


The ALTMEMPHY megafunction creates the datapath between the memory device and the memory controller, and user logic in various Altera devices. The ALTMEMPHY megafunction GUI helps you configure multiple variations of a memory interface. You can then connect the ALTMEMPHY megafunction variation with either a user-designed controller or with the Altera high-performance controller. In addition, the ALTMEMPHY megafunction and the Altera high-performance controller are available for half-rate DDR3 SDRAM interfaces.

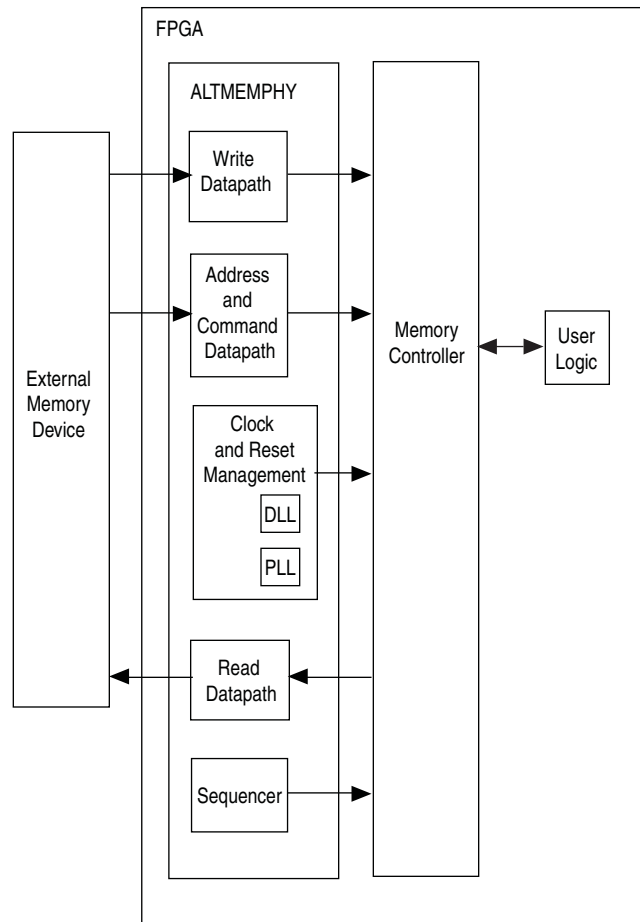
 If the ALTMEMPHY megafunction does not meet your requirements, you can also create your own memory interface datapath using the ALTDLL and ALTDQ\_DQS megafunctions, available in the Quartus II software. However, you are then responsible for every aspect of the interface, including timing analysis and debugging.

This chapter describes the DDR3 SDRAM ALTMEMPHY megafunction, which uses AFI as the interface between the PHY and the controller.

## Block Description

Figure 2-1 shows the major blocks of the ALTMEMPHY megafunction and how it interfaces with the external memory device and the controller. The ALTPLL megafunction is instantiated inside the ALTMEMPHY megafunction, so that you do not need to generate the clock to any of the ALTMEMPHY blocks.

**Figure 2-1. ALTMEMPHY Megafunction Interfacing with the Controller and the External Memory**



The ALTMEMPHY megafunction comprises the following blocks:

- Write datapath
- Address and command datapath
- Clock and reset management, including DLL and PLL
- Sequencer for calibration
- Read datapath

The major advantage of the ALTMEMPHY megafunction is that it supports an initial calibration sequence to remove process variations in both the Altera device and the memory device. In Arria series devices, the DDR3 SDRAM ALTMEMPHY calibration process centers the resynchronization clock phase into the middle of the captured data valid window to maximize the resynchronization setup and hold margin. During the user operation, the VT tracking mechanism eliminates the effects of VT variations on resynchronization timing margin.

## Calibration

The sequencer performs calibration to find the optimal clock phase for the memory interface.

For information about calibration, refer to [ALTMEMPHY Calibration Stages](#).

## Address and Command Datapath

This topic discusses the address and command datapath.

### Arria II GX Devices

The address and command datapath is responsible for taking the address and command outputs from the controller and converting them from half-rate clock to full-rate clock. Two types of addressing are possible:

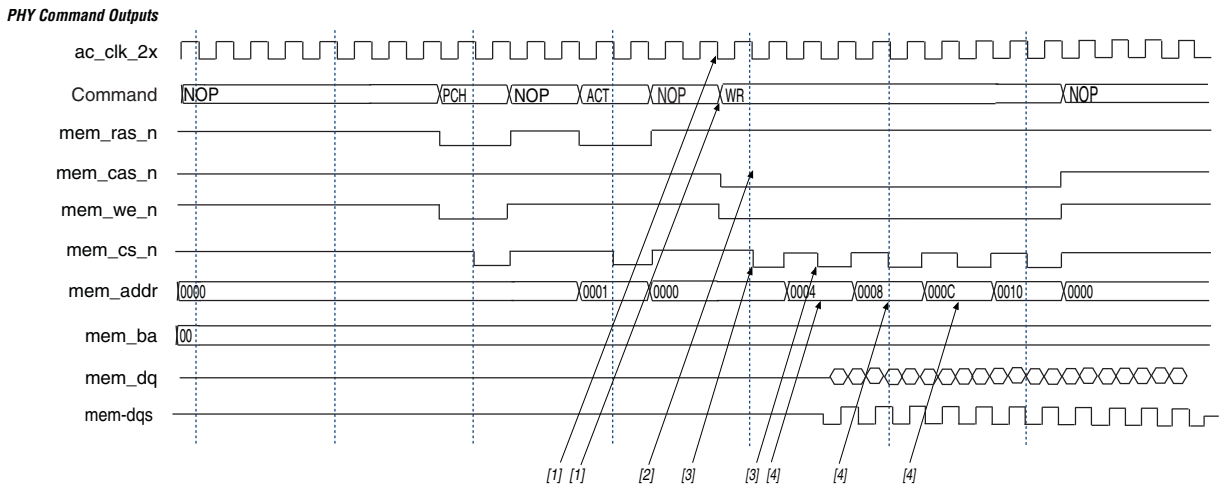
- 1T (full rate)—the duration of the address and command is a single memory clock cycle (`mem_clk_2x`, [Figure 2-2](#)). This applies to all address and command signals in full-rate designs or `mem_cs_n`, `mem_cke`, and `mem_odt` signals in half-rate designs.
- 2T (half rate)—the duration of the address and command is two memory clock cycles. For half-rate designs, the ALTMEMPHY megafunction supports only a burst size of four, which means the burst size on the local interface is always set to 1. The size of the data is  $4n$ -bits wide on the local side and is  $n$ -bits wide on the memory side. To transfer all the  $4n$ -bits at the double data rate, two memory-clock cycles are required. The new address and command can be issued to memory every two clock cycles. This scheme applies to all address and command signals, except for `mem_cs_n`, `mem_cke`, and `mem_odt` signals in half-rate mode.



Refer to [Table 2-1 on page 2-6](#) to see the frequency relationship of `mem_clk_2x` with the rest of the clocks.

Figure 2-2 shows a 1T chip select signal (`mem_cs_n`), which is active low, and disables the command in the memory device. All commands are masked when the chip-select signal is inactive. The `mem_cs_n` signal is considered part of the command code.


**Figure 2-2. Arria II GX Address and Command Datapath**




The command interface is made up of the signals `mem_ras_n`, `mem_cas_n`, `mem_we_n`, `mem_cs_n`, `mem_cke`, and `mem_odt`.

The waveform in Figure 2-2 shows a NOP command followed by five back-to-back write commands. The following sequence corresponds with the numbered items in Figure 2-2.

1. The commands are asserted either on the rising edge of `ac_clk_2x`. The `ac_clk_2x` is derived from either `mem_clk_2x` ( $0^\circ$ ), `write_clk_2x` ( $270^\circ$ ), or the inverted variations of those two clocks (for  $180^\circ$  and  $90^\circ$  phase shifts). This depends on the setting of the address and command clock in the ALTMEMPHY parameter editor. Refer to "Address and Command Datapath" on page 2-3 for illustrations of this clock in relation to the `mem_clk_2x` or `write_clk_2x` signals.
2. All address and command signals (except for `mem_cs_n`, `mem_cke`, and `mem_odt` signals) remain asserted on the bus for two clock cycles, allowing sufficient time for the signals to settle.
3. The `mem_cs_n`, `mem_cke`, and `mem_odt` signals are asserted during the second cycle of the address/command phase. By asserting the chip-select signal in alternative cycles, back-to-back read or write commands can be issued.
4. The address is incremented every other `ac_clk_2x` cycle.

 The `ac_clk_2x` clock is derived from either `mem_clk_2x` (when you choose  $0^\circ$  or  $180^\circ$  phase shift) or `write_clk_2x` (when you choose  $90^\circ$  or  $270^\circ$  phase shift).

 The address and command clock can be  $0$ ,  $90$ ,  $180$ , or  $270^\circ$  from the system clock.

## Clock and Reset Management

The clocking and reset block is responsible for clock generation, reset management, and phase shifting of clocks. It also has control of clock network types that route the clocks, which is handled in the `<variation_name>_alt_mem_phy_clk_reset` module in the `<variation_name>_alt_mem_phy.v/.vhd` file.

### Clock Management

The clock management feature allows the ALTMEMPHY megafunction to work out the optimum phase during calibration, and to track voltage and temperature variation relies on phase shifting the clocks relative to each other.



Certain clocks require phase shifting during the ALTMEMPHY megafunction operation.

You can implement clock management circuitry using PLLs and DLLs.

The ALTMEMPHY MegaWizard Plug-In Manager automatically generates an ALTPLL megafunction instance. The ALTPLL megafunction generates the different clock frequencies and relevant phases used within the ALTMEMPHY megafunction.

The available device families have different PLL capabilities. The minimum PHY requirement is to have 16 phases of the highest frequency clock. The PLL uses **With No Compensation** option to minimize jitter. Changing the PLL compensation to a different operation mode may result in inaccurate timing results.

The input clock to the PLL does not have any other fan-out to the PHY, so you do not have to use a global clock resource for the path between the clock input pin to the PLL. You must use the PLL located in the same device quadrant or side as the memory interface and the corresponding clock input pin for that PLL, to ensure optimal performance and accurate timing results from the Quartus II software.

You must choose a PLL and PLL input clock pin that are located on the same side of the device as the memory interface to ensure minimal jitter. Also, ensure that the input clock to the PLL is stable before the PLL locks. If not, you must perform a manual PLL reset (by driving the `global_reset_n` signal low) and relock the PLL to ensure that the phase relationship between all PLL outputs is properly set.



If the design cascades PLLs, the source (upstream) PLL should have a low-bandwidth setting, and the destination (downstream) PLL should have a high-bandwidth setting. Adjacent PLLs cascading is recommended to reduce clock jitter.



For more information about the VCO frequency range and the available phase shifts, refer to the *Clock Networks and PLLs* chapter in the respective device family handbook.

Table 2-1 shows the clock outputs that Arria II GX devices use.

**Table 2-1. DDR3 SDRAM Clocking in Arria II GX Devices (Part 1 of 2)**

Clock Name <sup>(1)</sup>	Postscale Counter	Phase (Degrees)	Clock Rate	Clock Network Type		Notes
				All Quadrants	Any 3 Quadrants <sup>(2)</sup>	
phy_clk_1x and aux_half_rate_clk	C0	0°	Half-Rate	Global	Global	The only clocks parameterizable for the ALTMEMPHY megafunction. These clocks also feed into a divider circuit to provide the PLL scan_clk signal (for reconfiguration) that must be lower than 100 MHz.
mem_clk_2x and aux_full_rate_clk	C1	0°	Full-Rate	Global	Regional <sup>(3)</sup> Global <sup>(4)</sup>	This clock is for clocking DQS and as a reference clock for the memory devices.
mem_clk_1x	C2	0°	Half-Rate	Global	Regional	This clock is for clocking DQS and as a reference clock for the memory devices.
write_clk_2x	C3	-90°	Full-Rate	Global	Regional	This clock is for clocking the data out of the DDR I/O (DDIO) pins in advance of the DQS strobe (or equivalent). As a result, its phase leads that of the mem_clk_2x by 90°.
ac_clk_2x	C3	-90°	Full-Rate	Global	Regional	Address and command clock. The ac_clk_2x clock is derived from either mem_clk_2x (when you choose 0° or 180° phase shift) or write_clk_2x (when you choose 90° or 270° phase shift). Refer to “Address and Command Datapath” on page 2-3 for illustrations of the address and command clock relationship with the mem_clk_2x or write_clk_2x signals.
cs_n_clk_2x	C3	-90°	Full-Rate	Global	Global	Memory chip-select clock. The cs_n_clk_2x clock is derived from ac_clk_2x.
resync_clk_2x	C4	Calibrated	Full-Rate	Global	Regional	Clocks the resynchronization registers after the capture registers. Its phase is adjusted to the center of the data valid window across all the DQS-clocked DDIO groups.

