

JEDEC STANDARD

Byte Addressable Energy Backed Interface

JESD245E

(Revision of JESD245D, July 2020)

APRIL 2022

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 2022
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.

This page left intentionally blank

BYTE ADDRESSABLE ENERGY BACKED INTERFACE**Contents**

	Page
Foreword	xii
Introduction	xii
1 Scope	1
2 Normative References	1
2.1 Overview	1
2.2 JEDEC Standards	1
2.3 Management Interface Standards	2
2.4 NIST (National Institute of Standards and Technology) Standards	2
2.5 NIST CAVP (Cryptographic Algorithm Validation Program)	3
2.6 NIST CMVP (Cryptographic Module Validation Program)	3
2.7 IEEE Standards	3
2.8 Common Criteria Standards	3
2.9 TCG (Trusted Computing Group) Standards	4
2.10 Best Practices	5
3 Terms and Definitions	5
3.1 Acronyms	6
3.2 Terms and Definitions	6
3.3 Keywords	8
3.4 Conventions	9
4 Introduction	10
5 I2C	11
5.1 I ² C Bus	11
5.2 I ² C Transactions	11
5.2.1 I ² C Read Byte Transaction	12
5.2.2 I ² C Write Byte Transaction	12
5.2.3 I ² C Read Word Transaction	12
5.2.4 I ² C Write Word Transaction	13
5.2.5 I ² C Block Read Transaction	13
5.2.6 I ² C Block Write Transaction	13
5.2.7 I ² C Transactions to Reserved Offsets	14
5.2.8 Illegal I ² C Transactions	14
5.2.9 I ² C Resets	14
5.3 Paging	14
5.4 Typed Block Data	15
6 Serial Presence Detect	16
6.1 Byte Addressable Energy Backed Interface Discovery	16
7 Features	17
7.1 Controller Ready	17
7.2 Operations	18
7.2.1 Catastrophic Save Operation	18
7.2.1.1 START_CSAVE Bit	19
7.2.1.2 SAVE_n Pin	19
7.2.1.3 RESET_n Pin	21

Contents (cont'd)

7.2.2	Restore Operation	21
7.2.3	Erase Operation.....	22
7.2.4	Arm Operation	23
7.2.5	Management Operations	24
7.2.6	Set Energy Source Policy Operation.....	24
7.2.7	Set Event Notification Operation.....	24
7.2.8	Firmware Operations	24
7.2.9	Factory Default Operation	26
7.2.10	Reset Controller Operation	26
7.2.11	Operational Unit Operations	27
7.2.12	Atomic Save and Erase Operation	28
7.2.13	Abort Operation	28
7.2.13.1	Aborting a Catastrophic Save Operation.....	28
7.2.13.2	Aborting a Restore Operation	29
7.2.13.3	Aborting an Erase Operation.....	29
7.2.13.4	Aborting an Arm Operation	29
7.2.13.5	Aborting Firmware Operations	29
7.2.13.6	Aborting a Factory Default Operation	29
7.2.13.7	Aborting Operational Unit Operations	29
7.3	Energy Source	30
7.3.1	Local Energy Source.....	30
7.3.2	Tethered Energy Source.....	30
7.3.3	Host Energy Source	30
7.3.4	Shared Energy Source.....	30
7.3.5	Energy Source Policy.....	30
7.4	Event Notification	31
7.5	Error and Warning Thresholds.....	31
7.6	Module Firmware.....	32
7.7	CRC Algorithm	34
7.8	Error Injection.....	35
7.9	Statistics	36
7.10	Thermal	36
7.11	Host Area	37
7.12	Vendor Log Page	37
7.13	Vendor-Specific	37
7.14	LCOM Interface.....	37
7.15	Label Data.....	38
7.16	Catastrophic Save Transition Registers	38
7.17	Security Protocol Data	39
8	Register Map	39
8.1	Page 0 Register Map	39
8.1.1	Paging Mechanism Registers	44
8.1.1.1	OPEN_PAGE – Offset 0x00.....	44
8.1.1.2	STD_NUM_PAGES – Offset 0x01	44
8.1.1.3	VENDOR_START_PAGES – Offset 0x02.....	44
8.1.1.4	VENDOR_NUM_PAGES – Offset 0x03	44
8.1.2	Version Registers	44

Contents (cont'd)

8.1.2.1	HWREV – Offset 0x04	44
8.1.2.2	SPECREV – Offset 0x06	45
8.1.2.3	SLOT0_FWREV0 – Offset 0x07	45
8.1.2.4	SLOT0_FWREV1 – Offset 0x08	45
8.1.2.5	SLOT1_FWREV0 – Offset 0x09	45
8.1.2.6	SLOT1_FWREV1 – Offset 0x0A	46
8.1.2.7	SLOT0_SUBFWREV – Offset 0x0B	46
8.1.2.8	SLOT1_SUBFWREV – Offset 0x0C	46
8.1.3	Characteristics Registers	46
8.1.3.1	CAPABILITIES0 – Offset 0x10	46
8.1.3.2	CAPABILITIES1 – Offset 0x11	47
8.1.3.3	ENERGY_SOURCE_POLICY – Offset 0x14	48
8.1.3.4	HOST_MAX_OPERATION_RETRY – Offset 0x15	48
8.1.3.5	CSAVE_TRIGGER_SUPPORT – Offset 0x16	48
8.1.3.6	EVENT_NOTIFICATION_SUPPORT – Offset 0x17	49
8.1.3.7	CSAVE_TIMEOUT0 – Offset 0x18	49
8.1.3.8	CSAVE_TIMEOUT1 – Offset 0x19	49
8.1.3.9	PAGE_SWITCH_LATENCY0 – Offset 0x1A	50
8.1.3.10	PAGE_SWITCH_LATENCY1 – Offset 0x1B	50
8.1.3.11	RESTORE_TIMEOUT0 – Offset 0x1C	50
8.1.3.12	RESTORE_TIMEOUT1 – Offset 0x1D	50
8.1.3.13	ERASE_TIMEOUT0 – Offset 0x1E	51
8.1.3.14	ERASE_TIMEOUT1 – Offset 0x1F	51
8.1.3.15	ARM_TIMEOUT0 – Offset 0x20	51
8.1.3.16	ARM_TIMEOUT1 – Offset 0x21	51
8.1.3.17	FIRMWARE_OPS_TIMEOUT0 – Offset 0x22	52
8.1.3.18	FIRMWARE_OPS_TIMEOUT1 – Offset 0x23	52
8.1.3.19	ABORT_CMD_TIMEOUT – Offset 0x24	52
8.1.3.20	MAX_RUNTIME_POWER0 – Offset 0x27	52
8.1.3.21	MAX_RUNTIME_POWER1 – Offset 0x28	53
8.1.3.22	CSAVE_POWER_REQ0 – Offset 0x29	53
8.1.3.23	CSAVE_POWER_REQ1 – Offset 0x2A	53
8.1.3.24	CSAVE_IDLE_POWER_REQ0 – Offset 0x2B	53
8.1.3.25	CSAVE_IDLE_POWER_REQ1 – Offset 0x2C	54
8.1.3.26	CSAVE_MIN_VOLT_REQ0 – Offset 0x2D	54
8.1.3.27	CSAVE_MIN_VOLT_REQ1 – Offset 0x2E	54
8.1.3.28	CSAVE_MAX_VOLT_REQ0 – Offset 0x2F	54
8.1.3.29	CSAVE_MAX_VOLT_REQ1 – Offset 0x30	55
8.1.3.30	VENDOR_LOG_PAGE_SIZE – Offset 0x31	55
8.1.3.31	REGION_BLOCK_SIZE – Offset 0x32	55
8.1.3.32	OPERATIONAL_UNIT_OPS_TIMEOUT0 – Offset 0x33	55
8.1.3.33	OPERATIONAL_UNIT_OPS_TIMEOUT1 – Offset 0x34	56
8.1.3.34	FACTORY_DEFAULT_TIMEOUT0 – Offset 0x35	56
8.1.3.35	FACTORY_DEFAULT_TIMEOUT1 – Offset 0x36	56
8.1.3.36	MIN_OPERATING_TEMP0 – Offset 0x38	56
8.1.3.37	MIN_OPERATING_TEMP1 – Offset 0x39	57
8.1.3.38	MAX_OPERATING_TEMP0 – Offset 0x3A	57

Contents (cont'd)

