I3C Controller Driver API Reference

Technical Note

FPGA-TN-02342-1.1

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.

Abbreviation	Definition
ACK	Acknowledgement
APB	Advanced Peripheral Bus
API	Application Programming Interfaces
BCR	Bus Characteristic Register
CCC	Common Control Code
CRC	Cyclic Redundancy Check
DAA	Dynamic Address Assignment
DCR	Device Characteristic Register
DDR	Double Data Rate
FIFO	First In First Out
FPGA	Field Programmable Gate Array
HDR	High Data Rate
HJ	Hot Join
I2C	Inter-Integrated Circuit
I3C	Improved Inter-Integrated Circuit
IBI	In Band Interrupt
IP	Intellectual Property
LMMI	Lattice Memory Mapped Interface
NACK	No Acknowledgement
PID	Provisional ID
RISC	Reduced Instruction Set Computer
Rx	Receive
SCL	Serial Clock
SDK	Software Development Kit
SDR	Single Data Rate
Tx	Transmit

1. Introduction

The I3C IP is a two-wire bi-directional serial bus, optimized for multiple sensor secondary devices and controlled by only one I3C Controller device at a time. The I3C protocol is backward compatible with many legacy I2C devices. The protocol supports significantly higher speeds, new communication modes, and new device roles, including an ability to change device roles over time when connected to I3C devices.

For more details on the I3C Controller IP core, refer to the I3C Controller IP User Guide (FPGA-IPUG-02228).

1.1. Purpose

I3C Controller and its SDK is a set of application programming interfaces (APIs) that provides access to Lattice-specific hardware and software capabilities. This document is intended to act as a reference guide for developers by providing details of the I3C Controller C-language driver API description, function call flow diagrams, API data structures, API variables, and API macros.

1.2. Audience

The intended audience for this document includes embedded system designers and embedded software developers using Lattice MachXO2™, MachXO3D™, MachXO3L™, MachXO3LF™, CrossLink™-NX, Certus™-NX, CertusPro™-NX, Mach™-NX, MachXO5™-NX, and Lattice Avant™ devices. The technical guidelines assume readers have expertise in the embedded system area and FPGA technologies.

1.3. Driver Versioning

1.3.1. Driver Version

I3C_CONTROLLER_DRV_VER "v1.1.0"

1.4. Driver and IP Compatibility

Driver Version	IP Version
1.1.0	3.7.1

Refer to the I3C Controller IP Release Notes (FPGA-RN-02017) for more information on the driver and IP versions.

2. API Description

The I3C Controller controls peripheral and Target devices by writing to and reading from configuration registers. The available APIs are listed below.

2.1. i3c_controller_init()

This API initializes the base address of the I3C Controller. It enables the clock signal and interrupt signal and sets the transfer mode.

```
uint8_t i3c_controller_init(struct i3c_controller_instance *this_i3cm, uint32_t addr,
uint8_t mode, uint8_t clk, uint8_t slave_nums, uint32_t fifo_depth)
```

In/Out	Parameter	Description	Returns
In	this_i3cm	Member of the i3c_controller_instance structure. Pointer to the Controller instance to initialize.	1: success 0: failure
In	addr	Member of the i3c_controller_instance structure. Base address of the I3C Controller registers.	
In	mode	Member of the i3c_controller_instance structure. Mode flags (I3C/I2C-specific behavior).	
In	clk	Member of the i3c_controller_instance structure. Clock (divider or pulse width) configuration.	
In	slave_nums	Member of the i3c_controller_instance structure. Number of connected Target devices.	
In	fifo_depth	Member of the i3c_controller_instance structure. FIFO depth of the I3C Controller.	

2.2. i3c_controller_daa()

This API performs dynamic address assignment (DAA). It assigns dynamic addresses and captures the PID, BCR, and DCR.

```
uint8_t i3c_controller_daa(struct i3c_controller_instance *this_i3cm, struct i3c_dev_info
*dev, uint8_t *dyn_addr, uint32_t device_count, uint8_t slave_addr)
```

In/Out	Parameter	Description	Returns
In	this_i3cm	Member of the i3c_controller_instance structure. Pointer to the Controller instance.	1: success 0: failure
Out	dev_info	Member of the i3c_dev_info structure. Pointer to the device information structure to be filled.	
In	dyn_addr	Member of the i3c_dev_info structure. Pointer to the dynamic address to assign.	
In	device_count	Member of the i3c_controller_instance structure. Number of devices to enumerate.	
In	slave_addr	Member of the i3c_dev_info structure. Static Target address used during enumeration.	

2.3. i3c_controller_private_i3c_write()

This API writes data in the I3C private transfer mode. It follows the defined user packet frame and writes data to the Tx FIFO through the APB interface.

```
uint8_t i3c_controller_private_i3c_write(struct i3c_controller_instance *this_i3cm, uint8_t
*wr_buf, uint32_t wr_len, uint8_t slave_dyn_addr)
```

In/Out	Parameter	Description	Returns
In	this_i3cm	Member of the i3c_controller_instance structure. Pointer to the Controller instance.	1: success 0: failure
In	wr_buf	Member of the i3c_controller_instance structure. Data buffer to write.	
In	wr_len	Member of the i3c_controller_instance structure. Length of data to write.	
In	slave_dyn_addr	Member of the i3c_dev_info structure. Dynamic address of the Target device.	

2.4. i3c_controller_private_i3c_read()

This API reads data in the I3C private transfer mode. The read response from the Target device is stored in the Rx FIFO and read through the APB interface.

```
uint8_t i3c_controller_private_i3c_read(struct i3c_controller_instance *this_i3cm, uint8_t *rd_buf, uint32_t rd_len, uint8_t slave_dyn_addr)
```

In/Out	Parameter	Description	Returns
In	this_i3cm	Member of the i3c_controller_instance structure. Pointer to the Controller instance.	1: success 0: failure
Out	rd_buf	Member of the i3c_controller_instance structure. Buffer pointer where data is collected.	
In	rd_len	Member of the i3c_controller_instance structure. Buffer length of read data.	
In	slave_dyn_addr	Member of the i3c_dev_info_ structure. Dynamic address of the Target device.	

2.5. i3c_controller_private_i2c_write()

This API writes data in the I2C private transfer mode. It follows the defined user packet frame and writes data to the Tx FIFO through the APB interface.

```
uint8_t i3c_controller_private_i2c_write(struct i3c_controller_instance *this_i3cm, uint8_t *wr_buf, uint32_t wr_len, uint8_t slave_dyn_addr)
```

In/Out	Parameter	Description	Returns
In	this_i3cm	Member of the i3c_controller_instance structure. Pointer to the Controller instance.	1: success 0: failure
In	wr_buf	Member of the i3c_controller_instance structure. Buffer pointer where data is assigned.	
In	wr_len	Member of the i3c_controller_instance structure. Buffer length of write data.	
In	slave_dyn_addr	Member of the i3c_dev_info_ structure. Dynamic address of the Target device.	

2.6. i3c_controller_private_i2c_read()

This API reads data in the I2C private transfer mode. The read response from the Target device is stored in the Rx FIFO and read through the APB interface.

```
uint8_t i3c_controller_private_i2c_read(struct i3c_controller_instance *this_i3cm, uint8_t *rd_buf, uint32_t rd_len, uint8_t slave_dyn_addr)
```

In/Out	Parameter	Description	Returns
In	this_i3cm	Member of the i3c_controller_instance structure. Pointer to the Controller instance.	1: success 0: failure
Out	rd_buf	Member of the i3c_controller_instance structure. Buffer pointer where data is collected.	
In	rd_len	Member of the i3c_controller_instance structure. Buffer length of read data.	
In	slave_dyn_addr	Member of the i3c_dev_info_ structure. Dynamic address of the Target device.	

