Nexus DDR3 Memory Controller Driver API Reference

User Guide

FPGA-TN-02401-1.2

June 2026

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviations	Definition
API	Application Programming Interface
APB	Advanced Peripheral Bus
DDR3	Double Data Rate Generation 3
FPGA	Field Programmable Gate Array
IP	Intellectual Property (FPGA)
JEDEC	Joint Electron Device Engineering Council
PHY	Physical Layer / Physical Interface
PLL	Phase-Locked Loop
SDK	Software Development Kit
SDRAM	Synchronous Dynamic Random Access Memory
ZQ	Output impedance

1. Introduction

The Lattice Semiconductor DDR3 Memory Controller for Lattice Nexus™ devices is a complete solution: a controller, a DDR PHY, and the clocking and training logic required to interface with DDR3 SDRAM.

For more information about the IP core and register descriptions, refer to the DDR3 SDRAM Controller IP for Nexus Devices (FPGA-IPUG-02086).

1.1. Purpose

The DDR3 driver and SDK provide a set of application programming interfaces (API) that provides access to specific Lattice hardware and software capabilities. Use this document as a reference guide; it describes the C-language driver APIs and function call flows.

1.2. Audience

This document is intended for embedded system designers and software developers who use Lattice Certus™-NX devices. This guide assumes you have expertise in embedded systems and FPGA technologies.

1.3. Driver and IP Compatibility

Table 1.1. Driver and IP Version

Driver version	IP version
26.01.00	2.4.0

For driver and IP version information, refer to the DDR3 SDRAM Controller IP Release Notes (FPGA-RN-02032).

Table 1.2. Quick Facts of Driver Tested Environment

Driver tested on HW Device	Nexus Family	Certus-NX
Tool Version	Lattice Propel™ Builder	2025.2
	Lattice Propel™ SDK	2025.2
	Lattice Radiant™ Software	2025.2
	Lattice Radiant™ Programmer	2025.2

2. API Description

2.1. ddr3_init()

This API is used to initialize the DDR3 instance.

```
unsigned int ddr3_init(ddr3 *instance_ptr, unsigned int base_addr);
```

Table 2.1. ddr3_init()

In/Out	Parameter	Description	Returns
In	instance_ptr	Handle the ddr3 structure.	0: failure 1: success
In	base_addr	Base address to be assigned to controller.	

2.2. ddr3_do_training()

This API performs basic training of the DDR3 IP. This starts the initialization and training sequence of the DDR3 SDRAM device. When the sequence completes, control of the PHY transfers to the memory controller, and you can access DDR3 memory through the data interface. Note: CBT_FAIL and WR_TRN_FAIL are reserved for future use and are not returned by the current implementation of ddr3_do_training().

```
DDR_RET ddr3_do_training(ddr3 *instance_ptr);
```

Table 2.2. ddr3_do_training()

In/Out	Parameter	Description	Returns
In	instance_ptr	Handle the ddr3 structure.	“Enum DDR_RET state” NO_FAIL = 0, CBT_FAIL, WR_LVL_FAIL, RD_TRN_FAIL, WR_TRN_FAIL, OTHER_FAIL

2.3. ddr3_PowerDownModeEnable()

This API enables power-down configuration settings in the Mode Control Register. You can set these attributes manually at runtime. After the DDR3 engine enters power-down mode, it remains in that mode until you manually issue a power-down exit command. The controller discards the power-down entry command if the memory is already in power-down mode. Refer to the IP user guide for register descriptions and user commands.

```
unsigned int ddr3_PowerDownModeEnable(ddr3 *instance_ptr);
```

Table 2.3. ddr3_PowerDownModeEnable()

In/Out	Parameter	Description	Returns
In	instance_ptr	Handle the ddr3 structure.	0: failure 1: success

2.4. ddr3_PowerDownModeDisable()

This API disables power-down configuration settings in the Mode Control Register. You can set these attributes manually at runtime. After the DDR3 engine enters power-down mode, this command brings the DDR3 controller out of power-down mode. The controller discards the power-down exit command if the memory is not already in power-down mode. Refer to the IP user guide for register descriptions and user commands.

```
unsigned int ddr3_PowerDownModeDisable(ddr3 *instance_ptr);
```

Table 2.4. ddr3_PowerDownModeDisable()

In/Out	Parameter	Description	Returns
In	instance_ptr	Handle the ddr3 structure.	0: failure 1: success

2.5. ddr3_SelfRefreshModeEnable()

This API enables self-refresh mode. The controller discards the self-refresh entry command if the memory is already in self-refresh mode. Refer to the IP user guide for register descriptions and user commands.

```
unsigned int ddr3_SelfRefreshModeEnable(ddr3 *instance_ptr);
```

Table 2.5. ddr3_SelfRefreshModeEnable()

In/Out	Parameter	Description	Returns
In	instance_ptr	Handle the ddr3 structure.	0: failure 1: success

2.6. ddr3_SelfRefreshModeDisable()

This API disables self-refresh mode. The controller discards the self-refresh exit command if the memory is not already in self-refresh mode. Refer to the IP user guide for register descriptions and user commands.

```
unsigned int ddr3_SelfRefreshModeDisable(ddr3 *instance_ptr);
```

Table 2.6. ddr3_SelfRefreshModeDisable()

In/Out	Parameter	Description	Returns
In	instance_ptr	Handle the ddr3 structure.	0: failure 1: success

2.7. ddr3_ZQcalibrationShortEnable()

This API enables short ZQ calibration during runtime. You can manually perform calibrations to account for voltage and temperature variations. Refer to the IP user guide for register descriptions and user commands.

```
unsigned int ddr3_ZQcalibrationShortEnable(ddr3 *instance_ptr);
```

Table 2.7. ddr3_ZQcalibrationShortEnable()

In/Out	Parameter	Description	Returns
In	instance_ptr	Handle the ddr3 structure.	0: failure 1: success

2.8. ddr3_ZQcalibrationLongEnable()

This API enables long ZQ calibration. Long calibration is a full calibration normally performed during reset/initialization. For quick calibration during runtime, use ddr3_ZQcalibrationShortEnable(). Refer to the IP user guide for register descriptions and user commands.

```
unsigned int ddr3_ZQcalibrationLongEnable(ddr3 *instance_ptr);
```

Table 2.8. ddr3_ZQcalibrationLongEnable()

In/Out	Parameter	Description	Returns
In	instance_ptr	Handle the ddr3 structure.	0: failure 1: success

2.9. ddr3_enable_data_mode()

This API enables data mode in the DDR3 controller. This switches the controller from configuration mode to data mode so you can access DDR3 memory through the data interface.

```
void ddr3_enable_data_mode(ddr3 *instance_ptr);
```

Table 2.9. ddr3_enable_data_mode()

In/Out	Parameter	Description	Returns
In	instance_ptr	Handle the ddr3 structure.	None

2.10. ddr3_enable_config_mode()

