**Document Type: Detail Design**

**Document Name:**

**ADSP FRAMEWORK: FIFO DRIVER**

Renesas Electronics Corporation

Renesas Design Vietnam Co. Ltd

R-Car Software Solution 2 Group

Middleware 1 Team

Table of Contents

[1. Overview 3](#_Toc534285282)

[2. Function list 4](#_Toc534285283)

[3. Detail information 5](#_Toc534285284)

[3.1 Data type definition 5](#_Toc534285285)

[3.1.1 FIFO\_MODULE 5](#_Toc534285286)

[3.1.2 FIFO\_ERROR\_CODE 5](#_Toc534285287)

[3.1.3 fifo\_type 5](#_Toc534285288)

[3.1.4 FIFO\_PARAM 6](#_Toc534285289)

[3.2 Macro definition 7](#_Toc534285290)

[3.3 Register definition 8](#_Toc534285291)

[3.4 Function definition 10](#_Toc534285292)

[3.4.1 xa\_fifo\_enable 10](#_Toc534285293)

[3.4.2 xa\_fifo\_config 12](#_Toc534285294)

[3.4.3 xa\_fifo\_get\_int\_num 15](#_Toc534285295)

[3.4.4 xa\_fifo\_register\_interrupt\_handler 16](#_Toc534285296)

[3.4.5 xa\_fifo\_unregister\_interrupt\_handler 17](#_Toc534285297)

[3.4.6 xa\_fifo\_disable\_interrupt 18](#_Toc534285298)

[3.4.7 xa\_fifo\_enable\_interrupt 19](#_Toc534285299)

[3.4.8 xa\_fifo\_clear\_interrupt 20](#_Toc534285300)

[3.4.9 xa\_fifo\_stop 21](#_Toc534285301)

[3.4.10 xa\_fifo\_check\_available 22](#_Toc534285302)

[4. Revision history 24](#_Toc534285303)

List of Figures

[Figure 1‑1 The software architecture 3](#_Toc529965764)

[Figure 3‑1 xa\_fifo\_enable flowchart 11](#_Toc529965765)

[Figure 3‑3 xa\_fifo\_config flowchart 14](#_Toc529965766)

[Figure 3‑4 xa\_fifo\_get\_int\_num flowchart 15](#_Toc529965767)

[Figure 3‑5 xa\_fifo\_register\_interrupt\_handler flowchart 16](#_Toc529965768)

[Figure 3‑6 xa\_fifo\_unregister\_interrupt\_handler flowchart 17](#_Toc529965769)

[Figure 3‑7 xa\_fifo\_disable\_interrupt flowchart 18](#_Toc529965770)

[Figure 3‑8 xa\_fifo\_enable\_interrupt flowchart 19](#_Toc529965771)

[Figure 3‑9 xa\_fifo\_clear\_interrupt 20](#_Toc529965772)

[Figure 3‑10 xa\_fifo\_stop flowchart 21](#_Toc529965773)

[Figure 3‑11 xa\_fifo\_check\_available 23](#_Toc529965774)

List of Table

[Table 2‑1 Function list 4](#_Toc529965775)

[Table 3‑1 FIFO\_MODULE type information 5](#_Toc529965776)

[Table 3‑2 FIFO\_ERROR\_CODE type information 5](#_Toc529965777)

[Table 3‑3 fifo\_type type information 5](#_Toc529965778)

[Table 3‑4 FIFO\_PARAM type information 6](#_Toc529965779)

[Table 3‑5 Setting registers OF\_EN and IF\_EN to enable FIFO 10](#_Toc529965780)

[Table 2‑1 Function list 4](#_Toc534285304)

[Table 3‑1 FIFO\_MODULE type information 5](#_Toc534285305)

[Table 3‑2 FIFO\_ERROR\_CODE type information 5](#_Toc534285306)

[Table 3‑3 fifo\_type type information 5](#_Toc534285307)

[Table 3‑4 FIFO\_PARAM type information 6](#_Toc534285308)

[Table 3‑5 Setting registers OF\_EN and IF\_EN to enable FIFO 10](#_Toc534285309)

# Overview

This section provides an overview of software architecture.

User Application

ADSP Interface

Kernel Space

User Space

ADSP Driver

ARM

Audio HW

**ADSP Framework**

TDM class

Renderer class

Equalizer class

Capture class

TDM Plugin

ADSP

Plugin

Equalizer Plugin\*

Capture Plugin

Renderer Plugin

This document’s target is in side of red square

\* not connect to hardware modules

DAC/

ADC

FIFO

SCU

PDMA

SSI

SSIU

ADMA

Figure 1‑1 ****The software architecture****

# Function list

The following is list of functions:

Table 2‑1 Function list

|  |  |  |
| --- | --- | --- |
| **Type** | **Function Name** | **Outline** |
| API | xa\_fifo\_enable | This API is to enable the input/output of FIFO |
| xa\_fifo\_config | This API is to set up registers necessary for FIFO module execution |
| xa\_fifo\_register\_interrupt\_handler | This API is to register interrupt handler for FIFO |
| xa\_fifo\_unregister\_interrupt\_handler | This API is to unregister interrupt handler for FIFO |
| xa\_fifo\_disable\_interrupt | This API is to disable FIFO interrupt |
| xa\_fifo\_enable\_interrupt | This API is to enable FIFO interrupt |
| xa\_fifo\_clear\_interrupt | This API is to clear FIFO interrupt |
| xa\_fifo\_stop | This API is to stop FIFO operation |
| xa\_fifo\_check\_available | This API is to check availability of FIFO module and choose an available one |
| Internal function | xa\_fifo\_get\_int\_num | This function is to get the interrupt number |

# Detail information

This section describes detail information of data types, macro definitions, implemented APIs and internal function units, global variable.

## Data type definition

### FIFO\_MODULE

The data type FIFO\_MODULE is a type-defined enumeration that lists all supported FIFO modules.

Table 3‑1 FIFO\_MODULE type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Value** | **Outline** |
| FIFO0 | 0 | 1st FIFO |
| FIFO1 | 1 | 2nd FIFO |
| FIFO2 | 2 | 3rd FIFO |
| FIFO3 | 3 | 4th FIFO |
| FIFOMAX | 4 | Number of FIFO modules |

### FIFO\_ERROR\_CODE

The data type FIFO\_ERROR\_CODE is a type-defined enumeration that lists all error codes for FIFO.