**Table 2-1. DDR3 SDRAM Clocking in Arria II GX Devices (Part 2 of 2)**

Clock Name <sup>(1)</sup>	Postscale Counter	Phase (Degrees)	Clock Rate	Clock Network Type		Notes
				All Quadrants	Any 3 Quadrants <sup>(2)</sup>	
measure_clk_2x	C5	Calibrated	Full-Rate	Global	Regional	This clock is for VT tracking. This free-running clock measures relative phase shifts between the internal clock(s) and those being fed back through a mimic path. As a result, the ALTMEMPHY megafunction can track VT effects on the FPGA and compensate for the effects.

**Note to Table 2-1:**

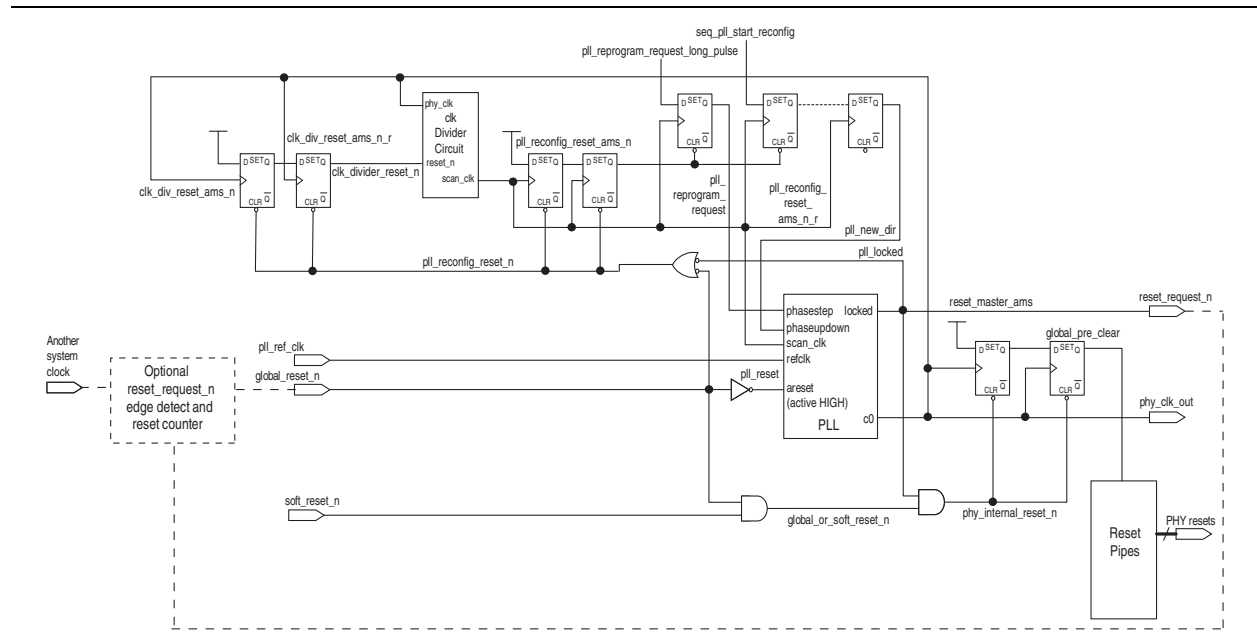
- (1) The `_1x` clock represents a frequency that is half of the memory clock frequency; the `_2x` clock represents the memory clock frequency.
- (2) The default clock network type is Global, however you can specify a regional clock network to improve clock jitter if your design uses any three quadrants.
- (3) For `mem_clk2x`.
- (4) For `aux_full_rate_clk`.

## Reset Management

Figure 2-3 shows the main features of the reset management block for the DDR3 SDRAM PHY. You can use the `pll_ref_clk` input to feed the optional `reset_request_n` edge detect and reset counter module. However, this requires the `pll_ref_clk` signal to use a global clock network resource.

There is a unique reset metastability protection circuit for the clock divider circuit because the `phy_clk` domain reset metastability protection registers have fan-in from the `soft_reset_n` input so these registers cannot be used.

Figure 2-3. ALTMEMPHY Reset Management Block for Arria II GX Devices



## Read Datapath

This topic discusses the read datapath.

### Arria II GX Devices

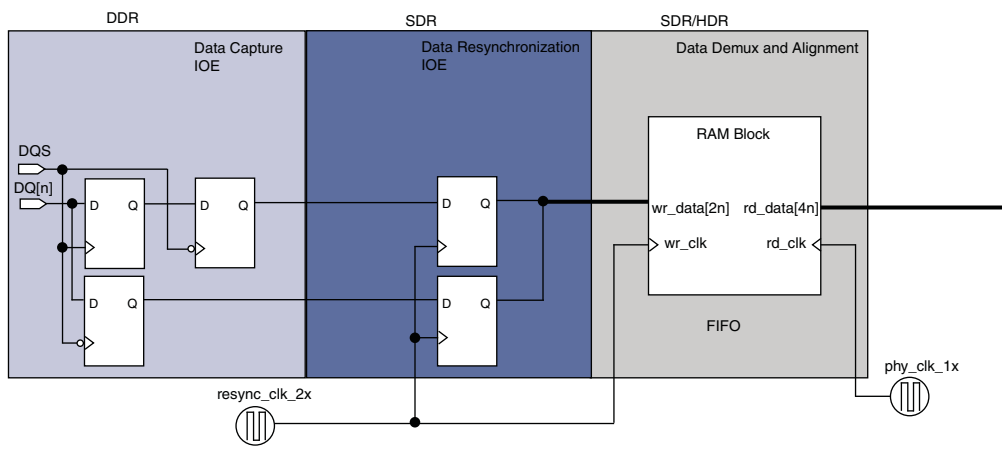
The read datapath logic captures data sent by the memory device and subsequently aligns the data back to the system clock domain. The read datapath for DDR3 SDRAM consists of the following three main blocks:

- Data capture
- Data resynchronization
- Data demultiplexing and alignment

As the DQS/DQS<sub>n</sub> signal is not continuous, the PHY also has postamble protection logic to ensure that any glitches on the DQS input signals at the end of the read postamble time do not cause erroneous data to be captured as a result of postamble glitches.

Figure 2-4 shows the order of the functions performed by the read datapath and the frequency at which the read data is handled.

**Figure 2-4. DDR3 SDRAM Read Datapath in Arria II GX Devices**



### Data Capture and Resynchronization

The data capture and resynchronization registers for Arria II GX devices are implemented in the I/O element (IOE) to achieve maximum performance. Data capture and resynchronization is the process of capturing the read data (DQ) with the DQS/DQSn strobes and resynchronizing the captured data to an internal free-running full-rate clock supplied by the enhanced PLL. The resynchronization clock is an intermediate clock whose phase shift is determined during the calibration stage. The captured data (`rdata_p_captured` and `rdata_n_captured`) is synchronized to the resynchronization clock (`resync_clk_2x`), refer to [Figure 2-4](#). For Arria II GX devices, the ALTMEMPHY instances an ALTDQ\_DQS megafunction that instantiates the required IOEs for all the DQ and DQS pins.

### Data Demultiplexing

Data demultiplexing is the process of changing the SDR data into HDR data. Data demultiplexing is required to bring the frequency of the resynchronized data down to the frequency of the system clock, so that data from the external memory device can ultimately be brought into the FPGA controller clock domain. Before data capture, the data is DDR and  $n$ -bit wide. After data capture, the data is SDR and  $2n$ -bit wide. After data demuxing, the data is HDR of width  $4n$ -bits wide. The system clock frequency is half the frequency of the memory clock. Demultiplexing is achieved using a dual-port memory with a  $2n$ -bit wide write-port operating on the resynchronization clock (SDR) and a  $4n$ -bit wide read-port operating on the PHY clock (HDR). The basic principle of operation is that data is written to the memory at the SDR rate and read from the memory at the HDR rate while incrementing the read- and write-address pointers. As the SDR and HDR clocks are generated, the read and write pointers are continuously incremented by the same PLL, and the  $4n$ -bit wide read data follows the  $2n$ -bit wide write data with a constant latency

### Read Data Alignment

Data alignment is the process controlled by the sequencer to ensure the correct captured read data is present in the same half-rate clock cycle at the output of the read data DPRAM. Data alignment is implemented using memory blocks in the core of devices.

### Postamble Protection

A dedicated postamble register controls the gating of the shifted DQS signal that clocks the DQ input registers at the end of a read operation. Any glitches on the DQS input signals at the end of the read postamble time do not cause erroneous data to be captured as a result of postamble glitches. The postamble path is also calibrated to determine the correct clock cycle, clock phase shift, and delay chain settings.

## ALTMEMPHY Signals

This section describes the ALTMEMPHY megafunction signals for DDR3 SDRAM variants.

Table 2-2 through Table 2-4 show the signals.



Signals with the prefix `mem_` connect the PHY with the memory device; ports with the prefix `ctl_` connect the PHY with the controller.

The signal lists include the following signal groups:

- I/O interface to the SDRAM devices
- Clocks and resets
- External DLL signals
- User-mode calibration OCT control
- Write data interface
- Read data interface
- Address and command interface
- Calibration control and status interface
- Debug interface

**Table 2-2. Interface to the DDR3 SDRAM Devices (Part 1 of 2) <sup>(1)</sup>**

Signal Name	Type	Width <sup>(2)</sup>	Description
<code>mem_addr</code>	Output	<code>MEM_IF_ROWADDR_WIDTH</code>	The memory row and column address bus.
<code>mem_ba</code>	Output	<code>MEM_IF_BANKADDR_WIDTH</code>	The memory bank address bus.
<code>mem_cas_n</code>	Output	1	The memory column address strobe.
<code>mem_cke</code>	Output	<code>MEM_IF_CS_WIDTH</code>	The memory clock enable.
<code>mem_clk</code>	Bidirectional	<code>MEM_IF_CLK_PAIR_COUNT</code>	The memory clock, positive edge clock. <sup>(3)</sup>
<code>mem_clk_n</code>	Bidirectional	<code>MEM_IF_CLK_PAIR_COUNT</code>	The memory clock, negative edge clock.
<code>mem_cs_n</code>	Output	<code>MEM_IF_CS_WIDTH</code>	The memory chip select signal.
<code>mem_dm</code>	Output	<code>MEM_IF_DM_WIDTH</code>	The optional memory DM bus.
<code>mem_dq</code>	Bidirectional	<code>MEM_IF_DWIDTH</code>	The memory bidirectional data bus.
<code>mem_dqs</code>	Bidirectional	<code>MEM_IF_DWIDTH/</code> <code>MEM_IF_DQ_PER_DQS</code>	The memory bidirectional data strobe bus.
<code>mem_dqs_n</code>	Bidirectional	<code>MEM_IF_DWIDTH/</code> <code>MEM_IF_DQ_PER_DQS</code>	The memory bidirectional data strobe bus.
<code>mem_odt</code>	Output	<code>MEM_IF_CS_WIDTH</code>	The memory on-die termination control signal.

**Table 2-2. Interface to the DDR3 SDRAM Devices (Part 2 of 2) (1)**

Signal Name	Type	Width (2)	Description
mem_ras_n	Output	1	The memory row address strobe.
mem_reset_n	Output	1	The memory reset signal. This signal is derived from the PHY's internal reset signal, which is generated by gating the global reset, soft reset, and the PLL locked signal.
mem_we_n	Output	1	The memory write enable signal.
mem_ac_parity (4)	Output	1	The address or command parity signal generated by the PHY and sent to the DIMM.
parity_error_n (4)	Output	1	The active-low signal that is asserted when a parity error occurs and stays asserted until the PHY is reset.
mem_err_out_n (4)	Input	1	The signal sent from the DIMM to the PHY to indicate that a parity error has occurred for a particular cycle.

**Notes to Table 2-2:**

- (1) Connected to I/O pads.
- (2) Refer to Table 2-5 for parameter description.
- (3) Output is for memory device, and input path is fed back to ALTMEMPHY megafunction for VT tracking.
- (4) This signal is for Registered DIMMs only.

**Table 2-3. AFI Signals (Part 1 of 4)**

Signal Name	Type	Width (1)	Description
<b>Clocks and Resets</b>			
pll_ref_clk	Input	1	The reference clock input to the PHY PLL.
global_reset_n	Input	1	Active-low global reset for PLL and all logic in the PHY. A level set reset signal, which causes a complete reset of the whole system. The PLL may maintain some state information.
soft_reset_n	Input	1	Edge detect reset input intended for SOPC Builder use or to be controlled by other system reset logic. Causes a complete reset of PHY, but not the PLL used in the PHY.
reset_request_n	Output	1	Directly connected to the locked output of the PLL and is intended for optional use either by automated tools such as SOPC Builder or could be manually ANDed with any other system-level signals and combined with any edge detect logic as required and then fed back to the global_reset_n input.  Reset request output that indicates when the PLL outputs are not locked. Use this as a reset request input to any system-level reset controller you may have. This signal is always low while the PLL is locking (but not locked), and so any reset logic using it is advised to detect a reset request on a falling-edge rather than by level detection.

