

JEDEC STANDARD

Byte Addressable Energy Backed Interface

JESD245B.01 (Revision of JESD245B, July 2017)

SEPTEMBER 2017

JEDEC SOLID STATE TECHNOLOGY ASSOCIATION



NOTICE

JEDEC standards and publications contain material that has been prepared, reviewed, and approved through the JEDEC Board of Directors level and subsequently reviewed and approved by the JEDEC legal counsel.

JEDEC standards and publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for use by those other than JEDEC members, whether the standard is to be used either domestically or internationally.

JEDEC standards and publications are adopted without regard to whether or not their adoption may involve patents or articles, materials, or processes. By such action JEDEC does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the JEDEC standards or publications.

The information included in JEDEC standards and publications represents a sound approach to product specification and application, principally from the solid state device manufacturer viewpoint. Within the JEDEC organization there are procedures whereby a JEDEC standard or publication may be further processed and ultimately become an ANSI standard.

No claims to be in conformance with this standard may be made unless all requirements stated in the standard are met.

Inquiries, comments, and suggestions relative to the content of this JEDEC standard or publication should be addressed to JEDEC at the address below, or refer to www.jedec.org under Standards and Documents for alternative contact information.

Published by
©JEDEC Solid State Technology Association 2017
3103 North 10th Street
Suite 240 South
Arlington, VA 22201-2107

This document may be downloaded free of charge; however JEDEC retains the copyright on this material. By downloading this file the individual agrees not to charge for or resell the resulting material.

PRICE: Contact JEDEC

Printed in the U.S.A.
All rights reserved

PLEASE!

DON'T VIOLATE
THE
LAW!

This document is copyrighted by JEDEC and may not be
reproduced without permission.

For information, contact:

JEDEC Solid State Technology Association
3103 North 10th Street
Suite 240 South
Arlington, VA 22201-2107

or refer to www.jedec.org under Standards-Documents/Copyright Information.

BYTE ADDRESSABLE ENERGY BACKED INTERFACE

(From JEDEC Board Ballot JCB-17-15, formulated under the cognizance of the JC-45.6 Subcommittee on Hybrid Modules.)

CONTENTS

Foreword	v
Introduction	v
1 Scope.....	1
2 Normative References.....	1
3 Terms and definitions	2
3.1 Acronyms	2
3.2 Terms and Definitions	3
3.3 Keywords.....	4
3.4 Conventions	5
4 Introduction.....	6
5 I²C.....	7
5.1 I ² C Bus.....	7
5.2 I ² C transactions.....	7
5.3 Paging.....	10
5.4 Typed Block Data.....	11
6 Serial Presence Detect.....	12
6.1 Byte Addressable Energy Backed Interface Discovery	12
7 Features.....	13
7.1 Controller Ready.....	13
7.2 Operations.....	13
7.2.1 Catastrophic Save operation	13
7.2.2 Restore operation.....	15
7.2.3 Erase operation	16
7.2.4 Arm operation.....	16
7.2.5 Management operations.....	17
7.2.6 Set Energy Source Policy operation	17
7.2.7 Set Event Notification operation.....	17
7.2.8 Firmware operations	18
7.2.9 Factory Default operation	19
7.2.10 Reset Controller operation	20
7.2.11 Operational Unit operations.....	21
7.2.12 Atomic Arm and Erase operation	21
7.2.13 Abort operation.....	22
7.3 Energy Source.....	23
7.3.1 Local Energy Source.....	23
7.3.2 Tethered Energy Source	23
7.3.3 Host Energy Source	23
7.3.4 Shared Energy Source.....	23
7.3.5 Energy Source Policy	24
7.4 Event Notification.....	24

7.5 Error and Warning Thresholds	25
7.6 Module Firmware	25
7.7 CRC Algorithm.....	28
7.8 Error Injection.....	29
7.9 Statistics.....	29
7.10 Thermal.....	30
7.11 Host Area.....	31
7.12 Vendor Log Page	31
7.13 Vendor-Specific.....	31
7.14 LCOM Interface	31
7.15 Label Data	32
7.16 Catastrophic Save transition registers.....	32
8 Register Map	33
8.1 Page 0 register map.....	33
8.1.1 Paging Mechanism registers	36
8.1.2 Version registers	37
8.1.3 Characteristics registers	39
8.1.4 Runtime Command registers	49
8.1.5 Runtime Command Status registers.....	54
8.1.6 Catastrophic Save registers.....	62
8.1.7 Thresholds registers	64
8.1.8 Module registers	67
8.1.9 NVM Subsystem registers	74
8.2 Page 1 register map.....	75
8.2.1 Paging Mechanism registers	76
8.2.2 Energy Source Version registers.....	76
8.2.3 Energy Source Characteristics registers.....	77
8.2.4 Energy Source Runtime Command registers.....	79
8.2.5 Energy Source Runtime Command Status registers	80
8.2.6 Energy Source registers	80
8.3 Page 2 register map.....	81
8.3.1 Paging Mechanism registers	83
8.3.2 Device Statistics registers	83
8.3.3 Error Injection registers	92
8.3.4 Host Area registers	96
8.4 Page 3 register map.....	98
8.4.1 Paging Mechanism registers	99
8.4.2 Typed Block Data registers	100
8.4.3 Firmware Update registers.....	103
8.4.4 Byte and Word Transaction Data registers for Typed Block Data	103
8.4.5 Block Transaction Data registers for Typed Blocked Data	108
8.5 Page 4 register map.....	109
8.5.1 OPEN_PAGE – Offset 0x00.....	109
8.5.2 PDA_SUPPORTED – Offset 0x04.....	109
8.5.3 PDA_VALID – Offset 0x05	109
8.5.4 MRx_SHARED registers – Offsets 0x10-0x11 to 0x1C-0x1D	110
8.6 Page 5 register map.....	110

8.6.1 OPEN_PAGE – Offset 0x00	111
8.6.2 MRx_PDA_SDRAMyy registers – Offsets 0x10-0x11 to 0xFE-0xFF	111
8.7 Page 6 register map	111
8.7.1 OPEN_PAGE – Offset 0x00	111
8.7.2 RCD_FUNCTIONS_SUPPORTED0 – Offset 0x04	111
8.7.3 RCD_FUNCTIONS_SUPPORTED1 – Offset 0x05	112
8.7.4 RCD_FUNCTIONS_VALID0 – Offset 0x06	112
8.7.5 RCD_FUNCTIONS_VALID1 – Offset 0x07	112
8.7.6 Control word registers	112
8.8 Page 7 register map	113
8.8.1 OPEN_PAGE – Offset 0x00	113
8.8.2 Control word registers	113
8.9 Page 8 register map	113
8.9.1 OPEN_PAGE – Offset 0x00	113
8.9.2 Control word registers	113
8.10 Page 9 register map	114
8.10.1 OPEN_PAGE – Offset 0x00	114
8.10.2 Control word registers	114
9 Host Operation Workflows	115
9.1 Controller Ready workflow	115
9.2 Catastrophic Save workflow	115
9.3 Restore workflow	115
9.4 Arm workflow	116
9.5 Erase workflow	116
9.6 Abort Running Operation workflow	116
9.7 Firmware Update workflow	117
9.8 Typed Block Data workflow	122
9.8.1 Reading Typed Block Data workflow	123
9.8.2 Writing Typed Block Data workflow	124
ANNEX A (normative) SDRAM and RCD Requirements for NVRDIMM-N Modules	125
A.1 SDRAM and RCD requirements during a Catastrophic Save operation	125
A.2 SDRAM and RCD requirements related to a Restore operation	125
A.2.1 Overview	125
A.2.2 Restore operation requirements	125
A.2.3 Post-restore transition entry requirements	126
A.2.4 Post-restore transition exit requirements	131
ANNEX B (informative) revision history	135
B.1 Differences Between JESD245 (v1.0) and JESD245A (v2.0)	137
B.2 Differences Between JESD245A (v2.0) and this standard (JESD245B)(v2.1)	135

FIGURES

Figure 1: NVDIMM-N overview	6
Figure 2: I ² C Read Byte transaction	8
Figure 3: I ² C Write Byte transaction	8
Figure 4: I ² C Read Word transaction	8
Figure 5: I ² C Write Word transaction	9
Figure 6: I ² C Block Read transaction	9
Figure 7: I ² C Block Write transaction	9
Figure 8: I ² C paging mechanism	11

TABLES

Table 1: Required SPD fields	12
Table 2: Firmware image header – common format	26
Table 3: Firmware image header format 0	26
Table 4: Firmware image header format 1	27
Table 5: Temperature value bit definition	30
Table 6: Mode register – lower byte	32
Table 7: Mode register – upper byte	32
Table 8: Page 0 register map categories	33
Table 9: Page 0 register map	34
Table 10: Major Revision and Minor Revision fields	37
Table 11: Reset Controller Scope field	40
Table 12: Page 1 register map categories	75
Table 13: Page 1 register map	75
Table 14: Page 2 register map categories	81
Table 15: Page 2 register map	82
Table 16: Page 3 register map categories	98
Table 17: Page 3 register map	98
Table 18: Page 4 register map	109
Table 19: Page 5 register map	110
Table 20: Page 6 register map	111
Table 21: Page 7 register map	113
Table 22: Page 8 register map	113
Table 23: Page 9 register map	114
Table 24: MR0 requirements during a post-restore transition	127
Table 25: MR1 requirements during a post-restore transition	127
Table 26: MR2 requirements during a post-restore transition	127
Table 27: MR3 requirements during a post-restore transition	128
Table 28: MR4 requirements during a post-restore transition	128
Table 29: MR5 requirements during a post-restore transition	129
Table 30: MR6 requirements during a post-restore transition	129
Table 31: RCD requirements during a post-restore transition	130
Table 32: Host reprogramming RCD function 0 control words after a post-restore transition	132
Table 33: Host reprogramming RCD function 1 control words after a post-restore transition	133
Table 34: Host reprogramming RCD function 4 control words after a post-restore transition	133

FOREWORD

This standard has been prepared by JEDEC. The purpose of this standard is definition of a byte addressable energy backed function on a non-volatile dual in-line memory module (NVDIMM). This standard defines the feature set and commands implemented by the byte addressable energy backed function on an NVDIMM-N.

INTRODUCTION

An NVDIMM-N is a memory module that can be integrated into a standard platform. A Byte Addressable Energy Backed Function on a NVDIMM is designed to preserve data in the event of the power failure. A Byte Address Energy Backed Function is backed by a combination of SDRAM and non-volatile memory (e.g., NAND flash) on the NVDIMM-N. It operates at SDRAM speeds and provides persistent storage by backing up the SDRAM contents into the non-volatile memory in the event of a power failure. This is made possible by an Energy Source (e.g., supercapacitor) which maintains charge on the module enabling back-up of data from SDRAM to the non-volatile memory, providing a storage-class memory solution.

To be able to provide interoperability and the ability for platform and platform software (e.g., BIOS) to support NVDIMM-Ns from various manufacturers, standardization of the host to module interface, discovery mechanism, the feature set and command operations are required, as described in this standard.

BYTE ADDRESSABLE ENERGY BACKED INTERFACE

(From JEDEC Board Ballot JCB-17-15, formulated under the cognizance of the JC-45.6 Subcommittee on Hybrid Modules.)

1 SCOPE

This standard specifies the host and device interface for a DDR4 NVDIMM-N, which is a DIMM that achieves non-volatility by copying SDRAM contents into non-volatile memory (NVM) when host power is lost using an Energy Source managed by either the module or the host.