This API enables configuration mode in the DDR3 controller, switching it from data mode so you can access the controller through the APB interface.

```
void ddr3_enable_config_mode(ddr3 *instance_ptr);
```

Table 2.10. ddr3_enable_config_mode()

In/Out	Parameter	Description	Returns
In	instance_ptr	Handle the ddr3 structure.	None

2.11. ddr3_get_ddr_width()

This API returns the DDR width of the DDR3 controller.

```
unsigned int ddr3_get_ddr_width(ddr3 *instance_ptr);
```

Table 2.11. ddr3_get_ddr_width()

In/Out	Parameter	Description	Returns
In	instance_ptr	Handle the ddr3 structure.	ddr_width: 0 : 8-bit 1 : 16-bit 2 : 24-bit 3 : 32-bit

2.12. wr_dqdm_calibration()

This API performs write DQ/DM (data qualifier / data mask) calibration as part of extended training. It sweeps the WR DQDM delay to find the valid timing window (maximum and minimum boundaries), optionally adjusts rdclksel when the coarse tune threshold is reached, then sets the trained delay to the midpoint and updates the controller registers.

```
unsigned int wr_dqdm_calibration(ddr3 *instance_ptr, int checkpoint_num, struct
dqdm_sweep_info *sweep, int wr_coarse_tune);
```

Table 2.12. wr_dqdm_calibration()

In/Out	Parameter	Description	Returns
In	instance_ptr	Handle the ddr3 structure.	Error count: 0: No error Non-zero: failure
In	checkpoint_num	Base checkpoint value for scratch register. For debugging purposes. Define CALIBRATION_CHECKPOINT_TO_SCRATCH_REG via #define in the driver implementation or in the Propel SDK settings to enable write to scratch register.	
In	sweep	Sweep state and boundaries for DQDM calibration.	
In	wr_coarse_tune	Coarse tune step count before rdclksel adjust	

2.13. rd_dq_calibration()

This API performs read DQ calibration as part of extended training. It resets the RD DQ delay to zero and then either sweeps to find the valid RD DQ delay window (minimum and maximum boundaries) or, if the initial check passes, finds only the maximum and computes a theoretical midpoint. Results are written to scratch checkpoints, and the sweep state is updated.

```
unsigned int rd_dq_calibration(ddr3 *instance_ptr, int checkpoint_num, struct
dqdm_sweep_info *sweep);
```

Table 2.13. rd_dq_calibration()

In/Out	Parameter	Description	Returns
In	instance_ptr	Handle the ddr3 structure.	Error count: 0: No error Non-zero: failure
In	checkpoint_num	Base checkpoint value written to the scratch register for debugging purposes. Define CALIBRATION_CHECKPOINT_TO_SCRATCH_REG via #define in the driver implementation or in the Propel SDK settings to enable write to scratch register.	
In	sweep	Sweep state and boundaries for DQDM calibration.	

2.14. coarse_dqdm_calibration()

This API performs coarse DQ/DM alignment as part of extended training. It first drives WR DQDM and RD DQ into a known-good (passing) region by iteratively coarse-adjusting WR DQDM together with rdclksel, in both decrement and, if needed, increment directions, until coarse_adjust_wr_dqdm_rd_dq passes or overflows. It then invokes wr_dqdm_calibration to perform the actual WR DQDM window sweep (maximum, minimum, and midpoint).

```
int coarse_dqdm_calibration(ddr3 *instance_ptr, struct dqdm_sweep_info *sweep);
```

Table 2.14. coarse_dqdm_calibration()

In/Out	Parameter	Description	Returns
In	instance_ptr	Handle the ddr3 structure.	Error count: 0: No error Non-zero: failure
In	dqdm_sweep_info	Sweep state and boundaries for DQDM calibration.	

2.15. update_trained_val()

Reserved for internal use.

2.16. reset_delay_code()

Reserved for internal use.

3. Function Call Flow Diagrams

3.1. ddr3_init()

This section shows the function call flow for the *ddr3_init()* API.

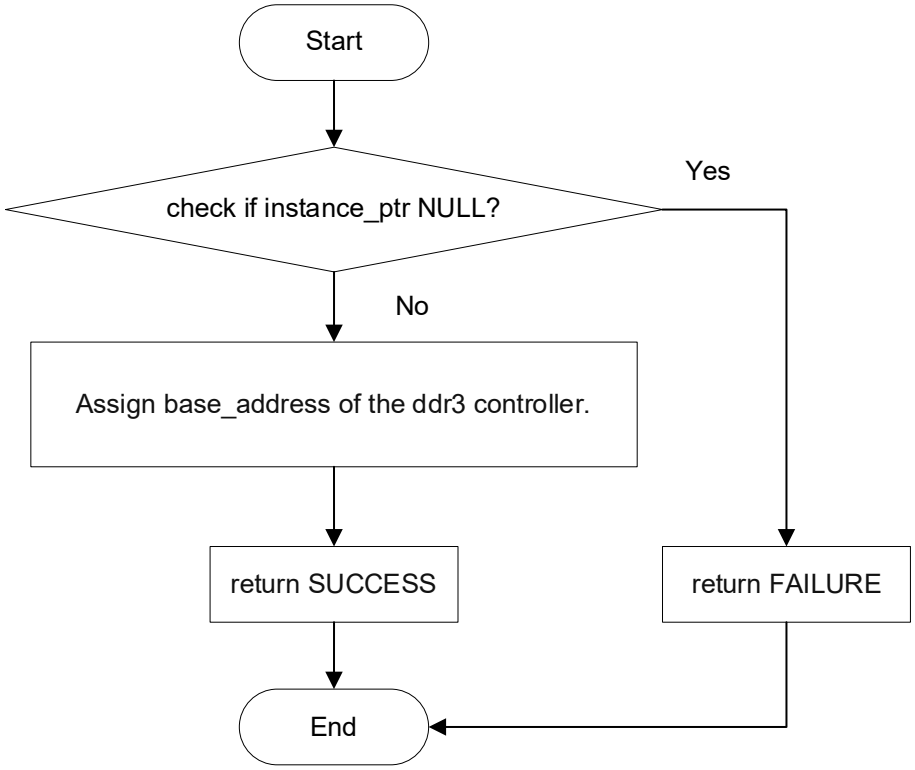

Figure 3.1. ddr3_init() Function Call Flow

3.2. ddr3_do_training()

This section shows the function call flow for the *ddr3_do_training()* API.