8.1.3.39	MAX_OPERATING_TEMP1 – Offset 0x3B.....	57
8.1.4	Runtime Command Registers	57
8.1.4.1	NVDIMM_MGT_CMD0 – Offset 0x40.....	58
8.1.4.2	NVDIMM_MGT_CMD1 – Offset 0x41.....	59
8.1.4.3	NVDIMM_FUNC_CMD – Offset 0x43.....	59
8.1.4.4	ARM_CMD – Offset 0x45	60
8.1.4.5	SET_EVENT_NOTIFICATION_CMD – Offset 0x47	61
8.1.4.6	SET_ES_POLICY_CMD – Offset 0x49	61
8.1.4.7	FIRMWARE_OPS_CMD – Offset 0x4A.....	62
8.1.4.8	OPERATIONAL_UNIT_OPS_CMD – Offset 0x4B	62
8.1.4.9	SECURITY_PROTOCOL_SPECIFIC0 – Offset 0x4C	63
8.1.4.10	SECURITY_PROTOCOL_SPECIFIC1 – Offset 0x4D	63
8.1.4.11	SECURITY_PROTOCOL_TYPE – Offset 0x4E.....	63
8.1.5	Runtime Command Status Registers.....	63
8.1.5.1	NVDIMM_READY – Offset 0x60.....	63
8.1.5.2	NVDIMM_CMD_STATUS0 – Offset 0x61	64
8.1.5.3	NVDIMM_CMD_STATUS1 – Offset 0x62	64
8.1.5.4	CSAVE_STATUS – Offset 0x64	65
8.1.5.5	RESTORE_STATUS – Offset 0x66.....	65
8.1.5.6	ERASE_STATUS – Offset 0x68	66
8.1.5.7	ARM_STATUS – Offset 0x6A.....	66
8.1.5.8	FACTORY_DEFAULT_STATUS – Offset 0x6C	67
8.1.5.9	SET_EVENT_NOTIFICATION_STATUS – Offset 0x6E.....	67
8.1.5.10	SET_ES_POLICY_STATUS – Offset 0x70.....	68
8.1.5.11	FIRMWARE_OPS_STATUS – Offset 0x71	68
8.1.5.12	OPERATIONAL_UNIT_OPS_STATUS – Offset 0x72.....	69
8.1.5.13	ERASE_FAIL_INFO Register – Offset 0x73.....	69
8.1.5.14	FACTORY_DEFAULT_STATUS_FAIL_INFO Register – Offset 0x74	70
8.1.5.15	FIRMWARE_OPS_FAIL_INFO Register – Offset 0x75	70
8.1.5.16	ARM_FAIL_INFO register – Offset 0x76.....	70
8.1.5.17	RESTORE_FAIL_INFO – Offset 0x88.....	71
8.1.5.18	OPERATIONAL_UNIT_FAIL_INFO – Offset 0x8F	71
8.1.6	Catastrophic Save Registers.....	72
8.1.6.1	CSAVE_INFO – Offset 0x80	72
8.1.6.2	CSAVE_FAIL_INFO0 – Offset 0x84	73
8.1.6.3	CSAVE_FAIL_INFO1 – Offset 0x85	74
8.1.7	Thresholds Registers.....	74
8.1.7.1	NVM_LIFETIME_ERROR_THRESHOLD – Offset 0x90	75
8.1.7.2	ES_LIFETIME_ERROR_THRESHOLD – Offset 0x91	75
8.1.7.3	ES_TEMP_ERROR_HIGH_THRESHOLD0 – Offset 0x94	75
8.1.7.4	ES_TEMP_ERROR_HIGH_THRESHOLD1 – Offset 0x95	75
8.1.7.5	ES_TEMP_ERROR_LOW_THRESHOLD0 – Offset 0x96	76
8.1.7.6	ES_TEMP_ERROR_LOW_THRESHOLD1 – Offset 0x97	76
8.1.7.7	NVM_LIFETIME_WARNING_THRESHOLD – Offset 0x98	76
8.1.7.8	ES_LIFETIME_WARNING_THRESHOLD – Offset 0x99.....	76
8.1.7.9	ES_TEMP_WARNING_HIGH_THRESHOLD0 – Offset 0x9C.....	77
8.1.7.10	ES_TEMP_WARNING_HIGH_THRESHOLD1 – Offset 0x9D.....	77

Contents (cont'd)

8.1.7.11	ES_TEMP_WARNING_LOW_THRESHOLD0 – Offset 0x9E	77
8.1.7.12	ES_TEMP_WARNING_LOW_THRESHOLD1 – Offset 0x9F	77
8.1.8	Module Registers	77
8.1.8.1	MODULE_HEALTH – Offset 0xA0	78
8.1.8.2	MODULE_HEALTH_STATUS0 – Offset 0xA1	79
8.1.8.3	MODULE_HEALTH_STATUS1 – Offset 0xA2	79
8.1.8.4	ERROR_THRESHOLD_STATUS – Offset 0xA5	80
8.1.8.5	WARNING_THRESHOLD_STATUS – Offset 0xA7	81
8.1.8.6	AUTO_ES_HEALTH_FREQUENCY – Offset 0xA9	81
8.1.8.7	MODULE_OPS_CONFIG – Offset 0xAA	82
8.1.9	NVM Subsystem registers	83
8.1.9.1	NVM_LIFETIME – Offset 0xC0	83
8.2	Page 1 Register Map	83
8.2.1	Paging Mechanism Registers	85
8.2.1.1	OPEN_PAGE – Offset 0x00	85
8.2.2	Energy Source Version Registers	85
8.2.2.1	ES_HWREV – Offset 0x04	85
8.2.2.2	ES_FWREV0 – Offset 0x06	85
8.2.2.3	ES_FWREV1 – Offset 0x07	85
8.2.2.4	SLOT0_ES_FWREV0 – Offset 0x08	85
8.2.2.5	SLOT0_ES_FWREV1 – Offset 0x09	86
8.2.2.6	SLOT1_ES_FWREV0 – Offset 0x0A	86
8.2.2.7	SLOT1_ES_FWREV1 – Offset 0x0B	86
8.2.3	Energy Source Characteristics Registers	86
8.2.3.1	ES_CHARGE_TIMEOUT0 – Offset 0x10	86
8.2.3.2	ES_CHARGE_TIMEOUT1 – Offset 0x11	86
8.2.3.3	ES_ATTRIBUTES – Offset 0x14	87
8.2.3.4	ES_TECH – Offset 0x15	87
8.2.3.5	MIN_ES_OPERATING_TEMP0 – Offset 0x16	87
8.2.3.6	MIN_ES_OPERATING_TEMP1 – Offset 0x17	88
8.2.3.7	MAX_ES_OPERATING_TEMP0 – Offset 0x18	88
8.2.3.8	MAX_ES_OPERATING_TEMP1 – Offset 0x19	88
8.2.4	Energy Source Runtime Command Registers	88
8.2.4.1	ES_FUNC_CMD0 – Offset 0x30	88
8.2.5	Energy Source Runtime Command Status Registers	89
8.2.5.1	ES_CMD_STATUS0 – Offset 0x50	89
8.2.6	Energy Source Registers	89
8.2.6.1	ES_LIFETIME – Offset 0x70	89
8.2.6.2	ES_TEMP0 – Offset 0x71	89
8.2.6.3	ES_TEMP1 – Offset 0x72	90
8.2.6.4	ES_RUNTIME0 – Offset 0x73	90
8.2.6.5	ES_RUNTIME1 – Offset 0x74	90
8.3	Page 2 Register Map	90
8.3.1	Paging Mechanism Registers	92
8.3.1.1	OPEN_PAGE – Offset 0x00	92
8.3.2	Device Statistics Registers	93
8.3.2.1	LAST_CSAVE_DURATION0 – Offset 0x04	93

Contents (cont'd)

8.3.2.2	LAST_CSAVE_DURATION1 – Offset 0x05	93
8.3.2.3	LAST_RESTORE_DURATION0 – Offset 0x06	93
8.3.2.4	LAST_RESTORE_DURATION1 – Offset 0x07	93
8.3.2.5	LAST_ERASE_DURATION0 – Offset 0x08	93
8.3.2.6	LAST_ERASE_DURATION1 – Offset 0x09	94
8.3.2.7	CSAVE_SUCCESS_COUNT0 – Offset 0x0A	94
8.3.2.8	CSAVE_SUCCESS_COUNT1 – Offset 0x0B	94
8.3.2.9	RESTORE_SUCCESS_COUNT0 – Offset 0x0C	94
8.3.2.10	RESTORE_SUCCESS_COUNT1 – Offset 0x0D	94
8.3.2.11	ERASE_SUCCESS_COUNT0 – Offset 0x0E	95
8.3.2.12	ERASE_SUCCESS_COUNT1 – Offset 0x0F	95
8.3.2.13	POWER_CYCLE_COUNT0 – Offset 0x10	95
8.3.2.14	POWER_CYCLE_COUNT1 – Offset 0x11	95
8.3.2.15	CSAVE_FAILURE_COUNT0 – Offset 0x12	95
8.3.2.16	CSAVE_FAILURE_COUNT1 – Offset 0x13	96
8.3.2.17	RESTORE_FAILURE_COUNT0 – Offset 0x14	96
8.3.2.18	RESTORE_FAILURE_COUNT1 – Offset 0x15	96
8.3.2.19	ERASE_FAILURE_COUNT0 – Offset 0x16	96
8.3.2.20	ERASE_FAILURE_COUNT1 – Offset 0x17	96
8.3.2.21	LAST_ARM_DURATION0 – Offset 0x18	97
8.3.2.22	LAST_ARM_DURATION1 – Offset 0x19	97
8.3.2.23	LAST_FACTORY_DEFAULT_DURATION0 – Offset 0x1A	97
8.3.2.24	LAST_FACTORY_DEFAULT_DURATION1 – Offset 0x1B	97
8.3.2.25	LAST_FIRMWARE_OPS_DURATION0 – Offset 0x1C	97
8.3.2.26	LAST_FIRMWARE_OPS_DURATION1 – Offset 0x1D	98
8.3.2.27	LAST_OPERATIONAL_UNIT_OPS_DURATION0 – Offset 0x1E	98
8.3.2.28	LAST_OPERATIONAL_UNIT_OPS_DURATION1 – Offset 0x1F	98
8.3.2.29	ARM_SUCCESS_COUNT0 – Offset 0x20	98
8.3.2.30	ARM_SUCCESS_COUNT1 – Offset 0x21	98
8.3.2.31	FACTORY_DEFAULT_SUCCESS_COUNT0 – Offset 0x22	98
8.3.2.32	FACTORY_DEFAULT_SUCCESS_COUNT1 – Offset 0x23	99
8.3.2.33	FIRMWARE_SUCCESS_COUNT0 – Offset 0x24	99
8.3.2.34	FIRMWARE_SUCCESS_COUNT1 – Offset 0x25	99
8.3.2.35	OPERATIONAL_UNIT_SUCCESS_COUNT0 – Offset 0x26	99
8.3.2.36	OPERATIONAL_UNIT_SUCCESS_COUNT1 – Offset 0x27	99
8.3.2.37	ARM_FAILURE_COUNT0 – Offset 0x28	99
8.3.2.38	ARM_FAILURE_COUNT1 – Offset 0x29	100
8.3.2.39	FACTORY_DEFAULT_FAILURE_COUNT0 – Offset 0x2A	100
8.3.2.40	FACTORY_DEFAULT_FAILURE_COUNT1 – Offset 0x2B	100
8.3.2.41	FIRMWARE_FAILURE_COUNT0 – Offset 0x2C	100
8.3.2.42	FIRMWARE_FAILURE_COUNT1 – Offset 0x2D	100
8.3.2.43	OPERATIONAL_UNIT_FAILURE_COUNT0 – Offset 0x2E	100
8.3.2.44	OPERATIONAL_UNIT_FAILURE_COUNT1 – Offset 0x2F	101
8.3.3	Error Injection Registers	101
8.3.3.1	INJECT_OPS_FAILURES0 – Offset 0x60	101
8.3.3.2	INJECT_OPS_FAILURES1 – Offset 0x61	102
8.3.3.3	INJECT_ES_FAILURES – Offset 0x64	102

