start Firmware Update
Read Capabilities (CMD 0041h)
Read size of START_LPL, erase byte
Optional
Required
Read WriteFile (Binary/Hex)
Size of Start_LPL bytes
Send CMD Start
(CMD 0101h)
Start_LPL bytes
CDB != Success
Start Failed. Module is not in firmware upgrade mode.
CDB == Success, set addr = 0.
Read File (Binary/Hex)
"block size"
addr += block size
CDB == Success
Send CMD Read LPL/EPL using addr (CMD 0105h/0106h)
CDB == Fail
Yes
CDB == Success
EOF
No
Block Data == Erase Byte
No
Send CMD Write LPL/EPL using addr (CMD 0103h/0104h)
CDB == Fail
Yes
Send CMD Complete (CMD 0107h)
CDB == Success or Fail
Send CMD Complete (CMD 0107h)
CDB != Success (E.g. Bad full Image CRC)
Update Failed. Cleanup
CDB == Success
Update Success

Figure 7-4 Firmware Update using CDB

The flow diagram in Figure 7-4 shows two optional blocks:

The first optional block describes that the host may optionally skip sending such blocks of data that consist entirely of bytes marking an erased state (this special byte value is advertised). Skipping the download of such blocks can speed up the firmware download process.

The second optional block describes that the host may optionally read back the most recently sent block. This slows down the firmware download process and is only recommended when needed.

A successful firmware update process ends when the host has downloaded all data from the download image and no unrecovered errors have occurred in the associated message exchanges. The host then completes the firmware update by sending the CMD 0107h (Complete Firmware Download). Receiving this command will trigger the module to perform any appropriate validation and stop the current firmware download process. If the validation fails, this command returns a failure code in the **CdbStatus** register.

Note: A defensive host should not assume that all modules will perform a full validation.

When the firmware download was unsuccessful, e.g. due to commands returning failure, or when it needs to be aborted for any reason, the host must also terminate the ongoing firmware update process either by sending a CMD 0107h (returning failure but terminating the download process) or, if supported, by using CMD 0102h (Abort Firmware Download). A new firmware update process may then be started to retry updating the firmware in the module.

7.3.1.3 Reference Firmware Upload Procedure using CDB

Reading back a complete or partial image, as controlled by the header in the readback firmware file, may be achieved using the flow chart shown in Figure 7-5.

Note: The upload retrieves data from the most recently updated image store.

Note: Modules may not support reading back firmware from the module. If readback is supported, a host may generate a file from the readback data such that the data of the file matches that of the download file used. At least one bit will be different as the headers of the readback and download files are different.

Readback is controlled by the header portion of a vendor-supplied **firmware readback file**, which serves two purposes: it contains a vendor specific header that controls which parts of the overall firmware of the module will be uploaded and its size defines the size of the expected readback data.

The readback file header has the same number of (StartCmdPayloadSize) bytes like the downloaded file. Within the vendor-specific header bytes, a field must indicate to the module that the operation is a readback, in which case CMD 0101h (Start Firmware Download) does not perform an image erasure.

To start the readback, the host reads the first number of StartCmdPayloadSize bytes from the binary readback image and sends these bytes using CMD 0101h (Start Download). The content of the readback image after the first "Start Command Payload Size" bytes is irrelevant and will not be sent to the module.

The host zero-initializes a variable containing the address of the next data block to be received in subsequent Write Firmware Block or Read Firmware Block commands, respectively. The number of bytes to readback as shown in Figure 7-5 is the size of the binary readback image (excluding the header).

When CMD 0101h was successful, the module is in a state ready to return data to the host using the advertised method, CMD 0105h (Read Firmware Block LPL) or CMD 0106h (Read Firmware Block EPL). These commands differ mainly in the allowed size of an upload image block, which is advertised for the EPL variant

After completion of the readback, but also on readback errors, the host sends CMD 0107h (Complete Firmware Download). In the reply, the module may return either nothing or an appropriate error code in subsequent Read LPL or Read EPL commands.

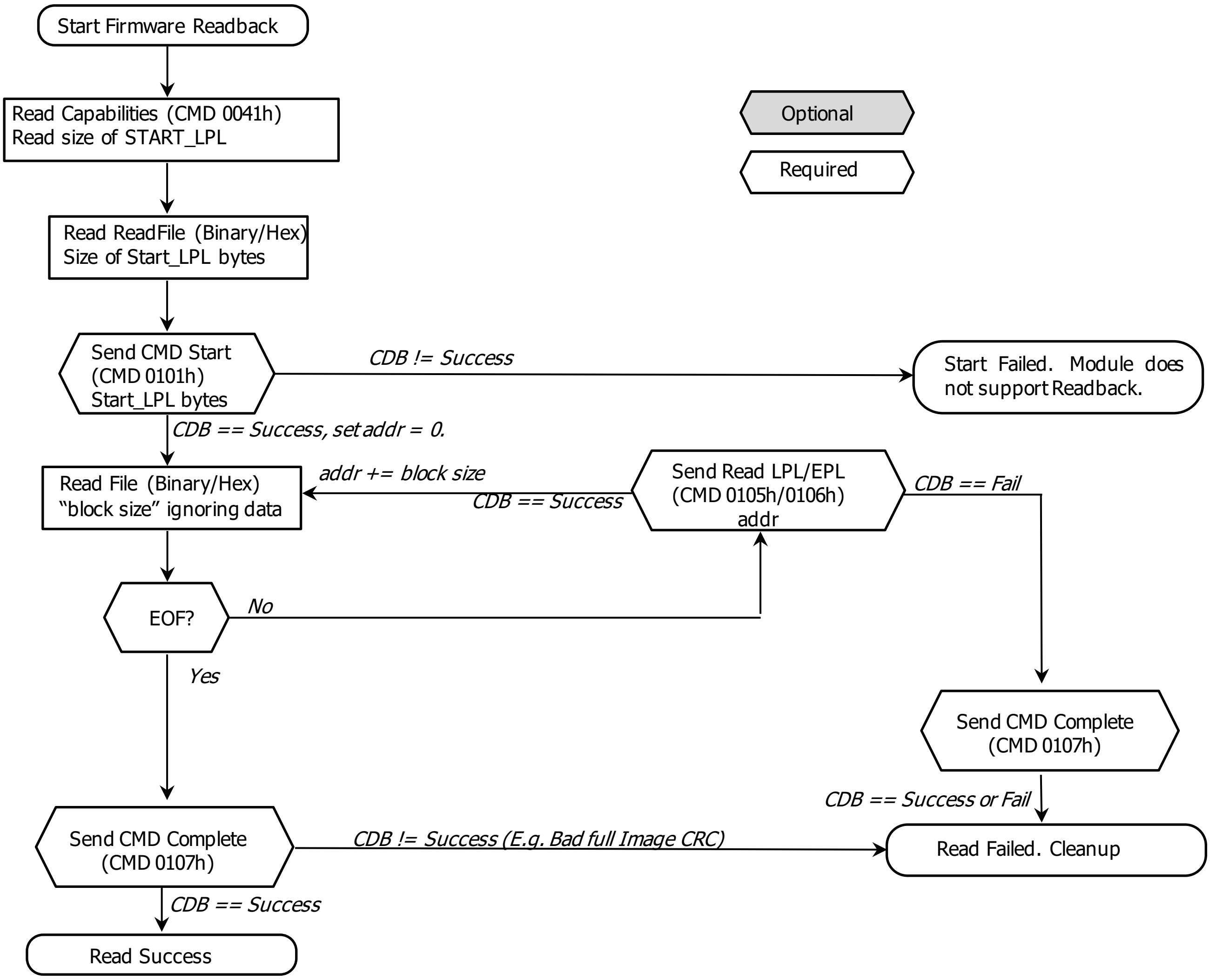


Figure 7-5 Firmware Upload using CDB

7.3.1.4 Firmware Administration and Status

This section describes the **FirmwareStatus** Byte 9Fh:136 returned by CMD 0100h (Get Firmware Info). This byte represents status information about the firmware images stored in the module. See section 9.7.1.

The Status of (the content of) an image storage bank is encoded in the bits of a nibble (half byte)

Table 7-1 Status of an Image Storage Bank

Nibble Bit	Status Aspect	Encoding
0	Operational Status	1: running 0: not running
1	Administrative Status	1: committed 0: uncommitted
2	Validity Status	1: invalid (not runnable) 0: valid (runnable)
3	-	0: reserved

Table 7-2 and Table 7-4 below show a non-exhaustive list of common FirmwareStatus codes. Table 7-2 shows typical codes returned by modules that supports two image banks A and B. The presence of a bootloader or Factory image is not important nor directly indicated here. In normal operating situations, there will not be a way for the bootloader to continue running.

Table 7-2 Typical FirmwareStatus Codes of Modules supporting image A and B.

Firmware Status BA	Firmware Running	Committed Firmware	Bank Validity Status	Description
03h	A	A	A,B	Normal Operation. A running and A committed. Image B valid, can be activated by Run B command.
43h	A	A	A,-	Normal Operation. A running and A committed. Image B update is in progress, has failed or has been aborted. There is no valid image in Bank B.
12h	B	A	A,B	Trial run of image in Bank B.
16h	B	A	-,B	Errored Mode (Bank A found invalid). Image in Bank A is committed but invalid. The module is running the valid image in Bank B.
30h	B	B	A,B	Normal Operation. B running and B committed. Image A valid, can be activated by Run A command.
34h	B	B	-,B	Normal Operation. B running and B committed. Image A update in progress, failed or aborted. There is no valid image in Bank A.
21h	A	B	A,B	Trial run of image in Bank A.
61h	A	B	A,-	Errored Mode (Bank B found invalid). Image in Bank B is committed but invalid. The module is running the valid image in Bank A.

Table 7-3 Typical FirmwareStatus Codes of Modules supporting image A

Firmware Status BA	Running	Committed	Bank Validity Status	Description
44h	-	-	-,-	Factory image or boot loader running, no valid image available
40h	-	-	A,-	Factory image or boot loader running, valid uncommitted image available
42h	-	A	A,-	Factory image or boot loader running, valid committed image available
41h	A	-	A,-	Running an uncommitted image
43h	A	A	A,-	Running a committed image

Table 7-4 shows FirmwareStatus codes that may be returned by a module in special situations, when neither image A nor B is running and neither image A nor B is committed. However, the firmware is still responding correctly to this CDB command. In this situation the host should interpret the running firmware as an internal factory image that cannot be updated via the CDB command set. The vendor may have other vendor specific methods to update the firmware and should be contacted for additional details if these codes are returned.

In some situations, these may be error conditions, for example, if the module supports image A and B as well as a backup factory image, these are error codes that may be returned by the module if an unexpected event has occurred that caused the factory image to be running.

Table 7-4 Special FirmwareStatus Codes

Firmware Status BA	Running	Committed	Bank Validity Status	Description
00h	Factory (none of A,B)	Factory (none of A,B)	A,B	This may be returned by a module that does not support firmware update and only has the factory image. A module that supports image A and B may return these codes when the module's boot control block that defines which image Bank to boot is damaged or when a default factory image is running.
40h			A,-	A module that supports image A, B, or A and B may return these codes when something has damaged the modules boot control block defining the image Bank to boot, or a default factory image is running.
04h			-,B	
44h			-,-	

7.3.2 Performance Monitoring (PM) using CDB

The optional CDB implementation of performance monitoring supports **statistics** such as minimum, average, maximum and current values for a subset of **monitored observables** (see Table 9-27 in section 9.8).

A host can query the list of supported CDB performance monitoring features using CMD 0042h and CMD 0201h (see sections 9.4.3 and 9.8.2).

CDB based performance monitoring is based on host polling and allows the host to read a large number of observables in a single CDB message transaction.

Alarm indications are not supported by the polling based CDB mechanism. If autonomous alarm reporting via Flags is required, Versatile Diagnostics Monitoring (VDM) implemented in Pages 20h-2Fh may be used.

CDB and VDM may both be supported and then used simultaneously. Both CDB and VDM can collect statistics (such as minimum, average, and maximum) for an overlapping subset of observables.

The CDB and VDM statistics may be collected either independently (**independent statistics mode**) or synchronously (**linked statistics mode**). The statistics collection mode defaults to independent mode.

A CDB command (CMD 0200h) is available for selecting the statistics collection mode at runtime, e.g. to switch from independent tracking to synchronous tracking.

Whenever the module restarts (including restart due to a Run Image CDB command), the performance monitoring mode reverts to independent tracking of all supported and advertised observables.

All statistics results are cleared and reset with a mode change.

In **independent statistics mode** statistics read using CDB and statistics read using VDM Pages 20h-2Fh will generally differ because start time and duration of the data collection periods are not synchronized.

Clearing and resetting of statistics result registers in CDB and VDM Page 2Fh will be completely independent, and the module maintains two separate internal data sets for statistics collecting (minimum, average and maximum values).

In **linked statistics mode**, the module maintains one common minimum, average, and maximum value for each observable that supports statistics and then generally returns the same information via CDB or VDM.

Furthermore, in a linked mode a statistics interval result will be cleared whether it is cleared by a CDB command or by writing to a register in VDM Page 2Fh.

For both performance monitoring methods, VDM or CDB, the time at which the module originally starts collecting statistics and the method or internal time granularity at which these data are processed remains undefined.

It is assumed that a host will initially setup Data Paths and later, when the operational conditions are satisfactory for the data of interest, read and discard the first set of statistics results, before actually (re-)starting statistics collection. This is recommended because the first statistics results will likely be abnormal due to initialization transients in the system or non-operational conditions.

7.4 Module Boot Record (MBR)

This section is reserved for a future feature extension.

7.5 Tunable Lasers, Channel Selection and Center Frequency Setting

7.5.1 Concepts and Background

An optical carrier or optical channel is usually characterized by its optical center frequency and the optical channel bandwidth available to carry a modulated signal.

Note: Instead of specifying the optical frequency, a (signed) channel number on a given channel grid is often used, representing the grid offset against a channel number 0, which is defined as the grid's anchor or origin. Instead of specifying the channel bandwidth, the frequency grid spacing between optical carriers on a grid is often used. See below for more information.

For management purposes an optical channel is represented in CMIS by a Media Lane together with associated Media Lane attributes (registers). CWDM module transmit over multiple channels and therefore have multiple Media Lanes. DWDM modules often use only a single optical channel and have only a single Media Lane.

From a management viewpoint, optical channels may be either **pre-defined** by an interface standard (often in single-channel and CWDM applications) or **host-programmable** (often in DWDM applications).

For **host-programmable** channels, conceptually there are two provisioning schemes:

- indirectly **selecting** a channel from a predefined set of channels (in **grid-based** applications), where a channel and its associated center frequency is selected by a channel number
- directly **setting** a center frequency arbitrarily within an overall supported band (in **grid-less** applications), i.e. without the help of a channel number

Note: In both cases, the module must ensure, for operation, that the relevant lasers are tuned to the center frequencies of the provisioned channels. When a tunable module wants to implement an Application with pre-defined channels, it will perform the tuning based on the selected Application, not based on the channel programming registers.

A particular **channel grid** is defined by four attributes

- a grid spacing bandwidth (the frequency difference between neighbors)
- a channel numbering scheme
- a numbering **origin** frequency (usually 193.1THz)
- a **range of channel numbers** supported on the grid (0 denoting the channel at the origin frequency)

7.5.2 Programming Overview

A module advertises the **grid types** it supports on Page 04h. The grid type definitions include grid spacing, channel numbering scheme including numbering origin, and the supported range of channel numbers.

The registers for programming a single optical channel (for one Media Lane) are located on Page 12h:

- For **grid-based** applications, the grid to be used and a desired channel number on that grid are set.
- For **grid-less** applications, for historical reasons[1] the center frequency is not programmed directly, but in a base-plus-offset approach: The host selects and tunes to a nearby channel from a coarse-grained grid and then add a suitable fine-grained frequency offset (called a fine tuning offset) to achieve any desired off-grid center frequency.

To perform this function, so-called fine-tuning support (adding a small frequency offset to a grid frequency) must be available and enabled. *For instance, selecting a grid with 50 GHz channel spacing (or less) and offset tuning with 1MHz resolution over a +/-32 GHz range allows to program arbitrary grid-less channels. Note that tuning is performed as an initial setup step.*

When selecting another optical channel (grid spacing or channel number), the module must be in the **DPDeactivated** state. Attempts to change the optical channel (grid selection or channel selection) while the corresponding Data Path is in any other Data Path state may result in unspecified behavior, as the module may tune the laser before or after the change has been made.

A **Tuning Not Accepted** Flag is raised when the module is currently unable to serve the tuning request following a host programming grid or channel. The Flag indicates that the nominally programmed channel or grid values do no longer reflect the actual laser condition.

[1] The absence of a direct frequency setting method has historical reasons: grid-less applications are a relatively recent development and modules with tunable lasers have traditionally been programmed via channel numbers. Future versions of CMIS may include a direct frequency setting method.

Unlike channel selection, laser frequency fine-tuning offsets, and target output power settings may be programmed as long as the corresponding Data Path is not in a transient state.

The module is expected to align the laser to the fine-tuning offset value regardless of whether the value has been changed before, during or after laser tuning.

Modules that support programmable output power are also expected to align the output power to the target output power value regardless of when the change was made.

8 Module Management Memory Map

This chapter defines the structure and the meaning of the registers and fields in the Management Memory Map of a CMIS compliant module.

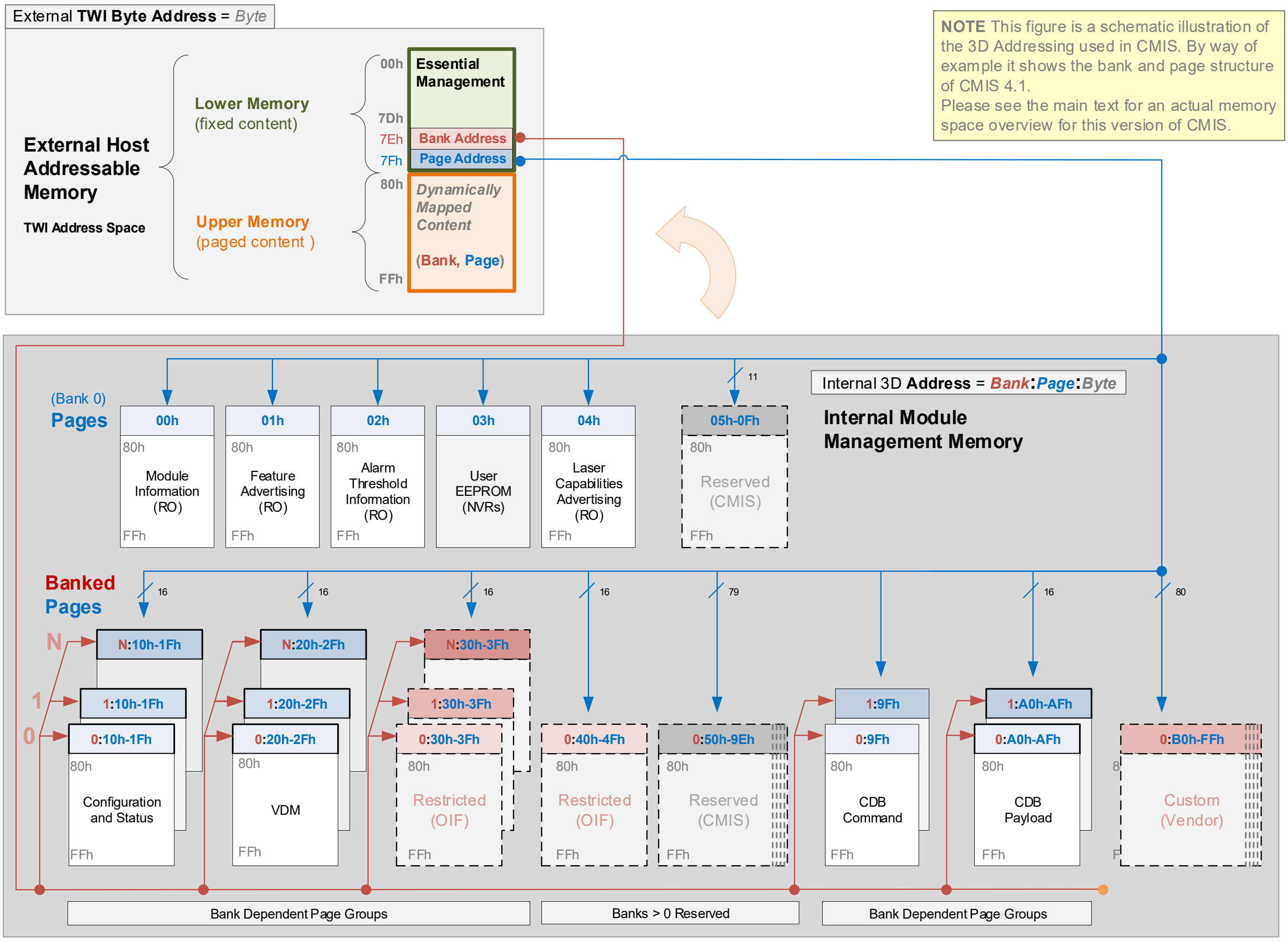
8.1 Overview and General Specifications

8.1.1 Management Memory Structure and Mapping

The register access protocol defined in section 5.2 only supports eight-bit Byte addresses. This limits the size of the **host addressable memory** that can be immediately accessed by the host to 256 bytes.

The host addressable memory is divided in two segments, **Lower Memory** (addresses 00h through 7Fh) and **Upper Memory** (addresses 80h through FFh).

This is illustrated in the upper part of Figure 8-1:



NOTE This figure is a schematic illustration of the 3D Addressing used in CMIS. By way of example it shows the bank and page structure of CMIS 4.1.
Please see the main text for an actual memory space overview for this version of CMIS.

Figure 8-1 CMIS Module Memory Map (Conceptual View)

A larger **accessible** management memory is required for all but the most basic modules. This additional memory is supported by a structure of 128-byte **Pages** and by a mechanism for dynamically mapping any of these 128-byte Pages from a larger internal management memory space into the Upper Memory of the host addressable space.

Basic modules that support only a fixed address space without Upper Memory content switching are called **flat memory modules**, whereas modules that support a larger internal management memory with dynamic mapping of Pages into Upper Memory are called **paged memory modules**.

Note: In this version of CMIS, flat memory modules are assumed to support only read-only static data as provided by an EEPROM. See sections 6.3.2.3. and 8.2.

The conceptual structure of the additional internal management memory[1] and the dynamic mapping into Upper Memory is shown in the lower part of Figure 8-1. The management memory inside the module is arranged as a unique and always immediately host accessible address space of 128 bytes (Lower Memory) and as multiple upper address subspaces of 128 bytes each (**Pages**), only one of which is selected at any one time to be host visible in Upper Memory. A second level of Page selection is available for Pages for which several instances exist (e.g. where a **Bank** of Pages with the same Page number is supported).

This structure supports the flat 256-byte memory for **flat memory** modules like passive copper modules and permits timely access to addresses in the Lower Memory, e.g. Flags and Monitors, also for **paged memory modules**.

Less time critical entries, e.g. serial ID information and threshold settings, are available using the Page Select function in Lower Memory. For more complex modules which require a larger amount of management memory the host needs to use dynamic mapping of the various Pages into the host addressable Upper Memory address space, whenever needed.

Note: The management Memory Map has been designed largely after the QSFP Memory Map. This Memory Map has been changed in order to accommodate 8 electrical lanes and to limit the required memory space. The single address approach is used as found in QSFP. Paging is used in order to enable time critical interactions between host and module.

8.1.2 Map of Supported Pages and Banks

A basic 256-byte subset of the Management Memory Map is mandatory for all CMIS compliant devices, flat memory modules and paged memory modules.

Other parts are only available for paged memory modules, where a few basic Pages are required for all paged memory modules, while most other pages are supported as explicitly advertised by the module. See Table 8-40 for details regarding the advertisement of supported management memory spaces (Banks, Pages).

In particular, Lower Memory and Page 00h in Upper Memory is supported by all modules.

Flat memory modules support only Lower Memory and Page 00h which is fixed mapped to Upper Memory.

Paged memory modules must additionally support Pages **01h**, **02h** and Bank 0 of Pages **10h** and **11h**.

Bank 0 of Pages 10h-1Fh, provides lane-specific registers for the first 8 lanes, and each additional Bank provides support for an additional 8 lanes.

Note, however, that generally the allocation of information over the Banks may be Page specific and may not to be related to grouping data for 8 lanes. For instance, additional Banks of Pages can also be supported for modules which require larger volume of management data.

The following Figure 8-2 and Table 8-1 provide an overview and a detailed map of all Pages in the Management Memory Map in the two-dimensional space spanned by Page Address and Bank Address.

[1] The actual physical storage structure of these memory pages is not in the scope of this specification. This specification only defines how the available memory pages can be addressed.

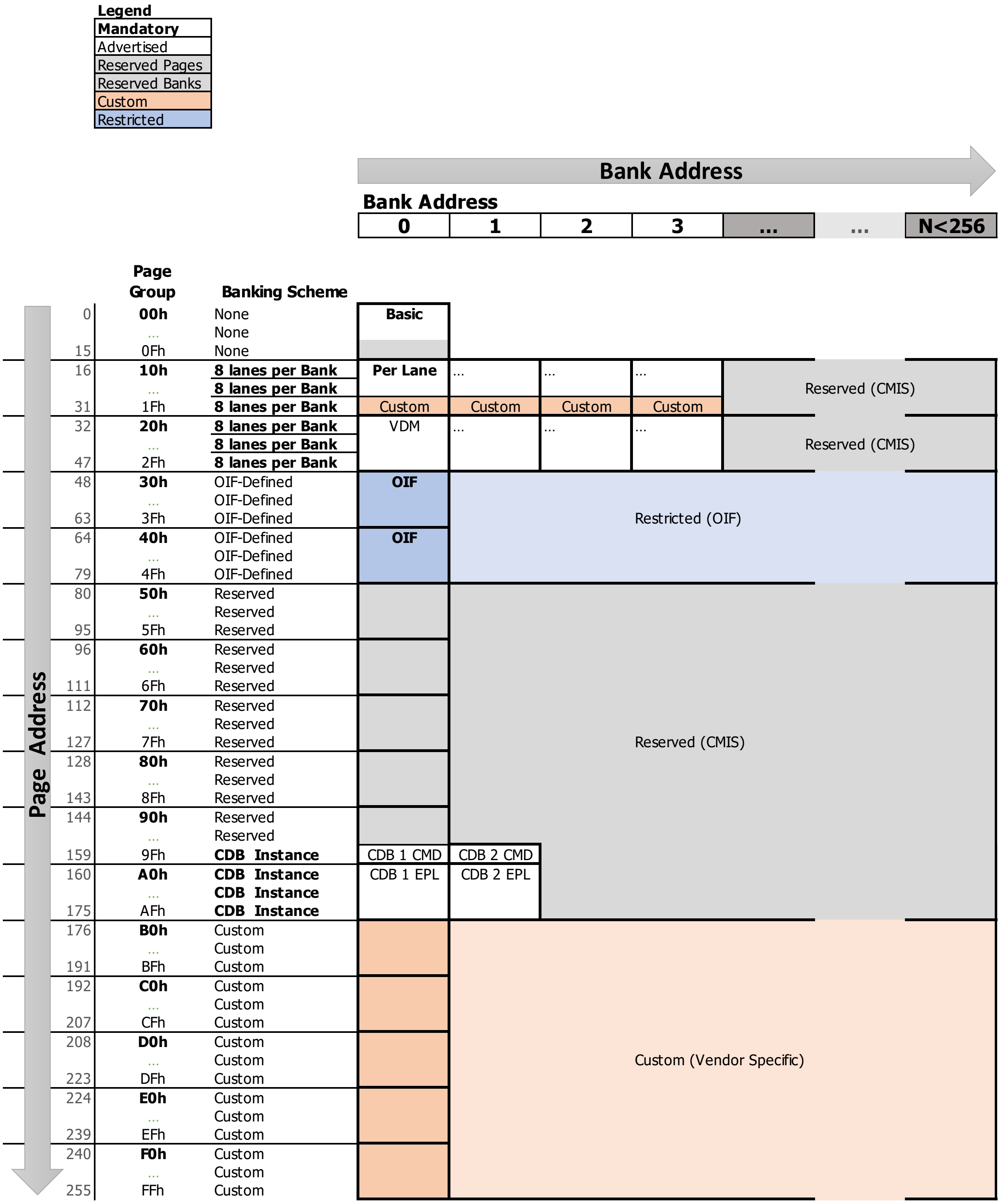


Figure 8-2 CMIS Bank and Page Group Iconic Memory Map Overview

Table 8-1 List of CMIS Pages

Page	#	Page Description	Data per Bank	# Banks	Type	See
00h	1	Administrative Information	n/a	1 of 1	RO	8.3
01h	1	Advertising	n/a	1 of 1	RO	8.4
02h	1	Thresholds Information	n/a	1 of 1	RO	8.5
03h	1	**User** NV RAM	n/a	1 of 1	RW	8.6
04h	1	Laser Capabilities Advertising	n/a	1 of 1	RO	8.7
05h-0Fh	11	**Reserved Pages** (CMIS)	n/a	1 of 1	-	-
10h	1	Lane and Data Path Configuration	8 lanes group	4 of 256	RW	8.8
11h	1	Lane and Data Path Status	8 lanes group	4 of 256	RO	8.9
12h	1	Tunable Laser Control and Status	8 lanes group	4 of 256	mixed	8.10
13h	1	Module Performance Diagnostics Control	8 lanes group	4 of 256	RW	8.11
14h	1	Module Performance Diagnostics Results	8 lanes group	4 of 256	RO	8.12
15h	1	Timing Characteristics	8 lanes group	4 of 256	RO	8.13
16h-1Dh	9	**Reserved Pages** (CMIS)	8 lanes group	4 of 256	-	-
1Eh-1Fh	2	**Custom Pages**	8 lanes group	4 of 256	-	-
20h	1	VDM Observable Descriptors[1][64]	8 lanes group	4 of 256	RO	8.14
21h	1	VDM Observable Descriptors[2][64]	8 lanes group	4 of 256	RO	
22h	1	VDM Observable Descriptors[3][64]	8 lanes group	4 of 256	RO	
23h	1	VDM Observable Descriptors[4][64]	8 lanes group	4 of 256	RO	
24h	1	VDM Samples[1][64]	8 lanes group	4 of 256	RO	8.14.2
25h	1	VDM Samples[2][64]	8 lanes group	4 of 256	RO	
26h	1	VDM Samples[3][64]	8 lanes group	4 of 256	RO	
27h	1	VDM Samples[4][64]	8 lanes group	4 of 256	RO	
28h	1	VDM TC Flagging ThresholdQuads[1][16] [1]	8 lanes group	4 of 256	RO	8.14.3
29h	1	VDM TC Flagging ThresholdQuads[2][16]	8 lanes group	4 of 256	RO	
2Ah	1	VDM TC Flagging ThresholdQuads[3][16]	8 lanes group	4 of 256	RO	
2Bh	1	VDM TC Flagging ThresholdQuads[4][16]	8 lanes group	4 of 256	RO	
2Ch	1	VDM TC FlagQuads[256] (threshold crossing)	8 lanes group	4 of 256	RO/COR	8.14.4
2Dh	1	VDM MaskQuads[256]	8 lanes group	4 of 256	RW	8.14.5
2Eh	1	**Reserved Pages** (CMIS)	8 lanes group	4 of 256	-	-
2Fh	1	VDM Advertisements and Dynamic Control	8 lanes group	4 of 256	RW	8.14.6
30h-3Fh	16	**Restricted Pages** OIF (C-CMIS)	-	-	-	-
40h-4Fh	16	**Restricted Pages** OIF (C-CMIS)	-	-	-	-
50h-9Eh	79	**Reserved Pages** (CMIS)	-	-	-	-
9Fh	1	CDB Command/Response with Local Payload	CDB Instance	2 of 256	RW	8.15
A0h	1	CDB EPL[1][128] Extended Payload Segments	CDB Instance	2 of 256	RW	8.16
A1h	1	CDB EPL[2][128]	CDB Instance	2 of 256	RW	
A2h	1	CDB EPL[3][128]	CDB Instance	2 of 256	RW	
A3h	1	CDB EPL[4][128]	CDB Instance	2 of 256	RW	
A4h	1	CDB EPL[5][128]	CDB Instance	2 of 256	RW	
A5h	1	CDB EPL[6][128]	CDB Instance	2 of 256	RW	
A6h	1	CDB EPL[7][128]	CDB Instance	2 of 256	RW	
A7h	1	CDB EPL[8][128]	CDB Instance	2 of 256	RW	
A8h	1	CDB EPL[9][128]	CDB Instance	2 of 256	RW	
A9h	1	CDB EPL[10][128]	CDB Instance	2 of 256	RW	
AAh	1	CDB EPL[11][128]	CDB Instance	2 of 256	RW	
ABh	1	CDB EPL[12][128]	CDB Instance	2 of 256	RW	
ACh	1	CDB EPL[13][128]	CDB Instance	2 of 256	RW	
ADh	1	CDB EPL[14][128]	CDB Instance	2 of 256	RW	
AEh	1	CDB EPL[15][128]	CDB Instance	2 of 256	RW	
AFh	1	CDB EPL[16][128]	CDB Instance	2 of 256	RW	
B0h-FFh	80	**Custom Pages**	Vendor specific	-	-	-

[1] Note that one ThresholdQuad set of thresholds serves several Observables.

8.1.3 Specifications Conventions

This section presents specifications conventions that are used in the Management Memory Map definitions.

8.1.3.1 Reserved Locations

Pages or Banks marked as **reserved** are reserved for future use in this specification. Reserved Pages or Banks are not accessible, i.e. neither functionality nor locations are supported (see section 8.2.13).

Locations within a Page (Bytes, Fields, or Bits) marked as **reserved** are also reserved for future use in this specification (see section 3.1). However, such locations are accessible, albeit without associated functionality.

The module shall zero-initialize accessible reserved locations. There are no other obligations for the module.

Both host and module shall neither use (evaluate) nor modify data from reserved locations.

Note: A subtle exception is for a defensive host to read (but not use) locations reserved for Flags.

The effects of a forbidden host WRITE to a reserved location is undefined.

Note: Other organizations must contact the managing organization or the editor of this specification to request allocations of registers, fields, or bits.

8.1.3.2 Custom Locations

Pages or Banks marked as **custom** are for vendor defined use. Access to custom Banks or Pages that are not functionally supported by a module shall not be supported, i.e. no access shall be possible (see section 8.2.13).

Locations within a Page (Bytes, Fields, or Bits) marked as **custom** are not governed by this specification and may be vendor defined. The use of registers defined as custom may be subject to additional agreements between module users and vendors.

Note: The requirement to generally enable vendor-agnostic management of fully CMIS compliant modules implies that the use of custom managed elements must be optional.

Note: The requirement to allow vendor-agnostic management of CMIS compliant modules implies that custom interrupt sources (Flags) must remain silent unless the host is known to be aware of the custom extensions.

8.1.3.3 Register Default Values

Default values for all **control** registers are 0 unless otherwise specified.

Note: Hosts are encouraged to review or explicitly set critical registers and not rely on module default values.

Note: For Control Set registers the default settings may be Application dependent, see section 6.2.3.

The default value of **Masks** for Flags residing on **optional** pages is 1 (Interrupt is masked by default).

Note: A default value of 1 is also recommended for all Masks of ***custom*** *Interrupt sources.*

8.1.3.4 Byte Order (Endianness)

The default byte order of multi-byte registers representing numerical types is **Big Endian**, i.e. the lowest byte address contains the most significant byte of the multi-byte value. Exceptions are stated explicitly

Note: Independent of proper representation, atomic accesses to multi-byte registers may require access synchronization protocols (see also section 5.2.3 for a discussion of data coherency)

8.1.3.5 Access Types

The following Field access types are distinguished and indicated (together with an optionality indication) in the Type column of the field definition tables defining the Management Memory Map later in this chapter. The word "element" is used here to refer to an entire Byte, a Field, or a Bit:

Table 8-2 Access Types

Access Type	Description
RW	A **readable** and **writeable** element.
RWW	A **readable** and **writeable** element that can also be **modified** by the module. *Note: This access type should be used only for interactions governed by a protocol.*
RO	A **read-only** element. A WRITE of a value to a read-only element is allowed but has no effect.
WO	A **write-only** element. A READ from a write-only element is **allowed** but delivers **unpredictable** values.
WO/SC	A **write-only** element with **self-clearing** side effect. A READ from a WO/SC element is **allowed** and delivers a **zero** value, except transiently when reading before the module has evaluated and cleared the non-zero bits written.
RO/COR	A **read-only** element with **clear-on-read** (clear-after-read) side effect. All bits in a RO/COR Byte are cleared by the module after the Byte value has been read. *Note: Modules may also combine clearing and update (clear-or-update-on-read).*

Note: The clear-on-read (COR) access type is used for latched Flags that indicate that an event has occurred, or a condition was present since the last read. Note that a Flag cleared by reading will be set again if the module evaluates the flagged condition as still being present. The timing of this Flag update after a clear-on-read is specified in section 10.

Note: Bits with WO or WO/SC access types are often used as command or trigger bits that start the execution of something. A WO/SC specification is mainly useful when privacy protection of written data is to be specified.

Note: The ACCESS protocol allows to read from or write to any addressable register; the Access Type is therefore only a specification of value related behavior, not an actual restriction. Especially there are neither protocol violations nor protocol exceptions when the host, e.g., writes to a RO element.

8.1.3.6 Optionality Indications

Identified elements of the Memory Map described in this chapter are conditionally or unconditionally optional. An optional element is either supported or not-supported by a specific module.

Three levels of optionality are distinguished: Optionality of Pages, of Banks, and of Fields on (banked) Pages.

For Pages, only two cases are distinguished, as documented in the text describing the Page

- Required: All paged memory modules must support required Pages.
- Optional: All other Pages are optional. The host can always determine at runtime, directly or indirectly from advertisements or other field values, if an optional Page is supported.

For Banks, the host can always determine at runtime, directly or indirectly from advertisements or other field values, if an optional Bank is supported.

For Fields, the following classification is documented in the Type column of the Field definition tables:

Required	**Rqd.**	always supported when the relevant Page is supported.
Advertised	**Adv.**	supported when indicated in a clearly associated feature advertisement
Conditional	**Cnd.**	supported when some (documented) run-time evaluated condition is present
Optional	**Opt.**	optionally supported, unknown conditions, inferred only from feature behavior

Note: Ideally the host should be able to find out if an optional field is actually supported. However, for historical reasons, there are some optional fields for which no advertisement exists. These tend to be features which have a meaningful default value where absence of support can be inferred from non-changing default values.

8.1.4 General Specifications

This section presents specification that are applicable to wide parts of the Management Memory Map.

8.1.4.1 Multi-Byte Reporting Registers

Scalar multi-byte reporting registers (e.g. monitored numerical values) with explicitly specified data type (i.e. specified as e.g. F16, S32, S64) are coherently and atomically readable in a multibyte READ over the length of the multi-byte reporting register. See section 3.4 for defined data types and section 5.2.5 for data coherency in multi-byte ACCESS operations.

Note: Unlike in previous revisions a multi-byte register is now described as one multi-byte register covering a byte address range, not as individual bytes with individual byte names.

Other multi-byte registers do not support coherent and atomic ACCESS unless explicitly specified.

8.1.4.2 Flags, Masks, and Interrupts

Flags

In this specification, a **Flag** is a latched indicator bit in management memory with associated maskable Interrupt request generation.

While a Flag is set, an **Interrupt** request is generated unless an associated **Mask** bit is set.

Once asserted, a Flag bit remains set (latched) until cleared by a READ of the Byte containing the Flag (or cleared in the course of the reaction to a Reset signal).

A Flag has **clear on read** access type.

After being read and cleared, a Flag indicating presence of a condition is raised again if the flagged condition persists; this will cause the Interrupt request to be asserted again (unless masked). Flags indicating the occurrence of an event will only be raised again when the next flagged event occurs.

Note: In a situation where a flagged condition itself is permanent, the module may first clear (on read) and then re-assert the cleared Flag after a finite condition re-evaluation time interval. A fast sequence of Flag readings may therefore result in toggling Flag values while the flagged condition continues to be present.

Note: Generally, the timing limits, sequencing relationships, and interdependencies with regard to state changes of Flags, Mask, and Interrupt Requests is not specified in this version of CMIS (see chapter 10). Hosts should therefore be aware that the synchronization between changes of a flagged condition or event, Flag or Mask state changes, and Interrupt request assertion changes (and hence the transient behavior between stable conditions) may vary significantly across modules.

Note: A module raises Flags only when allowed by Flagging Conformance rules (see section 6.3.4) which depend on the states of the Module State Machine and the Data Path State Machine.

Flag Summaries

To allow quick scanning of a Flag register tree, a bit in a Flag summary register indicates if a Flag in an associated underlying Flag register (or in a next level Flag summary) is set. Note that a Flag summary bit itself is not a Flag and reading a Flags summary register does not clear any underlying (summarized) Flag.

Interrupt Request

The Interrupt request hardware signal is asserted as long as there is any Flag set with its corresponding Mask bits cleared.

An Interrupt is raised at the onset of an unmasked flagged condition or at the occurrence of an unmasked flagged event and remains asserted until all asserted unmasked Flags (both module-level and lane-specific) either have been cleared by a host read or have been masked by the host (by setting the relevant Mask bits as described below).

Masks

In this specification a **Mask** bit is a host writable bit in management memory which, when set, suppresses future Interrupt generation, and ceases current Interrupt generation caused by its associated Flag bit.

Note: Mask bits may be used to prevent an Interrupt request from remaining active while the host performs actions to acknowledge and handle the condition causing the Flag to be set.

For each Flag there is a corresponding Mask bit, usually at the same Bit index in a Mask configuration Byte. A set Mask bit suppresses the contribution of its corresponding Flag to the hardware Interrupt signal generation. A cleared Mask bit enables the contribution of its corresponding Flag to the hardware Interrupt signal generation. Setting a Mask on a set Flag deasserts the hardware Interrupt unless there are other unmasked Flags set.

Mask bits are volatile: During module initialization all CMIS defined Mask bits are set to default values (at exit from **MgmtInit** state to **ModuleReady** state, see section 6.3.2.7).

Note: This choice exists for historical reasons. Normally default Interrupt silence would be preferable.

Note: The host may use known Mask bits to temporarily suppress a continued hardware Interrupt assertion due to stable conditions, i.e. from not cleared Flags, which would otherwise continually reassert the hardware Interrupt signal.

Note: Custom masks for custom functions should be masked by default.

8.1.4.3 Generic Checksums

Page Checksum
Unless specified differently, a Page Checksum value is the arithmetic sum of a given Byte range, modulo 256.

CDB Checksum
A CDB message checksum value is the one's complement of the arithmetic sum of a given Byte range, modulo 256.

Note: Both checksums may be applied to non-contiguous ranges (where Bytes are excluded or zeroed).

8.2 Lower Memory (Control and Status Essentials)

Lower Memory consists of the "lower" 128 bytes of the 256-byte host-addressable memory.

Lower Memory is addressed with byte addresses in the range 00h to 7Fh. The byte addresses of Lower Memory are statically interpreted: a given byte address always refers to the same physical register in Lower Memory.

Note: In contrast, Upper Memory byte addresses in the range 80h to FFh are dynamically interpreted: the actually addressed register depends on the current page mapping as described in section 8.1.1 and below.

Note: Lower Memory content is always immediately accessible and therefore contains mainly registers that are deemed to be most frequently accessed by the host, such as module state and interrupt status, module level monitors, module level Flags and Masks, lane level Flags summary, and global control functions.

Flat memory modules that support only constant read-only data are not required to support any writeable control fields or readable dynamic status reporting fields defined in Lower Memory.

Note: This enables ROM-based implementation of the addressable memory in flat memory modules.

Lower Memory is subdivided into subject areas as shown in the following table:

Table 8-3 Lower Memory Overview

Address	Size	Subject Area	Description
0–2	3	Management Characteristics	Basic Information about how this module is managed
3	1	Global Status Information	Current state of Module, Interrupt signal status
4–7	4	Flags Summary	Summary of Flags set on specific Pages (and Banks)
8-13	6	Module-Level Flags	Flags that are not lane or Data Path specific
14-25	12	Module-Level Monitors	Monitors that are not lane or Data Path specific
26-30	5	Module-Level Controls	Controls applicable to the module as a whole
31-36	6	Module-Level Masks	Mask bits for the Module-Level Flags
37-38	2	CDB Command Status	Status of current CDB command
39-40	2	Module Active Firmware Version	Module Active Firmware Version number
41	1	Fault Information	Fault cause for entering ModuleFault state
42-63	22	-	**Reserved[22]**
64-84	21	-	**Custom[21]**
85-117	33	Supported Applications Advertising	Applications supported by module Data Path(s)
118-125	8	Password Facilities	Password Entry and Change (mechanism only)
126-127	2	Page Mapping	Page mapping into host addressable Upper Memory

8.2.1 Management Characteristics

The Management Characteristics fields described in Table 8-4 provide fundamental management characteristics of the module that allow hosts to verify if, and to determine how, they can operate and manage the module.

The Management Characteristics fields include an SFF-8024 module type identifier (this is a value defined in the Identifier Values table in [5]) which implicitly contains information about the management protocol and the management Memory Map offered by the module.

Note: Since the management protocol is not systematically encoded in the SFF 8024 identifiers, hosts will have to test against a list of supported SFF 8024 module type identifiers.

The other fields indicating management characteristics like the CMIS version number, memory model (flat memory or paged memory), support for intervention-free reconfiguration, and management interface speed, can be interpreted once the module has been recognized as a CMIS compliant module.

*Note: Modules may also be classified by their functional "module type", depending on the unique or main Application that they support. This kind of "module type" is **not** encoded in the SFF8024Identifier field, but can instead be derived from other advertisements.*

Table 8-4 Management Characteristics

Byte	Bits	Field Name	Field Description	Type
0	7-0	SFF8024Identifier	**SFF8024Identifier** is an SFF-8024 module type Identifier from the Identifier Values table in [5] which allows to infer both physical form factor and management protocol of the module. *Note: The CMIS interpretation of all other registers or fields is valid only when the fundamental SFF8024Identifier indicates that the module uses the CMIS management protocol.*	RO Rqd.
1	7-0	CmisRevision	CMIS revision number (decimal): The upper nibble (bits 7-4) is the integer part (major number) The lower nibble (bits 3-0) is the decimal part (minor number) *Example: 01h indicates version 0.1, 21h indicates version 2.1.* *Note: See Appendix G.3 for interoperability implications of the major revision number (integer part).*	RO Rqd.
2	7	MemoryModel	Indicator of the memory model of the module: 0b: Paged memory (Pages 00h-02h, 10h-11h supported) 1b: Flat memory (Page 00h supported only)	RO Rqd.
	6	SteppedConfigOnly	0b: Module supports **intervention-free reconfiguration** 1b: Module supports **only step-by-step reconfiguration**, where a host WRITE to ApplyDPInit is (technically) accepted in **all** states **without** causing DPSM state changes, and where ApplyImmediate is not supported (i.e. a WRITE to ApplyImmediate is ignored). *Note: Support for intervention-free reconfiguration is required for any Application that supports or requires time critical speed negotiation with active modules that is not achievable with stepwise configuration, such as InfiniBand or Fibre Channel.* *Note: See section 6.2.4 for more information*	
	5-4	-	**Reserved**	
	3-2	MciMaxSpeed	Indicates maximum supported clock speed of Management Communication Interface (MCI): 00b: Module supports up to 400 kHz 01b: Module supports up to 1 MHz 10b: **Reserved** 11b: **Reserved**	RO Rqd.
	1-0	-	**Reserved**	

8.2.2 Global Status Information

The fields described in Table 8-5 provide fundamental module status indicators.

See section 6.3 for information on the Module State Machine

Table 8-5 Global Status Information

Byte	Bits	Field Name	Field Description	Type
3	7-4	-	**Reserved**	
	3-1	ModuleState	Current Module State (see Table 8-6 for encoding and section 6.3.2 for a description of the meaning of ModuleState) *Notes:* *- Flat memory modules always report ModuleReady.* *- Not all states of the Module State Machine are observable.*	RO Rqd.
	0	InterruptDeasserted	Status of Interrupt output signal **1b:** Interrupt not asserted (**default**) 0b: Interrupt asserted	

Table 8-6 Module State Encodings

Code	Module State	Description
000b	-	**Reserved**
001b	ModuleLowPwr	
010b	ModulePwrUp	
011b	ModuleReady	This is the only state reported by flat memory modules
100b	ModulePwrDn	
101b	ModuleFault	
110b	-	**Reserved**
111b	-	**Reserved**

8.2.3 Flags Summary

The Flags Summary bits indicate when any Flags are asserted on specific pages, for up to 4 Banks.

Note: To clear a summarized Flag, the Flag itself must be read from the relevant Page on the appropriate Bank.

Table 8-7 Lane-Level Flags Summary

Byte	Bit	Field Name	Field Description	Type
4	7	-	Reserved	
	6	-	Reserved	
	5	-	Reserved	
	4	-	Reserved	
	3	FlagsSummaryBank0Page2Ch	1b: at least one Flag is set on Bank 0, Page 2Ch	RO Adv.
	2	FlagsSummaryBank0Page14h	1b: at least one Flag is set on Bank 0, Page 14h	RO Adv.
	1	FlagsSummaryBank0Page12h	1b: at least one Flag is set on Bank 0, Page 12h	RO Adv.
	0	FlagsSummaryBank0Page11h	1b: at least one Flag is set on Bank 0, Page 11h	RO Rqd.
5	7	-	Reserved	
	6	-	Reserved	
	5	-	Reserved	
	4	-	Reserved	
	3	FlagsSummaryBank1Page2Ch	1b: at least one Flag is set on Bank 1, Page 2Ch	RO Adv.
	2	FlagsSummaryBank1Page14h	1b: at least one Flag is set on Bank 1, Page 14h	RO Adv.
	1	FlagsSummaryBank1Page12h	1b: at least one Flag is set on Bank 1, Page 12h	RO Adv.
	0	FlagsSummaryBank1Page11h	1b: at least one Flag is set on Bank 1, Page 11h	RO Adv.
6	7	-	Reserved	
	6	-	Reserved	
	5	-	Reserved	
	4	-	Reserved	
	3	FlagsSummaryBank2Page2Ch	1b: at least one Flag is set on Bank 2, Page 2Ch	RO Adv.
	2	FlagsSummaryBank2Page14h	1b: at least one Flag is set on Bank 2, Page 14h	RO Adv.
	1	FlagsSummaryBank2Page12h	1b: at least one Flag is set on Bank 2, Page 12h	RO Adv.
	0	FlagsSummaryBank2Page11h	1b: at least one Flag is set on Bank 2, Page 11h	RO Adv.
7	7	-	Reserved	
	6	-	Reserved	
	5	-	Reserved	
	4	-	Reserved	
	3	FlagsSummaryBank3Page2Ch	1b: at least one Flag is set on Bank 3, Page 2Ch	RO Adv.
	2	FlagsSummaryBank3Page14h	1b: at least one Flag is set on Bank 3, Page 14h	RO Adv.
	1	FlagsSummaryBank3Page12h	1b: at least one Flag is set on Bank 3, Page 12h	RO Adv.
	0	FlagsSummaryBank3Page11h	1b: at least one Flag is set on Bank 3, Page 11h	RO Adv.

8.2.4 Module-Level Flags

The registers described in Table 8-8 contain module-level (global) Flags, with Masks described in section 8.2.7.

Note: The general behavior of Flags, Masks, and Interrupts is specified in section 8.1.4.2.

Module-level Flags are used to report module-level status changes, operating failures, as well as threshold crossing alarms and warnings for monitored observables. Monitors with associated alarm and/or warning thresholds have associated alarm Flags, warning Flags. For normal operation and in default state, these Flags are cleared. Refer to section 6.3.4 for Flagging rules depending on ModuleState.

Byte 13 is provided for Custom Module Level Flags.

Table 8-8 Module Flags (paged memory modules only)

Byte	Bit	Name	Field Description	Type
8	7	CdbCmdCompleteFlag2	Latched Flag to indicate completion of a CDB command for CDB instance 2. Support is advertised in field 01h:163.7-6	RO/COR Adv.
	6	CdbCmdCompleteFlag1	Latched Flag to indicate completion of a CDB command for CDB instance 1. Support is advertised in field 01h:163.7-6	RO/COR Adv.
	5-3	-	Reserved	Rqd.
	2	DataPathFirmwareErrorFlag	Latched Flag to indicate that subordinated firmware in an auxiliary device for processing transmitted or received signals (e.g. a DSP) has failed.	RO/COR Adv.
	1	ModuleFirmwareErrorFlag	Latched Flag to indicate that self-supervision of the main module firmware has detected a failure in the main module firmware itself. There are several possible causes of the error such as program memory becoming corrupted and incomplete firmware loading.	RO/COR Adv.
	0	ModuleStateChangedFlag	Latched Flag to indicate a Module State Change	RO/COR Rqd.
9	7	VccMonLowWarningFlag	Latched Flag for low supply voltage warning	RO/COR Adv.
	6	VccMonHighWarningFlag	Latched Flag for high supply voltage warning	
	5	VccMonLowAlarmFlag	Latched Flag for low supply voltage alarm	
	4	VccMonHighAlarmFlag	Latched Flag for high supply voltage alarm	
	3	TempMonLowWarningFlag	Latched Flag for low temperature warning	
	2	TempMonHighWarningFlag	Latched Flag for high temperature warning	
	1	TempMonLowAlarmFlag	Latched Flag for low temperature alarm	
	0	TempMonHighAlarmFlag	Latched Flag for high temperature alarm	
10	7	Aux2MonLowWarningFlag	Latched Flag for low Aux 2 monitor warning	RO/COR Adv.
	6	Aux2MonHighWarningFlag	Latched Flag for high Aux 2 monitor warning	
	5	Aux2MonLowAlarmFlag	Latched Flag for low Aux 2 monitor alarm	
	4	Aux2MonHighAlarmFlag	Latched Flag for high Aux 2 monitor alarm	
	3	Aux1MonLowWarningFlag	Latched Flag for low Aux 1 monitor warning	
	2	Aux1MonHighWarningFlag	Latched Flag for high Aux 1 monitor warning	
	1	Aux1MonLowAlarmFlag	Latched Flag for low Aux 1 monitor alarm	
	0	Aux1MonHighAlarmFlag	Latched Flag for high Aux 1 monitor alarm	
11	7	CustomMonLowWarningFlag	Latched Flag for low Vendor Defined Monitor warning	RO/COR Adv.
	6	CustomMonHighWarningFlag	Latched Flag for high Vendor Defined Monitor warning	
	5	CustomMonLowAlarmFlag	Latched Flag for low Vendor Defined Monitor alarm	
	4	CustomMonHighAlarmFlag	Latched Flag for high Vendor Defined Monitor alarm	
	3	Aux3MonLowWarningFlag	Latched Flag for low Aux 3 monitor warning	
	2	Aux3MonHighWarningFlag	Latched Flag for high Aux 3 monitor warning	
	1	Aux3MonLowAlarmFlag	Latched Flag for low Aux 3 monitor alarm	
	0	Aux3MonHighAlarmFlag	Latched Flag for high Aux 3 monitor alarm	
12	7-0	-	**Reserved[1]**	
13	7-0	-	**Custom[1]**	

8.2.5 Module-Level Monitor Values

Real time monitoring for module-level observables includes two monitors with fixed observables (temperature and supply voltage) and four monitors with selectable function (3 auxiliary and 1 vendor defined) as shown in Table 8-9.

Note: The data format of a monitored value may have greater resolution and range than required.

Measurement accuracy is defined by the relevant interface standard or module product specification.

The reported monitoring results of supported module level monitors shall be within the relevant accuracy requirements when the module is in the ModuleReady state.

Table 8-9 Module-Level Monitor Values (paged memory modules only)

Byte	Bit	Register Name	Register Description	Type
14-15	7-0	TempMonValue	S16 **Module Temperature Monitor** (Current Value) internally measured temperature in 1/256 degree Celsius increments	RO Adv.
16-17	7-0	VccMonVoltage	U16 **Supply Voltage Monitor** (Current Value) internally measured input supply voltage in 100 µV increments	RO Adv.
18-19	7-0	Aux1MonValue	S16 **Aux1 Monitor** (see Table 8-43) (Current Value) The monitored observable is advertised in 01h:145.0: 0b: **Custom** 1b: **TEC Current** in 100%/32767 increments of maximum TEC current magnitude, i.e. of the larger of the heating or cooling current magnitudes +32767 (100%) of the max current magnitude when heating -32767 is -100% of the max current magnitude when cooling	RO Adv.
20-21	7-0	Aux2MonValue	S16 **Aux2 Monitor** (see Table 8-43) (Current Value) The monitored observable is advertised in 01h:145.1: 0b: **Laser Temperature**: in 1/256 degree Celsius increments 1b: **TEC Current** in 100%/32767 increments of maximum TEC current magnitude, i.e. of the larger of the heating or cooling current magnitudes +32767 (100%) of the max current magnitude when heating -32767 is -100% of the max current magnitude when cooling	RO Adv.
22-23	7-0	Aux3MonValue	S16 **Aux3 Monitor** (see Table 8-43) (Current Value) The monitored observable is advertised in 01h:145.2: 0b: **Laser Temperature**: in 1/256 degree Celsius increments 1b: **Additional Supply Voltage**: in 100 µV increments	RO Adv.
24-25	7-0	CustomMonValue	S16 or U16: **Custom** monitor (Current Value)	RO Adv.

8.2.6 Module-Level Controls

Module-Level (global) controls are applicable to the entire module or to all Lanes or Data Paths in the module.

Note: Lane-specific controls are located in Page 10h (see section 8.8).

Table 8-10 Module Global Controls (paged memory modules only)

Byte	Bit	Field Name	Field Description	Type
26	7	-	Reserved	
	6	LowPwrAllowRequestHW	Enables the evaluation of the LowPwrRequestHW hardware signal 0b: Module ignores the LowPwrRequestHW signal **1b**: Module evaluates the LowPwrRequestHW signal (**default**) *Note: See Table 6-12 and section 6.3.2.2 for more information* *Note: Since LowPwrRequestSW is cleared by default, the evaluation of LowPwrRequestHW is enabled by default, as to allow the host to request stopping in Low Power mode during start-up.*	RW Rqd.
	5	SquelchMethodSelect	0b: Squelching of Tx output reduces OMA 1b: Squelching of Tx output reduces P_{av} *Note: Method to choose depends on interface standard used.* *See Table 8-44 for SquelchMethodSelect capability advertising.*	RW Adv.
	4	LowPwrRequestSW	0b: No request 1b: Request for the module to stay in, or to return into, Low Power mode *Note: See Table 6-12 and section 6.3.2.4 for more information*	RW Rqd.
	3	SoftwareReset	Self-clearing trigger bit that causes the module to be reset when 1b is written to it. The effect of a SoftwareReset trigger is the same as asserting the Reset hardware signal for the appropriate hold time, followed by its de-assertion. 0b: No action 1b: Software reset	WO/SC Rqd.
	2-0	-	**Custom**	
27-28	All	-	**Reserved[2]**	
29-30	All	-	**Custom [2]**	

8.2.7 Module-Level Masks

The host can control which Flags may cause a hardware Interrupt by setting Mask bits described in Table 8-11.

Note: The general behavior of Flags and Masks is specified in section 8.1.4.2.

Table 8-11 Module Level Masks (paged memory modules only)

Byte	Bits	Field Name	Field Description	Type
31	7	CdbCmdCompleteMask2	Mask bit for CdbCmdCompleteFlag2	RW Adv.
	6	CdbCmdCompleteMask1	Mask bit for CdbCmdCompleteFlag1	RW Adv.
	5-3	-	Reserved	
	2	DataPathFirmwareErrorMask	Mask bit for DataPathFirmwareErrorFlag	RW Adv.
	1	ModuleFirmwareErrorMask	Mask bit for ModuleFirmwareErrorFlag	RW Adv.
	0	ModuleStateChangedMask	Mask bit for ModuleStateChangedFlag	RW Rqd.
32	7	VccMonLowWarningMask	Mask bit for VccMonLowWarningFlag	RW Adv.
	6	VccMonHighWarningMask	Mask bit for VccMonHighWarningFlag	
	5	VccMonLowAlarmMask	Mask bit for VccMonLowAlarmFlag	
	4	VccMonHighAlarmMask	Mask bit for VccMonHighAlarmFlag	
	3	TempMonLowWarningMask	Mask bit for TempMonLowWarningFlag	
	2	TempMonHighWarningMask	Mask bit for TempMonHighWarningFlag	
	1	TempMonLowAlarmMask	Mask bit for TempMonLowAlarmFlag	
	0	TempMonHighAlarmMask	Mask bit for TempMonHighAlarmFlag	
33	7	Aux2MonLowWarningMask	Mask bit for Aux2MonLowWarningFlag	RW Adv.
	6	Aux2MonHighWarningMask	Mask bit for Aux2MonHighWarningFlag	
	5	Aux2MonLowAlarmMask	Mask bit for Aux2MonLowAlarmFlag	
	4	Aux2MonHighAlarmMask	Mask bit for Aux2MonHighAlarmFlag	
	3	Aux1MonLowWarningMask	Mask bit for Aux1MonLowWarningFlag	
	2	Aux1MonHighWarningMask	Mask bit for Aux1MonHighWarningFlag	
	1	Aux1MonLowAlarmMask	Mask bit for Aux1MonLowAlarmFlag	
	0	Aux1MonHighAlarmMask	Mask bit for Aux1MonHighAlarmFlag	
34	7	CustomMonLowWarningMask	Mask bit for CustomMonLowWarningFlag	RW Adv.
	6	CustomMonHighWarningMask	Mask bit for CustomMonHighWarningFlag	
	5	CustomMonLowAlarmMask	Mask bit for CustomMonLowAlarmFlag	
	4	CustomMonHighAlarmMask	Mask bit for CustomMonHighAlarmFlag	
	3	Aux3MonLowWarningMask	Mask bit for Aux3MonLowWarningFlag	
	2	Aux3MonHighWarningMask	Mask bit for Aux3MonHighWarningFlag	
	1	Aux3MonLowAlarmMask	Mask bit for Aux3MonLowAlarmFlag	
	0	Aux3MonHighAlarmMask	Mask bit for Aux3MonHighAlarmFlag	
35	7-0	-	**Reserved[1]** for Masks	
36	7-0	-	**Custom[1]** Module level Masks	

8.2.8 CDB Command Status

The CDB command status fields **CdbStatus**<i> provide the status of the most recently triggered CDB command execution or its result, separately for each CDB Instance <i>.

See section 7.2 for a description of the optional CDB feature, and sections 8.15 and 9 for more technical details

When CDB is used in background operation mode (see Table 8-48), the host can read the relevant CDB command status field while a CDB command is executing to obtain its current status.

When CDB is used in foreground operation mode, the host can read the relevant CDB command status field only after the command has completed, as can be determined by Acknowledge polling.

Table 8-12 CdbStatus fields (paged memory modules only)

Byte	Bits	Register Name	Register Description	Type
37	7-0	CdbStatus1	Status of the most recent CDB command in CDB instance 1	RO Adv.
38	7-0	CdbStatus2	Status of the most recent CDB command in CDB instance 2	RO Adv.

A CdbStatus field has the following format:

Table 8-13 Bit definitions within CdbStatus fields

Bits	Field Name	Field Description
7	CdbIsBusy	Bool: CdbIsBusy status bit indicates whether the module is still busy, or idle and ready to accept a new CDB command 0b: Module idle, host can write 1b: Module busy, host needs to wait
6	CdbHasFailed	Bool: CdbHasFailed bit indicates if there was a failure, after the module has completed execution of the last CDB command 0b: Last triggered CDB command completed successfully 1b: Last triggered CDB command failed
5-0	CdbCommandResult	The CdbCommandResult field provides more detailed classification for each of the three coarse query results that are encoded by the pair of bit 7 (CdbIsBusy) and bit 6 (CdbHasFailed) Coarse Status CdbIsBusy CdbHasFailed **IN PROGRESS** **1** X (don't care) **SUCCESS** 0 **0** **FAILED**: 0 **1** The interpretation of CdbCommandResult therefore depends on the coarse status as follows: **IN PROGRESS** 00h=Reserved 01h=Command is captured but not processed 02h=Command checking is in progress 03h=Command execution is in progress 04h-2Fh=Reserved 30h-3Fh=Custom **SUCCESS** 00h=Reserved 01h=Command completed successfully 02h=Reserved 03h=Previous CMD was ABORTED by CMD Abort 04h-1Fh=Reserved 20h-2Fh=Reserved 30h-3Fh=Custom **FAILED** 00h=Reserved 01h=CMDID unknown 02h=Parameter range error or parameter not supported

Bits	Field Name	Field Description
		03h=Previous CMD was not properly ABORTED (by CMD Abort) 04h=Command checking time out 05h=CdbChkCode Error 06h=Password related error (command specific meaning) 07h=Command not compatible with operating status 08h-0Fh=Reserved for STS command checking error 10h-1Fh=Reserved 20h-2Fh=For individual STS command or task error 30h-3Fh=Custom

The module sets the **CdbIsBusy** bit when a CDB command is triggered on the respective CDB instance.

