NEC

MOS INTEGRATED CIRCUIT

μPD488170L

18M-BIT Rambus DRAM
1M-WORD X 9-BIT X 2-BANK

Description

The 18-Megabit Rambus™ DRAM (RDRAM™) is an extremely-high-speed CMOS DRAM organized as 2M words by 9 bits and capable of bursting up to 256 bytes of data at 2 ns per byte. The use of Rambus Signaling Logic (RSL) technology makes this 500 MHz transfer rate achievable while using conventional system and board design methodologies. Low latency is attained by using the RDRAM's large internal sense amplifier arrays as high speed caches.

RDRAMs are general purpose high-performance memory devices suitable for use in a broad range of applications including main memory, graphics, video, and any other application where high-performance and low cost are required.

Detailed information about product features and specifications can be found in the following document. Please make sure to read this document before starting design.

Rambus DRAM user's manual (Reference Manual)

Features

- Rambus Interface
- 500 MB/sec peak transfer rate per RDRAM
- RSL interface
- Synchronous protocol for fast block-oriented transfers
- Direct connection to Rambus ASICs, MPUs, and Peripherals
- 40 ns from start of read request to first byte; 2 ns per byte thereafter
- Features for graphics include random-access mode, write-per-bit and mask-per-bit operations
- Dual 2K-Byte sense amplifiers act as caches for low latency accesses
- Multiple power-saving modes
- On-chip registers for flexible addressing and timing
- Low pincount-only 15 active signals
- Standardized pinout across multiple generations of RDRAMs
- 3.3 volt operation

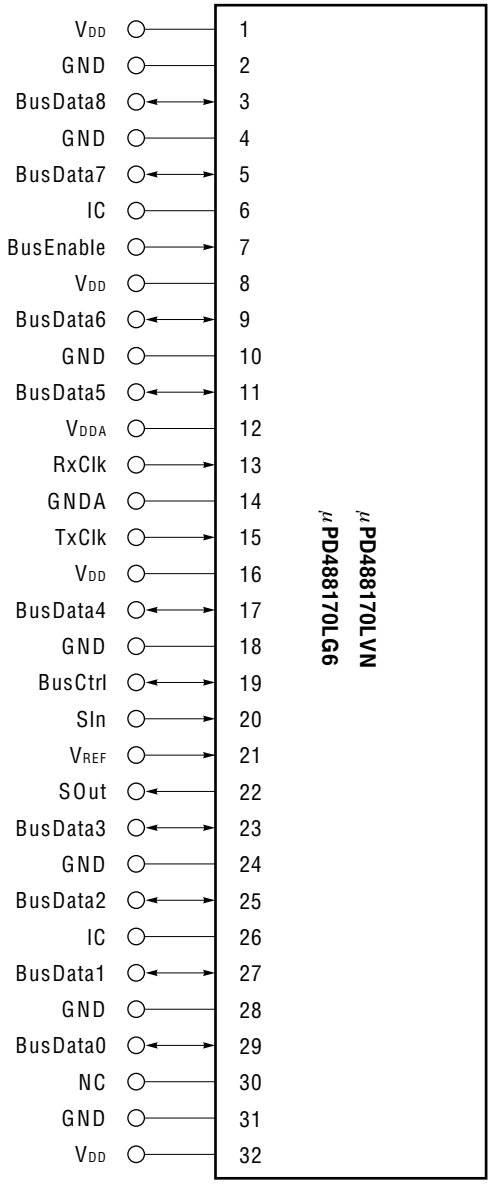
Ordering Information

Part Number	Clock Frequency	Operation Voltage	Package
μPD488170LVN-A50-9	250MHz	3.3±0.15 V	32-pin plastic SVP (11 × 25)
μPD488170LVN-A45-9	225MHz	3.3±0.15 V	32-pin plastic SVP (11 × 25)
μPD488170LG6-A50	250MHz	3.3±0.15 V	72/36-pin plastic SSOP type
μPD488170LG6-A45	225MHz	3.3±0.15 V	72/36-pin plastic SSOP type

The information in this document is subject to change without notice.

Pin Configuration (Marking Side)

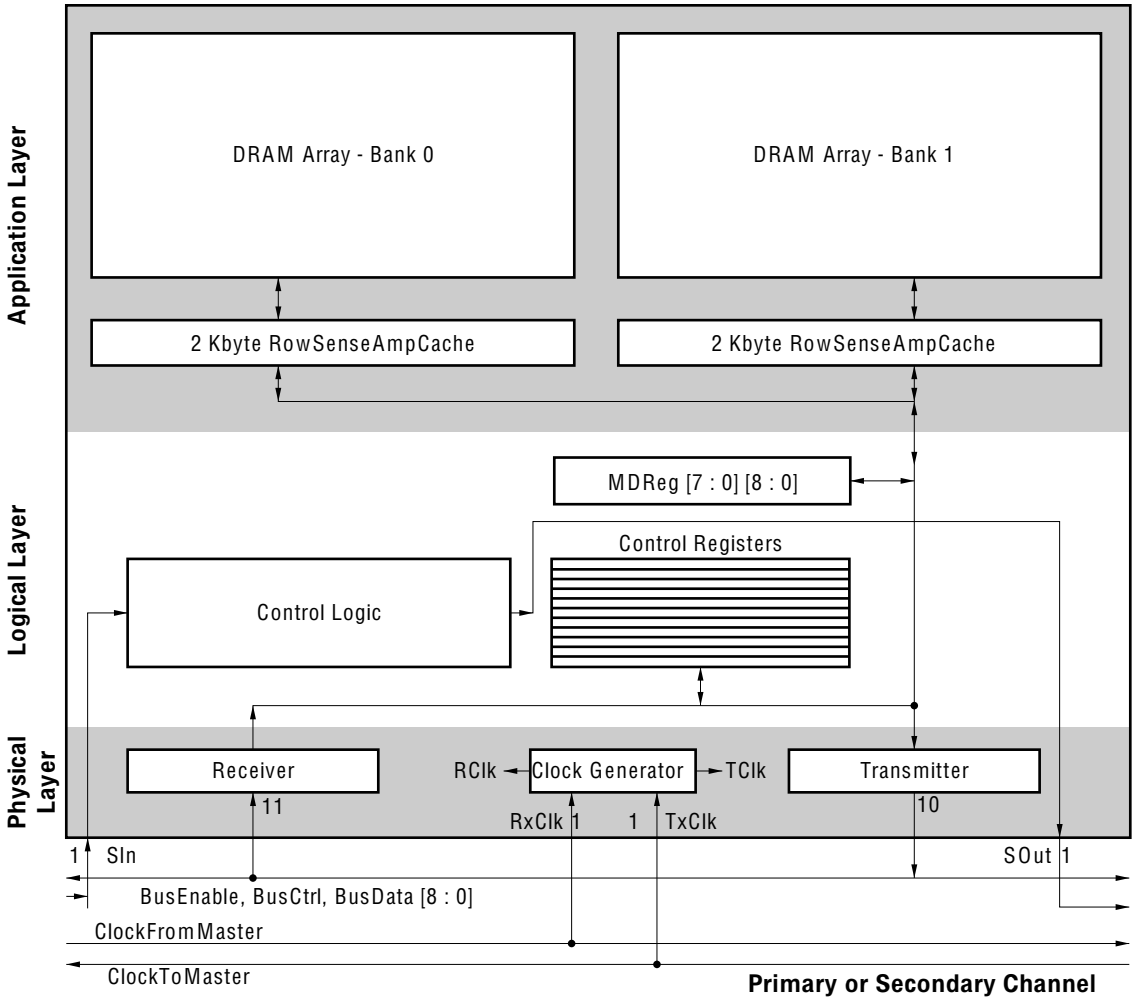
32-pin plastic SVP (11 × 25)
72/36-pin plastic SSOP type



BusData 0 - BusData 8	: Bus Data (Input/Output)
RxClk	: Receive Clock (Input)
TxClk	: Transmit Clock (Input)
V_{REF}	: Logic Threshold Voltage (Input)
BusCtrl	: BusCtrl (Input/Output)
BusEnable	: BusEnable (Input)
V_{DD}, V_{DDA}	: Power Supply
GND, GNDA	: Ground
SIn	: Serial Input (Input)
SOut	: Serial Output (Output)
NC	: No Connection IC
IC[Note]	: Internal Connection

Note Leave this pin unconnected.

Block Diagram



CONTENTS (1/2)

CONTENTS (2/2)

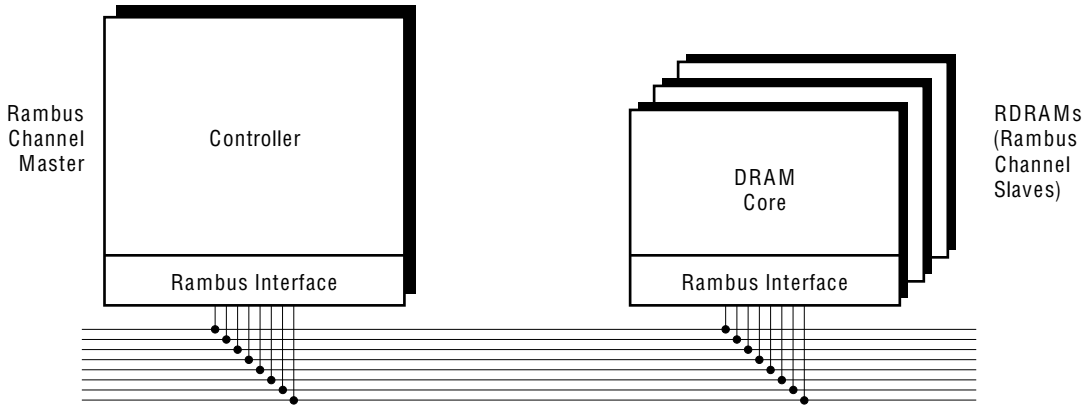
1. Pin Function

Signal	I/O	Description
BusData [8:0]	I/O	Signal lines for request, write data, and read data packets. The request packet contains the address, operation codes, and the count of the bytes to be transferred. This is a low-swing, active-low signal referenced to V_{REF}.
RxClk	I	Receive clock. Incoming request and write data packets are aligned to this clock. This is a low-swing, active-low signal referenced to V_{REF}.
TxClk	I	Transmit clock. Outgoing acknowledge and read packets are aligned with this clock. This is a low-swing, active-low signal referenced to V_{REF}.
V_{REF}	I	Logic threshold voltage for low swing signals.
BusCtrl	I/O	Control signal to frame packets, to transmit part of the operation code, and to acknowledge requests. Low-swing, active-low signal referenced to V_{REF}.
BusEnable	I	Control signal to enable the bus. Long assertions of this signal will reset all devices on the Channel. This is a low-swing, active-low signal referenced to V_{REF}.
V_{DD}, V_{DDA}	—	+3.3 V power supply. V_{DDA} is a separate analog supply.
GND, GNDA	—	Circuit ground. GNDA is a separate analog ground.
SIn	I	Initialization daisy chain input. TTL levels. Active high.
SOut	O	Initialization daisy chain output. TTL levels. Active high.

2. Rambus System Overview

A typical Rambus memory system has three main elements: the Rambus Channel, the RDRAMs, and a Rambus Interface on a controller. The logical representation of this is shown in the following figure.

Figure 2-1. Logical Representation

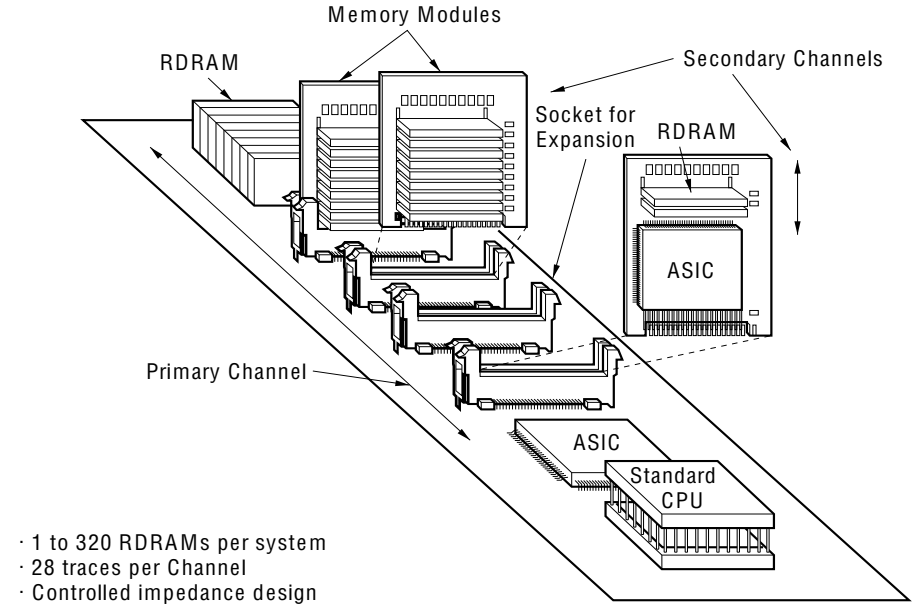


The Rambus Channel is a synchronous, high-speed, byte-wide bus that is used to directly connect Rambus devices together. Using only 13 high-speed signals, the Channel carries all address, data, and control information to and from devices through the use of a high level block-oriented protocol.

The Rambus Interface is implemented on both master and slave devices. Rambus masters are the only devices capable of generating transaction requests and can be ASIC devices, memory controllers, graphics engines, peripheral chips, or microprocessors. RDRAMs are slave devices and only respond to requests from master devices.

The following figure shows a typical physical implementation of a Rambus system. It includes a controller ASIC that acts as the Channel master and a base set of RDRAMs soldered directly to the board. An RSocket™ is included on the Channel for memory upgrade using RModule™ expansion cards.

Figure 2-2. A Rambus System Example

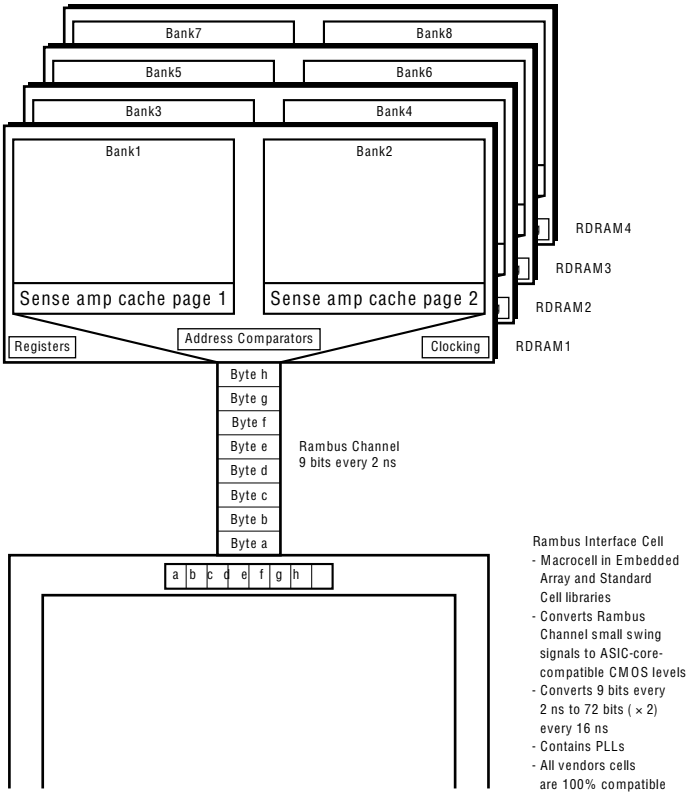


3. Rambus Signaling Logic

RSL technology is the key to attaining the high data rates available in Rambus systems. By employing high quality transmission lines, current-mode drivers, low capacitive loading, low-voltage signaling, and precise clocking, systems reliably transfer data at 2 nanosecond intervals on a Rambus Channel with signal quality that is superior to TTL or GTL-based interfaces.

