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

3103 North 10th Street

Suite 240 South

Arlington, VA 22201-2107

PRICE: Contact JEDEC

Printed in the U.S.A.

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^2C Bus 11
5.2 I^2C Transactions 11
5.2.1 I^2C Read Byte Transaction 12
5.2.2 I^2C Write Byte Transaction 12
5.2.3 I^2C Read Word Transaction 12
5.2.4 I^2C Write Word Transaction 13
5.2.5 I^2C Block Read Transaction 13
5.2.6 I^2C Block Write Transaction 13
5.2.7 I^2C Transactions to Reserved Offsets 14
5.2.8 Illegal I^2C Transactions 14
5.2.9 I^2C 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^2C Read Byte Transaction 12
Figure 3 — I^2C Write Byte Transaction 12
Figure 4 — I^2C Read Word Transaction 12
Figure 5 — I^2C Write Word Transaction 13
Figure 6 — I^2C Block Read Transaction 13
Figure 7 — I^2C Block Write Transaction 14
Figure 8 — I^2C 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

I^2C 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 (KBKDFVS)
  - https://csrc.nist.gov/groups/STM/cavp/documents/KBKDF800-108/kbkdfvs.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^2C	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 I^2C register occurs. See 7.2.1.

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

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

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

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.

I^2C 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 I^2C 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 I^2C registers. The NVDIMM-N draws power for the Catastrophic Save operation either from the host via the V_12 pins on module edge connector or from a Device-Managed Energy Source.

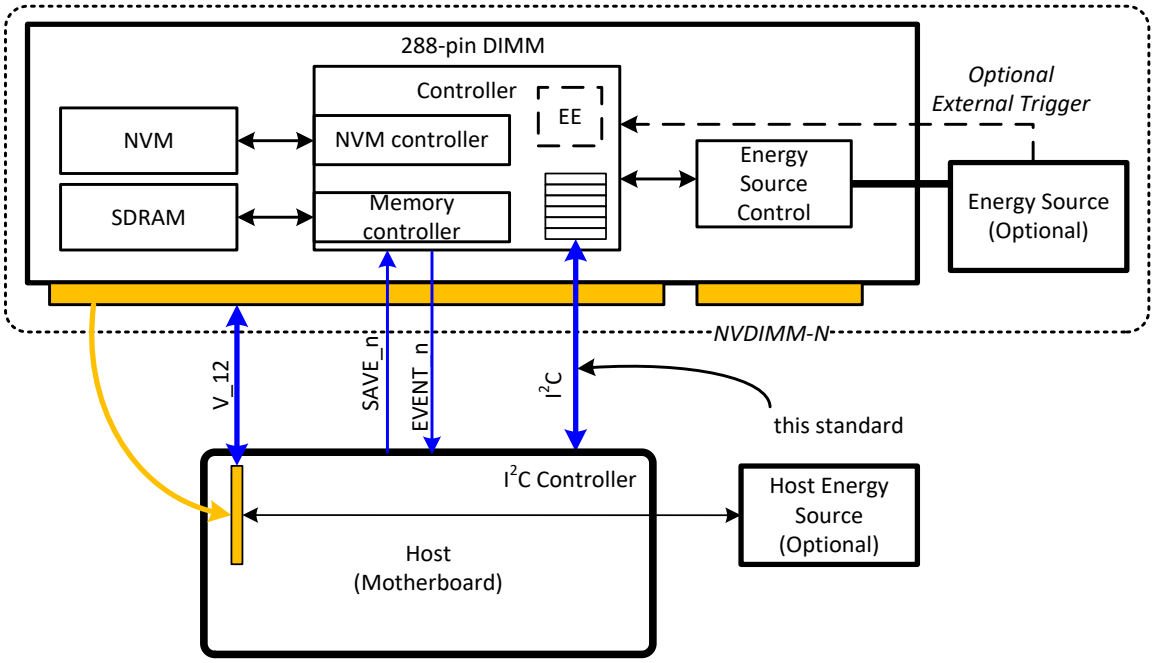

Figure 1 — NVDIMM-N Overview

5 I2C

This clause describes the I^2C interface that a module compliant with this standard shall support.

5.1 I^2C Bus

The module's I^2C bus support shall be compliant with SMBus 3.0. Modules shall support the SMBus 100 kHz Class, the SMBus 400 kHz Class (I^2C Fast-mode), and the SMBus 1 MHz Class (I^2C Fast-mode Plus). Modules shall not clock stretch any I^2C transactions.

The module shall implement an I^2C target with the seven-bit I^2C 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 I^2C Transactions

Modules:

- shall support I^2C Read Byte transactions (see 5.2.1) and I^2C Write Byte transactions (see 5.2.2);
- may support I^2C Read Word transactions (see 5.2.3) and I^2C 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 I^2C Block Read transactions (see 5.2.5) and I^2C Block Write transactions (see 5.2.6) to the TYPED_BLOCK_DATA_OFFSET register in page 3 (see 8.4.5.1). For each I^2C Block Read and I^2C Block Write transaction, the transfer size is 32 bytes.

Modules may support I^2C Read Word, I^2C Write Word, I^2C Block Read, and I^2C Block Write transactions for additional vendor-specific registers.

If a module supports I^2C Read Word and I^2C Write Word transactions, it shall set Bit 3 (I^2C Word Transactions Supported) in the CAPABILITIES1 register (see 8.1.3.2).

If a module supports I^2C Block Read and I^2C Block Write transactions, it shall set Bit 4 (I^2C Block Transactions Supported) in the CAPABILITIES0 register (see 8.1.3.1).

This subclause describe the I^2C transactions between the host and the module. The host is the I^2C controller and the module is the I^2C target. In the figures describing I^2C 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.

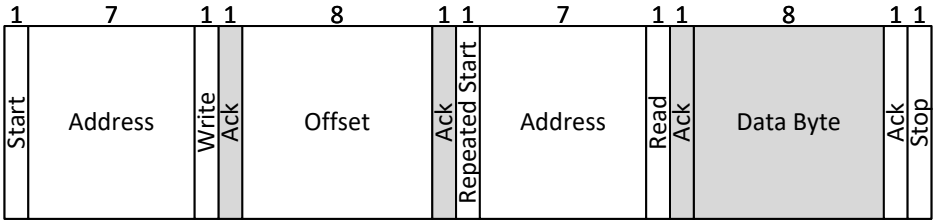

Figure 2 — I²C Read Byte Transaction

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.

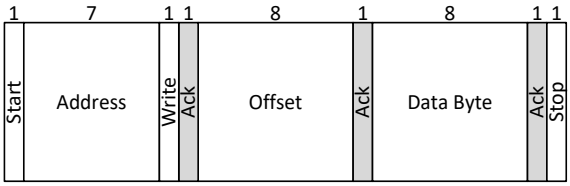

Figure 3 — I²C Write Byte Transaction

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.

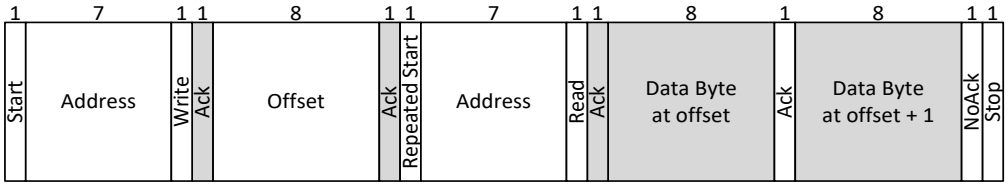

Figure 4 — I²C Read Word Transaction

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.

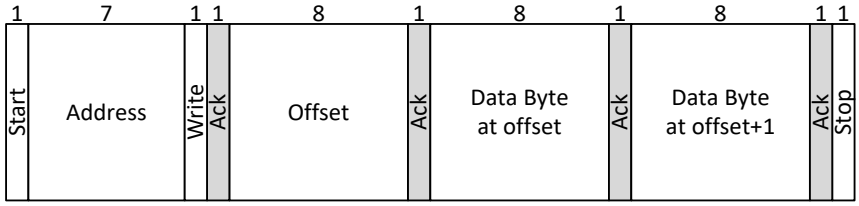

Figure 5 — I²C Write Word Transaction

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.

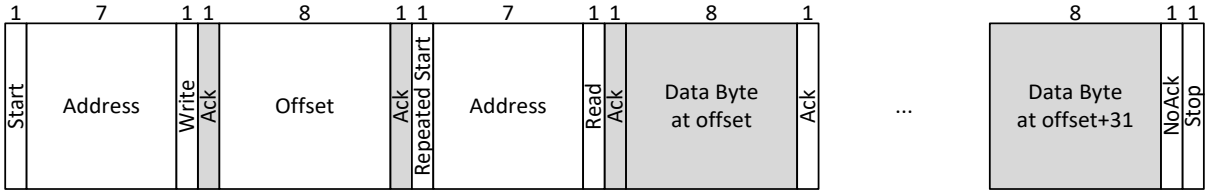

Figure 6 — I²C Block Read Transaction

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.

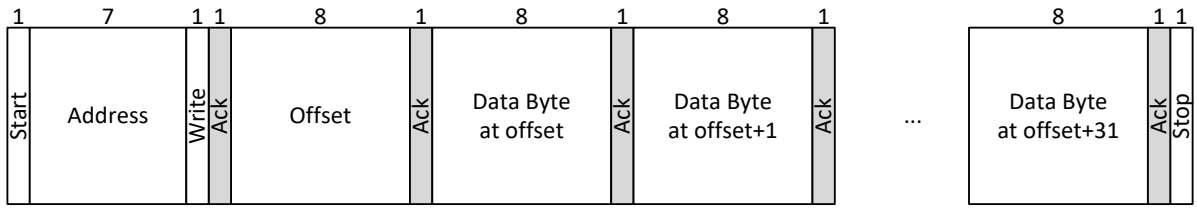

Figure 7 — I²C Block Write Transaction

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 µs 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.

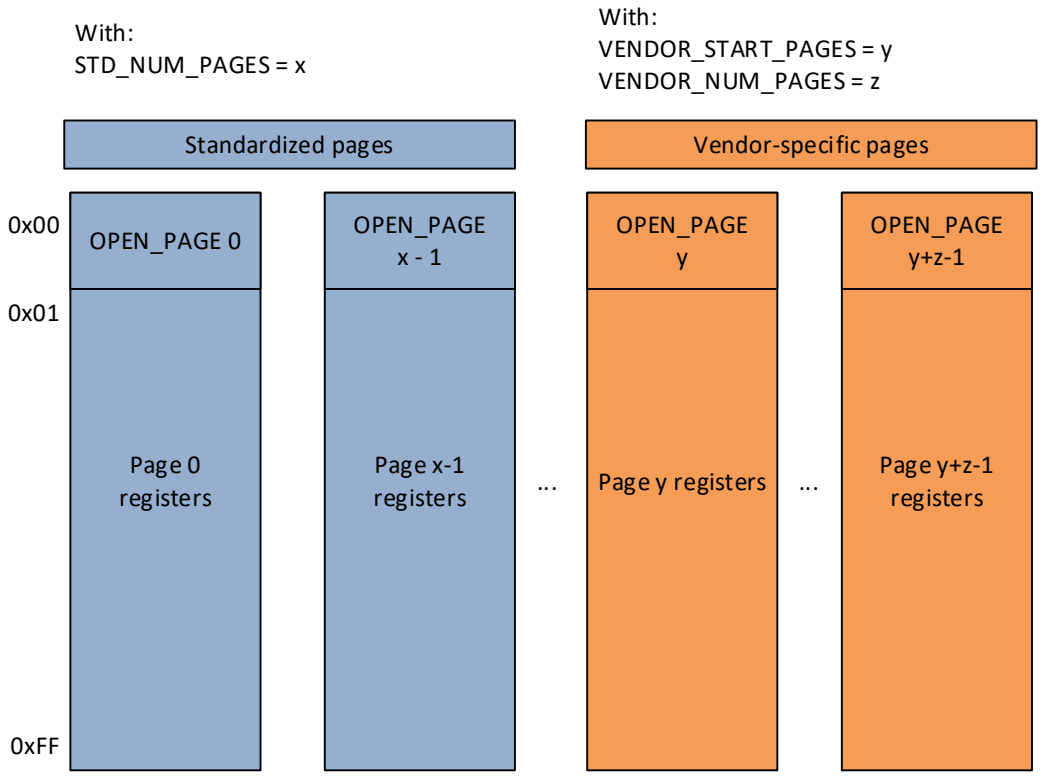

Figure 8 — I²C Paging Mechanism

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 I^2C transactions. Because the SDA signal is pulled up by the system, the host detects a NACK for the Address byte of any I^2C transactions.

After the module completes power on initialization and starts responding to I^2C 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
$t_{SAVE\_PW\_LOW}$	SAVE_n Pulse Width LOW Duration	1	-	-	µs
$t_{SAVE\_PW\_HIGH}$	SAVE_n Pulse Width HIGH Duration	1	-	-	µs
t_{SRE_Hold}	NVDIMM SDRAM Self Refresh Hold Time after end of SAVE_n pulse	25	-	-	µs

Figure 9 shows SAVE_n trigger mode timing.

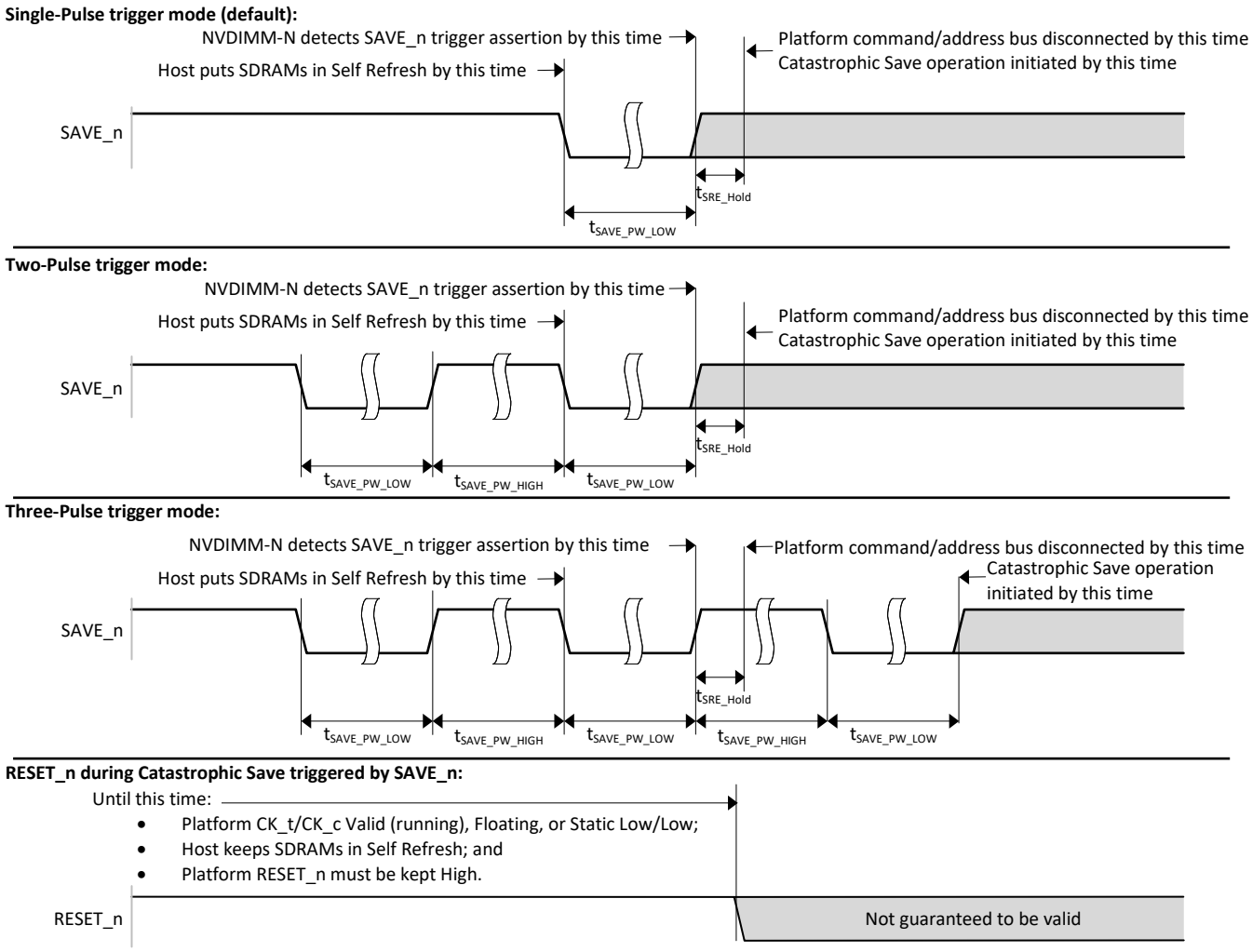

Figure 9 — SAVE_n Trigger Mode Timing Diagram

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
$t_{RESET\_PW\_LOW}$	RESET_n Pulse Width LOW Duration	1	-	-	µs
t_{SRE_Hold}	NVDIMM SDRAM Self Refresh Hold Time after end of RESET_n pulse	25	-	-	µs

Figure 10 shows RESET_n trigger mode timing.

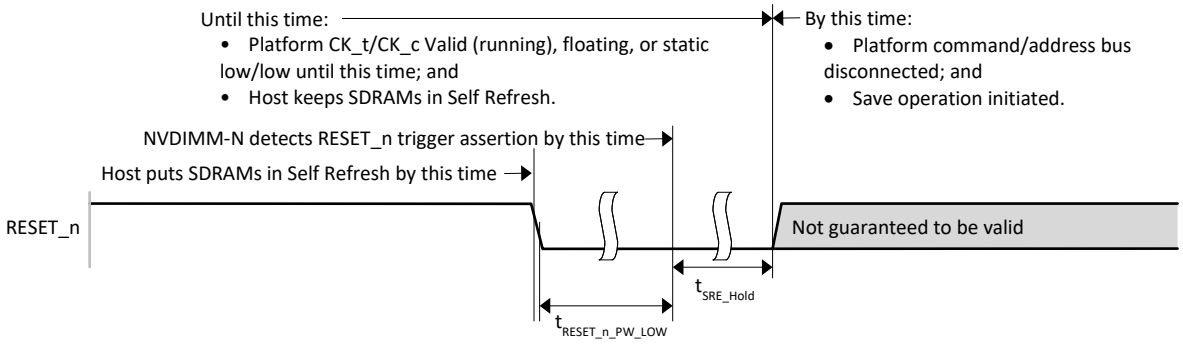

Figure 10 — RESET_n Trigger Mode Timing Diagram

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 Operation 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 V_12 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 V_12 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:

7.7 CRC Algorithm (cont'd)

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

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

7.8 Error Injection

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

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

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

- explicitly disabled by the host; or
- by the Factory Default operation; or

7.8 Error Injection (cont'd)

- disabled after being injected one time, if the ONE_TIME_USE bit in the INJECT_ERROR_TYPE register (see 8.3.3.6) is set.

Error injection shall be disabled by default.

7.9 Statistics

Modules may provide statistics on the following operations:

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

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

Additional statistics may be available in vendor specific registers.

7.10 Thermal

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

Modules shall report temperatures in registers defined in this standard with a minimum resolution of 0.25 °C. Temperature values defined in this standard use the format defined in Table 7.

Table 7 — Temperature Value Bit Definition

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

7.10 Thermal (cont'd)

Examples:

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

7.11 Host Area

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

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

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

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

7.12 Vendor Log Page

Modules shall support a Vendor Log Page that provides information useful to triage issues seen on the module. A module may use the VENDOR_LOG_PAGE_SIZE register (see 8.1.3.30) to communicate the number of valid vendor-specific bytes by specifying a non-zero value. If the module uses the VENDOR_LOG_PAGE_SIZE to communicate the number of valid bytes, the size is limited to a maximum value of 8160 bytes. The operational unit size shall be defined by the vendor (see 8.4.2.10). The size must be defined in multiples of 32 bytes.

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

7.13 Vendor-Specific

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

7.14 LCOM Interface

A module controller indicates LCOM interface (see JESD82-31A (DDR4RCD02)) support by setting status Bit 7 (LCOM Interface Supported) in the CAPABILITIES0 register (see 8.1.3.1). If the optional LCOM interface is supported, the module memory controller can communicate with the RCD through the LCOM interface. LCOM supports the following functions:

7.14 LCOM Interface (cont'd)

- controller sends command frames to the RCD; and
- controller receives read data frames from the RCD.

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

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

7.15 Label Data

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

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

7.16 Catastrophic Save Transition Registers

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

The host should program:

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

Table 8 defines the format of the register at the lowest offset of a pair of registers holding an SDRAM Mode Register value.

Table 8 — Mode Register – Lower Byte

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

7.16 Catastrophic Save Transition Registers (cont'd)

Table 9 defines the format of the register at the highest offset of a pair of registers holding an SDRAM Mode Register value.

Table 9 — Mode Register – Upper Byte

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

7.17 Security Protocol Data

If modules support encryption, they shall support the Security Protocol Data type. Security Protocol Data is Typed Block Data that is transferred between the host and the module to manage the encryption and security aspects of the module. The Operational Unit size for Security Protocol Data shall be 256 bytes.

The SECURITY_PROTOCOL_TYPE register (see 8.1.4.11), the SECURITY_PROTOCOL_SPECIFIC0 register (see 8.1.4.9), and the SECURITY_PROTOCOL_SPECIFIC1 register (see 8.1.4.10) describe the Security Protocol Data type information contained in the Typed Block Data. These registers are set by the host prior to transferring Security Protocol Data between the host and the module.

To read the Security Protocol Data, the host should follow the Reading Typed Block Data workflow (see 9.8.1). This is used to map the TCG IF-RECV function (see SIIS).

To write the Security Protocol Data, the host should follow the Writing Typed Block Data workflow (see 9.8.2). This is used to map the TCG IF-SEND function (see SIIS).

8 Register Map

This clause describes the I^2C registers that are defined in each supported page.

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

- All register values are 0-based unless stated otherwise in the register description.
- In register definition tables, "Access" means host access property (e.g., RW, RO, or WO).
- In register definition tables, "Mand" means mandatory (e.g., Y, N, or C; where Y means yes, N means no, and C means conditional where the conditions are described in the register description).
- In register definition tables, "Persist" means persistent through power cycles (e.g., Y or N).
- In register definition tables, "Default" means the value that will be set in register when Factory Default operation is completed.

The host access properties are:

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

8.1 Page 0 Register Map

The registers in page 0 are organized based on categories.

8.1 Page 0 Register Map (cont'd)

Table 10 shows the layout of the categories in page 0.

Table 10 — Page 0 Register Map Categories

Offset	Category	Description
0x00 – 0x03	Paging Mechanism	Registers related to I^2C page support
0x04 – 0x0F	Version	Registers providing version information
0x10 – 0x3F	Characteristics	Registers providing characteristics information
0x40 – 0x5F	Runtime Command	Registers related to runtime commands
0x60 – 0x7F	Runtime Command Status (part 1 of 2)	Registers providing runtime command status information
0x80 – 0x87	Catastrophic Save	Registers providing Catastrophic Save information
0x88 – 0x8F	Runtime Command Status (part 2 of 2)	Registers providing runtime command status information
0x90 – 0x9F	Thresholds	Registers related to thresholds
0xA0 – 0xBF	Module	Registers providing module related information
0xC0 – 0xDF	NVM Subsystem	Registers providing NVM subsystem related information
0xE0 – 0xFF	Reserved	Reserved

Table 11 lists the registers that are in page 0.

Table 11 — Page 0 Register Map

Offset	Register Name	Clause
Paging Mechanism Registers		
0x00	OPEN_PAGE	8.1.1.1
0x01	STD_NUM_PAGES	8.1.1.2
0x02	VENDOR_START_PAGES	8.1.1.3
0x03	VENDOR_NUM_PAGES	8.1.1.4
Version Registers		
0x04	HWREV	8.1.2.1
0x05	Reserved	
0x06	SPECREV	8.1.2.2
0x07	SLOT0_FWREV0	8.1.2.3
0x08	SLOT0_FWREV1	8.1.2.4
0x09	SLOT1_FWREV0	8.1.2.5
0x0A	SLOT1_FWREV1	8.1.2.6
0x0B	SLOT0_SUBFWREV	8.1.2.7
0x0C	SLOT1_SUBFWREV	8.1.2.8
0x0D-0x0F	Reserved	

Table 11 — Page 0 Register Map (cont'd)

Offset	Register Name	Clause
Characteristics Registers		
0x10	CAPABILITIES0	8.1.3.1
0x11	CAPABILITIES1	8.1.3.2
0x12-0x13	Reserved	
0x14	ENERGY_SOURCE_POLICY	8.1.3.3
0x15	HOST_MAX_OPERATION_RETRY	8.1.3.4
0x16	CSAVE_TRIGGER_SUPPORT	8.1.3.5
0x17	EVENT_NOTIFICATION_SUPPORT	8.1.3.6
0x18	CSAVE_TIMEOUT0	8.1.3.7
0x19	CSAVE_TIMEOUT1	8.1.3.8
0x1A	PAGE_SWITCH_LATENCY0	8.1.3.9
0x1B	PAGE_SWITCH_LATENCY1	8.1.3.10
0x1C	RESTORE_TIMEOUT0	8.1.3.11
0x1D	RESTORE_TIMEOUT1	8.1.3.12
0x1E	ERASE_TIMEOUT0	8.1.3.13
0x1F	ERASE_TIMEOUT1	8.1.3.14
0x20	ARM_TIMEOUT0	8.1.3.15
0x21	ARM_TIMEOUT1	8.1.3.16
0x22	FIRMWARE_OPS_TIMEOUT0	8.1.3.17
0x23	FIRMWARE_OPS_TIMEOUT1	8.1.3.18
0x24	ABORT_CMD_TIMEOUT	8.1.3.19
0x25 - 0x27	Obsolete	
0x27	MAX_RUNTIME_POWER0	8.1.3.20
0x28	MAX_RUNTIME_POWER1	8.1.3.21
0x29	CSAVE_POWER_REQ0	8.1.3.22
0x2A	CSAVE_POWER_REQ1	8.1.3.23
0x2B	CSAVE_IDLE_POWER_REQ0	8.1.3.24
0x2C	CSAVE_IDLE_POWER_REQ1	8.1.3.25
0x2D	CSAVE_MIN_VOLT_REQ0	8.1.3.26
0x2E	CSAVE_MIN_VOLT_REQ1	8.1.3.27
0x2F	CSAVE_MAX_VOLT_REQ0	8.1.3.28
0x30	CSAVE_MAX_VOLT_REQ1	8.1.3.29
0x31	VENDOR_LOG_PAGE_SIZE	8.1.3.30
0x32	REGION_BLOCK_SIZE	8.1.3.31
0x33	OPERATIONAL_UNIT_OPS_TIMEOUT0	8.1.3.32
0x34	OPERATIONAL_UNIT_OPS_TIMEOUT1	8.1.3.33
0x35	FACTORY_DEFAULT_TIMEOUT0	8.1.3.34
0x36	FACTORY_DEFAULT_TIMEOUT1	8.1.3.35
0x37	Reserved	
0x38	MIN_OPERATING_TEMP0	8.1.3.36
0x39	MIN_OPERATING_TEMP1	8.1.3.37
0x3A	MAX_OPERATING_TEMP0	8.1.3.38
0x3B	MAX_OPERATING_TEMP1	8.1.3.39
0x3C-0x3F	Reserved	

Table 11 — Page 0 Register Map (cont'd)

Offset	Register Name	Clause
Runtime Command Registers		
0x40	NVDIMM_MGT_CMD0	8.1.4.1
0x41	NVDIMM_MGT_CMD1	8.1.4.2
0x42	Reserved	
0x43	NVDIMM_FUNC_CMD	8.1.4.3
0x44	Reserved	
0x45	ARM_CMD	8.1.4.4
0x46	Reserved	
0x47	SET_EVENT_NOTIFICATION_CMD	8.1.4.5
0x48	Reserved	
0x49	SET_ES_POLICY_CMD	8.1.4.6
0x4A	FIRMWARE_OPS_CMD	8.1.4.7
0x4B	OPERATIONAL_UNIT_OPS_CMD	8.1.4.8
0x4C	SECURITY_PROTOCOL_SPECIFIC0	8.1.4.9
0x4D	SECURITY_PROTOCOL_SPECIFIC1	8.1.4.10
0x4E	SECURITY_PROTOCOL_TYPE	8.1.4.11
0x4F-0x5F	Reserved	
Runtime Command Status Registers (part 1 of 2)		
0x60	NVDIMM_READY	8.1.5.1
0x61	NVDIMM_CMD_STATUS0	8.1.5.2
0x62	NVDIMM_CMD_STATUS1	8.1.5.3
0x63	Reserved	
0x64	CSAVE_STATUS	8.1.5.4
0x65	Reserved	
0x66	RESTORE_STATUS	8.1.5.5
0x67	Reserved	
0x68	ERASE_STATUS	8.1.5.6
0x69	Reserved	
0x6A	ARM_STATUS	8.1.5.7
0x6B	Reserved	
0x6C	FACTORY_DEFAULT_STATUS	8.1.5.8
0x6D	Reserved	
0x6E	SET_EVENT_NOTIFICATION_STATUS	8.1.5.9
0x6F	Reserved	
0x70	SET_ES_POLICY_STATUS	8.1.5.10
0x71	FIRMWARE_OPS_STATUS	8.1.5.11
0x72	OPERATIONAL_UNIT_OPS_STATUS	8.1.5.12
0x73	ERASE_FAIL_INFO	8.1.5.13
0x74	FACTORY_DEFAULT_STATUS_FAIL_INFO	8.1.5.14
0x75	FIRMWARE_OPS_FAIL_INFO	8.1.5.15
0x76	ARM_FAIL_INFO	8.1.5.16
0x77-0x7F	Reserved	

Table 11 — Page 0 Register Map (cont'd)

Offset	Register Name	Clause
Catastrophic Save Registers		
0x80	CSAVE_INFO	8.1.6.1
0x81-0x83	Reserved	
0x84	CSAVE_FAIL_INFO0	8.1.6.2
0x85	CSAVE_FAIL_INFO1	8.1.6.3
0x86-0x87	Reserved	
Runtime Command Status Registers (part 2 of 2)		
0x88	RESTORE_FAIL_INFO	8.1.5.17
0x89-0x8E	Reserved	
0x8F	OPERATIONAL_UNIT_FAIL_INFO	8.1.5.18
Threshold Registers		
0x90	NVM_LIFETIME_ERROR_THRESHOLD	8.1.7.1
0x91	ES_LIFETIME_ERROR_THRESHOLD	8.1.7.2
0x92	Obsolete	
0x93	Reserved	
0x94	ES_TEMP_ERROR_HIGH_THRESHOLD0	8.1.7.3
0x95	ES_TEMP_ERROR_HIGH_THRESHOLD1	8.1.7.4
0x96	ES_TEMP_ERROR_LOW_THRESHOLD0	8.1.7.5
0x97	ES_TEMP_ERROR_LOW_THRESHOLD1	8.1.7.6
0x98	NVM_LIFETIME_WARNING_THRESHOLD	8.1.7.7
0x99	ES_LIFETIME_WARNING_THRESHOLD	8.1.7.8
0x9A	Obsolete	
0x9B	Reserved	
0x9C	ES_TEMP_WARNING_HIGH_THRESHOLD0	8.1.7.9
0x9D	ES_TEMP_WARNING_HIGH_THRESHOLD1	8.1.7.10
0x9E	ES_TEMP_WARNING_LOW_THRESHOLD0	8.1.7.11
0x9F	ES_TEMP_WARNING_LOW_THRESHOLD1	8.1.7.12
Module Registers		
0xA0	MODULE_HEALTH	8.1.8.1
0xA1	MODULE_HEALTH_STATUS0	8.1.8.2
0xA2	MODULE_HEALTH_STATUS1	8.1.8.3
0xA3-0xA4	Reserved	
0xA5	ERROR_THRESHOLD_STATUS	8.1.8.4
0xA6	Reserved	
0xA7	WARNING_THRESHOLD_STATUS	8.1.8.5
0xA8	Reserved	
0xA9	AUTO_ES_HEALTH_FREQUENCY	8.1.8.6
0xAA	MODULE_OPS_CONFIG	8.1.8.7
0xAB-0xBF	Reserved	
NVM Subsystem Registers		
0xC0	NVM_LIFETIME	8.1.9.1
0xC1-0xFF	Reserved	

8.1.1 Paging Mechanism Registers

The registers in this category are related to the I^2C paging mechanism (see 5.3).

8.1.1.1 OPEN_PAGE – Offset 0x00

Definition	Access	Mand	Persist	Default
[7:0] Open page number	RW	Y	N	0

When the OPEN_PAGE register is read, it returns the current opened page number. When the OPEN_PAGE register is written to, the module shall attempt to set the current opened page number to the value written. The default value of OPEN_PAGE shall be 0.

8.1.1.2 STD_NUM_PAGES – Offset 0x01

Definition	Access	Mand	Persist	Default
[7:0] Number of standard pages supported	RO	Y	Y	>= 4

The STD_NUM_PAGES register returns the highest number of a pages defined by this standard that is supported by the module.

8.1.1.3 VENDOR_START_PAGES – Offset 0x02

Definition	Access	Mand	Persist	Default
[7:0] Start page number for vendor-specific pages supported	RO	Y	Y	>= STD_NUM_PAGES

The VENDOR_START_PAGES register returns the starting page number for vendor-specific pages. The value returned from this register shall be greater than or equal to the value returned from STD_NUM_PAGES register. If the module does not support any vendor-specific pages, the value returned from this register shall be equal to the value returned from STD_NUM_PAGES register.

8.1.1.4 VENDOR_NUM_PAGES – Offset 0x03

Definition	Access	Mand	Persist	Default
[7:0] Number of vendor-specific pages supported	RO	Y	Y	Vendor specific

The VENDOR_NUM_PAGES register returns the number of vendor-specific pages supported by the module. If an implementation does not support any vendor-specific pages, the value returned shall be 0.

8.1.2 Version Registers

The registers in this category provide version information about the module.

8.1.2.1 HWREV – Offset 0x04

Definition	Access	Mand	Persist	Default
[7:0] Controller hardware revision	RO	Y	Y	Vendor specific

The HWREV register returns the controller hardware revision.

8.1.2.2 SPECREV – Offset 0x06

Definition	Access	Mand	Persist	Default
[3:0] Minor Revision [7:4] Major Revision	RO	Y	Y	Vendor specific

The SPECREV register indicates the revision of this standard to which the module complies. Table 12 defines the Major Revision and Minor Revision values. Modules compliant with this standard shall report revision 2.1.

Table 12 — Major Revision And Minor Revision Fields

Standard	Major Revision	Minor Revision
JESD245	1	0
JESD245A	2	0
JESD245B	2	1
JESD245C	2	2
JESD245D	2	3
This standard (i.e., JESD245E)	2	4

8.1.2.3 SLOT0_FWREV0 – Offset 0x07

Definition	Access	Mand	Persist	Default
[7:0] Slot 0 controller firmware revision bits 7:0	RO	Y	Y	Vendor specific

The SLOT0_FWREV0 register returns the least significant byte of the controller firmware revision of the firmware image in slot 0. The SLOT0_FWREV0 register is not impacted by the Factory Default operation.

The combination of the values returned by SLOT0_FWREV0 and SLOT0_FWREV1 shall be non-zero.

8.1.2.4 SLOT0_FWREV1 – Offset 0x08

Definition	Access	Mand	Persist	Default
[7:0] Slot 0 controller firmware revision bits 15:8	RO	Y	Y	Vendor specific

The SLOT0_FWREV1 register returns the most significant byte of the controller firmware revision of the firmware image in slot 0. The SLOT0_FWREV1 register is not impacted by the Factory Default operation.

The combination of the values returned by SLOT0_FWREV0 and SLOT0_FWREV1 shall be non-zero.

8.1.2.5 SLOT1_FWREV0 – Offset 0x09

Definition	Access	Mand	Persist	Default
[7:0] Slot 1 controller firmware revision bits 7:0	RO	Y	Y	0

The SLOT1_FWREV0 register returns the least significant byte of the controller firmware revision of the firmware image in slot 1. If slot 1 does not contain a firmware image or contains an invalid firmware image, the value returned shall be 0. The SLOT1_FWREV0 register is not impacted by the Factory Default operation.

8.1.2.6 SLOT1_FWREV1 – Offset 0x0A

Definition	Access	Mand	Persist	Default
[7:0] Slot 1 controller firmware revision bits 15:8	RO	Y	Y	0

The SLOT1_FWREV1 register returns the most significant byte of the controller firmware revision of the firmware image in slot 1. If slot 1 does not contain a firmware image or contains an invalid firmware image, the value returned shall be 0. The SLOT1_FWREV1 register is not impacted by the Factory Default operation.

8.1.2.7 SLOT0_SUBFWREV – Offset 0x0B

Definition	Access	Mand	Persist	Default
[7:0] Slot 0 subcomponent firmware revision	RO	Y	Y	0

The SLOT0_SUBFWREV register returns the subcomponent firmware revision of the firmware image in slot 0. The SLOT0_SUBFWREV register is not impacted by the Factory Default operation. A value of 0 indicates there is no subcomponent firmware image in slot 0.

8.1.2.8 SLOT1_SUBFWREV – Offset 0x0C

Definition	Access	Mand	Persist	Default
[7:0] Slot 1 subcomponent firmware revision	RO	Y	Y	0

The SLOT1_SUBFWREV register returns the subcomponent firmware revision of the firmware image in slot 1. The SLOT1_SUBFWREV register is not impacted by the Factory Default operation. A value of 0 indicates there is no subcomponent firmware image in slot 1.

8.1.3 Characteristics Registers

The registers in this category provide characteristics information supported by the module.

8.1.3.1 CAPABILITIES0 – Offset 0x10

Definition	Access	Mand	Persist	Default
[0] : Event Notification Supported [1] : Auto Energy Source Health Check Supported [2] : Error Injection Supported [3] : Device Statistics Supported [4] : I^2C Block Transactions Supported [5] : Two-pulse SAVE_n Trigger Mode Supported [6] : Three-pulse SAVE_n Trigger Mode Supported [7] : LCOM Interface Supported	RO	Y	Y	Vendor specific

The CAPABILITIES0 register provides information regarding the capabilities supported by the module. A bit that is set indicates that the module supports the corresponding capability. A bit that is clear indicates the module does not support the corresponding capability.

If Bit 0, Event Notification Supported, is set, the module supports asserting the EVENT_n pin when an enabled event occurs (see 7.4).

If Bit 1, Auto Energy Source Health Check, is set, the module performs periodic Energy Source health checks.

If Bit 2, Error Injection Supported, is set, the module supports error injection (see 7.8).

8.1.3.1 CAPABILITIES0 – Offset 0x10 (cont'd)

If Bit 3, Device Statistics Supported, is set, the module supports collecting the device statistics (see 7.9).

If Bit 4, I^2C Block Transactions Supported, is set, the module supports I^2C Block Read transactions (see 5.2.5) and I^2C Block Write transactions (see 5.2.6) for Typed Block Data.

If Bit 5, Two-pulse SAVE_n Trigger Mode Supported, is set, the module supports two-pulse SAVE_n trigger mode (see 8.1.8.7).

If Bit 6, Three-pulse SAVE_n Trigger Mode Supported, is set, the module supports three-pulse SAVE_n trigger mode (see 8.1.8.7).

If Bit 7, LCOM Interface Supported, is set, the module supports the LCOM interface (see 7.14).

8.1.3.2 CAPABILITIES1 – Offset 0x11

Definition	Access	Mand	Persist	Default
[0] : Atomic Save and Erase Supported [1] : Abort CMD Timeout in Seconds [2] : Operational Unit Buffer Per Typed Block Data [3] : I^2C Word Transactions Supported [5:4] : Reset Controller Scope [6] : Self-encrypting NVDIMM-N Supported and Enabled [7] : Catastrophic Save on RESET_n Supported	RO	Y	Y	Vendor specific

If Bit 0, Atomic Save and Erase Supported, is set, module supports the Atomic Save and Erase operation (see 7.2.12).

If Bit 1, Abort CMD Timeout in Seconds, is set, the abort timeout value listed in the ABORT_CMD_TIMEOUT register is in units of seconds. If Bit 1 is cleared, the abort timeout value listed in the ABORT_CMD_TIMEOUT register is in units of milliseconds.

If Bit 2, Operational Unit Buffer Per Typed Block Data, is set, the module supports an Operational Unit buffer for each Typed Block Data type.

If Bit 3, I^2C Word Transactions Supported, is set, the module supports I^2C Word Read transactions (see 5.2.3) and I^2C Word Write transactions (see 5.2.4) for Typed Block Data.

Table 13 defines the Reset Controller Scope field. The module shall set the Reset Controller Scope field to 0x1 or 0x2.

Table 13 — Reset Controller Scope Field

Value	Description
0x0	Reset Controller operation semantics are not reported
0x1	Reset Controller operation honors SAVE_n or RESET_n, whichever signal is the trigger
0x2	Reset Controller operation does not honor SAVE_n or RESET_n as a trigger
0x3	Reserved; host software should interpret this the same as 0x0
NOTE Modules compliant with JESD245 or JESD245A always reported a value of 0x0 in the Reset Controller Scope field.	

