**Document Type: Detail Design**

**Document Name:**

**ADSP FRAMEWORK: SSI DRIVER**

Renesas Electronics Corporation

Renesas Design Vietnam Co. Ltd

R-Car Software Solution 2 Group

Middleware 1 Team

Table of Contents

[1. Overview 3](#_Toc530075665)

[2. Function list 4](#_Toc530075666)

[3. Detail information 5](#_Toc530075667)

[3.1 Data type definition 5](#_Toc530075668)

[3.1.1 SSI\_MODULE 5](#_Toc530075669)

[3.1.2 ADG\_CLK 5](#_Toc530075670)

[3.1.3 SSI\_DATA\_MODE 6](#_Toc530075671)

[3.1.4 SSI\_CLK\_MODE 6](#_Toc530075672)

[3.1.5 SSI\_CLK\_MODE 6](#_Toc530075673)

[3.1.6 SSI\_ERR\_CODE 6](#_Toc530075674)

[3.1.7 SSI\_PARAMS 7](#_Toc530075675)

[3.2 Global variable definition 8](#_Toc530075676)

[3.3 Function definition 9](#_Toc530075677)

[3.3.1 xa\_ssi\_get\_register\_base 9](#_Toc530075678)

[3.3.2 xa\_ssi\_get\_module\_master 10](#_Toc530075679)

[3.3.3 xa\_ssi\_check\_valid\_module 11](#_Toc530075680)

[3.3.4 xa\_ssi\_setup 12](#_Toc530075681)

[3.3.5 ssi\_start 14](#_Toc530075682)

[3.3.6 xa\_ssi\_stop 15](#_Toc530075683)

[3.3.7 xa\_ssi\_check\_available\_module 16](#_Toc530075684)

[4. Revision history 17](#_Toc530075685)

List of Figures

[Figure 1‑1 The software architecture 3](#_Toc530075686)

[Figure 3‑1 xa\_ssi\_get\_register\_base flowchart 9](#_Toc530075687)

[Figure 3‑2 xa\_ssi\_get\_module\_master flowchart 10](#_Toc530075688)

[Figure 3‑3 xa\_ssi\_check\_valid\_module flowchart 11](#_Toc530075689)

[Figure 3‑4 xa\_ssi\_setup flowchart 13](#_Toc530075690)

[Figure 3‑5 xa\_ssi\_start flowchart 14](#_Toc530075691)

[Figure 3‑6 xa\_ssi\_stop flowchart 15](#_Toc530075692)

[Figure 3‑7 xa\_ssi\_check\_available\_module flowchart 16](#_Toc530075693)

List of Table

[Table 2‑1 Function list 4](#_Toc530075694)

[Table 3‑1 SSI\_MODULE type information 5](#_Toc530075695)

[Table 3‑2 ADG\_CLK type information 5](#_Toc530075696)

[Table 3‑3 SSI\_DATA\_MODE type information 6](#_Toc530075697)

[Table 3‑4 SSI\_CLK\_MODE type information 6](#_Toc530075698)

[Table 3‑5 SSI\_CLK\_MODE type information 6](#_Toc530075699)

[Table 3‑6 SSI\_ERR\_CODE type information 6](#_Toc530075700)

[Table 3‑7 SSI\_PARAMS type information 7](#_Toc530075701)

[Table 3‑8 Setting of clock supplier for SSI 12](#_Toc530075702)

[Table 2‑1 Function list 4](#_Toc534285937)

[Table 3‑1 SSI\_MODULE type information 5](#_Toc534285938)

[Table 3‑2 ADG\_CLK type information 5](#_Toc534285939)

[Table 3‑3 SSI\_DATA\_MODE type information 6](#_Toc534285940)

[Table 3‑4 SSI\_CLK\_MODE type information 6](#_Toc534285941)

[Table 3‑5 SSI\_CLK\_MODE type information 6](#_Toc534285942)

[Table 3‑6 SSI\_ERR\_CODE type information 6](#_Toc534285943)

[Table 3‑7 SSI\_PARAMS type information 7](#_Toc534285944)

[Table 3‑8 Setting of clock supplier for SSI 12](#_Toc534285945)

# 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

SCU

PDMA

FIFO

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\_ssi\_get\_module\_master | This API is to get the master module of SSI module |
| xa\_ssi\_check\_valid\_module | This API is to check validity of SSI module |
| xa\_ssi\_setup | This API is to set up registers necessary for SSI module execution |
| xa\_ssi\_start | This API is to start SSI module |
| xa\_ssi\_stop | This API is to stop SSI module |
| xa\_ssi\_check\_available\_module | This API is to check availability of SSI module |
| Internal functions | xa\_ssi\_get\_register\_base | This function is to get the address of the base register of SSI module |

# Detail information

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

## Data type definition

### SSI\_MODULE

The data type SSI\_MODULE is a type-defined enumeration that lists all supported SSI modules. It starts with SSI0 to SSIMAX.

