
JESD 21-C Section Title: Annex L: Serial Presence Detect (SPD) for DDR4 SDRAM Modules**Committee Document Reference Title: DDR4 SPD Document Release 4****Device Type Identifier: UDIMM Revision 1.1****RDIMM Revision 1.2****LRDIMM Revision 1.2****NVDIMM-N Revision 1.1**

1 Scope

This annex describes the serial presence detect (SPD) values for all DDR4 modules. The SPD data provides critical information about all modules on the memory channel and is intended to be use by the system's BIOS in order to properly initialize and optimize the system memory channels. The storage capacity of the SPD EEPROMs is limited, so a number of techniques are employed to optimize the use of these bytes, including overlays and run length limited coding.

All unused entries will be coded as 0x00. All unused bits in defined bytes will be coded as 0 except where noted.

Timing parameters in the SPD represent the operation of the module including all DRAMs and support devices at the lowest supported supply voltage (see SPD byte 11), and are valid from $t_{CKAVGmin}$ to $t_{CKAVGmax}$ (see SPD bytes 18 and 19).

To allow for maximum flexibility as devices evolve, SPD fields described in this document may support device configuration and timing options that are not included in the JEDEC DDR4 SDRAM data sheet (JESD79-4). Please refer to DRAM supplier data sheets or JESD79-4 to determine the compatibility of components.

2 History

Computer main memory buses have traditionally been defined by the generation of memory attached to the bus, e.g., EDO, SDRAM, DDR1, etc. The bus interface protocol and characteristics have largely been defined by the memory type. Clock frequency, CAS latencies, refresh recovery times and similar parameters defined the timing of signals between memory controller and the memory, and parameters such as number of ranks installed and device widths allowed system software to determine the memory capacity and similar high level characteristics of each module.

Over time, the memory bus has been extended to include additional features for application specific requirements. Registered DIMMs, for example, increased total capacity by buffering the loading of the address bus signals, allowing more DRAM to be installed. Similarly, Load Reduced DIMMs buffered the data bus as well, allowing even more ranks of memory to be supported. As each new extension to the function of the memory bus was introduced, system software combined knowledge of those extensions with information programmed into the EEPROM in the SPD to determine how to use and optimize the new features. Using the RDIMM as an example, systems understood that an additional clock of latency needed to be added to the DRAM latency to accommodate propagation delay through the register.

In later generations, the DRAM to host interface is completely virtualized. A memory module may have no DRAM at all, yet may use the DRAM bus to communicate with the host by emulating the DRAM channel interface. These virtual interfaces must appear to the system as one of the base module types, i.e., UDIMM, RDIMM, or LRDIMM. Modules that incorporate at least one non-DRAM media type for the purpose of main memory data storage are called “hybrid”, they act like a DRAM but on the other side of the interface protocol are some other memory type(s).

3 SPD Architecture

The SPD contents architecture must support the many variations of module types while remaining efficient. A system of overlay information selected through the use of “key bytes”, or selectors for the type of information to load has been implemented. The following DDR4 module SPD address map describes where the individual lookup table entries will be held in the serial EEPROM.

Consistent with the definition of DDR4 generation SPD devices (EE1004 and TSE2004) which have four individual write protection blocks of 128 bytes in length each, the SPD contents are aligned with these blocks as shown in Table 1:

Table 1 — SPD Contents Architecture

Block	Range		Description
0	0~127	0x000~0x07F	Base Configuration and DRAM Parameters
1	128~191	0x080~0x0BF	Standard Module Parameters -- See annexes L.1.x for details
	192~255	0x0C0~0x0FF	Hybrid Module Parameters -- See annexes L.2.x for details
2	256~319	0x100~0x13F	Hybrid Module Extended Function Parameters - See Annexes L.3.x for details
	320~383	0x140~0x17F	Manufacturing Information
3	384~511	0x180~0x1FF	End User Programmable

Operating parameters for the different module types are defined in the following subsections and will reside in the appropriate address ranges of the EEPROM address map depending on the module type. Please see Overlay Schema for further detail.

1. Section 9 - Standard Module Parameter - Overlay Bytes 128-191
 - Section 9.1 - UDIMMs
 - Section 9.2 - RDIMM
 - Section 9.3 - LRDIMM
2. Section 10 - Hybrid Module Parameters - Overlay Bytes 192 - 255
 - Section 10.1 NVDIMM
3. Section 11 - Hybrid Module Extended Function Parameters - Overlay Bytes 256-319
 - Section 11.1 - Energy Backed Byte Addressable NVDIMM
 - Section 11.2 - Energy Backed Block Addressable NVDIMM
 - Section 11.3 - Non-Energy Backed Byte Addressable NVDIMM

4 Overlay Schema

The following Schema exemplify the manner in which the base configuration information along with the appropriate subsections are to be overlaid onto the appropriate address spaces in order to provide a complete definition of the module.

4.1 UDIMM Overlay Schema

Table 2 shows the UDIMM overlay schema.

Key Byte 2 contains value 0x0C or 0x0E

Key Byte 3 contains any of the following values:

- 0x02, UDIMM
- 0x03, SO-DIMM
- 0x06, Mini-UDIMM
- 0x09, 72b-SO-UDIMM
- 0x0C, 16b-SO-DIMM
- 0x0D, 32b-SO-DIMM

4 Overlay Schema (Cont'd)

4.1 UDIMM Overlay Schema (Cont'd)

Table 2 — UDIMM Overlay Schema

Block	Range	Description
0	0~127	Base Configuration and DRAM Parameters
1	128~191	Insert Section 9.1: Unbuffered Memory Module Types
	192~255	Unused
2	256~319	Reserved
	320~383	Module Supplier's Data
3	384~511	End User Programmable

4.2 RDIMM Overlay Schema

Table 3 shows the RDIMM overlay schema.

Key Byte 2 contains value 0x0C or 0x0E

Key Byte 3 contains any of the following values:

- 0x01, RDIMM
- 0x05, Mini-RDIMM
- 0x08, 72b-SO-RDIMM

Table 3 — RDIMM Overlay Schema

Block	Range	Description
0	0~127	Base Configuration and DRAM Parameters
1	128~191	Insert Section 9.2: RDIMM Memory Module Types
	192~255	Unused
2	256~319	Reserved
	320~383	Module Supplier's Data
3	384~511	End User Programmable

4.3 LRDIMM Overlay Schema

Table 4 shows the LRDIMM overlay schema.

Key Byte 2 contains value 0x0C or 0x0E

Key Byte 3 contains any of the following values:

- 0x04, LRDIMM

Table 4 — LRDIMM Overlay Schema

Block	Range	Description
0	0~127	Base Configuration and DRAM Parameters
1	128~191	Insert Section 9.3: LRDIMM Memory Module Types
	192~255	Unused
2	256~319	Reserved
	320~383	Module Supplier's Data
3	384~511	End User Programmable

4.4 Hybrid DIMM Overlay Schema

Table 5 shows the Hybrid DIMM overlay schema.

Key Byte 2 contains value 0x0C or 0x0E

Key Byte 3 contains any of the following values:

- 0x9M, NVDIMM - Non-volatile DIMM; M defines base architecture

Table 5 — Hybrid DIMM Overlay Schema

Block	Range	Description
0	0~127	Base Configuration and DRAM Parameters
1	128~191	Insert Base Module Type Section 9.x
	192~255	Insert Hybrid Memory Parameter Section 9.x
2	256~319	Insert Extended Function Parameter Block Section 9.x
	320~383	Module Supplier's Data
3	384~511	End User Programmable

After programming the SPD contents, suppliers of JEDEC compliant modules must set the write protect bits for SPD device blocks 0 and 1 and must not set the write protect bit for block 3. Write protection of block 2 is required for modules using the Extended Function Parameter Block, such as NVDIMMs. See the EE1004-v/TSE2004av Device Specification for details on the SWPn command protocol.

5 Parsing the SPD

Table 6 provides relevant SPD bytes for parsing.

Table 6 — Some Relevant SPD Bytes for Parsing

SPD Byte(s)	Definition
1	SPD revision for this memory module
2	DRAM interface type presented or emulated
3	Memory module interface type
0~127	Base configuration and DRAM parameters
128~191	Module specific parameters
192~255	Hybrid memory parameters
204~219	NVDIMM function interface descriptors
256~319	Extended function parameter block

The system BIOS will acquire information from the SPD in order to properly configure the systems memory controller. It is assumed the BIOS will parse the SPD data in the order listed below.

Step 1: Parse Byte 2 - Verify the installed DRAM type is supported.

The first step in parsing the SPD is to verify that the DRAM type installed is supported by looking at DRAM device type byte 2. While it is usually not possible to physically plug in the wrong memory type, for example a DDR3 module size or key location should prevent insertion into a DDR4 system, there are cases where byte 2 is used to prevent accidental use of an incompatible memory type such as DDR4E modules in a DDR4-only system.

