**Document Type: Detail Design**

**Document Name:**

**ADSP FRAMEWORK: SSIU DRIVER**

Renesas Electronics Corporation

Renesas Design Vietnam Co. Ltd

R-Car Software Solution 2 Group

Middleware 1 Team

Table of Contents

[1. Overview 3](#_Toc529909215)

[2. Function list 4](#_Toc529909216)

[3. Detail information 5](#_Toc529909217)

[3.1 Data type definition 5](#_Toc529909218)

[3.1.1 SSIU\_SSI\_MODULE 5](#_Toc529909219)

[3.1.2 SSIU\_PIN\_MODE 5](#_Toc529909220)

[3.1.3 SSIU\_SWAP\_FUNC 6](#_Toc529909221)

[3.1.4 SSIU\_PARAMS 6](#_Toc529909222)

[3.1.5 SSIU\_ERR\_CODE 6](#_Toc529909223)

[3.1.6 SSIU\_PARAMS 6](#_Toc529909224)

[3.1.7 SSIU\_CONTROL 7](#_Toc529909225)

[3.2 Macro definition 8](#_Toc529909226)

[3.3 Function definition 9](#_Toc529909227)

[3.3.1 ssiu\_setup 9](#_Toc529909228)

[3.3.2 ssiu\_start 12](#_Toc529909229)

[3.3.3 ssiu\_stop 13](#_Toc529909230)

[3.3.5 ssiu\_get\_register\_base 14](#_Toc529909231)

[4. Revision history 15](#_Toc529909232)

List of Figures

[Figure 1‑1 The software architecture 3](#_Toc529909178)

[Figure 3‑1 ssiu\_setup flowchart 11](#_Toc529909179)

[Figure 3‑2 ssiu\_start flowchart 12](#_Toc529909180)

[Figure 3‑3 ssiu\_stop flowchart 13](#_Toc529909181)

[Figure 3‑4 ssiu\_get\_register\_base flowchart 14](#_Toc529909182)

List of Table

[Table 2‑1 Function list 4](#_Toc529909184)

[Table 3‑1 SSIU\_SSI\_MODULE type information 5](#_Toc529909185)

[Table 3‑2 SSIU\_PIN\_MODE type information 5](#_Toc529909186)

[Table 3‑3 SSIU\_SWAP\_FUNC type information 6](#_Toc529909187)

[Table 3‑4 SSIU\_PARAMS type information 6](#_Toc529909188)

[Table 3‑5 SSIU\_ERR\_CODE type information 6](#_Toc529909189)

[Table 3‑6 SSIU\_PARAMS type information 6](#_Toc529909190)

[Table 3‑7 SSIU\_CONTROL type information 7](#_Toc529909191)

[Table 3‑8 Setting of SSI\_BUSIF\_ADINR based on PCM width 9](#_Toc529909192)

[Table 3‑9 Setting of SSI\_BUSIF\_ADINR based on channel 9](#_Toc529909193)

[Table 3‑10 Setting pin mode 10](#_Toc529909194)

# Overview

This section provides an overview of software architecture.

User Application

ADSP Interface

Kernel Space

User Space

ADSP Driver

ARM

Audio HW

**ADSP Framework**

Renderer class

TDM class

Equalizer class

Capture class

ADSP

Plugin

Equalizer Plugin\*

TDM Plugin

Capture Plugin

Renderer Plugin

This document’s target is in side of red square

\* not connect to hardware modules

DAC/

ADC

SCU

PDMA

FIFO

SSI

SSIUU

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 | ssiu\_setup | This API is to set up registers necessary for SSIU module execution |
| ssiu\_start | This API is to start SSIU module |
| ssiu\_stop | This API is to stop SSIU module |
| Internal functions | ssiu\_get\_register\_base | This function is to get base address of registers for SSIU module |

# Detail information

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

## Data type definition

### SSIU\_SSI\_MODULE

The data type SSIU\_SSI\_MODULE is a type-defined structure that lists SSIU’s properties: module index and bus index.