Table 2-3. AFI Signals (Part 2 of 4)

Signal Name	Type	Width <sup>(1)</sup>	Description
ctl_clk	Output	1	Half-rate clock supplied to controller and system logic. The same signal as the non-AFI phy_clk.
ctl_reset_n	Output	1	Reset output on ctl_clk clock domain.
<b>Other Signals</b>			
aux_half_rate_clk	Output	1	In half-rate designs, a copy of the phy_clk_1x signal that you can use in other parts of your design, same as phy_clk port.
aux_full_rate_clk	Output	1	In full-rate designs, a copy of the mem_clk_2x signal that you can use in other parts of your design.
aux_scan_clk	Output	1	Low frequency scan clock supplied primarily to clock any user logic that interfaces to the PLL and DLL reconfiguration interfaces.
aux_scan_clk_reset_n	Output	1	This reset output asynchronously asserts (drives low) when global_reset_n is asserted and de-assert (drives high) synchronous to aux_scan_clk when global_reset_n is de-asserted. It allows you to reset any external circuitry clocked by aux_scan_clk.
<b>Write Data Interface</b>			
ctl_dqs_burst	Input	$\text{MEM\_IF\_DQS\_WIDTH} \times \text{DWIDTH\_RATIO} / 2$	When asserted, mem_dqs is driven. The ctl_dqs_burst signal must be asserted before the ctl_wdata_valid signal and must be driven for the correct duration to generate a correctly timed mem_dqs signal.
ctl_wdata_valid	Input	$\text{MEM\_IF\_DQS\_WIDTH} \times \text{DWIDTH\_RATIO} / 2$	Write data valid. Generates ctl_wdata and ctl_dm output enables.
ctl_wdata	Input	$\text{MEM\_IF\_DWIDTH} \times \text{DWIDTH\_RATIO}$	Write data input from the controller to the PHY to generate mem_dq.
ctl_dm	Input	$\text{MEM\_IF\_DM\_WIDTH} \times \text{DWIDTH\_RATIO}$	DM input from the controller to the PHY.
ctl_wlat	Output	5	Required write latency between address/command and write data that is issued to ALTMEMPHY controller local interface.  This signal is only valid when the ALTMEMPHY sequencer successfully completes calibration, and does not change at any point during normal operation.  The legal range of values for this signal is 0 to 31; and the typical values are between 0 and ten, 0 mostly for low CAS latency DDR memory types.

**Table 2-3. AFI Signals (Part 3 of 4)**

Signal Name	Type	Width <sup>(1)</sup>	Description
<b>Read Data Interface</b>			
ctl_doing_rd	Input	$\text{MEM\_IF\_DQS\_WIDTH} \times \text{DWIDTH\_RATIO} / 2$	Doing read input. Indicates that the DDR3 SDRAM controller is currently performing a read operation. The controller generates <code>ctl_doing_rd</code> to the ALTMEMPHY megafunction. The <code>ctl_doing_rd</code> signal is asserted for one <code>phy_clk</code> cycle for every read command it issues. If there are two read commands, <code>ctl_doing_rd</code> is asserted for two <code>phy_clk</code> cycles. The <code>ctl_doing_rd</code> signal also enables the capture registers and generates the <code>ctl_mem_rdata_valid</code> signal. The <code>ctl_doing_rd</code> signal should be issued at the same time the read command is sent to the ALTMEMPHY megafunction.
ctl_rdata	Output	$\text{DWIDTH\_RATIO} \times \text{MEM\_IF\_DWIDTH}$	Read data from the PHY to the controller.
ctl_rdata_valid	Output	$\text{DWIDTH\_RATIO} / 2$	Read data valid indicating valid read data on <code>ctl_rdata</code> . This signal is two-bits wide (as only half-rate or $\text{DWIDTH\_RATIO} = 4$ is supported) to allow controllers to issue reads and writes that are aligned to either the half-cycle of the half-rate clock.
ctl_rlat	Output	<code>READ_LAT_WIDTH</code>	Contains the number of clock cycles between the assertion of <code>ctl_doing_rd</code> and the return of valid read data ( <code>ctl_rdata</code> ). This signal is unused by the Altera high-performance controller.
<b>Address and Command Interface</b>			
ctl_addr	Input	$\text{MEM\_IF\_ROWADDR\_WIDTH} \times \text{DWIDTH\_RATIO} / 2$	Row address from the controller.
ctl_ba	Input	$\text{MEM\_IF\_BANKADDR\_WIDTH} \times \text{DWIDTH\_RATIO} / 2$	Bank address from the controller.
ctl_cke	Input	$\text{MEM\_IF\_CS\_WIDTH} \times \text{DWIDTH\_RATIO} / 2$	Clock enable from the controller.
ctl_cs_n	Input	$\text{MEM\_IF\_CS\_WIDTH} \times \text{DWIDTH\_RATIO} / 2$	Chip select from the controller.
ctl_odt	Input	$\text{MEM\_IF\_CS\_WIDTH} \times \text{DWIDTH\_RATIO} / 2$	On-die-termination control from the controller.
ctl_ras_n	Input	$\text{DWIDTH\_RATIO} / 2$	Row address strobe signal from the controller.
ctl_we_n	Input	$\text{DWIDTH\_RATIO} / 2$	Write enable.
ctl_cas_n	Input	$\text{DWIDTH\_RATIO} / 2$	Column address strobe signal from the controller.
ctl_rst_n	Input	$\text{DWIDTH\_RATIO} / 2$	Reset from the controller.
<b>Calibration Control and Status Interface</b>			
ctl_mem_clk_disable	Input	<code>MEM_IF_CLK_PAIR_COUNT</code>	When asserted, <code>mem_clk</code> and <code>mem_clk_n</code> are disabled.
ctl_cal_success	Output	1	A 1 indicates that calibration was successful.
ctl_cal_fail	Output	1	A 1 indicates that calibration has failed.

**Table 2-3. AFI Signals (Part 4 of 4)**

Signal Name	Type	Width <sup>(1)</sup>	Description
ctl_cal_req	Input	1	When asserted, a new calibration sequence is started. Currently not supported.
ctl_cal_byte_lane_sel_n	Input	MEM_IF_DQS_WIDTH × MEM_CS_WIDTH	Indicates which DQS groups should be calibrated. Not supported.

**Note to Table 2-2:**

(1) Refer to Table 2-5 for parameter descriptions.

**Table 2-4. Other Interface Signals (Part 1 of 2)**

Signal Name	Type	Width	Description
<b>External DLL Signals</b>			
dqs_delay_ctrl_export	Output	DQS_DELAY_CTL_WIDTH	Allows sharing DLL in this ALTMEMPHY instance with another ALTMEMPHY instance. Connect the dqs_delay_ctrl_export port on the ALTMEMPHY instance with a DLL to the dqs_delay_ctrl_import port on the other ALTMEMPHY instance.
dqs_delay_ctrl_import	Input	DQS_DELAY_CTL_WIDTH	Allows the use of DLL in another ALTMEMPHY instance in this ALTMEMPHY instance. Connect the dqs_delay_ctrl_export port on the ALTMEMPHY instance with a DLL to the dqs_delay_ctrl_import port on the other ALTMEMPHY instance.
dqs_offset_delay_ctrl_width	Input	DQS_DELAY_CTL_WIDTH	Connects to the DQS delay logic when dll_import_export is set to IMPORT. Only connect if you are using a DLL offset, which can otherwise be tied to zero. If you are using a DLL offset, connect this input to the offset_ctrl_out output of the dll_offset_ctrl block.
dll_reference_clk	Output	1	Reference clock to feed to an externally instantiated DLL. This clock is typically from one of the PHY PLL outputs.
<b>User-Mode Calibration OCT Control Signals</b>			
oct_ctl_rs_value	Input	14	OCT RS value port for use with ALT_OCT megafunction if you want to use OCT with user-mode calibration.
oct_ctl_rt_value	Input	14	OCT RT value port for use with ALT_OCT megafunction if you want to use OCT with user-mode calibration.
<b>Debug Interface Signals <sup>(1), (2)</sup></b>			
dbg_clk	Input	1	Debug interface clock.
dbg_reset_n	Input	1	Debug interface reset.
dbg_addr	Input	DBG_A_WIDTH	Address input.
dbg_wr	Input	1	Write request.
dbg_rd	Input	1	Read request.
dbg_cs	Input	1	Chip select.
dbg_wr_data	Input	32	Debug interface write data.
dbg_rd_data	Output	32	Debug interface read data.
dbg_waitrequest	Output	1	Wait signal.

**Table 2-4. Other Interface Signals (Part 2 of 2)**

Signal Name	Type	Width	Description
<b>Calibration Interface Signals—without leveling only</b>			
rsu_codvw_phase	Output	—	The sequencer sweeps the phase of a resynchronization clock across 360° or 720° of a memory clock cycle. Data reads from the DIMM are performed for each phase position, and a data valid window is located, which is the set of resynchronization clock phase positions where data is successfully read. The final resynchronization clock phase is set at the center of this range: the center of the data valid window or CODVW. This output is set to the current calculated value for the CODVW, and represents how many phase steps were performed by the PLL to offset the resynchronization clock from the memory clock.
rsu_codvw_size	Output	—	The final centre of data valid window size ( <i>rsu_codvw_size</i> ) is the number of phases where data was successfully read in the calculation of the resynchronization clock centre of data valid window phase ( <i>rsu_codvw_phase</i> ).
rsu_read_latency	Output	—	The <i>rsu_read_latency</i> output is then set to the read latency (in <i>phy_clk</i> cycles) using the <i>rsu_codvw_phase</i> resynchronization clock phase. If calibration is unsuccessful then this signal is undefined.
rsu_no_dvw_err	Output	—	If the sequencer sweeps the resynchronization clock across every phase and does not see any valid data at any phase position, then calibration fails and this output is set to 1.
rsu_grt_one_dvw_err	Output	—	If the sequencer sweeps the resynchronization clock across every phase and sees multiple data valid windows, this is indicative of unexpected read data (random bit errors) or an incorrectly configured PLL that must be resolved. Calibration has failed and this output is set to 1.

**Notes to Table 2-4:**

- (1) The debug interface uses the simple Avalon-MM interface protocol.
- (2) These ports exist in the Quartus II software, even though the debug interface is for Altera's use only.

Table 2-5 shows the parameters to which Table 2-2 through Table 2-4 refer.

**Table 2-5. Parameters (Part 1 of 2)**

Parameter Name	Description
DWIDTH_RATIO	The data width ratio from the local interface to the memory interface. DWIDTH_RATIO of 2 means full rate, while DWIDTH_RATIO of 4 means half rate.
LOCAL_IF_DWIDTH	The width of the local data bus must be quadrupled for half-rate and doubled for full-rate.
MEM_IF_DWIDTH	The data width at the memory interface. MEM_IF_DWIDTH can have values that are multiples of MEM_IF_DQ_PER_DQS.
MEM_IF_DQS_WIDTH	The number of DQS pins in the interface.
MEM_IF_ROWADDR_WIDTH	The row address width of the memory device.
MEM_IF_BANKADDR_WIDTH	The bank address with the memory device.
MEM_IF_CS_WIDTH	The number of chip select pins in the interface. The sequencer only calibrates one chip select pin.
MEM_IF_DM_WIDTH	The number of <i>mem_dm</i> pins on the memory interface.

Table 2-5. Parameters (Part 2 of 2)

Parameter Name	Description
MEM_IF_DQ_PER_DQS	The number of mem_dq[] pins per mem_dqs pin.
MEM_IF_CLK_PAIR_COUNT	The number of mem_clk/mem_clk_n pairs in the interface.

## PHY-to-Controller Interfaces

The following section describes the typical modules that are connected to the ALTMEMPHY variation and the port name prefixes each module uses. This section also describes using a custom controller. This section describes the AFI.

The AFI standardizes and simplifies the interface between controller and PHY for all Altera memory designs, thus allowing you to easily interchange your own controller code with Altera's high-performance controller. The AFI PHY includes an administration block that configures the memory for calibration and performs necessary mode registers accesses to configure the memory as required (these calibration processes are different). Figure 2-5 shows an overview of the connections between the PHY, the controller, and the memory device.