Table 3‑1 SSI\_MODULE type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Value** | **Outline** |
| SSI0 | 0 | SSI0 |
| SSI1 | 1 | SSI1 |
| SSI2 | 2 | SSI2 |
| SSI3 | 3 | SSI3 |
| SSI4 | 4 | SSI4 |
| SSI5 | 5 | SSI5 |
| SSI6 | 6 | SSI6 |
| SSI7 | 7 | SSI7 |
| SSI8 | 8 | SSI8 |
| SSI9 | 9 | SSI9 |
| SSIMAX | 10 | Number of SSI modules |

### ADG\_CLK

The data type ADG\_CLK is a type-defined enumeration that lists all supported ADG Clocks.

Table 3‑2 ADG\_CLK type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Value** | **Outline** |
| ADG\_CLKA | 0 | ADG Clock A, frequency is 22.5792MHz |
| ADG\_CLKB | 1 | ADG Clock B, frequency is 24.576MHz |
| ADG\_CLKC | 2 | ADG Clock C, frequency is 14.7456MHz |

### SSI\_DATA\_MODE

The data type SSI\_DATA\_MODE is a type-defined enumeration that lists all supported SSI modes: Transfer mode/Receive mode.

Table 3‑3 SSI\_DATA\_MODE type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Value** | **Outline** |
| SSI\_RCV\_MODE | 0 | SSI’s Receive mode |
| SSI\_TR\_MODE | 1 | SSI’s Transfer mode |

### SSI\_CLK\_MODE

The data type SSI\_CLK\_MODE is a type-defined enumeration that lists all supported SSI clock modes: Master/Slave.

Table 3‑4 SSI\_CLK\_MODE type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Value** | **Outline** |
| SSI\_MASTER\_MODE | 0 | SSI clock mode Master |
| SSI\_SLAVE\_MODE | 1 | SSI clock mode Slave |

### SSI\_CLK\_MODE

The data type SSI\_CLK\_MODE is a type-defined enumeration that lists all supported SSI clock modes: Master/Slave.

Table 3‑5 SSI\_CLK\_MODE type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Value** | **Outline** |
| SSI\_MASTER\_MODE | 0 | SSI clock mode Master |
| SSI\_SLAVE\_MODE | 1 | SSI clock mode Slave |

### SSI\_ERR\_CODE

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

Table 3‑6 SSI\_ERR\_CODE type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Value** | **Outline** |
| SSI\_ERR\_NONE | 0 | No errors |
| SSI\_ERR\_OUT\_RANGE | -1 | SSI module is out of range |
| SSI\_ERR\_BUSY | -2 | SSI module is busy |

### SSI\_PARAMS

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

Table 3‑7 SSI\_PARAMS type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Data type** | **Outline** |
| channels | UWORD32 | channel of data used for SSI module |
| pcm\_wdth | UWORD32 | PCM width of data used for SSI module |
| fs | UWORD32 | Sampling rate of data used for SSI moduel |
| clock\_type | ADG\_CLK\_MODE | Clock type (Clock A, B, C) |
| data\_mode | SSI\_DATA\_MODE | SSI data mode |
| clock\_mode | SSI\_CLK\_MODE | SSI clock mode |
| ssi\_format | SSI\_FORMAT | SSI format |

## Global variable definition

|  |  |
| --- | --- |
| static const UWORD32 xa\_ssi\_register\_base[SSIMAX] | |
| Array’s index | Value |
| SSI0 | XF\_RCAR\_REG\_SSICR0 |
| SSI1 | XF\_RCAR\_REG\_SSICR1 |
| SSI2 | XF\_RCAR\_REG\_SSICR2 |
| SSI3 | XF\_RCAR\_REG\_SSICR3, |
| SSI4 | XF\_RCAR\_REG\_SSICR4 |
| SSI5 | XF\_RCAR\_REG\_SSICR5 |
| SSI6 | XF\_RCAR\_REG\_SSICR6 |
| SSI7 | XF\_RCAR\_REG\_SSICR7 |
| SSI8 | XF\_RCAR\_REG\_SSICR8 |
| SSI9 | XF\_RCAR\_REG\_SSICR9 |

## Function definition

### xa\_ssi\_get\_register\_base

DD\_PLG\_TDM\_07\_001

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static inline SSI\_CONTROL \*xa\_ssi\_get\_register\_base(SSI\_MODULE module) | | | |
| **Function** | This function is to get the address of the base register of SSI module. | | | |
| **Arguments** | Type | Name | I/O | Description |
| SSI\_MODULE | module | I | SSI module  Valid value: [0: 9] |
| **Return value** | 0 | | SSI module is invalid | |
| Address of base register | | | |
| **Description** | * xa\_ssi\_get\_register\_base command processing:   - Get address of base register from xa\_ssi\_register\_base array. | | | |