8.1.3.2 CAPABILITIES1 – Offset 0x11 (cont'd)

If Bit 6, Self-encrypting NVDIMM-N Supported. This bit provides an insecure indication that the device supports encryption. If set to '1', the device supports encryption, if cleared to '0', the device does not support encryption. Security Protocol Data (see clause 7.17) is used for secure encryption management (e.g., to securely determine the algorithms supported, enable encryption, securely determine the current state of the device).

If Bit 7, Catastrophic Save on RESET_n, is set, the module supports catastrophic save operation when RESET_n is asserted (see clause 7.2.1.3).

8.1.3.3 ENERGY_SOURCE_POLICY – Offset 0x14

Definition	Access	Mand	Persist	Default
[0] : Device Managed Policy supported [1] : Host Managed Policy supported [7:2] : Reserved	RO	Y	Y	Non-zero value

The ENERGY_SOURCE_POLICY register provides information regarding the Energy Source policy supported by the module. A bit that is set indicates that the module supports the corresponding policy. A bit that is set indicates that the module does not support the corresponding policy.

In Device Managed Policy, the module is responsible for the management of the Energy Source used for the Catastrophic Save operation. In Host Managed Policy, the host is responsible for the management of the Energy Source used for the Catastrophic Save operation.

The default value shall be a non-zero value as either Device Managed Policy or Host Managed Policy shall be supported. A module may support both policies.

8.1.3.4 HOST_MAX_OPERATION_RETRY – Offset 0x15

Definition	Access	Mand	Persist	Default
[1:0] : Maximum Catastrophic Save retry count [3:2] : Maximum Restore retry count [5:4] : Maximum Erase retry count [7:6] : Reserved	RO	Y	Y	Non-zero

The HOST_MAX_OPERATION_RETRY register provides the recommended retry count to the host if a Catastrophic Save, Restore or Erase operation fails or exceeds the maximum timeout value.

After the host aborts a Catastrophic Save operation, the host should wait for the time indicated in the ERASE_TIMEOUT0 and ERASE_TIMEOUT1 registers (see 8.1.3.13 and 8.1.3.14) before issuing a subsequent Catastrophic Save operation.

8.1.3.5 CSAVE_TRIGGER_SUPPORT – Offset 0x16

Definition	Access	Mand	Persist	Default
[0] : SAVE_n Trigger [1] : RESET_n Trigger [7:2] : Reserved	RO	Y	Y	Non-zero

The CSAVE_TRIGGER_SUPPORT register provides the Catastrophic Save triggers supported by the module. A bit that is set indicates that the module supports the corresponding trigger. A bit that is clear indicates that the module does not support the corresponding trigger.

8.1.3.5 CSAVE_TRIGGER_SUPPORT – Offset 0x16 (cont'd)

If Bit 0, SAVE_n trigger, is set, the module is capable of triggering the Catastrophic Save operation based on the SAVE_n pin behavior defined in 8.1.8.7.

If Bit 1, RESET_n trigger, is set, the module is capable of triggering the Catastrophic Save operation based on the RESET_n pin.

The default value shall be a non-zero value as the SAVE_n trigger shall be supported (i.e., the module shall support at least one trigger). A module may support more than one trigger.

8.1.3.6 EVENT_NOTIFICATION_SUPPORT – Offset 0x17

Definition	Access	Mand	Persist	Default
[0] : Persistency notification [1] : Warning threshold notification [2] : Obsolete [3] : Firmware activation notification [7:4] : Reserved	RO	Y	Y	Vendor specific

The EVENT_NOTIFICATION_SUPPORT register provides information on the events for which the module will generate notification. A bit that is set indicates that the module supports generating a notification if the corresponding event occurs. A bit that is clear indicates that the module does not support generating a notification if the corresponding event occurs.

If bit 0 is set, the module supports generating a notification if there is a persistency loss or restored event.

If bit 1 is set, the module supports generating a notification if there is a threshold exceeded or below threshold event.

If bit 3 is set, the module supports generating a notification if new firmware is activated by a Reset Controller operation.

8.1.3.7 CSAVE_TIMEOUT0 – Offset 0x18

Definition	Access	Mand	Persist	Default
[7:0] : Worst case Catastrophic Save completion latency bits 7:0	RO	Y	Y	Vendor specific

The CSAVE_TIMEOUT0 register provides the least significant byte of the worst case Catastrophic Save completion latency in milliseconds or seconds and is used in conjunction with the CSAVE_TIMEOUT1 register (see 8.1.3.8).

8.1.3.8 CSAVE_TIMEOUT1 – Offset 0x19

Definition	Access	Mand	Persist	Default
[6:0] : Worst case Catastrophic Save completion latency bits 14:8 [7] : 0 –Completion latency value in milliseconds 1 –Completion latency value in seconds	RO	Y	Y	Vendor specific

The CSAVE_TIMEOUT1 register provides the upper bits of the worst case Catastrophic Save completion latency and the time unit for the worst case completion latency. If a Catastrophic Save operation is in progress when the host boots, the host should use the CSAVE_TIMEOUT0 and CSAVE_TIMEOUT1 registers to determine when a Catastrophic Save operation has timed out.

8.1.3.8 CSAVE_TIMEOUT1 – Offset 0x19 (cont'd)

If the module is capable of performing an Erase operation as part of a Catastrophic Save operation or supports the Atomic Save and Erase operation, then this value shall reflect the timeout for the combination of an Erase operation and a Catastrophic Save operation.

If Bit 7 is 0, the worst case completion latency value is in units of milliseconds. If Bit 7 is 1, the worst case completion latency value is in units of seconds.

8.1.3.9 PAGE_SWITCH_LATENCY0 – Offset 0x1A

Definition	Access	Mand	Persist	Default
[7:0] : Host wait time for I^2C page switch completion latency bits 7:0	RO	Y	Y	Vendor specific

The PAGE_SWITCH_LATENCY0 register provides the least significant byte of the host wait time for the I^2C page switch completion latency and is used in conjunction with the PAGE_SWITCH_LATENCY1 register (see 8.1.3.10).

8.1.3.10 PAGE_SWITCH_LATENCY1 – Offset 0x1B

Definition	Access	Mand	Persist	Default
[6:0] : Host wait time for I^2C page switch completion latency bits 14:8 [7] : 0 – Completion latency value in milliseconds 1 – Completion latency value in seconds	RO	Y	Y	Vendor specific

The PAGE_SWITCH_LATENCY1 register provides the upper bits of the host wait time for the I^2C page switch completion latency (see 5.3).

If Bit 7 is 0, the completion latency value is in units of milliseconds. If Bit 7 is 1, the completion latency value is in units of seconds.

8.1.3.11 RESTORE_TIMEOUT0 – Offset 0x1C

Definition	Access	Mand	Persist	Default
[7:0] : Worst case Restore completion latency bits 7:0	RO	Y	Y	Vendor specific

The RESTORE_TIMEOUT0 register provides the least significant byte of the worst case Restore completion latency and is used in conjunction with the RESTORE_TIMEOUT1 register (see 8.1.3.12).

8.1.3.12 RESTORE_TIMEOUT1 – Offset 0x1D

Definition	Access	Mand	Persist	Default
[6:0] : Worst case Restore completion latency bits 14:8 [7] : 0 – Completion latency value in milliseconds 1 – Completion latency value in seconds	RO	Y	Y	Vendor specific

The RESTORE_TIMEOUT1 register provides the upper bits of and the time unit for the worst case Restore completion latency (see 7.2.2 and 9.3).

If Bit 7 is 0, the worst case Restore completion latency value is in units of milliseconds. If Bit 7 is 1, the worst case Restore completion latency value is in units of seconds.

8.1.3.13 ERASE_TIMEOUT0 – Offset 0x1E

Definition	Access	Mand	Persist	Default
[7:0] : Worst case Erase completion latency bits 7:0	RO	Y	Y	Vendor specific

The ERASE_TIMEOUT0 register provides the least significant byte of the worst case Erase completion latency and is used in conjunction with the ERASE_TIMEOUT1 register (see 8.1.3.14).

8.1.3.14 ERASE_TIMEOUT1 – Offset 0x1F

Definition	Access	Mand	Persist	Default
[6:0] : Worst case Erase completion latency bits 14:8 [7] : 0 – Completion latency value in milliseconds 1 – Completion latency value in seconds	RO	Y	Y	Vendor specific

The ERASE_TIMEOUT1 register provides the upper bits of and the time unit for the worst case Erase completion latency (see 7.2.3 and 9.5).

If Bit 7 is 0, the worst case Erase completion latency value is in units of milliseconds. If Bit 7 is 1, the worst case Erase completion latency value is in units of seconds.

8.1.3.15 ARM_TIMEOUT0 – Offset 0x20

Definition	Access	Mand	Persist	Default
[7:0] : Worst case Arm completion latency bits 7:0	RO	Y	Y	Vendor specific

The ARM_TIMEOUT0 register provides the least significant byte of the worst case Arm completion latency and is used in conjunction with the CSAVE_TIMEOUT1 register (see 8.1.3.16).

8.1.3.16 ARM_TIMEOUT1 – Offset 0x21

Definition	Access	Mand	Persist	Default
[6:0] : Worst case Arm completion latency bits 14:8 [7] : 0 – Completion latency value in milliseconds 1 – Completion latency value in seconds	RO	Y	Y	Vendor specific

The ARM_TIMEOUT1 register provides the upper bits of and the time unit for the worst case Arm completion latency (see 7.2.4 and 9.4).

If Bit 7 is 0, the worst case Arm completion latency value is in units of milliseconds. If Bit 7 is 1, the worst case Arm completion latency value is in units of seconds.

If the module supports the Atomic Save and Erase operation, then this value shall reflect the timeout for the combination of an Arm operation and an Erase operation.

8.1.3.17 FIRMWARE_OPS_TIMEOUT0 – Offset 0x22

Definition	Access	Mand	Persist	Default
[7:0] : Worst case Firmware Operations completion latency bits 7:0	RO	Y	Y	Vendor specific

The FIRMWARE_OPS_TIMEOUT0 register provides the least significant byte of the worst case Firmware Operations completion latency and is used in conjunction with the CSAVE_TIMEOUT1 register (see 8.1.3.18).

8.1.3.18 FIRMWARE_OPS_TIMEOUT1 – Offset 0x23

Definition	Access	Mand	Persist	Default
[6:0] : Worst case Firmware Operations completion latency bits 14:8 [7] : 0 – Completion latency value in milliseconds 1 – Completion latency value in seconds	RO	Y	Y	Vendor specific

The FIRMWARE_OPS_TIMEOUT1 register provides the upper bits of and the time unit for the worst case Firmware Operations completion latency (see 7.2.8 and 9.7).

If Bit 7 is 0, the worst case Firmware Operations completion latency value is in units of milliseconds. If Bit 7 is 1, the worst case Firmware Operations completion latency value is in units of seconds.

8.1.3.19 ABORT_CMD_TIMEOUT – Offset 0x24

Definition	Access	Mand	Persist	Default
[7:0] : Maximum time to abort a running command	RO	Y	Y	Vendor specific

The ABORT_CMD_TIMEOUT register provides the maximum time to abort a running command. If Bit 1 (Abort CMD Timeout in Seconds) of the CAPABILITIES1 register is set, then this field is in seconds. If Bit 1 of the CAPABILITIES1 register is clear, then this field is in milliseconds.

NOTE: Modules compliant with JESD245 or JESD245A always indicated a value in milliseconds in this register, with a maximum possible value of 255 ms. A module compliant with this standard that requires more than 255 ms is not compatible with hosts compliant with JESD245 and JESD245A that strictly enforce this timeout.

8.1.3.20 MAX_RUNTIME_POWER0 – Offset 0x27

Definition	Access	Mand	Persist	Default
[7:0] : Maximum runtime power consumption from the V_12 pin in milliwatts bits 7:0	RO	N	Y	Vendor specific

The MAX_RUNTIME_POWER0 register provides the least significant byte of the maximum runtime power consumption from the V_12 pin in milliwatts and is used in conjunction with the MAX_RUNTIME_POWER0 register (see 8.1.3.21).

This register shall be supported if the module supports the Host Managed Policy or if the module draws power from the V_12 pin to charge an Energy Source connected to the module.

8.1.3.21 MAX_RUNTIME_POWER1 – Offset 0x28

Definition	Access	Mand	Persist	Default
[7:0] : Maximum runtime power consumption from the V_12 pin in milliwatts bits 15:8	RO	N	Y	Vendor specific

The MAX_RUNTIME_POWER1 register provides the most significant byte of the maximum runtime power consumption from the V_12 pin in milliwatts. A module may draw power from the V_12 pin to charge an Energy Source connected to the module or to provide power to components on the module.

This register shall be supported if the module supports the Host Managed Policy or if the module draws power from the V_12 pin to charge an Energy Source connected to the module.

8.1.3.22 CSAVE_POWER_REQ0 – Offset 0x29

Definition	Access	Mand	Persist	Default
[7:0] : Average power required for Catastrophic Save operation in milliwatts bits 7:0	RO	N	Y	Vendor specific

The CSAVE_POWER_REQ0 register provides the least significant byte of the average power required for the Catastrophic Save operation in milliwatts and is used in conjunction with the CSAVE_POWER_REQ1 register (see 8.1.3.23).

This register shall be supported if the module supports the Host Managed Policy.

8.1.3.23 CSAVE_POWER_REQ1 – Offset 0x2A

Definition	Access	Mand	Persist	Default
[7:0] : Average power required for Catastrophic Save operation in milliwatts bits 15:8	RO	N	Y	Vendor specific

The CSAVE_POWER_REQ1 register provides the most significant byte of the average power required for the Catastrophic Save operation in milliwatts.

For a host that uses a shared Energy Source, the host should use the CSAVE_POWER_REQ0 and CSAVE_POWER_REQ1 register values for energy budgeting. The energy required is calculated by multiplying the CSAVE_TIMEOUT0 and CSAVE_TIMEOUT1 register values by the CSAVE_POWER_REQ0 and CSAVE_POWER_REQ1 register values.

If the module is capable of performing an Erase operation as part of a Catastrophic Save operation or supports the Atomic Save and Erase operation, then this value shall reflect the average power value for the combination of an Erase operation and a Catastrophic Save operation.

This register shall be supported if the module supports the Host Managed Policy.

8.1.3.24 CSAVE_IDLE_POWER_REQ0 – Offset 0x2B

Definition	Access	Mand	Persist	Default
[7:0] : Average power required after the Catastrophic Save operation completes in milliwatts bits 7:0	RO	N	Y	Vendor specific

The CSAVE_IDLE_POWER_REQ0 register provides the least significant byte of the average power required by the module after the Catastrophic Save operation completes in milliwatts and is used in conjunction with the CSAVE_IDLE_POWER_REQ1 register (see 8.1.3.25).

This register shall be supported if the module supports the Host Managed Policy.

8.1.3.25 CSAVE_IDLE_POWER_REQ1 – Offset 0x2C

Definition	Access	Mand	Persist	Default
[7:0] : Average power required after the Catastrophic Save operation completes in milliwatts bits 15:8	RO	N	Y	Vendor specific

The CSAVE_IDLE_POWER_REQ1 register provides the most significant byte of the average power required by the module after the Catastrophic Save operation completes in milliwatts.

For a host that use a shared Energy Source, the host should use the CSAVE_IDLE_POWER_REQ0 and CSAVE_IDLE_POWER_REQ1 register values for energy budgeting.

This register shall be supported if the module supports the Host Managed Policy.

8.1.3.26 CSAVE_MIN_VOLT_REQ0 – Offset 0x2D

Definition	Access	Mand	Persist	Default
[7:0] : Minimum voltage the Energy Source has to service during a Catastrophic Save operation in millivolt bits 7:0	RO	N	Y	Vendor specific

The CSAVE_MIN_VOLT_REQ0 register provides the least significant byte of the minimum voltage in millivolts the Energy Source has to service during a Catastrophic Save operation and is used in conjunction with the CSAVE_MIN_VOLT_REQ1 register (see 8.1.3.27).

This register shall be supported if the module supports the Host Managed Policy.

8.1.3.27 CSAVE_MIN_VOLT_REQ1 – Offset 0x2E

Definition	Access	Mand	Persist	Default
[7:0] : Minimum voltage the Energy Source has to service during a Catastrophic Save operation millivolt bits 15:8	RO	N	Y	Vendor specific

The CSAVE_MIN_VOLT_REQ1 register provides the most significant byte of the minimum voltage in millivolts the Energy Source has to service during a Catastrophic Save operation.

This register shall be supported if the module supports the Host Managed Policy. Modules shall support a value of 16A8h (5800) or smaller for the CSAVE_MIN_VOLT_REQ0 and CSAVE_MIN_VOLT_REQ1 registers.

If the module is capable of performing an Erase operation as part of a Catastrophic Save operation or supports the Atomic Save and Erase operation, then this value shall reflect the minimum voltage for the combination of an Erase operation and a Catastrophic Save operation.

8.1.3.28 CSAVE_MAX_VOLT_REQ0 – Offset 0x2F

Definition	Access	Mand	Persist	Default
[7:0] : Maximum voltage the Energy Source has to service during a Catastrophic Save operation in millivolt bits 7:0	RO	N	Y	Vendor specific

The CSAVE_MAX_VOLT_REQ0 register provides the least significant byte of the maximum voltage in millivolts the Energy Source has to service during a Catastrophic Save operation and is used in conjunction with the CSAVE_MAX_VOLT_REQ1 register (see 8.1.3.29). This register shall be supported if the module supports the Host Managed Policy.

8.1.3.29 CSAVE_MAX_VOLT_REQ1 – Offset 0x30

Definition	Access	Mand	Persist	Default
[7:0] : Maximum voltage the Energy Source has to service during a Catastrophic Save operation millivolt bits 15:8	RO	N	Y	Vendor specific

The CSAVE_MAX_VOLT_REQ1 register provides the most significant byte of the maximum voltage in millivolts the Energy Source has to service during a Catastrophic Save operation.

This register shall be supported if the module supports the Host Managed Policy. Modules shall support a value of 35E8h (13800) or smaller for the CSAVE_MAX_VOLT_REQ0 and CSAVE_MAX_VOLT_REQ1 registers.

If the module is capable of performing an Erase operation as part of a Catastrophic Save operation or supports the Atomic Save and Erase operation, then this value shall reflect the maximum voltage for the combination of an Erase operation and a Catastrophic Save operation.

8.1.3.30 VENDOR_LOG_PAGE_SIZE – Offset 0x31

Definition	Access	Mand	Persist	Default
[7:0] : Vendor Log Page size in multiples of 32 bytes	RO	N	Y	Vendor specific

The VENDOR_LOG_PAGE_SIZE register provides the size in multiples of 32 bytes of the Vendor Log Page. The Vendor Log Page provides implementation specific diagnostic data that may be helpful to root cause issues on the module. If the module does not support a Vendor Log Page, this register shall return a data payload of all 0.

8.1.3.31 REGION_BLOCK_SIZE – Offset 0x32

Definition	Access	Mand	Persist	Default
[7:0] : Region size in multiples of 32 bytes	RO	Y	Y	>= 8

The REGION_BLOCK_SIZE register provides the supported region size in multiples of 32 bytes. Region is used in I^2C Block Read transactions and I^2C Block Write transactions to transfer data between the host and the module. The minimum value supported shall be 8. The maximum region size is 8160 bytes.

Example: A REGION_BLOCK_SIZE register value of 32 indicates that the module's region block size is 1024 bytes (i.e., 32 × 32).

8.1.3.32 OPERATIONAL_UNIT_OPS_TIMEOUT0 – Offset 0x33

Definition	Access	Mand	Persist	Default
[7:0] : Worst case Operational Unit Operations completion latency bits 7:0	RO	Y	Y	Vendor specific

The OPERATIONAL_UNIT_OPS_TIMEOUT0 register provides the least significant byte of the worst case Operational Unit operation completion latency in milliseconds or seconds and is used in conjunction with the OPERATIONAL_UNIT_OPS_TIMEOUT1 register (see 8.1.3.33).

8.1.3.33 OPERATIONAL_UNIT_OPS_TIMEOUT1 – Offset 0x34

Definition	Access	Mand	Persist	Default
[6:0] : Worst case Operational Unit Operations completion latency bits 14:8 [7] : 0 – Completion latency value in milliseconds 1 – Completion latency value in seconds	RO	Y	Y	Vendor specific

The OPERATIONAL_UNIT_OPS_TIMEOUT1 register provides the upper bits of the worst case Operational Unit operation completion latency in milliseconds or seconds. The host should use the OPERATIONAL_UNIT_OPS_TIMEOUT0 and OPERATIONAL_UNIT_OPS_TIMEOUT1 registers to determine when an Operational Unit operation has timed out.

If Bit 7 is 0, the worst case completion latency value is in units of milliseconds. If Bit 7 is 1, the worst case completion latency value is in units of seconds.

8.1.3.34 FACTORY_DEFAULT_TIMEOUT0 – Offset 0x35

Definition	Access	Mand	Persist	Default
[7:0] : Worst case Factory Default completion latency bits 7:0	RO	Y	Y	Vendor specific

The FACTORY_DEFAULT_TIMEOUT0 register provides the least significant byte of the worst case Factory Default completion latency in milliseconds or seconds and is used in conjunction with the FACTORY_DEFAULT_TIMEOUT1 register (see 8.1.3.35).

8.1.3.35 FACTORY_DEFAULT_TIMEOUT1 – Offset 0x36

Definition	Access	Mand	Persist	Default
[6:0] : Worst case Factory Default completion latency bits 14:8 [7] : 0 – Completion latency value in milliseconds 1 – Completion latency value in seconds	RO	Y	Y	Vendor specific

The FACTORY_DEFAULT_TIMEOUT1 register provides the upper bits of the worst case Factory Default completion latency in milliseconds or seconds. The host should use the FACTORY_DEFAULT_TIMEOUT0 and FACTORY_DEFAULT_TIMEOUT1 registers to determine when a Factory Default operation has timed out.

If Bit 7 is 0, the worst case completion latency value is in units of milliseconds. If Bit 7 is 1, the worst case completion latency value is in units of seconds.

The host should interpret a factory default latency of zero as indicating that a factory default latency is not reported.

8.1.3.36 MIN_OPERATING_TEMP0 – Offset 0x38

Definition	Access	Mand	Persist	Default
[7:0] : Minimum operating temperature bits 7:0	RO	Y	Y	Vendor specific

The MIN_OPERATING_TEMP0 register provides the least significant byte of the minimum operating temperature in the format defined in 7.10.

NOTE: Modules compliant with JESD245 or JESD245A reported their minimum operating temperature in a one-byte register at offset 0x25, which is obsolete in this standard.

8.1.3.37 MIN_OPERATING_TEMP1 – Offset 0x39

Definition	Access	Mand	Persist	Default
[7:0] : Minimum operating temperature bits 15:8	RO	Y	Y	Vendor specific

The MIN_OPERATING_TEMP1 register provides the most significant byte of the minimum operating temperature in the format defined in 7.10.

NOTE: Modules compliant with JESD245 or JESD245A reported their minimum operating temperature in a one-byte register at offset 0x25, which is obsolete in this standard.

8.1.3.38 MAX_OPERATING_TEMP0 – Offset 0x3A

Definition	Access	Mand	Persist	Default
[7:0] : Maximum operating temperature bits 7:0	RO	Y	Y	Vendor specific

The MAX_OPERATING_TEMP0 register provides the least significant byte of the maximum operating temperature in the format defined in 7.10.

NOTE: Modules compliant with JESD245 or JESD245A reported their maximum operating temperature in a one-byte register at offset 0x26, which is obsolete in this standard.

8.1.3.39 MAX_OPERATING_TEMP1 – Offset 0x3B

Definition	Access	Mand	Persist	Default
[7:0] : Maximum operating temperature bits 15:8	RO	Y	Y	Vendor specific

The MAX_OPERATING_TEMP1 register provides the most significant byte of the maximum operating temperature in the format defined in 7.10.

NOTE: Modules compliant with JESD245 or JESD245A reported their maximum operating temperature in a one-byte register at offset 0x26, which is obsolete in this standard.

8.1.4 Runtime Command Registers

The registers in this category are related to runtime commands supported by the module.

8.1.4.1 NVDIMM_MGT_CMD0 – Offset 0x40

Definition	Access	Mand	Persist	Default
[0] : Start a Reset Controller operation [1] : Clear all Command Status registers [2] : Clear the CSAVE_STATUS register [3] : Clear the RESTORE_STATUS register [4] : Clear the ERASE_STATUS register and the ERASE_FAIL_INFO register [5] : Clear the ARM_STATUS register and the ARM_FAIL_INFO register [6] : Clear the SET_EVENT_NOTIFICATION_STATUS register [7] : Clear the SET_ES_POLICY_STATUS register	WO	Y	N	N/A

The NVDIMM_MGT_CMD0 register allows the host to reset the controller or clear command status register(s) on the module. To start an operation, the host sets the bit corresponding to the operation desired and issues an I^2C Write Byte transaction to this register.

If bit 0 (Start a Reset Controller operation) is set, the module shall ignore all other bits in this register. If bit 0 is clear and bit 1 (Clear all Command Status registers) is set, the module shall ignore all other bits in this register.

The controller self clears the bit(s) after it executes the command.

If Bit 0 is set, the module shall start a Reset Controller operation (see 7.2.10).

If Bit 1 is set, the module shall:

- clear the CSAVE_STATUS register (see 8.1.5.4);
- clear the RESTORE_STATUS register (see 8.1.5.5);
- clear the ERASE_STATUS register (see 8.1.5.6);
- clear the ERASE_FAIL_INFO register (see 8.1.5.13);
- clear Bits 0 and 1 in the ARM_STATUS register (see 8.1.5.7);
- clear the ARM_FAIL_INFO register (see 8.1.5.16);
- clear the FACTORY_DEFAULT_STATUS register (see 8.1.5.8);
- clear the FACTORY_DEFAULT_STATUS_FAIL_INFO register (see 8.1.5.14);
- clear Bits 0 and 1 in the SET_EVENT_NOTIFICATION_STATUS register (see 8.1.5.9);
- clear Bits 0 and 1 in the SET_ES_POLICY_STATUS register (see 8.1.5.10);
- clear all bits except Bit 2 in the FIRMWARE_OPS_STATUS register (see 8.1.5.11);
- clear the FIRMWARE_OPS_FAIL_INFO register (see 8.1.5.15); and
- clear the OPERATIONAL_UNIT_OPS_STATUS register (see 8.1.5.12).
- clear the OPERATIONAL_UNIT_FAIL_INFO register (see 8.1.5.18).

If Bit 2 is set, the module shall clear the CSAVE_STATUS register.

If Bit 3 is set, the module shall clear the RESTORE_STATUS register.

If Bit 4 is set, the module shall clear the ERASE_STATUS register and the ERASE_FAIL_INFO register.

If Bit 5 is set, the module shall clear Bits 0 and 1 in the ARM_STATUS register and clear the ARM_FAIL_INFO register.

8.1.4.1 NVDIMM_MGT_CMD0 – Offset 0x40 (cont'd)

If Bit 6 is set, the module shall clear Bits 0 and 1 in the SET_EVENT_NOTIFICATION_STATUS register.

If Bit 7 is set, the module shall clear Bits 0 and 1 in the SET_ES_POLICY_STATUS register.

8.1.4.2 NVDIMM_MGT_CMD1 – Offset 0x41

Definition	Access	Mand	Persist	Default
[0] : Clear the FACTORY_DEFAULT_STATUS status register and the FACTORY_DEFAULT_STATUS_FAIL_INFO register [1] : Clear the FIRMWARE_OPS_STATUS register and the FIRMWARE_OPS_FAIL_INFO register [2] : Deassert the EVENT_n pin [3] : Clear the OPERATIONAL_UNIT_OPS_STATUS register and the OPERATIONAL_UNIT_FAIL_INFO register [7:4] : Reserved	WO	Y	N	N/A

The NVDIMM_MGT_CMD1 register allows the host to clear command status register(s) on the module. To start an operation, the host sets the bit corresponding to the operation desired and issues an I^2C Write Byte transaction to this register. The controller self clears the bit(s) after it executes the command.

If Bit 0 is set, the module shall clear the FACTORY_DEFAULT_STATUS register (see 8.1.5.8) and clear the FACTORY_DEFAULT_STATUS_FAIL_INFO register (see 8.1.5.14).

If Bit 1 is set, the module shall clear all bits except Bit 2 in the FIRMWARE_OPS_STATUS register (see 8.1.5.11) and clear the FIRMWARE_OPS_FAIL_INFO register (see 8.1.5.15).

If Bit 2 is set, the module shall stop asserting the EVENT_n pin for the events that caused EVENT_n to be asserted.

If Bit 3 is set, the module shall clear the OPERATIONAL_UNIT_OPS_STATUS register and the OPERATIONAL_UNIT_FAIL_INFO register.

8.1.4.3 NVDIMM_FUNC_CMD – Offset 0x43

Definition	Access	Mand	Persist	Default
[0] : START_FACTORY_DEFAULT: Start a Factory Default operation [1] : START_CSAVE : Start a Catastrophic Save operation [2] : START_RESTORE : Start a Restore operation [3] : START_ERASE : Start an Erase operation [4] : ABORT_CURRENT_OP : Abort the current operation, if any [7:5] : Reserved	WO	Y	N	N/A

The NVDIMM_FUNC_CMD register allows the host to start a Factory Default, Catastrophic Save, Restore, Erase or Abort operation on the module. To start an operation, the host sets the bit corresponding to the operation desired and issues an I^2C Write Byte transaction to this register. Upon receipt of the I^2C Write Byte transaction, the module shall update the corresponding In Progress bit in the NVDIMM_CMD_STATUS0 register (see 8.1.5.2).

8.1.4.3 NVDIMM_FUNC_CMD – Offset 0x43 (cont'd)

When the operation completes, the module shall clear the corresponding In Progress bit in the NVDIMM_CMD_STATUS0 register and update the corresponding command status register, if any. If more than one bit is set, the module shall ignore the request. The controller self clears the bit(s) after it executes the command.

If Bit 0, START_FACTORY_DEFAULT, is set, the module shall start a Factory Default operation (see 7.2.9).

If Bit 1, START_CSAVE, is set, the module shall start a Catastrophic Save operation (see 7.2.1).

If Bit 2, START_RESTORE, is set, the module shall start a Restore operation (see 7.2.2).

If Bit 3, START_ERASE, is set, the module shall start an Erase operation (see 7.2.3).

If Bit 4, ABORT_CURRENT_OP, is set, the module shall start an Abort operation (see 7.2.13) if the current operation is defined as abortable in 7.2.13.

8.1.4.4 ARM_CMD – Offset 0x45

Definition	Access	Mand	Persist	Default
[0] : Catastrophic Save on SAVE_n	WO	Y	N	N/A
[1] : Obsolete		N		
[2] : Catastrophic Save on RESET_n		N		
[6:3] : Reserved		N/A		
[7] : Atomic Save and Erase		N		

The ARM_CMD register allows the host to start an Arm operation (see 7.2.4). To enable a trigger, the host sets the bit(s) corresponding to the trigger(s) desired. To disable a trigger, the host clears the bit(s) corresponding to the trigger(s) desired. On receipt of the I^2C Write Byte transaction, the module shall update the Arm In Progress bit in the NVDIMM_CMD_STATUS0 register (see 8.1.5.2).

If Bit 0, Catastrophic Save on SAVE_n, is set, the module shall arm itself to start a Catastrophic Save operation based on the SAVE_n pin as defined in 7.2.1.2.

If Bit 2, Catastrophic Save on RESET_n, is set, the module shall arm itself to start a Catastrophic Save operation based on the RESET_n pin as defined in 7.2.1.3.

NOTE: Setting bit 0 and bit 2 at the same time may lead to unexpected behavior as some modules support a Catastrophic Save based either on SAVE_n or on RESET_n but not both. For modules that support both methods, it is up to the user to ensure that the bit for the method-of-choice is set.

If Bit 7, Atomic Save and Erase, is set in conjunction with one of the Arm triggers, the module shall atomically perform an Erase operation and start a Catastrophic Save operation when a trigger is detected.

8.1.4.5 SET_EVENT_NOTIFICATION_CMD – Offset 0x47

Definition	Access	Mand	Persist	Default
[0] : Persistency notification [1] : Warning threshold notification [2] : Obsolete [3] : Firmware activation notification [7:3] : Reserved	WO	Y	N	N/A

The SET_EVENT_NOTIFICATION_CMD register allows the host to enable or disable notification for the specified event(s). To enable event notification, the host sets the bit(s) corresponding to the event(s) desired. To disable event notification, the host clears the bit(s) corresponding to the event(s) desired.

If Bit 0, Persistency notification, is set, the module shall generate a notification by driving the EVENT_n pin low if the module detects a lost persistency or persistency restored event (see 8.1.8.1).

If Bit 1, Warning threshold notification, is set, the module shall generate a notification by driving the EVENT_n pin low if a warning threshold exceeded or below warning threshold event occurs (see 8.1.8.1).

If Bit 3, Firmware activation notification, is set, the module shall generate a notification by driving the EVENT_n pin low if new firmware is activated by a Reset Controller operation (see 8.1.8.1).

If an event notification is disabled, the module shall not generate a notification when the corresponding event occurred.

8.1.4.6 SET_ES_POLICY_CMD – Offset 0x49

Definition	Access	Mand	Persist	Default
[0] : Device Managed Policy [1] : Host Managed Policy [7:2] : Reserved	WO	N	N	N/A

The SET_ES_POLICY_CMD register allows the host to set the Energy Source policy on the module (see 7.2.6). Only a single bit can be set. To set the Energy Source policy, the host sets the bit corresponding to the desired Energy Source policy.

If Bit 0, Device Managed Policy, is set, the module shall manage the Energy Source used for the Catastrophic Save operation.

If Bit 1, Host Managed Policy, is set, the module shall use power from the V_12 pin for the Catastrophic Save operation.

The host should set the Energy Source policy on the module before starting the Arm operation.

8.1.4.7 FIRMWARE_OPS_CMD – Offset 0x4A

Definition	Access	Mand	Persist	Default
[0] : Firmware Update Mode - 0: Disable firmware update mode - 1: Enable firmware update mode [1] : Start a Clear Firmware operation [2] : Start a Generate Firmware Checksum operation [3] : Start a Commit Firmware operation [4] : Start a Validate Firmware Header operation [5] : Start a Validate Firmware Image operation [7:6] : Reserved	WO	Y	N	N/A

The FIRMWARE_OPS_CMD register allows the host to start Firmware operations. If more than one bit is set, the module shall ignore the request. To start the desired operation, the host sets the bit corresponding to the desired operation. On receipt of the I^2C Write Byte transaction, the module shall update the Firmware Ops In Progress bit in the NVDIMM_CMD_STATUS0 register (see 8.1.5.2).

Bit 0, Firmware Update Mode, allows the host to enable or disable firmware update mode on the module. A set bit indicates a desire to enable firmware update mode. A clear bit indicates a desire to disable firmware update mode.

If Bit 1, Start a Clear Firmware operation, is set, the module shall start a Clear Firmware operation.

If Bit 2, Start a Generate Firmware Checksum operation, is set, the module shall start a Generate Firmware Checksum operation.

If Bit 3, Start a Commit Firmware operation, is set, the module shall start a Commit Firmware operation.

If Bit 4, Start a Validate Firmware Header operation, is set, the module shall start a Validate Firmware Header operation.

If Bit 5, Start a Validate Firmware Image operation, is set, the module shall start a Validate Firmware Image operation.

8.1.4.8 OPERATIONAL_UNIT_OPS_CMD – Offset 0x4B

Definition	Access	Mand	Persist	Default
[0] : Get Operational Unit [1] : Set Operational Unit [2] : Clear Operational Unit Buffer [3] : Generate Operational Unit Checksum [7:4] : Reserved	WO	Y	N	N/A

The OPERATIONAL_UNIT_OPS_CMD register allows the host to start Operational Unit operations (see 7.2.11) against a specific Typed Block Data. If more than one bit is set, the module shall ignore the request. To start the desired operation, the host sets the bit corresponding to the desired operation. Upon receipt of the I^2C Write Byte transaction, the module shall update the Operational Unit Ops In Progress bit in the NVDIMM_CMD_STATUS1 register (see 8.1.4.2).

If Bit 0, Get Operational Unit, is set, the module shall retrieve the Operational Unit identified by the OPERATIONAL_UNIT_ID0 and OPERATIONAL_UNIT_ID1 registers (see 8.4.2.8 and 8.4.2.9) for the Typed Block Data specified in the TYPED_BLOCK_DATA register (see 8.4.2.1) into an internal buffer.

8.1.4.8 OPERATIONAL_UNIT_OPS_CMD – Offset 0x4B (cont'd)

If Bit 1, Set Operational Unit, is set, the module shall commit the data for the Operational Unit identified by the OPERATIONAL_UNIT_ID0 and OPERATIONAL_UNIT_ID1 registers for the Typed Block Data specified in the TYPED_BLOCK_DATA register to the corresponding location in the module.

If Bit 2, Clear Operational Unit Buffer, is set, the module shall set the internal buffer used for the Operational Unit corresponding to the Typed Block Data specified in the TYPED_BLOCK_DATA register to all zeroes.

If Bit 3, Generate Operational Unit Checksum, is set, the module shall calculate the checksum of the contents of the internal buffer associated with the Operational Unit identified by the OPERATIONAL_UNIT_ID0 and OPERATIONAL_UNIT_ID1 registers for the Typed Block Data specified in the TYPED_BLOCK_DATA register and return the checksum value in OPERATIONAL_UNIT_CRC0 and OPERATIONAL_UNIT_CRC1 registers (see 8.4.2.13 and 8.4.2.14). The checksum shall be calculated using the algorithm described in 7.7.

8.1.4.9 SECURITY_PROTOCOL_SPECIFIC0 – Offset 0x4C

Definition	Access	Mand	Persist	Default
[7:0] Security Protocol Specific bits 7:0	RW	C	N	0

The SECURITY_PROTOCOL_SPECIFIC0 register specifies the least significant byte of the Security Protocol Specific information. This register is mandatory if the module supports encryption.

8.1.4.10 SECURITY_PROTOCOL_SPECIFIC1 – Offset 0x4D

Definition	Access	Mand	Persist	Default
[7:0] Security Protocol Specific bits 15:8	RW	C	N	0

The SECURITY_PROTOCOL_SPECIFIC1 register specifies the most significant byte of the Security Protocol Specific information. This register is mandatory if the module supports encryption.

8.1.4.11 SECURITY_PROTOCOL_TYPE – Offset 0x4E

Definition	Access	Mand	Persist	Default
[7:0] Security Protocol Type	RW	C	N	0

The SECURITY_PROTOCOL_TYPE register specifies the type of the Security Protocol information contained in the Security Protocol Data (see 7.17). This register is mandatory if the module supports encryption.

8.1.5 Runtime Command Status Registers

The registers in this category provide status information about runtime commands.

8.1.5.1 NVDIMM_READY – Offset 0x60

Definition	Access	Mand	Persist	Default
[7:0] : Status value for NVDIMM controller	RO	Y	N	0

The NVDIMM_READY register provides information on whether the controller is ready for host access after power on or a Reset Controller operation (see 7.1). A value of 0xA5 indicates that the controller is ready for host access. The meanings of all other values are vendor specific. The NVDIMM_READY register is not impacted by the Factory Default operation.

8.1.5.2 NVDIMM_CMD_STATUS0 – Offset 0x61

Definition	Access	Mand	Persist	Default
[0] : 0 – Operation Not In Progress 1 – Operation In Progress [1] : Factory Default In Progress [2] : Catastrophic Save In Progress [3] : Restore In Progress [4] : Erase In Progress [5] : Abort In Progress [6] : Arm In Progress [7] : Firmware Ops In Progress	RO	Y	N	0

The NVDIMM_CMD_STATUS0 register provides information on whether there are operations in progress.

If Bit 0 is clear, the controller is not executing any operation. If Bit 0 is set, the controller is executing an operation.

If Bit 1 is set, the Factory Default operation is currently in progress.

If Bit 2 is set, the Catastrophic Save operation is currently in progress.

If Bit 3 is set, the Restore operation is currently in progress.

If Bit 4 is set, the Erase operation is currently in progress.

If Bit 5 is set, the Abort operation is currently in progress.

If Bit 6 is set, the Arm operation is currently in progress.

If Bit 7 is set, one of the Firmware operations is currently in progress.

The module shall automatically clear the In Progress bits when the corresponding operation completes.

8.1.5.3 NVDIMM_CMD_STATUS1 – Offset 0x62

Definition	Access	Mand	Persist	Default
[0] : Operational Unit Ops In Progress [7:1] : Reserved	RO	Y	N	0

The NVDIMM_CMD_STATUS1 register provides information on whether there are operations in progress.

If Bit 0 is set, one of the Operational Unit operations is currently in progress.

The module shall automatically clear the In Progress bits when the corresponding operation completes.

8.1.5.4 CSAVE_STATUS – Offset 0x64

Definition	Access	Mand	Persist	Default
[0] : CSAVE_SUCCESS [1] : CSAVE_ERROR [2] : CSAVE_REJECT [3] : Reserved [4] : ABORT_SUCCESS [5] : ABORT_ERROR [7:6] : Reserved	RO	Y	Y	0

The CSAVE_STATUS register provides information regarding the last Catastrophic Save operation (see 7.2.1).

If Bit 0, CSAVE_SUCCESS, is set, the last Catastrophic Save operation completed without any errors.

If Bit 1, CSAVE_ERROR, is set, the last Catastrophic Save operation failed or was rejected.

If Bit 2, CSAVE_REJECT, is set, a Catastrophic Save operation was requested via the START_CSAVE bit in the NVDIMM_FUNC_CMD register (see 8.1.4.3) and was rejected before starting.