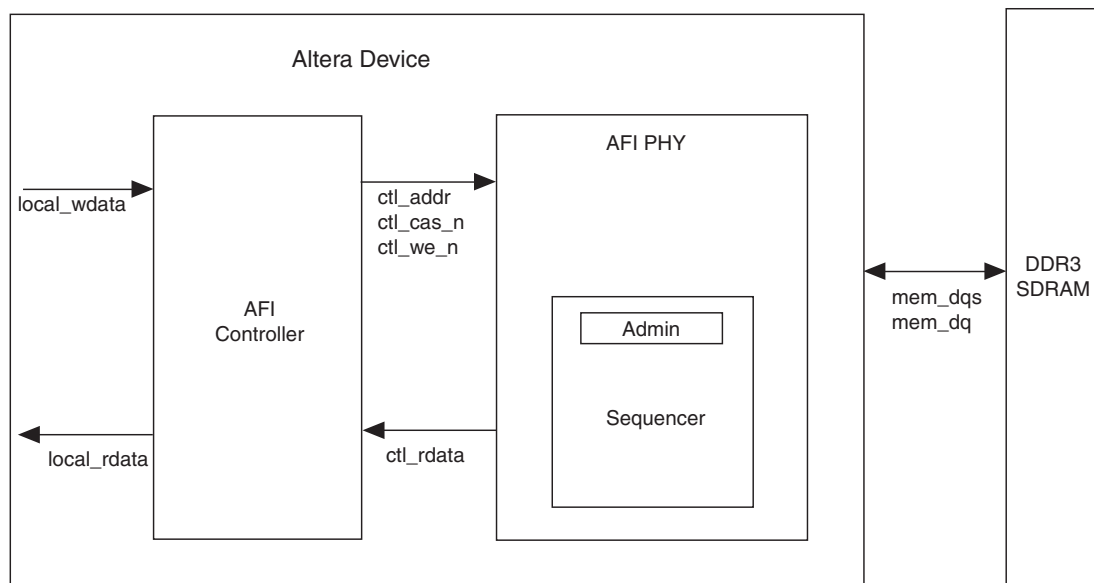
 Altera recommends that you use the AFI for new designs.

Figure 2-5. AFI PHY Connections

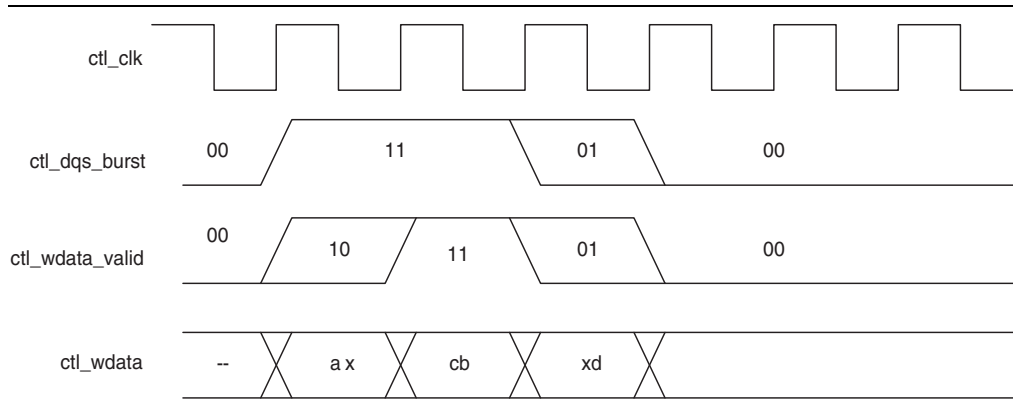


For half-rate designs, the address and command signals in the ALTMEMPHY megafunction are asserted for one mem\_clk cycle (1T addressing), such that there are two input bits per address and command pin in half-rate designs. If you require a more conservative 2T addressing, drive both input bits (of the address and command signal) identically in half-rate designs.

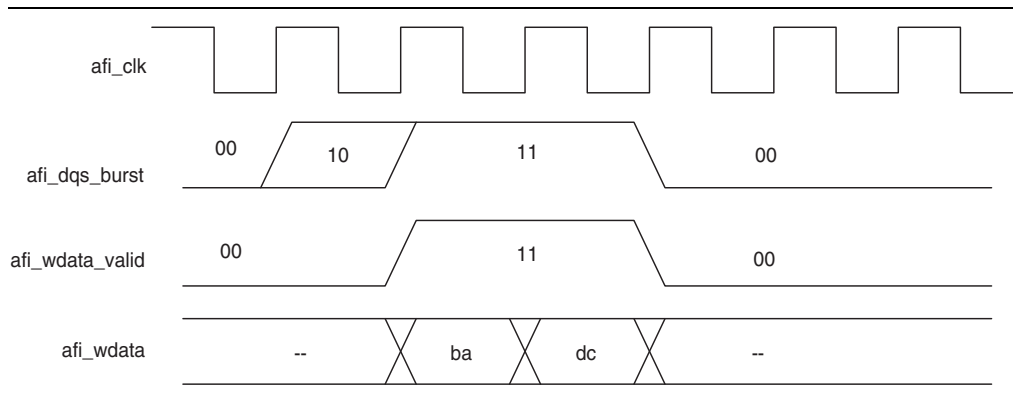
For DDR3 SDRAM with the AFI, the read and write control signals are on a per-DQS group basis. The controller can calibrate and use a subset of the available DDR3 SDRAM devices. For example, the controller can calibrate and use two devices out of a 64- or 72-bit DIMM for better debugging mechanism.

For half-rate designs, the AFI allows the controller to issue reads and writes that are aligned to either half-cycle of the half-rate `phy_clk`, which means that the datapaths can support multiple data alignments—word-unaligned and word-aligned writes and reads. [Figure 2-6](#) and [Figure 2-7](#) display the half-rate write operation.

**Figure 2-6. Half-Rate Write with Word-Unaligned Data**




**Figure 2-7. Half-Rate Write with Word-Aligned Data**




After calibration process is complete, the sequencer sends the write latency in number of clock cycles to the controller.

[Figure 2-8](#) and [Figure 2-9](#) show word-aligned writes and reads. In the following read and write examples the data is written to and read from the same address. In each example, `ctl_rdata` and `ctl_wdata` are aligned with controller clock (`ctl_clk`) cycles. All the data in the bit vector is valid at once. For comparison, refer [Figure 2-10](#) and [Figure 2-11](#) that show the word-unaligned writes and reads.


 The `ctl_doing_rd` is represented as a half-rate signal when passed into the PHY. Therefore, the lower half of this bit vector represents one memory clock cycle and the upper half the next memory clock cycle. [Figure 2-11 on page 2-22](#) shows separated word-unaligned reads as an example of two `ctl_doing_rd` bits are different. Therefore, for each x16 device, at least two `ctl_doing_rd` bits need to be driven, and two `ctl_rdata_valid` bits need to be interpreted.


The AFI has the following conventions:

- With the AFI, high and low signals are combined in one signal, so for a single chip select (`ctl_cs_n`) interface, `ctl_cs_n[1:0]`, where location 0 appears on the memory bus on one `mem_clk` cycle and location 1 on the next `mem_clk` cycle.

 This convention is maintained for all signals so for an 8 bit memory interface, the write data (`ctl_wdata`) signal is `ctl_wdata[31:0]`, where the first data on the DQ pins is `ctl_wdata[7:0]`, then `ctl_wdata[15:8]`, then `ctl_wdata[23:16]`, then `ctl_wdata[31:24]`.

- Word-aligned and word-unaligned reads and writes have the following definitions:
  - Word-aligned for the single chip select is active (low) in location 1 (`_1`). `ctl_cs_n[1:0] = 01` when a write occurs. This alignment is the easiest alignment to design with.
  - Word-unaligned is the opposite, so `ctl_cs_n[1:0] = 10` when a read or write occurs and the other control and data signals are distributed across consecutive `ctl_clk` cycles.

 The Altera high-performance controller uses word-aligned data only.

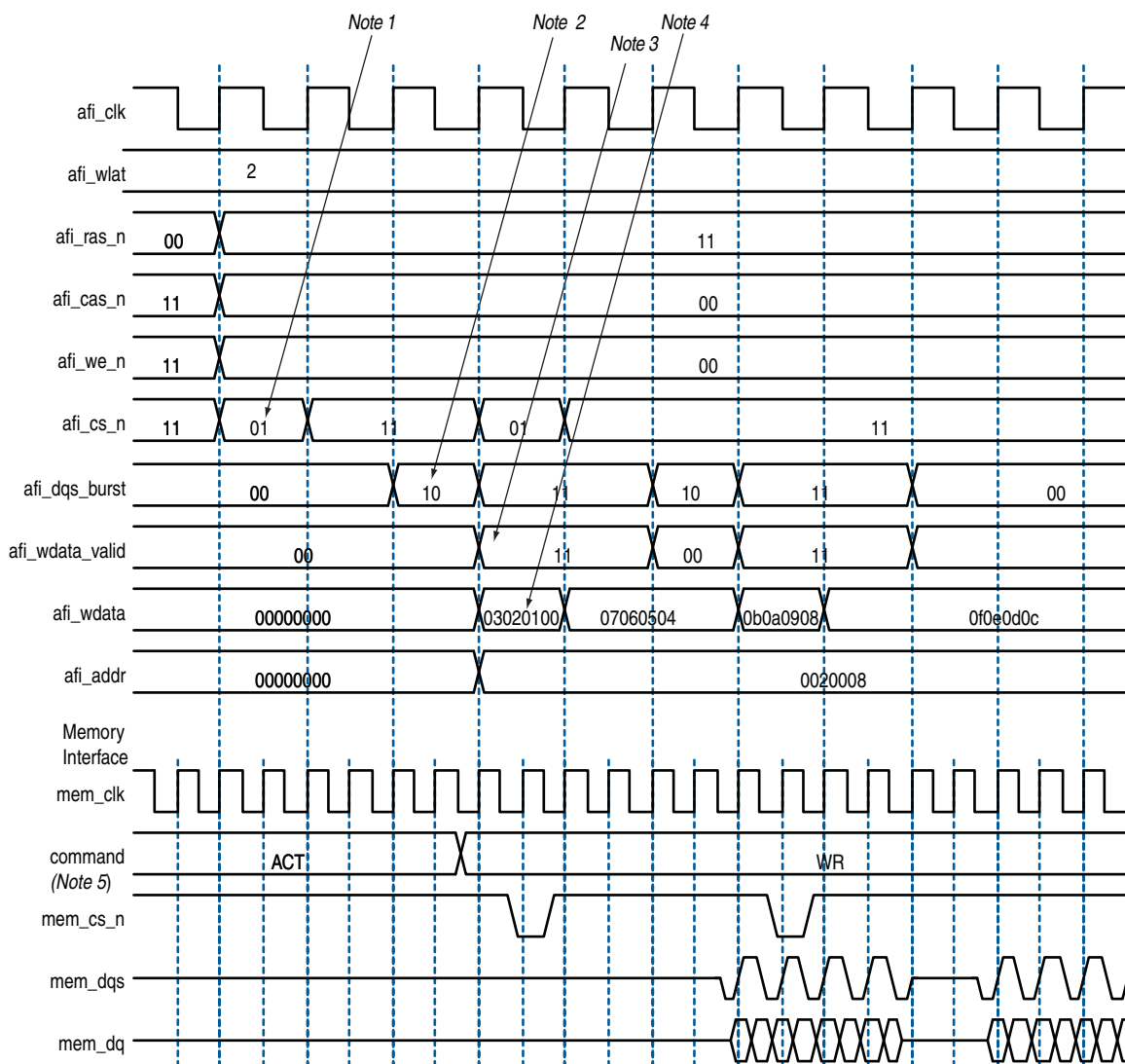
 The timing analysis script does not support word-unaligned reads and writes.

- Spaced reads and writes have the following definitions:
  - Spaced writes—write commands separated by a gap of one controller clock (`ctl_clk`) cycle
  - Spaced reads—read commands separated by a gap of one controller clock (`ctl_clk`) cycle

[Figure 2-8](#) through [Figure 2-11](#) assume the following general points:

- The burst length is four. A DDR2 SDRAM is used—the interface timing is identical for DDR3 devices.
- An 8-bit interface with one chip select.
- The data for one controller clock (`ctl_clk`) cycle represents data for two memory clock (`mem_clk`) cycles (half-rate interface).