Table 3‑2 FIFO\_ERROR\_CODE type information

|  |  |
| --- | --- |
| **Member name** | **Value** |
| FIFO\_ERROR\_NONE | 0 |
| FIFO\_ERROR\_INVALID | -1 |
| FIFO\_ERROR\_INTERNAL | -2 |
| FIFO\_ERROR\_BUSY | -3 |
| FIFO\_ERROR\_OUT\_RANGE | -4 |

### fifo\_type

The data type fifo\_type is a type-defined enumeration of indicators of whether FIFO’s input or output is used.

Table 3‑3 fifo\_type type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Value** | **Outline** |
| FIFO\_INPUT | 0 | FIFO input is used |
| FIFO\_OUTPUT | 1 | FIFO output is used |

### FIFO\_PARAM

The data type FIFO\_PARAM is a type-defined structure that possesses necessary parameters for FIFO module.

Table 3‑4 FIFO\_PARAM type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Data type** | **Outline** |
| channel | UWORD32 | channel of data used for FIFO module |
| pcm\_width | UWORD32 | PCM width of data used for FIFO module |
| sample\_rate | UWORD32 | Sampling rate of data used for FIFO module |
| fifo\_num | FIFO\_MODULE | FIFO module |
| type | fifo\_type | FIFO type |
| dev\_connect | UWORD32 | Indicator to show which module FIFO connects |
| ssi\_idx | UWORD32 | SSI module |

## Macro definition

|  |  |  |
| --- | --- | --- |
| **Macros** | **Value** | **Outline** |
| XA\_FIFO\_FULL | 1024 | FIFO full size |
| XA\_FIFO\_NEARFULL | 768 | FIFO nearly full size |
| XA\_FIFO\_NEAREMPTY | 64 | FIFO nearly empty size |
| XA\_FIFO\_EMPTY | 0 | FIFO empty size |
| XA\_FIFO\_CONNECT\_SRC | 0 | FIFO is connected to SRC |
| XA\_FIFO\_CONNECT\_SSI | 1 | FIFO is connected to SSI |

## Register definition

Below is the table listing all items that are macros representing register addresses located in repository s492d/include/sys/xt-shmem/board-rcar/xf-registers.h

|  |  |
| --- | --- |
| **Register** | **Outline** |
| XF\_RCAR\_REG\_OF\_EN | Register OF\_EN |
| XF\_RCAR\_REG\_IF\_EN | Register IF\_EN |
| XF\_RCAR\_REG\_ADSP\_TIMSEL | ADSP Output Timing Select Register |
| XF\_RCAR\_REG\_ADSP\_TIMSEL1 | ADSP Output Timing Select Register 1 |
| XF\_RCAR\_REG\_ADSP\_TIMSEL2 | ADSP Output Timing Select Register 2 |
| XF\_RCAR\_REG\_ADSP\_TIMSEL3 | ADSP Output Timing Select Register 3 |
| XF\_RCAR\_REG\_OF\_MSTP | Output FIFO Module Stop Register |
| XF\_RCAR\_REG\_OF\_PTRRST | Output FIFO Pointer Reset Register |
| XF\_RCAR\_REG\_OF0\_CH | Output FIFO0 Channel Register |
| XF\_RCAR\_REG\_OF1\_CH | Output FIFO1 Channel Register |
| XF\_RCAR\_REG\_OF2\_CH | Output FIFO2 Channel Register |
| XF\_RCAR\_REG\_OF3\_CH | Output FIFO3 Channel Register |
| XF\_RCAR\_REG\_OF\_ADINR | Output FIFO0 Audio Information Register |
| XF\_RCAR\_REG\_OF1\_ADINR | Output FIFO1 Audio Information Register |
| XF\_RCAR\_REG\_OF2\_ADINR | Output FIFO2 Audio Information Register |
| XF\_RCAR\_REG\_OF3\_ADINR | Output FIFO3 Audio Information Register |
| XF\_RCAR\_REG\_OF0\_INT\_TH | Output FIFO0 Interrupt Threshold Register |
| XF\_RCAR\_REG\_OF1\_INT\_TH | Output FIFO1 Interrupt Threshold Register |
| XF\_RCAR\_REG\_OF2\_INT\_TH | Output FIFO2 Interrupt Threshold Register |
| XF\_RCAR\_REG\_OF3\_INT\_TH | Output FIFO3 Interrupt Threshold Register |
| XF\_RCAR\_REG\_OF\_INT\_CLR | Output FIFO Interrupt Clear Register |
| XF\_RCAR\_REG\_IF\_MSTP | Input FIFO Module Stop Register |
| XF\_RCAR\_REG\_IF\_PTRRST | Input FIFO Pointer Reset Register |
| XF\_RCAR\_REG\_IF0\_CH | Input FIFO0 Channel Register |
| XF\_RCAR\_REG\_IF1\_CH | Input FIFO1 Channel Register |
| XF\_RCAR\_REG\_IF2\_CH | Input FIFO2 Channel Register |
| XF\_RCAR\_REG\_IF3\_CH | Input FIFO3 Channel Register |
| XF\_RCAR\_REG\_IF\_ADINR | Input FIFO0 Audio Information Register |
| XF\_RCAR\_REG\_IF1\_ADINR | Input FIFO1 Audio Information Register |
| XF\_RCAR\_REG\_IF2\_ADINR | Input FIFO2 Audio Information Register |
| XF\_RCAR\_REG\_IF3\_ADINR | Input FIFO3 Audio Information Register |
| XF\_RCAR\_REG\_IF0\_INT\_TH | Input FIFO0 Interrupt Threshold Register |
| XF\_RCAR\_REG\_IF1\_INT\_TH | Input FIFO1 Interrupt Threshold Register |
| XF\_RCAR\_REG\_IF2\_INT\_TH | Input FIFO2 Interrupt Threshold Register |
| XF\_RCAR\_REG\_IF3\_INT\_TH | Input FIFO3 Interrupt Threshold Register |
| XF\_RCAR\_REG\_IF\_INT\_CLR | Input FIFO Interrupt Clear Register |

## Function definition

### xa\_fifo\_enable

DD\_PLG\_TDM\_04\_001

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | inline FIFO\_ERROR\_CODE xa\_fifo\_enable(FIFO\_MODULE fifo\_num, fifo\_type type) | | | |
| **Function** | This function is to enable the input/output of FIFO. | | | |
| **Arguments** | Type | Name | I/O | Description |
| FIFO\_MODULE | fifo\_num | I | FIFO module  Valid values: [0: 3] |
| fifo\_type | type | I | FIFO uses its input or output  Valid values: 0/1 |
| **Return value** | FIFO\_ERROR\_BUSY | | FIFO module is busy | |
| FIFO\_ERROR\_NONE | | Normally end | |
| **Description** | * xa\_fifo\_enable command processing:   - Set either register OF\_EN or IF\_EN based on FIFO type | | | |