2.7. i3c_controller_broadcast_ccc()

This API checks whether the active I3C Controller sends Broadcast CCC commands to address all Target registers.

```
uint8_t i3c_controller_broadcast_ccc(struct i3c_controller_instance *this_i3cm, uint8_t *buf, uint32_t size)
```

In/Out	Parameter	Description	Returns
In	this_i3cm	Member of the i3c_controller_instance structure. Pointer to the Controller instance.	1: success 0: failure
In	buf	Member of the i3c_controller_instance structure. Buffer pointer where data is assigned.	
In	size	Member of the i3c_controller_instance structure. Buffer length of write CCC command.	

2.8. i3c_controller_direct_ccc()

This API checks whether the active I3C Controller sends Direct CCC read and write commands to address individual Target registers.

```
uint8_t i3c_controller_direct_ccc(struct i3c_controller_instance *this_i3cm, uint8_t slave_dyn_addr, uint8_t *buf, uint32_t buf_size, uint8_t *buf2, uint32_t buf_size2, uint8_t rnw, uint8_t *ret_buf, uint32_t ret_size)
```

In/Out	Parameter	Description	Returns
In	this_i3cm	Member of the i3c_controller_instance structure. Pointer to the Controller instance.	1: success 0: failure
In	slave_dyn_addr	Member of the i3c_controller_instance structure. Dynamic address of the I3C Target.	
In	buf	Buffer pointer assigned to the CCC Command.	
In	buf_size	Buffer length of write CCC command.	
In	buf2	Additional buffer for read/write operations.	
In	buf_size2	Length of additional address buffer.	
In	rnw	Read or Write Flag for the CCC Command. Assign *1* to read from *Rx FIFO*, and *0* to write to *Tx FIFO*.	
Out	ret_buf	Buffer to receive any returned data.	
Out	ret_size	Size of returned data.	

2.9. i3c_controller_hdr_ddr_write()

This API writes data in HDR mode. It configures and enters HDR mode, issues the HDR command followed by HDR data words and an HDR CRC word (automatically added by the I3C Controller IP), and applies start and stop conditions for HDR mode.

```
uint8_t i3c_controller_private_i3c_hdr_write(struct i3c_controller_instance *this_i3cm,
uint8_t *wr_buf, uint32_t wr_len, uint8_t slave_dyn_addr)
```

In/Out	Parameter	Description	Returns
In	this_i3cm	Member of the i3c_controller_instance structure. Pointer to the Controller instance.	1: success 0: failure
In	wr_buf	Member of the i3c_controller_instance structure. Buffer pointer where data is assigned in HDR mode.	
In	wr_len	Member of the i3c_controller_instance structure. Buffer length of write data in HDR mode.	
In	slave_dyn_addr	Member of the i3c_controller_instance structure. Dynamic address of the I3C Target.	

2.10. i3c_controller_hdr_ddr_read()

This API reads data in HDR mode. It configures and enters HDR mode, issues the HDR command followed by HDR data words and an HDR CRC word (automatically added by the I3C Controller IP), and applies start and stop conditions for HDR mode.

```
uint8_t i3c_controller_private_i3c_hdr_read(struct i3c_controller_instance *this_i3cm,
uint8_t *rd_buf, uint32_t rd_len, uint8_t slave_dyn_addr)
```

In/Out	Parameter	Description	Returns
In	this_i3cm	Member of the i3c_controller_instance structure. Pointer to the Controller instance.	1: success 0: failure
Out	rd_buf	Member of the i3c_controller_instance structure. Buffer pointer where data is collected in HDR mode.	
In	rd_len	Member of the i3c_controller_instance structure. Buffer length of read data in HDR mode.	
In	slave_dyn_addr	Member of the i3c_controller_instance structure. Dynamic address of the I3C Target.	

2.11. i3c_controller_ibi_init()

This API enables in-band interrupt (IBI) requests from the Target. It configures the Controller to pause SCL during the IBI ACK/NACK phase, allowing firmware to inspect the requesting address. The function programs the IBI read count and writes the response before the bus proceeds.

```
uint8_t i3c_controller_ibi_init(struct i3c_controller_instance *this_i3cm)
```

In/Out	Parameter	Description	Returns
In	this_i3cm	Member of the i3c_controller_instance structure. Pointer to the Controller instance.	1: success 0: failure
In	ibi_config	Set ibi_config to *1* to assign IBI auto response low.	

2.12. i3c_controller_ibi_hot_join_handler()

This API enables Hot-Join requests from the Target. It checks the status of the Tx FIFO, IBI read count, and IBI response registers, provides an ACK or NACK to the Controller, and reads mandatory and optional bytes and stores them into the Rx FIFO.

```
uint8_t i3c_controller_ibi_hot_join_handler(struct i3c_controller_instance *this_i3cm,
uint8_t *ibi_payload_data, uint32_t *rd_len, uint32_t ibi_payload_len)
```

In/Out	Parameter	Description	Returns
In	this_i3cm	Member of the i3c_controller_instance structure. Pointer to the Controller instance.	1: success 0: failure
Out	ibi_payload_data	Member of the i3c_controller_instance structure. Assigns the IBI payload data.	
Out	rd_len	Member of the i3c_controller_instance structure. Actual read count of the IBI payload.	
In	ibi_payload_len	Member of the i3c_controller_instance structure. Assigns the length of the IBI payload.	

2.13. i3c_controller_secondary_handoff()

This API enables transfer of the active Controller role to another Controller. This function applies only when the Secondary Controller Capability is enabled.

```
uint8_t  i3c_controller_secondary_handoff(i3c_controller_instance *this_i3cm)
```

In/Out	Parameter	Description	Returns
In	this_i3cm	Member of the i3c_controller_instance structure. Pointer to the Controller instance.	1: success 0: failure

2.14. i3c_controller_hdr_broadcast_ccc()

This API checks whether the active I3C Controller sends Broadcast CCC commands to all Target addresses in HDR mode.

```
uint8_t i3c_controller_hdr_broadcast_ccc(struct i3c_controller_instance *this_i3cm,
uint8_t *buf, uint32_t size)
```

In/Out	Parameter	Description	Returns
In	this_i3cm	Member of the i3c_controller_instance structure. Pointer to the Controller instance.	1: success 0: failure
In	buf	Member of the i3c_controller_instance structure. Buffer pointer where data is assigned in HDR mode.	
In	size	Member of the i3c_controller_instance structure. Buffer length of write data in HDR mode.	

2.15. i3c_controller_hdr_direct_ccc()

This API checks whether the active I3C Controller sends Direct CCC commands to individual Target addresses in HDR mode.

```
uint8_t i3c_controller_hdr_direct_ccc(struct i3c_controller_instance *this_i3cm, uint8_t
slave_dyn_addr, uint8_t *buf, uint32_t buf_size, uint8_t *ret_buf, uint32_t ret_size,
uint8_t *buf2, uint32_t buf_size2, uint8_t rnw)
```

In/Out	Parameter	Description	Returns
In	this_i3cm	Member of the i3c_controller_instance structure. Pointer to the Controller instance.	1: success 0: failure
In	slave_dyn_addr	Member of the i3c_controller_instance structure. Dynamic address of the I3C Target.	
In	buf	Buffer pointer assigned to the CCC Command.	
In	buf_size	Buffer length of write CCC command.	
In/Out	ret	Return buffer pointer for read operations.	
In	ret_size	Length of the return buffer.	