Table 3‑1 SSIU\_SSI\_MODULE type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Data type** | **Outline** |
| module\_index | UWORD32 | SSIU module index |
| bus\_index | UWORD32 | SSIU bus index |

### SSIU\_PIN\_MODE

The data type SSIU\_PIN\_MODE is a type-defined enumeration that lists all supported pin modes for SSIU.

Table 3‑2 SSIU\_PIN\_MODE type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Value** | **Outline** |
| SSIU\_NON\_PIN\_MODE | 0 | No setting pin mode for SSIU |
| SSIU\_S3\_S4\_PIN\_MODE | 1 | SSI3 works as slave and SSI4 works as slave |
| SSIU\_M3\_S4\_PIN\_MODE | 2 | SSI3 works as master and SSI4 works as slave |
| SSIU\_S0\_S2\_PIN\_MODE | 3 | SSI0 works as slave and SSI2 works as slave |
| SSIU\_M0\_S2\_PIN\_MODE | 4 | SSI0 works as master and SSI2 works as slave |
| SSIU\_S0\_S1\_PIN\_MODE | 5 | SSI0 works as slave and SSI1 works as slave |
| SSIU\_M0\_S1\_PIN\_MODE | 6 | SSI0 works as master and SSI1 works as slave |
| SSIU\_S0\_S9\_PIN\_MODE | 7 | SSI0 works as slave and SSI9 works as slave |
| SSIU\_M0\_S9\_PIN\_MODE | 8 | SSI0 works as master and SSI9 works as slave |
| SSIU\_S3\_S9\_PIN\_MODE | 9 | SSI3 works as slave and SSI9 works as slave |
| SSIU\_M3\_S9\_PIN\_MODE | 10 | SSI3 works as master and SSI9 works as slave |
| SSIU\_S0\_S3\_PIN\_MODE | 11 | SSI0 works as slave and SSI3 works as slave |
| SSIU\_M0\_S3\_PIN\_MODE | 12 | SSI0 works as master and SSI3 works as slave |

### SSIU\_SWAP\_FUNC

The data type SSIU\_SWAP\_FUNC is a type-defined enumeration of indicators of whether swap function is enabled or not.

Table 3‑3 SSIU\_SWAP\_FUNC type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Value** | **Outline** |
| SSIU\_WORD\_SWAP\_DISABLE | 0 | Swap function is disabled |
| SSIU\_WORD\_SWAP\_ENABLE | 1 | Swap function is enabled |

### SSIU\_PARAMS

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

Table 3‑4 SSIU\_PARAMS type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Data type** | **Outline** |
| channel | UWORD32 | channel of data used for SSIU module |
| pcm\_width | UWORD32 | PCM width of data used for SSIU module |
| pin\_mode | SSIU\_PIN\_MODE | pin mode for SSIU module |
| swap\_func | SSIU\_SWAP\_FUNC | Swap function indicator |

### SSIU\_ERR\_CODE

The data type SSIU\_ERR\_CODE is a type-defined enumeration that lists all error codes for SSIU module.

Table 3‑5 SSIU\_ERR\_CODE type information

|  |  |
| --- | --- |
| **Member name** | **Value** |
| SSIU\_ERR\_NONE | 0 |
| SSIU\_ERR\_OUT\_RANGE | -1 |

### SSIU\_PARAMS

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

Table 3‑6 SSIU\_PARAMS type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Data type** | **Outline** |
| channel | UWORD32 | channel of data used for SSIU module |
| pcm\_width | UWORD32 | PCM width of data used for SSIU module |
| pin\_mode | SSIU\_PIN\_MODE | pin mode for SSIU module |
| swap\_func | SSIU\_SWAP\_FUNC | Swap function indicator |

### SSIU\_CONTROL

The data type SSIU\_CONTROL is a type-defined structure that lists registers of SSIU module.

