**Document Type: Detail Design**

**Document Name:**

**ADSP FRAMEWORK: ADMA DRIVER**

Renesas Electronics Corporation

Renesas Design Vietnam Co. Ltd

R-Car Software Solution 2 Group

Middleware 1 Team

Table of Contents

[1. Overview 3](#_Toc534275273)

[2. Function list 4](#_Toc534275274)

[3. Detail information 5](#_Toc534275275)

[3.1 Data type definition 5](#_Toc534275276)

[3.1.1 ADMA\_CH 5](#_Toc534275277)

[3.1.2 dmac\_error\_code 6](#_Toc534275278)

[3.1.3 ADMA\_STAGE 6](#_Toc534275279)

[3.1.4 DMAC\_SRCDST 7](#_Toc534275280)

[3.1.5 ADMAC\_MODE 10](#_Toc534275281)

[3.1.6 DMAC\_SETTING 10](#_Toc534275282)

[3.1.7 DMAC\_AUDIO\_MODULE\_SETTING 11](#_Toc534275283)

[3.2 Macro definition 12](#_Toc534275284)

[3.3 Register definition 14](#_Toc534275285)

[3.4 Global variable definition 15](#_Toc534275286)

[3.5 Function definition 18](#_Toc534275287)

[3.5.1 admac\_get\_register\_addr 18](#_Toc534275288)

[3.5.2 admac\_set\_stage\_number 20](#_Toc534275289)

[3.5.3 admac\_reset\_descriptor 21](#_Toc534275290)

[3.5.4 admac\_start\_descriptor 22](#_Toc534275291)

[3.5.5 admac\_enable\_ch 23](#_Toc534275292)

[3.5.6 admac\_disable\_ch 24](#_Toc534275293)

[3.5.7 admac\_enable\_module 25](#_Toc534275294)

[3.5.8 admac\_clear\_channel 26](#_Toc534275295)

[3.5.9 admac\_check\_stage 27](#_Toc534275296)

[3.5.10 admac\_check\_and\_clear\_interrupt 28](#_Toc534275297)

[3.5.11 admac\_start 29](#_Toc534275298)

[3.5.12 admac\_stop 30](#_Toc534275299)

[3.5.13 admac\_descriptor\_setup 31](#_Toc534275300)

[3.5.14 admac\_stage\_config 32](#_Toc534275301)

[3.5.15 admac\_pre\_config 33](#_Toc534275302)

[3.5.16 admac\_post\_config 35](#_Toc534275303)

[3.5.17 admac\_check\_valid\_channel 37](#_Toc534275304)

[3.5.18 admac\_setting\_use\_descriptor 39](#_Toc534275305)

[3.5.19 admac\_setting\_nouse\_descriptor 40](#_Toc534275306)

[3.5.20 admac\_resetup 42](#_Toc534275307)

[3.5.21 admac\_pause 43](#_Toc534275308)

[4. Revision history 44](#_Toc534275309)

List of Figures

[Figure 1‑1 The software architecture 3](#_Toc529534679)

[Figure 3‑1 admac\_get\_register\_addr flowchart 19](#_Toc529534680)

[Figure 3‑2 admac\_set\_stage\_number flowchart 20](#_Toc529534681)

[Figure 3‑3 admac\_reset\_descriptor flowchart 21](#_Toc529534682)

[Figure 3‑4 admac\_start\_desciptor flowchart 22](#_Toc529534683)

[Figure 3‑5 admac\_enable\_ch flowchart 23](#_Toc529534684)

[Figure 3‑6 admac\_disable\_ch flowchart 24](#_Toc529534685)

[Figure 3‑7 admac\_enable\_module flowchart 25](#_Toc529534686)

[Figure 3‑8 admac\_clear\_channel flowchart 26](#_Toc529534687)

[Figure 3‑9 admac\_check\_stage flowchart 27](#_Toc529534688)

[Figure 3‑10 admac\_check\_and\_clear\_interrupt flowchart 28](#_Toc529534689)

[Figure 3‑11admac\_start flowchart 29](#_Toc529534690)

[Figure 3‑12 admac\_stop flowchart 30](#_Toc529534691)

[Figure 3‑13 admac\_descriptor\_setup flowchart 31](#_Toc529534692)

[Figure 3‑14 admac\_stage\_config flowchart 32](#_Toc529534693)

[Figure 3‑15 admac\_pre\_config flowchart 34](#_Toc529534694)

[Figure 3‑16 admac\_post\_config flowchart 35](#_Toc529534695)

[Figure 3‑17 admac\_check\_valid\_channel flowchart 38](#_Toc529534696)

[Figure 3‑18 admac\_setting\_use\_descriptor flowchart 39](#_Toc529534697)

[Figure 3‑19 admac\_setting\_nouse\_descriptor flowchart 41](#_Toc529534698)

[Figure 3‑20 admac\_resetup flowchart 42](#_Toc529534699)

[Figure 3‑21 admac\_pause flowchart 43](#_Toc529534700)

List of Table

[Table 2‑1 Function list 4](#_Toc528330044)

[Table 3‑1 ADMA\_CH type information 5](#_Toc528330045)

[Table 3‑2 dmac\_error\_code type information 6](#_Toc528330046)

[Table 3‑3 ADMA\_STAGE type information 6](#_Toc528330047)

[Table 3‑4 DMAC\_SRCDST type information 7](#_Toc528330048)

[Table 3‑5 ADMAC\_MODE type information 10](#_Toc528330049)

[Table 3‑6 DMAC\_SETTING type information 10](#_Toc528330050)

[Table 3‑7 DMAC\_AUDIO\_MODULE\_SETTING type information 11](#_Toc528330051)

[Table 2‑1 Function list 4](#_Toc534275310)

[Table 3‑1 ADMA\_CH type information 5](#_Toc534275311)

[Table 3‑2 dmac\_error\_code type information 6](#_Toc534275312)

[Table 3‑3 ADMA\_STAGE type information 6](#_Toc534275313)

[Table 3‑4 DMAC\_SRCDST type information 7](#_Toc534275314)

[Table 3‑5 ADMAC\_MODE type information 10](#_Toc534275315)

[Table 3‑6 DMAC\_SETTING type information 10](#_Toc534275316)

[Table 3‑7 DMAC\_AUDIO\_MODULE\_SETTING type information 11](#_Toc534275317)

# 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 | admac\_get\_register\_addr | This API is to get DMAC channel N register’s address |
| admac\_check\_stage | This API is to check the current stage of DMAC module |
| admac\_check\_and\_clear\_interrupt | This API is to check and clear interrupt of DMAC channel N |
| admac\_start | This API is to start DMAC channel N |
| admac\_stop | This API is to stop DMAC channel N |
| admac\_check\_valid\_channel | This API is to check availability of DMAC module and choose an available one |
| admac\_setting\_use\_descriptor | This API is to set DMAC module using descriptor memory |
| admac\_setting\_nouse\_descriptor | This API is to set DMAC module not using descriptor memory |
| admac\_resetup | This API is to re-setup DMAC module |
| admac\_pause | This API is to pause ADMAC channel N |
| Internal function | admac\_set\_stage\_number | This function is to set stage number |
| admac\_reset\_descriptor | This function is to reset descriptor |
| admac\_start\_descriptor | This function is to start descriptor |
| admac\_enable\_ch | This function is to enable a channel |
| admac\_disable\_ch | This function is to disable a channel |
| admac\_enable\_module | This function is to enable DMAC module |
| admac\_clear\_channel | This function is to clear value set in DMAC channel N register |
| admac\_descriptor\_setup | This function is to set up DMAC descriptor |
| admac\_stage\_config | This function is to set DMAC stage |
| admac\_pre\_config | This function is to set DMAC parameters |
| admac\_post\_config | This function is to post-set DMAC parameters |

# Detail information

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

## Data type definition

### ADMA\_CH

The data type ADMA\_CH is a type-defined enumeration that lists all supported ADMAC channels.

Table 3‑1 ADMA\_CH type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Value** | **Outline** |
| ADMAC\_CH00 | 0 | ADMAC channel 0 |
| ADMAC\_CH01 | 1 | ADMAC channel 1 |
| ADMAC\_CH02 | 2 | ADMAC channel 2 |
| ADMAC\_CH03 | 3 | ADMAC channel 3 |
| ADMAC\_CH04 | 4 | ADMAC channel 4 |
| ADMAC\_CH05 | 5 | ADMAC channel 5 |
| ADMAC\_CH06 | 6 | ADMAC channel 6 |
| ADMAC\_CH07 | 7 | ADMAC channel 7 |
| ADMAC\_CH08 | 8 | ADMAC channel 8 |
| ADMAC\_CH09 | 9 | ADMAC channel 9 |
| ADMAC\_CH10 | 10 | ADMAC channel 10 |
| ADMAC\_CH11 | 11 | ADMAC channel 11 |
| ADMAC\_CH12 | 12 | ADMAC channel 12 |
| ADMAC\_CH13 | 13 | ADMAC channel 13 |
| ADMAC\_CH14 | 14 | ADMAC channel 14 |
| ADMAC\_CH15 | 15 | ADMAC channel 15 |
| ADMAC\_CH16 | 16 | ADMAC channel 16 |
| ADMAC\_CH17 | 17 | ADMAC channel 17 |
| ADMAC\_CH18 | 18 | ADMAC channel 18 |
| ADMAC\_CH19 | 19 | ADMAC channel 19 |
| ADMAC\_CH20 | 20 | ADMAC channel 20 |
| ADMAC\_CH21 | 21 | ADMAC channel 21 |
| ADMAC\_CH22 | 22 | ADMAC channel 22 |
| ADMAC\_CH23 | 23 | ADMAC channel 23 |
| ADMAC\_CH24 | 24 | ADMAC channel 24 |
| ADMAC\_CH25 | 25 | ADMAC channel 25 |
| ADMAC\_CH26 | 26 | ADMAC channel 26 |
| ADMAC\_CH27 | 27 | ADMAC channel 27 |
| ADMAC\_CH28 | 28 | ADMAC channel 28 |
| ADMAC\_CH29 | 29 | ADMAC channel 29 |
| ADMAC\_CH30 | 30 | ADMAC channel 30 |
| ADMAC\_CH31 | 31 | ADMAC channel 31 |
| ADMAC\_CHMAX | 32 | Total number of ADMAC channels |

### dmac\_error\_code

The data type dmac\_error\_code is a type-defined enumeration that lists all error codes for ADMAC.

Table 3‑2 dmac\_error\_code type information

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

### ADMA\_STAGE

The data type ADMA\_STAGE is a type-defined enumeration of stages of ADMAC.

Table 3‑3 ADMA\_STAGE type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Value** | **Outline** |
| ADMAC\_STAGE0 | 0 | ADMAC stage 0 |
| ADMAC\_STAGE1 | 1 | ADMAC stage 1 |
| ADMAC\_STAGEMAX | 2 | Total stages of ADMAC |

### DMAC\_SRCDST

The data type DMAC\_SRCDST is a type-defined enumeration that shows connections between ADMAC and other devices.

Table 3‑4 DMAC\_SRCDST type information

|  |  |
| --- | --- |
| **Member name** | **Value** |
| DMAC\_SSI00 | DMAC\_SSI\_MODULE\_MIN |
| DMAC\_SSI01 | 1 |
| DMAC\_SSI02 | 2 |
| DMAC\_SSI03 | 3 |
| DMAC\_SSI04 | 4 |
| DMAC\_SSI05 | 5 |
| DMAC\_SSI06 | 6 |
| DMAC\_SSI07 | 7 |
| DMAC\_SSI10 | 10 |
| DMAC\_SSI11 | 11 |
| DMAC\_SSI12 | 12 |
| DMAC\_SSI13 | 13 |
| DMAC\_SSI14 | 14 |
| DMAC\_SSI15 | 15 |
| DMAC\_SSI16 | 16 |
| DMAC\_SSI17 | 17 |
| DMAC\_SSI20 | 20 |
| DMAC\_SSI21 | 21 |
| DMAC\_SSI22 | 22 |
| DMAC\_SSI23 | 23 |
| DMAC\_SSI24 | 24 |
| DMAC\_SSI25 | 25 |
| DMAC\_SSI26 | 26 |
| DMAC\_SSI27 | 27 |
| DMAC\_SSI30 | 30 |
| DMAC\_SSI31 | 31 |
| DMAC\_SSI32 | 32 |
| DMAC\_SSI33 | 33 |
| DMAC\_SSI34 | 34 |
| DMAC\_SSI35 | 35 |
| DMAC\_SSI36 | 36 |
| DMAC\_SSI37 | 37 |
| DMAC\_SSI40 | 40 |
| DMAC\_SSI41 | 41 |
| DMAC\_SSI42 | 42 |
| DMAC\_SSI43 | 43 |
| DMAC\_SSI44 | 44 |
| DMAC\_SSI45 | 45 |
| DMAC\_SSI46 | 46 |
| DMAC\_SSI47 | 47 |
| DMAC\_SSI5 | 50 |
| DMAC\_SSI6 | 60 |
| DMAC\_SSI7 | 70 |
| DMAC\_SSI8 | 80 |
| DMAC\_SSI90 | 90 |
| DMAC\_SSI91 | 91 |
| DMAC\_SSI92 | 92 |
| DMAC\_SSI93 | 93 |
| DMAC\_SSI94 | 94 |
| DMAC\_SSI95 | 95 |
| DMAC\_SSI96 | 96 |
| DMAC\_SSI97 | DMAC\_SSI\_MODULE\_MAX |
| DMAC\_DTCPPP0 | 98 |
| DMAC\_DTCPPP1 | 99 |
| DMAC\_DTCPCP0 | 100 |
| DMAC\_DTCPCP1 | 101 |
| DMAC\_SCU\_SRCI0 | DMAC\_SCU\_SRC\_INPUT\_MODULE\_MIN |
| DMAC\_SCU\_SRCI1 | 111 |
| DMAC\_SCU\_SRCI2 | 112 |
| DMAC\_SCU\_SRCI3 | 113 |
| DMAC\_SCU\_SRCI4 | 114 |
| DMAC\_SCU\_SRCI5 | 115 |
| DMAC\_SCU\_SRCI6 | 116 |
| DMAC\_SCU\_SRCI7 | 117 |
| DMAC\_SCU\_SRCI8 | 118 |
| DMAC\_SCU\_SRCI9 | DMAC\_SCU\_SRC\_INPUT\_MODULE\_MAX |
| DMAC\_SCU\_SRCO0 | DMAC\_SCU\_SRC\_OUTPUT\_MODULE\_MIN |
| DMAC\_SCU\_SRCO1 | 121 |
| DMAC\_SCU\_SRCO2 | 122 |
| DMAC\_SCU\_SRCO3 | 123 |
| DMAC\_SCU\_SRCO4 | 124 |
| DMAC\_SCU\_SRCO5 | 125 |
| DMAC\_SCU\_SRCO6 | 126 |
| DMAC\_SCU\_SRCO7 | 127 |
| DMAC\_SCU\_SRCO8 | 128 |
| DMAC\_SCU\_SRCO9 | DMAC\_SCU\_SRC\_OUTPUT\_MODULE\_MAX |
| DMAC\_SCU\_CMD0 | DMAC\_SCU\_CMD\_MODULE\_MIN |
| DMAC\_SCU\_CMD1 | DMAC\_SCU\_CMD\_MODULE\_MAX |
| DMAC\_MLM0 | 132 |
| DMAC\_MLM1 | 133 |
| DMAC\_MLM2 | 134 |
| DMAC\_MLM3 | 135 |
| DMAC\_MLM4 | 136 |
| DMAC\_MLM5 | 137 |
| DMAC\_MLM6 | 138 |
| DMAC\_MLM7 | 139 |
| DMAC\_NONCONTROL | 140 |
| DMAC\_MAX | 141 |

### ADMAC\_MODE

The data type ADMAC\_MODE is a type-defined enumeration of ADMAC operation mode.