In/Out	Parameter	Description	Returns
In/Out	buf2	Additional buffer for read/write operations.	
In	buf_size2	Length of additional address buffer.	
In	rnw	Read or Write Flag for the CCC Command. Assign *1* to read from *Rx FIFO*, and *0* to write to *Tx FIFO*.	

2.16. i3c_controller_private_i3c_read_handle_early_term()

This API supports I3C read operations with early Target termination.

```
uint8_t i3c_controller_private_i3c_read_handle_early_term(struct i3c_controller_instance *this_i3cm, uint8_t slave_dyn_addr, uint8_t *rd_buf, uint32_t expected_rd_len, uint32_t *actual_rd_len)
```

In/Out	Parameter	Description	Returns
In	this_i3cm	Member of the i3c_controller_instance structure. Pointer to the Controller instance.	1: success 0: failure
In	slave_dyn_addr	Member of the i3c_controller_instance structure. Dynamic address of the I3C Target.	
Out	rd_buf	Read buffer.	
In	expected_rd_len	Expected read length of the buffer.	
In/Out	actual_rd_len	Actual read length of the buffer.	

2.17. i3c_controller_private_i3c_write_repeated_start_read_handle_early_term()

This API supports I3C write operations with repeated-start read, including handling of early Target read termination.

```
uint8_t i3c_controller_private_i3c_write_repeated_start_read_handle_early_term(struct i3c_controller_instance *this_i3cm, uint8_t slave_dyn_addr, uint8_t *wr_buf, uint32_t wr_len, uint8_t *rd_buf, uint32_t expected_rd_len, uint32_t *actual_rd_len)
```

In/Out	Parameter	Description	Returns
In	this_i3cm	Member of the i3c_controller_instance structure. Pointer to the Controller instance.	1: success 0: failure
In	slave_dyn_addr	Member of the i3c_controller_instance structure. Dynamic address of the I3C Target.	
In	wr_buf	Write buffer.	
In	wr_len	Write buffer length.	
Out	rd_buf	Read buffer.	
In	expected_rd_len	Expected read length of the buffer.	
In/Out	actual_rd_len	Actual read length of the buffer.	

3. Function Call Flow Diagrams

3.1. i3c_controller_init()

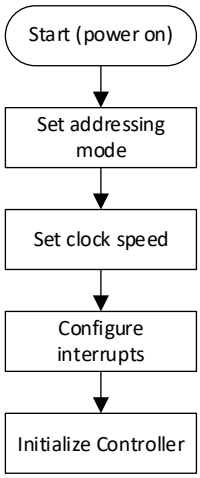

Figure 3.1. i3c_controller_init()

3.2. i3c_controller_daa()

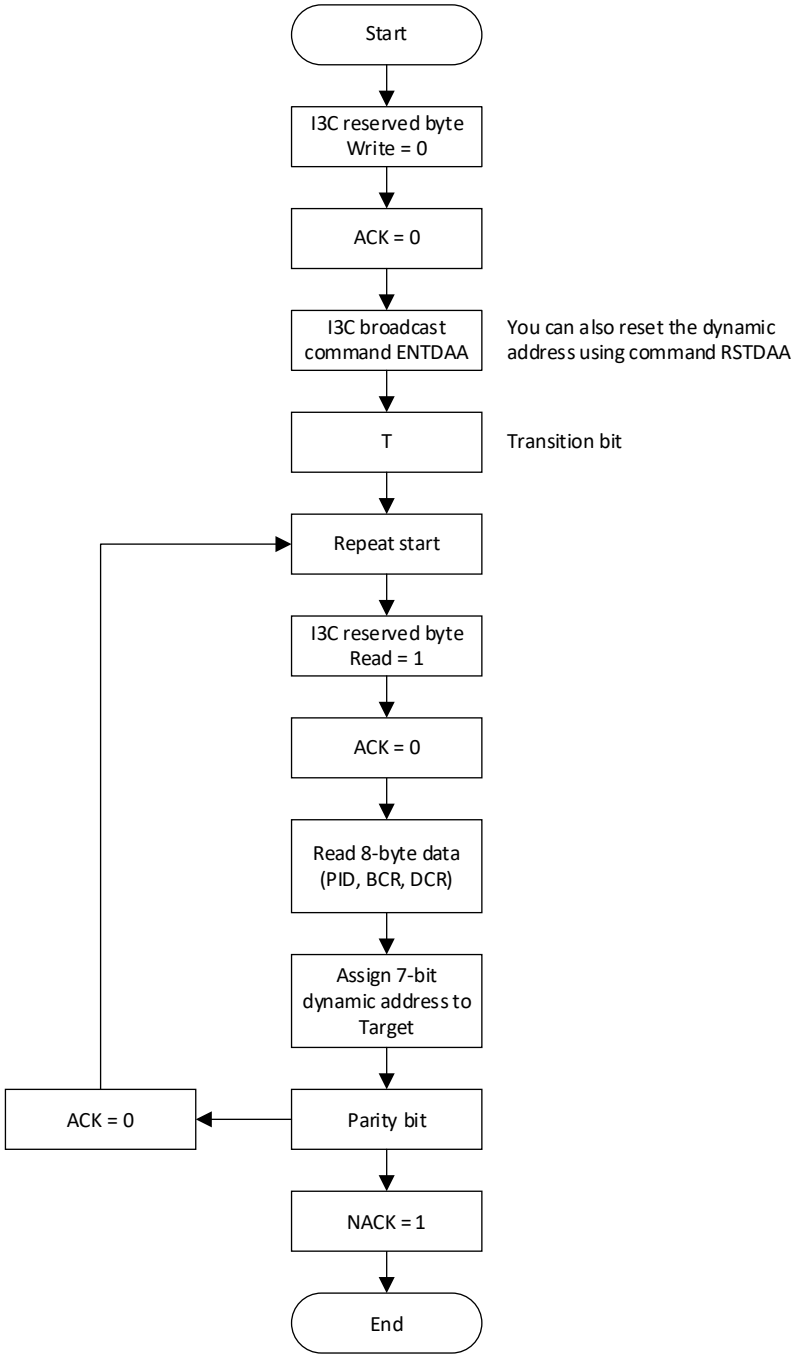

Figure 3.2. i3c_controller_daa()

3.3. i3c_controller_private_i3c_write()

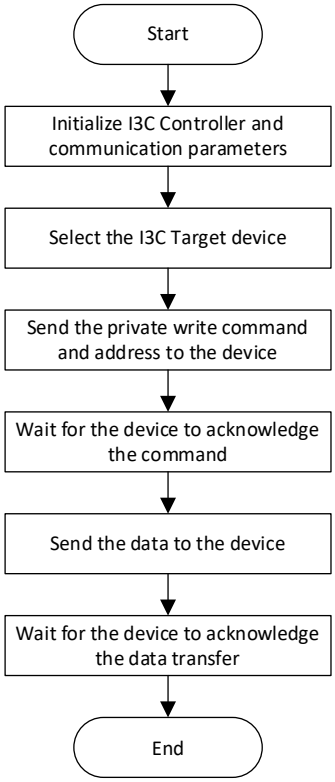

Figure 3.3. i3c_controller_private_i3c_write()

3.4. i3c_controller_private_i3c_read()