The module clears the **CdbIsBusy** bit on successful or unsuccessful completion of a CDB command, after updating the fields describing the result of the completed command

- The **CdbHasFailed** bit reports if the command has failed
- The **CdbCommandResult** field provides a detailed classification

After command completion, the module does not change the **CdbStatus** byte of a CDB Instance, until the next CDB command is triggered for that CDB instance.

Note: CdbStatus fields are cleared on exit from module state MgmtInit (see section 6.3.2.7).

8.2.9 Module Active Firmware Version

The Bytes described in Table 8-14 allow a module to report the firmware major and minor revision for the active (i.e. currently running) firmware, or to indicate that no firmware is running.

Note: Module firmware may consist of multiple firmware components (program images, non-volatile data). It is strongly recommended that the firmware version identifies the aggregate (or bundle) of all firmware elements that can be updated by the vendor or by the host.

Note: For modules supporting firmware update, it is also strongly recommended that the firmware major and minor revision numbers together with build number (available by CDB query) uniquely specifies exactly one aggregate firmware configuration. Firmware aggregates that differ in only a single component should never have the same version identification (major, minor, build).

Note: The identification of temporary firmware component versions for lab test or debug, or identification of individual firmware components is not in the scope of this specification. However, the CDB firmware query command supports additional fields which could be used to mark such temporary firmware aggregates.

Reporting firmware version information is generally required and independent of whether a module supports firmware update by any method.

Content and meaning of firmware major and minor revision information reported are vendor dependent, but the format of both fields is defined to be integer. The encoding of the major and minor revision fields is:

- Major Revision = 0 and Minor Revision = 0 indicates that a module does not have any firmware.
- Major Revision = FFh and Minor Revision = FFh indicates that the active firmware image is invalid[1].
- All other Major and Minor Revision combinations indicate the active firmware version.

Table 8-14 Module Active Firmware Version

Byte	Bits	Register Name	Register Description	Type
39	7-0	ModuleActiveFirmwareMajorRevision	U8 Numeric representation of the module's active firmware major revision	RO Rqd.
40	7-0	ModuleActiveFirmwareMinorRevision	U8 Numeric representation of the module's active firmware minor revision	RO Rqd.

Note: A module supporting a second image reports version information for the inactive image in the Bytes described in section 8.4.1 with the same encoding.

Note: A module may or may not support a second (inactive) firmware image.

Note: Flat memory modules with firmware may also report their active firmware version in these Bytes.

Note: Modules that support a proprietary vendor specific firmware update method (instead of CDB as described later) shall still report the active firmware version in these bytes.

8.2.10 Module Fault Information

The optional ModuleFaultCause Byte describes the reason for the module having entered the ModuleFault state.

Note: In certain fault reaction scenarios the management communication interface may not be available.

Table 8-15 Fault Information (paged memory modules only)

Byte	Bits	Register Name	Register Description	Type
41	7-0	ModuleFaultCause	Reason of entering the ModuleFault state **0**: No Fault detected (or field not supported) 1: TEC runaway 2: Data memory corrupted 3: Program memory corrupted 4-31: **Reserved** (fault codes) 32-63: **Custom** (fault codes) 64-255: **Reserved** (general)	RO Opt.

[1] Such an indication could e.g. be given by a fixed bootloader that is independent of loaded firmware.

8.2.11 Applications Advertising

Bytes 00h:86-117 (see Table 8-19) provide space for an array of the first four of five bytes (see Table 8-18) of eight **Application Descriptors** (see section 6.2.1.4).

Note: Application Descriptors are used by the module to advertise supported Applications. See section 6.2.1.4 for more information about Application advertising .

Note: The fifth byte (MediaLaneAssignmentOptions) of each Application Descriptor is stored separately in Byte array 01h:176-190 (see Table 8-51).

Note: When more than eight Applications need to be advertised, additional Application Descriptors can be stored in memory described in Table 8-52 and in the second half of memory described in Table 8-51.

All modules advertise at least one Application (so the first Application Descriptor is always used). The module indicates the end of the list of valid Application Descriptors by setting the HostInterfaceID field of the first unused Application Descriptor to a value of FFh.

Application Descriptor Fields

The **HostInterfaceID** indicates the interface standard describing the Host Interface of the Application. HostInterfaceID values and their meaning are specified in [5].

The **MediaInterfaceID** indicates the interface standard describing the Media Interface of the Application. MediaInterfaceID values and their meaning are specified in [5], depending on the module's Media Type as indicated in the **MediaType** field (Byte 00h:85).

The **MediaType** indicates the particular Media Interface Type table in [5] that applies to the module and hence the interpretation of MediaInterfaceID values. MediaType values are specified in Table 8-16.

The **HostLaneCount** and **MediaLaneCount** fields specify the number of lanes either explicitly (nonzero value) or by implicit reference, via the Interface ID, to the relevant interface specification (zero value).

The **HostLaneAssignmentOptions** register specifies which lane groups can be used for a Data Path carrying the advertised Application. Bits 0-7 form a bit map corresponding to Host Lanes 1-8. A bit value of 1 indicates that the lane group of the advertised Application can begin on the corresponding host lane. See section 6.2.1.4 for a more detailed description.

*Note: The **MediaLaneAssignmentOptions** parts of the Application Descriptor are stored separately.*

Both parts of the Application Descriptor are linked by the number known as AppSel Code. The AppSel code is simply the sequential number of the Application Descriptor in any of the memory locations storing (parts of) the Application Descriptor array.

Table 8-16 Media Type Encodings

Code	Media Type	Associated Interface ID Table
00h	Undefined	None, not applicable
01h	Optical Interfaces: MMF	[5] Table "850 nm MM media interface codes"
02h	Optical Interfaces: SMF	[5] Table "SM media interface codes"
03h	Passive Copper Cables	[5] Table "Passive Copper Cable interface codes"
04h	Active Cables	[5] Table "Active Cable assembly interface codes"
05h	BASE-T	[5] Table "BASE-T media interface codes"
06h-3Fh	-	**Reserved**
40h-8Fh	-	**Custom**
90h-FFh	-	**Reserved**

Table 8-17 Media Type Register (Lower Memory)

Byte	Bits	Register Name	Register Description	Type
85	7-0	MediaType	The MediaType field defines the interpretation of MediaInterfaceID values in the following Application Descriptors. See Table 8-16 for the MediaType encoding.	RO Rqd.

Table 8-18 Format of Application Descriptor Bytes 1-4

Byte Offset	Bits	Field Name	Field Description	Type
0	7-0	HostInterfaceID	ID from [5] or FFh to mark as unused (empty Application Descriptor)	RO Rqd.
1	7-0	MediaInterfaceID	ID from a suitable table of IDs in [5] that is identified by the MediaType Byte 00h:85 (see Table 8-16)	
2	7-4	HostLaneCount	0000b: lane count defined by interface ID 0001b: 1 lane, 0010b: 2 lanes …. 1000b: 8 lanes. 1001b-1111b: reserved	
	3-0	MediaLaneCount		
3	7-0	HostLaneAssignmentOptions	Bits 0-7 form a bit map corresponding to Host Lanes 1-8. A set bit indicates that the Application may begin on the corresponding host lane. *Note: Refer to section 6.2.1.4 for more information.*	

Note: The fifth bytes of these Application Descriptors are stored elsewhere (see Table 8-51)

Table 8-19 Application Descriptor Registers Bytes 1-4 (Lower Memory)

Byte	Bits	Field Name	Register Description	Type
86	7-0	HostInterfaceIDApp1	**AppDescriptor1** AppSel 1 (0001b) See Table 8-18	RO Rqd.
87	7-0	MediaInterfaceIDApp1		
88	7-4	HostLaneCountApp1		
	3-0	MediaLaneCountApp1		
89	7-0	HostLaneAssignmentOptionsApp1		
90	7-0	HostInterfaceIDApp2	**AppDescriptor2** AppSel 2 (0010b) See Table 8-18	RO Rqd.
91	7-0	MediaInterfaceIDApp2		
92	7-4	HostLaneCountApp2		
	3-0	MediaLaneCountApp2		
93	7-0	HostLaneAssignmentOptionsApp2		
94	7-0	HostInterfaceIDApp3	**AppDescriptor3** AppSel 3 (0011b) See Table 8-18	RO Cnd.
95	7-0	MediaInterfaceIDApp3		
96	7-4	HostLaneCountApp3		
	3-0	MediaLaneCountApp3		
97	7-0	HostLaneAssignmentOptionsApp3		
98	7-0	HostInterfaceIDApp4	**AppDescriptor4** AppSel 4 (0100b) See Table 8-18	RO Cnd.
99	7-0	MediaInterfaceIDApp4		
100	7-4	HostLaneCountApp4		
	3-0	MediaLaneCountApp4		
101	7-0	HostLaneAssignmentOptionsApp4		
102	7-0	HostInterfaceIDApp5	**AppDescriptor5** AppSel 5 (0101b) See Table 8-18	RO Cnd.
103	7-0	MediaInterfaceIDApp5		
104	7-4	HostLaneCountApp5		
	3-0	MediaLaneCountApp5		
105	7-0	HostLaneAssignmentOptionsApp5		
106	7-0	HostInterfaceIDApp6	**AppDescriptor6** AppSel 6 (0110b) See Table 8-18	RO Cnd.
107	7-0	MediaInterfaceIDApp6		
108	7-4	HostLaneCountApp6		
	3-0	MediaLaneCountApp6		
109	7-0	HostLaneAssignmentOptionsApp6		
110	7-0	HostInterfaceIDApp7	**AppDescriptor7** AppSel 7 (0111b) See Table 8-18	RO Cnd.
111	7-0	MediaInterfaceIDApp7		
112	7-4	HostLaneCountApp7		
	3-0	MediaLaneCountApp7		
113	7-0	HostLaneAssignmentOptionsApp7		
114	7-0	HostInterfaceIDApp8	**AppDescriptor8** AppSel 8 (1000b)	RO Cnd.
115	7-0	MediaInterfaceIDApp8		

116	7-4	HostLaneCountApp8	See Table 8-18	
	3-0	MediaLaneCountApp8		
117	7-0	HostLaneAssignmentOptionsApp8		

8.2.12 Password Entry and Change

The password entry and change facility described in this section provides a standardized **mechanism** to allow password protection of custom data or functionality, which itself are outside of the scope of this specification.

Note: An alternative password entry and password change mechanism providing feedback about success or failure is available via CDB commands CMD 0001h and CMD 0002h (see section 9.3).

Password protected custom functionality and password entry mechanism support is optional (there is no advertisement) but if it is supported, it shall conform to the following specifications.

Password protection of CMIS specified data or functionality is **prohibited** for CMIS compliant module, unless explicitly specified[1] or advertised[2].

The password entry and change facility uses 32-bit passwords (U32 values in Big-Endian hex notation).

A password entry register must be written using a size-matched four-byte WRITE access

Password entry registers are self-clearing; the module maintains the state of password protection internally.

In the password value range, two types of passwords are distinguished:

A **Host Password** value is in the range of 0000 0000h to 7FFF FFFFh. It can be changed by the host.

A **Module Password** value is in the range of 80000000h to FFFF FFFFh. It cannot be changed by the host.

Note: A host password may e.g. be used to protect module instance-specific data (e.g. inventory data) that are under control of the module user. A module password may e.g. be used by module vendors to grant access to custom features only to those hosts who have been given a certain module password.

The factory default of a Host Password is 0000 1011h.

The factory default of a Module Password is defined by the module vendor.

The host can change a current Host Password value by writing a new Host Password value into the Password Change Entry Area (Bytes 118-121) after the correct current Host Password value has been entered into the Password Entry Area (Bytes 122-125). The effect of writing a Module Password value is undefined.

Password-protected features are unlocked when a valid password is entered and remain unlocked until either an invalid password is entered or until the module is reinitialized.

The initial internal state of password protection when the module exits the MgmtInit state shall be locked.

Table 8-20 Password Change Entry

Byte	Bits	Register Name	Register Description	Type
118-121	7-0	PasswordChangeEntryArea	U32: new password value	WO/SC Opt.
122-125	7-0	PasswordEntryArea	U32: password value	WO/SC Opt.

[1] The non-volatile module data in Page 03h may be password protected

[2] In this version of CMIS, there are no such advertisements.

8.2.13 Page Mapping (Upper Memory Content Selection)

The **PageMapping** register (00h:126-127) is a **two-byte** register containing a **Page Address** value that determines which register in the internal Management Memory Map is **actually** accessed when the host performs an ACCESS addressing a Byte in Upper Memory (Byte address range 128-255).

Note: This programmable redirection from a fixed Upper Memory address window to a selected Page in the Management Memory Map is referred to as Page Mapping. Note that CMIS is agnostic of how a module actually implements or emulates page mapping.

The **PageMapping** register has two components storing the two components of a **Page Address**:

- The **BankSelect** Byte (00h:126) storing a **Bank Index**
- The **PageSelect** Byte (00h:127) storing a **Page Index**

Note: Beware of confusion – a Page Address is a two-dimensional value that uniquely identifies a Page, whereas a Page Index is a one-dimensional value identifying a set of Pages with the same Bank Index.

The Management Memory Map (see Table 8-1) defines **banked** Pages (with the potential of Bank Support) and **unbanked** Pages (without Bank Support), whereby the Bank Index of unbanked Pages is irrelevant (albeit using a nominal value of 0 is recommended).

Note: See section 8.1 for background on the host accessible Management Memory Map, on the split of host addressable memory into LowerMemory and UpperMemory, and on Banking.

For an arbitrary Page Address change or for just a Bank Index change, a host must write both BankSelect and PageSelect in one WRITE access, even if the PageSelect value does not change. The module does not begin processing the BankSelect value until after the PageSelect register has been written.

For just a Page Index change (mapping another Page in the current Bank), or for mapping an arbitrary unbanked Page to Upper Memory, a host may WRITE only the PageSelect Byte.

Table 8-21 Page Mapping Register Components

Byte	Bits	Register Name	Field Description	Type
126	7-0	BankSelect	**Bank Index** of Page mapped to Upper Memory (if applicable). The current BankSelect value determines which Bank of a Page is accessed (for Pages with Banking) when a host ACCESS addresses a Byte in Upper Memory (address 128 through 255). Ignored when the Page indexed by PageSelect is unbanked.	RW Cnd.
127	7-0	PageSelect	**Page Index** of Page mapped to Upper Memory The current PageSelect value determines which Page (or Page in a Bank) is accessed when a host ACCESS addresses a Byte in Upper Memory (address 128 through 255) *Note: The module may clear the PageSelect to prevent mapping an unsupported page.*	RWW Cnd.

PageMapping Validity

The module ensures that the **PageMapping** register always contains a **Page Address** that is **actually** supported by the module: When a host write would result in a not supported Page Address in the PageMapping register, the module **clears the PageSelect** Byte (without clearing the BankSelect Byte), such that the resulting **PageMapping** register selects Page 00h (as the BankSelect value is ignored unbanked Page 00h).

Note: This is a deliberate exception from the usual rule that either host or module modify a field, but not both.

Stale Data Access Prevention during Page Remapping

The module may need time to effectively switch the content of Upper Memory (i.e. to effectively map a new Page) after a change of the PageMapping register.

While performing the page switch, when an ACCESS to Upper Memory would or could access stale data, the module rejects such ACCESS, as described in section 5.2.3, using methods described in Appendix B. For simplicity, the module may at the same time also reject ACCESS to Lower Memory.

The maximum time allowed for the module to complete a page switch is specified in section 10.2.2 and applies both to supported and to non-supported values, as well as to the case where the register values do not change.

8.3 Page 00h (Administrative Information)

Page 00h contains static read-only module characteristic information.

Page 00h is supported by all modules (including cable assemblies).

Table 8-22 Page 00h Overview

Address	Size (bytes)	Subject Area or Field	Description
128	1	SFF8024IdentifierCopy	Copy of Byte 00h:0
129-144	16	VendorName	Vendor name (ASCII)
145-147	3	VendorOUI	Vendor IEEE company ID
148-163	16	VendorPN	Part number provided by vendor (ASCII)
164-165	2	VendorRev	Revision level for part number provided by vendor (ASCII)
166-181	16	VendorSN	Vendor Serial Number (ASCII)
182-189	8	DateCode	Manufacturing Date Code (ASCII)
190-199	10	CLEICode	Common Language Equipment Identification Code (ASCII)
200-201	2	ModulePowerCharacteristics	Module power characteristics
202	1	CableAssemblyLinkLength	Cable length (for cable assembly modules only)
203	1	ConnectorType	Connector type of the media interface
204-209	6	Copper Cable Attenuation	Attenuation characteristics (passive copper cables only)
210	1	MediaLaneInformation	Supported near end media lanes (all modules)
211	1	Cable Assembly Information	Far end modules information (cable assemblies only)
212	1	MediaInterfaceTechnology	Information on media side device or cable technology
213-220	8	-	**Reserved[8]**
221	1	-	**Custom[1]**
222	1	PageChecksum	Page Checksum over bytes 128-221
223-255	33	-	**Custom[33]** Information (**non-volatile**)

8.3.1 SFF-8024 Identifier Copy

This field shall contain the same value as Byte 00h:0.

Note: This duplication requirement is maintained for historical reasons and for similarity with predecessor SFF MIS specifications.

Table 8-23 SFF8024IdentifierCopy (Byte 00h:128)

Byte	Bits	Name	Description	Type
128	7-0	SFF8024Identifier	This Byte shall contain the same value as the SFF8024 Identifier Byte 00h:0. See Byte 00h:0 for a description of its meaning.	RO Rqd.

8.3.2 Vendor Information

Table 8-24 Vendor Information (Page 00h)

Bytes	Length (bytes)	Register Name	Register Description	Type
129-144	16	VendorName	Vendor name (ASCII)	RO Rqd.
145-147	3	VendorOUI	Vendor IEEE company ID	RO Rqd.
148-163	16	VendorPN	Part number provided by vendor (ASCII)	RO Rqd.
164-165	2	VendorRev	Revision level for part number provided by vendor (ASCII)	RO Rqd.
166-181	16	VendorSN	Vendor Serial Number (ASCII)	RO Rqd.
182-189	8	DateCode	Manufacturing Date Code (ASCII)	RO Rqd.
190-199	10	CLEICode	Common Language Equipment Identification Code (ASCII)	RO Rqd.

8.3.2.1 Vendor Name

The **VendorName** is a 16 character read-only field that contains ASCII characters, left aligned and padded on the right with ASCII spaces (20h).

The VendorName shall contain the full name of the corporation, a commonly accepted abbreviation of the name of the corporation, the SCSI company code for the corporation, or the stock exchange code for the corporation. The VendorName may be the original manufacturer of the module or the name of the module reseller. In both cases, the VendorName and VendorOUI (if specified) shall correlate to the same company. At least one of the vendor name or the vendor OUI fields shall contain valid serial number manufacturing data.

8.3.2.2 Vendor Organizationally Unique Identifier

The vendor organizationally unique identifier field (**VendorOUI**) is a 3-byte field that contains the IEEE Company Identifier for the vendor. A value of all zero in the 3-byte field indicates that the vendor's OUI is unspecified.

8.3.2.3 Vendor Part Number

The vendor part number (**VendorPN**) is a 16-byte field that contains ASCII characters, left aligned and padded on the right with ASCII spaces (20h), defining the vendor part number or product name. A value of all zero in the 16-byte field indicates that the vendor part number is unspecified.

8.3.2.4 Vendor Revision Number

The vendor revision number (**VendorRev**) is a 2-byte field that contains ASCII characters, left aligned and padded on the right with ASCII spaces (20h), defining the vendor's product revision number. A value of all zero in the field indicates that the vendor revision number is unspecified.

8.3.2.5 Vendor Serial Number

The vendor serial number (**VendorSN**) is a 16-character field that contains ASCII characters, left aligned and padded on the right with ASCII spaces (20h), defining the vendor's serial number for the Product. A value of all zero in the 16-byte field indicates that the vendor serial number is unspecified.

8.3.2.6 Date Code

The **DateCode** is an 8-byte field that contains the vendor's date code in ASCII characters. The date code is mandatory. The date code shall be in the following format:

Table 8-25 Date Code (Page 00h)

Byte	Bits	Register Name	Description	Type
182-183	All	Year	ASCII two low order digits of year (00=2000)	RO Rqd.
184-185	All	Month	ASCII digits of month (01=Jan through 12=Dec)	RO Rqd.
186-187	All	DayOfMonth	ASCII day of month (01-31)	RO Rqd.
188-189	All	LotCode	ASCII custom lot code, may be blank	RO Opt.

8.3.2.7 CLEI Code

The **CLEI** (Common Language Equipment Identification) code is a 10-byte field that contains the vendor's CLEI code in ASCII characters.

The CLEI code value is optional. If CLEI code value is not supported, a value of all ASCII 20h (spaces) shall be entered.

Table 8-26 CLEI Code (Page 00h)

Byte	Bits	Register Name	Description	Type
190-199	All	CLEICode	Vendor's CLEI Code (ASCII)	RO Opt.

8.3.3 Module Power Characteristics

Module power characteristics are advertised in Bytes 00h:200 and 00h:201 (see Table 8-27).

The **MaxPower** field specifies worst case maximum power consumption over operating conditions and lifetime.

The **ModulePowerClass** field provides a form factor specific classification of the MaxPower value.

Note: The preferred method for the host to verify if the maximum power consumption is acceptable in the host system, before allowing the module to enter High Power Mode, is to use the MaxPower field. See section 6.3.2.4.

Note: The host may also use the ModulePowerClass field to determine if the maximum power consumption is acceptable in the host system, but this indirect and form factor dependent method is not preferred.

Table 8-27 Module Power Class and Max Power (Page 00h)

Byte	Bits	Field Name	Field Description	Type
200	7-5	ModulePowerClass[1]	000: Power class 1 001: Power class 2 010: Power class 3 011: Power class 4 100: Power class 5 101: Power class 6 110: Power class 7 111: Power class 8	RO Rqd.
	4-0	-	**Reserved**	RO
201	7-0	MaxPower	Maximum power consumption in multiples of 0.25 W rounded up to the next whole multiple of 0.25 W	RO Rqd.

Note 1: See relevant hardware specification for maximum power allowed in each Power class

8.3.4 Cable Assembly Link Length

The **CableAssemblyLinkLength** Byte 00h:202 advertises the physical interconnect length of cable assemblies (including both passive copper cables and active optical or electrical cables).

Modules with separable optical media shall set the CableAssemblyLinkLength value to 0000 0000b.

A CableAssemblyLinkLength value of 1111 1111b indicates a link length greater than 6300 m.

Table 8-28 Cable Assembly Link Length

Byte	Bits	Field Name	Field Description	Type
202	7-6	LengthMultiplier	Multiplier for value in bits 5-0: 00: multiplier 0.1 01: multiplier 1 10: multiplier 10 11: multiplier 100	RO Rqd.
	5-0	BaseLength	Link length base value in meters. To calculate actual link length use multiplier in bits 7-6. A value of 0 indicates an undefined Link Length, e.g. when the physical media can be disconnected from the module.	

8.3.5 Media Connector Type

The **ConnectorType** field indicates the connector type for the media side of the module, as defined and maintained in the Connector References section of [5].

Table 8-29 Media Connector Type

Byte	Bits	Register Name	Description	Type
203	7-0	ConnectorType	Type of connector present in the module. See [5] for Connector Type codes.	RO Rqd.

8.3.6 Copper Cable Attenuation

These Bytes advertise the cable attenuation characteristics for passive copper cables.

For other modules bytes 204-209 are **reserved**.

Table 8-30 Copper Cable Attenuation (Page 00h)

Byte	Bits	Register Name	Register Description	Type
204	7-0	AttenuationAt5GHz	U8 Passive copper cable attenuation at 5 GHz in 1 dB increments	RO Cnd.
205	7-0	AttenuationAt7GHz	U8 Passive copper cable attenuation at 7 GHz in 1 dB increments	RO Cnd.
206	7-0	AttenuationAt12p9GHz	U8 Passive copper cable attenuation at 12.9 GHz in 1 dB increments	RO Cnd.
207	7-0	AttenuationAt25p8GHz	U8 Passive copper cable attenuation at 25.8 GHz in 1 dB increments	RO Cnd.
208-209	All	-	**Reserved[2]**	RO

8.3.7 Media Lane Information

The module indicates which Media Lanes are not supported (see Table 8-32).

Note: An optical media lane may consist of fibers or wavelengths.

Table 8-31 Media Lane Information (Page 00h)

Byte	Bits	Field Name	Field Description	Type
210	7	MediaLaneUnsupportedLane8	Bool: **MediaLaneUnsupportedLane<i>** 0b: Media Lane <i> supported 1b: Media Lane <i> not supported	RO Rqd.
	6	MediaLaneUnsupportedLane7		
	5	MediaLaneUnsupportedLane6		
	4	MediaLaneUnsupportedLane5		
	3	MediaLaneUnsupportedLane4		
	2	MediaLaneUnsupportedLane3		
	1	MediaLaneUnsupportedLane2		
	0	MediaLaneUnsupportedLane1		

8.3.8 Cable Assembly Lane Information

For Cable Assemblies, the **FarEndConfiguration** Byte (see Table 8-33) indicates the number of far end lanes supported and the far end lane configuration.

For modules with detachable media connectors the FarEndConfiguration byte is cleared.

Table 8-32 Cable Assembly Information (Page 00h)

Byte	Bits	Field Name	Field Description	Type
211	7-5	-	**Reserved**	RO
	4-0	FarEndConfiguration	See Table 8-33 for configuration codes	RO Cnd.

Table 8-33 contains codes for all near end supported combinations of far end modules (lane groups a to h), and maps those far end modules (lane groups) to the connected near end lane number. Unique letters indicate discrete far end modules. Note that the discrete far end modules may or may not be the same module type.

Table 8-33 Far end cable lane groups advertising codes (Page 00h)

Far End Cable Lane Groups Advertising Codes									
Config Code		Near End Host Lane Number							
Decimal	Binary	1	2	3	4	5	6	7	8
0	00000b	Undefined. Module with detachable media.							
1	00001b	a	b	c	d	e	f	g	h
2	00010b	a	a	a	a	a	a	a	a
3	00011b	a	a	a	a	e	e	e	e
4	00100b	a	b	c	d	e	e	e	e
5	00101b	a	b	c	c	e	e	e	e

6	00110b	a	a	c	d	e	e	e	e
7	00111b	a	a	c	c	e	e	e	e
8	01000b	a	a	a	a	e	f	g	h
9	01001b	a	a	a	a	e	f	g	g
10	01010b	a	a	a	a	e	e	g	h
11	01011b	a	a	a	a	e	e	g	g
12	01100b	a	a	c	c	e	e	g	g
13	01101b	a	b	c	c	e	e	g	g
14	01110b	a	a	c	d	e	e	g	g
15	01111b	a	b	c	d	e	e	g	g
16	10000b	a	a	c	c	e	f	g	g
17	10001b	a	b	c	c	e	f	g	g
18	10010b	a	a	c	d	e	f	g	g
19	10011b	a	b	c	d	e	f	g	g
20	10100b	a	a	c	c	e	e	g	h
21	10101b	a	b	c	c	e	e	g	h
22	10110b	a	a	c	d	e	e	g	h
23	10111b	a	b	c	d	e	e	g	h
24	11000b	a	a	c	c	e	f	g	h
25	11001b	a	b	c	c	e	f	g	h
26	11010b	a	a	c	d	e	f	g	h
27-31	11011b-11111b	reserved							

8.3.9 Media Interface Technology

The **MediaInterfaceTechnology** Byte 00h:212 classifies the media interface device or cable technology, using the encodings in Table 8-35.

An active optical cable may be distinguished from a separable module by examining the **ConnectorType** field Byte 00h:203 (see section 8.3.5), with values defined in the Connector References section of [5].

Table 8-34 Media Connector Type

Byte	Bits	Register Name	Register Description	Type
212	7-0	MediaInterfaceTechnology	Media Interface Technology as per Table 8-35	RO Rqd.

Table 8-35 Media Interface Technology encodings

Code	Description of physical device
00h	850 nm VCSEL
01h	1310 nm VCSEL
02h	1550 nm VCSEL
03h	1310 nm FP
04h	1310 nm DFB
05h	1550 nm DFB
06h	1310 nm EML
07h	1550 nm EML
08h	Others
09h	1490 nm DFB
0Ah	Copper cable unequalized
0Bh	Copper cable passive equalized
0Ch	Copper cable, near and far end limiting active equalizers
0Dh	Copper cable, far end limiting active equalizers

0Eh	Copper cable, near end limiting active equalizers
0Fh	Copper cable, linear active equalizers
10h	C-band tunable laser
11h	L-band tunable laser
12h-FFh	Reserved

8.3.10 Page 00h Page Checksum (required)

The page checksum is a one-byte code that can be used to verify that the read-only static data on Page 00h is valid. The page checksum value shall be the low order 8 bits of the arithmetic sum of all byte values from byte 128 to byte 221, inclusive.

8.3.11 Custom Info (non-volatile)

Bytes 223-255 are allocated in **non-volatile** storage for information provided by the original manufacturer of the module or the module reseller. This information persists across module reset and power cycle.

The contents of this area are not defined by this specification.

8.4 Page 01h (Advertising)

Page 01h is an optional Page containing advertising fields for properties of paged memory modules.

The module advertises support of Page 01h in the MemoryModel Bit 00h:2.7.

Note: Page 01h is mandatory for paged memory modules.

All fields on Page 01h are read-only and static.

See the following subsections for detailed information on the subject areas listed in Table 8-36

Table 8-36 Page 01h Overview

Byte	Size (bytes)	Subject Area	Description
128-131	4	Inactive Firmware and Hardware revisions	Inactive FW revision and HW revision
132-137	6	Supported link length	Supported lengths of various fiber media
138-141	4	Wavelength Information	(for single wavelength modules)
142	1	Supported Pages	
143-144	2	Durations Advertisements	
145-154	10	Module Characteristics	
155-156	2	Supported Controls	
157-158	2	Supported Flags	
159-160	2	Supported Monitors	
161-162	2	Supported Signal Integrity Controls	
163-166	4	Supported CDB Functionality	
167-169	3	Additional Durations Advertisements	
170-175	7	-	**Reserved[7]**
176-190	15	Media Lane advertising	
191-222	32	-	**Custom[32]**
223-250	28	Additional Application Descriptors	
251-254	4	-	**Reserved[4]**
255	1	Page Checksum	Page Checksum of bytes 130-254[1]

Note 1: The firmware version bytes 128-129 are intentionally excluded from the Page Checksum to avoid requiring a Memory Map update when firmware is updated.

8.4.1 Inactive Firmware and Hardware Revisions

Table 8-37 describes the fields for reporting the module's inactive firmware revision (if any) and the module's hardware revision.

Note: The active firmware revision is reported in the Module Active Firmware Version fields (see section 8.2.9) and the overall module revision number is reported in the Vendor Revision Number field (see section 8.3.2.4).

The inactive firmware is that firmware stored that is not currently executing, when the module supports a second firmware image.

Note: This inactive firmware image may be an alternate or backup firmware image, or a new firmware image downloaded but not yet activated.

The ModuleInactiveFirmwareMajorRevision and ModuleInactiveFirmwareMinorRevision fields contain numbers with the same encoding as the ModuleActiveFirmwareMajorRevision and ModuleActiveFirmwareMinorRevision fields (see section 8.2.9): a module without inactive firmware clears these fields.

Bytes 01h:128-29 are not included in the Page 01h Page Checksum as these bytes may change dynamically for modules that support switching firmware version between multiple images during firmware updates.

The numeric ModuleHardwareMajorRevision and ModuleHardwareMinorRevision fields contain version numbers. These two fields are included in the Page 01h Page Checksum.

Table 8-37 Module Inactive Firmware and Hardware Revisions (Page 01h)

Byte	Bits	Register Name	Register Description	Type
128	7-0	ModuleInactiveFirmwareMajorRevision	U8 Numeric representation of module inactive firmware major revision	RO Rqd.
129	7-0	ModuleInactiveFirmwareMinorRevision	U8 Numeric representation of module inactive firmware minor revision	RO Rqd.
130	7-0	ModuleHardwareMajorRevision	U8 Numeric representation of module hardware major revision	RO Rqd.
131	7-0	ModuleHardwareMinorRevision	U8 Numeric representation of module hardware minor revision	RO Rqd.

Note: Modules that support a proprietary vendor specific firmware update method with two firmware images (instead of CDB as described later) shall still report inactive firmware version in bytes 01h:128-129.

8.4.2 Supported Link Length

The Bytes described in Table 8-38 advertise the maximum supported fiber media length for each type of fiber media at the maximum module-supported bit rate for modules with a separable optical media interface. Unsupported media types shall be populated with zeroes.

Active optical cables shall populate the fields in this table with zeroes and instead report their actual length using the fields in Table 8-28.

Table 8-38 Supported Fiber Link Length (Page 01h)

Byte	Bits	Name	Description	Type
132	7-6	LengthMultiplierSMF	Link length multiplier for SMF fiber 00 = 0.1 (0.1 to 6.3 km) 01 = 1 (1 to 63 km) 10 = 10 (10 to 630 km) 11 = reserved	RO Rqd.
	5-0	BaseLengthSMF	Base link length for SMF fiber in km. Must be multiplied by multiplier defined in bits 7-6 to calculate actual link length.	RO Rqd.
133	7-0	LengthOM5	Link length supported for OM5 fiber, units of 2 m (2 to 510 m)	RO Rqd.
134	7-0	LengthOM4	Link length supported for OM4 fiber, units of 2 m (2 to 510 m)	RO Rqd.
135	7-0	LengthOM3	Link length supported for EBW 50/125 µm fiber (OM3), units of 2m (2 to 510 m)	RO Rqd.
136	7-0	LengthOM2	Link length supported for 50/125 µm fiber (OM2), units of 1m (1 to 255 m)	RO Rqd.
137	7-0	-	**Reserved[1]**	RO

The link length supported for SMF fiber specifies the link length that is supported by the device while operating in compliance with the applicable standards using single mode fiber. The supported link length is as specified in [7]. The value is in units of kilometers.

The link length supported for OM5 fiber specifies the link length that is supported by the device while operating in compliance with the applicable standards using 4700 MHz*km (850 nm) and 2470 MHz*km (953 nm) extended bandwidth 50 micron core multimode fiber. The value is in units of two meters.

The link length supported for OM4 fiber specifies the link length that is supported by the device while operating in compliance with the applicable standards using 4700 MHz*km (850 nm) extended bandwidth 50 micron core multimode fiber. The value is in units of two meters.

The link length supported for OM3 fiber specifies the link length that is supported by the device while operating in compliance with the applicable standards using 2000 MHz*km (850 nm) extended bandwidth 50 micron core multimode fiber. The value is in units of two meters.

The link length supported for OM2 fiber specifies the link length that is supported by the device while operating in compliance with the applicable standards using 500 MHz*km (850 nm and 1310 nm) 50 micron multi-mode fiber. The value is in units of one meter.

8.4.3 Wavelength Information

The **NominalWavelength** and **WavelengthTolerance** fields are defined for single wavelength modules.

Note: Nominal wavelength and wavelength tolerance are attributes of optical media lanes, whereas NominalWavelength and WavelengthTolerance are module level fields.

Note: All optical modules, whether single or multiple wavelengths, advertise their supported media specifications indirectly via the MediaInterfaceID of the relevant Application. In many of these Applications, fixed nominal wavelengths and maximum tolerances are known from the associated standard.

A single wavelength module with implicitly defined fixed wavelength and tolerance reports these specified values or more specific actual values (e.g. tighter tolerance).

A single wavelength module with (directly or indirectly) programmable wavelength reports actual nominal wavelength and actual wavelength tolerance.

A multi-wavelength module may optionally provide wavelength information, either for one of the wavelengths or for the entire wavelength range (as nominal center wavelength and overall tolerance). Since the interpretation is not uniquely defined, a host may ignore this field for multi-wavelength modules and use the advertised Application to determine module capabilities.

Table 8-39 Wavelength Information (Page 01h)

Byte	Bits	Register Name	Register Description	Type
138-139	7-0	NominalWavelength	U16 nominal transmitter output wavelength for a single wavelength module at room temperature in units of 0.05nm	RO Cnd.
140-141	7-0	WavelengthTolerance	U16 wavelength tolerance **tol** as the worst case +/-**tol** range around the NominalWavelength under all normal operating conditions in units of 0.005nm	RO Cnd.

Example 1 (Single Wavelength Module):
ITU-T Grid Wavelength = 1534.25 nm with 0.236 nm Tolerance
Nominal Wavelength = 1534.25 nm, represented as U16: (1534.25 nm * 20) = 30685
Wavelength Tolerance = 0.236 nm, represented as U16: (0.236 nm * 200) = 47

Example 2 (Multi-Wavelength Module):
100GBASE-LR4 Wavelength Range = 1294.53 to 1310.19 nm
Nominal Wavelength = 1302.36 nm, represented as U16: (1302.36 nm * 20) = 26047
Wavelength Tolerance = 7.83 nm, represented as U16: (7.83 nm * 200) = 1566

8.4.4 Supported Pages Advertising

Table 8-40 Supported Pages Advertising (Page 01h)

Byte	Bit	Field Name	Field Description	Type
142	7	-	**Reserved**	RO Rqd.
	6	VDMPagesSupported	VDM Pages 20h-2Fh (partially) supported (advertisement details in Page 2Fh)	RO Rqd.
	5	DiagnosticPagesSupported	Banked Page 13h-14h supported	RO Rqd.
	4	-	**Reserved**	
	3	-	**Reserved**	
	2	Page03hSupported	User Page 03h supported	
	1-0	BanksSupported [1]	Banks supported for Pages 10h-1Fh **00b:** Bank 0 supported (8 lanes) **01b:** Banks 0 and 1 supported (16 lanes) **10b:** Banks 0-3 supported (32 lanes) 11b: reserved	

Note 1: For response to hosts using an invalid Bank Page combination see section 8.2.13.

8.4.5 Durations Advertising

The **ModSelWaitTime*** fields define both the required **setup time** (for the ModSel signal after the host asserts ModSel and before the start of an MCI bus transaction) and the required **delay** (after completion of an MCI transaction before the host can deassert the ModSel signal).

For example, if the module wait time is 1.6 ms, the mantissa field (bits 4-0) will be 11001b (25) and the exponent field (bits 7-5) will be 110b (6) for a result of $25\cdot 2^6$ *= 11001000000b or 1600 us decimal.*

The **MaxDurationDPInit** and **MaxDurationDPDeinit** fields are defined so that hosts can determine when something failed in the module during these states, for example a module firmware hang up.

These maximum duration values represent worst-case durations across all advertised Applications and all possible combinations of Data Paths.

See sections 6.3.3.5 and 6.3.3.7 for details of the DPInit and DPDeinit states, respectively.

See section 8.4.12 for other maximum duration advertisements.

Table 8-41 Durations Advertising (Page 01h)

Byte	Bit	Field Name	Field Description	Type
143	7-5	ModSelWaitTimeExponent **e**	ModSel wait time as $\mathbf{m}\cdot 2^{e}$ in µs 00h: no data available	RO Rqd.
	4-0	ModSelWaitTimeMantissa **m**		
144	7-4	MaxDurationDPDeinit	Maximum duration of the DPDeinit state (encoded as per Table 8-42)	RO Rqd.
	3-0	MaxDurationDPInit	Maximum duration of the DPInit state (encoded as per Table 8-42)	

Table 8-42 State Duration Encoding (Page 01h)

Encoding	Maximum State Duration $\mathbf{T}_{\text{state}}$
0000b	$\mathbf{T}_{\text{state}}$ < 1 ms
0001b	1 ms <= $\mathbf{T}_{\text{state}}$ < 5 ms
0010b	5 ms <= $\mathbf{T}_{\text{state}}$ < 10 ms
0011b	10 ms <= $\mathbf{T}_{\text{state}}$ < 50 ms
0100b	50 ms <= $\mathbf{T}_{\text{state}}$ < 100 ms
0101b	100 ms <= $\mathbf{T}_{\text{state}}$ < 500 ms
0110b	500 ms <= $\mathbf{T}_{\text{state}}$ < 1 s
0111b	1 s <= $\mathbf{T}_{\text{state}}$ < 5 s
1000b	5 s <= $\mathbf{T}_{\text{state}}$ < 10 s
1001b	10 s <= $\mathbf{T}_{\text{state}}$ < 1 min
1010b	1 min <= $\mathbf{T}_{\text{state}}$ < 5 min
1011b	5 min <= $\mathbf{T}_{\text{state}}$ < 10 min
1100b	10 min <= $\mathbf{T}_{\text{state}}$ < 50 min
1101b	$\mathbf{T}_{\text{state}}$ >= 50 min
1110b	Reserved
1111b	Reserved

8.4.6 Module Characteristics Advertising

The fields in Table 8-43 describe the characteristics of certain module properties. Some features are optional. Advertisement of the implementation of optional features is described in sections 8.4.7 through 8.4.10.

A **Tx synchronous group** is defined as a Tx input lane or group of Tx input lanes sourced from the same clock domain. Two different Tx synchronous groups may be sourced from different clock domains. There may be a limit on the maximum permissible clock tolerance between two different Tx synchronous groups, as defined by the industry standard associated with a given Application. Refer to applicable industry standards.

A Tx synchronous group can contain one or more Data Paths, if the Tx lanes on all Data Paths are sourced from the same clock domain and the module takes measures to ensure that active Data Paths continue to operate undisturbed even as other Data Paths (and their associated Tx input lanes) are enabled/disabled by the host.

Table 8-43 Module Characteristics Advertising (Page 01h)

Byte	Bit	Field Name	Field Description	Type
145	7	CoolingImplemented	0b: Uncooled transmitter device 1b: Cooled transmitter	RO Rqd.
	6-5	TxInputClockingCapabilities	Defines which Tx input lanes must be frequency synchronous 00b: Tx input lanes 1-8 01b: Tx input lanes 1-4 and 5-8 10b: Tx input lanes 1-2, 3-4, 5-6, 7-8 11b: Lanes may be asynchronous in frequency When more than 8 lanes are supported, the above grouping applies to each group of 8 lanes (each Bank)	RO Rqd.
	4	ePPSSupported	Support of the Enhanced Pulse Per Second timing signal [2] 0b/1b: ePPS signal processing not supported/supported	RO Rqd.
	3	TimingPage15hSupported	0b: Timing characteristics (Page 15h) not supported 1b: Timing characteristics (Page 15h) supported	RO Rqd.
	2	Aux3MonObservable	0b: Aux 3 monitor monitors Laser Temperature 1b: Aux 3 monitor monitors Vcc2	RO Adv.
	1	Aux2MonObservable	0b: Aux 2 monitor monitors Laser Temperature 1b: Aux 2 monitor monitors TEC current	RO Adv.
	0	Aux1MonObservable	0b: Aux 1 monitor is **custom** 1b: Aux 1 monitor monitors TEC current	RO Adv.
146	7-0	ModuleTempMax	S8 Maximum allowed module case temperature in 1 deg C increments. ModuleTempMax = ModuleTempMin = 0 indicates 'not specified'.	RO Cnd.
147	7-0	ModuleTempMin	S8 Minimum allowed module case temperature in 1 deg C increments. ModuleTempMax = ModuleTempMin = 0 indicates 'not specified'.	RO Cnd.
148-149	7-0	PropagationDelay	U16 Propagation delay of a non-separable AOC in multiples of 10 ns rounded to the nearest 10 ns, or zero for 'not specified'.	RO Cnd.
150	7-0	OperatingVoltageMin	U8 Minimum supported module operating voltage, in 20 mV increments (0-5.1 V), or zero for 'not specified'.	RO Cnd.
151	7	OpticalDetectorType	0b: PIN detector 1b: APD detector	RO Rqd.
	6-5	RxOutputEqType	00b: Peak-to-peak (p-p) amplitude stays constant, or not implemented, or no information 01b: Steady-state amplitude stays constant 10b: Average of p-p and steady-state amplitude stays constant 11b: Reserved	
	4	RxPowerMeasurementType	0b: OMA 1b: average power	
	3	RxLOSType	0b: Rx LOS responds to OMA 1b: Rx LOS responds to P_{av} *Note: LOS Type depends on interface standards supported*	
	2	RxLOSIsFast	0b: Module raises Rx LOS within regular timing limits 1b: Module raises Rx LOS within "fast mode" timing limits	

Byte	Bit	Field Name	Field Description	Type
			Refer to form factor hardware specification for regular and "fast mode" timing limit requirements	
	1	TxDisableIsFast	0b: Module responds to Tx Output Disable with regular timing 1b: Module responds to Tx Output Disable in "fast mode" timing limits Refer to form factor hardware specification for regular and "fast mode" timing limit requirements	
	0	TxDisableIsModuleWide	0b: Tx output disable is controlled per lane 1b: All Tx output lanes disabled when any OutputDisableTx set	
152	7-0	CDRPowerSavedPerLane	U8 Minimum power consumption saved per CDR per lane when placed in CDR bypass, in multiples of 0.01 W rounded up to the next whole multiple of 0.01 W	RO Cnd.
153	7	RxOutputLevel3Supported [1]	0b/1b: Amplitude Code 3 not supported/supported	RO Cnd.
	6	RxOutputLevel2Supported [1]	0b/1b: Amplitude Code 2 not supported/supported	
	5	RxOutputLevel1Supported [1]	0b/1b: Amplitude Code 1 not supported/supported	
	4	RxOutputLevel0Supported [1]	0b/1b: Amplitude Code 0 not supported/supported	
	3-0	TxInputEqMax	Maximum supported value of the Tx Input Equalization control for manual/fixed programming (see section 6.2.5.1)	
154	7-4	RxOutputEqPostCursorMax	Maximum supported value of the Rx Output Eq Post-cursor control (see section 6.2.5.2)	RO Cnd.
	3-0	RxOutputEqPreCursorMax	Maximum supported value of the Rx Output Eq Pre-cursor control (see section 6.2.5.2)	

Note 1: See Table 6-8

8.4.7 Supported Controls Advertisement

Table 8-44 describes supported module and lane controls and related module functions (see section 8.8.2).

Table 8-44 Supported Controls Advertisement (Page 01h)

Byte	Bit	Field Name	Field Description	Type
155	7	WavelengthIsControllable	0b: No wavelength control 1b: Active wavelength control supported *Note: Active Control does not imply tunability*	RO Rqd.
	6	TransmitterIsTunable	0b: Transmitter not tunable 1b: Transmitter is tunable (Pages 04h and 12h supported)	
	5-4	SquelchMethodTx	00b: Tx output squelching function is not supported 01b: Tx output squelching function reduces OMA 10b: Tx output squelching function reduces P_{av} 11b: Host controls the method for Tx output squelching, reducing OMA or P_{av} (see Table 8-10)	
	3	ForcedSquelchTxSupported	0b/1b: Host cannot/can force squelching of Tx outputs using OutputSquelchForceTx*	
	2	AutoSquelchDisableTxSupported	0b/1b: Host cannot/can disable automatic squelching of Tx outputs using AutoSquelchDisableTx*	
	1	OutputDisableTxSupported	0b/1b: Host cannot/can disable Tx outputs using the OutputDisableTx register	
	0	InputPolarityFlipTxSupported	0b/1b: InputPolarityFlipTx control not supported/supported	
156	7-3	-	**Reserved**	RO Rqd.
	2	AutoSquelchDisableRxSupported	0b/1b: Host cannot/can disable automatic squelching of Rx outputs using AutoSquelchDisableRx*	RO Rqd.
	1	OutputDisableRxSupported	0b/1b: Host cannot/can disable Rx outputs using the OutputDisableRx register	
	0	OutputPolarityFlipRxSupported	0b/1b: PolarityFlipRx not supported/supported	

8.4.8 Supported Flags Advertisement

Table 8-45 describes supported module and lane Flags (see section 8.9.3).

Table 8-45 Supported Flags Advertisement (Page 01h)

Byte	Bit	Field Name	Field Description	Type
157	7-4	-	**Reserved**	RO
	3	AdaptiveInputEqFailFlagTxSupported	0b: Tx Adaptive Input Eq Fail Flags not supported 1b: supported	RO Rqd.
	2	CDRLOLFlagTxSupported	0b: Tx CDR Loss of Lock Flags not supported 1b: supported	
	1	LOSFlagTxSupported	0b: Tx Loss of Signal Flags not supported 1b: supported	
	0	FailureFlagTxSupported	0b: Tx Fault Flags not supported 1b: supported	
158	7-3	-	**Reserved**	RO
	2	CDRLOLFlagRxSupported	0b: Rx CDR Loss of Lock Flags not supported 1b: Rx CDR Loss of Lock Flags supported	RO Rqd.
	1	LOSFlagRxSupported	0b: Rx Loss of Signal Flags not supported 1b: Rx Loss of Signal Flags supported	
	0	-	**Reserved**	RO

8.4.9 Supported Monitors Advertisement

Table 8-46 describes supported module and lane monitors.

Table 8-46 Supported Monitors Advertisement (Page 01h)

Byte	Bit	Field Name	Field Description	Type
159	7-6	-	**Reserved**	RO
	5	CustomMonSupported	0b: Custom monitor not supported 1b: Custom monitor supported	RO Rqd.
	4	Aux3MonSupported	0b: Aux 3 monitor not supported 1b: Aux 3 monitor supported	
	3	Aux2MonSupported	0b: Aux 2 monitor not supported 1b: Aux 2 monitor supported	
	2	Aux1MonSupported	0b: Aux 1 monitor not supported 1b: Aux 1 monitor supported	
	1	VccMonSupported	0b: Internal 3.3 V monitor not supported 1b: Internal 3.3 V monitor supported	
	0	TempMonSupported	0b: Temperature monitor not supported 1b: Temperature monitor supported	
160	7-5	-	**Reserved**	RO Rqd.
	4-3	TxBiasCurrentScalingFactor	Multiplier for 2uA Bias current increment used in Tx Bias current monitor and threshold registers (see Table 8-55 and Table 8-79) 00b: multiply x1 01b: multiply x2 10b: multiply x4 11b: reserved	
	2	RxOpticalPowerMonSupported	0b: Rx Optical Input Power monitor not supported 1b: Rx Optical Input Power monitor supported	
	1	TxOpticalPowerMonSupported	0b: Tx Output Optical Power monitor not supported 1b: Tx Output Optical Power monitor supported	
	0	TxBiasMonSupported	0b: Tx Bias monitor not supported 1b: Tx Bias monitor supported	

8.4.10 Supported Signal Integrity Controls Advertisement

Table 8-47 describes the advertisement of signal integrity controls and related settings.

Table 8-47 Supported Signal Integrity Controls Advertisement(Page 01h)

Byte	Bit	Field Name	Field Description	Type
161	7	-	**Reserved**	RO
	6-5	TxInputEqRecallBuffersSupported	00b: Tx Input Eq Store/Recall not supported 01b: Tx Input Eq Store/Recall buffer count=1 10b: Tx Input Eq Store/Recall buffer count=2 11b: reserved	RO Rqd.
	4	TxInputEqFreezeSupported	0b/1b: Tx Input Eq Freeze not supported / supported	
	3	TxInputAdaptiveEqSupported	0b/1b: Adaptive Tx Input Eq not supported / supported	
	2	TxInputEqFixedManualControlSupported	0b: Tx Input Eq Fixed Manual control not supported 1b: Tx Input Eq Fixed Manual control supported	
	1	TxCDRBypassControlSupported	0b: If a Tx CDR is supported, it cannot be bypassed 1b: If a Tx CDR is supported, it can be bypassed	
	0	TxCDRSupported	0b/1b: Tx CDR not supported / supported	
162	7-6	-	**Reserved**	RO
	5	StagedSet1Supported	Staged Control Set 1 supported on Page 10h	RO Rqd.
	4-3	RxOutputEqControlSupported	00b: Rx Output Eq control not supported 01b: Rx Output Eq Pre-cursor control supported 10b: Rx Output Eq Post-cursor control supported 11b: Rx Output Eq Pre- and Post-cursor control supported	
	2	RxOutputAmplitudeControlSupported	0b: Rx Output Amplitude control not supported 1b: Rx Output Amplitude control supported	
	1	RxCDRBypassControlSupported	0b: Rx CDR Bypass control not supported (if a CDR is supported, it cannot be bypassed) 1b: Rx CDR Bypass control supported	
	0	RxCDRSupported	0b: Rx CDR not supported 1b: Rx CDR supported	

8.4.11 CDB Messaging Support Advertisement

Table 8-48 describes how the module advertises fundamental support of the Command Data Block (CDB) command and reply messaging functionality, as well as of some high-level CDB features and characteristics.

The host can query support for individual CDB commands through the CDB commands shown in Table 9-1.

See section 7.2 for conceptual information on CDB, section 8.15 for details of the messaging mechanism, and chapter 9 for the CDB Command Reference (message catalogue).

Table 8-48 CDB Advertisement (Page 01h)

Byte	Bit	Field Name	Field Description	Type
163	7-6	CdbInstancesSupported	00b: CDB functionality not supported 01b: One CDB instance supported 10b: Two CDB instances supported 11b: Reserved **One CDB Instance** Bank 0 of Pages 9Fh and the subset of Pages A0h-AFh advertised in CdbMaxPagesEPL (01h:163.3-0), CdbCmdCompleteFlag1 Flag (00h:8.6) and the associated Mask (00h:31.6) are supported. **Two CDB Instances** Banks 0 and 1 of Pages 9Fh and of the subset of Pages A0h-AFh advertised in CdbMaxPagesEPL (01h:163.3-0), CdbCmdCompleteFlag<i> (00h:8.7-6), <i> = 1,2, and the associated Masks (00h:31.7-6) are supported.	RO Rqd.
	5	CdbBackgroundModeSupported	0b: Background CDB operation not supported. 1b: Background CDB operation supported	RO Cnd.

Byte	Bit	Field Name	Field Description	Type
			Note: In Background Mode, register access is possible while a CDB command is still being processed.	
	4	CdbAutoPagingSupported	When the Management Memory current address pointer advances past the end of an Extended Payload (EPL) CDB Page (A0h-AFh), the page number in the PageSelect Byte will automatically increment and the Memory Map current address pointer automatically wraps to 128. When the last page number AFh is incremented, the page number wraps back to A0h. 0b: Auto Paging not supported 1b: Auto Paging and Auto Page wrap supported	RO Cnd.
	3-0	CdbMaxPagesEPL	This field encodes the EPL Page range supported or, equivalently, the maximum length of extended payload: **Value** **Supported** **Total Number of** **EPL Pages** **Pages** **EPL Bytes** 0: (none) 0 0 1: A0h 1 128 2: A0h-A1h 2 256 3 A0h-A2h 3 384 4 A0h-A3h 4 512 5 A0h-A7h 8 1024 6 A0h-ABh 12 1536 7 A0h-AFh 16 2048 *Note: A host can access all supported EPL Pages and the EPL Page range is sufficient for all supported CDB commands. The required number of EPL pages may be CDB command specific.*	RO Cnd.
164	7-0	CdbReadWriteLengthExtension	*Note: For READ and WRITE efficiency in CDB messaging, a module can support multi-byte ACCESS in the CDB Page range (9Fh-AFh) with more than 8 bytes.* **CdbReadWriteLengthExtension** = **i** specifies **i***8 allowable additional number of bytes in a WRITE or READ access to an EPL CDB Page (A0Fh-AFh), i.e. **i** is a length extension in units of byte octets (8 bytes). For page **9Fh** (without auto paging support), the allowable length extension is min(**i,**15)*8 = 120 Bytes. This leads to the maximum length of a READ or a WRITE **Value** **Maximum number of bytes (EPL)** 0: 8 bytes (no extension of general length limit) **i**: 8 * (1+**i**) bytes (0 ≤ i ≤ 255) 255: 8 * 256 = **2048** bytes max **Value** **Maximum Number of Bytes (LPL)** 0: 8 bytes (no extension of general length limit) **i**: 8 * (1+**i**) bytes (0 ≤ **i** ≤ 15) **i**: 8 * 16 = 128 bytes (16≤ **i** ≤ 256) *Note: If the MCI transaction from the host is longer than the length allowed as per this advertisement, the module may ignore bytes written beyond the allowed length and not return more that so many bytes in a read.*	RO Cnd.

Byte	Bit	Field Name	Field Description	Type
165	7	CdbCommandTriggerMethod	Determines how the host triggers CDB command processing in the module and when this occurs: 1b: when the MCI transaction of a WRITE access including the CMDID register 9Fh:129 is properly terminated by the host (STOP). 0b: when a single byte WRITE to 9Fh:129 or a two-byte WRITE to the CMDID register Byte 9Fh:128-129 is properly terminated by the host (STOP). *Note: Preferred method 1b enables the host to WRITE a complete CMD message (header and body) in one go, whereas in Method 0b the host composes the CMD message body first and then triggers CMD processing in a second step. See also section 7.2.*	RO Cnd.
	6-5	-	**Reserved**	RO
	4-0	CdbExtMaxBusyTime	When CdbMaxBusySpecMethod=1b: CdbExtMaxBusyTime = X encodes the maximum CDB busy time $\mathbf{T_{CDBB}}$ as max(1,X)*160 ms in a range of 160 ms to 4960 ms. When CdbMaxBusySpecMethod=0b: don't care	RO Cnd.
166	7	CdbMaxBusySpecMethod	0b: Indicates that the maximum CDB busy time $\mathbf{T_{CDBB}}$ is specified via CdbMaxBusyTime (01h:166.6-0) 1b: Indicates that the maximum CDB busy time $\mathbf{T_{CDBB}}$ is specified via Cdb**Ext**MaxBusyTime (01h:165.4-0).	RO Cnd.
	6-0	CdbMaxBusyTime	When CdbMaxBusySpecMethod=0b: CdbMaxBusyTime=X encodes the maximum CDB busy time $\mathbf{T_{CDBB}}$ as (80–max(80,X))ms in a range of 0 ms to 80 ms. When CdbMaxBusySpecMethod=1b: don't care	RO Cnd.

Editor's Note: The following text needs to be revised and cleared from redundancies in a future revision.

CDB instances, are supported by the module.

A value of CdbInstancesSupported = 0 indicates that CDB is not supported at all.

A CDB instance is identified by non-zero **CDB instance number**

The Bank Index of the Pages belonging to a CDB instance is the CDB instance number decremented by one.

All CDB instances behave identically and support the same set of CDB commands.

Note: Module support for multiple CDB instances can be useful when long-duration CDB commands operating in the background are supported, such as firmware update.

Background Operation

The **CdbBackgroundModeSupported** Bit 01h:163.5 defines if the host can ACCESS the addressable management memory while a CDB command is being processed by the module.

- If CdbBackgroundModeSupported is cleared, the module will hold-off ACCESS while a CDB command is being executed until the command is completed (see section 5.2.3 and section 10.2.2)
- If CdbBackgroundModeSupported is set, the module will possibly hold-off ACCESS only until the CDB command is parsed and captured or queued (see section 5.2.3 and section 10.2.2.

When CDB Background Operation is supported, the host can read the **CdbStatus** field to determine the status of in-progress CDB commands (see Table 8-12).

While a CDB Command is being executed in the background, the module ensures that the relevant internal background operations (such as flash EEPROM writes) do not affect other host interactions with the module that may concurrently occur in the foreground.

Auto Paging

The advertisements of Auto-Paging support (in **CdbAutoPagingSupported** Bit 01h:163.4), the number of supported EPL pages (in **CdbMaxPagesEPL** Field 01h:163.3-0), and the maximum write transaction length (in the **CdbReadWriteLengthExtension** Byte 01h:164) are interrelated, as described in Table 8-49, below.

Table 8-49 Overview of CDB advertising combinations

Auto Paging Supported? (01h:163.4)	# EPL Pages Supported? (01h:163.3-0)	Max # Bytes in Seq. Byte Write (01h:164)	Description
0	> 0	Any	Auto paging is not supported. Normal wrap of current address within page. The host should not write past the end of an EPL Page.
1	0	Any	Invalid (useless).
	> 0		Auto Paging is supported, a write past address 255 automatically increments the Page number and wraps the current address pointer to byte 128. If the Page number increment is past the last supported EPL Page, the Page number wraps back to A0h. *Note: The host may use the Auto Paging feature to write data in large chunks, without the overhead of explicitly programming Page changes.*

8.4.12 Additional Durations Advertising

The fields described in Table 8-50 advertise maximum durations of module operations. See also section 8.4.5.

Table 8-50 Additional State Machine Durations Advertising (Page 01h)

Byte	Bit	Field Name	Field Description	Type
167	7-4	MaxDurationModulePwrDn	Encoded maximum duration of the ModulePwrDn state (see Table 8-42)	RO Rqd.
	3-0	MaxDurationModulePwrUp	Encoded maximum duration of the ModulePwrUp state (see Table 8-42)	
168	7-4	MaxDurationDPTxTurnOff	Encoded maximum duration of the DPTxTurnOff state (see Table 8-42)	RO Rqd.
	3-0	MaxDurationDPTxTurnOn	Encoded maximum duration of the DPTxTurnOn state (see Table 8-42)	
169	7-0	-	**Reserved[1]**	

Note: The MaxDuration fields allow hosts to determine when something has gone wrong in the module during transient states, for example when a module firmware is hung up.*

The module shall report worst-case maximum state durations across all supported configurations.

See sections 6.3.2.9 and 6.3.2.11 for details of the ModulePwrUp and ModulePwrDn states and sections 6.3.3.8 and 6.3.3.10 for details of the DPTxTurnOn and DPTxTurnOff states.

8.4.13 Media Lane Assignment Options Advertising

Each Application Descriptor comprises five bytes to advertise an Application, as described in section 6.2.1.4.

The first four bytes of the Application Descriptors are advertised in Bytes 00h:86-117 (see Table 8-19) or in Bytes 01h:223-250 (see Table 8-52). The fifth bytes are advertised in Bytes 01h:176-190 (see Table 8-51).

Both parts of the Application Descriptor are linked by the number known as AppSel Code. The AppSel code is simply the sequential number of the Application Descriptor in any of the memory locations storing (parts of) the Application Descriptor array.

Table 8-51 Media Lane Assignment Advertising (Page 01h)

Byte	Bits	Register Name	Register Description	Type
176	7-0	MediaLaneAssignmentOptionsApp1	**MediaLaneAssignmentOptionsApp<i>** Media Lane Assignment Options for the Application advertised in Application descriptor identified by AppSel <i>. Bits 0-7 form a bit map corresponding to Media Lanes 1-8. A set bit indicates that a	RO Rqd.
177	7-0	MediaLaneAssignmentOptionsApp2		
178	7-0	MediaLaneAssignmentOptionsApp3		
179	7-0	MediaLaneAssignmentOptionsApp4		
180	7-0	MediaLaneAssignmentOptionsApp5		
181	7-0	MediaLaneAssignmentOptionsApp6		
182	7-0	MediaLaneAssignmentOptionsApp7		

183	7-0	MediaLaneAssignmentOptionsApp8	Data Path for the Application is allowed to begin on the corresponding Media Lane. Each instance of an Application uses contiguous Media Lane numbers. If multiple instances of a single Application are allowed, each starting point is identified, and all instance must be supported concurrently. (See section 6.2.1)	
184	7-0	MediaLaneAssignmentOptionsApp9		
185	7-0	MediaLaneAssignmentOptionsApp10		
186	7-0	MediaLaneAssignmentOptionsApp11		
187	7-0	MediaLaneAssignmentOptionsApp12		
188	7-0	MediaLaneAssignmentOptionsApp13		
189	7-0	MediaLaneAssignmentOptionsApp14		
190	7-0	MediaLaneAssignmentOptionsApp15		

8.4.14 Additional Application Advertising

Bytes 01h:223-250 (see Table 8-52) provide space for seven additional Application Descriptors (the first four bytes of each descriptor) in addition to the eight Application Descriptors in Bytes 86-177 (see Table 8-19).

See section 6.2.1.4 for information on Application Advertising and section 8.2.11 for information about the Application Descriptor and the array of Application Descriptors.

The HostInterfaceID field of the first unused descriptor in Table 8-52 shall have a value of FFh, indicating the end of the list of Application Descriptors.