Although this standard is targeted towards DDR4 NVDIMM only, it does not preclude adoption of this standard by other implementations (e.g., DDR3 NVDIMM).

2 NORMATIVE REFERENCES

The normative documents listed in this clause contain provisions that, through reference in this text, constitute provisions of this standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

JESD21C, 4.20.28 *DDR4 SDRAM Registered DIMM Design Specification*, Revision 1.00 (August 2015)

JESD21C, 4.1.2.L-4 *Annex L: Serial Presence Detect (SPD) for DDR4 SDRAM Modules* (DDR4 SPD Document Release 4)

JESD21C, 4.1.2.L-5 *Annex L: Serial Presence Detect (SPD) for DDR4 SDRAM Modules* (DDR4 SPD Document Release 5)(forthcoming)

JESD21C, 4.1.6 *Definitions of the EE1004-v 4 Kbit Serial Presence Detect (SPD) EEPROM and TSE2004av 4 Kbit SPD EEPROM with Temperature Sensor (TS) for Memory Module Applications*

JESD21C, 4.1.6-2 *Definitions of the EE1004av 4 Kbit Serial Presence Detect (SPD) EEPROM and TSE2004av 4 Kbit SPD EEPROM with Temperature Sensor (TS) for Memory Module Applications* (forthcoming)

JESD79-4A, *DDR4 SDRAM* (November 2013)

JESD79-4B, *DDR4 SDRAM* (June 2017)

JESD82-31, *DDR4 Registering Clock Driver – DDR4RCD01* (August 2016)

JESD82-31A, *DDR4 Registering Clock Driver – DDR4RCD02* (forthcoming)

JESD82-32, *DDR4 Data Buffer – DDR4DB01* (August 2016)

JESD82-32A, *DDR4 Data Buffer – DDR4DB02* (forthcoming)

JESD245, *Byte Addressable Energy Backed Interface* (December 2015)

JESD245A, *Byte Addressable Energy Backed Interface* (September 2016)

JESD248 *DDR4 NVDIMM-N Design Specification* (Revision 1.0)(September 2016)

JESD248A *DDR4 NVDIMM-N Design Specification* (forthcoming)

I²C Bus Specification Revision 6 (4 April 2014)

System Management Bus (SMBus) Specification Version 3.0 (20 December 2014)

3 TERMS AND DEFINITIONS

For the purposes of this standard, the terms and definitions given in the document included in clause 2 “Normative References” and this clause apply.

3.1 Acronyms

DDR3	Double Data Rate version 3
DDR4	Double Data Rate version 4
DIMM	Dual In-line Memory Module
ES	Energy Source
I ² C	Inter IC (inter integrated circuit)
LCOM	LCOM Bus
NVDIMM	Non-volatile Dual In-line Memory Module
NVDIMM-N	NVDIMM with a Byte Addressable Energy Backed Interface function
NVM	Non-Volatile Memory
NVRDIMM-N	NVDIMM-N that is an RDIMM
RCD	Registering Clock Driver
RDIMM	Registered DIMM
SDRAM	Synchronous Dynamic Random Access Memory
SMBus	System Management Bus
SPD	Serial Presence Detect

3.2 Terms and Definitions

Abort: Operation that stops the currently running operation on the module. See 7.2.13.

Arm: Operation that enables or disables trigger(s) for a Catastrophic Save operation. See 7.2.4.

Catastrophic Save: Operation that copies the SDRAM contents into NVM when power is lost. The Catastrophic Save operation is started when an enabled trigger occurs or a write to an I²C register occurs. See 7.2.1.

CKE Power Down mode: RCD mode in which all input buffers are disabled except for CK_t/CK_c, DCKEn, DRST_n, and sometimes DODTn and ERROR_IN_n. See JESD82-31A (DDR4RCD02).

Clock Stopped Power Down mode: RCD mode in which all input buffers are disabled except for CK_t/CK_c. See JESD82-31A (DDR4RCD02).

Device Managed Policy: Energy Source policy where the module manages the Energy Source used during the Catastrophic Save operation.

Energy Source: A device that is capable of storing and providing energy to the module during a Catastrophic Save operation. See 7.2.13.

Erase: Operation that deletes the previously saved SDRAM image in NVM. See 7.2.3.

Factory Default: Operation that erases all NVM on the module and resets readable registers to their factory default values except the data needed to determine warranty compliance. This operation does not impact firmware on the module. See 7.2.9.

Firmware Operations: Operations that are related to updating the firmware on the module. See 7.2.8.

Host: The system in which the module is installed.

Host Managed Policy: Energy Source policy where the host manages the Energy Source used during the Catastrophic Save operation.

I²C Bus: A bidirectional 2-wire bus for efficient inter-IC control.

LCOM: A local communication bus, bidirectional 6-wire interface that connects the module's controller to other components.

Management Operations: Operations that either reset the controller or clear status register(s). See 7.2.5.

Maximum Power Saving mode: SDRAM mode similar to Self-Refresh mode that does not perform internal refreshes. See JESD79-4B (DDR4 SDRAM).

Power Down mode: SDRAM mode in which all input buffers except CK_t/CK_c, CKE, RESET_n, and sometimes ODT are disabled. See JESD79-4B (DDR4 SDRAM).

Restore: Operation that restores previously saved SDRAM contents from NVM to SDRAM. See 7.2.2.

RCD Control Words: The registers on a RDIMM that configure the RCD for operational use.

Saturating counter: a counter that remains at its maximum value after reaching its maximum value

Self-Refresh mode: SDRAM mode in which all input buffers except CKE and RESET_n are disabled. See JESD79-4B (DDR4 SDRAM).

SDRAM Mode Registers: The registers on SDRAM that configure the SDRAM for operational use. Some of the mode registers are write-only registers.

Set Energy Source Policy: Operation that configures the Energy Source to be used by the module in the Catastrophic Save operation. See 7.2.6.

Set Event Notification: Operation that either enables or disables notification support on the module when certain events occur. See 7.2.7.

Typed Block Data: A collection of data that is transferred between the host and module in 32 byte blocks. See 5.4.

Vendor Log Page: An optional area on the module that is accessible by the host and containing vendor specific data useful to triage issues on the module. See 7.12.

3.3 Keywords

Several keywords are used to differentiate levels of requirements and options, as follow:

Can - A keyword used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

Expected - A keyword used to describe the behavior of the hardware or software in the design models assumed by this standard. Other hardware and software design models may also be implemented.

Ignored - A keyword that describes bits, bytes, quadlets, or fields whose values are not checked by the recipient.

Mandatory - A keyword that indicates items required to be implemented as defined by this standard.

May - A keyword that indicates a course of action permissible within the limits of the standard (*may* equals *is permitted*).

Must - The use of the word *must* is deprecated and shall not be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.

Optional - A keyword that describes features which are not required to be implemented by this standard. However, if any optional feature defined by the standard is implemented, it shall be implemented as defined by the standard.

Reserved - A keyword used to describe objects—bits, bytes, and fields—or the code values assigned to these objects in cases where either the object or the code value is set aside for future standardization. Usage and interpretation may be specified by future extensions to this or other standards. A reserved object shall be zeroed or, upon development of a future standard, set to a value specified by such a standard. The recipient of a reserved object shall not check its value. The recipient of a defined object shall check its value and reject reserved code values.

Shall - A keyword that indicates a mandatory requirement strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals *is required to*). Designers are required to implement all such mandatory requirements to assure interoperability with other products conforming to this standard.

Should - A keyword used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).

Will - The use of the word *will* is deprecated and shall not be used when stating mandatory requirements; *will* is only used in statements of fact.

3.4 Conventions

This standard uses the conventions described in this subclause.

A binary number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 and 1 immediately followed by a lower-case b (e.g., 0101b). Spaces may be included in binary number representations to increase readability or delineate field boundaries (e.g., 0 0101 1010b).

A hexadecimal number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 through 9 and/or the upper-case English letters A through F immediately followed by a lower-case h (e.g., FA23h). Spaces may be included in hexadecimal number representations to increase readability or delineate field boundaries (e.g., B FD8C FA23h).

A decimal number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 through 9 not immediately followed by a lower-case b or lower-case h (e.g., 25).

A range of numeric values is represented in this standard in the form "a to z", where a is the first value included in the range, all values between a and z are included in the range, and z is the last value included in the range (e.g., the representation "0h to 3h" includes the values 0h, 1h, 2h, and 3h).

When the value of the bit or field is not relevant, x or xx appears in place of a specific value.

4 INTRODUCTION

A nonvolatile DIMM (NVDIMM) is a DIMM that maintains the contents of SDRAM during power loss. An NVDIMM-N achieves non-volatility by:

- performing a Catastrophic Save operation to copy SDRAM contents into NVM when host power is lost using an Energy Source managed by either the module or the host; and
- performing a Restore operation to copy contents from the NVM to SDRAM when power is restored.

This standard defines an I²C Bus-based register interface for a NVDIMM-N called the Byte Addressable Energy Backed Interface. Figure 1 shows the components of an NVDIMM-N that differ from a regular DIMM. The Non-Volatile Memory Subsystem Controller (i.e., the controller) includes a memory controller for the SDRAM, an NVM controller for the NVM, and I²C registers. The NVDIMM-N draws power for the Catastrophic Save operation either from the host via the V₁₂ pin or from a Device-Managed Energy Source.

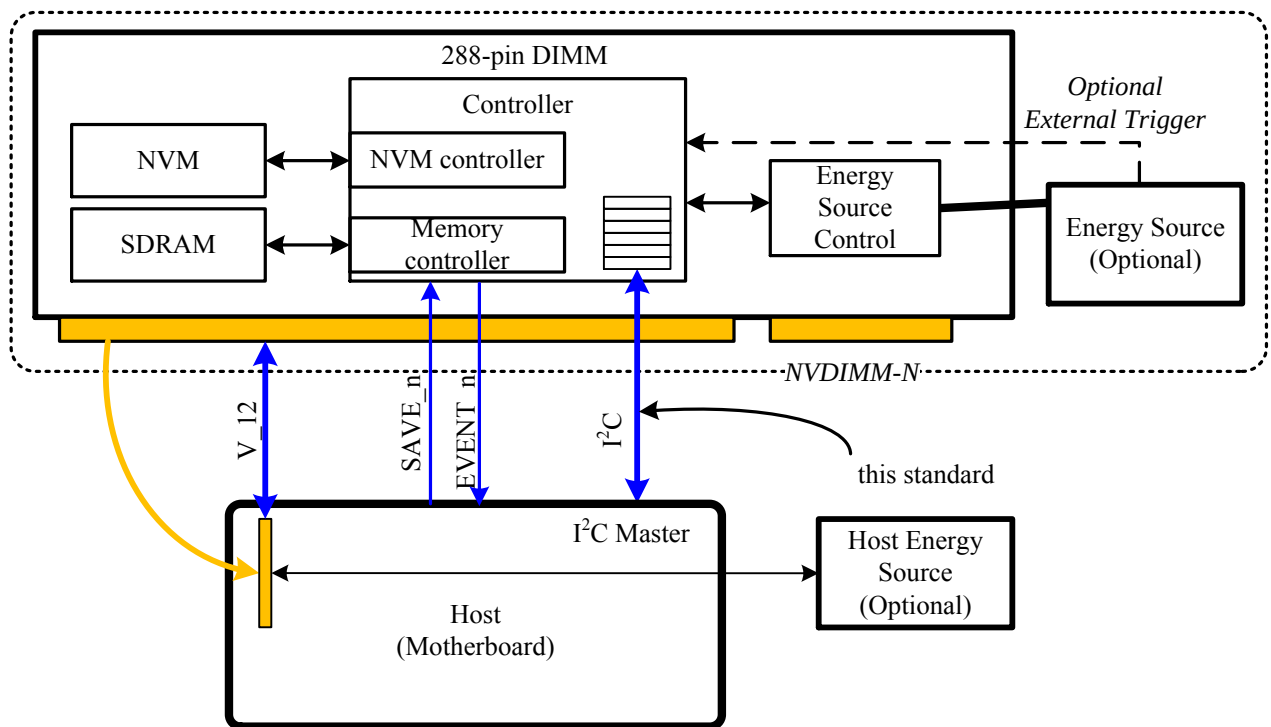


Figure 1 — NVDIMM-N overview

5 I²C

This section describes the I²C interface that a module compliant with this standard shall support.