Contents (cont'd)

8.3.3.4	INJECT_FW_FAILURES – Offset 0x65	103
8.3.3.5	INJECT_BAD_BLOCK_CAP – Offset 0x67	104
8.3.3.6	INJECT_ERROR_TYPE – Offset 0x68	104
8.3.4	Host Area Registers	104
8.3.4.1	DRAM_ECC_ERROR_COUNT – Offset 0x80	104
8.3.4.2	DRAM_THRESHOLD_ECC_COUNT – Offset 0x81	104
8.3.4.3	HOST_MANAGED_ES_ATTRIBUTES – Offset 0x82	105
8.3.4.4	HOST_CSAVE_FAIL – Offset 0x83	105
8.3.4.5	HOST_CSAVE_WORKFLOW_FAILURE_COUNT0 – Offset 0x84	106
8.3.4.6	HOST_CSAVE_WORKFLOW_FAILURE_COUNT1 – Offset 0x85	106
8.4	Page 3 Register Map	106
8.4.1	Paging Mechanism Registers	108
8.4.1.1	OPEN_PAGE – Offset 0x00	108
8.4.2	Typed Block Data Registers	108
8.4.2.1	TYPED_BLOCK_DATA – Offset 0x04	109
8.4.2.2	REGION_ID0 – Offset 0x05	109
8.4.2.3	REGION_ID1 – Offset 0x06	109
8.4.2.4	BLOCK_ID – Offset 0x07	110
8.4.2.5	TYPED_BLOCK_DATA_SIZE0 – Offset 0x08	110
8.4.2.6	TYPED_BLOCK_DATA_SIZE1 – Offset 0x09	110
8.4.2.7	TYPED_BLOCK_DATA_SIZE2 – Offset 0x0A	110
8.4.2.8	OPERATIONAL_UNIT_ID0 – Offset 0x0C	110
8.4.2.9	OPERATIONAL_UNIT_ID1 – Offset 0x0D	110
8.4.2.10	OPERATIONAL_UNIT_SIZE0 – Offset 0x10	111
8.4.2.11	OPERATIONAL_UNIT_SIZE1 – Offset 0x11	111
8.4.2.12	OPERATIONAL_UNIT_SIZE2 – Offset 0x12	111
8.4.2.13	OPERATIONAL_UNIT_CRC0 – Offset 0x14	111
8.4.2.14	OPERATIONAL_UNIT_CRC1 – Offset 0x15	111
8.4.3	Firmware Update Registers	111
8.4.3.1	FW_REGION_CRC0 – Offset 0x40	112
8.4.3.2	FW_REGION_CRC1 – Offset 0x41	112
8.4.3.3	FW_SLOT_INFO – Offset 0x42	112
8.4.4	Byte and Word Transaction Data Registers for Typed Block Data	112
8.4.4.1	TYPED_BLOCK_DATA_BYTE0 – Offset 0x80	112
8.4.4.2	TYPED_BLOCK_DATA_BYTE1 – Offset 0x81	113
8.4.4.3	TYPED_BLOCK_DATA_BYTE2 – Offset 0x82	113
8.4.4.4	TYPED_BLOCK_DATA_BYTE3 – Offset 0x83	113
8.4.4.5	TYPED_BLOCK_DATA_BYTE4 – Offset 0x84	113
8.4.4.6	TYPED_BLOCK_DATA_BYTE5 – Offset 0x85	113
8.4.4.7	TYPED_BLOCK_DATA_BYTE6 – Offset 0x86	113
8.4.4.8	TYPED_BLOCK_DATA_BYTE7 – Offset 0x87	113
8.4.4.9	TYPED_BLOCK_DATA_BYTE8 – Offset 0x88	114
8.4.4.10	TYPED_BLOCK_DATA_BYTE9 – Offset 0x89	114
8.4.4.11	TYPED_BLOCK_DATA_BYTE10 – Offset 0x8A	114
8.4.4.12	TYPED_BLOCK_DATA_BYTE11 – Offset 0x8B	114
8.4.4.13	TYPED_BLOCK_DATA_BYTE12 – Offset 0x8C	114
8.4.4.14	TYPED_BLOCK_DATA_BYTE13 – Offset 0x8D	114

Contents (cont'd)

8.4.4.15	TYPED_BLOCK_DATA_BYTE14 – Offset 0x8E	114
8.4.4.16	TYPED_BLOCK_DATA_BYTE15 – Offset 0x8F	115
8.4.4.17	TYPED_BLOCK_DATA_BYTE16 – Offset 0x90	115
8.4.4.18	TYPED_BLOCK_DATA_BYTE17 – Offset 0x91	115
8.4.4.19	TYPED_BLOCK_DATA_BYTE18 – Offset 0x92	115
8.4.4.20	TYPED_BLOCK_DATA_BYTE19 – Offset 0x93	115
8.4.4.21	TYPED_BLOCK_DATA_BYTE20 – Offset 0x94	115
8.4.4.22	TYPED_BLOCK_DATA_BYTE21 – Offset 0x95	115
8.4.4.23	TYPED_BLOCK_DATA_BYTE22 – Offset 0x96	116
8.4.4.24	TYPED_BLOCK_DATA_BYTE23 – Offset 0x97	116
8.4.4.25	TYPED_BLOCK_DATA_BYTE24 – Offset 0x98	116
8.4.4.26	TYPED_BLOCK_DATA_BYTE25 – Offset 0x99	116
8.4.4.27	TYPED_BLOCK_DATA_BYTE26 – Offset 0x9A	116
8.4.4.28	TYPED_BLOCK_DATA_BYTE27 – Offset 0x9B	116
8.4.4.29	TYPED_BLOCK_DATA_BYTE28 – Offset 0x9C	116
8.4.4.30	TYPED_BLOCK_DATA_BYTE29 – Offset 0x9D	117
8.4.4.31	TYPED_BLOCK_DATA_BYTE30 – Offset 0x9E	117
8.4.4.32	TYPED_BLOCK_DATA_BYTE31 – Offset 0x9F	117
8.4.5	Block Transaction Data Registers for Typed Blocked Data	117
8.4.5.1	TYPED_BLOCK_DATA_OFFSET – Offset 0xE0	117
8.5	Page 4 Register Map	117
8.5.1	OPEN_PAGE – Offset 0x00	118
8.5.2	PDA_SUPPORTED – Offset 0x04	118
8.5.3	PDA_VALID – Offset 0x05	119
8.5.4	MRx_SHARED Registers – Offsets 0x10-0x11 to 0x1C-0x1D	119
8.6	Page 5 Register Map	119
8.6.1	OPEN_PAGE – Offset 0x00	120
8.6.2	MRx_PDA_SDRAMyy Registers – Offsets 0x10-0x11 to 0xFE-0xFF	120
8.7	Page 6 Register Map	121
8.7.1	OPEN_PAGE – Offset 0x00	121
8.7.2	RCD_FUNCTIONS_SUPPORTED0 – Offset 0x04	121
8.7.3	RCD_FUNCTIONS_SUPPORTED1 – Offset 0x05	122
8.7.4	RCD_FUNCTIONS_VALID0 – Offset 0x06	122
8.7.5	RCD_FUNCTIONS_VALID1 – Offset 0x07	122
8.7.6	Control Word Registers	122
8.8	Page 7 Register Map	122
8.8.1	OPEN_PAGE – Offset 0x00	123
8.8.2	Control Word Registers	123
8.9	Page 8 Register Map	123
8.9.1	OPEN_PAGE – Offset 0x00	124
8.9.2	Control Word Registers	124
8.10	Page 9 Register Map	124
8.10.1	OPEN_PAGE – Offset 0x00	124
8.10.2	Control Word Registers	124
9	Host Operation Workflows	125
9.1	Controller Ready Workflow	125
9.2	Catastrophic Save Workflow	125

