RESILTECH | Technologies for Resilience

SWD_006_PIA141
0.22
22/12/2025
219 pages

SW User Guide

M33 STL SW User Guide

Addressee: Renesas Electronics Corporation	Author	Andrea Chimenti	
	Check	Leonardo Cappelli	
	Authorization	Francesco Rossi	
	Distribution	Confidence	● Confidential ○ Normal
		Controlled copy	○ Yes ● No

Commented [NO1]: Since this is the cover title, it is better to write the full product name without using an abbreviation.

Commented [MD2R1]: [UG_1] In general template-level information/structure should not be changed. However, as a first step, we should understand how Renesas wants to use UG/SM documents and evaluate an alignment to the other MCAL CorTst documents

Commented [LM3R1]: Feedback from Renesas to be provided

REVISIONS INDEX			
REVISION	DATE	DESCRIPTION	AUTHOR
0.01	26/05/2025	Initial draft	Marco Novi
0.02	16/07/2025	Updated sections 2.3 and 3.3.4. Added section 3.6.3.2	Marco Novi
0.03	17/07/2025	Updated sections 3.6.2, 3.6.3.1 and 3.6.3.2	Marco Novi
0.04	07/08/2025	Added sections for MPU and NVIC APIs	Andrea Chimenti
0.05	29/08/2025	Updated document to address Renesas open points	Andrea Chimenti
0.06	29/08/2025	Updated sections 3.6.2.3 and 3.6.3.2. Added sections 3.6.3.3 and 3.6.3.4	Marco Novi
0.07	03/09/2025	Added section 3.6.3.5 and sub-sections to describe the building and linking process to be followed to use Tasking	Lavinia Masini
0.08	04/09/2025	Added section 3.6.3.6 and sub-sections to describe the building and linking process to be followed to use Windriver	Marco Novi
0.09	17/09/2025	Updated Section 3.3.6 adding new STL Tests. Aligned Section 3.6.3.1 to the new approach. Added Append B to describe the STL Tests content in details	Lavinia Masini
0.10	18/09/2025	Updated section 3.6.3 and its sub-sections to align integration guidelines to the new approach	Lavinia Masini
0.11	22/09/2025	Updated document to address JIRA ticket RNSM33STL-8	Andrea Chimenti
0.12	25/09/2025	Updated sections 2.2 and 2.3. Added section 3.6.3.7 to describe the integration guidelines to be used for Hightec compiler. Updated section A to provide more details about the m33_stl_registerInfo_t array	Lavinia Masini
0.13	25/09/2025	Updated sections 3.2, 3.3.1, 3.3.4 and 3.6.2.1. Deleted reference to iodef.h and system_init.c files in the section 3.6.3	Marco Novi
0.14	23/10/2025	Updated section 3.6.3 and its sub-sections to align integration guidelines to the new approach	Marco Novi
0.15	28/10/2025	Updated the document to address JIRA ticket RNSM33STL-8 (open points from ID_8 to ID_25)	Andrea Chimenti
0.16	05/11/2025	Updated the document to address JIRA ticket RNSM33STL-8 (open points from ID_26 to ID_33)	Andrea Chimenti
0.17	03/12/2025	Updated section 3.6.3.5	Marco Novi
0.18	04/12/2025	Updated section 3.6.3.4	Marco Novi
0.19	10/12/2025	Updated 3.6.3 and sub-sections to address JIRA Ticket RNSM33STL-8	Marco Novi
0.20	11/12/2025	Updated section 3.6.3.2, adding a note regarding the GCC usage (Jira Ticket RNSM33STL-71)	Lavinia Masini
0.21	15/12/2025	Updated document to incorporate Renesas' feedback and findings on M2 milestone (see Jira Ticket RNSM33STL-90)	Lavinia Masini
0.22	22/12/2025	Updated section 2.2, removing the sentence related to latent faults to address Jira Ticket RNSM33STL-90	Lavinia Masini

List of Figures

List of Tables

1 Introduction

This document provides an overview of the Cortex-M33 processor Software Test Library (STL).

1.1 Purpose

The purpose of this document is to explain how to integrate the Cortex-M33 processor Software Test Library in an application running on R-Car U5L1, U5L2, and U5L4 devices.

1.2 Acronyms/Terms Definition

This section reports the definition of the main terms and acronyms used in this document.

Term/Acronym	Definition
API	Application Programming Interface
AoU	Assumption of Use
CPU	Central Processing Unit
FP	Floating Point
GP	General Purpose
HDL	Hardware Description Language
HW	Hardware
ID	Identifier
NVIC	Nested Vectored Interrupt Controller
MPU	Memory Protection Unit
STL	Software Test Library
SW	Software
WDT	Watchdog Timer

Table 1 - Acronyms

Commented [NL4]: Should it be "The Cortex-M33 processor"
See other cases in the document

Commented [MD5R4]: [UG_2] Ok to update

Commented [AC6R4]: Updated

Commented [NL7]: Should it be "The Cortex-M33 processor"
See other cases in the document

Commented [MD8R7]: [UG_3] Ok to update

Commented [AC9R7]: Updated

Commented [RV10]: Need to use formal names as R-Car/U5Lx

Commented [MD11R10]: [UG_4] Ok to update

Commented [AC12R10]: Updated

Commented [RV13R10]: OK

1.3 Referenced Document

Table 2 reports the reference documents indicated in this document.

Ref. and document code	Title
[1]. 100230_0100_08_en	Arm® Cortex®-M33 Processor - Technical Reference Manual
[2]. SWD_007_PIA141	M33 STL SW Safety Manual

Table 2 - Reference documents

2 Overview

2.1 About the processor

The Cortex®-M33 processor is a low gate count, highly energy efficient processor that is intended for microcontroller and deeply embedded applications. The processor is based on the Arm®v8-M architecture. More details about Cortex-M33 can be found in [1].

2.2 About STL

The STL provides diagnostic testing for the Cortex-M33 processor.

The tests can be run to prevent the following:

- Faults that cause single points of failure.

The STL is not a replacement for scan-based manufacturing tests but can be used for additional in-field testing at either boot time or runtime.

The STL is designed to detect faults in the functional logic (excluding memories). Any logic that is not used in a functional environment (such as debug and trace) is excluded. The fault detection is realized by software that performs self-tests using a sequence of instructions.

Commented [RV14]: Plus WDG?

Commented [MD15R14]: [UG_7] Ok to update the sentence. We will remove the word "entirely" and add a sentence to include the WDT

Commented [AC16R14]: Updated

Commented [RV17R14]: OK

Additionally, the STL is adjuvated by the required external WDT that adds detection capabilities for the hang of the CPU and other similar failures once they are exercised by SW, including the STL.

The M33_STL and M33_STL_NVIC APIs do not trigger any interrupt or exception and there are no STL Tests that test the CPU behavior when an interrupt or an exception is triggered.

Commented [NL18]: Is this right when The STL source code triggered a MemManage fault exception during the MPU module test.
Please help me check this if I misunderstand, please correct me.

Commented [LM19R18]: Rephrased and added more details

The M33_STL_MPU API, when executed to test the MPU functionality (e.g., mode parameter is 0 or 2), triggers two MPU exceptions, generating two MemManage faults, which are handled by the related exception handler, implemented by the STL.

More details on the STL can be found in sec. 3.3 and sub-sections.

2.3 Tools

The following table will report the list of tools used during the STL life cycle.

Purpose	Vendor	Tool	Version
HDL simulator	Synopsys	VC Z01X	W-2024.09-SP2-1
Software Development Tools	Green Hills Software	GHS	v2024.1.4
	Free Software Foundation	Arm GNU Toolchain	14.3.1.arm
	IAR Systems AB	IAR Embedded Workbench for Arm	EWARM V9.70.1

	Arm (Keil)	Arm Toolchain for Embedded	v6.22.1
	TASKING B.V.	TASKING VX-toolset for Arm	v8.0rb1
	Wind River Systems, Inc	Wind River Diab Compiler	v7.0.6.0
	HighTec EDV-Systeme GmbH	HighTec ARM Development Platform	v9.0.0
	Renesas Electronics Corporation	e² studio	2025-10 (25.10.0)

Table 3 – Tools

3 STL DESCRIPTION

The objective of the STL is to verify the correct functionality of the CPU by adopting an instruction-based diagnosis, to detect permanent hardware failures of the CPU Core.

The STL software comprises several main components.

The main components are:

- Two APIs that allow the user to configure and run the core STL (sections 3.3.1 and 3.3.2)
- An STL scheduler, which executes the STL Tests selected by the user and reports the result of the STL, at the end of its execution (section 3.3.5)
- A set of STL Tests, composed of one or more STL Test Elements, that test instructions or registers belonging to the processor core. Each STL Test and therefore each STL Test Element are implemented in Armv8-M assembly.
- One API that allows the user to run the NVIC STL test (section 3.3.3)
- One API that allows the user to run the MPU STL tests (section 3.3.4)

Commented [RV20]: Minor but remove space. See other cases in the document

Commented [MD21R20]: [UG_8] The document will be reviewed for clean up

Commented [AC22R20]: Updated

Commented [RV23R20]: OK

3.1 STL Overview

The STL for the Cortex-M33 processor has 3 functional levels:

- **STL APIs**: they represent the highest level of the STL architecture that allows the caller to configure the environment and execute the STL. These APIs are written in C language.
- **STL internal functions**: it is the level below, which implements the functions used by the APIs for internal operations required by the top-level function. The scheduler function that allows the execution of the user-selected STL Tests is implemented at this level. These functions are written in C language.
- **STL Test**: The STL is composed of several STL Tests, each of them focused on testing specific instructions and registers. The scheduler executes the STL Tests according to the user configuration. Each STL Test is independent of the others. It is written in Assembly language for Cortex-M33 with a C level interface. Each STL Test performs some operations to save the user application context and guarantee that the user configuration or General Purpose and control registers are restored when the STL returns the control to the caller.

3.2 STL files

The following table contains the full list of the STL files and the description of the content of each file.