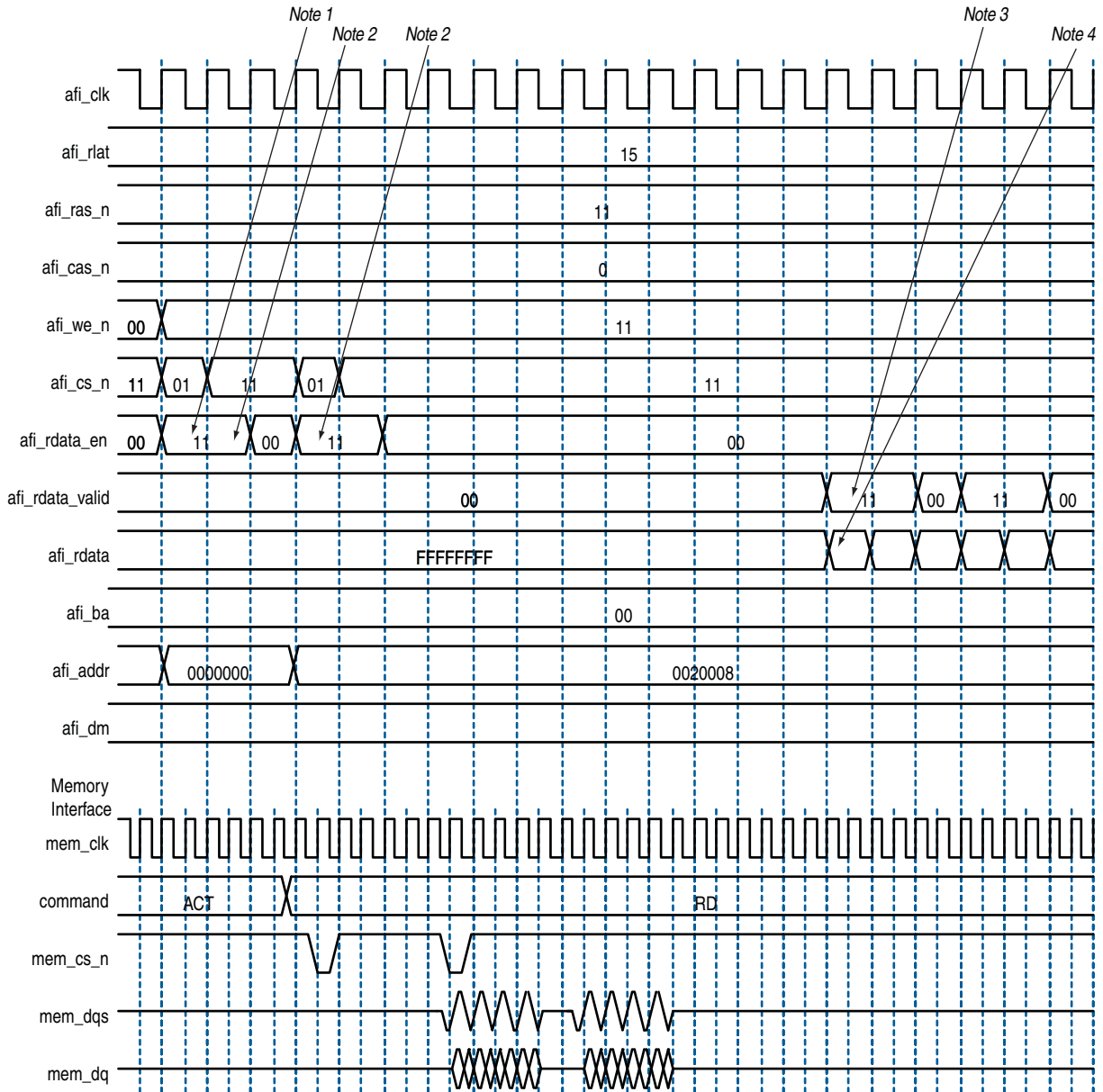
Figure 2-8. Word-Aligned Writes



Notes to Figure 2-8:

- (1) To show the even alignment of `ctl_cs_n`, expand the signal (this convention applies for all other signals).
- (2) The `ctl_dqs_burst` must go high one memory clock cycle before `ctl_wdata_valid`. Compare with the word-unaligned case.
- (3) The `ctl_wdata_valid` is asserted two `ctl_wlat` controller clock (`ctl_clk`) cycles after chip select (`ctl_cs_n`) is asserted. The `ctl_wlat` indicates the required write latency in the system. The value is determined during calibration and is dependant upon the relative delays in the address and command path and the write datapath in both the PHY and the external DDR SDRAM subsystem. The controller must drive `ctl_cs_n` and then wait `ctl_wlat` (two in this example) `ctl_clks` before driving `ctl_wdata_valid`.
- (4) Observe the ordering of write data (`ctl_wdata`). Compare this to data on the `mem_dq` signal.
- (5) In all waveforms a command record is added that combines the memory pins `ras_n`, `cas_n` and `we_n` into the current command that is issued. This command is registered by the memory when chip select (`mem_cs_n`) is low. The important commands in the presented waveforms are WR = write, ACT = activate.

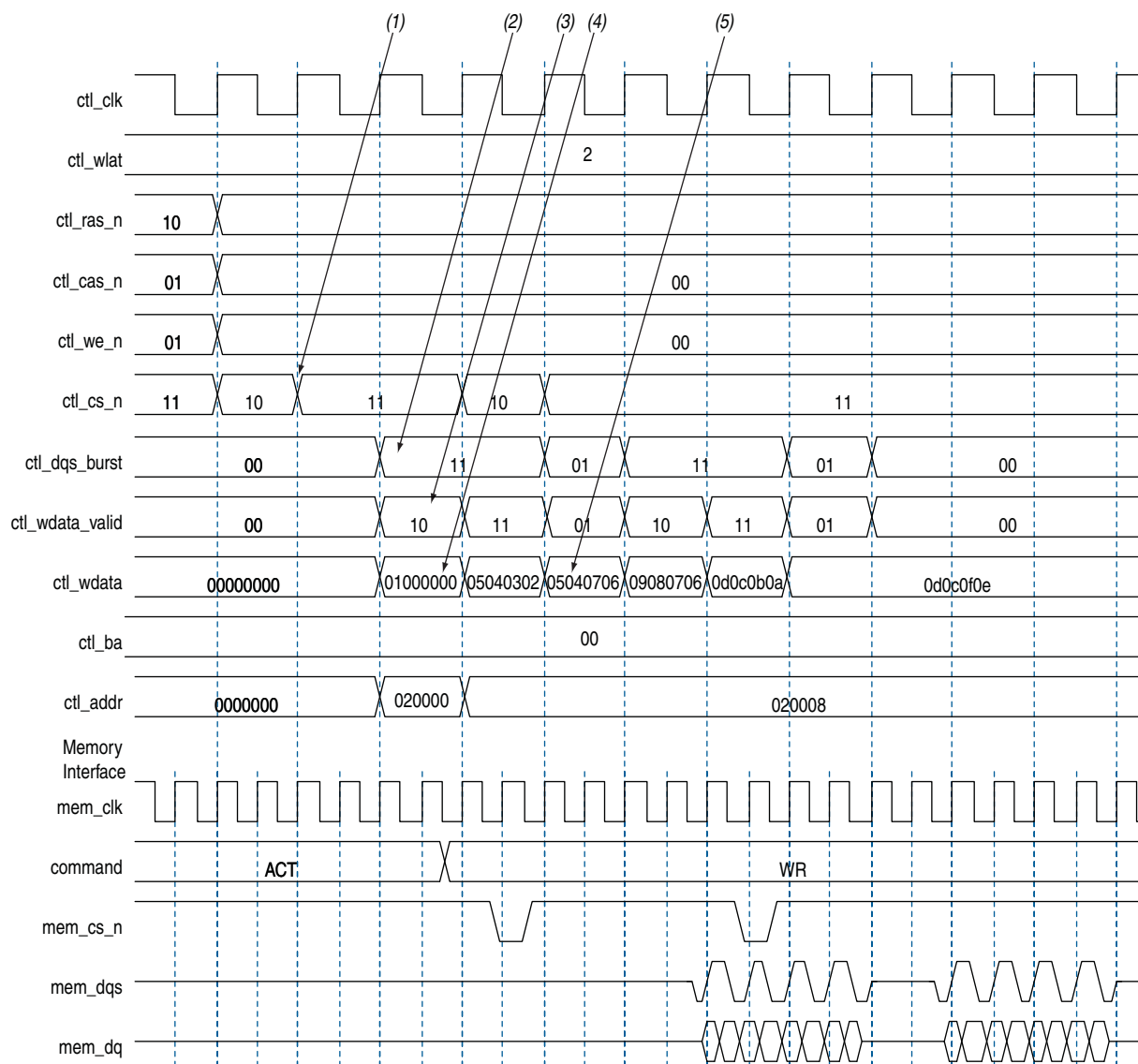
Figure 2-9. Word-Aligned Reads

**Notes to Figure 2-9:**

- (1) For AFI, **ctl\_doing\_rd** is required to be asserted one memory clock cycle before chip select (**ctl\_cs\_n**) is asserted. In the half-rate **ctl\_clk** domain, this requirement manifests as the controller driving 11 (as opposed to the 01) on **ctl\_doing\_rd**.
- (2) AFI requires that **ctl\_doing\_rd** is driven for the duration of the read. In this example, it is driven to 11 for two half-rate **ctl\_clks**, which equates to driving to 1, for the four memory clock cycles of this four-beat burst.
- (3) The **ctl\_rdata\_valid** returns 15 (**ctl\_rlat**) controller clock (**ctl\_clk**) cycles after **ctl\_doing\_rd** is asserted. Returned is when the **ctl\_rdata\_valid** signal is observed at the output of a register within the controller. A controller can use the **ctl\_rlat** value to determine when to register to returned data, but this is unnecessary as the **ctl\_rdata\_valid** is provided for the controller to use as an enable when registering read data.
- (4) Observe the alignment of returned read data with respect to data on the bus.

Figure 2-10 and Figure 2-11 show spaced word-unaligned writes and reads.

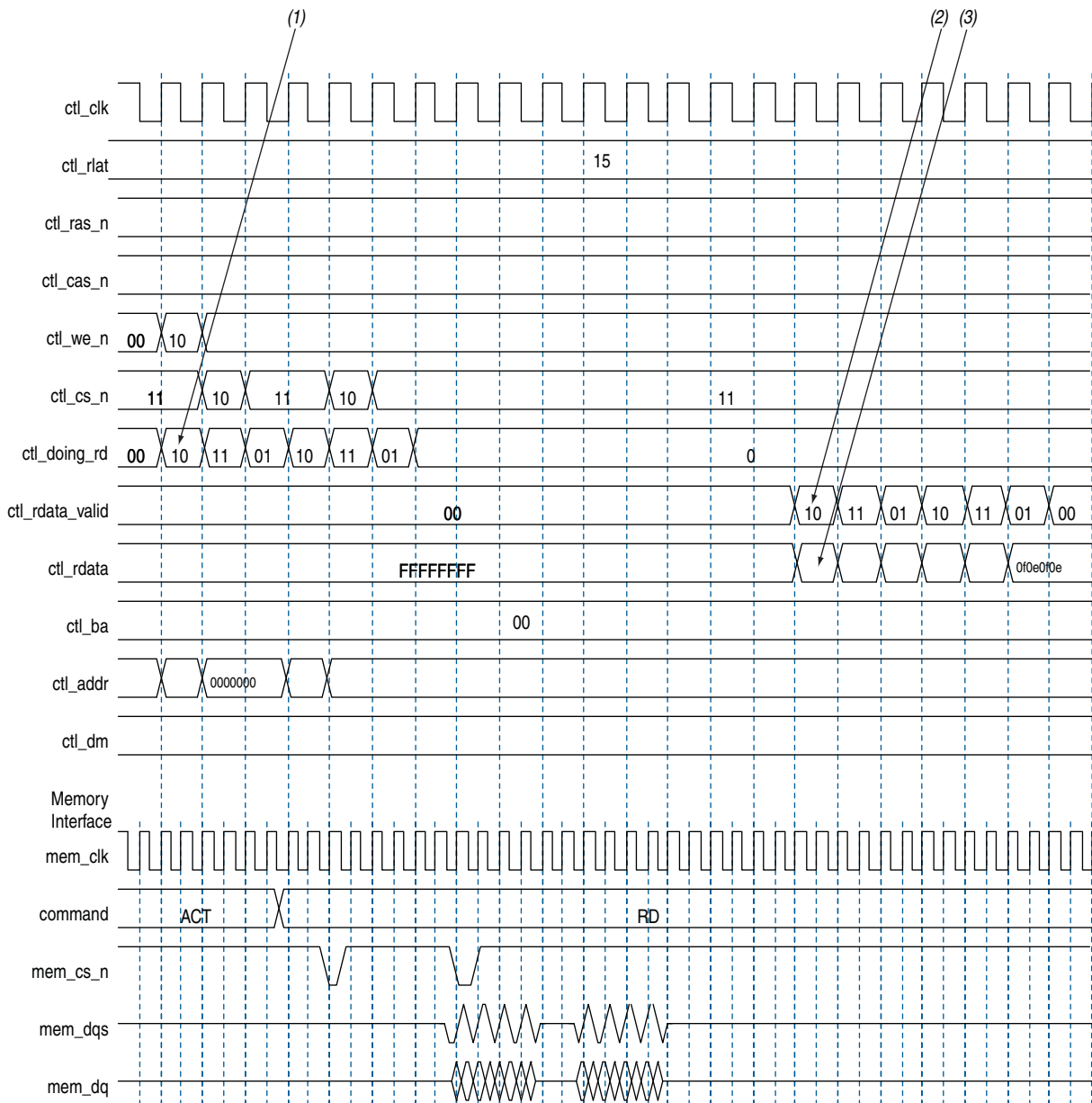
**Figure 2-10. Word-Unaligned Writes**



**Notes to Figure 2-10:**

- (1) Alternative word-unaligned chip select (`ctl_cs_n`).
- (2) As with word-aligned writes, `ctl_dqs_burst` is asserted one memory clock cycle before `ctl_wdata_valid`. You can see `ctl_dqs_burst` is 11 in the same cycle where `ctl_wdata_valid` is 10. The LSB of these two becomes the first value the signal takes in the `mem_clk` domain. You can see that `ctl_dqs_burst` has the necessary one `mem_clk` cycle lead on `ctl_wdata_valid`.
- (3) The latency between `ctl_cs_n` being asserted and `ctl_wdata_valid` going high is effectively `ctl_wlat` (in this example, two) controller clock (`ctl_clk`) cycles. This can be thought of in terms of relative memory clock (`mem_clk`) cycles, in which case the latency is four `mem_clk` cycles.
- (4) Only the upper half is valid (as the `ctl_wdata_valid` signal demonstrates, there is one `ctl_wdata_valid` bit to two 8-bit words). The write data bits go out on the bus in order, least significant byte first. So for a continuous burst of write data on the DQ pins, the most significant half of write data is used, which goes out on the bus last and is therefore contiguous with the following data. The converse is true for the end of the burst. Write data is spread across three controller clock (`ctl_clk`) cycles, but still only four memory clock (`mem_clk`) cycles. However, in relative memory clock cycles the latency is equivalent in the word-aligned and word-unaligned cases.
- (5) The 0504 here is residual from the previous clock cycle. In the same way that only the upper half of the write data is used for the first beat of the write, only the lower half of the write data is used in the last beat of the write. These upper bits can be driven to any value in this alignment.