5 Parsing the SPD (Cont'd)

Step 2: Parse Byte 1 - Verify SPD compatibility. See Section 6 - SPD Revision Progression

The SPD revision byte 1 “encoding” nibble may be used to force legacy systems to reject newer modules. This would typically only occur if a critical error were found in SPD encoding that would require a “fix”. In this case, as in the case of an unsupported DRAM type, system initialization must be halted immediately.

The SPD revision stored in Byte 1 applies to all information for the module, including base information, module specific information, and hybrid information. Each SPD revision exactly defines how many bytes are valid in all other SPD blocks. The number of supported bytes (or bits) may increase from one SPD revision to another within each block as indicated by the “additions” nibble of the SPD revision. For example, an SPD revision 1.3 has more bytes or bits defined than SPD revision 1.2.

This progression of SPD contents is important for the BIOS to DIMM compatibility model. An older system may have a BIOS that only understands SPD revision 1.2 encoding, so if a module is installed that contains revision 1.3 information, the system can accurately interpret all of the historical revision 1.2 information retained in the module that is the subset of the revision 1.3 specification. Similarly, if a module with SPD revision 1.1 information is installed, that same BIOS can interpret all information that was current at the time that SPD 1.1 was defined. Therefore, BIOSes must maintain a knowledge of the active information for each historical SPD revision in order to support older modules.

Step 3: Parse Byte 3 - Determine module type and appropriate overlays.

Key byte 3 for module type determines the subsequent use of overlay information. Byte 3 bits 3~0 define the host to module interface style: unbuffered, registered, or load reduced. Standard module types are defined in Annexes L.1.x.

Key byte 3 also determines the presence of hybrid module types. Byte 3, bit 7 asserts if the module is hybrid, and Byte 3, bits 6~4 define the type of hybrid module. A hybrid module may appear to the system as a superset of any of the base module types. For example, an NVDIMM may present a UDIMM, RDIMM, or LRDIMM compatible interface to the system. Because of this unique second overlay for hybrid information on top of the base module type, the SPD revision for Hybrid modules must include all relevant information for any of the base module types if applicable. Refer to SPD revision progression in Section 6 for more information.

If Byte 3, bits 7~4 indicate the presence of a Hybrid module type, the BIOS must parse two more overlays. Annex L.2, Hybrid Module Parameters (Bytes 192~255) and Section 11 Hybrid Modules Extended Function Parameters (Bytes 256~319). As with the module specific parameters, both of the Hybrid Module parameter sets may increase from SPD revision to revision.

Step 4: Parse Bytes 0-127 - Make base configuration settings to memory interface based on these bytes.

All module types are required to read and interpret this block of data to set up the DRAM type, maximum operating frequency, the number of row, column, bank bits, write recovery time, etc. While these bytes primarily describe the timing of the DRAMs, timing represents the capabilities of the module and it may be necessary to downgrade the timing based on other factors including layout or support components on the module, such as registers.

Step 5: Parse Bytes 128-191 - Configure Standard module memory interface.

These bytes will be referenced and used by the system as needed based on the Standard module type that was determined after Byte 3 was parsed as indicated in Step 2. Bytes 128~191 are coded differently for each standard module type as defined in Section 9.

Step 6: Parse Bytes 192-255 (If Hybrid) - Read Hybrid Module parameters settings.

These bytes will be referenced and used by the system as needed based on the Hybrid module type that was determined after Byte 3 was parsed as indicated in Step 2. Function Format Interface Descriptors defined in bytes 204~205 through 218~219 will be read and utilized as indicated in Step 7. Bytes 192~255 are coded differently for each Hybrid module type and defined in Section 10.

Step 7: Parse Bytes 256-319 (If Hybrid Extended Functions are specified).

These bytes will be referenced and used by the system as needed based on the Function Format interface descriptors read previously during Step 6 from SPD bytes 204~205 through 218~219. Support up to 8 functions per Hybrid module can be defined within Bytes 256~319 and are coded based on Section 11.

For NVDIMM hybrid modules, parsing the two data blocks of hybrid memory parameters in SPD bytes 192~255 and the extended function parameters in SPD bytes 256~319 is intimately related. Each function descriptor contains a 4-bit field in bits 13~10 that creates an address pointer into the extended function parameter block. The formula is: $256 + 4 * \text{function descriptor bits 13~10}$.

The extended function parameter blocks are run length limited, meaning that when one function parameter set stops the next set starts.

5 Parsing the SPD (Cont'd)

(with a granularity of 4 bytes). For example, imagine an NVDIMM with two functions, the first function is a byte addressable energy backed memory which requires 14 bytes of extended parameters and the second function is block addressed requiring 23 bytes of extended parameters. The resulting encoding for that NVDIMM are shown in Table 7:

Table 7 — Example with Imaginary SPD Revision 1.2

SPD Bytes	Coding (binary)	Definition
204~205	1000 0000 0000 0001	Implemented; byte addressable energy backed JEDEC byte addressable energy backed protocol Offset of extended function parameters = $256 + 4 * 0 = 256$
206~207	1001 0000 0100 0001	Implemented; block addressed JEDEC block addressed protocol Offset of extended function parameters = $256 + 4 * 4 = 272$
256~269	<valid>	Extended parameters for byte addressed energy backed function
272~294	<valid>	Extended parameters for block addressed function

The example in Table 7 assumes a certain SPD revision in order for the BIOS to know how many bytes are defined in the extended function parameter fields. If the above example were for SPD revision 1.2, then SPD revision 1.3 is released with 18 bytes for the first function and 26 bytes for the second function, the coding would change as shown in Table 8:

Table 8 — Example with Imaginary SPD Revision 1.3

SPD Bytes	Coding (binary)	Definition
204~205	1000 0000 0000 0001	Implemented; byte addressable energy backed JEDEC byte addressable energy backed protocol Offset of extended function parameters = $256 + 4 * 0 = 256$
206~207	1001 0100 0100 0001	Implemented; block addressed JEDEC block addressed protocol Offset of extended function parameters = $256 + 4 * 5 = 276$
256~273	<valid>	Extended parameters for byte addressed energy backed function
276~301	<valid>	Extended parameters for block addressed function

When an NVDIMM with SPD revision 1.3 is plugged into a legacy system written for SPD 1.2 or less, the BIOS would interpret the first 14 bytes of extended parameters for the byte addressed energy backed function (256~269) and the first 23 bytes of the block addressed function (276~294). The rest of the parameter bytes are ignored.

A newer system that supports SPD revisions up through 1.3 would interpret all bytes on either NVDIMM type in the correct SPD byte addresses, knowing that an NVDIMM with SPD revision 1.2 would have parameters in 256~269 and 272~294, and an NVDIMM with SPD revision 1.3 would have parameters in bytes 256~273 and 276~301.

6 SPD Revision Progression

The use of indexing for functional overlays creates a set of dependencies that affect how SPD revisions must be defined. All module types are dependent on the definition of the general DRAM section bytes 0~127, therefore any change in the general DRAM section requires an increase in the SPD revision for all module types UDIMM, RDIMM, LRDIMM and Hybrid DIMM.

A change in the indexed overlay information for specific module types does not require that the SPD revision for the other module types change. For example, adding information to an RDIMM 1.1 SPD specification would require increasing the RDIMM SPD specification to 1.2, however it would not affect UDIMM, LRDIMM or Hybrid DIMM SPD revisions.

Since hybrid modules may be implemented emulating any module type interface, then any change in the general DRAM section 0~127 or any change in any module type specific information 128~191 does require an increase in the NVDIMM SPD revision level. Since all NVDIMM types share a single overlay 192~255 and 256~319, a change in any NVDIMM SPD revision changes the revision for all NVDIMM types even if nothing changes for a specific NVDIMM type. Table 9 shows a hypothetical sequence starting from all module types at revision 1.0, then showing how SPD revisions would change based on certain actions.

6 SPD Revision Progression (Cont'd)

Table 9 — Hypothetical SPD Revision Progression Showing Revision Relationships

Event	UDIMM	RDIMM	LRDIMM	NVDIMM
Initial SPD release	1.0	1.0	1.0	1.0
Addition in RDIMM Annex	1.0	1.1	1.0	1.1
Addition in LRDIMM Annex	1.0	1.1	1.1	1.2
Addition in LRDIMM Annex	1.0	1.1	1.2	1.3
Addition in General Section	1.1	1.2	1.3	1.4
Addition in UDIMM Annex	1.2	1.2	1.3	1.5
Addition in NVDIMM Annex	1.2	1.2	1.3	1.6
Addition in RDIMM and UDIMM Annexes	1.3	1.3	1.3	1.7
Encoding change in LRDIMM Annex	1.3	1.3	2.3	2.7
Addition in LRDIMM Annex	1.3	1.3	2.4	2.8
Encoding change in General Section	2.3	2.3	3.4	3.8
Addition in RDIMM Annex	2.3	2.4	3.4	3.9

7 Address Map

This section provides the SPD address map for all DDR4 modules. It describes where the individual lookup table entries will be held in the serial EEPROM. Consistent with the definition of DDR4 generation SPD devices which have four individual write protection blocks of 128 bytes in length each, the SPD contents are aligned with these blocks as shown in Table 10:

Table 10 — Address Map

Block	Range		Description
0	0~127	0x000~0x07F	Base Configuration and DRAM Parameters
1	128~191	0x080~0x0BF	Module Specific Parameters -- See annexes for details
	192~255	0x0C0~0x0FF	Hybrid Memory Parameters -- See annexes for details
2	256~319	0x100~0x13F	Extended Function Parameter Block
	320~383	0x140~0x17F	Manufacturing Information
3	384~511	0x180~0x1FF	End User Programmable

After programming the SPD contents, suppliers of JEDEC compliant modules must set the write protect bits for SPD device blocks 0 and 1 and must not set the write protect bit for block 3. Write protection of block 2 is required for modules using the Extended Function Parameter Block, such as NVDIMMs. See the EE1004-v/TSE2004av Device Specification for details on the SWPn command protocol.