File name	Description
M33_STL.c	C code file containing the definitions of the STL API functions and other internal functions.
M33_STL.h	Header file containing the STL APIs declaration.
m33_stl_config_Var.h	This file contains the macros used by the user to enable or disable an STL Test compilation.
m33_stl_utility.h	Header file containing the #define directives for macros used by M33_STL.c file.
m33_stl_compare.asm	Assembly file containing the compare routines used by STL Test Elements to check the correctness of the instructions/registers under test.
m33_stl_constants.h	Header file containing the #define directives used by the assembly files.
m33_stl_datavect_Data.c	File containing the definition of the input data, used by those STL Test Elements that require a data vector as input.
m33_stl_datavect_Data.h	Header file containing the declaration of input data, used by those STL Test Elements that require a data vector as input.
m33_stl_datavector_test_elements.asm	Assembly file containing the diagnostic procedures implemented by those STL Test Elements that use data vectors as input.
m33_stl_in_context_switch.asm	Assembly file containing the routines used for saving in the stack General Purpose and Floating-Point Registers. Additionally, it contains the initialization procedures for these registers.
m33_stl_out_context_switch.asm	Assembly file containing the routines used for restoring from the stack General Purpose and Floating-Point Registers.
m33_stl_cpu_n<num>.asm	Assembly code files containing the implementation m33_stl_cpu_n<num> functions, where num goes from 0 to the total number of STL Tests - 1
m33_stl_cpu_n<num>.h	Header files containing the declaration of m33_stl_cpu_n<num> functions, where num goes from 0 to the total number of STL Tests - 1
m33_stl_mpu_n<num>.h	Header files containing the declaration of m33_stl_mpu_n<num> functions, where num is 0 or 1
m33_stl_mpu_n<num>.asm	Assembly code files containing the implementation m33_stl_mpu_n<num> functions, where num is 0 or 1
m33_stl_nvic_n<num>.h	Header files containing the declaration of m33_stl_nvic_n<num> functions, where num is 0
m33_stl_nvic_n<num>.asm	Assembly code files containing the implementation m33_stl_nvic_n<num> functions, where num is 0
m33_stl_registerInfo.h	Header file contains the #define directives used by the m33_stl_registerInfo.c and assembly files
m33_stl_registerInfo.c	C code containing the arrays to test the registers of the NVIC and MPU modules
m33_stl_utils.h	Header file containing the support functions used by the C code
m33_stl_utils.asm	Assembly file containing the support functions used by the C code

m33_stl_exceptions.c	C source file containing the exception handlers used by STL Test Elements, including m33_stl_mem_manage_handler, which handles memory faults triggered by MPU tests, and the m33_stl_mem_manage_user_handler.
m33_stl_exceptions.h	Header file containing the #define directives for macros used by m33_stl_exceptions.c file.

Table 4 - STL files description

3.3 M33 STL functions

The STL provides the following APIs:

- M33_STL_Config to allow the user to specify the base address of the data structure that contains the information related to the STL execution
- M33_STL to run the STL Tests selected by the user
- M33_STL_NVIC to verify the registers of the NVIC module by running specific STL Tests
- M33_STL_MPU to run the diagnostic tests for the correct functionality of the MPU module and to verify the MPU registers

M33_STL_Config function allows the user to specify the base address of the data structure used by the STL to keep information about its status (see section 3.3.2 for more details).

M33_STL runs specific STL Tests according to the user specification.

M33_STL_NVIC runs the STL Tests that verify the registers of the NVIC module.

M33_STL_MPU runs the STL Tests that verify the registers and the functionality of the MPU module.

3.3.1 M33_STL

This function runs the STL Tests, according to the user configuration and the values of bitMaskArray[], starting from the beginning until the final is reached or until a fault is detected. In case of fault detection, the STL stops its execution, does not execute the STL Tests following that one where the failure occurs, and returns control to the caller. The length of the bitMaskArray[] is obtained from the number of the STL Tests as follows:

length = (Test_ID of the last Test / 32) + 1

Additionally, the M33_STL function provides the user with the interface to enable or disable the force fail functionality.

The M33_STL function is written in the C programming language and its signature is defined as follows:

uint32_t M33_STL (const uint32_t bitMaskArray[], uint8_t forceFail);

Table 5 describes in more detail the inputs and outputs of the function.

Table ID	Parameter type	C type	Name	Description
1	Input	const uint32_t	bitMaskArray[]	It identifies the array of bitMask values required to specify the STL Tests to be executed among the compiled ones. The length of the bitMaskArray array is obtained from the number of the STL Tests as follows: length = (Test_ID of the last Test / 32) + 1 Every bit of each bitMask represents an STL Test (i.e., bit 0 of bitMaskArray[0] identifies the STL Test number 0, bit 1 of bitMaskArray[0] identifies the STL Test number 1, bit 0 of bitMaskArray[1] identifies the STL Test number 32 and so on) with the following convention: • **0**: indicates that the corresponding STL Test is not selected for the execution • **1**: indicates that the corresponding STL Test is selected for the execution The bitMaskArray[] is structured as in the following representation: • bitMaskArray[] = {bitMask0, bitMask1, ..., bitMaskX} with X = length – 1 The possible value of a single bitMaskX can be: • 0 <= bitMaskX < 2^32
2	Input	uint8_t	forceFail	This value allows the user to force the STL to fail. This parameter has two valid values: • **0**: disables force fail functionality • **1**: enable force to fail functionality

Table ID	Parameter type	C type	Name	Description
3	Output	uint32_t	N.A.	Global pass/fail result of the STL: • M33_STL_PASS (0x600D) No faults detected. All the executed STL Tests ended with success. • M33_STL_FAIL (0xBAD): ○ The STL detected a fault ○ The forceFail is enabled ○ The bitMaskArray or the forceFail parameters are not valid. More details on the STL result are reported in m33_stl_dataStruct_t structure (section 3.4).

Table 5 - M33_STL Interface

3.3.2 *M33_STL_Config*

This function is used to specify the base address of the data structure used to record information about the STL status, the last executed STL Test, and the fault type, if any.

It is written in C programming language and its signature is defined as follows:

uint32_t M33_STL_Config (m33_stl_dataStruct_t* const dataStruct)

Table 6 describes in more detail the input/output of the function.

Table ID	Parameter type	C type	Name	Description
1	Input	m33_stl_dataStruct_t* const	dataStruct	The base address of the data structure is used to save information about STL execution. Valid range is > 0.
2	Output	uint32_t	N.A.	Global pass/fail result of library initialization: • M33_STL_PASS (0x600D) *m33_stl*_dataStruct_t address successfully initialized • M33_STL_FAIL (0xBAD) m33_stl_dataStruct_t address initialization failed

Table 6 - M33_STL_Config Interface

3.3.3 M33_STL_NVIC

This function runs the STL tests for the register of the NVIC module. If a fault is detected, the STL stops execution and skips the remaining NVIC registers to be tested, following the one where the failure occurred, returning control to the caller.

If the test for all the NVIC registers completes successfully, the function sets the STL Status field of the data structure to Completed. Otherwise, it sets the status to Failed. In this case, if the fault type is not a Data Mismatch (i.e., not caused by a failed comparison operation within the Test Element), it also updates the fault type field to Unexpected Error.

For more details about the data structure, see section 3.4.

The M33_STL_NVIC function is written in the C programming language, and its signature is defined as follows:

uint32_t M33_STL_NVIC (void);

Table 7 describes in more detail the input/output of the function.

Table ID	Parameter type	C type	Name	Description
1	Output	uint32_t	N.A.	Global pass/fail result of the STL: • M33_STL_PASS (0x600D) No faults detected. All the values of the registers correspond with the expected values. • M33_STL_FAIL (0xBAD): ○ The dataStruct value is Null. ○ The value read from the register does not correspond with the expected value. More details on the STL result are reported in m33_stl_dataStruct_t structure (section 3.4.3).

Table 7 - M33_STL_NVIC Interface

3.3.4 M33_STL_MPU

This function executes the STL tests for the MPU module, which may include functional tests, register tests, or both, depending on the mode parameter.

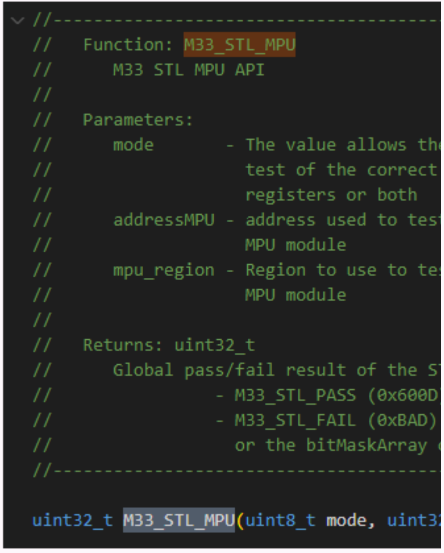
If a fault is detected, the STL stops execution, skips any remaining MPU functional or register tests following the one where the failure occurred, and returns control to the caller.

If all the selected STL tests complete successfully, the function sets the STL Status field of the data structure to Completed. Otherwise, it sets the status to Failed. In this case, if the Fault Type is not Data Mismatch (i.e., not caused by a failed comparison in a Test Element), the field is updated to Unexpected Error. For more details about the data structure, see section 3.4.

The M33_STL_MPU function is written in the C programming language, and its signature is defined as follows:

uint32_t M33_STL_MPU (uint8_t mode, const uint32_t *addressMPU, uint8_t mpu_region);

Commented [NL24]: Is there a mismatch between source code and UM?
Is it a pointer declaration?
E.g. uint32_t *addressMPU



```
//---------------------------------------
//   Function: M33_STL_MPU
//      M33 STL MPU API
//
//   Parameters:
//      mode       - The value allows the
//                   test of the correct
//                   registers or both
//      addressMPU - address used to test
//                   MPU module
//      mpu_region - Region to use to tes
//                   MPU module
//
//   Returns: uint32_t
//      Global pass/fail result of the S
//                - M33_STL_PASS (0x600D
//                - M33_STL_FAIL (0xBAD)
//                  or the bitMaskArray 
//---------------------------------------

uint32_t M33_STL_MPU(uint8_t mode, uint3
```

Commented [MD25R24]: [UG_9] It is a pointer the UG will be updated

Commented [AC26R24]: Updated

Table 8 describes in more detail the input/output of the function.