Table 3‑5 ADMAC\_MODE type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Value** | **Outline** |
| ADMAC\_NORMAL\_MODE | 1 | ADMAC normal mode |
| ADMAC\_REPEAT\_MODE | 2 | ADMAC repeating mode |

### DMAC\_SETTING

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

Table 3‑6 DMAC\_SETTING type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Data type** | **Outline** |
| dst | WORD32 | Destination address |
| src | WORD32 | Source address |
| transmem | pVOID | Transmission memory address |
| trans\_num | WORD32 | Transfer number |
| stg\_num | WORD32 | Stage number |
| stg\_size | WORD32 | Stage memory size |
| buf\_size | WORD32 | Ring buffer size |
| trans\_mode | WORD32 | Indicator of transfer/receive mode |
| mode | ADMAC\_MODE | Operation mode |
| dma\_ch | UWORD32 | ADMAC channel |
| src\_reqst | UWORD32 | Source request |
| trans\_size | UWORD32 | Transfer size |

### DMAC\_AUDIO\_MODULE\_SETTING

The data type DMAC\_AUDIO\_MODULE\_SETTING is a type-defined structure of ADMAC device information.

Table 3‑7 DMAC\_AUDIO\_MODULE\_SETTING type information

|  |  |  |
| --- | --- | --- |
| **Member name** | **Data type** | **Outline** |
| module\_address | UWORD32 | Destination address |
| trans | UWORD32 | Source address |
| recei | UWORD32 | Transmission memory address |

## Macro definition

|  |  |  |
| --- | --- | --- |
| **Macros** | **Value** | **Outline** |
| DMAC\_SSI\_MODULE\_MIN | 0 | SSI module min index |
| DMAC\_SSI\_MODULE\_MAX | 97 | SSI module max index |
| DMAC\_SCU\_SRC\_INPUT\_MODULE\_MIN | 110 | SCU input module min index |
| DMAC\_SCU\_SRC\_INPUT\_MODULE\_MAX | 119 | SCU input module max index |
| DMAC\_SCU\_SRC\_OUTPUT\_MODULE\_MIN | 120 | SCU output module min index |
| DMAC\_SCU\_SRC\_OUTPUT\_MODULE\_MAX | 129 | SCU output module max index |
| DMAC\_SCU\_CMD\_MODULE\_MIN | 130 | CMD module min index |
| DMAC\_SCU\_CMD\_MODULE\_MAX | 131 | CMD module max index |
| XF\_RCAR\_DMAOR\_L | XF\_RCAR\_REG\_DMAOR\_L | DMA operation register (for lower-numbered channels) |
| XF\_RCAR\_DMAOR\_U | XF\_RCAR\_REG\_DMAOR\_U | DMA channel clear register (for lower-numbered channels) |
| XF\_DMACHCLR\_L | XF\_RCAR\_REG\_DMACHCLR\_L | DMA channel clear register (for lower-numbered channels) |
| XF\_DMACHCLR\_U | XF\_RCAR\_REG\_DMACHCLR\_U | DMA channel clear register (for higher-numbered channels) |
| XF\_RCAR\_DMARS\_0 | XF\_RCAR\_REG\_DMARS\_0 | Channel 0 DMA Extended Resource Selectors Register |
| XF\_RCAR\_DMADPBASE\_0 | XF\_RCAR\_REG\_DMADPBASE\_0 | Channel 0 DMA Descriptor Base Address Register |
| XF\_RCAR\_DMACHCRB\_0 | XF\_RCAR\_REG\_DMACHCRB\_0 | Channel 0 DMA channel control register B |
| XF\_RCAR\_DMACHCR\_0 | XF\_RCAR\_REG\_DMACHCR\_0 | Channel 0 DMA Channel Control Register |
| XF\_RCAR\_DMATCR\_0 | XF\_RCAR\_REG\_DMATCR\_0 | Channel 0 DMA Transfer Count Register |
| XF\_RCAR\_DMABUFCR\_0 | XF\_RCAR\_REG\_DMABUFCR\_0 | Channel 0 DMA Buffer Control register |
| XF\_RCAR\_DMASAR\_0 | XF\_RCAR\_REG\_DMASAR\_0 | Channel 0 Source Address register |
| XF\_RCAR\_DMADAR\_0 | XF\_RCAR\_REG\_DMADAR\_0 | Channel 0 Destination Address register |
| DMAC\_TRANSMIT\_REQ | 1 | source request is source |
| DMAC\_RECEIVE\_REQ | 2 | source request is destination |

## 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\_DMAOR\_L | DMA operation register (for lower-numbered channels) |
| XF\_RCAR\_REG\_DMAOR\_U | DMA channel clear register (for lower-numbered channels) |
| XF\_RCAR\_REG\_DMACHCLR\_L | DMA channel clear register (for lower-numbered channels) |
| XF\_RCAR\_REG\_DMACHCLR\_U | DMA channel clear register (for higher-numbered channels) |
| XF\_RCAR\_REG\_DMARS\_0 | Channel 0 DMA Extended Resource Selectors Register |
| XF\_RCAR\_REG\_DMADPBASE\_0 | Channel 0 DMA Descriptor Base Address Register |
| XF\_RCAR\_REG\_DMACHCRB\_0 | Channel 0 DMA channel control register B |
| XF\_RCAR\_REG\_DMACHCR\_0 | Channel 0 DMA Channel Control Register |
| XF\_RCAR\_REG\_DMATCR\_0 | Channel 0 DMA Transfer Count Register |
| XF\_RCAR\_REG\_DMABUFCR\_0 | Channel 0 DMA Buffer Control register |
| XF\_RCAR\_REG\_DMASAR\_0 | Channel 0 Source Address register |
| XF\_RCAR\_REG\_DMADAR\_0 | Channel 0 Destination Address register |

## Global variable definition

Below is the array of struct DMAC\_AUDIO\_MODULE\_SETTING-datatype admac\_set\_value holds 141 (DMAC\_MAX) elements representing settings for ADMAC data transfer

|  |  |  |  |
| --- | --- | --- | --- |
| static DMAC\_AUDIO\_MODULE\_SETTING admac\_set\_value[DMAC\_MAX] | | | |
| Array’s index | module\_address | trans | recei |
| DMAC\_SSI00 | 0xEC100000 | 0x15 | 0x16 |
| DMAC\_SSI01 | 0xEC100400 | 0x35 | 0x36 |
| DMAC\_SSI02 | 0xEC100800 | 0x37 | 0x38 |
| DMAC\_SSI03 | 0xEC100C00 | 0x47 | 0x48 |
| DMAC\_SSI04 | 0xEC10A000 | 0x3F | 0x40 |
| DMAC\_SSI05 | 0xEC10A400 | 0x43 | 0x44 |
| DMAC\_SSI06 | 0xEC10A800 | 0x4F | 0x50 |
| DMAC\_SSI07 | 0xEC10AC00 | 0x53 | 0x54 |
| DMAC\_SSI10 | 0xEC101000 | 0x49 | 0x4A |
| DMAC\_SSI11 | 0xEC101400 | 0x4B | 0x4C |
| DMAC\_SSI12 | 0xEC101800 | 0x57 | 0x58 |
| DMAC\_SSI13 | 0xEC101C00 | 0x59 | 0x5A |
| DMAC\_SSI14 | 0xEC10B000 | 0x5F | 0x60 |
| DMAC\_SSI15 | 0xEC10B400 | 0xC3 | 0xC4 |
| DMAC\_SSI16 | 0xEC10B800 | 0xC7 | 0xC8 |
| DMAC\_SSI17 | 0xEC10BC00 | 0xCB | 0xCC |
| DMAC\_SSI20 | 0xEC102000 | 0x63 | 0x64 |
| DMAC\_SSI21 | 0xEC102400 | 0x67 | 0x68 |
| DMAC\_SSI22 | 0xEC102800 | 0x6B | 0x6C |
| DMAC\_SSI23 | 0xEC102C00 | 0x6D | 0x6E |
| DMAC\_SSI24 | 0xEC10C000 | 0xCF | 0xCE |
| DMAC\_SSI25 | 0xEC10C400 | 0xEB | 0xEC |
| DMAC\_SSI26 | 0xEC10C800 | 0xED | 0xEE |
| DMAC\_SSI27 | 0xEC10CC00 | 0xEF | 0xF0 |
| DMAC\_SSI30 | 0xEC103000 | 0x6F | 0x70 |
| DMAC\_SSI31 | 0xEC103400 | 0x21 | 0x22 |
| DMAC\_SSI32 | 0xEC103800 | 0x23 | 0x24 |
| DMAC\_SSI33 | 0xEC103C00 | 0x25 | 0x26 |
| DMAC\_SSI34 | 0xEC10D000 | 0x27 | 0x28 |
| DMAC\_SSI35 | 0xEC10D400 | 0x29 | 0x2A |
| DMAC\_SSI36 | 0xEC10D800 | 0x2B | 0x2C |
| DMAC\_SSI37 | 0xEC10DC00 | 0x2D | 0x2E |
| DMAC\_SSI40 | 0xEC104000 | 0x71 | 0x72 |
| DMAC\_SSI41 | 0xEC104400 | 0x17 | 0x18 |
| DMAC\_SSI42 | 0xEC104800 | 0x19 | 0x1A |
| DMAC\_SSI43 | 0xEC104C00 | 0x1B | 0x1C |
| DMAC\_SSI44 | 0xEC10E000 | 0x1D | 0x1E |
| DMAC\_SSI45 | 0xEC10E400 | 0x1F | 0x20 |
| DMAC\_SSI46 | 0xEC10E800 | 0x31 | 0x32 |
| DMAC\_SSI47 | 0xEC10EC00 | 0x33 | 0x34 |
| DMAC\_SSI5 | 0xEC105000 | 0x73 | 0x74 |
| DMAC\_SSI6 | 0xEC106000 | 0x75 | 0x76 |
| DMAC\_SSI7 | 0xEC107000 | 0x79 | 0x7A |
| DMAC\_SSI8 | 0xEC108000 | 0x7B | 0x7C |
| DMAC\_SSI90 | 0xEC109000 | 0x7D | 0x7E |
| DMAC\_SSI91 | 0xEC109400 | 0x7F | 0x80 |
| DMAC\_SSI92 | 0xEC109800 | 0x81 | 0x82 |
| DMAC\_SSI93 | 0xEC109C00 | 0x83 | 0x84 |
| DMAC\_SSI94 | 0xEC10F000 | 0xA3 | 0xA4 |
| DMAC\_SSI95 | 0xEC10F400 | 0xA5 | 0xA6 |
| DMAC\_SSI96 | 0xEC10F800 | 0xA7 | 0xA8 |
| DMAC\_SSI97 | 0xEC10FC00 | 0xA9 | 0xAA |
| DMAC\_DTCPPP0 | 0xEC120000 | 0xBF | 0xC0 |
| DMAC\_DTCPPP1 | 0xEC120400 | 0xD5 | 0xD6 |
| DMAC\_DTCPCP0 | 0xEC124000 | 0xD7 | 0xD8 |
| DMAC\_DTCPCP1 | 0xEC124400 | 0xD9 | 0xDA |
| DMAC\_SCU\_SRCI0 | 0xEC000000 | 0x85 | 0xff |
| DMAC\_SCU\_SRCI1 | 0xEC000400 | 0x87 | 0xff |
| DMAC\_SCU\_SRCI2 | 0xEC000800 | 0x89 | 0xff |
| DMAC\_SCU\_SRCI3 | 0xEC000C00 | 0x8B | 0xff |
| DMAC\_SCU\_SRCI4 | 0xEC001000 | 0x8D | 0xff |
| DMAC\_SCU\_SRCI5 | 0xEC001400 | 0x8F | 0xff |
| DMAC\_SCU\_SRCI6 | 0xEC001800 | 0x91 | 0xff |
| DMAC\_SCU\_SRCI7 | 0xEC001C00 | 0x93 | 0xff |
| DMAC\_SCU\_SRCI8 | 0xEC002000 | 0x95 | 0xff |
| DMAC\_SCU\_SRCI9 | 0xEC002400 | 0x97 | 0xff |
| DMAC\_SCU\_SRCO0 | 0xEC004000 | 0xff | 0x9A |
| DMAC\_SCU\_SRCO1 | 0xEC004400 | 0xff | 0x9C |
| DMAC\_SCU\_SRCO2 | 0xEC004800 | 0xff | 0x9E |
| DMAC\_SCU\_SRCO3 | 0xEC004C00 | 0xff | 0xA0 |
| DMAC\_SCU\_SRCO4 | 0xEC005000 | 0xff | 0xB0 |
| DMAC\_SCU\_SRCO5 | 0xEC005400 | 0xff | 0xB2 |
| DMAC\_SCU\_SRCO6 | 0xEC005800 | 0xff | 0xB4 |
| DMAC\_SCU\_SRCO7 | 0xEC005C00 | 0xff | 0xB6 |
| DMAC\_SCU\_SRCO8 | 0xEC006000 | 0xff | 0xB8 |
| DMAC\_SCU\_SRCO9 | 0xEC006400 | 0xff | 0xBA |
| DMAC\_SCU\_CMD0 | 0xEC008000 | 0xff | 0xBC |
| DMAC\_SCU\_CMD1 | 0xEC008400 | 0xff | 0xBE |
| DMAC\_MLM0 | 0xEC020000 | 0xDB | 0xDC |
| DMAC\_MLM1 | 0xEC020400 | 0xE3 | 0xE4 |
| DMAC\_MLM2 | 0xEC020800 | 0xE5 | 0xE6 |
| DMAC\_MLM3 | 0xEC020C00 | 0xE7 | 0xE8 |
| DMAC\_MLM4 | 0xEC021000 | 0xF3 | 0xF4 |
| DMAC\_MLM5 | 0xEC021400 | 0xF5 | 0xF6 |
| DMAC\_MLM6 | 0xEC021800 | 0xF7 | 0xF8 |
| DMAC\_MLM7 | 0xEC021C00 | 0xF9 | 0xFA |