Table 8-52 Additional Application Descriptor Registers (Page 01h)

Byte	Bits	Field Name	Register Description	Type
223	7-0	HostInterfaceIDApp9	**AppDescriptor9** AppSel 9 (1001b) See Table 8-18	RO Cnd.
224	7-0	MediaInterfaceIDApp9		
225	7-4	HostLaneCountApp9		
	3-0	MediaLaneCountApp9		
226	7-0	HostLaneAssignmentOptionsApp9		
227	7-0	HostInterfaceIDApp10	**AppDescriptor10** AppSel 10 (1010b) See Table 8-18	RO Cnd.
228	7-0	MediaInterfaceIDApp10		
229	7-4	HostLaneCountApp10		
	3-0	MediaLaneCountApp10		
230	7-0	HostLaneAssignmentOptionsApp10		
231	7-0	HostInterfaceIDApp11	**AppDescriptor11** AppSel 11 (1011b) See Table 8-18	RO Cnd.
232	7-0	MediaInterfaceIDApp11		
233	7-4	HostLaneCountApp11		
	3-0	MediaLaneCountApp11		
234	7-0	HostLaneAssignmentOptionsApp11		
235	7-0	HostInterfaceIDApp12	**AppDescriptor12** AppSel 12 (1100b) See Table 8-18	RO Cnd.
236	7-0	MediaInterfaceIDApp12		
237	7-4	HostLaneCountApp12		
	3-0	MediaLaneCountApp12		
238	7-0	HostLaneAssignmentOptionsApp12		
239	7-0	HostInterfaceIDApp13	**AppDescriptor13** AppSel 13 (1101b) See Table 8-18	RO Cnd.
240	7-0	MediaInterfaceIDApp13		
241	7-4	HostLaneCountApp13		
	3-0	MediaLaneCountApp13		
242	7-0	HostLaneAssignmentOptionsApp13		
243	7-0	HostInterfaceIDApp14	**AppDescriptor14** AppSel 14 (1110b) See Table 8-18	RO Cnd.
244	7-0	MediaInterfaceIDApp14		
245	7-4	HostLaneCountApp14		
	3-0	MediaLaneCountApp14		
246	7-0	HostLaneAssignmentOptionsApp14		
247	7-0	HostInterfaceIDApp15	**AppDescriptor15** AppSel 15 (1111b) See Table 8-18	RO Cnd.
248	7-0	MediaInterfaceIDApp15		
249	7-4	HostLaneCountApp15		
	3-0	MediaLaneCountApp15		
250	7-0	HostLaneAssignmentOptionsApp15		

Note: The fifth bytes of these Application Descriptors are stored elsewhere (see Table 8-51)

8.4.15 Page Checksum (Page 01h, Byte 255, RO RQD)

The Page Checksum is a one-byte code that can be used to verify that the read-only static data on Page 01h is valid. The checksum code shall be the low order 8 bits of the arithmetic sum of all byte values from byte 130 to byte 254, inclusive.

Note that the module firmware revision in bytes 128 and 129 is intentionally not included in the Page Checksum.

8.5 Page 02h (Module and Lane Thresholds)

Page 02h is an optional Page that informs about module-defined thresholds for module-level and lane-specific threshold crossing monitors.

The module advertises support of Page 02h in the MemoryModel Bit 00h:2.7.

Note: Page 02h is mandatory for paged memory modules.

All fields on Page 02h are read-only and static.

Table 8-53 Page 02h Overview

Byte	Size (bytes)	Subject Area	Description
128-175	48	Module-level monitor thresholds	
176-199	24	Lane-specific monitor thresholds	
200-229	30	-	**Reserved[30]**
230-254	25	-	**Custom[25]**
255	1	Page Checksum	Covers bytes 128-254

8.5.1 Module-Level Monitor Thresholds

The following thresholds are provided by the module to inform the host of the monitored observable levels where alarms and warnings will be triggered.

Table 8-54 Module-Level Monitor Thresholds (Page 02h)

Byte	Bit	Name	Description	Type
128-129	7-0	TempMonHighAlarmThreshold	S16 Thresholds for internal temperature monitor: 1/256 degree Celsius increments	RO Cnd.
130-131	7-0	TempMonLowAlarmThreshold		
132-133	7-0	TempMonHighWarningThreshold		
134-135	7-0	TempMonLowWarningThreshold		
136-137	7-0	VccMonHighAlarmThreshold	U16 Thresholds for internal 3.3 volt input supply voltage monitor: 100 µV increments	RO Cnd.
138-139	7-0	VccMonLowAlarmThreshold		
140-141	7-0	VccMonHighWarningThreshold		
142-143	7-0	VccMonLowWarningThreshold		
144-145	7-0	Aux1MonHighAlarmThreshold	S16 Thresholds for TEC Current monitor or custom Aux 1 monitor **TEC Current**: 100/32767% increments of maximum TEC current +32767 (100%) – Max Heating -32767 (-100%) – Max Cooling	RO Cnd.
146-147	7-0	Aux1MonLowAlarmThreshold		
148-149	7-0	Aux1MonHighWarningThreshold		
150-151	7-0	Aux1MonLowWarningThreshold		
152-153	7-0	Aux2MonHighAlarmThreshold	S16 Thresholds for TEC Current or Laser Temperature monitor **TEC Current**: 100/32767% increments of maximum TEC current +32767 (100%) – Max Heating -32767 (-100%) – Max Cooling **Laser Temperature**: 1/256 degree Celsius increments	RO Cnd.
154-155	7-0	Aux2MonLowAlarmThreshold		
156-157	7-0	Aux2MonHighWarningThreshold		
158-159	7-0	Aux2MonLowWarningThreshold		
160-161	7-0	Aux3MonHighAlarmThreshold	S16 Thresholds for Laser Temperature or additional supply voltage monitor **Laser Temperature**: 1/256 degree Celsius increments *NOTE: Laser Temp can be below 0 if uncooled or in Tx Output Disable.* **Additional supply voltage monitor**: 100 µV increments	RO Cnd.
162-163	7-0	Aux3MonLowAlarmThreshold		
164-165	7-0	Aux3MonHighWarningThreshold		
166-167	7-0	Aux3MonLowWarningThreshold		
168-169	7-0	CustomMonHighAlarmThreshold	S16 or U16 Custom monitor	RO Cnd.
170-171	7-0	CustomMonLowAlarmThreshold		
172-173	7-0	CustomMonHighWarningThreshold		

Byte	Bit	Name	Description	Type
174-175	7-0	CustomMonLowWarningThreshold		

8.5.2 Lane-Related Monitor Thresholds

The following thresholds are provided by the module to inform the host of the monitor levels where alarms and warnings will be triggered. These monitor thresholds apply to all lanes of the module.

Note: See section 8.9.4 for monitor details including accuracy.

Table 8-55 Lane-Related Monitor Thresholds (Page 02h)

Byte	Bit	Name	Description	Type
176-177	7-0	OpticalPowerTxHighAlarmThreshold	U16 Thresholds for **Tx optical power** monitor: in 0.1 uW increments Total measurement range of 0 to 6.5535 mW (~-40 dBm to +8.2 dBm for non-zero values)	RO Cnd.
178-179	7-0	OpticalPowerTxLowAlarmThreshold		
180-181	7-0	OpticalPowerTxHighWarningThreshold		
182-183	7-0	OpticalPowerTxLowWarningThreshold		
184-185	7-0	LaserBiasCurrentHighAlarmThreshold	U16 Thresholds for **Tx laser bias** monitor: 2 uA increments, times the multiplier encoded in 01h:160.4-3 (see Table 8-46)	RO Cnd.
186-187	7-0	LaserBiasCurrentLowAlarmThreshold		
188-189	7-0	LaserBiasCurrentHighWarningThreshold		
190-191	7-0	LaserBiasCurrentLowWarningThreshold		
192-193	7-0	OpticalPowerRxHighAlarmThreshold	U16 Thresholds for **Rx optical power** monitor: 0.1 uW increments Total measurement range of 0 to 6.5535 mW (~-40 dBm to +8.2 dBm for non-zero values)	RO Cnd.
194-195	7-0	OpticalPowerRxLowAlarmThreshold		
196-197	7-0	OpticalPowerRxHighWarningThreshold		
198-199	7-0	OpticalPowerRxLowWarningThreshold		

8.5.3 Page Checksum (Page 02h, Byte 255, RO RQD)

The Page Checksum code is a one-byte code that can be used to verify that the device property information in the module is valid. The Page Checksum code shall be the low order 8 bits of the arithmetic sum of all byte values from byte 128 to byte 254, inclusive.

8.6 Page 03h (User EEPROM)

Page 03h is an optional Page that allows the module to provide access to a host writeable EEPROM.

The module advertises support of Page 03h in Bit 01h:142.2 (see Table 8-40).

The host may read or write to this memory for any purpose.

Note: Actual usage of the EEPROM is not standardized by CMIS.

The maximum number of bytes in one WRITE access is 8 bytes.

The module rejects register ACCESS until a WRITE to EEPROM is completed internally. The maximum duration of the ACCESS hold-off period caused by internal write completion is tWRITENV as specified in section 10.2.2.

Note: In SFF-8636, a CLEI code may be present in the first 10 Bytes of Page 03h. In CMIS, this convention is aborted, because the CLEI code is allocated in Bytes 00h:190-199.

Table 8-56 Page 03h Overview

Byte	Size (bytes)	Subject Area	Description
128-255	128	User Data	Module user data stored in NV memory

8.7 Page 04h (Laser Capabilities Advertising)

Page 04h is an optional Page that allows the module to advertise tunable laser capabilities.

The module advertises support of Page 04h in Bit 01h:155.6 (see Table 8-44).

The tunable laser capabilities are defined at module level, for all media lanes.

Table 8-57 Page 04h Overview

Byte	**Size** (bytes)	**Subject Area**	**Description**
128-129	2	Wavelength grids	Supported grids and associated channel numbering
130-189	60	Channel number ranges	Channel number ranges per grid
190-197	8	Fine-Tuning Support	Wavelength fine tuning resolution and range
198-201	4	Programmable Output Power	Programmable range of output powers
202-254	53	-	**Reserved[53]**
255	1	Page Checksum	Covers bytes 128-254

Bytes 04h:128 advertises supported channel grids and 04h:129 advertises fine tuning support (see Table 8-58).

In the following Table 8-58 a local parameter **'n'** is used as a shorthand for a signed **channel offset number**, which is sometimes simply called a **channel number**.

This channel offset number effectively specifies a laser frequency in terms of a (signed) frequency offset from a reference channel at 193.1GHz, in units of the grid resolution (except for 75GHz grid, where the offset is defined in units of a third of the grid resolution).

The minimum and maximum channel offset numbers supported are used to define the tuning range on each grid in the **supported channel number range** registers 04h:130-161.

This channel offset number **n** is also used to specify the laser frequency in Page 12h.

Table 8-58 Laser capabilities for tunable lasers (Page 04h)

Byte	**Bits**	**Name**	**Description**	**Type**
128	7	GridSupported75GHz	Bool: Indicates whether the module supports channel-based tuning on the 75 GHz grid defined as: Frequency (THz) = 193.1 + n × 0.025 where n must be divisible by 3. 0b : 75 GHz Grid not supported 1b : 75 GHz Grid supported n is the 16-bit signed channel number that is referred to in the highest/lowest supported channel numbers in Page 04h	RO Rqd.
	6	GridSupported33GHz	Bool: Indicates whether the module supports channel-based tuning on the 33 GHz grid defined as: Frequency (THz) = 193.1 + n × 0.1/3 0b : 33 GHz Grid not supported 1b : 33 GHz Grid supported	RO Rqd.
	5	GridSupported100GHz	Bool: Indicates whether the module supports channel-based tuning on the 100 GHz grid defined as: Frequency (THz) = 193.1 + n × 0.1 0b : 100 GHz Grid not supported 1b : 100 GHz Grid supported	RO Rqd.
	4	GridSupported50GHz	Bool: Indicates whether the module supports channel-based tuning on the 50 GHz grid defined as: Frequency (THz) = 193.1 + n × 0.05 0b : 50 GHz Grid not supported 1b : 50 GHz Grid supported	RO Rqd.
	3	GridSupported25GHz	Bool: Indicates whether the module supports channel-based tuning on the 25 GHz grid defined as: Frequency (THz) = 193.1 + n × 0.025 0b : 25 GHz Grid not supported 1b : 25 GHz Grid supported	RO Rqd.

Byte	Bits	Name	Description	Type
	2	GridSupported12p5GHz	Bool: Indicates whether the module supports channel-based tuning on the 12.5 GHz grid defined as: Frequency (THz) = 193.1 + n × 0.0125 0b : 12.5 GHz Grid not supported 1b : 12.5 GHz Grid supported	RO Rqd.
	1	GridSupported6p25GHz	Bool: Indicates whether the module supports channel-based tuning on the 6.25 GHz grid defined as: Frequency (THz) = 193.1 + n × 0.00625 0b : 6.25 GHz Grid not supported 1b : 6.25 GHz Grid supported	RO Rqd.
	0	GridSupported3p125GHz	Bool: Indicates whether the module supports channel-based tuning on 3.125 GHz grid defined as: Frequency (THz) = 193.1 + n × 0.003125 0b : 3.125 GHz Grid not supported 1b : 3.125 GHz Grid supported	RO Rqd.
129	7	FineTuningSupported	Bool: Indicates whether the module supports fine-tuning of laser frequency in the vicinity of an on-grid channel. 0b : module does not support fine-tuning 1b : module supports fine-tuning	RO Rqd.
	6-0	-	**Reserved** for future channel spacing advertisements	RO Rqd.
130-131	7-0	GridLowChannel3p125GHz	S16 Lowest supported n for 3.125 GHz spacing	RO Rqd.
132-133	7-0	GridHighChannel3p125GHz	S16 Highest supported n for 3.125 GHz spacing	RO Rqd.
134-135	7-0	GridLowChannel6p25GHz	S16 Lowest supported n for 6.25 GHz spacing	RO Rqd.
136-137	7-0	GridHighChannel6p25GHz	S16 Highest supported n for 6.25 GHz spacing	RO Rqd.
138-139	7-0	GridLowChannel12p5GHz	S16 Lowest supported n for 12.5 GHz spacing	RO Rqd.
140-141	7-0	GridHighChannel12p5GHz	S16 Highest supported n for 12.5 GHz spacing	RO Rqd.
142-143	7-0	GridLowChannel25GHz	S16 Lowest supported n for 25 GHz spacing	RO Rqd.
144-145	7-0	GridHighChannel25GHz	S16 Highest supported n for 25 GHz spacing	RO Rqd.
146-147	7-0	GridLowChannel50GHz	S16 Lowest supported n for 50 GHz spacing	RO Rqd.
148-149	7-0	GridHighChannel50GHz	S16 Highest supported n for 50 GHz spacing	RO Rqd.
150-151	7-0	GridLowChannel100GHz	S16 Lowest supported n for 100 GHz spacing	RO Rqd.
152-153	7-0	GridHighChannel100GHz	S16 Highest supported n for 100 GHz spacing	RO Rqd.
154-155	7-0	GridLowChannel33GHz	S16 Lowest supported n for 33 GHz spacing	RO Rqd.
156-157	7-0	GridHighChannel33GHz	S16 Highest supported n for 33 GHz spacing	RO Rqd.
158-159	7-0	GridLowChannel75GHz	S16 Lowest supported n for 75 GHz spacing	RO Rqd.
160-161	7-0	GridHighChannel75GHz	S16 Highest supported n for 75 GHz spacing	RO Rqd.
162-189	7-0	-	**Reserved[28]** for future channel spacing support	RO Rqd.
190-191	7-0	FineTuningResolution	U16 Fine-tuning resolution, increments of 0.001 GHz	RO Rqd.
192-193	7-0	FineTuningLowOffset	S16 lowest fine-tuning offset with resolution of 0.001 GHz	RO Rqd.
194-195	7-0	FineTuningHighOffset	S16 highest fine-tuning offset in increments of 0.001 GHz	RO Rqd.
196	7	ProgOutputPowerPerLaneSupported	0b : Programmable output power per lane not supported 1b : Programmable output power per lane supported	RO Rqd.
	6-0	-	**Reserved**	
197	7-0	-	**Reserved[1]**	RO
198-199	7-0	ProgOutputPowerMin	S16 Minimum Programmable Output Power in increments of 0.01 dBm	RO Rqd.
200-201	7-0	ProgOutputPowerMax	S16 Maximum Programmable Output Power in increments of 0.01 dBm	RO Rqd.
202-254	7-0	-	**Reserved[53]**	RO
255	7-0	PageChecksum	Page Checksum over bytes 128-254 (see section 8.1.4.3)	RO Rqd.

8.8 Banked Page 10h (Lane Control and Data Path Control)

Page 10h is an optional Page that contains lane control bytes.

The module advertises support of Page 10h in the **MemoryModel** Bit 00h:2.7.

Note: Page 0:10h is mandatory for paged memory modules.

Page 10h may optionally be Banked. Each Bank of Page 10h refers to a group of 8 lanes.

Page 10h is subdivided into several areas as illustrated in the following table:

Table 8-59 Page 10h Overview

Byte	**Size** (bytes)	**Subject Area**	**Description**
128	1	Data Path Control	Data Path control bits for each lane, controlling associated Data Path State machines
129-142	14	Lane-Specific Control	Fields to control lane attributes independent of the Data Path State machine or control sets
143-177	35	Staged Control Set 0	Fields to select Applications and signal integrity settings
178-212	35	Staged Control Set 1	Fields to select Applications and signal integrity settings
213-232	20	Lane-Specific Masks	Masks to suppress Interrupts from lane-related Flags
233-239	7	-	**Reserved[7]**
240-255	16	-	**Custom[16]**

8.8.1 Data Path Initialization Control (DPDeinit Bits)

The **DPDeinit** byte controls the initialization of the lanes in all configured Data Paths that are associated with the 8 lanes represented in a Bank.

Note: See chapter 6 for general information on Data Paths and the Module State Machine, and section 6.3.3 for relevant information about the Data Path State Machine

The module evaluates this Byte only in Module State ModuleReady. When the Module State is ModuleReady, the Data Path associated with lanes whose DP**De**init bits are cleared will transition to the DPInit state and begin the initialization process.

Note: By default, all Data Paths will begin initializing when the Module State reaches ModuleReady. The host can prevent this auto-initialization behavior by setting all DPDeinit bits while the module is in the ModuleLowPwr state.

Note: Multiple Data Paths are mutually independent. They may be initialized or deinitialized at the same time or at different times.

Note: The number of lanes in any specific Data Path is a characteristic of the currently active Application that is selected by the AppSel code in the Active Control Set (see Table 8-83). Refer to section 6.2.1.4 for details on AppSel codes and section 6.2.3 for details on Control Sets.

Table 8-60 Data Path initialization control (Page 10h:128)

Byte	**Bit**	**Field Name**	**Register Description**	**Type**
128	7	DPDeinitLane8	**DPDeinitLane<i>** Data Path initialization control for host lane <i> 0b: Initialize the Data Path associated with host lane 1b: Deinitialize the Data Path associated with host lane All lanes of a Data Path must have the same value *Note: These bits represent static requests, not trigger events*	RW Rqd.
	6	DPDeinitLane7		
	5	DPDeinitLane6		
	4	DPDeinitLane5		
	3	DPDeinitLane4		
	2	DPDeinitLane3		
	1	DPDeinitLane2		
	0	DPDeinitLane1		

8.8.2 Lane-Specific Direct Effect Control Fields

Lane-specific direct effect control fields act on individual lanes in the module and are nominally independent of the Data Path (although all lanes of a Data Path are mostly treated identically, except for specific purposes).

Note: The lane-specific direct effect control field settings are not staged and not part of Control Sets.

A module advertises support for individual controls as described in section 8.4.7.

The direct effect control settings for muting a lane output, may remain temporarily ineffective as per Data Path State Machine specifications (e.g. clearing OutputDisableTx has no immediate effect in DPSM state DPInit).

*Note: Clear understanding requires careful distinction between the internal module **function** (of disabling an output or of squelching an output) and the ways how these functions are jointly **controlled** by the module itself and by the external host. The combined effect of host and module controlling these muting functions is determined depending on advertised module capabilities and host-controlled configuration.*

8.8.2.1 Tx Output Muting Functions and Their Control

A Tx output is muted when it is **disabled** or when it is **squelched**. Regarding the muting method applied and the resulting output signal characteristics, the **output disable takes precedence**.

Tx Output Disable Function

When an optical Tx output is **disabled**, its output signal has negligible optical average output power as defined by the relevant media interface standard (e.g. average power <-20dBm). The output is then **quiescent**.

Note: It is irrelevant if the implementation disables transmitter facilities (e.g. laser off) or just shuts down the output. That is why the function is named Tx Output Disable instead of Tx Disable as in earlier revisions.

When an electrical Tx output is **disabled**, it is **quiescent**.

Joint Control of Tx Output Disable Function

The Tx output disable function in the module is normally controlled by the host only, but the DPSM may override and keep an Tx output disabled when it is already nominally un-disabled by the host.

Tx Output Squelching Function

When a Tx output is **squelched**, either the optical modulation amplitude (OMA) or the average power (P_{av}) of the optical output is reduced such that the output is **quiescent**. The module advertises its **squelching method** or whether the host selects the squelching method (see Table 8-44 and Table 8-10). While a Tx output is disabled, the squelching function is masked or blocked (precedence of output disable).

Joint Control of Tx Output Squelch Function

The Tx output squelch function in the module can be controlled both by the host and by a module-internal **squelch controller** that automatically activates the output squelch function when no suitable host side input signal is available for forwarding on the media side output. The output squelch function activation on host command is called **forced** output **squelching**, while the activation due to internal controller decision is called **automatic** output **squelching**.

The automatic squelch controller itself (if supported) can be enabled or disabled by the host.

8.8.2.2 Rx Output Muting Functions and Control Overview

An Rx output is muted when it is **disabled** or when it is **squelched**. Regarding the muting method and the resulting output signal characteristics, there is no precedence defined.

Rx Output Disable Function

When an electrical Rx output is disabled, the output shall be **quiescent**.

Control of Rx Output Disable Function

The Rx output disable function in the module is controlled by the host.

Rx Output Squelching Function

When an electrical Rx output is squelched, the output shall be **quiescent**.

Control of Rx Output Squelching Function

The Rx output squelch function is controlled by a module-internal **squelch controller** that **automatically** activates the output squelch function when no suitable media side input signal is available to be forwarded on the host side output, i.e. when the associated Rx Media lane detects a loss of signal (e.g. Rx LOS is asserted).

Note: A single Rx input lane may be associated to more than one Rx output lane (depending on the Application). The internal controller squelches an Rx electrical output lane until the associated Rx input lane has detected a valid input signal (e.g. Rx LOS de-asserted) and all the associated internal module resources are fully initialized and capable of sending a valid and stable signal to the host. The intent here is to avoid link flaps.

Note: Unlike for Tx, the host cannot force an Rx output to be squelched.

The automatic squelch controller itself (if supported) can be enabled or disabled by the host.

Note: The auto-squelch setting should be identical for all lanes that are part of the same Data Path

8.8.2.3 Lane-specific Tx and Rx Control Fields

This section lists the specific controls for the purposes discussed above.

Tx Output Controls

The host can disable and un-disable Tx output lane N using **OutputDisableTx**<N>.

The host can force and unforce squelching of Tx output lane N using **OutputSquelchForceTx**<N>.

The host can disable or enable the internal squelch controller for lane N using **AutoSquelchDisableTx**<N>.

Tx Input Controls

The host can switch the host side input signal polarity of lane N using **InputPolarityFlipTx**<N>.

The host can freeze the host side input equalizer adaptation for lane N using **AdaptiveInputEqFreezeTx**<N>.

The host can store the current equalizer setting of lane N for later recall using **AdaptiveInputEqStoreTx**<N>.

Rx Output Controls

The host can disable and un-disable Rx output lane N using **OutputDisableRx**<N>.

Note: There is no OutputSquelchForceRx<N> corresponding to OutputSquelchForceTx<N>.

The host can switch the output signal polarity of lane N by **OutputPolarityFlip Rx**<N>.

The host can disable or enable the internal squelch controller for lane N using **AutoSquelchDisableRx**<N>.

Note: It is recommended that hosts configure the same auto-squelching settings for all lanes within a Data Path (media lanes for Tx and host lanes for Rx), otherwise unexpected behavior may result.

8.8.2.4 Register Lists

Table 8-61 Lane-specific Direct Effect Control Fields (Page 10h)

Byte	Bits	Field Name	Register Description	Type
129	7	InputPolarityFlipTx8	**InputPolarityFlipTx<i>** 0b: No Tx input polarity flip for lane <i> 1b: Tx input polarity flip for lane <i> Advertisement: 10h:155.0	RW Adv.
	6	InputPolarityFlipTx7		
	5	InputPolarityFlipTx6		
	4	InputPolarityFlipTx5		
	3	InputPolarityFlipTx4		
	2	InputPolarityFlipTx3		
	1	InputPolarityFlipTx2		
	0	InputPolarityFlipTx1		
130	7	OutputDisableTx8	**OutputDisableTx<i>** 0b: Tx output enabled for media lane <i> 1b: Tx output disabled for media lane <i> Advertisement: 10h:155.1	RW Adv.
	6	OutputDisableTx7		
	5	OutputDisableTx6		
	4	OutputDisableTx5		
	3	OutputDisableTx4		
	2	OutputDisableTx3		
	1	OutputDisableTx2		
	0	OutputDisableTx1		

Byte	Bits	Field Name	Register Description	Type
131	7	AutoSquelchDisableTx8	**AutoSquelchDisableTx<i>** 0b: Automatic Tx output squelching function enabled (when supported) for media lane <i> 1b: Automatic Tx output squelching function disabled for media lane<i> Advertisement: 10h:155.2	RW Adv.
	6	AutoSquelchDisableTx7		
	5	AutoSquelchDisableTx6		
	4	AutoSquelchDisableTx5		
	3	AutoSquelchDisableTx4		
	2	AutoSquelchDisableTx3		
	1	AutoSquelchDisableTx2		
	0	AutoSquelchDisableTx1		
132	7	OutputSquelchForceTx8	**OutputSquelchForceTx<i>** 0b: No impact on Tx output for media lane <i> 1b: Tx output squelched for media lane <i> Advertisement: 10h:155.3	RW Adv.
	6	OutputSquelchForceTx7		
	5	OutputSquelchForceTx6		
	4	OutputSquelchForceTx5		
	3	OutputSquelchForceTx4		
	2	OutputSquelchForceTx3		
	1	OutputSquelchForceTx2		
	0	OutputSquelchForceTx1		
133	7:0	-	**Reserved[1]**	RO
134	7	AdaptiveInputEqFreezeTx8	**AdaptiveInputEqFreezeTx<i>** 0b: No impact on Tx input eq adaptation behavior for lane <i> 1b: Tx input equalizer adaptation frozen at last value for lane <i> See section 6.2.5.4 Advertisement: 10h:161.4	RW Adv.
	6	AdaptiveInputEqFreezeTx7		
	5	AdaptiveInputEqFreezeTx6		
	4	AdaptiveInputEqFreezeTx5		
	3	AdaptiveInputEqFreezeTx4		
	2	AdaptiveInputEqFreezeTx3		
	1	AdaptiveInputEqFreezeTx2		
	0	AdaptiveInputEqFreezeTx1		
135	7-6	AdaptiveInputEqStoreTx4	**AdaptiveInputEqStoreTx<i>** Tx Input Equalizer Adaptation Store location for lane <i> 00b: reserved 01b: store to recall buffer 1 10b: store to recall buffer 2 11b: reserved See section 6.2.5.4 Advertisement: 10h:161.6-5	WO Adv.
	5-4	AdaptiveInputEqStoreTx3		
	3-2	AdaptiveInputEqStoreTx2		
	1-0	AdaptiveInputEqStoreTx1		
136	7-6	AdaptiveInputEqStoreTx8		WO Adv.
	5-4	AdaptiveInputEqStoreTx7		
	3-2	AdaptiveInputEqStoreTx6		
	1-0	AdaptiveInputEqStoreTx5		
137	7	OutputPolarityFlipRx8	**OutputPolarityFlipRx<i>** 0b: No Rx output polarity flip for lane <i> 1b: Rx output polarity flip for lane <i> Advertisement: 10h:155.0	RW Adv.
	6	OutputPolarityFlipRx7		
	5	OutputPolarityFlipRx6		
	4	OutputPolarityFlipRx5		
	3	OutputPolarityFlipRx4		
	2	OutputPolarityFlipRx3		
	1	OutputPolarityFlipRx2		
	0	OutputPolarityFlipRx1		
138	7	OutputDisableRx8	**OutputDisableRx<i>** 0b: Rx output enabled for lane <i> 1b: Rx output disabled for lane <i> Advertisement: 10h:155.1	RW Adv.
	6	OutputDisableRx7		
	5	OutputDisableRx6		
	4	OutputDisableRx5		
	3	OutputDisableRx4		
	2	OutputDisableRx3		
	1	OutputDisableRx2		
	0	OutputDisableRx1		
139	7	AutoSquelchDisableRx8	**AutoSquelchDisableRx<i>** 0b: Automatic Rx output squelching function enabled (when supported) for host lane <i> 1b: Automatic Rx output squelching function disabled for host lane <i> Advertisement: 10h:155.2	RW Adv.
	6	AutoSquelchDisableRx7		
	5	AutoSquelchDisableRx6		
	4	AutoSquelchDisableRx5		
	3	AutoSquelchDisableRx4		
	2	AutoSquelchDisableRx3		
	1	AutoSquelchDisableRx2		

Byte	Bits	Field Name	Register Description	Type
	0	AutoSquelchDisableRx1		
140-142	All	-	**Reserved[3]**	RO

8.8.3 Staged Control Set 0

Staged Control Set 0 is required for all paged memory modules.

Background on Control Sets and on the associated configuration and reconfiguration procedures (Provision and Provision-and-Commission) is described in sections 6.2.3. and 6.2.4

8.8.3.1 Apply Staged Control Set Triggers (Configuration Commands)

SCS0::ApplyDPInit and **SCS0::ApplyImmediate** are write-only (WO) trigger registers that allow the host to request execution of a configuration or reconfiguration procedure for one or more Data Paths selected by means of a lane bit mask in the value being written to the trigger register.

Note: Changes to per-lane configurations of a Staged Control Set have no effect on the behavior of a lane until the host has successfully triggered ApplyDPInit or ApplyImmediate for the lane (causing a copy of the lane configuration in the staged control set into the Active Control Set and, ultimately, into hardware or firmware).

Note: As described in sections 6.2.3.3, an ApplyDPInit trigger may cause commissioning to hardware without host intervention by a series of DPSM state transitions through DPInit, initiated via the DPInitPending bits.

*Note: ApplyImmediate is not supported when **SteppedConfigOnly**=1, WRITE access is then ignored.*

A successful **Provision** configuration procedure copies settings from a Staged Control Set to the Active Control set and sets DPInitPending bits.

A successful **Provision-and-Commission** reconfiguration procedure copies settings from a Staged Control Set to the Active Control set and commits the Active Control set to hardware.

Note: The ApplyDPInit and ApplyImmediate registers are stateless trigger registers with write-only access type. This implies that the value read from the register is not specified. Modules may use the bits in these registers for any purpose, including to signal command execution or acceptance status, e.g. for debug purposes.

Command Handling Protocol

The module response to a WRITE onto one of the Apply registers follows a four-step command handling protocol with status reporting in the **ConfigStatus** register (see also section 8.9.5)

(1) Command Acceptance

When a previously triggered configuration procedure is still being executed on the lanes of a Data Path, the module (silently) ignores any new trigger bits for that Data Path and does not execute the following steps.

Otherwise the module sets ConfigStatusLane<i> = ConfigInProgress for all Data Path lanes <i>, immediately, to indicate that a configuration procedure is being executed.

(2) Parameter Validation

The module validates the defined configuration settings in Staged Control Set 0, and optionally checks if the command is admissible in the current state, before making any changes.

Note: Module hardware and Active Control Set remain unchanged when a trigger is ignored (not accepted) or when the settings in the Staged Control Set are not successfully validated.

(3) Command Execution

The module first copies the validated settings from **Staged Control Set 0** to the Active Control Set, for all lanes <i> selected and accepted in the value written to the Apply trigger register. Further processing then depends on the current state of the Data Path and on the particular Apply trigger register used, as follows:

Provision procedure: When the host WRITE was to ApplyDPInit (or to ApplyImmediate for lanes indicating DPDeactivated state) the module sets the DPInitPendingLane<i> bits in the DPInitPending register.

Provision-and-Commission procedure: When the host WRITE was to ApplyImmediate for lanes indicating DPInitialized or DPActivated state, the module commits the configuration in the Active Control set to hardware.

(4) Result Feedback

Command processing terminates when the module eventually writes the command result status to the **ConfigStatus** fields: The same result status is written to all lanes of a Data Path.

Note: In case of multiple rejection reasons, the module choses the most important rejection reason. The reporting priority is not specified.

Table 8-62 Staged Control Set 0, Apply Triggers (Page 10h)

Byte	Bits	Field Name	Register Description	Type
143	7	ApplyDPInitLane8	**SCS0::ApplyDPInitLane<i>** 0b: No action for host lane <i> 1b: Trigger the **Provision** procedure using the Staged Control Set **0** settings for host lane <i>, with feedback provided in the associated ConfigStatusLane<i> field Restriction: This byte must be written in a single-byte WRITE *Note: See Table 6-3 for preconditions and state dependencies*	WO Rqd.
	6	ApplyDPInitLane7		
	5	ApplyDPInitLane6		
	4	ApplyDPInitLane5		
	3	ApplyDPInitLane4		
	2	ApplyDPInitLane3		
	1	ApplyDPInitLane2		
	0	ApplyDPInitLane1		
144	7	ApplyImmediateLane8	**SCS0::ApplyImmediate<i>** 0b: No action for host lane <i> 1b: Trigger the **Provision** or the **Provision-and-Commission** procedure using the Staged Control Set **0** settings for host lane <i>, with feedback provided in the associated ConfigStatusLane<i> field Restriction: This byte must be written in a single-byte WRITE Condition: **SteppedConfigOnly** (00h:2.6) = 0b *Note: See Table 6-3 for preconditions and state dependencies*	WO Cnd.
	6	ApplyImmediateLane7		
	5	ApplyImmediateLane6		
	4	ApplyImmediateLane5		
	3	ApplyImmediateLane4		
	2	ApplyImmediateLane3		
	1	ApplyImmediateLane2		
	0	ApplyImmediateLane1		

8.8.3.2 Data Path Configuration (Application Assignments)

The Data Path Configuration fields allow the host to **allocate** and configure the **Data Paths** consisting of one or more lanes for one or more of the **Applications** advertised by the module in the **Application Descriptor** registers (Table 8-19 and Table 8-52).

Table 8-64 describes the Data Path Configuration as a **DPConfigLane**<i> configuration register array.

Table 8-63 describes the fields in a DPConfigLane<i> configuration Byte, two of which are descriptive (AppSelCode and DataPathID), while one (ExplicitControl) optionally customizes standard lane configurations.

Table 8-63 Data Path Configuration per Lane (DPConfigLane<i>)

Lane	Bits	Field	Field Description	Type
<i>	7-4	AppSelCode	**SCS<k>::AppSelCodeLane<i>** If lane <i> is part of a Data Path for an Application instance, the AppSelCode field stores the AppSel code of the Descriptor of that Application (see Table 8-19 and Table 8-52) If lane <i> is not part of a Data Path for an Application and hence unused, the AppSelCode field is assigned the NULL value 0000b. All lanes of a Data Path for an Application that spans multiple lanes must have the same setting of AppSelCode.	RW Rqd.
	3-1	DataPathID	**SCS<k>::DataPathIDLane<i>** If lane <i> is part of a Data Path for an Application instance, this DataPathID field stores the DataPathID of that Data Path, i.e. the number of the first lane in the Data Path, decremented by one. If lane<i> is unused, the value of DataPathID is ignored. All lanes of a Data Path for an Application that spans multiple lanes must have the same setting of DataPathID. *Note: For example, the DataPathID of a Data Path including lane 1 is 0 (000b) and the DataPathID of a Data Path where lane 5 is the lowest lane number is 4 (100b).*	RW Rqd.

Lane	Bits	Field	Field Description	Type
	0	ExplicitControl	**SCS<k>::ExplicitControlLane<i>** If lane <i> is part of a Data Path for an Application instance, the ExplicitControl field specifies if lane <i> is configured with host-defined signal integrity settings (provisioned in registers described in Table 8-65 and Table 8-66) or with Application-dependent settings known by the module, when the Staged Control Set is Applied. If lane <i> is unused, the field is ignored. 0b: Use Application-dependent settings for lane <i> 1b: Use Staged Control Set 0 control values for lane <i>	RW Rqd.

Table 8-64 Staged Control Set 0, Data Path Configuration (Page 10h)

Byte	Bits	Field Name	Register Description	Type
145	7-4	AppSelCodeLane1	**SCS0::DPConfigLane1** See Table 8-63	RW Rqd.
	3-1	DataPathIDLane1		
	0	ExplicitControlLane1		
146	7-4	AppSelCodeLane2	**SCS0::DPConfigLane2** See Table 8-63	RW Rqd.
	3-1	DataPathIDLane2		
	0	ExplicitControlLane2		
147	7-4	AppSelCodeLane3	**SCS0::DPConfigLane3** See Table 8-63	RW Rqd.
	3-1	DataPathIDLane3		
	0	ExplicitControlLane3		
148	7-4	AppSelCodeLane4	**SCS0::DPConfigLane4** See Table 8-63	RW Rqd.
	3-1	DataPathIDLane4		
	0	ExplicitControlLane4		
149	7-4	AppSelCodeLane5	**SCS0::DPConfigLane5** See Table 8-63	RW Rqd.
	3-1	DataPathIDLane5		
	0	ExplicitControlLane5		
150	7-4	AppSelCodeLane6	**SCS0::DPConfigLane6** See Table 8-63	RW Rqd.
	3-1	DataPathIDLane6		
	0	ExplicitControlLane6		
151	7-4	AppSelCodeLane7	**SCS0::DPConfigLane7** See Table 8-63	RW Rqd.
	3-1	DataPathIDLane7		
	0	ExplicitControlLane7		
152	7-4	AppSelCodeLane8	**SCS0::DPConfigLane8** See Table 8-63	RW Rqd.
	3-1	DataPathIDLane8		
	0	ExplicitControlLane8		

8.8.3.3 Tx and Rx Signal Integrity Controls

The following control fields allow the host to pre-program signal integrity settings per lane, which the module uses instead of default values that are associated with the Application configured on the Data Path of the lane.

These fields are without effect unless the lane specific ExplicitControl bits are set, as described in section 6.2.3.

These fields have no effect until the staged control set is applied to the Active Control Set.

See section 6.2.3 for the dependency of these fields on the value of the ExplicitControl bit.

See section 6.2.5 for definitions of valid signal integrity control settings.

Table 8-65 Staged Control Set 0, Tx Controls (Page 10h)

Byte	Bits	Field Name	Register Description	Type
153	7	AdaptiveInputEqEnableTx8	**SCS0::AdaptiveInputEqEnableTx<i>** Adaptive input equalizer for host lane <i> 1b: Enable adaptive Tx input equalization 0b: Disable (use manual fixed equalizer) Advertisement: 01h:161.3	RW Adv.
	6	AdaptiveInputEqEnableTx7		
	5	AdaptiveInputEqEnableTx6		
	4	AdaptiveInputEqEnableTx5		
	3	AdaptiveInputEqEnableTx4		
	2	AdaptiveInputEqEnableTx3		
	1	AdaptiveInputEqEnableTx2		
	0	AdaptiveInputEqEnableTx1		
154	7-6	AdaptiveInputEqRecallTx4	**SCS0::AdaptiveInputEqRecallTx<i>** Recall stored Tx input equalizer adaptation settings for host lane <i> when Staged Control Set is copied to Active Control Set. *See section 6.2.5.4 for Store/Recall mechanism* 00b: do not recall 01b: recall buffer 1 10b: recall buffer 2 11b: reserved Advertisement: 01h:161.6-4	RW Adv.
	5-4	AdaptiveInputEqRecallTx3		
	3-2	AdaptiveInputEqRecallTx2		
	1-0	AdaptiveInputEqRecallTx1		
155	7-6	AdaptiveInputEqRecallTx8		RW Adv.
	5-4	AdaptiveInputEqRecallTx7		
	3-2	AdaptiveInputEqRecallTx6		
	1-0	AdaptiveInputEqRecallTx5		
156	7-4	FixedInputEqTargetTx2	**SCS0::FixedInputEqTargetTx<i>** Manual fixed Tx input equalizer control Advertisement: 01h:161.2	RW Adv.
	3-0	FixedInputEqTargetTx1		
157	7-4	FixedInputEqTargetTx4		RW Adv.
	3-0	FixedInputEqTargetTx3		
158	7-4	FixedInputEqTargetTx6		RW Adv.
	3-0	FixedInputEqTargetTx5		
159	7-4	FixedInputEqTargetTx8		RW Adv.
	3-0	FixedInputEqTargetTx7		
160	7	CDREnableTx8	**SCS0::CDREnableTx<i>** 1b: CDR enabled 0b: CDR bypassed Advertisement: 01h:161.0-1	RW Adv.
	6	CDREnableTx7		
	5	CDREnableTx6		
	4	CDREnableTx5		
	3	CDREnableTx4		
	2	CDREnableTx3		
	1	CDREnableTx2		
	0	CDREnableTx1		

Table 8-66 Staged Control Set 0, Rx Controls (Page 10h)

Byte	Bits	Name	Register Description	Type
161	7	CDREnableRx8	**SCS0::CDREnableRx<i>** 1b: CDR enabled 0b: CDR bypassed Advertisement: 01h:162.0-1	RW Adv.
	6	CDREnableRx7		
	5	CDREnableRx6		
	4	CDREnableRx5		
	3	CDREnableRx4		
	2	CDREnableRx3		
	1	CDREnableRx2		
	0	CDREnableRx1		
162	7-4	OutputEqPreCursorTargetRx2	**SCS0::OutputEqPreCursorTargetRx<i>** Rx output equalization pre-cursor target See Table 6-7 Advertisement: 01h:162.4-3	RW Adv.
	3-0	OutputEqPreCursorTargetRx1		
163	7-4	OutputEqPreCursorTargetRx4		RW Adv.
	3-0	OutputEqPreCursorTargetRx3		
164	7-4	OutputEqPreCursorTargetRx6		RW Adv.
	3-0	OutputEqPreCursorTargetRx5		
165	7-4	OutputEqPreCursorTargetRx8		RW Adv.
	3-0	OutputEqPreCursorTargetRx7		
166	7-4	OutputEqPostCursorTargetRx2	**SCS0::OutputEqPostCursorTargetRx<i>** Rx output equalization post-cursor target See Table 6-7 Advertisement: 01h:162.4-3	RW Adv.
	3-0	OutputEqPostCursorTargetRx1		
167	7-4	OutputEqPostCursorTargetRx4		RW Adv.
	3-0	OutputEqPostCursorTargetRx3		
168	7-4	OutputEqPostCursorTargetRx6		RW Adv.
	3-0	OutputEqPostCursorTargetRx5		
169	7-4	OutputEqPostCursorTargetRx8		RW Adv.
	3-0	OutputEqPostCursorTargetRx7		
170	7-4	OutputAmplitudeTargetRx2	**SCS0::OutputAmplitudeTargetRx<i>** Rx output amplitude target See Table 6-8 Advertisement: 01h:162.2	RW Adv.
	3-0	OutputAmplitudeTargetRx1		
171	7-4	OutputAmplitudeTargetRx4		RW Adv.
	3-0	OutputAmplitudeTargetRx3		
172	7-4	OutputAmplitudeTargetRx6		RW Adv.
	3-0	OutputAmplitudeTargetRx5		
173	7-4	OutputAmplitudeTargetRx8		RW Adv.
	3-0	OutputAmplitudeTargetRx7		
174-177	All	-	**Reserved[4]**	RO

8.8.4 Staged Control Set 1

Staged Control Set 1 is optional. If supported, it behaves exactly like Staged Control Set 0 (see section 8.8.3), except that it is zero initialized .

The module advertises support for Staged Control Set 1 in StagedSet1Supported (01h:162.5, see Table 8-47).

Note: Supporting Staged Control Set 1 may be useful for Applications where speed negotiation is performed.

8.8.4.1 Apply Staged Control Set Triggers

See section 8.8.3.1 for description and rationale.

Table 8-67 Staged Control Set 1, Apply Controls (Page 10h)

Byte	Bits	Name	Register Description	Type
178	7	ApplyDPInitLane8	**SCS1::ApplyDPInitLane<i>** 0b: No action for host lane <i> 1b: Trigger the **Provision** procedure using the Staged Control Set **1** settings for host lane <i>, with feedback provided in the associated ConfigStatusLane<i> field Restriction: This byte must be written in a single-byte WRITE *Note: See Table 6-3 for preconditions and state dependencies*	WO Adv.
	6	ApplyDPInitLane7		
	5	ApplyDPInitLane6		
	4	ApplyDPInitLane5		
	3	ApplyDPInitLane4		
	2	ApplyDPInitLane3		
	1	ApplyDPInitLane2		
	0	ApplyDPInitLane1		
179	7	ApplyImmediateLane8	**SCS1::ApplyImmediate<i>** 0b: No action for host lane <i> 1b: Trigger the **Provision** or the **Provision-and-Commission** procedure using the Staged Control Set **1** settings for host lane <i>, with feedback provided in the associated ConfigStatusLane<i> field Restriction: This byte must be written in a single-byte WRITE Condition: **SteppedConfigOnly** (00h:2.6) = 0b *Note: See Table 6-3 for preconditions and state dependencies*	WO Cnd.
	6	ApplyImmediateLane7		
	5	ApplyImmediateLane6		
	4	ApplyImmediateLane5		
	3	ApplyImmediateLane4		
	2	ApplyImmediateLane3		
	1	ApplyImmediateLane2		
	0	ApplyImmediateLane1		

8.8.4.2 Data Path Configuration (Application Assignment)

See section 8.8.3.2 and Table 8-63 for description.

Table 8-68 Staged Control Set 1, Data Path Configuration (Page 10h)

Byte	Bits	Name	Register Description	Type
180	7-4	AppSelCodeLane1	**SCS1::DPConfigLane1** See Table 8-63	RW Adv.
	3-1	DataPathIDLane1		
	0	ExplicitControlLane1		
181	7-4	AppSelCodeLane2	**SCS1::DPConfigLane2** See Table 8-63	RW Adv.
	3-1	DataPathIDLane2		
	0	ExplicitControlLane2		
182	7-4	AppSelCodeLane3	**SCS1::DPConfigLane3** See Table 8-63	RW Adv.
	3-1	DataPathIDLane3		
	0	ExplicitControlLane3		
183	7-4	AppSelCodeLane4	**SCS1::DPConfigLane4** See Table 8-63	RW Adv.
	3-1	DataPathIDLane4		
	0	ExplicitControlLane4		
184	7-4	AppSelCodeLane5	**SCS1::DPConfigLane5** See Table 8-63	RW Adv.
	3-1	DataPathIDLane5		
	0	ExplicitControlLane5		
185	7-4	AppSelCodeLane6	**SCS1::DPConfigLane6** See Table 8-63	RW Adv.
	3-1	DataPathIDLane6		
	0	ExplicitControlLane6		
186	7-4	AppSelCodeLane7	**SCS1::DPConfigLane7** See Table 8-63	RW Adv.
	3-1	DataPathIDLane7		
	0	ExplicitControlLane7		

Byte	Bits	Name	Register Description	Type
187	7-4	AppSelCodeLane8	**SCS1::DPConfigLane8** See Table 8-63	RW Adv.
	3-1	DataPathIDLane8		
	0	ExplicitControlLane8		

8.8.4.3 Tx and Rx Signal Integrity Controls

See section 8.8.3.3 for a description of the fully analogous controls in Staged Control Set 0.

Table 8-69 Staged Control Set 1, Tx Controls (Page 10h)

Byte	Bits	Name	Register Description	Type
188	7	AdaptiveInputEqEnableTx8	**SCS1::AdaptiveInputEqEnableTx<i>** Adaptive input equalizer for host lane <i> 1b: Enable 0b: Disable (use manual fixed equalizer)	RW Adv.
	6	AdaptiveInputEqEnableTx7		
	5	AdaptiveInputEqEnableTx6		
	4	AdaptiveInputEqEnableTx5		
	3	AdaptiveInputEqEnableTx4		
	2	AdaptiveInputEqEnableTx3		
	1	AdaptiveInputEqEnableTx2		
	0	AdaptiveInputEqEnableTx1		
189	7-6	AdaptiveInputEqRecallTx4	**SCS1::AdaptiveInputEqRecallTx<i>** Recall stored Tx input equalizer adaptation settings for host lane <i> when Staged Control Set is copied to Active Control Set. *See section 6.2.5.4 for Store/Recall mechanism* 00b: do not recall 01b: recall from recall buffer 1 10b: recall from recall buffer 2 11b: reserved	RW Adv.
	5-4	AdaptiveInputEqRecallTx3		
	3-2	AdaptiveInputEqRecallTx2		
	1-0	AdaptiveInputEqRecallTx1		
190	7-6	AdaptiveInputEqRecallTx8		RW Adv.
	5-4	AdaptiveInputEqRecallTx7		
	3-2	AdaptiveInputEqRecallTx6		
	1-0	AdaptiveInputEqRecallTx5		
191	7-4	FixedInputEqTargetTx2	**SCS1::FixedInputEqTargetTx<i>** Manual fixed Tx input equalizer control for host lane <i>. Encoding as described in Table 6-6	RW Adv.
	3-0	FixedInputEqTargetTx1		
192	7-4	FixedInputEqTargetTx4		RW Adv.
	3-0	FixedInputEqTargetTx3		
193	7-4	FixedInputEqTargetTx6		RW Adv.
	3-0	FixedInputEqTargetTx5		
194	7-4	FixedInputEqTargetTx8		RW Adv.
	3-0	FixedInputEqTargetTx7		
195	7	CDREnableTx8	**SCS1::CDREnableTx<i>** 1b: CDR enabled 0b: CDR bypassed	RW Adv.
	6	CDREnableTx7		
	5	CDREnableTx6		
	4	CDREnableTx5		
	3	CDREnableTx4		
	2	CDREnableTx3		
	1	CDREnableTx2		
	0	CDREnableTx1		

Table 8-70 Staged Control Set 1, Rx Controls (Page 10h)

Byte	Bits	Name	Register Description	Type
196	7	CDREnableRx8	**SCS1::CDREnableRx<i>** 1b: CDR enabled 0b: CDR bypassed	RW Adv.
	6	CDREnableRx7		
	5	CDREnableRx6		
	4	CDREnableRx5		
	3	CDREnableRx4		
	2	CDREnableRx3		
	1	CDREnableRx2		
	0	CDREnableRx1		
197	7-4	OutputEqPreCursorTargetRx2	**SCS1::OutputEqPreCursorTargetRx<i>** Rx output equalization pre-cursor	RW Adv.
	3-0	OutputEqPreCursorTargetRx1		

Byte	Bits	Name	Register Description	Type
198	7-4	OutputEqPreCursorTargetRx4	See Table 6-7	RW Adv.
	3-0	OutputEqPreCursorTargetRx3		
199	7-4	OutputEqPreCursorTargetRx6		RW Adv.
	3-0	OutputEqPreCursorTargetRx5		
200	7-4	OutputEqPreCursorTargetRx8		RW Adv.
	3-0	OutputEqPreCursorTargetRx7		
201	7-4	OutputEqPostCursorTargetRx2	**SCS1::OutputEqPostCursorTargetRx<i>** Rx output equalization post-cursor See Table 6-7	RW Adv.
	3-0	OutputEqPostCursorTargetRx1		
202	7-4	OutputEqPostCursorTargetRx4		RW Adv.
	3-0	OutputEqPostCursorTargetRx3		
203	7-4	OutputEqPostCursorTargetRx6		RW Adv.
	3-0	OutputEqPostCursorTargetRx5		
204	7-4	OutputEqPostCursorTargetRx8		RW Adv.
	3-0	OutputEqPostCursorTargetRx7		
205	7-4	OutputAmplitudeTargetRx2	**SCS1::OutputAmplitudeTargetRx<i>** Rx output amplitude encoding See Table 6-8	RW Adv.
	3-0	OutputAmplitudeTargetRx1		
206	7-4	OutputAmplitudeTargetRx4		RW Adv.
	3-0	OutputAmplitudeTargetRx3		
207	7-4	OutputAmplitudeTargetRx6		RW Adv.
	3-0	OutputAmplitudeTargetRx5		
208	7-4	OutputAmplitudeTargetRx8		RW Adv.
	3-0	OutputAmplitudeTargetRx7		
209-212	All	-	**Reserved[4]**	

8.8.5 Lane-Specific Masks

The host can control which lane-specific Flags contribute to hardware Interrupt request generation by setting Mask bits in the registers described in Table 8-71. A Mask bit is allocated for each Flag.

See section 8.1.4.2 for more information on Masks, Flags, and Interrupt requests.

Table 8-71 Lane-Specific Masks (Page 10h)

Byte	Bits	Name	Register Description	Type
213	7	DPStateChangedMask8	**DPStateChangedMask<i>** Mask for DPStateChangedFlag<i> for Data Path of host lane <i>	RW Rqd.
	6	DPStateChangedMask7		
	5	DPStateChangedMask6		
	4	DPStateChangedMask5		
	3	DPStateChangedMask4		
	2	DPStateChangedMask3		
	1	DPStateChangedMask2		
	0	DPStateChangedMask1		
214	7	FailureMaskTx8	**FailureMaskTx<i>** Mask for FailureFlagTx<i>, affecting media lane <i> Advertisement: 01h:157.0	RW Adv.
	6	FailureMaskTx7		
	5	FailureMaskTx6		
	4	FailureMaskTx5		
	3	FailureMaskTx4		
	2	FailureMaskTx3		
	1	FailureMaskTx2		
	0	FailureMaskTx1		
215	7	LOSMaskTx8	**LOSMaskTx<i>** Mask for LOSFlagTx, lane <i> Advertisement: 01h:157.1	RW Adv.
	6	LOSMaskTx7		
	5	LOSMaskTx6		
	4	LOSMaskTx5		
	3	LOSMaskTx4		
	2	LOSMaskTx3		
	1	LOSMaskTx2		

Byte	Bits	Name	Register Description	Type
	0	LOSMaskTx1		
216	7	CDRLOLMaskTx8	**CDRLOLMaskTx<i>** Mask for CDRLOLFlagTx<i>, lane <i> Advertisement: 01h:157.2	RW Adv.
	6	CDRLOLMaskTx7		
	5	CDRLOLMaskTx6		
	4	CDRLOLMaskTx5		
	3	CDRLOLMaskTx4		
	2	CDRLOLMaskTx3		
	1	CDRLOLMaskTx2		
	0	CDRLOLMaskTx1		
217	7	AdaptiveInputEqFailMaskTx8	**AdaptiveInputEqFailMaskTx<i>** Mask for AdaptiveInputEqFailFlagTx<i>, lane <i> Advertisement: 01h:157.3	RW Adv.
	6	AdaptiveInputEqFailMaskTx7		
	5	AdaptiveInputEqFailMaskTx6		
	4	AdaptiveInputEqFailMaskTx5		
	3	AdaptiveInputEqFailMaskTx4		
	2	AdaptiveInputEqFailMaskTx3		
	1	AdaptiveInputEqFailMaskTx2		
	0	AdaptiveInputEqFailMaskTx1		
218	7	OpticalPowerHighAlarmMaskTx8	**OpticalPowerHighAlarmMaskTx<i>** Mask for OpticalPowerHighAlarmFlagTx<i>, media lane <i> Advertisement: 01h:160.1	RW Adv.
	6	OpticalPowerHighAlarmMaskTx7		
	5	OpticalPowerHighAlarmMaskTx6		
	4	OpticalPowerHighAlarmMaskTx5		
	3	OpticalPowerHighAlarmMaskTx4		
	2	OpticalPowerHighAlarmMaskTx3		
	1	OpticalPowerHighAlarmMaskTx2		
	0	OpticalPowerHighAlarmMaskTx1		
219	7	OpticalPowerLowAlarmMaskTx8	**OpticalPowerLowAlarmMaskTx<i>** Mask for OpticalPowerLowAlarmFlagTx<i>, media lane <i> Advertisement: 01h:160.1	RW Adv.
	6	OpticalPowerLowAlarmMaskTx7		
	5	OpticalPowerLowAlarmMaskTx6		
	4	OpticalPowerLowAlarmMaskTx5		
	3	OpticalPowerLowAlarmMaskTx4		
	2	OpticalPowerLowAlarmMaskTx3		
	1	OpticalPowerLowAlarmMaskTx2		
	0	OpticalPowerLowAlarmMaskTx1		
220	7	OpticalPowerHighWarningMaskTx8	**OpticalPowerHighWarningMaskTx<i>** Mask for OpticalPowerHighWarningFlagTx<i>, media lane <i> Advertisement: 01h:160.1	RW Adv.
	6	OpticalPowerHighWarningMaskTx7		
	5	OpticalPowerHighWarningMaskTx6		
	4	OpticalPowerHighWarningMaskTx5		
	3	OpticalPowerHighWarningMaskTx4		
	2	OpticalPowerHighWarningMaskTx3		
	1	OpticalPowerHighWarningMaskTx2		
	0	OpticalPowerHighWarningMaskTx1		
221	7	OpticalPowerLowWarningMaskTx8	**OpticalPowerLowWarningMaskTx<i>** Mask for OpticalPowerLowWarningFlagTx<i>, media lane <i> Advertisement: 01h:160.1	RW Adv.
	6	OpticalPowerLowWarningMaskTx7		
	5	OpticalPowerLowWarningMaskTx6		
	4	OpticalPowerLowWarningMaskTx5		
	3	OpticalPowerLowWarningMaskTx4		
	2	OpticalPowerLowWarningMaskTx3		
	1	OpticalPowerLowWarningMaskTx2		
	0	OpticalPowerLowWarningMaskTx1		
222	7	LaserBiasHighAlarmMaskTx8	**LaserBiasHighAlarmMaskTx<i>** Mask for LaserBiasHighAlarmFlagTx<i>, media lane <i> Advertisement: 01h:160.0	RW Adv.
	6	LaserBiasHighAlarmMaskTx7		
	5	LaserBiasHighAlarmMaskTx6		
	4	LaserBiasHighAlarmMaskTx5		
	3	LaserBiasHighAlarmMaskTx4		
	2	LaserBiasHighAlarmMaskTx3		
	1	LaserBiasHighAlarmMaskTx2		

Byte	Bits	Name	Register Description	Type
	0	LaserBiasHighAlarmMaskTx1		
223	7	LaserBiasLowAlarmMaskTx8	**LaserBiasLowAlarmMaskTx<i>** Mask for LaserBiasLowAlarmFlagTx<i>, media lane <i> Advertisement: 01h:160.0	RW Adv.
	6	LaserBiasLowAlarmMaskTx7		
	5	LaserBiasLowAlarmMaskTx6		
	4	LaserBiasLowAlarmMaskTx5		
	3	LaserBiasLowAlarmMaskTx4		
	2	LaserBiasLowAlarmMaskTx3		
	1	LaserBiasLowAlarmMaskTx2		
	0	LaserBiasLowAlarmMaskTx1		
224	7	LaserBiasHighWarningMaskTx8	**LaserBiasHighWarningMaskTx<i>** Mask for LaserBiasHighWarningFlagTx<i>, media lane <i> Advertisement: 01h:160.0	RW Adv.
	6	LaserBiasHighWarningMaskTx7		
	5	LaserBiasHighWarningMaskTx6		
	4	LaserBiasHighWarningMaskTx5		
	3	LaserBiasHighWarningMaskTx4		
	2	LaserBiasHighWarningMaskTx3		
	1	LaserBiasHighWarningMaskTx2		
	0	LaserBiasHighWarningMaskTx1		
225	7	LaserBiasLowWarningMaskTx8	**LaserBiasLowWarningMaskTx<i>** Mask for LaserBiasLowWarningFlagTx<i>, media lane <i> Advertisement: 01h:160.0	RW Adv.
	6	LaserBiasLowWarningMaskTx7		
	5	LaserBiasLowWarningMaskTx6		
	4	LaserBiasLowWarningMaskTx5		
	3	LaserBiasLowWarningMaskTx4		
	2	LaserBiasLowWarningMaskTx3		
	1	LaserBiasLowWarningMaskTx2		
	0	LaserBiasLowWarningMaskTx1		
226	7	LOSMaskRx8	**LOSMaskRx<i>** Mask for LOSFlagRx<i>, media lane <i> Advertisement: 01h:158.1	RW Adv.
	6	LOSMaskRx7		
	5	LOSMaskRx6		
	4	LOSMaskRx5		
	3	LOSMaskRx4		
	2	LOSMaskRx3		
	1	LOSMaskRx2		
	0	LOSMaskRx1		
227	7	CDRLOLMaskRx8	**CDRLOLMaskRx<i>** Mask for CDRLOLFlagRx<i>, media lane <i> Advertisement: 01h:158.2	RW Adv.
	6	CDRLOLMaskRx7		
	5	CDRLOLMaskRx6		
	4	CDRLOLMaskRx5		
	3	CDRLOLMaskRx4		
	2	CDRLOLMaskRx3		
	1	CDRLOLMaskRx2		
	0	CDRLOLMaskRx1		
228	7	OpticalPowerHighAlarmMaskRx8	**OpticalPowerHighAlarmMaskRx<i>** Mask for OpticalPowerHighAlarmFlagRx<i>, media lane <i> Advertisement: 01h:160.2	RW Adv.
	6	OpticalPowerHighAlarmMaskRx7		
	5	OpticalPowerHighAlarmMaskRx6		
	4	OpticalPowerHighAlarmMaskRx5		
	3	OpticalPowerHighAlarmMaskRx4		
	2	OpticalPowerHighAlarmMaskRx3		
	1	OpticalPowerHighAlarmMaskRx2		
	0	OpticalPowerHighAlarmMaskRx1		
229	7	OpticalPowerLowAlarmMaskRx8	**OpticalPowerLowAlarmMaskRx<i>** Mask for OpticalPowerLowAlarmFlagRx<i>, media lane <i> Advertisement: 01h:160.2	RW Adv.
	6	OpticalPowerLowAlarmMaskRx7		
	5	OpticalPowerLowAlarmMaskRx6		
	4	OpticalPowerLowAlarmMaskRx5		
	3	OpticalPowerLowAlarmMaskRx4		
	2	OpticalPowerLowAlarmMaskRx3		
	1	OpticalPowerLowAlarmMaskRx2		

Byte	Bits	Name	Register Description	Type
	0	OpticalPowerLowAlarmMaskRx1		
230	7	OpticalPowerHighWarningMaskRx8	**OpticalPowerHighWarningMaskRx<i>** Mask for OpticalPowerHighWarningFlagRx<i>, media lane <i> Advertisement: 01h:160.2	RW Adv.
	6	OpticalPowerHighWarningMaskRx7		
	5	OpticalPowerHighWarningMaskRx6		
	4	OpticalPowerHighWarningMaskRx5		
	3	OpticalPowerHighWarningMaskRx4		
	2	OpticalPowerHighWarningMaskRx3		
	1	OpticalPowerHighWarningMaskRx2		
	0	OpticalPowerHighWarningMaskRx1		
231	7	OpticalPowerLowWarningMaskRx8	**OpticalPowerLowWarningMaskRx<i>** Mask for OpticalPowerLowWarningFlagRx<i>, media lane <i> Advertisement: 01h:160.2	RW Adv.
	6	OpticalPowerLowWarningMaskRx7		
	5	OpticalPowerLowWarningMaskRx6		
	4	OpticalPowerLowWarningMaskRx5		
	3	OpticalPowerLowWarningMaskRx4		
	2	OpticalPowerLowWarningMaskRx3		
	1	OpticalPowerLowWarningMaskRx2		
	0	OpticalPowerLowWarningMaskRx1		
232	7	OutputStatusChangedMaskRx8	**OutputStatusChangedMaskRx<i>** Mask for OutputStatusChangedFlagRx<i>, media lane <i>	RW Rqd.
	6	OutputStatusChangedMaskRx7		
	5	OutputStatusChangedMaskRx6		
	4	OutputStatusChangedMaskRx5		
	3	OutputStatusChangedMaskRx4		
	2	OutputStatusChangedMaskRx3		
	1	OutputStatusChangedMaskRx2		
	0	OutputStatusChangedMaskRx1		

8.9 Banked Page 11h (Lane Status and Data Path Status)

Page 11h is an optional Page that contains lane dynamic status bytes. All fields on Page 11h are read-only.

The module advertises support of Page 11h in the MemoryModel Bit 00h:2.7.

Note: Page 0:11h is mandatory for paged memory modules.

Page 11h may optionally be Banked. Each Bank of Page 11h refers to a group of 8 lanes.

Page 11h is subdivided into several areas as illustrated in the following table:

Table 8-72 Page 11h Overview

Byte	Size (bytes)	Subject Area	Description
128-131	4	Data Path States	State of Data Path State Machine associated with each lane
132-133	2	Lane Output Status	Output signal validity status (per lane)
134-153	20	Lane-specific Flags	Flags per lane or per Data Path
154-201	48	Lane-specific Monitors	Generic media side monitors for optical power and laser bias
202-205	4	Configuration Status	Status of configuration commands (Apply register writes)
206-234	29	Active Control Set	Nominal or actual Data Path configuration. See section 6.2.3
235-239	5	Data Path Conditions	Dynamic condition indications
240-255	16	Media Lane to media wavelengths and fibers mapping	Indicates the mapping of Media Lanes to Media Wavelengths and Fibers

8.9.1 Data Path States

The following fields report the Data Path States of the Data Path State Machines associated with each host lane.

Data Path States apply to both the host and the media interfaces, but are reported by host lane.

For Data Paths with multiple lanes, the module reports the same state for the lanes of each Data Path.

For unused lanes that are not part of a Data Path, the module reports DPDeactivated

An indication of DPDeactivated means that no Data Path is initialized on that lane.

Table 8-74 defines the Data Path State encodings.