Table ID	Parameter type	C type	Name	Description
1	Input	uint8_t	mode	The value allows the user to select whether to launch the test of the correct functionality, the test of the registers, or both. In particular, this parameter has three valid values: • 0: test of the correct functionality • 1: test of the registers • 2: test of the correct functionality and test of the registers
2	Input	const uint32_t *	addressMPU	The address of the memory area configured by the STL to test the correct functionality of the MPU
3	Input	uint8_t	mpu_region	MPU Region ID to be used to test the correct functionality of the MPU module. The valid range is 0x0 to 0xF.
4	Output	uint32_t	N.A.	Global pass/fail result of the STL: • M33_STL_PASS (0x600D) No faults detected. All the values of the registers correspond with the expected values. • M33_STL_FAIL (0xBAD): ○ The dataStruct value is Null. ○ The value read from the register does not correspond with the expected value. ○ The functionality of the MPU module is not correct More details on the STL result are reported in m33_stl_dataStruct_t structure (section 3.4.3).

Table 8 - M33_STL_MPU Interface

3.3.5 Scheduler

The STL runs the STL Tests specified by the user through the M33_STL API input parameters and the m33_stl_config_Var.h

Scheduler configuration

The user can specify the STL Tests to be compiled in the file m33_stl_config_Var.h. In particular, to compile an STL Test, the user has to add the corresponding macro M33_STL_TEST_XY_PRESENT (where XY goes from 0 to the total number of STL Tests - 1) in the m33_stl_config_Var.h file.

Below is an example of the m33_stl_config_Var.h file where the Tests 0, 1, 3, 5, 8 and 9 are compiled:

```
#define M33_STL_TEST_0_PRESENT
#define M33_STL_TEST_1_PRESENT
#define M33_STL_TEST_3_PRESENT
#define M33_STL_TEST_5_PRESENT
#define M33_STL_TEST_8_PRESENT
#define M33_STL_TEST_9_PRESENT
```

Commented [NO27]: It might be helpful to explain the meaning of this example (tests 0, 1, 3, 5, 8, and 9 are compiled).

Commented [MD28R27]: [UG_10] Ok to explain the meaning

Commented [AC29R27]: Updated

Commented [LM30R27]: Confirmation on the correctness of the updated from Renesas to be provided

Additionally, the user can select the STL Tests to be executed among the compiled ones. For example, if the user runs the M33_STL API with bitMaskArray[0] = bitMask0 with bitMask0 = 0x0000010B (assuming that forceFail is set to 0), considering also what has been specified above, the STL will only run the STL Tests 0, 1, 3, and 8.

For more information about the settings of the bitMaskArray, see Section 3.6.2.

3.3.6 STL Test Description

An STL Test is a routine that tests a part of an HW block. The table below reports the details of each implemented STL Test. Information about the specific instructions and registers tested in each STL Test can be found in Table 13.

STL Test Name	Test_ID Value (in hex)	The objective of the STL Test
m33_stl_cpu_n000	0x00	ADC, ADD, AND and TST instructions
m33_stl_cpu_n001	0x01	ASR, BIC and OR instructions

Commented [NO31]: Is the Test_ID assigned to the list order in this table? If so, it would be better to have a Test_ID column in this table.

Commented [MD32R31]: [UG_11] Not clear: do you mean to add a column with STL_TEST_# in line with the define #define M33_STL_TEST_#_PRESENT

Commented [RV33R31]: I have same question. The API requires the "Test_ID" but nowhere this is defined concretely.

Commented [LM34R31]: Test_ID column added to clearly report the Test_ID field of each STL Test. Is that in line with your expectation?

m33_stl_cpu_n002	0x02	Logical Shift instructions
m33_stl_cpu_n003	0x03	Move instructions
m33_stl_cpu_n004	0x04	MUL, RSBS, ROR and Subtract instructions
m33_stl_cpu_n005	0x05	Load and Store instructions
m33_stl_cpu_n006	0x06	Branch and Jump instructions (Control flow monitoring test)
m33_stl_cpu_n007	0x07	Division integer instructions
m33_stl_cpu_n008	0x08	Compare instructions
m33_stl_cpu_n009	0x09	Test FP Single Precision registers
m33_stl_cpu_n010	0x0A	Acquire and Multiple Load and Store instructions
m33_stl_cpu_n011	0x0B	Extend and Reverse instructions
m33_stl_cpu_n012	0x0C	Test GP registers
m33_stl_cpu_n013	0x0D	VFMA instruction with Round towards Minus Infinity mode and with AHP, DZ and FZ enabled - Single precision
m33_stl_cpu_n014	0x0E	Floating Point Convert instructions with Round towards Zero mode - 32-bit Integer
m33_stl_cpu_n015	0x0F	Floating Point Convert instructions with Round towards Minus Infinity mode - 32-bit Integer
m33_stl_cpu_n016	0x10	Floating Point Convert instructions with Round towards Plus Infinity mode - 32-bit Integer
m33_stl_cpu_n017	0x11	Floating Point Convert instructions with Round to Nearest mode - 32-bit Integer
m33_stl_cpu_n018	0x12	Floating Point Convert instructions - 32-bit Integer

Commented [NL35]: Could you please provide a clear description of the test function and list the instructions being tested?
E.g.
This test executes various data processing instructions to detect stuck-at faults.
The Logical Shift instructions tested are: LSL, LSR.

The same requirements apply to all STL test parts in this table.

Commented [MD36R35]: [UG_12] We will add an appendix to the UG with the required details

Commented [LM37R35]: [UG_12] The information about the detailed content of each Test will be added in a new appendix for the next document release

Commented [LM38R35]: Appendix B, Table 13 contains a detailed description of the content of each Test

Commented [LM39R35]: Confirmation on the correctness of the updated from Renesas to be provided

m33_stl_cpu_n019	0x13	VFMS instruction with Round to Nearest mode - Single precision
m33_stl_cpu_n020	0x14	Arithmetic instructions
m33_stl_cpu_n021	0x15	Load instructions
m33_stl_cpu_n022	0x16	Compare instructions
m33_stl_cpu_n023	0x17	Shift instructions
m33_stl_cpu_n024	0x18	Logical instructions
m33_stl_cpu_n025	0x19	Bit Manipulation and CLZ instructions
m33_stl_cpu_n026	0x1A	Multiply instructions
m33_stl_cpu_n027	0x1B	Move instructions
m33_stl_cpu_n028	0x1C	Pack Halfword instructions
m33_stl_cpu_n029	0x1D	Saturate instructions
m33_stl_cpu_n030	0x1E	Reverse and Rotate instructions
m33_stl_cpu_n031	0x1F	Signed Add and Sub instructions
m33_stl_cpu_n032	0x20	Signed Multiply instructions
m33_stl_cpu_n033	0x21	Signed Extend instructions
m33_stl_cpu_n034	0x22	TEQ and TST instructions
m33_stl_cpu_n035	0x23	Unsigned Add and Sub instructions
m33_stl_cpu_n036	0x24	Unsigned Multiply instructions
m33_stl_cpu_n037	0x25	Unsigned Saturate instructions
m33_stl_cpu_n038	0x26	Unsigned Extend instructions
m33_stl_cpu_n039	0x27	VADD, VDIV, VMUL, VSQRT and VSUB Floating Point instructions with Round towards Minus

Commented [NL40]: Should it list specific purpose of these test instructions as other cases?
E.g. Signed Extend instructions

Commented [MD41R40]: [UG_13] It will be updated with the specific purpose of the test

Commented [AC42R40]: Updated

		Infinity mode and with AHP, DZ and FZ enabled - Single precision
m33_stl_cpu_n040	0x28	VADD, VDIV, VMUL, VSQRT and VSUB Floating Point instructions with Round to Nearest mode - Single precision
m33_stl_cpu_n041	0x29	VADD, VDIV, VMUL, VSQRT and VSUB Floating Point instructions with Round towards Plus Infinity mode - Single precision
m33_stl_cpu_n042	0x2A	VADD, VDIV, VMUL, VSQRT and VSUB Floating Point instructions with Round towards Zero mode - Single precision
m33_stl_cpu_n043	0x2B	VABS and VNEG instructions - Single Precision
m33_stl_cpu_n044	0x2C	VCMP, VCMPE, VSEL, VMAXNM and VMINNM instructions - Single Precision
m33_stl_cpu_n045	0x2D	Floating Point Round instructions - Single Precision
m33_stl_cpu_n046	0x2E	Floating Point Round instructions with DN enabled - Single Precision
m33_stl_cpu_n047	0x2F	VCVTB.F16.F32 and VCVTT.F16.F32 instructions with Round to Nearest mode and AHP enabled
m33_stl_cpu_n048	0x30	VCVTT and VCVTB with Round to Nearest mode and DN enabled
m33_stl_cpu_n049	0x31	Floating Point Convert instructions - Fixed-Point
m33_stl_cpu_n050	0x32	Floating Point Convert instructions - Half Precision
m33_stl_cpu_n051	0x33	Floating Point Convert instructions - Single Precision
m33_stl_cpu_n052	0x34	VMOV instructions
m33_stl_cpu_n053	0x35	VNMLA instruction with Round towards Zero mode - Single precision