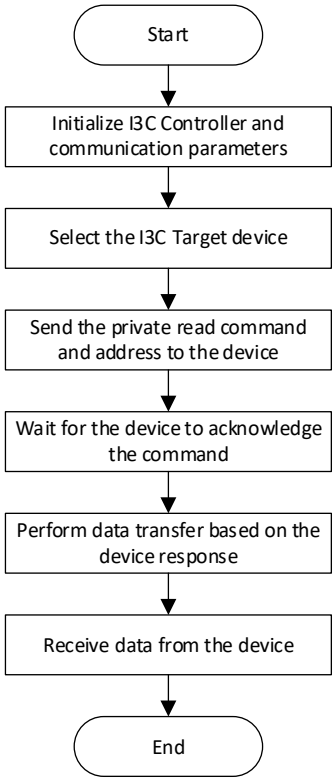

Figure 3.4. i3c_controller_private_i3c_read()

3.5. i3c_controller_private_i2c_write()

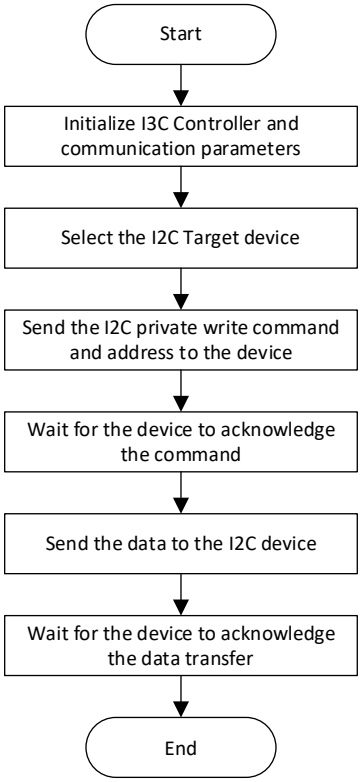

Figure 3.5. i3c_controller_private_i2c_write()

3.6. i3c_controller_private_i2c_read()

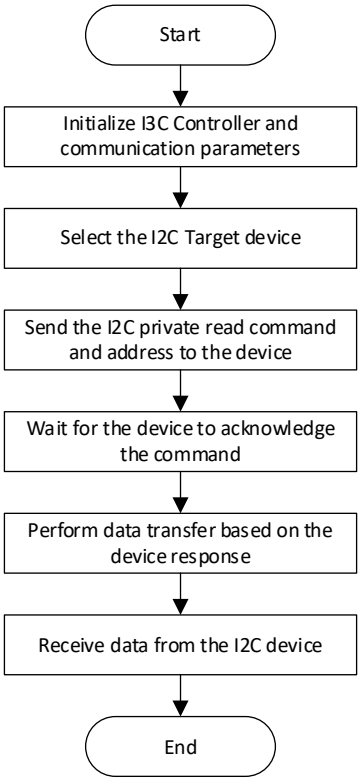

Figure 3.6. i3c_controller_private_i2c_read()

3.7. i3c_controller_broadcast_ccc()

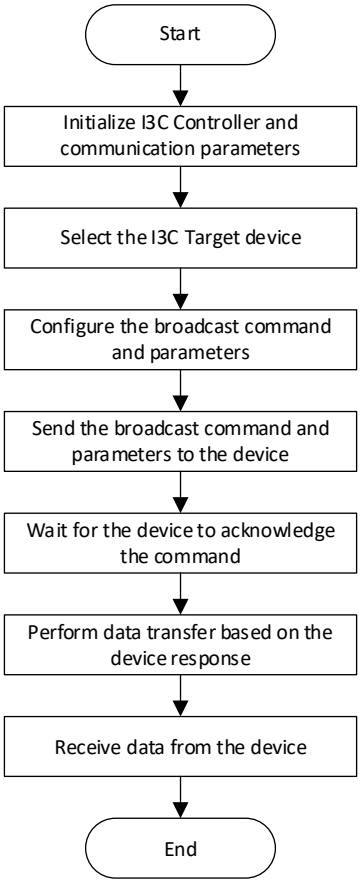

Figure 3.7. i3c_controller_broadcast_ccc()

3.8. i3c_controller_direct_ccc()

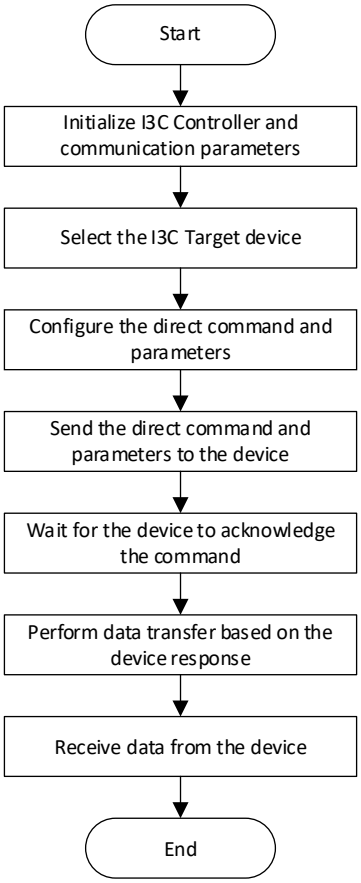

Figure 3.8. i3c_controller_direct_ccc()

3.9. i3c_controller_hdr_ddr_write()

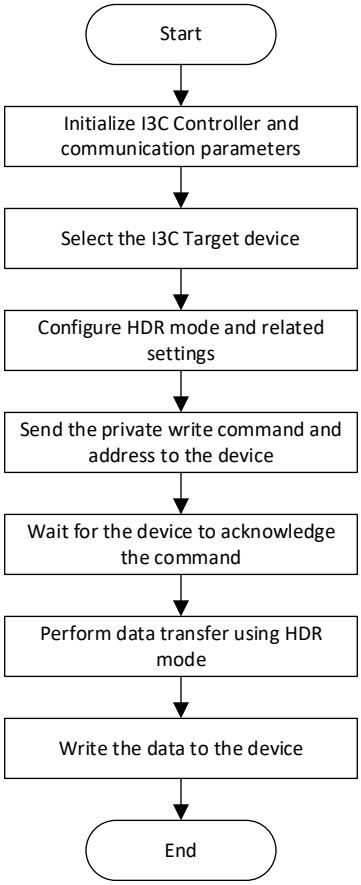

Figure 3.9. i3c_controller_hdr_ddr_write()

3.10. i3c_controller_hdr_ddr_read()

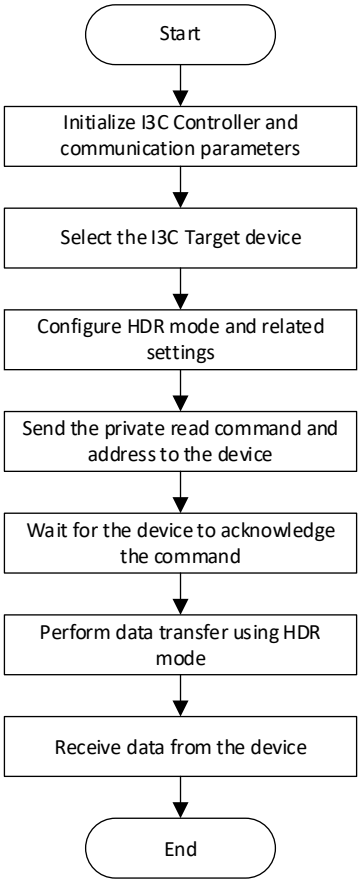

Figure 3.10. i3c_controller_hdr_ddr_read()

3.11. i3c_controller_ibi_init()