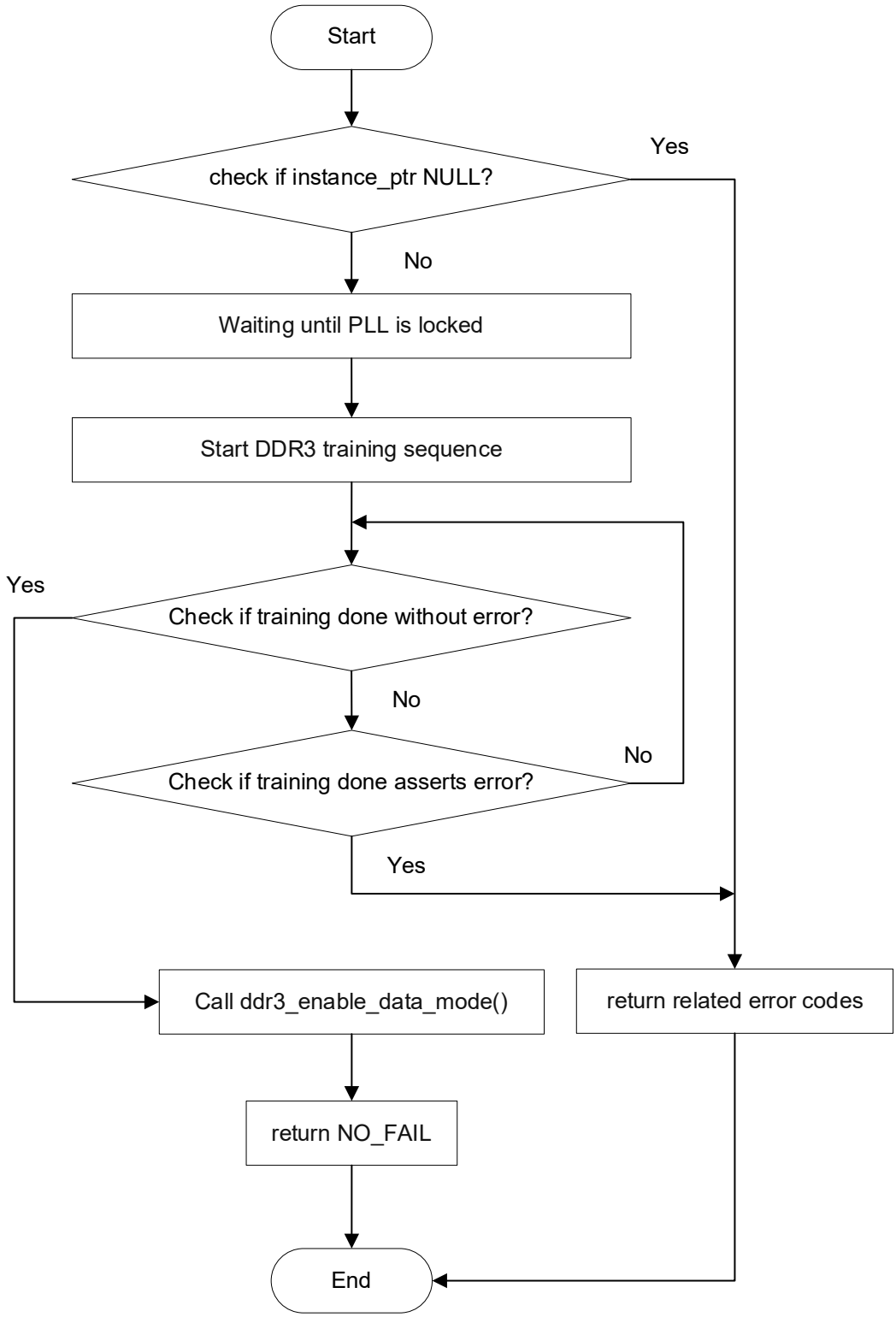

Figure 3.2. ddr3_do_training() Function Call Flow

3.3. ddr3_PowerDownModeEnable()

This section shows the function call flow for the *ddr3_PowerDownModeEnable()* API.

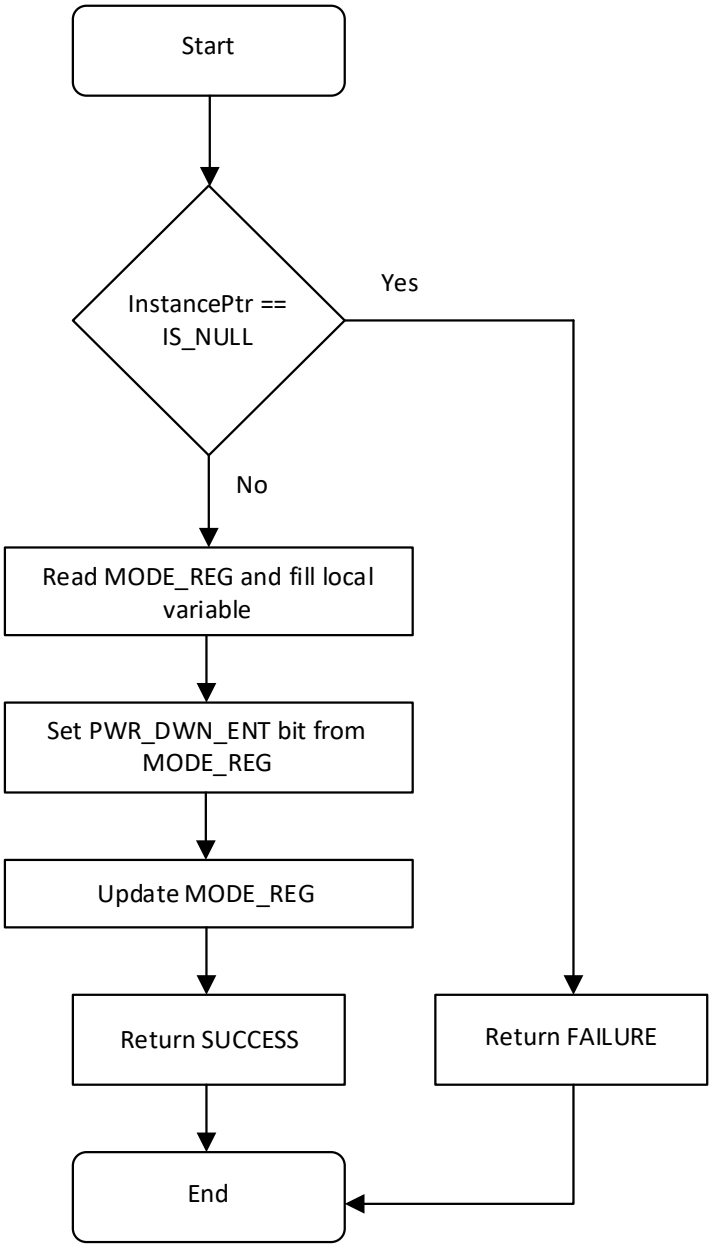

Figure 3.3. ddr3_PowerDownModeEnable() Function Call Flow

3.4. ddr3_PowerDownModeDisable()

This section shows the function call flow for the *ddr3_PowerDownModeDisable()* API.