Table 3‑7 SSIU\_CONTROL type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Data type** | **Register** |
| SSIn\_BUSIF\_MODE | volatile UWORD32 | SSI\_BUSIF\_MODE |
| SSIn\_BUSIF\_ADINR | volatile UWORD32 | SSI\_BUSIF\_ADINR |
| SSIn\_BUSIF\_DALIGN | volatile UWORD32 | SSI\_BUSIF\_DALIGN |
| SSIn\_MODE | volatile UWORD32 | SSI\_MODE |
| SSIn\_CONTROL | volatile UWORD32 | SSI\_CONTROL |
| SSIn\_STATUS | volatile UWORD32 | SSI\_STATUS |
| SSIn\_INT\_ENABLE\_MAIN | volatile UWORD32 | SSI\_INT\_ENABLE\_MAIN |

## Macro definition

|  |  |  |
| --- | --- | --- |
| **Macros** | **Value** | **Outline** |
| SSI\_MODE0 | XF\_RCAR\_REG\_SSI\_MODE0 | SSI Mode Register 0 |
| SSI\_MODE1 | XF\_RCAR\_REG\_SSI\_MODE1 | SSI Mode Register 1 |
| SSI\_MODE2 | XF\_RCAR\_REG\_SSI\_MODE2 | SSI Mode Register 2 |
| SSI\_MODE3 | XF\_RCAR\_REG\_SSI\_MODE3 | SSI Mode Register 3 |

Note: XF\_RCAR\_REG\_SSI\_MODEn (n = 0, 1, 2, 3) is in repository s492d/include/sys/xt-shmem/board-rcar/xf-registers.h

## Function definition

### ssiu\_setup

DD\_PLG\_RDR\_04\_001

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | SSIU\_ERR\_CODE ssiu\_setup(SSIU\_SSI\_MODULE module, SSIU\_PARAMS params) | | | |
| **Function** | This function is to set up registers necessary for SSIU module execution. | | | |
| **Arguments** | Type | Name | I/O | Description |
| SSIU\_SSI\_MODULE | module | I | SSIU module |
| SSIU\_PARAMS | params | I | Struct of parameters to set |
| **Return value** | SSIU\_ERR\_OUT\_RANGE | | Module index is invalid  Bus index is invalid  Channel is invalid  Pin mode is invalid  PCM width is invalid | |
| SSIU\_ERR\_NONE | | Normal end | |
| **Description** | * ssiu\_setup command processing:   - Set registers SSI\_BUSIF\_MODE, SSI\_BUSIF\_ADINR, SSI mode register | | | |

[Covers: FD\_PLG\_RDR\_005]