[Covers: FD\_PLG\_TDM\_005]

Table 3‑5 Setting registers OF\_EN and IF\_EN to enable FIFO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **fifo\_num** | **type** | **Register** | **Bit** | **Value** |
| FIFO0 | FIFO\_OUTPUT | XF\_RCAR\_REG\_OF\_EN | 0 | 1 |
| FIFO\_INPUT | XF\_RCAR\_REG\_IF\_EN | 0 | 1 |
| FIFO1 | FIFO\_OUTPUT | XF\_RCAR\_REG\_OF\_EN | 1 | 1 |
| FIFO\_INPUT | XF\_RCAR\_REG\_IF\_EN | 1 | 1 |
| FIFO2 | FIFO\_OUTPUT | XF\_RCAR\_REG\_OF\_EN | 2 | 1 |
| FIFO\_INPUT | XF\_RCAR\_REG\_IF\_EN | 2 | 1 |
| FIFO3 | FIFO\_OUTPUT | XF\_RCAR\_REG\_OF\_EN | 3 | 1 |
| FIFO\_INPUT | XF\_RCAR\_REG\_IF\_EN | 3 | 1 |

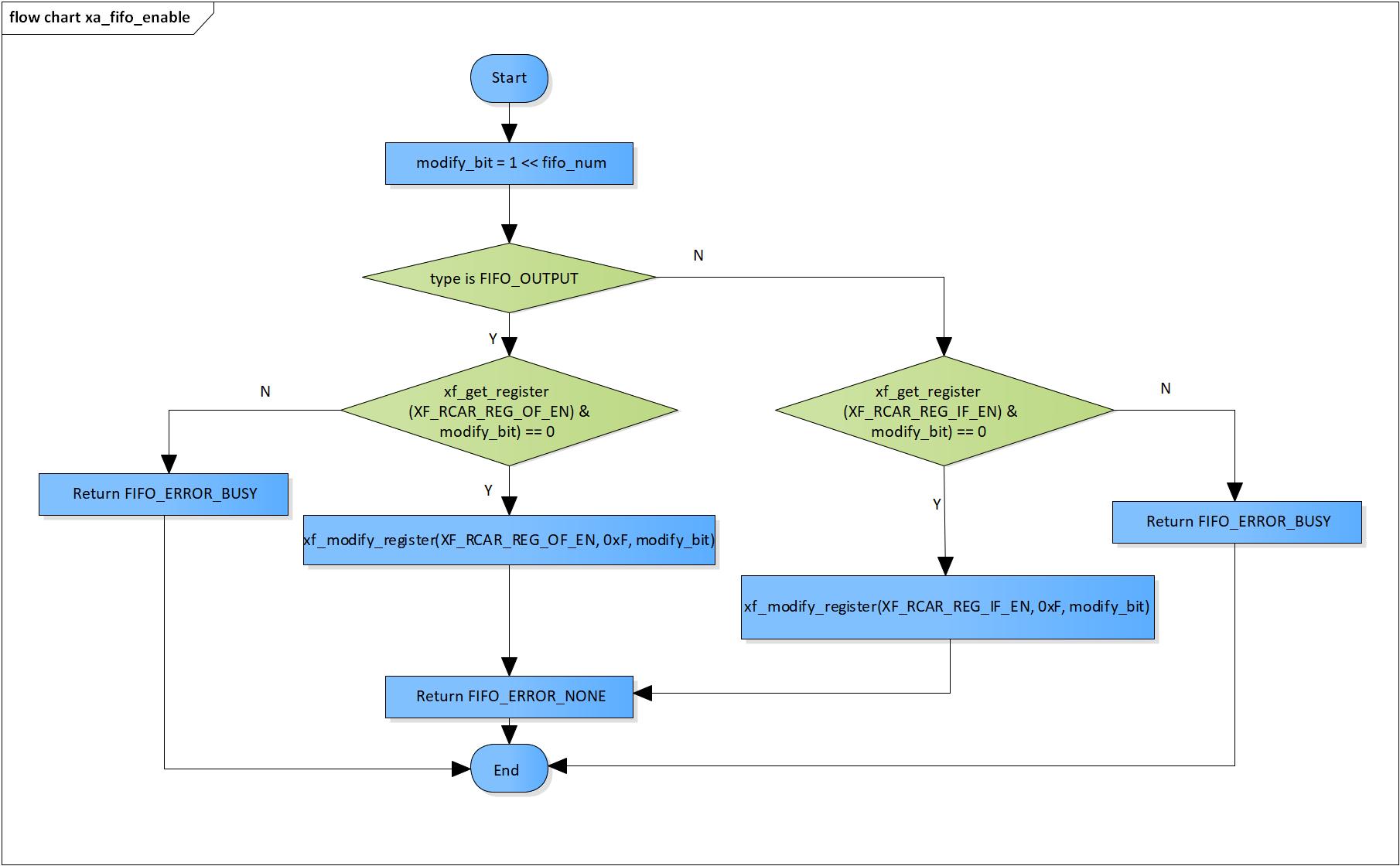


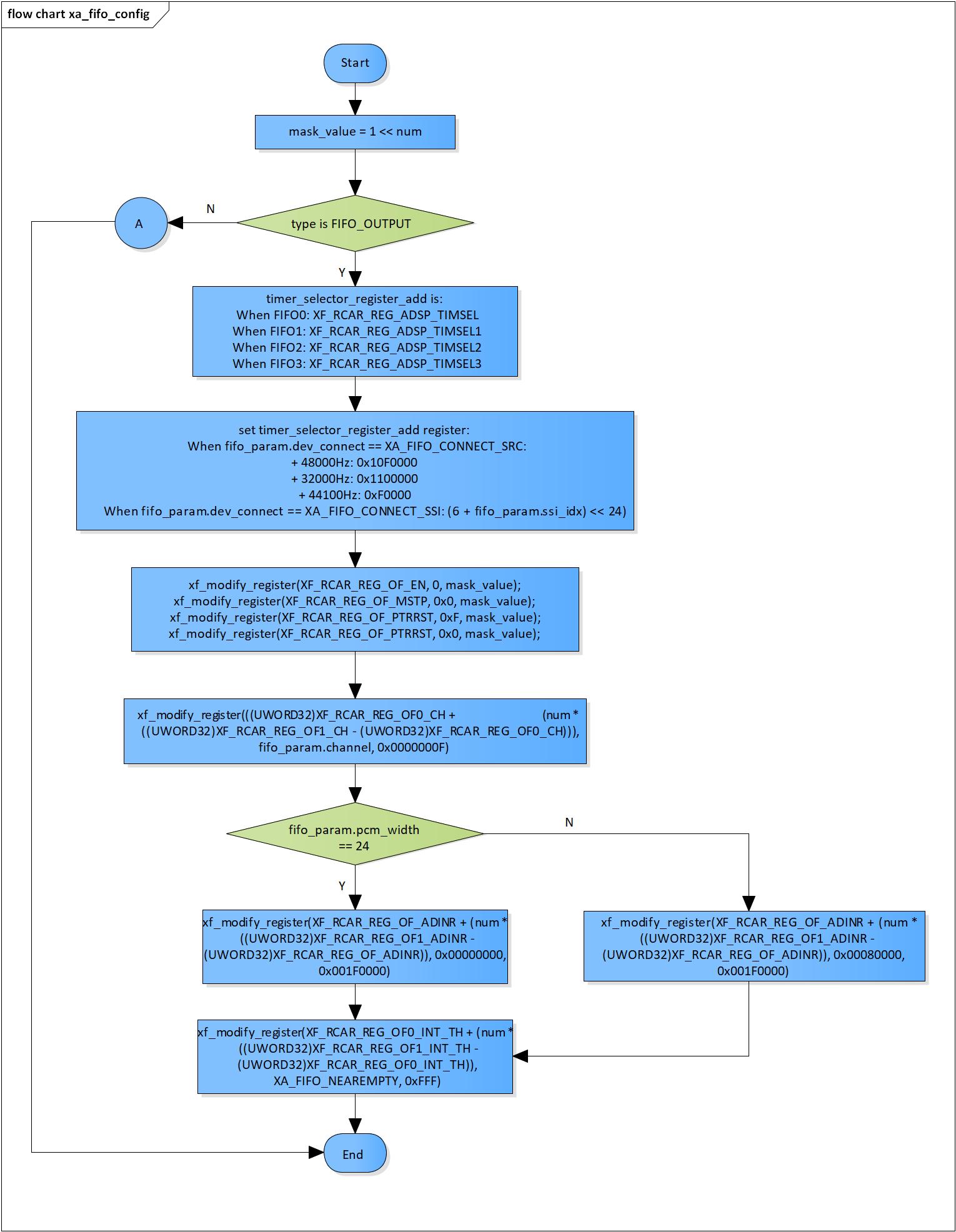
Figure 3‑1 xa\_fifo\_enable flowchart

### xa\_fifo\_config

DD\_PLG\_TDM\_04\_002

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | VOID xa\_fifo\_config(FIFO\_PARAM fifo\_param) | | | |
| **Function** | This function is to set up registers necessary for FIFO module execution. | | | |
| **Arguments** | Type | Name | I/O | Description |
| FIFO\_PARAM | fifo\_param | I | Struct of parameters of FIFO |
| **Return value** | None | | | |
| **Description** | * xa\_fifo\_config command processing:   - Set registers SSI\_BUSIF\_MODE, SSI\_BUSIF\_ADINR, SSI mode register | | | |

[Covers: FD\_PLG\_TDM\_005]



