I3C Target Driver API Reference

Technical Note

FPGA-TN-02338-1.0

December 2023

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 REDUDANCIES, 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

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
ACK	Acknowledgement
AHB	Advanced High-Performance Bus
APB	Advanced Peripheral Bus
API	Application Programming Interfaces
BCR	Bus Characteristic Register
CCC	Common Control Code
DA	Dynamic Address
DAA	Dynamic Address Assignment
DCR	Device Characteristic Register
DDR	Double Data Rate
DWORD	Double Word
FIFO	First In First Out
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
HDR	High Data Rate
HJ	Hot Join
IBI	In Band Interrupt
IP	Intellectual Property
I2C	Inter-Integrated Circuit
I3C	Improved Inter-Integrated Circuit
LMMI	Lattice Memory Mapped Interface
MDB	Mandatory Data Byte
NACK	No Acknowledgement
RISC	Reduced Instruction Set Computer
SA	Static Address
SCL	Serial Clock
SDA	Serial Data
SDK	Software Development Kit
SDR	Single Data Rate
TX	Transmit

1. Introduction

This document provides technical information about the I3C target driver and information for the IP development, verification, testing, validation, and integration software. The IP design is implemented in the Verilog HDL.

1.1. Purpose

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

Driver version: 1.0.0

1.3.2. IP Version

IP version: 3.2.0

1.4. Driver and IP Compatibility

The following IPs are used for the co-verification testing:

Table 1.1. IP and IP's version

IP	IP version
I3C Controller	3.2.0
I3C Target	3.2.0
Osc0	1.4.0
PLL	1.7.0
AHB Lite Interconnect	1.3.0
System Memory	2.0.0
AHB Lite to APB bridge	1.1.0
APB interconnect	1.2.0
RISC-V MC	2.4.0

1.5. System Topology

The I3C protocol operates in a control/target operation model where the data flows between them, and the communication network overall structure. For more details on the I3C Target IP Core, refer to the I3C Target IP Core User Guide (FPGA-IPUG-02227).

1.6. I3C Target Protocol Overview

The I3C protocol aims to address some of the I2C protocol's limitations while maintaining compatibility with the existing I2C devices. The protocol provides a more flexible and efficient communication solution for modern integrated circuits, especially in applications where multiple devices need to communicate on the same bus.

2. API Description

The I3C target driver can communicate with the I3C controller driver. The APIs are listed below:

2.1. i3c_target_init()

This API initializes the target. This function sets up the target configuration and manages the interrupts by enabling and disabling them.

```
unsigned int i3c_target_init(struct i3c_target_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	base_addr	• Member of the i3c_target_handle_t structure. • Base address of the I3C target.	1: success 0: failure

2.2. i3c_target_private_write()

This API is used to write data to the I3C bus. A dynamic address is used here to communicate between the target and the controller.

```
unsigned int i3c_target_private_write(struct i3c_target_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	base_addr	• Member of the i3c_target_handle_t structure. • Base address of the I3C target.	1: success 0: failure
In	wr_buf	• Member of the i3c_target_handle_t structure. • Buffer pointer where data is collected.	
In	wr_length	• Member of the i3c_target_handle_t structure. • Buffer length for write data.	

2.3. i2c_target_write()

This API is used to write data from the I3C bus in the I2C mode. A static address is used here to communicate between the controller and the target.

```
unsigned int i2c_target_write(struct i3c_target_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	base_addr	• Member of the i3c_target_handle_t structure. • Base address of the I3C target.	1: success 0: failure
In	wr_buf	• Member of the i3c_target_handle_t structure. • Buffer pointer where data is collected.	
In	wr_length	• Member of the i3c_target_handle_t structure. • Buffer length for write data.	

2.4. i3c_target_private_read()

This API is used to read data from the I3C bus. A dynamic address is used here to communicate between the target and the controller.

```
unsigned int i3c_target_private_read(struct i3c_target_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	base_addr	• Member of the i3c_target_handle_t structure. • Base address of the I3C target.	1: success 0: failure
In	rd_buf	• Member of the i3c_target_handle_t structure. • Buffer pointer where data is collected.	
In	rd_length	• Member of the i3c_target_handle_t structure. • Buffer length for data read.	

2.5. i2c_target_read()

This API is used to read data from the I3C bus in the I2C mode. A static address is used here to communicate between the controller and the target.

```
unsigned int i2c_target_read(struct i3c_target_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	base_addr	• Member of the i3c_target_handle_t structure. • Base address of the I3C target.	1: success 0: failure
In	rd_buf	• Member of the i3c_target_handle_t structure. • Buffer pointer where data is collected.	
In	rd_length	• Member of the i3c_target_handle_t structure. • Buffer length for data read.	

2.6. i3c_target_hj_req()

This API is used for target integration into the I3C bus through the execution of a *hot_join* request.

```
unsigned int i3c_target_hj_req(struct i3c_target_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	base_addr	• Member of the i3c_target_handle_t structure. • Base address of the I3C target.	1: success 0: failure
In	max_retry	• Member of the i3c_target_handle_t structure. • Number of retry to get an ACK from the controller.	