All Rambus Interfaces incorporate special logic to convert signals from RSL to CMOS levels for internal use. In addition, these interfaces convert the Channel data rate of one byte every 2 nanoseconds to an internal data rate of 8 bytes every 16 nanoseconds as shown in the following figure. Although the bandwidth remains the same, the use of a wide internal bus eases internal timing requirements for chip designers.

Figure 3-1. Converting the Channel Data Rate



4. Register Space Map

The following table summarizes the registers included in all 18M RDRAMs.

Table 4-1. Registers Space Map

Register Name	Adr[20:10]	Adr[9:2]	Register Number
Device Type[3:0][8:0]	××...××	00000000	0
DeviceId[3:0][8:0]	××...××	00000001	1
Delay[3:0][8:0]	××...××	00000010	2
Mode[3:0][8:0]	××...××	00000011	3
RefRow[3:0][8:0]	××...××	00000101	5
RasInterval[3:0][8:0]	××...××	00000110	6
MinInterval[3:0][8:0]	××...××	00000111	7
AddressSelect[3:0][8:0]	××...××	00001000	8
DeviceManufacturer[3:0][8:0]	××...××	00001001	9
Row[3:0][8:0]	××...××	10000000	128

(1) Device Type Register

This register specifies RDRAM configuration and size.

(2) DeviceId Register

This register specifies RDRAM base address.

(3) Delay Register

This register specifies RDRAM programmable CAS delay values.

(4) Mode Register

This register specifies RDRAM programmable output drive current.

(5) RefRow Register

This register specifies RDRAM refresh row and bank address.

The RefRow register contains read-write fields. It is used to keep track of the bank and row being refreshed. Normally this register is only read or written for testing purposes. The fields are aliased in the following way:

RowField[7:1] equals RefRow[0][7:1]

RowField[9:8] equals RefRow[2][1:0]

BankField[3] equals RefRow[1][3]

(6) RasInterval Register

This register specifies RDRAM programmable RAS delay values. The RasInterval Register contains four write-only fields. When a rowmiss occurs, or when a row is being refreshed during a burst refresh operation, it is necessary for the control logic of an RDRAM to count the appropriate number of clock cycles (t_{CYCLE}) for four intervals. This is done with a counter which is loaded successively with three values from the RasInterval Register.

(7) MinInterval Register

This register specifies RDRAM refresh control.

This register provides the minimum values for three time intervals for framing packets.

The time intervals are specified in clock cycle (t_{CYCLE}) units.

Caution MinInterval Register[3][2] = 0 is necessary. Because, 18M RDRAM don't support Power Down request.

(8) AddressSelect Register

This register specifies RDRAM address mapping.

(9) DeviceManufacturer Register

This register specifies RDRAM manufacturer information.

This register specifies the manufacturer of the device. Additional bits are available for manufacturer specific information, e.g. stepping or revision numbers.

(10) Row Register

This register specifies RDRAM current sensed row in each bank.

The detailed functional description is provided in RDRAM Reference Manual.

5. Packet Formation

5.1 Packet Summary

The following table summarizes the transmit/receive functionality of the two RDRAM types for the different packet classes.

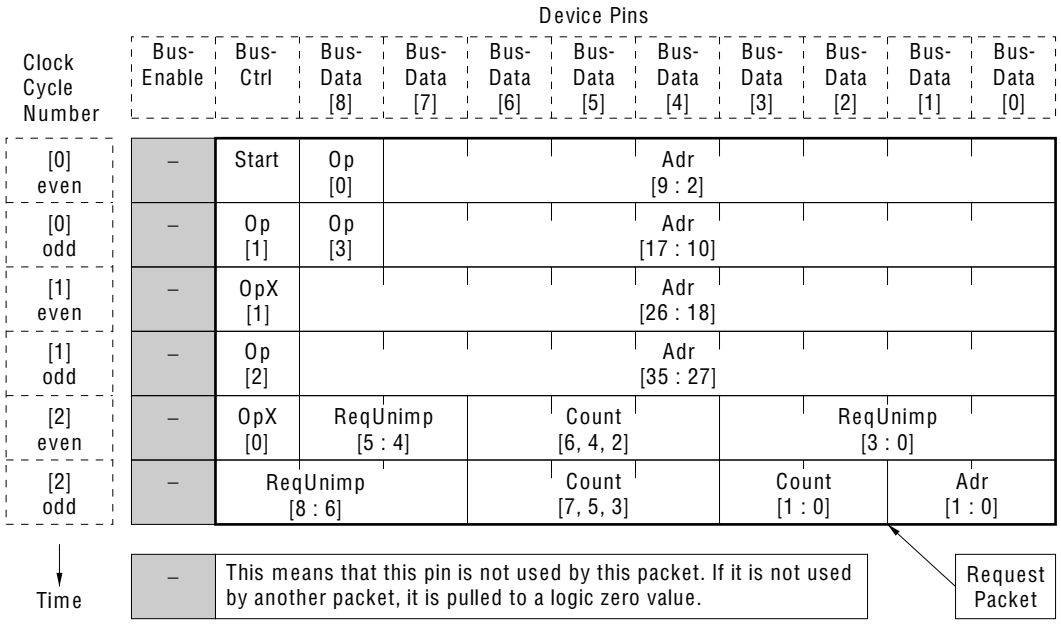
Table 5-1. Transmitting/Receiving Devices for Packet Types

Packet Type	Initiating Devices	μPD488170L
Request Packet	Transmit	Receive
Acknowledge Packet	Receive	Transmit
Read Data Packet	Receive	Transmit
Write Data Packet	Transmit	Receive
Serial Address Packet	Transmit	Receive
Serial Control Packet	Transmit	Receive
Serial Mode Packet	Transmit	Receive

5.2 Request Packet

The request packet format is shown in the following figure.

Figure 5-1. Request Packet Format



The vertical axis in all packet figures in the following sections shows time in units of clock cycles, with each clock cycle broken into even and odd bus ticks. The timing is relative, measured from the beginning of the packet.

5.2.1 Start Field

A device should start framing a request packet when it sees this bit asserted to a logical one and it is not looking for an acknowledge packet nor framing an earlier request packet.

5.2.2 Op[3:0], OpX[1:0] Fields

The Op and OpX fields are summarized in the following table.

Table 5-2. Op[3:0] and OpX[1:0] Fields - Command Encodings

Op[3:0]	OpX[1:0] = 00	OpX[1:0] = 01	OpX[1:0] = 10	OpX[1:0] = 11
0000	Rseq	Rnsq	Rsrv	Rsrv
0001	Rsrv	Rsrv	Rsrv	Rsrv
0010	Rsrv	Rsrv	Rsrv	Rsrv
0011	Rsrv	Rsrv	Rsrv	Rsrv
0100	WseqNpb	WseqDpb	WseqBpb	WseqMpb
0101	Rsrv	Rsrv	Rsrv	Rsrv
0110	Rreg	Rsrv	Rsrv	Rsrv
0111	Wreg	Rsrv	Rsrv	Rsrv
1000	WnsqNpb	WnsqDpb	WnsqBpb	WnsqMpb
1001	Rsrv	Rsrv	Rsrv	Rsrv
1010	Rsrv	Rsrv	Rsrv	Rsrv
1011	Rsrv	Rsrv	Rsrv	Rsrv
1100	WbnsNpb	WbnsDpb	Rsrv	WbnsMpb
1101	Rsrv	Rsrv	Rsrv	Rsrv
1110	Rsrv	Rsrv	Rsrv	Rsrv
1111	WregB	Rsrv	Rsrv	Rsrv

The command opcode also determines which packets (in addition to the request packet) will form the transaction. A detailed functional description of the actions that an RDRAM takes for each implemented command is provided in **"Rambus DRAM user's manual (Reference Manual)"**. The following table summarizes the functionality of each subcommand:

Table 5-3. Subcommand Summary

SubCommand	Description
Rseq	Read sequential data from memory space.
Rnsq	Read non-sequential (random-access) data from memory space.
Wseq	Write sequential data to memory space.
Wnsq	Write non-sequential (random-access) data to memory space.
Wbns	Write non-sequential (random-access) data to memory space with non-contiguous byte masking.
Npb	Write data is from data packet. There is no bit mask.
Dpb	Write data is from data packet. The bit mask is in the MDReg.
Mpb	Write data is from MDReg. The bit mask is from the data packet.
Bpb	Write data is from data packet. The bit mask is also from the data packet.
Rreg	Read sequential data from register space.
Wreg	Write sequential data to register space.
WregB	Broadcast write with no Okay acknowledge permitted.

The memory read commands are formed using the Rseq and Rnsq subcommands to select sequential or nonsequential (random) access.

- Rrrr = {Rseq, Rnsq}

The following table summarizes the available write commands and shows how they are formed from a 3×4 matrix of the Wwww and Bbb subcommands.

- WwwwBbb Wwww = {Wseq, Wnsq, Wbns}
  Bbb = {Npb, Dpb, Bpb, Mpb}

Table 5-4. Write Commands

Bbb subcommand	Wwww subcommands		
	Wseq (seqential-access with contiguous byte masking)	Wnsq (non-sequential-access)	Wbns (non-sequential-access with non-contiguous-byte-masking)
Npb	WseqNpb	WnsqNpb	WbnsNpb
Dpb	WseqDpb	WnsqDpb	WbnsDpb
Mpb	WseqMpb	WnsqMpb	WbnsMpb
Bpb	WseqBpb	WnsqBpb	Not implemented

There are three Wwww subcommands. They control the accessing pattern and the use of non-contiguous byte masking.

- Wseq - octbyte blocks in the RDRAM core are accessed in sequential (ascending little-endien) address order. Contiguous byte masking is controlled with the Adr[2:0] and Count[2:0] fields of the request packet.

- Wnsq - octbyte blocks in the RDRAM core are accessed in non-sequential address order. The addresses for the octbyte blocks within the sensed row come from serial address packets which are received on the BusEnable pin.
  The address order is arbitrary.

- Wbns - octbyte blocks in the RDRAM core are accessed in non-sequential address order, as in the Wnsq subcommand. In addition, byte masks are transmitted with the write data, permitting arbitrary non-contiguous byte masking of this write data. The bytemask octbytes are not included in the total octbyte transfer count ; i.e. a Count[7:3] field of 31 implies 4 bitmask octbytes and 32 write data octbytes, for a data packet size of 36 octbytes.

There are four Bbb subcommands. They select the type of bit masking to be applied to the write data.

- Npb (no-per-bit) - There is no bit mask applied to the write data. The MDReg is not used or modified.

- Dpb (data-per-bit) - The MDReg is used as a bit mask, the write data comes from the data packet. The same bit mask is used for each octbyte. This is also called persistent bit masking. The MDReg is not modified.
- Mpb (mask-per-bit) - The bit mask comes from the data packet, the write data comes from the MDReg. The same data is used for each octbyte. This is also called color masking. The MDReg is not modified.

- Bpb (both-per-bit) - The bit mask and the write data come from the data packet. The MDReg is not used, but is modified as a side effect (the WwwwBpb commands are used to load the MDReg for the WwwwDpb and WwwwMpb commands). This is also called non-persistent bit masking.
  The bitmask octbytes are included in the total octbyte transfer count ; i.e. a Count[7:3] field of 31 implies 16 bitmask octbytes and 16 write data octbytes.

5.2.3 Adr[35:0] Field

The Adr field is used as either a memory or register space address depending upon the OP[3:0] and OpX[1:0] fields. Devices extract a portion of the Adr field to match against their DeviceId register (IdMatch), thus selecting the device to which the request is directed. The remainder of the Adr field accesses the desired region of the device's memory or register space. The memory read and write commands and the Rreg and Wreg commands will only take place if there is an IdMatch. The IdMatch criteria is ignored for the WRegB commands, with all responding devices performing the required actions.

The Rambus protocol uses quadbyte resolution in the data packet for register space read and write commands; i.e. one quadbyte is the smallest data item that may be transferred, and all transfers are an integral number of quadbytes. The Adr[35:2] field is the quadbyte address. The Adr[1:0] field is Unimp for these commands, and should be driven with "00" by initiating devices.

The Rambus protocol uses octbyte resolution in the data packet for memory space read and write commands; i.e. one octbyte is the smallest data item that may be transferred, and all transfers are an integral number of octbytes. The Adr[35:3] field is the octbyte address.

Some commands use the Adr[2:0] field to specify contiguous byte masking. Refer to **"Rambus DRAM user's manual (Reference Manual)"**.

5.2.4 Count[7:0] Field

The following table summarizes the transfer count ranges for 18M RDRAMs:

Table 5-5. Transfer Count Summary

Count Range	μPD488170L
Maximum count for memory space	32 octbytes
Minimum count for memory space	1 octbyte
Maximum count for register space	1 quadbyte
Minimum count for register space	1 quadbyte

Register space read and write commands use a transfer count of one quadbyte, regardless of the Count[7:0] field value.

Memory space read and write commands specify the number of octbytes to be transferred with the Count[7:3] field. An offset-by-one-encoding is used so that "00000" specifies one octbyte, "00001" specifies two octbytes, and so on up to "11111" which specifies thirty-two octbytes. The transfer count does include the octbytes containing bitmasks (for commands using the Bpb subcommand). The transfer count does not include the octbytes containing non-contiguous ByteMasks (for commands using the Wbns subcommand).

Some commands use the Count[2:0] field to specify contiguous byte masking. Refer to **"Rambus DRAM user's manual (Reference Manual)"**.

Memory space transactions to RDRAMs are not allowed to cross internal row address boundaries within the device. Attempts to do so have Undef (undefined) results. These row boundaries are at 2kbyte intervals for 18M RDRAMs.

5.2.5 Adr[2:0] and Count[2:0] Fields for Contiguous Byte Masking

An initiating device wishing to write an arbitrary number of contiguous bytes to a starting address on an arbitrary byte boundary may do so with the Adr[2:0] and Count[2:0] fields with the Wseq subcommands. The transfer count and starting address are given by:

- MasterCount[7:0] specifies the number of bytes which the master device wishes to transfer.
- Adr[35:0] specifies the starting byte address (this is the same as the Adr[35:0] field in the request packet)

Where the convention used by the initiating device for the count is that Master-Count[7:0] = "00000000" means one byte, MasterCount[7:0] = "00000001" means two bytes and MasterCount[7:0] = "11111111" means 256 bytes (an offset-by-one encoding; the data block count is equal to MasterCount[7:0]+1).

The initiating device converts this internal count value into a value for the request packet with the following formula. Little-endien byte addressing is used for specifying bytes within octbytes.

Count[7:0] = Adr[2:0] + MasterCount[7:0] (Eq 5-1)

Where "+" denotes unsigned integer addition of two bit fields (short fields are zero-extended on the left). If the value of Adr[2:0] + MasterCount[7:0] is greater than 255 (it may be as much as 262), then the initiating device must break the request into two transactions.

The Adr[2:0] and Count[2:0] field generate masks for individual bytes within an octbyte. The Adr[35:3] and Count[7:3] field have the octbyte resolution previously described. The following tables show how the byte masks are generated. In the case of memory read transactions, the byte masks that are generated do not affect the data that is returned by the RDRAM; all data bytes in the first and last octbytes are returned in the read data packet.