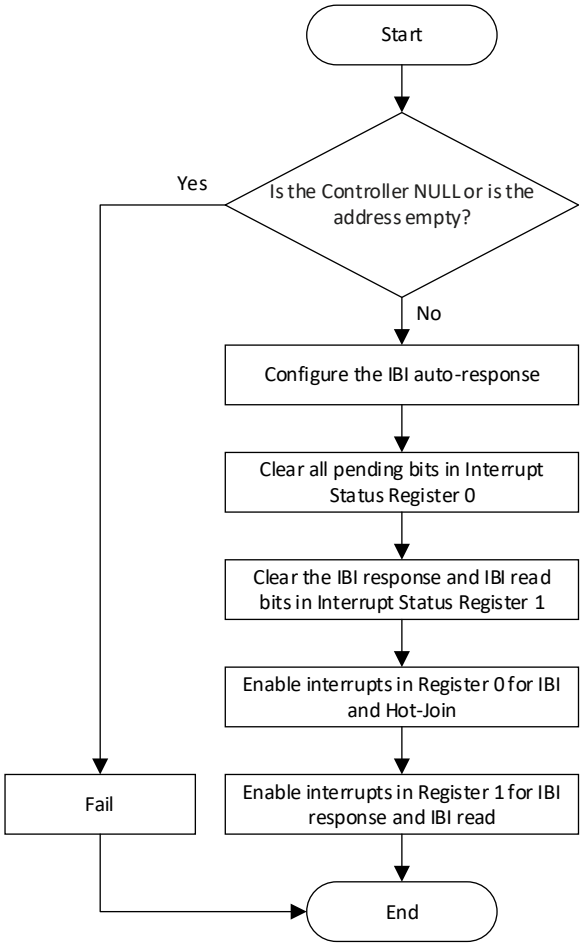

Figure 3.11. i3c_controller_ibi_init()

3.12. i3c_controller_ibi_hotjoin_handler()

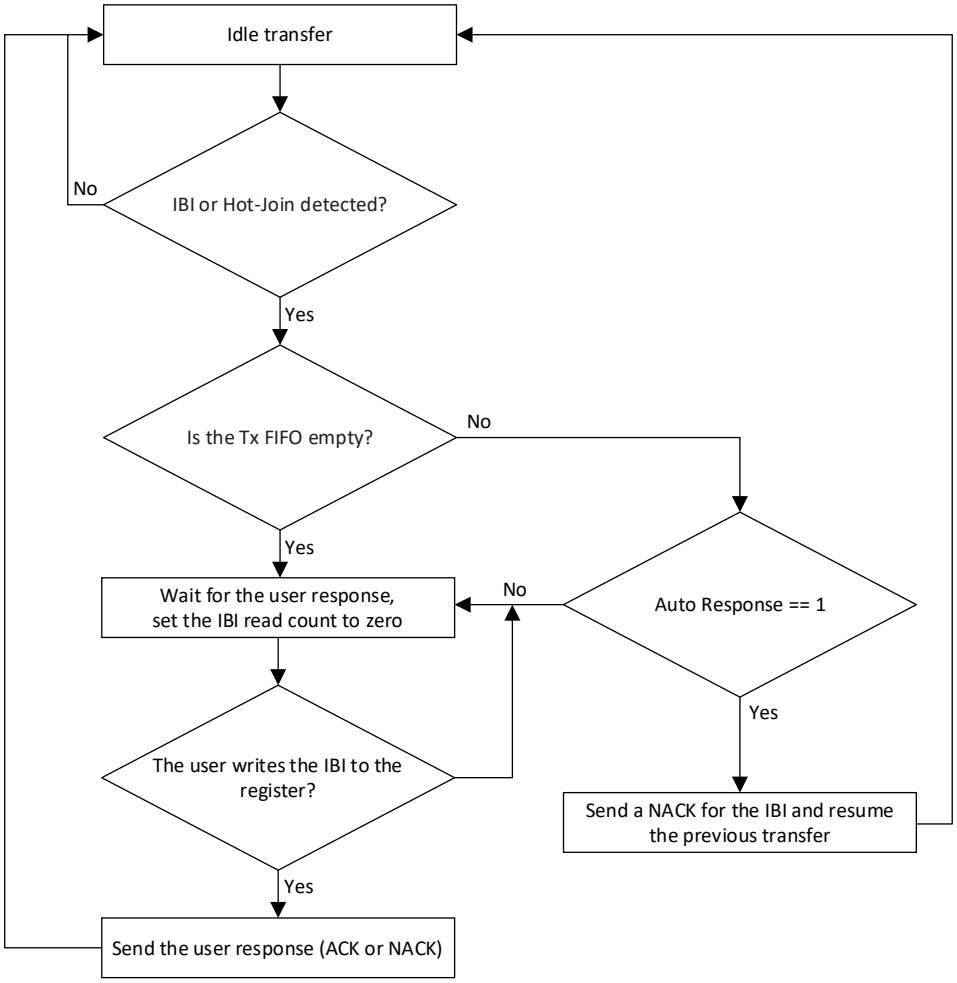

Figure 3.12. i3c_controller_ ibi_hotjoin_handler()

3.13. i3c_controller_secondary_handoff()

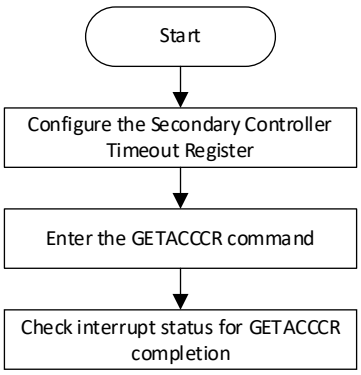

Figure 3.13. i3c_controller_secondary_handoff()

3.14. i3c_controller_hdr_broadcast_ccc()

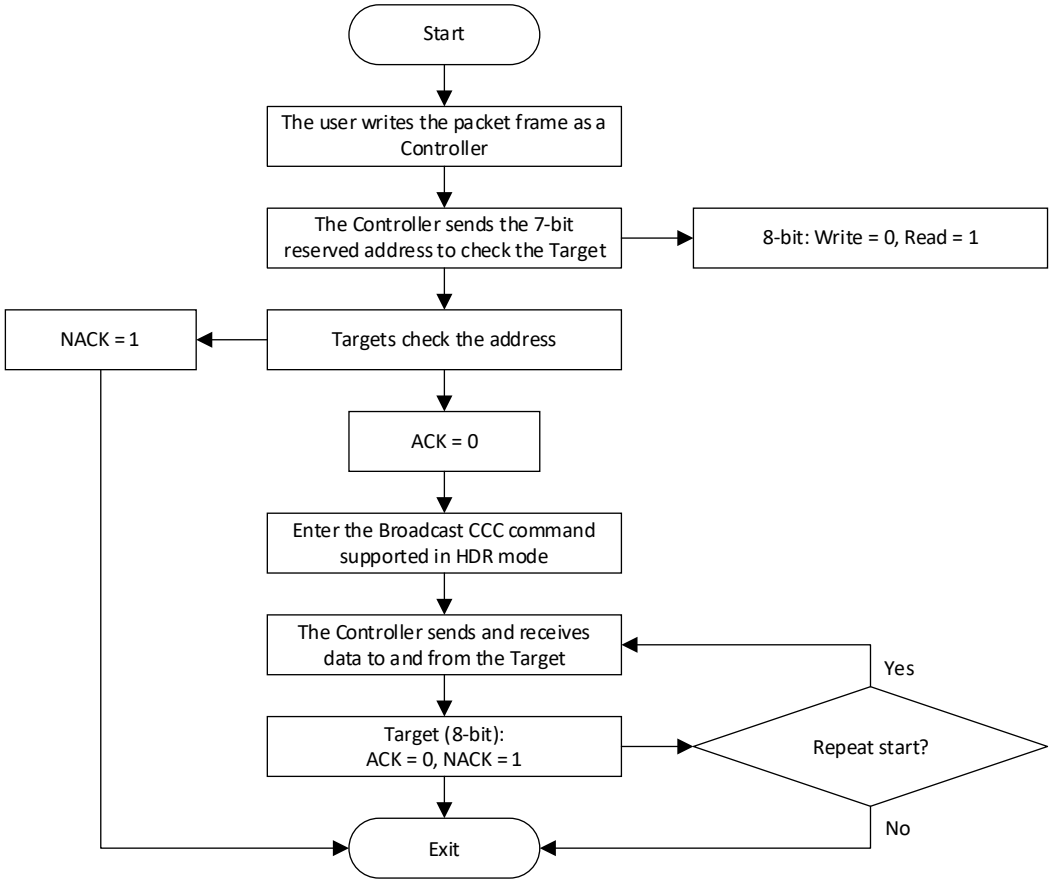

Figure 3.14. i3c_controller_hdr_broadcast_ccc()