Contents (cont'd)

9.3	Restore Workflow	125
9.4	Arm Workflow	126
9.5	Erase Workflow	126
9.6	Abort Running Operation Workflow	127
9.7	Firmware Update Workflow	127
9.8	Typed Block Data Workflow	132
9.8.1	Reading Typed Block Data Workflow	133
9.8.2	Writing Typed Block Data Workflow	134
9.8.3	Security Protocol Data Workflow	135
Annex A	- (Normative) SDRAM and RCD Requirements for NVRDIMM-N Modules	138
A.1	SDRAM and RCD Requirements During a Catastrophic Save Operation	138
A.2	SDRAM and RCD Requirements Related To A Restore Operation	138
A.2.1	Overview	138
A.2.2	Restore Operation Requirements	138
A.2.2.1	SDRAMs During a Restore Operation	138
A.2.2.2	RCD During a Restore Operation	139
A.2.3	Post-restore Transition Entry Requirements	139
A.2.3.1	SDRAMs for the Post-Restore Transition	139
A.2.3.2	RCD for a Post-Restore Transition	142
A.2.4	Post-Restore Transition Exit Requirements	144
A.2.4.1	Host Reprogramming the RCD after a Post-restore Transition	144
A.2.4.2	Host Reprogramming the SDRAMs after a Post-restore Transition	147
Annex B	- (Normative) Self-encrypting NVDIMM-N	149
B.1	Overview	149
B.2	Requirements	151
B.3	Data Encryption and Decryption Engines	151
B.3.1	Encryption Algorithm	151
B.3.2	Block Cipher Mode	152
B.4	RBG (Random Bit Generator) Engine	153
B.4.1	RBG Overview	153
B.4.2	Entropy Source	153
B.4.3	DBRG (Deterministic Random Bit Generator)	153
B.5	PBKDF (Password-based Key Derivation Function) Engine	154
B.6	Key Wrapping And Unwrapping Engines	155
B.7	Cryptographic Values	156
B.7.1	DEK (Data Encryption Key)	156
B.7.2	Salt	157
B.7.3	ICV (Integrity Check Value)	157
B.7.4	Password	157
B.7.5	KEK (Key Encryption Key)	157
B.7.6	Wrapped DEK	157
B.8	Authentication	158
B.8.1	TCG Storage Protocol	158
B.8.2	TCG Storage Ruby SSC	158
B.8.3	Using Passwords	159
B.8.4	Roles	160
B.8.5	BAEBI Operations	160

Contents (cont'd)

B.8.6	TCG Services	161
B.8.7	Key Usage	162
B.8.8	Using Ruby SSC Devices	163
B.9	Printed Label	163
B.10	Sanitization	163
B.11	Signed Firmware	164
B.11.1	Signed Firmware Overview	164
B.11.2	Access to Flash Memory	165
B.11.3	Signature Requirements	165
B.11.4	Public Key Handling	166
B.11.5	Firmware Version Numbers	166
B.11.6	Managing Production vs. Test Builds	166
B.11.7	FIPS 140-2 Compliance for Signed Firmware	166
B.12	FIPS 140-2 (Cryptographic Module) Compliance	167
B.13	Common Criteria Compliance	167
Annex C	- (Informative) Differences Between Revisions	168
C.1	Differences Between JESD245 (v1.0) and JESD245A (v2.0)	168
C.2	Differences Between JESD245A (v2.0) and (JESD245B)(v2.1)	168
C.3	Differences Between JESD245B (v2.1) and this Standard (JESD245C)(v2.2)	170
C.4	Differences Between JESD245C (v2.2) and this Standard (JESD245D)(v2.3)	171
C.5	Differences between JESD245D (v2.3) and this Standard (JESD245E)(v2.4)	172

Figures	Page
Figure 1 — NVDIMM-N Overview	10
Figure 2 — I ² C Read Byte Transaction	12
Figure 3 — I ² C Write Byte Transaction	12
Figure 4 — I ² C Read Word Transaction	12
Figure 5 — I ² C Write Word Transaction	13
Figure 6 — I ² C Block Read Transaction	13
Figure 7 — I ² C Block Write Transaction	14
Figure 8 — I ² C Paging Mechanism	15
Figure 9 — SAVE_n Trigger Mode Timing Diagram	20
Figure 10 — RESET_n Trigger Mode Timing Diagram	21
Figure 11 — Typed Block Data Block Diagram	135
Figure 12 — Some Cryptographic Components in a Self-encrypting NVDIMM-N	149
Figure 13 — XTS-AES Encryption	152
Figure 14 — XTS-AES Decryption	152
Figure 15 — Random Bit Generator (RBG)	153
Figure 16 — PBKDF Engine	154
Figure 17 — Key Wrap Engine	155
Figure 18 — Key Unwrap Engine	156
Figure 19 — Code Signing Process	164

Contents (cont'd)

Tables	Page
Table 1 — Required SPD Fields.....	17
Table 2 — SAVE_n Signal Timing.....	20
Table 3 — RESET_n Signal Timing.....	21
Table 4 — Firmware Image Header – Common Format.....	32
Table 5 — Firmware Image Header Format 0.....	33
Table 6 — Firmware Image Header Format 1.....	33
Table 7 — Temperature Value Bit Definition.....	36
Table 8 — Mode Register – Lower Byte.....	38
Table 9 — Mode Register – Upper Byte.....	39
Table 10 — Page 0 Register Map Categories.....	40
Table 11 — Page 0 Register Map.....	40
Table 12 — Major Revision And Minor Revision Fields.....	45
Table 13 — Reset Controller Scope Field.....	47
Table 14 — Page 1 Register Map Categories.....	83
Table 15 — Page 1 Register Map.....	84
Table 16 — Page 2 Register Map Categories.....	90
Table 17 — Page 2 Register Map.....	91
Table 18 — Page 3 Register Map Categories.....	106
Table 19 — Page 3 Register Map.....	107
Table 20 — Page 4 Register Map.....	118
Table 21 — Page 5 Register Map.....	120
Table 22 — Page 6 Register Map.....	121
Table 23 — Page 7 Register Map.....	123
Table 24 — Page 8 Register Map.....	123
Table 25 — Page 9 Register Map.....	124
Table 26 — MR0 Requirements During a Post-Restore Transition.....	140
Table 27 — MR1 Requirements During a Post-Restore Transition.....	140
Table 28 — MR2 Requirements During a Post-Restore Transition.....	140
Table 29 — MR3 Requirements During a Post-Restore Transition.....	141
Table 30 — MR4 Requirements During a Post-Restore Transition.....	141
Table 31 — MR5 Requirements During a Post-Restore Transition.....	142
Table 32 — MR6 Requirements During a Post-Restore Transition.....	142
Table 33 — RCD Requirements During a Post-Restore Transition.....	143
Table 34 — Host Reprogramming RCD Function 0 Control Words After a Post-Restore Transition.....	145
Table 35 — Host Reprogramming RCD Function 1 Control Words After a Post-Restore Transition.....	147
Table 36 — Host Reprogramming RCD Function 4 Control Words After a Post-Restore Transition.....	147
Table 37 — PBKDF Engine Parameters, Inputs, and Outputs.....	155
Table 38 — Key Wrap Engine Prerequisites, Inputs, and Outputs.....	156
Table 39 — Key Unwrap Engine Prerequisites, Inputs, and Outputs.....	156
Table 40 — Authenticated and Unauthenticated BAEBI Operations.....	160
Table 41 — Authenticated and Unauthenticated Services.....	162
Table 42 — Key Usage.....	162

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 or other triggers. 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 (NVM), providing a storage-class memory solution. The module may also support the ability to encrypt the SDRAM contents before backing up to the NVM.

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-21-43, formulated under the cognizance of the JC-45.6 Subcommittee on Hybrid Modules, item 2233.54G.)

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

2.1 Overview

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.

2.2 JEDEC Standards

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

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

JESD21C Page 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*

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)

JESD245B.01 *Byte Addressable Energy Backed Interface* (September 2017)

JESD248A *DDR4 NVDIMM-N Design Standard* (March 2018)

2.3 Management Interface Standards

I²C Bus Specification Revision 6 (4 April 2014)

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

2.4 NIST (National Institute of Standards and Technology) Standards

The U.S. Department of Commerce National Institute of Standards and Technology (NIST) Information Technology Laboratory (ITL) Computer Security Division (CSD) develops Federal Information Processing Standards (FIPS) and Special Publications (SPs) for computer security at <https://csrc.nist.gov>.