5.1 I²C Bus

The module's I²C bus support shall be compliant with SMBus 3.0. Modules shall support the SMBus 100 kHz Class, the SMBus 400 kHz Class (I²C Fast-mode), and the SMBus 1 MHz Class (I²C Fast-mode Plus). Modules shall not clock stretch any I²C transactions.

The module shall implement an I²C slave with the seven-bit I²C address of 1000YYY (binary) where YYY ranges from 0-7 (decimal) followed by the data direction bit (e.g., 1 indicates a read transaction and 0 indicates a write transaction).

5.2 I²C transactions

Modules:

- shall support I²C Read Byte transactions (see 5.2.1) and I²C Write Byte transactions (see 5.2.2);
- may support I²C Read Word transactions (see 5.2.3) and I²C Write Word transactions (see 5.2.4) to the TYPED_BLOCK_DATA_BYTE0 through TYPED_BLOCK_DATA_BYTE31 registers in page 3; and
- may support I²C Block Read transactions (see 5.2.5) and I²C Block Write transactions (see 5.2.6) to the TYPED_BLOCK_DATA_OFFSET register in page 3. For each I²C Block Read and I²C Block Write transaction, the transfer size is 32 bytes.

Modules may support I²C Read Word, I²C Write Word, I²C Block Read, and I²C Block Write transactions for additional vendor-specific registers.

If a module supports I²C Read Word and I²C Write Word transactions, it shall set Bit 3 (I²C Word Transactions Supported) in the CAPABILITIES1 register (see 8.1.3.2).

If a module supports I²C Block Read and I²C Block Write transactions, it shall set Bit 4 (I²C Block Transactions Supported) in the CAPABILITIES0 register (see 8.1.3.1).

This subclause describes the I²C transactions between the host and the module. The host is the I²C master and the module is the I²C slave. In the figures describing I²C transactions:

- White shading denotes bit(s) sent by the host;
- Gray shading denotes bit(s) sent by the module;
- “Start” represents the START condition, always sent by the host;
- “Repeated Start” represents the REPEATED START condition, always sent by the host;
- “Stop” represents the STOP condition, always sent by the host;
- “Read” represents the data direction bit with a value of 1, always sent by the host;
- “Write” represents the data direction bit with a value of 0, always sent by the host;
- “Ack” represents the Acknowledge bit, sent by the recipient. It is set to 0 (LOW) to indicate a positive acknowledgement (ACK) signal, and set to 1 (HIGH) to indicate a Not Acknowledge (NACK) signal; and
- “NoAck” represents the Acknowledge bit set to 1 (HIGH) to indicate a Not Acknowledge (NACK) signal, sent by the recipient.
- “Offset” refers to the register offset within the currently opened page as indicated by the OPEN_PAGE register (see 5.3).

5.2.1 I²C Read Byte transaction

The I²C Read Byte transaction is shown in Figure 2. This transaction matches the SMBus Read Byte transaction with the Command Code set to the BAEBI register offset.

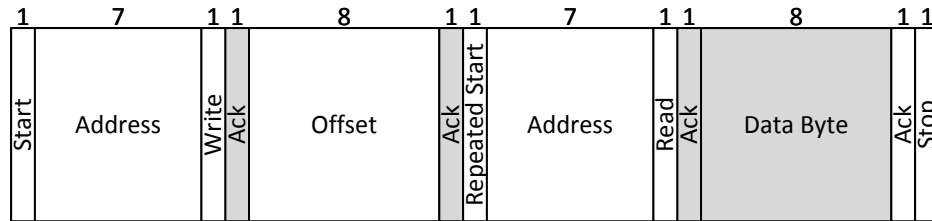


Figure 2 — I²C Read Byte transaction

5.2.2 I²C Write Byte transaction

The I²C Write Byte transaction is shown in Figure 3. This transaction matches the SMBus Write Byte transaction with the Command Code set to the BAEBI register offset.

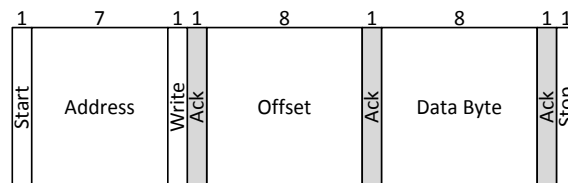


Figure 3 — I²C Write Byte transaction

5.2.3 I²C Read Word transaction

The I²C Read Word transaction is shown in Figure 4. This transaction transfers data in 2-byte blocks. This transaction matches the SMBus Read Word transaction with the Command Code set to the BAEBI register offset.

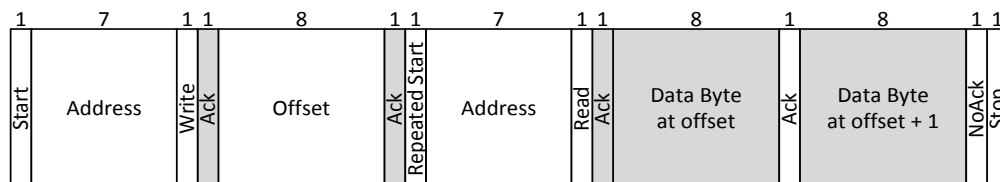


Figure 4 — I²C Read Word transaction

5.2.4 I²C Write Word transaction

The I²C Write Word transaction is shown in Figure 5. This transaction transfers data in 2-byte blocks. This transaction matches the SMBus Write Word transaction with the Command Code set to the BAEBI register offset.

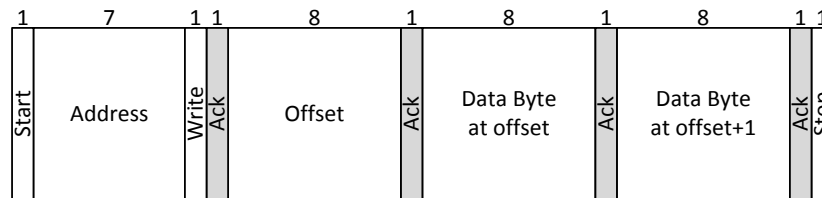


Figure 5 — I²C Write Word transaction

5.2.5 I²C Block Read transaction

The I²C Block Read transaction is shown in Figure 6. This transaction transfers data in 32-byte blocks. This transaction resembles the SMBus Block Read transaction with the Command Code set to the BAEBI register offset, omitting the Block Count byte before the first Data Byte, and transmitting exactly 32 data bytes.

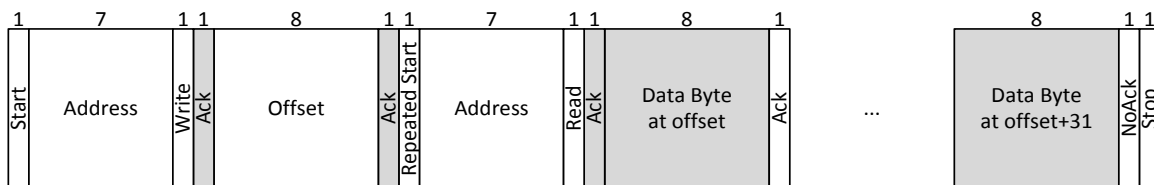


Figure 6 — I²C Block Read transaction

5.2.6 I²C Block Write transaction

The I²C Block Write transaction is shown in Figure 7. This transaction transfers data in 32-byte blocks. This transaction resembles the SMBus Block Read transaction with the Command Code set to the BAEBI register offset, omitting the Block Count byte before the first Data Byte, and transmitting exactly 32 data bytes.

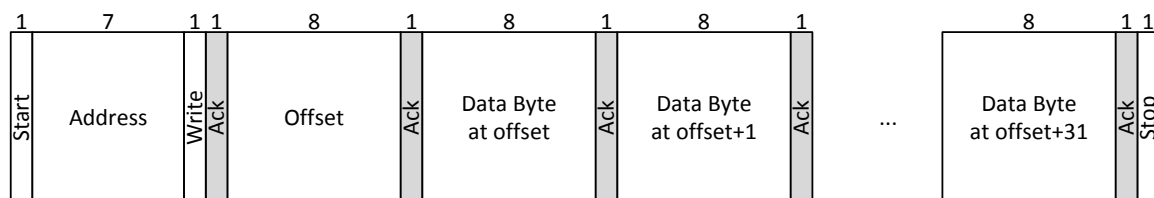


Figure 7 — I²C Block Write transaction

5.2.7 I²C Transactions to Reserved Offsets

Modules shall ignore writes to Reserved offsets. If the host attempts to read a Reserved offset, the module shall return the value 0.

5.2.8 Illegal I²C Transactions

Modules shall ignore writes to read-only registers. If the host attempts to read from a write-only register, the value returned by the module is vendor specific.

Modules shall ignore I²C Block Write transactions to a register that does not support I²C Block Write transactions. Modules shall return a data payload of all zeroes if the host issues an I²C Block Read transaction to a register that does not support I²C Block Read transactions.

5.2.9 I²C Resets

The controller is responsible for resetting its I²C interface state machine upon detecting SCL low for 35 ms cumulative between the START and STOP conditions or detecting SCL high continuously for 100 μ s between the START and STOP conditions.

5.3 Paging

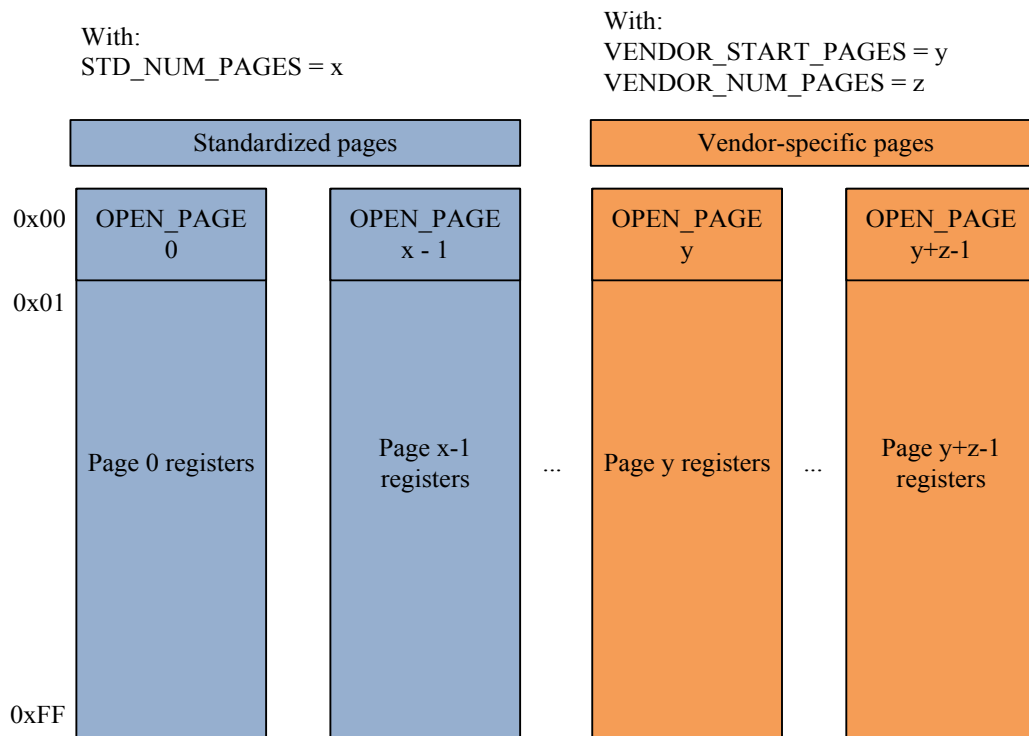
On I²C, the number of offsets available using traditional access method is 256. To access beyond 256 offsets, a paging mechanism is needed. All modules supporting the Byte Addressable Energy Backed Interface shall support a paging mechanism.