m33_stl_cpu_n054	0x36	Floating Point Convert instructions with Round to Nearest mode - Single Precision
m33_stl_cpu_n055	0x37	VFMLS instruction with Round towards Plus Infinity mode - Single precision
m33_stl_cpu_n056	0x38	VMLS instruction with Round to Nearest mode - Single precision
m33_stl_cpu_n057	0x39	Floating Point Convert instructions with Round towards Plus Infinity mode - Single Precision
m33_stl_cpu_n058	0x3A	VMLA and VNMUL instructions with Round towards Minus Infinity mode and with AHP, DZ and FZ enabled - Single precision
m33_stl_cpu_n059	0x3B	FP Load and FP Store instructions
m33_stl_cpu_n060	0x3C	Floating Point Convert instructions with Round towards Minus Infinity mode - Single Precision
m33_stl_cpu_n061	0x3D	VFNMS instruction with Round towards Zero mode - Single precision
m33_stl_cpu_n062	0x3E	VFNMA instruction with Round towards Plus Infinity mode - Single precision
m33_stl_cpu_n063	0x3F	Floating Point Convert instructions with Round towards Zero mode - Single Precision
m33_stl_cpu_n064	0x40	Test of control registers
m33_stl_cpu_n065	0x41	Test of control registers
m33_stl_cpu_n066	0x42	Test of control registers
m33_stl_cpu_n067	0x43	Test of the dual-issue strategy - Integer instructions
m33_stl_cpu_n068	0x44	Test of the dual-issue strategy - FP and Integer instructions
m33_stl_cpu_n069	0x45	VADD.F32, VCMP.F32, VCMPE.F32 and VMUL.F32 instructions

m33_stl_cpu_n070	0x46	CLZ instruction
m33_stl_cpu_n071	0x47	Strategy to test internal paths
m33_stl_cpu_n072	0x48	Strategy to test internal paths of FP block
m33_stl_mpu_n000	0x00	Functionality of MPU module
m33_stl_mpu_n001	0x01	Control Registers of the MPU module
m33_stl_nvic_n000	0x00	Control Registers of the NVIC module

Table 9 – STL Tests description

Commented [NL43]: Should it list specific registers **for diagnoses**?
e.g. MPU_CTRL, MPU_RNR, MPU_MAIR0/1 registers

Commented [MD44R43]: [UG_14] MPU register info in section A.2

Commented [NL45]: Should it list specific registers **for diagnoses**?

Commented [MD46R45]: [UG_15] NVIC register info in section A.1

3.4 Data Structure

The STL writes information about its status in a data structure. The base address of this data structure is chosen by the user through the M33_STL_Config API, and it is composed of the following fields:

- **Test_ID:** offset 0x0
- **STL_Status**: offset 0x4
- **Fault_Type:** offset 0x8

The STL initializes these fields to the default value 0x0.

3.4.1 Test_ID_ Field

This field contains the ID of the last executed STL Test or, in case of a failure, the ID of the first failed STL Test. In particular, the field’s format is the following:

<table>
<tr><td colspan="24">Reserved</td><td colspan="8">ID</td></tr>
<tr><td>31</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>8</td><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td><td>0</td></tr>
</table>

Figure 1 – Test_ID Field

Below is the definition of the Test_ID field.

- **Reserved [31-8]:** Reserved, Read as Zero, Write Ignored
- **ID [7-0]:** ID of the last executed STL Test or the first failed STL Test, if any

If the STL fails with the Fault_Type field as 0x2 (Force Fail Functionality enable), the Test_ID field is not updated by the STL and it will be the default one (0x0).

3.4.2 *STL_Status Field*

The field contains the current status of the STL. In particular, the field’s format is the following:

Reserved																												Status		
31																											3	2		0

Figure 2 – STL_Status Field

Below is the definition of the STL_Status field.

- **Reserved [31-4]:** Reserved, Read as Zero, Write Ignored
- **Status [2-0]:** STL Status. This field is used to indicate the current STL Status

0x0	**Setup**	STL has been called and it is in the set-up phase
0x1	**Completed**	The selected subset (or entire set) of tests has been successfully executed
0x2	**STL Test Completed**	One single STL Test successfully executed
0x3	**Failed**	Fault detected in the last executed STL Test
0x4	**Running**	STL Test running
0x5…F	**Error**	Illegal value in STL Status field

3.4.3 *Fault_Type Field*

The field contains information about the type of fault. The value of this field is valid only if the Status sub-field value in the STL_Status field is 0x3 (Failed).

Reserved																												Fault Type	
31																											2	1	0

Figure 3 – Fault_Type Field

Below is the definition of the Fault_Type field.

- **Reserved [31-4]:** Reserved, Read as Zero, Write Ignored
- **Fault Type [3-0]:** Type of the fault. Possible values are listed in the table below:

0x0	**Data Mismatch**	The comparison between the expected value and the result of the operation performed by the Test Element produces a mismatch
0x1	**Wrong Input Parameter**	One or more input parameters passed to M33_STL API are not correct

0x2	**Force Fail Functionality enabled**	The M33_STL_API has returned M33_STL_FAIL because the user has activated the force fail functionality
0x3	**Unexpected Error**	Unexpected error
0x4…F	**Error**	Illegal value in Fault_Type field

The Fault_Type field is set to “Data Mismatch” if an STL Test ends with a failure (i.e., the result of the operation under test in an STL Test Element is not equal to its expected value). Otherwise, the Fault_Type field is set to "Force Fail Functionality enabled" if the M33_STL function is passed the forceFail value set to 1, or it is set to “Wrong Input Parameter” if the M33_STL function is passed a bitMask value that attempts to select an STL test that was not configured to be present in the build by the m33_stl_config_Var.h file, or if the passed forceFail value is different from 0 (forceFail disabled) or 1 (forceFail enabled).

Fault_Type field is set to “Unexpected Error” to manage unexpected error situations due to faults in the memory area containing the m33_stl_dataStruct_t structure.

3.5 Release directory structure

This section shows the STL directory structure:

```
diagnostic
|   |-- common
    |   |-- inc
    |   |-- src
|   |-- scheduler
    |   |-- inc
    |   |-- src
|   |-- tests
    |   |-- cpu
        |   |-- inc
        |   |-- src
    |   |-- mpu
        |   |-- inc
        |   |-- src
    |   |-- nvic
        |   |-- inc
        |   |-- src
```

Commented [LM47]: [UG_16] Regarding this comment, since the directory structure has been updated, it has been removed

Commented [LM48R47]: Renesas to confirm this is ok

Details of the above directories structure:

Directory/File	Purpose
diagnostic	Top-level directory for STL code
diagnostic/common/inc	Directory containing header files related to the common functions
diagnostic/common/src	Directory containing source files related to the common functions
diagnostic/scheduler/inc	The directory containing header files related to the scheduler function
diagnostic/scheduler/src	The directory containing source files related to the scheduler function
diagnostic/tests/cpu/inc	Directory containing header files related to code implemented to test CPU components
diagnostic/tests/cpu/src	Directory containing source files related to the code implemented to test CPU components
diagnostic/tests/mpu/inc	Directory containing header files related to the code implemented to test the MPU module
diagnostic/tests/mpu/src	Directory containing source files related to the code implemented to test the MPU module
diagnostic/tests/nvic/inc	Directory containing header files related to the code implemented to test the NVIC module
diagnostic/tests/nvic/src	Directory containing source files related to the code implemented to test the NVIC module

Table 10 - STL directories

Commented [NL49]: s it “CPU components”

Commented [MD50R49]: [UG_17] It will be corrected

Commented [AC51R49]: Updated

3.6 STL Integration

3.6.1 Assumptions of Use

It is recommended that the STL’s users (i.e., the integrator or the developer) perform relevant operations to ensure that the CPU Software Test Library software meets the requirements of their application. This can be done by ensuring that the design is in line with the Assumptions of Use.

In [2], the list of the STL AoUs to be met is reported.

3.6.2 STL software

Guidelines for integrating the Software Test Library (STL) software are reported in the following paragraphs.

3.6.2.1 M33_STL

The M33_STL.h file has to be included in the .c file where the STL is called from. This is required to access the STL's APIs, whose prototype is defined in that header file. Moreover, a variable of type m33_stl_dataStruct_t (i.e., m33_stl_dataStruct = exampleDataStruct) has to be allocated in a memory area selected by the user, accessible by the STL for reading and writing and by the user application only for reading. Then, a variable to hold the STL result has to be initialized with a defensive value (i.e., uint32_t result = M33_STL_FAIL). Then, the bitmask is initialized to specify the STL Tests to be run, among the compiled ones. In particular, to execute an STL Test, the corresponding bit in the bitmask needs to be set to 1 (i.e., m33_stl_cpu_n000 can be selected by setting to 1 bit 0 in the bitmask, m33_stl_cpu_n001 setting the bit 1 and so on). Before launching the M33_STL function, the M33_STL_Config API needs to be called to configure the base address of the data structure containing info about the STL execution.

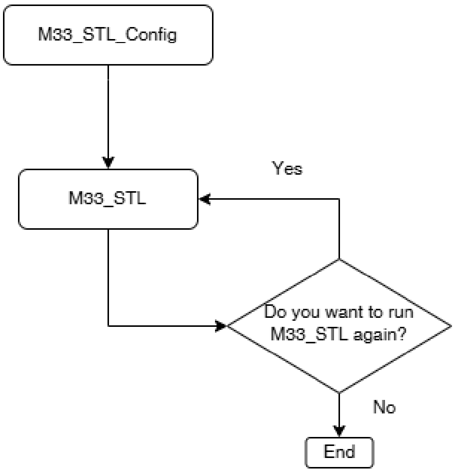


Figure 4 - STL call example

After calling the M33_STL_Config, its result is checked. If it is equal to M33_STL_PASS, then the M33_STL function is called. Once the function's execution is completed, its result is compared to M33_STL_PASS and if they are equal, it means that the STL ended with success, otherwise, a fault has been detected.

Below is an example of how the M33 STL APIs can be called:

```
#include <stdio.h>
#include "M33_STL.h"
```

Commented [NL52]: The M33_STL function tests both the CPU and FPU components.
If the FPU configuration is not enabled before launching the M33_STL function, the STL will fail.
However, I did not find any mention of the FPU configuration in this User Manual (UM).
Would it be necessary to add a note about this?