- FIPS 180-4 Secure Hash Standard (SHS) (August 2015)
 - <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>
- FIPS 186-4 Digital Signature Standard (DSS) (July 2013)
 - <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>
- FIPS 197 Advanced Encryption Standard (AES) (26 November 2001)
 - <https://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
- SP800-38D Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC (November 2007)
 - <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf>
- SP800-38E Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices (January 2010)
 - <https://csrc.nist.gov/publications/detail/sp/800-38e/final>
- SP800-38F Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping (December 2012)
 - <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38f.pdf>
- SP800-88 Guidelines for Media Sanitization (Rev 1, December 2014)
 - <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-88r1.pdf>
- SP900-90A Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revision 1, June 2015)
 - <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf>
- SP800-90B Recommendation for the Entropy Sources Used for Random Bit Generation (January 2018)
 - <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf>
 - overview at <https://csrc.nist.gov/groups/ST/rbg-workshop-2016/presentations/Session-I-1-john-kelsey-presentation.pdf>
- SP800-90C Recommendation for Random Bit Generator (RBG) Constructions (Second Draft) (April 2016)
 - https://csrc.nist.gov/publications/drafts/800-90/sp800_90c_second_draft.pdf
 - overview at <https://csrc.nist.gov/groups/ST/rbg-workshop-2016/presentations/SessionI-2-elaine-barker-presentation.pdf>
- SP800-108 Recommendation for Key Derivation Using Pseudorandom Functions (October 2009)
 - <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-108.pdf>
- SP800-132 Recommendation for Password-Based Key Derivation Part 1: Storage Applications (December 2010)
 - <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf>
- SP800-147 BIOS Protection Guidelines (April 2011)
 - <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-147.pdf>
- SP800-147B BIOS Protection Guidelines for Servers (August 2014)
 - <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-147B.pdf>

2.5 NIST CAVP (Cryptographic Algorithm Validation Program)

The Cryptographic Algorithm Validation Program (CAVP) provides validation testing of cryptographic algorithms at <https://csrc.nist.gov/groups/STM/cavp>.

- The Advanced Encryption Standard Algorithm Validation Suite (AESAVS) (15 November 2002)
 - <https://csrc.nist.gov/groups/STM/cavp/documents/aes/AESAVS.pdf>
- The XTS-AES Validation System (XTSVS) (5 September 2013)
 - <https://csrc.nist.gov/groups/STM/cavp/documents/aes/XTSVS.pdf>
- The NIST SP800-90A Deterministic Random Bit Generator Validation System (DRBGVS) (29 October 2015)
 - <https://csrc.nist.gov/groups/STM/cavp/documents/drbg/DRBGVS.pdf>
- Secure Hash Standard Validation System (SHAVS)
 - <https://csrc.nist.gov/groups/STM/cavp/documents/shs/SHAVS.pdf>
- Keyed-Hash Message Authentication Code Validation System (HMACVS)
 - <https://csrc.nist.gov/groups/STM/cavp/documents/mac/HMACVS.pdf>
- The Key Wrap Validation System (KWVS)
 - <https://csrc.nist.gov/groups/STM/cavp/documents/mac/KWVS.pdf>
- Key Derivation using Pseudorandom Functions (SP800-108) Validation System (KDKFVS)
 - <https://csrc.nist.gov/groups/STM/cavp/documents/KDKF800-108/kdkfvs.pdf>

2.6 NIST CMVP (Cryptographic Module Validation Program)

The Cryptographic Module Validation Program (CMVP) validates cryptographic modules at <http://csrc.nist.gov/groups/STM/cmvp>.

- NIST FIPS PUB 140-2 Security Requirements for Cryptographic Modules (25 May 2001)
 - <https://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf>
- Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program (7 May 2019)
 - <https://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf>
- Derived Test Requirements for FIPS PUB 140-2 (4 January 2011)
 - <https://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402DTR.pdf>

2.7 IEEE Standards

The IEEE Security in Storage Working group defined data-at-rest encryption modes at <http://siswg.net>.

- IEEE 1619-2018 Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices
 - For sale at <https://standards.ieee.org/findstds/standard/1619-2018.html>

2.8 Common Criteria Standards

Common Criteria standards and their testing requirements are developed by the 27 country members of the Common Criteria Recognition Agreement at <https://www.commoncriteriaportal.org>:

- Common Criteria for Information Technology Security Evaluation – Part 1: Introduction and general model (Version 3.1 Revision 4, September 2012)
 - <https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R4.pdf>
- Common Criteria for Information Technology Security Evaluation – Part 2: Security functional components (Version 3.1 Revision 4, September 2012)
 - <https://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R4.pdf>

2.8 Common Criteria Standards (cont'd)

- Common Criteria for Information Technology Security Evaluation – Part 3: Security assurance components (Version 3.1 Revision 4, September 2012)
 - <https://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R4.pdf>
- Common Methodology for Information Technology Security Evaluation – Evaluation methodology (Version 3.1, Revision 4, September 2012)
 - <https://www.commoncriteriaportal.org/files/ccfiles/CEMV3.1R4.pdf>

The USA representative to CCRA is the National Information Assurance Partnership at <https://www.niap-ccevs.org>.

Common Criteria Protection Profiles for Full Drive Encryption and their testing requirements are developed by the Common Criteria Full Disk Encryption work group at <https://www.commoncriteriaportal.org/communities/fde.cfm>:

- collaborative Protection Profile for Full Drive Encryption - Encryption Engine (cPP_FDE_EE)(Version 2.0, 22 September 2016)
 - https://www.commoncriteriaportal.org/files/ppfiles/CPP_FDE_EE_V2.0.pdf
- Supporting Document (Mandatory Technical Document) for Full Drive Encryption: Encryption Engine (Version 1.0, January 2015)
 - https://www.commoncriteriaportal.org/files/ppfiles/CPP_FDE_EE_V2.0_supporting_doc.pdf and <https://www.commoncriteriaportal.org/files/supdocs/CCDB-2015-01-004.pdf>
- collaborative Protection Profile for Full Drive Encryption - Authorization Acquisition (cPP_FDE_AA)(Version 2.0, 22 September 2016)
 - https://www.commoncriteriaportal.org/files/ppfiles/CPP_FDE_AA_V2.0.pdf
- Supporting Document (Mandatory Technical Document) for Full Drive Encryption: Authorization Acquisition (Version 1.0, January 2015)
 - https://www.commoncriteriaportal.org/files/ppfiles/CPP_FDE_AA_V2.0_supporting_doc.pdf and <https://www.commoncriteriaportal.org/files/supdocs/CCDB-2015-01-003.pdf>

2.9 TCG (Trusted Computing Group) Standards

The Trusted Computing Group defines standards for secure and trusted computing at <https://trustedcomputinggroup.org>. The Storage Working Group defines storage security interfaces at <https://trustedcomputinggroup.org/work-groups/storage>.

- TCG Storage Interface Interactions Specification (SIIS), Version 1.08, Revision 1.00 (26 October 2018)
 - <https://trustedcomputinggroup.org/resource/storage-work-group-storage-interface-interactions-specification>
- TCG Storage Security Subsystem Class: Opal Version 2.01 Revision 1.00 (5 August 2015)
 - <https://trustedcomputinggroup.org/resource/storage-work-group-storage-security-subsystem-class-opal>
- TCG Storage Security Subsystem Class: Ruby Version 1.00 Revision 1.26 (25 May 2019) (Work in Progress – PUBLIC REVIEW)
 - <https://trustedcomputinggroup.org/specifications-public-review>
 - <https://trustedcomputinggroup.org/resource/tcg-storage-security-subsystem-class-ruby>
- TCG Storage Opal SSC Feature Set: PSID Version 1.00 Revision 1.00 (5 August 2015)
 - <https://trustedcomputinggroup.org/resource/tcg-storage-opal-feature-set-psid>
- TCG Storage Opal SSC Feature Set: Single User Mode Version 1.00 Revision 2.00 (5 August 2015)
 - <https://trustedcomputinggroup.org/tcg-storage-opal-ssc-feature-set-single-user-mode>

2.9 TCG (Trusted Computing Group) Standards (cont'd)

- TCG Storage Feature Set: Block SID Authentication Version 1.00 Revision 1.00 (5 August 2015)
 - <https://trustedcomputinggroup.org/resource/tcg-storage-feature-set-block-sid-authentication-specification>
- TCG Storage Architecture Core Specification Version 2.01 Revision 1.00 (5 August 2015)
 - <https://trustedcomputinggroup.org/resource/tcg-storage-architecture-core-specification>

2.10 Best Practices

These papers discuss various attacks on cryptographic modules and provide advice on implementations.