If Bit 4, ABORT_SUCCESS, is set, the last Catastrophic Save operation was aborted by the host.

If Bit 5, ABORT_ERROR, is set, the abort of the last Catastrophic Save operation failed.

The module clears this register when the host requests clearing this register or all Command Status registers (see 8.1.4.1).

8.1.5.5 RESTORE_STATUS – Offset 0x66

Definition	Access	Mand	Persist	Default
[0] : RESTORE_SUCCESS [1] : RESTORE_ERROR [3:2] : Reserved [4] : ABORT_SUCCESS [5] : ABORT_ERROR [7:6] : Reserved	RO	Y	N	0

The RESTORE_STATUS register provides information regarding the last Restore operation.

If Bit 0, RESTORE_SUCCESS, is set, the last Restore operation completed without any errors.

If Bit 1, RESTORE_ERROR, is set, the last Restore operation failed and information on the failure is reported in the RESTORE_FAIL_INFO register (see 0).

If Bit 4, ABORT_SUCCESS, is set, the last Restore operation was aborted by the host.

If Bit 5, ABORT_ERROR, is set, the abort of the last Restore operation failed.

The module clears this register when the host requests clearing this register or all Command Status registers (see 8.1.4.1).

8.1.5.6 ERASE_STATUS – Offset 0x68

Definition	Access	Mand	Persist	Default
[0] : ERASE_SUCCESS [1] : ERASE_ERROR [3:2] : Reserved [4] : ABORT_SUCCESS [5] : ABORT_ERROR [7:6] : Reserved	RO	Y	N	0

The ERASE_STATUS register provides information regarding the last Erase operation.

If Bit 0, ERASE_SUCCESS, is set, the last Erase operation completed without any errors.

If Bit 1, ERASE_ERROR, is set, the last Erase operation failed. The ERASE_FAIL_INFO register (see 8.1.5.13) may contain additional information.

If Bit 4, ABORT_SUCCESS, is set, the last Erase operation was aborted by the host.

If Bit 5, ABORT_ERROR, is set, the abort of the last Erase operation failed.

The module clears this register when the host requests clearing this register or all Command Status registers (see 8.1.4.1 and 8.1.4.2).

8.1.5.7 ARM_STATUS – Offset 0x6A

Definition	Access	Mand	Persist	Default
[0] : ARM_SUCCESS [1] : ARM_ERROR [2] : SAVE_N_ARMED [3] : RESET_N_ARMED [4] : ABORT_SUCCESS [5] : ABORT_ERROR [7:6] : Reserved	RO	Y	N	0

The ARM_STATUS register provides information regarding the last Arm operation. It is updated after each Arm operation started by the host.

If Bit 0, ARM_SUCCESS, is set, the last Arm operation completed without any errors.

If Bit 1, ARM_ERROR, is set, the last Arm operation failed. The ARM_FAIL_INFO register (see 8.1.5.16) may contain additional information.

If Bit 2, SAVE_N_ARMED, is set, the module is armed to start a Catastrophic Save operation based on the SAVE_n pin as defined in 8.1.8.7. If Bit 2 is clear, the module is not armed to start a Catastrophic Save operation based on the SAVE_n pin.

If Bit 3, RESET_N_ARMED, is set, the module is armed to start a Catastrophic Save operation based on the RESET_n pin. If Bit 3 is cleared, the module is not armed to start a Catastrophic Save operation based on the RESET_n pin.

If Bit 4, ABORT_SUCCESS, is set, the last Arm operation was aborted by the host.

If Bit 5, ABORT_ERROR, is set, the abort of the last Arm operation failed.

The module shall not set Bit 2 and Bit 3 simultaneously.

The module clears ARM_SUCCESS and ARM_ERROR when the host requests clearing this register or all Command Status registers (see 8.1.4.1).

8.1.5.8 FACTORY_DEFAULT_STATUS – Offset 0x6C

Definition	Access	Mand	Persist	Default
[0] : FACTORY_DEFAULT_SUCCESS [1] : FACTORY_DEFAULT_ERROR [3:2] : Reserved [4] : ABORT_SUCCESS [5] : ABORT_ERROR [7:6] : Reserved	RO	Y	N	0

The FACTORY_DEFAULT_STATUS register provides information regarding the last Factory Default operation.

If Bit 0, FACTORY_DEFAULT_SUCCESS, is set, the last Factory Default operation completed without any errors.

If Bit 1, FACTORY_DEFAULT_ERROR, is set, the last Factory Default operation failed. The state of the module after a failed Factory Default operation is non-deterministic. The FACTORY_DEFAULT_STATUS_FAIL_INFO register (see 8.1.5.14) may contain additional information.

If Bit 4, ABORT_SUCCESS, is set, the last Factory Default operation was aborted by the host.

If Bit 5, ABORT_ERROR, is set, the abort of the last Factory Default operation failed.

The module clears this register when the host requests clearing this register or all Command Status registers (see 8.1.4.1 and 8.1.4.2).

8.1.5.9 SET_EVENT_NOTIFICATION_STATUS – Offset 0x6E

Definition	Access	Mand	Persist	Default
[0] : SET_EVENT_NOTIFICATION_SUCCESS [1] : SET_EVENT_NOTIFICATION_ERROR [2] : PERSISTENCY_ENABLED [3] : WARNING_THRESHOLD_ENABLED [4] : Obsolete [5] : FIRMWARE_ACTIVATION_ENABLED [7:6] : Reserved	RO	Y	N	0

The SET_EVENT_NOTIFICATION_STATUS register provides information regarding the last Set Event Notification operation.

If Bit 0, SET_EVENT_NOTIFICATION_SUCCESS, is set, the last Set Event Notification operation completed without any errors.

If Bit 1, SET_EVENT_NOTIFICATION_ERROR, is set, the last Set Event Notification operation failed.

If Bit 2, PERSISTENCY_ENABLED, is set, the module shall generate an event notification when a loss of persistency or persistency restored event has been detected. If Bit 2 is clear, the module shall not generate an event notification when a loss of persistency or persistency restored event has been detected.

If Bit 3, WARNING_THRESHOLD_ENABLED, is set, the module shall generate an event notification when a warning threshold has been exceeded or below warning threshold event has been detected. If Bit 3 is clear, the module shall not generate an event notification when a warning threshold has been exceeded or below warning threshold event has been detected.

If Bit 5, FIRMWARE_ACTIVATION_ENABLED, is set, the module shall generate an event notification when new firmware is activated by a Reset Controller operation. If Bit 5 is clear, the module shall not generate an event notification when new firmware is activated by a Reset Controller operation.

8.1.5.9 SET_EVENT_NOTIFICATION_STATUS – Offset 0x6E (cont'd)

The module clears the SET_EVENT_NOTIFICATION_SUCCESS and SET_EVENT_NOTIFICATION_ERROR bits when the host requests clearing this register or all Command Status registers (see 8.1.4.1).

8.1.5.10 SET_ES_POLICY_STATUS – Offset 0x70

Definition	Access	Mand	Persist	Default
[0] : SET_ES_POLICY_SUCCESS [1] : SET_ES_POLICY_ERROR [2] : DEVICE_MANAGED_POLICY_ENABLED [3] : HOST_MANAGED_POLICY_ENABLED [7:4] : Reserved	RO	Y	N	0

The SET_ES_POLICY_STATUS register provides information regarding the last Set Energy Source Policy operation.

If Bit 0, SET_ES_POLICY_SUCCESS, is set, the last Set Energy Source Policy operation completed without any errors.

If Bit 1, SET_ES_POLICY_ERROR, is set, the last Set Energy Source Policy operation failed.

If Bit 2, DEVICE_MANAGED_POLICY_ENABLED, is set, the module is configured to use a locally attached Energy Source and will monitor the health of the Energy Source. If Bit 2 is set, Bit 3 shall be clear.

If Bit 3, HOST_MANAGED_POLICY_ENABLED, is set, the module is configured to use the V_12 pin as the Energy Source for the Catastrophic Save operation. If Bit 3 is set, Bit 2 shall be clear.

The module clears the SET_ES_POLICY_SUCCESS and SET_ES_POLICY_ERROR bits when the host requests clearing this register or all Command Status registers (see 8.1.4.1).

8.1.5.11 FIRMWARE_OPS_STATUS – Offset 0x71

Definition	Access	Mand	Persist	Default
[0] : FIRMWARE_OPS_SUCCESS [1] : FIRMWARE_OPS_ERROR [2] : FIRMWARE_UPDATE_MODE [3] : FIRMWARE_BLOCK_RECEIVED [4] : ABORT_SUCCESS [5] : ABORT_ERROR [7:6] : Reserved	RO	Y	N	0

The FIRMWARE_OPS_STATUS register provides information regarding the last Firmware operation.

If Bit 0, FIRMWARE_OPS_SUCCESS, is set, the last Firmware operation completed without any errors.

If Bit 1, FIRMWARE_OPS_ERROR, is set, the last Firmware operation failed. The FIRMWARE_OPS_FAIL_INFO register (see 8.1.5.15) may contain additional information.

If Bit 2, FIRMWARE_UPDATE_MODE, is set, the module is in the Firmware Update mode where firmware on the module can be changed. If Bit 2 is clear, firmware on the module cannot be changed.

If Bit 3, FIRMWARE_BLOCK_RECEIVED, is set, the last block has been received successfully by the NVDIMM and the host may proceed with sending another block.

8.1.5.11 FIRMWARE_OPS_STATUS – Offset 0x71 (cont'd)

If Bit 4, ABORT_SUCCESS, is set, the last Firmware operation was aborted by the host.

If Bit 5, ABORT_ERROR, is set, the abort of the last Firmware operation failed.

The module clears all bits in this register except Bit 2, FIRMWARE_UPDATE_MODE, when the host requests clearing this register or all Command Status registers (see 8.1.4.1 and 8.1.4.2).

8.1.5.12 OPERATIONAL_UNIT_OPS_STATUS – Offset 0x72

Definition	Access	Mand	Persist	Default
[0] : OPERATIONAL_UNIT_OPS_SUCCESS [1] : OPERATIONAL_UNIT_OPS_ERROR [3:2] : Reserved [4] : ABORT_SUCCESS [5] : ABORT_ERROR [7:6] : Reserved	RO	Y	N	0

The OPERATIONAL_UNIT_OPS_STATUS register provides information regarding the last Operational Unit operation.

If Bit 0, OPERATIONAL_UNIT_OPS_SUCCESS, is set, the last Operational Unit operation completed without any errors.

If Bit 1, OPERATIONAL_UNIT_OPS_ERROR, is set, the last Operational Unit operation failed and information on the failure is reported in the OPERATIONAL_UNIT_FAIL_INFO register (see 8.1.5.18).

If Bit 4, ABORT_SUCCESS, is set, the last Operational Unit operation was aborted by the host.

If Bit 5, ABORT_ERROR, is set, the abort of the last Operational Unit operation failed.

The module clears all bits in this register when the host requests clearing this register or all Command Status registers (see 8.1.4.1 and 8.1.4.2).

8.1.5.13 ERASE_FAIL_INFO Register – Offset 0x73

Definition	Access	Mand	Persist	Default
[0] : SECURITY ERROR [7:1] : Reserved	RO	Y	N	0

The ERASE_FAIL_INFO register provides additional failure information for the last Erase operation, if any.

If Bit 0, SECURITY_ERROR, is set, the last Erase operation failed due to the module being 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).

The module clears this register when the host requests clearing this register or all Command Status registers (see 8.1.4.1).

8.1.5.14 FACTORY_DEFAULT_STATUS_FAIL_INFO Register – Offset 0x74

Definition	Access	Mand	Persist	Default
[0] : SECURITY_ERROR [7:1] : Reserved	RO	Y	N	0

The FACTORY_DEFAULT_STATUS_FAIL_INFO register provides additional failure information for the last Factory Default operation, if any.

If Bit 0, SECURITY_ERROR, is set, the last Factory Default operation failed due to the module being 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).

The module clears this register when the host requests clearing this register (see 8.1.4.2) or all Command Status registers (see 8.1.4.1).

8.1.5.15 FIRMWARE_OPS_FAIL_INFO Register – Offset 0x75

Definition	Access	Mand	Persist	Default
[0] : SECURITY_ERROR [7:1] : Reserved	RO	Y	N	0

The FIRMWARE_OPS_FAIL_INFO register provides additional failure information for the last Firmware operation, if any.

If Bit 0, SECURITY_ERROR, is set, the last Firmware operation failed due to the module being 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).

The module clears this register when the host requests clearing this register (see 8.1.4.2) or all Command Status registers (see 8.1.4.1).

8.1.5.16 ARM_FAIL_INFO register – Offset 0x76

Definition	Access	Mand	Persist	Default
[0] : SECURITY_ERROR [7:1] : Reserved	RO	Y	N	0

The ARM_FAIL_INFO register provides additional failure information for the last Arm operation, if any.

If Bit 0, SECURITY_ERROR, is set, the last Arm operation failed due to the module being locked (i.e., a write of Typed Block Data type 4h (see 8.1.4.1) to unlock the module (see SIIS) has not completed successfully).

The module clears this register when the host requests clearing this register or all Command Status registers (see 8.1.4.1).

8.1.5.17 RESTORE_FAIL_INFO – Offset 0x88

Definition	Access	Mand	Persist	Default
[0] : INVALID_IMAGE [1] : SECURITY_ERROR [3:2] : Reserved [4] : DRAM_NOT_SELF_REFRESH [5] : CONTROLLER_HARDWARE_ERROR [6] : NVM_CONTROLLER_ERROR [7] : NVM_MEDIA_ERROR	RO	Y	N	0

The RESTORE_FAIL_INFO register provides failure information for the last Restore operation, if any.

If Bit 0, INVALID_IMAGE, is set, the last Restore operation failed due to an invalid or non-existent saved image.

If Bit 1, SECURITY_ERROR, is set, the last Restore operation failed due to the module being 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).

If Bit 4, DRAM_NOT_SELF_REFRESH, is set, the last Restore operation failed due to the module detecting that the SDRAMs were not in Self-Refresh mode (i.e., CKE_n was asserted).

If Bit 5, CONTROLLER_HARDWARE_ERROR, is set, the last Restore operation failed due to the module encountering a controller hardware error.

If Bit 6, NVM_CONTROLLER_ERROR, is set, the last Restore operation failed due to the module encountering a non-volatile media controller error.

If Bit 7, NVM_MEDIA_ERROR, is set, the last Restore operation failed due to the module encountering a non-volatile media error.

The module may set more than one bit in the RESTORE_FAIL_INFO register.

8.1.5.18 OPERATIONAL_UNIT_FAIL_INFO – Offset 0x8F

Definition	Access	Mand	Persist	Default
[0] : ENDURANCE_LIMIT_REACHED [1] : SECURITY_ERROR [2] : TYPED_BLOCK_DATA_SIZE_ERROR [3] : INVALID_SEC_PROTOCOL [4] : INVALID_SEC_CMD [5] : SEC_TRANSFER_LENGTH_ERROR [6] : IF_SEND_CMD_ERROR [7] : Reserved	RO	Y	N	0

The OPERATIONAL_UNIT_FAIL_INFO register provides failure information for the last Operational Unit operation, if any.

If Bit 0, ENDURANCE_LIMIT_REACHED, is set, the Set Operational Unit operation failed due to the module reaching the endurance limit of the media.

If Bit 1, SECURITY_ERROR, is set, the last Set Operational Unit operation or last Get Operational Unit operation failed due to the module being 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).

8.1.5.18 OPERATIONAL_UNIT_FAIL_INFO – Offset 0x8F (cont'd)

If Bit 2, TYPED_BLOCK_DATA_SIZE_ERROR, is set, the last security operation failed due to the module receiving an unsupported Typed Block Data Size from the host.

If Bit 3, INVALID_SEC_PROTOCOL, is set, the last security operation failed due to the module receiving an unsupported security protocol request from the host.

If Bit 4, INVALID_SEC_CMD, is set, the last security operation failed due to the module receiving an unsupported security command from the host.

If Bit 5, SEC_TRANSFER_LENGTH_ERROR, is set, the last security operation failed due to the module receiving an incorrect number of bytes from the host.

If Bit 6, IF_SEND_CMD_ERROR, is set, the last security IF_SEND command failed due to the module receiving successive IF_SEND without IF_RECV.

8.1.6 Catastrophic Save Registers

The registers in this category provide Catastrophic Save related information.

8.1.6.1 CSAVE_INFO – Offset 0x80

Definition	Access	Mand	Persist	Default
[0] : NVM_Data_Valid [1] : Triggered by START_CSAVE [2] : Triggered by SAVE_n [3] : Triggered by RESET_n [7:4] : Reserved	RO	Y	Y	0

The CSAVE_INFO register provides information on whether there is a valid SDRAM image saved in the NVM subsystem and the trigger source of the last Catastrophic Save operation, if any.

If Bit 0, NVM_Data_Valid, is set, the module has a valid SDRAM image saved in the NVM subsystem.

If Bit 1, Triggered by START_CSAVE, is set, the last Catastrophic Save operation was started through the START_CSAVE bit in the NVDIMM_FUNC_CMD register (see 8.1.4.3).

If Bit 2, Triggered by SAVE_n, is set, the last Catastrophic Save operation was started by the SAVE_n pin.

If Bit 3, Triggered by RESET_n, is set, the last Catastrophic Save operation was started by the RESET_n pin.

8.1.6.2 CSAVE_FAIL_INFO0 – Offset 0x84

Definition	Access	Mand	Persist	Default
[0] : VOLTAGE_REGULATOR_FAILED [1] : VDD_LOST [2] : VPP_LOST [3] : VTT_LOST [4] : DRAM_NOT_SELF_REFRESH [5] : CONTROLLER_HARDWARE_ERROR [6] : NVM_CONTROLLER_ERROR [7] : NVM_MEDIA_ERROR	RO	Y	Y	0

The CSAVE_FAIL_INFO0 register provides failure information for the last Catastrophic Save operation, if any.

If Bit 0, VOLTAGE_REGULATOR_FAILED, is set, the last Catastrophic Save operation failed due to the module detecting a voltage regulator failure.

If Bit 1, VDD_LOST, is set, the last Catastrophic Save operation failed due to the module detecting a lost Vdd condition.

If Bit 2, VPP_LOST, is set, the last Catastrophic Save operation failed due to the module detecting a lost Vpp condition.

If Bit3, VTT_LOST, is set, the last Catastrophic Save operation failed due to the module detecting a lost Vtt condition.

If Bit 4, DRAM_NOT_SELF_REFRESH, is set, the last Catastrophic Save operation failed due to the module detecting SDRAM is not in self refresh.

If Bit 5, CONTROLLER_HARDWARE_ERROR, is set, the last Catastrophic Save operation failed due to the module encountering a controller hardware error.

If Bit 6, NVM_CONTROLLER_ERROR, is set, the last Catastrophic Save operation failed due to the module encountering an NVM controller error.

If Bit 7, NVM_MEDIA_ERROR, is set, the last Catastrophic Save operation failed due to the module encountering an NVM media error.

The module may set more than one bit in the CSAVE_FAIL_INFO0 and CSAVE_FAIL_INFO1 registers.

8.1.6.3 CSAVE_FAIL_INFO1 – Offset 0x85

Definition	Access	Mand	Persist	Default
[0] : NOT_ENOUGH_ENERGY_FOR_CSAVE	RO	Y	Y	0
[1] : PARTIAL_DATA_SAVED		N		
[2] : SAVE_ABORT		Y		
[3] : NO_SAVE_N		N(1)		
[4] : INSUFFICIENT_SAVE_N				
[5] : NO_RESET_N		N(1)		
[6]: SECURITY_ERROR				
[7] : Reserved		C		
NOTE 1 Bit 3 and bit 5 shall match the corresponding bits in CSAVE_TRIGGER_SUPPORT register (see 8.1.3.5).				

The CSAVE_FAIL_INFO1 register provides failure information for the last Catastrophic Save operation, if any.

If Bit 0, NOT_ENOUGH_ENERGY_FOR_CSAVE, is set, the last Catastrophic Save operation failed due to the module detecting there is not enough energy for a Catastrophic Save operation. This bit is set only if the module is in Device Managed Policy.

If Bit 1, PARTIAL_DATA_SAVED, is set, not all of the SDRAM data was saved during the last Catastrophic Save operation.

If Bit 2, SAVE_ABORT, is set, the last Catastrophic Save operation was aborted by the host.

If Bit 3, NO_SAVE_N, is set, the module, while it was armed for Catastrophic Save on SAVE_n, experienced a power loss but no Catastrophic Save operation was started as the module did not detect an asserted SAVE_n.

If Bit 4, INSUFFICIENT_SAVE_N, is set, the module, while it was armed for Catastrophic Save on SAVE_n, experienced a power loss but a Catastrophic Save was not started due to insufficient number of SAVE_n pulse(s).

If Bit 5, NO_RESET_N is set, the module, while it was armed for Catastrophic Save on RESET_n, experienced a power loss but no Catastrophic Save operation was started as the module did not detect an asserted RESET_n.

NOTE: Modules compliant with JESD245 and JESD245A always report Bit 2, Bit 3, Bit 4, and Bit 5 clear.

If Bit 6, SECURITY_ERROR, is set, the last Catastrophic Save operation failed due to the module being 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).

The module may set more than one bit in the CSAVE_FAIL_INFO0 and CSAVE_FAIL_INFO1 registers.

8.1.7 Thresholds Registers

The registers in this category are related to the thresholds support provided by the module.

Percentage fields shall contain a valid number between 0 and 100.

NOTE: Numbers between 101 and 255 are reserved.

8.1.7.1 NVM_LIFETIME_ERROR_THRESHOLD – Offset 0x90

Definition	Access	Mand	Persist	Default
[7:0] : NVM lifetime percentage error threshold	RO	Y	Y	Non-zero value

The NVM_LIFETIME_ERROR_THRESHOLD register provides the percentage of NVM lifetime after which the module is not capable of persisting data during a Catastrophic Save operation. The threshold is in 1% granularity.

8.1.7.2 ES_LIFETIME_ERROR_THRESHOLD – Offset 0x91

Definition	Access	Mand	Persist	Default
[7:0] : ES lifetime percentage error threshold	RO	N	Y	Vendor specific

The ES_LIFETIME_ERROR_THRESHOLD register provides the percentage value at which the lifetime of the Energy Source has exceeded its warranty. This register is mandatory if the module is configured for Device Managed Policy. The threshold is in 1% granularity.

For modules configured for Host Managed Policy, the module shall return the value 0.

8.1.7.3 ES_TEMP_ERROR_HIGH_THRESHOLD0 – Offset 0x94

Definition	Access	Mand	Persist	Default
[7:0] : ES temperature error high threshold bits	RO	N	Y	Vendor specific

The ES_TEMP_ERROR_HIGH_THRESHOLD0 register provides the least significant byte of the Energy Source high temperature value at which the Energy Source has exceeded its warranty in the format defined in 7.10. This register is mandatory if the module is configured for Device Managed Policy.

For modules configured for Host Managed Policy, the module shall return the value 0.

NOTE: Modules compliant with JESD245 or JESD245A reported their ES temperature error threshold in a one-byte register at offset 0x92, which is obsolete in this standard.

8.1.7.4 ES_TEMP_ERROR_HIGH_THRESHOLD1 – Offset 0x95

Definition	Access	Mand	Persist	Default
[7:0] : ES temperature error high threshold bits	RO	N	Y	Vendor specific

The ES_TEMP_ERROR_HIGH_THRESHOLD1 register provides the most significant byte of the Energy Source high temperature value at which the Energy Source has exceeded its warranty in the format defined in 7.10. This register is mandatory if the module is configured for Device Managed Policy.

For modules configured for Host Managed Policy, the module shall return the value 0.

NOTE: Modules compliant with JESD245 or JESD245A reported their ES temperature error threshold in a one-byte register at offset 0x92, which is obsolete in this standard.

8.1.7.5 ES_TEMP_ERROR_LOW_THRESHOLD0 – Offset 0x96

Definition	Access	Mand	Persist	Default
[7:0] : ES temperature error low threshold bits	RO	N	Y	Vendor specific

The ES_TEMP_ERROR_LOW_THRESHOLD0 register provides the least significant byte of the Energy Source low temperature value at which the Energy Source has exceeded its warranty in the format defined in 7.10. This register is mandatory if the module is configured for Device Managed Policy.

For modules configured for Host Managed Policy, the module shall return the value 0.

8.1.7.6 ES_TEMP_ERROR_LOW_THRESHOLD1 – Offset 0x97

Definition	Access	Mand	Persist	Default
[7:0] : ES temperature error low threshold bits	RO	N	Y	Vendor specific

The ES_TEMP_ERROR_LOW_THRESHOLD1 register provides the most significant byte of the Energy Source low temperature value at which the Energy Source has exceeded its warranty in the format defined in 7.10. This register is mandatory if the module is configured for Device Managed Policy.

For modules configured for Host Managed Policy, the module shall return the value 0.

8.1.7.7 NVM_LIFETIME_WARNING_THRESHOLD – Offset 0x98

Definition	Access	Mand	Persist	Default
[7:0] : NVM lifetime percentage warning threshold bits	RW	Y	Y	Non-zero value

The NVM_LIFETIME_WARNING_THRESHOLD register provides the warning threshold for the NVM lifetime percentage value. The value of this register shall be greater than NVM_LIFETIME_ERROR_THRESHOLD register. If the host attempts to set this register to a value that is less than or equal to NVM_LIFETIME_ERROR_THRESHOLD, the module shall not change this register value. The threshold is in 1% granularity.

8.1.7.8 ES_LIFETIME_WARNING_THRESHOLD – Offset 0x99

Definition	Access	Mand	Persist	Default
[7:0] : ES lifetime percentage warning threshold bits	RW	N	Y	Vendor specific

The ES_LIFETIME_WARNING_THRESHOLD register provides the warning threshold for the Energy Source lifetime percentage value. The value of this register shall be greater than ES_LIFETIME_ERROR_THRESHOLD register. If the host attempts to set this register to a value that is less than or equal to ES_LIFETIME_ERROR_THRESHOLD, the module shall not change this register value. This register is mandatory if the module is configured for Device Managed Policy. The threshold is in 1% granularity.

For modules configured for Host Managed Policy, the module shall return the value 0.

8.1.7.9 ES_TEMP_WARNING_HIGH_THRESHOLD0 – Offset 0x9C

Definition	Access	Mand	Persist	Default
[7:0] : ES temperature warning high threshold bits	RW	N	Y	Vendor specific

The ES_TEMP_WARNING_HIGH_THRESHOLD0 register provides the least significant byte of the high warning threshold for the Energy Source temperature value in the format defined in 7.10. This register is mandatory if the module is configured for Device Managed Policy.

For modules configured for Host Managed Policy, the module shall return the value 0.

NOTE: Modules compliant with JESD245 or JESD245A supported an ES temperature warning threshold in a one-byte register at offset 0x9A, which is obsolete in this standard.

8.1.7.10 ES_TEMP_WARNING_HIGH_THRESHOLD1 – Offset 0x9D

Definition	Access	Mand	Persist	Default
[7:0] : ES temperature warning high threshold bits	RW	N	Y	Vendor specific

The ES_TEMP_WARNING_HIGH_THRESHOLD1 register provides the most significant byte of the high warning threshold for the Energy Source temperature value in the format defined in 7.10. This register is mandatory if the module is configured for Device Managed Policy.

For modules configured for Host Managed Policy, the module shall return the value 0.

NOTE: Modules compliant with JESD245 or JESD245A supported an ES temperature warning threshold in a one-byte register at offset 0x9A, which is obsolete in this standard.

8.1.7.11 ES_TEMP_WARNING_LOW_THRESHOLD0 – Offset 0x9E

Definition	Access	Mand	Persist	Default
[7:0] : ES temperature warning low threshold bits	RW	N	Y	Vendor specific

The ES_TEMP_WARNING_LOW_THRESHOLD0 register provides the least significant byte of the low warning threshold for the Energy Source temperature value in the format defined in 7.10. This register is mandatory if the module is configured for Device Managed Policy.

For modules configured for Host Managed Policy, the module shall return the value 0.

8.1.7.12 ES_TEMP_WARNING_LOW_THRESHOLD1 – Offset 0x9F

Definition	Access	Mand	Persist	Default
[7:0] : ES temperature warning low threshold bits	RW	N	Y	Vendor specific

The ES_TEMP_WARNING_LOW_THRESHOLD1 register provides the most significant byte of the low warning threshold for the Energy Source temperature value in the format defined in 7.10. This register is mandatory if the module is configured for Device Managed Policy.

For modules configured for Host Managed Policy, the module shall return the value 0.

8.1.8 Module Registers

The registers in this category provide module related information.

8.1.8.1 MODULE_HEALTH – Offset 0xA0

Definition	Access	Mand	Persist	Default
[0] : PERSISTENCY_LOST_ERROR [1] : WARNING_THRESHOLD_EXCEEDED [2] : PERSISTENCY_RESTORED [3] : BELOW_WARNING_THRESHOLD [4] : PERMANENT_HARDWARE_FAILURE [5] : EVENT_N_LOW [6] : ARM_INFO [7] : Reserved	RO	Y	N	0

The MODULE_HEALTH register provides a high level status register that the host can read to determine if there are issues with the module.

If Bit 0, PERSISTENCY_LOST_ERROR, is set, this indicates that the module is not capable of persisting data during the Catastrophic Save operation. This bit shall be a logical OR of all defined bits from the MODULE_HEALTH_STATUS0, MODULE_HEALTH_STATUS1 and ERROR_THRESHOLD_STATUS registers.

If Bit 1, WARNING_THRESHOLD_EXCEEDED, is set, this indicates that one or more warning threshold exceeded events have been detected. This bit shall be a logical OR of all defined bits from the WARNING_THRESHOLD_STATUS register.

If Bit 2, PERSISTENCY_RESTORED, is set, this indicates that the previous condition(s) causing the module to be not capable of persisting data during the Catastrophic Save operation is not present and the module is capable of persisting data during the Catastrophic Save operation.

If Bit 3, BELOW_WARNING_THRESHOLD, is set, this indicates that a previous warning threshold exceeded threshold no longer occurs.

If Bit 4, PERMANENT_HARDWARE_FAILURE, is set, this indicates that the module has a permanent hardware failure and module replacement is needed.

NOTE: Modules compliant with JESD245 and JESD245A always report Bit 4 clear.

If Bit 5, EVENT_N_LOW, is set, this indicates that the module is driving the EVENT_n pin low.

NOTE: Modules compliant with JESD245 and JESD245A always report Bit 5 clear.

If Bit 6 ARM_INFO, is set, the module is Armed for a Catastrophic Save operation. The default value of this bit is 0. Bit 6 is the real-time NVDIMM Arm status bit. If the ARM_INFO bit is set, this indicates the NVDIMM is Armed for a Catastrophic Save operation. If it is not set, the NVDIMM is not Armed. The ARM_INFO status bit is read only and can only be updated by the controller based on whether or not the NVDIMM is armed. ARM status would change as a result of reset conditions or other losses of persistence capability. The host can use this register for real-time device status.

8.1.8.2 MODULE_HEALTH_STATUS0 – Offset 0xA1

Definition	Access	Mand	Persist	Default
[0] : VOLTAGE_REGULATOR_FAILED [1] : VDD_LOST [2] : VPP_LOST [3] : VTT_LOST [4] : DRAM_NOT_SELF_REFRESH [5] : CONTROLLER_HARDWARE_ERROR [6] : NVM_CONTROLLER_ERROR [7] : NVM_LIFETIME_ERROR	RO	Y	N	0

The MODULE_HEALTH_STATUS0 register provides more detailed information regarding the module health.

If Bit 0, VOLTAGE_REGULATOR_FAILED, is set, the controller has detected a voltage regulator failure.

If Bit 1, VDD_LOST, is set, the module has encountered an error on operational voltage regulator output. If voltage regulator is turned OFF, then module does not set this bit.

If Bit 2, VPP_LOST, is set, the module has encountered an error on operational voltage regulator output. If voltage regulator is turned OFF, then module does not set this bit.

If Bit 3, VTT_LOST, is set, the module has encountered an error on operational voltage regulator output. If voltage regulator is turned OFF, then module does not set this bit.

If Bit 4, DRAM_NOT_SELF_REFRESH, is set, the module has detected that the SDRAM was not in self refresh.

If Bit 5, CONTROLLER_HARDWARE_ERROR is set, the module has detected a controller hardware error that causes data persistency to be lost.

If Bit 6, NVM_CONTROLLER_ERROR is set, the module has detected an NVM controller error.

If Bit 7, NVM_LIFETIME_ERROR, is set, the module has detected an NVM lifetime error (see 8.1.7.1).

8.1.8.3 MODULE_HEALTH_STATUS1 – Offset 0xA2

Definition	Access	Mand	Persist	Default
[0] : NOT_ENOUGH_ENERGY_FOR_CSAVE [1] : INVALID_FIRMWARE_ERROR [2] : CONFIG_DATA_ERROR [3] : NO_ES_PRESENT [4] : ES_POLICY_NOT_SET [5] : ES_HARDWARE_FAILURE [6] : ES_HEALTH_ASSESSMENT_ERROR [7] : Reserved	RO	Y	N	0

The MODULE_HEALTH_STATUS1 register provides more detailed information regarding the module health.

If Bit 0, NOT_ENOUGH_ENERGY_FOR_CSAVE, is set, the module has detected that the Energy Source does not have enough energy to support a Catastrophic Save operation. This bit shall only be set when the module is in Device Managed Policy mode.

8.1.8.3 MODULE_HEALTH_STATUS1 – Offset 0xA2 (cont'd)

If Bit 1, INVALID_FIRMWARE_ERROR is set, the module has detected invalid firmware in both slots during power on or during a Reset Controller operation.

If Bit 2, CONFIG_DATA_ERROR, is set, the controller has detected a configuration data error. Typically, this condition will result in a controller initialization error.

If Bit 3, NO_ES_PRESENT, is set, the module has detected that there is no Energy Source. While configured to run in Device Managed Policy, this bit shall be set if there is Energy Source attached to the module. While configured to run in Host Managed Policy, this bit shall be set if the module does not detect any power on the V_12 pin.

If Bit 4, ES_POLICY_NOT_SET, is set, the Energy Source policy has not been set on the module. The module shall set this bit if an Arm operation is executed and the host has not set the Energy Source policy on the module.

If Bit 5, ES_HARDWARE_FAILURE, is set, the module has detected that the attached Energy Source has failed due to a hardware error. This bit shall only be set when the module is in Device Managed Policy mode.

If Bit 6, ES_HEALTH_ASSESSMENT_ERROR, is set, the module has encountered an error while trying to assess the health of the attached Energy Source. This bit shall only be set when the module is in Device Managed Policy mode.

8.1.8.4 ERROR_THRESHOLD_STATUS – Offset 0xA5

Definition	Access	Mand	Persist	Default
[0] : NVM_LIFETIME_ERROR [1] : ES_LIFETIME_ERROR [2] : ES_TEMP_ERROR [7:3] : Reserved	RO	Y	N	0

The ERROR_THRESHOLD_STATUS register provides status regarding the error thresholds on the modules. If a bit is set, this indicates that the error threshold has been met or exceeded. If the current value exceeds both the error and warning thresholds, only the error threshold bit shall be set.

If Bit 0, NVM_LIFETIME_ERROR, is set, the module has detected that the NVM lifetime is either at or exceeds the error threshold (see 8.1.7.1). If the bit is clear, the NVM lifetime is below the error threshold.

If Bit 1, ES_LIFETIME_ERROR, is set, the module has detected that the Energy Source lifetime is either at or exceeds the error threshold (see 8.1.7.2). If the bit is clear, the Energy Source lifetime is below the error threshold.

If Bit 2, ES_TEMP_ERROR, is set, the module has detected that the Energy Source temperature is above the high error threshold (see 8.1.7.3 and 8.1.7.4) or is below the low error threshold (see 8.1.7.5 and 8.1.7.6). If the bit is clear, the Energy Source temperature is at or below the high error threshold and at or above the low error threshold.

8.1.8.5 WARNING_THRESHOLD_STATUS – Offset 0xA7

Definition	Access	Mand	Persist	Default
[0] : NVM_LIFETIME_WARNING [1] : ES_LIFETIME_WARNING [2] : ES_TEMP_WARNING [7:3] : Reserved	RO	Y	N	0

The WARNING_THRESHOLD_STATUS register provides status regarding the warning thresholds on the modules. If a bit is set, this indicates that the warning threshold has been met or exceeded. If the current value exceeds both the error and warning thresholds, only the error threshold bit shall be set.

If Bit 0, NVM_LIFETIME_WARNING, is set, the module has detected that the NVM lifetime is either at or exceeds the warning threshold (see 8.1.7.7). If the bit is clear, the NVM lifetime is below the warning threshold.

If Bit 1, ES_LIFETIME_WARNING, is set, the module has detected that the Energy Source lifetime is either at or exceeds the warning threshold (see 8.1.7.8). If the bit is clear, the Energy Source lifetime is below the warning threshold.

If Bit 2, ES_TEMP_WARNING, is set, the module has detected that the Energy Source temperature is above the high warning threshold (see 8.1.7.9 and 8.1.7.10) or below the low warning threshold (see 8.1.7.11 and 8.1.7.12). If the bit is clear, the Energy Source temperature is at or below the high warning threshold and at or above the low warning threshold.

8.1.8.6 AUTO_ES_HEALTH_FREQUENCY – Offset 0xA9

Definition	Access	Mand	Persist	Default
[2:0] : Value in days [7:3] : Value in weeks	RW	N	Y	>= 1

The AUTO_ES_HEALTH_FREQUENCY register provides the current frequency of the Energy Source health assessment done by the module. This register shall be supported when the module is in Device Managed Policy mode. While the module is in the Host Managed Policy mode, the module shall return the value 0.

The minimum value supported is 1 day. The module needs to be running for the time period in this register before the first Energy Source health check is done. If this register is written to, the new value is used at the next Energy Source health check.

NOTE: An Energy Source health check may have an effect on the longevity of the Energy Source device.

8.1.8.7 MODULE_OPS_CONFIG – Offset 0xAA

Definition	Access	Mand	Persist	Default
[0] : SAVE_N_LOW_DURING_CSAVE [2:1] : SAVE_N Assertion Mode -00: Default Mode (Single-pulse Trigger) -01: Two-pulse Trigger Mode -10: Three-pulse Trigger Mode -11: Reserved [3] : LCOM_ENABLE [7:4] : Reserved	RW	Y	N	0

The MODULE_OPS_CONFIG register provides ability to configure the operational behavior of the module. The host uses this register to adjust the behavior of the module from the default behavior.

If Bit 0, SAVE_N_LOW_DURING_CSAVE, is set, the module shall drive the SAVE_n pin low during the duration of a Catastrophic Save operation. The default behavior of the module is to not drive the SAVE_n pin low during a Catastrophic Save operation.

If Bits[2:1], SAVE_N Assertion Mode, are set to 00b, the module starts a Catastrophic Save operation within 1 µs + t_{SRE_Hold} after the SAVE_n pin is detected low. The controller is required to disconnect the module from the platform command/address bus once SAVE_n is detected Low (i.e., within 1 µs + t_{SRE_Hold} from SAVE_n going low) since those signals are not guaranteed to be valid after that point.

If Bits[2:1], SAVE_N Assertion Mode, are set to 01b and Two-pulse SAVE_N Trigger Mode Supported in the CAPABILITIES0 register is set:

- the first time that the SAVE_n signal is pulsed Low indicates an early warning of an impending Catastrophic Save event due to power failure detection by the system;
- the second time that the SAVE_n signal is pulsed Low indicates the SDRAM is in Self Refresh state and that the module shall start a Catastrophic Save operation within 1 µs + t_{SRE_Hold} from the start of the second low pulse. The controller is required to disconnect the module from the platform command/address bus once SAVE_n is detected Low for a second time (i.e., within 1 µs + t_{SRE_Hold} from the start of the second low pulse) since those signals are not guaranteed to be valid after that point.

In order to use Two-pulse SAVE_N Trigger Mode in platforms where two or more modules use a shared SAVE_n signal, the host is required to configure the NVDIMMs properly so that the modules do not falsely trigger Save events in any of the modules connected to the same SAVE_n net. This restriction does not apply to systems where each NVDIMM module has a dedicated SAVE_n signal in the platform.

If Bits[2:1], SAVE_N Assertion Mode, are set to 10b and Three-pulse SAVE_N Trigger Mode Supported in the CAPABILITIES0 register is set:

- the first time when SAVE_n signal is pulsed Low indicates an early warning of an impending Catastrophic Save event due to power failure detection by the system.
- the second time the SAVE_n signal is pulsed Low indicates the SDRAM is in Self Refresh State. The controller is required to disconnect the module from the platform command/address bus once SAVE_n is detected Low for a second time (i.e., within 1 µs + t_{SRE_Hold} from the start of the second low pulse) since those signals are not guaranteed to be valid after that point.
- the third time the SAVE_n signal is pulsed Low, the module shall start a Catastrophic Save operation within 1 µs from the start of the third low pulse.

8.1.8.7 MODULE_OPS_CONFIG – Offset 0xAA (cont'd)

In order to use Three-pulse SAVE_N Trigger Mode in platforms where two or more modules use a shared SAVE_n signal, the host is required to configure the NVDIMMs properly so that the modules do not falsely trigger Catastrophic Save events in any of the modules connected to the same SAVE_n net. This restriction does not apply to systems where each NVDIMM module has a dedicated SAVE_n signal in the platform.