In the paging mechanism, there is an OPEN_PAGE register at offset 0x00 of all supported pages. When the OPEN_PAGE register is read, it returns the page that is open for access. When the OPEN_PAGE register is written, it changes the page that is open for access. Each page supports up to 255 offsets. If an attempt is made to access an unsupported page, the page access change shall be ignored and the OPEN_PAGE register value read shall not change from the previous value. The module shall complete the page switch within the time indicated in the PAGE_SWITCH_LATENCY0 and PAGE_SWITCH_LATENCY1 registers (see 8.1.3.9 and 8.1.3.10).

To open a page for access, the host should do the following:

1. Write the desired page number to access to the OPEN_PAGE register at offset 0x00.
2. Wait for the page switching latency as reported in the PAGE_SWITCH_LATENCY0 and PAGE_SWITCH_LATENCY1 registers.
3. Read the OPEN_PAGE register at offset 0 to validate that page access is updated to the desired page. If the OPEN_PAGE register value is not equal to the desired page, the page access change failed.
4. Access the desired offset.

The module shall support at least 4 pages and may support up to a maximum of 256 pages. The OPEN_PAGE register shall not return a page value from an unimplemented or unsupported page.

Figure 8 — I²C paging mechanism

5.4 Typed Block Data

Typed Block Data is a collection of data that is transferred between the host and the module in 32 byte blocks. Each Typed Block Data type has its own defined size, and operations against Typed Block Data are done in Operational Units. Each Typed Block data has its own Operational Unit size. The Typed Block Data size shall be a multiple of the Operational Unit size. Data transfers to an Operational Unit are done in 32 byte blocks using the BLOCK_ID, REGION_ID0 and REGION_ID1 registers in page 3.

To determine the size of Typed Block Data, the host should set the TYPED_BLOCK_DATA register in page 3 to the Typed Block Data type and then read the TYPED_BLOCK_DATA_SIZE0, TYPED_BLOCK_DATA_SIZE1 and TYPED_BLOCK_DATA_SIZE2 registers in page 3.

To determine the Operational Unit size of Typed Block Data, the host should set the TYPED_BLOCK_DATA register in page 3 to the Typed Block Data type and then read the OPERATIONAL_UNIT_SIZE0, OPERATIONAL_UNIT_SIZE1 and OPERATIONAL_UNIT_SIZE2 registers in page 3.

The module shall support the following Typed Block Data types:

- Firmware Image Data (see 7.6)
- Vendor Log Page (see 7.12)
- Label Data (see 7.15)

The module may support additional Typed Block Data types by specifying a vendor-specific Typed Block Data value. If the module does not support a Typed Block Data type, it shall return a Typed Block Data size of 0 for that Typed Block Data type.

The module shall ignore writes to unsupported Typed Block Data types. If the host attempts to read from an unsupported Typed Block Data type, the module shall return a data payload of all zeroes.

6 SERIAL PRESENCE DETECT

Modules shall have a Serial Presence Detect (SPD) EEPROM that is compliant with the EE1004-v and TSE2004av devices defined in JESD21C Page 4.1.6. The contents of the SPD EEPROM shall be compliant with JESD21C Page 4.1.2.L-5 and include the hybrid module parameters for a byte addressable energy backed function defined in JESD21C Page 4.1.2.L-5.

6.1 Byte Addressable Energy Backed Interface Discovery

The contents of the SPD on a module allows the host to discover whether the module implements the Byte Addressable Energy Backed Interface. A module that implements the Byte Addressable Energy Backed Interface shall have an SPD populated with value(s) described in Table 1. Additional SPD bytes are populated based on the specifics of the implementations and all SPD byte values shall comply with JESD21C Page 4.1.2.L-5. All multi-byte values in Table 1 are little endian.

Table 1 — Required SPD fields

SPD Field Name	SPD Byte Number(s)	Value(s)
SPD Revision	1	A value of 0x12 (i.e., NVDIMM-N Revision 1.2) or greater
Key Byte/Module Type	3	Bit 7: 1 (Hybrid module) Bits 6-4: 001 (NVDIMM Hybrid) Bits 3-0: Appropriate value based on implementation (e.g., 0001b if RDIMM)
Maximum Non-Volatile Memory Initialization Time	203	Appropriate value in seconds for the maximum time needed for the NVM hardware subsystem to complete its initialization
Function [0-7] Format Interface Descriptor	204 – 205 206 – 207 208 – 209 210 – 211 212 – 213 214 – 215 216 – 217 218 – 219	One of the Function [0-7] Format Interface Descriptor contains the value xxxxxx0000100001b (Byte Addressable Energy Backed, JEDEC Byte Addressable Energy Backed Protocol Function Interface 1)

7 FEATURES

7.1 Controller Ready

While the module is performing power on initialization, it may tristate its SDA pin and not respond to I²C transactions. Because the SDA signal is pulled up by the system, the host detects a NACK for the Address byte of any I²C transactions.

After the module completes power on initialization and starts responding to I²C transactions:

- if the value of the NVDIMM_READY register (see 8.1.5.1) is not 0xA5, the module shall allow access to and the host should only access the following registers defined by this standard:
 - OPEN_PAGE register (see 8.1.1.1);
 - NVDIMM_READY register (see 8.1.5.1);
 - MODULE_HEALTH register (see 8.1.8.1);
 - MODULE_HEALTH_STATUS0 register (see 8.1.8.2); and
 - MODULE_HEALTH_STATUS1 register (see 8.1.8.3).

NOTE Modules compliant with JESD245 and JESD245B do not allow access to the MODULE_HEALTH_STATUS1 register during this time.

- if the value of the NVDIMM_READY register is 0xA5, the module shall allow access to and the host may access all registers.

The module shall complete power on initialization within the timeout reported in SPD byte 203 (Maximum Non-Volatile Memory Initialization Time) (see JESD21C Page 4.1.2.L-5).

After power on or a Reset Controller operation, the host should perform the Controller Ready workflow (see 9.1) before performing any other workflows or register accesses.

7.2 Operations

This section describes the operations that the host may request the module to perform. While an operation is running on the module, the only operations that can be started are the Abort and Management operations. If the current running operation is the Abort operation, the only operation that can be started by the host is the Management operation. If the host attempts to start an operation that cannot be executed due to a current running operation, the module shall ignore the command.

7.2.1 Catastrophic Save operation

Modules shall support the Catastrophic Save operation.

The module may implement a Catastrophic Save Success saturating counter, reported in the CSAVE_SUCCESS_COUNT0 and CSAVE_SUCCESS_COUNT1 registers, that persists through power loss.

The module may implement a Catastrophic Save Failure saturating counter, reported in the CSAVE_FAILURE_COUNT0 and CSAVE_FAILURE_COUNT1 registers, that persists through power loss.

The module shall support the ability to start a Catastrophic Save operation through the START_CSAVE bit in the NVDIMM_FUNC_CMD register (see 8.1.4.3) or the SAVE_n pin. A module may support ability to start a Catastrophic Save operation through vendor specific methods. The module shall also support ability to configure whether the SAVE_n pin is driven low during the duration of the Catastrophic

Save operation through the SAVE_N_LOW_DURING_CSAVE bit in the MODULE_OPS_CONFIG register (see 8.1.8.7).

Upon receiving a START_CSAVE based request, if the NVM has not been erased, the module shall either:

- reject the request by setting the CSAVE_REJECT bit and the CSAVE_ERROR bit in the CSAVE_INFO register and not change any other bits in the CSAVE_STATUS, CSAVE_INFO, CSAVE_FAIL_INFO0, and CSAVE_FAIL_INFO1 registers; or
- perform an Erase operation as the first part of the Catastrophic Save operation.

Upon receiving a SAVE_n based request, if the NVM has not been erased, the module shall perform an Erase operation as the first part of the Catastrophic Save operation.

If the module was armed and did not start a Catastrophic Save operation when power is lost, this shall be considered a Catastrophic Save operation failure.

Upon the start of the Catastrophic Save operation, the module shall:

- if the Catastrophic Save operation was triggered by the START_CSAVE bit, set Bit 1 (Triggered by START_CSAVE) and clear Bit 2 (Triggered by SAVE_n) in the CSAVE_INFO register;
- if the Catastrophic Save operation was triggered by the SAVE_n pin, clear Bit 1 (Triggered by START_CSAVE) and set Bit 2 (Triggered by SAVE_n) in the CSAVE_INFO register;
- clear the CSAVE_FAIL_INFO0 and CSAVE_FAIL_INFO1 registers;
- drive the SAVE_n pin low for the duration of the Catastrophic Save operation if the SAVE_N_LOW_DURING_CSAVE bit is set in the MODULE_OPS_CONFIG register.

Upon the start of the Catastrophic Save operation, the module should detect if the SDRAM is not in Self-Refresh mode. If the module detects that the SDRAM is not in Self-Refresh mode, then the module shall:

- set the DRAM_NOT_SELF_REFRESH bit in the MODULE_HEALTH_STATUS0 register; and
- clear the NVM_Data_Valid bit in the CSAVE_INFO register.

The module shall complete the operation within the time indicated in the CSAVE_TIMEOUT0 and CSAVE_TIMEOUT1 registers.

During the Catastrophic Save operation, the module shall program the RCD Control Words and SDRAM Mode Registers with values compatible with the module memory controller as described in A.1 and shall copy the SDRAM contents to the NVM.

Upon completion of the Catastrophic Save operation, the module shall:

- clear the Catastrophic Save In Progress bit in the NVDIMM_CMD_STATUS0 register;
- update the CSAVE_STATUS and CSAVE_INFO registers;
- disable all triggers;
- unmask the RESET_n pin on the module;
- if the operation was not successful, persist information about the cause(s) of the Catastrophic Save operation failure in the CSAVE_FAIL_INFO0 and CSAVE_FAIL_INFO1 registers;
- if the operation was successful, increment the Catastrophic Save Success counter, if any; and
- if the operation was not successful, increment the Catastrophic Save Failure counter, if any.

Prior to initiating a Catastrophic Save operation, the host should flush all platform buffers to guarantee data consistency and then put the SDRAMs on the module into Self-Refresh mode (see JESD79-4B (DDR4 SDRAM)).

The host should also start a Catastrophic Save operation in scenarios where a platform hard reset is needed. Examples of these scenarios include fatal platform errors such as NMI or host memory controller errors.