- Power Analysis Resistant Hardware Implementations of AES. Levent Ordu and Berna Örs, 14th IEEE International Conference on Electronics, Circuits, and Systems, 2007.
 - for sale at <https://ieeexplore.ieee.org/document/4511263?arnumber=4511263>
- An Implementation of DES and AES, Secure against Some Attacks. Mehdi-Laurent Akkar and Christophe Giraud, Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems (CHES '01), 2001.
 - https://link.springer.com/chapter/10.1007/3-540-44709-1_26

3 Terms and Definitions

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

3.1 Acronyms

ASIC	application-specific integrated circuit
BAEBI	byte addressable energy backed interface
CSAVE	catastrophic save
DDR3	double data rate version 3 (see JESD79-3F)
DDR4	double data rate version 4 (see JESD79-4B)
DEK	data encryption key (see B.7.1)
DIMM	dual in-line memory module
DRBG	deterministic random bit generator (see B.4.3)
ES	Energy Source
FPGA	field-programmable gate array
ICV	integrity check value (see B.7.3)
I ² C	inter integrated circuit
KEY	key encryption key (see B.7.5)
LCOM	local communications interface
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
PBKDF	password-based key derivation function (see B.5)
PSID	physical owner security ID (see B.9)
RBG	random bit generator (see B.4)
RCD	registering clock driver
RDIMM	registered DIMM
SDRAM	synchronous dynamic random access memory
SMBus	System Management Bus
SP	security provider (see TCG Storage Architecture)
SPD	serial presence detect
SSC	security subsystem class (see TCG Storage Architecture)

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 (CSAVE): Operation that copies the SDRAM contents into NVM. 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 DODT_n 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.

3.2 Terms and Definitions (cont'd)

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. Factory Default values are indicated in the “Default” column of Register Definitions fields.

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 interface that connects the module’s controller to other components. See 7.14.

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 non-volatile DIMM (NVDIMM) is a DIMM that maintains the contents of SDRAM during power loss (see JESD248A). 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 (BAEBI).

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, an encryption engine (EE) if encryption is supported, and I²C registers. The NVDIMM-N draws power for the Catastrophic Save operation either from the host via the V₁₂ pins on module edge connector or from a Device-Managed Energy Source.

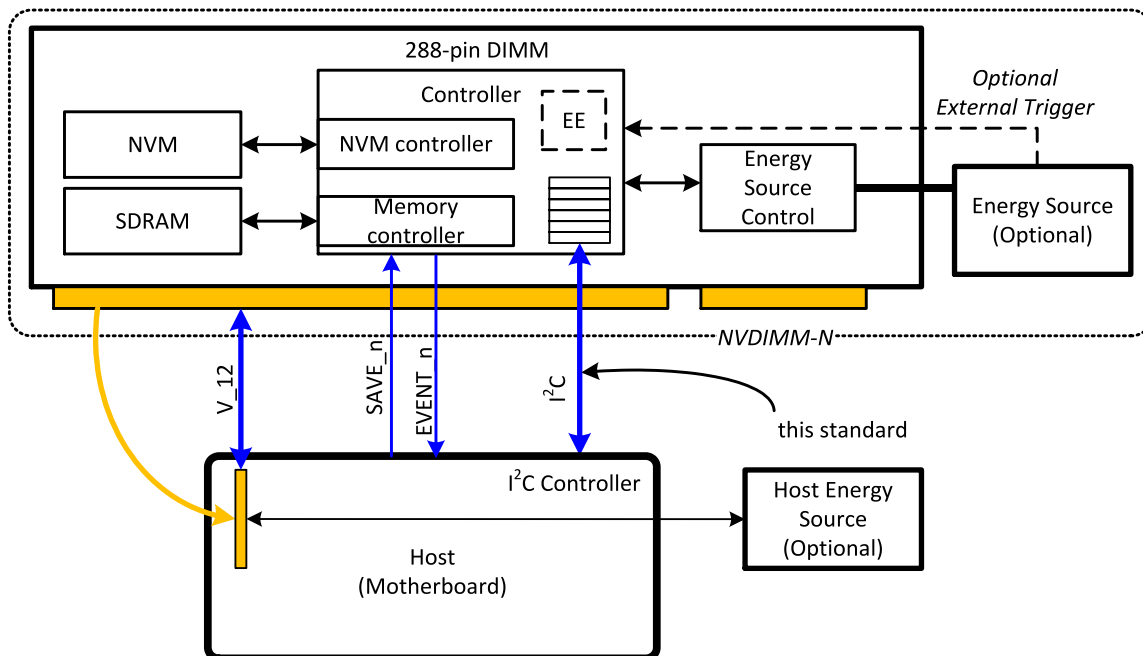


Figure 1 — NVDIMM-N Overview

5 I²C

This clause 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 target 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 (see 8.4.4); 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 (see 8.4.5.1). 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 describe the I²C transactions between the host and the module. The host is the I²C controller and the module is the I²C target. 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.

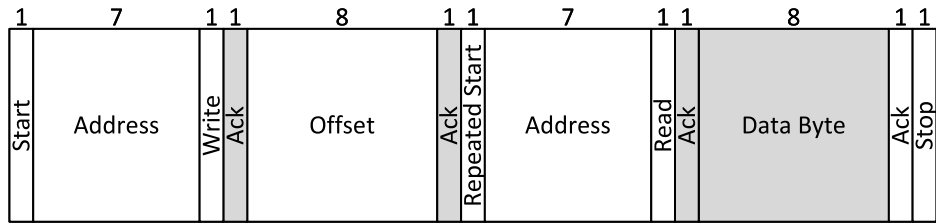


Figure 2 — I²C Read Byte Transaction

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.

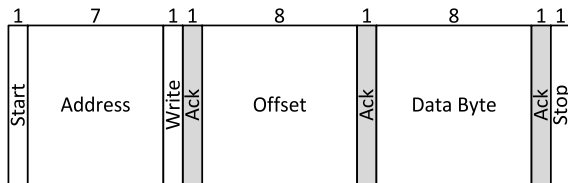


Figure 3 — I²C Write Byte Transaction

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.

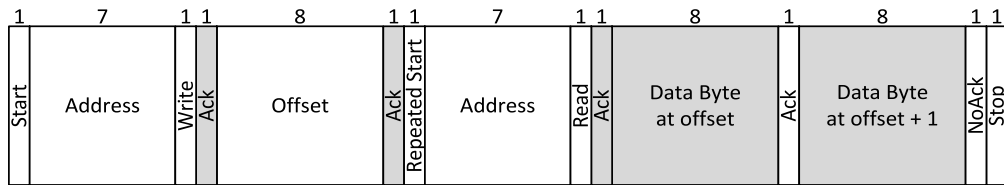


Figure 4 — I²C Read Word Transaction

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.

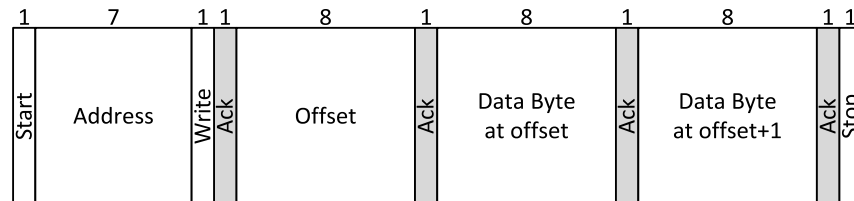


Figure 5 — I²C Write Word Transaction

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.

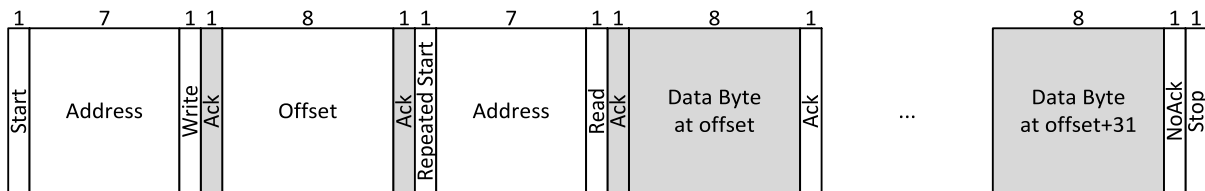


Figure 6 — I²C Block Read Transaction

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.

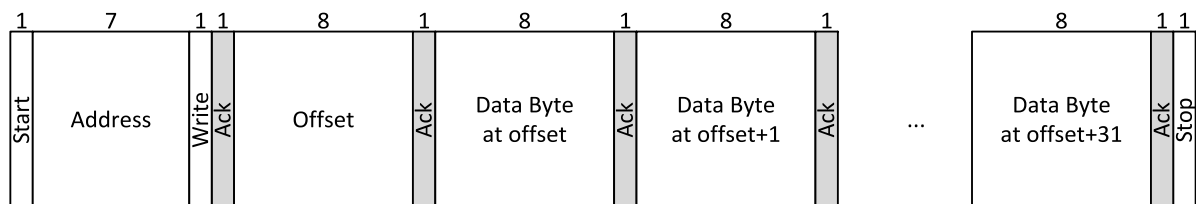


Figure 7 — I²C Block Write Transaction**5.2.7 I²C Transactions to Reserved Offsets**

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.

5.3 Paging (cont'd)

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.