Table 3‑8 Setting of SSI\_BUSIF\_ADINR based on PCM width

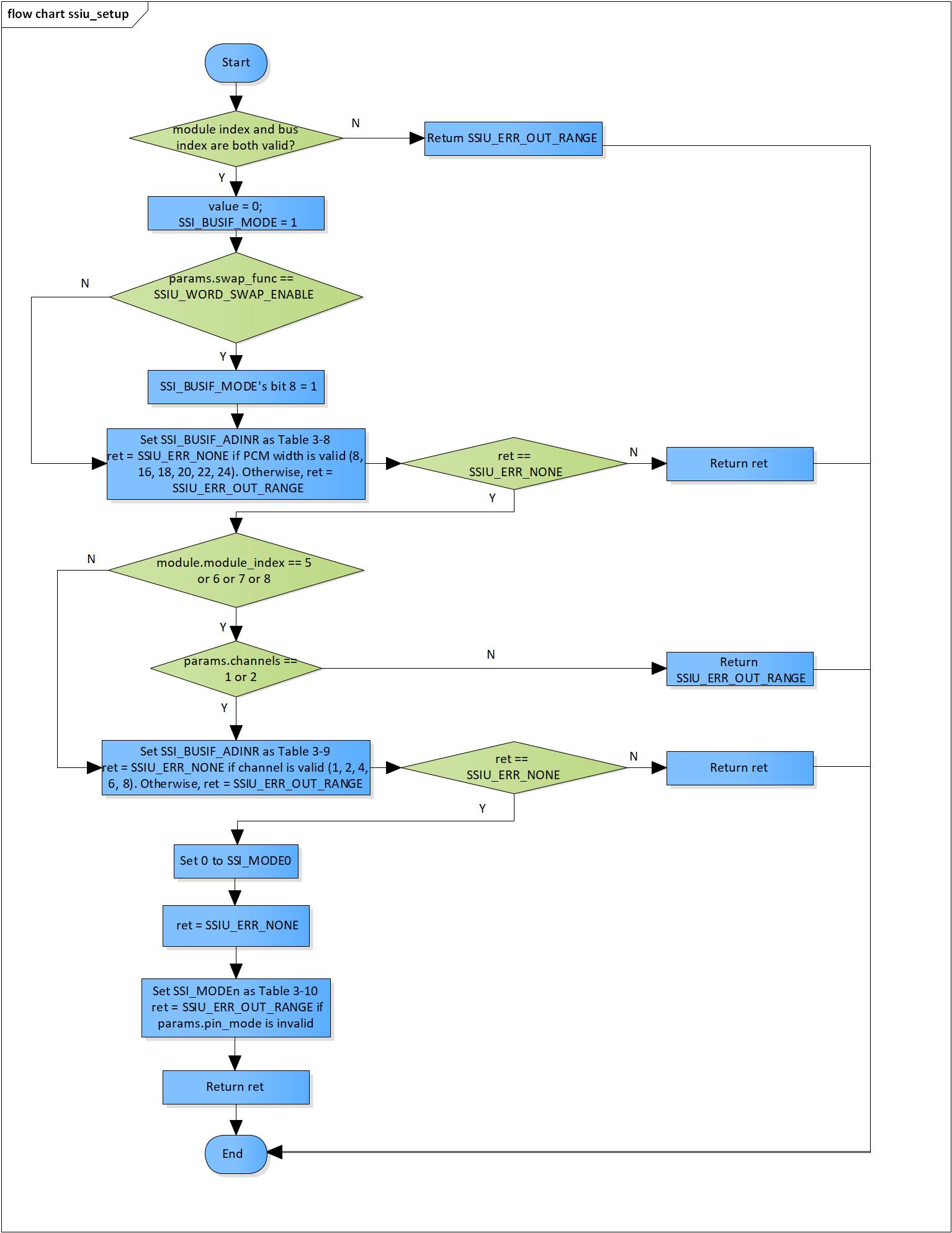
|  |  |
| --- | --- |
| **PCM width** | **SSI\_BUSIF\_ADINR[20:16]** |
| 8 | 0x10 |
| 16 | 0x08 |
| 18 | 0x06 |
| 20 | 0x04 |
| 22 | 0x02 |
| 24 | 0x00 |

Table 3‑9 Setting of SSI\_BUSIF\_ADINR based on channel

|  |  |
| --- | --- |
| **Channels** | **SSI\_BUSIF\_ADINR[4:0]** |
| 1 | 1 |
| 2 | 2 |
| 4 | 4 |
| 6 | 6 |
| 8 | 8 |