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

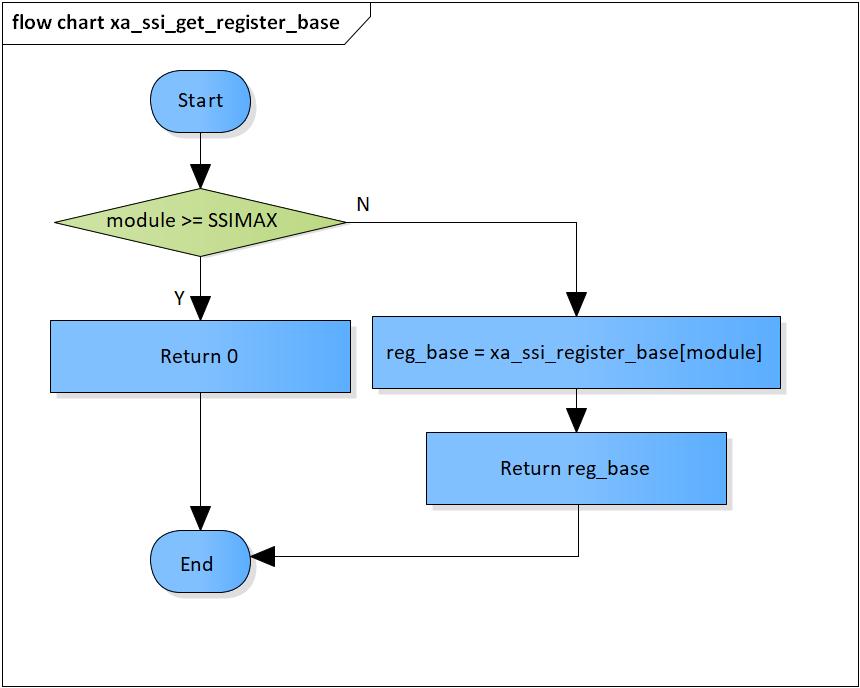


Figure 3‑1 xa\_ssi\_get\_register\_base flowchart

### xa\_ssi\_get\_module\_master

DD\_PLG\_TDM\_07\_002

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | SSI\_MODULE xa\_ssi\_get\_module\_master(SSI\_MODULE module) | | | |
| **Function** | This function is to get master module based on current SSI module. | | | |
| **Arguments** | Type | Name | I/O | Description |
| SSI\_MODULE | module | I | SSI module  Valid value: [0: 9] |
| **Return value** | master\_module | | Master module of current SSI module | |
| **Description** | * xa\_ssi\_get\_module\_master command processing:   - Get master module based on current SSI module | | | |

[Covers: FD\_PLG\_TDM\_005]