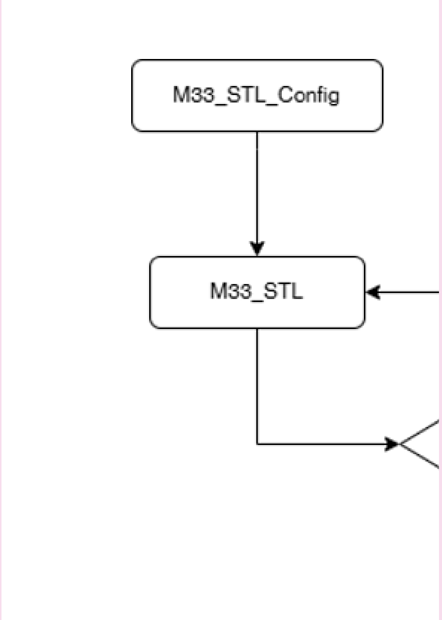


Figure 4 - STL call

Commented [LM53R52]: There is an AoU (M33_STL_AOU_016) in the Safety Manual (SWD_007, section 4, table 4) requiring that before running FP-related STL Tests, the user shall enable the access to the FP coprocessor.
Section 3.6.2.1 of the User Guide is meant to provide information about a generic execution of the STL, without reporting details about specific configurations

```
//Functions to support Fault Injection simulations
void testPassed(void);
void errorHandler(void);

void enableFP (void);

m33_stl_dataStruct_t __attribute__((section(".d_m33_stl_data_struct"))) exampleDataStruct;

void main_pe0(void)
{

    uint32_t bitMask0;
    uint32_t bitMask1;
    uint32_t bitMask2;
    uint32_t result;
    uint8_t forceFail;
    uint32_t bitMaskArray[3];

    // Defensive initialization for M33_STL API input parameters
    bitMask0 = M33_STL_FAIL;
    bitMask1 = M33_STL_FAIL;
    result = M33_STL_FAIL;
    // Defensive initialization of data structure fields
    exampleDataStruct.Fault_Type = 0xFFFFFFFFU;
    exampleDataStruct.STL_Status = 0xFFFFFFFFU;
    exampleDataStruct.Test_ID = 0xFFFFFFFFU;
    // Initialize bitMaskArray parameter
    bitMask0 =0xFFFFFFFF;
    bitMask1 =0xFFFFFFFF;
    bitMask2 =0x000001FF;

    enableFP();

    bitMaskArray[0] = bitMask0;
    bitMaskArray[1] = bitMask1;
    bitMaskArray[2] = bitMask2;

    // Set forceFail parameter to 0x0 - Disable force fail functionality
    forceFail = 0x0U;
    // Configure starting address of data structure containing info
    // about STL execution
    result = M33_STL_Config(&exampleDataStruct);
    if (result == M33_STL_PASS){
        // Calls all Tests indicated by bitMask
        result = M33_STL(bitMaskArray, forceFail);

        if (result == M33_STL_PASS) {
            testPassed();
        }
    }
    errorHandler();

}

void enableFP (void)
{

    uint32_t * cpacr_addr = (uint32_t *) 0xE000ED88;

    *cpacr_addr = *cpacr_addr | 0x00F00000;

}
void testPassed(void)
{
    __asm__ volatile ("add r5, r5, 5");
    __asm__ volatile ("add r5, r5, 5");
    __asm__ volatile ("add r5, r5, 5");
    while(1); // Test Passed
}

void errorHandler(void)
{
    __asm__ volatile ("sub r5, r5, 6");
    __asm__ volatile ("sub r5, r5, 6");
    __asm__ volatile ("sub r5, r5, 6");
    while(1); // Error
}
```

3.6.2.2 M33_STL_NVIC

The M33_STL.h file has to be included in the .c file where the STL is called from. This is required to access the STL's APIs, whose prototype is defined in that header file. Moreover, a variable of type m33_stl_dataStruct_t (i.e., m33_stl_dataStruct = exampleDataStruct) has to be allocated in a memory area selected by the user, accessible by the STL in reading and writing, and by the user application only for reading. Then, a variable to hold the STL result has to be initialized with a defensive value (i.e., uint32_t result = M33_STL_FAIL). Then, the third field of the m33_stl_nvicRegisterInfo array (see section A for details about the array fields) shall be initialized with the expected value. For more information on the m33_stl_nvicRegisterInfo array, see the section A.1.

Before launching the M33_STL_NVIC function, the M33_STL_Config API needs to be called to configure the base address of the data structure containing info about the STL execution.

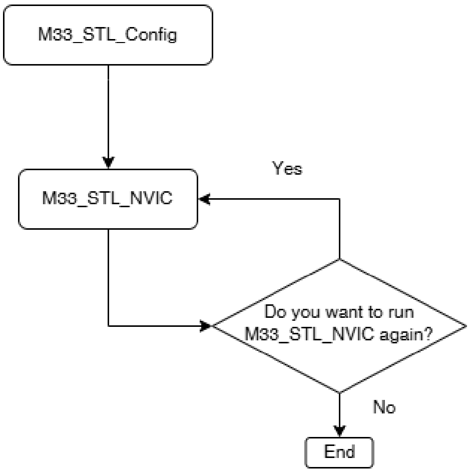


Figure 5 – M33_STL_NVIC call example

After calling the M33_STL_Config, its result is checked. If it is equal to M33_STL_PASS, then the M33_STL_NVIC function is called. Once the function's execution is completed, its result is compared to M33_STL_PASS and if they are equal, it means that the STL ended with success, otherwise, a fault has been detected.

Below is an example of how the M33 STL NVIC APIs can be called:

```
#include <stdio.h>
#include "M33_STL.h"


//Functions to support Fault Injection simulations
```

```
void testPassed(void);
void errorHandler(void);

void enableFP (void);

m33_stl_dataStruct_t __attribute__((section(".d_m33_stl_data_struct"))) exampleDataStruct;

void main_pe0(void)
{
    uint32_t result;

    // Defensive initialization for M33_STL API input parameters
    result = M33_STL_FAIL;
    // Defensive initialization of data structure fields
    exampleDataStruct.Fault_Type = 0xFFFFFFFFU;
    exampleDataStruct.STL_Status = 0xFFFFFFFFU;
    exampleDataStruct.Test_ID = 0xFFFFFFFFU;

    // Configure starting address of data structure containing info
    // about STL execution
    result = M33_STL_Config(&exampleDataStruct);
    if (result == M33_STL_PASS){
        // Calls all Tests indicated by bitMask
        result = M33_STL_NVIC();

        if (result == M33_STL_PASS) {
            testPassed();
        }
    }
    errorHandler();

}

void enableFP (void)
{

    uint32_t * cpacr_addr = (uint32_t *) 0xE000ED88;

    *cpacr_addr = *cpacr_addr | 0x00F00000;

}
void testPassed(void)
```

```
{
  __asm__ volatile ("add r5, r5, 5");
  __asm__ volatile ("add r5, r5, 5");
  __asm__ volatile ("add r5, r5, 5");
  while(1); // Test Passed
}

void errorHandler(void)
{
  __asm__ volatile ("sub r5, r5, 6");
  __asm__ volatile ("sub r5, r5, 6");
  __asm__ volatile ("sub r5, r5, 6");
  while(1); // Error
}
```

3.6.2.3 M33_STL_MPU

The M33_STL.h file has to be included in the .c file where the STL is called from. This is required to access the STL's APIs, whose prototype is defined in that header file. Moreover, a variable of type m33_stl_dataStruct_t (i.e., m33_stl_dataStruct = exampleDataStruct) has to be allocated in a memory area selected by the user, accessible by the STL in reading and writing, and by the user application only for reading. Then, a variable to hold the STL result has to be initialized with a defensive value (i.e., uint32_t result = M33_STL_FAIL). Then, the third field of the m33_stl_mpuRegisterInfo array (see section A for details about the array fields) shall be initialized with the expected value. For more information on the m33_stl_mpuRegisterInfo array, see the section A.2.

To test the functionality of the MPU, the m33_stl_mem_manage_handler shall be defined in the vector table at the field corresponding to the MPU Fault Handler. Additionally, the user will need to redefine the m33_stl_mem_manage_user_handler to handle MPU exceptions not caused by the MPU test.

Then, the following parameters shall be initialized:

- mode: the value allows the user to select whether to launch the test of the correct functionality, the test of the registers, or both. In particular, this parameter has three valid values:
  - 0: test of the correct functionality
  - 1: test of the registers
  - 2: test of the correct functionality and test of the registers
- addressMPU: The address of the memory area configured by the STL to test the correct functionality of the MPU
- mpu_region: MPU Region ID to be used to test the correct functionality of the MPU module. The valid range is 0x0 to 0xF.

Before launching the M33_STL_MPU function, the M33_STL_Config API needs to be called to configure the base address of the data structure containing info about the STL execution.

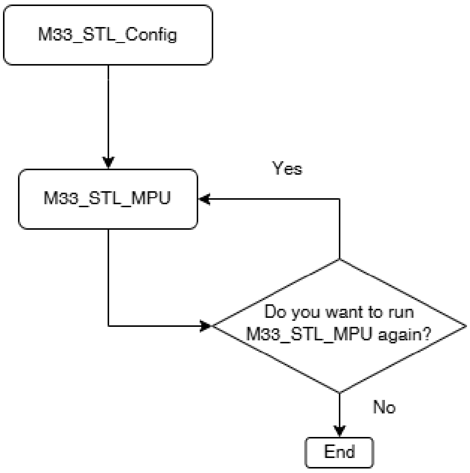


Figure 6 – M33_STL_MPU call example

After calling the M33_STL_Config, its result is checked. If it is equal to M33_STL_PASS, then the M33_STL_MPU function is called. Once the function's execution is completed, its result is compared to M33_STL_PASS and if they are equal, it means that the STL ended with success; otherwise, a fault has been detected.