Table 3‑10 Setting pin mode

|  |  |  |  |
| --- | --- | --- | --- |
| **Pin mode** | **Register** | **Bits** | **Value to set** |
| SSIU\_S3\_S4\_PIN\_MODE | SSI\_MODE1 | [20:16] | 0x11 |
| SSIU\_M3\_S4\_PIN\_MODE | SSI\_MODE1 | [20:16] | 0x12 |
| SSIU\_S0\_S2\_PIN\_MODE | SSI\_MODE1 | [4:0] | 0x04 |
| SSIU\_M0\_S2\_PIN\_MODE | SSI\_MODE1 | [4:0] | 0x08 |
| SSIU\_S0\_S1\_PIN\_MODE | SSI\_MODE1 | [4:0] | 0x01 |
| SSIU\_M0\_S1\_PIN\_MODE | SSI\_MODE1 | [4:0] | 0x02 |
| SSIU\_S0\_S9\_PIN\_MODE | SSI\_MODE2 | [4:0] | 0x01 |
| SSIU\_M0\_S9\_PIN\_MODE | SSI\_MODE2 | [4:0] | 0x02 |
| SSIU\_S3\_S9\_PIN\_MODE | SSI\_MODE2 | [4:0] | 0x05 |
| SSIU\_S3\_S9\_PIN\_MODE | SSI\_MODE2 | [4:0] | 0x06 |
| SSIU\_M3\_S9\_PIN\_MODE | SSI\_MODE3 | [4:0] | 0x01 |
| SSIU\_S0\_S3\_PIN\_MODE | SSI\_MODE3 | [4:0] | 0x02 |