7.1 Block 0: Base Configuration and DRAM Parameters

Table 11 details the location of each byte in this block.

Table 11 — Block 0: Base Configuration and DRAM Parameters

Byte Number	Function Described		Notes
0	0x000	Number of Serial PD Bytes Written / SPD Device Size	1, 2
1	0x001	SPD Revision	
2	0x002	Key Byte / DRAM Device Type	
3	0x003	Key Byte / Module Type	
NOTE 1 Number of SPD bytes written will typically be programmed as 384 bytes. NOTE 2 Size of SPD device will typically be programmed as 512 bytes. NOTE 3 From DDR4 SDRAM datasheet.			

7 Address Map (Cont'd)

7.1 Block 0: Base Configuration and DRAM Parameters (Cont'd)

Table 11 — Block 0: Base Configuration and DRAM Parameters (Cont'd)

Byte Number		Function Described	Notes
4	0x004	SDRAM Density and Banks	3
5	0x005	SDRAM Addressing	3
6	0x006	Primary SDRAM Package Type	3
7	0x007	SDRAM Optional Features	3
8	0x008	SDRAM Thermal and Refresh Options	3
9	0x009	Other SDRAM Optional Features	3
10	0x00A	Secondary SDRAM Package Type	3
11	0x00B	Module Nominal Voltage, VDD	3
12	0x00C	Module Organization	
13	0x00D	Module Memory Bus Width	
14	0x00E	Module Thermal Sensor	
15	0x00F	Extended module type	
16	0x010	Reserved -- must be coded as 0x00	
17	0x011	Timebases	
18	0x012	SDRAM Minimum Cycle Time ($t_{CKAVGmin}$)	3
19	0x013	SDRAM Maximum Cycle Time ($t_{CKAVGmax}$)	3
20	0x014	CAS Latencies Supported, First Byte	3
21	0x015	CAS Latencies Supported, Second Byte	3
22	0x016	CAS Latencies Supported, Third Byte	3
23	0x017	CAS Latencies Supported, Fourth Byte	3
24	0x018	Minimum CAS Latency Time (t_{AAmin})	3
25	0x019	Minimum RAS to CAS Delay Time (t_{RCDmin})	3
26	0x01A	Minimum Row Precharge Delay Time (t_{RPmin})	3
27	0x01B	Upper Nibbles for t_{RASmin} and t_{RCmin}	3
28	0x01C	Minimum Active to Precharge Delay Time (t_{RASmin}), Least Significant Byte	3
29	0x01D	Minimum Active to Active/Refresh Delay Time (t_{RCmin}), Least Significant Byte	3
30	0x01E	Minimum Refresh Recovery Delay Time ($t_{RFC1min}$), LSB	3
31	0x01F	Minimum Refresh Recovery Delay Time ($t_{RFC1min}$), MSB	3
32	0x020	Minimum Refresh Recovery Delay Time ($t_{RFC2min}$), LSB	3
33	0x021	Minimum Refresh Recovery Delay Time ($t_{RFC2min}$), MSB	3
34	0x022	Minimum Refresh Recovery Delay Time ($t_{RFC4min}$), LSB	3
35	0x023	Minimum Refresh Recovery Delay Time ($t_{RFC4min}$), MSB	3
36	0x024	Minimum Four Activate Window Time (t_{FAWmin}), Most Significant Nibble	3
37	0x025	Minimum Four Activate Window Time (t_{FAWmin}), Least Significant Byte	3
38	0x026	Minimum Activate to Activate Delay Time (t_{RRD_smin}), different bank group	3
39	0x027	Minimum Activate to Activate Delay Time (t_{RRD_Lmin}), same bank group	3
40	0x28	Minimum CAS to CAS Delay Time (t_{CCD_Lmin}), same bank group	3
41	0x29	Upper Nibble for t_{WRmin}	3

NOTE 1 Number of SPD bytes written will typically be programmed as 384 bytes.

NOTE 2 Size of SPD device will typically be programmed as 512 bytes.

NOTE 3 From DDR4 SDRAM datasheet.

7 Address Map (Cont'd)**7.1 Block 0: Base Configuration and DRAM Parameters (Cont'd)****Table 11 — Block 0: Base Configuration and DRAM Parameters (Cont'd)**

Byte Number		Function Described	Notes
42	0x2A	Minimum Write Recovery Time (t_{WRmin})	3
43	0x2B	Upper nibbles for t_{WTRmin}	3
44	0x02C	Minimum Write to Read Time (t_{WTR_Smin}), different bank group	3
45	0x02D	Minimum Write to Read Time (t_{WTR_Lmin}), same bank group	3
46~59	0x02E~0x03B	Reserved -- must be coded as 0x00	
60~77	0x03C~0x04D	Connector to SDRAM Bit Mapping	
78~116	0x04E~0x074	Reserved -- must be coded as 0x00	
117	0x75	Fine Offset for Minimum CAS to CAS Delay Time (t_{CCD_Lmin}), same bank group	
118	0x76	Fine Offset for Minimum Activate to Activate Delay Time (t_{RRD_Lmin}), same bank group	3
119	0x77	Fine Offset for Minimum Activate to Activate Delay Time (t_{RRD_Smin}), different bank group	3
120	0x078	Fine Offset for Minimum Activate to Activate/Refresh Delay Time (t_{RCmin})	3
121	0x079	Fine Offset for Minimum Row Precharge Delay Time (t_{RPmin})	3
122	0x07A	Fine Offset for Minimum RAS to CAS Delay Time (t_{RCDmin})	3
123	0x07B	Fine Offset for Minimum CAS Latency Time (t_{AAmin})	3
124	0x07C	Fine Offset for SDRAM Maximum Cycle Time ($t_{CKAVGmax}$)	3
125	0x07D	Fine Offset for SDRAM Minimum Cycle Time ($t_{CKAVGmin}$)	3
126	0x07E	CRC for Base Configuration Section, Least Significant Byte	
127	0x07F	CRC for Base Configuration Section, Most Significant Byte	
NOTE 1 Number of SPD bytes written will typically be programmed as 384 bytes.			
NOTE 2 Size of SPD device will typically be programmed as 512 bytes.			
NOTE 3 From DDR4 SDRAM datasheet.			

7.2 Block 1, lower half: Module Specific Parameters

Bytes 128~191 (0x080~0x0BF) Parameters in this block are specific to the module type as selected by the contents of SPD Key Byte 3 bits 3~0. Refer to the appropriate subsection for detailed byte descriptions.

7.3 Block 1, upper half: Hybrid Memory Parameters

Bytes 192~255 (0x0C0~0x0FF) Parameters in this block are specific to the hybrid memory type as selected by the contents of SPD Key Byte 3 bits 7~4. Refer to the appropriate subsection for detailed byte descriptions. The CRC field in bytes 254~255 covers all of block 1, bytes 128~255.

7.4 Block 2, lower half: Extended Function Parameters

Bytes 256~319 (0x100~0x13F). Parameters in this block are used to define the extended functions available on Hybrid modules. Values in this block are based on indices in the NVDIMM annex, and the size of each function parameter block is unique to that function type.

7.5 Block 2, upper half: Manufacturing Information

Bytes 320~383 (0x140~0x17F) Table 12 details the location of each byte in this block.

Table 12 — Block 2, upper half: Manufacturing Information

Byte Number	Function Described	Notes
320	0x140	Module Manufacturer's ID Code, Least Significant Byte
321	0x141	Module Manufacturer's ID Code, Most Significant Byte
322	0x142	Module Manufacturing Location
323~324	0x143~0x144	Module Manufacturing Date
325~328	0x145~0x148	Module Serial Number
329~348	0x149~0x15C	Module Part Number
349	0x15D	Module Revision Code
350	0x15E	DRAM Manufacturer's ID Code, Least Significant Byte
351	0x15F	DRAM Manufacturer's ID Code, Most Significant Byte
352	0x160	DRAM Stepping
353~381	0x161~0x17D	Module Manufacturer's Specific Data
382~383	0x17E~0x17F	Reserved; must be coded as 0x00

7.6 Block 3: End User Programmable

Bytes 384~511 (0x180~0x1FF) Bytes in this block are reserved for use by end users.