3.15. i3c_controller_hdr_direct_ccc()

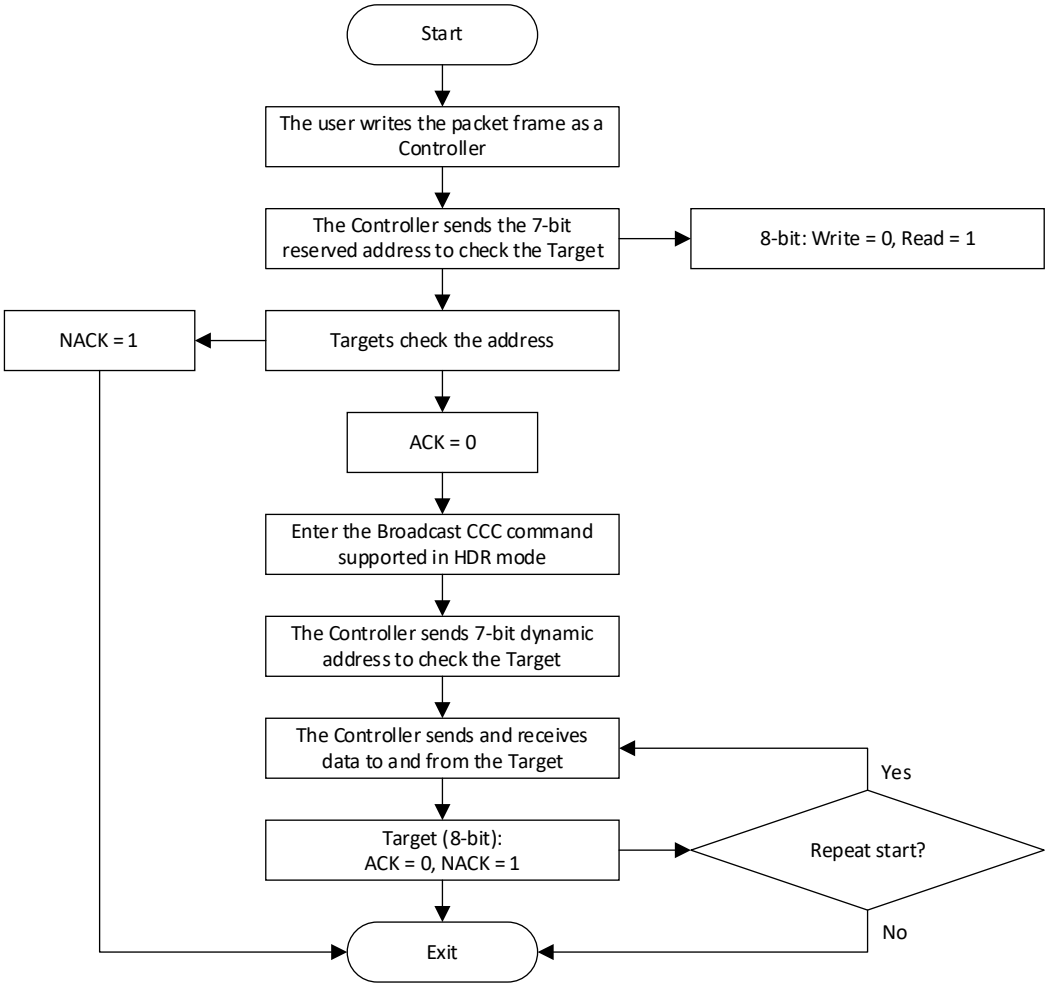

Figure 3.15. i3c_controller_hdr_direct_ccc()

3.16. i3c_controller_private_i3c_read_handle_early_term()

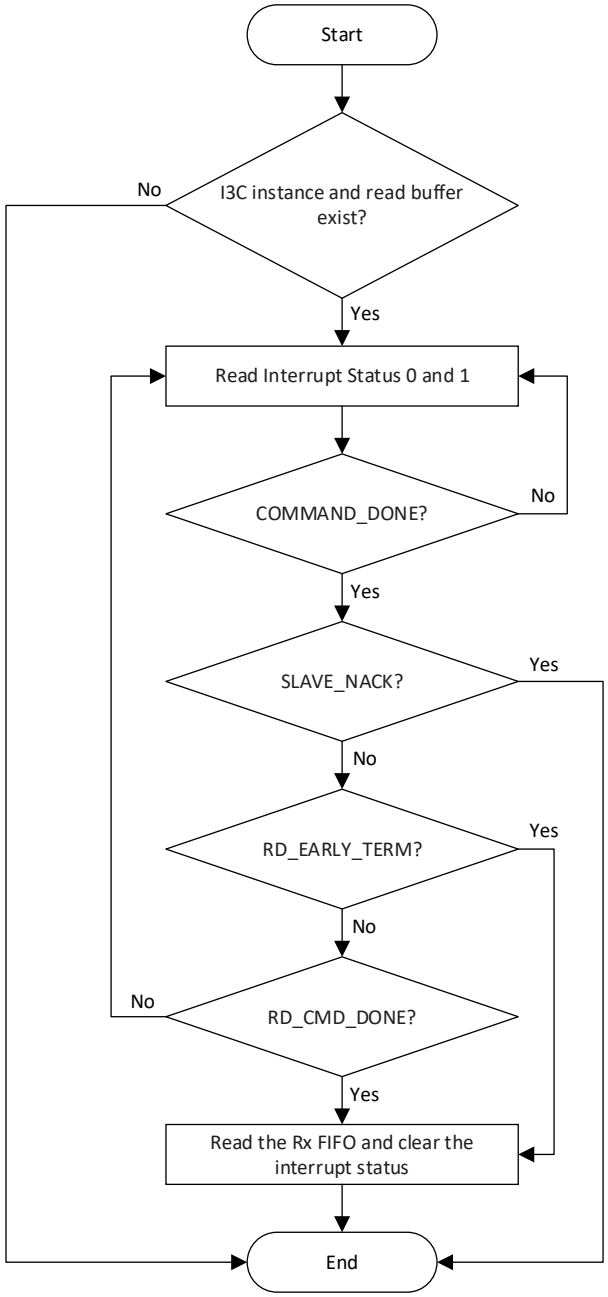

Figure 3.16. i3c_controller_private_i3c_read_handle_early_term()

3.17. i3c_controller_private_i3c_write_repeated_start_read_handle_early_term()