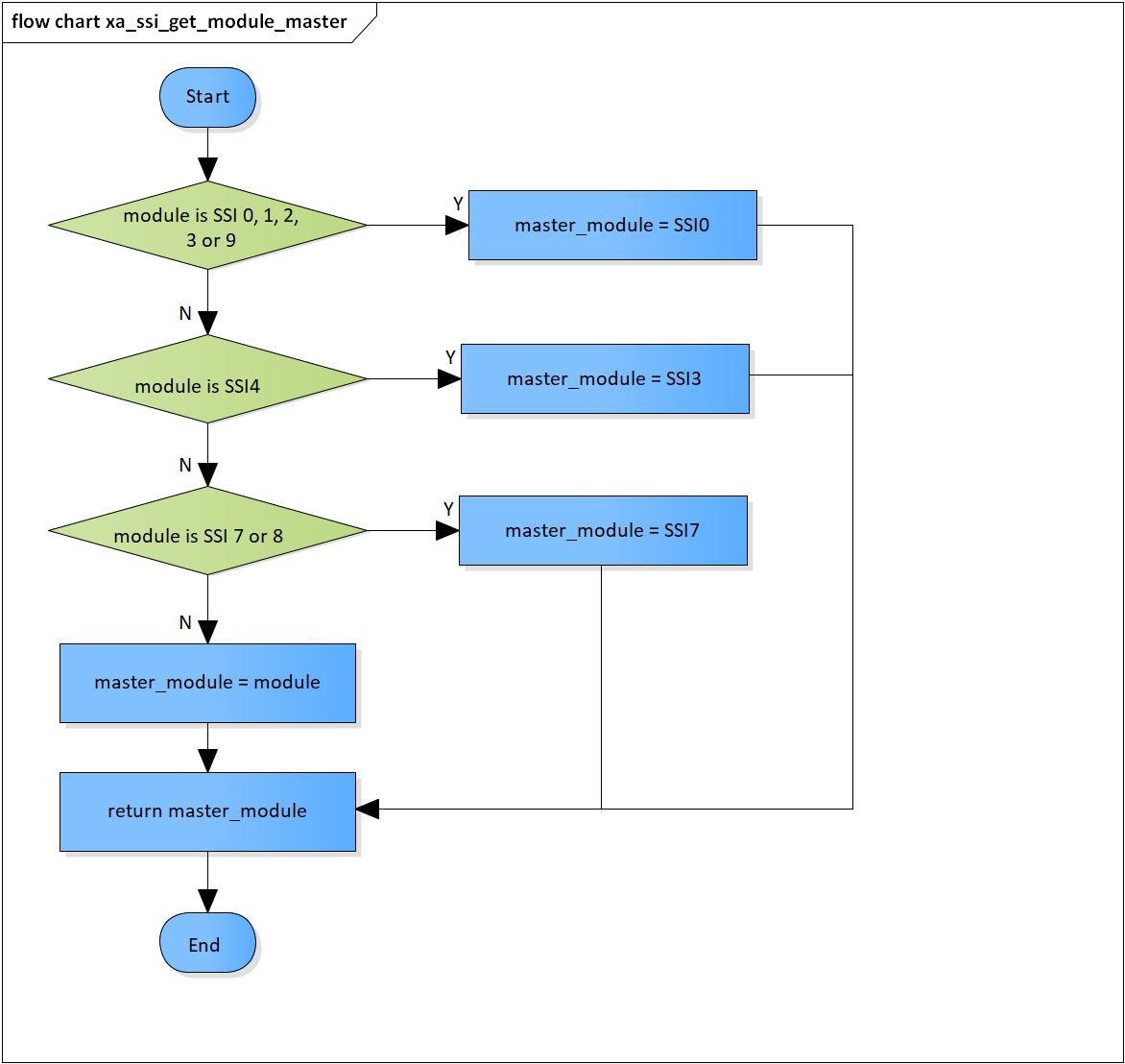


Figure 3‑2 xa\_ssi\_get\_module\_master flowchart

### xa\_ssi\_check\_valid\_module

DD\_PLG\_TDM\_07\_003

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | SSI\_ERROR\_CODE xa\_ssi\_check\_valid\_module(SSI\_MODULE module, SSI\_FORMAT format) | | | |
| **Function** | This function is to check validity of SSI module | | | |
| **Arguments** | Type | Name | I/O | Description |
| SSI\_MODULE | module | I | SSI module  Valid value: [0: 9] |
| SSI\_FORMAT | format | I | Format of the SSI module  Valid value: 0/1/2/3 |
| **Return value** | SSI\_ERR\_OUT\_RANGE | | SSI module is invalid | |
| SSI\_ERR\_NONE | | Normal end | |
| **Description** | * xa\_ssi\_check\_valid\_module command processing:   - Check validity of SSI module | | | |

[Covers: FD\_PLG\_TDM\_024, FD\_PLG\_TDM\_026, FD\_PLG\_TDM\_034, FD\_PLG\_TDM\_036]