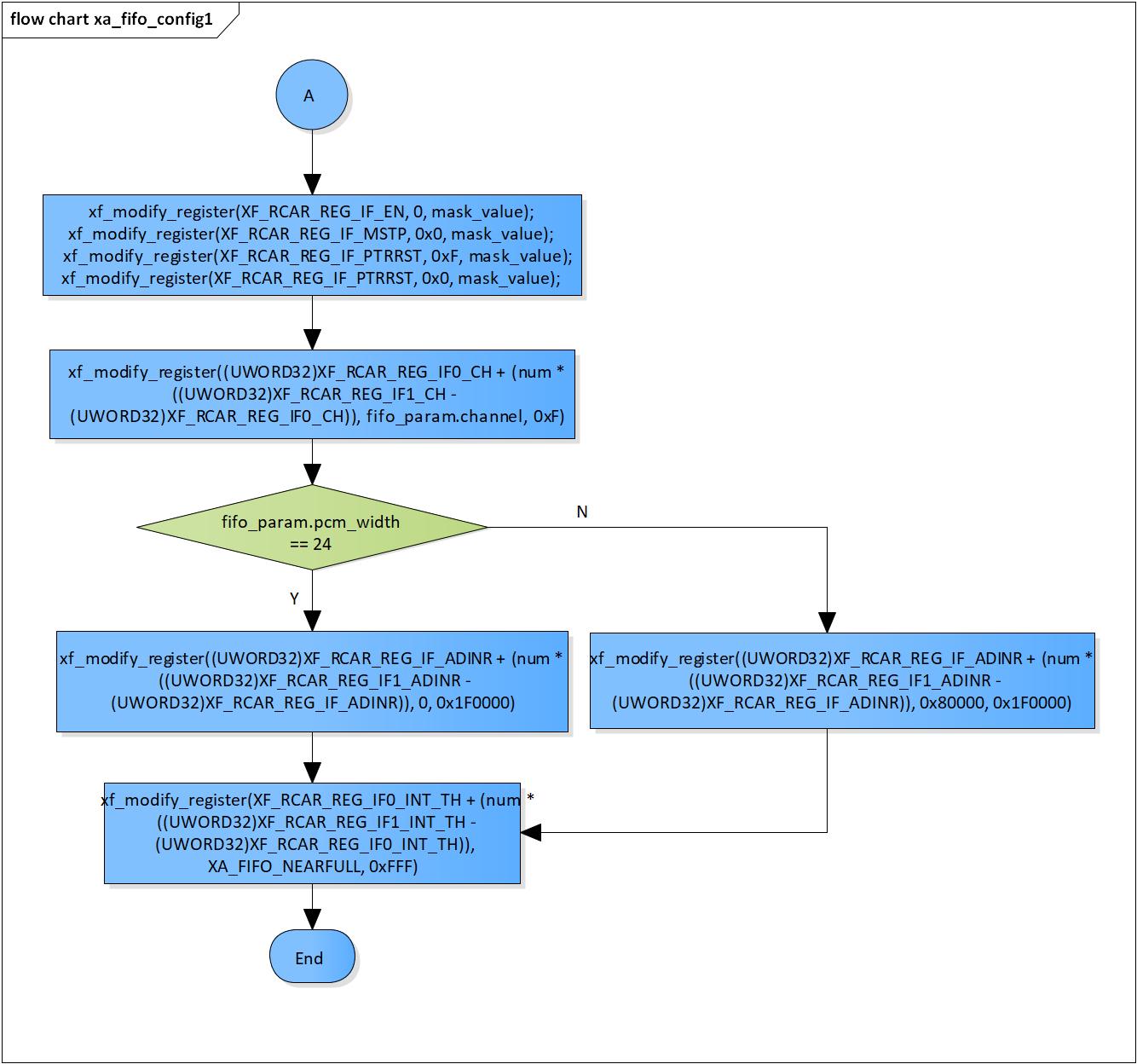


Figure 3‑2 xa\_fifo\_config flowchart

### xa\_fifo\_get\_int\_num

DD\_PLG\_TDM\_04\_003

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static inline WORD32 xa\_fifo\_get\_int\_num(FIFO\_MODULE fifo\_num, fifo\_type type) | | | |
| **Function** | This function is to get interrupt number of FIFO module. | | | |
| **Arguments** | Type | Name | I/O | Description |
| FIFO\_MODULE | fifo\_num | I | FIFO module  Valid values: [0: 3] |
| fifo\_type | type | I | FIFO type  Valid values: 0/1 |
| **Return value** | Interrupt number | | | |
| **Description** | * xa\_fifo\_get\_int\_num command processing:   - Get interrupt number based on FIFO module and its type (FIFO\_INPUT or FIFO\_OUTPUT) | | | |

[Covers: FD\_PLG\_TDM\_005, FD\_PLG\_TDM\_017]

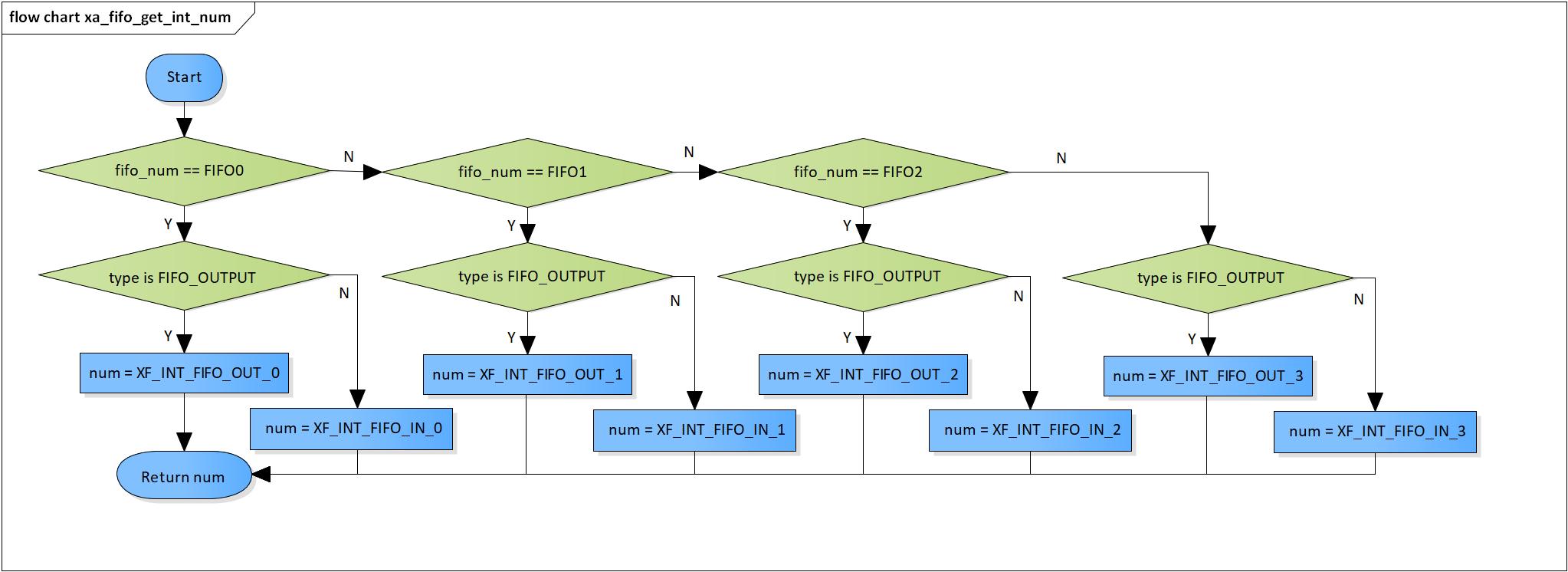


Figure 3‑3 xa\_fifo\_get\_int\_num flowchart

### xa\_fifo\_register\_interrupt\_handler

DD\_PLG\_TDM\_04\_004

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | VOID xa\_fifo\_register\_interrupt\_handler(FIFO\_MODULE fifo\_num, fifo\_type type, pVOID function\_addr, pVOID arg) | | | |
| **Function** | This function is to register interrupt handler for FIFO. | | | |