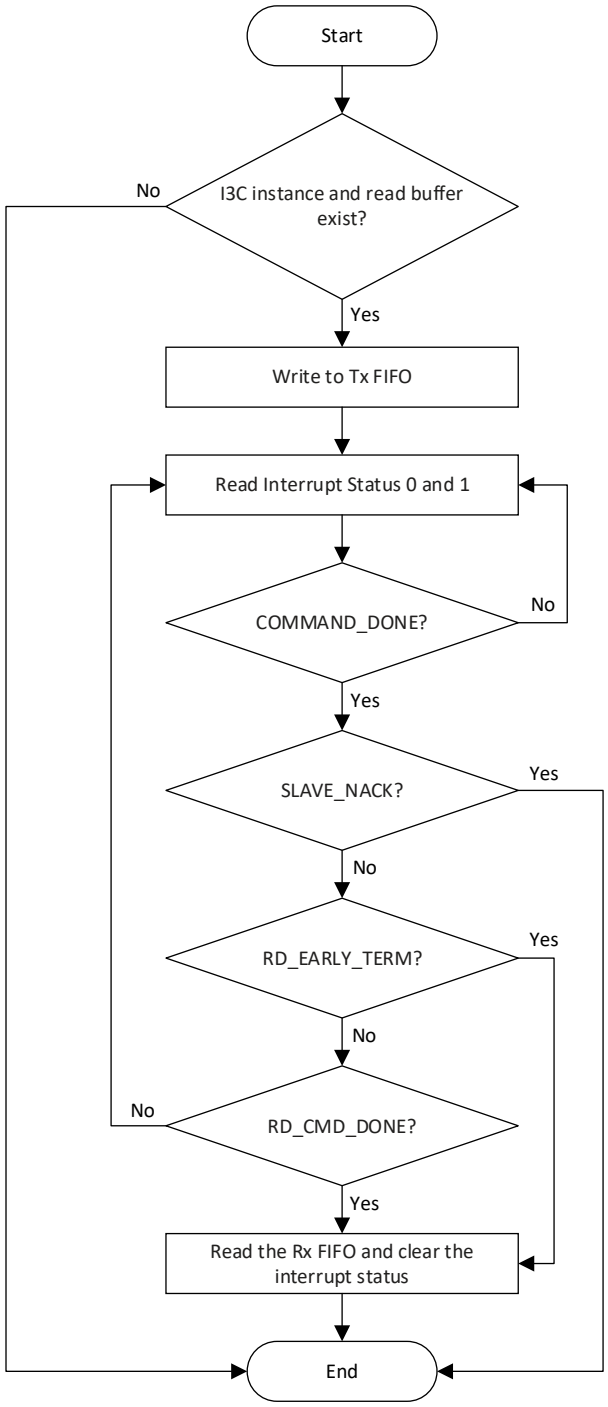

Figure 3.17. i3c_controller_private_i3c_write_repeated_start_read_handle_early_term()

4. API Data Structures

This section describes the structures used in the API.

4.1. i3c_controller_instance

This structure assigns offsets for the I3C Controller registers. The register offsets defined this structure are 32-bit and are used on the APB interface. The following table describes the available parameters.

```
uint32_t base_addr;
uint32_t status;
uint16_t devs_nums;
uint8_t  trans_mode;
uint8_t  *tx_buf;
uint8_t  *rx_buf;
uint8_t  addrs[MAX_DEVS];
uint8_t user_7bit_code;
```

Table 4.1. i3c_controller_instance Parameters

Parameter	Description
base_addr	Stores the I3C Controller base address.
status	Stores the state for the I3C Controller.
devs_nums	Stores the number of discovered I3C Target devices.
trans_mode	Specifies the I3C transfer mode.
tx_buf	Points to the transmit buffer that stores Tx data.
rx_buf	Points to the receive buffer that stores Rx data.
addrs[MAX_DEVS]	Stores the dynamic address of all Target devices.
user_7bit_code	Stores the 7-bit user command code.

4.2. i3c_dev_info_t

This structure stores the PID, DCR, and BCR of a Target device assigned during the dynamic address assignment. The following table describes the available parameters.

```
uint8_t init_dyn_addr;
uint8_t static_addr;
uint8_t bcr;
uint8_t dcr;
uint64_t pid;
```

Table 4.2. i3c_dev_info_t Parameters

Parameter	Description
init_dyn_addr	Stores the dynamic address of each Target device.
static_addr	Stores the static address of each Target device.
pid	Stores the provisional ID of each Target device.
dcr	Stores the device-characteristics register of each Target device.
bcr	Stores the bus-characteristics register of each Target device.

5. API Variables

The following table shows the variables and their descriptions.

Table 5.1. Variable Description

Variable	Description
handle	A structure variable that consists of base address, buffer, and length.
control	Gives packet frame values to APIs.
status	Gives return types for all APIs.
index	The increment values in for loop.

6. API Macros

6.1. Reg 32-bit and 8-bit

Only one option can be set by the user at a time.

```
#define REG_32_BIT                      1
#define REG_8_BIT                       0
```

6.2. Set and Clear

```
#define SET                             0x01
#define CLEAR                           0x00
#define ALL_BITS_EN                     0xff
```

6.3. Success and Failure

```
#define FAILURE                         0
#define SUCCESS                         1
```

6.4. Write and Read

```
#define WRITE                           0
#define READ                            1
```

6.5. User Packet Frame

```
#define CCC_IN_FRAME                    0x01
#define REPEATED_START_IN_FRAME         0x02
#define STOP_IN_FRAME                   0x04
#define START_OF_CCC                    0x08
#define I2C_COMMAND                     0x10
#define WAIT_NEXT_PACKET                0x20
#define HDR_MODE                        0x80
```

6.6. CCC Commands

6.6.1. Broadcast CCC

```
#define ENEC_B                          0x00        //Broadcast R
#define DISEC_B                         0x01        //Broadcast R
#define ENTAS0_B                        0x02        //Broadcast C
#define ENTAS1_B                        0x03        //Broadcast O
#define ENTAS2_B                        0x04        //Broadcast O
#define ENTAS3_B                        0x05        //Broadcast O
#define RSTDAA                          0x06        //Broadcast R
#define ENTDAA                          0x07        //Broadcast R
#define DEFTGTS                         0x08        //Broadcast C
#define SETMWL_B                        0x09        //Broadcast R
#define SETMRL_B                        0x0A        //Broadcast R
#define ENDXFER_B                       0x12        //Broadcast C
#define ENTHDR0                         0x20        //Broadcast C
#define SETXTIME                        0x28        //Broadcast C
```

```
#define SETAASA                                 0x29            //Broadcast O
#define RSTACT                                  0x2A            //Broadcast R
```

6.6.2. Direct CCC

```
#define ENEC_D                                  0x80            //Direct R
#define DISEC_D                                 0x81            //Direct R
#define ENTAS0_D                                0x82            //Direct C
#define ENTAS1_D                                0x83            //Direct O
#define ENTAS2_D                                0x84            //Direct O
#define ENTAS3_D                                0x85            //Direct O
#define SETDASA                                 0x87            //Direct O
#define SETNEWDA                                0x88            //Direct C
#define SETMWL_D                                0x89            //Direct R
#define SETMRL_D                                0x8A            //Direct R
#define GETMWL                                  0x8B            //Direct R
#define GETMRL                                  0x8C            //Direct R
#define GETPID                                  0x8D            //Direct C
#define GETBCR                                  0x8E            //Direct C
#define GETDCR                                  0x8F            //Direct C
#define GETSTATUS                               0x90            //Direct R
#define GETACCCR                                0x91            //Direct C
#define ENDXFER_D                               0x92            //Direct C
```