Table 8-73 Lane-associated Data Path States (Page 11h)

Byte	Bit	Field Name	Register Description (DPStateHostLane<i>)	Type
128	7-4	DPStateHostLane2	Data Path State of host lane 2 (see Table 8-74)	RO Rqd.
	3-0	DPStateHostLane1	Data Path State of host lane 1 (see Table 8-74)	
129	7-4	DPStateHostLane4	Data Path State of host lane 4 (see Table 8-74)	RO Rqd.
	3-0	DPStateHostLane3	Data Path State of host lane 3 (see Table 8-74)	
130	7-4	DPStateHostLane6	Data Path State of host lane 6 (see Table 8-74)	RO Rqd.
	3-0	DPStateHostLane5	Data Path State of host lane 5 (see Table 8-74)	
131	7-4	DPStateHostLane8	Data Path State of host lane 8 (see Table 8-74)	RO Rqd.
	3-0	DPStateHostLane7	Data Path State of host lane 7 (see Table 8-74)	

Table 8-74 Data Path State Encoding

Encoding	State
0h	**Reserved**
1h	DPDeactivated (or unused lane)
2h	DPInit
3h	DPDeinit
4h	DPActivated
5h	DPTxTurnOn
6h	DPTxTurnOff
7h	DPInitialized
8h-Fh	**Reserved**

8.9.2 Lane Output Status Indications

The **OutputStatusRx** and **OutputStatusTx** output status indication registers described in Table 8-75 report the status of the high speed outputs of a module, independent of the state of the DPSM instances associated with those output lanes (see also section 6.3.3).

The signal on an **Rx** output host lane is declared valid in the **OutputStatusRx** register (11h:132) only while the module is actually sending a **valid** signal to the host (see definition in section 3.3).

Example: In simple transceiver modules the Rx output is usually valid only after the associated Rx Media lane input has detected an input signal sufficient to initialize and activate all associated internal module resources, and only after those resources have been initialized and activated. A detected valid input Rx input signal is then, after Application specific processing in the module, forwarded to the host as a valid Rx output signal.

For each Rx output host lane status in the OutputStatusRx register there is an associated status change Flag (in Byte 11h:153), which the module sets on a change in the Rx Output Status (in Byte 11h:132) of that lane. The Mask associated with each Flag is found in Byte 10h:232 (see Table 8-71).

Note: The intent here is to avoid link flaps, by ensuring that the module declares its Rx outputs valid only when it is capable of sending a valid and stable signal to the host (after internal resources are initialized and activated).

The signal on an **Tx** output media lane is declared valid in the **OutputStatusTx** register (11h:133) only while the module is actually sending a **valid** media lane signal (see definition in section 3.3).

Note: It should be obvious that the module can neither judge nor ensure signal validity of higher layers that are not processed by the module.

Note: Unlike for the Rx output, there is no Tx output status change reporting Flag defined. This is deliberate because a status change does not trigger any regular configuration activity by the host.

Table 8-75 Lane-Specific Output Status (Page 11h)

Byte	Bit	Field Name	Register Description	Type
132	7	OutputStatusRx8	**OutputStatusRx<i>** 0b: Rx<i> output signal invalid or muted 1b: Rx<i> output signal **valid**	RO Rqd.
	6	OutputStatusRx7		
	5	OutputStatusRx6		
	4	OutputStatusRx5		
	3	OutputStatusRx4		
	2	OutputStatusRx3		
	1	OutputStatusRx2		
	0	OutputStatusRx1		
133	7	OutputStatusTx8	**OutputStatusTx<i>** 0b: Tx<i> output signal muted or invalid 1b: Tx<i> output signal **valid**	RO Rqd.
	6	OutputStatusTx7		
	5	OutputStatusTx6		
	4	OutputStatusTx5		
	3	OutputStatusTx4		
	2	OutputStatusTx3		
	1	OutputStatusTx2		
	0	OutputStatusTx1		

8.9.3 Lane-Specific Flags

This section of the Memory Map contains lane-specific Flags. These Flags provide a mechanism for reporting lane-specific status changes, operating failures, alarms and warnings for monitored observables, or event occurrences. Each lane-specific Flag has an associated Mask.

The general behavior of Flags, Masks, and Interrupt generation is described in section 8.1.4.2.

The lane-specific Flags are defined in Table 8-76, Table 8-77 and Table 8-78.

Table 8-76 Lane-Specific State Changed Flags (Page 11h)

Byte	Bit	Field Name	Register Description	Type
134	7	DPStateChangedFlag8	**DPStateChangedFlag<i>** Latched Data Path State Changed Flag, host lane <i>	RO/COR Rqd.
	6	DPStateChangedFlag7		
	5	DPStateChangedFlag6		
	4	DPStateChangedFlag5		
	3	DPStateChangedFlag4		
	2	DPStateChangedFlag3		
	1	DPStateChangedFlag2		
	0	DPStateChangedFlag1		

Table 8-77 Lane-Specific Tx Flags (Page 11h)

Byte	Bit	Field Name	Register Description	Type
135	7	FailureFlagTx8	**FailureFlagTx<i>** Latched Tx Failure Flag, affecting media lane <i> This Flag indicates an internal failure that causes an unspecified malfunction in the Tx facility used by media lane <i>. *Note: This Flag was formerly named Tx Fault. See glossary for definitions of Fault and Failure.* Advertisement: 01h:157.0	RO/COR Adv.
	6	FailureFlagTx7		
	5	FailureFlagTx6		
	4	FailureFlagTx5		
	3	FailureFlagTx4		
	2	FailureFlagTx3		
	1	FailureFlagTx2		
	0	FailureFlagTx1		
136	7	LOSFlagTx8	**LOSFlagTx<i>** Latched Tx LOS Flag, host lane <i> Advertisement: 01h:157.1	RO/COR Adv.
	6	LOSFlagTx7		
	5	LOSFlagTx6		
	4	LOSFlagTx5		
	3	LOSFlagTx4		
	2	LOSFlagTx3		
	1	LOSFlagTx2		
	0	LOSFlagTx1		
137	7	CDRLOLFlagTx8	**CDRLOLFlagTx<i>** Latched Tx CDR LOL Flag, host lane <i> Advertisement: 01h:157.2	RO/COR Adv.
	6	CDRLOLFlagTx7		
	5	CDRLOLFlagTx6		
	4	CDRLOLFlagTx5		
	3	CDRLOLFlagTx4		
	2	CDRLOLFlagTx3		
	1	CDRLOLFlagTx2		
	0	CDRLOLFlagTx1		
138	7	AdaptiveInputEqFailFlagTx8	**AdaptiveInputEqFailFlagTx<i>** Latched Tx Adaptive Input Eq Fail, host lane <i> Advertisement: 01h:157.3	RO/COR Adv.
	6	AdaptiveInputEqFailFlagTx7		
	5	AdaptiveInputEqFailFlagTx6		
	4	AdaptiveInputEqFailFlagTx5		
	3	AdaptiveInputEqFailFlagTx4		
	2	AdaptiveInputEqFailFlagTx3		
	1	AdaptiveInputEqFailFlagTx2		
	0	AdaptiveInputEqFailFlagTx1		
139	7	OpticalPowerHighAlarmFlagTx8	**OpticalPowerHighAlarmFlagTx<i>** Latched Tx output power High Alarm, media lane <i> Advertisement: 01h:160.1	RO/COR Adv.
	6	OpticalPowerHighAlarmFlagTx7		
	5	OpticalPowerHighAlarmFlagTx6		
	4	OpticalPowerHighAlarmFlagTx5		
	3	OpticalPowerHighAlarmFlagTx4		
	2	OpticalPowerHighAlarmFlagTx3		
	1	OpticalPowerHighAlarmFlagTx2		
	0	OpticalPowerHighAlarmFlagTx1		
140	7	OpticalPowerLowAlarmFlagTx8	**OpticalPowerLowAlarmFlagTx<i>** Latched Tx output power Low alarm, media lane <i>	RO/COR Adv.
	6	OpticalPowerLowAlarmFlagTx7		
	5	OpticalPowerLowAlarmFlagTx6		

Byte	Bit	Field Name	Register Description	Type
	4	OpticalPowerLowAlarmFlagTx5	Advertisement: 01h:160.1	
	3	OpticalPowerLowAlarmFlagTx4		
	2	OpticalPowerLowAlarmFlagTx3		
	1	OpticalPowerLowAlarmFlagTx2		
	0	OpticalPowerLowAlarmFlagTx1		
141	7	OpticalPowerHighWarningFlagTx8	**OpticalPowerHighWarningFlagTx<i>** Latched Tx output power High warning, media lane <i> Advertisement: 01h:160.1	RO/COR Adv.
	6	OpticalPowerHighWarningFlagTx7		
	5	OpticalPowerHighWarningFlagTx6		
	4	OpticalPowerHighWarningFlagTx5		
	3	OpticalPowerHighWarningFlagTx4		
	2	OpticalPowerHighWarningFlagTx3		
	1	OpticalPowerHighWarningFlagTx2		
	0	OpticalPowerHighWarningFlagTx1		
142	7	OpticalPowerLowWarningFlagTx8	**OpticalPowerLowWarningFlagTx<i>** Latched Tx output power Low warning, media lane <i> Advertisement: 01h:160.1	RO/COR Adv.
	6	OpticalPowerLowWarningFlagTx7		
	5	OpticalPowerLowWarningFlagTx6		
	4	OpticalPowerLowWarningFlagTx5		
	3	OpticalPowerLowWarningFlagTx4		
	2	OpticalPowerLowWarningFlagTx3		
	1	OpticalPowerLowWarningFlagTx2		
	0	OpticalPowerLowWarningFlagTx1		
143	7	LaserBiasHighAlarmFlagTx8	**LaserBiasHighAlarmFlagTx<i>** Latched Tx Bias High Alarm, media lane <i> Advertisement: 01h:160.0	RO/COR Adv.
	6	LaserBiasHighAlarmFlagTx7		
	5	LaserBiasHighAlarmFlagTx6		
	4	LaserBiasHighAlarmFlagTx5		
	3	LaserBiasHighAlarmFlagTx4		
	2	LaserBiasHighAlarmFlagTx3		
	1	LaserBiasHighAlarmFlagTx2		
	0	LaserBiasHighAlarmFlagTx1		
144	7	LaserBiasLowAlarmFlagTx8	**LaserBiasLowAlarmFlagTx<i>** Latched Tx Bias Low alarm, media lane <i> Advertisement: 01h:160.0	RO/COR Adv.
	6	LaserBiasLowAlarmFlagTx7		
	5	LaserBiasLowAlarmFlagTx6		
	4	LaserBiasLowAlarmFlagTx5		
	3	LaserBiasLowAlarmFlagTx4		
	2	LaserBiasLowAlarmFlagTx3		
	1	LaserBiasLowAlarmFlagTx2		
	0	LaserBiasLowAlarmFlagTx1		
145	7	LaserBiasHighWarningFlagTx8	**LaserBiasHighWarningFlagTx<i>** Latched Tx Bias High warning, media lane <i> Advertisement: 01h:160.0	RO/COR Adv.
	6	LaserBiasHighWarningFlagTx7		
	5	LaserBiasHighWarningFlagTx6		
	4	LaserBiasHighWarningFlagTx5		
	3	LaserBiasHighWarningFlagTx4		
	2	LaserBiasHighWarningFlagTx3		
	1	LaserBiasHighWarningFlagTx2		
	0	LaserBiasHighWarningFlagTx1		
146	7	LaserBiasLowWarningFlagTx8	**LaserBiasLowWarningFlagTx<i>** Latched Tx Bias Low warning, media lane <i> Advertisement: 01h:160.0	RO/COR Adv.
	6	LaserBiasLowWarningFlagTx7		
	5	LaserBiasLowWarningFlagTx6		
	4	LaserBiasLowWarningFlagTx5		
	3	LaserBiasLowWarningFlagTx4		
	2	LaserBiasLowWarningFlagTx3		
	1	LaserBiasLowWarningFlagTx2		
	0	LaserBiasLowWarningFlagTx1		

Table 8-78 Rx Flags (Page 11h)

Byte	Bit	Field Name	Register Description	Type
147	7	LOSFlagRx8	**LOSFlagRx<i>** Latched Rx LOS Flag, media lane <i> Advertisement: 01h:158.1	RO/COR Adv.
	6	LOSFlagRx7		
	5	LOSFlagRx6		
	4	LOSFlagRx5		
	3	LOSFlagRx4		
	2	LOSFlagRx3		
	1	LOSFlagRx2		
	0	LOSFlagRx1		
148	7	CDRLOLFlagRx8	**CDRLOLFlagRx<i>** Latched Rx CDR LOL Flag, media lane <i> Advertisement: 01h:158.2	RO/COR Adv.
	6	CDRLOLFlagRx7		
	5	CDRLOLFlagRx6		
	4	CDRLOLFlagRx5		
	3	CDRLOLFlagRx4		
	2	CDRLOLFlagRx3		
	1	CDRLOLFlagRx2		
	0	CDRLOLFlagRx1		
149	7	OpticalPowerHighAlarmFlagRx8	**OpticalPowerHighAlarmFlagRx<i>** Latched Rx input power High alarm, media lane <i> Advertisement: 01h:160.2	RO/COR Adv.
	6	OpticalPowerHighAlarmFlagRx7		
	5	OpticalPowerHighAlarmFlagRx6		
	4	OpticalPowerHighAlarmFlagRx5		
	3	OpticalPowerHighAlarmFlagRx4		
	2	OpticalPowerHighAlarmFlagRx3		
	1	OpticalPowerHighAlarmFlagRx2		
	0	OpticalPowerHighAlarmFlagRx1		
150	7	OpticalPowerLowAlarmFlagRx8	**OpticalPowerLowAlarmFlagRx<i>** Latched Rx input power Low alarm, media lane <i> Advertisement: 01h:160.2	RO/COR Adv.
	6	OpticalPowerLowAlarmFlagRx7		
	5	OpticalPowerLowAlarmFlagRx6		
	4	OpticalPowerLowAlarmFlagRx5		
	3	OpticalPowerLowAlarmFlagRx4		
	2	OpticalPowerLowAlarmFlagRx3		
	1	OpticalPowerLowAlarmFlagRx2		
	0	OpticalPowerLowAlarmFlagRx1		
151	7	OpticalPowerHighWarningFlagRx8	**OpticalPowerHighWarningFlagRx<i>** Latched Rx input power High warning, media lane <i> Advertisement: 01h:160.2	RO/COR Adv.
	6	OpticalPowerHighWarningFlagRx7		
	5	OpticalPowerHighWarningFlagRx6		
	4	OpticalPowerHighWarningFlagRx5		
	3	OpticalPowerHighWarningFlagRx4		
	2	OpticalPowerHighWarningFlagRx3		
	1	OpticalPowerHighWarningFlagRx2		
	0	OpticalPowerHighWarningFlagRx1		
152	7	OpticalPowerLowWarningFlagRx8	**OpticalPowerLowWarningFlagRx<i>** Latched Rx input power Low warning, media lane <i> Advertisement: 01h:160.2	RO/COR Adv.
	6	OpticalPowerLowWarningFlagRx7		
	5	OpticalPowerLowWarningFlagRx6		
	4	OpticalPowerLowWarningFlagRx5		
	3	OpticalPowerLowWarningFlagRx4		
	2	OpticalPowerLowWarningFlagRx3		
	1	OpticalPowerLowWarningFlagRx2		
	0	OpticalPowerLowWarningFlagRx1		

153	7	OutputStatusChangedFlagRx8	**OutputStatusChangedFlagRx<i>** Latched Output Status Changed Flag for Rx host lane <i>	RO/COR Rqd.
	6	OutputStatusChangedFlagRx7		
	5	OutputStatusChangedFlagRx6		
	4	OutputStatusChangedFlagRx5		
	3	OutputStatusChangedFlagRx4		
	2	OutputStatusChangedFlagRx3		
	1	OutputStatusChangedFlagRx2		
	0	OutputStatusChangedFlagRx1		

8.9.4 Lane-Specific Monitors

Real time lane monitoring may be performed for each transmit and receive lane and includes Tx output optical power, Rx input optical power, and Tx bias current.

The monitored observables defined here all have associated alarm and/or warning thresholds with associated threshold crossing alarm and/or warning Flags and Flags.

Alarm threshold values and warning threshold values have the same numerical value representation as the associated monitor values for which they specify threshold values.

Monitor results of supported lane monitors shall be within the relevant accuracy requirements when the module is in the DPActivated state.

Measured Tx laser bias current is represented as a 16-bit unsigned integer with the current defined as the full 16-bit value (0 to 65535) with LSB equal to 2 uA times the multiplier from Byte 01h:160. For a multiplier of 1, this yields a total measurement range of 0 to 131 mA.

Accuracy is Vendor Specific but must be better than +/-10% of the manufacturer's nominal value over specified operating temperature and voltage.

Measured Rx input optical power is in mW and can represent either average received power or OMA, depending on the Rx Optical Power Measurement type described in Table 8-43. The parameter is encoded as a 16-bit unsigned integer with the power defined as the full 16-bit value (0 to 65535) with LSB equal to 0.1 uW, yielding a total measurement range of 0 to 6.5535 mW (~-40 to +8.2 dBm for non-zero values).

Absolute accuracy is dependent upon the exact optical wavelength. For the vendor specified wavelength, accuracy shall be better than +/-3 dB over specified temperature and voltage. This accuracy shall be maintained for input power levels up to the lesser of maximum transmitted or maximum received optical power per the appropriate standard. It shall be maintained down to the minimum transmitted power minus cable plant loss (insertion loss or passive loss) per the appropriate standard. Absolute accuracy beyond this minimum required received input optical power range is Vendor Specific.

Measured Tx optical power is the average power represented in mW. The parameter is encoded as a 16-bit unsigned integer with the power defined as the full 16-bit value (0 to 65535) with LSB equal to 0.1 uW, yielding a total measurement range of 0 to 6.5535 mW (~-40 to +8.2 dBm).

Accuracy is Vendor Specific but shall be better than +/-3 dB over specified temperature and voltage for the vendor specified wavelength.

Table 8-79 Media Lane-Specific Monitors (Page 11h)

Byte	Bit	Field Name	Register Description	Type
154-155	7-0	OpticalPowerTx1	U16 **OpticalPowerTx<i>** Internally measured Tx output optical power: in 0.1 uW increments, yielding a total measurement range of 0 to 6.5535 mW (~-40 to +8.2 dBm for non-zero values) Advertisement: 01h:160.1	RO Adv.
156-157	7-0	OpticalPowerTx2		
158-159	7-0	OpticalPowerTx3		
160-161	7-0	OpticalPowerTx4		
162-163	7-0	OpticalPowerTx5		
164-165	7-0	OpticalPowerTx6		
166-167	7-0	OpticalPowerTx7		
168-169	7-0	OpticalPowerTx8		
170-171	7-0	LaserBiasTx1	U16 **LaserBiasTx<i>** Internally measured Tx bias current monitor: in 2 uA increments, times the multiplier from Table 8-46.	RO Adv.
172-173	7-0	LaserBiasTx2		
174-175	7-0	LaserBiasTx3		
176-177	7-0	LaserBiasTx4		

Byte	Bit	Field Name	Register Description	Type
178-179	7-0	LaserBiasTx5	Advertisement: 01h:160.0	
180-181	7-0	LaserBiasTx6		
182-183	7-0	LaserBiasTx7		
184-185	7-0	LaserBiasTx8		
186-187	7-0	OpticalPowerRx1	U16 **OpticalPowerRx<i>** Internally measured Rx input optical power: in 0.1 uW increments, yielding a total measurement range of 0 to 6.5535 mW (~-40 to +8.2 dBm for non-zero values) Advertisement: 01h:160.2	RO Adv.
188-189	7-0	OpticalPowerRx2		
190-191	7-0	OpticalPowerRx3		
192-193	7-0	OpticalPowerRx4		
194-195	7-0	OpticalPowerRx5		
196-197	7-0	OpticalPowerRx6		
198-199	7-0	OpticalPowerRx7		
200-201	7-0	OpticalPowerRx8		

8.9.5 Configuration Command Execution and Result Status (ConfigStatus)

As described in sections 6.2.3.3, 6.2.4, and 8.8.3.1, the host can command a configuration procedure to be executed by the module, by writing a lane trigger bit mask to an Apply register in that Staged Control Set where the desired configuration has been defined.

Both the **Provisioning** procedure and the **Provisioning-and-Commissioning** procedure may take significant time to execute and may be rejected, for a variety of reasons. Therefore, a synchronization protocol about currently ongoing execution and eventual result feedback is implemented, by means of the Configuration Command Execution and Result Status (**ConfigStatus**) register (see Table 8-80).

The field **ConfigStatusLane**<i> informs the host both on the **current command handling status** and on the **final result status** of the most recently accepted configuration command affecting lane <i>.

Configuration Command Handling Protocol

In response to a WRITE access to an Apply register, the module executes four generic command handling activities: (1) command acceptance, (2) parameter validation, (3) command execution, and (4) result feedback.

(1) In the **command acceptance** step the module **checks** for the precondition that none of the relevant fields **ConfigStatusLane**<i> indicate ConfigInProgress. If this check passes, the module immediately sets the ConfigStatusLane<i> field of all Data Path lanes <i> to ConfigInProgress. Otherwise, the module aborts all further command handling steps for the relevant Data Path silently (without feedback).

Note: Busy signaling to the host is specified via the separate ConfigStatus register, but module implementations are free to use the undefined Apply register bits as busy status bits for their own purpose or for debugging.

(2) In the **parameter validation** step the module validates the settings in the relevant Staged Control Set. If the module determines that the desired configuration is invalid, it skips the following command execution step.

Note: It is assumed that the following command execution step cannot fail after successful validation.

(3) In the **command execution** step, the module actually performs the desired procedure. See section 8.8.3.1.

(4) In the **result feedback** step, the module **updates** the fields **ConfigStatusLane**<i> of all lanes of the Data Path with the actual command result status, thereby clearing ConfigInProgress. The presence of a result status value in ConfigStatusLane<i> indicates to the host that the module will accept a new configuration command for the Data Path of lane <i>.

The module populates the ConfigStatusLane<i> fields for all triggered lanes of a Data Path with the same value.

Note: Recall that all configuration procedures are defined to operate on entire Data Paths, with the exception of hot reconfiguration of SI attributes by ApplyImmediate in DPActivated or DPInitialized, where triggers on a subset of lanes are allowed.

The reporting priority of the result status codes is not specified.

Table 8-80 Configuration Command Status registers (Page 11h)

Byte	Bit	Field Name	Field Description	Type
202	7-4	ConfigStatusLane2	**ConfigStatusLane<i>** Configuration Command Execution / Result Status for the Data Path of host lane <i>, during and after the most recent configuration command. See Table 8-81 for the encoding of values. *Note: There is no feedback to the host when an Apply trigger is ignored after failed readiness test (when another configuration is still in progress)*	RO Rqd.
	3-0	ConfigStatusLane1		
203	7-4	ConfigStatusLane4		RO Rqd.
	3-0	ConfigStatusLane3		
204	7-4	ConfigStatusLane6		RO Rqd.
	3-0	ConfigStatusLane5		
205	7-4	ConfigStatusLane8		RO Rqd.
	3-0	ConfigStatusLane7		

The codes in Table 8-81 represent both the current command handling status (**in-progress**, **ready**) and the result status information (**success**, **rejection** due to validation failure), whereby the **ready** status is implicitly indicated by the presence of result status information.

Table 8-81 Configuration Command Execution and Result Status Codes (Page 11h)

Encoding	Name	Value Description
0h	ConfigUndefined	No status information available (initial register value)
1h	ConfigSuccess	**Positive Result Status**: The last accepted configuration command has been completed successfully
2h	ConfigRejected	**Negative Result Status (2h-Bh. Dh-Fh)**: Configuration rejected: unspecific validation failure
3h	ConfigRejectedInvalidAppSel	Configuration rejected: invalid AppSel code
4h	ConfigRejectedInvalidDataPath	Configuration rejected: invalid set of lanes for AppSel
5h	ConfigRejectedInvalidSI	Configuration rejected: invalid SI control settings
6h	ConfigRejectedLanesInUse	Configuration rejected: some lanes not in DPDeactivated
7h	ConfigRejectedPartialDataPath	Configuration rejected: lanes are only subset of DataPath
8h	-	**Reserved** (other validation failures)
9h	-	
Ah	-	
Bh	-	
Ch	ConfigInProgress	**Execution Status**: A configuration command is still being processed by the module; a new configuration command is **ignored** for this lane while ConfigInProgress.
Dh	-	**Custom** Configuration rejected for custom reasons
Eh	-	
Fh	-	

8.9.6 Active Control Set

The Active Control Set is required for all paged memory modules.

Refer to section 6.2.3 for background on Control Sets.

Before exiting the MgmtInit state, the Active Control Set is populated with a default Application and with default signal integrity settings.

The host may update the Active Control Set by using the ApplyDPInit or ApplyImmediate trigger fields in either Staged Control Set 0 or Staged Control Set 1. See section 8.8.3 for a description of Staged Control Set usage.

8.9.6.1 Data Path Configuration (Application Assignment)

The following fields allow the host to infer the current baud rate, modulation format, and Data Path width, as well as other interface related specification elements, for each lane in the module.

Table 8-82 Data Path Configuration per Lane (DPConfigLane<i> Field)

Lane	Bits	Field	Field Description	Type
<i>	7-4	AppSelCode	**ACS::AppSelCodeLane<i>** Defines the Application assigned to the Data Path containing lane <i> by reference to the Application Descriptor of that Application: **0000b**: lane <i> is unused and unassigned (not part of a Data Path) **0001b**, …, **1111b**: lane <i> is part of a Data Path for the Application described in Application Descriptor with AppSel code AppSelCodeLane<i>	RO Rqd.
	3-1	DataPathID	**ACS::DataPathIDLane<i>** Index of first lane in the Data Path containing lane <i> 000b: Lane 1 001b: Lane 2 …. 111b: Lane 8	RO Rqd.
	0	ExplicitControl	**ACS::ExplicitControlLane<i>** 0b: Lane <i> SI settings are Application dependent 1b: Lane <i> SI settings are host defined	RO Rqd.

Table 8-83 Indicators for Active Control Set, Data Path Configuration (Page 11h)

Byte	Bits	Field Name	Register Description	Type
206	7-4	AppSelCodeLane1	**ACS::DPConfigLane1** See Table 8-82	RO Rqd.
	3-1	DataPathIDLane1		
	0	ExplicitControlLane1		
207	7-4	AppSelCodeLane2	**ACS::DPConfigLane2** See Table 8-82	RO Rqd.
	3-1	DataPathIDLane2		
	0	ExplicitControlLane2		
208	7-4	AppSelCodeLane3	**ACS::DPConfigLane3** See Table 8-82	RO Rqd.
	3-1	DataPathIDLane3		
	0	ExplicitControlLane3		
209	7-4	AppSelCodeLane4	**ACS::DPConfigLane4** See Table 8-82	RO Rqd.
	3-1	DataPathIDLane4		
	0	ExplicitControlLane4		
210	7-4	AppSelCodeLane5	**ACS::DPConfigLane5** See Table 8-82	RO Rqd.
	3-1	DataPathIDLane5		
	0	ExplicitControlLane5		
211	7-4	AppSelCodeLane6	**ACS::DPConfigLane6** See Table 8-82	RO Rqd.
	3-1	DataPathIDLane6		
	0	ExplicitControlLane6		
212	7-4	AppSelCodeLane7	**ACS::DPConfigLane7** See Table 8-82	RO Rqd.
	3-1	DataPathIDLane7		
	0	ExplicitControlLane7		
213	7-4	AppSelCodeLane8	**ACS::DPConfigLane8** See Table 8-82	RO Rqd.
	3-1	DataPathIDLane8		
	0	ExplicitControlLane8		

8.9.6.2 Tx and Rx Signal Integrity Fields

The fields described in Table 8-84 and Table 8-85 report the active signal integrity settings for each Tx and Rx lane, respectively.

If the ExplicitControl bit for a lane is set in the Data Path Configuration (see Table 8-83) , the contents of the registers for that lane described in Table 8-84 and Table 8-85 originate from corresponding registers in one of the Staged Control Sets.

If the ExplicitControl bit for a lane is cleared, the contents of the registers for that lane in Table 8-84 and Table 8-85 were determined by the module according to the selected Application.

See section 6.2.5 for definitions of valid signal integrity control settings.

Table 8-84 Indicators for Active Control Set, Tx Controls (Page 11h)

Byte	Bits	Field Name	Register Description	Type
214	7	AdaptiveInputEqEnableTx8	**ACS::AdaptiveInputEqEnableTx<i>** 1b: Enable adaptive Tx input equalizer for lane <i> 0b: Use fixed Tx input equalizer for lane <i> Settings in FixedInputEqTargetTx<i> Advertisement: 01h:161.3	RO Adv.
	6	AdaptiveInputEqEnableTx7		
	5	AdaptiveInputEqEnableTx6		
	4	AdaptiveInputEqEnableTx5		
	3	AdaptiveInputEqEnableTx4		
	2	AdaptiveInputEqEnableTx3		
	1	AdaptiveInputEqEnableTx2		
	0	AdaptiveInputEqEnableTx1		
215	7-6	AdaptiveInputEqRecalledTx4	**ACS::AdaptiveInputEqRecalledTx<i>** Recalled Tx Eq settings status for lane <i> 00b: settings are not recalled 01b: settings have been recalled from recall buffer 1 10b: settings have been recalled from recall buffer 2 11b: reserved *Note: See Table 8-65 and Table 8-69 for the recall settings in the staged control sets.* Advertisement: 01h:161.6-5	RO Adv.
	5-4	AdaptiveInputEqRecalledTx3		
	3-2	AdaptiveInputEqRecalledTx2		
	1-0	AdaptiveInputEqRecalledTx1		
216	7-6	AdaptiveInputEqRecalledTx8		RO Adv.
	5-4	AdaptiveInputEqRecalledTx7		
	3-2	AdaptiveInputEqRecalledTx6		
	1-0	AdaptiveInputEqRecalledTx5		
217	7-4	FixedInputEqTargetTx2	**ACS::FixedInputEqTargetTx<i>** Fixed Tx input equalization for lane <i> encoding as defined in Table 6-6 Advertisement: 01h:161.2	RO Adv.
	3-0	FixedInputEqTargetTx1		
218	7-4	FixedInputEqTargetTx4		RO Adv.
	3-0	FixedInputEqTargetTx3		
219	7-4	FixedInputEqTargetTx6		RO Adv.
	3-0	FixedInputEqTargetTx5		
220	7-4	FixedInputEqTargetTx8		RO Adv.
	3-0	FixedInputEqTargetTx7		
221	7	CDREnableTx8	**ACS::CDREnableTx<i>** 1b: CDR enabled 0b: CDR bypassed Advertisement: 01h:161.1	RO Adv.
	6	CDREnableTx7		
	5	CDREnableTx6		
	4	CDREnableTx5		
	3	CDREnableTx4		
	2	CDREnableTx3		
	1	CDREnableTx2		
	0	CDREnableTx1		

Table 8-85 Indicators for Active Control Set, Rx Controls (Page 11h)

Byte	Bits	Name	Register Description	Type
222	7	CDREnabledRx8	**ACS::CDREnableRx<i>** 1b: CDR enabled 0b: CDR bypassed Advertisement: 01h:162.1	RO Adv.
	6	CDREnabledRx7		
	5	CDREnabledRx6		
	4	CDREnabledRx5		
	3	CDREnabledRx4		
	2	CDREnabledRx3		
	1	CDREnabledRx2		
	0	CDREnabledRx1		
223	7-4	OutputEqPreCursorTargetRx2	**ACS::OutputEqPreCursorTargetRx<i>** Rx output pre-cursor equalization for lane <i> encoded as per Table 6-7 Advertisement: 01h:162.4-3	RO Adv.
	3-0	OutputEqPreCursorTargetRx1		
224	7-4	OutputEqPreCursorTargetRx4		RO Adv.
	3-0	OutputEqPreCursorTargetRx3		
225	7-4	OutputEqPreCursorTargetRx6		RO Adv.
	3-0	OutputEqPreCursorTargetRx5		
226	7-4	OutputEqPreCursorTargetRx8		RO Adv.
	3-0	OutputEqPreCursorTargetRx7		

Byte	Bits	Name	Register Description	Type
227	7-4	OutputEqPostCursorTargetRx2	**ACS::OutputEqPostCursorTargetRx<i>** Rx output post-cursor equalization for lane <i> encoded as per Table 6-7 Advertisement: 01h:162.4-3	RO Adv.
	3-0	OutputEqPostCursorTargetRx1		
228	7-4	OutputEqPostCursorTargetRx4		RO Adv.
	3-0	OutputEqPostCursorTargetRx3		
229	7-4	OutputEqPostCursorTargetRx6		RO Adv.
	3-0	OutputEqPostCursorTargetRx5		
230	7-4	OutputEqPostCursorTargetRx8		RO Adv.
	3-0	OutputEqPostCursorTargetRx7		
231	7-4	OutputAmplitudeTargetRx2	**ACS::OutputAmplitudeTargetRx<i>** Rx output amplitude level for lane <i> encoded as per Table 6-8 Advertisement: 01h:162.2	RO Adv.
	3-0	OutputAmplitudeTargetRx1		
232	7-4	OutputAmplitudeTargetRx4		RO Adv.
	3-0	OutputAmplitudeTargetRx3		
233	7-4	OutputAmplitudeTargetRx6		RO Adv.
	3-0	OutputAmplitudeTargetRx5		
234	7-4	OutputAmplitudeTargetRx8		RO Adv.
	3-0	OutputAmplitudeTargetRx7		

8.9.7 Data Path Conditions

After copying the settings for the lanes of a Data Path from a Staged Control Set to the Active Control Set during a successful **Provision** procedure triggered by ApplyDPInit (see Table 6-3 and Table 6-4 in section 6.2.4.2) the module simultaneously sets the bits **DPInitPendingLane**<i> for the lanes <i> of that Data Path.

This indicates to the host that the Data Path configuration associated with Lane<i> has been updated (not necessarily modified) in the Active Control Set, whereas the subsequent transit through the DPInit state that will eventually commit the nominal settings in the Active Control Set to hardware is still pending.

When **intervention-free** reconfiguration is supported (by default, SteppedConfigOnly=0), these bits may also cause **DPSM state transitions** without further host intervention, as described in section 6.3.3.1.

Table 8-86 Data Path Conditions (Page 11h)

Byte	Bits	Field Name	Register Description	Type
235	7	DPInitPendingLane8	**DPInitPendingLane<i>** 0b: DPInit not pending 1b: DPInit not yet executed after successful ApplyDPInit, hence the Active Control Set content may deviate from the actual hardware configuration. *Note: When SteppedConfigOnly=0, the DPInitPendingLane<i> bits are evaluated in DPSM exit condition equations, see section 6.3.3.1.*	RO Rqd.
	6	DPInitPendingLane7		
	5	DPInitPendingLane6		
	4	DPInitPendingLane5		
	3	DPInitPendingLane4		
	2	DPInitPendingLane3		
	1	DPInitPendingLane2		
	0	DPInitPendingLane1		
236-239			**Reserved[4]**	RO

8.9.8 Media Lane to Media Wavelength and Fiber Mapping

Table 8-87 describes the advertising fields to define the mapping of media lanes to media wavelengths and physical fibers for muxed or WDM implementations.

For WDM applications the shortest wavelength is always designated media wavelength 1 and starting from shortest wavelength through the longest all others are listed consecutively.

The fiber numbering and naming used in Table 8-87 is as defined in the appropriate hardware specification.

See the relevant hardware specification and Table 8-19, Table 8-33, and Table 8-35 for mapping constraints.

A mapping advertised in Table 8-87 that violates any of these constraints is invalid.

Table 8-87 Media Lane to Media Wavelength and Fiber mapping (Page 11h)

Byte	Bits	Name	Register Description	Type
240	7-4	MediaLaneToWavelengthMappingTx1	**MediaLaneToWavelengthMappingTx<i>** Mapping of media lane <i> 0000: Mapping unknown or undefined 0001: Maps to media wavelength 1 0010: Maps to media wavelength 2 0011: Maps to media wavelength 3 0100: Maps to media wavelength 4 0101: Maps to media wavelength 5 0110: Maps to media wavelength 6 0111: Maps to media wavelength 7 1000: Maps to media wavelength 8 1001-1111: Reserved **MediaLaneToFiberMappingTx<i>** Mapping of media lane <i> 0000: Mapping unknown or undefined 0001: Maps to media fiber 1 or TR1 0010: Maps to media fiber 2 or RT1 0011: Maps to media fiber 3 or TR2	RO Cnd.
	3-0	MediaLaneToFiberMappingTx1		
241	7-4	MediaLaneToWavelengthMappingTx2		RO Cnd.
	3-0	MediaLaneToFiberMappingTx2		
242	7-4	MediaLaneToWavelengthMappingTx3		RO Cnd.
	3-0	MediaLaneToFiberMappingTx3		
243	7-4	MediaLaneToWavelengthMappingTx4		RO Cnd.
	3-0	MediaLaneToFiberMappingTx4		
244	7-4	MediaLaneToWavelengthMappingTx5		RO Cnd.
	3-0	MediaLaneToFiberMappingTx5		
245	7-4	MediaLaneToWavelengthMappingTx6		RO Cnd.
	3-0	MediaLaneToFiberMappingTx6		
246	7-4	MediaLaneToWavelengthMappingTx7		RO Cnd.
	3-0	MediaLaneToFiberMappingTx7		
247	7-4	MediaLaneToWavelengthMappingTx8		RO Cnd.
	3-0	MediaLaneToFiberMappingTx8		
248	7-4	MediaLaneToWavelengthMappingRx1		RO Cnd.
	3-0	MediaLaneToFiberMappingRx1		
249	7-4	MediaLaneToWavelengthMappingRx2		RO

Byte	Bits	Name	Register Description	Type
	3-0	MediaLaneToFiberMappingRx2	0100: Maps to media fiber 4 or RT2	Cnd.
250	7-4	MediaLaneToWavelengthMappingRx3	0101: Maps to media fiber 5 or TR3	RO
	3-0	MediaLaneToFiberMappingRx3	0110: Maps to media fiber 6 or RT3	Cnd.
251	7-4	MediaLaneToWavelengthMappingRx4	0111: Maps to media fiber 7 or TR4	RO
	3-0	MediaLaneToFiberMappingRx4	1000: Maps to media fiber 8 or RT4	Cnd.
252	7-4	MediaLaneToWavelengthMappingRx5	1001-1111: Reserved	RO
	3-0	MediaLaneToFiberMappingRx5		Cnd.
253	7-4	MediaLaneToWavelengthMappingRx6		RO
	3-0	MediaLaneToFiberMappingRx6		Cnd.
254	7-4	MediaLaneToWavelengthMappingRx7		RO
	3-0	MediaLaneToFiberMappingRx7		Cnd.
255	7-4	MediaLaneToWavelengthMappingRx8		RO
	3-0	MediaLaneToFiberMappingRx8		Cnd.

8.10 Banked Page 12h (Tunable Laser Control and Status)

Page 12h is an optional Page for laser tuning control, status, and Flags, for transmitters with tunable technology.

The module advertises support of Page 12h in Bit 01h:155.6 (see Table 8-44).

Page 12h may optionally be Banked. Each Bank of Page 12h refers to 8 media lanes.

Table 8-88 Page 12h Overview

Byte	Size (bytes)	Subject Area	Description
128-135	8	Grid Spacings	array with one Byte per media lane
136-151	8 x 2	Channel Offset Numbers	array with one S16 Word per media lane
152-167	8 x 2	Fine Tuning Offsets	array with one S16 Word per media lane
168-199	8 x 4	Laser Frequencies	array with one U32 double word per media lane
200-215	8 x 2	Target Output Power	array with one S16 word per media lane
216-221	6	-	**Reserved[6]**
222-229	8	Status Indicators	array with one Byte per media lane
230	1	Flag Summary	one Bit per media lane
231-238	8	Flags	array with one Byte per media lane
239-246	8	Masks (default: 1)	array with one Byte per media lane, **Masks all set by default**
247-255	9	-	**Reserved[9]** *Note: This page has no Page Checksum*

In Byte register arrays with one byte per lane the lowest byte address represents lane 1.

In Bit arrays (within a Byte) with one bit per lane, the least significant bit 0 represents lane 1.

Table 8-65 Laser tuning, status, and Flags for tunable transmitters (Page 12h)

Byte	Bit	Field Name	Field Description	Type
128-135	7-4	GridSpacingTx<n>	Selected grid spacing of media lane <n>=1-8 0000b: 3.125 GHz 0001b: 6.25 GHz 0010b: 12.5 GHz 0011b: 25 GHz 0100b: 50 GHz 0101b: 100 GHz 0110b: 33 GHz 0111b: 75 GHz 8-14: Reserved 1111b: Not available	RW Rqd.
	3-1	-	**Reserved**	RO
	0	FineTuningEnableTx<n>	Bool: fine-tuning enabled for media lane <n>=1-8 0b: Fine-tuning disabled 1b: Fine-tuning enabled	RW Rqd.
136-151	7-0	ChannelNumberTx<n>	S16 Channel (offset) Number for media lane <n>=1-8 *The meaning of the signed channel (offset) number and its dependence on the selected grid spacing is defined in section 8.7.*	RW Rqd.
152-167	7-0	FineTuningOffsetTx<n>	S16 fine-tuning frequency offset for media lane <n>=1-8 in units of 0.001 GHz	RW Rqd.
168-199	7-0	CurrentLaserFrequencyTx<n>	U32 Current frequency for media lane <n>=1-8 in units of 0.001 GHz	RO Rqd.
200-215	7-0	TargetOutputPowerTx<n>	S16 Target programmable output power for media lane <n>=1-8 in units of 0.01 dBm	RW Rqd.
216-221	7-0	-	**Reserved[6]**	RO Rqd.
222-229	7-2	-	**Reserved**	RO Rqd.
	1	TuningInProgressTx<n>	Bool: Status indication for tuning in progress on media lane <n>=1-8	RO Rqd.

Byte	Bit	Field Name	Field Description	Type
			0b/1b: Tuning not in progress/in progress Tuning is in progress when the laser is tuning to an on-grid channel, or fine tuning to a frequency offset, or tuning to target output power.	
	0	WavelengthUnlockStatusTx<n>	Bool: Unlocked status indication for laser wavelength on media lane <n>=1-8 0b/1b: Wavelength locked/unlocked	RO Rqd.
230	7-0	LaserTuningFlagSummaryTx<n>	Laser tuning Flag summary for media lane <n>=1-8. The bit <n>-1 is set if and only if any of the Flags in Bytes 231-238 are 1 for the particular Lane <n>. *Note: The host should react to an interrupt by reading this Byte to indicate which Lane(s) are responsible for the interrupt. The host should then read the corresponding Lane Byte (231-238) to determine the specific interrupt and to clear the latch.*	RO Rqd.
231-238	7-6	-	**Reserved**	RO Rqd.
	5	TargetOutputPowerOORFlagTx<n>	Latched Flag indicating that a target output power value outside the allowed range was entered for media lane <n>=1-8	RO/COR Rqd.
	4	FineTuningOutOfRangeFlagTx<n>	Latched Flag indicating that a fine-tuning value outside the allowed range was given for media lane <n>=1-8	RO/COR Rqd.
	3	TuningNotAcceptedFlagTx<n>	Latched Flag indicating a failed tuning operation for media lane <n>=1-8: the module was **temporarily** unable to serve a tuning request in its current state, in response to the host programming grid or channel. The Flag indicates that the nominally programmed grid or channel do not match the actual condition of the laser. The Flag clears when channel or grid have been successfully re-programmed. *Note. This Flag may e.g. be asserted by the module if the host attempts to change the target output power or fine tune frequency while tuning in progress bit is '1'.*	RO/COR Rqd.
	2	InvalidChannelNumberFlagTx<n>	Latched Flag indicating that ChannelNumberTx<n> was not in advertised range of the channel spacing selected on media lane <n>=1-8	RO/COR Rqd.
	1	WavelengthUnlockedFlagTx<n>	Latched Flag version of WavelengthUnlockedTx<n>	RO/COR Rqd.
	0	TuningCompleteFlagTx<n>	Latched Flag indicating laser tuning has been completed for media lane <n>=1-8	RO/COR Rqd.
239-246	7-6	-	**Reserved**	RO Rqd.
	5	TargetOutputPowerOORMaskTx<n>	Mask for TargetOutputPowerOORFlagTx<n> Default: 1	RW Rqd.
	4	FineTuningOutOfRangeMaskTx<n>	Mask for FineTuningOutOfRangeFlagTx<n> Default: 1	RW Rqd.
	3	TuningNotAcceptedMaskTx<n>	Mask for TuningNotAcceptedFlagTx<n>=1-8 Default: 1	RW Rqd.
	2	InvalidChannelMaskTx<n>	Mask for InvalidChannelNumberFlagTx<n> Default: 1	RW Rqd.
	1	WavelengthUnlockedMaskTx<n>	Mask for WavelengthUnlockedFlagTx<n> Default: 1	RW Rqd.
	0	TuningCompleteMaskTx<n>	Mask for TuningCompleteFlagTx<n> Default: 1	RW Rqd.
247-255	7-0	-	**Reserved[9]**	RO

8.11 Banked Page 13h (Module Performance Diagnostics Control)

Pages 13h and 14h are optional Pages containing module diagnostic control and result status fields, respectively.

The module advertises support of Pages 13h and 14h jointly in Bit 01h:142.5 (see Table 8-40).

Page 13h may optionally be Banked. Each Bank of Page 13h refers to 8 lanes.

Module Performance Diagnostics allows the host to control and evaluate two types of measurements:

- **intrusive** measurements of **error performance metrics** using pattern generators and checkers
  - U64 bit error counts and total bits
  - F16 bit error ratio (BER)
- **non-intrusive** measurements or estimations of **physical channel metrics** (SNR)

For counting-based error performance metrics, two kinds of controlling the measurement interval are possible

- **host controlled** for on-demand single-shot or quasi-periodic measurements (with stop-restart gap)
- **module controlled** gating interval for single-shot or periodic measurements (gap-free diagnostics)

For both types of interval control, **progressive updates** may be available that also occur periodically using a subinterval of the overall single-shot or periodic measurement interval (gating period).

The host can read selected measurement results in the Diagnostics Data area of Page 14h (see Table 8-112). See Table 8-116 for the detailed list of selectable Diagnostics measurement results and Table 8-91 for the associated advertisement fields.

The actually available measurement methods depend on advertised module capabilities as defined in the following subsections and described in section 8.11.11.

Page 13h is subdivided into subject areas as illustrated in the following table:

Table 8-66 Page 13h Overview

Byte	Size (bytes)	Subject Area	Description
128	1	Loopback capabilities	Diagnostics capability advertisements
129	1	Diagnostics measurement capabilities	
130	1	Diagnostic reporting capabilities	
131	1	Pattern Generation and Checking locations	
132-142	11	Pattern Generation and Checking capabilities	
143	1	-	**Reserved[1]** for module advertisements
144-151	8	Pattern Generator controls, host side	Host controls
152-159	8	Pattern Generator controls, media side	
160-167	8	Pattern Checker controls, host side	
168-175	8	Pattern Checker controls, media side	
176-179	4	Clocking and Measurement controls	
180-183	4	Loopback controls	
184-195	12	-	**Reserved[12]**
196-205	10	**-**	**Custom[10]**
206-223	18	Masks for Diagnostics Flags	Masks for Flags in Bytes 14h:132-149
224-255	32	User Pattern	User defined 32-byte pattern

8.11.1 Loopback Capabilities Advertisement

Four different types of loopback are defined by this specification. Figure 8-3 illustrates each type characterized by the **location of the loopback** on Host or Media Side and the **direction of the signal** being looped-back.

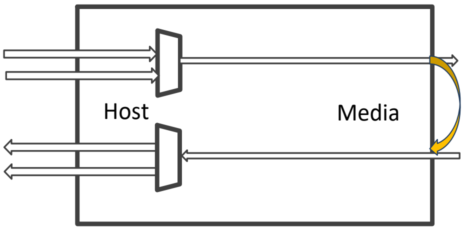
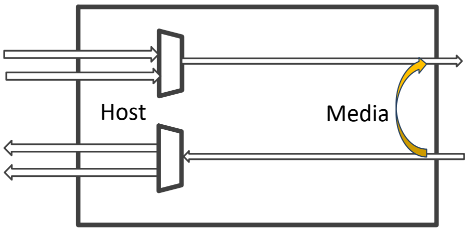
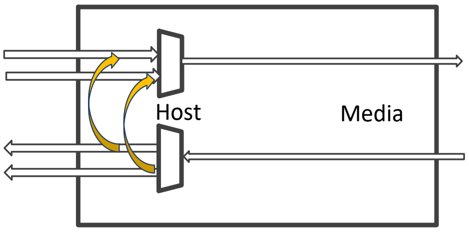
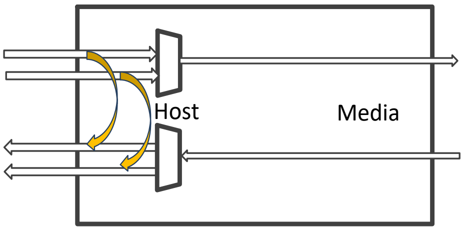
Name	Illustration
Media Side Output Loopback (only one media lane shown) advertised in 13h:128.0	
Media Side Input Loopback (only one media lane shown) advertised in 13h:128.1	
Host Side Output Loopback (only one media lane shown) advertised in 13h:128.2	
Host Side Input Loopback (only one media lane shown) advertised in 13h:128.3	

Figure 8-3 Loopback Type Illustrations

The loopback capabilities of the module are advertised as described in 13h:128 (see Table 8-89).

Table 8-89 Loopback Capabilities (Page 13h)

Byte	Bits	Field Name	Field Description	Type
128	7	-	**Reserved**	RO Rqd.
	6	SimultaneousHostAndMediaSideLoopbacks	0b: not supported 1b: supported	
	5	PerLaneMediaSideLoopbacks		
	4	PerLaneHostSideLoopbacks		
	3	HostSideInputLoopback		
	2	HostSideOutputLoopback		
	1	MediaSideInputLoopback		
	0	MediaSideOutputLoopback		

8.11.2 Diagnostics Measurement Capabilities Advertisement

Diagnostics measurement capabilities of the module are advertised in Table 8-90.

Table 8-90 Diagnostics Measurement Capabilities (Page 13h)

Byte	Bits	Field Name	Field Description	Type
129	7-6	GatingSupport	Gating (measurement over a given time interval) is 00b: Not supported (host defined measurement time) 01b: Supported with time accuracy <= 2 ms 10b: Supported with time accuracy <= 20 ms 11b: Supported with time accuracy > 20 ms *Note: The accuracy specifies the interval length error magnitude in the real-time time base of the module. Interval length may vary, e.g. by scheduling jitter in the module firmware.* *Note: The total bits counted and reported can be used as an accurate measure of the actual time interval length*	RO Rqd.
	5	GatingResultsSupported	Gating result statistics selectable by DiagnosticsSelector (14h:128) values **11h-15h** are 0b: Not supported 1b: Supported	
	4	PeriodicUpdatesSupported	Real-time statistics selectable by DiagnosticsSelector (14h:128) values **01h-06h** are periodically updated during measurement 0b: no periodic update during measurement 1b: periodic update during measurement *Note: The update rate of periodic update is selected by UpdatePeriodSelect (13h:177.0)* *Note: Single-shot measurements (un-gated non-periodic) can be started by enabling checkers and stopped by ResetErrorInformation (13h:177.5)*	
	3	PerLaneGatingTimersSupported	0b: Only two global gating timers are available for all lanes on all Banks, one for Host Side Measurements and one for Media Side Measurements. 1b: Per lane gating timers are supported in all Banks	
	2	AutoRestartGatingSupported	0b: AutoRestartGating control (13h:177.4) not supported 1b: AutoRestartGating control (13h:177.4) supported	
	1-0	-	**Reserved**	

8.11.3 Diagnostic Reporting Capabilities Advertisement

The diagnostic reporting capabilities of the module are advertised in Table 8-91.

Table 8-91 Diagnostic Reporting Capabilities (Page 13h)

Byte	Bits	Field Name	Field Description	Type
130	7	MediaSideFEC	1b: Supported (PRBS error information available) 0b: Not supported	RO Rqd.
	6	HostSideFEC	1b: Supported (PRBS error information available) 0b: Not supported	
	5	MediaSideInputSNRMeasurement	Indicates if media side SNR measurement reported via DiagnosticsSelector value 06h is supported (see Table 8-114) 1b: Supported 0b: Not supported	
	4	HostSideInputSNRMeasurement	Indicates if host side SNR measurement reported via Diagnostics Selection value **06h** is supported (Byte 14h:128, see Table 8-114) 1b: Supported 0b: Not supported	
	3	-	**Reserved**	
	2	-	**Reserved**	
	1	BitsAndErrorsCountingSupported	Indicates if DiagnosticsSelector values **02h-05h** are supported (Page 14h byte 128, Table 8-114) 1b: Supported 0b: Not supported *Some modules are unable to perform a 64-bit division to calculate and present BER. It is expected these types of module will present the BER as error counts and total bits elapsed for the error counts presented.*	
	0	BitErrorRatioResultsSupported	Indicates if DiagnosticsSelector value **01h** is supported (See Table 8-114) 1b: Supported 0b: Not supported	

8.11.4 Pattern Generation and Checking Location Advertisement

The support options for pattern generator and pattern checkers are advertised in 13h:131 (see Table 8-92).

Note: Following common practice, the terms 'pattern' and 'PRBS' are used interchangeably.

Figure 8-4 shows a reference diagram of possible locations for PRBS generators, PRBS checkers, FEC encoders and FEC decoders in the conceptual data path of an Application. The encircled numbers indicate the associated advertisement bit in 13h:131.

Note: The terms Pre-FEC and Post-FEC refer to the position of a generator or checker in the data path, before or after the FEC encoder for pattern generators, and before or after the FEC decoder for pattern checkers. In modules without FEC, only Post-FEC Encoders and Pre-FEC decoders exist.

Note: FEC usage and type of FEC depends on the Applications supported by the module (see section 6.2).

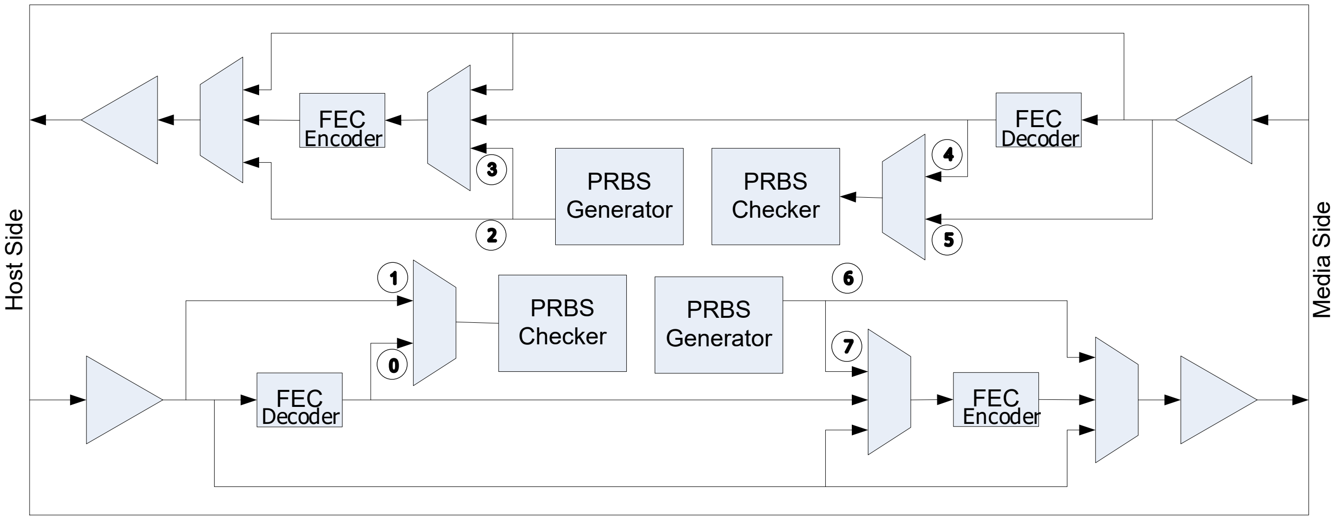


Figure 8-4 PRBS Paths Reference Diagram

Table 8-92 Pattern Generation and Checking Location (Page 13h)

Byte	Bits	Field Name	Field Description	Type
131	7	PRBSGeneratorMediaSidePreFEC	1b: Supported 0b: Not supported *Note: see reference diagram in Figure 8-4*	RO Rqd.
	6	PRBSGeneratorMediaSidePostFEC		
	5	PRBSCheckerMediaSidePreFEC		
	4	PRBSCheckerMediaSidePostFEC		
	3	PRBSGeneratorHostSidePreFEC		
	2	PRBSGeneratorHostSidePostFEC		
	1	PRBSCheckerHostSidePreFEC		
	0	PRBSCheckerHostSidePostFEC		

8.11.5 Pattern Generation and Checking Capabilities Advertisement

This specification defines sixteen distinguished patterns that may be generated and/or checked by the module.

These patterns are identified by a Pattern ID according to Table 8-93.

All patterns names ending in 'Q' denote patterns for PAM4 modulation using Gray coding (see section 3.3).

Note: The signal rate for these patterns is determined by Application on the associated lane.

Table 8-93 Pattern IDs

Pattern ID	Pattern Name	Pattern Description
0	PRBS-31Q	As defined in IEEE 802.3-2018 clause 120.5.11.2.2
1	PRBS-31	
2	PRBS-23Q	ITU-T Recommendation O.172, 2005
3	PRBS-23	
4	PRBS-15Q	x^15 + x^14 + 1
5	PRBS-15	
6	PRBS-13Q	As defined in IEEE 802.3-2018 clause 120.5.11.2.1
7	PRBS-13	
8	PRBS-9Q	As defined in IEEE 802.3-2018 clause 120.5.11
9	PRBS-9	
10	PRBS-7Q	x^7 + x^6 + 1
11	PRBS-7	
12	SSPRQ	As defined in IEEE 802.3-2018 clause 120.5.11.2.3
13	-	**Reserved**
14	Custom	Vendor defined pattern
15	User Pattern	Programmable pattern provided in Bytes 13h:224-255

The **pattern generation capabilities** of the module are advertised in registers described in Table 8-94.

Table 8-94 PRBS Pattern Generation Capabilities (Page 13h)

Byte	Bits	Field Name	Register Description	Type
132	7	HostSideGeneratorSupportsPattern7	**HostSideGeneratorSupportsPattern<i>** Pattern with ID <i> (see Table 8-93) 1b: Supported 0b: Not supported (Little Endian)	RO Rqd.
	6	HostSideGeneratorSupportsPattern6		
	5	HostSideGeneratorSupportsPattern5		
	4	HostSideGeneratorSupportsPattern4		
	3	HostSideGeneratorSupportsPattern3		
	2	HostSideGeneratorSupportsPattern2		
	1	HostSideGeneratorSupportsPattern1		
	0	HostSideGeneratorSupportsPattern0		
133	7	HostSideGeneratorSupportsPattern15		RO Rqd.
	6	HostSideGeneratorSupportsPattern14		
	5	HostSideGeneratorSupportsPattern13		
	4	HostSideGeneratorSupportsPattern12		
	3	HostSideGeneratorSupportsPattern11		
	2	HostSideGeneratorSupportsPattern10		
	1	HostSideGeneratorSupportsPattern9		
	0	HostSideGeneratorSupportsPattern8		
134	7	MediaSideGeneratorSupportsPattern7	**MediaSideGeneratorSupportsPattern<i>** Pattern <i> (see Table 8-93 for Pattern IDs) 1b: Supported 0b: Not supported (Little Endian)	RO Rqd.
	6	MediaSideGeneratorSupportsPattern6		
	5	MediaSideGeneratorSupportsPattern5		
	4	MediaSideGeneratorSupportsPattern4		
	3	MediaSideGeneratorSupportsPattern3		
	2	MediaSideGeneratorSupportsPattern2		
	1	MediaSideGeneratorSupportsPattern1		
	0	MediaSideGeneratorSupportsPattern0		
135	7	MediaSideGeneratorSupportsPattern15		RO

Byte	Bits	Field Name	Register Description	Type
	6	MediaSideGeneratorSupportsPattern14		Rqd.
	5	MediaSideGeneratorSupportsPattern13		
	4	MediaSideGeneratorSupportsPattern12		
	3	MediaSideGeneratorSupportsPattern11		
	2	MediaSideGeneratorSupportsPattern10		
	1	MediaSideGeneratorSupportsPattern9		
	0	MediaSideGeneratorSupportsPattern8		

The **pattern checking capabilities** of the module are advertised in registers described in Table 8-95.

Table 8-95 Pattern Checking Capabilities (Page 13h)

Byte	Bits	Name	Description	Type
136	7	HostSideCheckerSupportsPattern7	**HostSideCheckerSupportsPattern<i>** Pattern <i> (see Table 8-93 for Pattern IDs) 1b: Supported 0b: Not supported (Little Endian)	RO Rqd.
	6	HostSideCheckerSupportsPattern6		
	5	HostSideCheckerSupportsPattern5		
	4	HostSideCheckerSupportsPattern4		
	3	HostSideCheckerSupportsPattern3		
	2	HostSideCheckerSupportsPattern2		
	1	HostSideCheckerSupportsPattern1		
	0	HostSideCheckerSupportsPattern0		
137	7	HostSideCheckerSupportsPattern15		RO Rqd.
	6	HostSideCheckerSupportsPattern14		
	5	HostSideCheckerSupportsPattern13		
	4	HostSideCheckerSupportsPattern12		
	3	HostSideCheckerSupportsPattern11		
	2	HostSideCheckerSupportsPattern10		
	1	HostSideCheckerSupportsPattern9		
	0	HostSideCheckerSupportsPattern8		
138	7	MediaSideCheckerSupportsPattern7	**MediaSideCheckerSupportsPattern<i>** Pattern <i> (see Table 8-93 for Pattern IDs) 1b: Supported 0b: Not supported (Little Endian)	RO Rqd.
	6	MediaSideCheckerSupportsPattern6		
	5	MediaSideCheckerSupportsPattern5		
	4	MediaSideCheckerSupportsPattern4		
	3	MediaSideCheckerSupportsPattern3		
	2	MediaSideCheckerSupportsPattern2		
	1	MediaSideCheckerSupportsPattern1		
	0	MediaSideCheckerSupportsPattern0		
139	7	MediaSideCheckerSupportsPattern15		RO Rqd.
	6	MediaSideCheckerSupportsPattern14		
	5	MediaSideCheckerSupportsPattern13		
	4	MediaSideCheckerSupportsPattern12		
	3	MediaSideCheckerSupportsPattern11		
	2	MediaSideCheckerSupportsPattern10		
	1	MediaSideCheckerSupportsPattern9		
	0	MediaSideCheckerSupportsPattern8		

Additional pattern capabilities are advertised in registers described in Table 8-96.

Table 8-96 Pattern Generator and Checker swap and invert Capabilities (Page 13h)

Byte	Bits	Field Name	Field Description	Type
140	7-6	RecoveredClockForGeneratorOptions	Options to use recovered clock for contra-directional pattern generator on the same module side 00b: not supported 01b: supported without loopback 10b: supported with loopback 11b: supported with and without loopback	RO Rqd.

Byte	Bits	Field Name	Field Description	Type
	5	ReferenceClockForPatternsSupported	Option to use reference clock for pattern generation 1b/0b: supported/not supported	
	4		**Reserved**	
	3-0	UserPatternLengthSupported	U4 Maximum length L of the user defined pattern, where the field value n encodes L as L=2(n+1), i.e. 0000b: 2 bytes, …, 1111b: 32 bytes	
141	7	MediaSideCheckerSupportsDataSwap	0b/1b: Byte 13h:170 not supported/supported	RO Rqd.
	6	MediaSideCheckerSupportsDataInvert	0b/1b: Byte 13h:169 not supported/supported	
	5	MediaSideGeneratorSupportsDataSwap	0b/1b: Byte 13h:154 not supported/supported	
	4	MediaSideGeneratorSupportsDataInvert	0b/1b: Byte 13h:153 not supported/supported	
	3	HostSideCheckerSupportsDataSwap	0b/1b: Byte 13h:162 not supported/supported	
	2	HostSideCheckerSupportsDataInvert	0b/1b: Byte 13h:161 not supported/supported	
	1	HostSideGeneratorSupportsDataSwap	0b/1b: Byte 13h:146 not supported/supported	
	0	HostSideGeneratorSupportsDataInvert	0b/1b: Byte 13h:145 not supported/supported	
142	7	MediaCheckerSupportsPerLaneEnable	Media side pattern checker for lane i enabled in 13h:168 enables lane i (or all lanes of the Bank) 0b/1b: per lane enable not supported/supported	RO Rqd.
	6	MediaCheckerSupportsPerLanePattern	Media side pattern selection for checker 0b: Lane 1 pattern 13h:172.3-0 is used for all lanes 1b: Per lane pattern selection in 13h:172-175	
	5	MediaGeneratorSupportsPerLaneEnable	Media side pattern generator for lane i enabled in 13h:152 enables lane i (or all lanes of the Bank) 0b/1b: per lane enable not supported/supported	
	4	MediaGeneratorSupportsPerLanePattern	Media side pattern selection for generator 0b: Lane 1 pattern 13h:156.3-0 is used for all lanes 1b: Per lane pattern selection in 13h:156-159	
	3	HostCheckerSupportsPerLaneEnable	Host side pattern checker for lane i enabled in 13h:160 enables lane i (or all lanes of the Bank) 0b/1b: per lane enable not supported/supported	
	2	HostCheckerSupportsPerLanePattern	Host side pattern selection for checker 0b: Lane 1 pattern 13h:164.3-0 is used for all lanes 1b: Per lane pattern selection in 13h:164-167	
	1	HostGeneratorSupportsPerLaneEnable	Host side pattern generator for lane i enabled in 13h:144 enables lane i (or all lanes of the Bank) 0b/1b: per lane enable not supported/supported	
	0	HostGeneratorSupportsPerLanePattern	Host side pattern selection for generator 0b: Lane 1 pattern 13h:148.3-0 is used for all lanes 1b: Per lane pattern selection in 13h:148-151	
143	7-0	-	**Reserved[1]**	RO

8.11.6 Host Side Pattern Generator Controls

The controls in this section control pattern generation on the host side of the module in the direction of the Rx electrical output.