Figure 2-11. Word-Unaligned Reads



## Notes to Figure 2-11:

- (1) Similar to word-aligned reads, **ctl\_doing\_rd** is asserted one memory clock cycle before chip select (**ctl\_cs\_n**) is asserted, which for a word-unaligned read is in the previous controller clock (**ctl\_clk**) cycle. In this example the **ctl\_doing\_rd** signal is now spread over three controller clock (**ctl\_clk**) cycles, the high bits in the sequence '10', '11', '01', '10', '11', '01' providing the required four memory clock cycles of assertion for **ctl\_doing\_rd** for the two 4-beat reads in the full-rate memory clock domain, '011110', '011110'.
- (2) The return pattern of **ctl\_rdata\_valid** is a delayed version of **ctl\_doing\_rd**. Advertised read latency (**ctl\_rlat**) is the number of controller clock (**ctl\_clk**) cycles delay inserted between **ctl\_doing\_rd** and **ctl\_rdata\_valid**.
- (3) The read data (**ctl\_rdata**) is spread over three controller clock cycles and in the pointed to vector only the upper half of the **ctl\_rdata** bit vector is valid (denoted by **ctl\_rdata\_valid**).

## Using a Custom Controller

The ALTMEMPHY megafunction can be integrated with your own controller. This section describes the interface requirement and the handshake mechanism for efficient read and write transactions.

### Preliminary Steps

Perform the following steps to generate the ALTMEMPHY megafunction:

1. If you are creating a custom DDR3 SDRAM controller, generate the Altera high-performance controller targeting your chosen Altera and memory devices.
2. Compile and verify the timing. This step is optional.
3. If targeting a DDR3 SDRAM device, simulate the high-performance controller design so you can determine how to drive the PHY signals using your own controller.
4. Integrate the top-level ALTMEMPHY design with your controller. If you started with the high-performance controller, the PHY variation name is `<controller_name>_phy.v/.vhd`. Details about integrating your controller with Altera's ALTMEMPHY megafunction are described in the following sections.
5. Compile and simulate the whole interface to ensure that you are driving the PHY properly and that your commands are recognized by the memory device.

### Design Considerations

This section discusses the important considerations for implementing your own controller with the ALTMEMPHY megafunction. This section describes the design considerations for AFI variants.



Simulating the high-performance controller is useful if you do not know how to drive the PHY signals.

### Clocks and Resets

The ALTMEMPHY megafunction automatically generates a PLL instance, but you must still provide the reference clock input (`pll_ref_clk`) with a clock of the frequency that you specified in the MegaWizard Plug-In Manager. An active-low global reset input is also provided, which you can deassert asynchronously. The clock and reset management logic synchronizes this reset to the appropriate clock domains inside the ALTMEMPHY megafunction.

A clock output, half the memory clock frequency for a half-rate controller, is provided and all inputs and outputs of the ALTMEMPHY megafunction are synchronous to this clock. For AFIs, this signal is called `ctl_clk`.

There is also an active-low synchronous reset output signal provided, `ctl_reset_n`. This signal is synchronously de-asserted with respect to the `ctl_clk` or `phy_clk` clock domain and it can reset any additional user logic on that clock domain.

## Calibration Process Requirements

When the global `reset_n` is released the ALTMEMPHY handles the initialization and calibration sequence automatically. The sequencer calibrates memory interfaces by issuing reads to multiple ranks of DDR3 SDRAM (multiple chip select). Timing margins decrease as the number of ranks increases. It is impractical to supply one dedicated resynchronization clock for each rank of memory, as it consumes PLL resources for the relatively small benefit of improved timing margin. When calibration is complete `ctl_cal_success` goes high if successful; `ctl_cal_fail` goes high if calibration fails. Calibration can be repeated by the controller using the `soft_reset_n` signal, which when asserted puts the sequencer into a reset state and when released the calibration process begins again.

## Other Local Interface Requirements

The memory burst length for DDR3 SDRAM devices can be set at either four or eight; but when using the Altera high-performance controller, only burst length eight is supported. For a half-rate controller, the memory clock runs twice as fast as the clock provided to the local interface, so data buses on the local interface are four times as wide as the memory data bus.

## Address and Command Interfacing

Address and command signals are automatically sized for 1T operation, such that for full-rate designs there is one input bit per pin (for example, one `cs_n` input per chip select configured); for half-rate designs there are two. If you require a more conservative 2T address and command scheme, use a full-rate design and drive the address/command inputs for two clock cycles, or in a half-rate design drive both address/command bits for a given pin identically.



Although the PHY inherently supports 1T addressing, the high-performance controller supports only 2T addressing, so PHY timing analysis is performed assuming 2T address and command signals.

## Handshake Mechanism Between Read Commands and Read Data

When performing a read, a high-performance controller with the AFI asserts `ctl_doing_read` to indicate that a read command is requested and the byte lanes that it expects valid data to return on. ALTMEMPHY uses `ctl_doing_read` for the following actions:

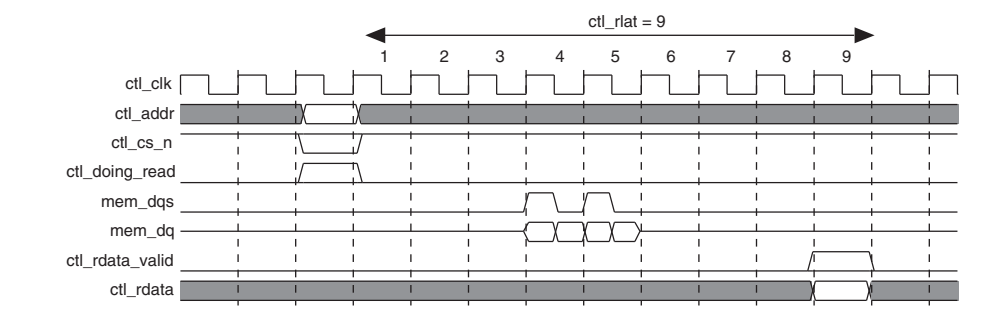
- Control of the postamble circuit
- Generation of `ctl_rdata_valid`
- Dynamic termination ( $R_t$ ) control timing

The read latency, `ctl_rlat`, is advertised back to the controller. This signal indicates how long it takes in `ctl_clk` clock cycles from assertion of `ctl_doing_read` to valid read data returning on `ctl_rdata`. The `ctl_rlat` signal is only valid when calibration has successfully completed and never changes values during normal user mode operation.

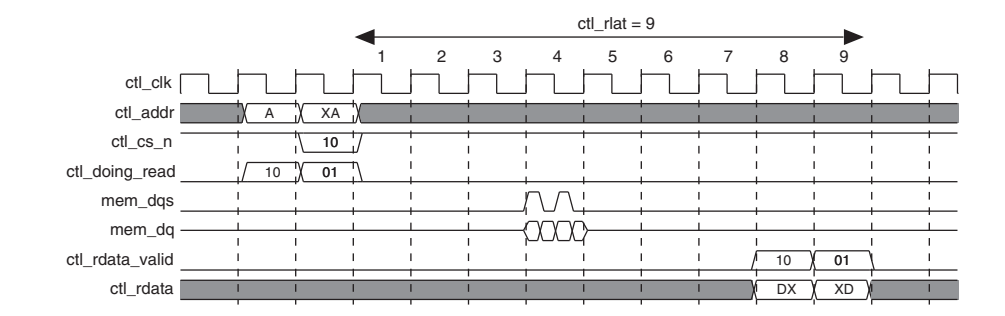
The ALTMEMPHY provides a signal, `ctl_rdata_valid`, to indicate that the data on read data bus is valid. The width of this signal varies between half-rate and full-rate designs to support the option to indicate that the read data is not word aligned.

Figure 2-12 and Figure 2-13 show these relationships.

**Figure 2-12. Address and Command and Read-Path Timing—Full-Rate Design**



**Figure 2-13. Second Read Alignment—Half-Rate Design**



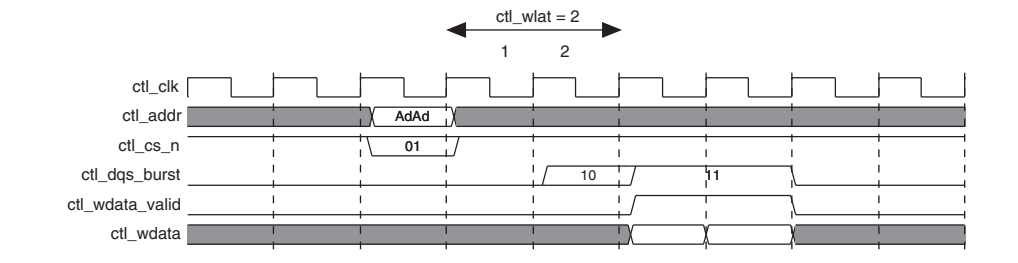
### Handshake Mechanism Between Write Commands and Write Data

In the AFI, the ALTMEMPHY output `ctl_wlat` gives the number of `ctl_clk` cycles between the write command that is issued `ctl_cs_n` asserted and `ctl_dqs_burst` asserted. The `ctl_wlat` signal considers the following actions to provide a single value in `ctl_clk` clock cycles:

- CAS write latency
- Additive latency
- Datapath latencies and relative phases
- Board layout
- Address and command path latency and 1T register setting, which is dynamically setup to take into account any leveling effects

The `ctl_wlat` signal is only valid when the calibration has been successfully completed by the ALTMEMPHY sequencer and does not change at any point during normal user mode operation. Figure 2-14 shows the operation of `ctl_wlat` port.

**Figure 2-14. Timing for `ctl_dqs_burst`, `ctl_wdata_valid`, Address, and Command—Half-Rate Design**



For a half-rate design `ctl_cs_n` is 2 bits, not 1. Also the `ctl_dqs_burst` and `ctl_wdata_valid` waveforms indicate a half-rate design. This write results in a burst of 8 at the DDR. Where `ctl_cs_n` is driven 2'b01, the LSB (1) is the first value driven out of `mem_cs_n`, and the MSB (0) follows on the next `mem_clk`. Similarly, for `ctl_dqs_burst`, the LSB is driven out of `mem_dqs` first (0), then a 1 follows on the next clock cycle. This sequence produces the continuous DQS pulse as required. Finally, the `ctl_addr` bus is twice `MEM_IF_ADDR_WIDTH` bits wide and so the address is concatenated to result in an address phase two `mem_clk` cycles wide.

## Partial Writes

As part of the DDR3 SDRAM memory specifications, you have the option for partial write operations by asserting the DM pins for part of the write signal.

For designs targeting the Arria II devices, deassert the `ctl_wdata_valid` signal during partial writes, when the write data is invalid, to save power by not driving the DQ outputs.

For designs targeting other devices, use only the DM pins if you require partial writes. Assert the `ctl_dqs_burst` and `ctl_wdata_valid` signals as for full write operations, so that the DQ and DQS pins are driven during partial writes.

## ALTMEMPHY Calibration Stages

In all configurations, the noncalibrated address, command and control interfaces must be correctly constrained and meet timing.