| **Arguments** | Type | Name | I/O | Description |
| FIFO\_MODULE | fifo\_num | I | FIFO module  Valid values: [0: 3] |
| fifo\_type | type | I | FIFO type  Valid values: 0/1 |
| pVOID | function\_addr | I | Pointer to handler function |
| pVOID | arg | I | Argument passed to handler |
| **Return value** | None | | | |
| **Description** | * xa\_fifo\_register\_interrupt\_handler command processing:   - Register interrupt handler for FIFO | | | |

[Covers: FD\_PLG\_TDM\_005]

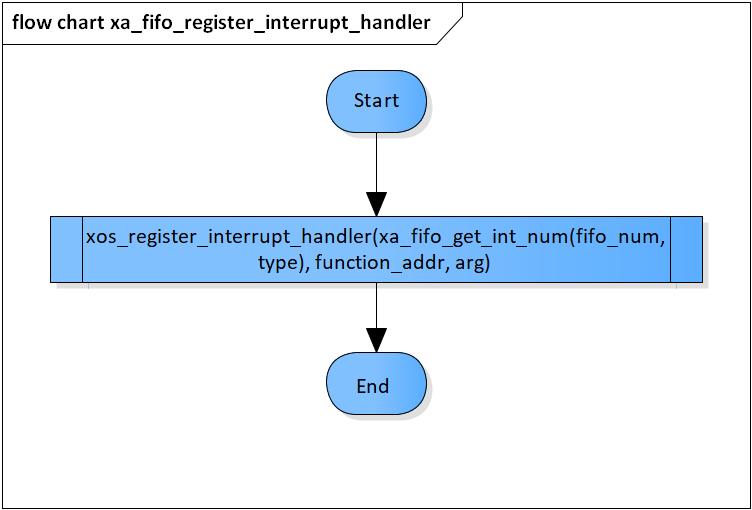


Figure 3‑4 xa\_fifo\_register\_interrupt\_handler flowchart

### xa\_fifo\_unregister\_interrupt\_handler

DD\_PLG\_TDM\_04\_005

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | VOID xa\_fifo\_unregister\_interrupt\_handler(FIFO\_MODULE fifo\_num,fifo\_type type) | | | |
| **Function** | This function is to unregister interrupt handler for FIFO. | | | |
| **Arguments** | Type | Name | I/O | Description |
| FIFO\_MODULE | fifo\_num | I | FIFO module  Valid values: [0: 3] |
| fifo\_type | type | I | FIFO type  Valid values: 0/1 |
| **Return value** | None | | | |
| **Description** | * xa\_fifo\_unregister\_interrupt\_handler command processing:   - Unregister interrupt handler for FIFO | | | |