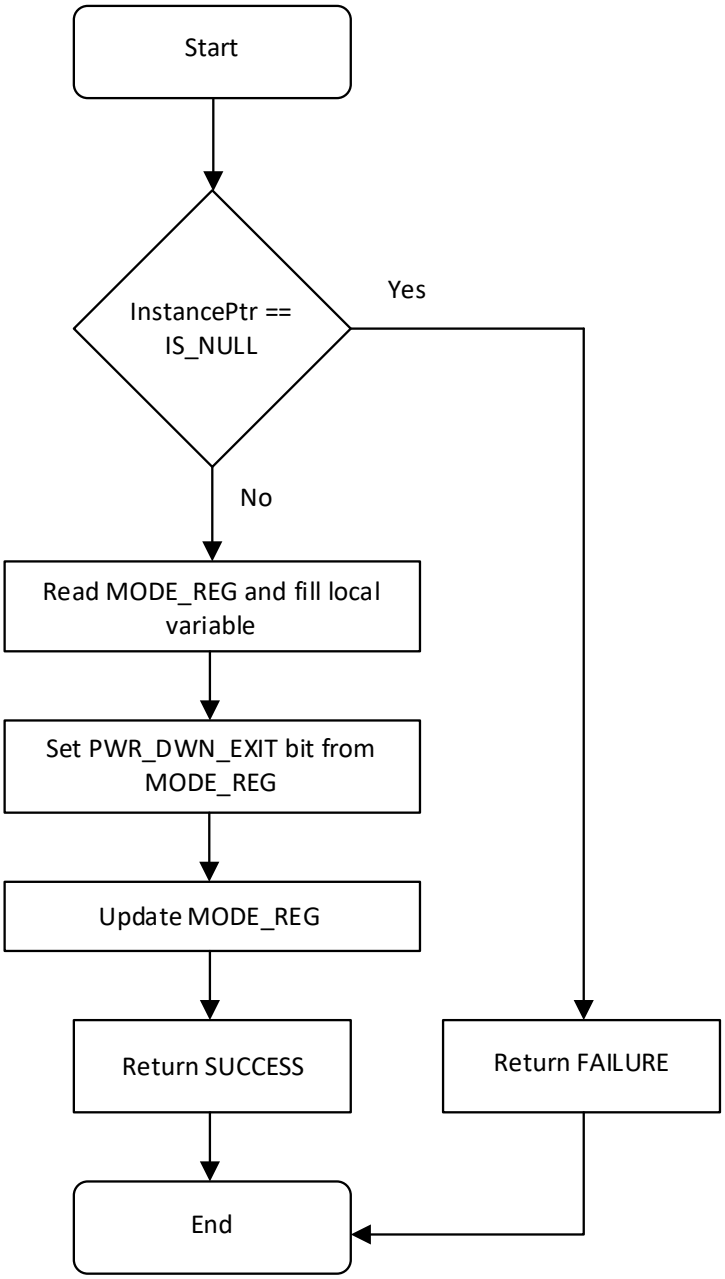

Figure 3.4. ddr3_PowerDownModeDisable() Function Call Flow

3.5. ddr3_SelfRefreshModeEnable()

This section shows the function call flow for the *ddr3_SelfRefreshModeEnable()* API.

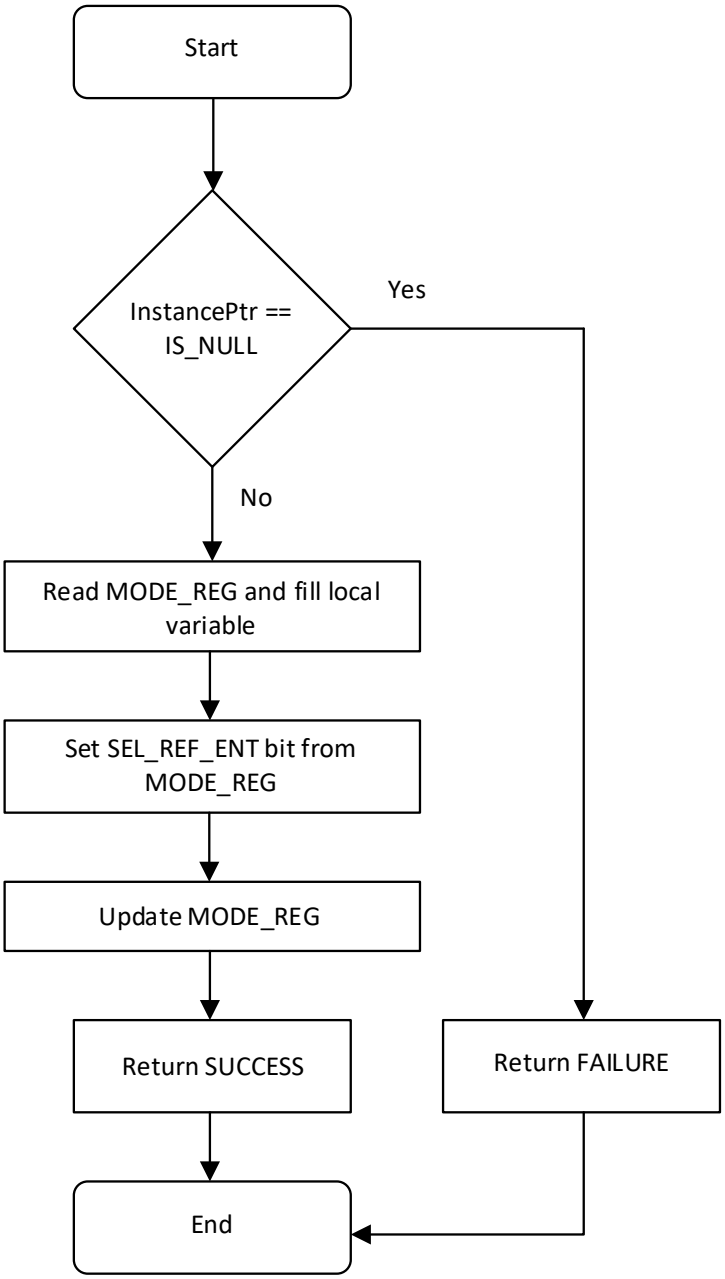

Figure 3.5. ddr3_SelfRefreshModeEnable() Function Call Flow

3.6. ddr3_SelfRefreshModeDisable()

This section shows the function call flow for the *ddr3_SelfRefreshModeDisable()* API.