2.7. i3c_target_ibi_req()

This API is used to indicate the pending actions to the I3C controller via In-Band Interrupts (IBI) request. The target initiates the IBI request. After the IBI request is done, target can send the mandatory and optional data bytes.

```
unsigned int i3c_target_ibi_req(struct i3c_target_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	base_addr	• Member of the i3c_target_handle_t structure. • Base address of the I3C target.	1: success 0: failure
In	mdb	• Member of the i3c_target_handle_t structure. • Mandatory data bytes passed by the user.	

In/Out	Parameter	Description	Returns
In	ibi_optional_payload	• Member of the i3c_target_handle_t structure. • Optional data sent to the bus.	
In	ibi_optional_payload_len	• Member of the i3c_target_handle_t structure. • Length of the optional data byte sent to the bus.	
In	max_retry	• Member of the i3c_target_handle_t structure. • Number of retry to get an ACK from the controller.	

2.8. i3c_target_cntrl_role_handoff()

This API is used to generate a control role request by the target which involves inspecting the enable control register and commencing the initialization process.

```
unsigned int i3c_target_cntrl_role_handoff(struct i3c_target_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	base_addr	• Member of the i3c_target_handle_t structure. • Base address of the I3C target.	1: success 0: failure
In	max_retry	• Member of the i3c_target_handle_t structure. • Number of retry to get an ACK from the controller.	

2.9. i3c_target_hdrddr_write()

This API is used to write in HDR DDR mode by the target.

```
unsigned int i3c_target_hdrddr_write(struct i3c_target_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	base_addr	• Member of the i3c_target_handle_t structure. • Base address of the I3C target.	1: success 0: failure
In	wr_buf	• Member of the i3c_target_handle_t structure. • Buffer pointer where data is collected.	
In	wr_length	• Member of the i3c_target_handle_t structure. • Buffer length for write data.	

2.10. i3c_target_hdrddr_read()

This API is used to read in HDR DDR mode by the target.

```
unsigned int i3c_target_hdrddr_read(struct i3c_target_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	base_addr	• Member of the i3c_target_handle_t structure. • Base address of the I3C target.	1: success 0: failure
In	rd_buf	• Member of the i3c_target_handle_t structure. • Buffer pointer where data is collected.	
In	rd_length	• Member of the i3c_target_handle_t structure. • Buffer length for data read.	

2.11. i3c_target_fifo_loopback_enable()

This API is used to loop back the FIFO register in the I3C target.

```
unsigned int i3c_target_fifo_loopback_enable(struct i3c_target_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	base_addr	• Member of the i3c_target_handle_t structure. • Base address of the I3C target.	1: success 0: failure

2.12. i3c_target_daa_wait()

This API is used to wait for the dynamic address assignment of the I3C target.

```
unsigned int i3c_target_daa_wait(struct i3c_target_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	base_addr	• Member of the i3c_target_handle_t structure. • Base address of the I3C target.	1: success 0: failure

2.13. i3c_target_static_addr_wait()

This API is used to wait for the static address assignment of the I3C target.

```
unsigned int i3c_target_static_addr_wait(struct i3c_target_handle_t *handle)
```

In/Out	Parameter	Description	Returns
In	base_addr	• Member of the i3c_target_handle_t structure. • Base address of the I3C target.	1: success 0: failure