## Function definition

### admac\_get\_register\_addr

DD\_PLG\_RDR\_07\_001

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | inline UWORD32 admac\_get\_register\_addr(UWORD32 reg\_addr0, ADMA\_CH n) | | | |
| **Function** | This function is to get DMAC channel N register’s address. | | | |
| **Arguments** | Type | Name | I/O | Description |
| UWORD32 | reg\_addr0 | I | Pointer to register of channel 0 |
| ADMA\_CH | n | I | Channel number  Valid range: [0: 31] |
| **Return value** | The value of register channel N | | | |
| **Description** | * admac\_get\_register\_addr command processing:   - Get DMAC channel N register’s address | | | |

[Covers: FD\_PLG\_RDR\_005, FD\_PLG\_RDR\_014, FD\_PLG\_RDR\_015, FD\_PLG\_RDR\_017, FD\_PLG\_RDR\_032, FD\_PLG\_RDR\_043]

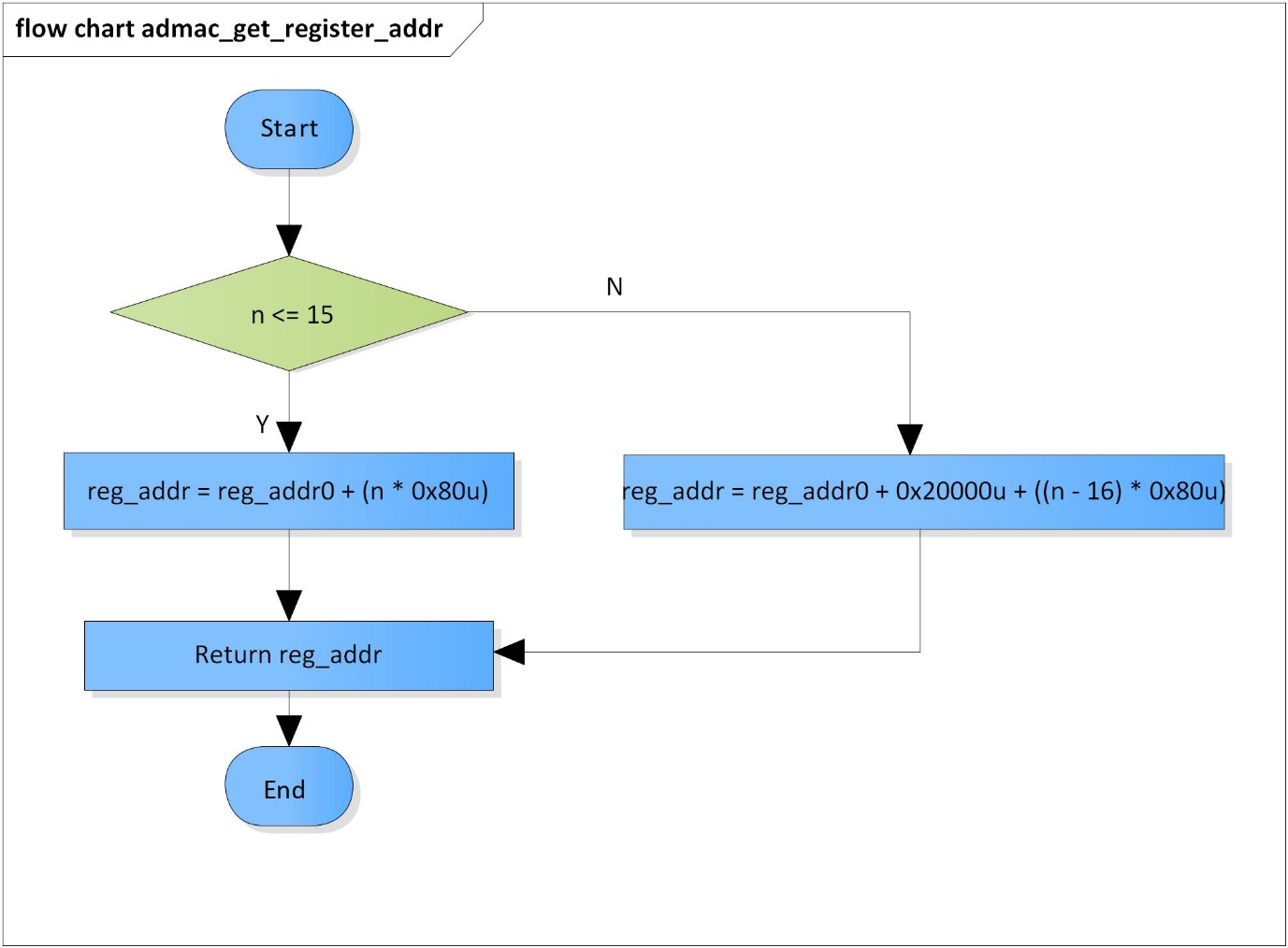


Figure 3‑1 admac\_get\_register\_addr flowchart

### admac\_set\_stage\_number

DD\_PLG\_RDR\_07\_002

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static inline void admac\_set\_stage\_number(UWORD32 n, UWORD32 stg\_num) | | | |
| **Function** | This function is to set stage number. | | | |
| **Arguments** | Type | Name | I/O | Description |
| UWORD32 | n | I | Channel number |
| UWORD32 | stg\_num | I | Stage number |
| **Return value** | None | | | |
| **Description** | * admac\_set\_stage\_number command processing:   - Set stage for DMA channel N | | | |

[Covers: FD\_PLG\_RDR\_005]

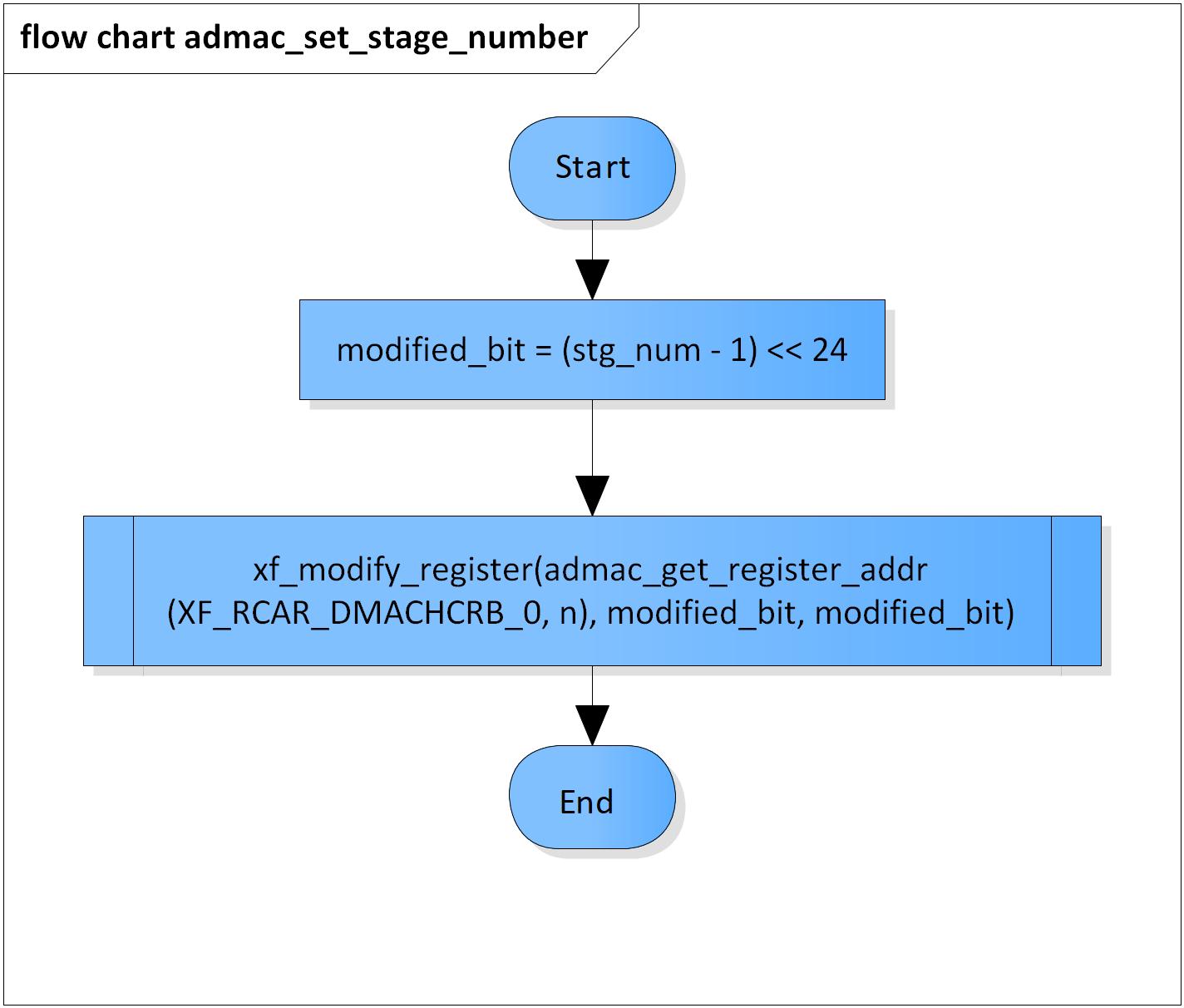


Figure 3‑2 admac\_set\_stage\_number flowchart

### admac\_reset\_descriptor

DD\_PLG\_RDR\_07\_003

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static inline void admac\_reset\_descriptor(ADMA\_CH n) | | | |
| **Function** | This function is to reset descriptor of DMA channel N. | | | |
| **Arguments** | Type | Name | I/O | Description |
| ADMA\_CH | n | I | Channel number  Valid range: [0: 31] |
| **Return value** | None | | | |
| **Description** | * admac\_reset\_descriptor command processing:   - Reset descriptor of DMA channel N | | | |

[Covers: FD\_PLG\_RDR\_005]

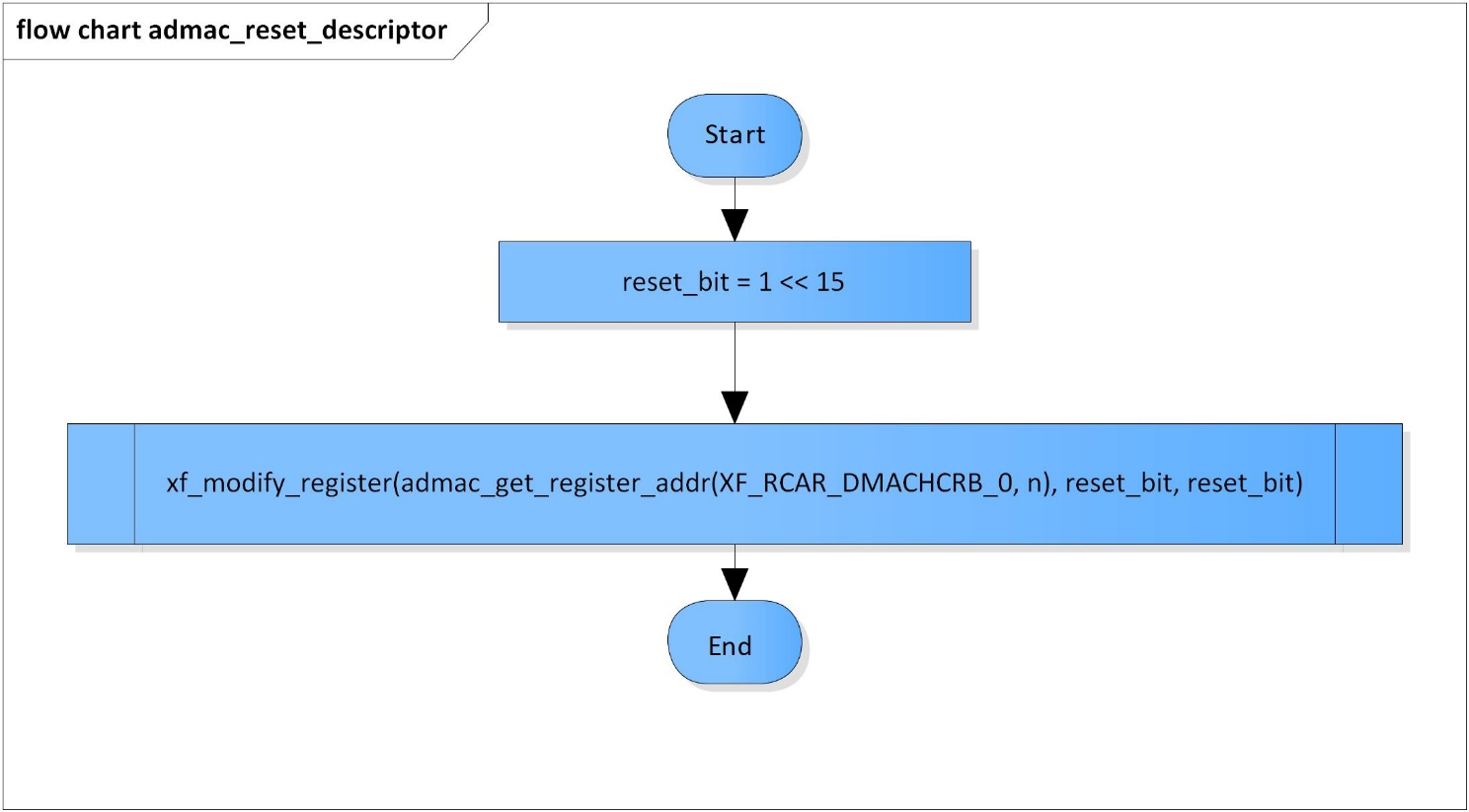
****

Figure 3‑3 admac\_reset\_descriptor flowchart

### admac\_start\_descriptor

DD\_PLG\_RDR\_07\_004

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static inline void admac\_start\_descriptor(ADMA\_CH n) | | | |
| **Function** | This function is to start descriptor of DMA channel N. | | | |
| **Arguments** | Type | Name | I/O | Description |
| ADMA\_CH | n | I | Channel number  Valid range: [0: 31] |
| **Return value** | None | | | |
| **Description** | * admac\_start\_descriptor command processing:   - Start descriptor of DMA channel N | | | |

[Covers: FD\_PLG\_RDR\_005]