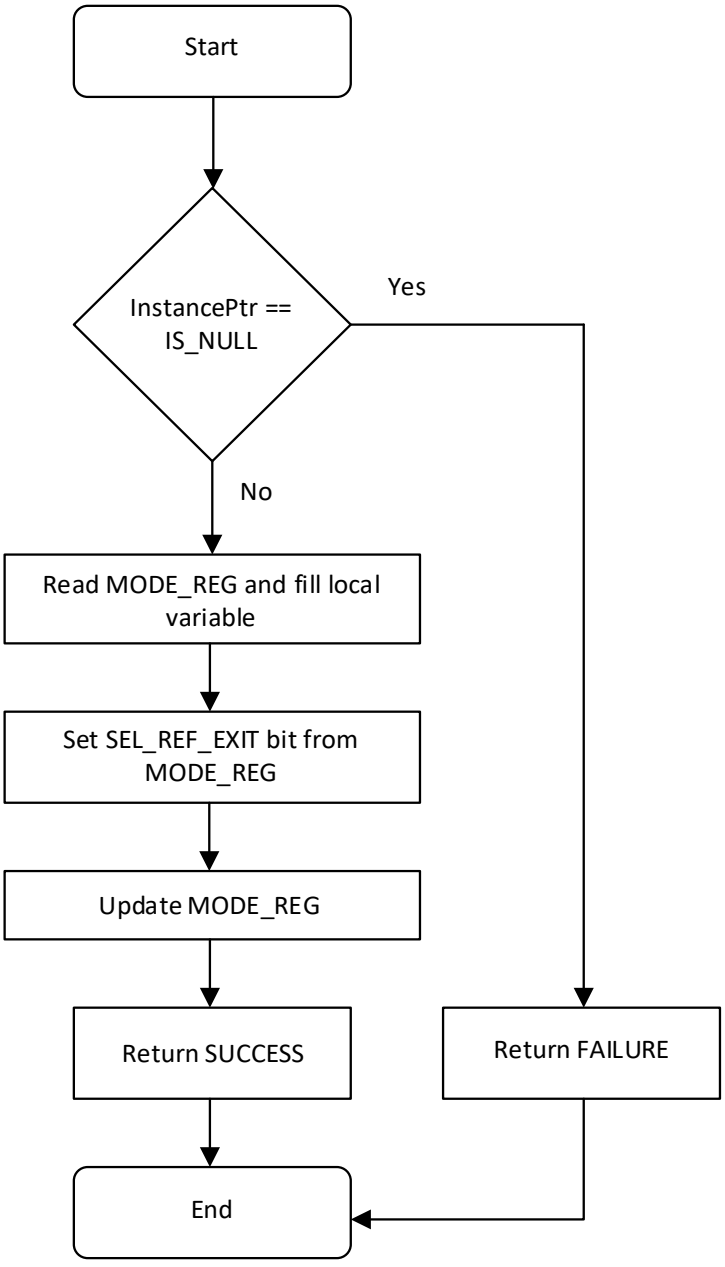

Figure 3.6. ddr3_SelfRefreshModeDisable() Function Call Flow

3.7. ddr3_ZQcalibrationShortEnable()

This section shows the function call flow for the *ddr3_ZQcalibrationShortEnable()* API.

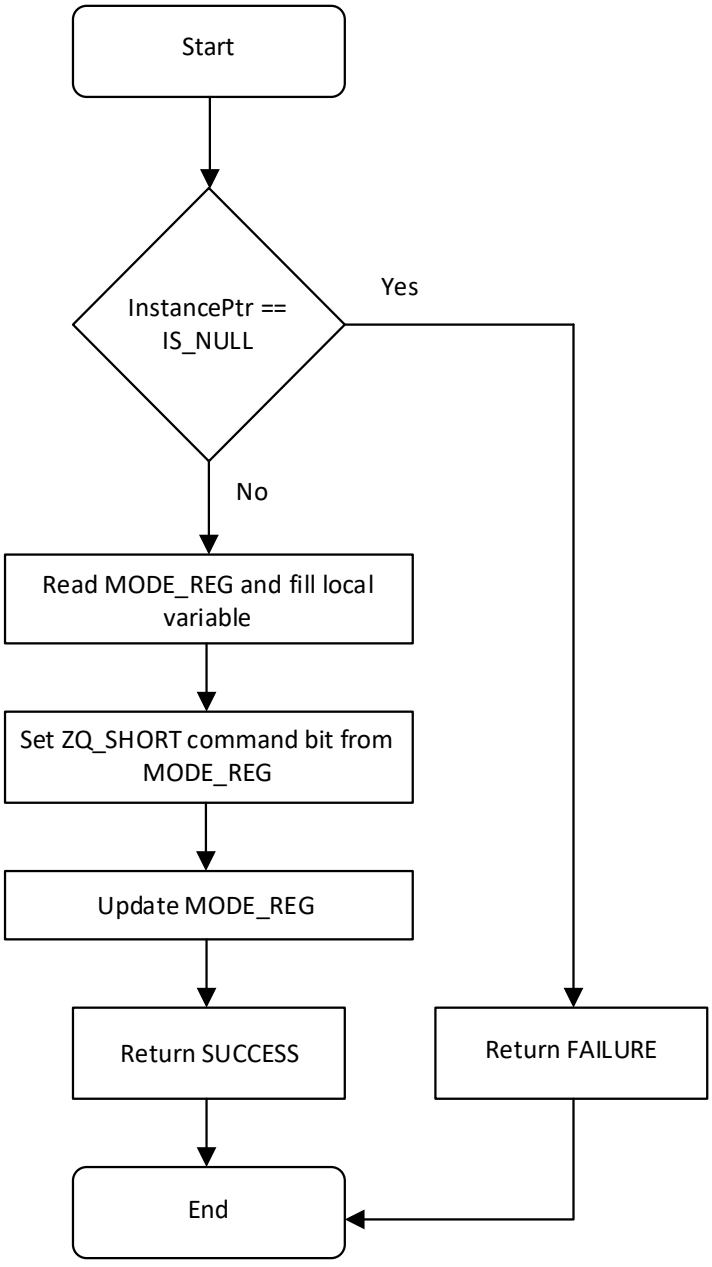

Figure 3.7. ddr3_ZQcalibrationShortEnable() Function Call Flow

3.8. ddr3_ZQcalibrationLongEnable()

This section shows the function call flow for the *ddr3_ZQcalibrationLongEnable()* API.