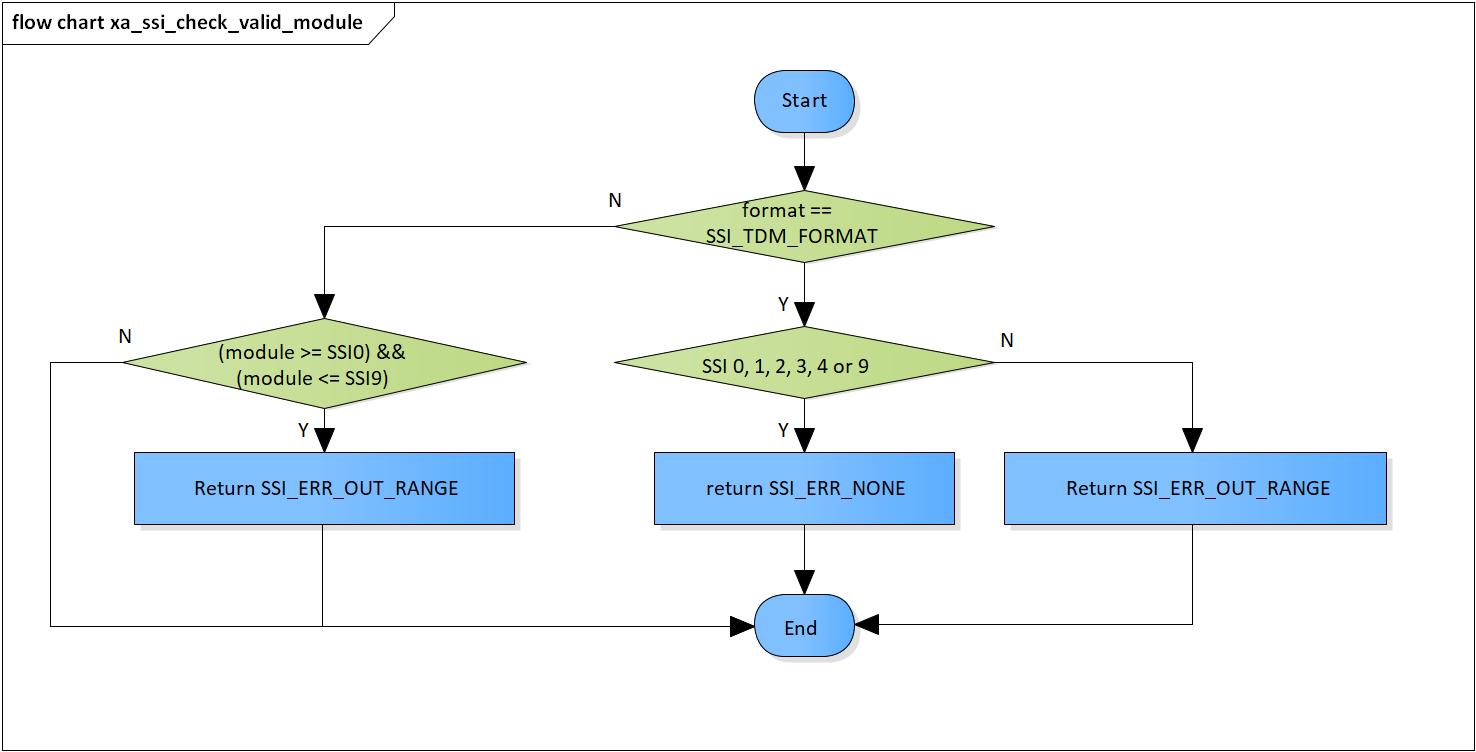


Figure 3‑3 xa\_ssi\_check\_valid\_module flowchart

### xa\_ssi\_setup

DD\_PLG\_TDM\_07\_004

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | SSI\_ERROR\_CODE xa\_ssi\_setup(SSI\_MODULE module, SSI\_PARAMS params) | | | |
| **Function** | This function is to set up registers necessary for SSI module execution. | | | |
| **Arguments** | Type | Name | I/O | Description |
| SSI\_MODULE | module | I | SSI module  Valid value: [0: 9] |
| SSI\_PARAMS | params | I | Struct of parameters to set |
| **Return value** | SSI\_ERR\_OUT\_RANGE | | SSI module is invalid  SSI format is invalid  Clock type is invalid  Channel is invalid  Clock ratio is invalid  Sampling rate is invalid  PCM width is invalid | |
| SSI\_ERR\_NONE | | Normal end | |
| **Description** | * xa\_ssi\_setup command processing:   - Set registers for SSI operation | | | |

[Covers: FD\_PLG\_TDM\_005]

Table 3‑8 Setting of clock supplier for SSI

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SSI module** | **Sampling rate** | **Register** | **Bits** | **Value** |
| SSI0 | 48000 | AUDIO\_CLK\_SEL0 | [7:0] | 0x2 |
| 44100 | 0x1 |
| SSI1 | 48000 | AUDIO\_CLK\_SEL0 | [15:8] | 0x2 |
| 44100 | 0x1 |
| SSI2 | 48000 | AUDIO\_CLK\_SEL0 | [23:16] | 0x2 |
| 44100 | 0x1 |
| SSI3 | 48000 | AUDIO\_CLK\_SEL0 | [31:24] | 0x2 |
| 44100 | 0x1 |
| SSI4 | 48000 | AUDIO\_CLK\_SEL1 | [7:0] | 0x2 |
| 44100 | 0x1 |
| SSI9 | 48000 | AUDIO\_CLK\_SEL2 | [15:8] | 0x2 |
| 44100 | 0x1 |