3. Function Call Flow Diagrams

3.1. Initialization

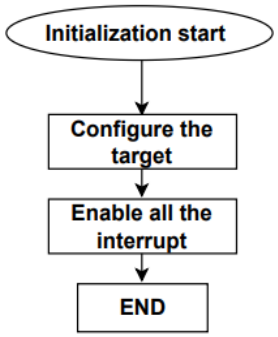

Figure 3.1. Initialization Flow Chart

3.2. I3C Write

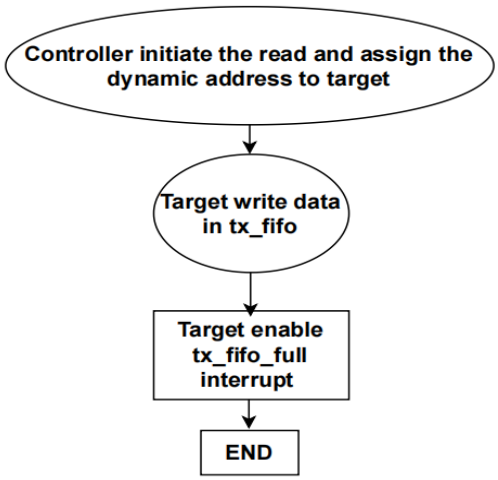

Figure 3.2. I3C Write Flow Chart

3.3. I2C Write

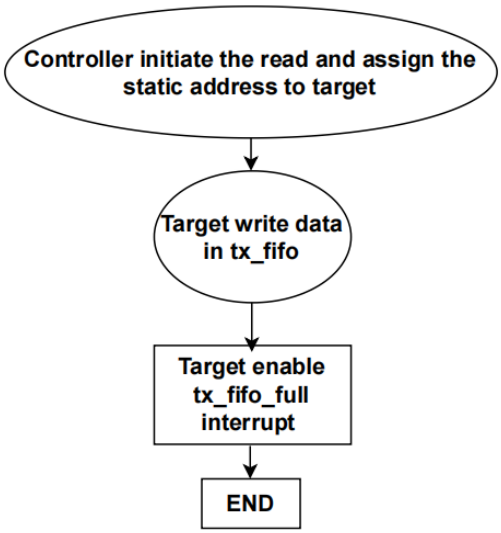

Figure 3.3. I2C Write Flow Chart

3.4. I3C Read

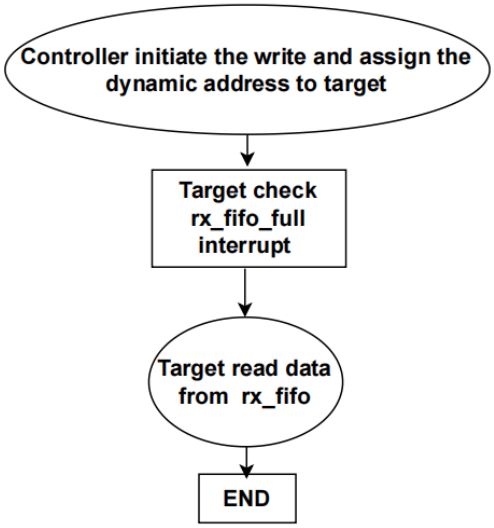

Figure 3.4. I3C Read Flow Chart

3.5. I2C Read

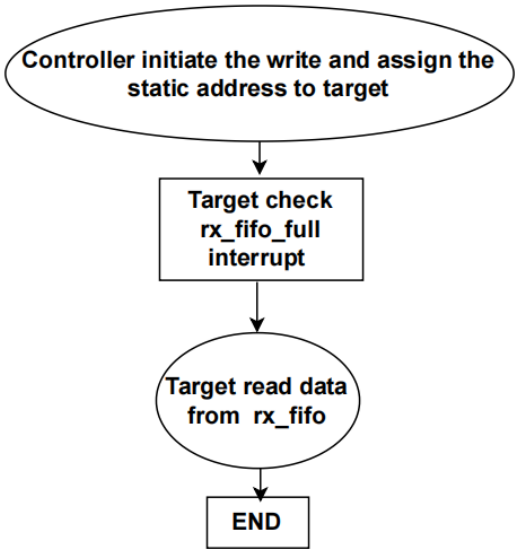

Figure 3.5. I2C Read Flow Chart

3.6. Hot Join

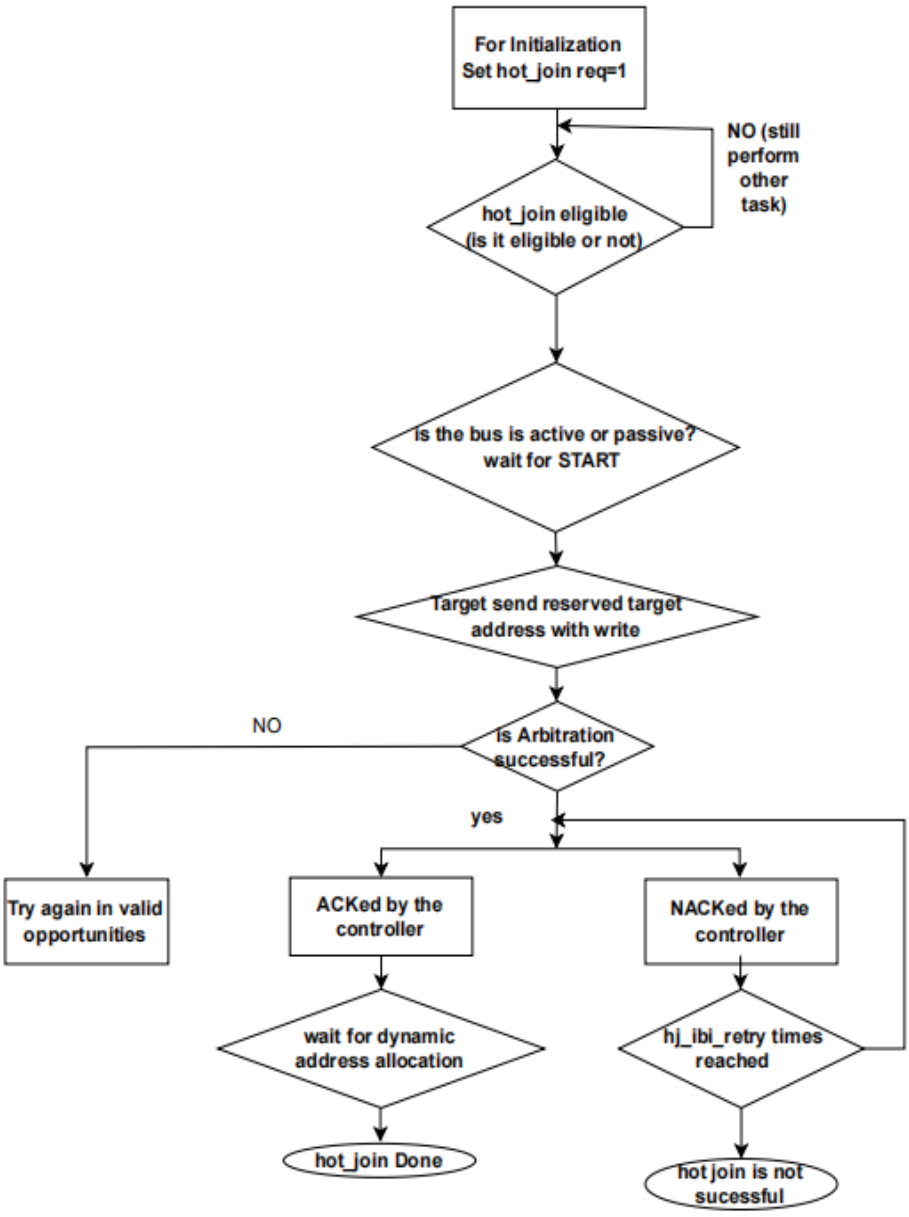

Figure 3.6. Hot Join Flow Chart

3.7. IBI

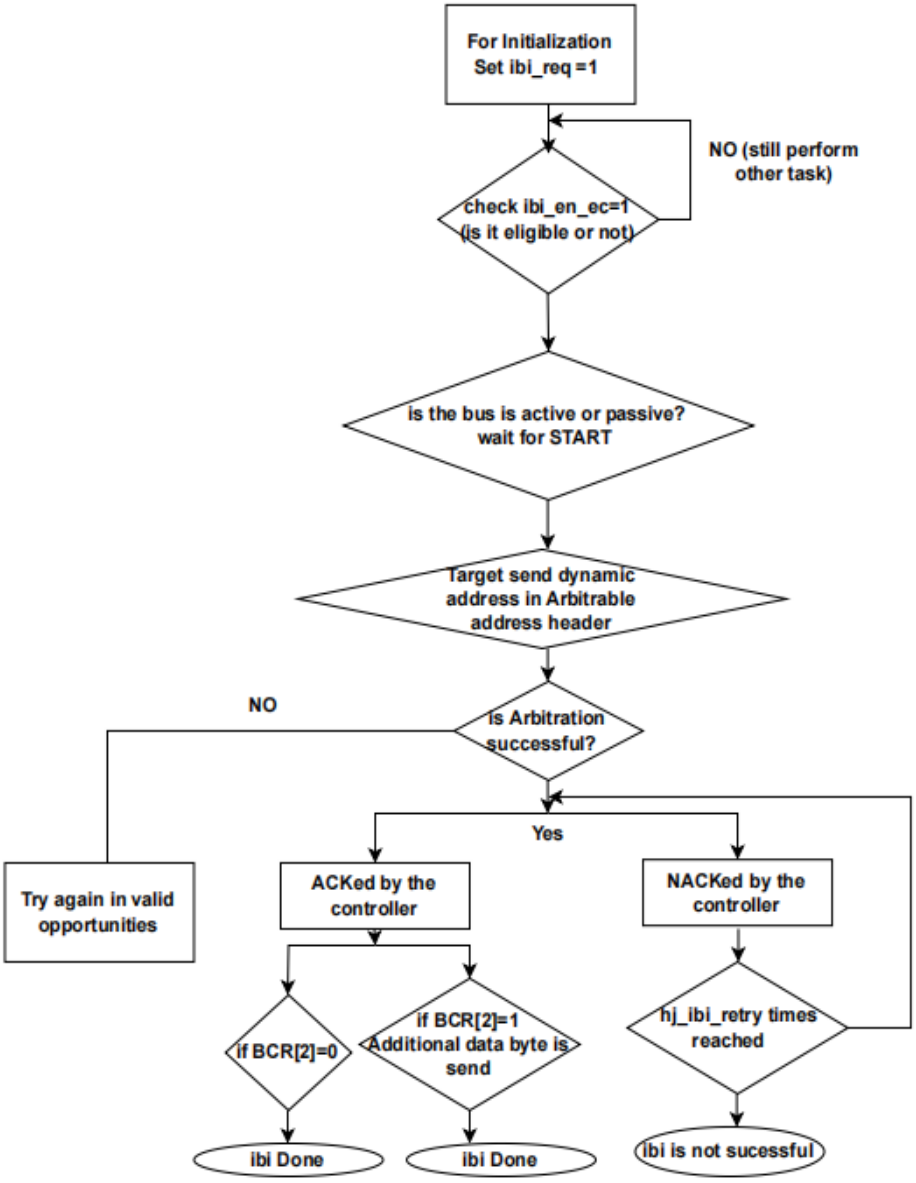

Figure 3.7. IBI Flow Chart

3.8. Secondary Control Connection

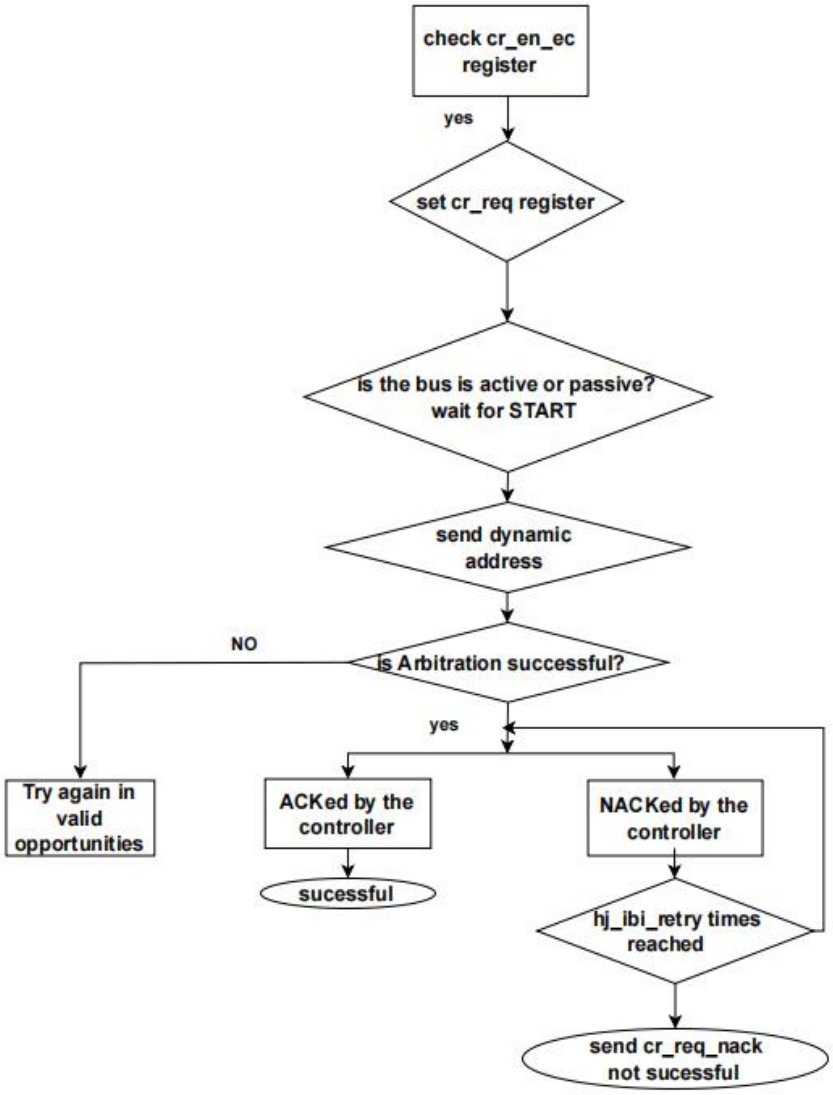

Figure 3.8. Secondary Control Connection Flow Chart

3.9. HDR Write

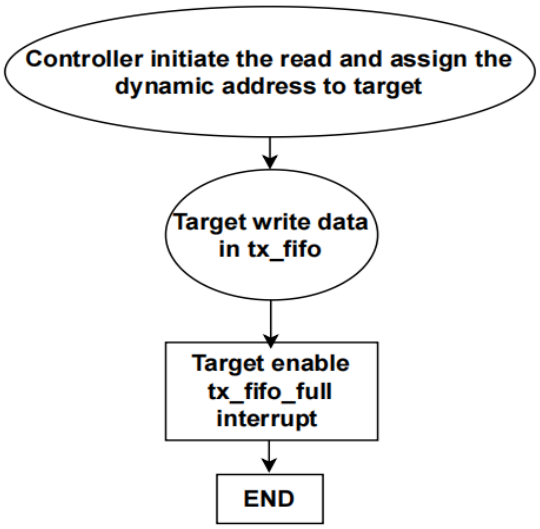

Figure 3.9. HDR Write Flow Chart

3.10. HDR Read

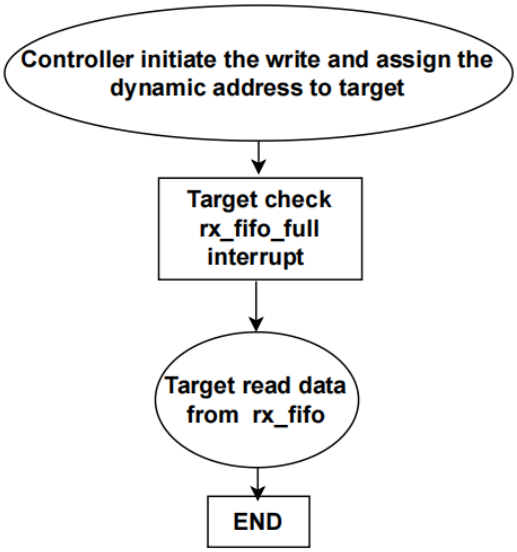

Figure 3.10. HDR Read Flow Chart

3.11. FIFO Loop Back Enable

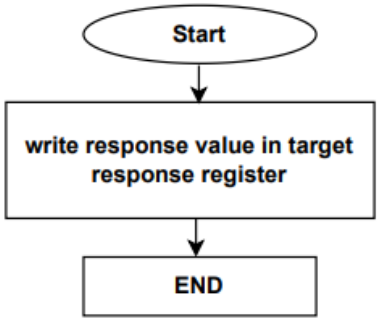

Figure 3.11. FIFO Loop Back Enable Flow Chart

3.12. Dynamic Address

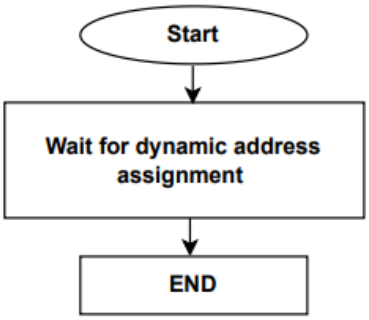

Figure 3.12. Wait for Dynamic Address Flow Chart

3.13. Static Address

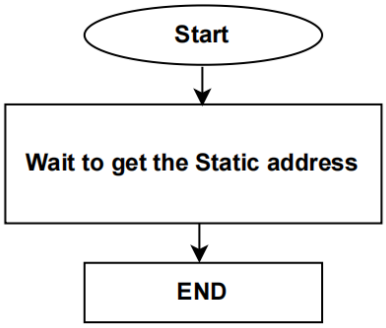

Figure 3.13. Wait for Static Address Flow Chart

4. API Data Structures

This section describes the structures used in the API. The *i3c_tgt_reg_type_t* structure is used for register offset, and *i3c_target_handle_t* structure is used for parameters set by the user.