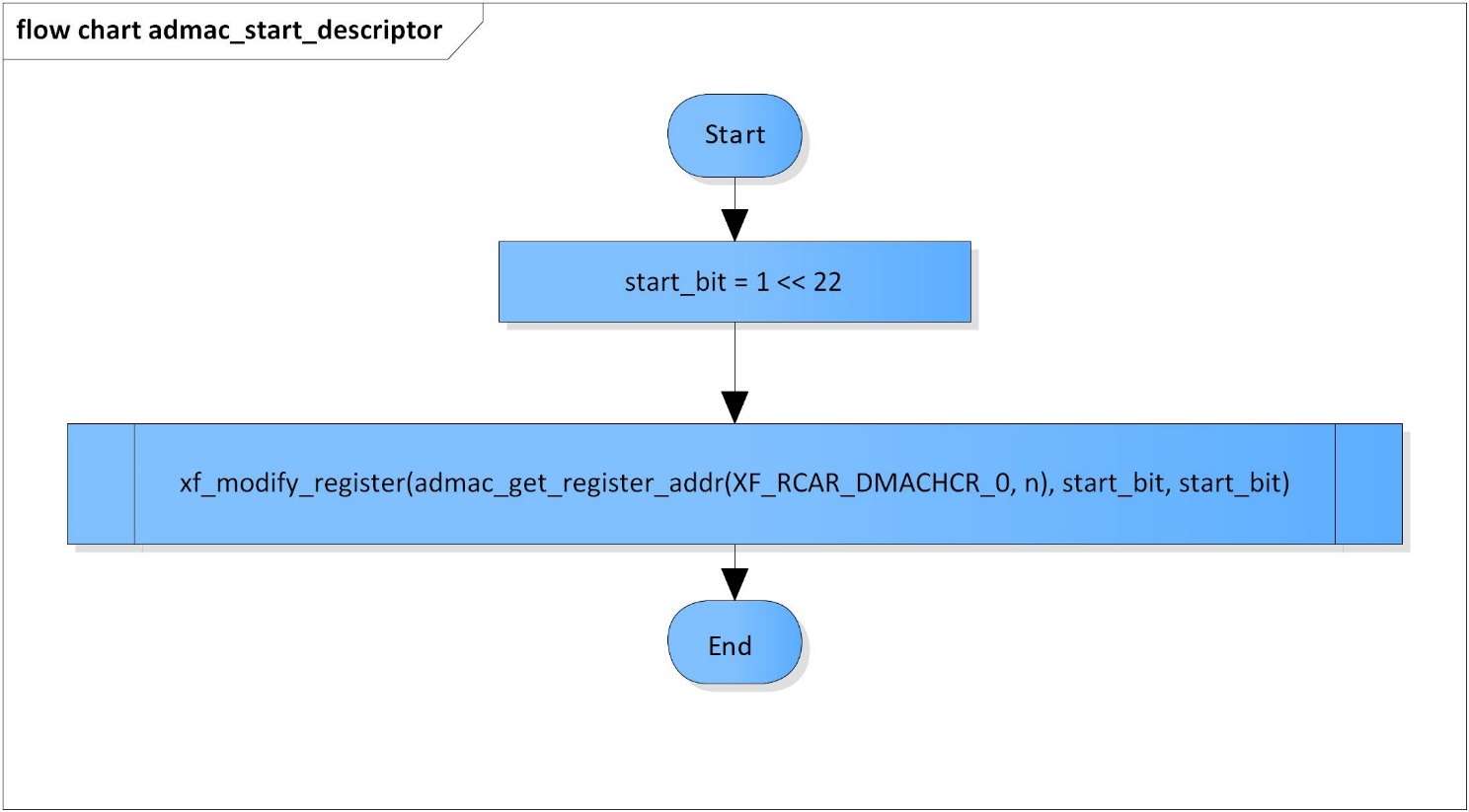


Figure 3‑4 admac\_start\_desciptor flowchart

### admac\_enable\_ch

DD\_PLG\_RDR\_07\_005

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static inline void admac\_enable\_ch(ADMA\_CH n) | | | |
| **Function** | This function is to enable DMA channel N. | | | |
| **Arguments** | Type | Name | I/O | Description |
| ADMA\_CH | n | I | Channel number  Valid range: [0: 31] |
| **Return value** | None | | | |
| **Description** | * admac\_enable\_ch command processing:   - Enable DMA channel N | | | |

[Covers: FD\_PLG\_RDR\_005, FD\_PLG\_RDR\_014, FD\_PLG\_RDR\_015, FD\_PLG\_RDR\_017, FD\_PLG\_RDR\_032, FD\_PLG\_RDR\_043]

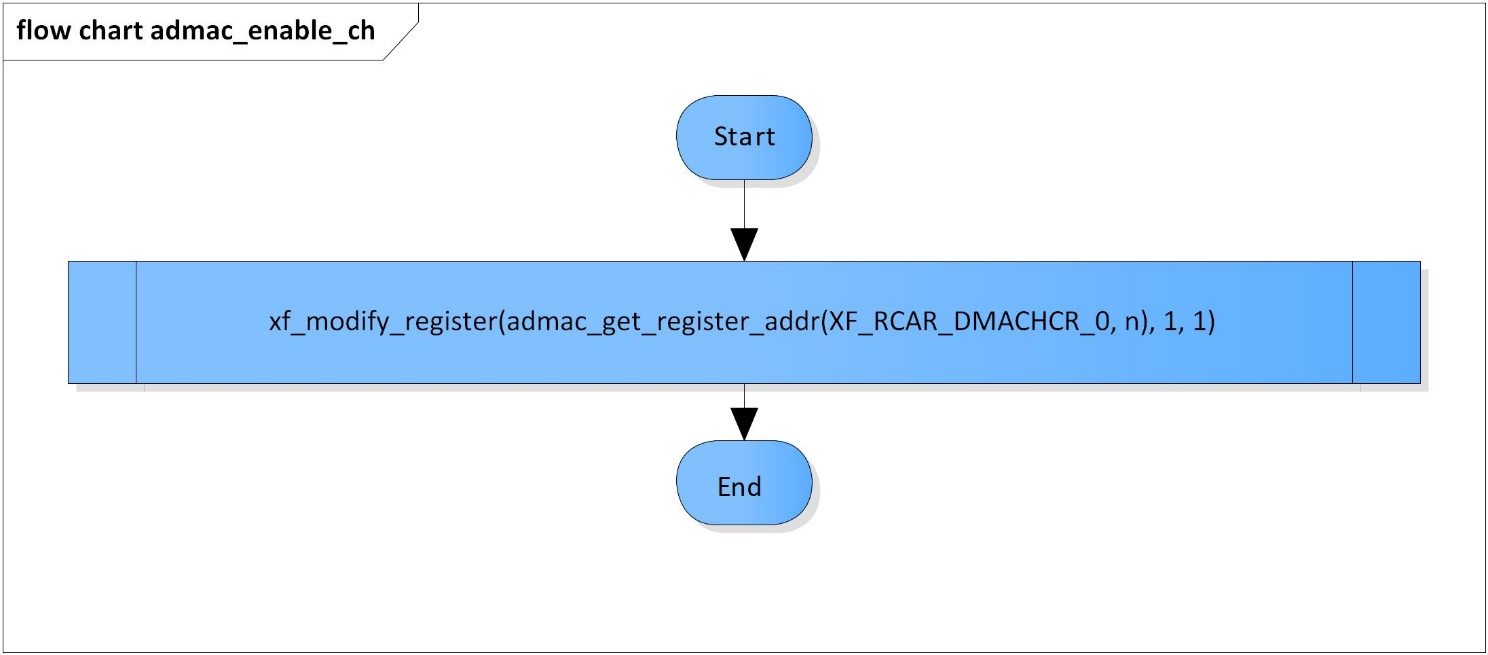


Figure 3‑5 admac\_enable\_ch flowchart

### admac\_disable\_ch

DD\_PLG\_RDR\_07\_006

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static inline void admac\_disable\_ch(ADMA\_CH n) | | | |
| **Function** | This function is to disable DMA channel N. | | | |
| **Arguments** | Type | Name | I/O | Description |
| ADMA\_CH | n | I | Channel number  Valid range: [0: 31] |
| **Return value** | None | | | |
| **Description** | * admac\_disable\_ch command processing:   - Disable DMA channel N | | | |

[Covers: FD\_PLG\_RDR\_005, FD\_PLG\_RDR\_014, FD\_PLG\_RDR\_015, FD\_PLG\_RDR\_017, FD\_PLG\_RDR\_032, FD\_PLG\_RDR\_043]

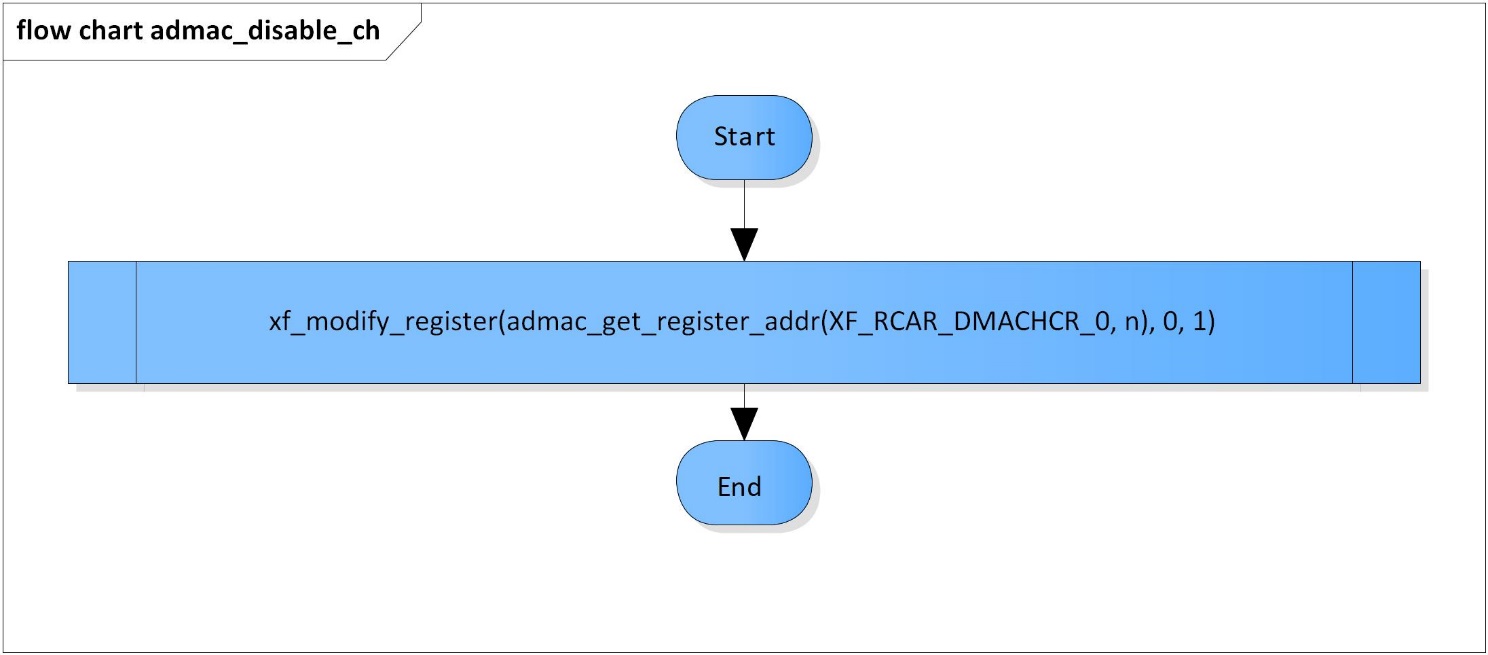


Figure 3‑6 admac\_disable\_ch flowchart

### admac\_enable\_module

DD\_PLG\_RDR\_07\_007

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static inline dmac\_error\_code admac\_enable\_module(ADMA\_CH n) | | | |
| **Function** | This function is to enable DMAC module. | | | |
| **Arguments** | Type | Name | I/O | Description |
| ADMA\_CH | n | I | Channel number  Valid range: [0: 31] |
| **Return value** | DMAC\_ERROR\_OUT\_RANGE | | DMAC module is invalid | |
| DMAC\_ERROR\_NONE | | Normal end | |
| **Description** | * admac\_enable\_module command processing:   - Set ‘1’ to DE bit of register XF\_RCAR\_DMAOR\_L or XF\_RCAR\_DMAOR\_U depending channel number to enable DMAC module | | | |

[Covers: FD\_PLG\_RDR\_005, FD\_PLG\_RDR\_015, FD\_PLG\_RDR\_017, FD\_PLG\_RDR\_032, FD\_PLG\_RDR\_043]

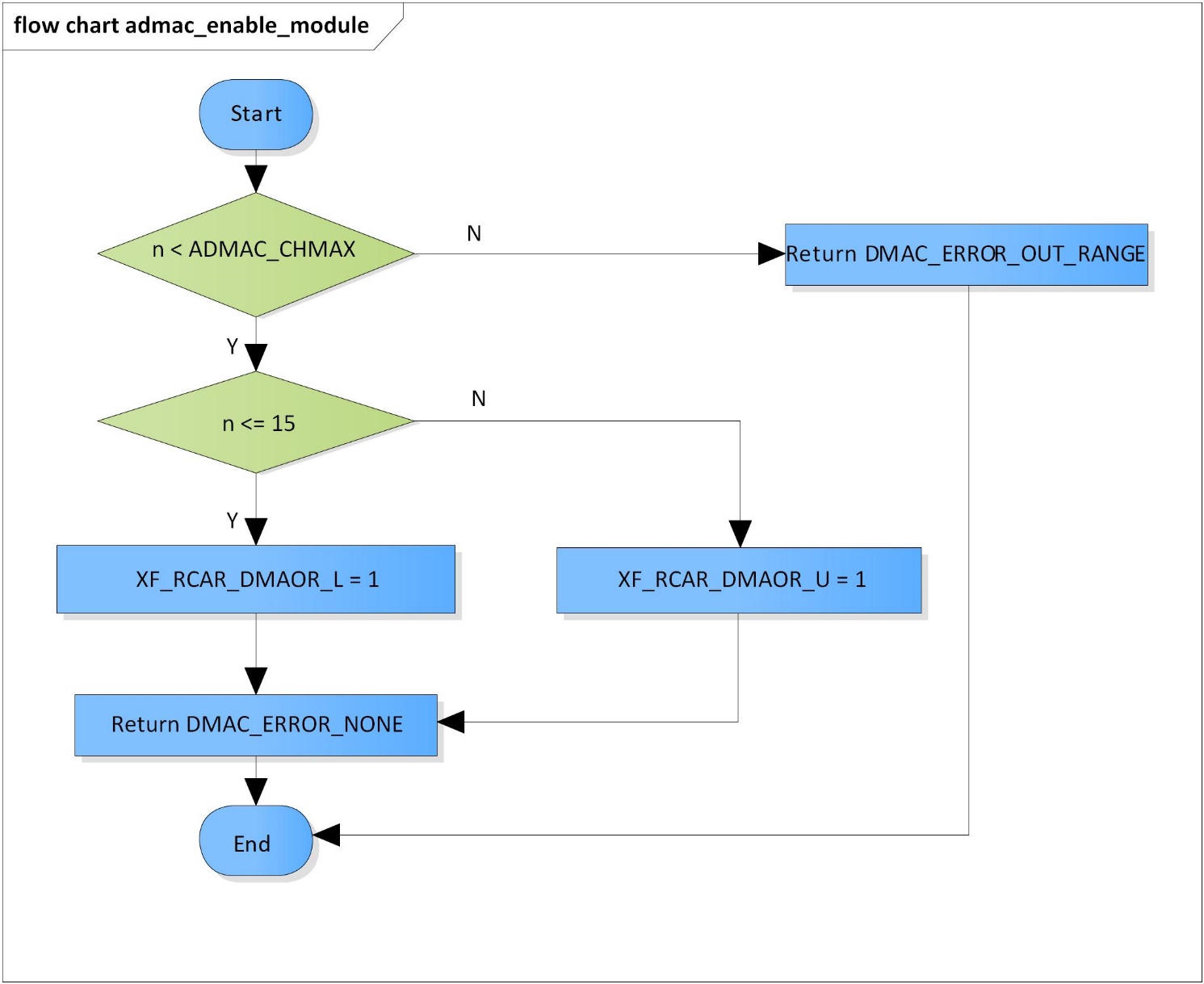


Figure 3‑7 admac\_enable\_module flowchart

### admac\_clear\_channel

DD\_PLG\_RDR\_07\_008

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static inline dmac\_error\_code admac\_clear\_module(ADMA\_CH n) | | | |
| **Function** | This function is to clear value set in DMAC channel N registers. | | | |
| **Arguments** | Type | Name | I/O | Description |
| ADMA\_CH | n | I | Channel number  Valid range: [0: 31] |
| **Return value** | DMAC\_ERROR\_OUT\_RANGE | | DMAC module is invalid | |
| DMAC\_ERROR\_NONE | | Normal end | |
| **Description** | * admac\_clear\_channel command processing:   - Clear value set in DMAC channel N registers | | | |