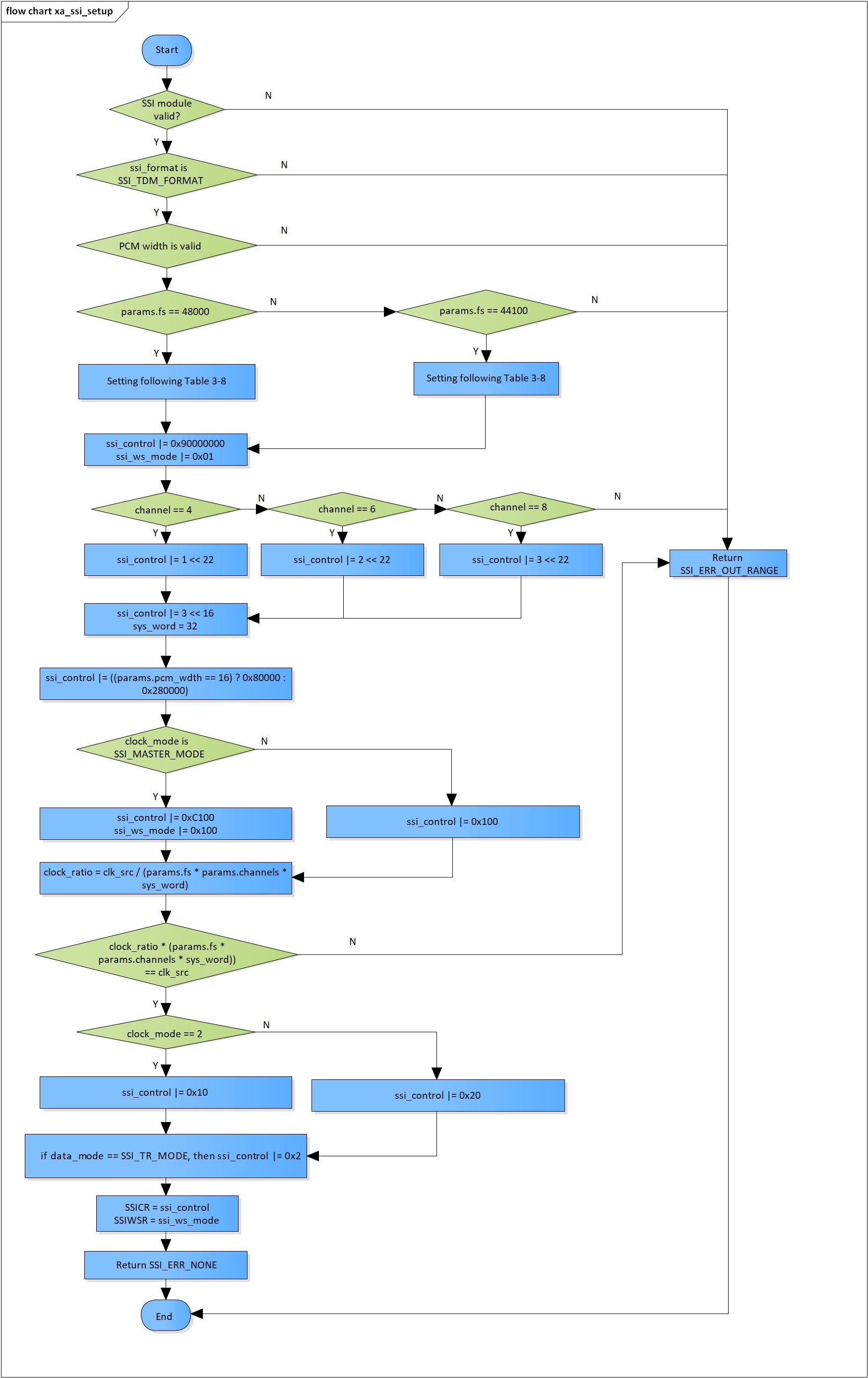


Figure 3‑4 xa\_ssi\_setup flowchart

### ssi\_start

DD\_PLG\_TDM\_07\_005

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | SSI\_ERR\_CODE xa\_ssi\_start(SSI\_MODULE module) | | | |
| **Function** | This function is to start SSI module. | | | |
| **Arguments** | Type | Name | I/O | Description |
| SSI\_MODULE | module | I | SSI module  Valid value: [0: 9] |
| **Return value** | SSI\_ERR\_BUSY | | SSI module is running | |
| SSI\_ERR\_NONE | | Normal end | |
| **Description** | * xa\_ssi\_start command processing:   - Set ‘1’ to SSICR’s bit 0 to start SSI | | | |

[Covers: FD\_PLG\_TDM\_005]