If Bit 4, RESET_N Assertion Mode is set, the host shall pulse the RESET_ pin low.

If Bit 3, LCOM_ENABLE, is set, the LCOM interface (see 7.14) is enabled.

8.1.9 NVM Subsystem registers

The registers in this category provide NVM subsystem related information.

8.1.9.1 NVM_LIFETIME – Offset 0xC0

Definition	Access	Mand	Persist	Default
[7:0] : NVM lifetime percentage from 0 to 100	RO	Y	Y	Non-zero value

The NVM_LIFETIME register provides the last known NVM lifetime percentage value. The percentage value shall be from 0 to 100 where 100 represents a healthy state. The NVM_LIFETIME register is not impacted by the Factory Default operation.

The module lifetime percentage shall decrease with use.

8.2 Page 1 Register Map

The registers in page 1 are related to Energy Source and organized based on categories. This page is mandatory when the module is in Device Managed Policy mode. When module is in Host Managed Policy mode, the module shall not allow a page switch to this page. Table 14 shows the layout of the categories in page 1.

Table 14 — Page 1 Register Map Categories

Offset	Category	Description
0x00 – 0x03	Paging Mechanism	Registers related to the I^2C paging mechanism
0x04 – 0x0F	Energy Source Version	Registers providing Energy Source version information
0x10 – 0x2F	Energy Source Characteristics	Registers providing Energy Source characteristics information
0x30 – 0x4F	Energy Source Runtime Command	Registers related to Energy Source runtime commands
0x50 – 0x6F	Energy Source Runtime Command Status	Registers providing Energy Source runtime command status information
0x70 – 0xFF	Energy Source	Registers providing Energy Source related information

Table 15 lists the registers that are in page 1.

8.2 Page 1 Register Map (cont'd)

Table 15 — Page 1 Register Map

Offset	Register Name	Clause
Paging Mechanism Registers		
0x00	OPEN_PAGE	8.2.1.1
0x01–0x03	Reserved	
Energy Source Version Registers		
0x04	ES_HWREV	8.2.2.1
0x05	Reserved	
0x06	ES_FWREV0	8.2.2.2
0x07	ES_FWREV1	8.2.2.3
0x08	SLOT0_ES_FWREV0	8.2.2.4
0x09	SLOT0_ES_FWREV1	8.2.2.5
0x0A	SLOT1_ES_FWREV0	8.2.2.6
0x0B	SLOT1_ES_FWREV1	8.2.2.7
0x0C–0x0F	Reserved	
Energy Source Characteristics Registers		
0x10	ES_CHARGE_TIMEOUT0	8.2.3.1
0x11	ES_CHARGE_TIMEOUT1	8.2.3.2
0x12	Obsolete	
0x13	Obsolete	
0x14	ES_ATTRIBUTES	8.2.3.3
0x15	ES_TECH	8.2.3.4
0x16	MIN_ES_OPERATING_TEMP0	8.2.3.5
0x17	MIN_ES_OPERATING_TEMP1	8.2.3.6
0x18	MAX_ES_OPERATING_TEMP 0	8.2.3.7
0x19	MAX_ES_OPERATING_TEMP 1	8.2.3.8
0x16–0x2F	Reserved	
Energy Source Runtime Command Registers		
0x30	ES_FUNC_CMD0	8.2.4.1
0x31-0x4F	Reserved	
Energy Source Runtime Command Status Registers		
0x50	ES_CMD_STATUS0	8.2.5.1
0x51-0x6F	Reserved	
Energy Source Registers		
0x70	ES_LIFETIME	8.2.6.1
0x71	ES_TEMP0	8.2.6.2
0x72	ES_TEMP1	8.2.6.3
0x73	ES_RUNTIME0	8.2.6.4
0x74	ES_RUNTIME1	8.2.6.5
0x75-0xFF	Reserved	

8.2.1 Paging Mechanism Registers

The registers in this category are related to the I^2C paging mechanism (see 5.3).

8.2.1.1 OPEN_PAGE – Offset 0x00

Definition	Access	Mand	Persist	Default
[7:0] Open page number	RW	Y	N	0

This register is the same as the OPEN_PAGE register in page 0 (see 8.1.1.1).

8.2.2 Energy Source Version Registers

The registers in this category provide Energy Source version information.

8.2.2.1 ES_HWREV – Offset 0x04

Definition	Access	Mand	Persist	Default
[7:0] Energy source hardware revision	RO	N	Y	Vendor specific

The ES_HWREV register returns the Energy Source hardware revision.

8.2.2.2 ES_FWREV0 – Offset 0x06

Definition	Access	Mand	Persist	Default
[7:0] Energy source firmware revision bits 7:0	RO	N	Y	Vendor specific

The ES_FWREV0 register returns the least significant byte of the current Energy Source firmware revision.

The combination of the values returned by ES_FWREV0 and ES_FWREV1 shall be non-zero.

8.2.2.3 ES_FWREV1 – Offset 0x07

Definition	Access	Mand	Persist	Default
[7:0] Energy source firmware revision bits 15:8	RO	N	Y	Vendor specific

The ES_FWREV1 register returns the most significant byte of the current Energy Source firmware revision.

The combination of the values returned by ES_FWREV0 and ES_FWREV1 shall be non-zero.

8.2.2.4 SLOT0_ES_FWREV0 – Offset 0x08

Definition	Access	Mand	Persist	Default
[7:0] Slot 0 Energy source firmware revision bits 7:0	RO	N	Y	Vendor specific

The SLOT0_ES_FWREV0 register returns the least significant byte of the Energy Source firmware revision in slot 0. A value of 0 in both SLOT0_ES_FWREV0 and SLOT0_ES_FWREV1 indicates there is no energy source firmware image in slot 0.

NOTE: Modules compliant with JESD245 and JESD245A do not support the SLOT0_ES_FWREV0, SLOT0_ES_FWREV1, SLOT1_ES_FWREV0, and SLOT1_ES_FWREV1 registers.

8.2.2.5 SLOT0_ES_FWREV1 – Offset 0x09

Definition	Access	Mand	Persist	Default
[7:0] Slot 0 Energy source firmware revision bits 15:8	RO	N	Y	Vendor specific

The SLOT0_ES_FWREV1 register returns the most significant byte of the Energy Source firmware revision in slot 0.

8.2.2.6 SLOT1_ES_FWREV0 – Offset 0x0A

Definition	Access	Mand	Persist	Default
[7:0] Slot 1 Energy source firmware revision bits 7:0	RO	N	Y	Vendor specific

The SLOT1_ES_FWREV0 register returns the least significant byte of the Energy Source firmware revision in slot 1. A value of 0 in both SLOT1_ES_FWREV0 and SLOT1_ES_FWREV1 indicates there is no energy source firmware image in slot 1.

8.2.2.7 SLOT1_ES_FWREV1 – Offset 0x0B

Definition	Access	Mand	Persist	Default
[7:0] Slot 1 Energy source firmware revision bits 15:8	RO	N	Y	Vendor specific

The SLOT1_ES_FWREV1 register returns the most significant byte of the Energy Source firmware revision in slot 1.

8.2.3 Energy Source Characteristics Registers

The registers in this category provide Energy Source characteristics information.

8.2.3.1 ES_CHARGE_TIMEOUT0 – Offset 0x10

Definition	Access	Mand	Persist	Default
[7:0] Worst case Energy Source charge time bits 7:0	RO	N	Y	Vendor specific

The ES_CHARGE_TIMEOUT0 returns the least significant byte of the worst case Energy Source charge time in seconds. The combination of ES_CHARGE_TIMEOUT0 and ES_CHARGE_TIMEOUT1 shall be non-zero.

8.2.3.2 ES_CHARGE_TIMEOUT1 – Offset 0x11

Definition	Access	Mand	Persist	Default
[7:0] Worst case Energy Source charge time bits 15:8	RO	N	Y	Vendor specific

The ES_CHARGE_TIMEOUT1 returns the most significant byte of the worst case Energy Source charge time in seconds. The combination of ES_CHARGE_TIMEOUT0 and ES_CHARGE_TIMEOUT1 shall be non-zero.

8.2.3.3 ES_ATTRIBUTES – Offset 0x14

Definition	Access	Mand	Persist	Default
[0] : On Module [1] : Tethered [2] : Shared [7:3] Reserved	RO	N	Y	Non-zero value

The ES_ATTRIBUTES register provides attributes regarding the Energy Source. A set bit indicates that the Energy Source has the corresponding attribute. One or more bits may be set.

If Bit 0, On Module, is set, this indicates that there is an Energy Source on the module (see 7.3.1).

If Bit 1, Tethered, is set, this indicates that there is an Energy Source tethered to the module (see 7.3.2).

If Bit 2, Shared, is set, this indicates that the Energy Source is used by more than one module (see 7.3.4).

8.2.3.4 ES_TECH – Offset 0x15

Definition	Access	Mand	Persist	Default
[0] : Undefined [1] : Supercapacitor [2] : Battery [3] : Hybrid capacitor [7:4] Reserved	RO	N	Y	Non-zero value

The ES_TECH register provides the technology used in the Energy Source. A set bit indicates that the corresponding technology is used in the Energy Source. One or more bits may be set.

If Bit 0, Undefined, is set, this indicates that the technology used in the Energy Source is not defined in this spec.

If Bit 1, Supercapacitor, is set, this indicates that the Energy Source uses supercapacitor.

If Bit 2, Battery, is set, this indicates that the Energy Source uses battery.

If Bit 3, Hybrid capacitor, is set, this indicates that the Energy Source uses hybrid capacitor.

8.2.3.5 MIN_ES_OPERATING_TEMP0 – Offset 0x16

Definition	Access	Mand	Persist	Default
[7:0] Minimum Energy Source operating temperature bits 7:0	RO	N	Y	Vendor specific

The MIN_ES_OPERATING_TEMP0 register returns the least significant byte of the minimum operating temperature of the Energy Source in the format defined in 7.10.

NOTE: Modules compliant with JESD245 or JESD245A reported their minimum Energy Source operating temperature in a one-byte register at offset 0x12, which is obsolete in this standard.

8.2.3.6 MIN_ES_OPERATING_TEMP1 – Offset 0x17

Definition	Access	Mand	Persist	Default
[7:0] Minimum Energy Source operating temperature bits 15:8	RO	N	Y	Vendor specific

The MIN_ES_OPERATING_TEMP1 register returns the most significant byte of the minimum operating temperature of the Energy Source in the format defined in 7.10.

NOTE: Modules compliant with JESD245 or JESD245A reported their minimum Energy Source operating temperature in a one-byte register at offset 0x12, which is obsolete in this standard.

8.2.3.7 MAX_ES_OPERATING_TEMP0 – Offset 0x18

Definition	Access	Mand	Persist	Default
[7:0] Maximum Energy Source operating temperature bits 7:0	RO	N	Y	Vendor specific

The MAX_ES_OPERATING_TEMP0 register returns the least significant byte of the maximum operating temperature of the Energy Source in the format defined in 7.10.

NOTE: Modules compliant with JESD245 or JESD245A reported their maximum Energy Source operating temperature in a one-byte register at offset 0x13, which is obsolete in this standard.

8.2.3.8 MAX_ES_OPERATING_TEMP1 – Offset 0x19

Definition	Access	Mand	Persist	Default
[7:0] Maximum Energy Source operating temperature bits 15:8	RO	N	Y	Vendor specific

The MAX_ES_OPERATING_TEMP1 register returns the most significant byte of the maximum operating temperature of the Energy Source in the format defined in 7.10.

NOTE: Modules compliant with JESD245 or JESD245A reported their maximum Energy Source operating temperature in a one-byte register at offset 0x13, which is obsolete in this standard.

8.2.4 Energy Source Runtime Command Registers

The registers in this category are related to Energy Source runtime commands.

8.2.4.1 ES_FUNC_CMD0 – Offset 0x30

Definition	Access	Mand	Persist	Default
[0] : MANUAL_ES_HEALTH_CHECK [1] : Clear the ES_CMD_STATUS0 register [7:2] : Reserved	WO	N	N	N/A

The ES_FUNC_CMD0 register allows the host to start Energy Source runtime related operations. If more than one bit is set, the module shall ignore the request.

If Bit 0, MANUAL_ES_HEALTH_CHECK, is set, the module shall start a health check of the Energy Source if there is no other Energy Source health check currently running. If there is a currently running Energy Source health check, the module shall ignore the request. On receipt of this write and at completion of the health check, then module shall update the ES_CMD_STATUS0 register.

If Bit 1 is set, the module shall clear the ES_CMD_STATUS0 register.

8.2.5 Energy Source Runtime Command Status Registers

The registers in this category provide Energy Source runtime command status information.

8.2.5.1 ES_CMD_STATUS0 – Offset 0x50

Definition	Access	Mand	Persist	Default
[0] : Health Check in Progress [1] : Health Check Succeeded [2] : Health Check Failed [3] : 0 – Automatic Health Check 1 – Manual Health Check [7:4] : Reserved	RO	N	N	0

The ES_CMD_STATUS0 register provides status information regarding the status of the Energy Source health check operation. Only one bit can be set for Bits 0 to 2.

If Bit 0, Health Check in Progress, is set, the module is executing an Energy Source health check operation.

If Bit 1, Health Check Succeeded, is set, the module has successfully completed an Energy Source health check.

If Bit 2, Health Check Failed, is set, the last Energy Source health check executed by the module failed.

If Bit 3 is clear, the currently running or last completed Energy Source health check operation was an automatic health check. If Bit 3 is set, the currently running or last completed Energy Source health check operation was manually started.

8.2.6 Energy Source Registers

The registers in this category provide Energy Source related information.

8.2.6.1 ES_LIFETIME – Offset 0x70

Definition	Access	Mand	Persist	Default
[7:0] : Energy source lifetime percentage from 0 to 100	RO	N	N	Non-zero value

The ES_LIFETIME register provides the last known Energy Source lifetime percentage from 0 to 100. 100 represents a healthy state. If no health check has been done yet, this register shall return the value 0xFF. The ES_LIFETIME register is not impacted by the Factory Default operation.

The ES lifetime percentage shall decrease with use.

8.2.6.2 ES_TEMP0 – Offset 0x71

Definition	Access	Mand	Persist	Default
[7:0] : Energy source temperature bits 7:0	RO	N	N	Environment -specific

8.2.6.2 ES_TEMP0 – Offset 0x71 (cont'd)

The ES_TEMP0 register provides the least significant byte of the last known Energy Source temperature in the format defined in 7.10. The minimum value is 0. The temperature measurement frequency is vendor specific. If no Energy Source temperature measurement has been done yet or the module does not support measurement of Energy Source temperature, this register shall return the value 0xFF. The ES_TEMP0 register is not impacted by the Factory Default operation.

NOTE: Modules compliant with JESD245 or JESD245A did not support bit 1 or bit 0 in this register.

8.2.6.3 ES_TEMP1 – Offset 0x72

Definition	Access	Mand	Persist	Default
[7:0] : Energy source temperature bits 15:8	RO	N	N	Environment -specific

The ES_TEMP1 register provides the most significant byte of the last known Energy Source temperature in the format defined in 7.10. The temperature measurement frequency is vendor specific. If no Energy Source temperature measurement has been done yet or the module does not support measurement of Energy Source temperature, this register shall return the value 0xFF. The ES_TEMP1 register is not impacted by the Factory Default operation.

NOTE: Modules compliant with JESD245 or JESD245A did not support the Sign bit in this register.

8.2.6.4 ES_RUNTIME0 – Offset 0x73

Definition	Access	Mand	Persist	Default
[7:0] : Energy Source Runtime Hours counter bits 7:0	RO	N	Y	0

The ES_RUNTIME0 register returns the least significant byte of Energy Source Runtime Hours counter (see 7.2.13). The ES_RUNTIME0 register is not impacted by the Factory Default operation.

8.2.6.5 ES_RUNTIME1 – Offset 0x74

Definition	Access	Mand	Persist	Default
[7:0] : Energy Source Runtime Hours counter bits 15:8	RO	N	Y	0

The ES_RUNTIME1 register returns the most significant byte of the Energy Source Runtime Hours counter. The ES_RUNTIME1 register is not impacted by the Factory Default operation.

8.3 Page 2 Register Map

The registers in page 2 are related to device statistics, error injection or host area and organized based on categories. Table 16 shows the layout of the categories in page 2.

Table 16 — Page 2 Register Map Categories

Offset	Category	Description
0x00 – 0x03	Paging Mechanism	Registers related to I^2C page support
0x04 – 0x5F	Device Statistics	Registers providing device statistics information
0x60 – 0x7F	Error Injection	Registers related to error injection support
0x80 – 0xFF	Host Area	Registers for host generated data

8.3 Page 2 Register Map (cont'd)

Table 17 lists the registers that are in page 2.

Table 17 — Page 2 Register Map

Offset	Register Name	Clause
Paging Mechanism Registers		
0x00	OPEN_PAGE	8.3.1.1
0x01-0x03	Reserved	
Device Statistics Registers		
0x04	LAST_CSAVE_DURATION0	8.3.2.1
0x05	LAST_CSAVE_DURATION1	8.3.2.2
0x06	LAST_RESTORE_DURATION0	8.3.2.3
0x07	LAST_RESTORE_DURATION1	8.3.2.4
0x08	LAST_ERASE_DURATION0	8.3.2.5
0x09	LAST_ERASE_DURATION1	8.3.2.6
0x0A	CSAVE_SUCCESS_COUNT0	8.3.2.7
0x0B	CSAVE_SUCCESS_COUNT1	8.3.2.8
0x0C	RESTORE_SUCCESS_COUNT0	8.3.2.9
0x0D	RESTORE_SUCCESS_COUNT1	8.3.2.10
0x0E	ERASE_SUCCESS_COUNT0	8.3.2.11
0x0F	ERASE_SUCCESS_COUNT1	8.3.2.12
0x10	POWER_CYCLE_COUNT0	8.3.2.13
0x11	POWER_CYCLE_COUNT1	8.3.2.14
0x12	CSAVE_FAILURE_COUNT0	8.3.2.15
0x13	CSAVE_FAILURE_COUNT1	8.3.2.16
0x14	RESTORE_FAILURE_COUNT0	8.3.2.17
0x15	RESTORE_FAILURE_COUNT1	8.3.2.18
0x16	ERASE_FAILURE_COUNT0	8.3.2.19
0x17	ERASE_FAILURE_COUNT1	8.3.2.20
0x18	LAST_ARM_DURATION0	8.3.2.21
0x19	LAST_ARM_DURATION1	8.3.2.22
0x1A	LAST_FACTORY_DEFAULT_DURATION0	8.3.2.23
0x1B	LAST_FACTORY_DEFAULT_DURATION1	8.3.2.24
0x1C	LAST_FIRMWARE_OPS_DURATION0	8.3.2.25
0x1D	LAST_FIRMWARE_OPS_DURATION1	8.3.2.26
0x1E	LAST_OPERATIONAL_UNIT_OPS_DURATION0	8.3.2.27
0x1F	LAST_OPERATIONAL_UNIT_OPS_DURATION1	8.3.2.28

Table 17 — Page 2 Register Map (cont'd)

Offset	Register Name	Clause
0x20	ARM_SUCCESS_COUNT0	8.3.2.29
0x21	ARM_SUCCESS_COUNT1	8.3.2.30
0x22	FACTORY_DEFAULT_SUCCESS_COUNT0	8.3.2.31
0x23	FACTORY_DEFAULT_SUCCESS_COUNT1	8.3.2.32
0x24	FIRMWARE_SUCCESS_COUNT0	8.3.2.33
0x25	FIRMWARE_SUCCESS_COUNT1	8.3.2.34
0x26	OPERATIONAL_UNIT_SUCCESS_COUNT0	8.3.2.35
0x27	OPERATIONAL_UNIT_SUCCESS_COUNT1	8.3.2.36
0x28	ARM_FAILURE_COUNT0	8.3.2.37
0x29	ARM_FAILURE_COUNT1	8.3.2.38
0x2A	FACTORY_DEFAULT_FAILURE_COUNT0	8.3.2.39
0x2B	FACTORY_DEFAULT_FAILURE_COUNT1	8.3.2.40
0x2C	FIRMWARE_FAILURE_COUNT0	8.3.2.41
0x2D	FIRMWARE_FAILURE_COUNT1	8.3.2.42
0x2E	OPERATIONAL_UNIT_FAILURE_COUNT0	8.3.2.43
0x2F	OPERATIONAL_UNIT_FAILURE_COUNT1	8.3.2.44
0x30-0x5F	Reserved	
Error Injection Registers		
0x60	INJECT_OPS_FAILURES0	8.3.3.1
0x61	INJECT_OPS_FAILURES1	8.3.3.2
0x62-0x63	Reserved	
0x64	INJECT_ES_FAILURES	8.3.3.3
0x65	INJECT_FW_FAILURES	8.3.3.4
0x66	Reserved	
0x67	INJECT_BAD_BLOCK_CAP	8.3.3.5
0x68	INJECT_ERROR_TYPE	8.3.3.6
0x69-0x7F	Reserved	
Host Area Registers		
0x80	DRAM_ECC_ERROR_COUNT	8.3.4.1 and 8.3.4.2
0x81	DRAM_THRESHOLD_ECC_COUNT	8.3.4.2
0x82	HOST_MANAGED_ES_ATTRIBUTES	8.3.4.3
0x83	HOST_CSAVE_FAIL	8.3.4.4
0x84	HOST_CSAVE_WORKFLOW_FAILURE_COUNT0	8.3.4.5
0x85	HOST_CSAVE_WORKFLOW_FAILURE_COUNT1	8.3.4.6
0x86-0xFF	Reserved	

8.3.1 Paging Mechanism Registers

The registers in this category are related to the I^2C paging mechanism (see 5.3).

8.3.1.1 OPEN_PAGE – Offset 0x00

Definition	Access	Mand	Persist	Default
[7:0] Open page number	RW	Y	N	0

This register is the same as the OPEN_PAGE register in page 0 (see 8.1.1.1).

8.3.2 Device Statistics Registers

The registers in this category provide device statistics information. If the module supports the register defined in this clause, it shall set Bit 3 (Device Statistics Supported) in the CAPABILITIES0 register (see 8.1.3.1). If the module does not support a device statistics register, the module shall return the value 0.

8.3.2.1 LAST_CSAVE_DURATION0 – Offset 0x04

Definition	Access	Mand	Persist	Default
[7:0] Last Catastrophic Save duration bits 7:0	RO	N	Y	0

The LAST_CSAVE_DURATION0 register provides the least significant byte of the last Catastrophic Save operation duration.

8.3.2.2 LAST_CSAVE_DURATION1 – Offset 0x05

Definition	Access	Mand	Persist	Default
[6:0] : Last Catastrophic Save duration bits 14:8 [7] : 0 – Duration in milliseconds 1 – Duration in seconds	RO	N	Y	0

The LAST_CSAVE_DURATION1 register provides the upper bits of and time units of the last Catastrophic Save operation duration.

If Bit 7 is 0, the duration value is in units of milliseconds. If Bit 7 is 1, the duration value is in units of seconds.

8.3.2.3 LAST_RESTORE_DURATION0 – Offset 0x06

Definition	Access	Mand	Persist	Default
[7:0] Last Restore duration bits 7:0	RO	N	Y	0

The LAST_RESTORE_DURATION0 register provides the least significant byte of the last Restore duration.

8.3.2.4 LAST_RESTORE_DURATION1 – Offset 0x07

Definition	Access	Mand	Persist	Default
[6:0] : Last Restore duration bits 14:8 [7] : 0 – Duration in milliseconds 1 – Duration in seconds	RO	N	Y	0

The LAST_RESTORE_DURATION1 register provides the upper bits of and time units of the last Restore operation duration.

If Bit 7 is 0, the duration value is in units of milliseconds. If Bit 7 is 1, the duration value is in units of seconds.

8.3.2.5 LAST_ERASE_DURATION0 – Offset 0x08

Definition	Access	Mand	Persist	Default
[7:0] Last Erase duration bits 7:0	RO	N	Y	0

The LAST_ERASE_DURATION0 register provides the least significant byte of the last Erase operation duration.

8.3.2.5 LAST_ERASE_DURATION0 – Offset 0x08 (cont'd)

If the Atomic Save and Erase capability is used, this value indicates the Erase duration of the Erase part of the Atomic Save and Erase operation.

8.3.2.6 LAST_ERASE_DURATION1 – Offset 0x09

Definition	Access	Mand	Persist	Default
[6:0] : Last Erase duration bits 14:8 [7] : 0 – Duration in milliseconds 1 – Duration in seconds	RO	N	Y	0

The LAST_ERASE_DURATION1 register provides the upper bits of and time units of the last Erase operation duration.

If the Atomic Save and Erase capability is used, this value indicates the Erase duration of the Erase part of the Atomic Save and Erase operation.

If Bit 7 is 0, the duration value is in units of milliseconds. If Bit 7 is 1, the duration value is in units of seconds.

8.3.2.7 CSAVE_SUCCESS_COUNT0 – Offset 0x0A

Definition	Access	Mand	Persist	Default
[7:0] Catastrophic Save Success counter bits 7:0	RO	N	Y	0

The CSAVE_SUCCESS_COUNT0 register provides the least significant byte of the Catastrophic Save Success counter.

The CSAVE_SUCCESS_COUNT0 register is not impacted by the Factory Default operation.

8.3.2.8 CSAVE_SUCCESS_COUNT1 – Offset 0x0B

Definition	Access	Mand	Persist	Default
[7:0] Catastrophic Save Success counter bits 15:8	RO	N	Y	0

The CSAVE_SUCCESS_COUNT1 register provides the most significant byte of the Catastrophic Save Success counter.

The CSAVE_SUCCESS_COUNT1 register is not impacted by the Factory Default operation.

8.3.2.9 RESTORE_SUCCESS_COUNT0 – Offset 0x0C

Definition	Access	Mand	Persist	Default
[7:0] Restore Success counter bits 7:0	RO	N	Y	0

The RESTORE_SUCCESS_COUNT0 register provides the least significant byte of the Restore Success counter.

8.3.2.10 RESTORE_SUCCESS_COUNT1 – Offset 0x0D

Definition	Access	Mand	Persist	Default
[7:0] Restore Success counter bits 15:8	RO	N	Y	0

The RESTORE_SUCCESS_COUNT1 register provides the most significant byte of the Restore Success counter.

8.3.2.11 ERASE_SUCCESS_COUNT0 – Offset 0x0E

Definition	Access	Mand	Persist	Default
[7:0] Erase Success counter bits 7:0	RO	N	Y	0

The ERASE_SUCCESS_COUNT0 register provides the least significant byte of the Erase Success counter.

If the Atomic Save and Erase capability is used, this value will increment by the Erase part of the Atomic Arm and Erase operation.

8.3.2.12 ERASE_SUCCESS_COUNT1 – Offset 0x0F

Definition	Access	Mand	Persist	Default
[7:0] Erase Success counter bits 15:8	RO	N	Y	0

The ERASE_SUCCESS_COUNT1 register provides the most significant byte of the Erase Success counter.

If the Atomic Save and Erase capability is used, this value will increment by the Erase part of the Atomic Arm and Erase operation.

8.3.2.13 POWER_CYCLE_COUNT0 – Offset 0x10

Definition	Access	Mand	Persist	Default
[7:0] Power cycles counter bits 7:0	RO	N	Y	0

The POWER_CYCLE_COUNT0 register provides the least significant byte of the Power Cycles counter (see 7.9). The POWER_CYCLE_COUNT0 register is not impacted by the Factory Default operation.

8.3.2.14 POWER_CYCLE_COUNT1 – Offset 0x11

Definition	Access	Mand	Persist	Default
[7:0] Power cycles counter bits 15:8	RO	N	Y	0

The POWER_CYCLE_COUNT1 register provides the most significant byte of the Power Cycles counter. The POWER_CYCLE_COUNT1 register is not impacted by the Factory Default operation.

8.3.2.15 CSAVE_FAILURE_COUNT0 – Offset 0x12

Definition	Access	Mand	Persist	Default
[7:0] Catastrophic Save Failure counter bits 7:0	RO	N	Y	0

The CSAVE_FAILURE_COUNT0 provides the least significant byte of the Catastrophic Save Failure counter.

NOTE: Modules compliant with JESD245 and JESD245A do not support the CSAVE_FAILURE_COUNT0 and CSAVE_FAILURE_COUNT1 registers.

The CSAVE_FAILURE_COUNT0 register is not impacted by the Factory Default operation.

8.3.2.16 CSAVE_FAILURE_COUNT1 – Offset 0x13

Definition	Access	Mand	Persist	Default
[7:0] Catastrophic Save Failure counter bits 15:8	RO	N	Y	0

The CSAVE_FAILURE_COUNT1 register provides the most significant byte of the Catastrophic Save Failure counter.

The CSAVE_FAILURE_COUNT1 register is not impacted by the Factory Default operation.

8.3.2.17 RESTORE_FAILURE_COUNT0 – Offset 0x14

Definition	Access	Mand	Persist	Default
[7:0] Restore Failure counter bits 7:0	RO	N	Y	0

The RESTORE_FAILURE_COUNT0 provides the least significant byte of the Restore Failure counter.

NOTE: Modules compliant with JESD245 and JESD245A do not support the RESTORE_FAILURE_COUNT0 and RESTORE_FAILURE_COUNT1 registers.

The RESTORE_FAILURE_COUNT0 register is not impacted by the Factory Default operation.

8.3.2.18 RESTORE_FAILURE_COUNT1 – Offset 0x15

Definition	Access	Mand	Persist	Default
[7:0] Restore Failure counter bits 15:8	RO	N	Y	0

The RESTORE_FAILURE_COUNT1 register provides the most significant byte of the Restore Failure counter.

The RESTORE_FAILURE_COUNT1 register is not impacted by the Factory Default operation.

8.3.2.19 ERASE_FAILURE_COUNT0 – Offset 0x16

Definition	Access	Mand	Persist	Default
[7:0] Erase Failure counter bits 7:0	RO	N	Y	0

The ERASE_FAILURE_COUNT0 provides the least significant byte of the Erase Failure counter.

NOTE: Modules compliant with JESD245 and JESD245A do not support the ERASE_FAILURE_COUNT0 and ERASE_FAILURE_COUNT1 registers.

The ERASE_FAILURE_COUNT0 register is not impacted by the Factory Default operation.

8.3.2.20 ERASE_FAILURE_COUNT1 – Offset 0x17

Definition	Access	Mand	Persist	Default
[7:0] Erase Failure counter bits 15:8	RO	N	Y	0

The ERASE_FAILURE_COUNT1 register provides the most significant byte of the Erase Failure counter.

The ERASE_FAILURE_COUNT1 register is not impacted by the Factory Default operation.

8.3.2.21 LAST_ARM_DURATION0 – Offset 0x18

Definition	Access	Mand	Persist	Default
[7:0] Last Arm duration bits 7:0	RO	N	Y	0

The LAST_ARM_DURATION0 register provides the least significant byte of the last Arm duration.

8.3.2.22 LAST_ARM_DURATION1 – Offset 0x19

Definition	Access	Mand	Persist	Default
[6:0] : Last Arm duration bits 14:8 [7] : 0 – Duration in milliseconds 1 – Duration in seconds	RO	N	Y	0

The LAST_ARM_DURATION1 register provides the upper bits of and time units of the last Arm operation duration.

If Bit 7 is 0, the duration value is in units of milliseconds. If Bit 7 is 1, the duration value is in units of seconds.

8.3.2.23 LAST_FACTORY_DEFAULT_DURATION0 – Offset 0x1A

Definition	Access	Mand	Persist	Default
[7:0] Last Factory Default duration bits 7:0	RO	N	Y	0

The LAST_FACTORY_DEFAULT_DURATION0 register provides the least significant byte of the last Factory Default operation duration.

8.3.2.24 LAST_FACTORY_DEFAULT_DURATION1 – Offset 0x1B

Definition	Access	Mand	Persist	Default
[6:0] : Last Factory Default duration bits 14:8 [7] : 0 – Duration in milliseconds 1 – Duration in seconds	RO	N	Y	0

The LAST_FACTORY_DEFAULT_DURATION1 register provides the upper bits of and time units of the last Factory Default operation duration.

If Bit 7 is 0, the duration value is in units of milliseconds. If Bit 7 is 1, the duration value is in units of seconds.

8.3.2.25 LAST_FIRMWARE_OPS_DURATION0 – Offset 0x1C

Definition	Access	Mand	Persist	Default
[7:0] Last Firmware Ops duration bits 7:0	RO	N	Y	0

The LAST_FIRMWARE_OPS_DURATION0 register provides the least significant byte of the last Firmware operation duration.

8.3.2.26 LAST_FIRMWARE_OPS_DURATION1 – Offset 0x1D

Definition	Access	Mand	Persist	Default
[6:0] : Last Firmware Ops duration bits 14:8 [7] : 0 – Duration in milliseconds 1 – Duration in seconds	RO	N	Y	0

The LAST_FIRMWARE_OPS_DURATION1 register provides the upper bits of and time units of the last Firmware operation duration.

If Bit 7 is 0, the duration value is in units of milliseconds. If Bit 7 is 1, the duration value is in units of seconds.

8.3.2.27 LAST_OPERATIONAL_UNIT_OPS_DURATION0 – Offset 0x1E

Definition	Access	Mand	Persist	Default
[7:0] Last Operational Unit Ops duration bits 7:0	RO	N	Y	0

The LAST_OPERATIONAL_UNIT_OPS_DURATION0 register provides the least significant byte of the last Operational Unit Ops duration.

8.3.2.28 LAST_OPERATIONAL_UNIT_OPS_DURATION1 – Offset 0x1F

Definition	Access	Mand	Persist	Default
[6:0] : Last Operational Unit Ops duration bits 14:8 [7] : 0 – Duration in milliseconds 1 – Duration in seconds	RO	N	Y	0

The LAST_OPERATIONAL_UNIT_OPS_DURATION1 register provides the upper bits of and time units of the last Operational Unit Ops duration.

If Bit 7 is 0, the duration value is in units of milliseconds. If Bit 7 is 1, the duration value is in units of seconds.

8.3.2.29 ARM_SUCCESS_COUNT0 – Offset 0x20

Definition	Access	Mand	Persist	Default
[7:0] Arm Success counter bits 7:0	RO	N	Y	0

The ARM_SUCCESS_COUNT0 register provides the least significant byte of the Arm Success counter.

8.3.2.30 ARM_SUCCESS_COUNT1 – Offset 0x21

Definition	Access	Mand	Persist	Default
[7:0] Arm Success counter bits 15:8	RO	N	Y	0

The ARM_SUCCESS_COUNT1 register provides the most significant byte of the Arm Success counter.

8.3.2.31 FACTORY_DEFAULT_SUCCESS_COUNT0 – Offset 0x22

Definition	Access	Mand	Persist	Default
[7:0] Factory Default Success counter bits 7:0	RO	N	Y	0

The FACTORY_DEFAULT_SUCCESS_COUNT0 register provides the least significant byte of the Factory Default Success counter.

8.3.2.32 FACTORY_DEFAULT_SUCCESS_COUNT1 – Offset 0x23

Definition	Access	Mand	Persist	Default
[7:0] Factory Default Success counter bits 15:8	RO	N	Y	0

The FACTORY_DEFAULT_SUCCESS_COUNT1 register provides the most significant byte of the Factory Default Success counter.

8.3.2.33 FIRMWARE_SUCCESS_COUNT0 – Offset 0x24

Definition	Access	Mand	Persist	Default
[7:0] Firmware Ops Success counter bits 7:0	RO	N	Y	0

The FIRMWARE_SUCCESS_COUNT0 register provides the least significant byte of the Firmware Ops Success counter.

8.3.2.34 FIRMWARE_SUCCESS_COUNT1 – Offset 0x25

Definition	Access	Mand	Persist	Default
[7:0] Firmware Ops Success counter bits 15:8	RO	N	Y	0

The FIRMWARE_SUCCESS_COUNT1 register provides the most significant byte of the Firmware Ops Success counter.

8.3.2.35 OPERATIONAL_UNIT_SUCCESS_COUNT0 – Offset 0x26

Definition	Access	Mand	Persist	Default
[7:0] Operational Unit Ops Success counter bits 7:0	RO	N	Y	0

The OPERATIONAL_UNIT_SUCCESS_COUNT0 register provides the least significant byte of the Operational Unit Ops Success counter.

8.3.2.36 OPERATIONAL_UNIT_SUCCESS_COUNT1 – Offset 0x27

Definition	Access	Mand	Persist	Default
[7:0] Operational Unit Ops Success counter bits 15:8	RO	N	Y	0

The OPERATIONAL_UNIT_SUCCESS_COUNT1 register provides the most significant byte of the Operational Unit Ops Success counter.

8.3.2.37 ARM_FAILURE_COUNT0 – Offset 0x28

Definition	Access	Mand	Persist	Default
[7:0] Arm Failure counter bits 7:0	RO	N	Y	0

The ARM_FAILURE_COUNT0 register provides the least significant byte of the Arm Failure counter.

NOTE: Modules compliant with JESD245 and JESD245A do not support the ARM_FAILURE_COUNT0 and ARM_FAILURE_COUNT1 registers.

8.3.2.38 ARM_FAILURE_COUNT1 – Offset 0x29

Definition	Access	Mand	Persist	Default
[7:0] Arm Failure counter bits 15:8	RO	N	Y	0

The ARM_FAILURE_COUNT1 register provides the most significant byte of the Arm Failure counter.

8.3.2.39 FACTORY_DEFAULT_FAILURE_COUNT0 – Offset 0x2A

Definition	Access	Mand	Persist	Default
[7:0] Factory Default Failure counter bits 7:0	RO	N	Y	0

The FACTORY_DEFAULT_FAILURE_COUNT0 register provides the least significant byte of the Factory Default Failure counter.

NOTE: Modules compliant with JESD245 and JESD245A do not support the FACTORY_DEFAULT_FAILURE_COUNT0 and FACTORY_DEFAULT_FAILURE_COUNT1 registers.

8.3.2.40 FACTORY_DEFAULT_FAILURE_COUNT1 – Offset 0x2B

Definition	Access	Mand	Persist	Default
[7:0] Factory Default Failure counter bits 15:8	RO	N	Y	0

The FACTORY_DEFAULT_FAILURE_COUNT1 register provides the most significant byte of the Factory Default Failure counter.

8.3.2.41 FIRMWARE_FAILURE_COUNT0 – Offset 0x2C

Definition	Access	Mand	Persist	Default
[7:0] Firmware Ops Failure counter bits 7:0	RO	N	Y	0

The FIRMWARE_FAILURE_COUNT0 register provides the least significant byte of the Firmware Ops Failure counter.

NOTE: Modules compliant with JESD245 and JESD245A do not support the FIRMWARE_FAILURE_COUNT0 and FIRMWARE_FAILURE_COUNT1 registers.

8.3.2.42 FIRMWARE_FAILURE_COUNT1 – Offset 0x2D

Definition	Access	Mand	Persist	Default
[7:0] Firmware Ops Failure counter bits 15:8	RO	N	Y	0

The FIRMWARE_FAILURE_COUNT1 register provides the most significant byte of the Firmware Ops Failure counter.

8.3.2.43 OPERATIONAL_UNIT_FAILURE_COUNT0 – Offset 0x2E

Definition	Access	Mand	Persist	Default
[7:0] Operational Unit Ops Failure counter bits 7:0	RO	N	Y	0

The OPERATIONAL_UNIT_FAILURE_COUNT0 register provides the least significant byte of the Operational Unit Ops Failure counter.

NOTE: Modules compliant with JESD245 and JESD245A do not support the OPERATIONAL_UNIT_FAILURE_COUNT0 and OPERATIONAL_UNIT_FAILURE_COUNT1 registers.

8.3.2.44 OPERATIONAL_UNIT_FAILURE_COUNT1 – Offset 0x2F

Definition	Access	Mand	Persist	Default
[7:0] Operational Unit Ops Failure counter bits 15:8	RO	N	Y	0

The OPERATIONAL_UNIT_FAILURE_COUNT1 register provides the most significant byte of the Operational Unit Ops Failure counter.

8.3.3 Error Injection Registers

The registers in this category are related to error injection support.

8.3.3.1 INJECT_OPS_FAILURES0 – Offset 0x60

Definition	Access	Mand	Persist	Default
[0] : INJECT_CSAVE_FAILURE [1] : INJECT_RESTORE_FAILURE [2] : INJECT_ERASE_FAILURE [3] : INJECT_ARM_FAILURE [4] : INJECT_INTERNAL_CONTROLLER_FAILURE [5] : INJECT_NVM_LIFETIME_WARNING [6] : INJECT_NVM_LIFETIME_ERROR [7] : INJECT_BAD_BLOCKS	RW	Y	Y	0

The INJECT_OPS_FAILURES0 register allows the host to inject operation or NVM related failures. A set bit indicates the corresponding error injection is enabled. A clear bit indicates the corresponding error injection is disabled. Error injection shall take place on the next applicable operation. If Bit 0 of the INJECT_ERROR_TYPE (see 8.3.3.6) is clear, then error injection is enabled until explicitly disabled. One or more bits can be set in this register.

If Bit 0, INJECT_CSAVE_FAILURE, is set, the module shall cause the next Catastrophic Save operation to fail with the SAVE_ERROR bit set in the SAVE_STATUS0 register. The module should not report any reasons in the CSAVE_FAIL_INFO0 and CSAVE_FAIL_INFO1 registers.