Table 8-97 defines the host side pattern generator controls and Table 8-98 defines the host side pattern generator selection controls.

Table 8-97 Host Side Pattern Generator Controls (Page 13h)

Byte	Bits	Field Name	Register Description	Type
144	7	HostSideGeneratorEnableLane8	1b: Enable generator with configuration defined in Bytes 145-151 0b: Disable pattern generator Advertisement: 13h:131.7-6	RW Adv.
	6	HostSideGeneratorEnableLane7		
	5	HostSideGeneratorEnableLane6		
	4	HostSideGeneratorEnableLane5		
	3	HostSideGeneratorEnableLane4		
	2	HostSideGeneratorEnableLane3		
	1	HostSideGeneratorEnableLane2		
	0	HostSideGeneratorEnableLane1		
145	7	HostSideGeneratorDataInvertLane8	1b: Invert the selected pattern 0b: Do not invert the selected pattern *Note: This control inverts the pattern; it does not swap the P and N signals (see polarity controls in Table 8-61.)*	RW Adv.
	6	HostSideGeneratorDataInvertLane7		
	5	HostSideGeneratorDataInvertLane6		
	4	HostSideGeneratorDataInvertLane5		
	3	HostSideGeneratorDataInvertLane4		
	2	HostSideGeneratorDataInvertLane3		
	1	HostSideGeneratorDataInvertLane2		
	0	HostSideGeneratorDataInvertLane1		
146	7	HostSideGeneratorByteSwapLane8	1b: Swap MSB and LSB for PAM4 patterns 0b: Do not swap MSB and LSB	RW Adv.
	6	HostSideGeneratorByteSwapLane7		
	5	HostSideGeneratorByteSwapLane6		
	4	HostSideGeneratorByteSwapLane5		
	3	HostSideGeneratorByteSwapLane4		
	2	HostSideGeneratorByteSwapLane3		
	1	HostSideGeneratorByteSwapLane2		
	0	HostSideGeneratorByteSwapLane1		
147	7	HostSideGeneratorPreFECEnableLane8	1b: Encoded pattern: Generate the selected pattern at the input to the internal host side FEC encoder 0b: Unencoded pattern: Generate the selected pattern at a location after the internal host side FEC encoder block	RW Adv.
	6	HostSideGeneratorPreFECEnableLane7		
	5	HostSideGeneratorPreFECEnableLane6		
	4	HostSideGeneratorPreFECEnableLane5		
	3	HostSideGeneratorPreFECEnableLane4		
	2	HostSideGeneratorPreFECEnableLane3		
	1	HostSideGeneratorPreFECEnableLane2		
	0	HostSideGeneratorPreFECEnableLane1		

Table 8-98 Host Side Pattern Generator Pattern Select Controls (Page 13h)

Byte	Bits	Field Name	Field Description	Type
148	7-4	HostSideGeneratorPatternSelectLane2	Selected pattern to be generated on each lane. See Table 8-93 for pattern coding.	RW Adv.
	3-0	HostSideGeneratorPatternSelectLane1		
149	7-4	HostSideGeneratorPatternSelectLane4		
	3-0	HostSideGeneratorPatternSelectLane3		
150	7-4	HostSideGeneratorPatternSelectLane6		
	3-0	HostSideGeneratorPatternSelectLane5		
151	7-4	HostSideGeneratorPatternSelectLane8		
	3-0	HostSideGeneratorPatternSelectLane7		

8.11.7 Media Side Pattern Generator Controls

The controls in this section control pattern generation on the media side of the module in the direction of the Tx electrical or optical output.

Table 8-99 defines the media side pattern generator controls and Table 8-100 defines the media side pattern generator selection controls.

Table 8-99 Media Side Pattern Generator Controls (Page 13h)

Byte	Bits	Field Name	Register Description	Type
152	7	MediaSideGeneratorEnableLane8	1b: Enable pattern generator with configuration defined in Bytes 153-159 0b: Disable pattern generator Advertisement: 01h:161.7-6	RW Adv.
	6	MediaSideGeneratorEnableLane7		
	5	MediaSideGeneratorEnableLane6		
	4	MediaSideGeneratorEnableLane5		
	3	MediaSideGeneratorEnableLane4		
	2	MediaSideGeneratorEnableLane3		
	1	MediaSideGeneratorEnableLane2		
	0	MediaSideGeneratorEnableLane1		
153	7	MediaSideGeneratorDataInvertLane8	1b: Invert the selected pattern 0b: Do not invert the selected pattern *Note: This control inverts the pattern; it does not swap the P and N signals (see polarity controls in Table 8-61)*	RW Adv.
	6	MediaSideGeneratorDataInvertLane7		
	5	MediaSideGeneratorDataInvertLane6		
	4	MediaSideGeneratorDataInvertLane5		
	3	MediaSideGeneratorDataInvertLane4		
	2	MediaSideGeneratorDataInvertLane3		
	1	MediaSideGeneratorDataInvertLane2		
	0	MediaSideGeneratorDataInvertLane1		
154	7	MediaSideGeneratorByteSwapLane8	1b: Swap MSB and LSB for PAM4 patterns 0b: Do not swap MSB and LSB	RW Adv.
	6	MediaSideGeneratorByteSwapLane7		
	5	MediaSideGeneratorByteSwapLane6		
	4	MediaSideGeneratorByteSwapLane5		
	3	MediaSideGeneratorByteSwapLane4		
	2	MediaSideGeneratorByteSwapLane3		
	1	MediaSideGeneratorByteSwapLane2		
	0	MediaSideGeneratorByteSwapLane1		
155	7	MediaSideGeneratorPreFECEnableLane8	1b: Encoded pattern: Generate the selected pattern at the input to the internal media side FEC encoder 0b: Unencoded pattern: Generate the selected pattern at a location after the internal media side FEC encoder block	RW Adv.
	6	MediaSideGeneratorPreFECEnableLane7		
	5	MediaSideGeneratorPreFECEnableLane6		
	4	MediaSideGeneratorPreFECEnableLane5		
	3	MediaSideGeneratorPreFECEnableLane4		
	2	MediaSideGeneratorPreFECEnableLane3		
	1	MediaSideGeneratorPreFECEnableLane2		
	0	MediaSideGeneratorPreFECEnableLane1		

Table 8-100 Media Side Pattern Generator Pattern Select Controls (Page 13h)

Byte	Bits	Field Name	Register Description	Type
156	7-4	MediaSideGeneratorPatternSelectLane2	Selected pattern to be generated on each lane. See Table 8-93 for pattern coding.	RW Adv.
	3-0	MediaSideGeneratorPatternSelectLane1		
157	7-4	MediaSideGeneratorPatternSelectLane4		
	3-0	MediaSideGeneratorPatternSelectLane3		
158	7-4	MediaSideGeneratorPatternSelectLane6		
	3-0	MediaSideGeneratorPatternSelectLane5		
159	7-4	MediaSideGeneratorPatternSelectLane8		
	3-0	MediaSideGeneratorPatternSelectLane7		

8.11.8 Host Side Pattern Checker Controls

The controls in this section control pattern checking on the host side of the module for data arriving in the direction of the Tx electrical input.

Table 8-101 defines the host side pattern checker controls and Table 8-102 defines the host side pattern checker selection controls.

Table 8-101 Host Side Pattern Checker Controls (Page 13h)

Byte	Bits	Field Name	Register Description	Type
160	7	HostSideCheckerEnableLane8	1b: Enable pattern checker with configuration defined in Bytes 161-167 0b: Disable pattern checker Advertisement: 01h:161.1-0	RW Adv.
	6	HostSideCheckerEnableLane7		
	5	HostSideCheckerEnableLane6		
	4	HostSideCheckerEnableLane5		
	3	HostSideCheckerEnableLane4		
	2	HostSideCheckerEnableLane3		
	1	HostSideCheckerEnableLane2		
	0	HostSideCheckerEnableLane1		
161	7	HostSideCheckerDataInvertLane8	1b: Invert the selected pattern 0b: Do not invert the selected pattern *Note: This control inverts the pattern; it does not swap the P and N signals (P/N swap for input signals is not currently supported by this specification).*	RW Adv.
	6	HostSideCheckerDataInvertLane7		
	5	HostSideCheckerDataInvertLane6		
	4	HostSideCheckerDataInvertLane5		
	3	HostSideCheckerDataInvertLane4		
	2	HostSideCheckerDataInvertLane3		
	1	HostSideCheckerDataInvertLane2		
	0	HostSideCheckerDataInvertLane1		
162	7	HostSideCheckerByteSwapLane8	1b: Swap MSB and LSB for PAM4 patterns 0b: Do not swap MSB and LSB	RW Adv.
	6	HostSideCheckerByteSwapLane7		
	5	HostSideCheckerByteSwapLane6		
	4	HostSideCheckerByteSwapLane5		
	3	HostSideCheckerByteSwapLane4		
	2	HostSideCheckerByteSwapLane3		
	1	HostSideCheckerByteSwapLane2		
	0	HostSideCheckerByteSwapLane1		
163	7	HostSideCheckerPostFECEnableLane8	1b: Check the selected encoded pattern at the output from the internal FEC decoder 0b: Check the selected unencoded pattern at the input to the internal FEC decoder block	RW Adv.
	6	HostSideCheckerPostFECEnableLane7		
	5	HostSideCheckerPostFECEnableLane6		
	4	HostSideCheckerPostFECEnableLane5		
	3	HostSideCheckerPostFECEnableLane4		
	2	HostSideCheckerPostFECEnableLane3		
	1	HostSideCheckerPostFECEnableLane2		
	0	HostSideCheckerPostFECEnableLane1		

Table 8-102 Host Side Pattern Checker Pattern Select Controls (Page 13h)

Byte	Bits	Name	Register Description	Type
164	7-4	HostSideCheckerPatternSelectLane2	Selected pattern to be generated on each lane. See Table 8-93 for pattern coding.	RW Adv.
	3-0	HostSideCheckerPatternSelectLane1		
165	7-4	HostSideCheckerPatternSelectLane4		
	3-0	HostSideCheckerPatternSelectLane3		
166	7-4	HostSideCheckerPatternSelectLane6		
	3-0	HostSideCheckerPatternSelectLane5		
167	7-4	HostSideCheckerPatternSelectLane8		
	3-0	HostSideCheckerPatternSelectLane7		

8.11.9 Media Side Pattern Checker Controls

The controls in this section control pattern checking on the media side of the module for data arriving in the direction of the Rx electrical or optical input.

Table 8-103 defines the media side pattern checker controls and Table 8-104 defines the media side pattern checker selection controls.

Table 8-103 Media Side Pattern Checker Controls (Page 13h)

Byte	Bits	Field Name	Register Description	Type
168	7	MediaSideCheckerEnableLane8	1b: Enable pattern checker with configuration defined in Bytes 169-175 0b: Disable pattern checker Advertisement: 01h:161.5-4	RW Adv.
	6	MediaSideCheckerEnableLane7		
	5	MediaSideCheckerEnableLane6		
	4	MediaSideCheckerEnableLane5		
	3	MediaSideCheckerEnableLane4		
	2	MediaSideCheckerEnableLane3		
	1	MediaSideCheckerEnableLane2		
	0	MediaSideCheckerEnableLane1		
169	7	MediaSideCheckerDataInvertLane8	1b: Invert the selected pattern 0b: Do not invert the selected pattern *Note: This control inverts the pattern; it does not swap the P and N signals (P/N swap for input signals is not currently supported by this specification).*	RW Adv.
	6	MediaSideCheckerDataInvertLane7		
	5	MediaSideCheckerDataInvertLane6		
	4	MediaSideCheckerDataInvertLane5		
	3	MediaSideCheckerDataInvertLane4		
	2	MediaSideCheckerDataInvertLane3		
	1	MediaSideCheckerDataInvertLane2		
	0	MediaSideCheckerDataInvertLane1		
170	7	MediaSideCheckerByteSwapLane8	1b: Swap MSB and LSB for PAM4 patterns 0b: Do not swap MSB and LSB	RW Adv.
	6	MediaSideCheckerByteSwapLane7		
	5	MediaSideCheckerByteSwapLane6		
	4	MediaSideCheckerByteSwapLane5		
	3	MediaSideCheckerByteSwapLane4		
	2	MediaSideCheckerByteSwapLane3		
	1	MediaSideCheckerByteSwapLane2		
	0	MediaSideCheckerByteSwapLane1		
171	7	MediaSideCheckerPostFECEnableLane8	1b: Check the selected encoded pattern at the output from the internal FEC decoder 0b: Check the selected unencoded pattern at the input to the internal FEC decoder block	RW Adv.
	6	MediaSideCheckerPostFECEnableLane7		
	5	MediaSideCheckerPostFECEnableLane6		
	4	MediaSideCheckerPostFECEnableLane5		
	3	MediaSideCheckerPostFECEnableLane4		
	2	MediaSideCheckerPostFECEnableLane3		
	1	MediaSideCheckerPostFECEnableLane2		
	0	MediaSideCheckerPostFECEnableLane1		

Table 8-104 Media Side Pattern Checker Select Controls (Page 13h)

Byte	Bits	Field Name	Register Description	Type
172	7-4	MediaSideCheckerPatternSelectLane2	Selected pattern to be generated on each lane. See Table 8-93 for pattern coding.	RW Adv.
	3-0	MediaSideCheckerPatternSelectLane1		
173	7-4	MediaSideCheckerPatternSelectLane4		
	3-0	MediaSideCheckerPatternSelectLane3		
174	7-4	MediaSideCheckerPatternSelectLane6		
	3-0	MediaSideCheckerPatternSelectLane5		
175	7-4	MediaSideCheckerPatternSelectLane8		
	3-0	MediaSideCheckerPatternSelectLane7		

8.11.10 Clocking and Measurement Controls

Table 8-105 describes general controls for the pattern generator and checker features.

Examples showing the usage of the Pattern Generator/Checker controls are described in Appendix E.

Table 8-105 Clocking and Measurement Controls (Page 13h)

Byte	Bits	Field Name	Field Description	Type
176	7-4	HostPRBSGeneratorClockSource	Clock source for Host Side PRBS Pattern Generation: 0: All lanes use Internal Clock **i**=1-8: All lanes use Reference Clock Media Lane **i** 9-14: **Reserved** 15: Recovered clock per Media Lane or Data Path	RW Opt.
	3-0	MediaPRBSGeneratorClockSource	Clock source for Media Side PRBS Pattern Generation: 0: All lanes use Internal Clock 1: All lanes use Reference Clock **i**= 2-9: All lanes use Reference Clock Host Lane **i-1** 10-14: **Reserved** 15: Recovered clock per Host Lane or Data Path	
177	7	StartStopIsGlobal	Bool **StartStopIsGlobal** controls whether writing a measurement Start/Stop control (see below for a list) on one Bank globally starts/stops measurements across all Banks **as if** the same control value **change** had occurred **in all** supported **Banks synchronously** (which is impossible): 0b: A start/stop control change acts on current Bank 1b: A start/stop control change acts on all Banks **Start/Stop controls** (instances) **Locations** ResetErrorInformation (4) (0-3:13h:177.5) HostSideCheckerEnableLane (32) (0-3:13h:160.0-7) MediaSideCheckerEnableLane (32) (0-3:13h:168.0-7) *Note: The **intended effect** of StartStopIsGlobal is that starting or stopping error counting measurement in one supported Bank occurs **on all affected lanes in all supported Banks synchronously**.* *Note: The actually affected lanes depend on per Bank capabilities like *CheckerSupportsPerLaneEnable.*	RW Opt.
	6	-	**Reserved**	
	5	ResetErrorInformation	This bit has effects on error information, and on gating timers when the value changes. **Effect on Error Information** **0b→ 1b**: Currently accumulating error statistics identified by DiagnosticsSelector 01h to 05h are frozen and, if supported, gated results identified by Selectors 11h-15h are updated with the frozen current error statistics. **1b→ 0b**: Error statistics identified by Selectors 01h-05h are reset to 0, whereas results identified by Selectors 11h-15h are unaffected. *Whenever a ResetErrorInformation operation on an individual lane or all lanes is triggered, it should always clear associated Error Information (both PRBS BER and Error Counters) in the Diagnostics Data memory at the same time.* The effect of ResetErrorInformation depends also on - PerLaneGatingTimersSupported (13h:129.3)	

Byte	Bits	Field Name	Field Description	Type
			- StartStopIsGlobal (13h:177.7) **Effect on global gating timers** (13h:129.3=0b) 0b→1b: single gate timer is stopped 1b→0b: starts the single gating timer **Effect on per lane gating timers** (13h:129.3=1b) 1b: keeps the gate timers stopped for the lanes in this Bank (13h:177.7=0) or on all. 0b: starts the individual gating timers for all enabled lanes in this Bank.	
	4	AutoRestartGating	0b: When Gate Time expires, the module will set the Gate Complete Flag in Bytes 14h:134-135. The module will update the error counter result and stop error detection. 1b: When Gate Time expires, the module will set the Gate Complete Flag in Bytes 14h:134-135. The module will update the error counter results, then immediately clear internal error counters and restart the Gate Timer, while continuing to count errors. *Note: With Auto-restart the host must read the error counter results before the Gate Time expires to avoid the result values to be overwritten with new results.*	
	3-1	MeasurementTime	Measurement (gating) time for one complete result over a defined measurement period. 000b: ungated, counters accrue indefinitely (infinite gate time 001b: 5 sec gate time 010b: 10 sec gate time 011b: 30 sec gate time 100b: 60 sec gate time 101b: 120 sec gate time 110b: 300 sec gate time 111b: **Custom**	
	0	UpdatePeriodSelect	Time between incremental updates to intermediate error counting results during a longer gating period 0b: 1 sec update interval 1b: 5 sec update interval This period is relevant only if PeriodicUpdatesSupported (13h:129.4=1b) advertises that the module will update the BER or Error counters while gating is in progress. ~~Then two update rates at which the module will process the internal error counters and update the error counter fields in the Memory Map are possible.~~ The host can then poll "real-time" error counts, up until the gating results are completed.	
178	7-4	-	**Reserved**	RW
	3-2	HostPRBSCheckerClockSource	The clock source used for the Host PRBS Pattern Checker can be configured using this control register: 0h: Recovered clocks from Host Lane/Data paths 1h: All lanes use Internal Clock 2h: All lanes use Reference Clock 3h: Reserved	RW
	1-0	MediaPRBSCheckerClockSource	The clock source used for the Media PRBS Pattern Checker can be configured using this control register: 0h: Recovered clocks from Media Lane/Data paths 1h: All lanes use Internal Clock	RW

Byte	Bits	Field Name	Field Description	Type
			2h: All lanes use Reference Clock 3h: Reserved.	
179	7-0	-	**Reserved[1]**	RW

8.11.11 Diagnostics Measurement Behavior

Editor's Note: Section should be revised for better clarity in the next revision of this specification

8.11.11.1 Un-Gated Measurements

This section describes error performance measurements when gating is not used (13h:177.3-1= 000b).

When intermediate periodic updates of error metrics are supported (PeriodicUpdatesSupported=1), the retrievable measurement results are cumulative, and progressively updated until the measurement may eventually be stopped by the host. These intermediate periodic updates are not indicated by a Flag, so the host needs to poll the updates.

Without periodic updates, measurements without gating are single-shot and results are available only when the measurement has been stopped by the host.

Editor's Note: The following table will be consolidated in the next revision of this specification.

Table 8-106 PRBS Checker Behavior Un-Gated Mode

Configuration	Byte 177 (D7-D0)	Description
177.3-1=000b **Not Gated** (Bit 4 ignored) 177.7 **StartStopIsGlobal** = 0	**0**00x 000**y**	In this mode, the error metrics measurement runs continuously. When the host enables disabled PRBS checkers (in 13h:160 or in 13h:168) all error counters for the enabled lanes are cleared and then start accumulating. When the host disables enabled PRBS checkers (in 13h:160 or in 13h:168) error counting is stopped, and error counting results will be available both via Selector 01-05h and 11h-15h (if supported). If 13h:129.4=0, real time error information is not updated and error information is only available when the error counting is stopped by checker disable. If 13h:129.4 = 1, real time error information is available with Selectors 01h-05h, updated period configured by **y**. Accurate error counter accumulation times can be derived from the total bit counters in the error information. A write to 13h:177. 5=1 also cause the error information registers to reset to 0 and restart accumulation on the enabled lanes.
177.3-1=000b. **Not Gated** (Bit 4 ignored) 177.7 **StartStopIsGlobal** = 1	**1**00x 000**y**	The behavior is the same as above with the exception that a write to 13h:177. 5=1 causes the error information registers of **all lanes in all Banks** to reset and restart accumulation. As described above just prior to reset of error information of all lanes in all Banks, the previous error information should be copied to both 01-05h (and 11h-15h if supported). Selector 01-05h restarts accumulation and 11h-15h (if supported) contains the error counting results of the period prior to reset.
Reset Error Counters (Bit 5) Assuming 13h:129.3=0 there is only one timer for host and one timer for media lanes and 13h:177.7 has no relevance.	x01x 000**y**	This bit is used to clear the error counters and restart accumulation of errors. When 13h:177.5 is set to 1b, all the enabled lanes error counters are frozen from the previous state and available from reading Selectors 01h-05h (and 11h-15h if supported). When 13h:177.5 is set to 0b, all the error information of enabled PRBS checker of <u>both</u> host and media lanes are reset. The host may also individually toggle Host Byte 160 or Media Byte 168 enable bits to restart error counting of the lanes independently.

8.11.11.2Gated Measurements with Global Gate Timer

This section describes error performance measurements with gating based on one Gate Timer for all lanes and all Banks (13h:129.3 = 0) on Host Side, or on Media Side, respectively.

Bit 13h:129.4 defines availability of real-time error information while the gating timer has not expired.

When gating is enabled, the Flags in Page 14h:s 134-135 will be raised at the expiry of the gate timers to indicate that new error information results are available.

If the host fails to read the error information registers prior to expiry of subsequent gate time expiry, only the latest error information will be available.

Table 8-107 PRBS Checker Behavior Single Gate Timer

Configuration	**Byte 177** (D7-D0)	**Description**
Gated, nnn configured secs. Since 13h:129.3=0 the control 13h:177.7 is ignored.	X000 *nnny* *nnn* ≠ 000	PRBS Error Counters are gated. Whenever PRBS checkers are enabled in 13h:160 or 13h:168, the respective Host or Media single gate timer resets to 0, all error counters for the enable lanes reset to 0 and start accumulating errors. When the gate timer expires, the error counters stop counting. At this time the module shall be able to present the error information collected within this gating period on Page 14h by using the Selectors 01h-05h (and 11h-15h if supported). If 13h:129.4 = 0, real time error information is not updated while gating is in progress. The error information will only be available at the end of the gate. If 13h:129.4 = 1, real time error information is available by using Page 14h using the Selectors 01h-05h. These real time error information will be updated by the module every configure "y" seconds. This mode is useful if the gating period is long. A host may choose to periodically read the real-time error information during gating and take any necessary action if a bad BER is detected. To restart gating, the host toggles the PRBS enable registers 160 and 168.
Gated, nnn configured secs. Since 13h:129.3=0 the control 13h:177.7 is ignored. If the Page 13h:129.2 is set, and the host writes a 1 to Bit4 of Byte 177, the gating timer will automatically restart.	Xx01 *nnny* *nnn* ≠ 000	This behavior is the same as the above row, except for the behavior when the gate timer expires and exceeds the configured gate time. Module should support Selectors 11h-15h to use this feature, otherwise error information from the previous gating period will be lost. Here at the end of the gate, that is when the gate timer expires and exceeds the configured "nnn" elapsed time, the PRBS error information will be presented in the error information results via Selectors 11h-15h. The current error information will reset, and the gate timer will be reset to 0 and restart accumulating errors for a new gating period. The host will have to read the error information from the previous gated time using the error information results Selectors 11h-15h.
Restart gate timer (Bit 5) aka ResetErrorInformation Since 13h:129.3=0 there is only one timer for host and one timer for media lanes and 13h:177.7 is irrelevant.	X01x *nnny* *nnn* ≠ 000	This bit is used to restart the both the single host and media gate timers. When Byte177.5 is set to 1b, all the enabled lanes error counters are frozen from the previous state and the gate timers are stopped. When Byte177.5 is set to 0b, both the host and media enabled the gate timers are restarted from 0 and the error information are reset, and a new error accumulation gate period is restarted. The host may also individually toggle host Byte 160 or media Byte 168 enable bits to restart the gate timer for the host and media lanes independently.

8.11.11.3Gated Measurements with Per Lane Gate Timer

This section describes error performance measurements with gating based on individual Gate Timers for all lanes and on all Banks.

Byte 13h:129.4 defines the behavior of the module availability of the real-time error information while the gating timer is running and has not expired.

In the following table whenever gating is enabled, the COR Flags in Page 14h:s 134-135 will be raised as well as Interrupt may be asserted at the expiry of the gate timers to indicate that error information registers are available. If the host fails to read the error information registers prior to expiry of subsequent gate time expiry, only the latest error information will be available.

Table 8-108 PRBS Checker Behavior Per Lane Gate Timer

Configuration	**Byte 177** (D7-D0)	**Description**
Gated, nnn configured secs. Since 13h:129.3=0 the control 13h:177.7 is ignored.	X000 *nnny* *nnn* !=000	PRBS Error Counters are gated. Whenever PRBS checkers are enabled in 13h:160 or 13h:168, the respective Host or Media lane gate timers will reset to 0, error counters for the enabled lanes will reset to 0 and start accumulating errors. Since there are individual per lane gate timers, these timers reset independently to provide the most accurate per lane gated time as possible. When the gate time configured by nnn expires, the error counters stop counting and the host can inspect pr collect the results of the finished gating period in the Diagnostics Area on Page 14h, using the Selectors 01h-05h and 11h-15h (if supported). If 13h:129.4 = 0, real time error information is not updated while gating is in progress, but only at the end of the gate. If 13h:129.4 = 1, real time error information is available on Page 14h using the Selectors 01h-05h. These real time error information will be updated with period configured by **y**. This mode is useful if the gating period is long. A host may periodically read the real-time error information during gating and react if bad performance is detected. To restart gating, the host toggles the PRBS checker enable registers 160 and 168 or sets and ceases 13h:177.5.
Gated, nnn configured secs. Since 13h:129.3=0 the control 13h:177.7 is ignored. If the Page 13h:129.2 is set, and the host writes a 1 to Bit4 of Byte 177, the gating timer will automatically restart.	Xx01 *nnny* *nnn* !=000	This behavior is the same as the above row, except for the behavior when the gate timer expires and exceeds the configured gate time. Module shall support Selectors 11h-15h to use this feature. Here at the end of the gate, that is when the individual per lane gate timer expires and exceeds the configured "nnn" elapsed time, the PRBS error information will be presented in the error information results via Selectors 11h-15h. The current error information will reset, and the gate timer will be reset to 0 and restart accumulating errors for a new gating period. The host will have to read the error information from the previous gated time using the error information results Selectors 11h-15h.
Restart gate timer (Bit 5) aka Reset Error Information	x01x *nnny* *nnn* !=000	This bit is used to restart all enabled Bank and lanes gate timers. When Bit 177.5 is raised, all enabled lanes error counters are frozen, and the gate timers are stopped. If Byte177.7=0 when Bit 177.5 is ceased, all enabled gate timers of the **current** Bank are restarted from 0, error information is reset, and a new error accumulation gate period is started. If Byte177.7=1 when Bit 177.5 is ceased, all enabled gate timers of **all** Banks **with Bit 177.7=1** are restarted from 0, error information is reset, and a new error accumulation gate period is started. The host may also individually toggle host Byte 160 or media Byte 168 enable bits to restart the gate timer for the host and media lanes independently. In this case, only the error information of the

Configuration	Byte 177 (D7-D0)	Description
		enable or disabled lane will reset and start count or freeze in its last value respectively.

8.11.12 Loopback Controls

Host and Media side loopback control registers and module behaviors depend on the advertised loopback capabilities described in Table 8-89.

Controls for loopback features are described in Table 8-109.

The module may **reject** unsupported host-written loopback settings (no change in affected register bits).

For example, if the module advertises that it can only perform host side or media side loopback one at a time and not simultaneously, the module may reject the command such that the rejected values in the loopback controls do not change.

Table 8-109 Loopback Controls (Page 13h)

Byte	Bits	Field Name	Register Description	Type
180	7	MediaSideOutputLoopbackEnableLane8	**MediaSideOutputLoopbackEnableLane<i>** 0b: normal non-loopback operation 1b: loopback enabled. If the Per-lane Media Side Loopback Supported field=1, loopback control is per lane. Otherwise, if any loopback enable bit is set to 1, all Media side lanes are in output loopback. Advertisement: 13h:128.	RWW Adv.
	6	MediaSideOutputLoopbackEnableLane7		
	5	MediaSideOutputLoopbackEnableLane6		
	4	MediaSideOutputLoopbackEnableLane5		
	3	MediaSideOutputLoopbackEnableLane4		
	2	MediaSideOutputLoopbackEnableLane3		
	1	MediaSideOutputLoopbackEnableLane2		
	0	MediaSideOutputLoopbackEnableLane1		
181	7	MediaSideInputLoopbackEnableLane8	**MediaSideInputLoopbackEnableLane<i>** 0b: normal non-loopback operation 1b: loopback enabled. If the Per-lane Media Side Loopback Supported field=1, loopback control is per lane. Otherwise, if any loopback enable bit is set to 1, all media side lanes are in input loopback. Advertisement: 13h:128.1	RWW Adv.
	6	MediaSideInputLoopbackEnableLane7		
	5	MediaSideInputLoopbackEnableLane6		
	4	MediaSideInputLoopbackEnableLane5		
	3	MediaSideInputLoopbackEnableLane4		
	2	MediaSideInputLoopbackEnableLane3		
	1	MediaSideInputLoopbackEnableLane2		
	0	MediaSideInputLoopbackEnableLane1		
182	7	HostSideOutputLoopbackEnableLane8	**HostSideOutputLoopbackEnableLane<i>** 0b: normal non-loopback operation 1b: loopback enabled. If the Per-lane Host Side Loopback Supported field=1, loopback control is per lane. Otherwise, if any loopback enable bit is set to 1, all host side lanes are in output loopback. Advertisement: 13h:128.2	RWW Adv.
	6	HostSideOutputLoopbackEnableLane7		
	5	HostSideOutputLoopbackEnableLane6		
	4	HostSideOutputLoopbackEnableLane5		
	3	HostSideOutputLoopbackEnableLane4		
	2	HostSideOutputLoopbackEnableLane3		
	1	HostSideOutputLoopbackEnableLane2		
	0	HostSideOutputLoopbackEnableLane1		
183	7	HostSideInputLoopbackEnableLane8	**HostSideInputLoopbackEnableLane<i>** 0b: normal non-loopback operation 1b: loopback enabled. If the Per-lane Host Side Loopback Supported field=1, loopback control is per lane. Otherwise, if any loopback enable bit is set to 1, all Host side lanes are in input loopback. Advertisement: 13h:128.3	RWW Adv.
	6	HostSideInputLoopbackEnableLane7		
	5	HostSideInputLoopbackEnableLane6		
	4	HostSideInputLoopbackEnableLane5		
	3	HostSideInputLoopbackEnableLane4		
	2	HostSideInputLoopbackEnableLane3		
	1	HostSideInputLoopbackEnableLane2		
	0	HostSideInputLoopbackEnableLane1		

8.11.13 Diagnostics Masks

Table 8-110 provides Mask bits for all diagnostics Flags.

The default value for all Mask bits on this page is 1 (masked).

Diagnostics Flags are located on Page 14h (see Table 8-115).

Table 8-110 Diagnostics Masks (Page 13h)

Byte	Bits	Field Name	Field/Register Description	Type
206	7	LossOfReferenceClockMask	Loss of reference clock Mask for the module	RW Opt.
	6-0	-	**Reserved**	
207	7-0	-	**Reserved[1]**	RW
208	7	PatternCheckGatingCompleteMaskHostLane8	**PatternCheckGatingCompleteMaskHostLane<i>** Per-host lane gating complete Mask. Default: 1	RW Adv.
	6	PatternCheckGatingCompleteMaskHostLane7		
	5	PatternCheckGatingCompleteMaskHostLane6		
	4	PatternCheckGatingCompleteMaskHostLane5		
	3	PatternCheckGatingCompleteMaskHostLane4		
	2	PatternCheckGatingCompleteMaskHostLane3		
	1	PatternCheckGatingCompleteMaskHostLane2		
	0	PatternCheckGatingCompleteMaskHostLane1		
209	7	PatternCheckGatingCompleteMaskMediaLane8	**PatternCheckGatingCompleteMaskMediaLane<i>** Per-media lane gating complete Mask. Default: 1	RW Adv.
	6	PatternCheckGatingCompleteMaskMediaLane7		
	5	PatternCheckGatingCompleteMaskMediaLane6		
	4	PatternCheckGatingCompleteMaskMediaLane5		
	3	PatternCheckGatingCompleteMaskMediaLane4		
	2	PatternCheckGatingCompleteMaskMediaLane3		
	1	PatternCheckGatingCompleteMaskMediaLane2		
	0	PatternCheckGatingCompleteMaskMediaLane1		
210	7	PatternGeneratorLOLMaskHostLane8	**PatternGeneratorLOLMaskHostLane<i>** Per-host lane pattern generator loss of lock Mask Default: 1	RW Adv.
	6	PatternGeneratorLOLMaskHostLane7		
	5	PatternGeneratorLOLMaskHostLane6		
	4	PatternGeneratorLOLMaskHostLane5		
	3	PatternGeneratorLOLMaskHostLane4		
	2	PatternGeneratorLOLMaskHostLane3		
	1	PatternGeneratorLOLMaskHostLane2		
	0	PatternGeneratorLOLMaskHostLane1		
211	7	PatternGeneratorLOLMaskMediaLane8	**PatternGeneratorLOLMaskMediaLane<i>** Per-media lane pattern generator loss of lock Mask Default: 1	RW Adv.
	6	PatternGeneratorLOLMaskMediaLane7		
	5	PatternGeneratorLOLMaskMediaLane6		
	4	PatternGeneratorLOLMaskMediaLane5		
	3	PatternGeneratorLOLMaskMediaLane4		
	2	PatternGeneratorLOLMaskMediaLane3		
	1	PatternGeneratorLOLMaskMediaLane2		
	0	PatternGeneratorLOLMaskMediaLane1		
212	7	PatternCheckerLOLMaskHostLane8	**PatternCheckerLOLMaskHostLane<i>** Per-host lane pattern checker loss of lock Mask Default: 1	RW Adv.
	6	PatternCheckerLOLMaskHostLane7		
	5	PatternCheckerLOLMaskHostLane6		
	4	PatternCheckerLOLMaskHostLane5		
	3	PatternCheckerLOLMaskHostLane4		
	2	PatternCheckerLOLMaskHostLane3		
	1	PatternCheckerLOLMaskHostLane2		
	0	PatternCheckerLOLMaskHostLane1		
213	7	PatternCheckerLOLMaskMediaLane8	**PatternCheckerLOLMaskMediaLane<i>** Per-media lane pattern checker loss of lock Mask Default: 1	RW Adv.
	6	PatternCheckerLOLMaskMediaLane7		
	5	PatternCheckerLOLMaskMediaLane6		
	4	PatternCheckerLOLMaskMediaLane5		
	3	PatternCheckerLOLMaskMediaLane4		
	2	PatternCheckerLOLMaskMediaLane3		

Byte	Bits	Field Name	Field/Register Description	Type
	1	PatternCheckerLOLMaskMediaLane2		
	0	PatternCheckerLOLMaskMediaLane1		
214-223	7-0	-	**Reserved[10]**	RW

8.11.14 User Pattern

Table 8-111 provides space for the host to define a user pattern of up to 32 bytes in length. The module may not support the full 32-byte length. Refer to Table 8-96 for the maximum supported user pattern length.

Table 8-111 User Pattern (Page 13h)

Byte	Bits	Register Name	Register Description	Type
224-255	7-0	UserPattern	Host defined user pattern (32 Bytes)	RW Opt.

8.12 Banked Page 14h (Module Performance Diagnostics Results)

Pages 13h and 14h are optional Pages and contain module diagnostic control and status fields.

The module advertises support of Pages 13h and 14h jointly in Bit 01h:142.5 (see Table 8-40).

Page 14h may optionally be Banked. Each Bank of Page 14h refers to 8 lanes.

Page 14h is subdivided into subject areas as illustrated in the following table:

Table 8-112 Page 14h Overview

Byte	Size (bytes)	Subject Area	Description
128	1	DiagnosticsSelector	Selects the content of Diagnostics Data (Bytes 192-255)
129	1	-	**Reserved[1]**
130-131	2	-	**Custom[2]**
132-139	18	Diagnostics Flags	Latched diagnostics Flags
140-149	10	-	**Reserved[10]**
192-255	64	Diagnostics Data	Diagnostics Data selected by Diagnostics Selector (a host visible set of Error Information Registers)

8.12.1 Diagnostics Selection

The **DiagnosticsSelector** value (Byte 14h:128) selects the type of diagnostics information that the module makes available to the host in the **Diagnostics Data** area (Bytes 14h:192-255).

Table 8-113 Diagnostics Selection Register (Page 14h)

Byte	Bits	Register Name	Register Description	Type
128	7-0	DiagnosticsSelector	Select content of Diagnostics Data, see Table 8-114. Reverts to 0 if value not supported	RWW Rqd.
129	7-0	-	**Reserved[1]**	
130	7-0	-	**Custom[1]**	
131	7-0	-	**Custom[1]**	

The possible types of diagnostics information (detection error performance information and channel metrics) and their associated DiagnosticsSelector values are listed in Table 8-114.

The DiagnosticsSelector values 01-05h allow the host to read **intermediate results** while gating is in progress.

The DiagnosticsSelector values 11-15h allow the host to read the **results** of the last completed gating period.

The module updates the values in **Diagnostics Data** area (Bytes 14h:192-255) with the selected diagnostic data within the Diagnostics Data Content Switch time (**tDDCS**) after the **DiagnosticsSelector** has been written.

Note that this is a pure timing specification; the module does not reject further register accesses. There is no confirmation by the module when the new data are available, so host should wait for a tDDCS guard period.

Table 8-114 Diagnostics Selector Options

Diagnostics Selector	Selection	Diagnostics Data Contents (Bytes 14h:192-255)
00h	None	All zeroes
Real-Time Results		
01h	Host/Media Input Lane 1-8 BER	F16 BER values (Pre- or Post-FEC)
02h	Host Lane 1-4 errors and bits counters	U64 **little endian** counters with PSL indicator
03h	Host Lane 5-8 errors and bits counters	U64 **little endian** counters with PSL indicator
04h	Media Lane 1-4 errors and bits counters	U64 **little endian** counters with PSL indicator
05h	Media Lane 5-8 errors and bits counters	U64 **little endian** counters with PSL indicator
06h	Host/Media Input Lane 1-8 SNR	U16 **little endian** in units of 1/256 dB
07h-10h	-	**Reserved**
Results over most recently completed Gating Period (stable for Gating Period)		
11h	Gated Host/Media Input Lane 1-8 BER	F16 BER values

Diagnostics Selector	Selection	Diagnostics Data Contents (Bytes 14h:192-255)
12h	Gated Host Lane 1-4 errors and bits counters	U64 **little endian** counters with PSL indicator
13h	Gated Host Lane 5-8 errors and bits counters	U64 **little endian** counters with PSL indicator
14h	Gated Media Lane 1-4 errors and bits counters	U64 **little endian** counters with PSL indicator
15h	Gated Media Lane 5-8 errors and bits counters	U64 **little endian** counters with PSL indicator
16h-BFh	-	**Reserved**
C0h-FFh	-	**Custom**

8.12.2 Diagnostics Flags

The diagnostics Pages contain Flags that are specific to diagnostics features.

The Masks associated with these diagnostics Flags are located on Page 13h (see Table 8-110).

Table 8-115 Latched Diagnostics Flags (Page 14h)

Byte	Bits	Field Name	Field/ Register Description	Type
132	7	LossOfReferenceClockFlag	Latched loss of reference clock Flag.	RO/COR Opt.
	6-0	-	**Reserved**	
133	7-0	-	**Reserved[1]**	RO/COR Opt.
134	7	PatternCheckGatingCompleteFlagHostLane8	**PatternCheckGatingCompleteFlagHostLane<i>** Latched per-host lane gating complete Flag. When gating is complete, this bit will be set.	RO/COR Adv.
	6	PatternCheckGatingCompleteFlagHostLane7		
	5	PatternCheckGatingCompleteFlagHostLane6		
	4	PatternCheckGatingCompleteFlagHostLane5		
	3	PatternCheckGatingCompleteFlagHostLane4		
	2	PatternCheckGatingCompleteFlagHostLane3		
	1	PatternCheckGatingCompleteFlagHostLane2		
	0	PatternCheckGatingCompleteFlagHostLane1		
135	7	PatternCheckGatingCompleteFlagMediaLane8	**PatternCheckGatingCompleteFlagMediaLane<i>** Latched per-media lane gating complete Flag. When gating is complete, this bit will be set.	RO/COR Adv.
	6	PatternCheckGatingCompleteFlagMediaLane7		
	5	PatternCheckGatingCompleteFlagMediaLane6		
	4	PatternCheckGatingCompleteFlagMediaLane5		
	3	PatternCheckGatingCompleteFlagMediaLane4		
	2	PatternCheckGatingCompleteFlagMediaLane3		
	1	PatternCheckGatingCompleteFlagMediaLane2		
	0	PatternCheckGatingCompleteFlagMediaLane1		
136	7	PatternGeneratorLOLFlagHostLane8	**PatternGeneratorLOLFlagHostLane<i>** Latched per-host lane pattern generator failure Flag. Indicates that the generator has failed to generate a valid pattern.	RO/COR Adv.
	6	PatternGeneratorLOLFlagHostLane7		
	5	PatternGeneratorLOLFlagHostLane6		
	4	PatternGeneratorLOLFlagHostLane5		
	3	PatternGeneratorLOLFlagHostLane4		
	2	PatternGeneratorLOLFlagHostLane3		
	1	PatternGeneratorLOLFlagHostLane2		
	0	PatternGeneratorLOLFlagHostLane1		
137	7	PatternGeneratorLOLFlagMediaLane8	**PatternGeneratorLOLFlagMediaLane<i>** Latched per-media lane pattern generator failure Flag. Indicates that the generator has failed to generate a valid pattern.	RO/COR Adv.
	6	PatternGeneratorLOLFlagMediaLane7		
	5	PatternGeneratorLOLFlagMediaLane6		
	4	PatternGeneratorLOLFlagMediaLane5		
	3	PatternGeneratorLOLFlagMediaLane4		
	2	PatternGeneratorLOLFlagMediaLane3		
	1	PatternGeneratorLOLFlagMediaLane2		
	0	PatternGeneratorLOLFlagMediaLane1		
138	7	PatternCheckerLOLFlagHostLane8	**PatternCheckerLOLFlagHostLane<i>** Latched per-host lane pattern checker loss of pattern lock Flag.	RO/COR Adv.
	6	PatternCheckerLOLFlagHostLane7		
	5	PatternCheckerLOLFlagHostLane6		
	4	PatternCheckerLOLFlagHostLane5		
	3	PatternCheckerLOLFlagHostLane4		

Byte	Bits	Field Name	Field/ Register Description	Type
	2	PatternCheckerLOLFlagHostLane3		
	1	PatternCheckerLOLFlagHostLane2		
	0	PatternCheckerLOLFlagHostLane1		
139	7	PatternCheckerLOLFlagMediaLane8	**PatternCheckerLOLFlagMediaLane <i>** Latched per-media lane pattern checker loss of pattern lock Flag.	RO/COR Adv.
	6	PatternCheckerLOLFlagMediaLane7		
	5	PatternCheckerLOLFlagMediaLane6		
	4	PatternCheckerLOLFlagMediaLane5		
	3	PatternCheckerLOLFlagMediaLane4		
	2	PatternCheckerLOLFlagMediaLane3		
	1	PatternCheckerLOLFlagMediaLane2		
	0	PatternCheckerLOLFlagMediaLane1		
140-149	7-0	-	**Reserved[10]**	

8.12.3 Diagnostics Data

The **Diagnostics Selector** field value (Byte 14h:128) determines which diagnostics information is reported in the **Diagnostics Data** area (Bytes 192-255). The meaning of the selector field value is specified in Table 8-116.

Note: The following required functionality for a Pattern Synchronization Loss indication in a counter should be noted especially by module vendors.

Embedded Pattern Checker Sync Loss Indication in Total Bits Counters

The least significant bit of any *TotalBitsCount* register indicates if the pattern checker was always in pattern sync lock during accumulation since total bits counting had started. This bit serves as a latched pattern sync loss Flag.

An **even** total bits counter value encodes that there was not a single pattern checker sync loss while counting, so BER can be calculated reliably from counted errors and counted total bits received.

An **odd** total bits counter value encodes that there was at least one pattern checker sync loss while counting, hence the counted errors are suspicious.

Note: Regular pattern checker loss of synchronization lock (LOL) Flags are also reported in Byte 14h:136. However, embedding a latched pattern lock indication in total bits counter allows the host to read a single 2x64 byte block to measure BER from bit error counts and an associated total bits counter. Note that the error rate distortion due to this encoding is negligible.

Table 8-116 Diagnostics Data (Bytes 192-255) Contents per Diagnostics Selector (Page 14h)

Diag. Selector	Bytes	Size	Diagnostics Field	Diagnostics Data Description
00h	192-255	64	-	**Reserved**
01h	192-193	2	HostSideBERLane1	F16 BER of host and media lanes 1-8 (periodically updated while gating)
	194-195	2	HostSideBERLane2	
	196-197	2	HostSideBERLane3	
	198-199	2	HostSideBERLane4	
	200-201	2	HostSideBERLane5	
	202-203	2	HostSideBERLane6	
	204-205	2	HostSideBERLane7	
	206-207	2	HostSideBERLane8	
	208-209	2	MediaSideBERLane1	
	210-211	2	MediaSideBERLane2	
	212-213	2	MediaSideBERLane3	
	214-215	2	MediaSideBERLane4	
	216-217	2	MediaSideBERLane5	
	218-219	2	MediaSideBERLane6	
	220-221	2	MediaSideBERLane7	
	222-211	2	MediaSideBERLane8	
02h	192-199	8	HostSideErrorCountLane1	U64 Little-endian (LSB first) Counters for host lanes 1-4
	200-207	8	HostTotalBitsCountLane1	

	208-215	8	HostSideErrorCountLane2	(periodically updated while gating) *Note: Odd TotalBits counter values indicate transient or permanent pattern sync loss during the accumulation period.*
	216-223	8	HostTotalBitsCountLane2	
	224-231	8	HostSideErrorCountLane3	
	232-239	8	HostTotalBitsCountLane3	
	240-247	8	HostSideErrorCountLane4	
	248-255	8	HostTotalBitsCountLane4	
03h	192-199	8	HostSideErrorCountLane5	U64 Little-endian (LSB first) Counters for host lanes 5-8 (periodically updated while gating) *Note: Odd TotalBits counter values indicate transient or permanent pattern sync loss during the accumulation period.*
	200-207	8	HostTotalBitsCountLane5	
	208-215	8	HostSideErrorCountLane6	
	216-223	8	HostTotalBitsCountLane6	
	224-231	8	HostSideErrorCountLane7	
	232-239	8	HostTotalBitsCountLane7	
	240-247	8	HostSideErrorCountLane8	
	248-255	8	HostTotalBitsCountLane8	
04h	192-199	8	MediaSideErrorCountLane1	U64 **Little-endian** (LSB first) Counters for media lanes 1-4 (periodically updated while gating) *Note: Odd TotalBits counter values indicate transient or permanent pattern sync loss during the accumulation period.*
	200-207	8	MediaSideTotalBitsCountLane1	
	208-215	8	MediaSideErrorCountLane2	
	216-223	8	MediaSideTotalBitsCountLane2	
	224-231	8	MediaSideErrorCountLane3	
	232-239	8	MediaSideTotalBitsCountLane3	
	240-247	8	MediaSideErrorCountLane4	
	248-255	8	MediaSideTotalBitsCountLane4	
05h	192-199	8	MediaSideErrorCountLane5	U64 **Little-endian** (LSB first) Counters for media lanes 5-8 (periodically updated while gating) *Note: Odd TotalBits counter values indicate transient or permanent pattern sync loss during the accumulation period.*
	200-207	8	MediaSideTotalBitsCountLane5	
	208-215	8	MediaSideErrorCountLane6	
	216-223	8	MediaSideTotalBitsCountLane6	
	224-231	8	MediaSideErrorCountLane7	
	232-239	8	MediaSideTotalBitsCountLane7	
	240-247	8	MediaSideErrorCountLane8	
	248-255	8	MediaSideTotalBitsCountLane8	
06h	192-193	2	-	**Reserved**
	194-195	2	-	
	196-197	2	-	
	198-199	2	-	
	200-201	2	-	
	202-203	2	-	
	204-205	2	-	
	206-207	2	-	
	208-209	2	HostSideSNRLane1	U16 **Little-endian** in units of 1/256dB SNR for host lanes 1-8 (real time status)
	210-211	2	HostSideSNRLane2	
	212-213	2	HostSideSNRLane3	
	214-215	2	HostSideSNRLane4	
	216-217	2	HostSideSNRLane5	
	218-219	2	HostSideSNRLane6	
	220-221	2	HostSideSNRLane7	
	222-223	2	HostSideSNRLane8	
	224-225	2	-	**Reserved**
	226-227	2	-	
	228-229	2	-	
	230-231	2	-	
	232-233	2	-	
	234-235	2	-	
	236-237	2	-	
	238-239	2	-	
	240-241	2	MediaSideSNRLane1	U16 **Little-endian** in units of 1/256dB SNR for media lanes 1-8 (real time status)
	242-243	2	MediaSideSNRLane2	
	244-245	2	MediaSideSNRLane3	

	246-247	2	MediaSideSNRLane4	
	248-249	2	MediaSideSNRLane5	
	250-251	2	MediaSideSNRLane6	
	252-253	2	MediaSideSNRLane7	
	254-255	2	MediaSideSNRLane8	
11h	192-193	2	GatedHostSideBERLane1	F16 BER BER of host and media lanes 1-8 (stable gating results)
	194-195	2	GatedHostSideBERLane2	
	196-197	2	GatedHostSideBERLane3	
	198-199	2	GatedHostSideBERLane4	
	200-201	2	GatedHostSideBERLane5	
	202-203	2	GatedHostSideBERLane6	
	204-205	2	GatedHostSideBERLane7	
	206-207	2	GatedHostSideBERLane8	
	208-209	2	GatedMediaSideBERLane1	
	210-211	2	GatedMediaSideBERLane2	
	212-213	2	GatedMediaSideBERLane3	
	214-215	2	GatedMediaSideBERLane4	
	216-217	2	GatedMediaSideBERLane5	
	218-219	2	GatedMediaSideBERLane6	
	220-221	2	GatedMediaSideBERLane7	
	222-211	2	GatedMediaSideBERLane8	
12h	192-199	8	GatedHostSideErrorCountLane1	U64 **Little-endian** (LSB first) Counters for host lanes 1-4 (stable gating results)
	200-207	8	GatedHostSideTotalBitsCountLane1	
	208-215	8	GatedHostSideErrorCountLane2	
	216-223	8	GatedHostSideTotalBitsCountLane2	
	224-231	8	GatedHostSideErrorCountLane3	
	232-239	8	GatedHostSideTotalBitsCountLane3	
	240-247	8	GatedHostSideErrorCountLane4	
	248-255	8	GatedHostSideTotalBitsCountLane4	
13h	192-199	8	GatedHostSideErrorCountLane5	U64 **Little-endian** (LSB first) Counters for host lanes 5-8 (stable gating results)
	200-207	8	GatedHostSideTotalBitsCountLane5	
	208-215	8	GatedHostSideErrorCountLane6	
	216-223	8	GatedHostSideTotalBitsCountLane6	
	224-231	8	GatedHostSideErrorCountLane7	
	232-239	8	GatedHostSideTotalBitsCountLane7	
	240-247	8	GatedHostSideErrorCountLane8	
	248-255	8	GatedHostSideTotalBitsCountLane8	
14h	192-199	8	GatedMediaSideErrorCountLane1	U64 **Little-endian** (LSB first) Counters for media lanes 1-4 (stable gating results)
	200-207	8	GatedMediaSideTotalBitsCountLane1	
	208-215	8	GatedMediaSideErrorCountLane2	
	216-223	8	GatedMediaSideTotalBitsCountLane2	
	224-231	8	GatedMediaSideErrorCountLane3	
	232-239	8	GatedMediaSideTotalBitsCountLane3	
	240-247	8	GatedMediaSideErrorCountLane4	
	248-255	8	GatedMediaSideTotalBitsCountLane4	
15h	192-199	8	GatedMediaSideErrorCountLane5	U64 Little**-endian** (LSB first) Counters for media lanes 5-8 (stable gating results)
	200-207	8	GatedMediaSideTotalBitsCountLane5	
	208-215	8	GatedMediaSideErrorCountLane6	
	216-223	8	GatedMediaSideTotalBitsCountLane6	
	224-231	8	GatedMediaSideErrorCountLane7	
	232-239	8	GatedMediaSideTotalBitsCountLane7	
	240-247	8	GatedMediaSideErrorCountLane8	
	248-255	8	GatedMediaSideTotalBitsCountLane8	

8.13 Banked Page 15h (Timing Characteristics)

Page 15h is an optional Page containing per-lane timing characteristics.

The module advertises support of Page 15h in Bit 01h:145.3 (see Table 8-43).

Page 15h may optionally be Banked. Each Bank of Page 15h refers to 8 lanes.

All fields in Page 15h are read-only reporting registers (not necessarily static).

Page 15h is subdivided in subject areas as illustrated in the following table:

Table 8-117 Page 15h Overview

Byte	Size (Bytes)	Subject Area	Description
128 – 223	96	-	**Reserved**
224 – 239	8 x 2	Data Path Rx Latency	Total Rx delay thru the module reported by host lane
240 – 255	8 x 2	Data Path Tx Latency	Total Tx delay thru the module reported by host lane

Data Path Rx and Tx Latency

The following fields report the Data Path Rx and Tx Latency associated with each host lane in the module.

Data Path Rx Latency and Data Path Tx Latency convey the total delay thru the module, in nanoseconds, and are reported by host lane. For Data Paths with multiple lanes, all lanes shall report the same latency.

The accuracy of the reported latency values is not specified in this version of CMIS.

Note: These registers may be used to either convey static average or max latencies, per-module values that have been characterized by the manufacturer, or to convey dynamic/per-reset values that are changed by the module after each initialization. There may be other possibilities.

Note: For modules which perform dynamic updates of these registers, it is currently undefined as to when the values in these registers are guaranteed to be valid.

Table 8-118 Data Path Rx and Tx Latency, per lane (Page 15h)

Byte	Bit	Register Name	Register Array Description	Type
224-225	7-0	DataPathRxLatencyHostLane1	**DataPathRxLatencyHostLane<i>** U16 lane <i> Rx delay in ns	RO Rqd.
226-227	7-0	DataPathRxLatencyHostLane2		RO Rqd.
228-229	7-0	DataPathRxLatencyHostLane3		RO Rqd.
230-231	7-0	DataPathRxLatencyHostLane4		RO Rqd.
232-233	7-0	DataPathRxLatencyHostLane5		RO Rqd.
234-235	7-0	DataPathRxLatencyHostLane6		RO Rqd.
236-237	7-0	DataPathRxLatencyHostLane7		RO Rqd.
238-239	7-0	DataPathRxLatencyHostLane8		RO Rqd.
240-241	7-0	DataPathTxLatencyHostLane1	**DataPathTxLatencyHostLane<i>** U16 lane <i> Tx delay in ns	RO Rqd.
242-243	7-0	DataPathTxLatencyHostLane2		RO Rqd.
244-245	7-0	DataPathTxLatencyHostLane3		RO Rqd.
246-247	7-0	DataPathTxLatencyHostLane4		RO Rqd.
248-249	7-0	DataPathTxLatencyHostLane5		RO Rqd.
250-251	7-0	DataPathTxLatencyHostLane6		RO Rqd.
252-253	7-0	DataPathTxLatencyHostLane7		RO Rqd.
254-255	7-0	DataPathTxLatencyHostLane8		RO Rqd.

8.14 Banked Pages Range 20h-2Fh (VDM)

Versatile diagnostics monitoring (VDM) is an optional feature allowing extension of the types of observables that can be monitored beyond those defined in Lower Memory and in Page 11h.

Basic monitoring functionality includes both access to a recent sample and threshold crossing detection.

Advanced monitoring functionality provides interval statistics (min, max, average) for certain observables.

The module advertises support of the VDM feature in bit 01h:142.6 (see Table 8-40)

A module may offer up to 256 so called VDM instances ("slots") for up to 256 observables.

The data of VDM instances are available in the VDM Page range 20h-2Fh in groups of 64 VDM instances.

Additional Banks of Pages can provide VDM instances for modules with more than 8 lanes.

See also section 7.1 for more information and an introduction to VDM functionality.

Introduction and Overview

A **VDM instance** is a numbered "slot" in the VDM Pages which is either active and associated with a basic observable or with a statistic of a basic observable (min, max, average), or passive and unused.

The observable associated with a VDM instance can be related to a lane, to a Data Path, or to the entire module.

The association of VDM instances and basic observables or statistics is module-defined: a module describes each supported VDM instance in an array of **VDM instance descriptors** which is distributed over up to four Pages (in groups of 64 descriptors per Page).

The descriptor of a VDM instance either identifies the associated observable (basic or statistic) from a list of supported observable types, or marks the VDM instance as unused (passive). For active VDM instances, the descriptor points to an associated set of four module-defined thresholds that the module uses to detect and flag threshold crossings. The host can set a Mask for each threshold crossing Flag to suppress contribution of that Flag to Interrupt request generation.

Note: Different modules will often support monitoring the same observables, but the VDM instance used for each observable is not pre-defined. The host will have to search and map desired monitors from the VDM instance descriptor array.

In case of a **basic observable**, VDM periodically reports "current value" samples of that observable via the sample register of its VDM instance (in Pages 24h-27h) and moreover it provides high/low alarm/warning threshold crossing supervision for that observable by setting the appropriate Flags on threshold crossing.

In case of a **statistics observable** (min, max, average) the host can read current values of the statistics "live" as updated by the module, but it can also request the module to stop (freeze) the current value update in order to ensure consistent reading of statistics results across one or more VDM instances (see section 8.14.6). The module performs threshold crossing detection also for statistics observables.

An important side effect of (globally) freezing the statistics reporting registers is that the module internally starts a new statistics interval by resetting its internal statistics variables at precisely the time when it performs the last update on a then frozen register.

This freeze-and-reset mechanism enables gap-free **interval statistics** over host-defined intervals.

Overview of VDM Pages

The following table provides an overview of the VDM Pages and their purpose:

Table 8-119 Summary of Page Definitions for Page 20h-2Fh

Page	Subject Area	Description
20h	Descriptors for VDM Instances 1-64 (Group 1)	Every 2-byte value describes a VDM instance offered by the module. (RO access)
21h	Descriptors for VDM Instances 65-128 (Group 2)	
22h	Descriptors for VDM Instances 129-192 (Group 3)	
23h	Descriptors for VDM Instances 193-256 (Group 4)	
24h	Samples of VDM Instances 1-64 (Group 1)	Every 2-byte value is a (possibly frozen) sample of the observable monitored by a VDM instance. (RO access)
25h	Samples of VDM Instances 65-128 (Group 2)	
26h	Samples of VDM Instances 129-192 (Group 3)	
27h	Samples of VDM Instances 193-256 (Group 4)	

Page	Subject Area	Description
28h	Thresholds 1-16 (Group 1)	Every set of four 2-byte values describes a threshold set of a threshold crossing detector, possibly shared by several VDM instances. The association of these threshold sets and VDM instances is described in the VDM descriptors. (RO access)
29h	Thresholds 17-32 (Group 2)	
2Ah	Thresholds 33-48 (Group 3)	
2Bh	Thresholds 49-64 (Group 4)	
2Ch	VDM Flags (Groups 1-4)	Every Byte contains the latched threshold crossing Flags of 2 VDM instances (4 bits each) (RO/COR access)
2Dh	VDM Masks (Groups 1-4)	Every Byte contains the Masks for the threshold crossing Flags of 2 VDM instances (4 bits each) (RW access) The default of each Mask is 1 (masked)
2Eh	-	**Reserved**
2Fh	Advertisement and Dynamic Controls	VDM support details and dynamic controls (Mixed RO and RW access)

8.14.1 Pages 20h-23h (VDM Descriptor Groups Pages)

There can be up to 256 VDM instances to monitor up to 256 VDM observables.

The module describes the list of VDM instances it supports in VDM instance descriptor registers.

Each Page contains one of four VDM instance groups.

Table 8-120 Page 20h,21h,22h,23h VDM Configuration

Page	Byte	Register Name	Description	Type
20h	128-129	VDMDescriptor1	Descriptor register for VDM instance 1, in group 1	RO
	130-131	VDMDescriptor2	Descriptor register for VDM instance 2, in group 1	RO
	132-251	…	Descriptor registers for VDM instance **3 to 62**, in group 1	RO
	252-253	VDMDescriptor63	Descriptor register for VDM instance 63, in group 1	RO
	254-255	VDMDescriptor64	Descriptor register for VDM instance 64, in group 1	RO
21h	128-129	VDMDescriptor65	Descriptor register for VDM instance 65, in group 2	RO
	130-131	VDMDescriptor66	Descriptor register for VDM instance 66, in group 2	RO
	132-251	…	Descriptor registers for VDM instance **67 to 126**, in group 2	RO
	252-253	VDMDescriptor127	Descriptor register for VDM instance 127, in group 2	RO
	254-255	VDMDescriptor128	Descriptor register for VDM instance 128, in group 2	RO
22h	128-129	VDMDescriptor129	Descriptor register for VDM instance 129, in group 3	RO
	130-131	VDMDescriptor130	Descriptor register for VDM instance 130, in group 3	RO
	132-251	…	Descriptor registers for VDM instance **131 to 190**, in group 3	RO
	252-253	VDMDescriptor191	Descriptor register for VDM instance 191, in group 3	RO
	254-255	VDMDescriptor192	Descriptor register for VDM instance 192, in group 3	RO
23h	128-129	VDMDescriptor193	Descriptor register for VDM instance 193, in group 4	RO
	130-131	VDMDescriptor194	Descriptor register for VDM instance 194, in group 4	RO
	132-251	…	Descriptor registers for VDM instance **195 to 254**, in group 4	RO
	252-253	VDMDescriptor255	Descriptor register for VDM instance 255, in group 4	RO
	254-255	VDMDescriptor256	Descriptor register for VDM instance 256, in group 4	RO

8.14.1.1 VDM Instance Descriptors

The module describes each available VDM instance by a two-byte descriptor field.

Table 8-121 Definition of 2-byte VDM Instance Descriptor

Byte	Bits	Field Name and Description
Even Address	7-4	**LocalThresholdSetID** This number determines which threshold set will be used for this observable. The threshold set is in the same group as the observable descriptor. The global ThresholdSetID (1-64) is defined as follows: Page 20h: (group 1): ThresholdSetID = 1 + LocalThresholdSetID Page 21h: (group 2): ThresholdSetID = 17 + LocalThresholdSetID Page 22h: (group 3): ThresholdSetID = 33 + LocalThresholdSetID Page 23h: (group 4): ThresholdSetID = 49 + LocalThresholdSetID
	3-0	**Monitored Resource** 0: Lane 1 or Data Path starting on lane 1 1: Lane 2 or Data Path starting on lane 2 2: Lane 3 or Data Path starting on lane 3 3: Lane 4 or Data Path starting on lane 4 4: Lane 5 or Data Path starting on lane 5 5: Lane 6 or Data Path starting on lane 6 6: Lane 7 or Data Path starting on lane 7 7: Lane 8 or Data Path starting on lane 8 15: Module (not associated with a lane or Data Path)
Odd Address	7-0	Observable Type (see Table 8-122 for observables and encodings)

8.14.1.2 VDM Observable Types

The list of defined VDM observable types, an associated Type ID, and the data type of samples of the observable type is defined in Table 8-122.