8 Details of Each Byte

8.1 General Configuration Section: Bytes 0~127 (0x000~0x07F)

This section defines parameters that are common to all DDR4 module types.

8.1.1 Byte 0 (0x000): Number of Bytes Used / Number of Bytes in SPD Device

The least significant nibble of this byte, shown in Table 13, describes the total number of bytes used by the module manufacturer for the SPD data and any (optional) specific supplier information. The byte count includes the fields for all required and optional data. Bits 6~4 describe the total size of the serial memory used to hold the Serial Presence Detect data.

Table 13 — Byte 0 (0x000): Number of Bytes Used / Number of Bytes in SPD Device

Bit 7	Bits 6~4	Bits 3~0
Reserved	SPD Bytes Total	SPD Bytes Used
Reserved; must be coded as 0	Bit [6, 5, 4] : 000 = Undefined 001 = 256 010 = 512 All others reserved	Bit [3, 2, 1, 0] : 0000 = Undefined 0001 = 128 0010 = 256 0011 = 384 0100 = 512 All others reserved
NOTE Typical programming of bits 3~0 will be 0011 (384 bytes).		

8.1.2 Byte 1 (0x001): SPD Revision

This byte, shown in Table 14, describes the compatibility level of the encoding of the bytes contained in the SPD EEPROM, and the current collection of valid defined bytes. Software should examine the upper nibble (Encoding Level) to determine if it can correctly interpret the contents of the module SPD. The lower nibble (Additions Level) can optionally be used to determine which additional bytes or attribute bits have been defined; however, since any undefined additional byte must be encoded as 0x00, undefined

8.1 General Configuration Section: Bytes 0~127 (0x000~0x07F) (Cont'd)

8.1.2 Byte 1 (0x001): SPD Revision (Cont'd)

attribute bit must be defined as 0, software can safely detect additional bytes and use safe defaults if a zero encoding is read for these bytes.

Table 14 — Byte 1 (0x001): SPD Revision

Production Status	SPD Revision	Encoding Level				Additions Level				Hex
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Pre-production	Revision 0.0	0	0	0	0	0	0	0	0	00
	Revision 0.1	0	0	0	0	0	0	0	1	01
	...	.	.	.	.	.	.	.	.	.
	Revision 0.9	0	0	0	0	1	0	0	1	09
Production	Revision 1.0	0	0	0	1	0	0	0	0	10
	Revision 1.1	0	0	0	1	0	0	0	1	11
	...	.	.	.	.	.	.	.	.	...
Undefined	Undefined	1	1	1	1	1	1	1	1	FF

The Additions Level is never reduced even after an increment of the Encoding Level. For example, if the current SPD revision level were 1.2 and a change in Encoding Level were approved, the next revision level would be 2.2. If additions to revision 2.2 were approved, the next revision would be 2.3. Changes in the Encoding Level are extremely rare, however, since they can create incompatibilities with older systems.

The exceptions to the above rule are the SPD revision levels used during development prior to the Revision 1.0 release. Revisions 0.0 through 0.9 are used to indicate sequential pre-production SPD revision levels, however the first production release will be Revision 1.0.

This document defines the SPD contents for multiple families of DDR4 memory modules, with a separate section for each family that defines the bytes in SPD locations 128~255 (0x080~0x0FF). These module families and their respective sections are:

- Section 9.1: Unbuffered Memory Modules
- Section 9.2: Registered Memory Modules
- Section 9.3: Load Reduced Memory Modules
- Section 10.1: Non-Volatile Memory Modules

The SPD revision level for each module family type is independent. This allows changes to be made to the Registered DIMM annex, for example, without necessarily changing the revision of Unbuffered DIMMs. In this context, the SPD revision value corresponds to all SPD bytes for that DIMM type. It also means that over time, the revisions for each module type may vary. Note that changes to a DIMM specific annex does not affect the revisions of other module types, but changes in the General Section of the SPD affect all DIMM types. The example in Table 15 suggests a possible historical progression:

Table 15 — Hypothetical Historic Progression of SPD Revisions by DIMM Type

Event	UDIMM	RDIMM	LRDIMM
Initial SPD release	1.0	1.0	1.0
Addition in RDIMM Annex	1.0	1.1	1.0
Addition in LRDIMM Annex	1.0	1.1	1.1
Addition in LRDIMM Annex	1.0	1.1	1.2
Addition in General Section	1.1	1.2	1.3
Addition in UDIMM Annex	1.2	1.2	1.3

Table 15 — Hypothetical Historic Progression of SPD Revisions by DIMM Type (Cont'd)

Event	UDIMM	RDIMM	LRDIMM
Encoding change in LRDIMM Annex	1.2	1.2	2.3
Addition in LRDIMM Annex	1.2	1.2	2.4
Encoding change in General Section	2.2	2.2	3.4
Addition in RDIMM Annex	2.2	2.3	3.4

8.1.3 Byte 2 (0x002): Key Byte / DRAM Device Type

This byte, shown in Table 16, is the key byte used by the system BIOS to determine how to interpret all other bytes in the SPD EEPROM. The BIOS must check this byte first to ensure that the EEPROM data is interpreted correctly. Any DRAM or Module type that requires significant changes to the SPD format (beyond defining previously undefined bytes or bits) also requires a new entry in the key byte table below.

Table 16 — Byte 2 (0x002): Key Byte / DRAM Device Type

Line #	Hex	SDRAM / Module Type Corresponding to Key Byte
0	00	Reserved
1	01	Fast Page Mode
2	02	EDO
3	03	Pipelined Nibble
4	04	SDRAM
5	05	ROM
6	06	DDR SGRAM
7	07	DDR SDRAM
8	08	DDR2 SDRAM
9	09	DDR2 SDRAM FB-DIMM
10	0A	DDR2 SDRAM FB-DIMM PROBE
11	0B	DDR3 SDRAM
12	0C	DDR4 SDRAM
13	0D	Reserved
14	0E	DDR4E SDRAM
15	0F	LPDDR3 SDRAM
16	10	LPDDR4 SDRAM
17	11	LPDDR4X SDRAM
-	-	-
253	FD	Reserved
254	FE	Reserved
255	FF	Reserved

The DRAM Device Type byte defines an interface compatibility family more than it identifies a specific memory device. For example, where an unbuffered memory module may expose the devices directly to the edge connector contacts, a load reduced memory module presents the interfaces of registering clock drivers and data buffers to the edge connector contacts. There may be memory modules with completely different devices, such as non-volatile memories, however these may define themselves as a base memory type such as DDR4 SDRAM for interface compatibility with the memory controller. The memory controller must be aware of its capabilities when parsing the SPD.

8 Details of Each Byte (Cont'd)**8.1 General Configuration Section: Bytes 0~127 (0x000~0x07F) (Cont'd)****8.1.4 Byte 3 (0x003): Key Byte / Module Type**

This byte, shown in Table 17, is a Key Byte used to index the module specific section of the SPD from bytes 128~191 and bytes 192~255. Byte 3 bits 3~0 identifies the SDRAM memory module type.

Bits 7~4 identifies the architecture of the secondary memory type, if any, present on the module in addition to the base DRAM. These secondary hybrid architectures may apply to any Base Module Type. The index in bits 7~4 are used to index the hybrid memory architecture specific section of the SPD from bytes 192~255.

System software supporting hybrid memory modules should parse the DRAM attributes per the Base Module Type and combine the indexed information per the Hybrid Memory Type to determine the combined memory module capabilities.

Some modules may have no base memory, but will have only a secondary memory type. For example, a Flash-only memory module. These are classified as “hybrid” for the purposes of interpreting the SPD. Where base memory parameters apply to this class of hybrid module, these will be documented with those bytes in the base section. Extended parameters are encoded in SPD bytes 192~255.