If Bit 1, INJECT_RESTORE_FAILURE, is set, the module shall cause the next Restore operation to fail with the RESTORE_ERROR bit set in the RESTORE_STATUS register. The module should not report any reasons in the RESTORE_FAIL_INFO register.

If Bit 2, INJECT_ERASE_FAILURE, is set, the module shall cause the next Erase operation to fail with the ERASE_ERROR bit set in the ERASE_STATUS register.

If Bit 3, INJECT_ARM_FAILURE, is set, the module shall cause the next Arm operation to fail with the ARM_ERROR bit set in the ARM_STATUS register.

If Bit 4, INJECT_INTERNAL_CONTROLLER_FAILURE, is set, the module shall inject an internal controller failure at the most appropriate time.

If Bit 5, INJECT_NVM_LIFETIME_WARNING, is set, the module shall simulate a NVM lifetime warning threshold exceeded event.

If Bit 6, INJECT_NVM_LIFETIME_ERROR, is set, the module shall simulate a NVM lifetime error threshold exceeded event.

If Bit 7, INJECT_BAD_BLOCKS, is set, the module shall disable bad block correction until the percentage of NVM that contains a bad block matches INJECT_BAD_BLOCK_CAP register value. This support is optional.

8.3.3.2 INJECT_OPS_FAILURES1 – Offset 0x61

Definition	Access	Mand	Persist	Default
[0] : INJECT_PERMANENT_HARDWARE_FAILURE [1] : INJECT_FACTORY_DEFAULT_FAILURE [2] : INJECT_SET_EVENT_NOTIFICATION_FAILURE [3] : INJECT_SET_ES_POLICY_FAILURE [4] : INJECT_FIRMWARE_OPS_FAILURE [5] : INJECT_OPERATIONAL_UNIT_OPS_FAILURE [7:6] : Reserved	RW	Y	Y	0

The INJECT_OPS_FAILURES1 register allows the host to inject failures. A set bit indicates the corresponding error injection is enabled. A clear bit indicates the corresponding error injection is disabled. Error injection shall take place on the next applicable operation. If Bit 0 of the INJECT_ERROR_TYPE register (see 8.3.3.6) is clear, then error injection is enabled until explicitly disabled. One or more bits can be set in this register.

NOTE: Modules compliant with JESD245 and JESD245A do not implement the INJECT_OPS_FAILURES1 register.

If Bit 0, INJECT_PERMANENT_HARDWARE_FAILURE, is set, the module shall report that it has a permanent hardware failure.

If Bit 1, INJECT_FACTORY_DEFAULT_FAILURE, is set, the module shall cause the next Factory Default operation to fail with the FACTORY_DEFAULT_ERROR bit set in the FACTORY_DEFAULT_STATUS register.

If Bit 2, INJECT_SET_EVENT_NOTIFICATION_FAILURE, is set, the module shall cause the next Set Event Notification operation to fail with the SET_EVENT_NOTIFICATION_ERROR bit set in the SET_EVENT_NOTIFICATION_STATUS register.

If Bit 3, INJECT_SET_ES_POLICY_FAILURE, is set, the module shall cause the next Set Energy Source Policy operation to fail with the SET_ES_POLICY_ERROR bit set in the SET_ES_POLICY_STATUS register.

If Bit 4, INJECT_FIRMWARE_OPS_FAILURE, is set, the module shall cause the next Firmware operation to fail with the FIRMWARE_OPS_ERROR bit set in the FIRMWARE_OPS_STATUS register.

If Bit 5, INJECT_OPERATIONAL_UNIT_OPS_FAILURE, is set, the module shall cause the next Operational Unit operation to fail with the OPERATIONAL_UNIT_OPS_ERROR bit set in the OPERATIONAL_UNIT_OPS_STATUS register.

8.3.3.3 INJECT_ES_FAILURES – Offset 0x64

Definition	Access	Mand	Persist	Default
[0] : INJECT_ES_FAILURE [1] : INJECT_ES_ASSESSMENT_FAILURE [2] : INJECT_ES_LIFETIME_WARNING [3] : INJECT_ES_LIFETIME_ERROR [4] : INJECT_ES_TEMP_WARNING [5] : INJECT_ES_TEMP_ERROR [7:6] : Reserved	RW	N	Y	0

8.3.3.3 INJECT_ES_FAILURES – Offset 0x64 (cont'd)

The INJECT_ES_FAILURES register allows the host to inject errors into a Device Managed Energy Source. A set bit indicates the corresponding error injection is enabled. A clear bit indicates the corresponding error injection is disabled. Error injection shall take place on the next applicable operation. If Bit 0 of the INJECT_ERROR_TYPE register (see 8.3.3.6) is not set, the error injection is enabled until explicitly disabled. One or more bits can be set in this register.

This register shall be supported when module is in Device Managed Policy mode. When the module is in Host Managed Policy mode, the module shall return the value 0 if this register is read and ignore writes to this register.

If Bit 0, INJECT_ES_FAILURE, is set, the module shall simulate an Energy Source hardware failure.

If Bit 1, INJECT_ES_ASSESSMENT_FAILURE, is set, the module shall fail the next Energy Source health check.

If Bit 2, INJECT_ES_LIFETIME_WARNING, is set, the module shall simulate an Energy Source lifetime warning threshold exceeded event.

If Bit 3, INJECT_ES_LIFETIME_ERROR, is set, the module shall simulate an Energy Source lifetime error threshold exceeded event.

If Bit 4, INJECT_ES_TEMP_WARNING, is set, the module shall simulate an Energy Source temperature warning threshold exceeded event.

If Bit 5, INJECT_ES_TEMP_ERROR, is set, the module shall simulate an Energy Source temperature error threshold exceeded event.

8.3.3.4 INJECT_FW_FAILURES – Offset 0x65

Definition	Access	Mand	Persist	Default
[0] : INJECT_FW_COMMIT_FAILURE [1] : INJECT_FW_CHECKSUM_FAILURE [2] : INJECT_INVALID_FW [7:3] : Reserved	RW	Y	Y	0

The INJECT_FW_FAILURES register allows the host to inject errors for Firmware operations. A set bit indicates the corresponding error injection is enabled. A clear bit indicates the corresponding error injection is disabled. Error injection shall take place on the next applicable operation. If Bit 0 of the INJECT_ERROR_TYPE register (see 8.3.3.6) is not set, the error injection is enabled until explicitly disabled. One or more bits can be set in this register.

If Bit 0, INJECT_FW_COMMIT_FAILURE, is set, the module shall cause the next Commit Firmware operation to fail with the FIRMWARE_OPS_ERROR bit set in the FIRMWARE_OPS_STATUS register.

If Bit 1, INJECT_FW_CHECKSUM_FAILURE, is set, the module shall cause the next Generate Firmware Checksum operation to inject a bad checksum value to FW_REGION_CRC0 and FW_REGION_CRC1 registers.

If Bit 2, INJECT_INVALID_FW, is set, the module shall cause:

- the next Validate Firmware Image operation to fail with the FIRMWARE_OPS_ERROR bit set in the FIRMWARE_OPS_STATUS register (see 8.1.5.11).

8.3.3.4 INJECT_FW_FAILURES – Offset 0x65 (cont'd)

- the next power on or Reset Controller operation to treat the firmware in slot 1 as having an invalid firmware image (i.e., failover to slot 0 and return zeros in the SLOT1_FWREV and SLOT1_FWREV1 registers (see 8.1.2.5 and 8.1.2.6)).

The INJECT_INVALID_FW bit shall not cause the module to set the INVALID_FIRMWARE_ERROR bit in the MODULE_HEALTH_STATUS1 register (see 8.1.8.3).

8.3.3.5 INJECT_BAD_BLOCK_CAP – Offset 0x67

Definition	Access	Mand	Persist	Default
[7:0] Maximum percentage of NVM containing bad block	RW	N	Y	0

The INJECT_BAD_BLOCK_CAP register allows the host to set the maximum percentage of NVM that contains a bad block when INJECT_BAD_BLOCKS error injection is enabled. The host should set this register to a non-zero value prior to enabling INJECT_BAD_BLOCKS.

8.3.3.6 INJECT_ERROR_TYPE – Offset 0x68

Definition	Access	Mand	Persist	Default
[0] : ONE_TIME_USE [7:1] Reserved	RW	N	Y	0

The INJECT_ERROR_TYPE register allows the host to set whether the error injection is one-time only or persistent until disabled.

If Bit 0, ONE_TIME_USE, is 0, the error injection is persistent until disabled. If Bit 0 is 1, the error injection is one-time only.

8.3.4 Host Area Registers

The registers in this category are related to host-generated data. Modules shall only provide storage for this data and shall not act upon the data.

8.3.4.1 DRAM_ECC_ERROR_COUNT – Offset 0x80

Definition	Access	Mand	Persist	Default
[7:0] Uncorrectable Errors count	RW	Y	Y	0

The DRAM_ECC_ERROR_COUNT register provides storage for the host's Uncorrectable Errors counter value (see 7.11).

8.3.4.2 DRAM_THRESHOLD_ECC_COUNT – Offset 0x81

Definition	Access	Mand	Persist	Default
[7:0] Correctable Errors Threshold Exceeded count	RW	Y	Y	0

The DRAM_THRESHOLD_ECC_COUNT register provides storage for the host's Correctable Errors Threshold Exceeded counter value (see 7.11).

8.3.4.3 HOST_MANAGED_ES_ATTRIBUTES – Offset 0x82

Definition	Access	Mand	Persist	Default
[0] : Shared - 0: Dedicated Energy Source - 1: Shared Energy Source [7:1] Reserved	RW	N	Y	0

The HOST_MANAGED_ES_ATTRIBUTES register allows the host to store attributes for the Host Managed Energy Source that moves with the module. A module in Host Managed Policy mode shall support this register. A module in Device Managed Policy mode shall return the value 0 if this is register is read and ignore any writes to this register.

If Bit 0 is clear, the Host Managed Energy Source is used only by this module. If Bit 0 is set, the Host Managed Energy Source is shared by two or more modules.

8.3.4.4 HOST_CSAVE_FAIL – Offset 0x83

Definition	Access	Mand	Persist	Default
[0] : HOST_ES_FAIL [1] : MC_WRITE_BUFFER_FLUSH_FAIL [2] : CPU_CACHE_FLUSH_FAIL [3] : IO_DMA_WRITE_BUFFER_FLUSH_FAIL [4] : HOST_CSAVE_WORKFLOW_FAIL [7:5] Reserved	RW	Y	Y	0

The HOST_CSAVE_FAIL register allows the host to optionally store information about issues on the host that caused a Catastrophic Save operation to fail or save inconsistent data. The host should clear this register before setting the bit to arm the NVDIMM-N.

NOTE: Modules compliant with JESD245 and JESD245A do not support the HOST_CSAVE_FAIL register.

If Bit 0, HOST_ES_FAIL is set, the Catastrophic Save operation failed due to a Host Energy Source issue.

If Bit 1, MC_WRITE_BUFFER_FLUSH_FAIL, is set, the host encountered a failure flushing the host memory controller's write buffer(s).

If Bit 2, CPU_CACHE_FLUSH_FAIL, is set, the host encountered a failure flushing the host processor cache(s).

If Bit 3, IO_DMA_WRITE_BUFFER_FLUSH_FAIL, is set, the host encountered a failure flushing the host memory controller's write buffer(s) used for IO DMA.

If Bit 4, HOST_CSAVE_WORKFLOW_FAIL, is set, the host encountered a host non-cache or write buffer flush failure in its Catastrophic Save workflow.

8.3.4.5 HOST_CSAVE_WORKFLOW_FAILURE_COUNT0 – Offset 0x84

Definition	Access	Mand	Persist	Default
[7:0] Catastrophic Save Workflow Failure count bits 7:0	RO	N	Y	0

The HOST_CSAVE_WORKFLOW_FAILURE_COUNT0 register provides storage for the least significant byte of the host's Catastrophic Save Workflow Failures counter value (see 7.11).

NOTE: Modules compliant with JESD245 and JESD245A do not support the HOST_CSAVE_WORKFLOW_FAILURE_COUNT0 and HOST_CSAVE_WORKFLOW_FAILURE_COUNT1 registers.

8.3.4.6 HOST_CSAVE_WORKFLOW_FAILURE_COUNT1 – Offset 0x85

Definition	Access	Mand	Persist	Default
[7:0] Catastrophic Save Workflow Failure count bits 15:8	RO	N	Y	0

The HOST_CSAVE_WORKFLOW_FAILURE_COUNT1 register provides storage for the most significant byte of the host's Catastrophic Save Workflow Failures counter value.

8.4 Page 3 Register Map

The registers in page 3 are related to Typed Block Data support or firmware update and organized based on categories. Table 18 shows the layout of the categories in page 3.

Table 18 — Page 3 Register Map Categories

Offset	Category	Description
0x00 – 0x03	Paging Mechanism	Registers related to I^2C page support
0x04 – 0x3F	Typed Block Data	Registers related to Typed Block Data support
0x40 – 0x6F	Firmware Update	Registers related to firmware update support
0x70 – 0x7F	Reserved	
0x80 – 0xBF	Typed Block Data Byte Data	Registers for Typed Block Data byte transfer
0xC0 – 0xDF	Reserved	
0xE0 – 0xFF	Typed Block Data Block Data	Registers for Typed Block Data block transfer

8.4 Page 3 Register Map (cont'd)

Table 19 lists the registers that are in page 3.

Table 19 — Page 3 Register Map

Offset	Register Name	Clause
Paging Mechanism Registers		
0x00	OPEN_PAGE	8.4.1.1
Clause0x01-0x03	Reserved	
Typed Block Data Registers		
0x04	TYPED_BLOCK_DATA	8.4.2.1
0x05	REGION_ID0	8.4.2.2
0x06	REGION_ID1	8.4.2.3
0x07	BLOCK_ID	8.4.2.4
0x08	TYPED_BLOCK_DATA_SIZE0	8.4.2.5
0x09	TYPED_BLOCK_DATA_SIZE1	8.4.2.6
0x0A	TYPED_BLOCK_DATA_SIZE2	8.4.2.7
0x0B	Reserved	
0x0C	OPERATIONAL_UNIT_ID0	8.4.2.8
0x0D	OPERATIONAL_UNIT_ID1	8.4.2.9
0x0E-0x0F	Reserved	
0x10	OPERATIONAL_UNIT_SIZE0	8.4.2.10
0x11	OPERATIONAL_UNIT_SIZE1	8.4.2.11
0x12	OPERATIONAL_UNIT_SIZE2	8.4.2.12
0x13	Reserved	
0x14	OPERATIONAL_UNIT_CRC0	8.4.2.13
0x15	OPERATIONAL_UNIT_CRC1	8.4.2.14
0x16-0x3F	Reserved	
Firmware Update Registers		
0x40	FW_REGION_CRC0	8.4.3.1
0x41	FW_REGION_CRC1	8.4.3.2
0x42	FW_SLOT_INFO	8.4.3.3
0x43-0x7F	Reserved	
Byte and Word Transaction Data Registers for Typed Block Data		
0x80	TYPED_BLOCK_DATA_BYTE0	8.4.4.1
0x81	TYPED_BLOCK_DATA_BYTE1	8.4.4.2
0x82	TYPED_BLOCK_DATA_BYTE2	8.4.4.3
0x83	TYPED_BLOCK_DATA_BYTE3	8.4.4.4
0x84	TYPED_BLOCK_DATA_BYTE4	8.4.4.5
0x85	TYPED_BLOCK_DATA_BYTE5	8.4.4.6
0x86	TYPED_BLOCK_DATA_BYTE6	8.4.4.7
0x87	TYPED_BLOCK_DATA_BYTE7	8.4.4.8

Table 19 — Page 3 Register Map (cont'd)

Offset	Register Name	Clause
0x88	TYPED_BLOCK_DATA_BYTE8	8.4.4.9
0x89	TYPED_BLOCK_DATA_BYTE9	8.4.4.10
0x8A	TYPED_BLOCK_DATA_BYTE10	8.4.4.11
0x8B	TYPED_BLOCK_DATA_BYTE11	8.4.4.12
0x8C	TYPED_BLOCK_DATA_BYTE12	8.4.4.13
0x8D	TYPED_BLOCK_DATA_BYTE13	8.4.4.14
0x8E	TYPED_BLOCK_DATA_BYTE14	8.4.4.15
0x8F	TYPED_BLOCK_DATA_BYTE15	8.4.4.16
0x90	TYPED_BLOCK_DATA_BYTE16	8.4.4.17
0x91	TYPED_BLOCK_DATA_BYTE17	8.4.4.18
0x92	TYPED_BLOCK_DATA_BYTE18	8.4.4.19
0x93	TYPED_BLOCK_DATA_BYTE19	8.4.4.20
0x94	TYPED_BLOCK_DATA_BYTE20	8.4.4.21
0x95	TYPED_BLOCK_DATA_BYTE21	8.4.4.22
0x96	TYPED_BLOCK_DATA_BYTE22	8.4.4.23
0x97	TYPED_BLOCK_DATA_BYTE23	8.4.4.24
0x98	TYPED_BLOCK_DATA_BYTE24	8.4.4.25
0x99	TYPED_BLOCK_DATA_BYTE25	8.4.4.26
0x9A	TYPED_BLOCK_DATA_BYTE26	8.4.4.27
0x9B	TYPED_BLOCK_DATA_BYTE27	8.4.4.28
0x9C	TYPED_BLOCK_DATA_BYTE28	8.4.4.29
0x9D	TYPED_BLOCK_DATA_BYTE29	8.4.4.30
0x9E	TYPED_BLOCK_DATA_BYTE30	8.4.4.31
0x9F	TYPED_BLOCK_DATA_BYTE31	8.4.4.32
0xA0-0xDF	Reserved	
Block Transaction Data Registers for Typed Block Data		
0xE0	TYPED_BLOCK_DATA_OFFSET	8.4.5.1
0xE1-0xFF	Reserved	

8.4.1 Paging Mechanism Registers

The registers in this category are related to the I^2C paging mechanism (see 5.3).

8.4.1.1 OPEN_PAGE – Offset 0x00

Definition	Access	Mand	Persist	Default
[7:0] Open page number	RW	Y	N	0

This register is the same as the OPEN_PAGE register in page 0 (see 8.1.1.1).

8.4.2 Typed Block Data Registers

The registers in this category are related to the Typed Block Data feature (see 5.4).

8.4.2.1 TYPED_BLOCK_DATA – Offset 0x04

Definition	Access	Mand	Persist	Default
[7:0] : TYPED_BLOCK_DATA - 0: Reserved - 1: Firmware Image Data - 2: Vendor Log Page - 3: Label Data - 4: Security Protocol Data - 5 to 200: Reserved - 201 to 255: Vendor Specific	RW	Y	N	0

When read from, the TYPED_BLOCK_DATA register provides the last Typed Block Data type that was written by the host. When written to, the TYPED_BLOCK_DATA register sets the Typed Block Data type of subsequent reads or writes to Typed Block Data register offsets. The Typed Block Data type should be written before transferring any Typed Block Data bytes.

If the TYPED_BLOCK_DATA register is set to 1, the Typed Block Data type is Firmware Image Data (see 7.6).

If the TYPED_BLOCK_DATA register is set to 2, the Typed Block Data type is Vendor Log Page (see 7.12).

If the TYPED_BLOCK_DATA register is set to 3, the Typed Block Data type is Label Data (see 7.15).

If the TYPED_BLOCK_DATA register is set to 4, the Typed Block Data type is Security Protocol Data (see 7.17).

TYPED_BLOCK_DATA register values from 201 to 255 are reserved for vendor specific usage.

8.4.2.2 REGION_ID0 – Offset 0x05

Definition	Access	Mand	Persist	Default
[7:0] : REGION_ID bits 7:0	RW	Y	N	0

When read from, the REGION_ID0 register provides the least significant byte of the region identifier. When written to, the host sets the least significant byte of the region identifier of subsequent Typed Block Data transfers.

The host uses the REGION_ID0 and REGION_ID1 registers to communicate to the module the region identifier of subsequent Typed Block Data transfers. The first region identifier has a value of 0.

8.4.2.3 REGION_ID1 – Offset 0x06

Definition	Access	Mand	Persist	Default
[7:0] : REGION_ID bits 15:8	RW	Y	N	0

When read from, the REGION_ID1 register provides the most significant byte of the region identifier. When written to, the host sets the most significant byte of the region identifier of subsequent Typed Block Data transfers.

The host uses the REGION_ID0 and REGION_ID1 registers to communicate to the module the region identifier of subsequent Typed Block Data transfers. The first region identifier has a value of 0.

8.4.2.4 BLOCK_ID – Offset 0x07

Definition	Access	Mand	Persist	Default
[7:0] : BLOCK_ID	RW	Y	N	0

When read from, the BLOCK_ID register provides the block identifier that was last written. When written to, the host sets the block identifier of subsequent Typed Block Data transfers.

The first block identifier has a value of 0.

8.4.2.5 TYPED_BLOCK_DATA_SIZE0 – Offset 0x08

Definition	Access	Mand	Persist	Default
[7:0] : TYPED_BLOCK_DATA_SIZE bits 7:0	RO/RW	Y	N	0

The TYPED_BLOCK_DATA_SIZE0 register contains the least significant byte of the size in multiples of 32 bytes of the current TYPED_BLOCK_DATA type. A reserved TYPED_BLOCK_DATA type shall return a Typed Block Data Size of 0. The register access is RO for the non-security protocol data and RW for security protocol data.

8.4.2.6 TYPED_BLOCK_DATA_SIZE1 – Offset 0x09

Definition	Access	Mand	Persist	Default
[7:0] : TYPED_BLOCK_DATA_SIZE bits 15:8	RO/RW	Y	N	0

The TYPED_BLOCK_DATA_SIZE1 register contains bits 15:8 of the size in multiples of 32 bytes of the current TYPED_BLOCK_DATA type. A reserved TYPED_BLOCK_DATA type shall return a Typed Block Data Size of 0. The register access is RO for the non-security protocol data and RW for security protocol data.

8.4.2.7 TYPED_BLOCK_DATA_SIZE2 – Offset 0x0A

Definition	Access	Mand	Persist	Default
[7:0] : TYPED_BLOCK_DATA_SIZE bits 23:16	RO/RW	Y	N	0

The TYPED_BLOCK_DATA_SIZE2 register contains the most significant byte of the size in multiples of 32 bytes of the current TYPED_BLOCK_DATA type. A reserved TYPED_BLOCK_DATA type shall return a Typed Block Data Size of 0. The register access is RO for the non-security protocol data and RW for security protocol data.

8.4.2.8 OPERATIONAL_UNIT_ID0 – Offset 0x0C

Definition	Access	Mand	Persist	Default
[7:0] : OPERATIONAL_UNIT_ID bits 7:0	RW	Y	N	0

The OPERATIONAL_UNIT_ID0 contains the least significant byte of the Operational Unit ID that will be used for subsequent Typed Block Data transfers.

8.4.2.9 OPERATIONAL_UNIT_ID1 – Offset 0x0D

Definition	Access	Mand	Persist	Default
[7:0] : OPERATIONAL_UNIT_ID bits 15:8	RW	Y	N	0

The OPERATIONAL_UNIT_ID1 contains the most significant byte of the Operational Unit ID that will be used for subsequent Typed Block Data transfers.

8.4.2.10 OPERATIONAL_UNIT_SIZE0 – Offset 0x10

Definition	Access	Mand	Persist	Default
[7:0] : OPERATIONAL_UNIT_SIZE bits 7:0	RO	Y	N	0

The OPERATIONAL_UNIT_SIZE0 contains the least significant byte of the Operational Unit size for the current TYPED_BLOCK_DATA type. The Operational Unit size is in multiples of 32 bytes. A reserved TYPED_BLOCK_DATA type shall return an Operational Unit size of 0.

8.4.2.11 OPERATIONAL_UNIT_SIZE1 – Offset 0x11

Definition	Access	Mand	Persist	Default
[7:0] : OPERATIONAL_UNIT_SIZE bits 15:8	RO	Y	N	0

The OPERATIONAL_UNIT_SIZE1 contains bits 15:8 of the Operational Unit size for the current TYPED_BLOCK_DATA type. The Operational Unit size is in multiples of 32 bytes. A reserved TYPED_BLOCK_DATA type shall return an Operational Unit size of 0.

8.4.2.12 OPERATIONAL_UNIT_SIZE2 – Offset 0x12

Definition	Access	Mand	Persist	Default
[7:0] : OPERATIONAL_UNIT_SIZE bits 23:16	RO	Y	N	0

The OPERATIONAL_UNIT_SIZE2 contains the most significant byte of the Operational Unit size for the current TYPED_BLOCK_DATA type. The Operational Unit size is in multiples of 32 bytes. A reserved TYPED_BLOCK_DATA type shall return an Operational Unit size of 0.

8.4.2.13 OPERATIONAL_UNIT_CRC0 – Offset 0x14

Definition	Access	Mand	Persist	Default
[7:0] : OPERATIONAL_UNIT_CRC bits 7:0	RO	Y	N	0

The OPERATIONAL_UNIT_CRC0 contains the least significant byte of the CRC that is generated by the Generate Operational Unit Checksum operation. If a CRC has not been generated for the current Operational Unit, the module shall return a value of 0.

8.4.2.14 OPERATIONAL_UNIT_CRC1 – Offset 0x15

Definition	Access	Mand	Persist	Default
[7:0] : OPERATIONAL_UNIT_CRC bits 15:8	RO	Y	N	0

The OPERATIONAL_UNIT_CRC1 contains the most significant byte of the CRC that is generated by the Generate Operational Unit Checksum operation. If a CRC has not been generated for the current Operational Unit, the module shall return a value of 0.

8.4.3 Firmware Update Registers

The registers in this category are related to firmware update (see 7.2.8, 7.6, and 9.7).

8.4.3.1 FW_REGION_CRC0 – Offset 0x40

Definition	Access	Mand	Persist	Default
[7:0] : Calculated checksum for firmware data region bits 7:0	RO	Y	N	0

The FW_REGION_CRC0 register provides the least significant byte of the calculated checksum for the firmware data region.

The host should read the FW_REGION_CRC0 and FW_REGION_CRC1 registers after transfer of each firmware data region to confirm data was transferred correctly between the host and module.

8.4.3.2 FW_REGION_CRC1 – Offset 0x41

Definition	Access	Mand	Persist	Default
[7:0] : Calculated checksum for firmware data region bits 15:8	RO	Y	N	0

The FW_REGION_CRC1 register provides the most significant byte of the calculated checksum for the firmware data region.

The host should read the FW_REGION_CRC0 and FW_REGION_CRC1 registers after transfer of each firmware data region to confirm data was transferred correctly between the host and module.

8.4.3.3 FW_SLOT_INFO – Offset 0x42

Definition	Access	Mand	Persist	Default
[3:0] : SELECT_FW_SLOT – Host selected firmware slot [7:4] : RUNNING_FW_SLOT – slot number of running firmware	RW	Y	Y	0

The FW_SLOT_INFO register provides information on the slot number of the running firmware image and the slot number of intended firmware image. The values of SELECT_FW_SLOT and RUNNING_FW_SLOT shall be either 0 or 1. All other values are reserved. The FW_SLOT_INFO register is not impacted by the Factory Default operation.

If SELECT_FW_SLOT is set to 1 and firmware slot 1 is empty or contains an invalid firmware image, the value of RUNNING_FW_SLOT shall be 0.

8.4.4 Byte and Word Transaction Data Registers for Typed Block Data

The registers in this category support transferring Typed Block Data bytes using I^2C Read Byte transactions (see 5.2.1), I^2C Write Byte transactions (see 5.2.2), I^2C Read Word transactions (see 5.2.3), and I^2C Write Word transactions (see 5.2.4).

8.4.4.1 TYPED_BLOCK_DATA_BYTE0 – Offset 0x80

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 0	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE0 register contains byte 0 of the block with identifier BLOCK_ID (see 8.4.2.4) and region identifier REGION_ID0 (see 8.4.2.2) and REGION_ID1 (see 8.4.2.3).

8.4.4.2 TYPED_BLOCK_DATA_BYTE1 – Offset 0x81

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 1	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE1 register contains byte 1 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.3 TYPED_BLOCK_DATA_BYTE2 – Offset 0x82

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 2	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE2 register contains byte 2 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.4 TYPED_BLOCK_DATA_BYTE3 – Offset 0x83

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 3	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE3 register contains byte 3 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.5 TYPED_BLOCK_DATA_BYTE4 – Offset 0x84

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 4	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE4 register contains byte 4 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.6 TYPED_BLOCK_DATA_BYTE5 – Offset 0x85

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 5	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE5 register contains byte 5 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.7 TYPED_BLOCK_DATA_BYTE6 – Offset 0x86

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 6	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE6 register contains byte 6 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.8 TYPED_BLOCK_DATA_BYTE7 – Offset 0x87

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 7	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE7 register contains byte 7 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.9 TYPED_BLOCK_DATA_BYTE8 – Offset 0x88

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 8	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE8 register contains byte 8 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.10 TYPED_BLOCK_DATA_BYTE9 – Offset 0x89

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 9	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE9 register contains byte 9 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.11 TYPED_BLOCK_DATA_BYTE10 – Offset 0x8A

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 10	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE10 register contains byte 10 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.12 TYPED_BLOCK_DATA_BYTE11 – Offset 0x8B

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 11	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE11 register contains byte 11 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.13 TYPED_BLOCK_DATA_BYTE12 – Offset 0x8C

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 12	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE12 register contains byte 12 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.14 TYPED_BLOCK_DATA_BYTE13 – Offset 0x8D

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 13	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE13 register contains byte 13 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.15 TYPED_BLOCK_DATA_BYTE14 – Offset 0x8E

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 14	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE14 register contains byte 14 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.16 TYPED_BLOCK_DATA_BYTE15 – Offset 0x8F

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 15	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE15 register contains byte 15 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.17 TYPED_BLOCK_DATA_BYTE16 – Offset 0x90

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 16	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE16 register contains byte 16 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.18 TYPED_BLOCK_DATA_BYTE17 – Offset 0x91

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 17	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE17 register contains byte 17 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.19 TYPED_BLOCK_DATA_BYTE18 – Offset 0x92

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 18	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE18 register contains byte 18 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.20 TYPED_BLOCK_DATA_BYTE19 – Offset 0x93

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 19	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE19 register contains byte 19 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.21 TYPED_BLOCK_DATA_BYTE20 – Offset 0x94

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 20	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE20 register contains byte 20 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.22 TYPED_BLOCK_DATA_BYTE21 – Offset 0x95

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 21	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE21 register contains byte 21 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.23 TYPED_BLOCK_DATA_BYTE22 – Offset 0x96

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 22	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE22 register contains byte 22 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.24 TYPED_BLOCK_DATA_BYTE23 – Offset 0x97

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 23	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE23 register contains byte 23 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.25 TYPED_BLOCK_DATA_BYTE24 – Offset 0x98

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 24	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE24 register contains byte 24 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.26 TYPED_BLOCK_DATA_BYTE25 – Offset 0x99

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 25	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE25 register contains byte 25 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.27 TYPED_BLOCK_DATA_BYTE26 – Offset 0x9A

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 26	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE26 register contains byte 26 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.28 TYPED_BLOCK_DATA_BYTE27 – Offset 0x9B

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 27	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE27 register contains byte 27 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.29 TYPED_BLOCK_DATA_BYTE28 – Offset 0x9C

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 28	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE28 register contains byte 28 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.30 TYPED_BLOCK_DATA_BYTE29 – Offset 0x9D

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 29	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE29 register contains byte 29 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.31 TYPED_BLOCK_DATA_BYTE30 – Offset 0x9E

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 30	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE30 register contains byte 30 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.4.32 TYPED_BLOCK_DATA_BYTE31 – Offset 0x9F

Definition	Access	Mand	Persist	Default
[7:0] : Typed Block Data Byte 31	RW	Y	N	0

The TYPED_BLOCK_DATA_BYTE31 register contains byte 31 of the block with identifier BLOCK_ID and region identifier REGION_ID0 and REGION_ID1.

8.4.5 Block Transaction Data Registers for Typed Blocked Data

The registers in this category support transferring Typed Block Data bytes using I^2C Block Read transactions (see 5.2.5) and I^2C Block Write transactions (see 5.2.6).

8.4.5.1 TYPED_BLOCK_DATA_OFFSET – Offset 0xE0

Definition	Access	Mand	Persist	Default
[7:0] : TYPED_BLOCK_DATA_OFFSET	RW	N	N	0

The TYPED_BLOCK_DATA_OFFSET register is the offset used to transfer a block of Typed Block Data bytes using I^2C Block Read transactions and I^2C Block Write transactions. The data transferred is for the block with identifier specified in the BLOCK_ID register (see 8.4.2.4) and the region identifier specified in the REGION_ID0 (see 8.4.2.2) and REGION_ID1 (see 8.4.2.3) registers.

8.5 Page 4 Register Map

Table 20 defines the registers that are in page 4. This page supports the Catastrophic Save transition (see 7.16). This page is mandatory if the module implements page 5.

8.5 Page 4 Register Map (cont'd)

Table 20 — Page 4 Register Map

Offset	Register Name	Clause
Paging Mechanism Registers		
0x00	OPEN_PAGE	8.5.1
0x01-0x03	Reserved	
PDA Support Registers		
0x04	PDA_SUPPORTED	8.5.2
0x05	PDA_VALID	8.5.3
0x06-0x9F	Reserved	
MRx Shared Registers		
0x10-0x11	MR0_SHARED	8.5.4
0x12-0x13	MR1_SHARED	8.5.4
0x14-0x15	MR2_SHARED	8.5.4
0x16-0x17	MR3_SHARED	8.5.4
0x18-0x19	MR4_SHARED	8.5.4
0x1A-0x1B	MR5_SHARED	8.5.4
0x1C-0x1D	MR6_SHARED	8.5.4
All others	Reserved	

8.5.1 OPEN_PAGE – Offset 0x00

Definition	Access	Mand	Persist	Default
[7:0] Open page number	RW	Y	N	0

This register is the same as the OPEN_PAGE register in page 0 (see 8.1.1.1).

8.5.2 PDA_SUPPORTED – Offset 0x04

Definition	Access	Mand	Persist	Default
[7] : Reserved [6] : MR6 PDA registers are supported [5] : MR5 PDA registers are supported [4] : MR4 PDA registers are supported [3] : MR3 PDA registers are supported [2] : MR2 PDA registers are supported [1] : MR1 PDA registers are supported [0] : MR0 PDA registers are supported	RO	Y	N	Vendor specific

The PDA_SUPPORTED register indicates if the module supports PDA registers for each of the SDRAM Mode Registers. The module should implement PDA values for MR6.

8.5.3 PDA_VALID – Offset 0x05

Definition	Access	Mand	Persist	Default
[7] : Reserved [6] : MR6 PDA values are valid [5] : MR5 PDA values are valid [4] : MR4 PDA values are valid [3] : MR3 PDA values are valid [2] : MR2 PDA values are valid [1] : MR1 PDA values are valid [0] : MR0 PDA values are valid	RW	N	N	0

The PDA_VALID register specifies that the host has written PDA values for the corresponding SDRAM Mode Registers to the corresponding registers in page 5.

8.5.4 MRx_SHARED Registers – Offsets 0x10-0x11 to 0x1C-0x1D

Each pair of MRx_SHARED registers follows the format defined in Table 8 and Table 9.

8.6 Page 5 Register Map

Table 21 defines the registers that are in page 5. This page supports the Catastrophic Save transition (see 7.16). This page contains MRx_PDA_SDRAMyy register pairs where:

- x represents the SDRAM Mode Register number, ranging from 0 to 6;
- yy represents the SDRAM, ranging from 0 to 17.

Table 21 show the MRx_PDA_SDRAMyy register pairs for MR0.

If the PDA_SUPPORTED register (see 8.5.2) indicates that the module does not support PDA values for a particular SDRAM Mode Register, then the module shall ignores writes to the corresponding MRx_PDA_SDRAMyy registers and returns 0 on reads from those registers.

8.6 Page 5 Register Map (cont'd)

Table 21 — Page 5 Register Map

Offset	Register Name	Clause
Paging Mechanism Registers		
0x00	OPEN_PAGE	8.6.1
0x01-0x03	Reserved	
MRx PDA SDRAMyy Registers		
0x04-0x27	MR0_PDA_SDRAMyy registers, if any	8.6.2
0x04-0x05	MR0_PDA_SDRAM0	8.6.2
0x06-0x07	MR0_PDA_SDRAM1	8.6.2
0x08-0x09	MR0_PDA_SDRAM2	8.6.2
0x0A-0x0B	MR0_PDA_SDRAM3	8.6.2
0x0C-0x0D	MR0_PDA_SDRAM4	8.6.2
0x0E-0x0F	MR0_PDA_SDRAM5	8.6.2
0x10-0x11	MR0_PDA_SDRAM6	8.6.2
0x12-0x13	MR0_PDA_SDRAM7	8.6.2
0x14-0x15	MR0_PDA_SDRAM8	8.6.2
0x16-0x17	MR0_PDA_SDRAM9	8.6.2
0x18-0x19	MR0_PDA_SDRAM10	8.6.2
0x1A-0x1B	MR0_PDA_SDRAM11	8.6.2
0x1C-0x1D	MR0_PDA_SDRAM12	8.6.2
0x1E-0x1F	MR0_PDA_SDRAM13	8.6.2
0x20-0x21	MR0_PDA_SDRAM14	8.6.2
0x22-0x23	MR0_PDA_SDRAM15	8.6.2
0x24-0x25	MR0_PDA_SDRAM16	8.6.2
0x26-0x27	MR0_PDA_SDRAM17	8.6.2
0x28-0x4B	MR1_PDA_SDRAMyy registers, if any	8.6.2
0x4C-0x6F	MR2_PDA_SDRAMyy registers, if any	8.6.2
0x70-0x93	MR3_PDA_SDRAMyy registers, if any	8.6.2
0x94-0xB7	MR4_PDA_SDRAMyy registers, if any	8.6.2
0xB8-0xDB	MR5_PDA_SDRAMyy registers, if any	8.6.2
0xDC-0xFF	MR6_PDA_SDRAMyy registers, if any	8.6.2

NOTE: Page 5 was not defined in JESD245 or JESD245A.

8.6.1 OPEN_PAGE – Offset 0x00

Definition	Access	Mand	Persist	Default
[7:0] Open page number	RW	Y	N	0

This register is the same as the OPEN_PAGE register in page 0 (see 8.1.1.1).

8.6.2 MRx_PDA_SDRAMyy Registers – Offsets 0x10-0x11 to 0xFE-0xFF

Each pair of MRx_PDA_SDRAMyy registers follows the format defined in Table 8 and Table 9.

8.7 Page 6 Register Map

Table 22 defines the registers that are in page 6. This page supports the Catastrophic Save transition (see 7.16). This page is mandatory if the module implements page 7, page 8, or page 9.

Table 22 — Page 6 Register Map

Offset	Register Name	Clause
Paging Mechanism Registers		
0x00	OPEN_PAGE	8.7.1
0x01-0x03	Reserved	
Supported RCD Functions Registers		
0x04	RCD_FUNCTIONS_SUPPORTED0	8.7.2
0x05	RCD_FUNCTIONS_SUPPORTED1	8.7.3
0x06	RCD_FUNCTIONS_VALID0	8.7.4
0x07	RCD_FUNCTIONS_VALID1	8.7.5
0x08-0x7F	Reserved	
Control Word Registers		
0x80-0x9F	Control word registers for RCD function 0	8.7.6
0xA0-0xBF	Control word registers for RCD function 1	8.7.6
0xC0-0xDF	Control word registers for RCD function 2	8.7.6
0xE0-0xFF	Control word registers for RCD function 3	8.7.6

NOTE: Page 6 was not defined in JESD245 or JESD245A.

8.7.1 OPEN_PAGE – Offset 0x00

Definition	Access	Mand	Persist	Default
[7:0] Open page number	RW	Y	N	0

This register is the same as the OPEN_PAGE register in page 0 (see 8.1.1.1).

8.7.2 RCD_FUNCTIONS_SUPPORTED0 – Offset 0x04

Definition	Access	Mand	Persist	Default
[7] : Function 7 registers are supported [6] : Function 6 registers are supported [5] : Function 5 registers are supported [4] : Function 4 registers are supported [3] : Function 3 registers are supported [2] : Function 2 registers are supported [1] : Function 1 registers are supported [0] : Function 0 registers are supported	RO	Y	N	Vendor specific

This register indicates if the module supports registers for each of the RCD functions, along with the corresponding bits in the RCD_FUNCTIONS_VALID0 register. The module should implement registers for RCD function 0 and RCD function 1.

8.7.3 RCD_FUNCTIONS_SUPPORTED1 – Offset 0x05

Definition	Access	Mand	Persist	Default
[7] : Function 15 registers are supported [6] : Function 14 registers are supported [5] : Function 13 registers are supported [4] : Function 12 registers are supported [3] : Function 11 registers are supported [2] : Function 10 registers are supported [1] : Function 9 registers are supported [0] : Function 8 registers are supported	RO	Y	N	Vendor specific

This register indicates if the module supports registers for each of the RCD functions, along with the corresponding bits in the RCD_FUNCTIONS_VALID1 register.

8.7.4 RCD_FUNCTIONS_VALID0 – Offset 0x06

Definition	Access	Mand	Persist	Default
[7] : Function 7 registers are valid [6] : Function 6 registers are valid [5] : Function 5 registers are valid [4] : Function 4 registers are valid [3] : Function 3 registers are valid [2] : Function 2 registers are valid [1] : Function 1 registers are valid [0] : Function 0 registers are valid	RW	Y	N	Vendor specific

Each bit in this register specifies that the host has written values for control words for the corresponding RCD function to the corresponding registers in page 7, page 8, and page 9.