Below is an example of how the M33 STL MPU APIs can be called:

```
#include <stdio.h>
#include "M33_STL.h"


//Functions to support Fault Injection simulations
void testPassed(void);
void errorHandler(void);

void enableFP (void);

m33_stl_dataStruct_t __attribute__((section(".d_m33_stl_data_struct"))) exampleDataStruct;

void main_pe0(void)
{

  uint32_t result;
  uint8_t forceFail;

  // Defensive initialization for M33_STL API input parameters
  result = M33_STL_FAIL;
```

```
    uint32_t mode;
    uint8_t mpu_region;
    uint32_t* addressMPU = (uint32_t*) 0x204E0000;

    mode = 2;
    mpu_region = 0;

    // Defensive initialization of data structure fields
    exampleDataStruct.Fault_Type = 0xFFFFFFFFU;
    exampleDataStruct.STL_Status = 0xFFFFFFFFU;
    exampleDataStruct.Test_ID = 0xFFFFFFFFU;

    // Configure the starting address of the data structure containing info
    // about STL execution
    result = M33_STL_Config(&exampleDataStruct);
    if (result == M33_STL_PASS){
      // Calls all Tests indicated by bitMask
      result = M33_STL_MPU(mode, addressMPU, mpu_region);
      if (result == M33_STL_PASS) {
        testPassed();
      }
    }
    errorHandler();
}

void enableFP (void)
{

    uint32_t * cpacr_addr = (uint32_t *) 0xE000ED88;

    *cpacr_addr = *cpacr_addr | 0x00F00000;

}
void testPassed(void)
{
    __asm__ volatile ("add r5, r5, 5");
    __asm__ volatile ("add r5, r5, 5");
    __asm__ volatile ("add r5, r5, 5");
    while(1); // Test Passed
}

void errorHandler(void)
```

```
{
  __asm__ volatile ("sub r5, r5, 6");
  __asm__ volatile ("sub r5, r5, 6");
  __asm__ volatile ("sub r5, r5, 6");
  while(1); // Error
}
```

3.6.3 Compiler Integration Guidelines

This section provides the user with more detailed instructions on how to compile the STL code using all supported compilers. The following sections and the mentioned Makefiles assume that the STL folder is contained in “<Project_folder>\rcar_u5lx\”. Additionally, when the make command is executed, it shall be ensured that the path of the used toolchain has been added to the environment variables. If this cannot be guaranteed, the config.mk file of the used toolchain shall be updated, specifying in the CC, AS, and LD variables the proper paths of the C compiler, Assembly compiler, and linker, respectively.

Only for the Tasking Compiler, the config.mk file of the used toolchain shall also be updated in the PP variable to set the proper paths of the C compiler (carm).

3.6.3.1 GHS Compiler

To compile and run the M33 STL code using the GreenHills (GHS) compiler, the file “Makefile” in “<Project_folder>\rcar_u5lx\ghs” shall be used. This makefile has been prepared to execute the proper sequence of commands required to compile and link the M33 STL. In particular, it performs the following steps:

1. Building C files, contained in STL/diagnostic/scheduler and STL/diagnostic/common folders
2. Building the assembly files, contained in STL/diagnostic/common, STL/diagnostic/tests/cpu, STL/diagnostic/tests/mpu, and STL/diagnostic/tests/nvic folders
3. Linking the outcomes of the previous steps (for each step, a set of .o files is created) and producing a .out file to be used for running the M33 STL

The “Makefile” shall be called by the user from the folder “<Project_folder>\rcar_u5lx\ghs” with the command make all DEVICE=<device_name>, where device_name can be one of the following:

- U5L1
- U5L2
- U5L4 (default if the DEVICE parameter is not specified)

This command performs the following operations:

1. It deletes the outcomes of the previous building process if present (it removes the “Debug” and “objects” folders and their content)
2. It executes the building and linking process:
    a. If everything ends with success, in the “Debug” folder, a file named “M*33_STL.out”* is created (it is the binary to be used for running the M33 STL). The addresses of the different sections are aligned to what has been specified in the “section.ld” file (contained in “<Project_folder>\rcar_u5lx\ghs\<DEVICE>” folder)
    b. If there is an error during the building/linking process, no .out file is generated in the “Debug” folder

The “Makefile” can also be used to delete the objects and Debug folders and their content, without starting the compilation, using the command “make clean”.

The compiler and linking flags are specified in the “config.mk” file (contained in “<Project_folder>\rcar_u5lx\ghs” folder). In particular, the following options are used to build and link the M33 STL Code:

GHS Compiler – C Code – ccarm command

- -cpu=cortexm33
- -fpu=vfpv5_d16
- -Ogeneral
- --short_enum
- -dual_debug
- -delete
- --no_commons
- -g
- -gsize
- -passsource
- -Wundef
- --prototype_errors
- --diag_error 193
- -Wshadow
- -MMD
- --gnu_asm
- --diag_suppress=2309

- -fsingle

GHS Compiler – Assembly Code – ccarm command

- -cpu=cortexm33
- -fpu=vfpv5_d16
- -preprocess_assembly_files
- -D__GHS__
- -D__ASSEMBLER__

For these 2 steps, the following directories are included in the command options, as follows:

- -I<Project_folder>/rcar_u5lx/ghs/../STL/diagnostic/common/inc
- -I<Project_folder>/rcar_u5lx/ghs/../STL/diagnostic/scheduler/inc
- -I<Project_folder>/rcar_u5lx/ghs/../STL/diagnostic/tests/cpu/inc
- -I<Project_folder>/rcar_u5lx/ghs/../STL/diagnostic/tests/mpu/inc
- -I<Project_folder>/rcar_u5lx/ghs/../STL/diagnostic/tests/nvic/inc

GHS Linker – ccarm command

- -object_dir=Debug
- -cpu=cortexm33
- -delete
- -e __start
- <DEVICE>/section.ld

3.6.3.2 GCC Compiler

To compile and run the M33 STL code using the GCC compiler, the file “Makefile” in “<Project_folder>\rcar_u5lx\gcc” shall be used. This makefile has been prepared to execute the proper sequence of commands required to compile and link the M33 STL. In particular, it performs the following steps:

1. Building C files, contained in STL/diagnostic/scheduler and STL/diagnostic/common folders
2. Building the assembly files, contained in STL/diagnostic/common, STL/diagnostic/tests/cpu, STL/diagnostic/tests/mpu, and STL/diagnostic/tests/nvic folders
3. Linking the outcomes of the previous steps (for each step, a set of .o files is created) and producing a .elf file to be used for running the M33 STL

The “Makefile” shall be called by the user from the folder “<Project_folder>\rcar_u5lx\gcc” with the command make all DEVICE=<device_name>, where device_name can be one of the following:

- U5L1
- U5L2
- U5L4 (default if the DEVICE parameter is not specified)

This command performs the following operations:

1. It deletes the outcomes of the previous building process if present (it removes the “Debug” and “objects” folders and their content)
2. It executes the building and linking process:
   a. If everything ends with success, in the “Debug” folder, a file named “M*33_STL.elf”* is created (it is the binary to be used for running the M33 STL). The addresses of the different sections are aligned to what has been specified in the “section.ld” file (contained in “<Project_folder>\rcar_u5lx\gcc\<DEVICE>” folder)
   b. If there is an error during the building/linking process, no .elf file is generated in the “Debug” folder

The “Makefile” can also be used to delete the objects and Debug folders and their content, without starting the compilation, using the command “make clean”.

The compiler and linking flags are specified in the “config.mk” file (contained in “<Project_folder>\rcar_u5lx\gcc” folder). In particular, the following options are used to build and link the M33 STL Code:

GCC Compiler – C Code – arm-none-eabi-gcc command

- -mcpu=cortex-m33
- -O2
- -fshort-enums
- -mfpu=fpv5-d16
- -mthumb
- -mfloat-abi=softfp
- -fno-common
- -g
- -MMD
- -gdwarf-4

- -Wno-attributes

GCC Compiler – Assembly Code – arm-none-eabi-gcc command

- -Xassembler
- -mcpu=cortex-m33
- -x assembler-with-cpp
- -mfpu=fpv5-d16
- -mthumb
- -mfloat-abi=softfp
- -DGCC_KEIL_HIGHTEC_CMP
- -D__ASSEMBLER__

For these 2 steps, the following directories are included in the command options, as follows:

- -I<Project_folder>/rcar_u5lx/gcc/../STL/diagnostic/common/inc
- -I<Project_folder>/rcar_u5lx/gcc/../STL/diagnostic/scheduler/inc
- -I<Project_folder>/rcar_u5lx/gcc/../STL/diagnostic/tests/cpu/inc
- -I<Project_folder>/rcar_u5lx/gcc/../STL/diagnostic/tests/mpu/inc
- -I<Project_folder>/rcar_u5lx/gcc/../STL/diagnostic/tests/nvic/inc

GCC Linker – arm-none-eabi-gcc command

- -mcpu=cortex-m33
- -mfpu=fpv5-d16
- -Xlinker --gc-sections
- -T $(PROJ_DIR)/<DEVICE>/section.ld
- -nostartfiles

Where $(PROJ_DIR) is the <Project_folder>/rcar_u5lx/gcc/ folder

Note: Since the GCC compiler is outside the scope of ISO 26262 compliance, it shall not be used in safety-related applications.

3.6.3.3 IAR Compiler

To compile and run the M33 STL code using the IAR Embedded Workbench for Arm, the file "Makefile" in "<Project_folder>\rcar_u5lx\IAR" shall be used. This makefile has been prepared to execute the proper sequence of commands required to compile and link the M33 STL. In particular, it performs the following steps:

1. Building C files, contained in STL/diagnostic/scheduler and STL/diagnostic/common folders
2. Building the assembly files, contained in STL/diagnostic/common, STL/diagnostic/tests/cpu, STL/diagnostic/tests/mpu, and STL/diagnostic/tests/nvic folders
3. Linking the outcomes of the previous steps (for each step, a set of .o files is created) and producing a .out file to be used for running the M33 STL

The "Makefile" shall be called by the user from the folder "<Project_folder>\rcar_u5lx\IAR" with the command make all DEVICE=<device_name>, where device_name can be one of the following:

- U5L1
- U5L2
- U5L4 (default if the DEVICE parameter is not specified)

This command performs the following operations:

1. It deletes the outcomes of the previous building process if present (it removes the "Debug" and "objects" folders and their content)
2. It executes the building and linking process:
   a. If everything ends with success, in the "Debug" folder, a file named "M*33_STL.out"* is created (it is the binary to be used for running the M33 STL). The addresses of the different sections are aligned to what has been specified in the "cm33.icf" file (contained in "<Project_folder>\rcar_u5lx\IAR\<DEVICE>" folder)
   b. If there is an error during the building/linking process, no .out file is generated in the "Debug" folder

The "Makefile" can also be used to delete the objects and Debug folders and their content, without starting the compilation, using the command "make clean".