The host should record information about failures in the Catastrophic Save operation in the HOST_CSAVE_FAIL register (see 8.3.4.4) and in the Catastrophic Save Workflow Failure saturating counter (see 7.11).

7.2.1.1 SAVE_n pin

For modules supporting Catastrophic Save operation initiation through the SAVE_n pin, the module shall start the Catastrophic Save operation based on the SAVE_n pin as defined in 8.1.8.7.

7.2.2 Restore operation

Modules shall support the Restore operation.

The module may implement a Restore Success saturating counter, reported in the RESTORE_SUCCESS_COUNT0 and RESTORE_SUCCESS_COUNT1 registers, that persists through power loss.

The module may implement a Restore Failure saturating counter, reported in the RESTORE_FAILURE_COUNT0 and RESTORE_FAILURE_COUNT1 registers, that persists through power loss.

The Restore operation is started through the START_RESTORE bit in the NVDIMM_FUNC_CMD register (see 8.1.4.3).

During the Restore operation, the module shall program the RCD Control Words and SDRAM Mode Registers to values compatible with the module memory controller as described in A.2.1 and A.2.2 and copies the previously saved SDRAM contents from NVM into the SDRAM. The module shall complete the operation within the time indicated in the RESTORE_TIMEOUT0 and RESTORE_TIMEOUT1 registers.

Upon completion of the Restore operation, the module shall:

- put the SDRAMs into Self-Refresh mode (see JESD79-4B (DDR4 SDRAM));
- clear the Restore In Progress bit in the NVDIMM_CMD_STATUS0 register;
- update the RESTORE_STATUS register (see 8.1.5.5);
- if the operation was successful, increment the Restore Success counter, if any; and
- if the operation was not successful, increment the Restore Failure counter, if any.

After a Restore operation is completed, the host should take the SDRAMs on the module out of Self-Refresh mode and reprogram the RCD Control Words and SDRAM Mode Registers as described in A.2.3.

7.2.3 Erase operation

Modules shall support the Erase operation.

The module may implement an Erase Success saturating counter, reported in the `ERASE_SUCCESS_COUNT0` and `ERASE_SUCCESS_COUNT1` registers, that persists through power loss.

The module may implement an Erase Failure saturating counter, reported in the `ERASE_FAILURE_COUNT0` and `ERASE_FAILURE_COUNT1` registers, that persists through power loss.

The Erase operation is started through the `START_ERASE` bit in the `NVDIMM_FUNC_CMD` register (see 8.1.4.3) or by the Catastrophic Save operation (see 7.2.1).

During the Erase operation, the module shall erase the last saved SDRAM image in NVM and clear the `NVM_Data_Valid` bit in the `CSAVE_INFO` register (see 8.1.6.1). If the NVM does not contain any saved SDRAM image, the Erase operation shall still succeed. The module shall complete the operation within the time indicated in the `ERASE_TIMEOUT0` and `ERASE_TIMEOUT1` registers.

Upon completion of the Erase operation, the module shall:

- clear the Erase In Progress bit in the `NVDIMM_CMD_STATUS0` register;
- update the `ERASE_STATUS` register (see 8.1.5.6);
- if the operation was successful, increment the Erase Success counter, if any; and
- if the operation was not successful, increment the Erase Failure counter, if any.

7.2.4 Arm operation

Modules shall support the Arm operation.

The module may implement an Arm Success saturating counter, reported in the `ARM_SUCCESS_COUNT0` and `ARM_SUCCESS_COUNT1` registers, that persists through power loss.

The module may implement an Arm Failure saturating counter, reported in the `ARM_FAILURE_COUNT0` and `ARM_FAILURE_COUNT1` registers, that persists through power loss.

When starting the Arm operation, the host specifies the trigger(s) to enable or disable through the `ARM_CMD` register (see 8.1.4.4).

During the Arm operation, the module shall either enable or disable specific triggers for the Catastrophic Save operation. If the host is attempting to enable a trigger, the module shall fail the Arm operation if it detects any conditions that would cause the module to fail to persist the SDRAM content during a Catastrophic Save operation. The module shall also verify that the host has set the Energy Source policy on the module. If the host has not set the Energy Source policy on the module, the module shall fail the Arm operation and set the `ES_POLICY_NOT_SET` bit in the `MODULE_HEALTH_STATUS1` register (see 8.1.8.3) if the module supports more than one Energy Source policy. While the module is armed for the Catastrophic Save operation, the module shall mask the `RESET_n` pin on the module. While the module is not armed for Catastrophic Save operation, the module shall unmask the `RESET_n` pin on the module. The module shall complete the operation within the time indicated in the `ARM_TIMEOUT0` and `ARM_TIMEOUT1` registers. The module shall fail the Arm operation if the persistency cannot be maintained during a power loss.

Upon completion of the Arm operation, the module shall:

- clear the Arm In Progress bit in the NVDIMM_CMD_STATUS0 register;
- update the ARM_STATUS register (see 8.1.5.7);
- if the operation was successful, increment the Arm Success counter, if any; and
- if the operation was not successful, increment the Arm Failure counter, if any.

The Arm operation has no impact on a Catastrophic Save operation started through the START_CSAVE bit in the NVDIMM_FUNC_CMD register.

7.2.5 Management operations

Modules shall support Management Operations. Management Operations are operations that allow the host to reset the controller (see 7.2.10), command the module to not assert the EVENT_n pin or clear specific or all operation status registers on the module.

Management operations are started through the NVDIMM_MGT_CMD0 or NVDIMM_MGT_CMD1 registers (see 8.1.4.1 and 8.1.4.2).

7.2.6 Set Energy Source Policy operation

Modules may support the Set Energy Source Policy operation.

During the Set Energy Source Policy operation, the module shall configure the Energy Source (see 7.2.13) to use during the Catastrophic Save operation. If the module supports more than one Energy Source policy, it shall support the Set Energy Source Policy operation. For modules that support the Set Energy Source Policy operation, the host should execute this operation before the Arm operation. A module can only be configured to one Energy Source policy.

Upon completion of the Set Energy Source Policy operation, the module shall update the SET_ES_POLICY_STATUS register (see 8.1.5.10).

7.2.7 Set Event Notification operation

Modules shall support the Set Event Notification operation.

When starting the Set Event Notification operation, the host specifies the event(s) for which notification shall be enabled or disabled through the SET_EVENT_NOTIFICATION_CMD register (see 8.1.4.5).

During the Set Event Notification operation, the module shall either enable or disable notification about specified events (see 7.4).

Upon completion of the Set Event Notification operation, the module shall update the SET_EVENT_NOTIFICATION_STATUS register (see 8.1.5.9).

7.2.8 Firmware operations

Modules shall support Firmware Operations. Firmware Operations are operations related to firmware update and consists of the following operations:

- Set Firmware Update mode
- Clear Firmware operation
- Generate Firmware Checksum operation
- Commit Firmware operation
- Validate Firmware Header operation
- Validate Firmware Image operation

The module may implement a Firmware Ops Success saturating counter, reported in the FIRMWARE_SUCCESS_COUNT0 and FIRMWARE_SUCCESS_COUNT1 registers, that persists through power loss.

The module may implement a Firmware Ops Failure saturating counter, reported in the FIRMWARE_FAILURE_COUNT0 and FIRMWARE_FAILURE_COUNT1 registers, that persists through power loss.

To start a Firmware operation, the host sets the appropriate bit in the FIRMWARE_OPS_CMD register (see 8.1.4.7).

To update firmware, the host should first enable Firmware Update mode on the module. If the host has not enabled Firmware Update mode on the module, the module shall fail all Firmware Operations except the Set Firmware Update mode operation. While the module is in Firmware Update mode, the module shall fail all other commands except the Abort command.

The module shall complete the Firmware operation within the time indicated in the FIRMWARE_OPS_TIMEOUT0 and FIRMWARE_OPS_TIMEOUT1 registers.

During a Clear Firmware operation, the module shall set the firmware image data region to 0.

During a Generate Firmware Checksum operation, the module shall calculate the checksum of the contents of the firmware image data region and return the checksum value in the FW_REGION_CRC0 and FW_REGION_CRC1 registers. The checksum shall be calculated using the algorithm described in 7.7.

During a Commit Firmware operation, the module shall commit the data in the firmware image data region to the corresponding location in the module where the firmware for slot 1 is stored.

During a Validate Firmware Header operation, the module shall validate that common firmware header is applicable to the module. The module shall validate that the common header transferred has the correct data. If it does not, the module shall fail the operation. If the common header is valid, the module shall persist at least the FIRMWARE IMAGE SIZE and FIRMWARE IMAGE CHECKSUM fields from the common header. These fields shall be used to determine whether a firmware image is valid or not.

During a Validate Firmware Image operation, the module shall validate that the size and calculated checksum of the firmware image in slot 1 matches the size and checksum in the common firmware image header. The module may do additional checks to verify the firmware image in slot 1 is valid. The module shall validate that the firmware image data size matches the FIRMWARE IMAGE SIZE value in the common header and the calculated checksum of the firmware image data matches the FIRMWARE IMAGE CHECKSUM value in the common header. The module may do additional checking to verify the

firmware image committed is the correct image. If an incorrect image has been committed, the module shall fail the Firmware operation. If a correct image has been committed, the module shall update the SLOT1_FWREV0, SLOT1_FWREV1, SLOT1_SUBFWREV, SLOT1_ES_FWREV0, and SLOT1_ES_FWREV1 registers with the appropriate values. If the firmware image in slot 1 is validated, the module shall update the following registers with the revision of the firmware image in slot 1:

- SLOT1_FWREV0 and SLOT1_FWREV1;
- SLOT1_ES_FWREV0 and SLOT1_ES_FWREV1, if there is an energy source firmware image; and
- SLOT1_SUBFWREV, if there is a subcomponent firmware image.

Upon completion of a Firmware operation, the module shall:

- clear the Firmware Ops In Progress bit in the NVDIMM_CMD_STATUS0 register;
- update the FIRMWARE_OPS_STATUS register (see 8.1.5.11);
- if the operation was successful, increment the Firmware Ops Success counter, if any; and
- if the operation was not successful, increment the Firmware Ops Failure counter, if any.

7.2.9 Factory Default operation

Modules shall support the Factory Default operation which returns the module to the factory default state.

The Factory Default operation is started through the START_FACTORY_DEFAULT bit in the NVDIMM_FUNC_CMD register (see 8.1.4.3).

The module may implement a Factory Default Success saturating counter, reported in the FACTORY_DEFAULT_SUCCESS_COUNT0 and FACTORY_DEFAULT_SUCCESS_COUNT1 registers, that persists through power loss.

The module may implement a Factory Default Failure saturating counter, reported in the FACTORY_DEFAULT_FAILURE_COUNT0 and FACTORY_DEFAULT_FAILURE_COUNT1 registers, that persists through power loss.

During the Factory Default operation, the module shall:

- erase all data in the NVM except the data needed to determine warranty compliance;
- disable all event notifications and enabled Catastrophic Save triggers;
- clear the Energy Source policy on the module;
- reset all readable registers to their default values except the registers needed to determine warranty compliance. The register descriptions in Clause 8 define which registers are not impacted by the Factory Default operation;
- not activate new firmware, if any;
- not disrupt the DDR4 bus (e.g., interfere with host access to SDRAMs and the RCD);
- complete the operation within the time indicated in the FACTORY_DEFAULT_TIMEOUT0 and FACTORY_DEFAULT_TIMEOUT1 registers.