In the case of memory write transactions, ByteMaskLS[7:0] applies to the first octbyte at Mem[AV][7:0][8:0]. Byte MaskMS[7:0] applies to the last octbyte at Mem[AV+CV][7:0][8:0]. All intermediate octbytes use a byte mask of 11111111 (a one means the byte is written, a zero means it is not). Here AV is the value of the Adr[35:3] field when interpreted as an unsigned, 33 bit integer, and CV is the value of the Count[7:3] field when interpreted as an unsigned, 5 bit interger. If the Count[7:3] is "00000" (one octbyte), the ByteMaskLS[7:0] and ByteMaskMS[7:0] masks are logically 'anded' together to give the effective byte mask.:

Table 5-6. Adr[2:0] to ByteMaskLS[7:0] Encoding

Adr[2:0]	ByteMaskLS[7:0]	Adr[2:0]	ByteMaskLS[7:0]
000	11111111	100	11110000
001	11111110	101	11100000
010	11111100	110	11000000
011	11111000	111	10000000

Table 5-7. Count[2:0] to ByteMaskMS[7:0] Encoding

Count[2:0]	ByteMaskMS[7:0]	Count[2:0]	ByteMaskMS[7:0]
000	00000001	100	00011111
001	00000011	101	00111111
010	00000111	110	01111111
011	00001111	111	11111111

The detailed functional description is provided in **"Rambus DRAM user's manual (Reference Manual)"**.

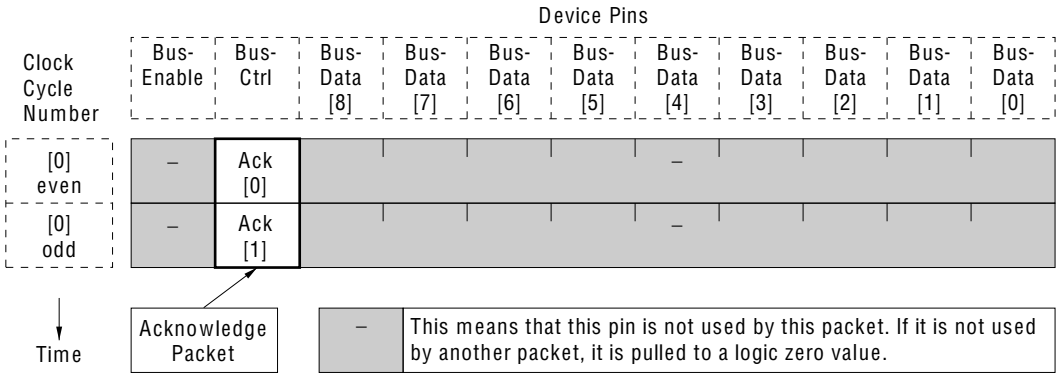
5.2.6 ReqUnimp[8:0] Fields

These fields are unimplemented (Unimp) in the request packet. They should be driven as zeroes by initiating devices.

5.3 Acknowledge Packet

The Ack[1:0] field carries the acknowledge encoding from the responding device(s) to the initiating device and any other listening devices. The following figure shows the format of the acknowledge packet.

Figure 5-2. Acknowledge Packet Format



The following table summarizes the four combinations of the Ack[1:0] field. The Ack3 combination is Undef. The Okay combination indicates that the read or write access to the specified space will take place.

When a responding device acknowledges a request with a Nack, then there will be no immediate change in the state of the device's memory space or register space. The responding device will take the appropriate steps to make the requested region of memory or register space accessible when the initiating device makes a subsequent request. The initiating device will need to wait some device-dependent length of time until the requested region is available.

There are three possible reasons for an RDRAM to respond with Nack. They are summarized below. The detailed functional description is provided in **"Rambus DRAM user's manual (Reference Manual)"**.

- $t_{PostMemWriteDelay}$ or $t_{PostRegWriteDelay}$ violation
- RowMiss (this causes a delay of $t_{RetrySensedClean}$ or $t_{RetrySensedDirty}$)
- ongoing refresh (this causes a delay of up to $t_{RetryRefresh}$)

Table 5-8. Ack[1:0] Encodings

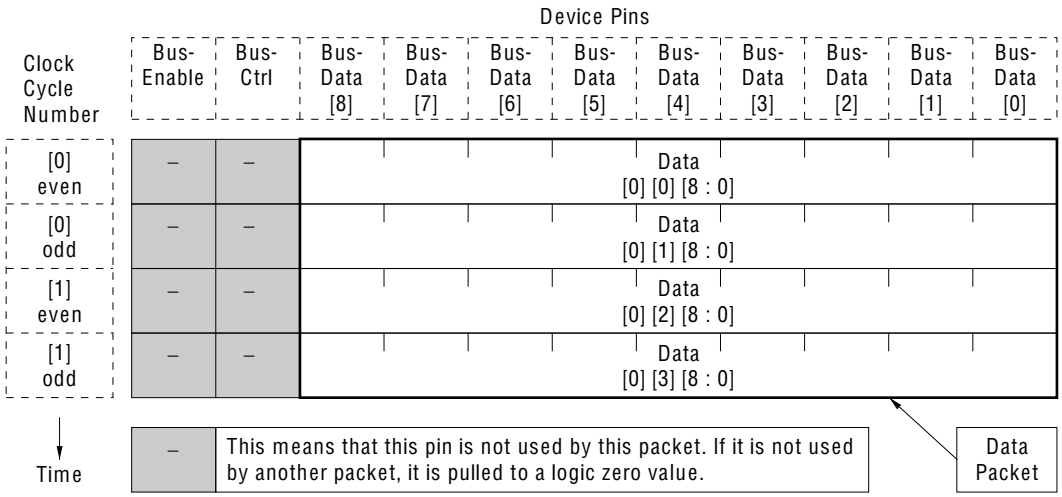
Commands allowed to use the Ack Combination	Ack [1:0]	Name	Description	Spec Undef
All commands	00	Nonexistent	Indicates passive acceptance of the request (WregB), or indicates that the addressed device did not respond (all other commands).	Spec
All commands but WregB	01	Okay	Indicates that the request was accepted by the addressed by the addressed (responding) device.	Spec
All commands	10	Nack	Indicates that the request could not be accepted because the state of the responding device prevented an access at the fixed timing slot.	Spec
All commands but WregB	11	Ack3	This should not be returned by this responding device. Initiating devices will, when presented with this combination, have an undefined response.	Undef

5.4 Data Packet

The following figure shows the format of a data packet for register space read and write commands. It consists of 1 quadbyte driven on the BusData[8:0] wires for RDRAMs.

Other responding devices may support data packet lengths longer than one quadbyte.

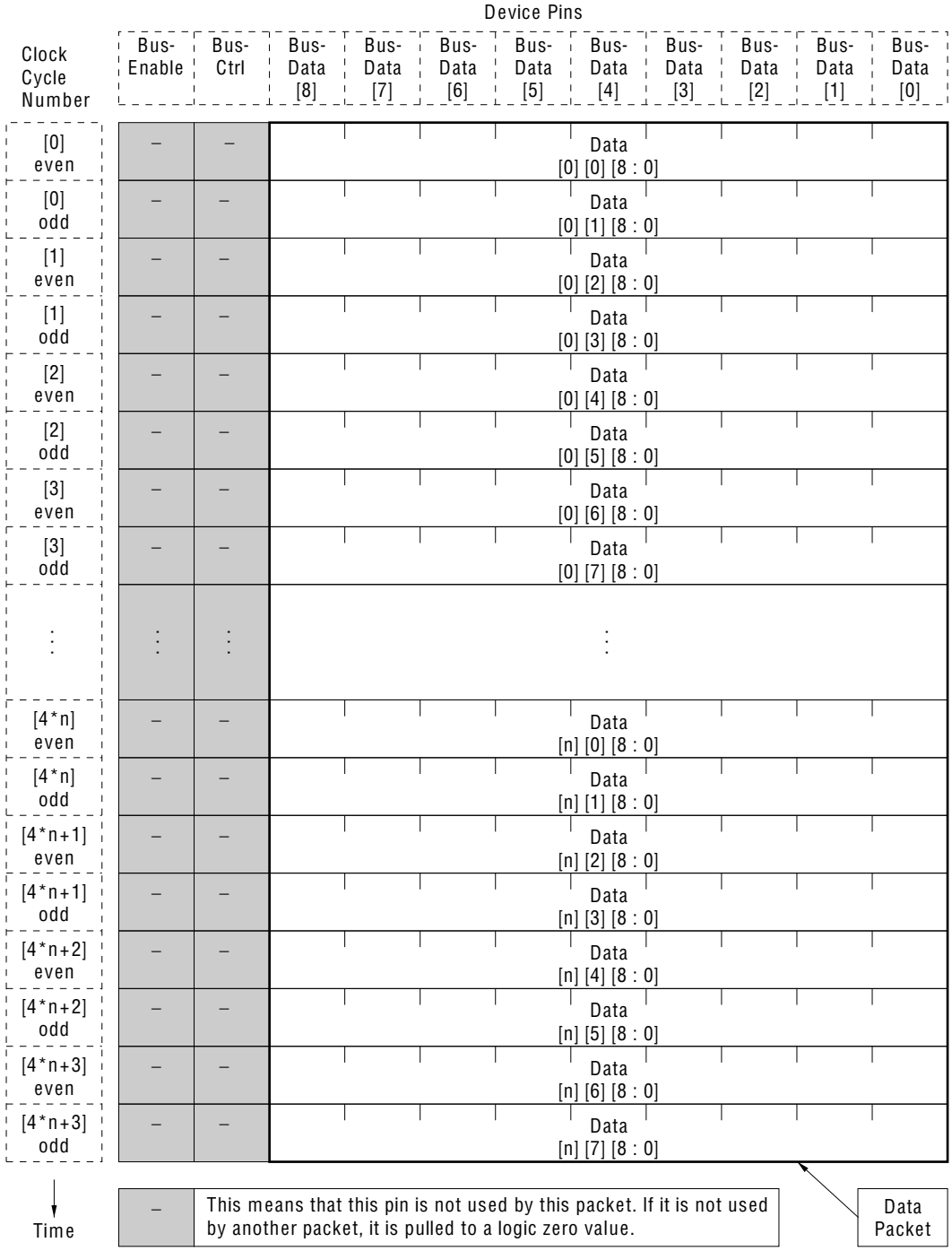
Figure 5-3. Data Packet Format (Register Space)



The following figure shows the format of a data packet for memory space read and write commands. For most of these commands, it consists of 1 to 32 octbytes driven on the BusData[8:0] wires. In the figure, "n" is either the CV value (if the transaction is allowed to complete) or the last count value (if the transaction is terminated prematurely by the serial control packet). "CV" is the value of the Count[7:3] field when interpreted as an unsigned, 5 bit integer.

The detailed functional description is provided in **"Rambus DRAM user's manual (Reference Manual)"**.

Figure 5-4. Data Packet Format (Memory Space)



5.5 Serial Address Packet Format

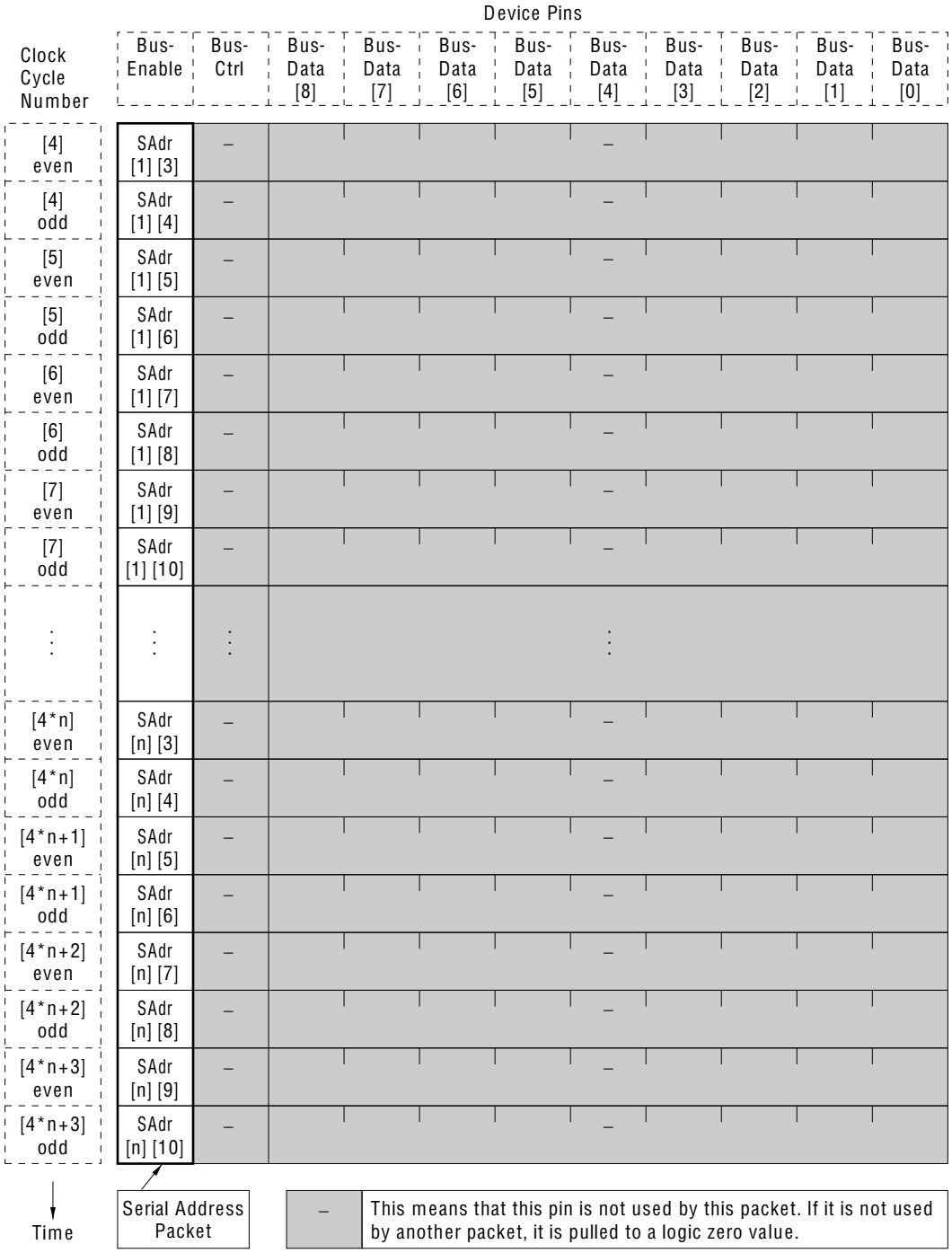
The serial address packet is transmitted by the initiating device and received by the responding devices. It provides eight low-order address bits for each octbyte which is accessed in memory space (a non-sequential or random-access transfer). These eight address bits are transferred serially on the BusEnable pin of the RDRAM, and are thus called a serial address. Each eight bit serial address accesses an octbyte of data within the RowSenseAmpCache of one of the two banks of the RDRAM. The complete set of serial addresses transmitted by the initiating device during the transaction are referred to as a serial address packet. The commands which use this packet are the Rnsq, WnsqBbb, and WbnsBbb classes of commands.

The high order bits for each octbyte are provided by the Adr[35:11] address bits from the request packet. The low-order address bits for the first octbyte are Adr[10:3], also from the request packet. The low-order address bits for octbytes [n:1] are provided by the serial address packet. As before, "n" is either the CV value (if the transaction is allowed to complete) or the last count value (if the transaction is terminated prematurely by the serial control packet). "CV" is the value of the Count[7:3] field when interpreted as an unsigned, 5 bit integer. The detailed functional description is provided in **"Rambus DRAM user's manual (Reference Manual)"**.

Table 5-9. Serial Address Fields (i = n:1)

Serial Address Field	Description	Unimp Imp
SAdr[i][10:3]	Low-order address bits for each octbyte.	Imp

Figure 5-5. Serial Address Packet Format



5.5.1 Serial Control Packet Format

The serial control packet is transmitted by the initiating device and received by the responding devices. It provides for the early termination of a memory space read or write transaction (before the specified data count in the Count[7:3] field has elapsed). It consists of eight bits transferred serially on the BusCtrl pin of the device, thus it is referred to as a serial control packet. The eight bits have the same timing alignment as the serial address packet. The commands which use this packet are all of those which access memory space. The register read and write commands do not use the serial control packet. The 18M RDRAM implements this packet.