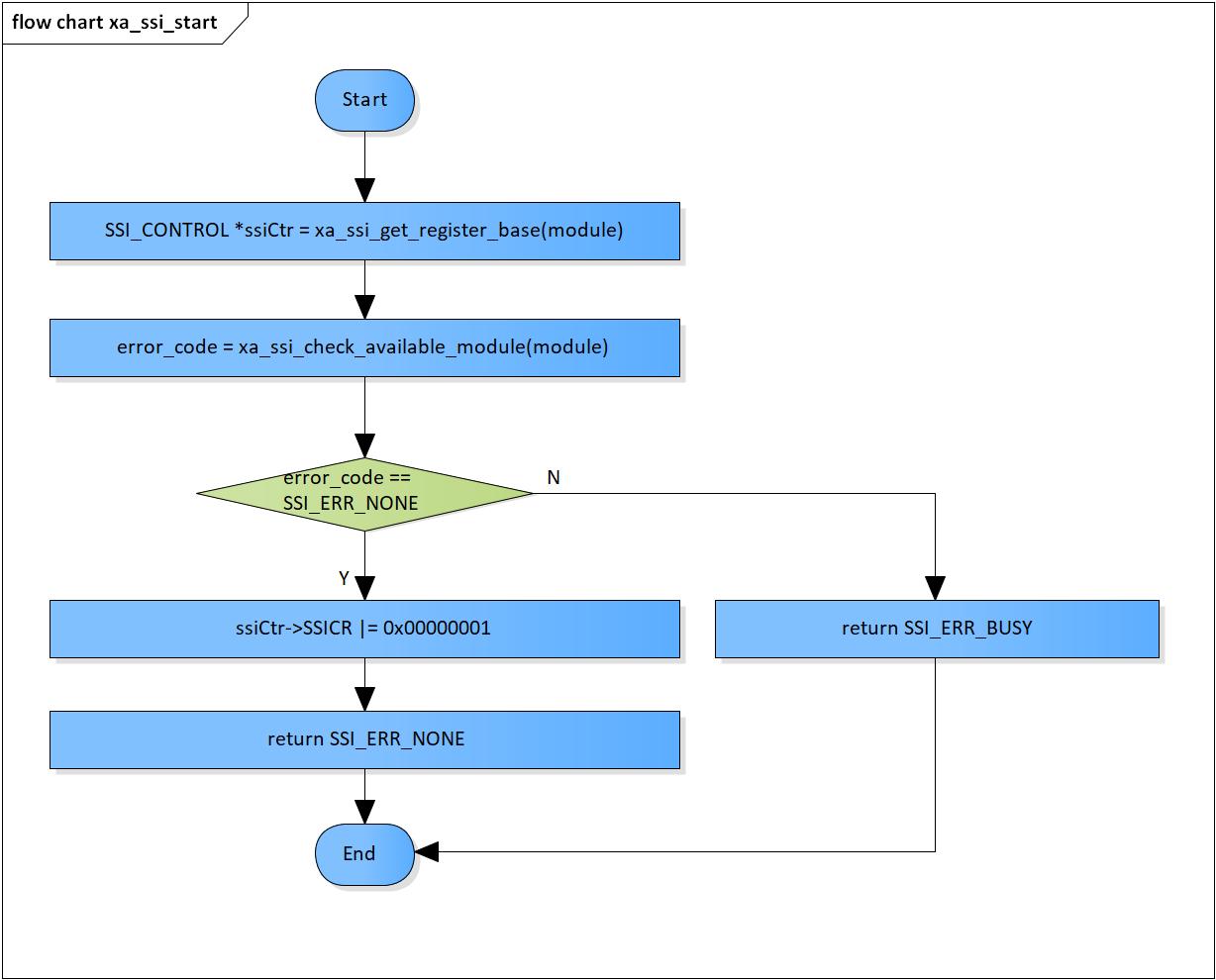


Figure 3‑5 xa\_ssi\_start flowchart

### xa\_ssi\_stop

DD\_PLG\_TDM\_07\_006

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | SSI\_ERR\_CODE xa\_ssi\_stop(SSI\_MODULE module) | | | |
| **Function** | This function is to stop SSI module. | | | |
| **Arguments** | Type | Name | I/O | Description |
| SSI\_MODULE | module | I | SSI module  Valid value: [0: 9] |
| **Return value** | SSI\_ERR\_OUT\_RANGE | | SSI module is invalid | |
| SSI\_ERR\_NONE | | Nomal end | |
| **Description** | * xa\_ssi\_start command processing:   - Set ‘0’ to SSICR’s bit 0 to stop SSI | | | |

[Covers: FD\_PLG\_TDM\_017]