Note: Unlike the regular Module Diagnostics observables (on Page 13h-14h) all VDM data types are Big Endian.

Table 8-122 VDM Observable Types (Type Coding)

Type ID	Observable Type	Instance Type	Data Type	Unit Scale	Unit
0	Not Used indicator[1]	N/A	N/A		
1	Laser Age (0% at BOL, 100% EOL) (Data Path)	Basic	U16	1	%
2	TEC Current (Module)	Basic	S16	100/32767	%
3	Laser Frequency Error (Media Lane)	Basic	S16	10	MHz
4	Laser Temperature (Media Lane)	Basic	S16	1/256	C
5	eSNR Media Input (Media Lane)	Basic	U16	1/256	dB
6	eSNR Host Input (Lane)	Basic	U16	1/256	dB
7	PAM4 Level Transition Parameter Media Input (Media Lane)	Basic	U16	1/256	dB
8	PAM4 Level Transition Parameter Host Input (Lane)	Basic	U16	1/256	dB
9	Pre-FEC BER Minimum Media Input (Data Path)	Statistic	F16	N/A	
10	Pre-FEC BER Minimum Host Input (Data Path)	Statistic	F16	N/A	
11	Pre-FEC BER Maximum Media Input (Data Path)	Statistic	F16	N/A	
12	Pre-FEC BER Maximum Host Input (Data Path)	Statistic	F16	N/A	
13	Pre-FEC BER Average Media Input (Data Path)	Statistic	F16	N/A	
14	Pre-FEC BER Average Host Input (Data Path)	Statistic	F16	N/A	
15	Pre-FEC BER Current Value Media Input (Data Path)	Basic	F16	N/A	
16	Pre-FEC BER Current Value Host Input (Data Path)	Basic	F16	N/A	
17	FERC Minimum Media Input (Data Path)	Statistic	F16	N/A	
18	FERC Minimum Host Input (Data Path)	Statistic	F16	N/A	
19	FERC Maximum Media Input (Data Path)	Statistic	F16	N/A	
20	FERC Maximum Host Input (Data Path)	Statistic	F16	N/A	
21	FERC Average Media Input (Data Path)	Statistic	F16	N/A	
22	FERC Average Host Input (Data Path)	Statistic	F16	N/A	
23	FERC Current Value Media Input (Data Path)	Basic	F16	N/A	
24	FERC Current Value Host Input (Data Path)	Basic	F16	N/A	
25-99	Reserved				
100-127	**Custom** Observables				
128-255	**Restricted** OIF				

[1] Not Used means that the module does not present data for the VDM Instance described by the descriptor.

8.14.2 Pages 24h-27h (VDM Sample Groups Pages)

There are up to 256 read-only sample registers (real time value registers) for current values of basic observables or statistics associated with up to 256 VDM instances.

All sample values are store in **Big Endian** order (MSB in lower address).

The observable actually reported in the VDM<i>Sample register of VDM instance <i> is defined in the descriptor VDM<i>Descriptor of that VDM instance in Pages 20h-23h.

Table 8-123 Page 24h-27h VDM Real-Time Values

Page	Byte	Register Name	Register Description	Type
24h	128-129	VDMSample1	X16 Real-time value of VDM instance 1 in group 1	RO
	130-131	VDMSample2	X16 Real-time value for VDM instance 2 in group 1	RO
	132-251	VDM **3…62**	X16 Real-time values for VDM instance 3 to 62 in group 1	RO
	252-253	VDMSample63	X16 Real-time value for VDM instance 63 in group 1	RO
	254-255	VDMSample64	X16 Real-time value for VDM instance 64 in group 1	RO
25h	128-129	VDMSample65	X16 Real-time value of VDM instance 65 in group 2	RO
	130-131	VDMSample66	X16 Real-time value for VDM instance 66 in group 2	RO
	132-251	VDM **67..126**	X16 Real-time values for VDM instance 67 to 126 in group 2	RO
	252-253	VDMSample127	X16 Real-time value for VDM instance 127 in group 2	RO
	254-255	VDMSample128	X16 Real-time value for VDM instance 128 in group 2	RO
26h	128-129	VDMSample129	X16 Real-time value of VDM instance 129 in group 3	RO
	130-131	VDMSample130	X16 Real-time value for VDM instance 130 in group 3	RO
	132-251	VDM **131..190**	X16 Real-time values for VDM instance 131 to 190 in group 3	RO
	252-253	VDMSample191	X16 Real-time value for VDM instance 191 in group 3	RO
	254-255	VDMSample192	X16 Real-time value for VDM instance 192 in group 3	RO
27h	128-129	VDMSample193	X16 Real-time value of VDM instance 193 in group 4	RO
	130-131	VDMSample194	X16 Real-time value for VDM instance 194 in group 4	RO
	132-251	VDM **195..254**	X16 Real-time values for VDM instance 195 to 254 in group 4	RO
	252-253	VDMSample255	X16 Real-time value for VDM instance 255 in group 4	RO
	254-255	VDMSample256	X16 Real-time value for VDM instance 256 in group 4	RO

8.14.3 Pages 28h-2Bh (VDM Threshold Set Groups Pages)

Pages 28h-2Bh contain a read only array of threshold sets, stored in groups of 16 threshold sets per Page.

Each used threshold set (quad) contains four threshold values for threshold crossing supervision at high/low alarm/warning thresholds.

The usage of a threshold sets for one or more observables is defined in the VDM instance descriptors of those observables on Pages 20h-23h.

Table 8-124 Page 28h-2Bh Alarm/Warning Thresholds

Page	Byte	Register Name	Register Description	Type
28h	128-129	HighAlarmThreshold1	X16 Thresholds for group 1 monitored observables. Threshold type and storage same as supervised real-time value.	RO
	130-131	LowAlarmThreshold1		RO
	132-133	HighWarningThreshold1		RO
	134-135	LowWarningThreshold1		RO
	136-137	HighAlarmThreshold2		RO
	138-139	LowAlarmThreshold2		RO
	140-141	HighWarningThreshold2		RO
	142-143	LowWarningThreshold2		RO
	144-247	… **3 to 15**		RO
	248-249	HighAlarmThreshold16		RO
	250-251	LowAlarmThreshold16		RO
	252-253	HighWarningThreshold16		RO
	254-255	LowWarningThreshold16		RO
29h	128-129	HighAlarmThreshold17	X16 Thresholds for group 2 monitored observables. Threshold type and storage same as supervised real-time value.	RO
	130-131	LowAlarmThreshold17		RO
	132-133	HighWarningThreshold17		RO
	134-135	LowWarningThreshold17		RO
	136-137	HighAlarmThreshold18		RO
	138-139	LowAlarmThreshold18		RO
	140-141	HighWarningThreshold18		RO
	142-143	LowWarningThreshold18		RO
	144-247	… **19 to 31**		RO
	248-249	HighAlarmThreshold32		RO
	250-251	LowAlarmThreshold32		RO
	252-253	HighWarningThreshold32		RO
	254-255	LowWarningThreshold32		RO
2Ah	128-129	HighAlarmThreshold33	X16 Thresholds for group 3 monitored observables. Threshold type and storage same as supervised real-time value.	RO
	130-131	LowAlarmThreshold33		RO
	132-133	HighWarningThreshold33		RO
	134-135	LowWarningThreshold33		RO
	136-247	… **34 to 47**		RO
	248-249	HighAlarmThreshold48		RO
	250-251	LowAlarmThreshold48		RO
	252-253	HighWarningThreshold48		RO
	254-255	LowWarningThreshold48		RO
2Bh	128-129	HighAlarmThreshold49	X16 Thresholds for group 4 monitored observables. Threshold type and storage same as supervised real-time value.	RO
	130-131	LowAlarmThreshold49		RO
	132-133	HighWarningThreshold49		RO
	134-135	LowWarningThreshold49		RO
	144-247	… **50 to 63**		RO
	248-249	HighAlarmThreshold64		RO
	250-251	LowAlarmThreshold64		RO
	252-253	HighWarningThreshold64		RO
	254-255	LowWarningThreshold64		RO

8.14.4 Page 2Ch (VDM Flags Page)

Each VDM instance has four Flags for threshold crossing events: alarms and warnings for crossing high and low thresholds.

Page 2Ch contains the Flags for these threshold crossing alarms and warnings, for all active VDM instances.

Table 8-125 VDM Threshold Crossing (TC) Flags Byte

Byte	Bit	Field Name	Field Description	Type
Any	0 or 4	HighTCAlarmFlag	Latched when monitored observable > high alarm threshold	RO/COR
	1 or 5	LowTCAlarmFlag	Latched when monitored observable < low alarm threshold	RO/COR
	2 or 6	HighTCWarningFlag	Latched when monitored observable > high warning threshold	RO/COR
	3 or 7	LowTCWarningFlag	Latched when monitored observable < low warning threshold	RO/COR

Table 8-126 Page 2Ch Alarm and Warning Configuration

Bytes	Bits	Field Name	Field Description	Type
128	7-4	VDMFlags2	High/Low Alarm/Warning TC Flags for VDM instance 2, group 1	RO/COR
	3-0	VDMFlags1	High/Low Alarm/Warning TC Flags for VDM instance 1, group 1	RO/COR
129-158	7-0	VDMFlags3-62	High/Low Alarm/Warning TC Flags as described in Table 8-124, for VDM instances 3-62, group 1	RO/COR
159	7-4	VDMFlags64	High/Low Alarm/Warning TC Flags for VDM instance 64, group 1	RO/COR
	3-0	VDMFlags63	High/Low Alarm/Warning TC Flags for VDM instance 63, group 1	RO/COR
160	7-4	VDMFlags66	High/Low Alarm/Warning TC Flags for VDM instance 66, group 2	RO/COR
	3-0	VDMFlags65	High/Low Alarm/Warning TC Flags for VDM instance 65, group 2	RO/COR
161-190	7-0	VDMFlags**67-126**	High/Low Alarm/Warning TC Flags as described in Table 8-124, for VDM instances 67-126, group 2	RO/COR
191	7-4	VDMFlags128	High/Low Alarm/Warning TC Flags for VDM instance 128, group 2	RO/COR
	3-0	VDMFlags127	High/Low Alarm/Warning TC Flags for VDM instance 127, group 2	RO/COR
192	7-4	VDMFlags130	High/Low Alarm/Warning TC Flags for VDM instance 130, group 3	RO/COR
	3-0	VDMFlags129	High/Low Alarm/Warning TC Flags for VDM instance 129, group 3	RO/COR
193-222	7-0	VDMFlags**131-190**	High/Low Alarm/Warning TC Flags as described in Table 8-124, for VDM instances 131-190, group 3	RO/COR
223	7-4	VDMFlags192	High/Low Alarm/Warning TC Flags for VDM instance 192, group 3	RO/COR
	3-0	VDMFlags191	High/Low Alarm/Warning TC Flags for VDM instance 191, group 3	RO/COR
224	7-4	VDMFlags194	High/Low Alarm/Warning TC Flags for VDM instance 194, group 4	RO/COR
	3-0	VDMFlags193	High/Low Alarm/Warning TC Flags for VDM instance 193, group 4	RO/COR
224-254	7-0	VDMFlags**195-254**	High/Low Alarm/Warning Flags as described in Table 8-124, for VDM instances 195-254, group 4,	RO/COR
255	7-4	VDMFlags256	High/Low Alarm/Warning TC Flags for VDM instance 256, group 4	RO/COR
	3-0	VDMFlags255	High/Low Alarm/Warning TC Flags for VDM instance 255, group 4	RO/COR

8.14.5 Page 2Dh (VDM Masks Page)

Each VDM instance can set Flags as described in Page 2Ch.

Each set Flag asserts the Interrupt request signal unless the corresponding Mask bit is set.

The default value for all VDM Mask bits is 1 (masked).

Page 2Dh defines a writeable Page of alarm Mask bits for the corresponding Flags defined in Page 2Ch.

Table 8-127 Page 2Dh ThresholdSet0 to 15 Alarm and Warning Configuration

Byte	Bit	Field Name	Field Description	Type
128	7-4	VDMMasks2	High/Low Alarm/Warning Masks for VDM instance 2, group 1	RW
	3-0	VDMMasks1	High/Low Alarm/Warning Masks for VDM instance 1, group 1	RW
129-158	7-4 3-0	VDMMasks3-62	High/Low Alarm/Warning Masks as described in Table 8-124, for VDM instances 3-62, group 1	RW
159	7-4	VDMMasks64	High/Low Alarm/Warning Masks for VDM instance 64, group 1	RW
	3-0	VDMMasks63	High/Low Alarm/Warning Masks for VDM instance 63, group 1	RW
160	7-4	VDMMasks66	High/Low Alarm/Warning Masks for VDM instance 66, group 2	RW
	3-0	VDMMasks65	High/Low Alarm/Warning Masks for VDM instance 65, group 2	RW
161-190	7-4 3-0	VDMMasks67-126	High/Low Alarm/Warning Masks as described in Table 8-124, for VDM instances 67-126, group 2	RW
191	7-4	VDMMasks128	High/Low Alarm/Warning Masks for VDM instance 128, group 2	RW
	3-0	VDMMasks127	High/Low Alarm/Warning Masks for VDM instance 127, group 2	RW
192	7-4	VDMMasks130	High/Low Alarm/Warning Masks for VDM instance 130, group 3	RW
	3-0	VDMMasks129	High/Low Alarm/Warning Masks for VDM instance 129, group 3	RW
193-222	7-4 3-0	VDMMasks131-190	High/Low Alarm/Warning Masks as described in Table 8-124, for VDM instances 131-190, group 3	RW
223	7-4	VDMMasks192	High/Low Alarm/Warning Masks for VDM instance 192, group 3	RW
	3-0	VDMMasks191	High/Low Alarm/Warning Masks for VDM instance 191, group 3	RW
224	7-4	VDMMasks194	High/Low Alarm/Warning Masks for VDM instance 194, group 4	RW
	3-0	VDMMasks193	High/Low Alarm/Warning Masks for VDM instance 193, group 4	RW
224-254	7-4 3-0	VDMMasks195-254	High/Low Alarm/Warning Masks as described in Table 8-124, for VDM instances 195-254, group 4,	RW
255	7-4	VDMMasks256	High/Low Alarm/Warning Masks for VDM instance 256, group 4	RW
	3-0	VDMMasks255	High/Low Alarm/Warning Masks for VDM instance 255, group 4	RW

8.14.6 Page 2Fh (VDM Advertisement and Dynamic Controls)

Page 2Fh is used for two purposes

- **advertisement** of the number of supported VDM groups (i.e. availability of VDM Pages)
- **dynamic control** for freezing and unfreezing statistics reporting registers

VDM Groups Advertisement

The VDMSupport Byte 2Fh:128 advertises how many VDM instance groups (of 64 VDM instances each) are supported, i.e. which of the Pages 20h-2Bh, and which parts of Pages 2Ch and 2Dh are available.

The individual VDM instances that the module actually provides to report samples of basic observables or of statistics of basic observables are described in the VDM Descriptor pages (see section 8.14.1).

Freezing and Unfreezing Statistics Reporting Registers

To ease the following description, the sample registers of VDM instances (in Pages 24h-27h) that report a statistic (minimum, maximum, or average) of a basic observable are called **statistics reporting registers**.

Note: Please see the beginning of section 8.14 for an introduction to statistics support in VDM

During statistics collection, a module always updates internal statistics variables (e.g. counters) but it updates the associated statistics reporting registers only when the statistics reporting is not frozen as described below.

Globally freezing the statistics reporting implies that the module terminates the current statistics collection interval (providing the results in the now frozen statistics reporting registers) and atomically starts a new statistics collection interval internally (without losing any sample of the basic observables).

Note: Raising the FreezeRequest bit effectively defines a host-controlled Performance Monitoring Interval.

Note: The initial behavior of statistics collection is not specified; it is unknow if statistics collection is initially halted or running. To achieve a well-defined starting point, hosts are advised to initially raise a FreezeRequest and then to start statistics collection at a suitable point in time after the module confirmed that the Freeze has been executed .

Dynamic Control of Statistics Intervals and Freezing

The host raises the FreezeRequest Bit 2Fh:144.7 to request that the module shall stop updating the statistics reporting registers (in order to allow consistent results readout) and at the same to start a new statistics interval.

When finished reading, the host may clear the FreezeRequest bit to signal that the module can resume updating the Interval Statistics registers. As a preparation for requesting the next freeze, the Host must eventually clear the FreezeRequest bit.

When freezing the reporting registers, the module starts a new statistics interval and ensures that no sample of any observable is lost. To this end the module atomically resets and restarts its internal statistics variables when it performs the last update on a then frozen register and then just stops updating the statistics reporting registers until the host releases the reporting registers again by terminating the freeze period.

Scenario

A scenario of host-module interactions for (repeatedly) obtaining statistics results over contiguous statistics intervals defined by the host (i.e. over host-defined Performance Monitoring Intervals) is as follows:

1. Host ceases a **FreezeRequest** (if still set) in order to request unfreezing the statistics reporting registers
2. Module clears **UnfreezeDone** (within **tVDMF** time) to prepare for a later Unfreeze Done indication
3. Module copies current values of its internally collected statistics (internally collected during the Freeze) to the statistics reporting registers and then raises **UnfreezeDone**
4. Host may now read live statistics from the statistics reporting registers to get incrementally updated statistics results. The module now continues to update the statistics reporting registers.
5. To request termination of the current statistics interval and prepare for statistics results readout, the host raises **FreezeRequest**
6. Module clears **FreezeDone** (within **tVDMF** time) to prepare for a later FreezeDone indication
7. Module stops updating statistics reporting registers after freeze, resets its internal statistics variable, and immediately continues to update the internal statistics variables (just without copying updates to the frozen statistics reporting registers) and then sets **FreezeDone**
8. The statistics reporting registers are now frozen and the host is free to read the frozen statistics reporting registers with the results of the statistics interval just ended.
9. While the statistics results registers are frozen, the module updates its internal statistics variables, but does not update the statistics reporting registers.

10. Host may cease the still active **FreezeRequest** at any time before ending the current statistics interval, either immediately after having read the statistics of the previous interval, or just prior to ending the current interval, and the scenario repeats .

Table 8-128 VDM Advertisement and Control Registers Summary (Page 2Fh)

Byte	Bit	Field Name	Field Description	Type
128	7-2	-	**Reserved**	RO
	1-0	VDMSupport	Advertisement 0: Group 1 (Page 20h, 24h, 28h, first ¼ of 2Ch, 2Dh) 1: Groups 1-2 (Page 20h-21h, 24h-25h, 28h-29h, first ½ of 2Ch, 2Dh) 2: Groups 1-3 (Page 20h-22h, 24h-26h, 28h-2Ah, first ¾ of 2Ch, 2Dh) 3: Groups 1-4 (Page 20h-23h, 24h-27h, 28h-2Bh, 2Ch, 2Dh)	RO
129-130	7-0	FineIntervalLength	U16 Length of fine interval (measurement time for one sample) used for internal BER/FERC monitoring, in units of 0.1 ms	RO
131-143	7-0	-	**Reserved[13]**	RO
144	7	FreezeRequest	When raised by the host, causes the module to freeze and hold all reported statistics reporting registers (minimum, maximum and average values) in Pages 24h-27h. When ceased by the host, releases the freeze request, allowing the reported minimum, maximum and average values to update again. *Note. The module internally records new statistics while the reporting registers are frozen, so that no data is lost.*	RW
	6-0	-	**Reserved**	RO
145	7	FreezeDone	Raised by the module when it has finished freezing the reporting registers (such that the host can now safely read) and simultaneously restarted all supported statistics internally. Ceased by the module (within $\mathbf{T}_{\mathbf{VDMF}}$ time from when Freeze Request is raised by the host). Until a first FreezeRequest is served, the module reports 0b.	RO
	6	UnfreezeDone	Raised by the module when real-time updates of the reporting registers are performed again Ceased by the module (within $\mathbf{T}_{\mathbf{VDMF}}$ time from when Freeze Request is ceased by the host). Until a first request to unfreeze is served, the module reports 0b.	RO
	5-0	-	**Reserved**	RO
146-255	7-0	-	**Reserved[110]**	RO

8.15 Banked Page 9Fh (CDB Message)

Page 9Fh is the main Page of the optional Command Data Block (CDB) messaging feature (see section 7.2).

The module advertises support of Page 9Fh in the **CdbInstancesSupported** field (see Table 8-48).

Each supported Bank of Page 9Fh (with the same Bank supported for the supported subset of pages A0h-AFh) implements one **CDB instance** as a separate memory mapped command/reply message exchange facility

Note: Concurrent messaging over different CDB instances is possible only when the module supports background mode message processing.

Table 8-129 Page 9Fh Overview (CDB Message)

Byte	Size (Bytes)	Subject Area	Description
128 – 133	6	Command Message Header	Host-written command message header
134 – 135	2	Reply Message Header	Module-written header additions in reply message
136 – 255	120	Local Payload (LPL)	Area to store command and/or reply message body (or parts)

The **CDB Message Page** 9Fh contains fields for the **message header** and for (parts of) the **message body**, both for **command messages** (composed by host) and for **reply messages** (composed by module).

The **CDB Command Message Header** Bytes (9Fh:128-133) described in Table 8-130 are filled by the host to specify the CDB command to be executed by the module, the length of the used local command payload (LPLLength), of the used extended command payload (EPLLength), and a Message Checksum (CdbChkCode). These fields are filled by the host when composing the command message and evaluated by the module when receiving the command message. The module does not change the command header fields in a reply.

Table 8-130 CDB Command Message Header (Page 9Fh)

Byte	Bits	Register Name	Register Description	Type
128-129	7-0	CMDID	U16 **CDB Command Code** (CMDID) identifies a CDB command to be executed and writing this field also "sends" the CMD message from host to module for processing. *Note: See Table 9-7 for a list of commands and their CMD IDs.* *Note: The module advertises the **actual trigger condition** for the "message transfer". Ignoring some details, it may either be a write to 9Fh:128-129 or 9Fh:129, or it may be end of a multibyte write that includes Byte 9Fh:129. The detailed specification is found in the main text.* *Note: This means that the host must compose header and body either before writing to Byte 9Fh:129, or during a multi-byte write operation that includes Byte 9Fh:129.*	RW Rqd.
130-131	7-0	EPLLength	U16 **Extended Payload Length** (EPLLength) specifies the host-written number of command message body bytes in EPL, in Pages A0h-AFh. Valid lengths are 0-2048. *Note: The EPLLength field is included in the calculation of the CdbChkCode field value.*	RW Cnd.
132	7-0	LPLLength	U8 **Local Payload Length** (LPLLength) specifies the host-written number of command message body bytes in LPL, on this Page 9Fh. *Note: The LPLLength field is included in the calculation of the CdbChkCode field value.*	RW Rqd.
133	7-0	CdbChkCode	U8 **CDB Check Code** (CdbChkCode) is computed by the host as the one's complement of the arithmetic sum of Bytes 9Fh:128 to 9Fh:(136+LPLLength-1) **excluding Bytes 9Fh:133-135**. Integrity checks of EPL Pages (if any) are defined individually for each CMD as defined in chapter 9. *Note: Neither the reply header fields 9Fh:134-135 nor any EPL data used by the command are included in the check computation.* *Note: Modules can compare a correctly recomputed CdbChkCode against the received value to check message integrity.*	RW Rqd.

The **CDB Reply Message Header** Bytes (9Fh:134-135) described in Table 8-131 are filled by the module to provide information about the **reply message body** (if any) carried in one or both of the payload areas, LPL or EPL. These reply header fields are ignored by the module when receiving a command; instead they are determined and filled by the module when composing the reply.

Note: To verify later that the module has actually filled the reply header fields properly, the host may set all reply header fields to values not expected in a reply, when composing the command message.

Table 8-131 CDB Reply Message Header (Page 9Fh)

Byte	Bits	Register Name	Register Description	Type
134	7-0	RPLLength	U8 **REPLY Payload Length** (RPLLength) is computed by the module and encodes the length of REPLY data returned: 0-120: Number of LPL bytes used in the LPL area 240≤**i**≤255: Number of EPL Pages used, encoded as **i**-239 (Pages A0 to A0h+**i**-240) 121-239: **Reserved** The exact number of bytes returned in EPL Pages is determined in a CMDID specific way. RPLLength is computed in a CMDID specific way when data are returned in both LPL and EPL areas. *Note: This REPLY header field is **not** included in the CdbChkCode computation. Hosts may want to set this byte to a suitable value when composing a CMD, in order to later check if the module has actually set this field in the REPLY.*	RW Cnd.
135	7-0	RPLChkCode	U8 **REPLY Payload Check Code** (RPLChkCode) is computed by the module as follows: **No REPLY message body**: RPLChkCode = 0 **REPLY message body in Local Payload**: RPLChkCode is the one's complement of the sum of LPL Bytes 136 to (136+RPLLength-1). **REPLY message body in Extended Payload**: RPLChkCode is the one's complement of the sum of all used bytes in EPL Pages A0h to (A0h+RPLLength-240). **REPLY message body in LPL and EPL**: RPLChkCode is computed as specified for the particular CMD. *Note: This REPLY header field is **not** included in the CdbChkCode computation. Hosts may want to set this byte to a suitable value when composing a CMD, in order to later check if the module has actually set this field in the REPLY.*	RW Rqd.

The **Local Payload** (**LPL**) area 9Fh:136-255 described in Table 8-132 can carry a message body in the same Page as the message header, for message body lengths not exceeding 120 bytes.

The LPL area is used both for CMD arguments and for REPLY data (i.e. REPLY LPL may overwrite CMD LPL).

*Note: The **CDB EPL Pages** range (A0h-AFh) is called **Extended Payload** (**EPL**) and allows for large message body lengths of up to 2048 bytes, as described in the next section.*

Table 8-132 CDB Message Body (Page 9Fh)

Byte	Bits	Register Name	Register Description	Type
136-255	7-0	**LPL**	**Local Payload (LPL)**: Message body area sufficient for lengths not exceeding 120 bytes for host-written CMD data or module-written REPLY data (possibly overwriting CMD data) as specified individually for each CDB Command in chapter 9.	RW Cnd.

Message Exchange Protocol

The host must write the length, Message Checksum, payload (if applicable), and CMDID.

A host WRITE either ending at or including the LSB of the CMDID field "sends the message" and triggers the module to execute the command.

Note: If the host writes a new CMDID before a current CDB command completes the behavior is unpredictable.

While processing a CDB command, the module updates command execution status in **CdbStatus** 00h:38 or 00h:37 and signals completion in the CDB **CMD Completion Flags** 00h:8.7 or 00h:8.6.

Note: The command message header fields EPLLength and LPLLength are always included in the command message checksum CdbChkCode (even when they are not used), while the reply header fields RPLLength and RPLChkCode are never included.

Editor's Note: In future revisions of this specification, the following text should be consolidated.

Triggering CDB Command Execution

The event triggering the module to start CDB command processing depends on the advertising bit 01h:165.7.

Assuming that the current Page is 9Fh, CDB processing in the module is triggered as follows:

- When 01h:165.7=0b: when the I2CMCI STOP condition of the I2CMCI transaction for a two-byte or one-byte WRITE ending at Byte 9Fh:129 is received
- When 01h:165.7=1b: when the I2CMCI STOP condition of the I2CMCI transaction for a multi-byte WRITE that includes Byte 9Fh:129 is received

Note: In the first method, at least two WRITEs are required to invoke a CDB command, unless the host writes all 128 Bytes of Page 9Fh with wrapping. In the second method a single WRITE may be sufficient to invoke a CDB command represented fully in Page 9Fh.

Note: The host must use the advertised method.

Note: The maximum number of bytes per single ACCESS for CDB is advertised in Page 01h Byte 164.

The LPLLength field and/or the EPLLength field is populated by the host and is CDB command dependent.

A command may use either LPL or EPL or both LPL and EPL.

The length of the EPL is specified in the EPLLength field. Any checks to ensure validity of EPL data are not defined generally. If additional checks are required, they are to be specified independently for each type or group of command codes.

For example, firmware download assumes that the CDB data written in the EPL block come directly from a binary byte stream representing the vendor provided file and it is up to the vendor to add any additional checks to any data embedded within the file.

8.15.1 Triggering CDB Command on WRITE ending at 9Fh:129

The module advertises 01h:165.7=0b when CDB command execution is triggered by WRITE to Byte 9Fh:129 or to 9Fh:128-129.

Note: In this case the recommended method to invoke a CDB command with LPL consists of at least two WRITEs:

1. Write command parameters to bytes 9Fh:130-(135+LPLLength).
    a. These writes may have to be broken into multiple WRITE accesses.
    b. The maximum allowed number bytes per WRITE is advertised in 01h:164.
2. Write the 16-bit CDBID field one two-byte WRITE to 9Fh:128-129 or in two one-byte WRITEs to 9Fh:128 followed by 9Fh:129

Note: The outcome of writing bytes 9Fh:128-(135+LPLLength) in one WRITE are unpredictable in this mode.

8.15.2 Triggering CDB Command on WRITE including 9Fh:129

The module advertises 01h:165.7=1b when CDB command execution is triggered by the STOP condition of an MCI transaction for a WRITE that includes Byte 129, i.e. after the MCI write transaction has been completed.

Note: With this option the recommended method to invoke a CDB command with LPL involves one WRITE to 9Fh:128-(135+LPLLength) if the maximum WRITE length in CDB pages allows.

Note: Although Byte 129 is sent as the 2nd byte, the command is not processed until the STOP bit is received.

Note: If the maximum bytes per WRITE is at least 8, then commands without LPL (LPLLength of 0) can be invoked with a single WRITE.

Note: If the maximum bytes per WRITE is 128, then any CDB command that uses LPL only may be triggered in one WRITE.

Note: The host may still use multiple WRITEs or byte by byte WRITEs, as a single byte write to Byte 129 ends with a STOP condition and will also invoke the CDB command.

8.16 Banked Pages Range A0h-AFh (CDB Extended Payload Pages)

Pages A0-AFh contain the optional **Extended Payload** (**EPL**) area for the optional Command Data Block (CDB) messaging feature (see section 7.2), providing space for long message body data beyond the capacity of the Local Payload (LPL) area on **CDB Message Page** 9Fh (see previous section).

Each Bank of Pages A0h-AFh refers to one CDB messaging instance.

The actually supported set of EPL Pages is advertised in the **CdbMaxPagesEPL** field and is conditional on CDB support advertised in the general **CdbInstancesSupported** field (see Table 8-48).

For efficient READ or WRITE access to the multi-page EPL area modules may support Auto Paging and READ and WRITE length extensions.

Auto Paging

If the CdbAutoPagingSupported bit is set (see Table 8-48), except for the last **supported** EPL Page, host READ or WRITE accesses past the end of a Page in the EPL Page Range A0h-A**E**h cause the module to automatically increment the Page number and wrap the current address pointer to the beginning of the next Page (Byte 128). At the end of the last **supported** EPL Page, Auto Paging wraps the Page number back to A0h and the current address pointer to Byte A0h:128.

Otherwise, if the CdbAutoPagingSupported bit is cleared, host READ or WRITE accesses past the end of a Page will wrap around to Byte 128 of the same Page, as described in section B.1 and section B.2.5.2.1 respectively.

Length Extension

The allowed length of READ and WRITE operations is determined from the **CdbReadWriteLengthExtension** advertisement field (see Table 8-48).

Table 8-133 EPL Segments (Pages A0h-AFh)

Byte	Bits	Page Name	Page Description	Type
128-255	7-0	EPLSegment<i>	EPLSegment<i> is the <i>th 128 bytes segment of the overall Extended Payload (EPL) area, where <i> is a sequential segment number, starting with EPLSegment1 (on Page A0h) and counting up to EPLSegment16 (on Page AFh). The maximum usable EPL length depends on the number of EPL segments supported and does not exceed 2048 Bytes. Advertisement: 01h:163 *Note: The EPL area is used both for host-written CMD data and for module-written REPLY data (possibly overwriting CMD data), as specified for each CDB Command in chapter 9.*	RW Adv.

9 CDB Command Reference

CDB is an optional feature for command-reply message exchange between host and module.

Support of CDB is advertised in 01h:163 (section 8.4.11).

When a module supports CDB, a subset of all defined CDB commands is (conditionally) required. Information about conditionally required command groups can be found in Table 9-1. Within each conditionally required command group there is then further per-command information whether a command is conditionally required.

All other commands are optional.

As advertised in 01h:163.7-6 a module may offer more than one CDB message exchange instance.

CDB messaging instances are distinguished by the Bank Address of the CDB message exchange Pages.

9.1 CDB Command Group Summary

A single CDB data exchange between host and module follows a command-reply pattern.

The individual message interactions are therefore referred to as **CDB commands** (for short, despite the fact that there are both commands and associated responses).

Each type of CDB interaction (CDB command) is identified by a CDB command identifier (**CMD ID**) that determines purpose, structure, and semantics of a CDB data exchange.

Table 9-1 CDB Command Groups

CMD IDs		**Command Group**	**Description**	**Group**	**See**
from	**to**				
0000h	003Fh	Module Commands	CDB module level commands.	Rqd.	9.3
0040h	005Fh	Capability Advertisement	Query advertised CDB features and capabilities	Rqd.	9.4
0060h	006Fh	-	**Reserved**, for Bulk Read of Banks and Pages.		9.5
0070h	007Fh		**Reserved**, for Bulk Write of Banks and Pages.		9.6
0080h	00FFh	-	**Reserved**		
0100h	011Fh	Firmware Management	CDB Firmware Management	Adv.	9.7
0120h	01FFh	-	**Reserved**		
0200h	027Fh	Performance Monitoring	CDB Performance Monitoring	Adv.	9.8
0280h	02FFh	Data Recording	Non-volatile Data Recording and Monitoring	Adv.	9.9
0300h	037Fh	-	**Reserved**, for BERT functionality		9.10
0380h	03FFh	Diagnostics and Debug		Adv.	9.11
0400h	3FFFh	-	**Reserved**		
4000h	7FFFh	-	**Restricted** for use by OIF. This CMD ID range allows the OIF to define new groups of messages specific for the application. *Note: The CMD ID ranges of each group may provide additional sub-ranges restricted for use by OIF.*		
8000h	FFFFh	-	**Custom**		

9.2 General Messaging Rules

9.2.1 Command and Reply

One message interaction consisting of a host command (CMD) and the associated module response (REPLY).

The host prepares the CMD parameters and triggers the module to execute the CMD.

The module processes the requested command and prepares REPLY header and REPLY data.

When processing is complete, the module sets the command completion Flag and the command status Byte, depending on the CDB instance used, as follows:

- CdbCmdCompleteFlag1 (00h:8.6) and CdbStatus1 (00h:37) for CDB instance 1
- CdbCmdCompleteFlag2 (00h:8.7) and CdbStatus2 (00h:38) for CDB instance 2

These rules apply to all CDB commands and with both background mode and foreground mode processing.

9.2.2 Use of Multiple CDB Instances

One command-reply message interaction always uses the same CDB instance (i.e. the same Bank address).

For protocols involving sequences of CDB commands (such as Firmware download), each protocol execution is bound to one of the possibly several CDB messaging instances. Sending protocol messages in parallel or sequentially over different CDB instances may result in undefined module behavior.

9.3 CDB Module Commands

Table 9-2 CDB Module Commands Summary

ID	Command Title	Description	Type	Section
0000h	Query Status	The module returns a CdbStatus of success (01h) as well as basic status information of the module, such as password unlock status. *Note: This can also be used to check if CDB is supported.*	Rqd.	9.3.1
0001h	Enter Password	This command allows the host to enter a password. The module reports success or failure, especially if the password was accepted or rejected. *Note: Entering passwords is also possible by writing to Bytes 00h:122-125, but this method lacks feedback if the password was accepted or rejected.*	Adv.	9.3.2
0002h	Change Password	This command allows the host to change the Host Password. The module reply reports success or failure, i.e. if the new Host Password has been accepted and stored successfully in non-volatile memory, or if there were any errors. *Note: Changing the Host Password persistently is also possible by writing to Bytes 00h:118-121, but without feedback if the operation completed successfully.*	Adv.	9.3.3
0003h	-	**Reserved**		
0004h	Abort Processing	Abort Current Background Operation	Adv.	9.3.4
0005h-001Fh	-	**Reserved**		

9.3.1 CMD 0000h: Query Status

This Query Status command may be used to retrieve the password acceptance status (if any) and to perform a test of the CDB messaging .

The **ResponseDelay** time parameter in the message defines a host specified delay before the module is to return with the response to this command by asserting the CDB complete Flag.

Note: This delay insertion can be used as a test especially if the module advertises via 01h:163.5 that CDB commands are processed in the background. If the MCI is enabled during the delay time, CdbStatus reports "In Progress" status until the delay time has expired. If MCI is disabled, then the MCI will NACK until the delay time has elapsed and the CDB response is ready.

The return message shows the current module unlock level.

Table 9-3 CDB Command 0000h: Query Status

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Query Status CMD ID	0000h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL length	02h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	comp.
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-137	ResponseDelay	U16. Programmable delay in ms for module responding to this command. A value of 0 asks for module response as fast as possible	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**In Progress** 10 000001b: Busy capturing command 10 000010b: Busy checking/validating command 10 000011b: Busy executing command **On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	Encoded Length and Check Code for REPLY message body in LPL or EPL. See Table 8-130	2
9Fh	135	RPLChkCode		comp.
9Fh	136	Length	Length of this message payload (including this byte)	2
9Fh	137	Status	0000 0000b: Module Boot Up. 0000 0001b: Host Password Accepted. 1xxx xxxxb: Module Password accepted. Bits 'x' may contain internal vendor information. Module passwords are passwords with Bit 31 set in the password field (see section 8.2.12).	
9Fh	138-255	-	Unmodified	

9.3.2 CMD 0001h: Enter Password

The Enter Password command allows the host to enter a host password

Table 9-4 CDB Command 0001h: Enter Password

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Enter Password CMD ID.	0001h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL length	04h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	comp.
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-139	Password	Password to be entered	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000101b: CdbChkCode error 01 000110b: Password error – not accepted	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	0
9Fh	135	RPLChkCode	See Table 8-130	0
9Fh	136-255	-	No data returned. Content not specified.	

9.3.3 CMD 0002h: Change Password

The Change Password command allows the host to change the Host Password.

Table 9-5 CDB Command 0002h: Change Password

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Change Password CMD ID.	0002h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL length	04h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	comp.
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-139	New password		
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error (e.g. Bit 31 is set). 01 000101b: CdbChkCode error 01 000110b: Insufficient privilege to change password	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	0
9Fh	135	RPLChkCode	See Table 8-130	0
9Fh	136-255	-	No data returned. Content not specified.	

9.3.4 CMD 0004h: Abort Processing

The Abort command allows the host to abort any current background operation.

Table 9-6 CDB Command 0004h: Abort

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Abort Processing CMD ID	0004h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL is not used	00h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	FCh
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-255	-	**Reserved**	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	0
9Fh	135	RPLChkCode	See Table 8-130	0
9Fh	136-255	-	No data returned. Content not specified.	

9.4 CDB Features and Capabilities Advertisement

Table 9-7 CDB Feature and Capabilities Commands Overview

ID	Command Title	Description	Type	Section
0040h	Module Features	Identify which module level commands are supported. CMD IDs 0000h to 00FFh.	Rqd.	9.4.1
0041h	Firmware Management Features	Identify which firmware management features are supported. CMD IDs 0100h to 011Fh.	Rqd.	9.4.2
0042h	Performance Monitoring Features	Identify which Performance and Data Monitoring commands are supported	Rqd.	9.4.3
0043h	BERT and Diagnostic Features	Identify which BERT and Diagnostic features are supported	Rqd.	9.4.4
0044h-005Fh	-	**Reserved**		

9.4.1 CMD 0040h: Module Features

This command is used to query which CDB commands are supported.

Table 9-8 CDB Command 0040h: Module Features

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Module Features CMD ID	0040h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL is not used	00h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	BFh
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-255	-	**Reserved**	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	36
9Fh	135	RPLChkCode	See Table 8-130	comp.
9Fh	136	CDB Flags	**Reserved** for additional CDB Flags.	00h
9Fh	137	**-**	**Reserved**	00h
9Fh	138	CMDs 0000h-0007h	Support advertisement for Module Commands A set bit indicates that the command is supported	
9Fh	139	CMDs 0008h-000Fh		
9Fh	140	CMDs 0010h-0017h		
9Fh	141	CMDs 0018h-001Fh		
9Fh	142	CMDs 0020h-0027h		
9Fh	143	CMDs 0028h-002Fh		
9Fh	144	CMDs 0030h-0037h		
9Fh	145	CMDs 0038h-003Fh		
9Fh	146	CMDs 0040h-0047h		
9Fh	147	CMDs 0048h-004Fh		
9Fh	148	CMDs 0050h-0057h		
9Fh	149	CMDs 0058h-005Fh		
9Fh	150	CMDs 0060h-0067h	Support advertisement for Bulk Read commands	
9Fh	151	CMDs 0068h-006Fh	Support advertisement for Bulk Read commands	
9Fh	152	CMDs 0070h-0077h	Support advertisement for Bulk Write commands	
9Fh	153	CMDs 0078h-007Fh	Support advertisement for Bulk Write commands	
9Fh	154-169	CMDs 0080h-00FFh		
9Fh	170-171	MaxCompletionTime	U16 Maximum CDB command execution time in ms, of all supported CDB commands. *Note: If exceeded, the host may send the CDB Abort Command.* *Note: The maximum possible MaxCompletionTime is about one minute (65.535 seconds).*	
9Fh	172-255	-	Content not specified.	

9.4.2 CMD 0041h: Firmware Management Features

This command is used to query which firmware management features are supported and to query command performance attributes.

Table 9-9 CDB Command 0041h: Firmware Management Features

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Firmware Management Features CMD ID	0041h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL is not used	00h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	BEh
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-255	-	**Reserved**	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	18
9Fh	135	RPLChkCode	See Table 8-130	comp.
9Fh	136	-	**Reserved**	0h
9Fh	137.7	ImageReadback	0b = Full Image Readback Not Supported 1b = Full Image Readback Supported (see section 7.3.1)	
	137.6-4	-	**Reserved**	000b
	137.3	MaxDurationCoding	0b = max duration multiplier **M** is 1 1b = max duration multiplier **M** is 10 This bit encodes a multiplier value **M** which governs the interpretation of values found in the U16 array of advertised max durations in Bytes 144-153 of this message: These advertised values are multiplied by **M**.	
	137.2	SkippingErasedBlocks	0b = Skipping erased blocks Not Supported 1b = Skipping erased blocks Supported	
	137.1	CopyCmd	0b = CMD 0108h (Copy image) Not Supported 1b = CMD 0108h (Copy image) Supported	
	137.0	AbortCmd	0b = CMD 0102h (Abort) Not Supported 1b = CMD 0102h (Abort) Supported	
9Fh	138	StartCmdPayloadSize	This defines the number of bytes that the host must extract from the beginning of the vendor-delivered binary firmware image file and send to the module in CMD 0101h (Start).	
9Fh	139	ErasedByte	This is the value representing an erased byte. The purpose of advertising this byte is to optionally reduce download time by allowing the host to skip sending blocks of the image containing ErasedByte values only. *Note: Typically for NAND flash the erased state is FFh, for other flash or EEPROM technology it is 00h.*	
9Fh	140	ReadWriteLengthExt	ReadWriteLengthExt = **i** specifies the allowable **additional** number of byte octets in a READ or a WRITE, specifically for **Firmware Management Commands** (IDs 0100h-01FFh) as follows	

Page	Byte	Field Name	Description	Value
			EPL: For accessing the multi-page **EPL** field, the allowable length extension is **i** byte octets (8 bytes). **LPL**: For accessing the **LPL** field on page **9Fh**, the allowable length extension is min(**i**, 15) byte octets. This leads to the maximum length of a READ or a WRITE **Value** **Maximum Number of Bytes (EPL)** 0: 8 bytes (no extension of general length limit) **i**: 8 * (1+**i**) bytes (0 ≤ **i** ≤ 255) 255: 8 * 256 = **2048** bytes **Value** **Maximum Number of Bytes (LPL)** 0: 8 bytes (no extension of general length limit) **i**: 8 * (1+**i**) bytes (0 ≤ **i** ≤ 15) **i**: 8 * 16 = **128** bytes (16 ≤ **i** ≤ 256) *Note: See also the CdbReadWriteLengthExtension Byte 01h:164 which defines a similar length extension for arbitrary CDB commands.*	
9Fh	141	WriteMechanism	Firmware update supported mechanism. 00h: None Supported. 01h: Write to LPL supported. 10h: Write to EPL supported. 11h: Both Write to LPL and EPL supported.	
9Fh	142	ReadMechanism	Firmware read / readback support mechanism. 00h: None Supported. 01h: Read via LPL supported. 10h: Read via EPL supported. 11h: Both Read via LPL and EPL supported.	
9Fh	143	HitlessRestart	0: CMD Run Image causes a reset. Traffic is affected. 1: CMD Run Image may reset but module will do its best to maintain traffic and management states. Data path functions are not reset.	
9Fh	144-145	MaxDurationStart	U16 Maximum time in **M** ms for a CDB Start command to complete execution, where **M** is defined by bit 137.3	
9Fh	146-147	MaxDurationAbort	U16 Maximum time in **M** ms for a CDB Abort command to complete execution, where **M** is defined by bit 137.3	
9Fh	148-149	MaxDurationWrite	U16 Maximum time in **M** ms for a CDB Write command to complete execution, where **M** is defined by bit 137.3	
9Fh	150-151	MaxDurationComplete	U16 Maximum time in **M** ms for a CDB Complete command to complete execution, **M** defined by bit 137.3	
9Fh	152-153	MaxDurationCopy	U16 Maximum time in **M** ms for a CDB Copy command to complete execution, where **M** is defined by bit 137.3	
9Fh	154-255	-	Content not specified.	

9.4.3 CMD 0042h: Performance Monitoring Features

Table 9-10 CDB Command 0042h: Performance Monitoring Features

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Performance Monitoring Features CMD ID	0042h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL is not used	00h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	BDh
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-255	-	**Reserved**	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	32
9Fh	135	RPLChkCode	See Table 8-130	comp.
9Fh	136	CMDs 0200h-0207h support	Each bit represents a mask. If bit is "1" then command is supported D0: CMD 0200h is supported. .. D7: CMD 0207h is supported.	
9Fh	137	CMDs 0208h-020Fh	Each bit represents a mask. If bit is "1" then command is supported D0: CMD 0208h is supported. .. D7: CMD 020Fh is supported.	
9Fh	138-151	CMDs 0210h-027Fh	Bitmask defines command supported.	
9Fh	152	CMDs 0280h-0287h	Data Monitoring Command Supported. D0: CMD 0280h supported. .. D7: CMD 0287h is supported.	
9Fh	153-167	CMDs 0288h-02FFh		
9Fh	168-255	-	Content not specified.	

9.4.4 CMD 0043h: BERT and Diagnostics Features

Table 9-11 CDB Command 0043h: BERT and Diagnostics Features

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	BERT and Diagnostics Features CMD ID	0043h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL is not used	00h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	BCh
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-255	-	**Reserved**	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	32
9Fh	135	RPLChkCode	See Table 8-130	comp.
9Fh	136	CMDs 0300h-0307h support	Each bit represents a mask. If bit is "1" then command is supported D0: CMD 0300h is supported. .. D7: CMD 0307h is supported.	
9Fh	137	CMDs 0308h-030Fh	Each bit represents a mask. If bit is "1" then command is supported D0: CMD 0308h is supported. .. D7: CMD 030Fh is supported.	
9Fh	**138-151**	CMDs 0310h-037Fh	Bitmask defines command supported.	
9Fh	152	CMDs 0380h-0387h	Data Monitoring Command Supported. D0: CMD 0380h supported. .. D7: CMD 0387h is supported.	
9Fh	153-167	CMDs 0388h-03FFh		
9Fh	168-255	-	Content not specified.	

9.5 CDB Bulk Read Commands

Table 9-12 CDB Bulk Read Commands Overview

ID	Command Group	Description	Type	Section
0060h-006Fh	-	**Reserved**		

9.6 CDB Bulk Write Commands

Table 9-13 CDB Bulk Write Commands Overview

ID	Command Group	Description	Type	Section
0070h-007Fh	-	**Reserved**		

9.7 CDB Firmware Management Commands

Firmware management using CDB is an optional feature.

See section 8.2.9 for background on module firmware as a potential aggregate of several firmware components and on aggregate firmware version identification.

Table 9-14 CDB Firmware Download Commands Overview

ID	Command Title	Description	Type	Section
0100h	Get Firmware Info	When the host issues this command, the module returns the requested FW information of all field-updateable firmware in the module.	Rqd.	9.7.1
0101h	Start Firmware Download	The host issues this command to initiate a firmware update. The module may completely erase the target or simply acknowledge and prepare the module firmware for any appropriate update or dynamically erase as each data block arrives. On success, the host may begin sending the firmware image using command codes 0103h-0107h.	Adv.	9.7.2
0102h	Abort Firmware Download	Aborts the FW Download if a FW Start has been issued.	Adv.	9.7.3
0103h	Write Firmware Block LPL	With this command, the host downloads a firmware image block previously stored in the LPL area. The module transfers that firmware image block into non-volatile storage. *Note: Each image block transfer from host to module is covered by the CDB command block checksum; this checksum does not ensure that the image has been properly transferred to non-volatile storage.*	Adv.	9.7.4
0104h	Write Firmware Block EPL	With this command, the host downloads a firmware image block previously stored in the EPL Page(s). The module transfers that firmware image block into non-volatile storage.	Adv.	9.7.5
0105h	Read Firmware Block LPL	The host may use this command to read back the firmware image that was most recently written to non-volatile storage. The module copies the image from non-volatile storage to the LPL Page. The firmware image transfer from the module to the host is covered by the CDB command block checksum, but this checksum does not ensure that the image has been properly transferred from non-volatile storage to the LPL Page.	Adv.	9.7.6
0106h	Read Firmware Block EPL	The host may use this command to read back the firmware image that was most recently written to non-volatile storage. The module copies the image from non-volatile storage to the EPL Page(s). The firmware image transfer from the module to host is covered by a block checksum, but this checksum does not ensure that the image has been properly transferred from non-volatile storage to the EPL Page(s).	Adv.	9.7.7
0107h	Complete Firmware Download	The host issues this command when the entire firmware image has been written via LPL or via EPL Pages, or to stop the download after failure. If this command is not issued, the firmware cannot be run even if the image is properly loaded to non-volatile storage. The module validates the checksum associated with the image when the host issues this command.	Adv.	9.7.8

ID	Command Title	Description	Type	Section
0108h	Copy Firmware Image	When multiple images are supported for a given subsystem in the module, this command causes the module to copy an image from one non-volatile storage location to another one. It is assumed that the copy firmware image command includes a validation process by the module firmware to ensure the copied image is valid. The CDB complete firmware image command 0107h does not need to be called after a copy firmware image command.	Adv.	9.7.9
0109h	Run Firmware Image	This command is used to start and run a selected image. This command transfers control from the currently running firmware to a selected firmware that is started. It can be used to switch between firmware versions, or to perform a restart of the currently running firmware.	Adv.	9.7.10
010Ah	Commit Firmware Image	The host uses this command to commit the running image so that the module will boot from it on future boots. *The assumption is that the host has a time period where it runs the new firmware and "accepts" the new firmware. During this time, if a reset occurs, the previously committed image will be run. When the host issues this command, the module will mark a non-volatile storage location to be used after future module resets. Only the running image can be committed. This is to avoid committing a "bad" image.*	Adv.	9.7.11
010Bh - 011Fh	-	**Reserved**		

1

9.7.1 CMD 0100h: Get Firmware Info

This command returns the firmware versions and firmware default running images that reside in the module. Firmware images A, B and either a factory or boot firmware image version.

Note: See section 8.2.9 for semantical expectations about unique identification of the entire firmware aggregate by the version triple (major revision, minor revision, build number).

Table 9-15 CDB Command 0100h: Get Firmware Info

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Get Firmware Info CMD ID	0100h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL is not used	00h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	FEh
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-255	-	No host-written payload	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	110
9Fh	135	RPLChkCode	See Table 8-130	comp.
9Fh	136	FirmwareStatus	Bitmask to indicate FW Status. Image in **Bank A**: Bit 0: Operational Status Bit 1: Administrative Status Bit 2: Validity Status Bit 3: Reserved Image in **Bank B**: Bit 4: Operational Status Bit 5: Administrative Status Bit 6: Validity Status Bit 7: Reserved Encoding as follows: **Operational Status**: 1 = running, 0 = not running **Administrative Status**: 1=committed, 0=uncommitted **Validity Status**: 1 = invalid, 0 = valid *Note: Zero-encoding of valid maintains backwards compatibility with CMIS 4.0* Hints: 0x00h Factory image is running (if supported) See also section 7.3.1.4 for a more detailed description.	
9Fh	137	ImageInformation	Bit 0: Firmware image A information in 9Fh:138-173 Bit 1: Firmware image B information in 9Fh:174-209 Bit 2: Factory or Boot image information in 9Fh:201-245	
9Fh	138	ImageAMajor	Image A firmware major revision	
9Fh	139	ImageAMinor	Image A firmware minor revision	
9Fh	140-141	ImageABuild	Image A firmware build number	

Page	Byte	Field Name	Description	Value
9Fh	142-173	ImageAExtraString	Additional information	
9Fh	174	ImageBMajor	Image B firmware major revision	
9Fh	175	ImageBMinor	Image B firmware minor revision	
9Fh	176-177	ImageBBuild	Image B firmware build number	
9Fh	178-209	ImageBExtraString	Additional information	
9Fh	210	FactoryBootMajor	Factory or Boot image firmware major revision	
9Fh	211	FactoryBootMinor	Factory or Boot image firmware minor revision	
9Fh	212-213	FactoryBootBuild	Factory or Boot image firmware build number	
9Fh	214-245	FactoryBootExtraStr	Additional information	
9Fh	246-255	-	Content unspecified	

9.7.2 CMD 0101h: Start Firmware Download

The module may erase the whole image or simply prepare for the firmware to be updated. The duration of this command depends on the host-written payload of the command. A complete erase of an image may take several seconds.

Table 9-16 CDB Command 0101h: Start Firmware Download

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Start firmware download process CMD ID	0101h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL length	comp.
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	comp.
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-139	ImageSize	U32 Size of firmware image to download into the module. This should be the file size including the LPL bytes sent as vendor data in this message.	var.
9Fh	140-143	-	**Reserved**	0h
9Fh	144-255	VendorData	The vendor may send up to 112 bytes of information in the Start Firmware Download command. It is recommended that the binary file delivered has up to 112 bytes of header that is sent to the module. The information within this field can e.g. be used by a vendor to reject an incorrect file (binary file) and prevent firmware loading of an incorrect file presented to the module.	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**In Progress** 10 000001b: Busy processing command, CMD captured 10 000010b: Busy processing command, CMD checking 10 000011b: Busy processing command, CMD execution **On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	0
9Fh	135	RPLChkCode	See Table 8-130	0
9Fh	136-255	-	No data returned. Content not specified.	

9.7.3 CMD 0102h: Abort Firmware Download

Aborts the firmware update process.

Modules supporting only a single image should not implement this command and advertise accordingly.

Table 9-17 CDB Command 0102h: Abort Firmware Download

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Abort Firmware Download CMD ID	0102h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL is not used	00h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	FCh
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-255	No host-written payload		
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**In Progress** 10 000001b: Busy processing command, CMD captured 10 000010b: Busy processing command, CMD checking 10 000011b: Busy processing command, CMD execution **On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	0
9Fh	135	RPLChkCode	See Table 8-130	0
9Fh	136-255	-	No data returned. Content not specified.	

9.7.4 CMD 0103h: Write Firmware Block LPL

Download one block of the firmware image via LPL

Table 9-18 CDB Command 0103h: Write Firmware Block LPL

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Write Firmware Block LPL CMD ID	0103h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	The actual length of the firmware block in the FirmwareBlock field + 4.	comp.
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	comp.
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-139	BlockAddress	U32 Starting byte address of this block of data within the supplied image file minus the size of the "Start Command Payload Size". See section 7.3.1.2.	
9Fh	140-255	FirmwareBlock	One block of the firmware image. The actually needed length may be shorter than the available FirmwareBlock field size. This actual length of the block is defined in Byte 132 (LPLLength), see above.	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**In Progress** 10 000001b: Busy processing command, CMD captured 10 000010b: Busy processing command, CMD checking 10 000011b: Busy processing command, CMD execution **On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	0
9Fh	135	RPLChkCode	See Table 8-130	0
9Fh	136-255	-	No data returned. Content not specified.	

9.7.5 CMD 0104h: Write Firmware Block EPL

Download one block of the firmware image via EPL

Table 9-19 CDB Command 0104h: Write Firmware Block EPL

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Write Firmware Block EPL CMD ID	0104h
9Fh	130-131	EPLLength	Length of the firmware block in the FirmwareBlock field, which may span over multiple Pages.	comp.
9Fh	132	LPLLength	LPL length	04h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	comp.
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-139	BlockAddress	U32 Starting byte address of this block of data within the supplied image file minus the size of the "Start Command Payload Size". See section 7.3.1.2.	
9Fh	140-255	-	**Reserved**	
CMD Data (EPL)				
A0h-AFh	128-255	FirmwareBlock	Up to 2048 Bytes. Actual Length specified in EPLLength	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**In Progress** 10 000001b: Busy processing command, CMD captured 10 000010b: Busy processing command, CMD checking 10 000011b: Busy processing command, CMD execution **On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	0
9Fh	135	RPLChkCode	See Table 8-130	0
9Fh	136-255	-	No data returned. Content not specified.	
REPLY Data (EPL)				
A0h-AFh	128-255	-	No data returned. Content not specified.	

9.7.6 CMD 0105h: Read Firmware Block LPL

Upload one block of firmware image from the indicated block in non-volatile storage via LPL

Note: The image being read is the most recently written, implicitly.

Table 9-20 CDB Command 0105h: Read Firmware Block LPL

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Read Firmware Block LPL CMD ID	0105h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL length	06h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	comp.
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-139	BlockAddress	U32 Starting byte address of this block of data within the supplied image file minus the size of the size of the "Start Command Payload Size". See section 7.3.1.2.	
9Fh	140-141	Length	U16 Number of bytes to read back to the LPL in this command, starting at the indicated address.	
9Fh	142-255	-	**Reserved**	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**In Progress** 10 000001b: Busy processing command, CMD captured 10 000010b: Busy processing command, CMD checking 10 000011b: Busy processing command, CMD execution **On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	var.
9Fh	135	RPLChkCode	See Table 8-130	comp.
9Fh	136-139	Address of block	U32 Base address of the data block within the firmware image	
9Fh	140-255	ImageData	Up to 116 Bytes	

9.7.7 CMD 0106h: Read Firmware Block EPL

Upload one block of firmware image from the indicated block in non-volatile storage via EPL

Note: The image being read is the most recently written, implicitly.

Table 9-21 CDB Command 0106h: Read Firmware Block EPL

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Read Firmware Block EPL CMD ID	0106h
9Fh	130-131	EPLLength	Length of EPL	variable
9Fh	132	LPLLength	LPL length	06h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	comp.
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-139	BlockAddress	U32 Starting byte address of this block of data within the supplied image file minus the size of the size of the "Start Command Payload Size". See section 7.3.1.2.	
9Fh	140-141	Length	U16 Number of bytes to read back to the EPL in this command, starting at the indicated address.	
9Fh	142-255	-	**Reserved**	
CMD Data (EPL)				
A0h-AFh	128-255	-	**Reserved**	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**In Progress** 10 000001b: Busy processing command, CMD captured 10 000010b: Busy processing command, CMD checking 10 000011b: Busy processing command, CMD execution **On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	variable
9Fh	135	RPLChkCode	See Table 8-130	comp.
9Fh	136-255	-	LPL unused. Contents unspecified	
REPLY Data (EPL)				
A0h-AFh	128-255	ImageData	Up to 128 Bytes. Actual Length specified in RPLLength	

9.7.8 CMD 0107h: Complete Firmware Download

When the host issues this command, the module validates the complete firmware image and then return success or failure (could be checksum failure).

Note: This command can also be used to abort a firmware download. The module detects an incomplete download and is expected to return a failure.

Table 9-22 CDB Command 0107h: Complete Firmware Download

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Complete Firmware Download CMD ID	0107h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL is not used	00h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	F7h
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-255	-	No host-written payload	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**In Progress** 10 000001b: Busy processing command, CMD captured 10 000010b: Busy processing command, CMD checking 10 000011b: Busy processing command, CMD execution **On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	0
9Fh	135	RPLChkCode	See Table 8-130	0
9Fh	136-255	-	Nothing returned	

9.7.9 CMD 0108h: Copy Firmware Image

Copy Firmware Image is an optional command within the firmware download commands that may be used in the firmware update process.

Note: This command is typically used in a system where both images that are written to flash are identical, and thus if the host desires to have both images A and B be identical, it can simply tell the module to copy from the running image which is committed to the uncommitted backup image.

Table 9-23 CDB Command 0108h: Copy Firmware Image

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Copy Firmware Image CMD ID	0108h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL length	01h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	comp.
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136	Copy Direction	ABh: Copy Image A into Image B BAh: Copy Image B into Image A	
9Fh	137-255	-	**Reserved**	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**In Progress** 10 000001b: Busy processing command, CMD captured 10 000010b: Busy processing command, CMD checking 10 000011b: Busy processing command, CMD execution **On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	6
9Fh	135	RPLChkCode	See Table 8-130	comp.
9Fh	136-139	Length	U32 Number of bytes copied	
9Fh	140	CopyDirection	ABh: Image A was copied into Image B BAh: Image B was copied into Image A	
9Fh	141	CopyStatus	00h : Copy Successful 01h : Copy Failed	
9Fh	142-255	-	**Reserved**	

9.7.10 CMD 0109h: Run Firmware Image

The host uses this command to run a selected image from module internal firmware banks.

For example, after the firmware has been updated, this command may be used to switch to the new firmware version. The host may use CMD 0100h to determine both the active and inactive firmware versions.

Executing the Run Image command may potentially be hitless, non-disruptive or minimally disruptive to high-speed traffic.

Behavior of the module and its control loop transients during resets are vendor and technology dependent.

Table 9-24 CDB Command 0109h: Run Firmware Image

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Run FW Image CMD ID	0109h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL length	04h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	comp.
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136	-	**Reserved**	
9Fh	137	ImageToRun	00h = Traffic affecting Reset to Inactive Image. 01h = Attempt Hitless Reset to Inactive Image 02h = Traffic affecting Reset to Running Image. 03h = Attempt Hitless Reset to Running Image	
9Fh	138-139	DelayToReset	U16 Indicates the delay in ms after receiving this command before a reset will occur, starting from the time the CDB complete Flag is set ~~(or NACK clearing if the CDB background mode is not set)~~. *Note: When DelayToReset is 0, the module may reset before the host has read the CdbStatus message.*	
9Fh	140-255	-	**Reserved**	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**In Progress** 10 000001b: Busy processing command, CMD captured 10 000010b: Busy processing command, CMD checking 10 000011b: Busy processing command, CMD execution **On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	
9Fh	135	RPLChkCode	See Table 8-130	
9Fh	136-255	-	Nothing returned. Contents unspecified	

9.7.11 CMD 010Ah: Commit Firmware Image

A Commit is the process where the "running" image is set to be the image to be used on exit from module reset. In other words, a committed image is the image that will run and is expected to be a 'good' firmware version to run upon any resets (including watch dog).

This command is used to switch the committed image after the firmware update process, when the new firmware is running and when the host has determined that the new firmware is working properly.

Note: Commit Image commits only the image that it is currently running. If it was possible to commit a non-running, a bad version may be committed and attempted to run after the next module reset.

Table 9-25 CDB Command 010Ah: Commit Image

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Commit Image CMD ID	010Ah
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL is not used	00h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	comp.
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-255	-	**Reserved**	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**In Progress** 10 000001b: Busy processing command, CMD captured 10 000010b: Busy processing command, CMD checking 10 000011b: Busy processing command, CMD execution **On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	0
9Fh	135	RPLChkCode	See Table 8-130	0
9Fh	136-255	-	Nothing returned. Contents unspecified	

9.8 CDB Performance Monitoring Commands

Table 9-26 CDB Performance Monitoring Commands Overview

ID	Command Title	Description	Type	Section
0200h	Control PM	General Performance Monitoring Controls	Rqd.	9.8.1
0201h	Get PM Feature Information	Advertisement on optional PM is supported.	Rqd.	9.8.2
0202h-020Fh	-	**Reserved**		
0210h	Get PM Module LPL	Get Module-level X16 PM using LPL	Adv.	9.8.3
0211h	Get PM Module EPL	Get Module-level X16 PM using EPL	Adv.	9.8.3
0212h	Get PM Host Side LPL	Get Lane-specific host side X16 PM using LPL	Adv.	9.8.4
0213h	Get PM Host Side EPL	Get Lane-specific host side X16 PM using EPL	Adv.	9.8.4
0214h	Get PM Media Side LPL	Get Lane-specific media side X16PM using LPL	Adv.	9.8.5
0215h	Get PM Media Side EPL	Get Lane-specific media side X16 PM using EPL	Adv.	9.8.5
0216h	Get PM Data Path LPL	Get Lane-specific Data Path X16 PM using LPL	Adv.	9.8.6
0217h	Get PM Data Path EPL	Get Lane-specific Data Path X16 PM using EPL	Adv.	9.8.6
0218h-027Fh	-	**Reserved**		

The following Table provides an Overview of the PM Observables available for retrieval by CDB commands.