[Covers: FD\_PLG\_TDM\_017]

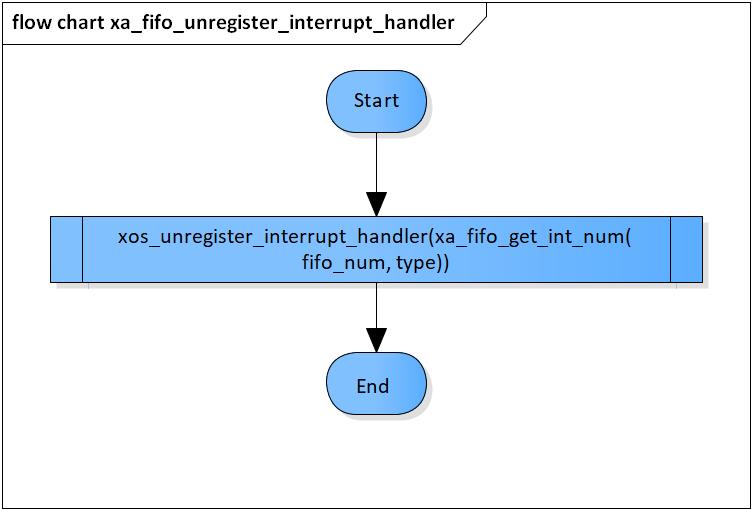


Figure 3‑5 xa\_fifo\_unregister\_interrupt\_handler flowchart

### xa\_fifo\_disable\_interrupt

DD\_PLG\_TDM\_04\_006

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | inline VOID xa\_fifo\_disable\_interrupt(FIFO\_MODULE fifo\_num, fifo\_type type) | | | |
| **Function** | This function is to disable interrupt for FIFO. | | | |
| **Arguments** | Type | Name | I/O | Description |
| FIFO\_MODULE | fifo\_num | I | FIFO module  Valid values: [0: 3] |
| fifo\_type | type | I | FIFO type  Valid values: 0/1 |
| **Return value** | None | | | |
| **Description** | * xa\_fifo\_disable\_interrupt command processing:   - Disable FIFO interrupt | | | |

[Covers: FD\_PLG\_TDM\_017]

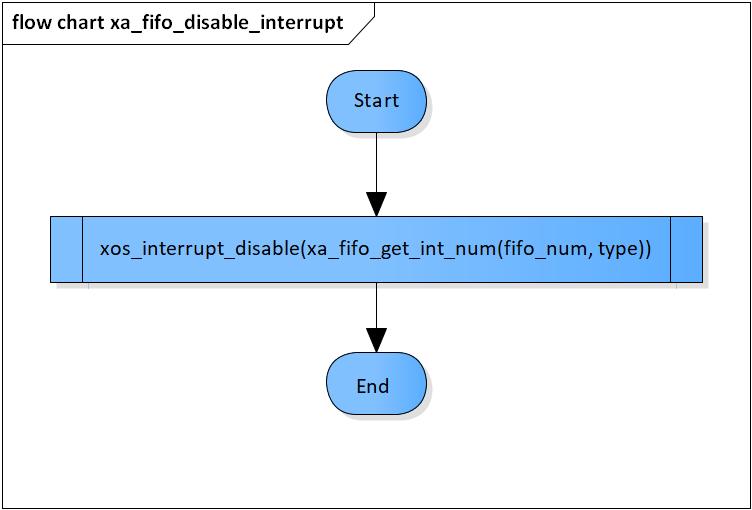


Figure 3‑6 xa\_fifo\_disable\_interrupt flowchart

### xa\_fifo\_enable\_interrupt

DD\_PLG\_TDM\_04\_007

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | inline VOID xa\_fifo\_enable\_interrupt(FIFO\_MODULE fifo\_num, fifo\_type type) | | | |
| **Function** | This function is to enable interrupt for FIFO. | | | |
| **Arguments** | Type | Name | I/O | Description |
| FIFO\_MODULE | fifo\_num | I | FIFO module  Valid values: [0: 3] |
| fifo\_type | type | I | FIFO type  Valid values: 0/1 |
| **Return value** | None | | | |
| **Description** | * xa\_fifo\_enable\_interrupt command processing:   - Enable FIFO interrupt | | | |

[Covers: FD\_PLG\_TDM\_005, FD\_PLG\_TDM\_017]

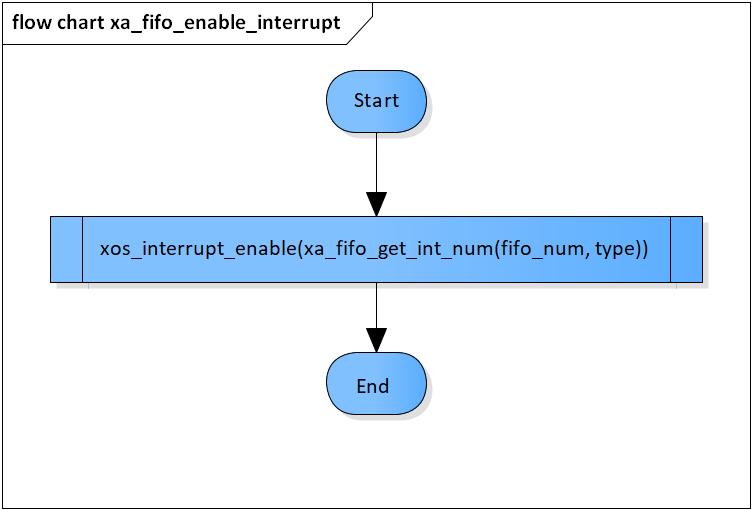
****

Figure 3‑7 xa\_fifo\_enable\_interrupt flowchart

### xa\_fifo\_clear\_interrupt

DD\_PLG\_TDM\_04\_008

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | inline VOID xa\_fifo\_clear\_interrupt(FIFO\_MODULE fifo\_num, fifo\_type type) | | | |
| **Function** | This function is to clear FIFO interrupt. | | | |
| **Arguments** | Type | Name | I/O | Description |
| FIFO\_MODULE | fifo\_num | I | FIFO module  Valid values: [0: 3] |
| fifo\_type | type | I | FIFO type  Valid values: 0/1 |
| **Return value** | None | | | |
| **Description** | * xa\_fifo\_clear\_interrupt command processing:   - Clear FIFO interrupt | | | |