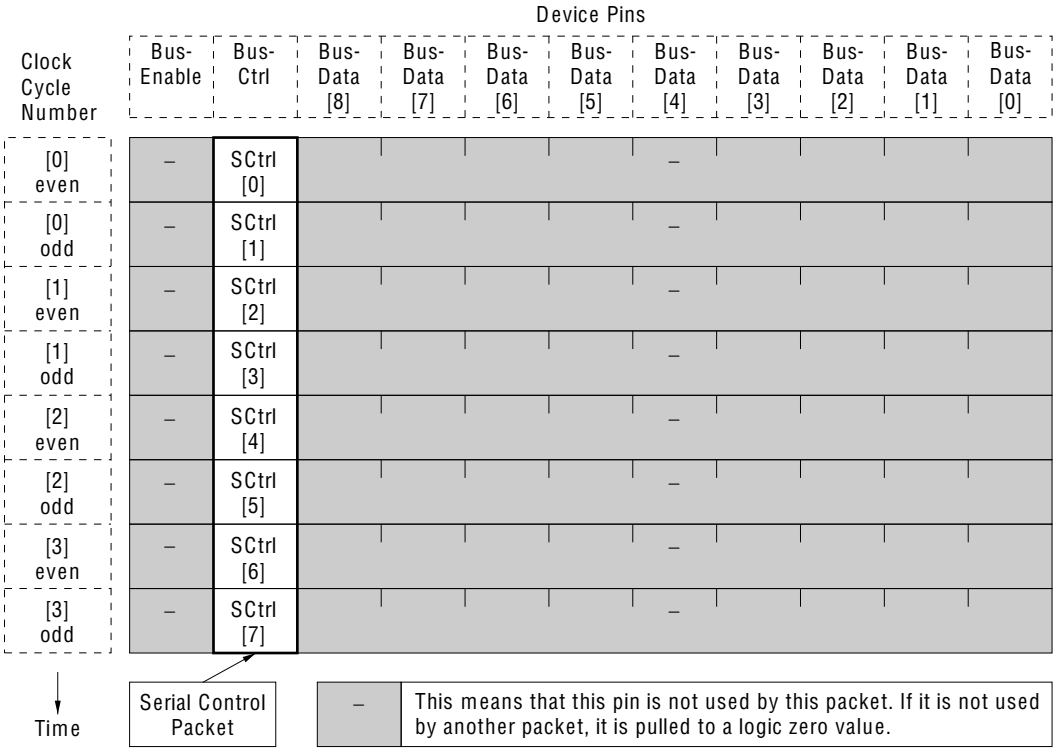
The termination occurs on octbyte data packet boundaries. The next figure shows the format of the serial control packet. The following table summarizes the function of the bits within the serial control packet. Note that the bits in the even bus ticks must be zero in order for framing to work properly (otherwise, one of these bits would be interpreted as the Start bit of a new request packet). The SCtrl[5] bit is used to control termination, and the other three odd bus tick bits are unimplemented.

Table 5-10. Serial Control Fields

Serial Control Fields	Description	Unimp Imp
SCtrl[0]	This bit must be a zero due to framing requirements.	Imp
SCtrl[1]	unimplemented	Unimp(0)
SCtrl[2]	This bit must be a zero due to framing requirements.	Imp
SCtrl[3]	unimplemented	Unimp(0)
SCtrl[4]	This bit must be a zero due to framing requirements.	Imp
SCtrl[5]	0 means don't terminate the current access. 1 means terminate the current access.	Imp
SCtrl[6]	This bit must be a zero due to framing requirements.	Imp
SCtrl[7]	unimplemented	Unimp(0)

If a memory read transaction (RrrrAaa) is terminated by asserting the SCtrl[5] bit to a logical one, the data octbyte with which it is associated is not transmitted by the responding device. The initiating device may start a new transaction once the transmission of the read data packet has ceased. The detailed functional description is provided in **"Rambus DRAM user's manual (Reference Manual)"**.

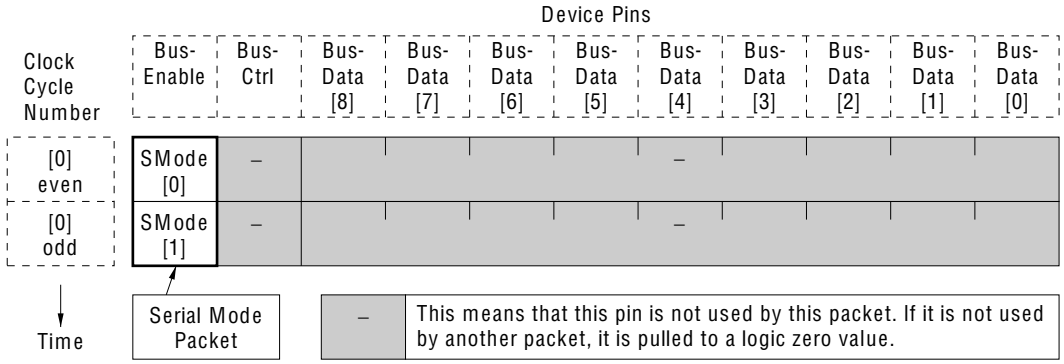
Figure 5-6. Serial Control Packet Format



5.5.2 Serial Mode Packet Format

The serial mode packet transmitted by initiating devices, and received by responding device. Its format is shown in the following figure.

Figure 5-7. Serial Mode Packet Format



The serial mode packet modifies the state of the Count00[7:0] and Count11[7:0] counters.

These counters cause operating mode transitions when they reach special values. The detailed functional description is provided in **"Rambus DRAM user's manual (Reference Manual)"**.

A serial mode packet with the SMode[1:0] field set to 00 is the default. Most transitions are caused by blocks of sequential serial mode packets, each with the SMode[1:0] field set to 11. The serial mode packets should never set SMode[1:0] field to 01 or 10. This is because in some of the operating modes, the clock generator is unlocked (the frequency is correct but not the phase). When this happens, the BusEnable receiver is unable to discriminate anything other than long pulses of zeros or ones. Because the frequency of the clock generator is correct, it can count the length of these pulses with moderate accuracy.

Table 5-11. Serial Mode Fields

SMode[1:0]	Description	Spec/Rsrv/ Undef
00	Increments Count00[3:0], clears Count11[7:0].	Spec
01	–	Undef
10	–	Undef
11	Increments Count11[7:0], clears Count00[3:0]	Spec

6. State Diagram

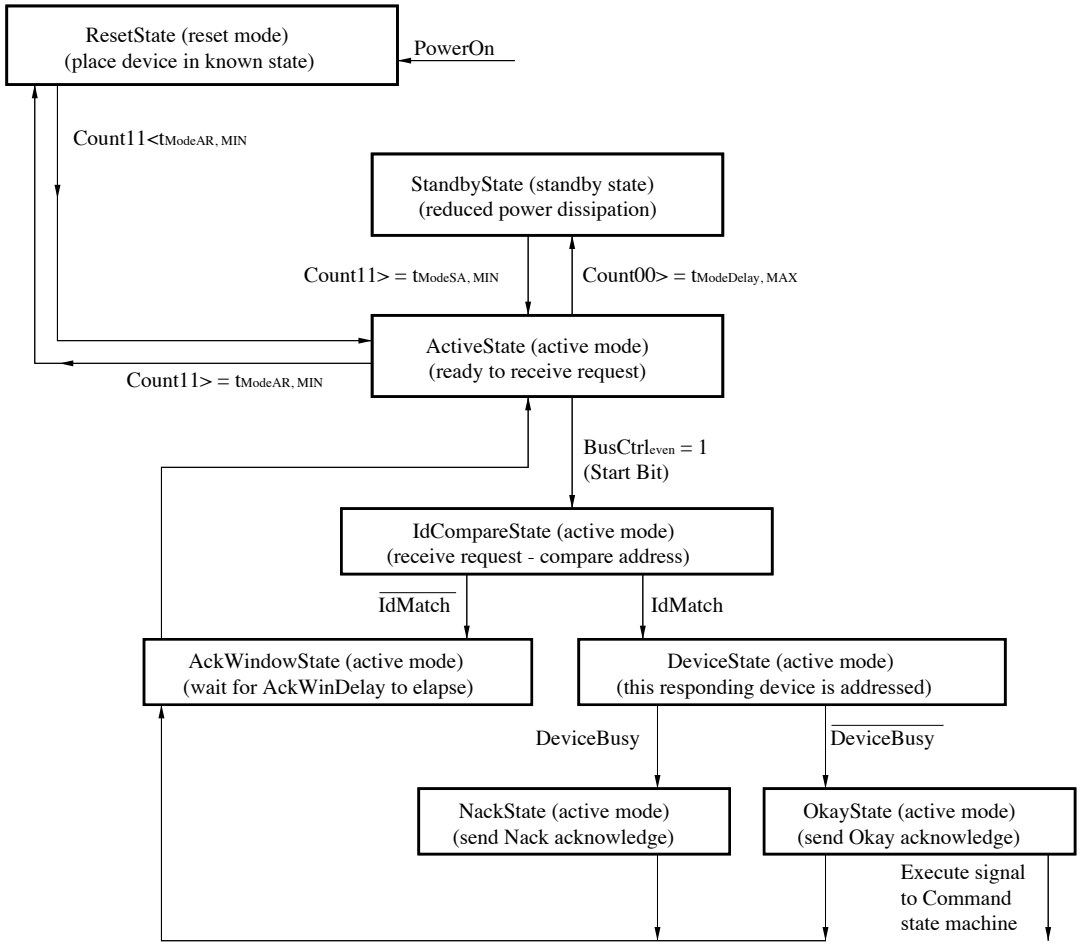
The following figure is a state diagram of the Frame state machine. The operating mode of the device depends upon which of the nine states it is in:

- reset mode - ResetState
- standby mode - StandbyState
- active mode - ActiveState, IdCompareState, DeviceState, OkayState, NackState, AckWindowState

This section will only discuss the first three states (ResetState, StandbyState, ActiveState). The remaining five states which are shown shaded in the state diagram (IdCompareState, DeviceState, OkayState, NackState, AckWindowState) will be dealt with in the **"Rambus DRAM user's manual (Reference Manual)"**.

The device will enter ResetState when power is initially applied (PowerOn). In ResetState, the device will be in the reset operating mode, in which all control registers assume a known state. If power has just been applied, the device will pass through ActiveState and settle in StandbyState, and remain there until serial mode packets are received from an initiating device.

Figure 6-1. Frame State Machine - State Diagram



ActiveState is the state in which all decisions are made to transition to the states for the other operating modes. From here, the device will also enter the transaction-framing states. Refer to **"Rambus DRAM user's manual (Reference Manual)"**.

After poweron, the device will re-enter ResetState when the value of the Count11[7:0] counter is greater than or equal to $t_{ModeAR,MIN}$. The device will leave ResetState when the value of the Count11[7:0] counter is less than $t_{ModeSA,MIN}$. This will happen when an SMode[1:0] field of 00 is received, causing the Count11[7:0] counter to clear.

The device will enter StandbyState when the value of the Count00[3:0] counter is greater than or equal to $t_{ModeDelay,MAX}$. The device will leave StandbyState when the value of the Count11[7:0] counter is greater than or equal to $t_{ModeSA,MIN}$.

Caution PD (MinInterval Register [3][2]) = 0 is necessary. Because, 18M RDRAM don't support Power Down request.

6.1 Parameters for Operating Mode Transitions

The following table summarizes the parameter values associated with operating mode transitions of a responding device. A minimum and maximum value are given for the parameters to account for implementation differences. In all cases, the SMode[1:0] field of the consecutive serial mode packets must have the value 11 to cause an operating mode transition (with the exception of the $t_{ModeDelay,MAX}$ as mentioned in the previous section). Initiating devices must use values within the minimum and maximum SMode packet count requirements shown above to control operating mode transitions.

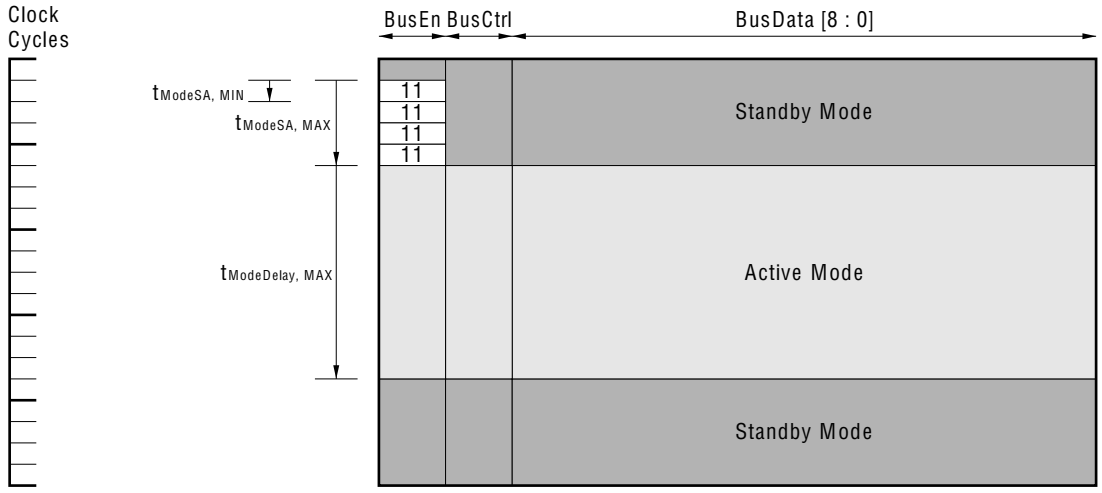
Table 6-1. Responding Device Parameters for Operating Mode Transitions

Count Parameter Name	Minimum (clock cycles)	Maximum (clock cycles)	Description
t_{ModeSA}	1	4	Number of SMode packets to cause a transition from Standby-Mode to ActiveMode
$t_{ModeOffSet}$	4	7	Offset from beginning of SMode packet to request packet for standby to active transition
$t_{ModeDelay}$	–	20	Delay from end of SMode packet to request packet for standby to active transition
$t_{ModeSwitchReset}$	320	–	Number of SMode packets to cause a transition from Active-Mode to ResetMode
t_{Reset}	32	–	Time required for an RDRAM's internal nodes to settle to their reset values.
$t_{Lock, Reset}$	750	–	Time required for an RDRAM's internal clock generator to lock to the external clock.

6.2 Standby Mode and Active Mode

The following figure shows the basic transitions between active and standby modes in response to serial mode packets

Figure 6-2. Basic ActiveMode/StandbyMode Transitions



This is a timing diagram, with time increasing in the downward direction. The time scale is in clock cycles, as shown on the left scale. The value of each of the eleven low-swing signal pins of the responding device is shown with the assumption that t_{TR} is zero (the responding device is located at the master end of the Channel).

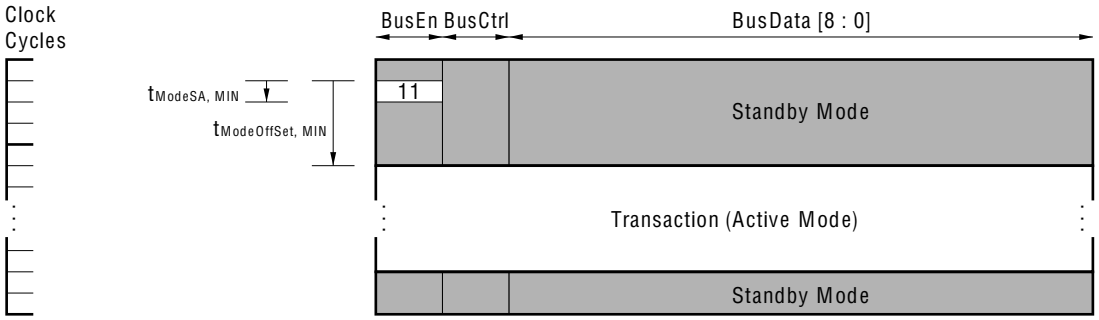
Serial mode packets with an SMode[1:0] field are shown as a box with a "11" label in the BusEn column. The BusEnable defaults to a logical zero value. The initiating device has transmitted $t_{ModeSA,MAX}$ serial mode packets with SMode[1:0] equal to 11 (this is the longest sequence permitted for invoking a standby to active transition). After the first $t_{ModeSA,MIN}$ serial mode packets, the device begins the transition to active mode. It reaches active mode after $t_{ModeOffSet, MIN}$ clock cycles after the start of the first serial mode packet. It remains there for $t_{ModeDelay,MAX}$ clock cycles after the last serial mode packet.