Table 17 — Byte 3 (0x003): Key Byte / Module Type

Bit 7	Bits 6~4	Bits 3~0
Hybrid	Hybrid Media	Base Module Type
0 = Not hybrid (Module is DRAM only) 1 = Hybrid module (See bits 6~4 for hybrid type)	Bits [6, 5, 4]: 000 = Not hybrid 001 = NVDIMM Hybrid All other codes reserved	Bits [3, 2, 1, 0]: 0000 = Extended DIMM type, see byte 15 (0x00F) 0001 = RDIMM 0010 = UDIMM 0011 = SO-DIMM 0100 = LRDIMM 0101 = Mini-RDIMM 0110 = Mini-UDIMM 0111 = Reserved 1000 = 72b-SO-RDIMM 1001 = 72b-SO-UDIMM 1010 = Reserved 1011 = Reserved 1100 = 16b-SO-DIMM 1101 = 32b-SO-DIMM 1110 = Reserved 1111 = Reserved
Base Module Type Definitions: RDIMM: Registered Dual In-Line Memory Module UDIMM: Unbuffered Dual In-Line Memory Module SO-DIMM: Unbuffered Small Outline Dual In-Line Memory Module, 64-bit data bus LRDIMM: Load Reduced Dual In-Line Memory Module Mini-RDIMM: Mini Registered Dual In-Line Memory Module Mini-UDIMM: Mini Unbuffered Dual In-Line Memory Module 72b-SO-RDIMM: Small Outline Registered Dual In-Line Memory Module, 72-bit data bus 72b-SO-UDIMM: Small Outline Unbuffered Dual In-Line Memory Module, 72-bit data bus 16b-SO-DIMM: Small Outline Unbuffered Dual In-Line Memory Module, 16-bit data bus 32b-SO-DIMM: Small Outline Unbuffered Dual In-Line Memory Module, 32-bit data bus Hybrid Memory Type Definitions: NVDIMM: Non-Volatile Dual In-Line Memory Module, Hybrid module with a DRAM-style interface with one or more non-DRAM components for data storage		

Examples:

0x01 = RDIMM, no hybrid memory present

0x91 = RDIMM, NVDIMM hybrid memory present

0x94 = LRDIMM, NVDIMM hybrid memory present

8 Details of Each Byte (Cont'd)

8.1 General Configuration Section: Bytes 0~127 (0x000~0x07F) (Cont'd)

8.1.5 Byte 4 (0x004): SDRAM Density and Banks

This byte, shown in Table 18, defines the total density of the DDR4 SDRAM, in bits, and the number of internal banks and bank groups into which the memory array is divided. For multi-die stacks (DDP, QDP, or 3DS), this represents the capacity of each DRAM die in the stack. These values come from the DDR4 SDRAM data sheet.

Table 18 — Byte 4 (0x004): SDRAM Density and Banks

Bits 7~6	Bits 5~4	Bits 3~0
Bank Group Bits	Bank Address Bits ¹	Total SDRAM capacity per die, in megabits
Bits [7, 6]: 00 = 0 (no bank groups) 01 = 1 (2 bank groups) 10 = 2 (4 bank groups) 11 = reserved	Bit [5, 4] : 00 = 2 (4 banks) 01 = 3 (8 banks) All others reserved	Bit [3, 2, 1, 0] : 0000 = 256 Mb 0001 = 512 Mb 0010 = 1 Gb 0011 = 2 Gb 0100 = 4 Gb 0101 = 8 Gb 0110 = 16 Gb 0111 = 32 Gb 1000 = 12 Gb 1001 = 24 Gb All others reserved
NOTE 1 Bank Address Bits determine the number of banks in each Bank Group.		

8.1.6 Byte 5 (0x005): SDRAM Addressing

This byte, shown in Table 19, describes the row addressing and the column addressing in the SDRAM device. Bits 2~0 encode the number of column address bits, and bits 5~3 encode the number of row address bits. These values come from the DDR4 SDRAM data sheet.

Table 19 — Byte 5 (0x005): SDRAM Addressing

Bits 7~6	Bits 5~3	Bits 2~0
Reserved	Row Address Bits	Column Address Bits
Reserved; must be coded as 00	Bit [5, 4, 3] : 000 = 12 001 = 13 010 = 14 011 = 15 100 = 16 101 = 17 110 = 18 All others reserved	Bit [2, 1, 0] : 000 = 9 001 = 10 010 = 11 011 = 12 All others reserved

8 Details of Each Byte (Cont'd)

8.1 General Configuration Section: Bytes 0~127 (0x000~0x07F) (Cont'd)

8.1.7 Byte 6 (0x006): Primary SDRAM Package Type

This byte, shown in Table 20, describes the type of SDRAM devices on the module. For modules having asymmetrical assembly of multiple SDRAM package types, this byte defines the primary set of SDRAMs; Byte 10 defines the secondary set.

Table 20 — Byte 6 (0x006): Primary SDRAM Package Type

Bit 7	Bits 6~4	Bits 3~2	Bit 1~0
Primary SDRAM Package Type	Die Count	Reserved	Signal Loading ²
0 = Monolithic DRAM Device 1 = Non-Monolithic Device ¹	000 = Single die 001 = 2 die 010 = 3 die 011 = 4 die 100 = 5 die 101 = 6 die 110 = 7 die 111 = 8 die	00	00 = Not specified ³ 01 = Multi load stack 10 = Single load stack (3DS) 11 = Reserved
NOTE 1 This includes Dual Die, Quad Die, Multi-Die, 3DS, or physically stacked devices - anything that is outside the standard monolithic device. NOTE 2 Refers to loading on signals at the SDRAM balls. Loading on certain signals (CKE, ODT, etc.) per specification of device stacking as defined in JESD79-4. NOTE 3 Monolithic DRAM device coded as 00 in bits 1~0.			

Terminology: See Table 21.

Table 21 — Package Type Terminology

SDRAM Package Type	Abbreviation	Description	# Electrical Loads ...			
			... On data, mask, and strobe signals	... On address and command signals	... On control signals except CKE	... On CKE signals
Monolithic	SDP	Single die package	1	1	1	1
Multi-load stack	DDP	Dual Die Package	2	2	1	1
	QDP	Quad Die Package	4	4	1	2
Single load stack	2H 3DS	Two SDRAM die single load stack	1	1	1	1
	3H 3DS	Three SDRAM die single load stack	1	1	1	1
	4H 3DS	Four SDRAM die single load stack	1	1	1	1
	5H 3DS	Five SDRAM die single load stack	1	1	1	1
	6H 3DS	Six SDRAM die single load stack	1	1	1	1
	7H 3DS	Seven SDRAM die single load stack	1	1	1	1
	8H 3DS	Eight SDRAM die single load stack	1	1	1	1

8.1.8 Byte 7 (0x007): SDRAM Optional Features

This byte, shown in Table 22, defines support for certain SDRAM features. This value comes from the DDR4 SDRAM data sheet.

Table 22 — Byte 7 (0x007): SDRAM Optional Features

Bits 7~6	Bits 5~4	Bits 3~0
Reserved	Maximum Activate Window (tMAW)	Maximum Activate Count (MAC)
Reserved; must be coded as 00	Bits [5, 4]: 00 = 8192 * tREFI 01 = 4096 * tREFI 10 = 2048 * tREFI 11 = Reserved	Bits [3, 2, 1, 0]: 0000 = Untested MAC ¹ 0001 = 700 K 0010 = 600 K 0011 = 500 K 0100 = 400 K 0101 = 300 K 0110 = 200 K 0111 = Reserved 1000 = Unlimited MAC ² All others reserved
NOTE 1 Untested MAC means the device is not tested for tMAW and/or MAC; no particular value should be assumed. NOTE 2 Unlimited MAC means there is no restriction to the number of activates to a given row in a refresh period providing DRAM timing requirements such as tRCmin and refresh requirements are not violated.		

8.1.9 Byte 8 (0x008): SDRAM Thermal and Refresh Options

This byte, shown in Table 23, describes the module's supported operating temperature ranges and refresh options. These values come from the DDR4 SDRAM data sheet. Please refer to the DDR4 SDRAM data sheet (JESD79-4 or supplier data sheet) for a complete description of these options.

Table 23 — Byte 8 (0x008): SDRAM Thermal and Refresh Options

Bits 7~0
Reserved
Reserved; must be coded as 0x00

8.1.10 Byte 9 (0x009): Other SDRAM Optional Features

This byte, shown in Table 24, defines support for certain SDRAM features. This value comes from the DDR4 SDRAM data sheet.

Table 24 — Byte 9 (0x009): Other SDRAM Optional Features

Bits 7~6	Bit 5	Bits 4~0
Post Package Repair (PPR)	Soft PPR	Reserved
00: PPR not supported 01: Post package repair supported, one row per bank group 10: Reserved 11: Reserved	0 = Soft PPR not supported 1 = Soft PPR supported	Reserved; must be coded as 00000
NOTE If PPR is supported, Hard PPR is always supported by the device. Optional additional support for Soft PPR is indicated in bit 5.		

8 Details of Each Byte (Cont'd)

8.1 General Configuration Section: Bytes 0~127 (0x000~0x07F) (Cont'd)

8.1.11 Byte 10 (0x00A): Secondary SDRAM Package Type

For modules having asymmetrical assembly of multiple SDRAM package types, this byte, shown in Table 25, defines the secondary set of SDRAMs. For modules with symmetrical assembly, this byte must be coded as 0x00.

Table 25 — Byte 10 (0x00A): Secondary SDRAM Package Type

Bit 7	Bits 6~4	Bits 3~2	Bits 1~0
SDRAM Package Type	Die Count	DRAM Density Ratio ¹	Signal Loading
0 = Monolithic DRAM Device 1 = Non-Monolithic Device ¹	000 = Single die 001 = 2 die 010 = 3 die 011 = 4 die 100 = 5 die 101 = 6 die 110 = 7 die 111 = 8 die	00 = Rank 1 and 3 device densities are the same as rank 0 and 2 densities 01 = Rank 1 and 3 are one standard device density smaller than rank 0 and 2 10 = Rank 1 and 3 are two standard device densities smaller than rank 0 and 2 11 = Reserved	00 = Not specified 01 = Multi load stack 10 = Single load stack (3DS) 11 = Reserved
NOTE 1 When DRAM Density Ratio (bits 3~2) is non-zero, all SPD settings except timing parameters are specified for the higher density devices in rank 0 (e.g., density, row, column, etc.). SPD timing parameters are specified for the slowest device timings of any rank (e.g., tCK, tAA, tRFC, etc.). NOTE 2 See Byte 6 for packaging notes.			