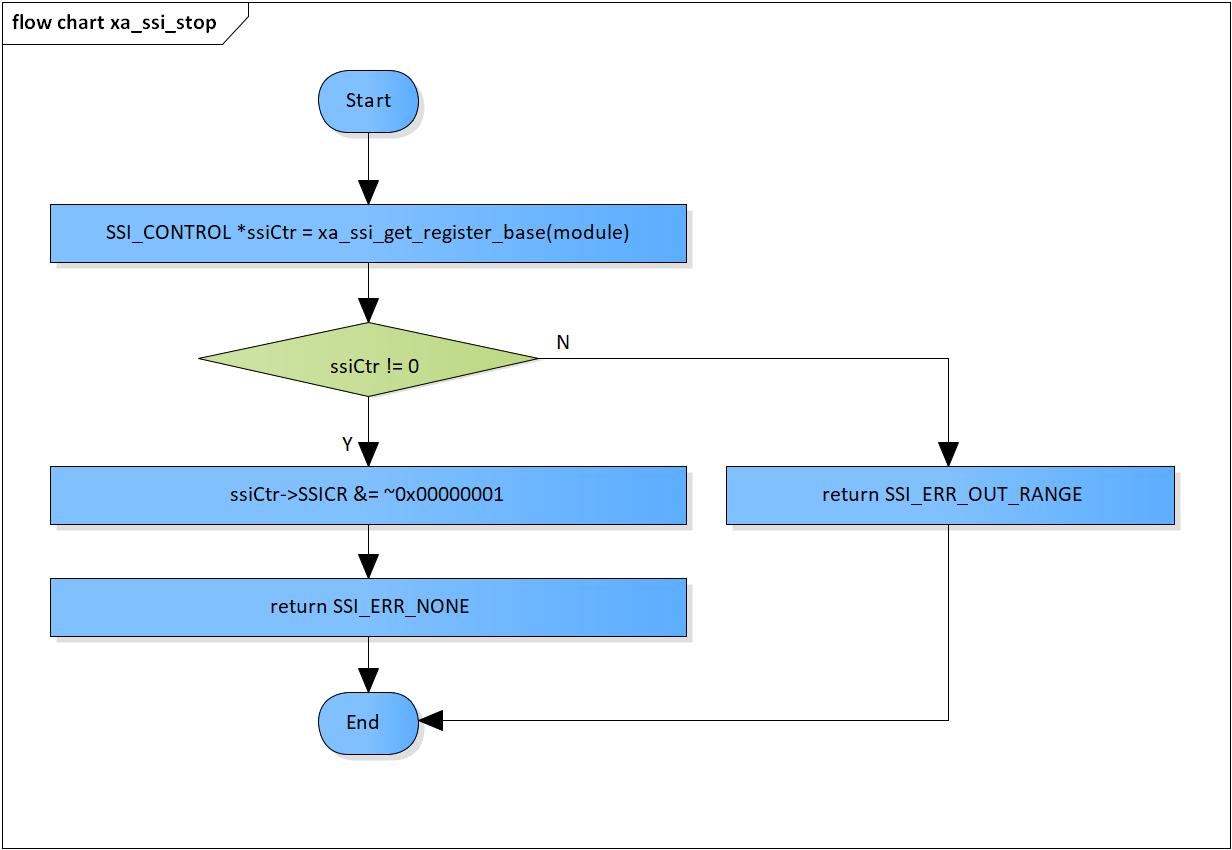


Figure 3‑6 xa\_ssi\_stop flowchart

### xa\_ssi\_check\_available\_module

DD\_PLG\_TDM\_07\_007

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | SSI\_ERROR\_CODE xa\_ssi\_check\_available\_module(SSI\_MODULE module\_idx) | | | |
| **Function** | This function is check availability of SSI module. | | | |
| **Arguments** | Type | Name | I/O | Description |
| UWORD32 | module\_idx | I | SSI module  Valid value: [0: 9] |
| **Return value** | SSI\_ERR\_OUT\_RANGE | | SSI module is invalid | |
| SSI\_ERR\_BUSY | | SSI module is busy | |
| SSI\_ERR\_NONE | | Normal end | |
| **Description** | * xa\_ssi\_check\_available\_module command processing:   - Check if the current SSI module is running or not | | | |

[Covers: FD\_PLG\_TDM\_005]

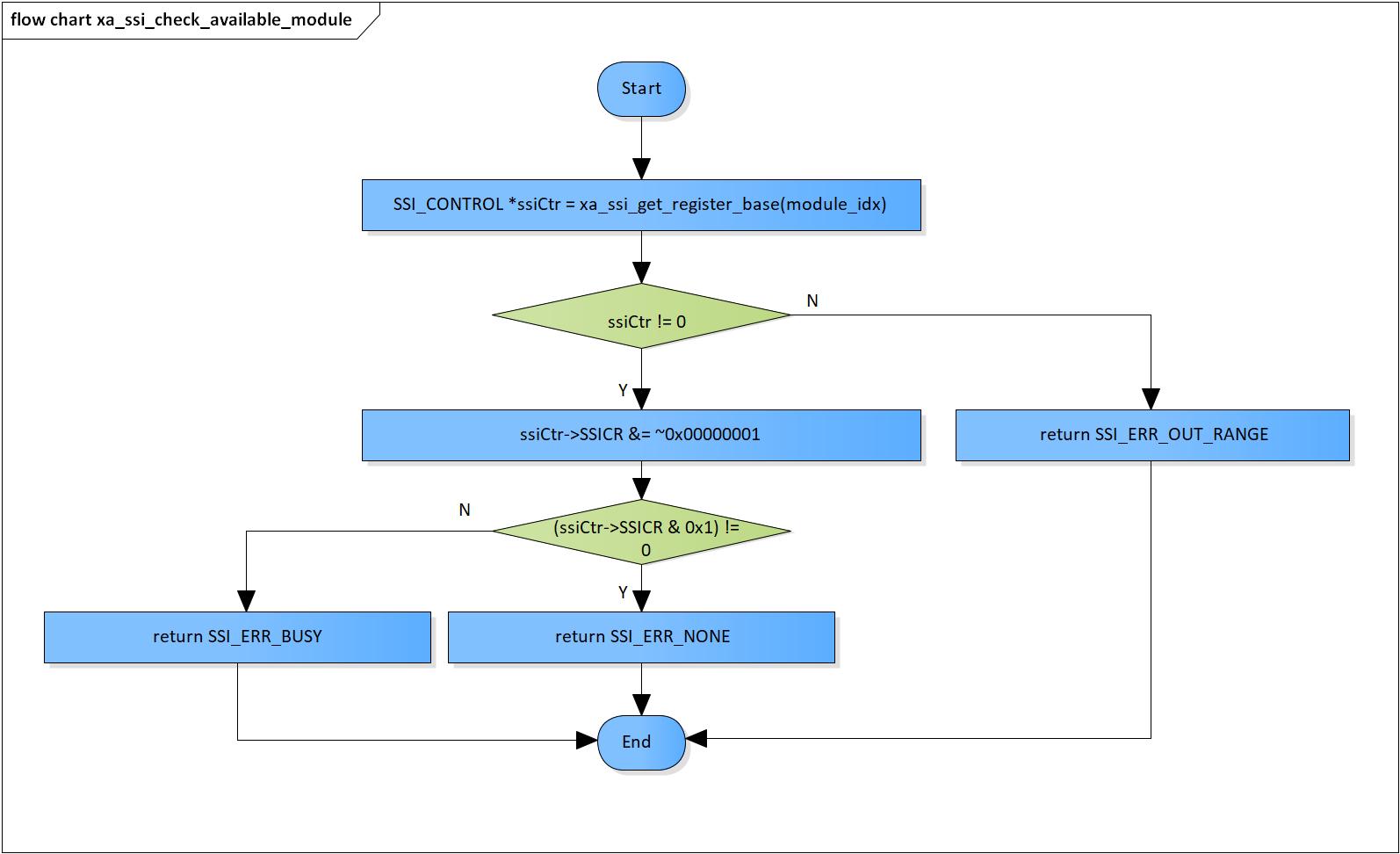


Figure 3‑7 xa\_ssi\_check\_available\_module flowchart

# 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 | Nguyen Dang |
| 1.2.0 | Jan 03, 2019 | - | Add range for input parameters | Vu Phan | Tien Tran |