8.7.5 RCD_FUNCTIONS_VALID1 – Offset 0x07

Definition	Access	Mand	Persist	Default
[7] : Function 15 registers are valid [6] : Function 14 registers are valid [5] : Function 13 registers are valid [4] : Function 12 registers are valid [3] : Function 11 registers are valid [2] : Function 10 registers are valid [1] : Function 9 registers are valid [0] : Function 8 registers are valid	RW	Y	N	Vendor specific

Each bit in this register specifies that the host has written values for control words for the corresponding RCD function to the corresponding registers in page 7, page 8, and page 9.

8.7.6 Control Word Registers

Each set of function registers matches the I^2C bus address map for the corresponding RCD function in JESD82-31A (DDR4RCD02).

8.8 Page 7 Register Map

Table 23 defines the registers that are in page 7. This page supports the Catastrophic Save transition (see 7.16).

8.8 Page 7 Register Map (cont'd)

Table 23 — Page 7 Register Map

Offset	Register Name	Clause
Paging Mechanism Registers		
0x00	OPEN_PAGE	8.8.1
0x01-0x03	Reserved	
Reserved Registers		
0x04-0x7F	Reserved	
Control Word Registers		
0x80-0x9F	Control word registers for RCD function 4	8.8.2
0xA0-0xBF	Control word registers for RCD function 5	8.8.2
0xC0-0xDF	Control word registers for RCD function 6	8.8.2
0xE0-0xFF	Control word registers for RCD function 7	8.8.2

NOTE: Page 7 was not defined in JESD245 or JESD245A.

8.8.1 OPEN_PAGE – Offset 0x00

Definition	Access	Mand	Persist	Default
[7:0] Open page number	RW	Y	N	0

This register is the same as the OPEN_PAGE register in page 0 (see 8.1.1.1).

8.8.2 Control Word Registers

Each set of function registers matches the I^2C bus address map for the corresponding RCD function in JESD82-31A (DDR4RCD02).

8.9 Page 8 Register Map

Table 24 defines the registers that are in page 8. This page supports the Catastrophic Save transition (see 7.16).

Table 24 — Page 8 Register Map

Offset	Register Name	Clause
Paging Mechanism Registers		
0x00	OPEN_PAGE	8.9.1
0x01-0x03	Reserved	
Reserved Registers		
0x04-0x7F	Reserved	
Control Word Registers		
0x80-0x9F	Control word registers for RCD function 8	8.9.2
0xA0-0xBF	Control word registers for RCD function 9	8.9.2
0xC0-0xDF	Control word registers for RCD function 10	8.9.2
0xE0-0xFF	Control word registers for RCD function 11	8.9.2

NOTE: Page 8 was not defined in JESD245 or JESD245A.

8.9.1 OPEN_PAGE – Offset 0x00

Definition	Access	Mand	Persist	Default
[7:0] Open page number	RW	Y	N	0

This register is the same as the OPEN_PAGE register in page 0 (see 8.1.1.1).

8.9.2 Control Word Registers

Each set of function registers matches the I^2C bus address map for the corresponding RCD function in JESD82-31A (DDR4RCD02).

8.10 Page 9 Register Map

Table 25 defines the registers that are in page 9. This page supports the Catastrophic Save transition (see 7.16).

Table 25 — Page 9 Register Map

Offset	Register Name	Clause
Paging Mechanism Registers		
0x00	OPEN_PAGE	8.10.1
0x01-0x03	Reserved	
Reserved Registers		
0x04-0x7F	Reserved	
Control Word Registers		
0x80-0x9F	Control word registers for RCD function 12	8.10.2
0xA0-0xBF	Control word registers for RCD function 13	8.10.2
0xC0-0xDF	Control word registers for RCD function 14	8.10.2
0xE0-0xFF	Control word registers for RCD function 15	8.10.2

NOTE: Page 9 was not defined in JESD245 or JESD245A.

8.10.1 OPEN_PAGE – Offset 0x00

Definition	Access	Mand	Persist	Default
[7:0] Open page number	RW	Y	N	0

This register is the same as the OPEN_PAGE register in page 0 (see 8.1.1.1).

8.10.2 Control Word Registers

Each set of function registers matches the I^2C bus address map for the corresponding RCD function in JESD82-31A (DDR4RCD02).

9 Host Operation Workflows

This clause describes the workflow for various operations from the host perspective.

9.1 Controller Ready Workflow

To determine whether the module is ready for host access on system boot or after a Reset Controller operation, the host should do the following:

1. Wait for the NVDIMM_READY register to be set to 0xA5. If the NVDIMM_READY register is not set to 0xA5 within the timeout reported in SPD byte 203 (Maximum Non-Volatile Memory Initialization Time) (see JESD21C Page 4.1.2.L-4), then report an error and abort the Controller Ready workflow.

All other workflows defined in clause 9 assume that the Controller Ready workflow has been performed.

9.2 Catastrophic Save Workflow

To start a Catastrophic Save operation (see 7.2.1) through the I^2C register interface, the host should do the following:

1. Wait for the Operation In Progress bit in the NVDIMM_CMD_STATUS0 register to be clear.
2. Set bit 2 (Clear the CSAVE_STATUS register) in the NVDIMM_MGT_CMD0 register.
3. 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)). If the host does not complete this task prior to initiating a Catastrophic Save operation through the I2C register interface can result in a failed CSAVE operation.
4. Set the START_CSAVE bit in the NVDIMM_FUNC_CMD register.
5. Check the Catastrophic Save In Progress bit in the NVDIMM_CMD_STATUS0 register. If clear, check the CSAVE_STATUS register. If the Catastrophic Save operation is not done, retry step 3.
6. Wait for the Catastrophic Save In Progress bit in the NVDIMM_CMD_STATUS0 register to be clear. If the Catastrophic Save In Progress bit is not clear within the time indicated in the CSAVE_TIMEOUT0 and CSAVE_TIMEOUT1 registers, abort the Catastrophic Save operation (see 9.6).
7. Check the CSAVE_STATUS register. If the Catastrophic Save operation was rejected, had an error, or was aborted, report an error. The host may retry up to the Maximum Catastrophic Save retry count indicated in HOST_MAX_OPERATION_RETRY.

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

9.3 Restore Workflow

To start a Restore operation (see 7.2.2), the host should do the following:

1. Wait for the Operation In Progress bit in the NVDIMM_CMD_STATUS0 register to be clear.
2. Set bit 3 (Clear the RESTORE_STATUS register) in the NVDIMM_MGT_CMD0 register.
3. Set the START_RESTORE bit in the NVDIMM_FUNC_CMD register.

9.3 Restore Workflow (cont'd)

4. Check the Restore In Progress bit in the NVDIMM_CMD_STATUS0 register. If clear, check the RESTORE_STATUS register. If the Restore operation is not done, retry step 3.
5. Wait for the Restore In Progress bit in the NVDIMM_CMD_STATUS0 register to be clear. If the Restore In Progress bit is not clear within the time indicated in the RESTORE_TIMEOUT0 and RESTORE_TIMEOUT1 registers, abort the Restore operation (see 9.6).
6. Check the RESTORE_STATUS register. If the Restore operation had an error or was aborted, report an error and do not proceed to the Arm and Erase workflows. The host may retry up to the Maximum Restore retry count indicated in HOST_MAX_OPERATION_RETRY.

9.4 Arm Workflow

To start an Arm operation (see 7.2.4), the host should do the following:

1. Wait for the Operation In Progress bit in the NVDIMM_CMD_STATUS0 register to be clear.
2. Ensure the module is unlocked (i.e., a write of Typed Block Data type 4h (see 8.4.2.1) to unlock the module (see SIIS) has completed successfully).
3. Set bit 5 (Clear the ARM_STATUS register and the ARM_FAIL_INFO register) in the NVDIMM_MGT_CMD0 register.
4. Set the appropriate bits in the ARM_CMD register to reflect the Catastrophic Save operation triggers to be enabled.
5. Check the Arm In Progress bit in the NVDIMM_CMD_STATUS0 register. If clear, check the ARM_STATUS register. If the Arm operation is not done, retry step 3.
6. NVDIMM_CMD_STATUS0: wait for the Arm In Progress bit to be clear. If the Arm In Progress bit is not clear within the time indicated in the ARM_TIMEOUT0 and ARM_TIMEOUT1 registers, abort the Arm operation (see 9.6).
7. Check the ARM_STATUS register. If the Arm operation had an error or was aborted, report an error and do not proceed to the Erase workflow.

9.5 Erase Workflow

To start an Erase operation (see 7.2.3), the host should do the following:

1. Wait for the Operation In Progress bit in the NVDIMM_CMD_STATUS0 register to be clear.
2. Ensure the module is unlocked (i.e., a write of Typed Block Data type 4h (see 8.4.2.1) to unlock the module (see SIIS) has completed successfully).
3. Set bit 4 (Clear the ERASE_STATUS register and ERASE_FAIL_INFO register) in the NVDIMM_MGT_CMD0 register.
4. Set the START_ERASE bit in the NVDIMM_FUNC_CMD register.
5. Check the Erase In Progress bit in the NVDIMM_CMD_STATUS0 register. If clear, check the ERASE_STATUS register. If the Erase operation is not done, retry step 3.
6. Wait for the Erase In Progress bit in the NVDIMM_CMD_STATUS0 register to be clear. If the Erase In Progress bit is not clear within the time indicated in the ERASE_TIMEOUT0 and ERASE_TIMEOUT1 registers, abort the Erase operation (see 9.6).

9.5 Erase Workflow (cont'd)

7. Check the ERASE_STATUS register. If the Erase operation had an error or was aborted, then check the ERASE_FAIL_INFO register and report an error. The host may retry up to the Maximum Erase retry count indicated in HOST_MAX_OPERATION_RETRY.

9.6 Abort Running Operation Workflow

To start an Abort operation (see 7.2.13), the host should do the following:

1. Set the ABORT_CURRENT_OP bit in the NVDIMM_FUNC_CMD register.
2. Check the Abort In Progress bit in the NVDIMM_CMD_STATUS0 register. If clear, check the _STATUS register for the operation being aborted. If the Abort operation is not done, retry step 1.
3. Wait for the Abort In Progress bit in the NVDIMM_CMD_STATUS0 register and the In Progress bit for the operation being aborted to be clear. If the bits are not clear within the time indicated in the ABORT_CMD_TIMEOUT register, abort the Abort Running Operation workflow.
4. Check the _STATUS register for the operation being aborted for ABORT_SUCCESS or ABORT_ERROR. If the Abort operation had an error, report an error.
5. If an ABORT operation does timeout the host should not perform any additional operations to the NVDIMM.

9.7 Firmware Update Workflow

Firmware image data is transferred and committed to the module in REGION_BLOCK_SIZE size blocks. To update a firmware image on the module, the host should do the following:

1. Ensure the module is unlocked (i.e., a write of Typed Block Data type 4h (see 8.4.2.1) to unlock the module (see SIIS) has completed successfully).
2. Validate that the Module Manufacturer ID and Module Product Identifier in the common header of the firmware image matches the values stored in the module's SPD (bytes 320 and 321 for the Module Manufacturer ID and bytes 192 and 193 for the Module Product Identifier)(see JESD21C Page 4.1.2.L-4). If the values do not match, abort the firmware update workflow and do not transfer any data to the module. The host may do additional verification by verifying the calculated checksum of the image data matches the checksum in the common header.
3. Wait for the Operation In Progress bit in the NVDIMM_CMD_STATUS0 register to be clear.
4. Set bit 1 (Clear the FIRMWARE_OPS_STATUS register and the FIRMWARE_OPS_FAIL_INFO register)) in the NVDIMM_MGT_CMD1 register.
5. Enable firmware update mode.
    a. Write the FIRMWARE_OPS_CMD register with the value 0x1 to enable firmware update mode.
    b. Check the Firmware Ops In Progress bit in the NVDIMM_CMD_STATUS0 register.
    c. If the Firmware Ops In Progress bit is set, wait for the Firmware Ops In Progress bit to be clear. If the Firmware Ops In Progress bit is not clear within the time indicated in the FIRMWARE_OPS_TIMEOUT0 and FIRMWARE_OPS_TIMEOUT1 registers, abort the Firmware operation and abort the firmware update workflow.

9.7 Firmware Update Workflow (cont'd)

d. If the Firmware Ops In Progress is not set, check FIRMWARE_OPS_STATUS register for status of enable firmware update mode. If not successful, abort the firmware update workflow.

e. In firmware update mode, no other operations are supported.

f. Verify the firmware update mode bit (bit 2) it set in FIRMWARE_OPS_STATUS register.

6. Set bit 1 (Clear the FIRMWARE_OPS_STATUS register and the FIRMWARE_OPS_FAIL_INFO register) in the NVDIMM_MGT_CMD1 register.

7. Clear the firmware data block to ensure there is no residual data.

   a. Write the FIRMWARE_OPS_CMD register with value 0x2 (Start a Clear Firmware operation).

   b. Check the Firmware Ops In Progress bit in the NVDIMM_CMD_STATUS0 register.

   c. If the Firmware Ops In Progress bit is set, wait for the Firmware Ops In Progress bit to be clear. If the Firmware Ops In Progress bit is not clear within the time indicated in the FIRMWARE_OPS_TIMEOUT0 and FIRMWARE_OPS_TIMEOUT1 registers, abort the Firmware operation and abort the firmware update workflow.

   d. If the Firmware Ops In Progress bit is not set, check the FIRMWARE_OPS_STATUS register for status of the Clear Firmware operation. If the operation is not successful, abort the firmware update workflow.

8. Send the data for the first firmware data region.

   a. Write the TYPED_BLOCK_DATA register with value 0x1 (Firmware Image Data).

   b. Write the BLOCK_ID, REGION_ID0 and REGION_ID1 registers with value 0.

   c. Send the bytes from the first firmware data region to the module using either an I^2C Block Write transaction to the TYPED_BLOCK_DATA_OFFSET register or I^2C Write Byte transactions or I^2C Write Word transactions to the TYPED_BLOCK_DATA_BYTE0 to TYPED_BLOCK_DATA_BYTE31 registers. Data transfer will be done in 32 byte blocks. After a block has been transferred, verify that the 32-byte block was received by polling FIRMWARE_OPS_STATUS offset for FIRMWARE_BLOCK_RECEIVED. Once the block is successfully received by the module, increment and write the updated block identifier to the BLOCK_ID register and keep transferring blocks of data until all data in the region has been sent.

   d. Validate that the data was transferred correctly to the module by ensuring checksum calculated by module matches the host calculated checksum for the firmware data region that has been transferred.

      i. Calculate the checksum for the firmware data region transferred using the checksum algorithm described in 7.7.

      ii. Set bit 1 (Clear the FIRMWARE_OPS_STATUS register and the FIRMWARE_OPS_FAIL_INFO register) in the NVDIMM_MGT_CMD1 register.

9.7 Firmware Update Workflow (cont'd)

iii. Command the module to calculate checksum for the firmware data region. Write the FIRMWARE_OPS_CMD register with value 0x4 (Start a Generate Firmware Checksum operation).

iv. Check the Firmware Ops In Progress bit in the NVDIMM_CMD_STATUS0 register.

v. If the Firmware Ops In Progress bit is set, wait for the Firmware Ops In Progress bit to be clear. If the Firmware Ops In Progress bit is not clear within the time indicated in the FIRMWARE_OPS_TIMEOUT0 and FIRMWARE_OPS_TIMEOUT1 registers, abort the Firmware operation and abort the firmware update workflow.

vi. If the Firmware Ops In Progress bit is not set, check the FIRMWARE_OPS_STATUS register for status of the Generate Firmware Checksum operation.

vii. Read the FW_REGION_CRC0 and FW_REGION_CRC1 registers and verify the module calculated checksum matches the host calculated checksum. If the checksums do not match, the host may attempt to transfer the bytes again or abort the firmware update workflow.

9. Command the module to validate that the firmware image is valid for the module based on the header.

    a. Set bit 1 (Clear the FIRMWARE_OPS_STATUS register) in the NVDIMM_MGT_CMD1 register.

    b. Write the FIRMWARE_OPS_CMD register with value 0x10 (Start a Validate Firmware Header operation) to ensure the host is transferring the correct image.

    c. Check the Firmware Ops In Progress bit in the NVDIMM_CMD_STATUS0 register.

    d. If the Firmware Ops In Progress bit is set, wait for the Firmware Ops In Progress bit to be clear. If the Firmware Ops In Progress bit is not clear within the time indicated in the FIRMWARE_OPS_TIMEOUT0 and FIRMWARE_OPS_TIMEOUT1 registers, abort the Firmware operation and abort the firmware update workflow.

    e. If the Firmware Ops In Progress bit is not set, check FIRMWARE_OPS_STATUS register for status of the Validate Firmware Header operation. If the operation fails, abort the firmware update workflow.

10. Commit the first firmware data region.

    a. Set bit 1 (Clear the FIRMWARE_OPS_STATUS register and the FIRMWARE_OPS_FAIL_INFO register) in the NVDIMM_MGT_CMD1 register.

    b. Write the FIRMWARE_OPS_CMD register with value 0x8 (Start a Commit Firmware operation).

    c. Check the Firmware Ops In Progress bit in the NVDIMM_CMD_STATUS0 register.

    d. If the Firmware Ops In Progress bit is set, wait for the Firmware Ops In Progress bit to be clear. If the Firmware Ops In Progress bit is not clear within the time indicated in the FIRMWARE_OPS_TIMEOUT0 and FIRMWARE_OPS_TIMEOUT1 registers, abort the Firmware operation and abort the firmware update workflow.

9.7 Firmware Update Workflow (cont'd)

e. If the Firmware Ops In Progress bit is not set, check FIRMWARE_OPS_STATUS register for status of the Commit Firmware operation. If the operation fails, the host may retry the operation or abort the firmware update workflow.

11. Send and commit the rest of the firmware image data in REGION_BLOCK_SIZE sized blocks. If the firmware image data size is not an integer multiple of 32 bytes, the host should pad the last block with zeroes so the transfer size is 32 bytes. If the firmware image data size is not an integer multiple of the region block size, the host should pad the last block with zeroes so the checksum calculation covers known values. For each firmware data region transfer, the host does the following:

    a. Write the BLOCK_ID, REGION_ID0 and REGION_ID1 registers with the appropriate value starting at 0.

    b. Clear the firmware data block to ensure there is no residual data.

        i. Write the FIRMWARE_OPS_CMD register with value 0x2 (Start a Clear Firmware operation).

        ii. Check the Firmware Ops In Progress bit in the NVDIMM_CMD_STATUS0 register.

        iii. If the Firmware Ops In Progress bit is set, wait for the Firmware Ops In Progress bit to be clear. If the Firmware Ops In Progress bit is not clear within the time indicated in the FIRMWARE_OPS_TIMEOUT0 and FIRMWARE_OPS_TIMEOUT1 registers, abort the Firmware operation and abort firmware update workflow.

        iv. If the Firmware Ops In Progress bit is not set, check the FIRMWARE_OPS_STATUS register for status of the Clear Firmware operation. If the operation is not successful, abort the firmware update workflow.

    c. Send the bytes from the first firmware data region to the module using either an I^2C Block Write transaction to the TYPED_BLOCK_DATA_OFFSET register or I^2C Write Byte transactions or I^2C Write Word transactions to the TYPED_BLOCK_DATA_BYTE0 to TYPED_BLOCK_DATA_BYTE31 registers. Data transfer will be done in 32 byte multiples. After a block has been transferred, increment and write the updated block identifier to the BLOCK_ID register and keep transferring blocks of data until all data in the region has been sent.

    d. Assuming bytes were sent successfully, validate that the data was transferred correctly to the module by ensuring checksum calculated by module matches the host's checksum for the firmware data region that has been transferred.

        i. Calculate the checksum for the firmware data region transferred using the checksum algorithm described in 7.7.

        ii. Set bit 1 (Clear the FIRMWARE_OPS_STATUS register and the FIRMWARE_OPS_FAIL_INFO register) in the NVDIMM_MGT_CMD1 register.

        iii. Command the module to calculate checksum for the firmware data region. Write the FIRMWARE_OPS_CMD register with value 0x4 (Start a Generate Firmware Checksum operation).

9.7 Firmware Update Workflow (cont'd)

iv. Check the Firmware Ops In Progress bit in the NVDIMM_CMD_STATUS0 register.

v. If the Firmware Ops In Progress bit is set, wait for the Firmware Ops In Progress bit to be clear. If the Firmware Ops In Progress bit is not clear within the time indicated in the FIRMWARE_OPS_TIMEOUT0 and FIRMWARE_OPS_TIMEOUT1 registers, abort the Firmware operation and abort the firmware update workflow.

vi. If the Firmware Ops In Progress bit is not set, check the FIRMWARE_OPS_STATUS register for status of the Generate Firmware Checksum operation. This operation shall never result in a failure from the module.

e. Read the FW_REGION_CRC0 and FW_REGION_CRC1 registers and verify the module calculated checksum matches the host calculated checksum. If checksums do not match, the host may attempt to transfer the bytes again or abort the firmware update workflow.

f. Commit the firmware data region transferred.

i. Set bit 1 (Clear the FIRMWARE_OPS_STATUS register and the FIRMWARE_OPS_FAIL_INFO register) in the NVDIMM_MGT_CMD1 register.

ii. Write the FIRMWARE_OPS_CMD register with value 0x8 (Start a Commit Firmware operation).

iii. Check the Firmware Ops In Progress bit in the NVDIMM_CMD_STATUS0 register.

iv. If the Firmware Ops In Progress bit is set, wait for the Firmware Ops In Progress bit to be clear. If the Firmware Ops In Progress bit is not clear within the time indicated in the FIRMWARE_OPS_TIMEOUT0 and FIRMWARE_OPS_TIMEOUT1 registers, abort the Firmware operation and abort the firmware update workflow.

v. If the Firmware Ops In Progress bit is not set, check FIRMWARE_OPS_STATUS register for status of the Commit Firmware operation. If the operation fails, the host may retry the operation or abort the firmware update workflow.

12. After all of the firmware data has been transferred to the module, ask the module to validate the firmware image data by the following:

a. Set bit 1 (Clear the FIRMWARE_OPS_STATUS register and the FIRMWARE_OPS_FAIL_INFO register) in the NVDIMM_MGT_CMD1 register.

b. Write the FIRMWARE_OPS_CMD register with value 0x20 (Start a Validate Firmware Image operation) to command the module to validate that a correct firmware image was committed.

c. Check the Firmware Ops In Progress bit in the NVDIMM_CMD_STATUS0 register.

9.7 Firmware Update Workflow (cont'd)

d. If the Firmware Ops In Progress bit is set, wait for the Firmware Ops In Progress to be clear. If the Firmware Ops In Progress bit is not clear within the time indicated in the FIRMWARE_OPS_TIMEOUT0 and FIRMWARE_OPS_TIMEOUT1 registers, abort the Firmware operation and abort the firmware update workflow.

e. If the Firmware Ops In Progress bit is not set, check the FIRMWARE_OPS_STATUS register for status of the Validate Firmware Image operation. If the operation fails, abort the firmware update workflow.

13. The last step of aborting or completing the firmware update workflow is to disable firmware update mode on the module by the following:

a. Set bit 1 (Clear the FIRMWARE_OPS_STATUS register and the FIRMWARE_OPS_FAIL_INFO register) in the NVDIMM_MGT_CMD1 register.

b. Write the FIRMWARE_OPS_CMD register with the value 0x0 to disable firmware update mode.

c. Check the Firmware Ops In Progress bit in the NVDIMM_CMD_STATUS0 register.

d. If the Firmware Ops In Progress bit is set, wait for the Firmware Ops In Progress bit to be clear. If the Firmware Ops In Progress bit is not clear within the time indicated in the FIRMWARE_OPS_TIMEOUT0 and FIRMWARE_OPS_TIMEOUT1 registers, abort the Firmware operation and abort the firmware update workflow.

e. If the Firmware Ops In Progress bit is not set, check FIRMWARE_OPS_STATUS register for status of disable firmware update mode.

To select which firmware image to run, the host writes to the SELECT~~ED~~_FW_SLOT field in the FW_SLOT_INFO register with either 0 or 1. On the next power cycle or Reset Controller operation, the module shall use the firmware image from the selected slot. If the selected slot does not have a valid image, the module shall revert to the firmware image in the other slot.

If a Firmware operation fails, the host may retrieve information about the cause of the failure by examining the FIRMWARE_OPS_FAIL_INFO register and/or reading the Vendor Log Page.

9.8 Typed Block Data Workflow

This subclause describes the recommended workflow for the host to read and write to Typed Block Data. If the module supports an Operational Unit Buffer for each Typed Block Data type (see 8.1.3.2), the host can do transfers to multiple Typed Block Data types in parallel. If the module does not support an Operational Unit Buffer for each Typed Block Data type, the host should do transfers to one Typed Block Data type at a time.

Operational Unit operations may time out. The host should use the OPERATIONAL_UNIT_OPS_TIMEOUT0 and OPERATIONAL_UNIT_OPS_TIMEOUT1 registers to determine the maximum amount of time to wait for an Operational Unit operation to complete.

If 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:

a) shall fail attempts to access (i.e., read or write) Typed Block Data for:
- Firmware Image data (i.e., type 1h);
- Vendor Log Page (i.e., type 2h); or
- Label Data (i.e., type 3h), and

9.8 Typed Block Data Workflow (cont'd)

b) may fail attempts to access (i.e. read or write) Typed Block Data for:
   - Vendor Specific (i.e., 201 to 255),

by setting the OPERATIONAL_UNIT_OPS_ERROR bit in the OPERATIONAL_UNIT_OPS_STATUS register and the SECURITY_ERROR bit in the OPERATIONAL_UNIT_FAIL_INFO register.

9.8.1 Reading Typed Block Data Workflow

To read from Typed Block Data for non-security protocol data type, the host should do the following:

1. Set the TYPED_BLOCK_DATA register to the desired Typed Block Data type.
2. Read the TYPED_BLOCK_DATA_SIZE0, TYPED_BLOCK_DATA_SIZE1, and TYPED_BLOCK_DATA_SIZE2 registers to determine the number of blocks to read for the Typed Block Data and calculate the starting Operational Unit ID for the desired data.
3. Set Bit 2, Clear Operational Unit Buffer, to 1 in the OPERATIONAL_UNIT_OPS_CMD register to clear the intermediate buffer on the module. Poll the OPERATIONAL_UNIT_OPS_STATUS register for completion of the Clear Operational Unit Buffer operation.
4. Set the OPERATIONAL_UNIT_ID0 and OPERATIONAL_UNIT_ID1 registers to the Operational Unit ID of the desired data.
5. Set Bit 0, Get Operational Unit, to 1 in the OPERATIONAL_UNIT_OPS_CMD register to retrieve the data into an intermediate buffer in the module. Poll the OPERATIONAL_UNIT_OPS_STATUS register for completion of the Get Operational Unit operation.
6. Set the BLOCK_ID, REGION_ID0 and REGION_ID1 registers to the appropriate value within the Operational Unit.
7. Read either the TYPED_BLOCK_DATA_BYTE0 to TYPED_BLOCK_DATA_BYTE31 registers or the TYPED_BLOCK_DATA_OFFSET register to read the data for the block from the module.
8. If the BLOCK_ID is not the last block of the operational unit, increment the BLOCK_ID register. If the BLOCK_ID register has the value 255 prior to the increment, set the BLOCK_ID register to 0 and increment the REGION_ID0 and REGION_ID1 registers.
9. Repeat Steps 7 to 8 until all the data from the Operational Unit is received.
10. Set Bit 3, Generate Operational Unit Checksum, to 1 in the OPERATIONAL_UNIT_OPS_CMD register to request the module to calculate the Operational Unit CRC value for the data retrieved. Poll the OPERATIONAL_UNIT_OPS_STATUS register for completion of the Generate Operational Unit Checksum operation.
11. To ensure there was no data error during transfer of the Operational Unit data, the host should calculate the CRC for Operational Unit data received from the module and verify that the CRC value matches the values in the OPERATIONAL_UNIT_CRC0 and OPERATIONAL_UNIT_CRC1 registers.
12. Increment the Operational Unit ID in the OPERATIONAL_UNIT_ID0 and OPERATIONAL_UNIT_ID1 registers.
13. Repeat Steps 3 to 12 until all desired data is received.

9.8.2 Writing Typed Block Data Workflow

To write to Typed Block Data for non-security protocol data type, the host should do the following:

1. Set the TYPED_BLOCK_DATA register to the desired Typed Block Data type.
2. Read the TYPED_BLOCK_DATA_SIZE0, TYPED_BLOCK_DATA_SIZE1, and TYPED_BLOCK_DATA_SIZE2 registers. Determine the number of blocks to write for the Typed Block Data and calculate the starting Operational Unit ID of the data to be written. If the data size is not an integer multiple of the operational unit size, the host should pad the last block with zeroes so the checksum calculation covers known values.
3. Set Bit 2, Clear Operational Unit Buffer, to 1 in the OPERATIONAL_UNIT_OPS_CMD register to clear the intermediate buffer on the module. Poll the OPERATIONAL_UNIT_OPS_STATUS register for completion of the Clear Operational Unit Buffer operation.
4. Set the OPERATIONAL_UNIT_ID0 and OPERATIONAL_UNIT_ID1 registers to the Operational Unit ID of the data to be written.
5. Calculate the Operational Unit CRC for the Operational Unit data to be written.
6. Set the BLOCK_ID, REGION_ID0 and REGION_ID1 registers to 0.
7. Write data to either the TYPED_BLOCK_DATA_BYTE0 to TYPED_BLOCK_DATA_BYTE31 registers or the TYPED_BLOCK_DATA_OFFSET register to send data to the module.
8. If the BLOCK_ID is not the last block of the operational unit, increment the BLOCK_ID register. If the BLOCK_ID register has the value 255 prior to the increment, set the BLOCK_ID register to 0 and increment the REGION_ID0 and REGION_ID1 registers.
9. Repeat Steps 7 to 8 until all the data from the Operational Unit is sent.
10. Set Bit 3, Generate Operational Unit Checksum, to 1 in the OPERATIONAL_UNIT_OPS_CMD register to request the module to calculate the Operational Unit CRC value for the data received by the module. Poll the OPERATIONAL_UNIT_OPS_STATUS register for completion of the Generate Operational Unit Checksum operation.
11. To ensure there was no data error during transfer of the Operational Unit data, verify that the CRC value calculated in step 5 matches the values in the OPERATIONAL_UNIT_CRC0 and OPERATIONAL_UNIT_CRC1 registers.
12. Set Bit 1, Set Operational Unit, to 1 in the OPERATIONAL_UNIT_OPS_CMD register to commit the Operational Unit data on the module. Poll the OPERATIONAL_UNIT_OPS_STATUS register for completion of the Set Operational Unit operation.
13. Increment the Operational Unit ID in the OPERATIONAL_UNIT_ID0 and OPERATIONAL_UNIT_ID1 registers.
14. Repeat Steps 3 to 13 until all desired data is sent.

Typed Block Data updates shall be supported only at Operational Unit size granularity. If the host is updating data that does not start at the beginning of an Operational Unit or the last byte is not at the end of an Operational Unit, the host needs to first read in the Operational Unit data before modifying it.

9.8.2 Writing Typed Block Data Workflow (cont'd)

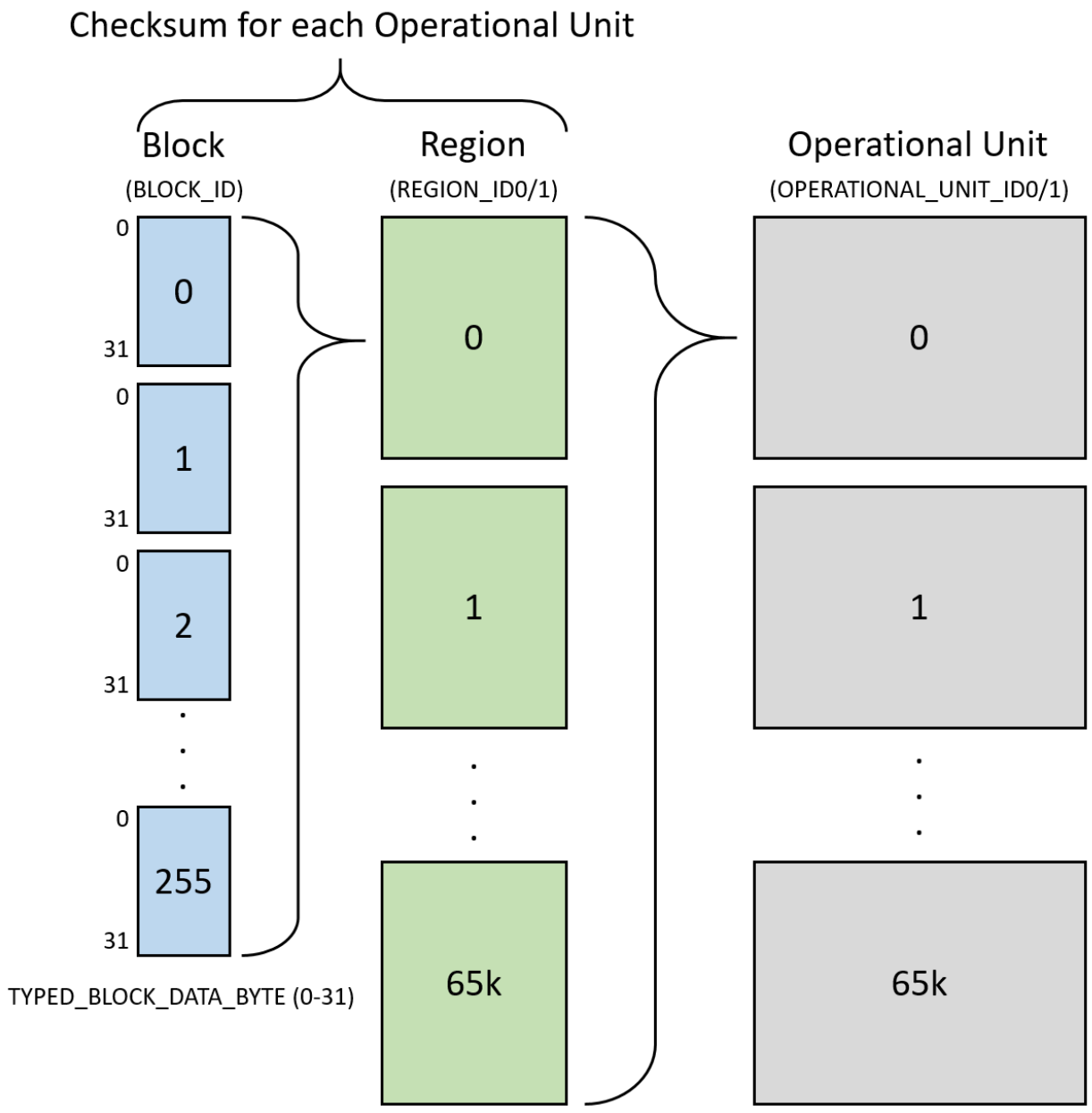

Figure 11 — Typed Block Data Block Diagram

9.8.3 Security Protocol Data Workflow

This subclause describes the recommended workflow for the host to read and write Security Protocol Data. If the module supports encryption, Security Protocol Data is the Typed Block Data that is transferred between the host and the module to manage the encryption and security aspects of the module.

Prior to transferring Security Protocol Data between the host and the module, the host should set the SECURITY_PROTOCOL_TYPE (see 8.1.4.11) register, the SECURITY_PROTOCOL_SPECIFIC0 register (see 8.1.4.9), and the SECURITY_PROTOCOL_SPECIFIC1 register (see 8.1.4.10) to describe the Security Protocol Data type information contained in the Typed Block Data.

9.8.3 Security Protocol Data Workflow (cont'd)

The Host shall set the TYPED_BLOCK_DATA register to Security Protocol Data 4h (see 7.17) to access the Security Protocol Data.

Since the Data transfers to an Operational Unit are done in 32 byte blocks and the Operational Unit size for Security Protocol Data is 256 bytes, REGION_ID0 and REGION_ID1 are not used for transfer between the host and the module.

Operational Unit operations may time out. The host should use the OPERATIONAL_UNIT_OPS_TIMEOUT0 and OPERATIONAL_UNIT_OPS_TIMEOUT1 registers to determine the maximum amount of time to wait for an Operational Unit operation to complete.

The Security Protocol Data write is used to map the TCG IF-SEND function (see SIIS). To write the data, the host should do the following:

1. Set the OPERATIONAL_UNIT_ID0 and OPERATIONAL_UNIT_ID1 registers to 0.
2. Set the TYPED_BLOCK_DATA_SIZE0, TYPED_BLOCK_DATA_SIZE1, and TYPED_BLOCK_DATA_SIZE2 registers to the appropriate value for the operation.
3. Set Bit 2, Clear Operational Unit Buffer, to 1 in the OPERATIONAL_UNIT_OPS_CMD register to clear the intermediate buffer on the module. Poll the OPERATIONAL_UNIT_OPS_STATUS register for completion of the Clear Operational Unit Buffer operation. If step 2 is performed and OPERATIONAL_UNIT_OPS_STATUS returns OPERATIONAL_UNIT_OPS_ERROR, the host shall check the OPERATIONAL_UNIT_FAIL_INFO register. The module shall provide failure information such as ENDURANCE_LIMIT_REACHED in OPERATIONAL_UNIT_FAIL_INFO indicating the module reached the endurance limit of the media or TYPED_BLOCK_DATA_SIZE_ERROR indicating the module received unsupported Typed Block Data Size from the host. The host shall repeat step 2 to perform a retry or use the value read from TYPED_BLOCK_DATA_SIZE0, TYPED_BLOCK_DATA_SIZE1, and TYPED_BLOCK_DATA_SIZE2 registers.
4. Set the OPERATIONAL_UNIT_ID0 and OPERATIONAL_UNIT_ID1 registers to the Operational Unit ID of the data to be written.
5. Set the BLOCK_ID to 0.
6. Calculate the Operational Unit CRC for the Operational Unit data to be written.
7. Write data to either the TYPED_BLOCK_DATA_BYTE0 to TYPED_BLOCK_DATA_BYTE31 registers or the TYPED_BLOCK_DATA_OFFSET register to send data to the module.
8. If the BLOCK_ID register is less than 8, increment the BLOCK_ID register and repeat step 7 until all the data in the Operational Unit is written.
9. Set Bit 3, Generate Operational Unit Checksum, to 1 in the OPERATIONAL_UNIT_OPS_CMD register to request the module to calculate the Operational Unit CRC value for the data received by the module. Poll the OPERATIONAL_UNIT_OPS_STATUS register for completion of the Generate Operational Unit Checksum operation.
10. To ensure there was no data error during the transfer of the Operational Unit data, verify that the CRC value calculated in step 6 matches the values in the OPERATIONAL_UNIT_CRC0 and OPERATIONAL_UNIT_CRC1 registers.
11. Set Bit 1, Set Operational Unit, to 1 in the OPERATIONAL_UNIT_OPS_CMD register to commit the Operational Unit data on the module. Poll the OPERATIONAL_UNIT_OPS_STATUS register for completion of the Set Operational Unit operation.

9.8.3 Security Protocol Data Workflow (cont'd)

12. If OPERATIONAL_UNIT_OPS_STATUS is checked in any above steps and the status is OPERATIONAL_UNIT_OPS_ERROR, the module shall provide failure information in the OPERATIONAL_UNIT_FAIL_INFO register. If the status is failed or the Operational Unit operation times out, the host may either abort the Security Protocol Data write or retry from step 1.
13. Increment the Operational Unit ID in the OPERATIONAL_UNIT_ID0 and OPERATIONAL_UNIT_ID1 registers.
14. Repeat Steps 3 to 13 until all desired data is sent.

The Security Protocol Data read is used to map the TCG IF- RECV function (see SIIS). To read the data, the host should do the following:

1. Set the OPERATIONAL_UNIT_ID0 and OPERATIONAL_UNIT_ID1 registers to 0.
2. Read the TYPED_BLOCK_DATA_SIZE0, TYPED_BLOCK_DATA_SIZE1, and TYPED_BLOCK_DATA_SIZE2, registers to determine the number of Operational Units to read for the Typed Block Data.
3. Set Bit 2, Clear Operational Unit Buffer, to 1 in the OPERATIONAL_UNIT_OPS_CMD register to clear the intermediate buffer on the module. Poll the OPERATIONAL_UNIT_OPS_STATUS register for completion of the Clear Operational Unit Buffer operation.
4. Set the OPERATIONAL_UNIT_ID0 and OPERATIONAL_UNIT_ID1 registers to the Operational Unit ID of the desired data.
5. Set the BLOCK_ID to 0.
6. Set Bit 0, Get Operational Unit, to 1 in the OPERATIONAL_UNIT_OPS_CMD register to retrieve the data into an intermediate buffer in the module. Poll the OPERATIONAL_UNIT_OPS_STATUS register for completion of the Get Operational Unit operation.
7. Read either the TYPED_BLOCK_DATA_BYTE0 to TYPED_BLOCK_DATA_BYTE31 registers or the TYPED_BLOCK_DATA_OFFSET register to read the data for the block from the module.
8. If the BLOCK_ID register is less than 8, increment the BLOCK_ID register and repeat step 7 until all the data from the Operational Unit is received.
9. Set Bit 3, Generate Operational Unit Checksum, to 1 in the OPERATIONAL_UNIT_OPS_CMD register to request the module to calculate the Operational Unit CRC value for the data retrieved. Poll the OPERATIONAL_UNIT_OPS_STATUS register for completion of the Generate Operational Unit Checksum operation.
10. To ensure there was no data error during transfer of the Operational Unit data, the host should calculate the CRC for Operational Unit data received from the module and verify that the CRC value matches the values in the OPERATIONAL_UNIT_CRC0 and OPERATIONAL_UNIT_CRC1 registers.
11. If the OPERATIONAL_UNIT_OPS_STATUS is checked in any above steps and the status is OPERATIONAL_UNIT_OPS_ERROR, the module should provide failure information in OPERATIONAL_UNIT_FAIL_INFO register. If the status is failed or Operational Unit operation times out, host may either abort the Security Protocol Data read or retry from step 1.
12. Repeat Steps 3 to 11 until all desired data is received.