The Factory Default operation shall not affect the firmware on the module.

Upon completion of the Factory Default operation, the module shall:

- clear the Factory Default In Progress bit in the NVDIMM_CMD_STATUS0 register;
- update the FACTORY_DEFAULT_STATUS register (see 8.1.5.8);
- if the operation was successful, increment the Factory Default Success counter, if any; and
- if the operation was not successful, increment the Factory Default Failure counter, if any.

7.2.10 Reset Controller operation

Modules shall support the Reset Controller command which resets the controller.

The Reset Controller operation is started through the NVDIMM_MGT_CMD0 register (see 8.1.4.1).

During the Reset Controller operation, the module shall:

- activate new firmware, if any (i.e., honor the FW_SLOT_INFO register SELECT_FW_SLOT field, and activate newly downloaded firmware, if any).

During the Reset Controller operation, if:

- the Reset Controller Scope field (see 8.1.3.2) is set to 0x1; and
- new firmware is activated and the new subcomponent revision equals the previous subcomponent revision,

then the module shall:

- not disrupt the DDR4 bus (e.g., interfere with Host access to SDRAMs and the RCD);
- if armed, latch any SAVE_n pin assertions and respond to them when the module becomes available; and
- become available within the time indicated in the Maximum Non-Volatile Memory Initialization Time field (SPD byte 203)(see JESD21C Page 4.1.2.L-5).

During the Reset Controller operation, the module may:

- stop responding to BAEBI register accesses as described in 8.1.5.1.

During the Reset Controller operation, if:

- the Reset Controller Scope field is not set to 0x1; or
- new firmware is activated and the new subcomponent revision is not equal to the previous subcomponent revision,

then the module may:

- disable all event notifications and enabled Catastrophic Save triggers;
- clear the Energy Source policy on the module;
- reset all readable registers to their default values except the registers needed to determine warranty compliance – the same registers as for the Factory Default operation.

Host software should not send a Reset Controller command to an armed module if the Reset Controller Scope field is not set to 0x1.

Host software should not send a Reset Controller command to an armed module if it has selected a new firmware slot with a different subcomponent revision.

Host software should not send a Reset Controller command to an armed module if:

- the Energy Source policy is Host Managed; and
- the time indicated in the Maximum Non-Volatile Memory Initialization Time field plus the time indicated in the CSAVE_TIMEOUT registers is greater than the amount of time for a Catastrophic Save operation supported by the host.

Upon completion of the Reset Controller operation, the module shall:

- allow access to all registers (see 7.1); and
- set the NVDIMM_READY register (see 8.1.5.1) to 0xA5.

7.2.11 Operational Unit operations

Modules may support Operational Unit operations. Operational Unit Operations are operations related to Typed Block Data and consist of the following operations:

- Get Operational Unit operation
- Set Operational Unit operation
- Clear Operational Unit Buffer operation
- Generate Operational Unit Checksum operation

NOTE: Modules compliant with JESD245 and JESD245A do not support Operational Unit operations.

To start an Operational Unit operation, the host sets the appropriate bit in the OPERATIONAL_UNIT_OPS_CMD register (see 8.1.4.8).

The module may implement an Operational Unit Ops Success saturating counter, reported in the OPERATIONAL_UNIT_SUCCESS_COUNT0 and OPERATIONAL_UNIT_SUCCESS_COUNT1 registers, that persists through power loss.

The module may implement an Operational Unit Ops Failure saturating counter, reported in the OPERATIONAL_UNIT_FAILURE_COUNT0 and OPERATIONAL_UNIT_FAILURE_COUNT1 registers, that persists through power loss.

Upon completion of the Operational Unit operation, the module shall:

- clear the Operational Unit Ops In Progress bit in the NVDIMM_CMD_STATUS1 register;
- update the OPERATIONAL_UNIT_OPS_STATUS register (see 8.1.5.12);
- if the operation was successful, increment the Operational Unit Ops Success counter, if any; and
- if the operation was not successful, increment the Operational Unit Ops Failure counter, if any.

7.2.12 Atomic Arm and Erase operation

Modules may support the Atomic Arm and Erase operation. When the Atomic Arm and Erase operation is supported, the controller will guarantee that the Arm and Erase commands will be done in a single operation before the operation is reported as completed. When this capability is used, it is unnecessary to use the Erase operation to erase a valid SDRAM image. This feature ensures continuous persistency for the contents of the module by eliminating a potential loss in persistence race condition between the Arm and Erase operations.

A Catastrophic Save operation may occur during an Atomic Arm and Erase operation. If a Catastrophic Save trigger is detected, as defined in 8.1.8.7, during the Atomic Arm and Erase operation, the controller shall pause the Atomic Arm and Erase operation and disconnect the module from the platform command/address bus. After the module has been disconnected from the platform command/address bus, the module shall continue to service the Atomic Arm and Erase operation and then start a Catastrophic Save operation.

7.2.13 Abort operation

Modules shall support the Abort operation for Catastrophic Save, Restore, Erase, Arm, Firmware, Factory Default, and Operational Unit operations.

The Abort operation is started through the ABORT_CURRENT_OP bit in the NVDIMM_FUNC_CMD register (see 8.1.4.3). The module shall complete the operation within the time indicated in the ABORT_CMD_TIMEOUT register.

During the Abort operation, the module shall stop the currently running Catastrophic Save, Restore, Erase, Arm, Firmware, Factory Default, or Operational Unit operation.

The host should use the Abort operation if it detects that a pending operation has exceeded its worst case completion latency.

7.2.13.1 Aborting a Catastrophic Save operation

When a Catastrophic Save operation is aborted, the module shall clear the NVM_Data_Valid bit in the CSAVE_INFO register (see 8.1.6.1). The contents of SDRAM saved to NVM are indeterminate. The module is responsible for doing any cleanup of the NVM used.

When a Catastrophic Save operation is aborted, the module shall set the ABORT_SUCCESS bit and the CSAVE_ERROR bit in the CSAVE_STATUS register (see 8.1.5.4).

7.2.13.2 Aborting a Restore operation

When a Restore operation is aborted, the contents of SDRAM are indeterminate.

When a Restore operation is aborted, the module shall set the ABORT_SUCCESS bit and the RESTORE_ERROR bit in the RESTORE_STATUS register (see 8.1.5.5).

7.2.13.3 Aborting an Erase operation

When an Erase operation is aborted, the module shall:

- set the ABORT_SUCCESS bit and the ARM_ERROR bit in the ARM_STATUS register (see 8.1.5.6).

7.2.13.4 Aborting an Arm operation

When an Arm operation is aborted, the contents of SDRAM are indeterminate.

When an Arm operation is aborted, the module shall set the ABORT_SUCCESS bit and the RESTORE_ERROR bit in the ARM_STATUS register (see 8.1.5.7).

7.2.13.5 Aborting Firmware operations

The following Firmware operations can be aborted:

- Clear Firmware operation
- Generate Firmware Checksum operation
- Commit Firmware operation
- Validate Firmware Image operation

When a Clear Firmware operation is aborted, the contents of the firmware data region are indeterminate.

When a Generate Firmware Checksum operation is aborted, the contents of the FW_REGION_CRC0 and FW_REGION_CRC1 registers shall not be changed.

When a Commit Firmware operation is aborted, the contents of firmware slot 1 are indeterminate. The SLOT1_FWREV0, SLOT1_FWREV1, SLOT1_ES_FWREV0, SLOT1_ES_FWREV1, and SLOT1_SUBFWREV registers shall be set to 0 to indicate an invalid firmware image is present.

When a Firmware operation is aborted, the module shall set the ABORT_SUCCESS bit and the FIRMWARE_OPS_ERROR bit in the FIRMWARE_OPS_STATUS register (see 8.1.5.11).

7.2.13.6 Aborting a Factory Default operation

When a Factory Default operation is aborted, the module shall set the ABORT_SUCCESS bit and the FACTORY_DEFAULT_ERROR bit in the FACTORY_DEFAULT_STATUS register (see 8.1.5.8).

7.2.13.7 Aborting Operational Unit operations

When an Operational Unit operation is aborted, the module shall set the ABORT_SUCCESS bit and the OPERATIONAL_UNIT_OPS_ERROR bit in the OPERATIONAL_UNIT_OPS_STATUS register (see 8.1.5.12).

7.3 Energy Source

A Byte Addressable Energy Backed Interface module requires an Energy Source in order to function correctly. This section describes the various Energy Sources that may be supported by a module.

The module may implement an Energy Source Runtime Hours saturating counter, reported in the ES_RUNTIME0 and ES_RUNTIME1 registers, that persists through power loss and measures the time in hours that the Energy Source has been operational since the Energy Source was manufactured.

7.3.1 Local Energy Source

A local Energy Source is an Energy Source that is located on the module itself. A module uses the local Energy Source in Device Managed Policy.

7.3.2 Tethered Energy Source

A tethered Energy Source is an Energy Source that is connected to the module through a connector on the module. A module uses a tethered Energy Source in Device Managed Policy.

7.3.3 Host Energy Source

A host Energy Source is an Energy Source that is located on the host. The energy from a Host Energy Source is provided to the module through the V₁₂ pin. A module uses a host Energy Source in Host Managed Policy.

7.3.4 Shared Energy Source

A shared Energy Source is an Energy Source that is shared across multiple modules. Shared Energy Source may be used when the module is in either Device Managed Policy or Host Managed Policy.

7.3.5 Energy Source Policy

The module shall support at least one of the following Energy Source policies and may support both Energy Source policies:

- **Device Managed:** The module is responsible for monitoring the health of the Energy Source and ensuring there is sufficient energy to support a Catastrophic Save operation. The Energy Source used is either located on the module or tethered to the module. A tethered Energy Source may be dedicated to the module or shared between modules. The module shall not draw power from the V₁₂ pin during a Catastrophic Save operation. A module that supports the Device Managed Policy shall support page 1.
- **Host Managed:** The host is responsible for monitoring the health of the Energy Source and ensuring there is sufficient energy to support a Catastrophic Save operation. The Energy Source used is from the DIMM connector and may be dedicated to the module or shared between modules.

In Device Managed mode, the module shall communicate information about the Energy Source used. The Energy Source may be located on the module or tethered to the module.

7.4 Event Notification

Modules may support asynchronous notification for events that require the host's attention. Event notification is implemented through the EVENT_n pin on the DDR4 bus. An enabled event occurs when the value of the state changes (i.e., edge type event and not level type event). When an enabled event occurs, the module shall drive the EVENT_n pin low until the corresponding event notification is disabled or the host commands the module to deassert the EVENT_n pin.

If the module supports notification, the module shall support notification for persistency and warning threshold events. A module may support notification for vendor specific events.

When an event triggers, the host should do the following:

1. Set bit 2 (Deassert the EVENT_n pin) in the NVDIMM_MGT_CMD1 register.
2. Read the MODULE_HEALTH register to determine the high level event source for the notification.
3. If the event is caused by PERSISTENCY_LOST_ERROR, read the MODULE_HEALTH_STATUS0, MODULE_HEALTH_STATUS1 and ERROR_THRESHOLD_STATUS registers for the specific cause of the PERSISTENCY_LOST_ERROR.
4. If the event is caused by WARNING_THRESHOLD_EXCEEDED, read the WARNING_THRESHOLD_STATUS register to determine the warning threshold that caused the notification. If desired, readjust the warning threshold and enable event notification.
5. If the event is caused by PERSISTENCY_RESTORED, this is to alert the host that the previous error condition is not present anymore. To determine the specific condition(s) that is not present, read the MODULE_HEALTH_STATUS0, MODULE_HEALTH_STATUS1 and ERROR_THRESHOLD_STATUS registers.
6. If event is caused by BELOW_WARNING_THRESHOLD, this is to alert the host that the previous warning condition is not present anymore. To determine the specific condition(s) that is not present, read the WARNING_THRESHOLD_STATUS register.