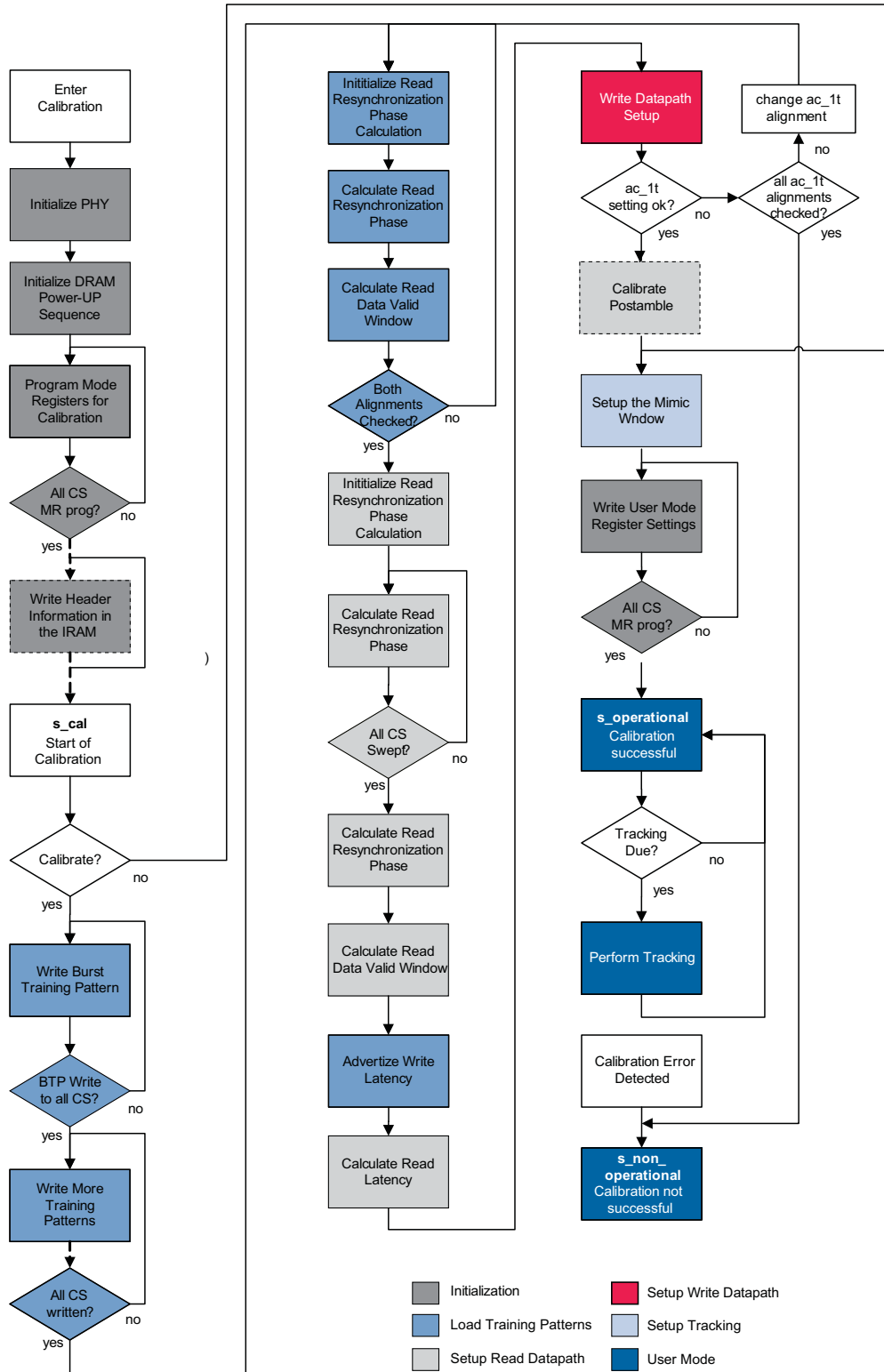
If calibration fails at a specific stage, use this chapter to understand what functionally happens at that stage, to assist with the debug.

The ALTMEMPHY IP performs the following calibration stages:

1. Enter Calibration (s\_reset)
2. Initialize PHY (s\_phy\_initialize)
3. Initialize DRAM
4. Write Header Information in the internal RAM (s\_write\_ihi)
5. Load Training Patterns
6. Test More Pattern Writes
7. Calibrate Read Resynchronization Phase
8. Advertize Write Latency (s\_was)
9. Calculate Read Latency (s\_adv\_rlat)
10. Output Write Latency (s\_adv\_wlat)
11. Calibrate Postamble (s\_poa)
12. Set Up Address and Command Clock Cycle
13. Write User Mode Register Settings (s\_prep\_customer\_mr\_setup)
14. Voltage and Temperature Tracking

This chapter discusses these stages. [Figure 2-15 on page 2-28](#) shows a flow chart of the calibration stages for the ALTMEMPHY IP.

Figure 2-15. Calibration Stages



## Enter Calibration (s\_reset)

Calibration starts when the ALTMEMPHY IP deasserts the PHY reset signal and the AFI signal `ctl_cal_req` is low.

## Initialize PHY (s\_phy\_initialize)

This stage holds off calibration until the DLL has locked, and (if debug toolkit is enabled) internal RAM contents are all reset to zero.

## Initialize DRAM

Initializing the DRAM has the following two stages:

- Initialize DRAM Power Up Sequence (s\_int\_dram)
- Program Mode Registers for Calibration (s\_prog\_mr)

### Initialize DRAM Power Up Sequence (s\_int\_dram)

This stage brings the SDRAM out of a reset state (from any previous state) through the initialization sequence specified in the JEDEC specification for each device type, up to but not including mode register set commands. At the end of this stage, the SDRAM is ready to receive mode register load commands, which must occur (on each rank) before refreshes can occur.




This calibration stage applies to all chip selects in parallel.

### Program Mode Registers for Calibration (s\_prog\_mr)

The ALTMEMPHY IP issues mode register set commands on a per chip select basis, which allows you great flexibility to issue different mode register settings to different chip selects. When all chip selects have mode registers programmed (the initialization of that chip select is complete), refreshes are enabled.


The following overrides apply to user settings:


- For DDR and DDR2 SDRAM:
  - DLL enable
  - Burst length 4
  - OCD calibration (DDR2 only)
- For DDR3 SDRAM:
  - DLL enable
  - Output buffer enable
  - Disable write leveling
  - Runtime burst length select
  - Test mode disabled
  - DLL reset

 For DDR3 SDRAM this stage also includes a ZQ-cal long operation (refer to the JEDEC specification).

## Write Header Information in the internal RAM (s\_write\_ghi)

In this stage, the ALTMEMPHY IP loads the internal header information in the first eight locations in the internal RAM through the parameterization of the ALTMEMPHY IP. The debug toolkit uses this information to provide the current ALTMEMPHY IP parameterization and IP version numbers.

 The ALTMEMPHY IP only executes this stage when you enable debug toolkit.


 For information about the debug toolkit, refer to the [ALTMEMPHY External Memory Interface Debug Toolkit](#).

## Load Training Patterns

In this stage, the ALTMEMPHY IP writes training patterns to the memory to be read in later calibration stages. Because of the matched trace lengths to DDR SDRAM components, after memory initialization, you can assume write capture works.

You can divide the training pattern writes into the following two stages:

- [Write Block Training Pattern \(s\\_write\\_btp\)](#)
- [Write More Training Patterns \(s\\_write\\_mtp\)](#)

 The ALTMEMPHY IP writes further training pattern in the calibration of the write datapath, refer to [Advertise Write Latency \(s\\_was\)](#).

### Write Block Training Pattern (s\_write\_btp)


This stage applies to read data valid alignment (s\_rdv), advertise read latency (s\_adv\_rd\_lat) and postamble calibration (s\_poa). For these calibration stages a pattern of all 1s and all 0s is sufficient to set up the PHY.

Writing of these two patterns is trivial and requires all DDIO outputs (high and low phases (bits)) to be held at either 1 or 0, for all 1 and all 0 patterns, respectively. To write these patterns, the ALTMEMPHY IP holds the DDIO outputs low (or high) and toggles DQS for a predetermined length of time and issues a single write command. The ALTMEMPHY IP tests a full range of memory write latencies.

To support DDR3 SDRAM discrete components (burst of eight reads), the ALTMEMPHY IP loads eight memory locations with 1s and eight with 0s.

The following memory locations contain the following patterns:

- Locations [7:0], all 0s
- Locations [15:8], all 1s

 You need patterns of all 1s and all 0s for calibrating the read resynchronization phase.

## Write More Training Patterns (s\_write\_mtp)

This stage calculates the read resynchronization phase (s\_rrp\_sweep).

The pattern is 0x30F5 and comprises separately written patterns. The ALTMEMPHY IP requires this pattern to match the characterization behavior for nonDQS capture based schemes (for example, Cyclone III devices). All device families use the following pattern:

- All 0: 'b0000 to DDIO high and low bits held at 0
- All 1: 'b1111 to DDIO high and low bits held at 1
- Toggle: 'b0101 to DDIO high bits held at 0 and DDIO low bits held at 1
- Mixed: 'b0011 to DDIO high and low bits have to toggle

While you can ensure that all zeros, all ones, or toggle are written into a burst of memory locations (output DDIO bits are held at constant values), it is challenging to ensure that a pattern of 0011 is written into memory. The challenge occurs because the write latency is unknown at this time. For example, if this pattern is repeated on DQ pins (0011 0011 0011) and a single write command issued (as for the other patterns), it is not known whether the memory location contains the pattern 0011 or 1100.

The ALTMEMPHY IP provides a methodology to robustly write these patterns ([Test More Pattern Writes](#)). For this section two locations (X and Y) are populated with write data, one contains the pattern 0011; the other contains 1100.

The memory locations contain the following patterns:

- x30 alignment 0 to location (X): 23 . . 16
- x30 alignment 1 to location (Y): 31 . . 24
- xF5 to location: 39 . . 32

The ALTMEMPHY IP writes these patterns in bursts of four beats, so in the pattern xF5, F is written separately to 5. The ALTMEMPHY IP writes patterns F and 0 as a part of the writing of the block training pattern ([Write Block Training Pattern \(s\\_write\\_btp\)](#)).

## Test More Pattern Writes

This stage comprises a number of calibration stages of the PHY, but the stage is described as one entity. This stage comprises:

- Initialize read resynchronization phase calculation (s\_rrp\_reset)
- Calculate read resynchronization phase (s\_rrp\_sweep)
- Calculate read data valid window (s\_read\_mtp)

The algorithm ensures the pattern 0011 is written to a known location in memory. The following assumptions and PHY settings apply to this stage:

- Assumptions:
  - Burst length of 4 writes
  - Write capture in the memory device works. Data can be safely written to memory. No write leveling is required. (Refer to JEDEC specification for more information on DDR3 SDRAM).
  - Writes are aligned on a clock cycle basis. You know which beats of DQ data to write on the low and high phases of DQS (ultimately DQ and DQS board delays are well matched).
- Settings:
  - Address and command 1T setting. You can add additional latency to the address and command path (specified as a maximum of one memory clock cycles (t)). This setting aligns write data to address and command signals relative to the controller clock domain, where address and command signals are issued in a given alignment. This setting is not required (0t) for a full-rate PHY.
  - Read data 1T alignment. Additional delay of captured read data to align with read commands in the half rate controller clock domain. The interpretation of 1T is the same as for address and command, but applied to read data. This setting does not apply to a full-rate PHY.
  - Read resynchronization phase setting. This setting is the primary task of the training pattern to correctly set the resynchronization phase in the middle of data valid window for the read data (on DQ pins) to be captured.

From this algorithm, to determine PHY settings B and C, given that A is not set, follow these steps:

1. Try to write the pattern and indistinguishable variations of it to different memory locations. For example, write to different locations with the following two patterns on the DQ bus (timed to a local controller rate clock), refer to [Write More Training Patterns \(s\\_write\\_mtp\)](#):
  - a. 0011 0011 0011 (try to write 0011) in location X
  - b. 1100 1100 1100 (try to write 1100) in location Y
2. Perform two single-pin DQ pin and single-chip select read resynchronization phase calibrations using location X and location Y, as part of the larger training pattern (0x30F5). You do not know at this time which locations X and Y contain the pattern 0011. This stage iterates through the following stages:
  - Initialize read resynchronization phase calculation (s\_rrp\_reset)
  - Calculate read resynchronization phase (s\_rrp\_sweep)
  - Calculate read data valid window (s\_read\_mtp)
  - Calculate read data valid window (s\_read\_mtp) is a special case of calibrate read resynchronization phase (s\_rrp\_sweep), refer to [Calibrate Read Resynchronization Phase \(s\\_rrp\\_sweep\)](#). This stage reports the size of the returned window without setting up the PLL phase or producing an error if no window is observed.

3. The single pin read resynchronization calibration (using pattern X or Y), which results in the largest data valid window, contains the optimal pattern. The read resynchronization calibration with the largest window indicates the location (X or Y) that contains the correct alignment (0011). Read resynchronization phase calibration uses this alignment (X or Y), to perform the full resynchronization phase calibration across all pins and chip selects.



During calibration of the read resynchronization phase, the ALTMEMPHY IP captures the DQ pin using a free running clock, phase shifted through a given number of steps. You should try to match a training pattern against read data, for each phase shift, and a window of valid phases is composed.

In waveforms, you observe two resynchronization phase sweeps, over single pins, before the larger phase sweep. However in fast simulation mode, you observe two resynchronization phase sweeps when the duration of all three sweeps is equal. For half-rate interfaces, you may observe a total of six phase sweeps, where the entire calibration is repeated when the address and command 1T setting is toggled.

## Calibrate Read Resynchronization Phase

This stage encompasses the following calibration stages:

- Initialize Read Resynchronisation Phase Calibration (s\_rrp\_reset)
- Calibrate Read Resynchronization Phase (s\_rrp\_sweep)
- Calculate Read Resynchronization Phase (s\_rrp\_seek)

This stage adjusts the phase of the resynchronization (or capture) clock to determine the optimal phase that gives the greatest margin. For DQS based capture schemes the resynchronization clock captures the outputs of DQS capture registers (DQS is the capture clock). In a non-DQS capture based scheme, the capture clock captures the input DQ pin data (the DQS signal is unused).