Annex A - (Normative) SDRAM and RCD Requirements for NVRDIMM-N Modules

A.1 SDRAM and RCD Requirements During a Catastrophic Save Operation

During a Catastrophic Save operation (see 7.2.1), an NVRDIMM-N module:

- may program the SDRAM mode registers to any values;
- if the SDRAMs support DLL-off mode, then may switch the SDRAMs to DLL-off mode (see JESD79-4B (DDR4 SDRAM));
- may put the SDRAMs in Self-Refresh mode using the Self-Refresh Entry (SRE) command and take them out of Self-Refresh mode using the Self-Refresh Exit (SRX) command;
- should program MR2 bits 7:6 (Low Power Array Self Refresh) to 10'b (i.e., Manual Mode (Extended Operating Temperature Range));
- should send REFRESH commands at a tREFI of 3.9 µs (e.g., twice the normal rate) while the SDRAMs are not in Self-Refresh mode.

NOTE: Since the system might not be able to provide thermal control while running off the Energy Source, this ensures the SDRAMs are prepared for the temperature to rise to 95 °C.

- should set F0RC2x bit 0 (I^2C Bus Interface Disable) to 1 (i.e., disabled), disabling SMBus access to the RCD;
- if the NVRDIMM-N module does not use DA17, should set F0RC08 bit 3 (DA17 Input Buffer and QxA17 Outputs Disable) to 1 (i.e., disabled);
- if the NVRDIMM-N module does not use DCS1_n, should set F0RC09 bit 1 (DCS1_n Input Buffer and QxCS1 Outputs Disable) to 1 (i.e., disabled); and
- if the NVRDIMM-N module does not use DCKE1, should set F0RCBx bit 6 (DCKE1 Input Buffer and QxCKE1 Output Driver Disable) to 1 (i.e., disabled).

A.2 SDRAM and RCD Requirements Related To A Restore Operation

A.2.1 Overview

During a Restore operation (see 7.2.27.1), an NVRDIMM-N module shall comply with A.2.2.

Before a post-restore transition, an NVRDIMM-N module shall comply with A.2.3.

After a post-restore transition, a host supporting an NVRDIMM-N module shall comply with A.2.4.

A.2.2 Restore Operation Requirements

During a Restore operation, the NVRDIMM-N module:

- shall program the SDRAMs as specified in A.2.2.1; and
- shall program the RCD as specified in A.2.2.2.

A.2.2.1 SDRAMs During a Restore Operation

Upon entry into a Restore operation, the NVRDIMM-N module shall assert RESET_n to the SDRAMs.

NOTE: The SDRAMs could be in any state (e.g., they might be in Power Down mode, Self-Refresh mode, or Maximum Power Saving mode, and might have mode registers set to any values).

During a Restore operation, the NVRDIMM-N module:

- may program the SDRAM mode registers to any values;

A.2.2.1 SDRAMs During a Restore Operation (cont'd)

- may put the SDRAMs in Self-Refresh mode using the Self-Refresh Entry (SRE) command and take them out of Self-Refresh mode using the Self-Refresh Exit (SRX) command; and
- should program MR2 bits 7:6 (Low Power Array Self Refresh) to 10'b (i.e., Manual Mode (Extended Operating Temperature Range));
- should send REFRESH commands at a tREFI of 3.9 µs (e.g., twice the normal rate) while they are not in Self-Refresh mode.

The NVRDIMM-N module shall send a ZQCL command before sending any READ or WRITE commands to the SDRAMs.

As specified in JESD79-4B (DDR4 SDRAM), the NVRDIMM-N module shall use tZQinit (e.g., 1024 DCLKs) rather than tZQoper (e.g., 512 DCLKs) as the time after its first ZQCL.

NOTE: The voltage input to the SDRAMs might change between host operation and the Restore operation (e.g., power flows through different FET paths), so previous ZQ calibrations might no longer be correct. SDRAMs are not required to interpret any particular number of ZQCS commands as providing the same calibration as a ZQCL, so the NVRDIMM-N module is required to use ZQCL rather than ZQCS.

A.2.2.2 RCD During a Restore Operation

During a Restore operation, the NVRDIMM-N module:

- may program the RCD control words to any values;
- should set F0RC2x bit 0 (I^2C Bus Interface Disable) to 1 (i.e., disabled), disabling SMBus access to the RCD;
- if the NVRDIMM-N module does not use DA17, should set F0RC08 bit 3 (DA17 Input Buffer and QxA17 Outputs Disable) to 1 (i.e., disabled);
- if the NVRDIMM-N module does not use DCS1_n, should set F0RC09 bit 1 (DCS1_n Input Buffer and QxCS1 Outputs Disable) to 1 (i.e., disabled); and
- if the NVRDIMM-N module does not use DCKE1, should set F0RCBx bit 6 (DCKE1 Input Buffer and QxCKE1 Output Driver Disable) to 1 (i.e., disabled).

A.2.3 Post-restore Transition Entry Requirements

Before the post-restore transition, the NVRDIMM-N module:

- shall program the SDRAMs as specified in A.2.3.1; and
- shall program the RCD as specified in A.2.3.2.

A.2.3.1 SDRAMs for the Post-Restore Transition

The NVRDIMM-N module shall put the SDRAMs in Self-Refresh mode using the Self-Refresh Entry (SRE) command (see JESD79-4B (DDR4 SDRAM)) before the post-restore transition.

The NVRDIMM-N module shall program all fields defined as RFU (reserved for future use) to zero in all mode registers before the post-restore transition.

NOTE: Table 26, Table 27, Table 28, Table 29, Table 30, Table 31, and Table 32 include fields that might interfere with the host's ability to reprogram the SDRAM.

The NVRDIMM-N module shall program MR0 as specified in Table 26 before the post-restore transition.

A.2.3.1 SDRAMs for the Post-Restore Transition (cont'd)

Table 26 — MR0 Requirements During a Post-Restore Transition

Field	Requirement
WR and RTP (bits 13,11:9)	none
DLL Reset (bit 8)	none
TM (bit 7)	normal mode (i.e., 0)
CAS Latency (bits 12,6:4,2)	none
Read Burst Type (bit 3)	none
Burst Length (buts 1:0)	none

The NVRDIMM-N module shall program MR1 as specified in Table 27 before the post-restore transition.

Table 27 — MR1 Requirements During a Post-Restore Transition

Field	Requirement
Qoff (bit 12)	output buffer enabled (i.e., 0)
TDQS enable (bit 11)	none
RTT_NOM (bits 10:8)	none
Write Leveling Enable (bit 7)	disabled (i.e., 0)
Additive Latency (bits 4:3)	none
Output Driver Impedance Control (bits 2:1)	none
DLL Enable (bit 0)	none

The NVRDIMM-N module shall program MR2 as specified in Table 28 before the post-restore transition.

Table 28 — MR2 Requirements During a Post-Restore Transition

Field	Requirement	Notes
TRR (bit 13)	disabled (i.e., 0)	
Write CRC (bit 12)	none	
RTT_WR (bits 10:9)	none	
TRR Mode – BGn (bits 8,2)	none	
Low Power Array Self Refresh (bits 7:6)	Manual Mode (Extended Operating Temperature Range) (i.e., 10'b)	This increases the time after the post-restore transition before the host is required to send REFRESH commands
CAS Write Latency (bits 5:3)	none	
TRR Mode – BAn (bits 1:0)	none	

The NVRDIMM-N module shall program MR3 as specified in Table 29 before the post-restore transition.

A.2.3.1 SDRAMs for the Post-Restore Transition (cont'd)

Table 29 — MR3 Requirements During a Post-Restore Transition

Field	Requirement	Notes
MPR Read Format (bits 12:11)	none	
Write CMD Latency when CRC and DM are enabled (bits 10:9)	none	
Fine Granularity Refresh mode (bits 8:6)	normal (fixed 1x) (i.e., 000'b)	This ensures that the host is not obligated to send more frequent REFRESH commands after SRX and that the host need not understand if the NVRDIMM-N module ended on an even or odd number of them.
Temperature sensor readout (bit 5)	none	
Per DRAM addressability (bit 4)	disabled (i.e., 0)	
Geardown mode (bit 3)	½ Rate (i.e., 0)	
MPR operation (bit 2)	normal (i.e., 0)	
MPR page selection (bits 1:0)	none	Ignored since MPR is not being accessed

The NVRDIMM-N module shall program MR4 as specified in Table 30 before the post-restore transition.

Table 30 — MR4 Requirements During a Post-Restore Transition

Field	Requirement	Notes
Post Package Repair mode (bit 13)	disabled (i.e., 0)	
Write Preamble (bit 12)	none	
Read Preamble (bit 11)	none	
Read Preamble Training mode (bit 10)	disabled (i.e., 0)	
Self-Refresh Abort mode (bit 9)	disabled (i.e., 0)	
CS_n to CMD/ADDR Latency (CAL) mode (bits 8:6)	disabled (i.e., 000'b)	Otherwise, the host does not know the latency value to use to send an MRS command to turn off CAL mode.
Soft Post Package Repair mode (bit 5)	disabled (i.e., 0)	
Internal Vref Monitor (bit 4)	none	
Temperature Controlled Refresh mode (bit 3)	disabled (i.e., 0)	
Temperature Controlled Refresh Range (bit 2)	normal (i.e., 0)	
Maximum Power Saving mode (bit 1)	disabled (i.e., 0)	Maximum power saving mode does not preserve data.

The NVRDIMM-N module shall program MR5 as specified in Table 31 before the post-restore transition.

A.2.3.1 SDRAMs for the Post-Restore Transition (cont'd)

Table 31 — MR5 Requirements During a Post-Restore Transition

Field	Requirement	Notes
Read DBI (bit 12)	none	
Write DBI (bit 11)	none	
Data Mask (bit 10)	none	
Command Address Parity Persistent Error (bit 9)	none pending (i.e., 0)	
RTT_PARK (bits 8:6)	none	
ODT Input Buffer during Power Down mode (bit 5)	none	
Command Address Parity Error Status (bit 4)	no parity errors pending that need to be cleared	
CRC Error Status (bit 3)	no CRC errors pending that need to be cleared	
Command/Address Parity Latency mode (bits 2:0)	disabled (i.e., 000'b)	Along with the requirement for F0RC0E (see table 8), this ensures that the host is not obligated to send correct parity and be prepared to handle parity errors during the initial MRS commands after the handover. The host should enable command/address parity later.

The NVRDIMM-N module shall program MR6 as specified in Table 32 before the post-restore transition.

Table 32 — MR6 Requirements During a Post-Restore Transition

Field	Requirement	Notes
tCCD_L and tDLLK (bits 12:10)	none	
VrefDQ Training Enable (bit 7)	disabled (i.e., 0)	
VrefDQ Training Range (bit 6)	none	Ignored since VrefDQ training is not enabled
VrefDQ Training Value (bits 5:0)	none	Ignored since VrefDQ training is not enabled

A.2.3.2 RCD for a Post-Restore Transition

The NVRDIMM-N module may leave the RCD control words set to any values for a post-restore transition except as specified in this subclause.

The NVRDIMM-N module shall program the RCD as specified in Table 33 before a post-restore transition.

A.2.3.2 RCD for a Post-restore Transition (cont'd)

Table 33 — RCD Requirements During a Post-Restore Transition

Field	Requirement	Notes
F0RC00 bit 0 (Output Inversion Disable)	0 (i.e., enabled)	
F0RC02 bit 0 (DA17 Input Bus Termination Disable)	0 (i.e., enabled)	
F0RC02 bit 1 (DPAR Input Bus Termination Disable)	0 (i.e., enabled)	
F0RC02 bit 2 (Transparent Mode)	0 (i.e., disabled)	
F0RC02 bit 3 (Frequency Band Select)	0 (i.e., operation (Frequency Band 1))	
F0RC09 bit 2 (i.e., CKE Power Down Mode)	0 (i.e., CKE power down with IBT ON)	
F0RC09 bit 3 (i.e., CKE Power Down Mode Enable)	1 (i.e., Enabled)	While in CKE Power Down mode, the RCD ignores all inputs except CK_t/CK_c, DCKEn, DODTn, ERROR_IN_n, and DRST_n and disables all outputs except Yn_t/Yn_c (which still carries the clock), QxODTn, and QxCKEn (driven LOW). The RCD exits this mode when it detects DCS[n:0]_n asserted with DCKEn HIGH.
F0RC0C bits 2:0 (Training mode selection)	000'b (i.e., normal operating mode)	
F0RC0D bit 2 (DIMM Type)	1 (i.e., RDIMM)	
F0RC0E bit 0 (Parity Enable)	0 (i.e., disabled)	Along with the requirement for MR5 (see table 6), this ensures that the host is not obligated to send correct parity and be prepared to handle parity errors during the initial MRS commands after the handover. The host should enable command/address parity later.
F0RC2x bit 0 (I^2C Bus Interface Disable)	0 (i.e., enabled)	Allow the host to restore certain RCD registers via SMBus before asserting CKE
NOTE Table 33 includes fields that might interfere with the host's ability to reprogram the RCD.		

A.2.3.2 RCD for a Post-restore Transition (cont'd)

For the post-restore transition, the NVRDIMM-N module shall put the RCD in Clock Stopped Power Down mode (see JESD82-31A (DDR4RCD02)) by:

- driving both CK_t and CK_c to LOW.

NOTE: While in Clock Stopped Power Down mode, the RCD stops its PLL, ignores all inputs except CK_t/CK_c, and disables all outputs except QxCKEn. The RCD exits this mode when it detects an input clock signal on CK_t/CK_c. The SDRAMs are required to be in Self-Refresh mode as well. Using this mode prevents problems from runt clock pulses during the transition.

A.2.4 Post-Restore Transition Exit Requirements

After a post-restore transition, the host:

1. shall reprogram the RCD as specified in A.2.4.1; and
2. shall reprogram the DRAMs as specified in A.2.4.2.

A.2.4.1 Host Reprogramming the RCD after a Post-restore Transition

The host should only reprogram the RCD control words one time after a post-restore transition.

NOTE: Although an NVRDIMM-N has SDRAMs on multiple sides and might have multiple ranks in the SDRAMs, there is only one RCD per NVRDIMM-N.

To regain control of the NVRDIMM-N, the host:

1. shall run the clock signal at host frequency and wait tSTAB (see JESD82-31A (DDR4RCD02)). This wakes the RCD from Clock Stopped Power Down mode;
2. shall restore F0RC0A (DIMM Operating Speed) and F0RC3x (Fine Granularity RDIMM Operating Speed) to the host clock frequency and wait tSTAB;
3. shall assert CKE0 to both:
    - wake the RCD from CKE Power Down mode (see JESD82-31A (DDR4RCD02)); and
    - cause the SDRAMs to exit Self-Refresh mode (see JESD79-4B (DDR4 SDRAM)).

The host should not assert DCSn_n, CKEn and ODTn signals that are not supported by the NVRDIMM-N module.

After asserting CKE0, the host should reprogram RCD control words as follows:

- function 0 as specified in
- Table 34;
- function 1 as specified in Table 35; and
- function 4 as specified in Table 36.

The host should not reprogram RCD control words for any other functions.

A.2.4.1 Host Reprogramming the RCD after a Post-restore Transition (cont'd)

Table 34 — Host Reprogramming RCD Function 0 Control Words After a Post-Restore Transition

Control Word	Name	Reprogram?	Notes
F0RC00	Global Features	DDR4	
F0RC01	Clock Driver Enable	DDR4	
F0RC02	Timing and IBT	DDR4	
F0RC03	CA and CS Driver Characteristics	DDR4	
F0RC04	ODT and CKE Driver Characteristics	DDR4	
F0RC05	Clock Driver Characteristics	DDR4	
F0RC06	Command Space	no	Used to issue self-clearing commands
F0RC07	LCOM Interface Command Space	no	Used to access DB in LRDIMMs (see JESD82-32A (DDR4DB02)) and to access optional NVDIMM mode function 4
F0RC08	I/O Configuration	DDR4	
F0RC09	Power Savings Settings	DDR4	
F0RC0A	RDIMM Operating Speed	SMBus	Program the DDR4 interface rate before enabling CKE.
F0RC0B	Vdd and Vref Source	SMBus	Due to the SMBus address map, writing to F0RC0A also requires writing to F0RC0B.
F0RC0C	Training	DDR4	
F0RC0D	DIMM Configuration	DDR4	
F0RC0E	Parity	DDR4	
F0RC0F	Command Latency Adder	DDR4	

Table 34 — Host Reprogramming RCD Function 0 Control Words After a Post-Restore Transition (cont'd)

Control Word	Name	Reprogram?	Notes
F0RC1x	Internal Vref	DDR4	
F0RC2x	I^2C Bus	DDR4	The host should disable I^2C bus access after the host is done issuing I^2C Bus cycles (e.g., after reprogramming F0RC0A and F0RC3x).
F0RC3x	Fine Granularity RDIMM Operating Speed	SMBus	Program the DDR4 interface rate before enabling CKE.
F0RC4x	CW Source Selection	no	Used to access other functions
F0RC5x	CW Destination Selection	no	Used to access other functions
F0RC6x	CW Data	no	Used to access other functions
F0RC7x	Input Buffer Termination	DDR4	
F0RC8x	ODT	DDR4	
F0RC9x	QxODT Write Pattern	DDR4	
F0RCAx	QxODT Read Pattern	DDR4	
F0RCBx	IBT and MRS Snoop	DDR4	
F0RCCx	Error Log Register	no	
F0RCDx	Error Log Register	no	
F0RCEx	Error Log Register	no	
F0RCFx	Error Log Register	no	

If the NVRDIMM-N module does not use DA17, then the host should set F0RC08 bit 3 (DA17 Input Buffer and QxA17 Outputs Disable) to 1 (i.e., disabled).

If the host does not use DA17 for the NVRDIMM-N module, then the host should set F0RC02 bit 0 (DA17 Input Bus Termination Disable) to 1 (i.e., disabled).

If the NVRDIMM-N module does not use DCS1_n, then the host should set F0RC09 bit 1 (DCS1_n Input Buffer and QxCS1 Outputs Disable) to 1 (i.e., disabled).

If the host does not use DCS1_n for the NVRDIMM-N module, then the host should set F0RC09 bit 0 (DCS1_n Input Bus Termination Disable) to 1 (i.e., disabled).

If the NVRDIMM-N module does not use DCKE1, then the host should set F0RCBx bit 6 (DCKE1 Input Buffer and QxCKE1 Output Driver Disable) to 1 (i.e., disabled).

If the host does not use DCKE1 for the NVRDIMM-N module, then the host should set F0RCBx bit 5 (DCKE1 Input Bus Termination Disable) to 1 (i.e., disabled).

A.2.4.1 Host Reprogramming the RCD after a Post-restore Transition (cont'd)

Table 35 — Host Reprogramming RCD Function 1 Control Words After a Post-Restore Transition

Control Word	Name	Reprogram Over DDR4?
F1RC00	Data Buffer Interface Driver Characteristics	yes
F1RC01	CAL Mode Snoop Enable	yes
F1RC02	CA and CS Output Slew Rate Control	yes
F1RC03	ODT and CKEn Output Slew Rate Control	yes
F1RC04	Clock Driver Output Slew Rate Control	yes
F1RC05	QxCSn_n Output Delay Control Word	yes
F1RC06 – F1RC0F	RFU	no
F1RC1x	QxCSn_n Output Delay Control Word	yes
F1RC2x	QxCn Output Delay Control Word	yes
F1RC3x	QxCKEn Output Delay Control Word	yes
F1RC4x	QxODTn Output Delay Control Word	yes
F1RC5x	QxCA Output Delay Control Word	yes
F1RC6x	Y1/Y3 Output Delay Control Word	yes
F1RC7x	Y0/Y2 Output Delay Control Word	yes
F1RC8x	BCOM[3:0]/BCKE/BODT Output Delay Control Word	yes
F1RC9x	BCK Output Delay Control Word	yes
F1RCAx – F1RCFx	RFU	no

Table 36 — Host Reprogramming RCD Function 4 Control Words After a Post-Restore Transition

Control Word	Name	Reprogram Over DDR4?	Notes
F4RC00	NVDIMM Mode Enable Control Word	yes	
F4RC01	NVDIMM Asynchronous Interrupt Control Word	yes	
F4RC02	Read Frame Gear Ratio and Yn/BCK Frequency Ratio Control Word	yes	
F4RC03	LCOM Receiver Termination Control Word	yes	
F4RC04	LCOM Receiver Vref Control Word	yes	
F4RC05 – F4RC0F	RFU	no	
F4RC1x – F4RC4x	Multi-Purpose Register	no	Used for commands
F4RC5x	Data Transfer (Copy and Restore) Control Word	no	
F4RC6x	Self-Refresh Status Word	no	only currently defined bits are read-only
F4RC7x –F4RCFx	RFU	no	

A.2.4.2 Host Reprogramming the SDRAMs after a Post-restore Transition

The host should reprogram the SDRAM mode registers using the DDR4 DLL off-to-on sequence defined in JESD79-4B (DDR4 SDRAM). This includes sending the following commands:

1. Self-^{oRefresh} Exit (SRX) command;

A.2.4.2 Host Reprogramming the SDRAMs after a Post-restore Transition (cont'd)

2. MRS command to MR1 with bit 0 set (i.e., enable DLL) and all other MR1 fields set to the appropriate values;
3. MRS command to MR0 with bit 8 set (i.e., DLL reset) and all other MR0 fields set to the appropriate values;
4. MRS command to MR3;
5. MRS command to MR6 to set VrefDQ to "safe values." This includes three steps:
    a. MRS command to MR6 to enable VrefDQ training;
    b. MRS command to MR6 to program the VrefDQ value; and
    c. MRS command to MR6 to exit VrefDQ training;
6. MRS commands to MR5, MR4, and MR2;
7. ZQCL command; and
8. MRS commands using per-DRAM addressing (PDA) to update any mode registers that were trained with PDA values (e.g., MR6 with PDA VrefDQ values).

After asserting CKEn (i.e., sending the SRX command), the host should send REFRESH commands as required by DDR4. The host should not wait for the DLL off-to-on sequence to complete before starting REFRESH commands, because the SDRAMs contain valid data. If the host asserts CKEn to multiple NVRDIMM-Ns in a channel at the same time, then it is required to start sending REFRESH commands to all of them at that time; the host should not perform the DLL off-to-on sequence one at a time in that case. If the system includes multiple DDR4 channels and asserts CKEn on multiple channels at the same time, then it should not wait for the sequence on one channel to complete before starting REFRESH commands on other channels.

The host should send MRS commands to SDRAMs on both side A and side B, as defined in the JESD21C Page 4.20.28 (DDR4 RDIMM) and JESD82-31A (DDR4RCD02).

The host should send per-DRAM addressing (PDA) MRS commands to both side A and side B if it does not know which side a particular SDRAM is on, and should send them to only the appropriate side if it does know which side a particular SDRAM is on.

As specified in JESD79-4B (DDR4 SDRAM), the host should not send REFRESH, ZQCL, or ZQCS commands during VrefDQ training.

As specified in JESD79-4B, the host should not send REFRESH, ZQCL, or ZQCS commands during PDA MRS sequences.

The host should send a ZQCL after reprogramming DRAM mode registers and before performing PDA MRS commands. As specified in JESD79-4B, the host should periodically send ZQCS commands thereafter.

The host should use tZQinit (e.g., 1024 DCLKs) rather than tZQoper (e.g., 512 DCLKs) as the time after its first ZQCL, since it does not know if the NVRDIMM-N module has already sent the first ZQCL after a RESET_n.

NOTE: The voltage input to the SDRAMs might change between the Restore operation and host operation (e.g., power flows through different FET paths), so previous ZQ calibrations might no longer be correct. SDRAMs are not required to interpret any particular number of ZQCS commands as providing the same calibration as a ZQCL, so the host is required to use ZQCL rather than ZQCS.

Annex B (Normative) Self-encrypting NVDIMM-N

B.1 Overview

An NVDIMM-N may be a self-encrypting device (SED) that protects data at rest. This means the NVDIMM-N controller:

- encrypts data during a Catastrophic Save operation; and
- decrypts data during a Restore operation

and the data is:

- plaintext while sitting in SDRAM; and
- ciphertext while sitting in NVM (e.g., flash memory).

Figure 12 shows some of the cryptographic modules in a self-encrypting NVDIMM-N.

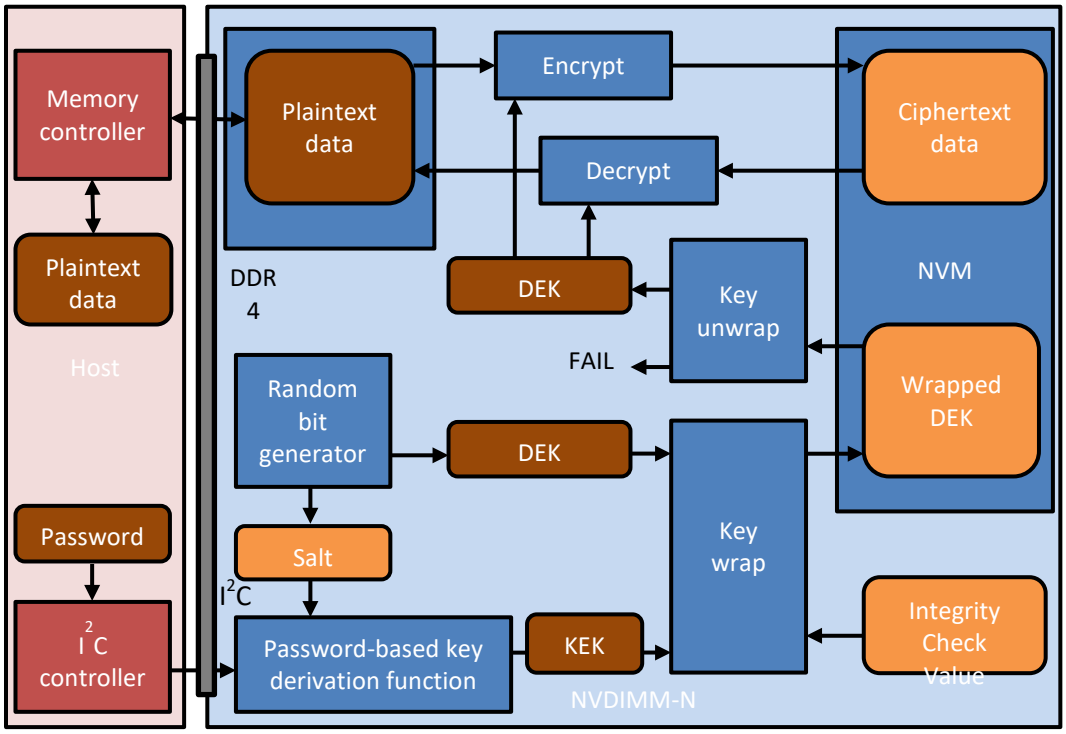

Figure 12 — Some Cryptographic Components in a Self-encrypting NVDIMM-N

The NVDIMM-N uses these engines:

- encrypt and decrypt (see B.3)
  - encrypt/decrypt data using an Advanced Encryption Standard (AES) encryption engine
  - XEX encryption mode with Tweak and ciphertext Stealing (XTS)
    - a block cipher mode appropriate for the usage model
- random bit generator (RBG) (see B.4)
  - hardware entropy source seeding a NIST-approved deterministic random bit generator algorithm
  - generates data encryption keys
  - generates salt for PBKDF

B.1 Overview (cont'd)

- password-based key derivation function (PBKDF) (see B.5)
  - adds salt from RBG to strengthen the externally provided password
  - includes an integrity check value to ensure the password was correct
- key wrap and unwrap (see B.6)
  - wrap/unwrap the DEK using the key derived from the password

The module uses these cryptographic values:

- DEK (data encryption key) (see B.7.1)
  - 256-bit AES key + 256-bit XTS key
  - generated internally by RBG
  - stored in NVM wrapped with the password
    - not accessible via debug ports, scan chains, etc.
  - created and wrapped by each new Arm operation
  - used by the Catastrophic Save operation to encrypt data written to NVM
  - unwrapped and used by Restore operations to decrypt data read from NVM
- salt (see B.7.2)
  - 16-byte random value generated internally by the RBG
  - used to strengthen the password
- integrity check value (see B.7.3)
- password (see B.7.4)
  - defined by the TCG storage protocol (e.g., a 32-byte value)
  - generated externally
  - received via the I^2C management interface over the TCG storage protocol
  - only held internally in volatile registers while running PBKDF
- KEK (key encryption key) (see B.7.5)
- wrapped DEK (see B.7.6)
  - DEK wrapped with key derived from the password and salt

Other aspects of the NVDIMM-N cryptographic features:

- authentication (see B.8)
  - Trusted Computing Group (TCG) Storage management protocol
  - Ruby SSC customized for an NVDIMM-N
- labeling (see B.9)
  - physical owner security ID (PSID) on printed label
  - avoids turning NVDIMM-N into a brick by forgetting the password
- sanitization (see B.10)
  - "instant secure erase" cryptographic erase for data sanitization
  - purge level of data sanitization
  - implemented by overwriting/erasing the wrapped DEK
- signed firmware (see B.11)
  - SHA-256 hash of the unsigned image
  - RSA-2048 encryption of the hash
  - RSASSA-PKCS1-v1.5 digital signature format
- optional FIPS 140-2 compliance (see B.12)
- optional Common Criteria compliance (see B.13)

B.1 Overview (cont'd)

The module does not encrypt the following according to this standard:

- SPD data (see 6.1)
- firmware image data (see 7.6)
- vendor log page (see 7.12)
- label data (see 7.15)

B.2 Requirements

The module shall maintain the plaintext data encryption key inside the module inaccessible by all entities except the encryption engines. The module shall not allow the data encryption key to be retrievable via any means, including debug ports, scan chains, or reads by an embedded CPU. The module shall exclude flip-flops holding the data encryption key from JTAG-accessible scan chains.

NVDIMM-N operations affected by self-encryption include:

- Arm operation
  - Set an access password
  - Create a data encryption key kept internal to the module
- Catastrophic Save operation
  - Encrypt while writing to NVM
- Power loss
- Restore operation
  - Unlock with access password
  - Decrypt while reading from NVM
- Factory Default operation
- TCG Authenticate method to the Locking SP

B.3 Data Encryption and Decryption Engines

B.3.1 Encryption Algorithm

The module shall include an Advanced Encryption Standard (AES) encryption engine as defined in FIPS 197 and tested according to the *Advanced Encryption Standard Algorithm Validation System (AESAVS)*.

The module shall use an AES encryption key size of 256 bits (i.e., 32 bytes).

NOTE: AES has a block size of 128 bits (i.e., 16 bytes).

The module should protect against non-invasive simple and differential side-channel analysis (e.g., see Akkar and Ordu papers) for:

- timing: exploiting timing information
- power: exploiting dynamic power consumption (e.g., additive data masking, cycle hiding)
- electromagnetic: capturing electromagnetic radiation from the chip

The module may protect against non-invasive fault injection attacks and electromagnetic pulse injection (e.g., see Dehbaoui).

The module need not protect against invasive analysis methods.

B.3.2 Block Cipher Mode

Figure 13 shows XTS-AES encryption.

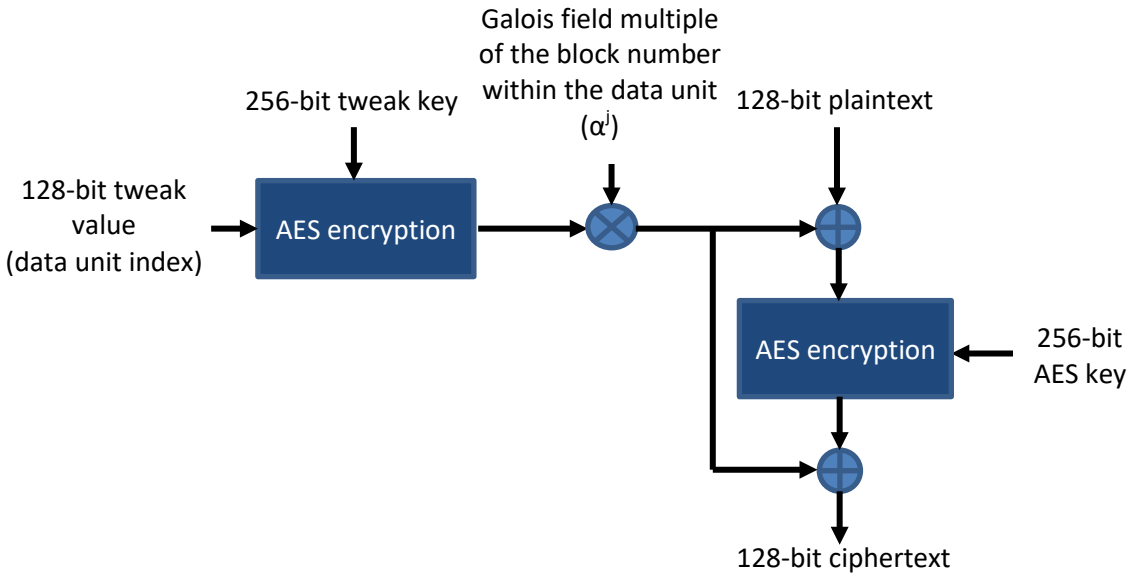

Figure 13 — XTS-AES Encryption

Figure 14 shows XTS-AES decryption.

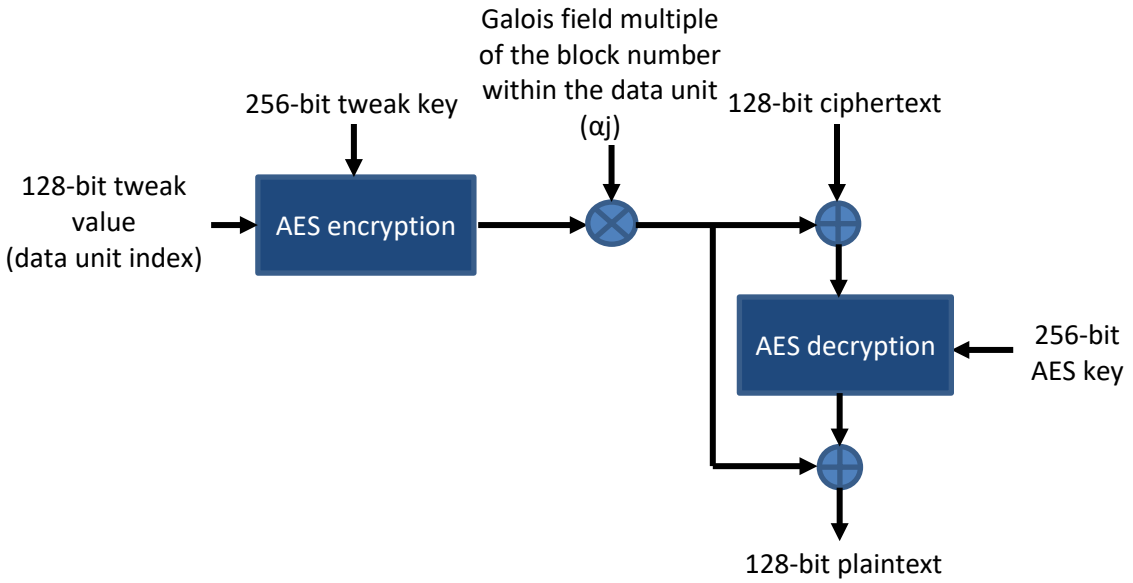

Figure 14 — XTS-AES Decryption

The module shall implement the XTS block cipher mode with AES (i.e., XTS-AES-256) as defined in IEEE 1619 and SP800-38E and tested according to *The XTS-AES Validation System (XTSVS)*.

B.3.2 Block Cipher Mode (cont'd)

The XTS-AES key shall be 512 bits (i.e., 64 bytes), meaning that:

- the AES key (i.e., Key_1) shall be 256 bits (i.e., 32 bytes); and
- the XTS tweak key (i.e., Key_2) shall be 256 bits (i.e., 32 bytes).

The data unit size:

- shall be at least 128 bits (i.e., 16 bytes) and no larger than 16 MiB; and
- should be a multiple of 128 bits to avoid using ciphertext stealing.

The module shall perform known answer tests during power-up as defined in FIPS 140-2.

B.4 RBG (Random Bit Generator) Engine

B.4.1 RBG Overview

Figure 15 shows the architecture of a NIST-approved random bit generator (RBG).

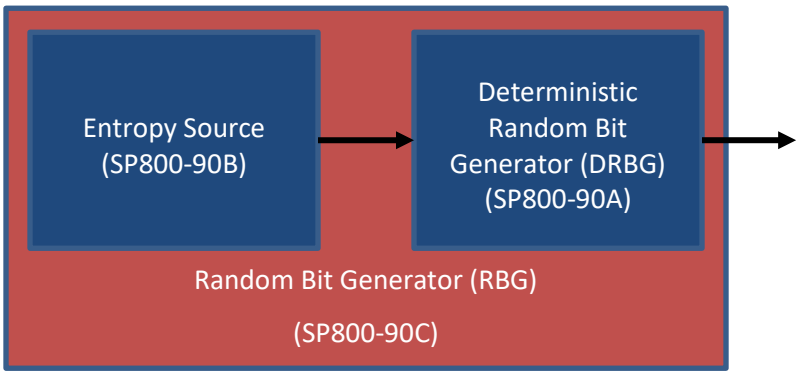

Figure 15 — Random Bit Generator (RBG)

The module shall include a random bit generator engine that uses an entropy source (see B.4.2) for the DRBG (see B.4.2) as defined in SP800-90C and tested as described in SP800-90C.

The RBG is used for generating DEKs (see B.7.1) and salt (see B.7.2).

B.4.2 Entropy Source

The module shall include an entropy source (i.e., a true hardware random number generator) as defined in SP800-90B and tested as described in SP800-90B. The module shall include startup, continuous, and on-demand health tests (e.g., Repetition Count and Adaptive Proportion tests) to detect if the entropy source is not working properly (e.g., subject to a hardware attack). The module should include conditioning.

B.4.3 DBRG (Deterministic Random Bit Generator)

The module shall include a deterministic random bit generator (DRBG) as defined in SP800-90A Revision 1 and tested according to SP800-90A and *The NIST SP800-90A Deterministic Random Bit Generator Validation System (DRBGVS)*. The module shall perform known answer tests during power-up and continuous random number generator tests during runtime as defined in FIPS 140-2, and known answer tests for manufacturing. The module shall not include DRBG logic in scan chains, instead relying upon known answer tests.

B.4.3 DBRG (Deterministic Random Bit Generator) (cont'd)

The module shall implement a DRBG with a security strength of 256 bits. The DBRG mechanisms providing a security strength of 256 bits include:

- Hash_DRBG:
  - allowed: SHA-256, SHA-512/256, SHA-384, and SHA-512
  - not allowed: SHA-1, SHA-224, SHA-512/224
  - if chosen, the hash engine shall be tested according to the *Secure Hash Standard Validation System (SHAVS)*
- HMAC_DRBG:
  - allowed: SHA-256, SHA-512/256, SHA-384, and SHA-512
  - not allowed: SHA-1, SHA-224, SHA-512/224
  - if chosen, the HMAC engine shall be tested according to the *Keyed-Hash Message Authentication Code Validation System (HMACVS)*
- CTR_DRBG:
  - allowed: AES-256
  - not allowed: 3 Key TDEA, AES-128, AES-192
  - if chosen, the encryption engine shall be tested according to the *Advanced Encryption Standard Algorithm Validation System (AESAVS)*

The module may reuse a hardware SHA-256 hash engine for signed firmware for Hash_DRBG. The module may reuse a hardware AES encryption engine for data encryption for CTR_DRBG.

B.5 PBKDF (Password-based Key Derivation Function) Engine

The module shall implement a password-based key derivation function compliant with SP800-132 (i.e.., the PBKDF2 algorithm) using one of the option 2 choices:

- the module should use option 2a (i.e., output an MK generated from the password)
- the module may use option 2b (i.e., output derived keying material generated from the password run through a KDF as defined in SP800-108 and tested according to KDKDFVS)

Figure 17 shows the PBKDF engine.