[Covers: FD\_PLG\_TDM\_005, FD\_PLG\_TDM\_017]

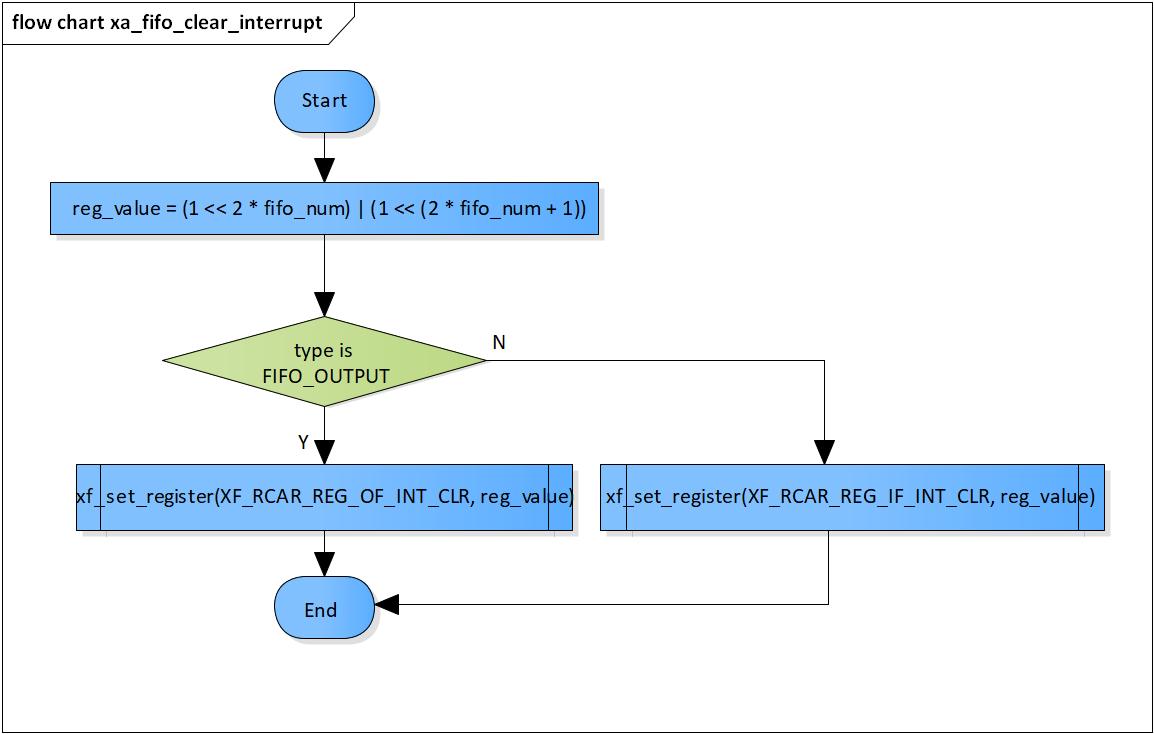
****

Figure 3‑8 xa\_fifo\_clear\_interrupt

### xa\_fifo\_stop

DD\_PLG\_TDM\_04\_009

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | FIFO\_ERROR\_CODE xa\_fifo\_stop(FIFO\_MODULE fifo\_num, fifo\_type type) | | | |
| **Function** | This function is to stop FIFO. | | | |
| **Arguments** | Type | Name | I/O | Description |
| FIFO\_MODULE | fifo\_num | I | FIFO module  Valid values: [0: 3] |
| fifo\_type | type | I | FIFO type  Valid values: 0/1 |
| **Return value** | FIFO\_ERROR\_OUT\_RANGE | | FIFO module is invalid | |
| FIFO\_ERROR\_NONE | | Normally end | |
| **Description** | * xa\_fifo\_stop command processing:   - Stop FIFO operation | | | |

[Covers: FD\_PLG\_TDM\_017]