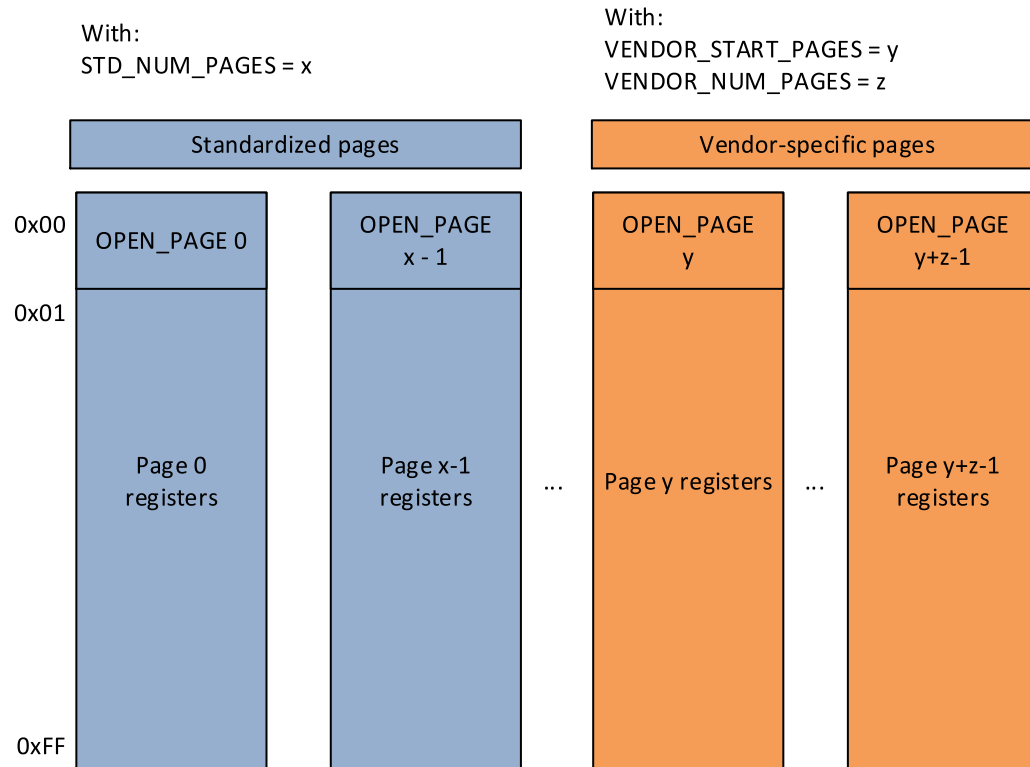


Figure 8 — I²C Paging Mechanism

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.

5.4 Typed Block Data (cont'd)

The module shall support the following Typed Block Data types:

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

If encryption is supported, then the module shall support the following Typed Block Data type:

- Security Protocol Data (see 7.17).

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.

If Typed Block Data type is set to Security Protocol Data (see 7.17), the host can write the TYPED_BLOCK_DATA_SIZE0, TYPED_BLOCK_DATA_SIZE1 and TYPED_BLOCK_DATA_SIZE2 registers in page 3.

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-4 and include the hybrid module parameters for a byte addressable energy backed function defined in JESD21C Page 4.1.2.L-4.

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-4. All multi-byte values in Table 1 are little endian.

6.1 Byte Addressable Energy Backed Interface Discovery (cont'd)

Table 1 — Required SPD Fields

SPD Field Name	SPD Byte Number(s)	Value(s)
SPD Revision	1	A value of 0x11 (i.e., NVDIMM-N Revision 1.1) 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:
 - the OPEN_PAGE register (see 8.1.1.1);
 - the NVDIMM_READY register (see 8.1.5.1);
 - the MODULE_HEALTH register (see 8.1.8.1);
 - the MODULE_HEALTH_STATUS0 register (see 8.1.8.2); and
 - the MODULE_HEALTH_STATUS1 register (see 8.1.8.3).

NOTE: Modules compliant with JESD245 and JESD245A 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-4).

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 clause describes the operations that the host may request the module to perform. If the host attempts to start an operation that cannot be executed due to a current running operation, the module shall ignore the command. While an operation is running on the module, the only operations that can be started are the Abort operation (see 7.2.13) and the Management operations (see 7.2.5). If the current running operation is the Abort operation, the only operation that can be started by the host is the Management operation.

7.2.1 Catastrophic Save Operation

The module shall support the Catastrophic Save operation.

The module shall support the ability to start a Catastrophic Save operation through one of the following mechanisms:

- the START_CSAVE bit in the NVDIMM_FUNC_CMD register (see 8.1.4.3);
- the SAVE_n pin (see 7.2.1.2);
- the RESET_n pin (see 7.2.1.3); or
- vendor-specific methods (see 7.13).

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.

A module is required to be armed to start a Catastrophic Save operation at power loss. If the module was armed and did not start a Catastrophic Save operation when power is lost, it shall be considered a Catastrophic Save operation failure (see 8.1.6.2).

The module is not required to be armed to start a Catastrophic Save operation by the START_CSAVE bit in the NVDIMM_FUNC_CMD register.

The module shall 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).

If a START_CSAVE based request is received while the module is locked (i.e., a write of Typed Block Data type 4h (see 8.4.2.1) to unlock the module (see SIIS) has not completed successfully), then the module shall reject the request by setting the CSAVE_REJECT bit and the CSAVE_ERROR bit in the CSAVE_STATUS register and the SECURITY_ERROR bit in the CSAVE_FAIL_INFO1 register.

Upon receiving a START_CSAVE, SAVE_n, or RESET_n 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_STATUS 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 (see 7.2.12).

Regardless of how a CSAVE was initiated, START_CSAVE, SAVE_n, or RESET_n based request, the non-volatile controller shall mask the RESET_n pin on the module while armed.

7.2.1 Catastrophic Save Operation (cont'd)

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:

- if encryption has not been enabled, copy the SDRAM contents to the NVM; and
- if encryption has been enabled, encrypt the contents of SDRAM and store the encrypted 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.

7.2.1.1 START_CSAVE Bit

When the Catastrophic Save operation is triggered by the START_CSAVE bit, the module shall:

- set Bit 1 (Triggered by START_CSAVE) in the CSAVE_INFO register (see 8.1.6.1);
- clear Bit 2 (Triggered by SAVE_n) in the CSAVE_INFO register;
- clear Bit 3 (Triggered by RESET_n) in the CSAVE_INFO register;
- clear the CSAVE_FAIL_INFO0 and CSAVE_FAIL_INFO1 registers (see 8.1.6.2 and 8.1.6.3); and
- mask the RESET_n pin.

7.2.1.2 SAVE_n Pin

When the Catastrophic Save operation is triggered by the SAVE_n pin, the module shall:

- clear Bit 1 (Triggered by START_CSAVE) in the CSAVE_INFO register (see 8.1.6.1);
- clear Bit 3 (Triggered by RESET_n) in the CSAVE_INFO register;
- set Bit 2 (Triggered by SAVE_n) in the CSAVE_INFO register;
- clear the CSAVE_FAIL_INFO0 and CSAVE_FAIL_INFO1 registers (see 8.1.6.2 and 8.1.6.3);

7.2.1.2 SAVE_n Pin (cont'd)

- 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;
- mask the RESET_n pin.

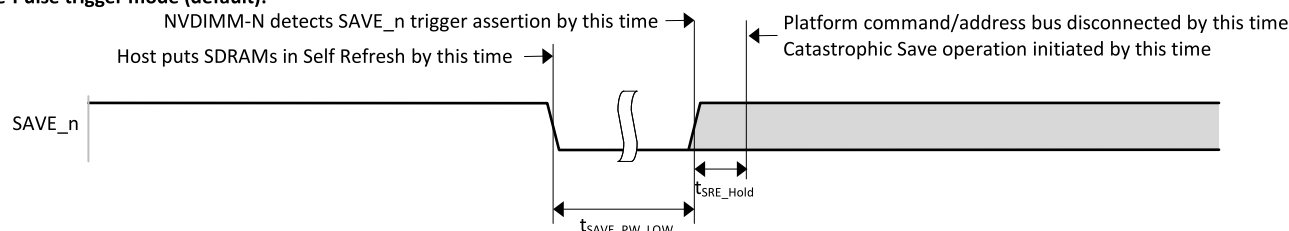
Table 2 defines the critical timing parameters for a Catastrophic Save triggered by SAVE_n.

Table 2 — SAVE_n Signal Timing

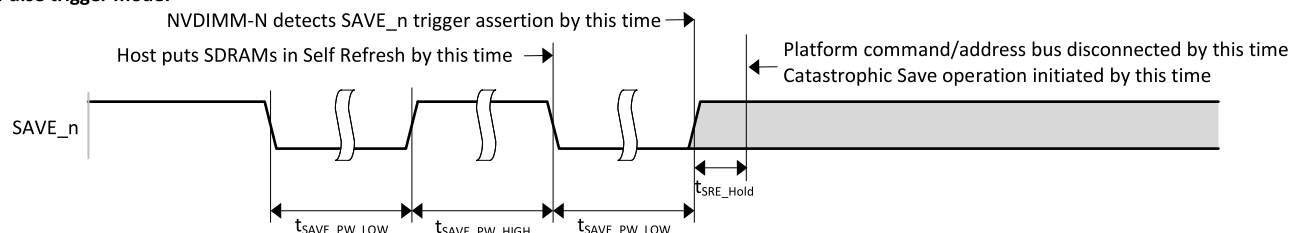
Symbol	Parameter	Min	Typ	Max	Unit
$t_{\text{SAVE_PW_LOW}}$	SAVE_n Pulse Width LOW Duration	1	-	-	μs
$t_{\text{SAVE_PW_HIGH}}$	SAVE_n Pulse Width HIGH Duration	1	-	-	μs
$t_{\text{SRE_Hold}}$	NVDIMM SDRAM Self Refresh Hold Time after end of SAVE_n pulse	25	-	-	μs

Figure 9 shows SAVE_n trigger mode timing.