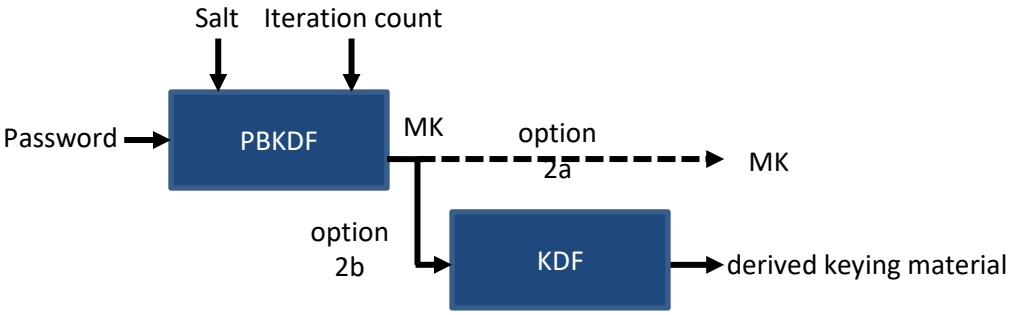

Figure 16 — PBKDF Engine

B.5 PBKDF (Password-based Key Derivation Function) Engine (cont'd)

Table 37 defines how the PBKDF engine implements the SP800-132 parameters, inputs, and outputs.

Table 37 — PBKDF Engine Parameters, Inputs, and Outputs

SP800-132			NVDIMM-N
Parameter	Name	Type	
PRF	HMAC with an approved hash function	parameter	HMAC using SHA-256, SHA-512/256, SHA-384, or SHA-512 (i.e., not using SHA-1, SHA-224, SHA-512/224)
hlen	digest size of the hash function	parameter	
P	password	input	user password from the TCG storage subsystem password length is 256 bits (i.e., 32 bytes)
S	salt	input	Salt (see B.7.2) 128 bits (i.e., 16 bytes)
C	iteration count	input	at least 10,000
kLen	length of MK in bits	input	
mk	main key	output	ICV **NOTE**: external documents use the master terminology

B.6 Key Wrapping And Unwrapping Engines

The module shall implement a key wrap engine and a key unwrap engine compliant with the AES Key Wrap mode (KW) defined in SP800-38F and tested according to KWVS.

Figure 17 shows the key wrap engine.

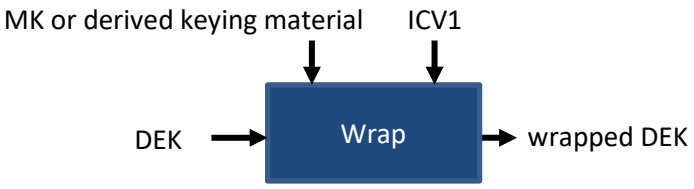

Figure 17 — Key Wrap Engine

Table 38 defines how the key wrap engine implements the SP800-38F parameters, inputs, and outputs. The key wrap engine uses the KW-AE(P) authenticated-encryption function.

B.6 Key Wrapping And Unwrapping Engines (cont'd)

Table 38 — Key Wrap Engine Prerequisites, Inputs, and Outputs

SP800-38F			NVDIMM-N Implementation
Parameter	Name	Type	
$CIPH_K$	designated cipher function	prerequisite	AES-128 engine
K	key encryption key	prerequisite	the MK or the derived keying material (i.e., the output of the PBKDF engine)
ICV1	integrity check value	fixed value	as defined in SP800-38F
P	plaintext	input	DEK (see B.7.1)
C	ciphertext	output	wrapped DEK (see B.7.6)

Figure 18 shows the key unwrap engine.

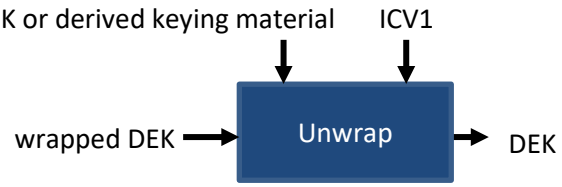

Figure 18 — Key Unwrap Engine

Table 39 defines how the key unwrap engine implements the SP800-38F parameters, inputs, and outputs. The key wrap engine uses the KW-AD(C) authenticated-decryption function.

Table 39 — Key Unwrap Engine Prerequisites, Inputs, and Outputs

SP800-38F			NVDIMM-N Implementation
Parameter	Name	Type	
$CIPH^{-1}_K$	designated cipher function	prerequisite	AES-128 engine
K	key encryption key	prerequisite	the MK or the derived keying material (i.e., the output of the PBKDF engine)
ICV1	integrity check value	fixed value	as defined in SP800-38F
C	purported ciphertext	input	wrapped DEK (see B.7.6)
P	plaintext	output	DEK (see B.7.1)

B.7 Cryptographic Values

B.7.1 DEK (Data Encryption Key)

The module shall generate data encryption keys (DEKs) (i.e., data protection keys (DPKs)) using the RBG as defined in B.4.

While processing an Arm operation, the module shall:

- create a new DEK.

B.7.1 DEK (Data Encryption Key) (cont'd)

While processing a Catastrophic Save operation, the module shall:

- use the DEK.

The DEK is only used one time; this prevents "reply attacks" replacing certain cipherblocks with older versions in the same position.

The module shall not expose the DEK through any interface (e.g., only hold the DEK in registers inside a hardware encryption core feeding that encryption logic). The module shall not make the DEK accessible via the programming interface, debug ports, or scan chains. The module should include a BIST mode that loads known key values to test for stuck-at faults.

B.7.2 Salt

The module shall generate salt values using the RBG engine as defined in B.4.

The PBKDF engine (see B.5) uses salt to generate a KEK (see B.7.4).

B.7.3 ICV (Integrity Check Value)

The module uses ICVs for PBKDF (see B.5) and key wrap and key unwrap (see B.6).

B.7.4 Password

The module shall obtain user passwords from the TCG storage security subsystem (see B.8.2).

The key wrap engine (see B.6) uses a user password to create a wrapped DEK (see B.7.6).

The module shall not expose any credential through any interface (e.g., only hold the password in registers inside a PBKDF hardware core). The module shall not make a password accessible via the programming interface, debug ports, or scan chains.

The module shall unwrap the wrapped DEK and use the Integrity Check Value to determine if the password is correct.

B.7.5 KEK (Key Encryption Key)

The module shall generate key encryption keys (KEKs) by running the password (see B.7.4) through a PBKDF and KDF, if any, as defined in B.5.

NOTE: In SP800-132, DEKs are called data protection keys (DPKs) and KEKs are called master keys (MKs).

The KEK is a 256-bit (i.e., 32-byte) value.

B.7.6 Wrapped DEK

The module shall generate wrapped DEKs by wrapping a DEK (see B.7.1) in a key encryption key (KEK) (see B.7.4) using a key wrap engine as defined in B.6.

While processing an Arm operation, the module shall:

- generate a wrapped DEK; and
- store the wrapped DEK inside the NVDIMM-N (e.g., in the NVM).

The module shall unwrap a wrapped DEK into a DEK using a key unwrap engine as defined in B.6.

B.7.6 Wrapped DEK (cont'd)

While processing a TCG Unlock method, the module shall:

- read the wrapped DEK from inside the NVDIMM-N;
- unwrap the wrapped DEK; and
- retain the DEK in a volatile location.

The module shall not expose the wrapped DEK through any interface.

B.8 Authentication

B.8.1 TCG Storage Protocol

The Trusted Computing Group defines a byte-coded interpreted language for managing security functionality in a device, communicated over protocol-specific interfaces (e.g., SECURITY PROTOCOL IN/OUT commands in SCSI, SECURITY SEND/RECEIVE commands in NVMe).

B.8.2 TCG Storage Ruby SSC

The module shall implement a security subsystem compliant with the TCG Storage *Ruby SSC* with modifications for NVDIMM-Ns as described in this subclause or described in the Ruby SSC.

NOTE: Some of the main requirements in Ruby SSC are:

- password authorities and authentication
- 32-byte passwords
- Admin SP
    - Templates: Base, Admin, Crypto
    - Methods: Next, GetACL, Get, Set, Authenticate, Revert, Activate, Random
    - Activate = managing the life cycle of SPs created in manufacturing
    - Revert = managing the life cycle of SPs created in manufacturing
- Locking SP
    - Templates: Base, Crypto, Locking
    - Methods: Next, GetACL, GenKey, RevertSP, Get, Set, Authenticate, Random
- PSID (Physical Presence Security Identifier (SID))
    - this ensures the physical owner of the device can always use it; no "bricking" problems
- Block Security Identifier (SID) Authentication
- optional PSK Secure Messaging

The module shall implement the following feature sets specified as optional in Ruby SSC:

- Single User Mode
- The Locking SP shall be created by the module manufacturer

The module shall implement the following choices chosen from those allowed in Ruby SSC:

- AES-256, not AES-128
- XTS block cipher mode, not any block cipher mode

The module shall not implement the following feature sets specified as optional in Ruby SSC:

- Configurable Namespace Locking
- MBR Shadowing
- returning the Geometry Reporting Feature Descriptor

B.8.2 TCG Storage Ruby SSC (cont'd)

The module shall implement the following features differently than specified in Ruby SSC:

- 1 KiB DataStore Table rather than the 128 KiB DataStore Table

NOTE: The Ruby SSC defines several credentials:

- Manufacturer (C_PIN_MSID)
- Physical Driver Owner PIN (C_PIN_PSID)
- Trusted Peripheral Owner Security ID (C_PIN_SID)
- Administrator (C_PIN_Admin1 .. AdminNN)
- Users (C_PIN_User1 .. UserNN)

B.8.3 Using Passwords

The system is responsible for selecting good passwords. Since the NVDIMM-N only accepts 32-byte passwords (which is all that Ruby SSC supports), but cPP_FDE_AA requires that full drive encryption authorization acquisition support 64-byte passwords, system software should run a user-provided ASCII string through PBKDF2 to mix it with a random salt value to generate a 32-byte password for use as the device password.

Software tools that accept password strings should use the same PBKDF2 algorithm:

- HMAC-SHA-256
- input passphrase
    - 64 bytes
    - ASCII printable characters (0x20 to 0x7e)
- input salt
    - 16 bytes (i.e., 128 bits) generated with a RBG compliant with SP800-90C
    - unique for each password
    - not a secret
    - saved with output password
- input number of iterations
    - support at least 1000 iterations
    - not a secret
    - saved with output password
- output password
    - 32 bytes (i.e., 256 bits)
    - secret
    - saved with output password

Software should save the salt value (which is not secret) and number of iterations (which is not secret) alongside that 32-byte password, so tools that accept user-provided ASCII strings work the same.

B.8.4 Roles

In FIPS 140-2 terms, the roles are:

- FIPS Crypto Officer:
- Drive owner: SID authority for the Admin SP; download FW images, enter/exit Ruby mode
- Administrator: manage users, create LBA ranges, crypto erase LBA ranges, set LBA range attributes
- FIPS User: unlock/lock drive, invoke crypto erase
- Unauthenticated: use PSID to restore to factory defaults

B.8.5 BAEBI Operations

Table 40 describes how various BAEBI operations are impacted by security.

Table 40 — Authenticated and Unauthenticated BAEBI Operations

Operation	Operator Access Control	Behavior if Locked?	Security Behavior if Unlocked
Catastrophic Save via register	Owner, Admin, User	set CSAVE_ERROR and CSAVE_REJECT in the CSAVE_STATUS register (see 8.1.5.4) and SECURITY_ERROR in the CSAVE_FAIL_INFO1 register (see 8.1.6.3)	Encrypt using DEK
Catastrophic Save via trigger	Admin, User	Not possible – arming is rejected	Encrypt using DEK
Restore	Admin, User	set RESTORE_ERROR in the RESTORE_STATUS register (see 8.1.5.5) and SECURITY_ERROR in the RESTORE_FAIL_INFO register (see 8.1.5.17)	Decrypt using unwrapped DEK
Arm	Admin, User	set ARM_ERROR in the ARM_STATUS register (see 8.1.5.7) and SECURITY_ERROR in the ARM_FAIL_INFO register (see 8.1.5.16)	Delete the wrapped DEK (providing instant secure erase) and generate a new DEK.
Erase	Admin, User	set ERASE_ERROR in the ERASE_STATUS register (see 8.1.5.6) and SECURITY_ERROR in the ERASE_FAIL_INFO register (see 8.1.5.13)	Delete the wrapped DEK (providing instant secure erase)
Reset Controller	none	allowed	none
Clear <various>	none	allowed	none
Deassert EVENT_n	none	allowed	none
Set Energy Source Policy	none	allowed	none
Set Event Notification	none	allowed	none

Table 40 — Authenticated and Unauthenticated BAEBI Operations (cont'd)

Operation	Operator Access Control	Behavior if Locked?	Security Behavior if Unlocked
Factory Default	Owner, Admin, User	set FACTORY_DEFAULT_ERROR in the FACTORY_DEFAULT_STATUS register (see 8.1.5.8) and SECURITY_ERROR in the FACTORY_DEFAULT_STATUS_FAIL_INFO register (see 8.1.5.14)	Delete the wrapped DEK (providing instant secure erase)
Firmware	Admin, Owner	set FIRMWARE_OPS_ERROR in the FIRMWARE_OPS _STATUS register (see 8.1.5.11) and SECURITY_ERROR in the FIRMWARE_OPS_FAIL_FAIL_INFO register (see 8.1.5.15)	none
Operational Unit – Firmware Image Data	Admin, User	set OPERATIONAL_UNIT_OPS_ERROR in the OPERATIONAL_UNIT_OPS _STATUS register (see 8.1.5.12) and SECURITY_ERROR in the OPERATIONAL_UNIT_FAIL_INFO register (see 8.1.5.15)	none
Operational Unit – Vendor Log Page	Owner	set OPERATIONAL_UNIT_OPS_ERROR in the OPERATIONAL_UNIT_OPS _STATUS register (see 8.1.5.12) and SECURITY_ERROR in the OPERATIONAL_UNIT_FAIL_INFO register (see 8.1.5.15)	none
Operational Unit – Label Data	Admin, User	set OPERATIONAL_UNIT_OPS_ERROR in the OPERATIONAL_UNIT_OPS _STATUS register (see 8.1.5.12) and SECURITY_ERROR in the OPERATIONAL_UNIT_FAIL_INFO register (see 8.1.5.15)	none
Operational Unit – Security Protocol	none	allowed	none
Abort	none	allowed	none

B.8.6 TCG Services

B.8.6 TCG Services (cont'd)

Table 41 describes how various TCG services are impacted by security.

B.8.6 TCG Services (cont'd)

Table 41 — Authenticated and Unauthenticated Services

Operation	Operator Access Control	Behavior if Locked?	Security Behavior if Unlocked
TCG Set PIN	Owner, Admin, User		
Lock/Unlock FW Download	Owner, Admin		
Enable/Disable Admins or Users	Admin		
Set Range Attributes	Admin		
Lock/Unlock Range	Admin, User		
Read/write			
Cryptographic Erase	Admin, User		Delete the wrapped DEK (providing instant secure erase)
Exit Security Mode	Owner, Admin		

B.8.7 Key Usage

Table 42 defines key usage.

Table 42 — Key Usage

Key	Description	Type	Role	Services
PSID (Physical Presence Security ID)	Owner authentication	PIN 32 byte ASCII string printed on label	Owner	Exit Security Mode
SID (Security ID)	Owner authentication	PIN 32 byte value	Owner	Set PIN
Admin Password(s)	Administrator authentication	PIN 32 byte value	Admin	Set PIN
User Password(s)	User authentication	PIN 32 byte value	Admin, User	Set PIN
LBA Range MEK	MEK mixed with MEKEK	AES key 32 bytes (256 bits)	Admin, User	Unlock
MEKEK	Media Encryption Key Encryption Key	AES key 32 bytes (256 bits)	Admin, User	Unlock
Seed Key	RNG key	Hash key 64 bytes	Creation of MEKs	
Seed	RNG seed (entropy)	520	RNG	

B.8.8 Using Ruby SSC Devices

The host should perform the following steps to “taking ownership” of a module:

1. Set an administrator C_PIN (take ownership)
2. Activate the Locking Security Provider
3. Set up the global locking range
4. Set the users’ C_PIN (as administrator)
5. Enable users

B.9 Printed Label

The module shall include a Physical Presence Security Identifier (PSID) as defined in TCG Storage *Opal SSC Feature Set: PSID*. The PSID:

- shall be randomly assigned
- shall be unique per module
- shall consist of 32 of the following ASCII characters: [A-Z0-9]
- shall not be included in the SPD EEPROM or anywhere else that is externally accessible in the NVDIMM-N, including NVDIMM-N controller registers, debug ports, and scan chains.

The printed label shall include the PSID in text and in the DataMatrix ECC 200 barcode.

In text, the PSID shall be printed as a 32 byte ASCII string without any separator characters. Example:

PSID:ZLZADSD5422DZD1GLCWSSE0X8QML98E3

The PSID shall be included in the DataMatrix ECC 200 barcode after “(K)” (meaning “key”). Example:

(K)ZLZADSD5422DZD1GLCWSSE0X8QML98E3

B.10 Sanitization

Media sanitization is defined by SP800-88 *Guidelines for Media Sanitization* (Rev 1, Dec 2014) as "a general term referring to the actions taken to render data written on media unrecoverable by both ordinary and extraordinary means."

SP800-88 defines the following levels:

- Clear — Overwrite user-addressable storage space using standard write commands; may not sanitize data in areas not currently user-addressable (such as bad blocks and overprovisioned areas)
- Purge — Overwrite or erase all storage space that might have been used to store data using dedicated device sanitize commands, such that data retrieval is "infeasible using state of the art laboratory techniques"
- Destroy — Ensure that data retrieval is "infeasible using state of the art laboratory techniques" and render the media unable to store data (such as disintegrate, pulverize, melt, incinerate, or shred)

An NVDIMM-N holds user data between a Catastrophic Save operation until an Arm operation or an Erase operation.

The module shall provide two mechanisms that comply with the SP800-88 Purge level of sanitization:

- The Factory Default operation (see 7.2.9)
- TCG RevertSP method to the Locking SP, or Revert method to the Admin SP

B.10 Sanitization (cont'd)

The module shall sanitize all places that might have held user data (including mapped out blocks).

The module shall use cryptographic erasure for both operations. This means the module shall erase the data encryption key (which is only stored inside the module) and start using a new data encryption key.

- Factory Default: cryptographic erase + flash block erase
- RevertSP: cryptographic erase

NOTE: A module implementing XTS-AES and changing DEKs on each Arm operation effectively performs a Purge on each Arm operation.

B.11 Signed Firmware

B.11.1 Signed Firmware Overview

A module with programmable components shall implement signed firmware and check the signature before saving the firmware image. The code performing this checking shall itself be signed (e.g., the previous version of that same firmware image, or another component that is itself signed).

The downloaded image is a concatenation of:

- unsigned firmware image
- signature

The firmware image may contain multiple images, including FPGA bit streams and content for DIMM SPDs.

Figure 19 shows the code signing process.

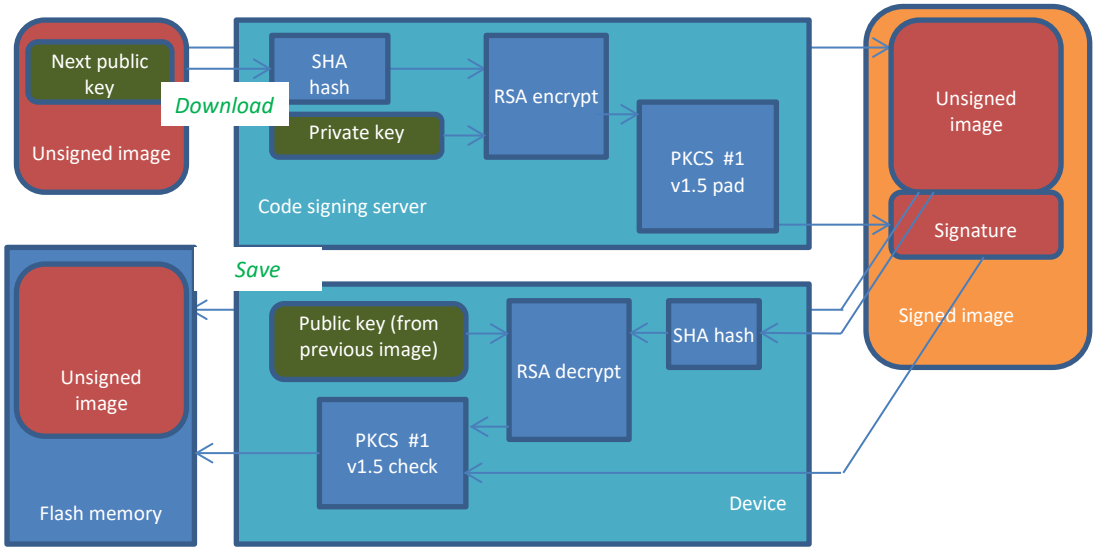

Figure 19 — Code Signing Process

B.11.2 Access to Flash Memory

If the flash memory is only writeable by signed code, then the signature need not be stored in the flash memory and validated on each boot. If the flash memory can be written by any other mechanism (e.g., if the flash ROM is on an I^2C bus and the I^2C host is accessible to more than just the signed code), then the module shall store the signature in flash memory and validate the signature on each boot.

B.11.3 Signature Requirements

The module shall accept firmware signed with the RSA digital signature algorithm specified in FIPS 186-4 chapter 5, which includes ANSI X9.31, PKCS #1, and SP800-90 by reference. The RSA digital signature algorithm includes:

- SHA-256 hash of the unsigned image
- RSA-2048 encryption of the hash
- RSASSA-PKCS1-v1.5 digital signature format

PKCS #1 v1.5 creates a 256 byte signature, carrying the encrypted 16-byte SHA-256 hash value plus padding values. The module shall check all 256 bytes in the signature:

- bytes 0-31 (32 bytes): SHA-256 hash
- bytes 32-51 (20 bytes): DER encoding of DigestInfo (constant values described in PKCS-1 v2.1 page 38)
- bytes 52-253 (202 bytes): padding string (PS) bytes each containing 0xff (for block type 0x01)
- byte 254 (1 byte):block type containing 0x01 (indicates private-key operation, and ensures encryption block is a large value)
- byte 255 (1 byte): leading value of 0x00 (value that ensures the encryption block is less than the modulus)

NOTE: Those byte numbers are based on an implementation in which the bytes in the signature are reversed from the mathematical endianness.

These additional features are not required:

- A SHA hash stronger than SHA-256 (e.g., SHA-384, SHA-512) is not required.
- An RSA modulus greater than 2048 bits is not required.
- RSASSA-PSS is not required. This adds optimal asymmetric encryption padding (OAEP), which includes a random value during encryption that is later discarded during decryption.

Although the signature could be placed at the beginning or interspersed into the firmware image, there is no advantage in doing so; simply appending with the “cat” command suffices. The device may assume that the last 256 bytes are the signature, implying the length of the unsigned image.

The firmware image need not be encrypted. Even though it carries the future public key and the code to check the signature, any tampering is expected to cause the signature to fail for this download.

For devices containing an FPGA that supports it, bitstream encryption may be used on the FPGA image.

NOTE: Firmware implementations of the SHA and RSA algorithms are slow but usually sufficient - firmware downloads are rare.

B.11.4 Public Key Handling

The public key, although not a secret, must not be able to be replaced by an attacker. Module firmware should include the public key inside the signed firmware image itself, not stored in a separate NVRAM or ROM on the device. If a private key is ever compromised, this allows migrating to a new private/public key pair with a firmware upgrade. After upgrading, the user is again safe from rogue firmware being loaded based on the old key pair.

Module firmware may accept images signed with alternative keys. This requires performing the RSA algorithm multiple times to see if it decodes properly. This might be useful to allow switching between test images and production images, provided there is a mechanism to clearly warn users that they have switched from the production stream to the test stream ("tamper-evident").

Certificates are not used to deliver the public key to the device, certificates are not checked with a chain of trust, and certificate revocation is not involved.

B.11.5 Firmware Version Numbers

As described in SP800-147B, the supplier shall assign firmware version numbers that can be checked to prevent rollback after the signature is checked before programming flash memory. Whether a given version prevents rollback shall be determined on a version-by-version basis.

The rationale for this is to prevent loading an old version that has security flaws, or an image that was signed by a compromised security key.

Example: firmware version 1.5 containing public key X allows downloading firmware version 1.6 signed with private key X, containing public key Y.

If a release needs to be revoked (e.g., because severe bugs were found), then a new version should be issued instead (e.g., if 1.5 is good, 1.6 is released but turns out bad, release 1.7 that is the same as 1.5 rather than allow re-downloading 1.5).

B.11.6 Managing Production vs. Test Builds

It is undesirable that an end user might accidentally download a test image that has severe bugs. However, production firmware that prevent loading test images means it is hard for engineers to switch back from production firmware to test firmware.

Some approaches:

- Use a ROM programmer: this bypasses all checks, and suffices for very small development quantities.
- Physical presence detection: if a jumper is present, bypass all checks. This suffices for small quantities, but can be a problem for wide area testing with hundreds of systems.
- Notification: allow downloading both, but provide clear notification to the user when changing from the production series back to the test series or rolling back to an earlier version, and provide notification to the user on every boot that the device is not running production firmware. This is similar to a FIPS 140-2 security module exiting FIPS 140-2 mode.

B.11.7 FIPS 140-2 Compliance for Signed Firmware

For FIPS 140-2 compliance, it is important that the module cannot be knocked out of a FIPS 140-2 approved mode of operation without that being visible to the user. Since the module is inside the server chassis, and the server is likely to be just one of many in a data center, a physical indicator like an LED has limited utility.

B.11.7 FIPS 140-2 Compliance for Signed Firmware (cont'd)

FIPS 140-2 certifications are for a specific firmware revision, so it is acceptable to require the system to confirm that the module is still in a FIPS 140-2 approved mode of operation during POST by:

- querying the firmware revision via the BAEBI FW_REV registers;
- ensuring the BAEBI Security Typed Block Data area is supported; and
- ensuring the TPer Feature descriptor, Locking Feature descriptor, and Ruby SSC Feature descriptor are present and returning the values defined in the SSC.

Assuming the module only accepts signed firmware, and the signer promises to not sign firmware images that report bogus firmware versions, the BAEBI firmware revision registers are sufficient; there is no risk within scope of hacked firmware pretending to be a compliant version. Physically reprogramming the NOR flash ROM is outside of the scope.

B.12 FIPS 140-2 (Cryptographic Module) Compliance

The module manufacturer should define a cryptographic module boundary within the NVDIMM-N. This shall include the hardware implementing encryption (e.g., FPGA or ASIC, microcontroller, NOR flash containing microcontroller firmware). This need not include the NVM (e.g., flash memory) or SDRAM. All chips inside the cryptographic boundary should be surface-mount, not through-hole, unless the back side of the board is also enclosed.

The module should be designed to comply with FIPS 140-2 *Security Requirements For Cryptographic Modules* Security Level 2 (tamper-evident physical, role-based authentication).

The module should be capable of being certified at Security Level 1.

The module should be capable of being certified at Security Level 2 by adding tamper-evident stickers or obscuring all components within the cryptographic module boundary with an opaque coating.

The module may be designed to comply with Security Level 3 (e.g., tamper-resistant, identity-based authentication, encrypted entry of critical security parameters such as keys).

B.13 Common Criteria Compliance

The module manufacturer should provide a *Security Target* document describing how the module conforms to the *collaborative Protection Profile for Full Drive Encryption – Encryption Engine (cPP_FDE_EE)*.

The module should be designed to support a host complying with the *collaborative Protection Profile Protection Profiles for Full Drive Encryption – Authorization Acquisition (cPP_FDE_AA)*.

Annex C - (Informative) Differences Between Revisions

This annex briefly describes changes in the revisions of this standard that might affect interoperability between the host and the module. The SPECREV register (see 8.1.2.2) indicates the revision implemented by the module.

C.1 Differences Between JESD245 (v1.0) and JESD245A (v2.0)

Description of Change
Add references to DDR4 RCD02 and DDR4 DB02 specs.
Add LCOM and RCD.
Add LCOM definition.
Fix typo in target address bit size.
Add new Clause for I^2C Resets.
Add new Clause for LCOM Interface.
Add CAPABILITIES1 register.
Add new example for spec version 2.0.
Add definitions for multi-pulse save and LCOM Interface support.
Add new Clause for CAPABILITIES1.
Update SAVE_N trigger description.
Add support for Arm and Erase operation.
Add support for FIRMWARE_BLOCK_RECEIVED support.

C.2 Differences Between JESD245A (v2.0) and (JESD245B)(v2.1)

Description of Change
Added a new Abort CMD Timeout in Seconds bit in the CAPABILITIES1 register that affects the interpretation of the ABORT_CMD_TIMEOUT register value.
Added the SAVE_ABORT, NO_SAVE_N, and INSUFFICIENT_SAVE_N bits to the CSAVE_FAIL_INFO1 register.
Added the HOST_CSAVE_FAIL and HOST_CSAVE_WORKFLOW_FAILURE_COUNTn registers.
Added the PERMANENT_HARDWARE_FAILURE bit to the MODULE_HEALTH register.
Added the EVENT_N_LOW bit to the MODULE_HEALTH register.
Obsoleted Voltage Regulator Failed as an event notification trigger (in the EVENT_NOTIFICATION_SUPPORT, SET_EVENT_NOTIFICATION_CMD, and SET_EVENT_NOTIFICATION_STATUS0 registers), since it is also part of Persistency Enabled.
Clarified that event notification triggers are edge-based, not level-based.
Added a one-time use error injection mode, with a new ONE_TIME_USE bit in the INJECT_ERROR_TYPE register.
Added the INJECT_OPS_FAILURES1 register with bits to inject additional types of failures.
Defined Operational Unit operations for accessing Typed Block Data using a small buffer.
Defined Label Data type of Typed Block Data.
Added annex describing requirements for handling NVRDIMM-N SDRAM and RCD in the Restore transition.
Added support for I^2C word transactions. Includes a new bit in the CAPABILITIES1 register.
Clarified the SPECREV register value mapping to JESD245x standards.

C.2 Differences Between JESD245A (v2.0) and (JESD245B)(v2.1) (cont'd)

Added support for using the Reset Controller operation to activate firmware. Includes: • a new Reset Controller Scope field in the CAPABILITIES1 register that indicates if the module still detects SAVE_N during Reset Controller operations • a new event notification reason in the EVENT_NOTIFICATION_SUPPORT register, SET_EVENT_NOTIFICATION_CMD register, SET_EVENT_NOTIFICATION_STATUS0 register
Added support for Subcomponent Revision Numbers to allow downloading smaller firmware images (e.g., when an FPGA image does not need to be updated). Includes: • New firmware image header format 0x01 including more SPD field values (Non-Volatile Memory Subsystem Controller Manufacturer ID, Non-Volatile Memory Subsystem Controller Product Identifier, Module Revision Code, Non-Volatile Memory Subsystem Controller Revision Code), a Controller Firmware Revision field, an Energy Source Firmware Revision field, and a Subcomponent Firmware Revision field. • New slot-specific Energy Source Revision registers (SLOT0_ES_FWREV0, SLOT0_ES_FWREV1, SLOT1_ES_FWREV0, SLOT1_ES_FWREV1) • New slot-specific Subcomponent Firmware Revision registers (SLOT0_SUBFWREV, SLOT1_SUBFWREV)
Added duration, count, and timeout registers: • Last Duration: Arm, Factory Default, Firmware, Operational Unit • Success Count: Arm, Factory Default, Firmware, Operational Unit, and renamed all such registers from NUM_<operation>_COUNTn to <operation>_SUCCESS_COUNTn. Added support for aborting the Factory Default operation. Marked formerly defined bits in ARM_CMD and SET_EVENT_NOTIFICATION_STATUS as obsolete. Define OPERATIONAL_UNIT_IDx registers as RW not WO.
Added <operation>_FAILURE_COUNTn registers for all operations.
Clarified that all device statistics counters are saturating counters that stop at 0xFFFF.
Changed all SAVE register names and field names to CSAVE
Obsoleted all the 8-bit temperature registers, replacing them with 16-bit registers compliant with the TSE2004 format (including Sign bits): MIN/MAX_OPERATING_TEMP, ES_TEMP_ERROR_THRESHOLD, ES_TEMP_WARNING_THRESHOLD, and MIN/MAX_ES_OPERATING_TEMP. For the thresholds, split into LOW and HIGH as well, like TSE2004 supports (going below the low threshold is as bad as going above the high threshold). To match TSE2004 semantics, made the notable event be exceeding the threshold rather than equaling or exceeding it – although this differs from the NVM/ES_LIFETIME_ERROR/WARNING thresholds.
Added a RESTORE_FAIL_INFO register.
Require SPD version to be at least 0x12 (revision 1.2) rather than 0x5 (revision 0.5), since JESD21C Page 4.1.2-L-5 (release 5) will specify that it documents NVDIMM-N Revision 1.2. Modules compliant with previous versions might use older values, but modules compliant with this version of JESD245 should comply with the latest SPD definitions.
Added registers to let the host tell the NVDIMM the SDRAM mode register and RCD control word values it is using to facilitate interoperability for the Catastrophic Save transition. This includes new register page 4 through page 9.
Added a SAVE_REJECT bit to the SAVE_STATUS0 register to indicate the operation was rejected without even being started.
Made the Clear All bit in the NVDIMM_MGT_CMD0 register not touch the In Progress registers.

C.2 Differences Between JESD245A (v2.0) and (JESD245B)(v2.1) (cont'd)

While NVDIMM_READY is not 0xA5, also allow access to the MODULE_HEALTH_STATUS1 register (in addition to OPEN_PAGE, NVDIMM_READY, MODULE_HEALTH, and MODULE_HEALTH_STATUS0).
Removed 0 from register names that don't yet have a 1 counterpart: NVDIMM_FUNC_CMD0, CSAVE_INFO0, CSAVE_STATUS0, ERASE_STATUS0, ARM_STATUS0, FACTORY_DEFAULT_STATUS0, SET_EVENT_NOTIFICATION_STATUS0
Added 0 to register names for those that have a 1 counterpart: CAPABILITIES, INJECT_OPS_FAILURES
Changed the INVALID_FIRMWARE bit in the MODULE_HEALTH_STATUS1 register to mean both images are lost, not just slot 1 (so it is a valid contributor to the PERSISTENCY_LOST bit in the MODULE_HEALTH register)
Clarify that the Catastrophic Save operation shall perform an erase operation if still needed when triggered
Clarify that these registers are accessible while the module is responding to I^2C transactions but reporting that NVDIMM_READY is not 0xA5: OPEN_PAGE, NVDIMM_READY, MODULE_HEALTH, MODULE_HEALTH_STATUS0, and MODULE_HEALTH_STATUS1.
Clarify that the NVM lifetime error indicates the operational, not warranty, lifetime has been reached.

C.3 Differences Between JESD245B (v2.1) and this Standard (JESD245C)(v2.2)

Description of Change
Clarified Figure 10
Added ARM_INFO bit to MODULE_HEALTH register in Clause 8.1.8.1
Added two Normative References regarding storage
Changed the Atomic Arm and Erase command to Atomic Save and Erase
Restructured Clause 7.2 and added Clause 7.2.1.3 including Catastrophic Save triggered by RESET_n
Added Security Protocol Data Clause 7.17
Clarified module behavior for START_CSAVE request while locked in Clause 7.2.1
Clarified module behavior for START_CSAVE request while module is not erased in Clause 7.2.1
Clarified module behavior for START_CSAVE request with data encryption in Clause 7.2.1
Clarified module behavior for START_CSAVE request triggered by RESET_n pin in Clause 7.2.1.1
Clarified module behavior for START_CSAVE request triggered by SAVE_n pin in Clause 7.2.1.2
Added timing information in Table 2 for SAVE_n
Clarified module behavior for Restore operation in Clause 7.2.2
Clarified module behavior for Erase operation in Clause 7.2.3
Clarified module behavior for Arm operation in Clause 7.2.4
Clarified module behavior for Management operation in Clause 7.2.5
Clarified module behavior for Firmware operation in Clause 7.2.8
Clarified module behavior for Factory Default operation in Clause 7.2.9
Clarified module behavior for Reset Controller operation in Clause 7.2.10
Clarified module behavior for Atomic Save and Erase operation in Clause 7.2.12
Added C for conditional in register definition tables in Clause 8
Added register type groupings in Page 0 – Page 9 register tables in Clause 8 for better readability
Added links to register definitions in Page 0 – Page 9 register tables in Clause 8
Added Security Protocol Registers in Table 11 (Page 0) registers 0x4C – 0x4E
Added Runtime Command Status Registers in Table 11 (Page 0) registers 0x73 – 0x76
Clarified module behavior for CSAVE_TRIGGER_SUPPORT in Clause 8.1.3.5

C.3 Differences Between JESD245B (v2.1) and this Standard (JESD245C)(v2.2) (cont'd)

Clarified bit behavior(s) in NVDIMM_MGT_CMD0 – Offset 0x40 register in Clause 8.1.4.1
Clarified bit behavior(s) in NVDIMM_MGT_CMD1 – Offset 0x41 register in Clause 8.1.4.2
Clarified bit behavior(s) in ARM_CMD – Offset 0x45 register in Clause 8.1.4.4
Added Security Protocol register information in Clauses 8.1.4.9 through 8.1.4.11
Clarified bit behavior(s) in ERASE_STATUS – Offset 0x68 register in Clause 8.1.5.6
Clarified bit behavior(s) in ARM_STATUS – Offset 0x6A register in Clause 8.1.5.7
Clarified bit behavior(s) in FACTORY_DEFAULT_STATUS – Offset 0x6C register in Clause 8.1.5.8
Clarified bit behavior(s) in FIRMWARE_OPS_STATUS – Offset 0x71 register in Clause 8.1.5.11
Added Runtime Command Status register information in Clauses 8.1.5.13 through 8.1.5.16
Added SECURITY_ERROR bit to RESTORE_FAIL_INFO – Offset 0x88 register in Clause 8.1.5.17
Added SECURITY_ERROR bit to OPERATIONAL_UNIT_FAIL_INFO – Offset 0x8F register in Clause 8.1.5.18
Added Triggered_by_RESET_n bit to CSAVE_INFO – Offset 0x80 register in Clause 8.1.6.1
Added SECURITY_ERROR bit to CSAVE_FAIL_INFO1 – Offset 0x85 register and clarified bit(s) behaviors in Clause 8.1.6.3
Deleted a bit definition from MODULE_OPS_CONFIG – Offset 0xAA register in Clause 8.1.8.7
Clarified host behavior expectation in Clause 8.3.4.4
Added SECURITY_ERROR bit to TYPED_BLOCK_DATA – Offset 0x04 register and clarified bit(s) behaviors in Clause 8.4.2.1
Clarified module behavior for Arm workflow in Clause 9.4
Clarified module behavior for Erase workflow in Clause 9.5
Clarified host behavior for Abort Running Operation workflow in Clause 9.6
Clarified module behavior for Firmware Update workflow in Clause 9.7
Clarified module behavior for Typed Block Data workflow in Clause 9.8
Added Annex B for self-encrypting NVDIMM-Ns
Changed the SPD Revision field from 0x12 (NVDIMM Revision v1.2) to 0x11 (v1.1), referencing Annex L Release 4 rather than Release 5 (which was never released…the need to add a bit reporting if the NVDIMM-N included a pullup resistor became moot).
Removed reference to a forthcoming JESD21C Page 4.1.6-2 defining an EE10004av; that was never balloted

C.4 Differences Between JESD245C (v2.2) and this Standard (JESD245D)(v2.3)

Description of Change
Added an “encryption” bit to the CAPABILITIES1 register in Clause 8.1.3.2
Added an Catastrophic Save on RESET_n bit to the CAPABILITIES1 register in Clause 8.1.3.2
Changed the Operational Workflow Clause 9.8.1 and 9.8.2
Changed the SPECREV Register Clause 8.1.2.2 to indicate the correct specification revision

C.5 Differences between JESD245D (v2.3) and this Standard (JESD245E)(v2.4)

Description of Change
Changed the Operational Unit size for Security Protocol Data from 32 to 256 bytes
Modified Table 10 to remove the reserved registers offsets and included the offset 0x88 – 0x8F description
Modified Table 12 to include JESD245E
Added statement to clear the OPERATIONAL_UNIT_FAIL_INFO register for Bit 1 of the NVDIMM_MGT_CMD0 register 8.1.4.1
Modified the Bit 3 definition for the NVDIMM_MGT_CMD1 register in Clause 8.1.4.2
Added Bit Definitions 2 - 6 in the OPERATIONAL_UNIT_FAIL_INFO register 8.1.5.18
Changed the TYPE_BLOCK_DATA_SIZE0,1 and 2 registers access from RW to RO/RW for security protocol data
Clarified Reading Typed Block Data workflow for non-security protocol data types
Clarified Writing Typed Block Data workflow for non-security protocol data types
Added Figure 11 to clarify the Typed Block Data structure
Added Clause 9.8.3 Security Protocol Data Workflow
Changed references of “Master” to “Controller” and “Slave” to “Target”

Standard Improvement Form **JEDEC Standard** **JESD245E**

The purpose of this form is to provide the Technical Committees of JEDEC with input from the industry regarding usage of the subject standard. Individuals or companies are invited to submit comments to JEDEC. All comments will be collected and dispersed to the appropriate committee(s).

If you can provide input, please complete this form, and return to:

JEDEC
Attn: Publications Department
3103 North 10th Street
Suite 240 South
Arlington, VA 22201-2107

Fax: 703.907.7583

1. I recommend changes to the following:

☐ Requirement, clause number _____

☐ Test method number _____ Clause number _____

The referenced clause number has proven to be:

☐ Unclear ☐ Too Rigid ☐ In Error

☐ Other _____

2. Recommendations for correction:

_____

_____

_____

_____

3. Other suggestions for document improvement:

_____

_____

_____

_____

Submitted by

Name: _____ Phone: _____

Company: _____ E-mail: _____

Address: _____

City/State/Zip: _____ Date: _____


JEDEC®