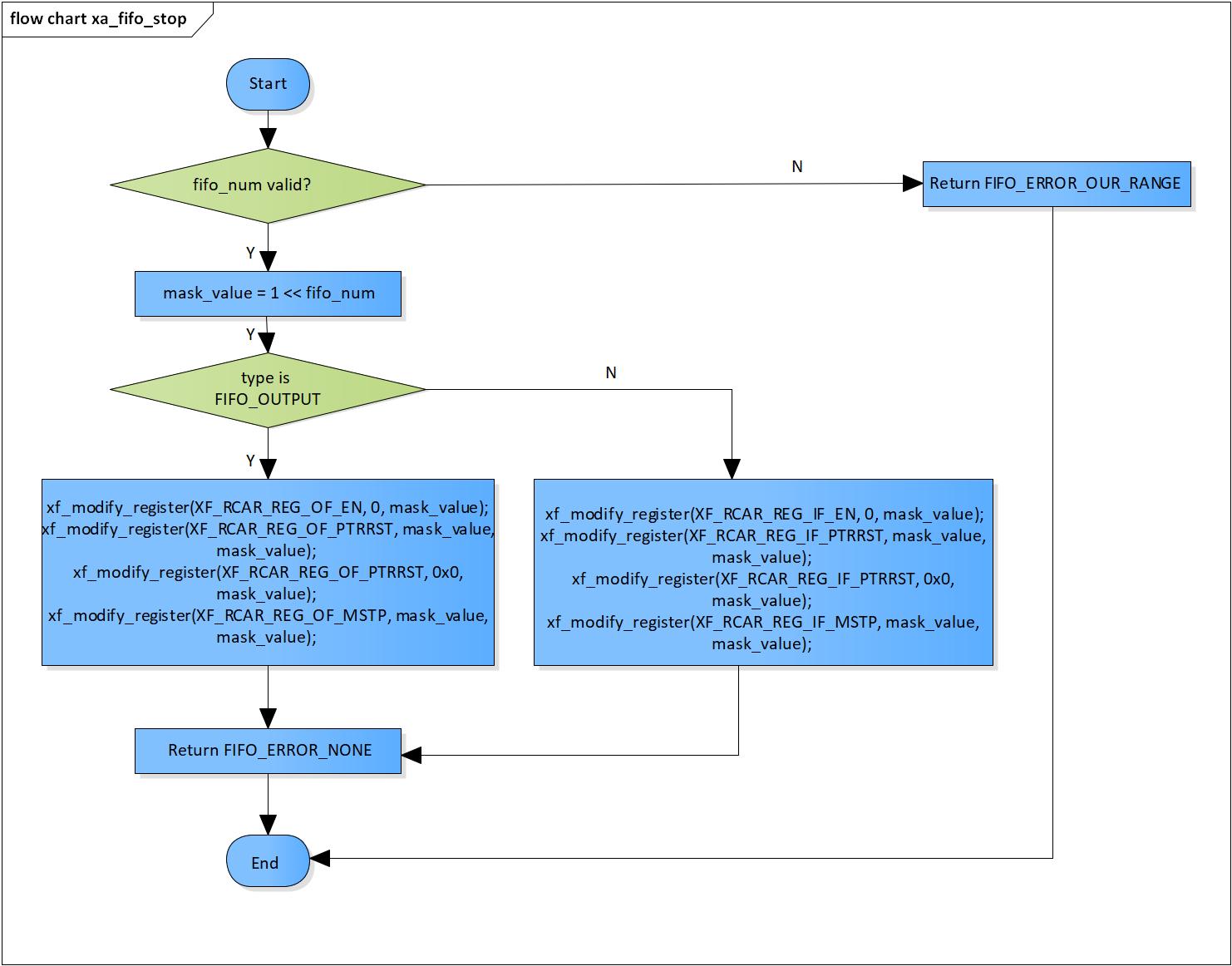


Figure 3‑9 xa\_fifo\_stop flowchart

### xa\_fifo\_check\_available

DD\_PLG\_TDM\_04\_010

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | inline FIFO\_ERROR\_CODE xa\_fifo\_check\_available(UWORD32 \*fifo\_num, fifo\_type type) | | | |
| **Function** | This function is to check availability of FIFOs and choose an available one for operation. | | | |
| **Arguments** | Type | Name | I/O | Description |
| UWORD32 \* | fifo\_num | I/O | Pointer to FIFO number  Valid values: [0: 3] |
| fifo\_type | type | I | FIFO type  Valid values: 0/1 |
| **Return value** | FIFO\_ERROR\_OUT\_RANGE | | FIFO module is invalid | |
| FIFO\_ERROR\_BUSY | | No available FIFO module | |
| FIFO\_ERROR\_NONE | | Normally end | |
| **Description** | * xa\_fifo\_check\_available command processing:   - Check availability of FIFOs  - Choose an available one for operation | | | |

[Covers: FD\_PLG\_TDM\_005]

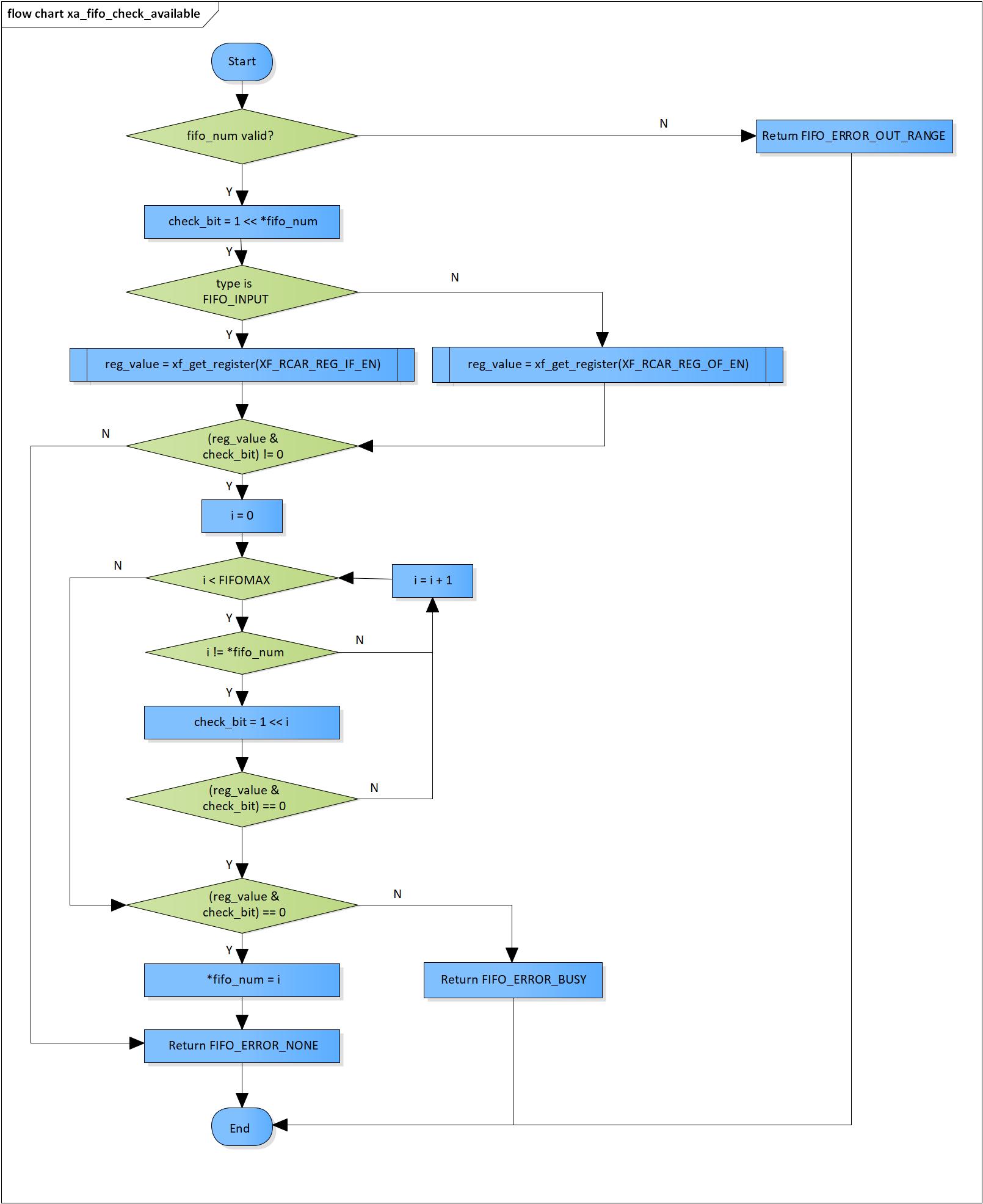


Figure 3‑10 xa\_fifo\_check\_available

# Revision history

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Date** | **Page** | **Content** | **Approved** | **Changed** |
| 1.0.0 | Nov 14 2018 | - | First Edition issued | Vu Phan | Nguyen Dang |
| 1.1.0 | Dec 10 2018 | - | Add traceability ID | Vu Phan | Nguyen Dang |
| 1.2.0 | Jan 03, 2019 | - | Add range for input parameters | Vu Phan | Tien Tran |