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

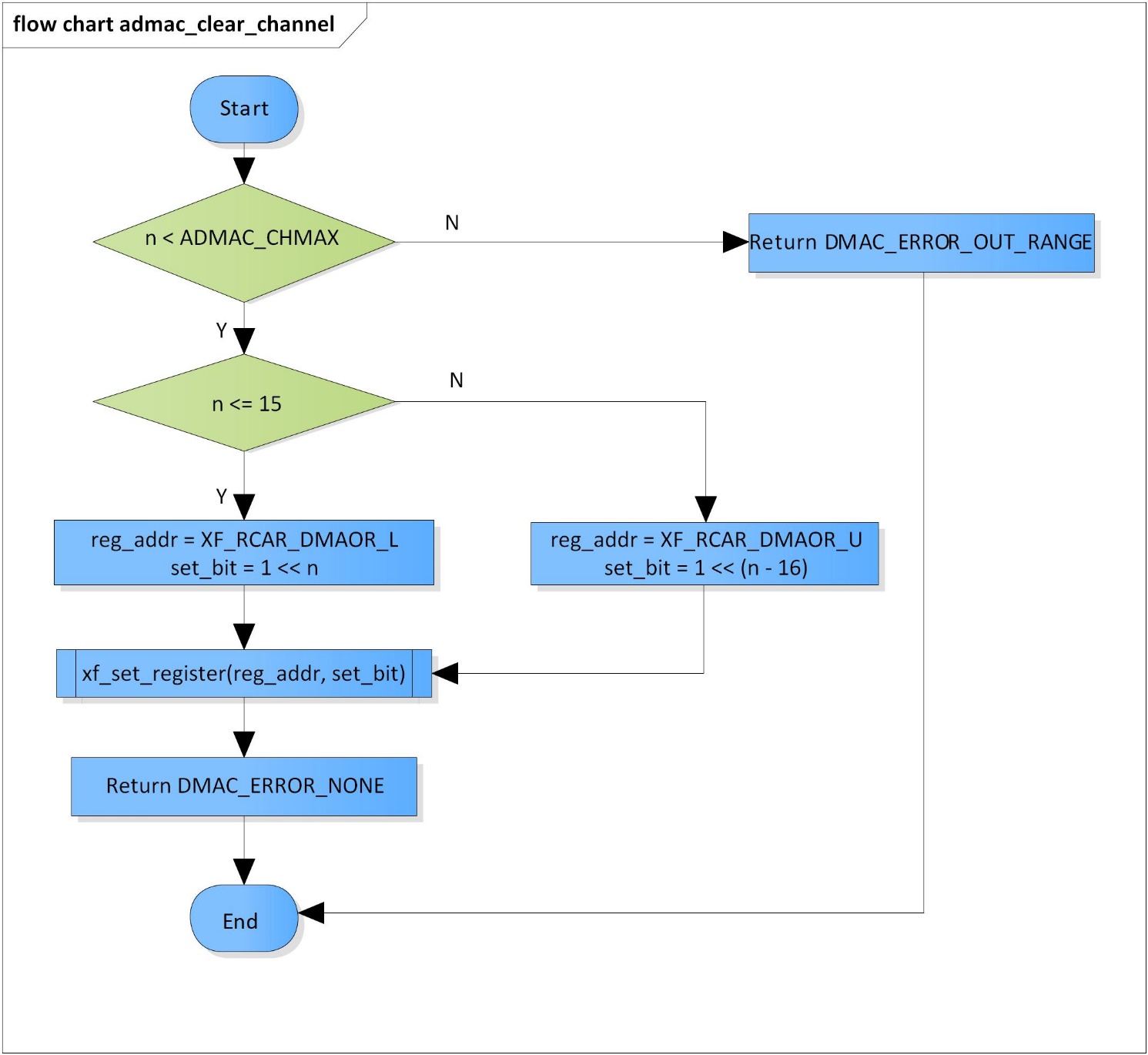


Figure 3‑8 admac\_clear\_channel flowchart

### admac\_check\_stage

DD\_PLG\_RDR\_07\_009

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | inline WORD32 admac\_check\_stage(ADMA\_CH n) | | | |
| **Function** | This function is to check the current stage of DMAC module. | | | |
| **Arguments** | Type | Name | I/O | Description |
| ADMA\_CH | n | I | Channel number  Valid range: [0: 31] |
| **Return value** | The stage of ADMAC module | | | |
| **Description** | * admac\_check\_stage command processing:   - Check the current stage of DMAC module | | | |

[Covers: FD\_PLG\_RDR\_015, FD\_PLG\_RDR\_017]

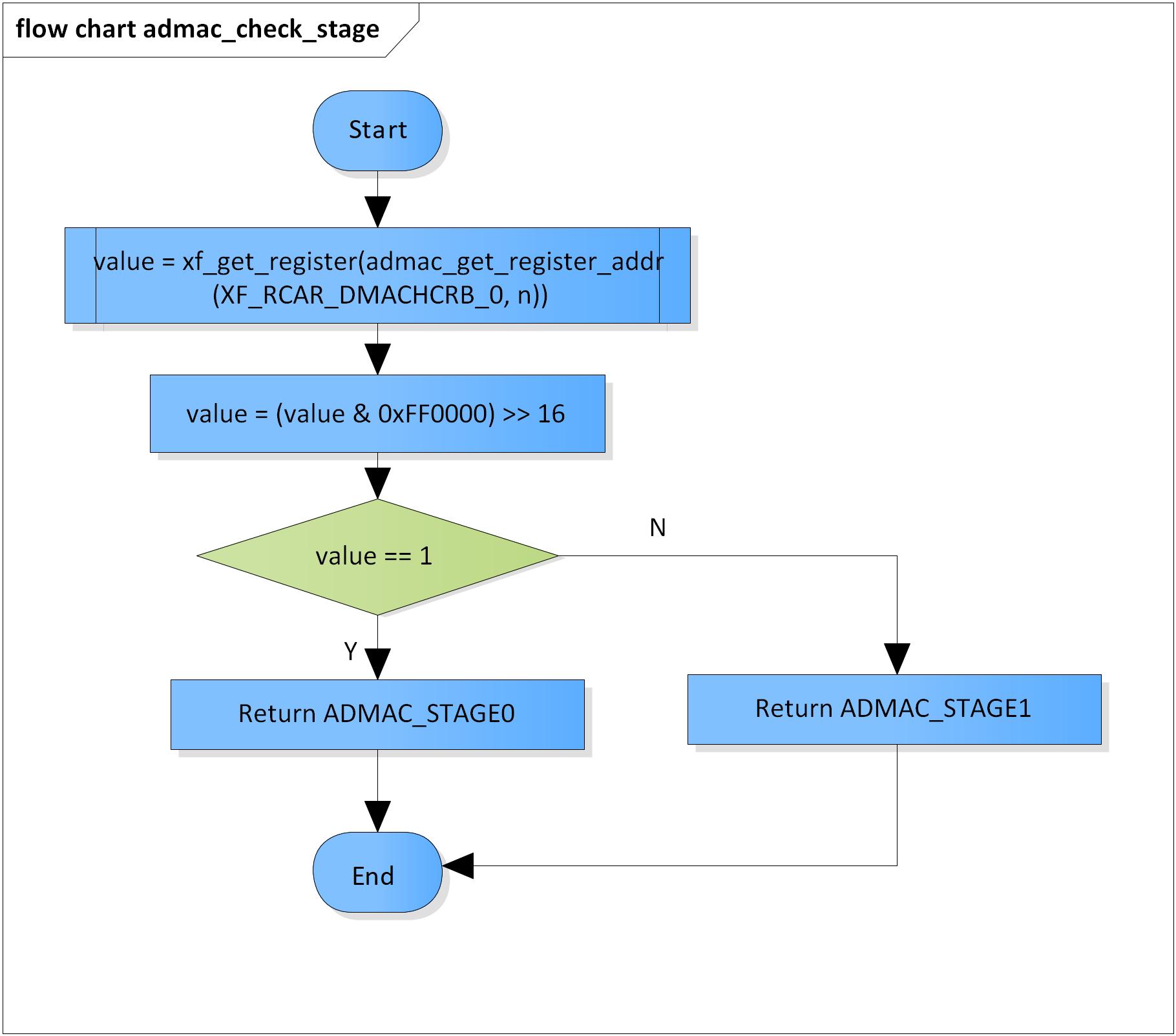


Figure 3‑9 admac\_check\_stage flowchart

### admac\_check\_and\_clear\_interrupt

DD\_PLG\_RDR\_07\_010

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | inline WORD32 admac\_check\_and\_clear\_interrupt(ADMA\_CH n) | | | |
| **Function** | This function is to check and clear interrupt of DMAC channel N. | | | |
| **Arguments** | Type | Name | I/O | Description |
| ADMA\_CH | n | I | Channel number  Valid range: [0: 31] |
| **Return value** | The interrupt\_flag | | | |
| **Description** | * admac\_check\_and\_clear\_interrupt command processing:   - Check and clear the interrupt of DMAC channel N | | | |

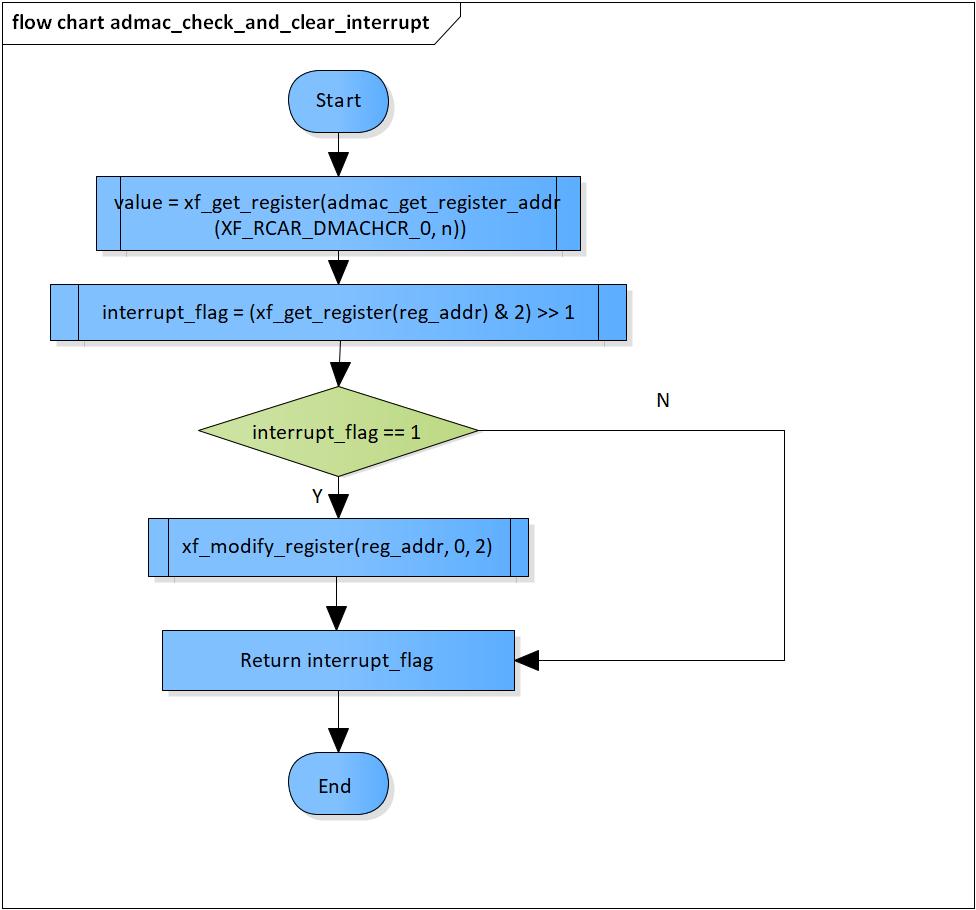
[Covers: FD\_PLG\_RDR\_015, FD\_PLG\_RDR\_017]

Figure 3‑10 admac\_check\_and\_clear\_interrupt flowchart

### admac\_start

DD\_PLG\_RDR\_07\_011

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | inline dmac\_error\_code admac\_start(ADMA\_CH n) | | | |
| **Function** | This function is to start DMAC channel N. | | | |
| **Arguments** | Type | Name | I/O | Description |
| ADMA\_CH | n | I | Channel number  Valid range: [0: 31] |
| **Return value** | DMAC\_ERROR\_OUT\_RANGE | | DMAC module is invalid | |
| DMAC\_ERROR\_NONE | | Normal end | |
| **Description** | * admac\_start command processing:   - Start the DMAC channel N | | | |

[Covers: FD\_PLG\_RDR\_005, FD\_PLG\_RDR\_015, FD\_PLG\_RDR\_017, FD\_PLG\_RDR\_032, FD\_PLG\_RDR\_043]