For all half-rate PHY interfaces, a 720° resynchronization or capture clock phase sweep is performed. For a half-rate PHY this sweep is effectively 360° of the half-rate clock, because resynchronization or capture clock is at the memory clock rate. A 720° sweep is required, so that read data can be presented to a controller aligned to one half-rate controller clock cycle.

For full-rate DQS based capture, because the DQ pins are captured using the DQS signal, in a 360° phase sweep, all resynchronization clock phases may pass. In this case the correct resynchronization phase to set cannot be determined. The correct phase is the one in the center of a valid window, where returned read data is correct. Thus a 720° phase sweep is performed. From 360 to 720°, a clock cycle of latency may be added to a 0 to 360° sweep. The returned read data is compared to a training pattern pseudo half-rate (at half the clock rate of the sequencer), so data can only be valid for 360° of the sweep. This method introduces edges to the data valid window such that a correct phase can be chosen.

For non-DQS capture in general, up to half (180°) of the swept capture clock phases can result in correct capture data, because the DDR to SDR conversion is performed by the capture clock, and thus high and low phases of DQ are captured in the incorrect alignment for half of the capture clock phases. Therefore, only 360° of capture clock need be swept for full-rate non-DQS capture based PHYs.

The pseudo half-rate case potentially adds one clock cycle of latency into the read datapath because of the 720° sweep. The ALTMEMPHY IP detects this occurrence and removes the clock cycle of latency in the calculate read resynchronization phase (`s_rrp_seek`).

### Initialize Read Resynchronisation Phase Calibration (`s_rrp_reset`)

This stage returns the PLL to a nominal zero phase shift.

### Calibrate Read Resynchronization Phase (`s_rrp_sweep`)

This stage performs a sweep through a parameterised 360° or 720° of resynchronization clock phase. The ALTMEMPHY IP optionally stores these results in the internal RAM.

The command has the following attributes:

- `single_pin` to indicate just to sweep DQ pin 0 as for the use in testing the write more training patterns stage.
- `mtp_alignment` to say which location (X/Y) to use from the write more training patterns stage.

### Calculate Read Resynchronization Phase (`s_rrp_seek`)

This stage calculates the size and center (in phase steps) of the largest data valid window found during the calibrate read resynchronization phase and sets PLL phase to the center phase.

Calculate read data valid window (`s_read_mtp`) is a special case of this stage which reports no errors for an invalid window (a failure is expected in one case) and does not setup the PLL.

### Calculate Read Data Valid Window (`s_rdv`)

This stage sets the latency on a delayed version of the `doing_rd` signal, so that it is aligned with the `rdata_valid` signal for the read data it is incident with the read command for.

This stage has the following process:

1. Reads a continuous stream of 1s followed by one read of zeros. The sequencer only asserts `doing_rd` when read command for zeros is issued.
2. Checks for alignment of read data valid signal (delayed version of `doing_rd`) to the zeros (`rdata = 0, rdata_valid = 1`).
3. If not aligned, reduces latency between `doing_rd` and `rdata_valid` signal and loop.



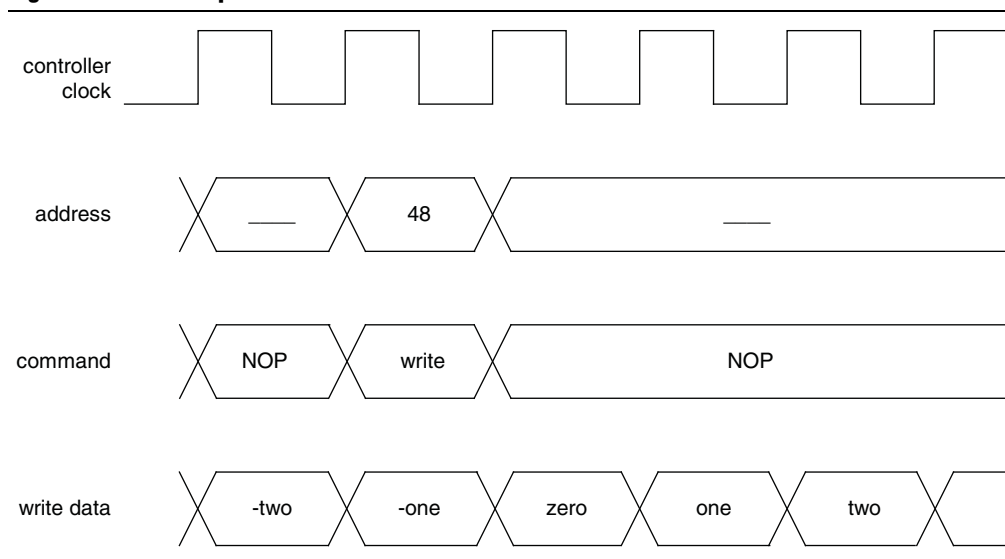
Read data valid latency is reset to a high value (before calibration) and then reduced until it matches the correct alignment.

## Advertize Write Latency (s\_was)

This stage writes a suitable pattern to the DRAM to calculate the write latency.

A write command is issued to a memory address 48 (Figure 2-16) while driving a count of frequency controller clock rate to the DQ pins (Figure 2-16), using the write datapath observed by the controller. For half-rate interfaces, each four beats of write data on DQ are identical. For full-rate interfaces, each two beats of write data are identical. In general, the write latency is written into addresses  $A \dots A + (n - 1)$ , where  $n$  is two times the ratio of controller to memory clock rate and each  $n$  bits of write data must be identical. The pattern is written into memory locations 48 to 55.

**Figure 2-16. Description of Write Pattern**




## Calculate Read Latency (s\_adv\_rlat)

In addition to read data valid window calculation ( $s_{rdv}$ ), the advertised read latency is calculated in this stage in the following way:

- Issues a read command (with `doing_rd`) and starts a counter at PHY clock rate
- When `rdata_valid` returns, outputs the value of the counter to `ctl_rlat` signal.

This signal is redundant, because a controller can use the `rdata_valid` signal to determine when valid read data is returned.

 The read data from the DRAM is not important here. The count is performed between the issue of `doing_rd` and `rdata_valid` returning.

## Output Write Latency (`s_adv_wlat`)

To calibrate a PHY write datapath to a minimum latency, a robust process is required to determine the write latency (WL) between a memory controller write command and the associated write data. Factors that can contribute to WL include memory CAS latency, arbitrary additive delays in the PHY, and board trace lengths. The presented approach extends from calibrating a PHY, where the controller operates at the memory clock frequency, to controller operation at half or a quarter of the memory clock frequency.

After a predetermined maximum read latency clock cycles have passed, the contents of the chosen memory address (0x48 in [Figure 2-16](#)) are read to recover the write latency. The first *n* beats of read data contain the write latency.

While this method recovers the write latency it can determine the address and command 1T setting in half rate mode.

The returned read data, in the controller clock domain, should be equal across the first four read data beats, as aligned to the controller clock domain. For this check to work an alternate read location must be read immediately before and after that containing the write latency.

If read data is not correctly aligned then the address and command 1T setting is toggled and calibration is rerun from write training patterns stage.

## Calibrate Postamble (`s_poa`)

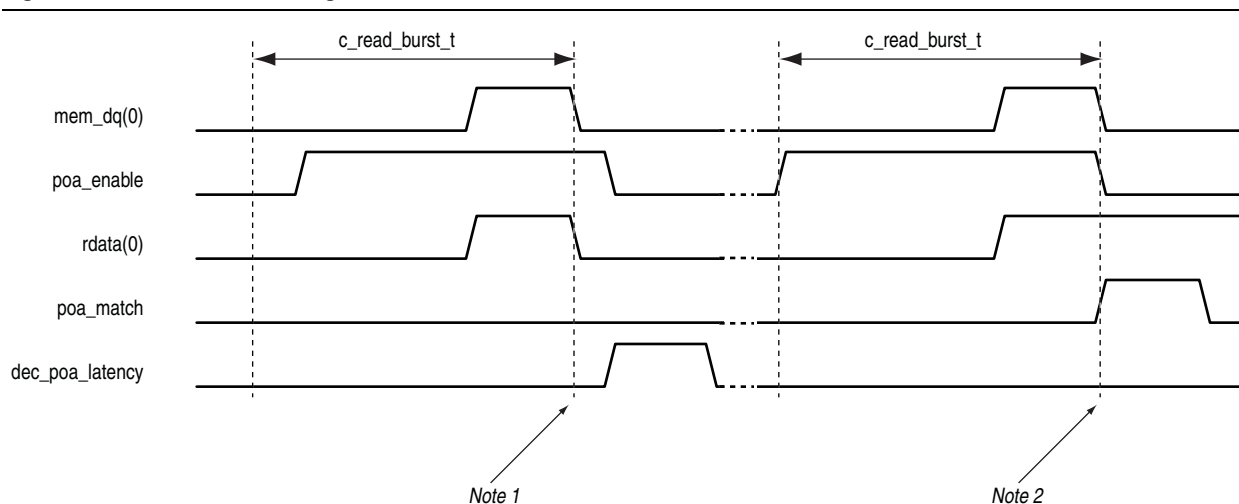
The ALTMEMPHY IP only implements this stage for DQS capture based PHYs (not used for Cyclone III and Cyclone IV devices).

For this stage, the PHY reads the pattern 0x30 from memory, so that the deassertion of the postamble protection signal (`poa_enable`) can be aligned to the 1's in this pattern.

This stage sets the correct clock cycle for the postamble path. The aim of the postamble path is to eliminate false DQ data capture because of postamble glitches on the DQS signal, through an override on DQS. This stage ensures the correct clock cycle timing of the postamble enable (override) signal.

The delay on the postamble enable signal starts off too large. It is then iteratively reduced until postamble enable de-asserts in the clock cycle before the last falling edge on DQS. Figure 2-17 shows the calibration timing diagram.

**Figure 2-17. Calibration Timing**



**Note to Figure 2-17:**

- (1) The poa\_enable signal is late, and the zeros on mem\_dq after here are captured.
- (2) The poa\_enable signal is aligned. Zeros following here are not captured and rdata remains at 1.

## Set Up Address and Command Clock Cycle

For half-rate interfaces, this stage also optionally adds an additional memory clock cycle of delay from the address and command path. This stage aligns write data to memory commands given in the controller clock domain. You see this stage in the waveform as a rerun of calibration (from the writing of training patterns) to calibrate to the new setting.

This stage is detected in the advertise write latency stage (s\_adv\_wlat)

## Write User Mode Register Settings (s\_prep\_customer\_mr\_setup)

User mode register setting applies on a per chip select basis without the overrides in the program mode registers for calibration (s\_prog\_mr) stage.

## Voltage and Temperature Tracking

Voltage and temperature tracking is a background process that tracks the voltage and temperature variations to maintain the relationship between the resynchronization or capture clock and the data valid window that were achieved at calibration. When the data calibration phase completes, the sequencer issues the mimic calibration sequence every 128 ms (in user mode).

### Setup the Mimic Window (s\_tracking\_setup)

During initial calibration, the mimic path is sampled using the measure clock. The measure\_clk signal has a \_1x or \_2x suffix, depending whether the ALTMEMPHY IP is a full-rate or half-rate design. The sampled value is then stored by the sequencer. After a sample value is stored, the sequencer uses the PLL reconfiguration logic to change the phase of the measure clock by one voltage-controlled oscillator (VCO) phase tap. The sequencer then stores the sampled value for the new mimic path clock phase. This sequence continues until all mimic path clock phase steps are swept. After the sequencer stores all the mimic path sample values, it calculates the phase which corresponds to the center of the high period of the mimic path waveform. This reference mimic path sampling phase is used during the voltage and temperature tracking phase.

### Perform Tracking (s\_tracking)

In user mode, the sequencer periodically performs a tracking operation. At the end of the tracking calibration, the sequencer compares the most recent optimum tracking phase against the reference sampling phase. If the sampling phases do not match, the mimic path delays have changed because of voltage and temperature variations. When the sequencer detects that the mimic path reference and most recent sampling phases do not match, the sequencer uses the PLL reconfiguration logic to change the phase of the resynchronization clock by the VCO taps in the same direction. This procedure allows the tracking process to maintain the near-optimum capture clock phase setup during data tracking calibration as voltage and temperature vary over time. The relationship between the resynchronization or capture clock and the data valid window is maintained by measuring the mimic path variations because of the voltage and temperature variations and applying the same variation to the resynchronization clock.

## Document Revision History

Table 2-6 lists the revision history for this document.

**Table 2-6. Document Revision History**

Date	Version	Changes
November 2011	3.1	<ul style="list-style-type: none"> <li data-bbox="508 1482 1398 1570">■ Consolidated ALTMEMPHY FD information from 11.0 version <b>DDR and DDR2 SDRAM Controller with ALTMEMPHY IP User Guide</b> and <b>DDR3 SDRAM Controller with ALTMEMPHY IP User Guide</b>.</li> <li data-bbox="508 1581 1052 1612">■ Added <b>ALTMEMPHY Calibration Stages</b> information.</li> </ul>