The responding device is in active mode when it begins framing the request packet. A transaction may begin in any of the clock cycles with the light shading above (labeled "Active Mode").

If the serial mode packet(s) causing a standby to active mode transition are not followed by a transaction with $t_{ModeDelay,MAX}$ clock cycles after the last serial mode packet, then the responding device will return to standby mode.

The next figure shows the case in which a transaction is started as early as possible after a serial mode packet which causes a standby to active mode transition.

Figure 6-3. ActiveMode/StandbyMode Transition - Early Transaction



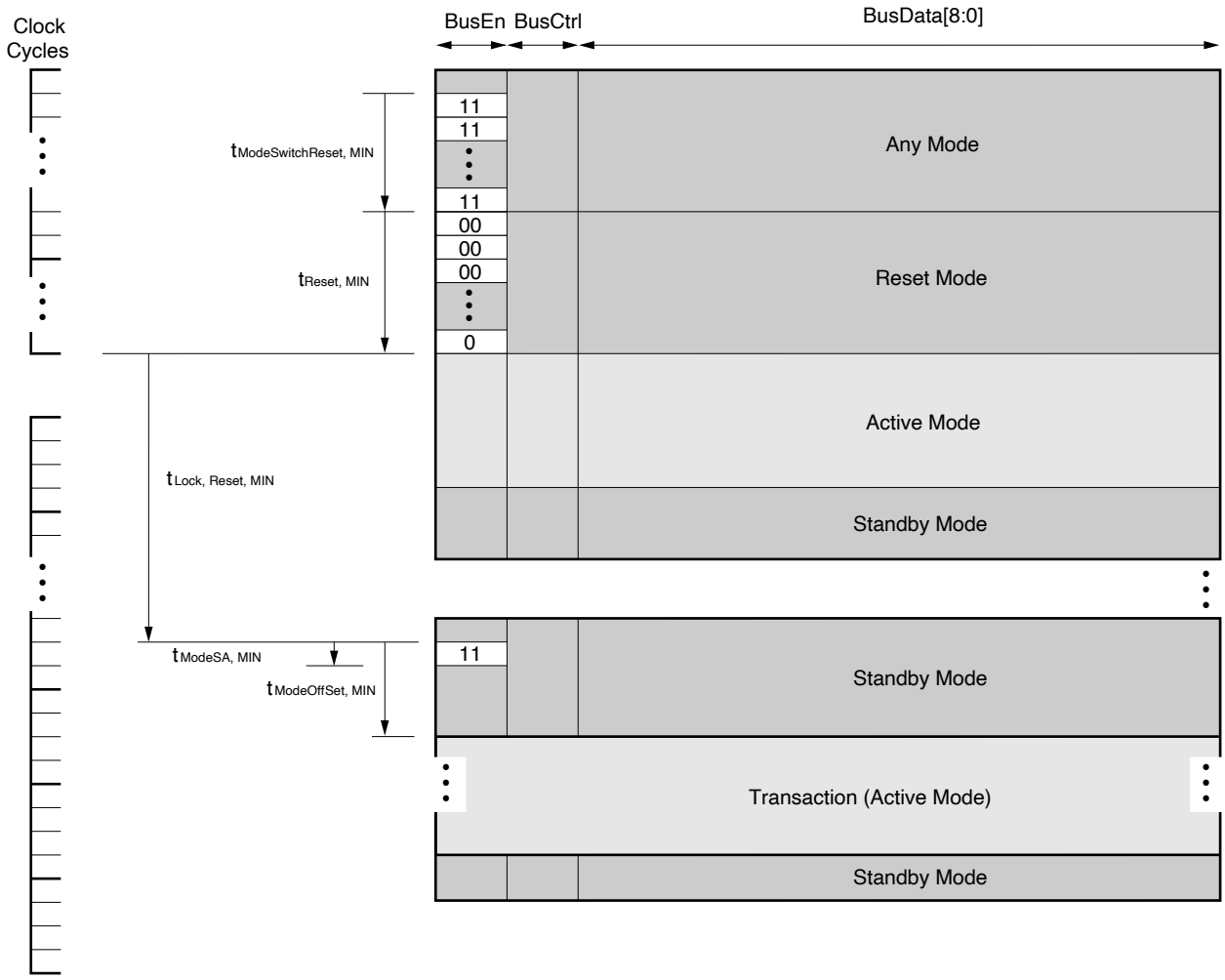
A transaction is composed of packet types other than serial mode packets, and will be defined in the next chapter. These other packet types lie entirely inside the heavy black box in the above two figures. When a transaction has completed, the device returns to standby mode. The detailed functional description is provided in **"Rambus DRAM user's manual (Reference Manual)"**.

6.3 ResetMode

Reset mode is entered when a consecutive sequence of $t_{ModeRA,MIN}$ serial mode packets with a value of 11 are seen by a responding device (shown in the following figure). In reset mode, all devices enter a known state from which they may be Initialized. The device remains in reset mode for as long as serial mode packets with 11 value are received. When one or more serial mode packets with a value of 00 are seen, the responding device enters the active mode state.

Although devices enter the active mode state immediately, their clock circuitry requires a time $t_{Lock,MIN}$ to resynchronize. Initiating devices must wait this long after the transition out of reset mode before starting any transactions.

Figure 6-4. ResetMode to ActiveMode Transition



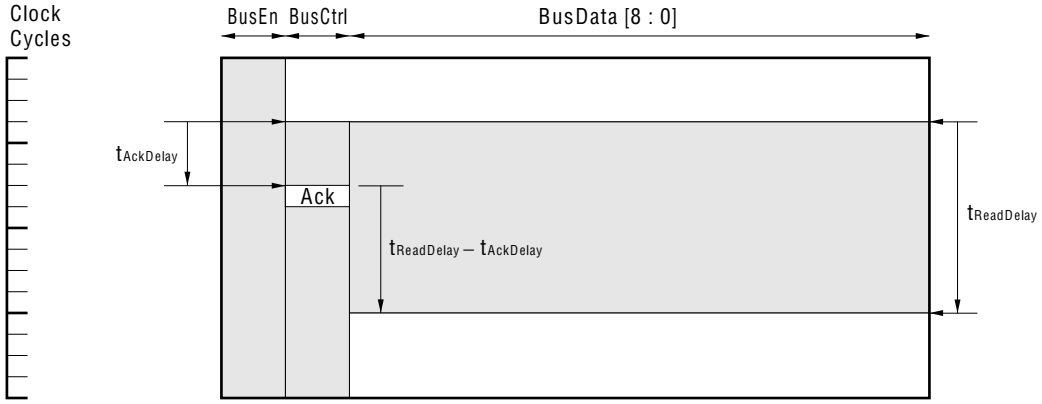
7. Transactions

7.1 Read Transactions

The following figure shows the basic form of a memory space or register space read transaction. There are request and acknowledge packets, with the same $t_{AckDelay}$ and $t_{AckWinDelay}$ timing constraints. $t_{AckWinDelay}$ will not be shown explicitly on any further transaction diagrams in this document.

When the responding device transmits an Okay acknowledge packet to the initiating device, it will also transmit a data packet with read data. This packet is sent a time $t_{ReadDelay}$ after the end of the request packet. The $t_{ReadDelay}$ value is in t_{CYCLE} units and is programmed into the ReadDelay field of the Delay register of each responding device. It is not required to be the same for all devices within a Rambus system, but the difference ($t_{ReadDelay}$ - $t_{AckDelay}$) is required to be the same. This allows initiating devices to use the acknowledge packet to determine when the read data packet begins. The detailed functional description is provided in **"Rambus DRAM user's manual (Reference Manual)"**.

Figure 7-1. Read Transaction



7.2 Write Transactions

The following figure shows the basic form of a memory space or register space write transaction. There are request and acknowledge packets, with the same $t_{AckDelay}$ and $t_{AckWinDelay}$ timing constraints as already discussed.

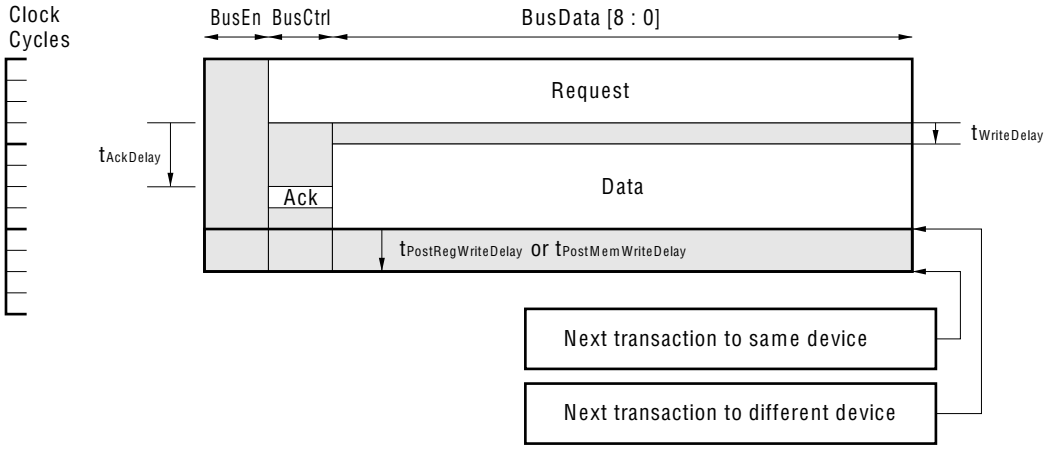
When the initiating device transmits a request packet to the responding devices, it will also transmit a data packet with write data. This packet is sent a time $t_{WriteDelay}$ cycles after the end of the request packet. The $t_{WriteDelay}$ is in t_{CYCLE} units and is programmed into the WriteDelay field of the Delay register of each responding device. It is required to be the same for all devices within a Rambus system. A responding device will see the same $t_{WriteDelay}$ interval between the request and write data packets whether the device is on the Primary Channel or on a Secondary Channel.

If the responding device returns an Okay acknowledge packet, then the transaction is complete at the end of the acknowledge window or at the end of the write data packet, whichever is later. The next request packet can be transmitted in the following clock cycle except for the case in which a register or memory space write to a device is followed by any other transaction to that device. In that case, one of the following two intervals must be inserted between the two transactions, where the memory or register case depends upon the first transaction.

- $t_{PostRegWriteDelay}$ if the current transaction is a register space access
- $t_{PostMemWriteDelay}$ if the current transaction is a memory space access

If the responding device returns a Nack or Nonexistent acknowledge packet for a write command, then no write data packet is required by the responding device. The current transaction is complete at the end of the acknowledge window, or when the initiating device stops transmitting the write data packet, whichever is later. The next request packet can be transmitted in the following clock cycle. For the case of a Nack or Nonexistent, the initiating device must terminate the write data packet before another initiating device is given control of the Rambus Channel for a transaction. This is part of the arbitration mechanism used by the initiating devices. The arbitration mechanism is not specified in this document because it does not use the Rambus Channel. The detailed functional description is provided in **"Rambus DRAM user's manual (Reference Manual)"**.

Figure 7-2. Write Transaction



7.3 Read Transactions with Serial Address Packet

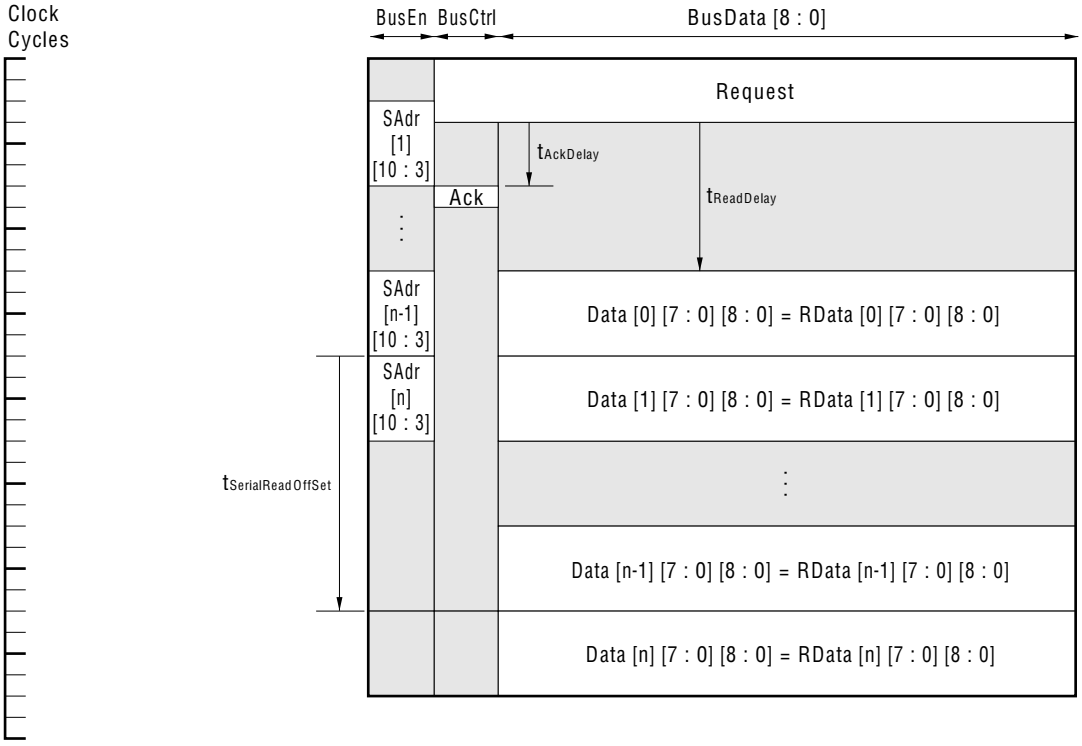
The following figure shows a memory space read transaction for a command which uses the serial address packet. For a transaction which moves (n+1) octbytes of read data, the serial address packet will be (4 × n) clock cycles in length (recall that the low-order address bits for the first octbyte of read data come from the request packet).

Each serial address subpacket (each SAdr[i][10:3] field) is transmitted by the initiating device a time $t_{SerialReadOffSet}$ clock cycles before the octbyte of read data to which it corresponds. This means that the serial address packet will move with the read data packet, with a constant offset.

- $t_{SerialReadOffSet}$ is the delay from the beginning of a serial address subpacket to the beginning of the read data subpacket (octbyte) with which it is associated.

The detailed functional description is provided in **"Rambus DRAM user's manual (Reference Manual)"**.