4.1. i3c_tgt_reg_type_t

This structure is used to assign offset for the I3C target register. 8-bit or 32-bit register can be used for the drivers.

Use REG32_BIT macro for the 32-bit register, and REG8_BIT macro for the 8-bit register. Use the 32-bit register for the DWORD address offset, and 8-bit register for the Byte address offset.

The SECONDARY_CONTROLLER_ENABLE macro is set when target acts as a secondary controller.

4.1.1. DWORD Address Offset

The structure used for the DWORD address offset is given as:

```
#if SECONDARY_CONTROLLER_ENABLE     //Used for secondary controller connection
volatile unsigned int reserved[128];
#endif
volatile unsigned int bcr;                                          //00
volatile unsigned int dcr;
volatile unsigned int dyn_addr;
volatile unsigned int event_command_en;
volatile unsigned int event_command_dev_config;
volatile unsigned int event_command_req;                            //05
volatile unsigned int hj_ibi_retry;
volatile unsigned int max_wr_legth_msb;
volatile unsigned int max_wr_legth_lsb;
volatile unsigned int max_rd_legth_msb;
volatile unsigned int max_rd_legth_lsb;                             //A
volatile unsigned int max_ibi_payload;
volatile unsigned int max_wr_data_speed;
volatile unsigned int max_rd_data_spped;
volatile unsigned int max_rd_turnout_time_msb;
volatile unsigned int max_rd_turnout_time;                          //F
volatile unsigned int max_rd_turnout_time_lsb;
volatile unsigned int dev_provisioned_id6;
volatile unsigned int dev_provisioned_id5;