See Table 8-122 for VDM Observable IDs

Table 9-27 CDB Performance Monitoring Observables

PM Group	Instances	ID: Type and Observable	Unit	Corresponds to Observables
Module	1	0: S16 Module Temperature 1: S16 Vcc 2: S16 Aux1Mon 3: S16 Aux2Mon 4: S16 Aux3Mon	1/256 degC 100 uV see → see → see →	0:00h:14-15 0:00h:16-17 0:00h:18-19 0:00h:20-21 0:00h:22-23
Host Side	Per Lane (across Banks)	0: U16 Host Side Lane SNR 1: F16 Host Side PAM4 LTP 2: F16 Host Side Pre-FEC BER	1/256 dB 1/256 dB	VDM ID 6 VDM ID 8 n/a
Media Side	Per Media Lane (across Banks)	0: U16 Tx Laser Bias 1: U16 Tx Optical Power 2: U16 Rx Optical Power 3: S16 Per-Lane Laser Temperature	0.1 uW 2uA * x 0.1 uW	0-3:11h:170-185 0-3:11h:154-169 0-3:11h:186-201 VDM ID 4
Data Path	Per Data Path (across Banks)	0: F16 Frame Error Count (FERC) 1: F16 Media Side Pre-FEC BER		VDM ID 19 VDM ID 13

9.8.1 CMD 0200h: Control PM

Controls the behavior of CDB Performance Monitoring and its behavior with respect to the Versatile Diagnostic Monitoring in Page 20h-2Fh.

This section details the messages used to extract PM data records such as minimum, average, maximum values.

Note: Unless otherwise specified, a 2-byte, 4-byte, or 8-byte value is encoded in Big Endian format, i.e. the lowest byte address stores the most significant byte of the word.

Table 9-28 CDB Command 0200h: Control PM

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Control PM CMD ID	0200h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL length	04h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	comp.
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136.1-7	-	**Reserved**	0
	136.0	LinkMode	0b: PM Objects are Independent 1b: PM Objects are the same as 20h-2Fh (linked) When PM objects are linked, this means that PM data in Page 20-2Fh are based on the same objects as the PM records that are returned by CDB Get PM commands. It also means that clearing statistics using either method in Page 2Fh will clear the statistics in CDB, too.	
9Fh	137	-	**Reserved**	00h
9Fh	138.1-7	-	**Reserved**. Set to 0.	0
	138.0	ClearAllStatistics	0b: No operation 1b: Clear all statistics (minimum, average, maximum) for all observables for all lanes at the same time, across all Banks, in a best-effort manner	
9Fh	139	-	**Reserved**	00h
9Fh	140-255	-	**Reserved** Not sent	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete	1
00h	37 or 38	CdbStatus	**On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	0
9Fh	135	RPLChkCode	See Table 8-130	0
9Fh	136-255	-	**Reserved**	

9.8.2 CMD 0201h: Get PM Feature Information

Identifies which of the PM monitors defined in CMD 0210h to 0217h is supported by the module.

Table 9-29 CDB Command 0201h: Get PM Feature Information

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Get PM Feature Information CMD ID	0201h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL is not used	00h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	FCh
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-255	-	**Reserved**	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	4
9Fh	135	RPLChkCode	See Table 8-130	comp.
9Fh	136	HostSideMonitors	Bit 0: Bool: Host Side SNR monitor available Bit 1: Bool: Host Side LTP monitor available	X
9Fh	137	MediaSideMonitors	Bit 0: Bool: Media Side SNR monitor available Bit 1: Bool: Media Side LTP monitor available	X
9Fh	138	-	**Reserved**	00h
9Fh	139	-	**Reserved**	00h
9Fh	140-255	-	**Reserved** Not sent	

9.8.3 CMD 0210h/0211h: Get Module PM LPL/EPL

Table 9-30 CDB Command 0210h/0211h: Get Module PM LPL/EPL

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Get Module PM using LPL CMD ID	0210h
			Get Module PM using EPL CMD ID	0211h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL length	05h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	comp.
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136.7	ClearOnRead	0b: return selected PM data 1b: return selected PM data and then reset their statistics	
	136.1-6	-	Reserved	
	136.0	RecordType	0b: Return 6-byte PM record (min, mean, max) 1b: Return 8-byte PM record (append "current" value)	
9Fh	137	Observables	Bit 0: Module Temperature Bit 1: Vcc Bit 2: Aux1 Bit 3: Aux2 Bit 4: Aux3 Bit 5-7: Reserved	
9Fh	138	-	**Reserved**	
9Fh	139	-	**Restricted** (OIF)	
9Fh	140	-	**Custom**	
9Fh	141-255	-	**Reserved**	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**In Progress** 10 000001b: Busy processing command, CMD captured 10 000010b: Busy processing command, CMD checking 10 000011b: Busy processing command, CMD execution **On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header				
9Fh	134	RPLLength	See Table 8-130	comp.
9Fh	135	RPLChkCode	See Table 8-130	comp.
REPLY Data (LPL) (CMD 0210h)				
9Fh	136-255	LPL PM data (up to 120 bytes)	PM record of one observable consists of 6 or 8 bytes: X16 minimum value X16 average (mean) value X16 maximum value X16 current value (if requested) *Note: A maximum of 15 to 20 records can be returned using LPL, depending on the requested PM record length.* The sequence of PM records is the same as the sequence of set bits in the Observables field, in order of ascending significance.	

Page	Byte	Field Name	Description	Value
REPLY Data (EPL) (CMD 0211h)				
A0h to AFh	128-255	EPL PM data (maximum number of bytes depends on available EPL Pages)	PM record of one observable consists of 6 or 8 bytes: X16 minimum value X16 average (mean) value X16 maximum value X16 current value (if requested) *Note: The maximum number of records depends on the size of the EPL and the requested PM record length.* Data is contiguous across EPL Pages and the sequence of PM records corresponds to the sequence of set bits in the Observables field, in order of ascending significance.	

9.8.4 CMD 0212h/0213h: Get PM Host Side LPL/EPL

Table 9-31 CDB Command 0212h/0213h: Get PM Host Side LPL/EPL

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Get PM Host Side LPL CMD ID	0212h
			Get PM Host Side EPL CMD ID	0213h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL length	14h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	comp.
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136.7	ClearOnRead	0b: return selected PM data 1b: return selected PM data and then reset their statistics	
	136.1-6	-	Reserved	
	136.0	RecordType	0b: Return 6-byte PM record (min, mean, max) 1b: Return 8-byte PM record (append "current" value)	
9Fh	137-139	-	**Reserved** *In future, we could define start/stop/increment lanes here if needed for modules with more than 32 lanes.*	
9Fh	140-143	Lanes	U32 Lanes is a bitmask indicating which host lane is present, where bit i represents lane i+1	
9Fh	144	Observables	Bit 0: Host Side Lane SNR Bit 1: Host Side PAM4 LTP Bit 2: Host Side Pre-FEC BER Bits 3-7: Reserved	
9Fh	145-147	-	**Reserved**	
9Fh	148-151	-	**Restricted** (OIF)	
9Fh	152-155	-	**Custom**	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**In Progress** 10 000001b: Busy processing command, CMD captured 10 000010b: Busy processing command, CMD checking 10 000011b: Busy processing command, CMD execution **On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header				
9Fh	134	RPLLength	See Table 8-130	comp.
9Fh	135	RPLChkCode	See Table 8-130	comp.
REPLY Data (LPL) (CMD 0212h)				
9Fh	136-255	LPL PM data (up to 120 bytes)	PM record of one PM observable consists of 6 or 8 bytes: X16 minimum value X16 average (mean) value X16 maximum value X16 current value (if requested) *Note: A maximum of 15 to 20 records can be returned using LPL depending on the requested PM record length.* The sequence of PM records is the same as the sequence of set bits in the Observables field, in order of ascending significance.	

Page	Byte	Field Name	Description	Value
REPLY Data (EPL) (CMD 0213h)				
A0h to AFh	128-255	EPL PM data (number of bytes depends on available Pages)	PM record of one observable consists of 6 or 8 bytes: X16 minimum value X16 average (mean) value X16 maximum value X16 current value (if requested) *Note: The maximum number of records depends on the size of the EPL and the requested PM record length.* Data is contiguous across EPL Pages and the sequence of PM records corresponds to the sequence of set bits in the Observables field, in order of ascending significance.	

9.8.5 CMD 0214h/0215h: Get PM Media Side LPL/EPL

Table 9-32 CDB Command 0214h/0215h: Get PM Media Side LPL/EPL

Page	Byte	Field Name (Type)	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID (U16)	Get PM Media Side LPL CMD ID	0214h
			Get PM Media Side EPL CMD ID	0215h
9Fh	130-131	EPLLength (U16)	EPL is not used	0000h
9Fh	132	LPLLength	LPL length	14h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	comp.
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136.7	ClearOnRead	0b: return selected PM data 1b: return selected PM data and then reset their statistics	
	136.1-6	-	Reserved	
	136.0	RecordType	0b: Return 6-byte PM record (min, mean, max) 1b: Return 8-byte PM record (append "current" value)	
9Fh	137-139	-	**Reserved** for modules with more than 32 lanes.	
9Fh	140-143	Lanes	U32 Lanes is bitmask indicating which media lane is present, where bit i represents lane i+1.	
9Fh	144	Selected PM data	Bit 0: Media Side SNR Bit 1: Media Side PAM4 LTP Bits 2-7 : Reserved	
9Fh	145	Selected PM data	Bit 0: Tx Laser Bias Bit 1: Tx Power Bit 2: Rx Power Bit 3: Per-Lane Laser Temperature Bits 4-7 : Reserved	
9Fh	146-147		**Reserved**	
9Fh	148-151		**Restricted** (OIF)	
9Fh	152-155		**Custom**	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**In Progress** 10 000001b: Busy processing command, CMD captured 10 000010b: Busy processing command, CMD checking 10 000011b: Busy processing command, CMD execution **On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header				
9Fh	134	RPLLength	See Table 8-130	comp.
9Fh	135	RPLChkCode	See Table 8-130	comp.

Page	Byte	Field Name (Type)	Description	Value
REPLY Data (LPL) (CMD 0214h)				
9Fh	136-255	LPL PM data (up to 120 bytes)	PM record of one observable consists of 6 or 8 bytes: X16 minimum value X16 average (mean) value X16 maximum value X16 current value (if requested(*Note: A maximum of 15 to 20 records can be returned using LPL, depending on the requested PM record length.* The sequence of PM records is the same as the sequence of set bits in the Observables field, in order of ascending significance.	
REPLY Data (EPL) (CMD 0215h)				
A0h to AFh	128-255	EPL PM data (maximum number of bytes depends on available EPL Pages)	PM record of one PM parameter consists of 6 or 8 bytes: X16 minimum value X16 average (mean) value X16 maximum value X16 current value (if requested) *Note: The maximum number of records depends on the size of the EPL and the requested PM record length.* Data is contiguous across EPL Pages and the sequence of PM records corresponds to the sequence of set bits in the Observables field, in order of ascending significance.	

9.8.6 CMD 0216h/0217h: Get Data Path PM LPL/EPL

Table 9-33 CDB Command 0216/0217h: Get Data Path PM LPL/EPL

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Get PM Data Path LPL CMD ID	0216h
			Get PM Data Path EPL CMD ID	0217h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL length	14h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	comp.
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136.7	ClearOnRead	0b: return selected PM data 1b: return selected PM data and then reset their statistics	
	136.1-6	-	Reserved	
	136.0	RecordType	0b: Return 6-byte PM record (min, mean, max) 1b: Return 8-byte PM record (append "current" value)	
9Fh	137-139	-	**Reserved** for modules with more than 32 lanes.	
9Fh	140-143	DataPaths	U32 DataPaths is a mask indicating which Data Path is present, where bit i represents the Data Path with DataPathID (lowest lane number) i+1	
9Fh	144	Observables	Bit 0: Frame Error Count (FERC, uncorrectable frames) Bit 1: Media Side Pre-FEC BER Bits 2-7 : Reserved	
9Fh	145-147	-	**Reserved**	
9Fh	148-151	-	**Restricted** (OIF)	
9Fh	152-155	-	**Custom**	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**In Progress** 10 000001b: Busy processing command, CMD captured 10 000010b: Busy processing command, CMD checking 10 000011b: Busy processing command, CMD execution **On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header				
9Fh	134	RPLLength	See Table 8-130	comp.
9Fh	135	RPLChkCode	See Table 8-130	comp.
Returned Data Path Lane PM Using LPL (0216h)				
9Fh	136-255	LPL PM data (up to 120 bytes)	PM record of one observable consists of 6 or 8 bytes: X16 minimum value X16 average (mean) value X16 maximum value X16 current value (if requested) *Note: A maximum of 15 to 20 records can be returned using LPL depending on the requested PM record length.* The sequence of PM records data is the same as the sequence of set bits in the Observables field, in order of ascending significance.	

Page	Byte	Field Name	Description	Value
Returned Data Path Lane PM Using EPL (0217h)				
A0h to AFh	128-255	EPL PM data (maximum number of bytes depends on available EPL Pages)	PM record of one observable consists of 6 or 8 bytes: X16 minimum value X16 average (mean) value X16 maximum value X16 current value (if requested) *Note: The maximum number of records depends on the size of the EPL and the requested PM record length*. Data is contiguous across EPL Pages and the sequence of PM records corresponds to the sequence of set bits in the Observables field, in order of ascending significance.	

9.9 CDB Data Monitoring and Recording Commands

Table 9-34 CDB Data Monitoring and Recording Commands Overview

ID	Command Title	Description	Type	Section
0280h	Data Monitoring and Recording Controls		Rqd.	0
0281h	-	**Reserved** (for Advertising)		0
0282h-02FFh	-	**Reserved**		
0290h	Temperature Histogram		Opt.	9.9.3
0291h-02FFh	-	**Reserved**		

9.9.1 CMD 0280h: Data Monitoring and Recording Controls

Table 9-35 CDB Command 0280h: Data Monitoring and Recording Controls

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Data Monitoring Controls CMD ID	0280h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL length	04h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	comp.
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136	-	**Reserved**	00h
9Fh	137	-	**Reserved**	00h
9Fh	138.1-7	-	Reserved. Set to 0	0
	138.0	ClearAllStatistics	0b: No operation. 1b: **Clear all statistics** (minimum, average, maximum) for all monitored observables for all lanes, at the same time (best-effort).	
9Fh	139	-	**Reserved**	00h
9Fh	140-255	-	**Unused**	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	
9Fh	135	RPLChkCode	See Table 8-130	
9Fh	136-255	-	**Unused**	

9.9.2 CMD 0281h: Data Monitoring and Recording Advertisement

Reserved for advertisement of the PM features defined in CMDs 0290-029Fh

Table 9-36 CDB Command 0281h: Data Monitoring and Recording Advertisements

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID		0281h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL is not used	00h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	7Ch
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-255	-	**Reserved**. Not sent	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	See Table 8-130	
9Fh	135	RPLChkCode	See Table 8-130	
9Fh	136-255	-	**Reserved**	

9.9.3 CMD 0290h: Temperature Histogram

This is an optional feature of a module to track the number of seconds in nonvolatile memory, in which the module has operated within the defined temperature bins, since the last time the host or user has cleared the temperature histogram.

The sampling interval and the interval at which the module writes to nonvolatile memory is module dependent. In order for the host not to lose histogram data, the host may send a command to save the current histogram to NVR before decommissioning or unplugging a module from its slot.

The timing accuracy of the internal clock is not defined here.

Table 9-37 CDB Command 0290h: Temperature Histogram

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Temperature Histogram CMD ID	0290h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL length	01h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	comp.
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply. See Table 8-130*	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136	SubCommands	Bit 0: Reserved Bit 1: Save current histogram to NVR (host triggered) Bit 2: Return histogram (10degC bins) Bit 3-6: Reserved Bit 7: Clear Temperature Histogram (should be NV)	
9Fh	137-255	--	**Reserved**	

Page	Byte	Field Name	Description	Value
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**In Progress** 10 000001b: Busy processing command, CMD captured 10 000010b: Busy processing command, CMD checking 10 000011b: Busy processing command, CMD execution **On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	Encoded Length and Check Code for REPLY message body in LPL or EPL. See Table 8-130	52
9Fh	135	RPLChkCode		comp.
9Fh	136	SubCommands	Echo of host-written values	
9Fh	137	NHoursToNextWrite	0: Module does not indicate hours to next write to NVR. 1-254: Temperature Histogram will be written to NVR within this specified number of hours. *Note: Depends on device technology use, NVR writes may need to be triggered by host.* 255: NVR write needs to be triggered by host. If the module were to write NVR autonomously, some operations may share EEPROM or flash device and may cause smaller microcontrollers to stop responding to MCI while the autonomous write is in progress.	
9Fh	138-139	-	**Reserved**	
9Fh	140-143	TotalSeconds	U32 Total time of temperature histogram accumulation (s)	
9Fh	144-147	TenDegBinBelowM5	U32 Total seconds operated with temp. below -5 degC.	
9Fh	148-151	TenDegBin0	U32 Total seconds operated in temp. interval [- 5, 5[degC	
9Fh	152-155	TenDegBin10	U32 Total seconds operated in temp. interval [5, 15[degC	
9Fh	156-159	TenDegBin20	U32 Total seconds operated in temp. interval [15, 25[degC	
9Fh	160-163	TenDegBin30	U32 Total seconds operated in temp. interval [25, 35[degC	
9Fh	164-167	TenDegBin40	U32 Total seconds operated in temp. interval [35, 45[degC	
9Fh	168-171	TenDegBin50	U32 Total seconds operated in temp. interval [45, 55[degC	
9Fh	172-175	TenDegBin60	U32 Total seconds operated in temp. interval [55, 65[degC	
9Fh	176-179	TenDegBin70	U32 Total seconds operated in temp. interval [65, 75[degC	
9Fh	180-183	TenDegBin80	U32 Total seconds operated in temp. interval [75, 85[degC	
9Fh	184-187	TenDegBinAbove85	U32 Total seconds operated with temp. above 85 degC	

9.10 CDB PRBS BERT Commands

See also features available using Pages 13h-14h.

Table 9-38 CDB BERT Commands Overview

ID	Command Title	Description	Type	Section
0300h	-	**Reserved** for PRBS related capabilities		
030xh	-	**Reserved** for PRBS Generator Config, Enable and Disable		
030xh	-	**Reserved** for PRBS Error Injector		
030xh	-	**Reserved** for PRBS Detector (Checker) Config, Enable and Disable		
030xh	-	**Reserved** for PRBS Error Counts		
030xh	-	**Reserved** for PRBS BER		
030xh-037Fh	-	**Reserved**		

9.11 CDB Diagnostics and Debug Commands

Table 9-39 CDB Diagnostics and Debug Commands Overview

ID	Command Title	Description	Type	Section
0380h	Loopbacks	**Reserved** for Loopbacks	Adv.	9.11.1
0390h	-	**Reserved** for PAM4 Histogram		
03A0h	-	**Reserved** for Eye Monitors		

9.11.1 CMD 0380h: Loopbacks

Basic loopback modes are available using features supported via Page 13h and Page 14h.

This CDB command is **reserved** for additional loopback capabilities.

Table 9-40 CDB Command 0380h: Loopbacks

Page	Byte	Field Name	Description	Value
CMD Header Fields				
9Fh	128-129	CMDID	Loopbacks CMD ID	0380h
9Fh	130-131	EPLLength	EPL is not used	0000h
9Fh	132	LPLLength	LPL is not used	00h
9Fh	133	CdbChkCode	Check Code over 9Fh:128-132 and LPL. See Table 8-130	7Ch
9Fh	134	RPLLength	*Note: Initiator may fill those reply fields, to later verify field updates by the target in the reply.* See Table 8-130	undef.
9Fh	135	RPLChkCode		undef.
CMD Data (LPL)				
9Fh	136-255	-	**Reserved**	
REPLY Status				
00h	8.6 or 8.7	CdbCmdCompleteFlag	Set by module when the CDB command is complete.	1
00h	37 or 38	CdbStatus	**In Progress** 10 000001b: Busy processing command, CMD captured 10 000010b: Busy processing command, CMD checking 10 000011b: Busy processing command, CMD execution **On Success** 00 000001b: Success **On Failure** 01 000000b: Failed, no specific failure 01 000010b: Parameter range error or not supported 01 000101b: CdbChkCode error	
REPLY Header and Data (LPL)				
9Fh	134	RPLLength	Empty reply message body. See Table 8-130	0
9Fh	135	RPLChkCode		0
9Fh	136-255	-	**Reserved**	

10 Management Timing Specifications

This chapter provides fixed, unadvertised timing specifications for

- management control signals (Management Signal Layer)
- register access and register interdependencies (Register Access Layer)
- interdependencies between signal conditions and the register map
- interdependencies between high speed signal conditions

In certain cases, advertisement registers may exist that allow a module to advertise tighter timing specifications.

Note: The timing specifications in this chapter are intentionally limited to timing specifications related to module management, management registers, or management control signals. Timing specifications for the underlying Management Communication Interface (MCI) are defined in Appendix B and must be consistent with timings specified here.

Note: The timings between high speed signals conditions are listed here because they are relevant for a host managing the module.

10.1 Timings for Management Control Signals (MSL)

Table 10-1 defines timing parameters and requirements for management control signal waveforms.

Table 10-2 defines timing parameters and requirements related to the latency of effects caused by management control signals or by other host or operator actions (power supply, hot-plug).

Table 10-1 Signal Waveform Timings

Parameter	Symbol	Min	Unit	Conditions
Min Reset Assert Duration	t_reset_init	10	us	Minimum pulse duration of the Reset signal to initiate a module reset

Table 10-2 Effect Latency Timings

Parameter	Symbol	Max	Unit	Conditions
Max Time to reach manageability in ModuleLowPwr	tMgmtInit	2000	ms	Time from power-on, hot plug, or Reset release until the START condition of a READ retrieving the default register value of an arbitrary register (including a page switch).

Note: Power on and hot plug events are defined as the instant when supply voltages reach and remain at or above the minimum electrical level specified in the form factor-dependent specification. The Reset release event is defined as the time of the most recent DEASSERTING edge of a correctly ASSERTED Reset signal.

10.2 Timings for Register Access (RAL)

Register ACCESS related timings are defined with reference to observable events at the Management Communication Interface port of the module.

Note: Future CMIS versions should seek specification methods that are independent of the MCI variant used.

10.2.1 Single ACCESS Timings

Table 10-3 defines timing parameters and requirements related to one individual register ACCESS.

Note: Refer to Appendix B for MCI implementation dependent timings related to register access

Table 10-3 Timings for Register Access

Parameter	Symbol	Max	Unit	Conditions
Max READ Latency Variation	tREAD	500	µs	Allowable Latency Variation, i.e. the maximum time the module may internally delay returning the requested data in an MCI transaction for a READ, beyond the fastest transaction execution. *Note: This RAL parameter is an abstraction of the I2CMCI clock stretch time*

10.2.2 Consecutive ACCESS Timings

This subsection defines timing requirements involving two adjacent register accesses with the possibility of an ACCESS hold-off period after the first ACCESS.

An **ACCESS hold-off period** is a period of time during which a module rejects READ or WRITE accesses for reasons specified in the relevant sections of this specification (see sections 5.2.3 and 5.2.4).

The **actual duration** of an ACCESS hold-off period is defined as the **time interval between** the event when the module receives the **I2CMCI STOP** token of the MCI transaction representing the WRITE causing the begin of the ACCESS hold-off period **and** the event when the module sends the **I2CMCI ACK** token for the I2CMCI command byte in the first MCI transaction representing a non-rejected ACCESS.

Note: A host not willing to wait for specified maximum durations can retry a rejected ACCESS or use the TEST primitive (defined in section 5.2.2.3) to poll if earlier ACCESS is possible.

Table 10-4 defines the **maximum duration** of each type of ACCESS hold-off period.

Table 10-4 Maximum ACCESS Hold-Off Durations

Parameter	Symbol	Max	Unit	Conditions
Max WRITE Latency for regular (volatile) register or memory *Note: Almost all CMIS memory locations are classified as volatile.*	tWRITE	10	ms	Maximum time for the result of a WRITE to be completed (written value retrieved in a READ), expressed as the longest ACCESS hold-off period after a WRITE to an address representing volatile memory or management register. *Note: This RAL parameter is an abstraction of the I2CMCI parameter tNACK.*
Max WRITE Latency for **NV** (persistent) register or memory	tWRITENV	80	ms	Maximum time for the result of a WRITE to be completed (written value retrieved in a READ), expressed as the longest ACCESS hold-off period after a WRITE to an address representing non-volatile memory or management register. *Note: This RAL parameter is an abstraction of the I2CMCI parameter tWR.*
Maximum **B**ank/**P**age **C**hange Time	tBPC	10	ms	Maximum duration of the ACCESS hold-off after a PageMapping register change (see section 8.2.13)
Maximum **CDB** command **f**oreground processing completion time	tCDBF	4960	ms	Maximum advertised length of the ACCESS hold-off period while module processes a CDB command in foreground processing mode. *Note: The maximum CDB busy time defaults to*

				80 ms but the module may advertise a different value ranging from 0 to 4960 ms (see section 8.4.11)
Maximum CDB command **c**apture time	tCDBC	80	ms	Maximum advertised length of the ACCESS hold-off period while module captures a CDB command for subsequent execution.

10.2.3 Register Content Dependencies

This section defines timings for cause-to-effect latencies of changes in certain registers to become visible in dependent registers, especially for bulk data.

The module does not prevent access to stale data in these cases (i.e. ACCESS that is too early is not rejected).

Measurement condition: The effect latency of a WRITE to a register causing an effect in specified dependent registers is the **time interval between** the event when the module receives the **I2CMCI STOP** token of the MCI transaction representing the WRITE **and** the event when the module sends the **I2CMCI ACK** token for the I2CMCI command byte in the first MCI transaction representing a READ of an arbitrary byte in the dependent registers that retrieves non-stale, correct data.

Table 10-5 Content Dependency Timings

Parameter	Symbol	Max	Unit	Conditions
Maximum **D**iagnostics **D**ata **C**ontent **S**witch Time	tDDCS	10	ms	Maximum time from STOP of the WRITE access to the Diagnostics Data Selector (see section 8.12.) to the START of a READ access retrieving correct data
Maximum **VDM** Statistics **F**reeze Time	tVDMF	10	ms	Maximum time required by the module to freeze all VDM statistics reporting registers after the STOP of the Freeze request WRITE (see section 8.14.6)

Note: In future versions of this specification, timings may be added for other interdependencies such as Flag summary registers (dependent on Flag updates), statis registers dependent on State Machine state changes, or register effects depending on hardware reconfigurations being completed, etc.

10.3 Timings between Conditions and Management Registers or Signals

This section describes timing requirements for interactions between signals and the management register map.

10.3.1 Interrupt and Flag Related Timings

Note: The relative timing behavior of different but correlated management information representations such as Interrupt request signal, and Flag and Flag bits, is largely unspecified and can vary significantly across modules.

The condition interrupt signal assertion or de-assertion refers to the voltage of the interrupt signal crossing the voltage threshold associated with assertion or de-assertion, respectively.

Table 10-6 Condition to Interrupt Timings

Parameter	Symbol	Max	Unit	Conditions
Max Interrupt Assert Time	ton_Int	200	ms	Time from onset of condition or occurrence of event until associated unmasked Interrupt asserted
Max Rx LOS Interrupt Assert Time	ton_los	100	ms	Time from onset of Rx LOS condition to Rx LOS Flag raised and Interrupt asserted
Max Rx LOS Interrupt Assert Time (optional fast mode)	ton_losf	1	ms	
Max Tx Failure Interrupt Assert Time	ton_Txfail	200	ms	Time from Tx Failure state to Tx Failure Flag raised and Interrupt asserted
Max Flag Assert Time	ton_flag	200	ms	Time from onset of condition or occurrence of event to associated Flag bit raised and Interrupt asserted

Table 10-7 Register to Interrupt Timings

Parameter	Symbol	Max	Unit	Conditions
Max Interrupt Deassert Time	toff_Int	0.5	ms	Time from reading (clear on read) last Flag set until Interrupt deasserted, assuming that no conditions nor events causing a Flag setting are present
Max Mask Assert Time	ton_mask	100	ms	Time from Mask bit raised while associated Flag bit is set until Interrupt deasserted
Max Mask Deassert Time	toff_mask	100	ms	Time from Mask bit ceased while associated Flag bit is set until Interrupt asserted

10.3.2 High Speed Signal Related Timings

Table 10-8 defines causal timing requirements between register access and high-speed signals.

Table 10-8 Register to High Speed Signal Timings

Parameter	Symbol	Max	Unit	Conditions
Max Tx Output Disable Latency [1]	ton_txdis	100	ms	Time from setting a OutputDisableTx<i> bit until optical output falls below 10% of nominal. For I2CMCI the time interval begins with the STOP token ending the MCI write transaction
Max Fast Mode Tx Output Disable Latency (optional) [1]	ton_txdisf	3	ms	
Max Tx Output UnDisable Latency [1]	toff_txdis	400	ms	Time from clearing a OutputDisableTx<i> bit until an enabled optical output rises above 90% of nominal. For I2CMCI the time interval begins with the STOP token ending the MCI write transaction
Max Fast Mode Tx Output UnDisable Latency (optional) [1]	toff_txdisf	10	ms	
Max Rx Output Disable Assert Time	ton_rxdis	100	ms	Time from the termination of the Rx Output Disable register write sequence until an enabled Rx output falls below 10% of nominal
Max Rx Output Disable Deassert Time	toff_rxdis	100	ms	Time from the termination of the Rx Output Disable register write sequence until an enable Rx output rises above 90% of nominal,
Max Auto Squelch Disable Assert Time	ton_sqdis	100	ms	This applies to Rx and Tx Auto Squelch and is the time from the termination of the register write sequence until auto squelch functionality is disabled.
Max Auto Squelch Disable	toff_sqdis	100	ms	This applies to Rx and Tx Auto Squelch and is the time

Parameter	Symbol	Max	Unit	Conditions
Deassert Time				from the termination of the register write sequence until auto squelch functionality is enabled.

Note 1: Values specified here place an upper limit on advertised timings like MaxDurationDPTxTurnOff and MaxDurationDPTxTurnOn (01h:168).

10.4 Timings between High Speed Signal Conditions

Table 10-9 defines causal timing requirements between conditions of high-speed signals.

Table 10-9 Signal Condition to Signal Condition Timings

Parameter	Symbol	Max	Unit	Conditions
Max Rx Auto Squelch Reaction Latency	ton_Rxsq	15	ms	Time from onset of loss of Rx input signal condition until the squelched output condition is reached
Max Rx Auto UnSquelch Reaction Latency	toff_Rxsq	-	ms	Obsoleted: time for normal Rx output signal condition restored when loss of Rx input signal condition clears
Max Tx Auto Squelch Reaction Latency	ton_Txsq	400	ms	Time from onset of loss of Tx input signal condition until the squelched output condition is reached
Max Tx Auto UnSquelch Reaction Latency	toff_Txsq	-	ms	Obsoleted: time for normal Tx output signal condition restored when loss of Tx input signal condition clears

Note: These timings are not directly related to management registers or to management signals. They provide an upper bound for module timing behavior as may be assumed by a host managing a CMIS module. Form factor specific timing specifications may be tighter.

Appendix A Form Factor Specific Signal Names

Table A-1 associates generic names for management related signals used in this specification with form factor-specific signal names for four form factors.

Table A-1 Form Factor Dependent Signal Name Associations

CMIS Signal Name	Layer	QSFP-DD	QSFP	OSFP	COBO
Reset	MSL	ResetL	ResetL	RSTn	ResetL
Interrupt		IntL	IntL	INT	IntL
LowPwrRequestHW		LPMode	LPMode	LPWn	LPMode
ModSel	MCI	ModSelL	ModSelL	-	ModSelL

These generic CMIS hardware signals assume two **symbolic** signal levels, ASSERTED and DEASSERTED (section 2.3.2), in order to address realizations both in active-low logic and in active-high logic.

Note: Active-high voltage signals are sometimes equated, silently, as positive logic bits with a representation of TRUE=1 and FALSE=0, while active-low voltage signals are equated as negative or inverted logic bits, with TRUE=0 and FALSE=1. These silent associations can be confusing and are therefore avoided by introducing symbolic signal levels with a fixed association to the truth values TRUE and FALSE.

The correspondence between the symbolic signal levels ASSERTED and DEASSERTED, and the voltage levels of a given logic convention used in the hardware specification of a form factor is defined in the following table:

Table A-2 Symbolic Logical Signal Values

CMIS Symbolic Value	**Active-High Logic**	**Active-Low Logic**	**Positive Bit Logic**	**Boolean Logic**
Signal Level	Voltage Level	Voltage Level	CMIS Bool Type	Truth Values
ASSERTED	High	Low	1	TRUE
DEASSERTED	Low	High	0	FALSE

Note: Boolean logic expressions and equations in CMIS are mostly formulated in terms of numerical ***bit values*** *0 and 1, thereby assuming a positive logic representation (TRUE=1 and FALSE=0). Mapping conventions defined in section 2.3.2 allow the specification to use the abstract signal levels ASSERTED and DEASSERTED for hardware signals together with bit values in those pseudo-Boolean expressions.*

Appendix B Management Communication Interface Definitions

This appendix contains specification variants for the Management Communication Interface (MCI) for use by reference in module hardware (form factor) specifications.

Relationship to Hardware Specifications

The MCI variant used by a particular module is generally assumed to be specified in a form-factor-dependent hardware specification. Please refer to the relevant form factor hardware specification to determine which MCI variant is employed for a given case.

Note: This documentation structure inverts the referential dependency of previous CMIS revisions. In the new structure the hardware specifications refer to CMIS when CMIS management is to be supported.

Relationship to other parts of the CMIS Specification

Each MCI variant specified in this appendix provides the necessary data transfer services for the basic READ, WRITE, and TEST register access operations specified in chapter 5.

Note: This separation between the register access layer (see section 5.2) and the underlying management communication interface in this specification does not need to be visible in an implementation.

Note: The main purpose of separating data transfer mechanism across the MCI and the basic CMIS management operations (READ from or WRITE to management addressable memory in a module) is to allow the CMIS management application layer to be easily migrated to other physical communication interfaces[1], in the future.

MCI Variants Available

In this CMIS revision, the following MCI variants are available

Table B-1 Management Communication Interface Variants

MCI Variant	Description	Specification
I2CMCI	The initial I2C based MCI specification for CMIS *Note: This was called TWI in previous versions*	Section B.2

Note: New MCI variants may be added to future CMIS revisions on request, for form factors that intend to implement CMIS based management over a new, e.g. higher bandwidth, communication infrastructure.

B.1 Generic Definitions

A **management communication interface** provides transactional data transfer services to implement the basic management register access operations specified in chapter 5: READ, WRITE, TEST.

The data transfer occurs between a managing host and a managed module, which take the roles of initiator and target, respectively, in the neutral data communication perspective of the MCI.

B.1.1 Communication Roles and Transactions

The management communication interface distinguishes two roles.

An **initiator** initiates and terminates a **transaction** for data transfer.

A **target** responds to the initiator within a transaction.

A transaction to implement a READ access is called a read transaction.

A transaction to implement a WRITE access is called a write transaction.

Within a transaction data can be transferred from the initiator to the target and from the target to the initiator.

Note: In CMIS the host is always the initiator, and the module is always the target.

[1] Editor’s note: It might be argued that the low abstraction level of the CMIS management operations may not be ideal for the management of future modules with increasingly complex functionality. Even if that is, hypothetically, assumed to be true, there is certainly value in allowing higher bandwidth CMIS implementations, both for porting existing management applications and for expanding the use of CDB messaging.

Note: Completion of a data transfer transaction and associated register or memory modifications in the module are not necessarily synchronous: internal modifications may occur only after the transaction is terminated by the initiator. Any necessary synchronization conditions are specified in section 5.2.4.

B.1.2 Current Byte Address

The target maintains one internal **current byte address** variable containing the Byte address of the Byte to be accessed in the next access in case that no address is provided in that next access. When an address is specified in the access, the current byte address is set to that address.

The target increments the current byte address by one after each byte read or byte written and rolls over after incrementing past the end of the current 128-byte memory area (see section 8.1).

Roll-over in Lower Memory sets the current byte address to 0 when the address to be incremented is 127.

Roll-over in Upper Memory sets the current byte address to 128 when the address to be incremented is 255.

Note: The current byte address roll-over may be combined with other mechanisms in certain (optional) address ranges, such as auto paging in the CDB page range described in Section 7.2.

The current byte address remains valid between two register accesses as long as power to the target is maintained and as long as no MCI protocol violations occur.

The current byte address is indeterminate upon loss of power to or reset of the module (i.e. after initialization), or upon an access failure (e.g. I2CMCI transactions not terminated by a STOP condition).

B.2 I2C-Based Management Communication Interface (I2CMCI)

The I2CMCI specification for communication between an initiator and a target is a tailored and simplified variant of the general I2C [1] specification.

B.2.1 Communication Topology

The I2CMCI protocol for data transfer appears as a point-to-point interaction between initiator and target because the protocol frame does not provide for address information (see section B.2.4).

The physical management communication topology in a host system might still be a multi-point I2C bus when host and module support an optional generic hardware control signal for module selection (ModSel).

B.2.2 I2CMCI Control Signals

The following discrete hardware control signal is not a required part of the I2CMCI but needs to be accounted for when present

- a module selection signal (**ModSel**) instructing the module to activate or passivate its management interface, in order to effectively allow MCI bus sharing (when such precautions are needed)

Note: This is motivated by the fact that the I2CMCI data transfer protocol described in Appendix B does not employ endpoint addressing and therefore inherently allows only point to point communication at any one time.

When module selection is used in a host system, the host must provide setup time on the ModSel line of all modules on the same bus and must not change the ModSel line of any module until the management communication is complete and hold time requirements are satisfied, as specified in section B.2.7.1.

B.2.3 Physical Layer Signals

The I2CMCI electrical layer consists of a clock signal (**SCL**) and a data signal (**SDA**).

The initiator drives the SCL signal to clock data and control information on SDA onto the I2C bus.

Both initiator and target latch the state of SDA on the positive transitioning edge of SCL.

The initiator shall initially use a 0-400 kHz SCL clock speed. If a higher management interface speed is supported by the target (see appropriate Hardware Specification) the initiator may later switch to a faster 0-1 MHz SCL clock speed.

The SDA signal is bi-directional.

During binary data transfer, the SDA signal transitions when SCL is low.

Between binary data transfers, SDA transitions when SCL is high are used to mark either the beginning (**START**) or ending (**STOP**) of a data transfer.

Note: An I2CMCI data transfer from initiator to target can be viewed as a serial stream of four tokens, 1, 0, START, STOP, while the reverse data stream is binary.

All bytes (control bytes, address bytes, and data bytes) are transferred bit-serially over the SDA, with the most significant bit (MSB) of each byte transferred first.

A LOW voltage encodes 0. A HIGH voltage encodes 1.

Other electrical signal specifications are found in the appropriate hardware/module specification.

The serial data communication protocol for the data transfer associated with CMIS register access operations over the I2C bit-serial communication link is described in detail in the remainder of this section.

B.2.4 Serial Communication Protocol

The I2CMCI serial data communication protocol governs the I2C bus transactions used for the data transfer associated with CMIS register access operations.

Note: In this section a subtle distinction needs to be made between a I2C bus transaction (a data transfer which begin with START and end either with a STOP or with a new START) and the CMIS register access operation (READ or WRITE) that is conveyed from the host to the module by means of that bus transaction. As will be seen below, register modifications caused by bus transactions occur after the transaction is terminated.

B.2.4.1. Basic Definitions and Protocol Elements

B.2.4.1.1. Start Condition (START)

A high-to-low transition of SDA with SCL high is a **START** condition.

All I2CMCI bus transactions begin with a START condition generated by the initiator.

Note: To distinguish from binary data 0 and 1, a START condition can be represented by a backslash \ character

B.2.4.1.2. Stop Condition (STOP)

A low-to-high transition of SDA with SCL high is a **STOP** condition.

All regular I2CMCI bus transactions end with a STOP condition generated by the initiator.

Note: To distinguish from binary data 0 and 1, a STOP condition can be represented by a slash / character

B.2.4.1.3. Word Size (Byte) and Bit Serial Transmission Order

The I2CMCI word size is 8-bits.

I2CMCI words are transferred bit-serially with the most significant bit (MSB) first.

Note: The following text uses the term byte to refer to 8-bit words.

B.2.4.1.4. Basic Operation Encoding (Control Byte)

The generic I2C address byte is used in I2CMCI as a control byte indicating the type of basic management operation that is conveyed in the bus transaction.

In I2CMCI the I2C address 10100001b (A1h) indicates a READ access and is called the **read control byte**.

In I2CMCI the I2C address 10100000b (A0h) indicates a WRITE access and is called the **write control byte**.

B.2.4.1.5. Acknowledge (ACK and NACK)

After sending a byte, the side driving the I2C bus releases the SDA line for one-bit time. During this period the receiving side of the I2C bus pulls SDA low (zero) in order to acknowledge (**ACK**) that it has received the byte.

Not pulling the SDA low (one) in this period is interpreted as a negative acknowledge (**NACK**).

The target sends ACK after each individual control byte received during a read or write transaction, and after each individual data byte received during a write transaction.

The initiator sends ACK after each individual data byte received during a read transaction, except for the last data byte, where it terminates the read transaction by sending NACK and then STOP.

B.2.4.2. Protocol Reset and Recovery

B.2.4.2.1. Power-On Reset

The I2CMCI adapter (interface circuitry) enters a reset state upon application of power.

B.2.4.2.2. Protocol Reset and Recovery

Synchronization issues may cause the initiator and target to disagree on the specific bit location currently being transferred, the type of transaction, or even if a transaction is in progress.

The I2CMCI protocol has no explicitly defined reset mechanism. The following procedure may force completion of a currently pending transaction and cause the target to release SDA.

1. The initiator shall provide up to nine SCL clock cycles (drive low, then high) to the target
2. The initiator shall monitor SDA while SCL is high on each cycle.
3. On any clock cycle, if the target releases SDA, the initiator shall initiate a STOP condition. The initiator is then free to send a START condition for the next transaction
4. If SDA remains low after nine clock cycles the I2CMCI protocol reset has failed

B.2.5 Serial MCI Transactions for READ/WRITE/TEST Access

This section describes the I2CMCI transactions between initiator and target, for READ, WRITE, TEST.

I2CMCI provides defined data transfer transactions to implement the following basic register access operations

Table B-2 I2C MCI Transactions

I2C MCI Transaction	Register Access Primitive	Section
Read one byte from current Byte address	VALUE = READ(1)	B.2.5.1.1
Read n>1 bytes sequentially, starting at current Byte address	VALUES = READ(n)	B.2.5.1.2
Read one byte from given Byte address	VALUE=READ(ByteAddress, 1)	B.2.5.2.1
Read n>1 bytes sequentially, starting at given Byte address	VALUES=READ(ByteAddress, n)	B.2.5.2.2
Write one byte to a given address	WRITE(ByteAddress, Value)	B.2.5.3
Write N>1 bytes sequentially, starting at given Byte address	WRITE(ByteAddress, Val1, …, ValN)	B.2.5.4
Test if target is ready to accept transaction (ACK polling)	TEST()	B.2.5.5

Note: Standard length limitations for reading or writing multiple bytes are specified in section 5.2 with possible exceptions explicitly specified elsewhere.

Read/Write Control Byte and Response

In the first part of a transaction, after the START condition, the first byte of a I2CMCI bus transaction is a control byte consisting of a fixed 7-bit part 1010000b followed by a bit indicating the type of transaction: A **read** transaction is requested if this bit is 1 (high). A **write** transaction is requested if this bit is 0 (low).

Upon reception of the control byte the target asserts the SDA signal low to acknowledge (ACK) receipt unless a transaction rejection is required (see section B.2.6.2).

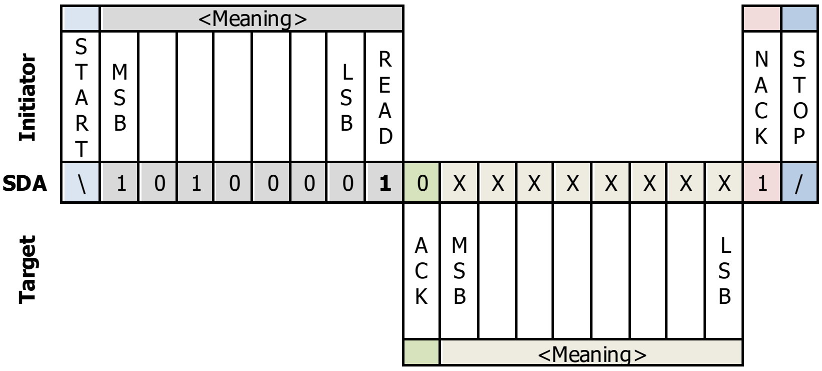
Byte Address and Data Bytes

When the first part of a transaction was acknowledged, addresses and/or data are transmitted in units of bytes, in the second part of a transaction. Format and interpretation of this transferred byte sequence is specific for each basic transaction (as described section B.2.4.2).

In transactions transferring more than one data byte, the data bytes are transmitted in the order of increasing byte addresses (cyclically increasing in case of roll-over described in section B.1.2).

Illustrations

To visually illustrate the bus transactions, figures like the following are used to show the bidirectional token stream on the SDA line, consisting of 0, 1, \ (START), and / (STOP) tokens. Colors and structure around the SDA line are used to show when Initiator or Target drive the SDA wire, and for what purpose.



B.2.5.1. Transactions for a Byte Read Operation

B.2.5.1.1. Transaction for a Current Address Read Operation

The initiator generates a START condition and sends the read control byte (10100001b).

The target sends ACK followed by the data byte retrieved from the address in the current byte address.

The initiator terminates the transaction with NACK and STOP.

The target increments the current byte address after orderly termination of the transaction (STOP) and before accepting a new transaction.

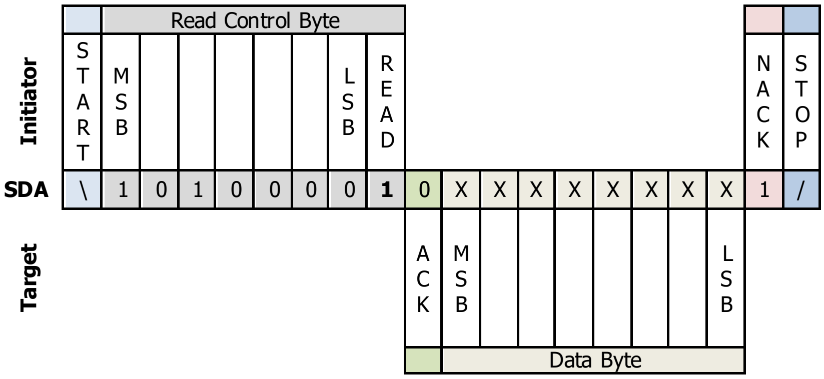


Figure B-1 Current Address Read

B.2.5.1.2. Transaction for a Random Read Operation

A random read transaction is implemented as an aborted dummy write transaction, followed by a current address read transaction.

The dummy write transaction is used to load (but not increment) the target byte address into the current byte address, from which the subsequent current address read transaction then reads. The procedure is as follows:

The initiator generates a START condition and sends the target byte address after the write control byte (10100000b).

The initiator then generates another START condition (aborting the dummy write) and begins a current address read transaction by sending a read control byte (10100001b).

The target acknowledges each byte received.

When the byte address of the dummy write is received, the target updates the current byte address. When the target then sees the write transaction aborted (i.e. when it receives a START instead of a byte to be written), the current byte address is not incremented and so contains the address to be read in the subsequent current address read transaction.

The initiator terminates the transaction with NACK and STOP when it has received the requested byte.

The target increments the current byte address after orderly termination of the transaction (STOP) and before accepting a new transaction.

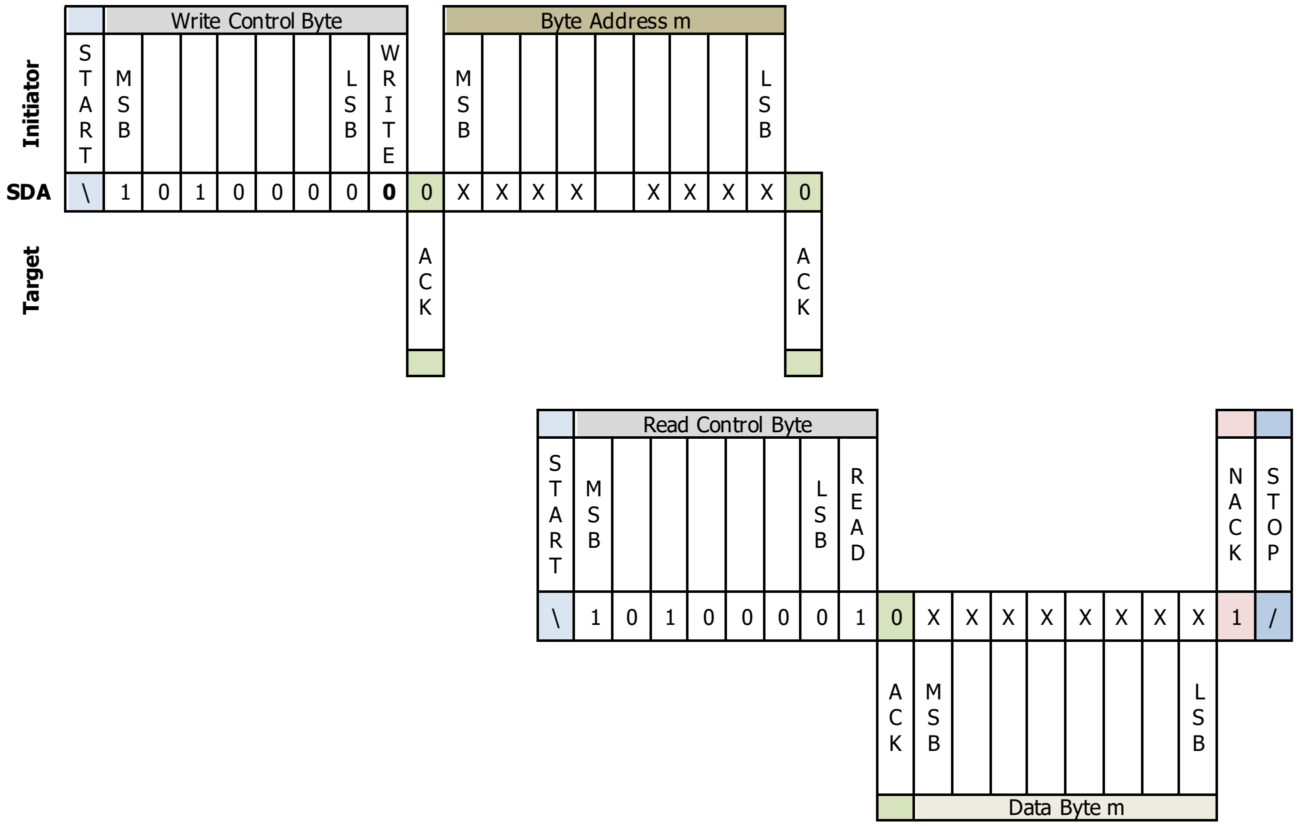


Figure B-2 Random Read

B.2.5.2. Transaction for a Sequential Bytes Read Operation

A sequential bytes read transaction is implemented as a continuation within either a current address read (see Figure B-3) or a random address read (see Figure B-4).

The initiator indicates the continuation of a reading sequence to the target by sending an ACK after a data byte received (instead of terminating the transaction with NACK and STOP).

When the target receives an ACK after sending a byte to the initiator, it increments the current byte address and sends the byte that is stored at this address.

The sequential bytes read is terminated when the initiator sends a NACK and a STOP (instead of ACK).

The target increments the current byte address after orderly termination of the transaction (STOP) and before accepting a new transaction.

B.2.5.2.1. Transaction for a Sequential Bytes Read from Current Start Address

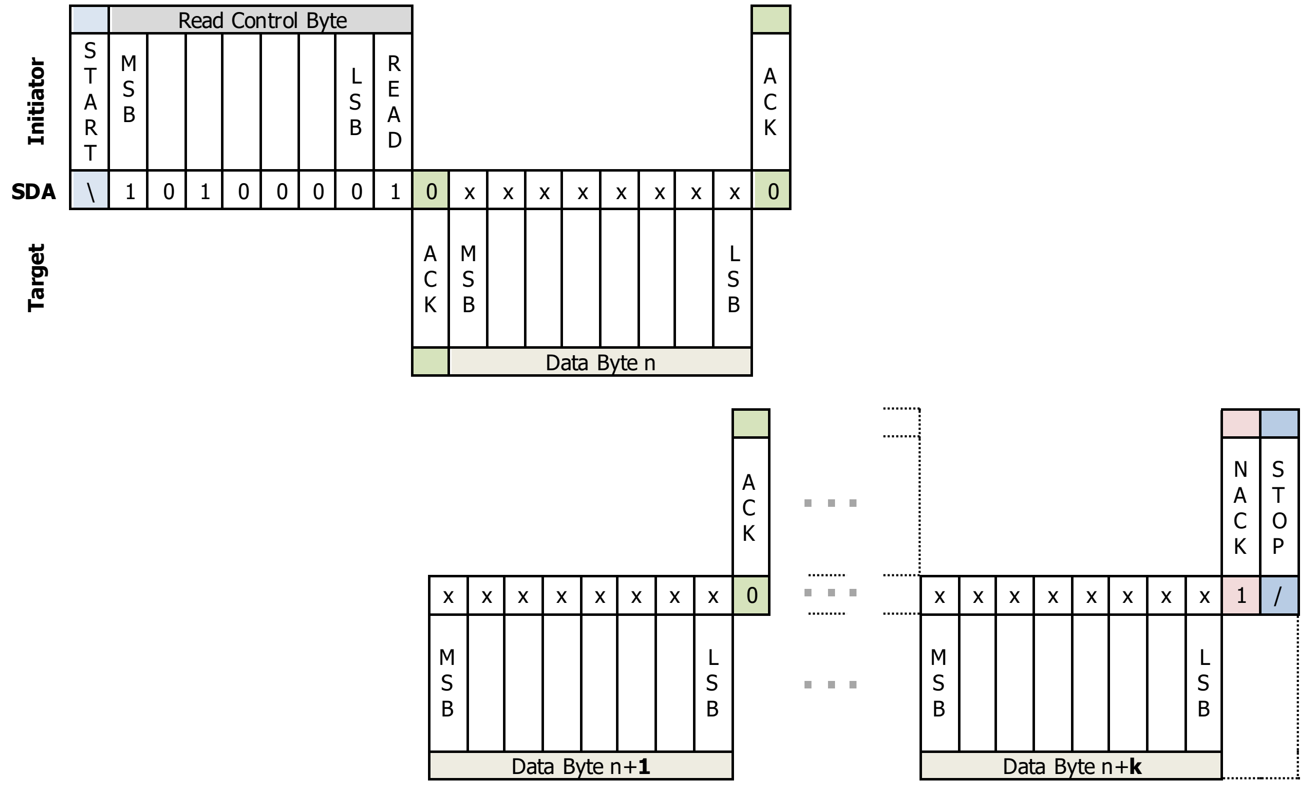
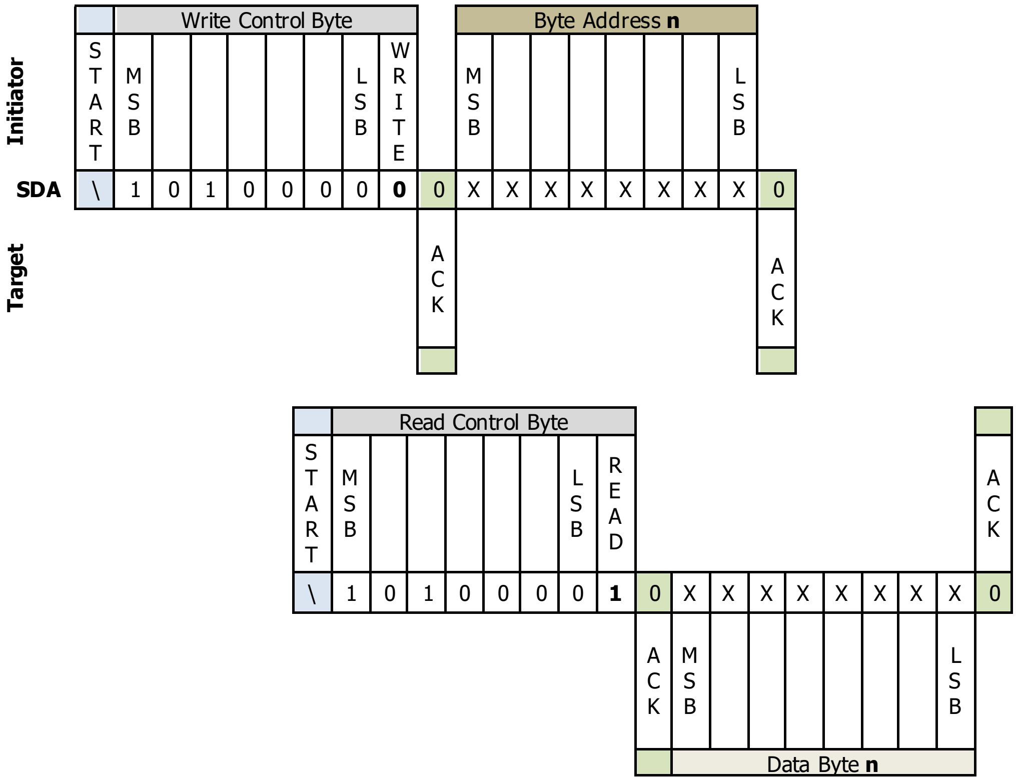


Figure B-3 Sequential Bytes Read Starting at Current Address

B.2.5.2.2. Transaction for a Sequential Bytes Read from Random Start Address



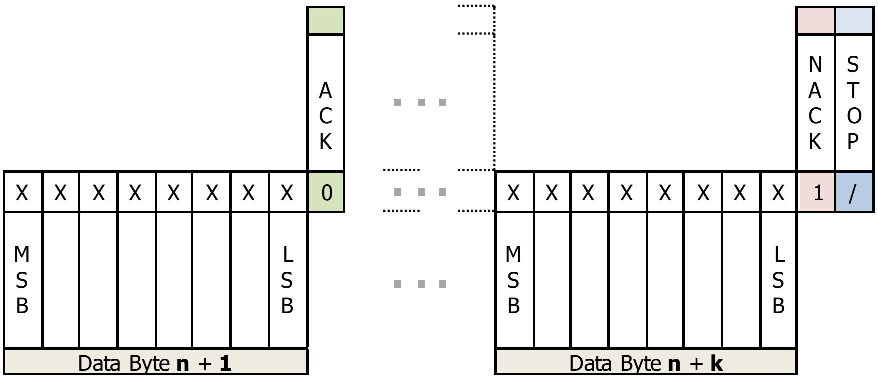


Figure B-4 Sequential Bytes Read Starting with Random Read

B.2.5.3. Transaction for a Single Byte Write Operation

The initiator generates a START condition and sends the write control byte (10100000b).

When the target has responded to the write control byte with ACK, the initiator sends the target byte address.

When the target has responded to the target byte address with ACK, the initiator sends the data byte value.

When the target has responded to the data byte value with ACK, the initiator sends STOP to terminate the write transaction. Otherwise the write transaction is either continued (see section B.2.5.4) or aborted.

On receipt of the target byte address, the target immediately updates its current byte address.

However, the target begins its internal memory write cycle of the received data byte to the address in the current byte address not before the write transaction is properly terminated by STOP.

Note: In a future revision, these register access related specifications may be moved to the appropriate section.

After receiving STOP, the target increments the current byte address before accepting the next transaction.

If a START condition is received in place of a STOP condition the target discards the data byte received and does not increment the current byte address.

For writes to non-volatile memory, upon receipt of the proper STOP condition, the target must enter and complete an internally timed write cycle before the next transaction can be accepted. If needed, the target must hold-off a subsequent transaction for a maximum duration tWR.

For writes to volatile memory, upon receipt of the proper STOP condition, the target must enter and complete an internally timed write cycle before the next transaction can be accepted. If needed, the target must hold-off subsequent transaction for a maximum duration tNACK.

Note: In a future revision, these register access related specifications may be moved to the appropriate section.

Note: To hold-off a subsequent transaction the target may e.g. disable its management interface and not respond or acknowledge subsequent commands until the internal memory write cycle is complete.

Note: the I2C 'Combined Format' using repeated START conditions is not supported on write transactions.

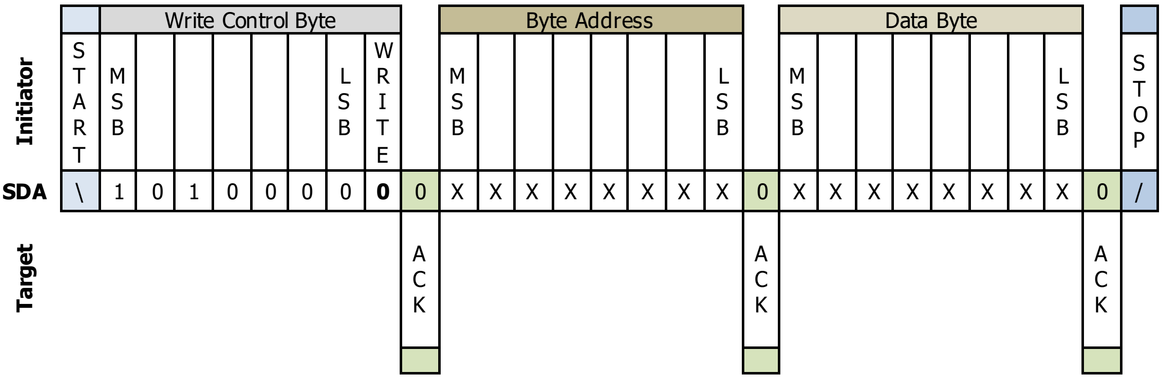


Figure B-5 Write Byte Transaction

B.2.5.4. Transaction for a Sequential Bytes Write Operation

The initiator initiates a sequential write transaction for several bytes in the same way as a single byte write, but it then does not send a STOP condition after the first byte. Instead, after the target acknowledges (ACK) receipt of the first data byte, the initiator transmits the additional data bytes without transmitting new explicit byte address information or control bytes.

The initiator terminates the sequential write sequence with a STOP condition, or otherwise, the transaction is aborted, and the results of the sequential write are undetermined.

The target shall ACK each byte received.

The target may either store a data byte after sending ACK or it may decide to buffer all bytes of the sequential write transaction until the transaction is terminated by STOP.

Note: In a future revision, these register access related specifications may be moved to the appropriate section.

After a properly terminated sequential write transaction (STOP received) the target ensures that the current byte address contains the address of the next byte after the last byte written, before accepting the next transaction. Otherwise, the value of the current byte address is undetermined.

At the end of each 128-byte Page, the current byte address counter rolls over to the first byte of that Page.[1]

For writes to **non-volatile** memory, upon receipt of the proper STOP condition the target must complete its internally timed write cycle before the next basic register access operation can be accepted. If needed, the target must hold-off subsequent transaction for a maximum duration **tWR**, to internal memory.

For writes to **volatile** memory, upon receipt of the proper STOP condition the target must complete its internally timed write cycle before the next basic register access operation can be accepted. If needed, the target must hold-off subsequent transaction for a maximum duration **tNACK**, to internal memory.

Note: In a future revision, these register access related specifications may be moved to the appropriate section.

Note: To hold-off a subsequent transaction the target may e.g. disable its management interface input and not respond or acknowledge subsequent commands until the internal memory write cycle is complete.

Note: the I2C 'combined format' using repeated START conditions is not supported.

[1] There may be exceptions to this basic rule for advanced management features described in chapter 7. Such exceptions are described where those features are specified.