Figure 7-3. Read Transaction with Serial Address Packet



7.4 Write Transactions with Serial Address Packet

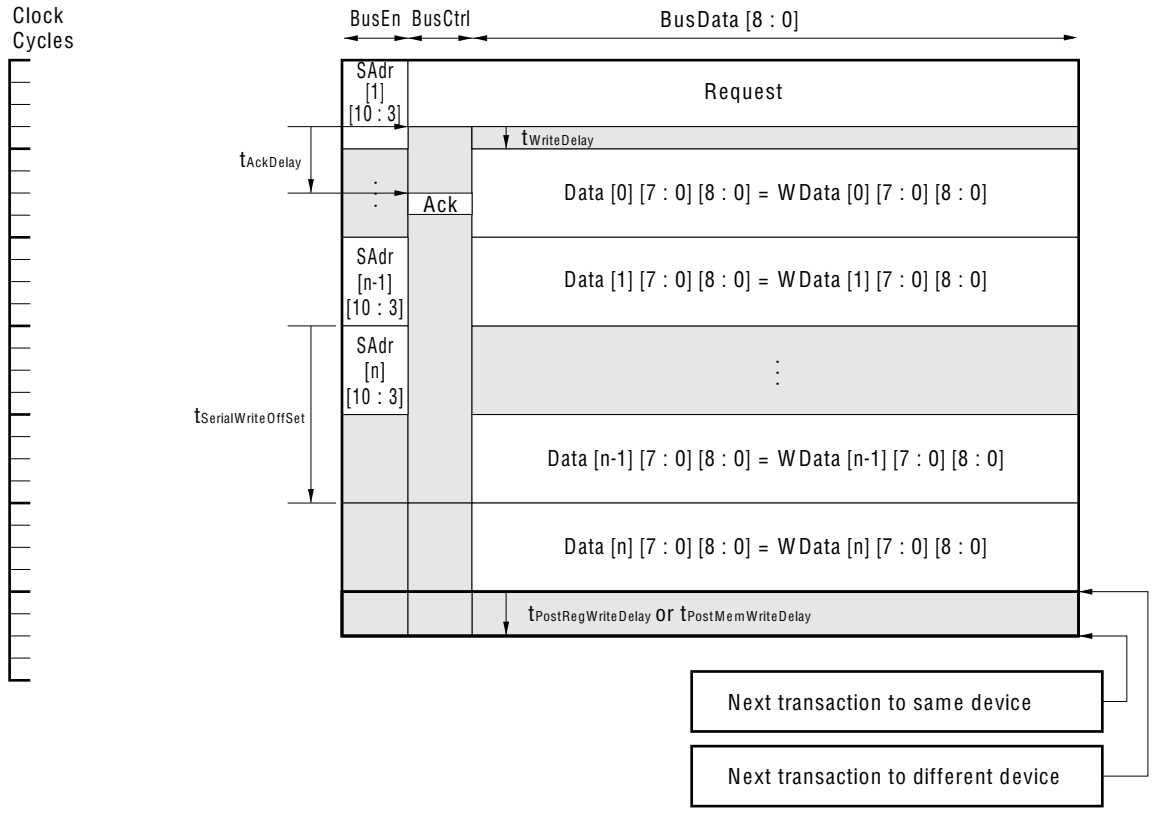
The following figure shows a memory space write transaction for a command which uses the serial address packet. For a transaction which moves (n+1) octbytes of write data, the serial address packet will be (4 × n) clock cycles in length (recall that the low-order address bits for the first octbyte of write data come from the request packet).

Each serial address subpacket (each SAdr[i][10:3] field) is transmitted by the initiating device a time $t_{SerialWriteOffSet}$ clock cycles before the octbyte of write data to which it corresponds. This means that the serial address packet will move with the write data packet, with a constant offset.

- $t_{SerialWriteOffSet}$ is the delay from the beginning of a serial address subpacket to the beginning of the write data subpacket (octbyte) with which it is associated.

Note that this offset interval is measured at the initiating device or the responding device; it will be the same at either point since the serial address packet and write data packet are moving in the same direction - from initiating device to responding device.

Figure 7-4. Write Transaction with Serial Address Packet



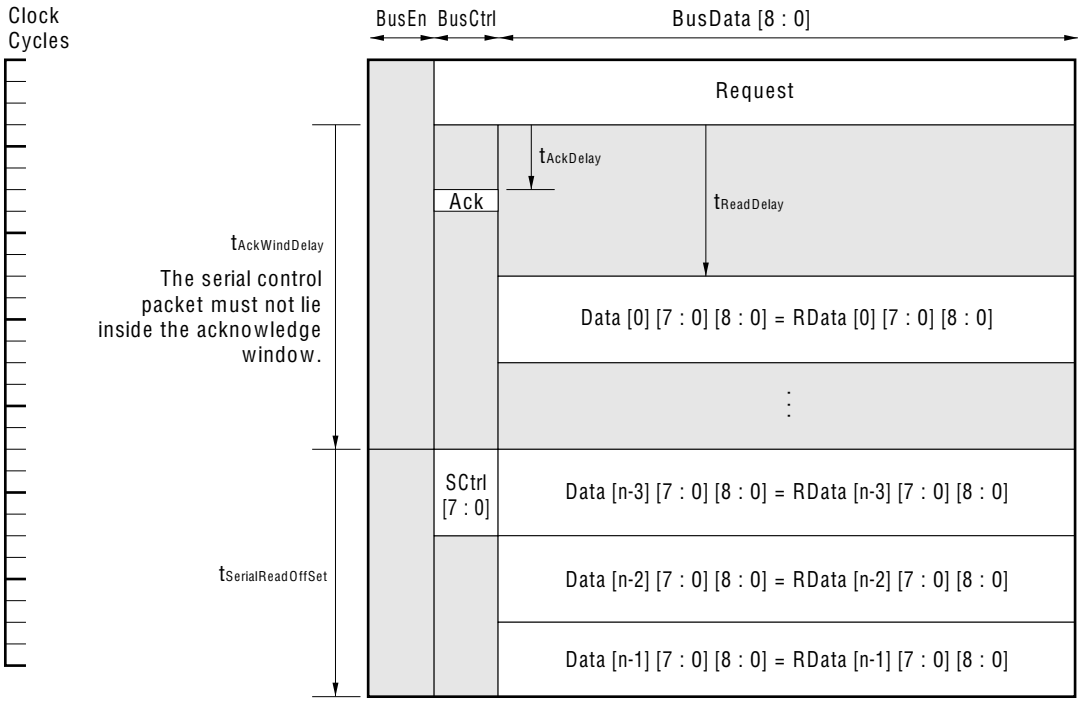
7.5 Read Transactions with Serial Control Packet

The following figure shows a memory space read transaction for a command which uses the serial control packet. This packet is used to terminate a transaction before the (CV+1) octbytes of read data have been transferred, where CV is the value of the Count[7:3] Field when interpreted as an unsigned, five bit integer. In the example shown, the read data is terminated after (n) octbytes have been transferred.

The serial control packet is transmitted by the initiating device a time $t_{SerialReadOffSet}$ clock cycles before the end of the last read data octbyte which is transmitted by the responding device.

The serial control packet is also constrained to lie entirely outside the $t_{AckWinDelay}$ interval, as shown in the figure, in order to avoid interference with the acknowledge packet which is being returned by the responding device. Violation of this constraint will produce undefined (Undef) results. The detailed functional description is provided in **"Rambus DRAM user's manual (Reference Manual)"**.

Figure 7-5. Read Transaction with Serial Control Packet



7.6 Write Transactions with Serial Control Packet

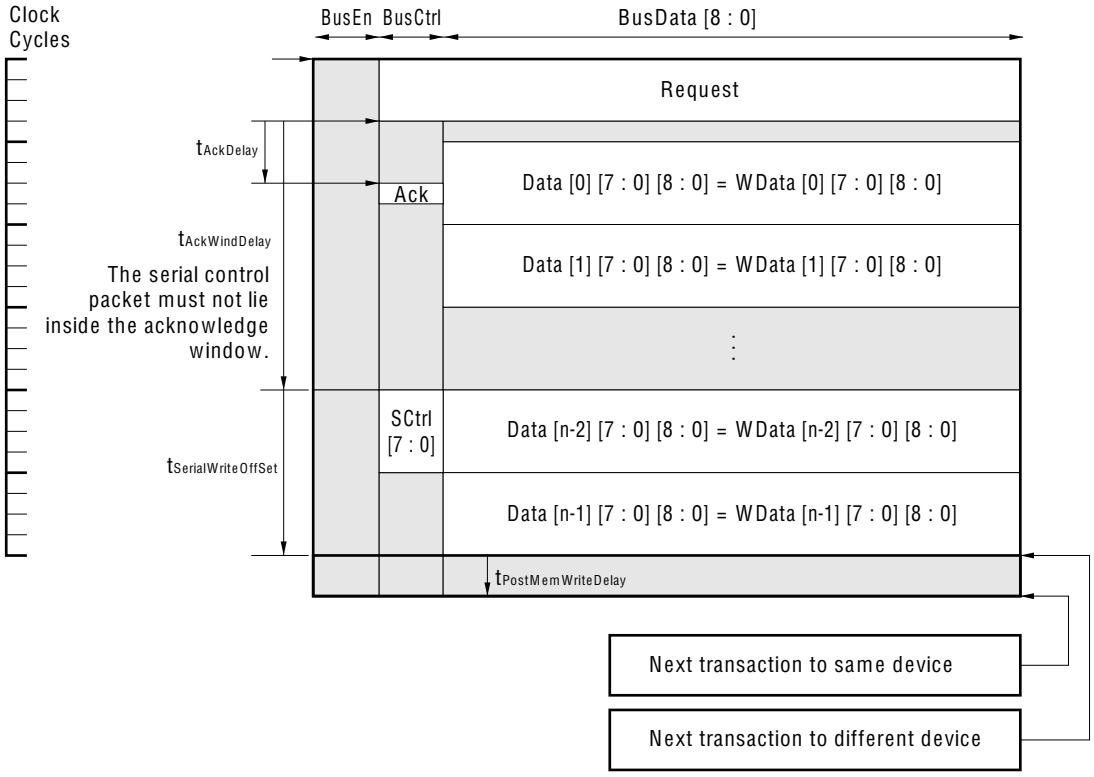
The following figure shows a memory space write transaction for a command which uses the serial control packet. This packet is used to terminate a transaction before the (CV+1) octbytes of write data have been transferred, where CV is the value of the Count[7:3] field when interpreted as an unsigned, five bit integer. In the example shown, the write data is terminated after (n) octbytes have been transferred.

The serial control packet is transmitted by the initiating device a time $t_{SerialWriteOffSet}$ clock cycles before the end of the last write data octbyte which is transmitted by the initiating device.

Note that this offset interval is measured at the initiating device or the responding device; it will be the same at either since the serial address packet and write data packet are moving in the same direction - from initiating device to responding device.

The serial control packet is also constrained to lie entirely outside the $t_{AckWinDelay}$ interval, as shown in the figure, in order to avoid interference with the acknowledge packet which is being returned by the responding device. Violation of this constraint will produce undefined (Undef) results.

Figure 7-6. Write Transaction with Serial Control Packet

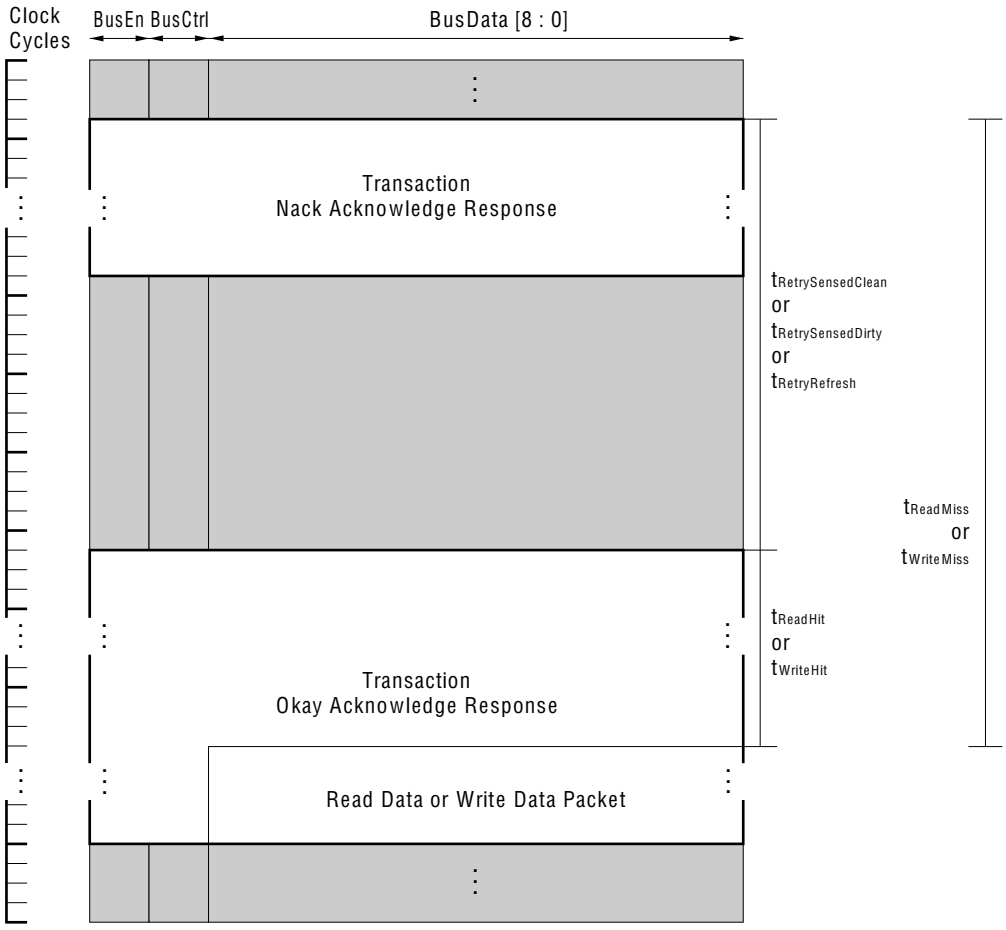


8. Nack Acknowledge Response

8.1 Retry and Miss Latency

If a responding device returns a Nack acknowledge packet, then no read or write data packet is transacted. The current transaction is complete at the end of the acknowledge window. It will be necessary to wait for an interval of time (called a t_{RETRY} interval) before resubmitting the transaction. The following figure illustrates this case.

Figure 8-1. Nack Acknowledge Response



Once the t_{RETRY} interval has elapsed, the transaction may be restarted by the initiating device, and the RDRAM will return an Okay acknowledge packet and the data packet will be transferred. An RDRAM will Nack any other transactions which are issued during the t_{RETRY} interval.

Two miss latency parameters may be derived with the following equations:

$$t_{ReadMiss} = t_{RETRY} + t_{ReadHit} \quad \text{(Eq 8-1)}$$

$$t_{WriteMiss} = t_{RETRY} + t_{WriteHit} \quad \text{(Eq 8-2)}$$

where $t_{RETRY} = \{t_{RetrySensedClean}, t_{RetrySensedDirty}, t_{RetryRefresh}\}$. The $t_{ReadMiss}$ and $t_{WriteMiss}$ parameters are the time from the beginning of the original (Nacked) request packet to the beginning of the data packet which is eventually transferred.

8.2 t_{RETRY} Interval

8.2.1 Retry Due to RowMiss

If an initiating device requests a region of memory space in an RDRAM slave which is not currently held in the RowSenseAmpCache, the RDRAM will respond with a Nackacknowledge packet. The RDRAM will then begin a RowMiss operation to get the proper row into the RowSenseAmpCache. During the RowMiss, the RDRAM will Nack any request it is given. When the RowMiss is complete, the new row may be accessed.

Each bank has a Valid flag and a Dirty flag for its Row register. After reset, both are zero. After a RowMiss has caused a new row to be placed into the RowSenseAmpCache, the Row register contains its row address and the Valid flag is set to a one. If the RowSenseAmpCache contents are modified with a memory write transaction, the dirty flag will be set. These flags are not directly accessible to initiating devices.