volatile unsigned int dev_provisioned_id4;                          //13
volatile unsigned int dev_provisioned_id3;
volatile unsigned int dev_provisioned_id2;
volatile unsigned int dev_provisioned_id1;
volatile unsigned int static_addr;                                  //17
volatile unsigned int dev_capabilities_byte1;
volatile unsigned int dev_capabilities_byte2;
volatile unsigned int dev_capabilities_byte3;
volatile unsigned int reserved1;
volatile unsigned int osc_inacc;
volatile unsigned int reserved2;
volatile unsigned int reserved3;
volatile unsigned int reserved4;
volatile unsigned int rx_fifo;                                      //20
volatile unsigned int reserved5;
```

```
volatile unsigned int tx_fifo;                                  //22
volatile unsigned int reserved6;
volatile unsigned int reserved7;
volatile unsigned int reserved8;
volatile unsigned int reserved9;
volatile unsigned int reserved10;
volatile unsigned int soft_reset;
volatile unsigned int target_response;                          //29
volatile unsigned int get_status_msb;
volatile unsigned int get_status_lsb;
volatile unsigned int bus_activity_state;
volatile unsigned int tgt_reaction_action_1;
volatile unsigned int tgt_reaction_action_2;
volatile unsigned int tgt_reaction_action_3;                    //2F
volatile unsigned int interrupt_1;
volatile unsigned int interrupt_1_en;                           //31
volatile unsigned int interrupt_1_set;
volatile unsigned int interrupt_2;
volatile unsigned int interrupt_2_en;                           //34