NOTE Another possible source of EVENT_n pin assertion on the module is the TSE2004av SPD EEPROM with Temperature Sensor, which uses that pin to report thermal events.

7.5 Error and Warning Thresholds

Error thresholds allow the host to know the operational limits of the module. Warning thresholds allow the host to get advance notification of when the module is reaching its operational limit.

All threshold events shall be normalized to a range of 0 to 100% where 100% represents a good status. Module may support event notification if a value exceeds the threshold and if the value meets or fall below the previously exceeded threshold.

Warning thresholds shall have a default value set by the module manufacturer. The module shall allow the host to modify a warning threshold if it is supported. A warning threshold cannot conflict with an error threshold.

To set a warning threshold, the host does the following:

1. Disable the event notification for the desired threshold.
2. Update the threshold value to the desired value.
3. Enable the event notification for the desired threshold.

7.6 Module Firmware

Modules shall support 2 firmware images – slot 0 and slot 1. There shall be a default image at firmware slot 0 that is flashed during manufacturing and not updatable afterwards. Modules shall support updating the firmware in slot 1 only if the host enables firmware update mode. If the host has not enabled firmware update mode on the module, the module shall fail all Firmware operations.

The default image shall be a full functional image.

The host firmware update workflow is described in 9.7.

A firmware image includes a controller firmware image, may contain an energy source firmware image, and may contain a subcomponent (e.g., FPGA) firmware image.

Controller firmware revision values shall be assigned in increasing order so that a larger revision value represents a more recent firmware image.

Subcomponent firmware revision values shall be assigned in increasing order so that a larger revision value represents a more recent subcomponent firmware image.

Energy source firmware revision values shall be assigned in increasing order so that a larger revision value represents a more recent firmware image.

Host software:

- should only send a firmware image to a module if all of the valid fields specifying SPD contents in the firmware image header match the NVDIMM SPD contents;
- should only send a firmware image to a module containing a subcomponent firmware image if the subcomponent firmware revision is different; and
- should only send a firmware image to a module containing an energy source firmware image if the energy source firmware revision is different.

The module should only accept a firmware image if all of the valid fields specifying SPD contents in the firmware image header match the NVDIMM SPD contents.

The firmware image shall begin with a 32 byte common header with the format defined in 7.6 Module Firmware (cont')

Table 2.

7.6 Module Firmware (cont')

Table 2 — Firmware image header – common format

Field Name	Byte Offset(s)	Description
Module Manufacturer ID Code	0 – 1	SPD bytes 320 – 321
Module Product Identifier	2 – 3	SPD bytes 192 – 193
Firmware Image Header Format-specific	4 – 19	See Table 3 or Table 4
Firmware Image Size	4 – 23	Size in bytes of the firmware image including the header.
Firmware Image Checksum	24 – 25	CRC of the firmware image excluding the header. The CRC is calculated using the CRC algorithm described in 7.7.
Reserved	26 – 30	Reserved
Firmware Image Header Format	31	Format of the firmware image header 0x00: firmware image header format 0 0x01: firmware image header format 1 All others: reserved

Table 3 defines firmware image header format 0.

Table 3 — Firmware image header format 0

Field Name	Byte Offset(s)	Description
Module Manufacturer ID Code	0 – 1	See 7.6 Module Firmware (cont') Table 2
Module Product Identifier	2 – 3	See 7.6 Module Firmware (cont') Table 2
Vendor-Defined	4 – 19	Vendor defined 16 byte value
Firmware Image Size	20 – 23	See 7.6 Module Firmware (cont') Table 2
Firmware Image Checksum	24 – 25	See 7.6 Module Firmware (cont') Table 2
Reserved	26 – 30	See 7.6 Module Firmware (cont') Table 2
Firmware Image Header Format	31	0x00

7.6 Module Firmware (cont')

Table 4 defines firmware image header format 1.

Table 4 — Firmware image header format 1

Field Name	Byte Offset(s)	Description
Module Manufacturer ID Code	0 – 1	See 7.6 Module Firmware (cont') Table 2
Module Product Identifier	2 – 3	See 7.6 Module Firmware (cont') Table 2
Non-Volatile Memory Subsystem Controller Manufacturer ID Code	4 – 5	SPD bytes 194-195
Non-Volatile Memory Subsystem Controller Product Identifier	6 – 7	SPD bytes 196-197
Module Revision Code	8	SPD byte 349
Non-Volatile Memory Subsystem Controller Revision Code	9	SPD byte 198
Controller Firmware Revision	10 – 11	Revision of the controller firmware image (see 8.1.2.3, 8.1.2.4, 8.1.2.5, and 8.1.2.6)
Energy Source Firmware Revision	12 – 13	Revision of the energy source firmware image (see 8.2.2.2, 8.2.2.3, 8.2.2.4, 8.2.2.5, 8.2.2.6, and 8.2.2.7), if any
Subcomponent Firmware Revision	14	Revision of the subcomponent firmware image (see 8.1.2.7 and 0), if any
Reserved	15 – 18	Reserved
Valid	19	Bitmask that specifies which fields in the Firmware Image Header are valid. Bit 0: Module Manufacturer ID Code Bit 1: Module Manufacturer Product Identifier Bit 2: Non-Volatile Memory Subsystem Controller Manufacturer ID Code Bit 3: Non-Volatile Memory Subsystem Controller Product Identifier Bit 4: Module Revision Code Bit 5: Non-Volatile Memory Subsystem Controller Revision Code Bit 6: Energy Source Firmware Revision Bit 7: Subcomponent Firmware Revision A field covered by a Valid bit that is not valid may be set to any value.
Firmware Image Size	20 – 23	See 7.6 Module Firmware (cont') Table 2

Firmware Image Checksum	24 – 25	See 7.6 Module Firmware (cont') Table 2
Reserved	26 – 30	See 7.6 Module Firmware (cont') Table 2
Firmware Image Header Format	31	0x01

7.7 CRC Algorithm

The checksum in:

- the firmware image header (see 7.6);
- in the FW_REGION_CRC0 and FW_REGION_CRC1 registers (see 8.1.4.7);
- in the OPERATIONAL_UNIT_CRC0 and OPERATIONAL_UNIT_CRC1 registers (see 8.1.4.8);

shall be calculated using the algorithm described in this subclause. This CRC algorithm is the same algorithm used to calculate the CRC fields in the SPD (see JESD21C Page 4.1.2.L-5).

The following algorithm (shown in C) is to be followed in calculating and checking the code:

```
int Crc16 (char *ptr, int count)
{
    int crc, i;

    crc = 0;
    while (--count >= 0) {
        crc = crc ^ (int)*ptr++ << 8;
        for (i = 0; i < 8; ++i) {
            if (crc & 0x8000) {
                crc = crc << 1 ^ 0x1021;
            } else {
                crc = crc << 1;
            }
        }
    }
    return (crc & 0xFFFF);
}
```

7.8 Error Injection

To enable validation of error handling on the host, error injection support is required. Error injection support shall be available in non-production firmware images. The ONE_TIME_USE bit in the INJECT_ERROR_TYPE register (see 8.3.3.6) determines whether the injected failure is for the next instance of the operation or for all subsequent instances of the operation until the error injection is disabled. The following error injection support shall be supported by the module:

- Force a Catastrophic Save, Restore, Erase, and Arm operation failure.
- Force an internal controller error.
- Force a permanent hardware failure.
- Force a NVM lifetime percentage value lower than the warning and error threshold.
- Inject a certain percentage of errors during writes to the NVM subsystem.
- Force an Energy Source failure if the module is using a local or tethered Energy Source.
- Force an Energy Source assessment failure if the module is using a local or tethered Energy Source.
- Force an Energy Source lifetime percentage value lower than the warning and error threshold if the module is using a local or tethered Energy Source.
- Force an Energy Source temperature beyond the warning and error thresholds if the module is using a local or tethered Energy Source.
- Force a commit error during firmware update.
- Force a firmware image checksum error.
- Force an invalid firmware image error.

Error injection shall take place during the next instance of the applicable operation. Any enabled error injection shall be persisted across power cycles and resets until:

- explicitly disabled by the host;
- or by the Factory Default operation; or
- disabled after being injected one time, if the ONE_TIME_USE bit in the INJECT_ERROR_TYPE register (see 8.3.3.6) is set.

Error injection shall be disabled by default.

7.9 Statistics

Modules may provide statistics on the following operations:

- Duration of the last Catastrophic Save, Restore, Erase, Arm, Factory Default, Firmware, and Operational Unit operations.
- Count of the completed Catastrophic Save, Restore, Erase, Arm, Factory Default, Firmware, and Operational Unit operations over the module lifetime.
- Count of module power cycles over the module lifetime.

The module may implement a Power Cycles saturating counter, reported in the POWER_CYCLE_COUNT0 and POWER_CYCLE_COUNT1 registers, that persists through power loss and measures power cycles over the module lifetime. The module shall increment this counter after successful controller initialization.

Additional statistics may be available in vendor specific registers.

7.10 Thermal

Modules shall include an SPD EEPROM with Temperature Sensor compliant with JESD21C Page 4.1.6-2. Modules may have multiple thermal sensors monitoring temperature of various components on the module. Additional thermal sensors are supported through vendor specific registers.

Modules shall report temperatures in registers defined in this standard with a minimum resolution of 0.25 °C. Temperature values defined in this standard use the format defined in Table 5.

Table 5 — Temperature value bit definition

Bit(s)	Description	Notes
[15:13]	Reserved	
[12]	Sign	0 = positive, 1 = negative The value of 0 °C should be expressed as a positive value.
[11]	128 °C	
[10]	64 °C	
[9]	32 °C	
[8]	16 °C	
[7]	8 °C	
[6]	4 °C	
[5]	2 °C	
[4]	1 °C	
[3]	0.5 °C	
[2]	0.25 °C	
[1]	0.125 °C	Optional for temperature reporting fields; not used for temperature threshold fields
[0]	0.0625 °C	Optional for temperature reporting fields; not used for temperature threshold fields

Examples:

- A temperature value of -40 °C is represented as 0x1280.
- A temperature value of 0 °C is represented as 0x0000 or 0x1000.
- A temperature value of 85 °C is represented as 0x0550.
- A temperature value of 95 °C is represented as 0x05f0.
- A temperature value of 10.5 °C is represented as 0x00a8 (i.e., 0000000010101000b).
- A temperature value of 64.75 °C is represented as 0x040c (i.e., 0000010000001100b).

7.11 Host Area

Modules shall provide persistent storage to hosts to store data that applies to the module but can only be detected by the host (see 8.3.4). Modules do not act upon any of this data.

The host may implement an Uncorrectable Errors saturating counter. The host should increment this counter and write the value to the DRAM_ECC_ERROR_COUNT register every time it detects an uncorrectable ECC error from the SDRAM on the module.

The host may implement a Correctable Errors Threshold Exceeded saturating counter. The host should increment this counter and write the value to the DRAM_THRESHOLD_ECC_COUNT register every time it detects a correctable ECC error from the SDRAM on the module that has exceeded the configured correctable ECC threshold for the module.