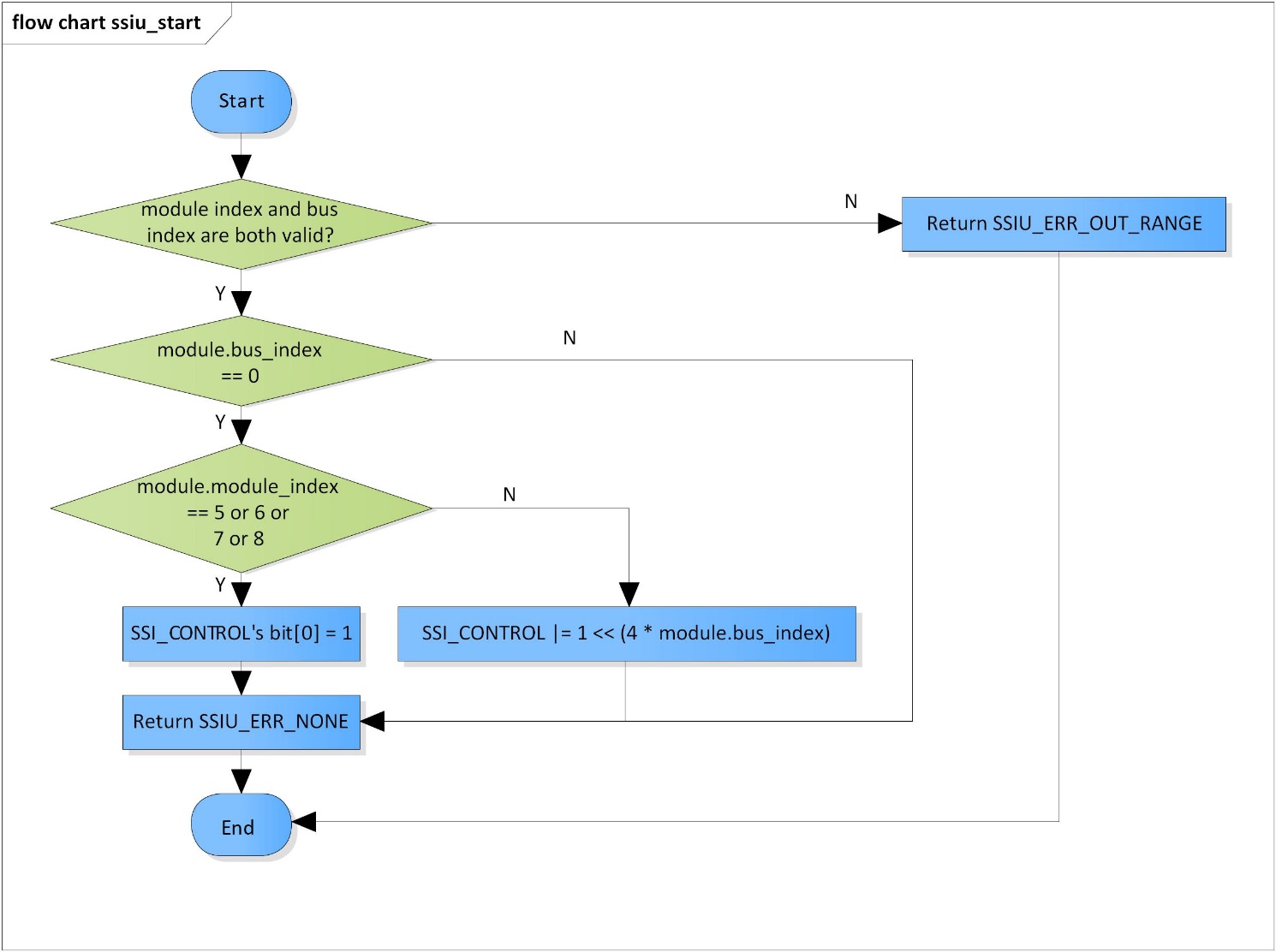
Figure 3‑1 ssiu\_setup flowchart

### ssiu\_start

DD\_PLG\_RDR\_04\_002

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | SSIU\_ERR\_CODE ssiu\_start(SSIU\_SSI\_MODULE module) | | | |
| **Function** | This function is to start SSIU module. | | | |
| **Arguments** | Type | Name | I/O | Description |
| SSIU\_SSI\_MODULE | module | I | SSIU module |
| **Return value** | SSIU\_ERR\_OUT\_RANGE | | Module index or bus index is invalid | |
| SSIU\_ERR\_NONE | | Normal end | |
| **Description** | * ssiu\_start command processing:   - Set SSI\_CONTROL register based on the bus index | | | |

[Covers: FD\_PLG\_RDR\_005]

Figure 3‑2 ssiu\_start flowchart

### ssiu\_stop

DD\_PLG\_RDR\_04\_003

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | SSIU\_ERR\_CODE ssiu\_stop(SSIU\_SSI\_MODULE module) | | | |
| **Function** | This function is to stop SSIU module. | | | |
| **Arguments** | Type | Name | I/O | Description |
| SSIU\_SSI\_MODULE | module | I | SSIU module |
| **Return value** | SSIU\_ERR\_OUT\_RANGE | | Module index or bus index is invalid | |
| SSIU\_ERR\_NONE | | Normal end | |
| **Description** | * ssiu\_stop command processing:   - Set SSI\_CONTROL register based on the bus index | | | |

[Covers: FD\_PLG\_RDR\_005, FD\_PLG\_RDR\_014, FD\_PLG\_RDR\_032, FD\_PLG\_RDR\_043]