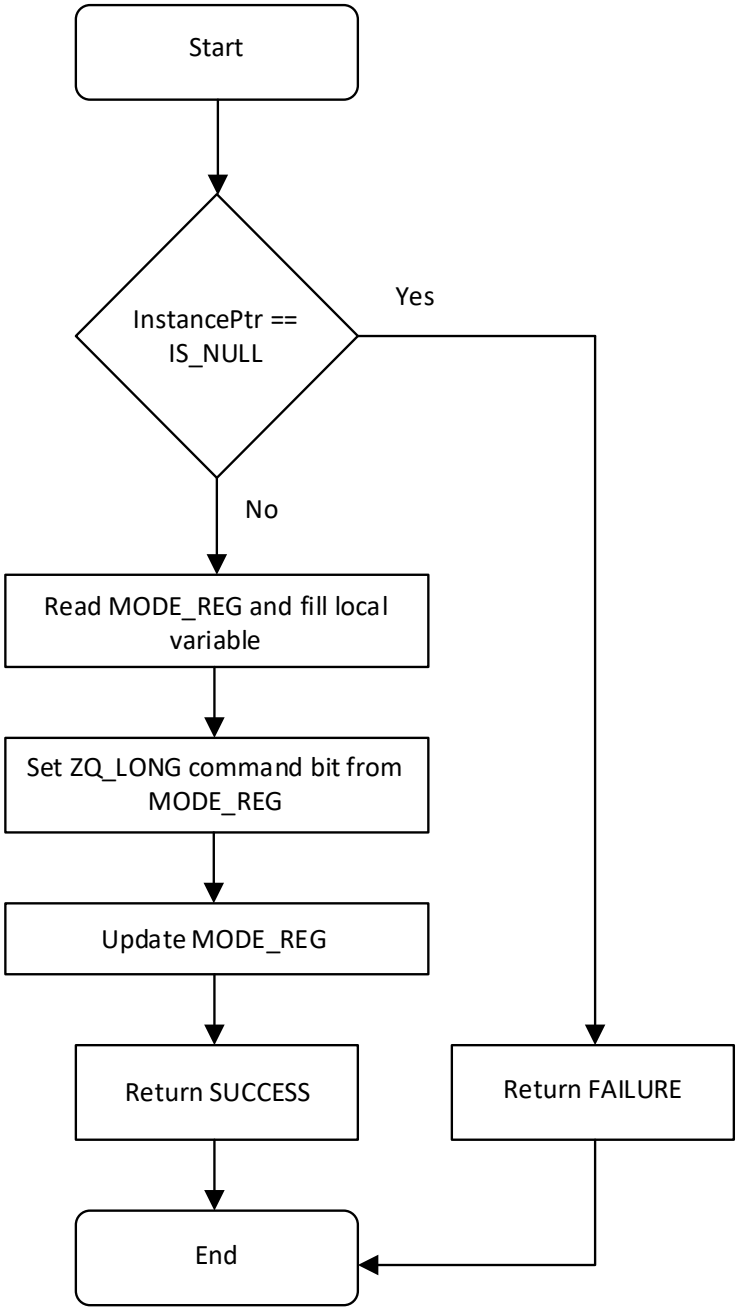

Figure 3.8. ddr3_ZQcalibrationLongEnable() Function Call Flow

3.9. ddr3_enable_data_mode()

This section shows the function call flow for the *ddr3_enable_data_mode()* API.

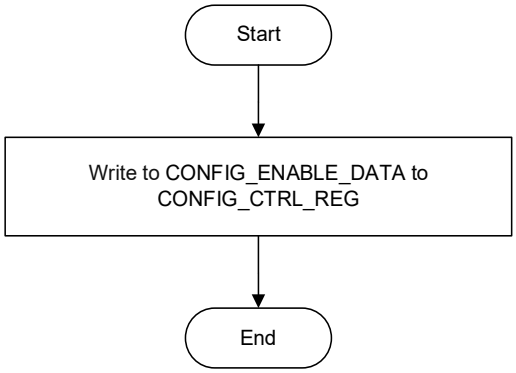

Figure 3.9. ddr3_enable_data_mode() Function Call Flow

3.10. ddr3_enable_config_mode()

This section shows the function call flow for the *ddr3_enable_config_mode()* API.

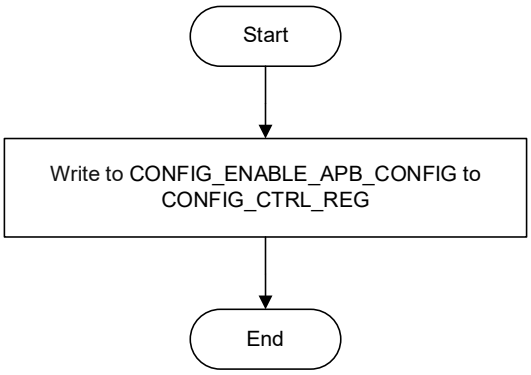

Figure 3.10. ddr3_enable_config_mode() Function Call Flow

3.11. ddr3_get_ddr_width()

This section shows the function call flow for the *ddr3_get_ddr_width()* API.

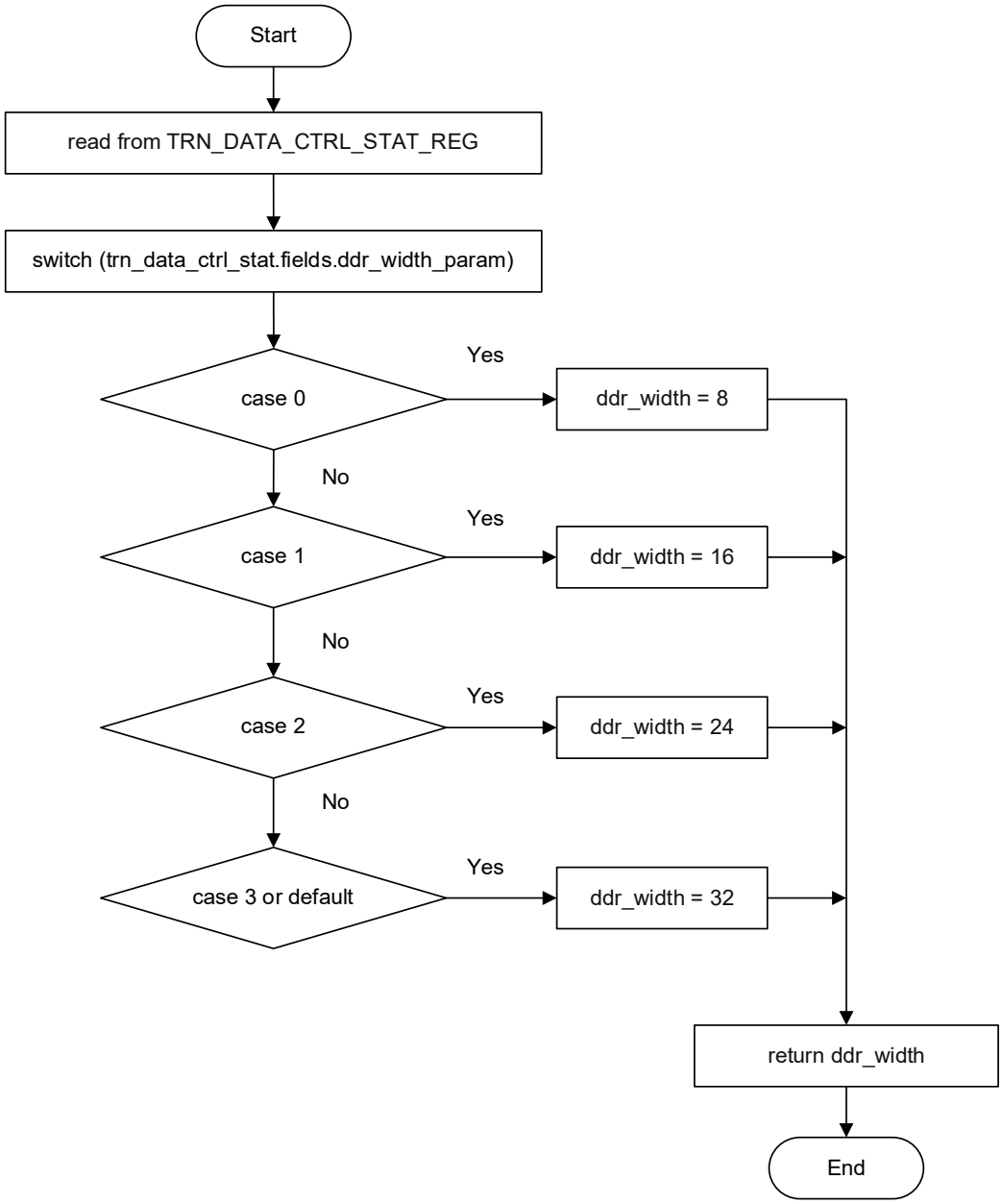

Figure 3.11.ddr3_get_ddr_width() Function Call Flow

The following APIs are reserved for internal use:

- wr_dqdm_calibration()
- rd_dq_calibration()
- coarse_dqdm_calibration()
- update_trained_val()
- reset_delay_code()

4. API Data Structures

4.1. Struct ddr3

Table 4.1. ddr3 Parameters

Data Type	Struct Member	Description
unsigned int	ddr3_base_address	Base address of the DDR3 memory controller.

4.2. Union and Struct config_ctrl_reg_t

Refer to the DDR3 SDRAM Controller IP for Nexus Devices (FPGA-IPUG-02086) user guide for detailed register description.

Table 4.2. config_ctrl_reg_t Parameters

Data Type	Union Member	Description
Struct	fields	Variable to access the config_ctrl_reg_t register members.
unsigned int	reg	Variable to access the config_ctrl_reg_t register.
Data Type	**fields struct Members**	**Description**
unsigned int	init_start	Write a '1' to this bit to begin initialization. This is a strobe bit.
unsigned int	config_rsvd1	Reserved
unsigned int	start_wl	Write a '1' to this bit to begin write leveling. This is a strobe bit.
unsigned int	config_data_en	Write a '1' to this bit to enable configuration access mode. This bit is sticky (it retains its value until explicitly cleared). To return to data access mode, write a '0' to this bit.
unsigned int	config_rsvd2	Reserved

4.3. Union and Struct mode_settings_reg_t

Refer to the DDR3 SDRAM Controller IP for Nexus Devices (FPGA-IPUG-02086) user guide for detailed register description.

Table 4.3. mode_settings_reg_t Parameters

Data Type	Union Member	Description
Struct	fields	Variable to access the mode_settings_reg_t register members.
unsigned int	reg	Variable to access the mode_settings_reg_t register.
Data Type	**fields struct Members**	**Description**
unsigned int	user_cmd	Supported user commands: • Power down entry • Self-refresh entry • Self-refresh exit • Power down exit • ZQ calibration long • ZQ calibration short Refer to the User Commands section in the DDR3 SDRAM Controller IP for Nexus Devices (FPGA-IPUG-02086) for the complete list of supported user commands via Native interface.
unsigned int	mode_reg_rsvd1	Reserved

4.4. Union and Struct status_reg_t

Refer to the DDR3 SDRAM Controller IP for Nexus Devices (FPGA-IPUG-02086) user guide for detailed register description.

Table 4.4. status_reg_t Parameters

Data Type	Union Member	Description
Struct	fields	Variable to access the status_reg_t register members.
unsigned int	reg	Variable to access the status_reg_t register.
Data Type	**fields struct Members**	**Description**
unsigned int	init_done	1: Initialization done 0: Initialization not done
unsigned int	write_lvl_done	1: Write leveling done 0: Write leveling not done
unsigned int	write_lvl_err	1: Write leveling failed 0: Write leveling successful
unsigned int	read_trn_err	1: Read training failed 0: Read training successful
unsigned int	clk_good	1: Clock is ready 0: Clock not ready
unsigned int	cmd_ready	1: Ready to accept commands 0: Not ready to accept commands
unsigned int	pll_lock	1: PLL Lock asserted 0: PLL Lock not asserted
unsigned int	self_ref_mode	1: Self-refresh mode 0: Out of Self-Refresh mode
unsigned int	pwr_down_mode	1: In power-down mode 0: Out of power down mode
unsigned int	wr_dqdm_del_err	1: WR DQ/DM delay error 0: No WR DQ/DM delay error
unsigned int	rd_dq_del_err	1: RD DQ delay error 0: No RD DQ delay error
unsigned int	ck_delay_actual	Actual CK delay value
unsigned int	trn_data_check_err	1: Training-data check error 0: No Training-data check error
unsigned int	trn_data_rd_done	1: Training-data read complete 0: No Training-data read complete
unsigned int	trn_data_rd_timeout	1: Training-data read timeout 0: No Training-data read timeout
unsigned int	sts_rsvd1	Reserved

The following APIs are reserved for internal use:

- Union and Struct delay_code_rst_t
- Union and Struct delay_code_dir_t
- Union and Struct delay_code_move_t
- Union and Struct delay_code_co_t
- Union and Struct wr_dqdm_0_t
- Union and Struct rd_dq_0_t
- Union and Struct rd_dq_phase_t
- Union and Struct common_dqdm_reg_t
- Union and Struct ddr_dll_delay_t
- Union and Struct trn_data_ctrl_stat_t
- Struct dqdm_sweep_info

5. API Enum

5.1. Enum DDR_RET

Table 5.1. DDR_RET Variables

Enum Member	Enum Decimal Value
NO_FAIL	0
CBT_FAIL	1
WR_LVL_FAIL	2
RD_TRN_FAIL	3
WR_TRN_FAIL	4
OTHER_FAIL	5