volatile unsigned int interrupt_2_set;                          //35
volatile unsigned int interrupt_3;                              //36
volatile unsigned int interrupt_3_en;
volatile unsigned int interrupt_3_set;
volatile unsigned int interrupt_4;                              //39
volatile unsigned int interrupt_4_en;
volatile unsigned int interrupt_4_set;
volatile unsigned int interrupt_5;                              //3C
volatile unsigned int interrupt_5_en;
volatile unsigned int interrupt_5_set;                          //3E
volatile unsigned int reserved11;
volatile unsigned int deftgts_count;                            //40
volatile unsigned int deftgtsrxfifo_start;
volatile unsigned int deftgtsrxfifo_count;
volatile unsigned int controller_role_handoff;
volatile unsigned int getmxds_con_cap;
volatile unsigned int getmxds_con_cap_lsb;                      //45
volatile unsigned int getmxds_con_cap1;                         //46
volatile unsigned int getmxds_con_cap2;
volatile unsigned int reserved12;
volatile unsigned int reserved13;
volatile unsigned int reserved14;
volatile unsigned int reserved15;                               //4A
volatile unsigned int reserved16;
volatile unsigned int reserved17;
volatile unsigned int reserved18;
volatile unsigned int reserved19;
volatile unsigned int bus_mode;                                 //50
volatile unsigned int hdr_ddr_target_config;
volatile unsigned int reserved20;
volatile unsigned int reserved21;
volatile unsigned int hdr_ddr_target_abr_config;                //54
```