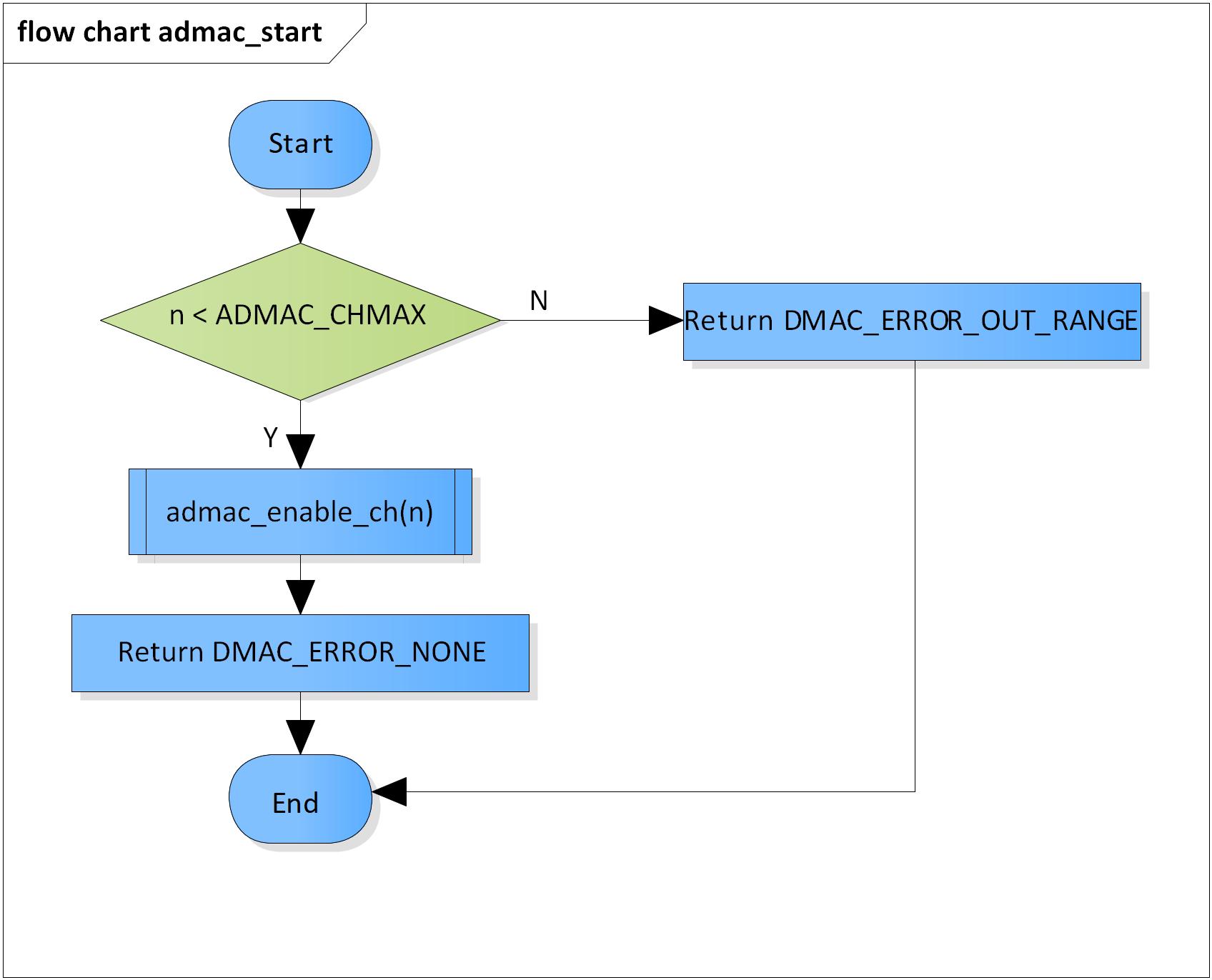


Figure 3‑11admac\_start flowchart

### admac\_stop

DD\_PLG\_RDR\_07\_012

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | inline dmac\_error\_code admac\_stop(ADMA\_CH n) | | | |
| **Function** | This function is to stop DMAC channel N. | | | |
| **Arguments** | Type | Name | I/O | Description |
| ADMA\_CH | n | I | Channel number  Valid range: [0: 31] |
| **Return value** | DMAC\_ERROR\_OUT\_RANGE | | DMAC module is invalid | |
| DMAC\_ERROR\_NONE | | Normal end | |
| **Description** | * admac\_stop command processing:   - Stop the DMAC channel N and clear value set on all of its registers | | | |

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

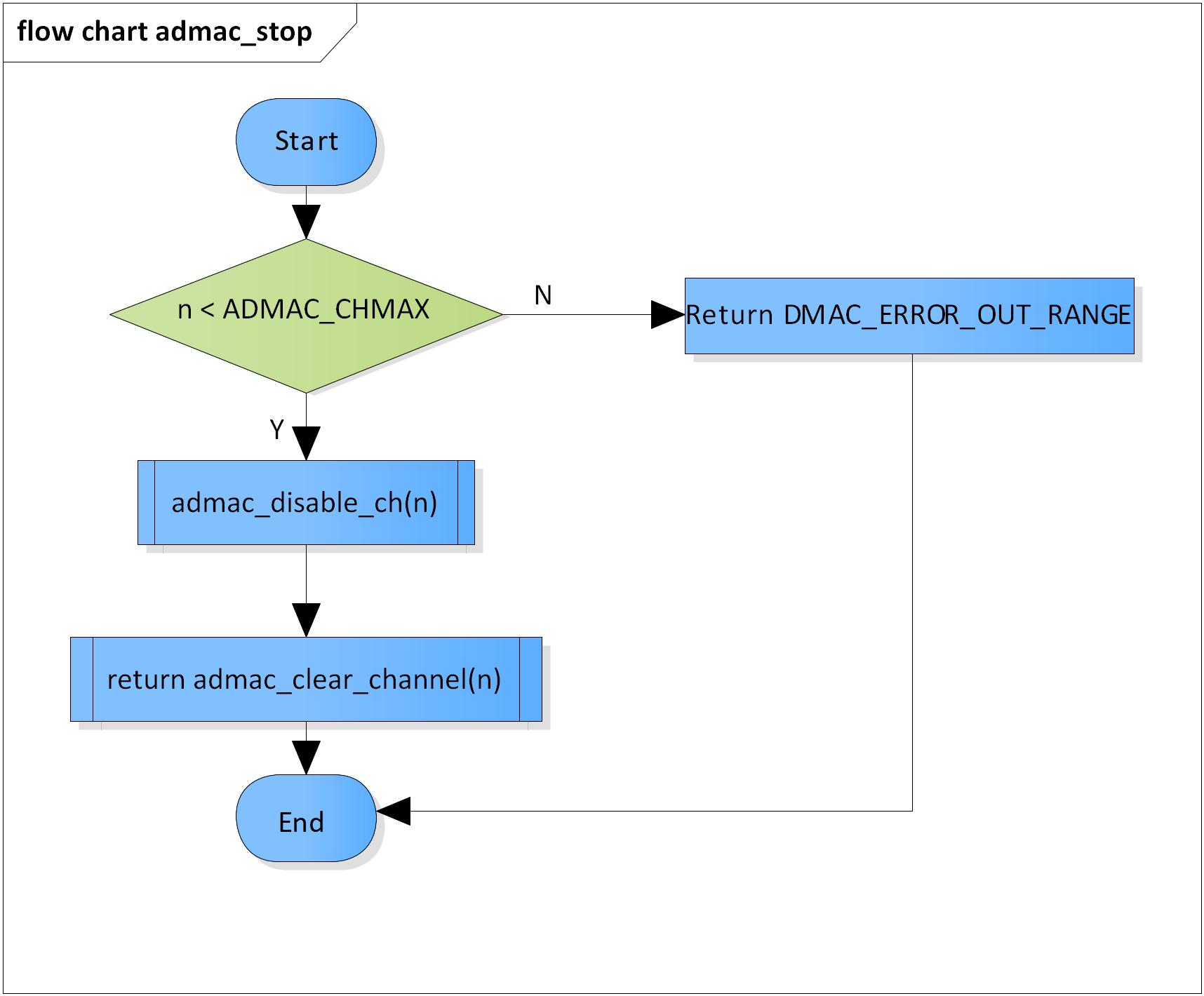


Figure 3‑12 admac\_stop flowchart

### admac\_descriptor\_setup

DD\_PLG\_RDR\_07\_013

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static inline VOID admac\_descriptor\_setup(ADMA\_CH n, const void \*descript\_mem) | | | |
| **Function** | This function is to set DMAC channel N’s descriptor. | | | |
| **Arguments** | Type | Name | I/O | Description |
| ADMA\_CH | n | I | Channel number  Valid range: [0: 31] |
| const void \* | descript\_mem | I | Pointer to descriptor memory |
| **Return value** | None | | | |
| **Description** | * admac\_descriptor\_setup command processing:   - Set DMAC channel N’s descriptor | | | |

[Covers: FD\_PLG\_RDR\_005]

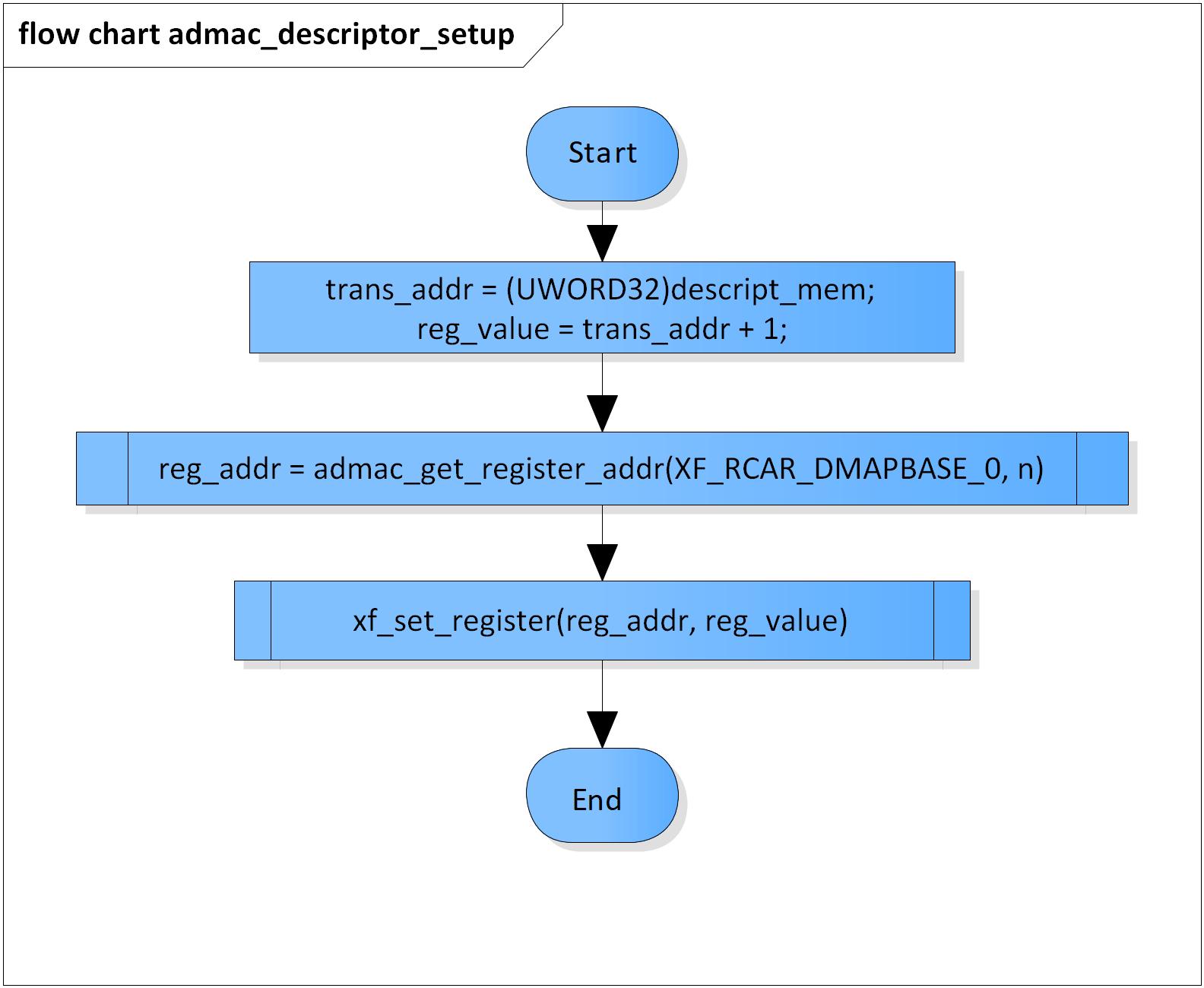
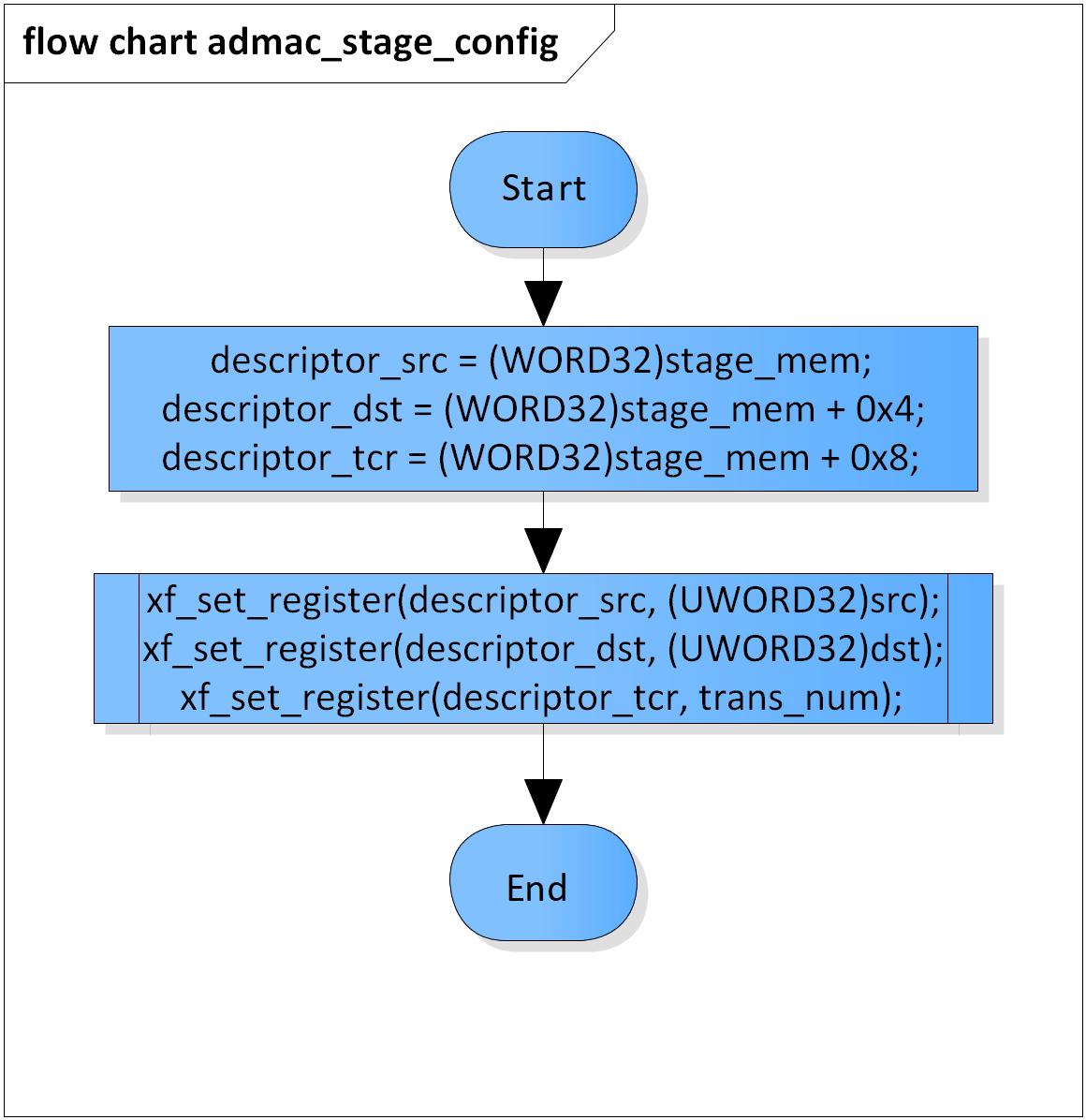