DRAM Density Ratio Settings (Table 26):

Table 26 — DRAM Density Ratio Settings

SPD Byte 4, Bits 3~0	Ranks 0, 2 Device Density	SPD Byte 10, Bits 3~2	Ranks 1, 3 Device Density	SPD Byte 10, Bits 3~2	Ranks 1, 3 Device Density
0000	256 Mb	01	Not Defined	10	Not Defined
0001	512 Mb	01	256 Mb	10	Not Defined
0010	1 Gb	01	512 Mb	10	256 Mb
0011	2 Gb	01	1 Gb	10	512 Mb
0100	4 Gb	01	2 Gb	10	1 Gb
0101	8 Gb	01	4 Gb	10	2 Gb
0110	16 Gb	01	12 Gb	10	8 Gb
0111	32 Gb	01	24 Gb	10	16 Gb
1000	12 Gb	01	8 Gb	10	4 Gb
1001	24 Gb	01	16 Gb	10	12 Gb

8.1.12 Byte 11 (0x00B): Module Nominal Voltage, VDD

This byte, shown in Table 27, describes the Voltage Level for DRAM and other components on the module such as the register or memory buffer if applicable. Note that SPDs or thermal sensor components are on the VDDSPD supply and are not affected by this byte.

'Operable' is defined as the VDD voltage at which module operation is allowed using the performance values programmed in the SPD.

'Endurant' is defined as the VDD voltage at which the module may be powered without adversely affecting the life expectancy or

reliability. Operation is not supported at this voltage.

Table 27 — Byte 11 (0x00B): Module Nominal Voltage, VDD

Byte 11: Module Nominal Voltage, VDD		
Reserved	DRAM VDD 1.2 V	
Bits 7~2	Bit 1	Bit 0
Reserved; must be coded as 000000	0 = not endurant 1 = endurant	0 = not operable 1 = operable

Examples:

A value on bits 1~0 of 11 defines DRAM support of a nominal operable voltage of 1.2 V only.

8.1.13 Byte 12 (0x00C): Module Organization

This byte, shown in Table 28, describes the organization of the SDRAM module. Bits 2~0 encode the device width of the SDRAM devices. Bits 5~3 encode the number of package ranks on the module.

Table 28 — Byte 12 (0x00C): Module Organization

Bit 7	Bit 6	Bits 5~3	Bits 2~0
Reserved	Rank Mix	Number of Package Ranks per DIMM	SDRAM Device Width
Reserved; must be coded as 0	0 = Symmetrical 1 = Asymmetrical	Bit [5, 4, 3] : 000 = 1 Package Rank 001 = 2 Package Ranks 010 = 3 Package Ranks 011 = 4 Package Ranks 100 = 5 Package Ranks 101 = 6 Package Ranks 110 = 7 Package Ranks 111 = 8 Package Ranks	Bit [2, 1, 0] : 000 = 4 bits 001 = 8 bits 010 = 16 bits 011 = 32 bits All others reserved

“Package ranks per DIMM” refers to the collections of devices on the module sharing common chip select signals (across the data width of the DIMM), either from the edge connector for unbuffered modules or from the outputs of a registering clock driver for RDIMMs and LRDIMMs.

“Logical rank” refers the individually addressable die in a 3DS stack and has no meaning for monolithic or multi-load stacked SDRAMs, however, for the purposes of calculating the capacity of the module, one should treat monolithic and multi-load stack SDRAMs as having one logical rank per package rank.

8.1.13.1 Symmetrical Configurations

Byte 12 Bit 6 = 0 defines a memory module as being “symmetrical” where all DRAM devices on the module are the same type as described in Byte 6.

Symmetrical configurations are shown in Figure 1 and Table 29.

8.1.13 Byte 12 (0x00C): Module Organization (Cont'd)
8.1.13.1 Symmetrical Configurations (Cont'd)

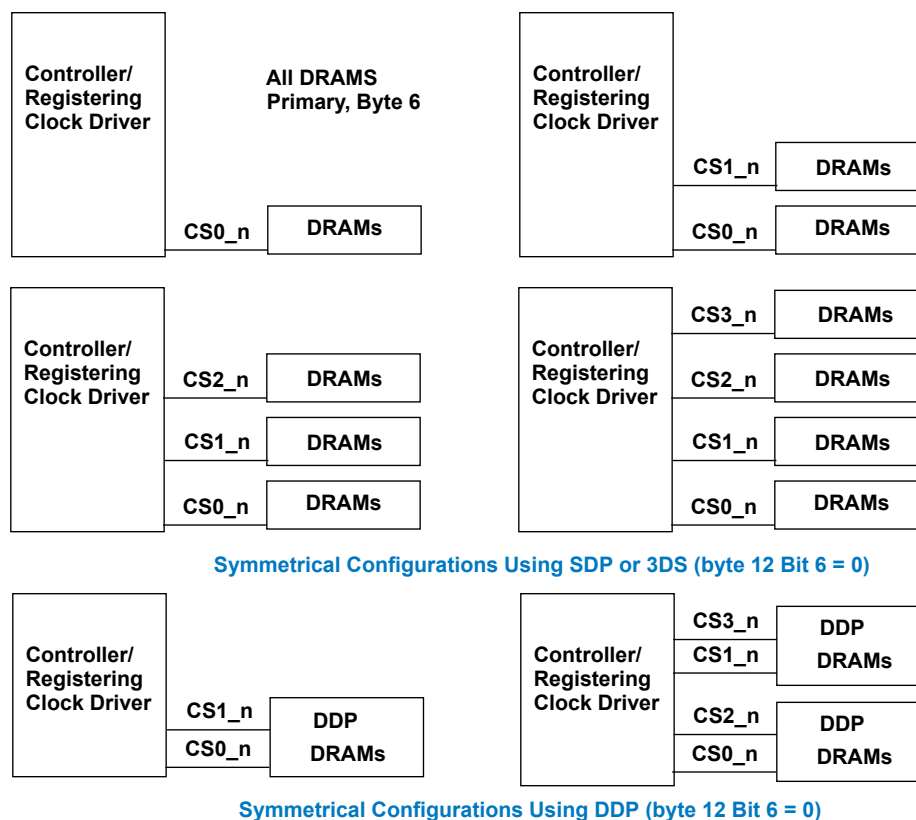


Figure 1 — Module Organization: Symmetrical Configurations

Table 29 — Rank Matrix for Symmetrical Modules (Byte 12 Bit 6 = 0)

SDRAM Package Type(s) (Byte 6)	# Package Ranks per DIMM	# Logical Ranks per Package Rank (for calculation only)	# Logical Ranks per DIMM
SDP	1	1	1
	2		2
	3		3
	4		4
DDP	2	1	2
	4		4
2H 3DS	1	2	2
	2		4
3H 3DS	1	3	3
	2		6
4H 3DS	1	4	4
	2		8
5H 3DS	1	5	5
	2		10
6H 3DS	1	6	6
	2		12

Table 29 — Rank Matrix for Symmetrical Modules (Byte 12 Bit 6 = 0) (Cont'd)

SDRAM Package Type(s) (Byte 6)	# Package Ranks per DIMM	# Logical Ranks per Package Rank (for calculation only)	# Logical Ranks per DIMM
7H 3DS	1	7	7
	2		14
8H 3DS	1	8	8
	2		16
Logical Ranks per DIMM = # Package Ranks per DIMM * # Logical Ranks per Package Rank Byte 10 coded as 0x00 for all symmetrical configurations			

8.1.13.2 Asymmetrical Configurations

Byte 12 Bit 6 = 1 describes support for modules having a asymmetrical DRAM capacities, for example a 3 package rank module comprised of a combination of SDP and DDP devices, or 3DS-based designs having different numbers of logical ranks in each package rank. When Byte 12 Bit 6 = 1, Byte 6 indicates the package type for the first collection of devices on the module and Byte 10 indicates the package type for the second collection of devices on the module.

Asymmetrical configurations are shown in Figure 2 and Table 30.