4.1.2. BYTES Address Offset

The structure used for the BYTES address offset is given as:

```
#if SECONDARY_CONTROLLER_ENABLE     //Used for secondary controller connection
volatile unsigned int reserved[128];
#endif
volatile unsigned char bcr;                                          //00
volatile unsigned char dcr;
volatile unsigned char dyn_addr;
volatile unsigned char event_command_en;
volatile unsigned char event_command_dev_config;
volatile unsigned char event_command_req;                            //05
volatile unsigned char hj_ibi_retry;
volatile unsigned char max_wr_legth_msb;
volatile unsigned char max_wr_legth_lsb;
volatile unsigned char max_rd_legth_msb;
volatile unsigned char max_rd_legth_lsb;                             //A
volatile unsigned char max_ibi_payload;
volatile unsigned char max_wr_data_speed;
volatile unsigned char max_rd_data_spped;
volatile unsigned char max_rd_turnout_time_msb;
volatile unsigned char max_rd_turnout_time;                          //F
volatile unsigned char max_rd_turnout_time_lsb;
volatile unsigned char dev_provisioned_id6;
volatile unsigned char dev_provisioned_id5;
volatile unsigned char dev_provisioned_id4;                          //13
volatile unsigned char dev_provisioned_id3;
volatile unsigned char dev_provisioned_id2;
volatile unsigned char dev_provisioned_id1;
volatile unsigned char static_addr;                                  //17
volatile unsigned char dev_capabilities_byte1;
volatile unsigned char dev_capabilities_byte2;
volatile unsigned char dev_capabilities_byte3;
volatile unsigned char reserved1;
volatile unsigned char osc_inacc;
volatile unsigned char reserved2;
volatile unsigned char reserved3;
volatile unsigned char reserved4;
volatile unsigned char rx_fifo;                                      //20
volatile unsigned char reserved5;
volatile unsigned char tx_fifo;                                      //22
volatile unsigned char reserved6;
volatile unsigned char reserved7;
volatile unsigned char reserved8;
volatile unsigned char reserved9;
volatile unsigned char reserved10;
volatile unsigned char soft_reset;
volatile unsigned char target_response;                              //29
volatile unsigned char get_status_msb;
volatile unsigned char get_status_lsb;
volatile unsigned char bus_activity_state;
volatile unsigned char tgt_reaction_action_1;
```

```
volatile unsigned char tgt_reaction_action_2;
volatile unsigned char tgt_reaction_action_3;                    //2F
volatile unsigned char interrupt_1;
volatile unsigned char interrupt_1_en;                           //31
volatile unsigned char interrupt_1_set;
volatile unsigned char interrupt_2;
volatile unsigned char interrupt_2_en;                           //34
volatile unsigned char interrupt_2_set;                          //35
volatile unsigned char interrupt_3;                              //36
volatile unsigned char interrupt_3_en;
volatile unsigned char interrupt_3_set;
volatile unsigned char interrupt_4;                              //39
volatile unsigned char interrupt_4_en;
volatile unsigned char interrupt_4_set;
volatile unsigned char interrupt_5;                              //3C
volatile unsigned char interrupt_5_en;
volatile unsigned char interrupt_5_set;                          //3E
volatile unsigned char reserved11;
volatile unsigned char deftgts_count;                            //40
volatile unsigned char deftgtsrxfifo_start;
volatile unsigned char deftgtsrxfifo_count;
volatile unsigned char controller_role_handoff;
volatile unsigned char getmxds_con_cap;
volatile unsigned char getmxds_con_cap_lsb;                      //45
volatile unsigned char getmxds_con_cap1;                         //46
volatile unsigned char getmxds_con_cap2;
volatile unsigned char reserved12;
volatile unsigned char reserved13;
volatile unsigned char reserved14;
volatile unsigned char reserved15;                               //4A
volatile unsigned char reserved16;
volatile unsigned char reserved17;
volatile unsigned char reserved18;
volatile unsigned char reserved19;
volatile unsigned char bus_mode;                                 //50
volatile unsigned char hdr_ddr_target_config;
volatile unsigned char reserved20;
volatile unsigned char reserved21;
volatile unsigned char hdr_ddr_target_abr_config;                //54
```