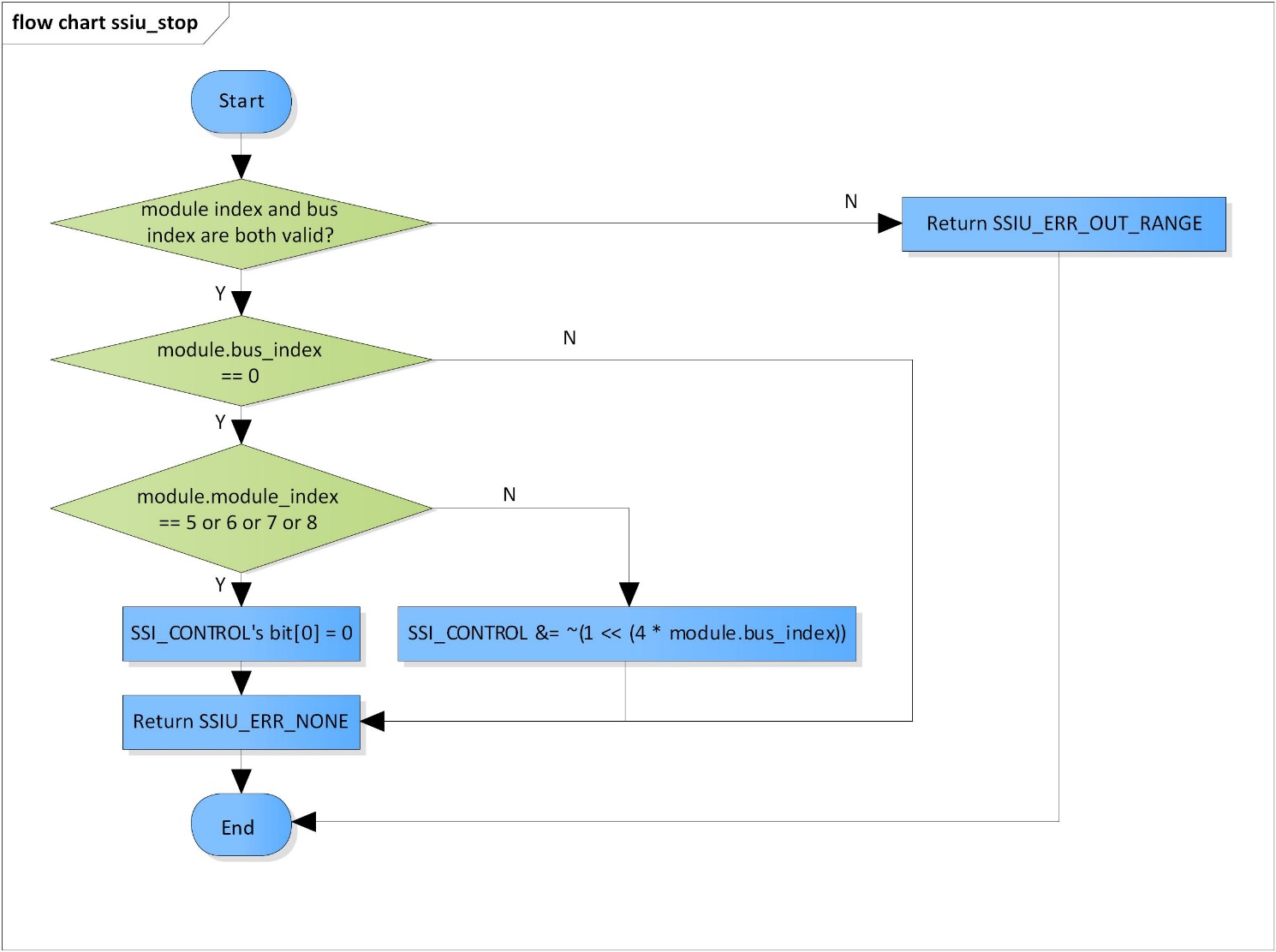


Figure 3‑3 ssiu\_stop flowchart

### ssiu\_get\_register\_base

DD\_PLG\_RDR\_04\_004

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static inline SSIU\_CONTROL \*ssiu\_get\_register\_base(SSIU\_SSI\_MODULE module) | | | |
| **Function** | This function is to get base address of registers for SSIU module. | | | |
| **Arguments** | Type | Name | I/O | Description |
| SSIU\_SSI\_MODULE | module | I | SSIU module |
| **Return value** | 0 | | Module index is invalid | |
| ssiu\_base | | Base address of registers for SSIU module | |
| **Description** | * ssiu\_get\_register\_base command processing:   - Get base address of registers for SSIU module | | | |

[Covers: FD\_PLG\_RDR\_005, FD\_PLG\_RDR\_014, FD\_PLG\_RDR\_032, FD\_PLG\_RDR\_043]