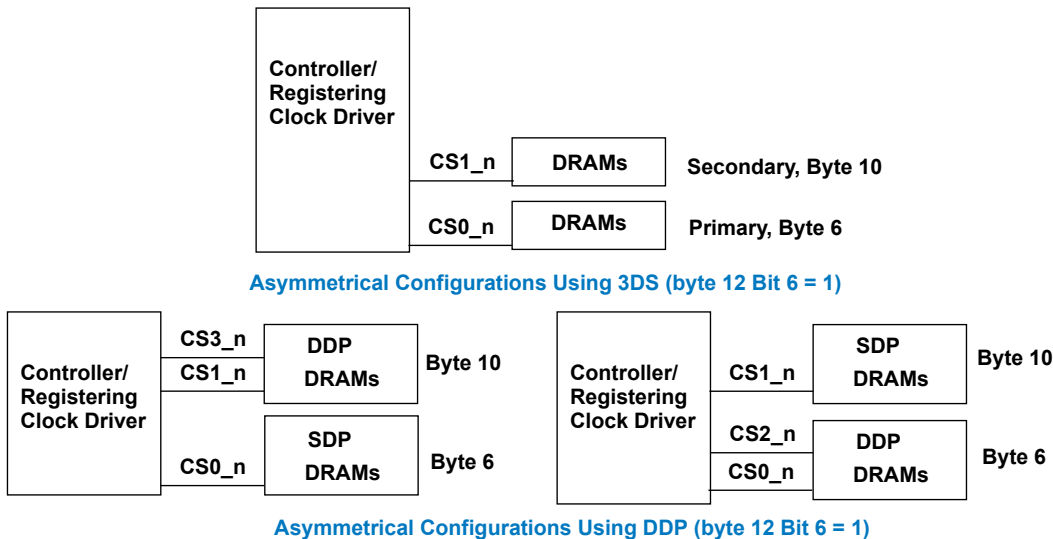


Figure 2 — Module Organization: Asymmetrical Configurations

8.1.13 Byte 12 (0x00C): Module Organization (Cont'd)

8.1.13.2 Asymmetrical Configurations (Cont'd)

Table 30 — Rank Mix for Asymmetrical Modules (Byte 12 Bit 6 = 1)

First SDRAM Package Type(s) (Byte 6) and Chip Select Alignment		Second SDRAM Package Type(s) (Byte 10) and Chip Select Alignment		# Package Ranks per DIMM	# Logical Ranks per DIMM
Non-3DS Asymmetrical Configurations					
SDP	CS0_n	DDP	CS1_n, CS3_n	3	3
DDP	CS0_n, CS2_n	SDP	CS1_n	3	3
3DS Asymmetrical Configurations					
2H 3DS	CS0_n	3H 3DS	CS1_n	2	5
2H 3DS	CS0_n	4H 3DS	CS1_n	2	6
...		...		...	...
7H 3DS	CS0_n	8H 3DS	CS1_n	2	15
Logical Ranks per DIMM = # Logical Ranks in Primary SDRAM type + # Logical Ranks in Secondary SDRAM type					

Examples: see Table 31

Table 31 — Module Configuration Examples

Byte 12, Bit 6	Byte 6	Byte 10	Byte 12, Bits 5~3, 2~0	Logical Ranks per DIMM	Module Configuration
Symmetry	Primary	Secondary	Ranks, Width		
0 = Sym.	0x00	0x00	000, 001	1	1 Package Rank x8, Monolithic
0 = Sym.	0x00	0x00	001, 001	2	2 Package Ranks x8, Monolithic
0 = Sym.	0x91	0x00	001, 000	2	2 Package Ranks x4, DDP
0 = Sym.	0xB2	0x00	000, 000	4	1 Package Rank x4, 4H 3DS
0 = Sym.	0xB2	0x00	001, 000	8	2 Package Ranks x4, 4H 3DS
1 = Asym.	0x00	0x91	010, 000	3	3 Package Ranks x4, SDP + DDP
1 = Asym.	0xF2	0xE2	001, 000	15	2 Package Ranks x4, 8H 3DS + 7H 3DS

8.1.14 Byte 13 (0x00D): Module Memory Bus Width

This byte, shown in Table 32, describes the width of the SDRAM memory bus on the module. Bits 2~0 encode the primary bus width. Bits 4~3 encode the bus extensions such as parity or ECC.

Table 32 — Byte 13 (0x00D): Module Memory Bus Width

Bits 7~5	Bits 4~3	Bits 2~0
Reserved	Bus width extension, in bits	Primary bus width, in bits
Reserved; must be coded as 000	Bit [4, 3] : 000 = 0 bits (no extension) 001 = 8 bits All others reserved	Bit [2, 1, 0] : 000 = 8 bits 001 = 16 bits 010 = 32 bits 011 = 64 bits All others reserved

Examples:

- 64 bit primary bus, no parity or ECC (64 bits total width): xxx 000 011
- 64 bit primary bus, with 8 bit ECC (72 bits total width): xxx 001 011

8.1.14.1 Calculating Module DRAM Capacity

The total memory capacity of the DRAM on the module may be calculated from SPD values. For example, to calculate the total capacity, in megabytes or gigabytes, of a typical module:

$$\text{Total} = \text{SDRAM Capacity} \div 8 * \text{Primary Bus Width} \div \text{SDRAM Width} * \text{Logical Ranks per DIMM}$$

where:

- SDRAM Capacity = SPD byte 4 bits 3~0
- Primary Bus Width = SPD byte 13 bits 2~0
- SDRAM Width = SPD byte 12 bits 2~0
- Logical Ranks per DIMM =

for SDP, DDP, QDP: = SPD byte 12 bits 5~3

for 3DS: = SPD byte 12 bits 5~3 times SPD byte 6 bits 6~4

Examples:

Two package ranks per DIMM using monolithic SDRAMs, 2 Gb per die, with x4 organization on a module with a 64 bit primary bus:

- Total = 2 Gb ÷ 8 * 64 ÷ 4 * 2 * 1 = 8 GB

Two package ranks per DIMM using DDP SDRAMs, 4 Gb per die, with x4 organization on a module with a 64 bit primary bus:

- Total = 4 Gb ÷ 8 * 64 ÷ 4 * 2 = 16 GB

Two package ranks per DIMM using 4H 3DS SDRAMs, 2 Gb per die, having a x8 organization on a module with a 64 bit primary bus:

- Total = 2 Gb ÷ 8 * 64 ÷ 8 * 2 * 4 = 16 GB

Commonly, parity or ECC are not counted in total module capacity, though they can also be included by adding the bus width extension in SPD byte 13 bits 4~3 to the primary bus width in the previous examples.

8.1.15 Byte 14 (0x00E): Module Thermal Sensor

This byte, shown in Table 33, describes the module's supported thermal options.

Table 33 — Byte 14 (0x00E): Module Thermal Sensor

Bit 7	Bits 6~0
Thermal Sensor ¹	Reserved
0 = Thermal sensor not incorporated onto this assembly 1 = Thermal sensor incorporated onto this assembly	0 = Undefined All others reserved
NOTE 1 Thermal sensor compliant with TSE2004av specifications.	

8 Details of Each Byte (Cont'd)

8.1 General Configuration Section: Bytes 0~127 (0x000~0x07F) (Cont'd)

8.1.16 Byte 15 (0x00F): Extended Module Type

This byte, shown in Table 34, extends the module type field of byte 3. Used when byte 3 bits 3~0 = 0000.

Table 34 — Byte 15 (0x00F): Extended Module Type

Bit 7~4	Bits 3~0
Reserved	Extended Base Module Type
Reserved; must be coded as 0000	Bits [3, 2, 1, 0]: 0000 = Reserved; must be coded as 0000 ... 1111 = Reserved; must be coded as 0000

8.1.17 Byte 16 (0x010):

Reserved, must be coded as 0x00

8.1.18 Byte 17 (0x011): Timebases

This byte, shown in Table 35, defines a value in picoseconds that represents the fundamental timebase for fine grain and medium grain timing calculations. These values are used as a multiplier for formulating subsequent timing parameters.

Table 35 — Byte 17 (0x011): Timebases

Bits 7~4	Bits 3~2	Bits 1~0
Reserved	Medium Timebase (MTB)	Fine Timebase (FTB)
Reserved; must be coded as 0000	Bits [3, 2]: 00 = 125 ps All others reserved	Bits [0, 1]: 00 = 1 ps All others reserved

8.1.18.1 Relating the MTB and FTB

When a timing value tXX cannot be expressed by an integer number of MTB units, the SPD must be encoded using both the MTB and FTB. The Fine Offsets are encoded using a two's complement value which, when multiplied by the FTB yields a positive or negative correction factor. Typically, for safety and for legacy compatibility, the MTB portion is rounded UP and the FTB correction is a negative value. The general algorithm for programming SPD values is:

```

Temp_val = tXX / MTB           // Calculate as real number
Remainder = Temp_val modulo 1   // Determine if integer # MTBs
Fine_Correction = 1 - Remainder // If needed, what correction
if (Remainder == 0) then        // Integer # MTBs?
    tXX(MTB) = Temp_val         // Convert to integer
    tXX(FTB) = 0                // No correction needed
else                             // Needs correction
    tXX(MTB) = ceiling (Temp_val) // Round up for safety in legacy systems
    tXX(FTB) = Fine_Correction * MTB / FTB // Correction is negative offset
endif

```

To recalculate the value of tXX from the SPD values, a general formula BIOSes may use is:

$$tXX = tXX(MTB) * MTB + tXX(FTB) * FTB$$

Example (see Table 36):