A subsequent RowMiss will cause the old row to be written back to the bank (if it was dirty and an explicit restore was not forced with the Close bit in the request packet) and a new row to be placed into the RowSenseAmpCache. The time required for this is called the t_{RETRY} time, and is added to the normal read and write hit latency times, as shown in the preceding figure. These times are given by the following equations. The component parameters are shown in a subsequent table. All of these t_{RETRY} intervals correspond roughly to the cycle time parameter t_{RC} of a conventional page mode DRAM. This is because RDRAMs use CAS-type accesses for all memory read and write transactions.

After a new row is sensed and placed into the RowSenseAmpCache, a final interval $t_{RowImprestore}$ is used to restore the row in core back to its original state. This is necessary because the DRAM sense operation is destructive. This interval is not in the critical timing path, and is performed in parallel with a subsequent data transfer. It can extend a subsequent retry operation.

There are two t_{RETRY} equations for the 18M RDRAM:

$$t_{RetrySensedClean} = t_{RowOverHead} + t_{RowPrecharge} + t_{RowSense} \quad \text{(Eq 8-3)}$$

$$t_{RetrySensedDirty} = t_{RowOverHead} + t_{RowExprestore} + t_{RowPrecharge} + t_{RowSense} \quad \text{(Eq 8-4)}$$

The detailed functional description is provided in **"Rambus DRAM user's manual (Reference Manual)"**.

8.2.2 Retry Due to Pending Burst Refresh

In a 18M RDRAM, a refresh burst will first restore the currently accessed row if it is dirty. This requires a $t_{RowExprestore}$ interval. If the row is clean, this interval is not required. A burst of four rows are precharged/sensed/restored (using the $t_{RowPrecharge}$, $t_{RowSense}$ and $t_{RowImprestore}$ intervals), and the current row is precharged/sensed so the RDRAM is left with its RowSenseAmpCache state unaltered (except the row's dirty flag will be cleared):

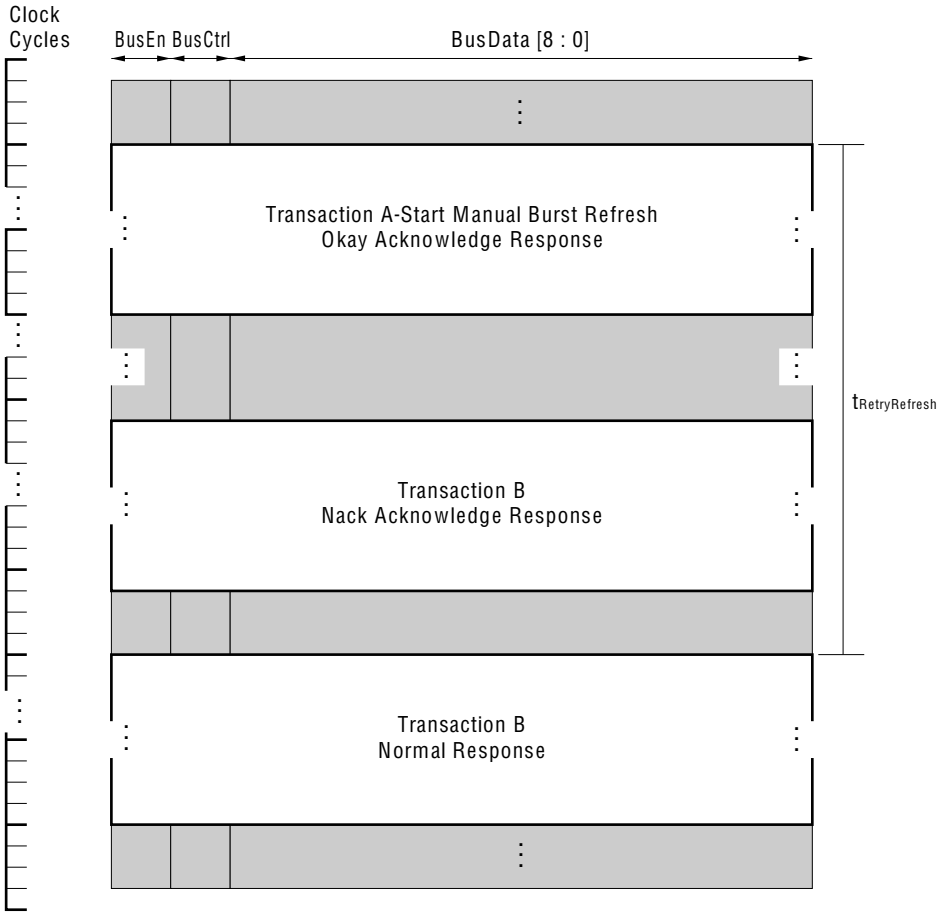
$$t_{RetryRefreshClean} = t_{WriteHit} + t_{RefRequestIdleOverHead} + 5xt_{LessRowRefreshOverHead} + 5x(2xRowPrecharge[0:4] + RowSense[0:4] + RowImpRestore[0:4]) \quad \text{(Eq 8-5)}$$

$$t_{RetryRefreshDirty} = t_{WriteHit} + t_{RefRequestIdleOverHead} + 5xt_{LessRowRefreshOverHead} + 5x(2xRowPrecharge[0:4] + RowSense[0:4] + RowImpRestore[0:4] + t_{RowExpRestore}) \quad \text{(Eq 8-6)}$$

RowPrecharge[0:4], RowSense[0:4], RowImpRestore[0:4] are the value of every Register.

When a transaction initiates a manual burst refresh in an RDRAM (transaction "A" in the figure below), the RDRAM will Nack all further transactions directed to in during the $t_{RetryRefresh}$ interval after. No information from these Nacked transactions will be retained after the $t_{RetryRefresh}$ interval. After the $t_{RetryRefresh}$ interval, transactions will be handled in a normal fashion. The detailed functional description is provided in **"Rambus DRAM user's manual (Reference Manual)"**.

Figure 8-2. Transaction Holdoff Due to Burst Refresh



8.3 Retry Component Intervals

The t_{RETRY} and $t_{RasAgain}$ intervals are built from the $t_{RowOverHead}$, $t_{RowPrecharge}$, $t_{RowSense}$, $t_{RowImprestore}$, $t_{RowExprestore}$, $t_{RefRequestIdleOverHead}$, $t_{LessRowRefreshOverHead}$, and $t_{HoldOff}$ intervals. All eight intervals are measured in t_{CYCLE} units, and thus scale with the clock frequency.

The $t_{RowOverHead}$, $t_{RefRequestIdleOverHead}$, and $t_{LessRowRefreshOverHead}$ intervals consist of the RowMiss and Refresh state machine overheads. The remaining five intervals represent the width of intervals used for timing core operations. These core operations have minimum times measured in nanosecond units (this is shown in the "core timing(ns)" columns in the table below). The five intervals are composed of a fixed part and a variable (programmable) part. If the clock frequency is reduced, the variable part may be reduced so the sum of the fixed and variable parts remain greater than or equal to the minimum core operation time (in nanoseconds).

Table 8-1. Retry Components

Delay Parameter	Fixed Part (overhead) and Variable Part[Note 1]	18M RDRAM	
		t_{CYCLE} Units (4 ns)	core timing (ns) with t_{CYCLE} = 4ns
$t_{RowOverHead}$	Row overhead	7	28
	–	n/a	
$t_{RefRequestIdleOverHead}$	Row overhead	14	56
	–	n/a	
$t_{LessRowRefreshOverHead}$	Row overhead	20	80
	–	n/a	
$t_{RowPrecharge}$	RowPrecharge overhead	6	28
	RowPrecharge[4:0]	1	
$t_{RowSense}$	RowSense overhead	1	32
	RowSense[4:0]	7	
$t_{RowImprestore}$	RowImpRestore overhead	5	60
	RowImpRestore[4:0]	10	
$t_{RowExprestore}$	RowExpRestore overhead	4	32
	RowExpRestore[4:0][Note 2]	4	
$t_{HoldOff}$	HoldOff overhead	1	8
	RowPrecharge [4:0]	1	

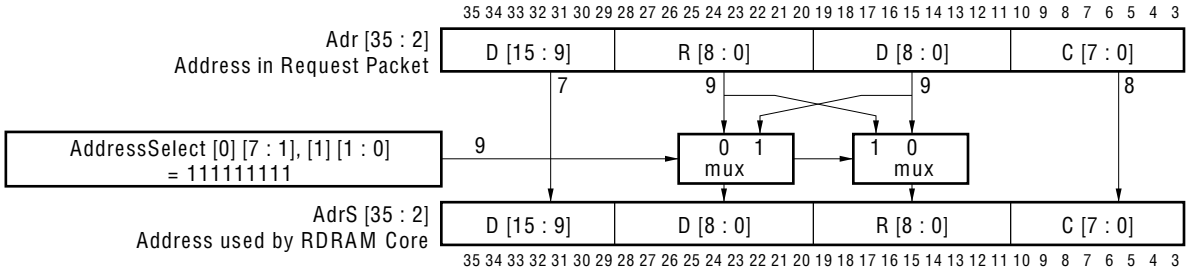
Notes 1. The variable part is programmed into the indicated field of the RasInterval register.

2. The RowPrecharge [4:0] field is used for both the precharge interval and the holdoff interval.

9. AddressMapping

The address space decoding logic contained in a 18M RDRAM is shown in the following figure. The initiating device places a 33 bit physical octbyte address Adr[35:3] on the Channel. This address is received by the RDRAM slave. The AddressSelect[1][1:0], [0][7:1] control register allows individual bits of the Adr[28:20] and Adr [19:11] fields to be swapped to produce the AdrS[28:20] and AdrS[19:11] fields. The Adr[35:29] and Adr[10:3] fields pass through unaltered to the AdrS[35:29] and AdrS[10:3] fields. The figure shows the case when AddressSelect[0][7:1],[1][1:0] = 111111111, and the two nine bit address fields are exchanged. The detailed functional description is provided in **"Rambus DRAM user's manual (Reference Manual)"**.

Figure 9-1. AddressMapping Hardware



10. Electrical Characteristics (Preliminary)

Absolute Maximum Ratings

Symbol	Parameter	MIN.	MAX.	Unit	Note
$V_{I,ABS}$	Voltage applied to any RSL pin with respect to GND	–0.5	V_{DD}+0.5	V	
$V_{I,TTL,ABS}$	Voltage applied to any TTL pin with respect to GND	–0.5	V_{DD}+0.5	V	
$V_{DD,ABS}$	Voltage on V_{DD} with respect to GND	–0.5	$V_{DD,MAX}$+1.0	V	
T_{OPT}	Operation temperature	0	+70	°C	1
T_{STORE}	Storage temperature	–55	+125	°C	

Caution The following table represents stress ratings only, and functional operation at the maximums is not guaranteed. Extended exposure to the maximum ratings may affect device reliability. Furthermore, although devices contain protective circuitry to resist damage from static electric discharge, always take precautions to avoid high static voltages or electric fields.

Note 1 This parameter apply at the status of using 50% Rambus channel by Read or Write and a transverse air flow greater than 1.5m/s maintained.

Thermal Parameters

Symbol	Parameter	MIN.	MAX.	Unit
T_J	Junction operating temperature		100	°C
Θ_{JC}	Junction-to-Case thermal resistance		5	°C/W

Capacitance

Symbol	Parameter	MIN.	MAX.	Unit
$C_{I/O}$	Low-swing input/output parasitic capacitance		2	pF
$C_{I,TTL}$	TTL input parasitic capacitance		8	pF

Power Consumption

Mode	Parameter		MIN.	MAX.	Unit
I_{CC1}	Standby Current	–A45		125	mA
		–A50		135	
I_{CC2}	Active Current	–A45		350	mA
		–A50		380	
I_{CC3}	Read Operation Current	–A45		440	mA
	(Burst Length = 256)	–A50		480	
I_{CC4}	Write Operation Current	–A45		440	mA
	(Burst Length = 256)	–A50		480	

Caution These do not include the I_{OL} current passing through the low-swing pins to ground.

Recommended Operating Conditions

Symbol	Parameter	MIN.	MAX.	Unit
V_{DD}, V_{DDA}	Supply voltage	3.15	3.45	V
V_{REF}	Reference voltage	1.95	2.15	V
V_{swing}	Input voltage range	1.0	1.4	V
V_{IH}	High level input voltage	V_{REF}+0.5	V_{REF}+0.7	V
V_{IL}	Low level input voltage	V_{REF}–0.7	V_{REF}–0.5	V
$V_{IH, TTL}$	High level TTL input voltage	2.0	V_{DD}+0.5	V
$V_{IL, TTL}$	Low level TTL input voltage	–0.5	+0.8	V

DC Characteristics (Recommended operating conditions unless otherwise noted)

Symbol	Parameter	Conditions	MIN.	MAX.	Unit
I_{REF}	V_{REF} current	V_{REF}=Maximum	–10	+10	μA
I_{OH}	High level output current	$0 \le V_{OUT} \le V_{DD}$	–10	+10	μA
I_{OL}	Low level output current	V_{OUT}=1.6 V		25	mA
$I_{I, TTL}$	TTL input leakage current	$0 \le V_{I, TTL} \le V_{DD}$	–10	+10	μA
$V_{OH, TTL}$	High level TTL output voltage	$I_{OH, TTL}$=–0.25 mA	2.4	V_{DD}	V
$V_{OL, TTL}$	Low level TTL output voltage	$I_{OL, TTL}$=1.0 mA	0	0.4	V

Recommended Timing Conditions

Symbol	Parameter		MIN.	MAX.	Unit
t_{PAUSE}	Pause time after Power On			200	μs
t_{CR}, t_{CF}	TxClk and RxClk input rise and fall times		0.3	0.7	ns
t_{CYCLE}	TxClk and RxClk cycle times	–A45	4.45	5	ns
		–A50	4	5	ns
t_{TICK}	Transport time per bit per pin (this timing interval is synthesized by the RDRAM's internal clock generator)		t_{CYCLE}/2	t_{CYCLE}/2	ns
t_{CH}, t_{CL}	TxClk and RxClk high and low times		47%	53%	t_{CYCLE}
t_{TR}	TxClk-RxClk differential		0.25	0.7	ns
t_{SD}	SIn-to-SOut propagation delay			50	ns
t_Q	TxClk-to-Data/Control output time		1–0.45	1+0.45	t_{CYCLE}/4
t_S	Data/Control-to-RxClk setup time		0.45		t_{CYCLE}/4
t_H	RxClk-to-Data/Control hold time		0.45		t_{CYCLE}/4
t_{REF}	Refresh interval			17	ms
t_{LOCK}	RDRAM internal clock generator lock time		750		t_{CYCLE}

Transaction Timing Characteristics

Symbol	Parameter	MIN.	Unit
$t_{PostRegWriteDelay}$	Delay from the end of the current transaction to the beginning of the next transaction if the current transaction is a write to register space and the next transaction is made to the same device. Use zero delay if the next transaction is to a different device.	6	t_{CYCLE}
$t_{PostMemWriteDelay}$	Delay from the end of the current transaction to the beginning of the next transaction if the current transaction is a write to memory space and the next transaction is made to the same device. Use zero delay if the next transaction is to a different device.	4	t_{CYCLE}
$t_{PostMemReadDelay}$	Delay from the end of the current memory read transaction to the beginning of the next transaction.	2	t_{CYCLE}
$t_{SerialReadOffSet}$	Delay from the beginning of a serial address subpacket or serial control packet to the beginning of the read data subpacket (octbyte) with which it is associated.	12	t_{CYCLE}
$t_{SerialWriteOffSet}$	Delay from the beginning of a serial address subpacket or serial control packet to the beginning of the write data subpacket (octbyte) with which it is associated.	8	t_{CYCLE}

Data and Transaction Latency Characteristics

Symbol	Parameter	MIN.	Unit	Notes
$t_{ReadDelay}$	Delay from the end of a read request packet to the beginning of the read data packet.	7	t_{CYCLE}	1
$t_{WriteDelay}$	Delay from the end of a write request packet to the beginning of the write data packet.	1	t_{CYCLE}	2

Notes 1. $t_{ReadDelay}$ is programmed to its minimum value.