[Note]: XF\_RCAR\_REG\_SSI\_BUSIF\_MODE(n) is defined in xf-registers.h

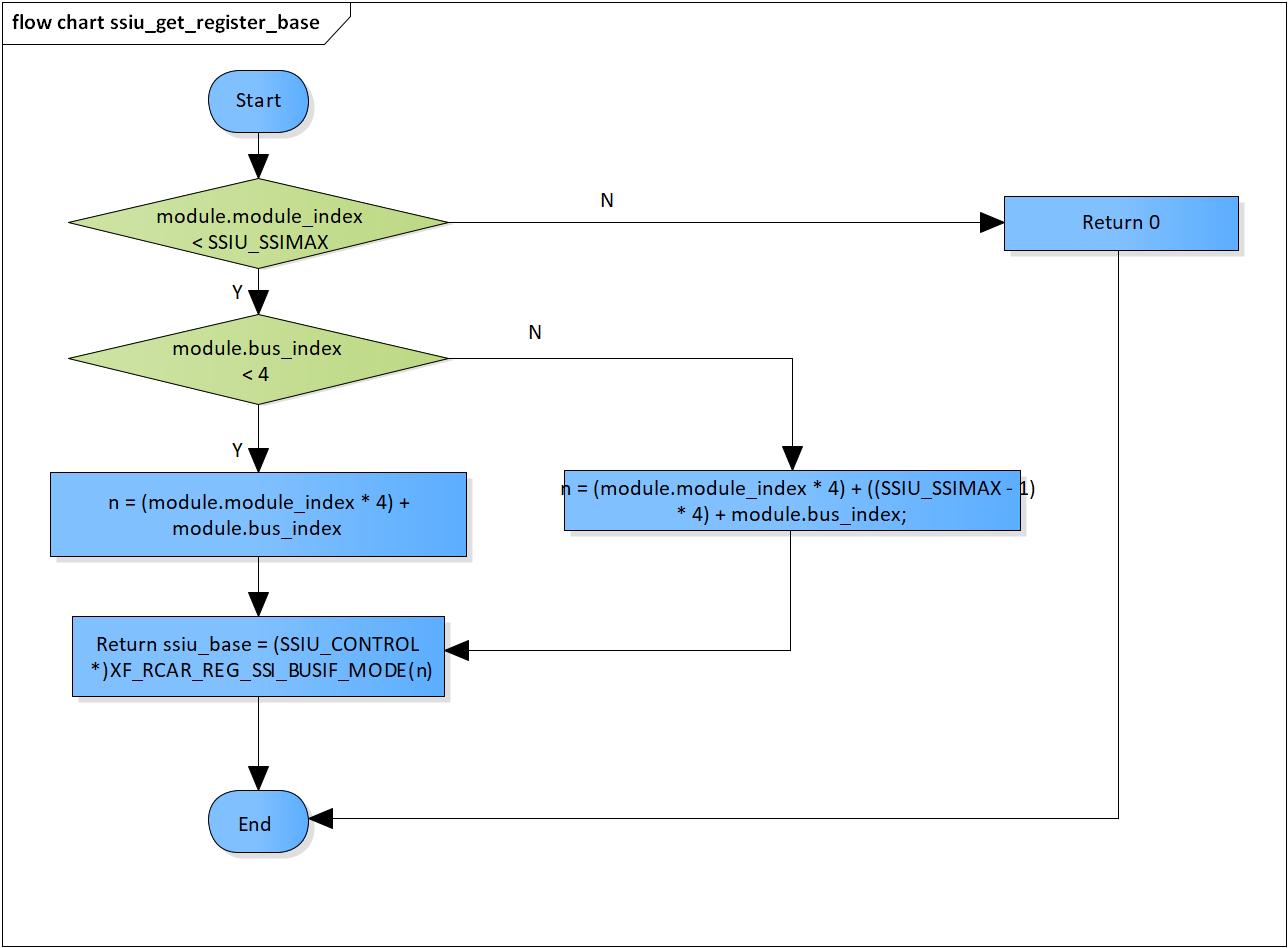


Figure 3‑4 ssiu\_get\_register\_base flowchart

# 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 | Ngu Pham |