Figure 3‑13 admac\_descriptor\_setup flowchart

### admac\_stage\_config

DD\_PLG\_RDR\_07\_014

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static inline VOID admac\_stage\_config(const void \*src, const void \*dst, const void \*stage\_mem, UWORD32 trans\_num) | | | |
| **Function** | This function is to set DMAC stage. | | | |
| **Arguments** | Type | Name | I/O | Description |
| const void \* | src | I | Source address |
| const void \* | dst | I | Destination address |
| const void \* | stage\_mem | I | Stage memory |
| UWORD32 | trans\_num | I | Transmission number in a time |
| **Return value** | None | | | |
| **Description** | * admac\_stage\_config command processing:   - Set DMAC stage | | | |

[Covers: FD\_PLG\_RDR\_005]

Figure 3‑14 admac\_stage\_config flowchart

### admac\_pre\_config

DD\_PLG\_RDR\_07\_015

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Syntax** | static inline dmac\_error\_code admac\_pre\_config(ADMA\_CH dma\_channel, UWORD32 src\_reqst, UWORD32 reqst\_flag, const void \*descript\_mem, UWORD32 stg\_num, ADMAC\_MODE mode, UWORD32 transfer\_size) | | | | |
| **Function** | This function is to set DMAC parameters. | | | | |
| **Arguments** | Type | Name | I/O | | Description |
| ADMA\_CH | dma\_channel | I | | DMAC channel  Valid range: [0: 31] |
| UWORD32 | src\_reqst | I | | Source request |
| UWORD32 | reqst\_flag | I | | Source or destination is requested  Valid values:  DMAC\_RECEIVE\_REQ  DMAC\_TRANSMIT\_REQ |
| const void \* | descript\_mem | I | | Descriptor memory address |
| UWORD32 | stg\_num | I | | Stage number  Valid values: greater than zero |
| ADMAC\_MODE | mode | I | | Operating mode |
| UWORD32 | transfer\_size | I | | Transmission unit  Valid values: [4] |
| **Return value** | DMAC\_ERROR\_OUT\_RANGE | | | DMAC module is invalid | |
| DMAC\_ERROR\_INVALID | | | Stage number is invalid  Request address is invalid  Transfer size is invalid | |
| DMAC\_ERROR\_NONE | | | Normal end | |
| **Description** | * admac\_pre\_config command processing:   - Set DMAC parameters | | | | |

[Covers: FD\_PLG\_RDR\_005]

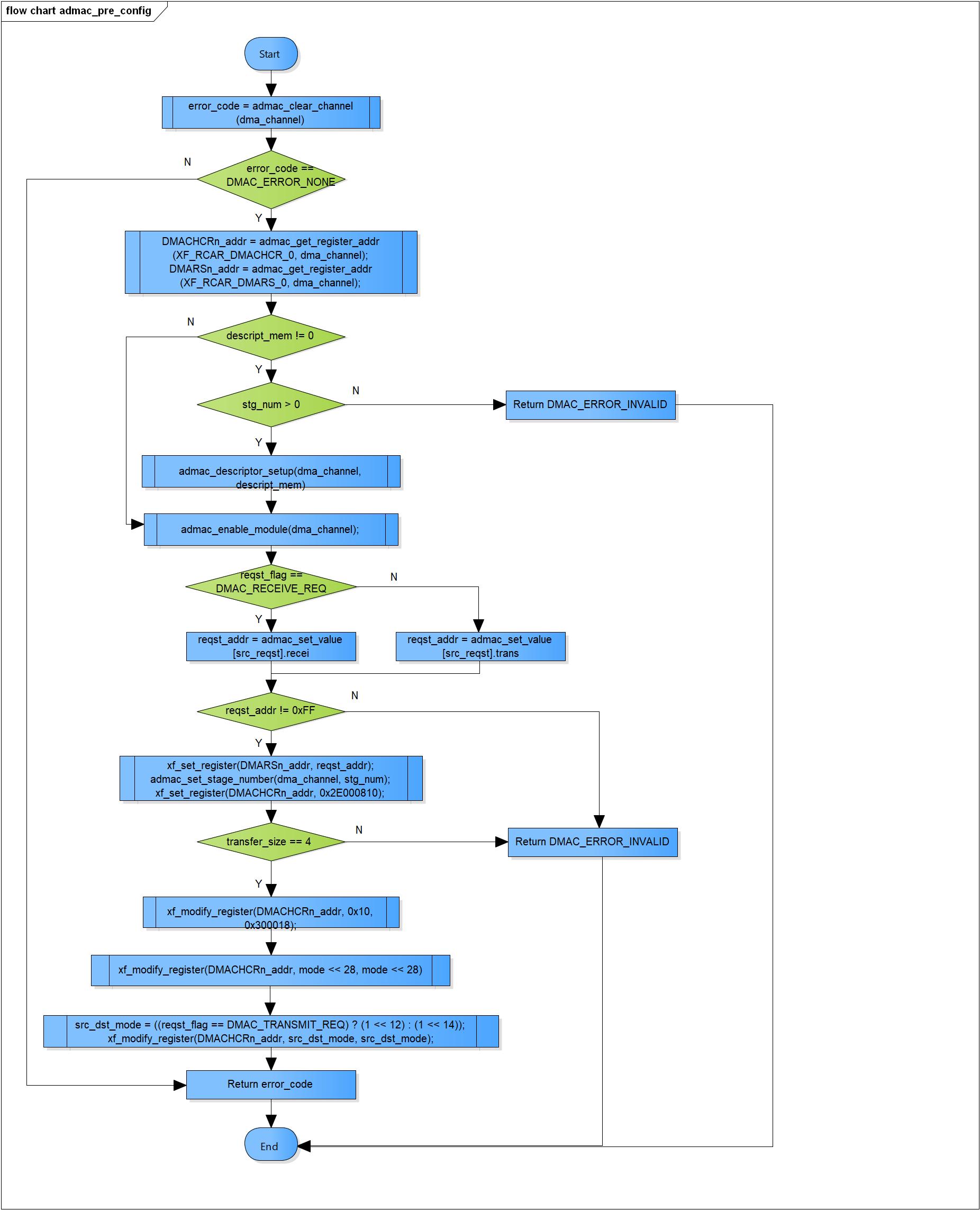
****

Figure 3‑15 admac\_pre\_config flowchart

### admac\_post\_config

DD\_PLG\_RDR\_07\_016

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | static inline VOID admac\_post\_config(ADMA\_CH n) | | | |
| **Function** | This function is to post set DMAC parameters for DMAC channel N | | | |
| **Arguments** | Type | Name | I/O | Description |
| ADMA\_CH | n | I | Channel number  Valid range: [0: 31] |
| **Return value** | None | | | |
| **Description** | * admac\_post\_config command processing:   - Reset the descriptor and then start it | | | |

[Covers: FD\_PLG\_RDR\_005]

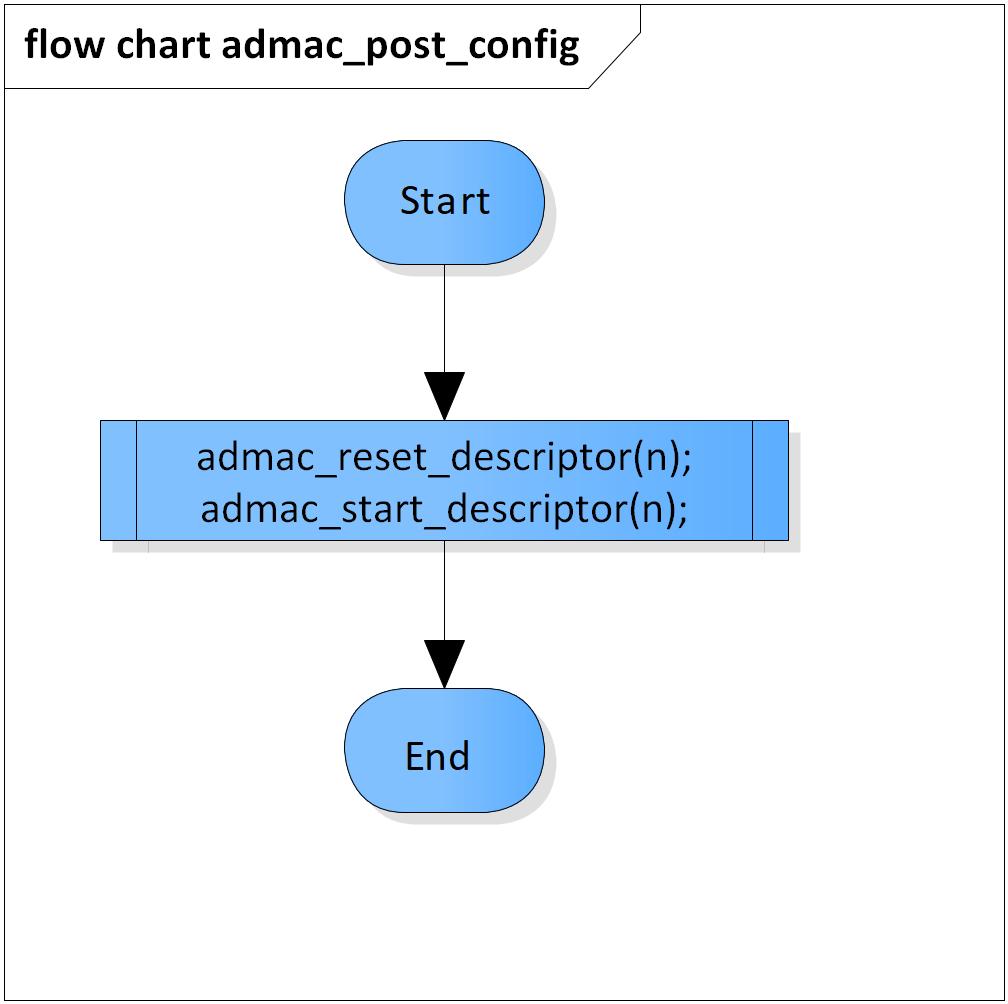
****

Figure 3‑16 admac\_post\_config flowchart

### admac\_check\_valid\_channel

DD\_PLG\_RDR\_07\_017

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | inline dmac\_error\_code admac\_check\_valid\_channel(pVOID pch) | | | |
| **Function** | This function is to check availability of DMACs and choose an available one for operation. | | | |
| **Arguments** | Type | Name | I/O | Description |
| pVOID | pch | I/O | Pointer to channel of DMAC  Valid range: [0: 31] |
| **Return value** | DMAC\_ERROR\_OUT\_RANGE | | DMAC module is invalid | |
| DMAC\_ERROR\_BUSY | | No available DMAC module | |
| DMAC\_ERROR\_NONE | | Normal end | |
| **Description** | * admac\_check\_valid\_channel command processing:   - Check availability of DMACs  - Choose an available one for operation | | | |

[Covers: FD\_PLG\_RDR\_005]

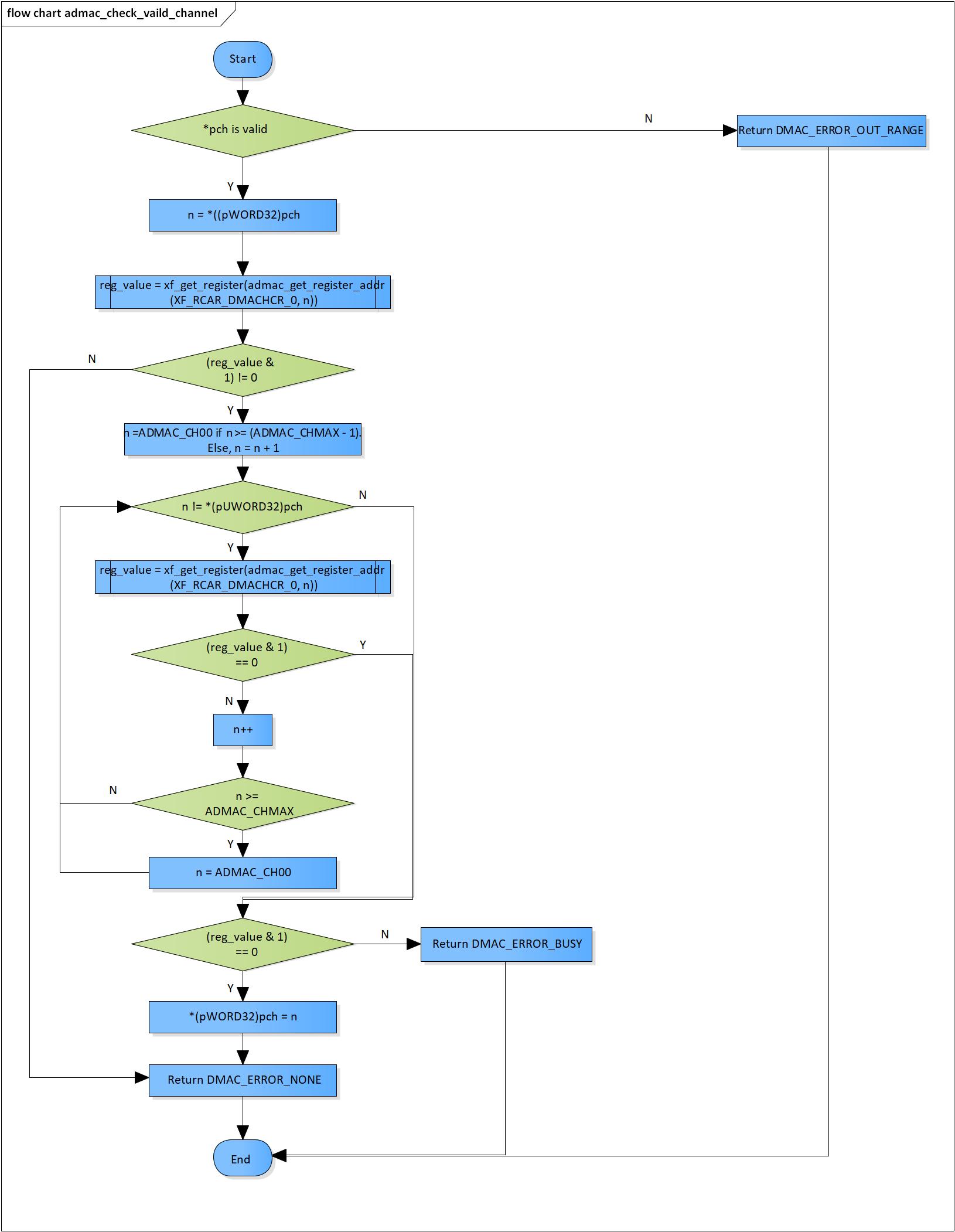


Figure 3‑17 admac\_check\_valid\_channel flowchart

### admac\_setting\_use\_descriptor

DD\_PLG\_RDR\_07\_018

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | inline dmac\_error\_code admac\_setting\_use\_descriptor(const DMAC\_SETTING \*dmac\_set) | | | |
| **Function** | This function is to set DMAC module using descriptor memory. | | | |
| **Arguments** | Type | Name | I/O | Description |
| const DMAC\_SETTING \* | dmac\_set | I | Pointer to DMAC parameter structure |
| **Return value** | Error returned from function admac\_pre\_config if failed | | | |
| **Description** | * admac\_setting\_use\_descriptor command processing:   - Pre set up DMAC module  - Set up all stages based on transmission mode  - Post set up DMAC module | | | |