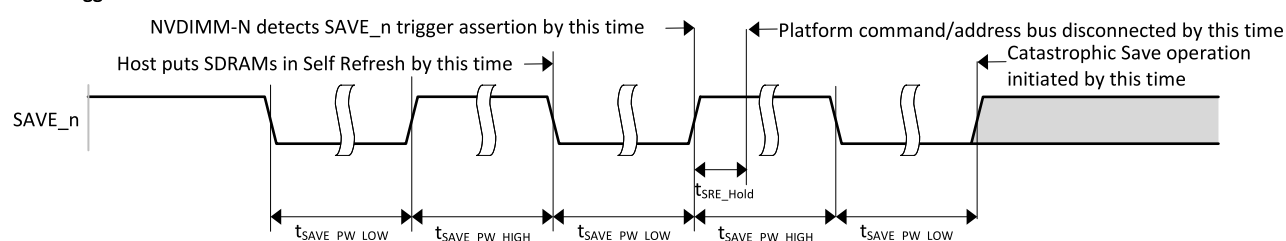
Single-Pulse trigger mode (default):



Two-Pulse trigger mode:



Three-Pulse trigger mode:



RESET_n during Catastrophic Save triggered by SAVE_n:

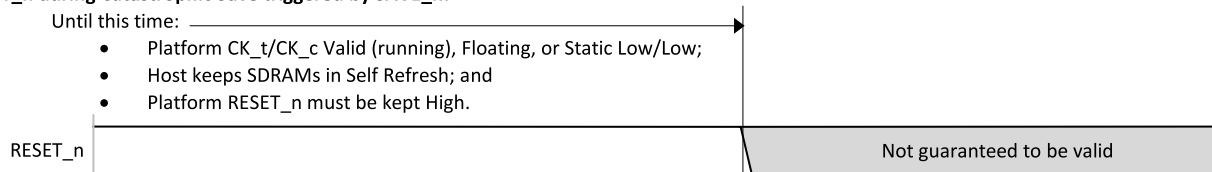


Figure 9 — SAVE_n Trigger Mode Timing Diagram

7.2.1.2 SAVE_n Pin (cont'd)

SDRAMs are kept in Self Refresh mode by continuously driving their CKE input signals LOW. When an RCD device is present, this can be achieved by keeping the RCD in Clock Stopped Power Down mode. In this mode the platform CKEn signals are ignored and the CK_t/CK_c clock signals have to either be driven both LOW or they have to be allowed to float (i.e., no drive). As an alternative, the host memory controller is allowed to keep the RCD in CKE Power Down mode. In this mode the platform CKEn signals are required to be driven all Low and the CK_t/CK_c clock signals are required to continue running at stable frequency. Just as in the latter case, the host controller is responsible for ensuring the platform CKEn signals are driven all Low.

7.2.1.3 RESET_n Pin

When the Catastrophic Save operation was triggered by the RESET_n pin, the module shall:

- clear Bit 1 (Triggered by START_SAVE) in the CSAVE_INFO register (see 8.1.6.1);
- clear Bit 2 (Triggered by SAVE_n) in the CSAVE_INFO register;
- set Bit 3 (Triggered by RESET_n) in the CSAVE_INFO register; and
- clear the CSAVE_FAIL_INFO0 and CSAVE_FAIL_INFO1 registers (see 8.1.6.2 and 8.1.6.3).

Table 3 defines the critical timing parameters for a Catastrophic Save triggered by RESET_n.

Table 3 — RESET_n Signal Timing

Symbol	Parameter	Min	Typ	Max	Unit
$t_{\text{RESET_PW_LOW}}$	RESET_n Pulse Width LOW Duration	1	-	-	μs
$t_{\text{SRE_Hold}}$	NVDIMM SDRAM Self Refresh Hold Time after end of RESET_n pulse	25	-	-	μs

Figure 10 shows RESET_n trigger mode timing.

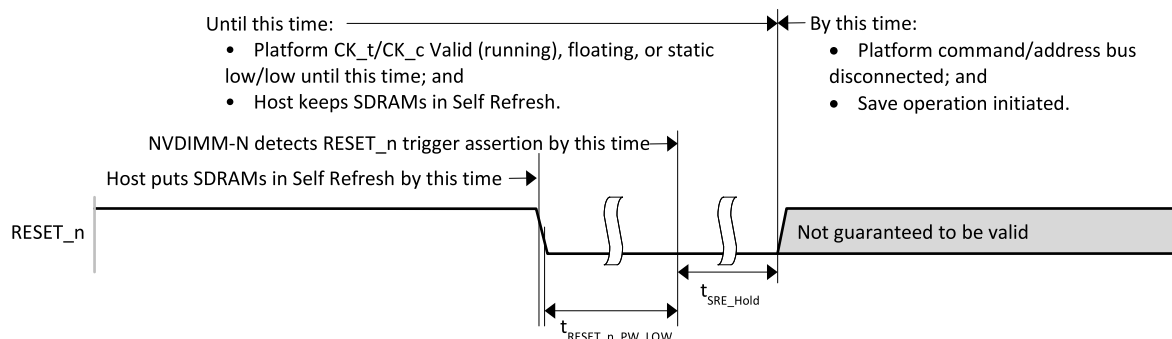


Figure 10 — RESET_n Trigger Mode Timing Diagram

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.

7.2.2 Restore Operation (cont'd)

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).

If a Restore operation is requested while the module is locked (i.e., a write of Typed Block Data type 4h (see 8.4.2.1) to unlock the module (see SIIS) has not completed successfully), then the module shall not perform the Restore operation and shall fail the operation by setting the `RESTORE_ERROR` bit in the `RESTORE_STATUS` register (see 8.1.5.5) and the `SECURITY_ERROR` bit in the `RESTORE_FAIL_INFO` register (see 8.1.5.17).

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:

- if encryption has not been enabled, copy the previously saved SDRAM contents from NVM into the SDRAM; and
- if encryption has been enabled, decrypt the previously saved SDRAM contents from the NVM and store the decrypted data 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).

If an Erase operation is started while the module is locked (i.e., a write of Typed Block Data type 4h (see 8.4.2.1) to unlock the module (see SIIS) has not completed successfully), then the module shall not perform the Erase operation and shall fail the operation by setting the `ERASE_ERROR` bit in the `ERASE_STATUS` register and the `SECURITY_ERROR` bit in the `ERASE_FAIL_INFO` register.

7.2.3 Erase Operation (cont'd)

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) and the ERASE_FAIL_INFO register (see 8.1.5.13);
- 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 from being detected by the SDRAMs on the module. While the module is not armed for Catastrophic Save operation, the module shall enable the RESET_n pin on the module for detection by the SDRAMs on the module. The module shall complete the Arm 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);) and the ARM_FAIL_INFO register (see 8.1.5.16)
- 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 is not required for 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).

Management operation may be started by the host while any other operation is running on the module.

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; and
- 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.

7.2.8 Firmware Operations (cont'd)

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) and the FIRMWARE_OPS_FAIL_INFO register (see 8.1.5.15);
- 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;
- clear the Label Data (see 7.15);
- clear the Vendor Log Page (see 7.12);
- 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 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); and
- 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) and the FACTORY_DEFAULT_STATUS_FAIL_INFO register (see 8.1.5.14);
- 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 operation 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,

7.2.10 Reset Controller Operation (cont'd)

then the module shall:

- not disrupt the DDR4 bus (e.g., interfere with Host access to SDRAMs and the RCD);
- if armed with SAVE_n as a trigger, latch any SAVE_n pin assertions and respond to them when the module becomes available;
- if armed with RESET_n as a trigger, latch any RESET_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-4).

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.

7.2.11 Operational Unit Operations (cont'd)

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 Save and Erase Operation

NOTE: In JESD245B this operation was referred to as “Atomic Arm and Erase.”

Modules may support the Atomic Save and Erase operation. When the Atomic Save and Erase operation is supported, the NVM is not erased immediately; but rather immediately after the next Catastrophic Save has been triggered... 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 Save and Erase operation. If a Catastrophic Save trigger is detected, as defined in 8.1.8.7, during the Atomic Save and Erase operation, the controller shall pause the Atomic Save 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 Save 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.

7.2.13.1 Aborting a Catastrophic Save Operation (cont'd)

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:

- the Clear Firmware operation;
- the Generate Firmware Checksum operation;
- the Commit Firmware operation; and
- the 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

NVDIMM-N module requires an Energy Source in order to function correctly. This clause 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 Table 4.

Table 4 — 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 5 or Table 6
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

7.6 Module Firmware (cont'd)

Table 5 defines firmware image header format 0.

Table 5 — Firmware Image Header Format 0

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

Table 6 defines firmware image header format 1.

Table 6 — Firmware Image Header Format 1

Field Name	Byte Offset(s)	Description
Module Manufacturer ID Code	0 – 1	See Table 4
Module Product Identifier	2 – 3	See Table 4
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 8.1.2.8), if any
Reserved	15 – 18	Reserved

Table 6 — Firmware Image Header Format 1 (cont'd)

Field Name	Byte Offset(s)	Description
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 Table 4
Firmware Image Checksum	24 – 25	See Table 4
Reserved	26 – 30	See Table 4
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-4).

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