2. $t_{WriteDelay}$ is programmed to its minimum value.

Hit, Retry and Miss Delay Characteristics

Symbol	Parameter		MIN.	Unit	Notes
$t_{ReadHit}$	Start of request packet to start of read data packet for row hit (Okay).		10	t_{CYCLE}	1
$t_{WriteHit}$	Start of request packet to start of write data packet for row hit (Okay).		4	t_{CYCLE}	1
$t_{RetrySensedClean}$	Start of request packet for row miss (Nack) to start of request packet for row hit (Okay). The previous row is unmodified.		22	t_{CYCLE}	2
$t_{RetrySensedDirty}$	Start of request packet for row miss (Nack) to start of request packet for row hit (Okay). The previous row is modified.		30	t_{CYCLE}	2
$t_{RetryRefresh}$	Start of request packet for row miss (Nack) to start of request packet for row hit (Okay).	Clean	213	t_{CYCLE}	2
		Dirty	221		
$t_{ReadMiss}$	Start of request packet for row miss (Nack) to start of Read Data packet for row hit (Okay).		32	t_{CYCLE}	3
$t_{WriteMiss}$	Start of request packet for row miss (Nack) to start of Write Data packet for row hit (Okay).		26	t_{CYCLE}	3

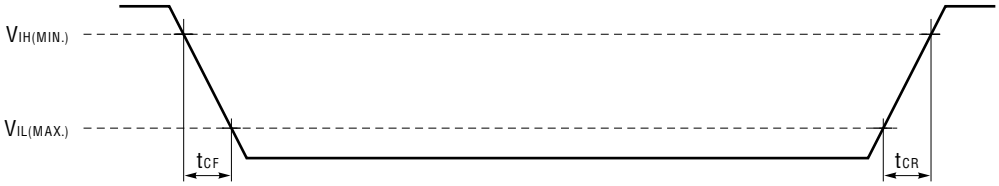
Notes 1. Programmable

2. $t_{RowExprestore}$, $t_{Precharge}$, and t_{Sense} are programmed to there minimum value.

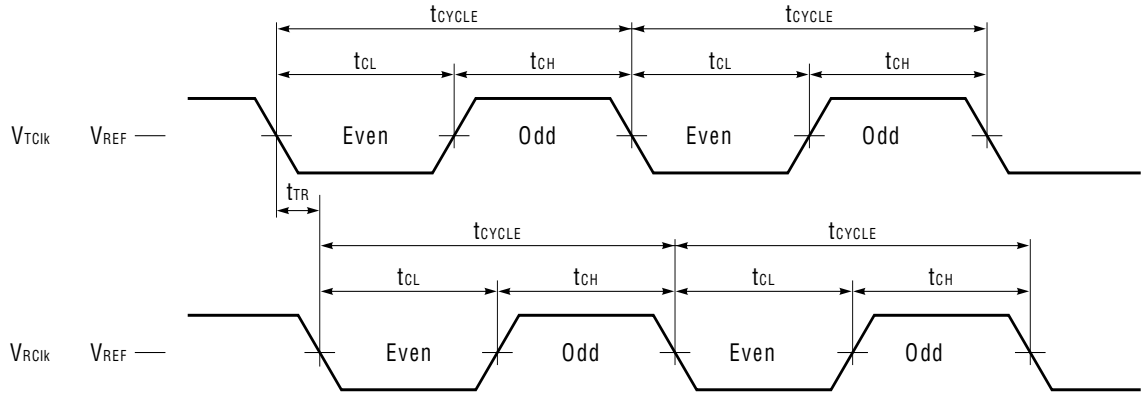
3. Calculated with $t_{RetrySensedClean(MIN)}$.

Rise/Fall Timing Chart

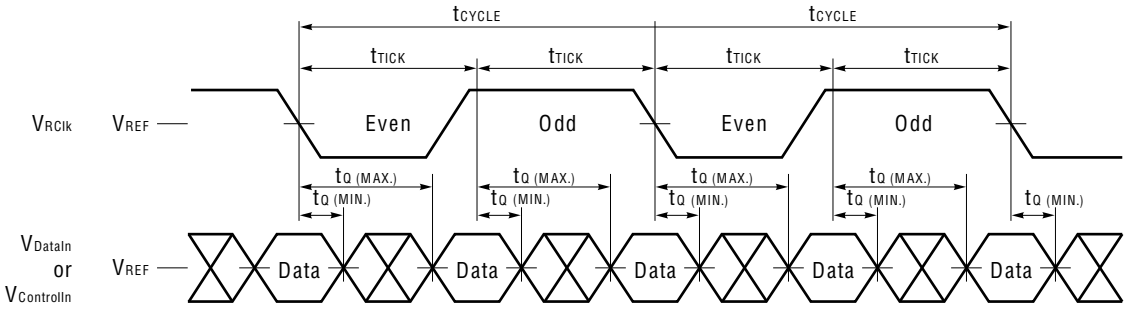
V_{RxClk}, V_{TxClk}



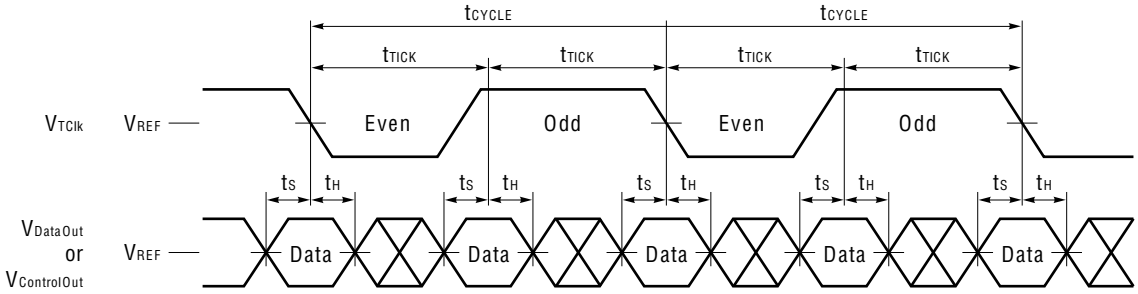
Clock Timing Chart



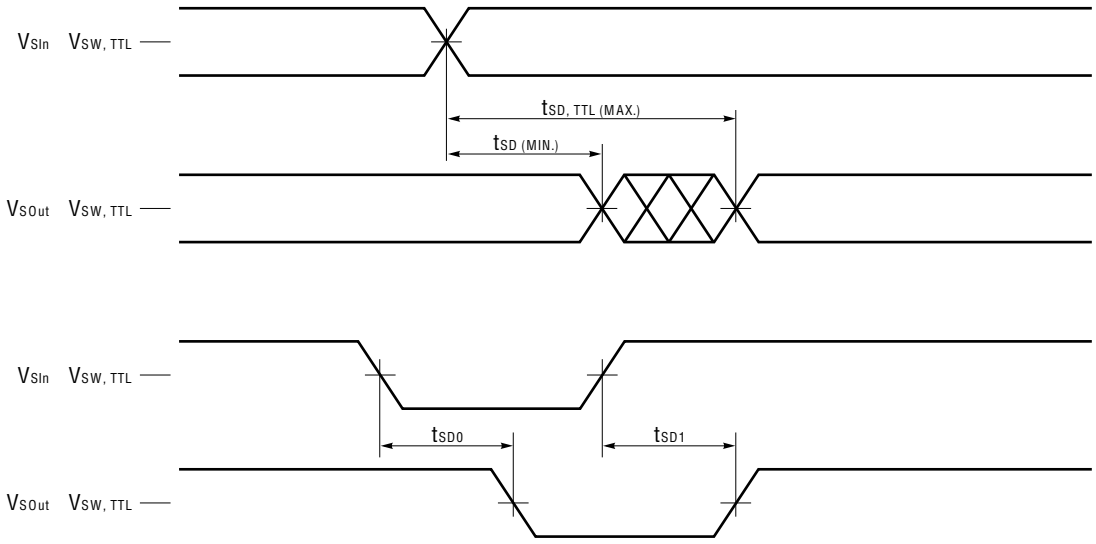
Receive Data Timing Chart



Transmit Data Timing Chart



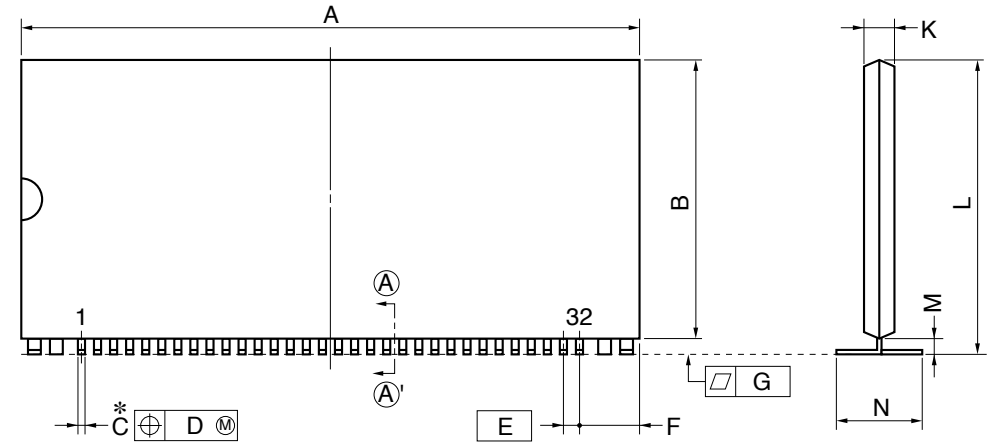
Serial Configuration Pin Timing Chart



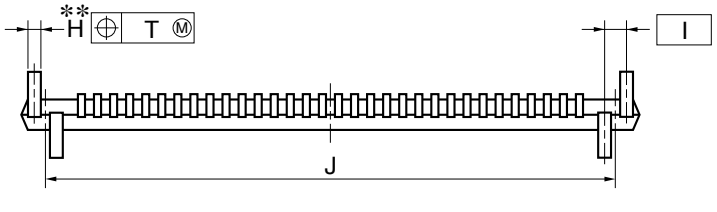
Remark $V_{SW, TTL}$ = 1.5 V

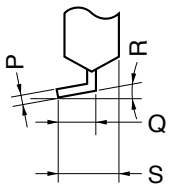
11. Package Drawings

32 PIN PLASTIC SVP (11×25)



Detail of Ⓐ - Ⓐ' part





NOTE

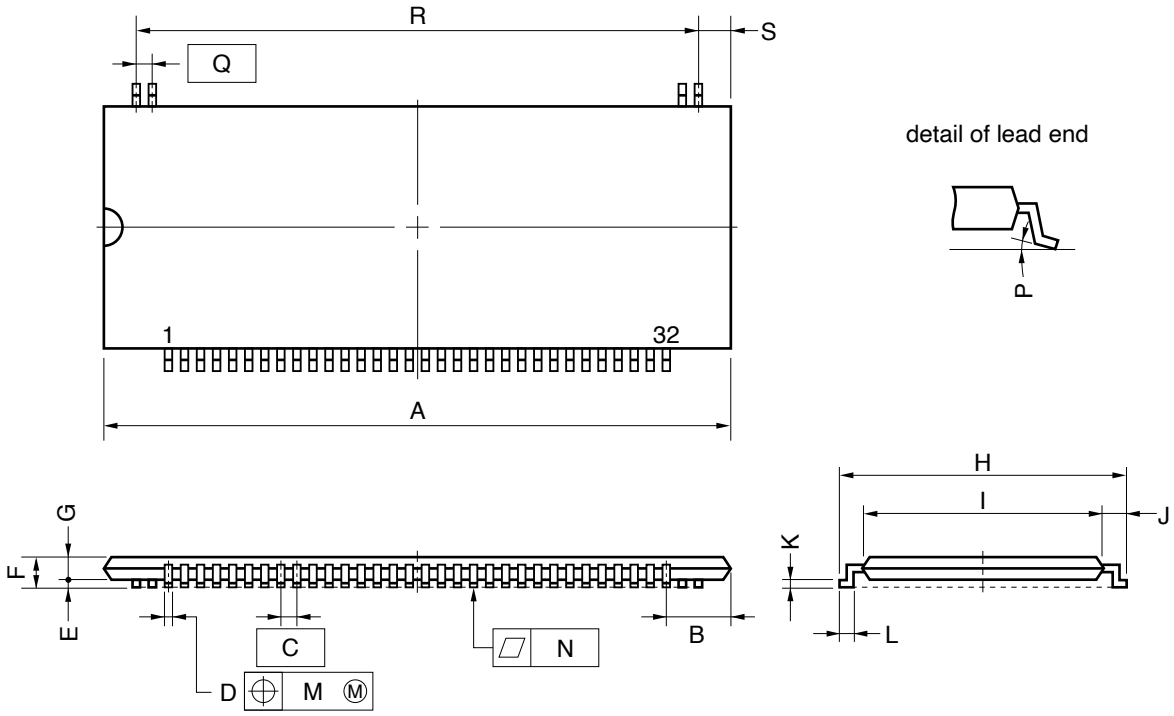
∗ Each I/O lead centerline is located within 0.13 mm (0.005 inch) of its true position (T.P.) at maximum material condition.

∗∗ Each support lead centerline is located within 0.18 mm (0.007 inch) of its true position (T.P.) at maximum material condition.

ITEM	MILLIMETERS	INCHES
A	25.30 MAX.	0.996 MAX.
B	11.0±0.1	0.433±0.004
C	0.24±0.06	$0.009^{+0.003}_{-0.002}$
D	0.13	0.005
E	0.65 (T.P.)	0.026 (T.P.)
F	2.575 MAX.	0.102 MAX.
G	0.10	0.004
H	0.52±0.06	0.020±0.002
I	0.9 (T.P.)	0.035 (T.P.)
J	23.20	0.913
K	1.25	0.049
L	11.80 MAX.	0.465 MAX.
M	0.5±0.1	$0.020^{+0.004}_{-0.005}$
N	3.70 MAX.	0.146 MAX.
P	$0.17^{+0.025}_{-0.015}$	0.007±0.001
Q	0.9±0.25	$0.035^{+0.011}_{-0.010}$
R	$3^{\circ}{}^{+7^{\circ}}_{-3^{\circ}}$	$3^{\circ}{}^{+7^{\circ}}_{-3^{\circ}}$
S	1.90 MAX.	0.075 MAX.
T	0.18	0.007

S32VN-65-9

72/36 PIN PLASTIC SSOP TYPE



NOTE

Each lead centerline is located within 0.13 mm (0.005 inch) of its true position (T.P.) at maximum material condition.

ITEM	MILLIMETERS	INCHES
A	25.30 MAX.	0.996 MAX.
B	2.575 MAX.	0.102 MAX.
C	0.65 (T.P.)	0.026 (T.P.)
D	0.24 ± 0.06	$0.009^{+0.003}_{-0.002}$
E	0.25 ± 0.05	$0.010^{+0.002}_{-0.003}$
F	1.6 MAX.	0.063 MAX.
G	1.25	0.049
H	13.0 ± 0.2	0.512 ± 0.008
I	11.0 ± 0.1	0.433 ± 0.004
J	1.0 ± 0.2	$0.039^{+0.009}_{-0.008}$
K	$0.17^{+0.025}_{-0.015}$	0.007 ± 0.001
L	0.5 ± 0.1	$0.020^{+0.004}_{-0.005}$
M	0.13	0.005
N	0.10	0.004
P	$3^{\circ}{}^{+7^{\circ}}_{-3^{\circ}}$	$3^{\circ}{}^{+7^{\circ}}_{-3^{\circ}}$
Q	0.65 (T.P.)	0.026 (T.P.)
R	22.75	0.896
S	1.275 MAX.	0.051 MAX.

P32G6-65A