Table 36 — $t_{CKAVGmin}$ SPD Calculations Using MTB and FTB

Speed Bin	t _{CKAVG} min Value Decimal	SPD byte 18 Decimal (Hexadecimal)	SPD byte 125 Decimal (Hexadecimal)
DDR4-1866	1.071 ns	9 (0x09)	-54 (0xCA)
	=	(9 * 0.125) + (-54 * 0.001)	
NOTE Examples assume MTB of 0.125 ns and FTB of 0.001 ns			

Timing parameters using both MTB and FTB are shown in Table 37:

Table 37 — Timing parameters using both MTB and FTB

Parameter	MTB Byte(s)	FTB Byte
$t_{CKAVGmin}$	18 (0x012)	125 (0x07D)
$t_{CKAVGmax}$	19 (0x013)	124 (0x07C)
t_{AAmin}	24 (0x018)	123 (0x07B)
t_{RCDmin}	25 (0x019)	122 (0x07A)
t_{RPmin}	26 (0x01A)	121 (0x079)
t_{RCmin}	27, 29 (0x01B, 0x01D)	120 (0x078)
t_{RRD_Smin}	39 (0x027)	119 (0x077)
t_{RRD_Lmin}	40 (0x028)	118 (0x076)

The encoding of two's complement fine timebase offsets are shown in Table 38:

Table 38 — eEncoding of Two's Complement Fine Timebase Offsets

Coding		Value (Dec)	Value (Hex)	FTB Timebase
Bit 7	Bits 6~0			1 ps
0	1111111	+127	7F	+127 ps
0	1111110	+126	7E	+126 ps
...	...	...	...	
0	0000001	+1	01	+1 ps
0	0000000	0	00	0
1	1111111	-1	FF	-1 ps
1	1111110	-2	FE	-2 ps
...	...	...	...	
1	0000000	-128	80	-128 ps

8.1.18.2 Rounding Algorithms

Software algorithms for calculation of timing parameters are subject to rounding errors from many sources. For example, a system may use a memory clock with a nominal frequency of 933.33... MHz, or a clock period of 1.0714... ns. Similarly, a system with a memory clock frequency of 1066.66... MHz yields mathematically a clock period of 0.9375... ns. In most cases, it is impossible to express all digits after the decimal point exactly, and rounding must be done because the SPD establishes a minimum granularity for timing parameters of 1 ps.

Rules for rounding must be defined to allow optimization of memory module performance without violating device parameters. In some cases, rounding errors cause an unnecessary clock of delay, affecting system performance. These rounding algorithms rely on results that are within correction factors on device testing and specification to avoid losing performance due to rounding errors.

8.1.18 Byte 17 (0x011): Timebases (Cont'd)

8.1.18.2 Rounding Algorithms (Cont'd)

These rules are:

- Clock periods such as $t_{CKAVGmin}$ are defined to 1 ps of accuracy; for example, 0.9375... ns is defined as 937 ps and 1.0714... ns is defined as 1071 ps.
- Using real math, parameters like t_{AAmin} , t_{RCDmin} , etc. which are programmed in systems in numbers of clocks (nCK) but expressed in the SPD in units of time (in ns) are divided by the clock period (in ns) yielding a ratio of clock units (nCK), a correction factor of 2.5% is subtracted, then the result is set to the next higher integer number of clocks:

$$nCK = \text{ceiling} \left[\left(\frac{\text{parameter_in_ns}}{\text{application_t}_{CK_in_ns}} \right) - 0.025 \right]$$

- Alternatively, programmers may prefer to use integer math instead of real math by expressing timing in ps, scaling the desired parameter value by 1000, dividing by the application clock period, adding an inverse correction factor of 97.4%, dividing the result by 1000, then truncating down to the next lower integer value:

$$nCK = \text{truncate} \left[\frac{\left(\frac{\text{parameter_in_ps} \times 1000}{\text{application_t}_{CK_in_ps}} \right) + 974}{1000} \right]$$

- Either algorithm should yield identical results. In case of any conflict between the two methodologies, the integer method shall prevail.

Examples

Example 1, using REAL math to convert t_{AAmin} from ns to nCK (see Table 39):

```
// This algorithm subtracts 2.5% correction factor and rounds up to next integer value
real MTB, FTB, TaaMin, Correction, ClockPeriod, TempNck;
int TaaInNck;

MTB = 0.125; // From SPD[17], in ns
FTB = 0.001; // From SPD[17], in ns
TaaMin = (SPD[18] * MTB) + (SPD[123] * FTB); // Calculate tAAmin in ns (FTB is negative offset)
Correction = 0.025; // 2.5%, per rounding algorithm
ClockPeriod = ApplicationTck; // Frequency (clock period) is application dependent
TempNck = TaaMin / ClockPeriod; // Initial calculation of nCK
TempNck = TempNck - Correction; // Subtract correction factor from nCK
TaaInNck = (int)ceiling(TempNck); // Ceiling to next higher integer value
```

Table 39 — Example 1, using REAL math to convert t_{RCDmin} from ns to nCK

DDR4-2666W Device Operating at Standard Application Data Rates (Full & Downbinned)						
Timing Parameter: $t_{AAmin} = 15.0$ ns						
Application Speed Grade	Device t_{AA}	Application t_{CK}	Device $t_{AA} \div$ Application t_{CK}	2.5% Correction	$t_{AA} / t_{CK} -$ Correction	Ceiling Result
	ns	ns	ratio (real)	(real)	ratio (real)	nCK (integer)
2666	15.000	0.750	20.0	0.025	19.975	20
2400	15.000	0.833	18.0072	0.025	17.9822	18
2133	15.000	0.937	16.00854	0.025	15.9835	16
1866	15.000	1.071	14.0056	0.025	13.9806	14

Table 39 — Example 1, using REAL math to convert t_{RCDmin} from ns to nCK (Cont'd)

DDR4-2666W Device Operating at Standard Application Data Rates (Full & Downbinned) Timing Parameter: $t_{AAmin} = 15.0$ ns						
Application Speed Grade	Device t_{AA}	Application t_{CK}	Device $t_{AA} \div$ Application t_{CK}	2.5% Correction	t_{AA} / t_{CK} — Correction	Ceiling Result
	ns	ns	ratio (real)	(real)	ratio (real)	nCK (integer)
1600	15.000	1.250	12.0	0.025	11.975	12
NOTE Roundup values for bins 2400, 2133, and 1866 would have lost one clock of performance without the application of the rounding algorithm. For example, a DDR4-2666W device running at DDR4-2400 data rates would have been required to set t_{AA} to 19 clocks without correction, but with correction t_{AA} may be safely programmed to 18 clocks.						

Example 2, using INT math to convert t_{AAmin} from ns to nCK (see Table 40):

```
// This algorithm uses adds 97.4% of a clock and truncates down to the next lower integer value
int MTB, FTB, TaaMin, ClockPeriod, TempNck, TaaInNck;

MTB = 125; // From SPD[17], in ps
FTB = 1; // From SPD[17], in ps
TaaMin = (SPD[18] * MTB) + (SPD[123] * FTB); // Calculate tAAmin in ps (FTB is negative offset)
ClockPeriod = ApplicationTckInPs; // Clock period is application specific
TempNck = (TaaMin * 1000) / ApplicationTckInPs; // Preliminary nCK calculation, scaled by 1000
TaaInNck = TempNck + 974; // Apply inverse of 2.5% correction factor
TaaInNck = (int)(TempNck / 1000); // Truncate to next lower integer
```

Table 40 — Example 2, using INT math to convert t_{RCDmin} from ns to nCK

DDR4-2666W Device Operating at Standard Application Data Rates (Full & Downbinned) Timing Parameter: $t_{AAmin} = 15.0$ ns (15000 ps)					
Application Speed Grade	Device t_{AA}	Application t_{CK}	(Device $t_{AA} * 1000$) Application t_{CK}	Add Inverse Correction	Truncate Corrected nCK / 1000
	ps	ps	Scaled nCK	Scaled nCK	nCK (integer)
2666	15000	750	20000	20974	20
2400	15000	833	18007	18981	18
2133	15000	937	16008	16982	16
1866	15000	1071	14005	14979	14
1600	15000	1250	12000	12974	12

8.1.19 Byte 18 (0x012): SDRAM Minimum Cycle Time ($t_{CKAVGmin}$)

This byte, shown in Table 41, defines the minimum cycle time for the SDRAM module, in medium timebase (MTB) units. This number applies to all applicable components on the module. This byte applies to SDRAM and support components as well as the overall capability of the DIMM. This value comes from the DDR4 SDRAM and support component data sheets.

Table 41 — Byte 18 (0x012): SDRAM Minimum Cycle Time ($t_{CKAVGmin}$)

Bits 7~0
Minimum SDRAM Cycle Time ($t_{CKAVGmin}$) MTB Units
Values defined from 1 to 255

If $t_{CKAVGmin}$ cannot be divided evenly by the MTB, this byte must be rounded up to the next larger integer and the Fine Offset for $t_{CKAVGmin}$ (SPD byte 125) used for correction to get the actual value.