4.2. i3c_target_handle_t

This structure is used to declare different types of parameters which are used for the I3C target API. It is given below. Table 4.1 shows the available parameters and descriptions.

```
unsigned int base_addr;
unsigned int len;
unsigned int *buf;
unsigned int max_retry;
unsigned int mdb;
unsigned int *ibi_optional_payload;
unsigned int ibi_optional_payload_len;
```

Table 4.1. i3c_target_handle_t Parameters

Parameter	Description
base_addr	• Member of the i3c_target_handle_t structure. • Base address of the I3C target.
len	• Member of the i3c_target_handle_t structure. • Buffer length of the I3C target for read and write data.
buf	• Member of the i3c_target_handle_t structure. • Buffer pointer of the I3C target for where data is collected.
mdb	• Member of the i3c_target_handle_t structure. • Mandatory data bytes about the type of interrupt sent by the user. • Used in the IBI API.
*ibi_optional_payload	• Member of the i3c_target_handle_t structure. • Optional data bytes sent by the user. • Used in the IBI API.
ibi_optional_payload_len	• Member of the i3c_target_handle_t structure. • Payload length of the optional data byte sent by the user. • Used in the IBI API.
max_retry	• Member of the i3c_target_handle_t structure. • Number of retry to get an ACK from the controller API. • Used in the HJ API, IBI API, and secondary controller API.

5. API Macros

This section describes the macro variables used in the API.

5.1. Driver Details

```
#define DRIVER_MAJOR_VERSION          1
#define DRIVER_MINOR_VERSION          0
#define DRIVER_TEST_VERSION           0
```

5.2. Register Option and Secondary Controller

Use this macro to choose the register bit option:

```
#define REG32_BIT                     1
#define REG8_BIT                      0
```

Set this macro to *1*, when the target is used as a secondary controller:

```
#define SECONDARY_CONTROLLER_ENABLE   0
```

5.3. Set and Reset

```
#define   SET_TGT                     1
#define   RESET                       0
```

5.4. Success and Failure

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

5.5. Initialization

```
#define ZERO                          0
#define CLEAR                         0x00
#define INTERRUPT_ENABLE              0xFF
```

5.6. Tx FIFO and Rx FIFO

```
#define EIGHT_BIT                     8
#define TX_FIFO_FULL_TGT              0x80
#define RX_FIFO_FULL                  0x20
#define RXFIFO_NOTEMPTY               0x40
```

5.7. Hot Join

```
#define EC_REQ_HJIN                   0x08
#define HJ_CAPABLE                    0x08
#define HJ_ENABLE                     0x08
#define BUSAVL                        0x02
#define BUSIDLE                       0x01
#define EC_REQ_HJ                     0x08
#define HJ_REQ_STATUS                 0x80
#define HJ_NACK                       0x20
#define HJ_DONE                       0x40
```

5.8. IBI

```
#define RST_TXFIFO                  0x04
#define IBI_CAPABLE                 0x01
#define IBI_ENABLE                  0x01
#define EC_REQ_IBI                  0x01
#define IBI_REQ_STATUS              0x08
#define IBI_NACK                    0x02
#define BCR_BIT2                    0x04
#define IBI_DONE                    0x04
#define IBI_PAYLD_ERR               0x01
```

5.9. Secondary Controller Request

```
#define EC_EN_CNTRL_ROLE            0x02
#define CNTRL_REQ                   0x02
#define CNTRL_REQ_GEN               0x08
#define CNTRL_REQ_NACK              0x02
#define CNTRL_ROLE_DONE             0x04
```

5.10. FIFO Loop-back

```
#define FIFO_LOOPBACK_EN            0x10
```

References

- I3C Target IP Core User Guide (FPGA-IPUG-02227)
- MachXO2 web page
- MachXO3D web page
- MachXO3 web page
- CrossLink-NX web page
- Certus-NX web page
- CertusPro-NX web page
- Mach-NX web page
- MachXO5-NX web page
- Avant-E web page
- Avant-G web page
- Avant-X 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 1.0, December 2023

Section	Change Summary
All	Initial release.

www.latticesemi.com