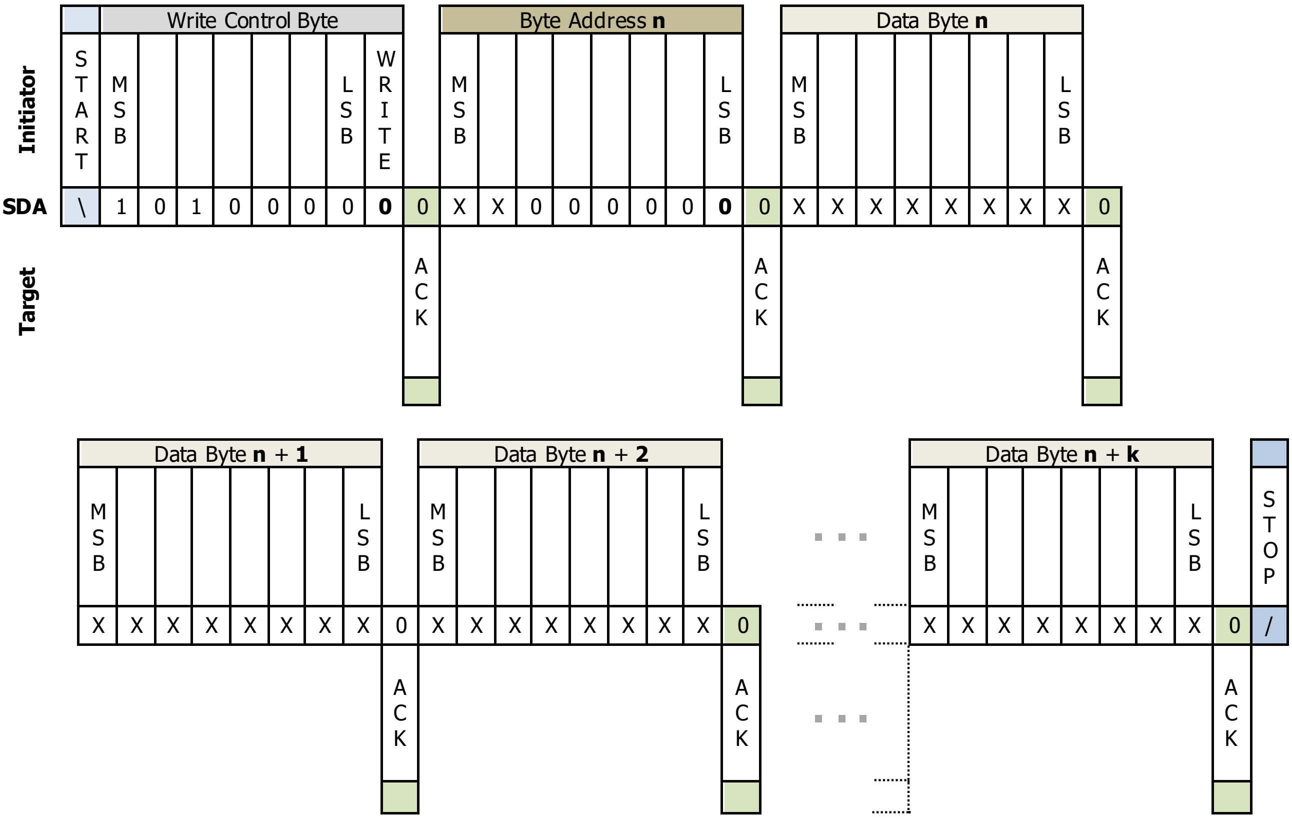


Figure B-6 Sequential Bytes Write Transaction

B.2.5.5. Transaction for a Test Operation

To test if the target is ready to accept a new transaction, the initiator can issue a **dummy** write transaction (START, **write** control byte, STOP) to see of the module ACKs the write control byte.

Note: This method of explicit Acknowledge Polling is the most lightweight method to synchronize to the target becoming ready without side effects on the target.

Note: A dummy read transaction (START, read control byte, STOP) cannot be used for Acknowledge Polling as this may have unintended side effects, when a target interprets this as a current address read transaction (see Section 5.4.3.1) and starts to drive SDA at the same time the initiator is driving SDA for the STOP.

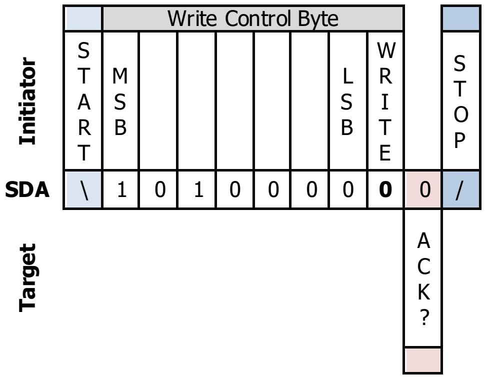


Figure B-7 Test Readiness Transaction

B.2.6 Transaction Flow Control Mechanisms

The MCI offers two flow control mechanisms to the local CMIS register access layer (RAL) instance.

B.2.6.1. Delaying Current Transaction (Clock Stretching)

Clock stretching is a mechanism for the target to delay completion of the current I2CMCI transaction.

The target exercises clock stretching by pulling the clock signal SCL low for a certain duration.

The duration while the target pulls the SCL signal low is the clock stretching duration.

The target is allowed to initiate clock stretching (SCL pull down) only while SCL is low.

The maximum total allowed clock stretching duration during one transaction is specified in section B.2.7.3.

Note: The target may use the clock stretching mechanism whenever needed to implement the register access associated with the transfer, e.g. to fetch a register value in a read transaction or, theoretically, to complete a register update from a preceding write transaction.

B.2.6.2. Rejecting Subsequent Transaction (Transaction Hold-Off)

Transaction hold-off is a temporary phase when new transactions are rejected by signaling NACK (instead of ACK) after having received the control byte of a new transaction (see figure below).

To implement hold-off the target may choose to simply disable its I2C inputs, as this results in the host receiving a NACK after the first byte of a transaction following the START condition.

The maximum allowed transaction hold-off duration is specified in sections B.2.7.3.

Note: The target exercises transaction hold-off when it is unable or not allowed to accept a new transaction in its current state or condition. For instance, after a properly terminated write transaction, the target may need to reject subsequent transactions in order to internally finalize the WRITE operation represented by the previous transaction, due to synchronization requirements specified in section 5.2.4).

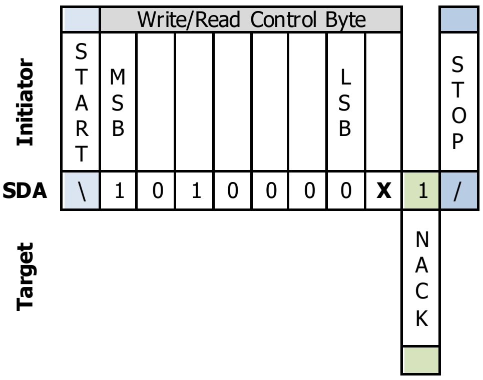


Figure B-8 Test Readiness Transaction

B.2.7 Timing Specifications

Note: The I2CMCI timing specifications must be compatible with the timing specifications of the register access layer specified in chapter 10.

B.2.7.1. Module Select Timings

The following timing parameters apply to host-module interactions over the MCI in form factors implementing the ModSelL module select signal.

Table 10-10 Module Select Timings

Parameter	Symbol	Min	Max	Unit	Conditions
Max Aborted sequence – bus release	Deselect _Abort		2	ms	Delay from a host de-asserting ModSelL (at any point in a bus sequence) to the module releasing SCL and SDA
Min ModSelL Setup Time[1]	tSU_ModSelL	2		ms	ModSelL Setup Time is the setup time on the select line before the start of a host initiated serial bus sequence.
Max ModSelL Hold Time[1]	tHD_ModSelL		500	us	ModSelL Hold Time is the delay from completion of a serial bus sequence to changes of module select status.
Note 1: Support for these setup and hold times are required for all modules. Once the module has initialized the management interface, the host may read advertised ModSelL setup and hold times from the register map.					

B.2.7.2. Waveform Timings

An example I2C waveform is shown in Figure B-9, and the timing parameters are specified in Table B-3 . The default clock rate is a maximum of 400 kHz with an option to support up to a maximum of 1 MHz.

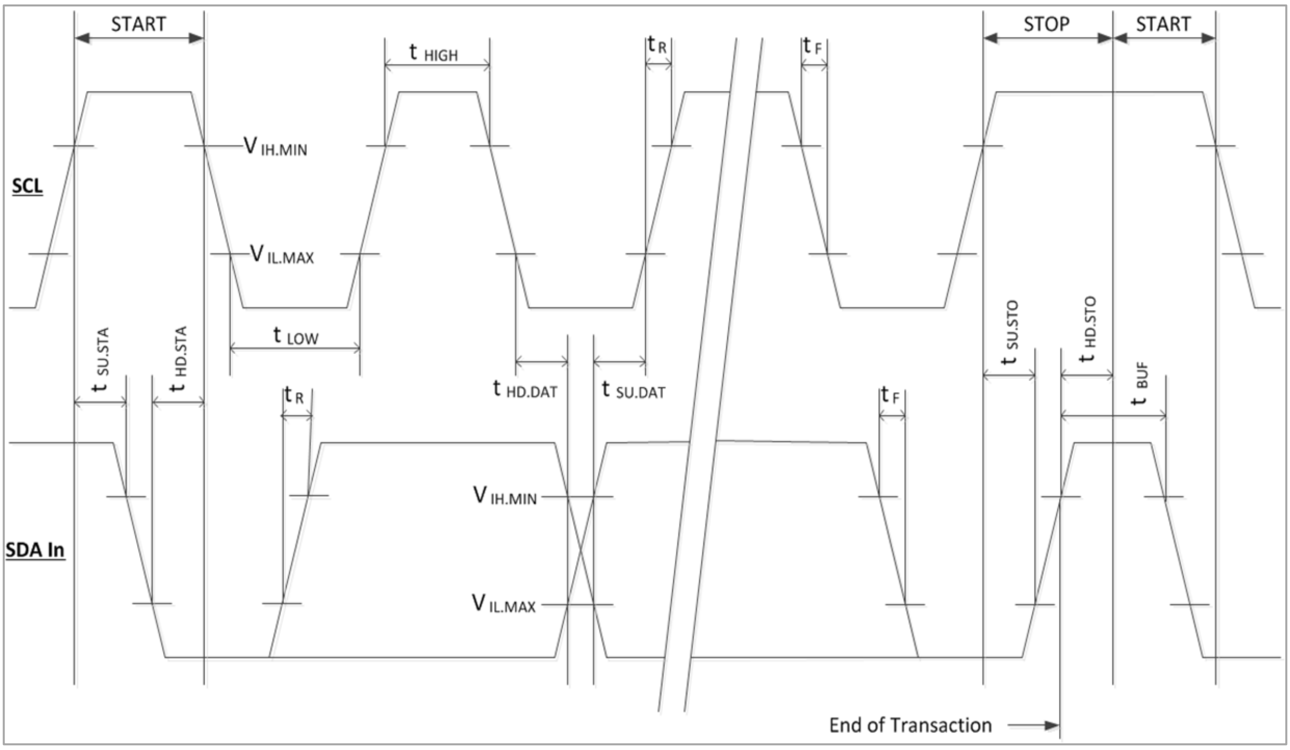


Figure B-9 I2C Waveform Timing Diagram

Table B-3 I2C Waveform Timing Parameters

		0.4 MHz		1 MHz			
Parameter	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Unit**	**Conditions**
Max Clock Frequency	fSCL	0	400	0	1000	kHz	
Min Clock Pulse Width Low	tLOW	1.3		0.50		µs	
Min Clock Pulse Width High	tHIGH	0.6		0.26		µs	
Min START Hold Time	tHD.STA	0.6		0.26		µs	Delay required between SDA becoming low and SCL starting to go low in a START
Min START Setup Time	tSU.STA	0.6		0.26		µs	Delay required between SCL becoming high and SDA starting to go low in a START
Min Data In Hold Time	tHD.DAT	0		0		µs	
Min Data In Setup Time	tSU.DAT	0.1		0.1		µs	
Max Input Rise Time	tR		300		120	ns	From (VIL,MAX=0.3*Vcc) to (VIH, MIN=0.7*Vcc), see Figure B-9.
Max Input Fall Time	tF		300		120	ns	From (VIH,MIN=0.7*Vcc) to (VIL,MAX=0.3*Vcc), see Figure B-9.
Min STOP Setup Time	tSU.STO	0.6		0.26		µs	
Min STOP Hold Time	tHD.STO	0.6		0.26		µs	

B.2.7.3. Transaction Timings

Table B-4 Flow Control Timings

Parameter	Symbol	Min	Max	Unit	Conditions
Max I2CMCI READ Delay (Clock Stretch) (formerly: T_clock_hold)	tRD		500	µs	Maximum time the module may delay returning the requested data in an MCI Read transaction (e.g. I2C clock stretching)
Max time to complete a standard WRITE to volatile memory or register	tNACK		10	ms	Maximum transaction hold-off time after a write transaction to an address representing volatile memory or a volatile management register
Max time to complete a standard WRITE access to non-volatile memory or register	tWR		80	ms	Maximum transaction hold-off time after a write transaction to an address representing non-volatile memory or a non-volatile management register
Min inter-transaction idle time	tBUF	20		µs	Min time bus free before new transaction can start, between STOP and START and between ACK and START

B.2.7.3.1. tWR Timing

The timing attribute tWR is the maximum time allowed for a module to complete its internally timed write cycle after a single or sequential write to non-volatile memory before the next basic management operation can be accepted.

The write cycle completion time is measured from the low to high SDA edge of the STOP condition of the Write transaction to the high to low SDA edge of the START condition for the next transaction.

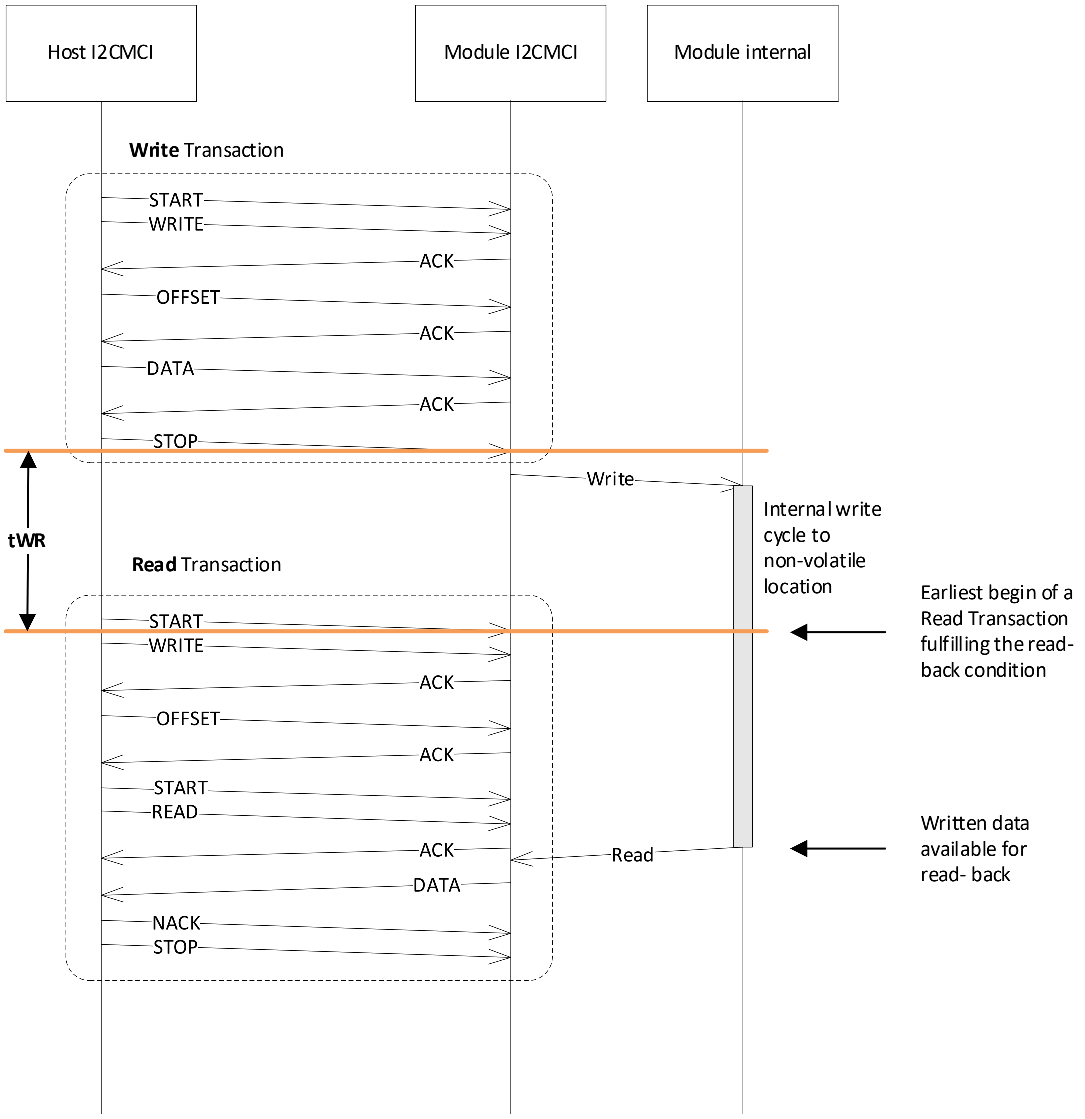


Figure B-10 tWR bus timing

B.2.7.3.2. tNACK Timing

The timing attribute tNACK is the maximum time allowed for a module to complete its internally timed write cycle after a single or sequential write to a volatile memory location before the next basic management operation can be accepted.

The write cycle completion time is measured from the low to high SDA edge of the STOP condition of the Write transaction to the high to low SDA edge of the START condition for the next transaction.

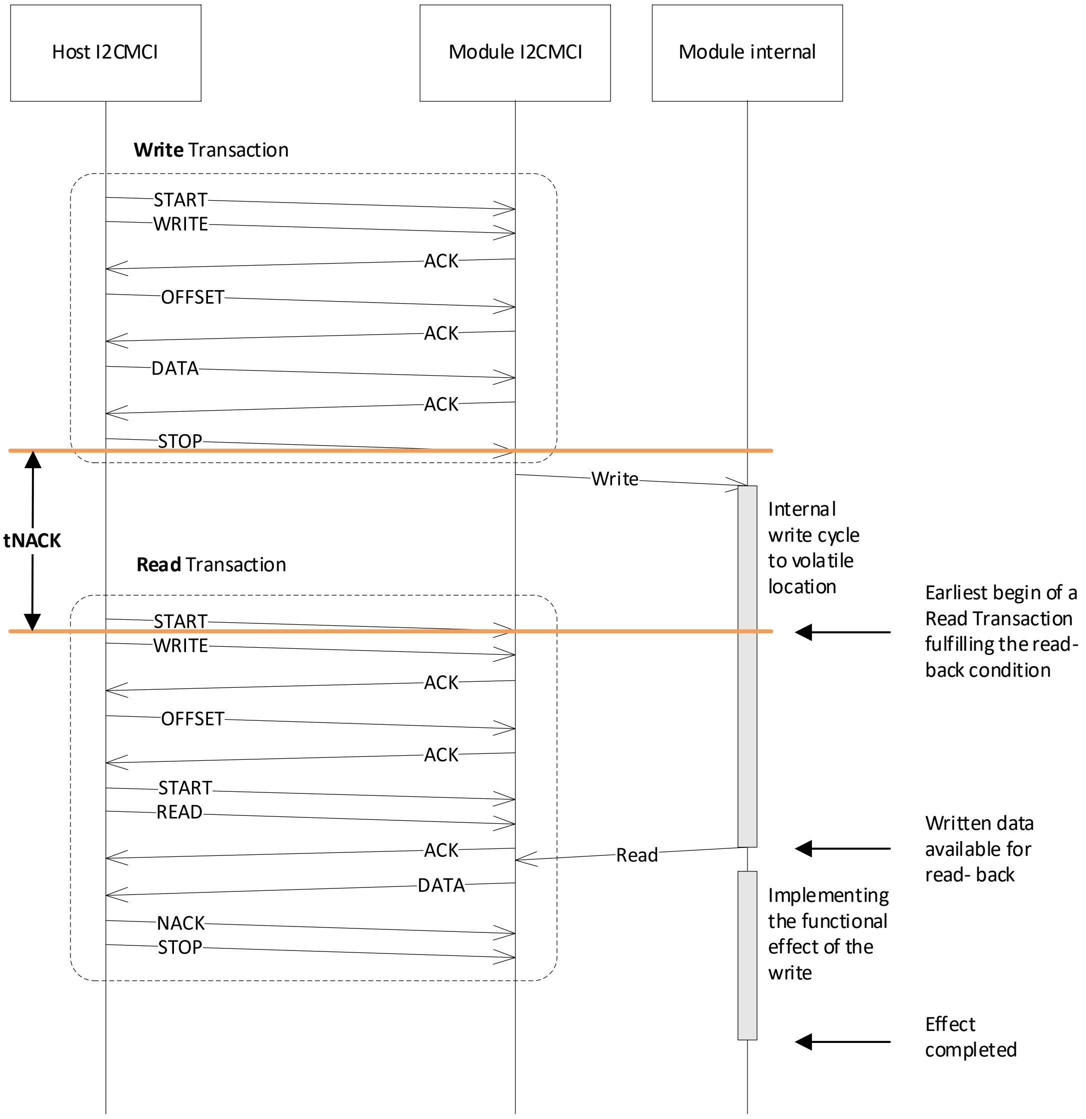

Host I2CMCI
Module I2CMCI
Module internal
Write Transaction
START
WRITE
ACK
OFFSET
ACK
DATA
ACK
STOP
Write
tNACK
Internal write cycle to volatile location
Read Transaction
START
WRITE
ACK
OFFSET
ACK
START
READ
ACK
Read
DATA
NACK
STOP
Earliest begin of a Read Transaction fulfilling the read-back condition
Written data available for read- back
Implementing the functional effect of the write
Effect completed

Figure B-11 tNACK bus timing

Appendix C Examples of Application Advertisements

This appendix presents Application advertisement examples to illustrate how fixed or programmable modules will advertise the Applications they support (as described in chapter 6). The following cases are described:

- Programmable 400GBASE-DR4 module that can also operate as 4x100GBASE-DR
- Fixed 400BASE-SR8 module
- Programmable 400G-SR8 module that can also operate as a 2x200G, 4x100G, or 8x50G breakout.
- Dual Use Active Optical Cable for 1x400G or 2x200G Applications

The first example in Table C-1 400GBASE-DR4 Transceiver with Dual Application Advertising illustrates the Application advertisement of a module that operates as a 400GBASE-DR4 transceiver but supports alternative operation as four integrated parallel 100GBASE-DR transceivers.

With this example, the host can select Application 1 or Application 2 (by programming the AppSel value AppSel=1 or AppSel=2, respectively) in one of the Staged Control Sets. AppSel=1 (400GAUI-8 to 400GBASE-DR4) is the default Application populated in the Active Control Set at power-on.

Table C-1 400GBASE-DR4 Transceiver with Dual Application Advertising

Byte	Bits	AppSel Code	Name	Value	Description
85	7-0	N/A	Module Type encoding	02h	Optical Interfaces: **SMF**
86	7-0	0001b	HostInterfaceID	11h	400GAUI-8 C2M
87	7-0		MediaInterfaceID	1Ch	**400GBASE-DR4**
88	7-4		HostLaneCount	8	8 host lanes
	3-0		MediaLaneCount	4	4 media lanes
89	7-0		HostLaneAssignmentOptions	01h	Permissible first host lane number for Application: lane 1
01h:176	7-0		MediaLaneAssignmentOptions	01h	Permissible first media lane number for Application: lane 1
90	7-0	0010b	HostInterfaceID	0Dh	100GAUI-2 C2M
91	7-0		MediaInterfaceID	14h	**100GBASE-DR**
92	7-4		HostLaneCount	2	2 host lanes
	3-0		MediaLaneCount	1	1 media lane
93	7-0		HostLaneAssignmentOptions	55h	Permissible first host lane number for Application: lanes 1, 3, 5, and 7
01h:177	7-0		MediaLaneAssignmentOptions	0Fh	Permissible first media lane number for Application: lanes 1, 2, 3, 4
94	7-0	0011b	HostInterfaceID	FFh	End of list of supported Applications
95	7-0		MediaInterfaceID	00h	
96	7-4		HostLaneCount	0	
	3-0		MediaLaneCount	0	
97	7-0		HostLaneAssignmentOptions	00h	
01h:178	7-0		MediaLaneAssignmentOptions	00h	

The second example in Table C-2 400GBASE-SR8 Fixed Transceiver Application Advertising illustrates the Application advertisement for a fixed 400GBASE-SR8 transceiver that does not support other Applications.

With this example, the host can set only AppSel=1 in one of the Staged Control Sets

Table C-2 400GBASE-SR8 Fixed Transceiver Application Advertising

Byte	Bits	AppSel Code	Name	Value	Description
85	7-0	N/A	Module Type encoding	01h	Optical Interfaces: **MMF**
86	7-0	0001b	HostInterfaceID	11h	400GAUI-8 C2M
87	7-0		MediaInterfaceID	10h	**400GBASE-SR8**
88	7-4		HostLaneCount	8	8 host lanes
	3-0		MediaLaneCount	8	8 media lanes
89	7-0		HostLaneAssignmentOptions	01h	Permissible first host lane number for Application: lane 1
01h:176	7-0		MediaLaneAssignmentOptions	01h	Permissible first media lane number for Application: lane 1
90	7-0	0010b	HostInterfaceID	FFh	End of list of supported Applications
91	7-0		MediaInterfaceID	00h	
92	7-4		HostLaneCount	0	
	3-0		MediaLaneCount	0	
93	7-0		HostLaneAssignmentOptions	00h	
01h:177	7-0		MediaLaneAssignmentOptions	00h	

The third example in Table C-3 400GBASE-SR8 Transceiver supporting 200GBASE-SR4, 100GBASE-SR2 and 50GBASE-SR illustrates the Application advertisement of a module that operates as a 400GBASE-SR8 transceiver by default but supports breakout to integrated parallel transceivers at a variety of speeds (2x200G, 4x100G, 8x50G).

With this example, the host can write either AppSel 1, 2, 3, or 4 in one of the Staged Control Sets, where AppSel=1 (400GAUI-8 to 400G-SR8) is the default Application populated in the Active Control Set at power-on.

Table C-3 400GBASE-SR8 Transceiver supporting 200GBASE-SR4, 100GBASE-SR2 and 50GBASE-SR

Byte	Bits	AppSel Code	Name	Value	Description
85	7-0	N/A	Module Type encoding	01h	Optical Interfaces: **MMF**
86	7-0	0001b	HostInterfaceID	11h	400GAUI-8 C2M
87	7-0		MediaInterfaceID	10h	**400GBASE-SR8**
88	7-4		HostLaneCount	8	8 host lanes
	3-0		MediaLaneCount	8	8 media lanes
89	7-0		HostLaneAssignmentOptions	01h	Permissible first host lane number for Application: lane 1
01h:176	7-0		MediaLaneAssignmentOptions	01h	Permissible first media lane number for Application: lane 1
90	7-0	0010b	HostInterfaceID	0Fh	200GAUI-4 C2M
91	7-0		MediaInterfaceID	0Eh	**200GBASE-SR4**
92	7-4		HostLaneCount	4	4 host lanes
	3-0		MediaLaneCount	4	4 media lanes
93	7-0		HostLaneAssignmentOptions	11h	Permissible first host lane number for Application: lanes 1 and 5
01h:177	7-0		MediaLaneAssignmentOptions	11h	Permissible first media lane number for Application: lanes 1, 5
94	7-0	0011b	HostInterfaceID	0Dh	100GAUI-2 C2M
95	7-0		MediaInterfaceID	0Ch	**100GBASE-SR2**
96	7-4		HostLaneCount	2	2 host lanes
	3-0		MediaLaneCount	2	2 media lanes
97	7-0		HostLaneAssignmentOptions	55h	Permissible first host lane number for Application: lanes 1, 3, 5, and 7
01h:178	7-0		MediaLaneAssignmentOptions	55h	Permissible first media lane number for Application: lanes 1, 3, 5, 7
98	7-0	0100b	HostInterfaceID	0Ah	50GAUI-1 C2M
99	7-0		MediaInterfaceID	07h	**50GBASE-SR**
100	7-4		HostLaneCount	1	1 host lane
	3-0		MediaLaneCount	1	1 media lane
101	7-0		HostLaneAssignmentOptions	FFh	Permissible first host lane number for Application: lanes 1, 2, 3, 4, 5, 6, 7, 8
01h:179	7-0		MediaLaneAssignmentOptions	FFh	Permissible first media lane number for Application: lanes 1, 2, 3, 4, 5, 6, 7, 8
102	7-0	0101b	HostInterfaceID	FFh	End of list of supported Applications
103	7-0		MediaInterfaceID	00h	
104	7-4		HostLaneCount	0	
	3-0		MediaLaneCount	0	
105	7-0		HostLaneAssignmentOptions	00h	
01h:180	7-0		MediaLaneAssignmentOptions	00h	

The fourth example in Table C-4 8x50G AOC Application Advertising Example illustrates the Application advertisement for an 8x50G Active Optical Cable (AOC) that supports one 8x50G host interface or two parallel 4x50G host interfaces.

With this example, the host could write either AppSel=1 or AppSel=2 in one of the Staged Control Sets. AppSel=1 (400GAUI-8) is the default Application populated in the Active Control Set at power-on.

Table C-4 8x50G AOC Application Advertising Example

Byte	Bits	AppSel Code	Name	Value	Description
85	7-0	N/A	Module Type encoding	04h	**Active cables**
86	7-0	0001b	HostInterfaceID	11h	400GAUI-8 C2M
87	7-0		MediaInterfaceID	03h	**AOC with BER < 2.4e-4**
88	7-4		HostLaneCount	8	8 host lanes
	3-0		MediaLaneCount	8	**8 media lanes**
89	7-0		HostLaneAssignmentOptions	01h	Permissible first host lane number for Application: lane 1
01h:176	7-0		MediaLaneAssignmentOptions	01h	Permissible first media lane number for Application: lane 1
90	7-0	0010b	HostInterfaceID	0Fh	200GAUI-4 C2M
91	7-0		MediaInterfaceID	03h	**AOC with BER < 2.4e-4**
92	7-4		HostLaneCount	4	4 host lanes
	3-0		MediaLaneCount	4	**4 media lanes**
93	7-0		HostLaneAssignmentOptions	11h	Permissible first host lane number for Application: lanes 1 and 5
01h:177	7-0		MediaLaneAssignmentOptions	11h	Permissible first media lane number for Application: lanes 1, 5
94	7-0	0011b	HostInterfaceID	FFh	End of list of supported Applications
95	7-0		MediaInterfaceID	00h	
96	7-4		HostLaneCount	0	
	3-0		MediaLaneCount	0	
97	7-0		HostLaneAssignmentOptions	00h	
01h:178	7-0		MediaLaneAssignmentOptions	00h	

Appendix D Examples of Initialization and Deinitialization

This appendix includes example scenarios illustrating possible sequences of events and interactions between host and module during module and Data Path initialization or deinitialization.

Refer to section 6.3.2 for information on the Module State Machine and to section 6.3.3 for information on Data Path State Machines, Applications, and Control Sets.

The following example scenarios illustrate how a host could power up and initialize, or deinitialize and power down a module:

Scenario	Description	Section
Quick hardware initialization	Power up and initialization without host software interaction	0
Quick software initialization	Power up and initialization with minimal host software interaction	D.1.2
Software configuration and initialization	Power up and initialization with module configuration by host	D.1.3
Hardware deinitialization	Power-down sequence using hardware control	D.2.1
Software deinitialization	Power-down sequence using software control	D.2.2

Note: For simplicity, the examples may deliberately omit minor behavioral variations depending on module advertisement or configuration (for instance it is assumed that all state change notifications are flagged)

D.1 Initialization Examples

The power-up initialization behavior of pausing in the ModuleLowPwr state (for host software control), or of passing to full operation through the ModuleLowPwr state (without host control) is determined by the settings LowPwrRequestSW and LowPwrAllowRequestHW and by the LowPwrRequestHW hardware signal (Table 6-12).

The default register settings LowPwrAllowRequestHW=1 and LowPwrRequestHW=0 (see Table 8-10) require the LowPwrRequestHW hardware signal to be ASSERTED when host software controlled start-up is desired.

D.1.1 Quick Hardware Initialization

This section describes an example of a simple module power-up sequence where the module fully powers up under hardware control, without host software intervention.

This scenario has the following characteristics:

a. Module is powered-up from an un-powered state (LowPwrAllowRequestHW=1 and LowPwrRequestSW=0)
b. Module is fully powered-up under hardware control (LowPwrRequestHW=**DEASSERTED**)
c. Host uses the default settings for the selected Application.
d. Host does not use custom signal integrity settings (i.e. ExplicitControl indicator)
e. Host does not perform speed negotiation

#	Host Action	Module Action	Module State (M) Data Path State (D)
0	Host applies Vcc, deasserts LowPwrRequestHW, asserts ModSel (if supported), and deasserts Reset. Host ensures that host transmitters are configured and enabled to output a valid signal for the default Application in the module, prior to step 1 below.		
1	Hot Plug		

2	Host detects module presence, waits for Interrupt assertion	Module powers up and initializes management interface, sets power on defaults, such as LowPwrRequestSW=0, LowPwrAllowRequestHW=1, DPDeinit=00h, OutputDisableTx = 0 and OutputSquelchForceTx = 0, and writes the power on default Data Path configurations into the Active Control Set and Staged Control Set 0	M=MgmtInit
			D=DPDeactivated

#	Host Action	Module Action	Module State (M) Data Path State (D)

3		Module sees LowPwrS transition signal is FALSE on entry into ModuleLowPwr and transitions to ModulePwrUp	M=ModuleLowPwr
			D=DPDeactivated

4		Module powers up to High Power Mode	M=ModulePwrUp
			D=DPDeactivated

5		Module sets ModuleStateChangedFlag on entry into ModuleReady	M=ModuleReady
			D=DPDeactivated

6	Host detects assertion of Interrupt and reads all Flag registers, which deasserts Interrupt	Module sees DPDeinitS transition signal is FALSE and transitions all Data Path states to DPInit	

7	Host waits for second Interrupt assertion to indicate completion of Data Path initialization	Module initializes all Data Paths according to the configuration in the Active Control Set.	M=ModuleReady
			D=DPInit

8		Module sees DPDeactivateS transition signal is FALSE and transitions all Data Path states to DPTxTurnOn	M=ModuleReady
			D=DPInitialized

9		Modules enables all Tx outputs	M=ModuleReady
			D=DPTxTurnOn

10	←	Module sets the Data Path State Changed Flags to 1 on entry into DPActivated	M=ModuleReady
			D=DPActivated

11	Host detects assertion of Interrupt and reads all Flag registers, which deasserts Interrupt	Module waits for host action	
12	Host uses the activated Data Path to carry live traffic		

D.1.2 Quick Software Initialization

This section describes an example of a simple module power-up sequence where the module gets under host software control in ModuleLowPwr, such that the host can validate module capabilities and power dissipation before letting the module complete its initialization, using the default Application and Data Path configuration.

This scenario has the following characteristics:

a. Module is powered-up from an un-powered state (LowPwrAllowRequestHW=1 and LowPwrRequestSW=0)
b. Module function is initialized under software control (LowPwrRequestHW=**ASSERTED**)
c. Host uses the default settings for the selected Application.
d. Host does not use custom signal integrity settings (i.e. ExplicitControl indicator)
e. Host does not perform speed negotiation

#	Host Action	Module Action	Module State (M) Data Path State (D)
0	Host applies Vcc, asserts LowPwrRequestHW and ModSel (if supported), and deasserts Reset		
1	Hot Plug		

2	Host detects module presence, waits for Interrupt assertion	Module powers up and initializes management interface, sets power on defaults, such as LowPwrRequestSW=0, LowPwrAllowRequestHW=1, DPDeinit=00h, OutputDisableTx = 0 and OutputSquelchForceTx = 0, and writes the power on default Data Path configurations into the Active Control Set and Staged Control Set 0	M=MgmtInit
			D=DPDeactivated

#	Host Action		Module Action	Module State (M) Data Path State (D)

3		←	Module sets ModuleStateChangedFlag on entry into ModuleLowPwr	M=ModuleLowPwr
				D=DPDeactivated

4	Host detects assertion of Interrupt and reads the Flag registers, which deasserts Interrupt		Module waits for host action	
5	Host reads module power requirements and Data Path configuration information			
6	Host configures and enables host transmitters such that they output a valid signal for the default Application towards the module			
7	Host clears LowPwrAllowRequestHW[1] bit, which effectively initiates a module transition to High Power Mode (ModuleReady)	→		
8	Host waits for Interrupt assertion to indicate completion of transition to High Power Mode		Module sees LowPwrS transition signal become FALSE and transitions to ModulePwrUp	

9			Module powers up to High Power Mode	M=ModulePwrUp
				D=DPDeactivated

10			Module sets ModuleStateChangedFlag on entry into ModuleReady	M=ModuleReady
				D=DPDeactivated

11	Host detects assertion of Interrupt and reads all Flag registers, which deasserts Interrupt		Module sees DPDeinitS transition signal is FALSE and transitions all Data Path states to DPInit	

12	Host waits for second Interrupt assertion to indicate completion of Data Path initialization		Module initializes all Data Paths according to the configuration in the Active Control Set.	M=ModuleReady
				D=DPInit

13			Module sees DPDeactivateS transition signal is FALSE and transitions all Data Path states to DPTxTurnOn	M=ModuleReady
				D=DPInitialized

14			Modules enables all Tx outputs	M=ModuleReady
				D=DPTxTurnOn

15		←	Module sets the DPStateChangedFlag bits to 1 on entry into DPActivated	M=ModuleReady
				D=DPActivated

16	Host detects assertion of Interrupt and reads all Flag registers, which deasserts Interrupt		Module waits for host action	
17	Host uses the activated Data Path to carry live traffic			

D.1.3 Software Configuration and Initialization

This section describes an example of a simple module power-up sequence where the module gets under host software control in ModuleLowPwr state, such that the host can validate module capabilities and its maximum power dissipation and subsequently configure the module function, before letting the configured module complete its functional initialization. This method may be the most common initialization method used for Ethernet applications.

In this example, host software powers up the module and the Data Paths in two separate steps.

This scenario has the following characteristics:

a. Module is powered-up from an un-powered state (LowPwrAllowRequestHW=1 and LowPwrRequestSW=0)

[1] To allow full SW control from now on, the host disables the LowPwrRequestHW signal. Alternatively the host could also deassert LowPwrRequestHW signal.

b. Module function is initialized under host software control (LowPwrRequestHW=**ASSERTED**)
c. Host selects one of the Applications advertised by the module
d. Host initializes one instance of this Application (i.e. one Data Path).
e. Host uses the default settings for the selected Application.
f. Host does not use custom signal integrity settings (i.e. ExplicitControl indicator is 0)
g. Host uses only staged Control Set 0 to configure the module (optional Staged Control Set 1 is not used).
h. Host does not perform speed negotiation

Steps 1-11 provide an example of how to power-up a module into the ModuleReady state, starting from an unpowered state. Steps 12-28 provide an example for the initialization and activation of a single Data Path within a module (these steps can also be used for the initialization and activation of subsequent Data Paths, e.g. in the case of breakout, etc.).

#	Host Action		Module Action	Module State (M) Data Path State (D)
0	Host applies Vcc, asserts LowPwrRequestHW and ModSel (if supported), and deasserts Reset			
1	Hot Plug			

2	Host detects module presence, waits for Interrupt assertion		Module powers up and initializes management interface, sets power on defaults, such as LowPwrRequestSW=0, LowPwrAllowRequestHW=1, DPDeinit=00h, OutputDisableTx = 0 and OutputSquelchForceTx = 0, and writes the power on default Data Path configurations into the Active Control Set and Staged Control Set 0.	M=MgmtInit
				D=DPDeactivated

3		←	Module sets ModuleStateChangedFlag on entry into ModuleLowPwr	M=ModuleLowPwr
				D=DPDeactivated

4	Host detects assertion of Interrupt and reads all Flag registers, which deasserts Interrupt		Module waits for host action	
5	Host reads module power requirements			
6	Host writes FFh to the DPDeinit register to prevent automatic Data Path initialization when the module reaches ModuleReady, and writes FFh into the OutputDisableTx register to prevent automatic Data Path activation when the Data Path state reaches DPInitialized.			
7	Host clears LowPwrAllowRequestHW[1] bit, which effectively initiates a module transition to High Power Mode (ModuleReady)	→		
8	Host waits for Interrupt assertion to indicate completion of transition to High Power Mode		Module sees LowPwrS transition signal become FALSE and transitions to ModulePwrUp	

9			Module powers up to High Power Mode	M=ModulePwrUp
				D=DPDeactivated

10		←	Module sets ModuleStateChangedFlag on entry into ModuleReady	M=ModuleReady
				D=DPDeactivated

[1] To allow full SW control from now on, the host disables the LowPwrRequestHW signal. Alternatively the host could also deassert LowPwrRequestHW signal.

#	Host Action		Module Action	Module State (M) Data Path State (D)
11	Host detects assertion of Interrupt and reads all Flag registers, which deasserts Interrupt		Module waits for host action	
12	Host reads Application advertising registers			
13	Host writes desired AppSel code into applicable Data Path Configuration Control registers in Staged Control Set 0			
14	Host configures and enables host transmitters such that they output a valid signal for the selected Application towards the module			
15	Host writes a 1 to the bits in the ApplyDPInit register in Staged Control Set 0, corresponding to host lanes of the Data Path the host wants to configure. The host lanes that are not part of the Data Path are masked (bits cleared) in the register write. The host performs this step with a single write to the ApplyDPInit register.	→		
16	Host waits for module action	←	Module validates the configuration requested in Staged Control Set 0. If the configuration was found to be valid, the module copies the contents to the Active Control Set, updates DPInitPending, and finally updates the ConfigStatus fields.	
17	Host reads the ConfigStatus fields to confirm that the requested configuration was validated and accepted by the module for all lanes of the selected Data Path.		Module waits for host action	
18	Host requests initialization of the newly configured Data Path by clearing the DPDeinit bits representing the host lanes of the Data Path being initialized. The host preserves the existing bit values for the other lanes in the DPDeinit register (using read-modify-write). The new contents of the DPDeinit register are written by the host in a single write transaction.	→		
19	Host waits for Interrupt assertion to indicate completion of Data Path initialization		Module sees DPDeinitS transition signal is FALSE and transitions the Data Path state to DPInit	

20			Module initializes the Data Path and clears its DPInitPending bits	M=ModuleReady
				D=DPInit

21		←	Once the Data Path is initialized the module sets the DPStateChangedFlag bits, on entry into DPInitialized	M=ModuleReady
				D=DPInitialized

22	Host detects assertion of Interrupt and reads all Flags to clear the interrupt		Module waits for host action	

#	Host Action		Module Action	Module State (M) Data Path State (D)
23	Host requests activation of the Data Path by clearing the bits representing its media lanes in the OutputDisableTx register. The host preserves the existing bit values for the other media lanes in the OutputDisableTx register (using read-modify-write). Host writes the new contents of the OutputDisableTx register in a single write transaction.	→		
24	Host waits for Interrupt assertion to indicate completion of Data Path activation		Module sees DPDeactivateS transition signal is FALSE and transitions the Data Path state to DPTxTurnOn.	

25			Modules enables all Tx outputs for the Data Path being activated	M=ModuleReady
				D=DPTxTurnOn

26		←	Once the Data Path is activated the module sets the DPStateChangedFlag bits, on entry into DPActivated	M=ModuleReady
				D=DPActivated

27	Host detects assertion of Interrupt and reads all Flag registers, which deasserts Interrupt.		Module waits for host action	
28	Host uses the activated Data Path to carry live traffic			

D.2 Deinitialization Examples

D.2.1 Hardware Deinitialization

This section describes an example of a simple module power-down sequence where the module powers down under hardware control (LowPwrRequestHW hardware signal transitions from DEASSERTED to ASSERTED) without software interaction.

This scenario has the following characteristics:
a. Module was previously powered up under hardware control (LowPwrRequestHW=DEASSERTED)
b. At least one Data Path is in the DPActivated state

#	Host Action		Module Action	Module State (M) Data Path State (D)

0	Module is powered up with at least one Data Path activated.
	LowPwrRequestHW = DEASSERTED,
	Reset = DEASSERTED,
	ModSel=ASSERTED (if supported).
			Initial conditions: Module fully configured and powered, and with at least one Data Path in the DPActivated state.	M=ModuleReady
				D=DPActivated

1	Host asserts the LowPwrRequestHW signal	→	Module sees DPDeactivateS transition signal is TRUE and transitions all Data Path states to DPTxTurnOff	

2			All Data Path states transition to DPInitialized without any further host action.	M=ModuleReady
				D=DPTxTurnOff

3			Module sees DPReDeinitS is TRUE and transitions all Data Path states to DPDeinit	M=ModuleReady
				D=DPInitialized

4			Module disables all Tx outputs and deinitializes Data Path resources	M=ModuleReady
				D=DPDeinit

5			Module sets the Data Path State Changed Flags to 1 on entry into DPDeactivated	M=ModuleReady
				D=DPDeactivated

6	Host detects assertion of Interrupt and reads all Flag registers, which deasserts Interrupt		Module sees LowPwrExS is TRUE and transitions the module state to ModulePwrDn	

7	Host waits for second Interrupt assertion to indicate completion of module power down		Module reduces the module power to low power mode levels	M=ModulePwrDn
				D=DPDeactivated

8		←	Module sets ModuleStateChangedFlag on entry into ModuleLowPwr	M=ModuleLowPwr
				D=DPDeactivated

9	Host detects assertion of Interrupt and reads all Flags to clear the interrupt		Module waits for host action	

D.2.2 Software Deinitialization

This section describes an example of a simple module power-down sequence where the module powers down under software control (LowPwrRequestHW hardware signal = ASSERTED). This example uses a two-step power down process where all Data Paths are first deactivated and then the module is powered down using the LowPwrAllowRequestHW bit. One scenario where this might be used is where a host has deactivated all Data Paths, but wants to go one step further and also transition the module to low power mode. Note that this approach does require a two-step power up process if the host wants to reactivate one of the Data Paths at a later point in time, where the host has to first transition the module to high power mode before activating the Data Path.

This scenario has the following characteristics:

a. Module was previously powered up under software control (LowPwrRequestHW=ASSERTED)

b. At least one Data Path is in the DPActivated state

#	Host Action		Module Action	Module State (M) Data Path State (D)

0	Module is powered up with at least one Data Path activated.
	LowPwrRequestHW = ASSERTED,
	ResetL = DEASSERTED,
	ModSel=ASSERTED (if supported).
			Initial condition: Module fully configured and powered, and with at least one Data Path in the DPActivated state.	M=ModuleReady
				D=DPActivated

1	The host sets the DPDeinit bits for all host lanes in the applicable Data Path to 1.	→	Module sees DPDeactivateS transition signal is TRUE and transitions the Data Path state to DPTxTurnOff	

2			All Data Path states transition to DPInitialized without any further host action.	M=ModuleReady
				D=DPTxTurnOff

3			Module sees DPReDeinitS is TRUE and transitions all Data Path states to DPDeinit	M=ModuleReady
				D=DPInitialized

4			Module disables the Tx outputs and deinitializes Data Path resources for all Data Paths.	M=ModuleReady
				D=DPDeinit

5		←	Module sets the Data Path State Changed Flags to 1 on entry into DPDeactivated	M=ModuleReady
				D=DPDeactivated

6	Host detects assertion of Interrupt and reads all Flag registers, which deasserts Interrupt		Module waits for host action	
7	Host sets the LowPwrAllowRequestHW bit to 1	→	Module sees LowPwrExS is TRUE and transitions the module state to ModulePwrDn	

8	Host waits for Interrupt assertion to indicate completion of module power down		Module reduces the module power to low power mode levels	M=ModulePwrDn
				D=DPDeactivated

9		←	Module sets ModuleStateChangedFlag on entry into ModuleLowPwr	M=ModuleLowPwr
				D=DPDeactivated

10	Host detects assertion of Interrupt and reads all Flags to clear the interrupt		Module waits for host action	

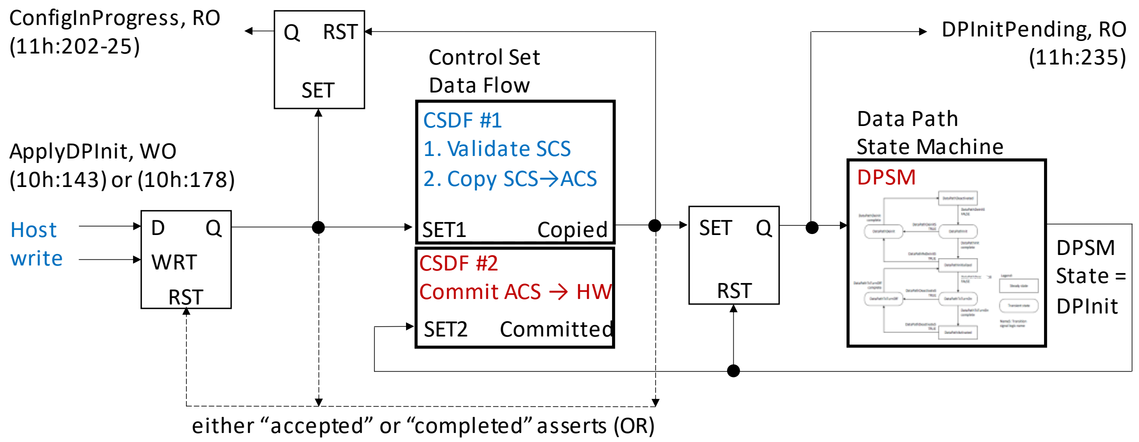
Appendix E Illustration of Applying Control Sets

This appendix illustrates different module behaviors after a host writing to one of the Apply register, depending on the advertisement SteppedConfigOnly (00h:2.6)

E.1 Default Behavior (SteppedConfigOnly = 0)

CMIS 5.0 Default ApplyDPInit Processing

SteppedConfigOnly = **0**, RO (00h:2.6) – DPInitPending affects DPSM and may initiate automatic commissioning



Not shown: Module must ignore host write to ApplyDPInit while ConfigStatus = ConfigInProgress
Module is free to use ApplyDPInit and to clear on command completion or on acceptance
Module ignores writes in transient DPSM states

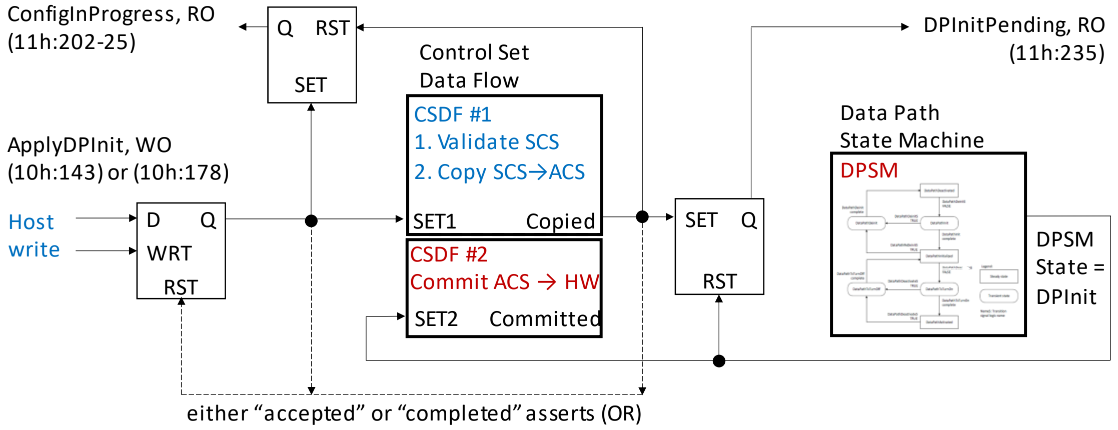
Figure E-12 ApplyDPInit (default: SteppedConfigOnly=0)

Note that ApplyImmediate with state dependent behavior is supported in steady DPSM states as per Table 6-3

E.2 Restricted Behavior (SteppedConfigOnly = 1)

CMIS 5.0 Restricted ApplyDPInit Processing

SteppedConfigOnly = **1**, RO (00h:2.6) – DPInitPending is pure provisioning and does **not** affect DPSM



Not shown: Module must ignore host write to ApplyDPInit while ConfigStatus = ConfigInProgress
Module is free to use ApplyDPInit and to clear on command completion or on acceptance

Figure E-13 ApplyDPInit (restricted: SteppedConfigOnly=1)

Note that ApplyImmediate is not supported in this configuration.

Appendix F Examples of Diagnostic Features Usage

This Appendix contains usage recommendations and examples for the diagnostic features available on Pages 13h and 14h.

Note: For use of these diagnostic features the module should be in the ModuleReady state. An Application should be selected but otherwise no specific relation to Data Path states is defined for these diagnostic features.

F.1 Enabling and Disabling Pattern Generator (Host or Media Side)

The following procedure contains a recommended sequence of register accesses to set the module into Host side (or Media side) pattern generation mode. Host side registers are used in the examples, with media side registers shown in parentheses.

1. Write Bank Select and Page Select registers 00h:127-128 to select Page 13h on the appropriate Bank
2. Configure desired mode of operation in Byte 13h:177
3. Configure desired pattern generator configuration and lane pattern in Bytes 13h:145-151 (13h:153-159)
4. Configure the desired control options in Bytes 13h:176-179
5. Enable the pattern generator on the selected lanes by writing to 13h:144.7-0 (13h:152:7-0)

After the above sequence of commands, the host side electrical (media side electrical or optical) output will be generating the selected pattern for the enabled lanes.

Note: The pattern generation feature may vary by module. On some modules, when in pattern generation mode, per lane control is not provided and all lanes may be generating the pattern. On other modules, such as modules design to support break-out, the selected lanes may be in pattern generation mode while other lanes are either disabled or in normal mission mode.

To disable the pattern generator on selected host side (media side) lanes:

6. Write to 13h:144.7-0 (13h:152.7-0) with the relevant bits representing the selected lanes cleared

When pattern generation is disabled on selected lanes, those lanes are expected to revert to mission mode if possible. In some modules, mission mode can only be achieved if none of the module interfaces, host side or media side are in pattern generation mode. A module reset may be used to guarantee that the module reverts back to mission mode. Otherwise, the host has to ensure that all pattern generation modes on all lanes for both the host and media sides are disabled for these types of modules.

F.2 Enabling and Disabling Pattern Checker (Host or Media Side)

The following procedure contains the recommended sequence of register accesses to set the module into Host side (or Media side) pattern checking mode. Host side registers are used in the example, with media side registers shown in parentheses.

1. Write Bank Select and Page Select registers 00h:127-128 to select Page 13h on the appropriate Bank.
2. Configure desired mode of operation in Byte 13h:177
3. Configure desired pattern checker configuration and lane pattern in 13h:161-167 (13h:169-175)
4. Configure desired control options in 13h:176-179
5. Enable the pattern checker on the selected lanes by writing to 13h:160.7-0 (13h:168.7-0)

When the host enables the PRBS checker, the module is expected to reset the error counters and enable the error counters to begin counting. The behavior of the error counters is defined by Byte 13h:177.

To disable the pattern checker (at any time) including in the middle of a gated error count operation

6. Clear the bits representing the desired lanes in byte 13h:160.7-0 (13h:168.7-0)

If the pattern checker is disabled in the middle of a gated operation, all of the error counters are undefined.

Disabling in the middle of a gated count is considered an abort operation by the module.

The following section details some of the error counter configurations and their expected behavior.

F.3 Reading Pattern Checker Error Counters

There are many scenarios in which pattern checkers can be used, based on the configuration in Byte 13h:177.

The following sections describe recommended host write sequences for some commonly used scenarios.

These recommended sequences are provided for information. Media side registers are provided in these examples, but the same sequences can also be applied to host side registers.

The example Application for these sequences has 8 host side lanes and 4 media side lanes.

F.3.1 Not Gated (Continuous) Error Counters, Individual Lanes

In this scenario the error statistics are collected as cumulative (non-periodic, open-end) statistics

Host Selected Mode of Operation

a. 13h:177.3-1 = 000b (not gated)
b. 13h:177.5 = 0 (do not hold checkers in reset)
c. 13h:177.7 = 0 (reset error counts per lane)
d. 13h:129.4 = 0 (periodic update disabled)
e. 13h:177.0 = X (don't care update interval)

Host Write Sequence

1. Write 13h:160.7-0 (13h:168.7-0) to enable the pattern checker on selected media side lanes.
   a. The module will apply the configuration and control options to the enabled lanes.
   b. Since the configuration is not gated and periodic update is disabled, the latest Error counters are available "on demand" when the host reads the current pattern checker data.
2. Write 14h to the Page select register. Since error information is on demand, module may take additional time to provide the selected diagnostics data.
3. Write 02h (04h) into the Diagnostics Selector 14h:128 to select lanes 1-4 for the host side (media side) or write a 03h (05h) to select lanes 5-8 (e.g. for DR4, FR4, LR4)
4. Module will perform a read of the pattern checker error counters when byte 128 is written. (Module will not clear the error counters.)
5. Read Byte 14h:138 (14h:139) to ensure the pattern checker has not lost lock.
6. Read Bytes 14h:192-255 to obtain error counters and total bits.

F.3.2 Not Gated (Continuous) Error Counters, Individual Lanes, Reset Error Counter

In this scenario error statistics are collected as interval statistics. The interval length per lane is under host control

Configuration Assumptions

a. 13h:177.3-1 = 000b (not gated)
b. 13h:177.5 = 0 (do not hold checkers in reset)
c. 13h:177.7 = 0 (reset error counts per lane)
d. 13h:129.4 = 1 (periodic update enabled)
e. 13h:177.0 = 0 (**1 second update interval**)

Host Write Sequence

1. Write bits 13h:160.7-0 (13h:168.7-0) to enable the pattern checker on selected media side lanes.
   a. The module will apply the configuration and control options to the enabled lanes.
   b. Since the configuration is not gated and periodic update is enabled, the module may provide the error information from the last update period.
   c. Error counters may also be available "on demand" when the host reads the current pattern checker data (this behavior is deliberately left to be vendor dependent).
2. Write 14h to the Page select register. To provide better response to host, the module may return the last updated error information.
3. Write 02h (or 04h) into the Diagnostics Selector 14h:128 to select lanes 1-4 for the host side (media side) or write a 03h (05h) to select host (media) lanes 5-8 (e.g. for DR8, FR8, LR8)
4. Read byte 14h:138 (14h:139) to ensure the pattern checker has not lost lock.
5. To reset the error information the host may set Bit 13h:177.5. When set:
   a. Module (may also read from the data path chips and then) freezes the current error information.
   b. Module copies current internal pattern checker counters into the counter results.
6. Host may write 11h-15h to the Diagnostics Selector 14h:128 to select the collected error information results, and then read bytes 14h:192-255 to obtain the results of BER or of error counters and total bits.

7. **To restart error counting** the host may clear Bit 13h:177.5. This will reset the current error information (selector 01h-05h) without affecting the error information results (selector 11h-15h).

F.3.3 Not Gated (Continuous) Error Counters, All Lanes, all Banks

In this scenario the error statistics are operated as host-controlled interval statistics

Configuration Assumptions

a.	13h:177.3-1 = 000b	(not gated)
b.	13h:177.5 = 0	(do not hold checkers in reset)
c.	13h:177.7 = 1	(**reset error counts on all Banks all enabled lanes**)
d.	13h:129.4 = 0	(periodic update enabled)
e.	13h:177.0 = 1	(**5 second update interval**)

Host Write Sequence

1. Write bits 13h:160.7-0 (13h:168.7-0) to enable the pattern checker on selected media lanes.
    a. The module will apply the configuration and control options to the enabled lanes.
    b. Since the configuration is not gated and periodic update is enabled, the module may provide the error information from the last update period.
    c. Error counters may also available "on demand" when the host reads the current pattern checker data, but this behavior is deliberately left as vendor dependent.
2. Write 14h to the Page select register. To provide better response to host, the module may return the last polled error information.
3. Write 02h (04h) to the Diagnostics Selector 14h:128 to select lanes 1-4 for the host side (media side) or write 03h (05h) to select host (media) lanes 5-8 (e.g. for DR8, FR8, LR8)
4. Host should read byte 14h:138 (14h:139) to ensure the pattern checker has not lost lock.
5. If the host wants to reset error information, the host shall set Bit 13h:177.5. When set:
    a. Module (may also read from the data path chips and then) freezes the current error information. Since this Bank's Bit 13h:177.7 is set, the module will also freeze the current error information of all enabled Banks with Bit 13h:177.7 set.
    b. Module copies current internal pattern checker counters into counting results. The module will copy error information of all enabled lanes and Banks with Bit 13h:177.7 set to error information results.
6. Host may write 11h-15h to the Diagnostics Selector 14h:128 to select the collected error information results of all Banks, and then read Bytes 14h:192-255 to obtain the results of BER or of error counters and total bits for the lanes represented in the respective Bank.
7. **To restart error counting** the host may clear Bit 13h:177.5. This will reset the current error information (selector 01h-05h) without affecting the error information results (selector 11h-15h).

Appendix G Specification Evolution and Maintenance Notes

This appendix documents considerations for future evolution of the specification.

The audience of this appendix is the editor and the group involved in maintaining and evolving CMIS.

The desired property of "backwards compatibility" of a new CMIS revision leads to subtle constraints both for allowable specification changes and for later extensions of the specification.

The purpose of this appendix is to make those subtle constraints explicit and clearly visible to the CMIS audience and the specification evolution group in particular.

As will be shown below, the following guideline should be used whenever possible:

> **CMIS Evolution Guideline**: New CMIS functionality for modules should be specified in a way that it either works silently and unattended in the default configuration of the module, or that it is passivated by default until explicitly enabled by a host who is aware of the new functionality.

.

G.1 Definitions

A CMIS version is indicated by a version number with two components called **major revision number** and **minor revision number**.

- Backwards compatibility from a **host perspective** means that a host can, in principle, always interwork with older modules (i.e. a host can manage modules using an earlier version of CMIS than the host).
- Backwards compatibility from a **module perspective** means that a module can interwork with older hosts (i.e. the module can be managed by hosts implementing an earlier version of CMIS than the module).

G.2 Cross-Version Compatibility

A host can query the CMIS version of a module at runtime, while the module does not know about the CMIS version implemented in the host.

When the CMIS version supported by a module is **older** than the CMIS version run by the host, the host can, at least in principle, adapt to a known older CMIS version and manage the module.

When the CMIS version supported by a module is **newer** and hence unknown to the host, it depends on the nature of the changes between the host's CMIS version and the newer CMIS version: If the new version provides only **compatible** extensions, or **incompatible but passivated** extensions that work well with their default settings and behavior, the host can still manage the module using the functionality supported by the host's older CMIS version.

G.3 Interpretation of CMIS Version Numbers

The CMIS version of a new CMIS revision will be defined to enable the following runtime rules for hosts:

- If a module reports the **same or a smaller CMIS major revision number** than the host, the module can be managed (in principle, possibly depending on the host dynamically adapting to an older version)
- If the module reports a **higher CMIS major revision number** than the host, the module may not behave as per host expectations and therefore cannot be managed

Appendix H Name Mapping CMIS 4.0 to CMIS 5.0 (Temporary)

The previously informal field names (naming phrases) in the management Memory Map have been converted to syntactically recognizable formal field names.

To support implementations that might have generated source code based on the field name wording in CMIS 4.0, the name translations are described in this appendix.

This appendix will be removed in the next CMIS revision.

H.1 Transformation Rules

When converting naming phrases to formal names, the following conversion rules and strategies have been applied.

The naming in various sections of CMIS 4.0 was heterogeneous, so mechanistic conversion was not possible.

	Transformation	**Examples**
Space Removal	Spaces in register, bit, or field naming phrases have been removed. Word separation by CamelCase convention.	Rx Lane → RxLane
Decimal Separator	. replaced by p	3.33GHz → 3p33GHz 3.3V → 3p3V
Special Characters	Dashes removed. L- and M- prefix converted to Flag and Mask suffix) Otherwise ad-hoc replacements	Pre-FEC → PreFEC L-XXX → XXXFlag / → Or
Word Separators	Capitalization of first letter of each word, with exceptions for readability when adjacent parts are all uppercase. Existing underscores removed.	Rx Lane → RxLane Apply_Imm* → ApplyImm*
Acronyms	Commonly recognized acronyms ABC were usually retained, even when word boundaries are no longer recognized. In other cases, the acronym is written Abc	PRBSGen (PrbsGen) PostFec
Shortening	Some descriptive phrases were too long and have been reasonably shortened	
Replacements	A few name components have been mechanistically changed (find-replace)	Implemented → Supported
Renaming	Some poor names without verb, object, or context were replaced to better represent their meaning. Where possible an <Object><Verb> pattern was preferred.	
Instance Numbers	Instance numbers (especially lane instance indicators Rx<i>, Tx<i>, Lane<i>) that were already embedded in names were retained but moved to the end (this allows to treat a sequence of like registers as a flattened array, such that the left prefix can serve as array name).	Rx1Attribute→AttributeRx1 Tx7Attribute→AttributeTx7 Lane1Attr→AttrLane1
Arrays	Elements of arrays in a Page which are not fully unfolded into a list of individually named instances are assumed to be numbered from the containing array name.	RxDataArray<n=1-8>
Placeholders	Variables (placeholders) in names are marked-up by angle brackets.	RxData<n> LaneAttributeTx<n>

H.2 Name Mapping Table

A name mapping table is provided in a separate spreadsheet named "CMIS4.0to5.0FieldNameChanges.xlsx"

End of Document