6.7. Registers

```
#define I3C_MASTER_CLKPERIOD                    0x01*4
#define I3C_MASTER_CFG0                         0x02*4
#define I3C_MASTER_OPEN_DRAIN_TIMER             0x03*4
#define I2C_CLK_DIVIDER                         0x04*4
#define I3C_MASTER_PULL_UP_RESISTOR_DISABLE     0x05*4
#define I3C_MASTER_HDR_DDR_CFG0                 0x06*4
#define I3C_MASTER_SOFT_RESET                   0x08*4
#define I3C_MASTER_START_REG                    0x11*4
#define I3C_MASTER_DA_ACK_REG                   0x1C*4
#define I3C_MASTER_IBI_RD_CNT                   0x1D*4
#define I3C_MASTER_IBI_RESP                     0x1E*4
#define I3C_MASTER_IBI_ADDR                     0x1F*4
#define I3C_MASTER_INTR_STA                     0x20*4
#define I3C_MASTER_INTR_SET                     0x21*4
#define I3C_MASTER_INTR_EN                      0x22*4
#define I3C_MASTER_INTR_STA1                    0x24*4
#define I3C_MASTER_INTR1_SET                    0x25*4
#define I3C_MASTER_INTR_EN1                     0x26*4
#define I3C_MASTER_BUS_COND_DEBUG               0x28*4
#define I3C_MASTER_LAST_NAK                     0x29*4
#define I3C_MASTER_LAST_ACK                     0x2A*4
#define I3C_MASTER_INTR_STA2                    0x2C*4
#define I3C_MASTER_INTR_EN2                     0x2E*4
#define I3C_MASTER_WRITE_FIFO                   0x30*4
#define I3C_MASTER_READ_FIFO                    0x40*4
```

6.8. Configuration Register 0 Settings

```
#define IBI_AUTO_RESP                   0x20
#define IGNORE_CMD_DONE                 0x10
#define I3C_MODE_ALLOWED                0x20
#define I2C_MODE_ALLOWED                0x08
#define IGNORE_RCVD_NAK                 0x04
#define EN_DAA_UID_IN_RXFIFO            0x02
#define I3C_PRIV_RW_NO_7E               0x01
```

6.9. Secondary Controller Handoff

```
#define CRH_TIMEOUT_VALUE               0x009C4
#define CRH_TIMEOUT_VALUE_REG17         0x000ff
#define CRH_TIMEOUT_VALUE_REG18         0x0ff00
#define CRH_TIMEOUT_VALUE_REG19         0xf0000
#define GET_ACCR_DONE                   0x08
#define GET_ACCR_RESULT                 0x04
#define SEC_CTRRL_INTRPT_RCVD           0X20
```

6.10. Interrupts

```
/* int status0 */
#define I3C_INT0_SLAVE_NACK             0x80
#define I3C_INT0_COMMAND_DONE           0x40
#define I3C_INT0_RCVD_IBI               0x20
#define I3C_INT0_RCVD_SEC_IBI           0x10
#define I3C_INT0_RCVD_HOT_JOIN          0x08
#define I3C_INT0_W_FIFO_FULL            0x04
#define I3C_INT0_R_FIFO_N_EMPTY         0x02
#define I3C_INT0_RD_CMD_DONE            0x01

/* int status1 */
#define I3C_INT1_RESERVED               0x80
#define I3C_INT1_WAITTING_IBI_RESP      0x40
#define I3C_INT1_RX_FIFO_FULL           0x20
#define I3C_INT1_CRH_TIMEOUT_EXPIRED    0x10
#define I3C_INT1_GET_ACCCR_DONE         0x08
#define I3C_INT1_IBI_RD_DONE            0x04
#define I3C_INT1_WR_EARLY_TERM          0x02
#define I3C_INT1_RD_EARLY_TERM          0x01

/* int status2 */
#define HDR_RD_CMD_DONE                 0x01
#define HDR_SLAVE_NACK                  0x80
#define HDR_CMND_DONE                   0x40

/* int enable 0 */
#define RCVD_IBI_EN                     0x10
#define RCVD_HOT_JOIN_EN                0x08
```

6.11. Reset Fields

```
#define IP_CSR_RST                      0x10
#define IP_CORE_RST                     0x08
#define TX_FIFO_RST                     0x04
#define RX_FIFO_RST                     0x02
#define IP_MAIN_RST                     0x01
#define START                           0x01
```

6.12. IBI Settings

```
#define RCVD_IBI                        0x10
#define ACK_IBI                         0x00
#define NACK_IBI                        0x01
#define RCVD_HJ                         0x08
#define ACK_HJ                          0x00
#define NACK_HJ                         0x01
```

6.13. Shift Values

```
#define READ_LEFT_SHIFT                 0x01
#define READ_RIGHT_SHIFT                0x80
```

6.14. Values

```
#define I3C_BROADCAST_ADDR              0x7e
#define I3C_SLAVE_NUMS                  1
#define I3C_SLAVE_ADDRESS               0x09
#define ENABLE_DAA_UID
#define I3C_INT0_DEFAULT_STATUS         0xc7
#define MASK_CLEAR                      0xFF
#define I3C_HOT_JOIN_ADDR               0x02
```

6.15. HDR-DDR Settings

```
#define DDR_IGNORE_RCVD_NACK            0x10
#define DDR_IGNORE_CMD_DONE             0x08
#define NO_CRC_AFTER_TERM               0x04
#define EN_WR_EARLY_TERM                0x02
#define EN_WRCMD_ACKNAK_CAP             0x01
```

6.16. Bus Condition Mode

```
#define I3C_MASTER_BUSCOND_HDR_DDR_MODE         0x80u
```

6.17. I3C Enum Mode

```
enum {
    I3C_MODE = 0,
    I2C_I3C_MODE = 1
};
```

References

- I3C Controller IP Release Notes (FPGA-RN-02017)
- I3C Controller IP User Guide (FPGA-IPUG-02228)
- Avant-E web page
- Avant-G web page
- Avant-X web page
- Certus-NX web page
- CertusPro-NX web page
- CrossLink-NX web page
- MachXO2 web page
- MachXO3 web page
- MachXO3D web page
- Mach-NX web page
- MachXO5-NX web page
- Lattice Solutions IP Cores web page
- Lattice Solutions Reference Designs web page
- Lattice Radiant Software web page
- Lattice Insights web page 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 1.1, June 2026

Section	Change Summary
All	Made editorial fixes.
Abbreviations in This Document	• Updated *acronym* to *abbreviation*. • Added *Rx* and *Tx*. • Removed *AHB* and *DWORD*.
Introduction	Updated the Driver Versioning and Driver and IP Compatibility sections.
API Description	Updated this section.
Function Call Flow Diagrams	Updated this section.
API Data Structures	Updated this section.

API Macros	• Removed the *Hot Join and IBI* section.
	• Added the following sections:
	  • Registers
	  • Configuration Register 0 Settings
	  • Reset Fields
	  • IBI Settings
	  • HDR-DDR Settings
	  • Bus Condition Mode
	  • I3C Enum Mode
	• Updated the following sections:
	  • Reg 32-bit and 8-bit
	  • User Packet Frame
	  • Interrupts
	  • Shift Values
	  • Values

References	Added the *I3C Controller IP Release Notes (FPGA-RN-02017)*, *Lattice Solutions IP Cores* web page, and *Lattice Solutions Reference Designs* web page.

Revision 1.0, December 2023

Section	Change Summary
All	Initial release.

www.latticesemi.com