The host may implement a Catastrophic Save Workflow Failure saturating counter. The host should increment this counter and write the value to the HOST_CSAVE_WORKFLOW_FAILURE_COUNT0 and HOST_CSAVE_WORKFLOW_FAILURE_COUNT1 registers every time it detects a failure on the host that cause a Catastrophic Save operation to fail or inconsistent data to be saved.

7.12 Vendor Log Page

Modules shall support a Vendor Log Page that provides information useful to triage issues seen on the module. A module may use the VENDOR_LOG_PAGE_SIZE register (see 8.1.3.30) to communicate the number of valid vendor-specific bytes by specifying a non-zero value. If the module uses the VENDOR_LOG_PAGE_SIZE to communicate the number of valid bytes, the size is limited to a maximum value of 8160 bytes.

To read the Vendor Log Page data, the host should follow the Read Typed Block Data workflow (see 9.8.1).

7.13 Vendor-Specific

Modules may implement vendor-specific functionality beyond what is described in this standard. Vendor-specific functionalities are accessed through the vendor specific page(s) that may be supported by a module.

7.14 LCOM Interface

A module controller indicates LCOM interface support by setting status Bit 7 (LCOM Interface Supported) in the CAPABILITIES0 register (see 8.1.3.1). If the optional LCOM interface is supported, the module memory controller can communicate with the RCD through the LCOM interface. LCOM supports the following functions:

- Controller sends command frames to the RCD
- Controller receives read data frames from the RCD

Modules supporting the LCOM interface shall be compliant with JESD82-31A (DDR4RCD02).

The host should write the LCOM_ENABLE bit in the MODULE_OPS_CONFIG register (see 8.1.8.7) as described in the NVDIMM Initialization Sequence defined in JESD82-31A (DDR4RCD02).

7.15 Label Data

Modules shall support the Label Data Typed Block Data type. Label Data is a region on the module that is used by the host to store metadata about how the memory exposed by the module is used. The Operational Unit size for Label Data shall be 256 bytes.

Modules shall support a minimum of 1024 bytes for Label Data. For modules exposing more than 32 GiB of memory, the module shall support at least 256 bytes of additional data for every 32 GiB of memory usable by the host. As an example, a module with 60 GiB of memory usable by the host shall support at least 1280 bytes of Label Data. A module with 150 GiB of memory usable by the host shall support at least 2048 bytes of Label Data.

7.16 Catastrophic Save transition registers

The module may implement registers to assist in the transition from the host to the module for a Catastrophic Save operation.

The host should program:

- the registers in page 4 with the values that the host has programmed into SDRAM Mode Registers without using per DRAM addressing (PDA);
- the registers in page 5 with the values that the host has programmed into SDRAM Mode Registers using PDA;
- the registers in page 6 with the values that the host has programmed into control words in RCD functions 0 through 3;
- the registers in page 7 with the values that the host has programmed into control words in RCD functions 4 through 7;
- the registers in page 8 with the values that the host has programmed into control words in RCD functions 8 through 11; and
- the registers in page 9 with the values that the host has programmed into control words in RCD functions 12 through 15.

Table 6 defines the format of the register at the lowest offset of a pair of registers holding an SDRAM Mode Register value.

Table 6 — Mode register – lower byte

Definition	Access	Mand	Persist	Default
[7:0] : Mode Register bits A7:A0	RW	Y	N	0

Table 7 defines the format of the register at the highest offset of a pair of registers holding an SDRAM Mode Register value.

Table 7 — Mode register – upper byte

Definition	Access	Mand	Persist	Default
[7] : Reserved	RW	Y	N	0
[6] : Mode Register bit A17				
[5:0] : Mode Register bits A13:A8				

8 REGISTER MAP

This clause describes the I²C registers that are defined in each supported page.

The following conventions are used for registers defined in this standard:

- All register values are 0-based unless stated otherwise in the register description.
- In register definition tables, “Access” means host access property (e.g., RW, RO, or WO).
- In register definition tables, “Mand” means mandatory (e.g., Y or N).
- In register definition tables, “Persist” means persistent through power cycles (e.g., Y or N).

The host access properties are:

- RW = read/write. The module accepts writes to the register and returns its contents on reads.
- RO = read-only. The module ignores writes to the register and returns its contents on reads.
- WO = write-only. The module accepts writes to the register and returns 0x00 on reads.

8.1 Page 0 register map

The registers in page 0 are organized based on categories.

Table 8 shows the layout of the categories in page 0.

Table 8 — Page 0 register map categories

Offset	Category	Description
0x00 – 0x03	Paging Mechanism	Registers related to I2C page support
0x04 – 0x0F	Version	Registers providing version information
0x10 – 0x3F	Characteristics	Registers providing characteristics information
0x40 – 0x5F	Runtime Command	Registers related to runtime commands
0x60 – 0x7F	Runtime Command Status	Registers providing runtime command status information
0x80 – 0x87	Catastrophic Save	Registers providing Catastrophic Save information
0x88 – 0x8F	Reserved	Reserved
0x90 – 0x9F	Thresholds	Registers related to thresholds
0xA0 – 0xBF	Module	Registers providing module related information
0xC0 – 0xDF	NVM Subsystem	Registers providing NVM subsystem related information
0xE0 – 0xFF	Reserved	Reserved

Table 9 lists the registers that are in page 0.

Table 9 — Page 0 register map

Offset	Register Name	Access	Mandatory	Persist
0x00	OPEN_PAGE	RW	Y	N
0x01	STD_NUM_PAGES	RO	Y	Y
0x02	VENDOR_START_PAGES	RO	Y	Y
0x03	VENDOR_NUM_PAGES	RO	Y	Y
0x04	HWREV	RO	Y	Y
0x05	Reserved			
0x06	SPECREV	RO	Y	Y
0x07	SLOT0_FWREV0	RO	Y	Y
0x08	SLOT0_FWREV1	RO	Y	Y
0x09	SLOT1_FWREV0	RO	Y	Y
0x0A	SLOT1_FWREV1	RO	Y	Y
0x0B	SLOT0_SUBFWREV	RO	Y	Y
0x0C	SLOT1_SUBFWREV	RO	Y	Y
0x0D-0x0F	Reserved			
0x10	CAPABILITIES0	RO	Y	Y
0x11	CAPABILITIES1	RO	Y	Y
0x12-0x13	Reserved			
0x14	ENERGY_SOURCE_POLICY	RO	Y	Y
0x15	HOST_MAX_OPERATION_RETRY	RO	Y	Y
0x16	CSAVE_TRIGGER_SUPPORT	RO	Y	Y
0x17	EVENT_NOTIFICATION_SUPPORT	RO	Y	Y
0x18	CSAVE_TIMEOUT0	RO	Y	Y
0x19	CSAVE_TIMEOUT1	RO	Y	Y
0x1A	PAGE_SWITCH_LATENCY0	RO	Y	Y
0x1B	PAGE_SWITCH_LATENCY1	RO	Y	Y
0x1C	RESTORE_TIMEOUT0	RO	Y	Y
0x1D	RESTORE_TIMEOUT1	RO	Y	Y
0x1E	ERASE_TIMEOUT0	RO	Y	Y
0x1F	ERASE_TIMEOUT1	RO	Y	Y
0x20	ARM_TIMEOUT0	RO	Y	Y
0x21	ARM_TIMEOUT1	RO	Y	Y
0x22	FIRMWARE_OPS_TIMEOUT0	RO	Y	Y
0x23	FIRMWARE_OPS_TIMEOUT1	RO	Y	Y
0x24	ABORT_CMD_TIMEOUT	RO	Y	Y
0x25	Obsolete			
0x26	Obsolete			
0x27	MAX_RUNTIME_POWER0	RO	N	Y
0x28	MAX_RUNTIME_POWER1	RO	N	Y
0x29	CSAVE_POWER_REQ0	RO	N	Y
0x2A	CSAVE_POWER_REQ1	RO	N	Y
0x2B	CSAVE_IDLE_POWER_REQ0	RO	N	Y
0x2C	CSAVE_IDLE_POWER_REQ1	RO	N	Y
0x2D	CSAVE_MIN_VOLT_REQ0	RO	N	Y
0x2E	CSAVE_MIN_VOLT_REQ1	RO	N	Y
0x2F	CSAVE_MAX_VOLT_REQ0	RO	N	Y
0x30	CSAVE_MAX_VOLT_REQ1	RO	N	Y
0x31	VENDOR_LOG_PAGE_SIZE	RO	N	Y
0x32	REGION_BLOCK_SIZE	RO	Y	Y

0x33	OPERATIONAL_UNIT_OPS_TIMEOUT0	RO	Y	Y
0x34	OPERATIONAL_UNIT_OPS_TIMEOUT1	RO	Y	Y
0x35	FACTORY_DEFAULT_TIMEOUT0	RO	Y	Y
0x36	FACTORY_DEFAULT_TIMEOUT1	RO	Y	Y
0x37	Reserved			
0x38	MIN_OPERATING_TEMP0	RO	Y	Y
0x39	MIN_OPERATING_TEMP1	RO	Y	Y
0x3A	MAX_OPERATING_TEMP0	RO	Y	Y
0x3B	MAX_OPERATING_TEMP1	RO	Y	Y
0x3C-0x3F	Reserved			
0x40	NVDIMM_MGT_CMD0	WO	Y	N
0x41	NVDIMM_MGT_CMD1	WO	Y	N
0x42	Reserved			
0x43	NVDIMM_FUNC_CMD	WO	Y	N
0x44	Reserved			
0x45	ARM_CMD	WO	Y	N
0x46	Reserved			
0x47	SET_EVENT_NOTIFICATION_CMD	WO	Y	N
0x48	Reserved			
0x49	SET_ES_POLICY_CMD	WO	N	N
0x4A	FIRMWARE_OPS_CMD	WO	Y	N
0x4B	OPERATIONAL_UNIT_OPS_CMD	WO	Y	N
0x4C-0x5F	Reserved			
0x60	NVDIMM_READY	RO	Y	N
0x61	NVDIMM_CMD_STATUS0	RO	Y	N
0x62	NVDIMM_CMD_STATUS1	RO	Y	N
0x63	Reserved			
0x64	CSAVE_STATUS	RO	Y	Y
0x65	Reserved			
0x66	RESTORE_STATUS	RO	Y	N
0x67	Reserved			
0x68	ERASE_STATUS	RO	Y	N
0x69	Reserved			
0x6A	ARM_STATUS	RO	Y	N
0x6B	Reserved			
0x6C	FACTORY_DEFAULT_STATUS	RO	Y	N
0x6D	Reserved			
0x6E	SET_EVENT_NOTIFICATION_STATUS	RO	Y	N
0x6F	Reserved			
0x70	SET_ES_POLICY_STATUS	RO	Y	N
0x71	FIRMWARE_OPS_STATUS	RO	Y	N
0x72	OPERATIONAL_UNIT_OPS_STATUS	RO	Y	N
0x73-0x7F	Reserved			
0x80	CSAVE_INFO	RO	Y	Y
0x81-0x83	Reserved			
0x84	CSAVE_FAIL_INFO0	RO	Y	Y
0x85	CSAVE_FAIL_INFO1	RO	Y	Y
0x86-0x87	Reserved			
0x88	RESTORE_FAIL_INFO	RO	Y	N
0x89-0x8E	Reserved			
0x8F	OPERATIONAL_UNIT_OPS_FAIL_INFO	RO	Y	N
0x90	NVM_LIFETIME_ERROR_THRESHOLD	RO	Y	Y