6. API Macros

Table 6.1. API Macros Description

Macro	Description
#define DDR3_DRV_VER	DDR3 driver version
#define CONFIG_CTRL_REG	Register address of config control register
#define STATUS_REG	Register address of status register
#define MODE_REG	Register address of mode settings register
#define DELAY_CODE_RST_REG	Reserved for internal use
#define DELAY_CODE_DIR_REG	Reserved for internal use
#define DELAY_CODE_MOVE_REG	Reserved for internal use
#define DELAY_CODE_CO_REG	Reserved for internal use
#define SCRATCH_0_REG	Reserved for internal use
#define SCRATCH_1_REG	Reserved for internal use
#define WR_DQDM_0_REG	Reserved for internal use
#define RD_DQ_0_REG	Reserved for internal use
#define RD_DQ_PHASE_REG	Reserved for internal use
#define DDR_DLL_DELAY_REG	Reserved for internal use
#define RDCLKSEL_REG	Reserved for internal use
#define WR_TRN_DATA_REG	Reserved for internal use
#define TRN_DATA_CTRL_STAT_REG	Reserved for internal use
#define CONFIG_ENABLE_APB_CONFIG	Configuration for APB mode
#define CONFIG_ENABLE_DATA	Configuration for data mode
#define CONFIG_START_WRITE_LEVELING	Configuration to start write leveling
#define CONFIG_INITIALISATION	Configuration to start initialisation
#define STATUS_TRN_DATA_RD_DONE	Extended training data read done status
#define STATUS_TRN_DATA_CHECK_ERR	Extended training data check error status
#define STATUS_RD_DQ_DEL_ERR	Read DQ delay calibration error status
#define STATUS_WR_DQDM_DEL_ERR	Write DQDM phase adjustment error status
#define STATUS_PWR_DOWN_MODE	Power down mode status
#define STATUS_SELF_REFRESH_MODE	Self-refresh mode status
#define STATUS_PLL_LOCK	PLL lock status
#define STATUS_CMD_READY	Command ready to be accepted status
#define STATUS_CLOCK_GOOD	Clock good status
#define STATUS_READ_TRAIN_ERR	Read training error status
#define STATUS_WRITE_LEVEL_ERR	Write leveling error status
#define STATUS_WRITE_LEVEL_DONE	Write leveling done status
#define STATUS_INIT_DONE	Initialization done status
#define DDR3_DONE_BITS	DDR3 initialization done bits
#define DDR3_DONE_BITS_MASK	Mask for DDR3 initialization done bits
#define TRN_EN	DDR3 training enable bits
#define TWO_US_DELAY	2 microsecond delay
#define PWR_DWN_ENT	Power down entry
#define PWR_DWN_EXIT	Power down exit
#define SEL_REF_ENT	Self-refresh entry
#define SEL_REF_EXIT	Self-refresh exit
#define ZQ_SHORT	ZQ calibration short
#define ZQ_LONG	ZQ calibration long
#define COARSE_TUNE 16	Reserved for internal use
#define NORMAL_TUNE	Reserved for internal use

Macro	Description
#define FINE_TUNE	Reserved for internal use
#define RDCLKSEL_ADJUST	Reserved for internal use
#define MAX_RDCLKSEL	Reserved for internal use
#define MIN_RDCLKSEL	Reserved for internal use
#define WR_DQDM	Reserved for internal use
#define RD_DQ	Reserved for internal use
#define RD_DQ_PHASE	Reserved for internal use
#define RDCLKSEL	Reserved for internal use
#define UPDATE_MAX	Reserved for internal use
#define UPDATE_MIN	Reserved for internal use
#define UPDATE_CNT	Reserved for internal use
#define DQDM_MAX_LIMIT	Reserved for internal use
#define DQDM_MIN_LIMIT	Reserved for internal use
#define TRN_EXP_DATA_W0	Reserved for internal use
#define TRN_EXP_DATA_W1	Reserved for internal use
#define TRN_EXP_DATA_W2	Reserved for internal use
#define TRN_EXP_DATA_W3	Reserved for internal use
#define TRN_EXP_DATA_W4	Reserved for internal use
#define TRN_EXP_DATA_W5	Reserved for internal use
#define TRN_EXP_DATA_W6	Reserved for internal use
#define TRN_EXP_DATA_W7	Reserved for internal use
#define SUCCESS	Success status
#define FAILURE	Failure status

References

- DDR3 SDRAM Controller IP for Nexus Devices (FPGA-IPUG-02086)
- DDR3-SDRAM-Controller-IP-Release-Notes (FPGA-RN-02032)
- Lattice Propel Builder User Guide (FPGA-UG-02243)
- Lattice Solutions IP Cores web page
- Lattice Propel Design Environment web page
- Lattice Radiant Software User Guide
- Lattice Radiant FPGA design software
- Lattice Insights for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.2, June 2026

Section	Change Summary
All	Minor editorial fixes.
Abbreviations in This Document	• Added *APB*, *PHY*, *PLL*, and *ZQ* in this section. • Removed *SCLK* in this section.
Introduction	• Updated driver version to *26.01.00* in Table 1.1. Driver and IP Version. • Updated IP version to *2.4.0* in Table 1.1. Driver and IP Version. • Updated all tool version to *2025.2* in Table 1.2. Quick Facts of Driver Tested Environment.

API Description	• Changed all parameter name *InstancePtr* to *instance_ptr* in this section.
	• Updated ddr3_init() section.
	  • Changed *DDR3_init()* to *ddr3_init()*.
	  • Updated return type from *DDR_RET* enum to *unsigned int*.
	  • Updated the *API returns* in Table 2.1. ddr3_init().
	• Added the following APIs:
	  • ddr3_do_training()
	  • ddr3_enable_data_mode()
	  • ddr3_enable_config_mode()
	  • ddr3_get_ddr_width()
	  • wr_dqdm_calibration()
	  • rd_dq_calibration()
	  • coarse_dqdm_calibration()
	• Listed the following API's reserved for internal use.
	  • update_trained_val()
	  • reset_delay_code()

Function Call Flow Diagrams	• Updated figure captions to include *Function Call Flow* in this section.
	• Updated Figure 3.1. ddr3_init() Function Call Flow.
	• Added the following function call flow diagrams:
	  • ddr3_do_training()
	  • ddr3_enable_data_mode()
	  • ddr3_enable_config_mode()
	  • ddr3_get_ddr_width()
	• Listed the following APIs reserved for internal use.
	  • wr_dqdm_calibration()
	  • rd_dq_calibration()
	  • coarse_dqdm_calibration()
	  • update_trained_val()
	  • reset_delay_code()

API Data Structures	• Changed *Struct DDR3* to *Struct ddr3*.
	• Updated description of config_data_en to clarify sticky bit behavior in Table 4.2. config_ctrl_reg_t Parameters.
	• Added the following APIs in Table 4.4. status_reg_t Parameters.
	  • wr_dqdm_del_err
	  • rd_dq_del_err
	  • ck_delay_actual
	  • trn_data_check_err
	  • trn_data_rd_done
	  • trn_data_rd_timeout
	• Added the following:
	  • Union and Struct delay_code_rst_t

Section	Change Summary
	• Union and Struct delay_code_dir_t • Union and Struct delay_code_move_t • Union and Struct delay_code_co_t • Union and Struct wr_dqdm_0_t • Union and Struct rd_dq_0_t • Union and Struct rd_dq_phase_t • Union and Struct common_dqdm_reg_t • Union and Struct ddr_dll_delay_t • Union and Struct trn_data_ctrl_stat_t • Struct dqdm_sweep_info
API Variables	Removed this section.

API Macros	• Reworked section contents.
	• Updated Table 6.1. API Macros Description.
	  • Replaced *DATA_ACCESS_EN* with *CONFIG_ENABLE_DATA*.
	  • Removed *IS_NULL* macro.

Revision 1.1, July 2025

Section	Change Summary
All	Minor editorial fixes.
Introduction	• Updated the driver version to *25.01.00*. • Updated the IP version to *2.2.0*.
Function Call Flow Diagrams	Updated Figure 3.1. ddr3_init().
API Data Structures	Added the DDR3 SDRAM Controller IP for Nexus Devices (FPGA-IPUG-02086) reference in Table 4.3. mode_settings_reg_t Parameters.
API Macros	Added DATA_ACCESS_EN API Macro.

Revision 1.0, March 2025

Section	Change Summary
All	Initial release.

www.latticesemi.com