[Covers: FD\_PLG\_RDR\_005]

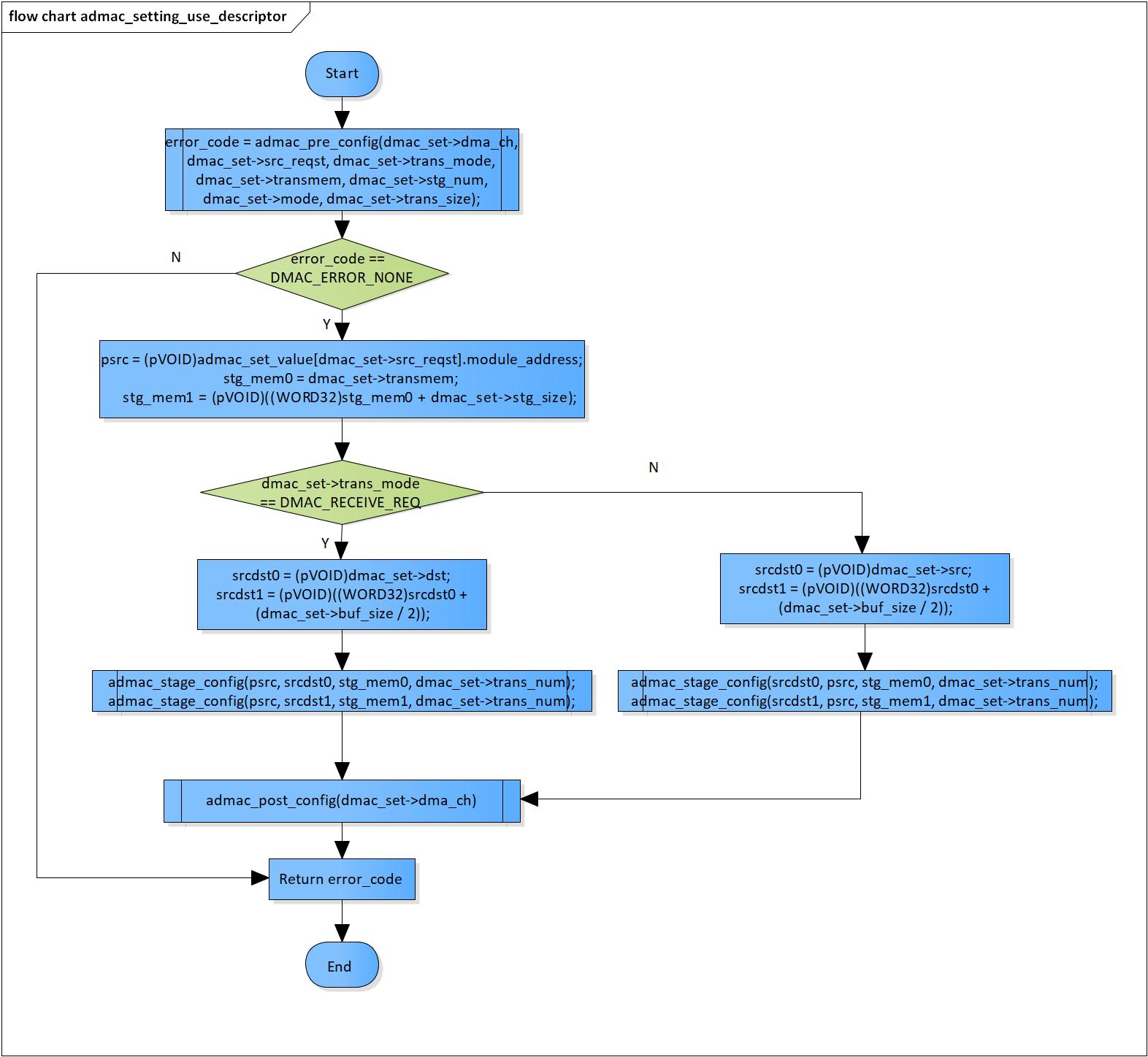


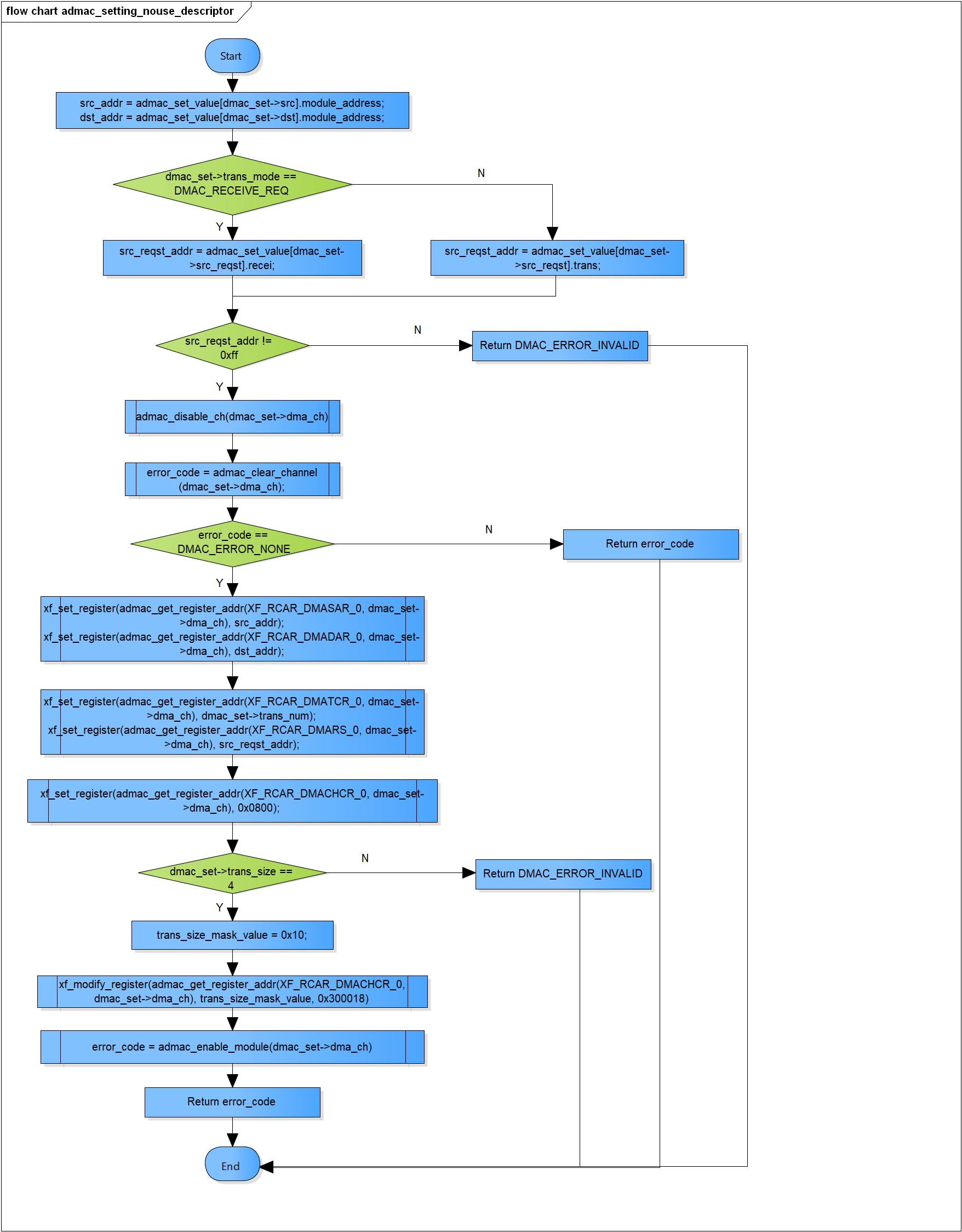
Figure 3‑18 admac\_setting\_use\_descriptor flowchart

### admac\_setting\_nouse\_descriptor

DD\_PLG\_RDR\_07\_019

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | inline dmac\_error\_code admac\_setting\_nouse\_descriptor(const DMAC\_SETTING \*dmac\_set) | | | |
| **Function** | This function is to set DMAC module without using descriptor memory. | | | |
| **Arguments** | Type | Name | I/O | Description |
| const DMAC\_SETTING \* | dmac\_set | I | Pointer to DMAC parameter structure |
| **Return value** | DMAC\_ERROR\_OUT\_RANGE | | Cannot clear value in DMAC’s registers  Cannot enable DMAC module | |
| DMAC\_ERROR\_NONE | | Normal end | |
| **Description** | * admac\_setting\_nouse\_descriptor command processing:   - Set necessary registers for ADMAC not using descriptor memory | | | |

[Covers: FD\_PLG\_RDR\_005]

Figure 3‑19 admac\_setting\_nouse\_descriptor flowchart

### admac\_resetup

DD\_PLG\_RDR\_07\_020

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Syntax** | inline dmac\_error\_code admac\_resetup(DMAC\_SETTING dmac\_set)s | | | | |
| **Function** | This function is to re-setup DMAC module. | | | | |
| **Arguments** | Type | Name | I/O | | Description |
| DMAC\_SETTING | dmac\_set | I | | DMAC parameter structure |
| **Return value** | DMAC\_ERROR\_OUT\_RANGE | | | Cannot enable DMAC module | |
| DMAC\_ERROR\_NONE | | | Normal end | |
| **Description** | * admac\_resetup command processing:   - Re-setup DMAC module | | | | |

[Covers: FD\_PLG\_RDR\_015, FD\_PLG\_RDR\_017]

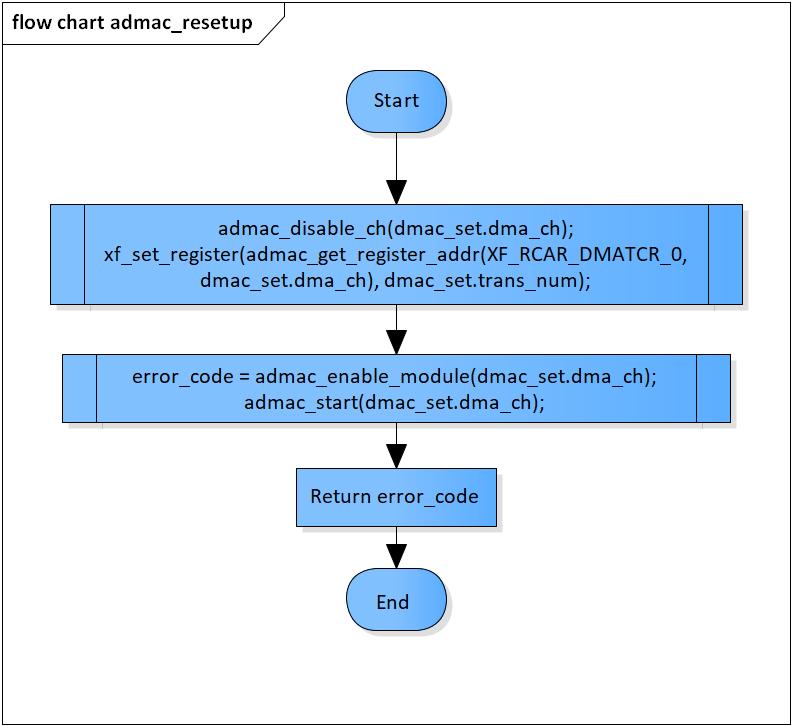


Figure 3‑20 admac\_resetup flowchart

### admac\_pause

DD\_PLG\_RDR\_07\_021

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Syntax** | VOID admac\_pause(ADMA\_CH n) | | | |
| **Function** | This function is to pause DMAC channel N | | | |
| **Arguments** | Type | Name | I/O | Description |
| ADMA\_CH | n | I | Channel number  Valid range: [0: 31] |
| **Return value** | None | | | |
| **Description** | * admac\_pause command processing:   - Disable DMAC channel N | | | |

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

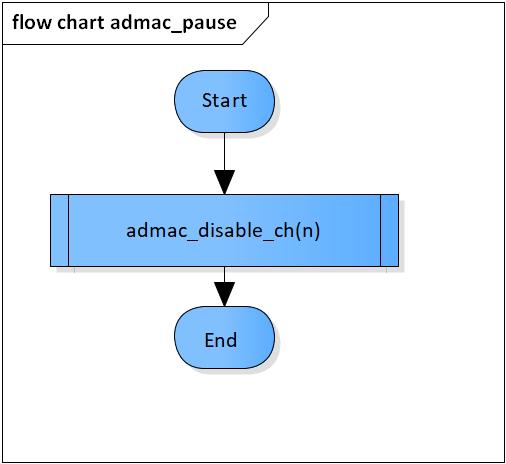


Figure 3‑21 admac\_pause flowchart

# Revision history

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Date** | **Page** | **Content** | **Approved** | **Changed** |
| 1.0.0 | Nov 14 2018 | - | First Edition issued | Vu Phan | Vu Phan |
| 1.1.0 | Dec 10 2018 | - | Add traceability ID | Vu Phan | Ngu Pham |
| 1.2.0 | Jan 03, 2019 | - | Add range for input parameters | Vu Phan | Tien Tran |
| 1.3.0 | Mar 20, 2019 | 33, 34 | Modify transfer size correct to swap channel issue |  | Ngu Pham |