The compiler and linking flags are specified in the "config.mk" file (contained in "<Project_folder>\rcar_u5lx\IAR" folder). In particular, the following options are used to build and link the M33 STL Code:

IAR Compiler – C Code – iccarm command

- --cpu=Cortex-M33
- -Oh
- --thumb

- --fpu=VFPv5_D16
- --discard_unused_publics
- --debug
- --diag_error=Pe193
- --enum_is_int
- --dependencies ms
- --endian=little
- -e
- --mfc
- -D__IAR__

IAR Compiler – Assembly Code – iasmarm command

- --cpu=cortex-M33
- --thumb
- --fpu=VFPv5_D16
- -r
- -D__IAR__
- -D__ASSEMBLER__

For these 2 steps, the following directories are included in the command options, as follows:

- -I<Project_folder>/rcar_u5lx/IAR/../STL/diagnostic/common/inc
- -I<Project_folder>/rcar_u5lx/IAR/../STL/diagnostic/scheduler/inc
- -I<Project_folder>/rcar_u5lx/IAR/../STL/diagnostic/tests/cpu/inc
- -I<Project_folder>/rcar_u5lx/IAR/../STL/diagnostic/tests/mpu/inc
- -I<Project_folder>/rcar_u5lx/IAR/../STL/diagnostic/tests/nvic/inc

IAR Linker – ilinkarm command

- --cpu=cortex-M33
- --fpu=VFPv5_D16
- --entry __iar_program_start
- --config $(PROJ_DIR)/<DEVICE>/cm33.icf
- --map=$(OUT_DIR)/M33_STL.map

Where $(PROJ_DIR) is the <Project_folder>/rcar_u5lx/IAR/ folder

Where $(OUT_DIR) is the <Project_folder>/rcar_u5lx/IAR/Debug folder

3.6.3.4 Arm Keil Compiler

To compile and run the M33 STL code using the Arm Toolchain for Embedded, the file “Makefile” in “<Project_folder>\rcar_u5lx\Keil” shall be used. This makefile has been prepared to execute the proper sequence of commands required to compile and link the M33 STL. In particular, it performs the following steps:

1. Building C files, contained in STL/diagnostic/scheduler and STL/diagnostic/common folders
2. Building the assembly files, contained in STL/diagnostic/common, STL/diagnostic/tests/cpu, STL/diagnostic/tests/mpu, and STL/diagnostic/tests/nvic folders
3. Linking the outcomes of the previous steps (for each step, a set of .o files is created) and producing a .elf file to be used for running the M33 STL

The “Makefile” shall be called by the user from the folder “<Project_folder>\rcar_u5lx\Keil” with the command make all DEVICE=<device_name>, where device_name can be one of the following:

- U5L1
- U5L2
- U5L4 (default if the DEVICE parameter is not specified)

This command performs the following operations:

1. It deletes the outcomes of the previous building process if present (it removes the “Debug” and “objects” folders and their content)
2. It executes the building and linking process:
    a. If everything ends with success, in the “Debug” folder, a file named “M*33_STL.elf”* is created (it is the binary to be used for running the M33 STL). The addresses of the different sections are aligned to what has been specified in the “ARMCM33_ac6.sct” file (contained in “<Project_folder>\rcar_u5lx\Keil\<DEVICE>” folder)
    b. If there is an error during the building/linking process, no .elf file is generated in the “Debug” folder

The “Makefile” can also be used to delete the objects and Debug folders and their content, without starting the compilation, using the command “make clean”.

The compiler and linking flags are specified in the “config.mk” file (contained in “<Project_folder>\rcar_u5lx\Keil” folder). In particular, the following options are used to build and link the M33 STL Code:

Before compiling the code, the startup.c file shall be updated. For the U5L4 and U5L2 devices, use the following startup.c file:

```
#define SCB_VTOR          (*((volatile int *)(0xE000ED08UL)))   /*< SCB configuration struct */
#define SCB_CPACR          (*((volatile int *)(0xE000ED88UL)))

#define __INITIAL_SP  Image$$ARM_LIB_STACK$$ZI$$Limit
typedef void(*VectorTableType)(void);
/*----------------------------------------------------------------------------
  External References
 *----------------------------------------------------------------------------*/
const VectorTableType IntVectors[496];
/*----------------------------------------------------------------------------
  Internal References
 *----------------------------------------------------------------------------*/
extern int __INITIAL_SP;
extern int __main(void);
extern void m33_stl_mem_manage_handler(void);

void Dummy_Handler(void);
void NMI_Dummy_Handler(void);
void NMI_Common_Handler(void);
void HardFault_Dummy_Handler(void);
void MemManage_Dummy_Handler(void);
void BusFault_Dummy_Handler(void);
void UsageFault_Dummy_Handler(void);
void SecureFault_Dummy_Handler(void);
void SVC_Dummy_Handler(void);
void DebugMon_Dummy_Handler(void);
void PendSV_Dummy_Handler(void);
void SysTick_Dummy_Handler(void);
void SystemInit (void);
void Init_gregs_8_12 (void);
void Init_regs_mpu_debug (void);
void Reset_Handler  (void);

/*----------------------------------------------------------------------------
  Exception / Interrupt Vector table
 *----------------------------------------------------------------------------*/
 #if defined ( __GNUC__ )
#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wpedantic"
#endif

 const VectorTableType IntVectors[496] __attribute__((used, section("RESET"))) = {
  (VectorTableType)(&__INITIAL_SP),      /*     Initial Stack Pointer */
  Reset_Handler,
  NMI_Common_Handler,                               /* -14 NMI Handler */
  HardFault_Dummy_Handler,                           /* -13 Hard Fault Handler */
  m33_stl_mem_manage_handler,                             /* -12 MPU Fault Handler */
  BusFault_Dummy_Handler,                            /* -11 Bus Fault Handler */
  UsageFault_Dummy_Handler,                           /* -10 Usage Fault Handler */
  SecureFault_Dummy_Handler,                          /*  -9 Secure Fault Handler */
  0,                            /*     Reserved */
  0,                            /*     Reserved */
  0,                            /*     Reserved */
  SVC_Dummy_Handler,                              /*  -5 SVCall Handler */
  DebugMon_Dummy_Handler,                              /*  -4 Debug Monitor Handler */
  0,                            /*     Reserved */
  PendSV_Dummy_Handler,                             /*  -2 PendSV Handler */
  SysTick_Dummy_Handler,                           /*  -1 SysTick Handler */

  /* Interrupts IRQn*/
#ifdef Interrupt0_Handler
  Interrupt0_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt1_Handler
 Interrupt1_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt2_Handler
 Interrupt2_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt3_Handler
 Interrupt3_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt4_Handler
 Interrupt4_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt5_Handler
 Interrupt5_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt6_Handler
 Interrupt6_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt7_Handler
 Interrupt7_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt8_Handler
  Interrupt8_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt9_Handler
  Interrupt9_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt10_Handler
  Interrupt10_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt11_Handler
  Interrupt11_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt12_Handler
  Interrupt12_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt13_Handler
  Interrupt13_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt14_Handler
  Interrupt14_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt15_Handler
  Interrupt15_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt16_Handler
  Interrupt16_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt17_Handler
  Interrupt17_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt18_Handler
  Interrupt18_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt19_Handler
  Interrupt19_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt20_Handler
  Interrupt20_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt21_Handler
  Interrupt21_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt22_Handler
  Interrupt22_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt23_Handler
  Interrupt23_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt24_Handler
  Interrupt24_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt25_Handler
  Interrupt25_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt26_Handler
  Interrupt26_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt27_Handler
  Interrupt27_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt28_Handler
```

```
  Interrupt28_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt29_Handler
  Interrupt29_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt30_Handler
  Interrupt30_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt31_Handler
  Interrupt31_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt32_Handler
  Interrupt32_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt33_Handler
  Interrupt33_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt34_Handler
  Interrupt34_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt35_Handler
 Interrupt35_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt36_Handler
 Interrupt36_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt37_Handler
 Interrupt37_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt38_Handler
 Interrupt38_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt39_Handler
 Interrupt39_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt40_Handler
 Interrupt40_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt41_Handler
 Interrupt41_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt42_Handler
  Interrupt42_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt43_Handler
  Interrupt43_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt44_Handler
  Interrupt44_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt45_Handler
  Interrupt45_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt46_Handler
  Interrupt46_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt47_Handler
  Interrupt47_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt48_Handler
  Interrupt48_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt49_Handler
  Interrupt49_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt50_Handler
  Interrupt50_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt51_Handler
  Interrupt51_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt52_Handler
  Interrupt52_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt53_Handler
  Interrupt53_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt54_Handler
  Interrupt54_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt55_Handler
  Interrupt55_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt56_Handler
  Interrupt56_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt57_Handler
  Interrupt57_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt58_Handler
  Interrupt58_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt59_Handler
  Interrupt59_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt60_Handler
  Interrupt60_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt61_Handler
  Interrupt61_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt62_Handler
```

```
  Interrupt62_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt63_Handler
  Interrupt63_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt64_Handler
  Interrupt64_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt65_Handler
  Interrupt65_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt66_Handler
  Interrupt66_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt67_Handler
  Interrupt67_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt68_Handler
  Interrupt68_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt69_Handler
  Interrupt69_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt70_Handler
  Interrupt70_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt71_Handler
  Interrupt71_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt72_Handler
  Interrupt72_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt73_Handler
  Interrupt73_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt74_Handler
  Interrupt74_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt75_Handler
  Interrupt75_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt76_Handler
  Interrupt76_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt77_Handler
  Interrupt77_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt78_Handler
  Interrupt78_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt79_Handler
  Interrupt79_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt80_Handler
  Interrupt80_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt81_Handler
  Interrupt81_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt82_Handler
  Interrupt82_Handler,
#else
```

```
 Dummy_Handler,
#endif
#ifdef Interrupt83_Handler
 Interrupt83_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt84_Handler
 Interrupt84_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt85_Handler
 Interrupt85_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt86_Handler
 Interrupt86_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt87_Handler
 Interrupt87_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt88_Handler
 Interrupt88_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt89_Handler
 Interrupt89_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt90_Handler
  Interrupt90_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt91_Handler
  Interrupt91_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt92_Handler
  Interrupt92_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt93_Handler
  Interrupt93_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt94_Handler
  Interrupt94_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt95_Handler
  Interrupt95_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt96_Handler
```

```
  Interrupt96_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt97_Handler
  Interrupt97_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt98_Handler
  Interrupt98_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt99_Handler
  Interrupt99_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt100_Handler
  Interrupt100_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt101_Handler
  Interrupt101_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt102_Handler
  Interrupt102_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt103_Handler
 Interrupt103_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt104_Handler
 Interrupt104_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt105_Handler
 Interrupt105_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt106_Handler
 Interrupt106_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt107_Handler
 Interrupt107_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt108_Handler
 Interrupt108_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt109_Handler
 Interrupt109_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt110_Handler
  Interrupt110_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt111_Handler
  Interrupt111_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt112_Handler
  Interrupt112_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt113_Handler
  Interrupt113_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt114_Handler
  Interrupt114_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt115_Handler
  Interrupt115_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt116_Handler
  Interrupt116_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt117_Handler
  Interrupt117_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt118_Handler
  Interrupt118_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt119_Handler
  Interrupt119_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt120_Handler
  Interrupt120_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt121_Handler
  Interrupt121_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt122_Handler
  Interrupt122_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt123_Handler
  Interrupt123_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt124_Handler
 Interrupt124_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt125_Handler
 Interrupt125_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt126_Handler
 Interrupt126_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt127_Handler
 Interrupt127_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt128_Handler
 Interrupt128_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt129_Handler
 Interrupt129_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt130_Handler
```

```
  Interrupt130_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt131_Handler
  Interrupt131_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt132_Handler
  Interrupt132_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt133_Handler
  Interrupt133_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt134_Handler
  Interrupt134_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt135_Handler
  Interrupt135_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt136_Handler
  Interrupt136_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt137_Handler
 Interrupt137_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt138_Handler
 Interrupt138_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt139_Handler
 Interrupt139_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt140_Handler
 Interrupt140_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt141_Handler
 Interrupt141_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt142_Handler
 Interrupt142_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt143_Handler
 Interrupt143_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt144_Handler
  Interrupt144_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt145_Handler
  Interrupt145_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt146_Handler
  Interrupt146_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt147_Handler
  Interrupt147_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt148_Handler
  Interrupt148_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt149_Handler
  Interrupt149_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt150_Handler
  Interrupt150_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt151_Handler
  Interrupt151_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt152_Handler
  Interrupt152_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt153_Handler
  Interrupt153_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt154_Handler
  Interrupt154_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt155_Handler
  Interrupt155_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt156_Handler
  Interrupt156_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt157_Handler
  Interrupt157_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt158_Handler
  Interrupt158_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt159_Handler
  Interrupt159_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt160_Handler
  Interrupt160_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt161_Handler
  Interrupt161_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt162_Handler
  Interrupt162_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt163_Handler
  Interrupt163_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt164_Handler
```

```
  Interrupt164_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt165_Handler
  Interrupt165_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt166_Handler
  Interrupt166_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt167_Handler
  Interrupt167_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt168_Handler
  Interrupt168_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt169_Handler
  Interrupt169_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt170_Handler
  Interrupt170_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt171_Handler
  Interrupt171_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt172_Handler
  Interrupt172_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt173_Handler
  Interrupt173_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt174_Handler
  Interrupt174_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt175_Handler
  Interrupt175_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt176_Handler
  Interrupt176_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt177_Handler
  Interrupt177_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt178_Handler
  Interrupt178_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt179_Handler
  Interrupt179_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt180_Handler
  Interrupt180_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt181_Handler
  Interrupt181_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt182_Handler
  Interrupt182_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt183_Handler
  Interrupt183_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt184_Handler
  Interrupt184_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt185_Handler
  Interrupt185_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt186_Handler
  Interrupt186_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt187_Handler
  Interrupt187_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt188_Handler
  Interrupt188_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt189_Handler
  Interrupt189_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt190_Handler
  Interrupt190_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt191_Handler
  Interrupt191_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt192_Handler
 Interrupt192_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt193_Handler
 Interrupt193_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt194_Handler
 Interrupt194_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt195_Handler
 Interrupt195_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt196_Handler
 Interrupt196_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt197_Handler
 Interrupt197_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt198_Handler
```

```
  Interrupt198_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt199_Handler
  Interrupt199_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt200_Handler
  Interrupt200_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt201_Handler
  Interrupt201_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt202_Handler
  Interrupt202_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt203_Handler
  Interrupt203_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt204_Handler
  Interrupt204_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt205_Handler
  Interrupt205_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt206_Handler
  Interrupt206_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt207_Handler
  Interrupt207_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt208_Handler
  Interrupt208_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt209_Handler
  Interrupt209_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt210_Handler
  Interrupt210_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt211_Handler
  Interrupt211_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt212_Handler
  Interrupt212_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt213_Handler
  Interrupt213_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt214_Handler
  Interrupt214_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt215_Handler
  Interrupt215_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt216_Handler
  Interrupt216_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt217_Handler
  Interrupt217_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt218_Handler
  Interrupt218_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt219_Handler
  Interrupt219_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt220_Handler
  Interrupt220_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt221_Handler
  Interrupt221_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt222_Handler
  Interrupt222_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt223_Handler
  Interrupt223_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt224_Handler
  Interrupt224_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt225_Handler
  Interrupt225_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt226_Handler
  Interrupt226_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt227_Handler
  Interrupt227_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt228_Handler
  Interrupt228_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt229_Handler
  Interrupt229_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt230_Handler
  Interrupt230_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt231_Handler
  Interrupt231_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt232_Handler
```

```
  Interrupt232_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt233_Handler
  Interrupt233_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt234_Handler
  Interrupt234_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt235_Handler
  Interrupt235_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt236_Handler
  Interrupt236_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt237_Handler
  Interrupt237_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt238_Handler
  Interrupt238_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt239_Handler
 Interrupt239_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt240_Handler
 Interrupt240_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt241_Handler
 Interrupt241_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt242_Handler
 Interrupt242_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt243_Handler
 Interrupt243_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt244_Handler
 Interrupt244_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt245_Handler
 Interrupt245_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt246_Handler
  Interrupt246_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt247_Handler
  Interrupt247_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt248_Handler
  Interrupt248_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt249_Handler
  Interrupt249_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt250_Handler
  Interrupt250_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt251_Handler
  Interrupt251_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt252_Handler
  Interrupt252_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt253_Handler
  Interrupt253_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt254_Handler
  Interrupt254_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt255_Handler
  Interrupt255_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt256_Handler
  Interrupt256_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt257_Handler
  Interrupt257_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt258_Handler
  Interrupt258_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt259_Handler
  Interrupt259_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt260_Handler
  Interrupt260_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt261_Handler
  Interrupt261_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt262_Handler
  Interrupt262_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt263_Handler
  Interrupt263_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt264_Handler
  Interrupt264_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt265_Handler
  Interrupt265_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt266_Handler
```

```
  Interrupt266_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt267_Handler
  Interrupt267_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt268_Handler
  Interrupt268_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt269_Handler
  Interrupt269_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt270_Handler
  Interrupt270_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt271_Handler
  Interrupt271_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt272_Handler
  Interrupt272_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt273_Handler
  Interrupt273_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt274_Handler
  Interrupt274_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt275_Handler
  Interrupt275_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt276_Handler
  Interrupt276_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt277_Handler
  Interrupt277_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt278_Handler
  Interrupt278_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt279_Handler
  Interrupt279_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt280_Handler
  Interrupt280_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt281_Handler
  Interrupt281_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt282_Handler
  Interrupt282_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt283_Handler
  Interrupt283_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt284_Handler
  Interrupt284_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt285_Handler
  Interrupt285_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt286_Handler
  Interrupt286_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt287_Handler
  Interrupt287_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt288_Handler
  Interrupt288_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt289_Handler
  Interrupt289_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt290_Handler
  Interrupt290_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt291_Handler
  Interrupt291_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt292_Handler
  Interrupt292_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt293_Handler
  Interrupt293_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt294_Handler
  Interrupt294_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt295_Handler
  Interrupt295_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt296_Handler
  Interrupt296_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt297_Handler
  Interrupt297_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt298_Handler
  Interrupt298_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt299_Handler
  Interrupt299_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt300_Handler
```

```
  Interrupt300_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt301_Handler
  Interrupt301_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt302_Handler
  Interrupt302_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt303_Handler
  Interrupt303_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt304_Handler
  Interrupt304_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt305_Handler
  Interrupt305_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt306_Handler
  Interrupt306_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt307_Handler
 Interrupt307_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt308_Handler
 Interrupt308_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt309_Handler
 Interrupt309_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt310_Handler
 Interrupt310_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt311_Handler
 Interrupt311_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt312_Handler
 Interrupt312_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt313_Handler
 Interrupt313_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt314_Handler
  Interrupt314_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt315_Handler
  Interrupt315_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt316_Handler
  Interrupt316_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt317_Handler
  Interrupt317_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt318_Handler
  Interrupt318_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt319_Handler
  Interrupt319_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt320_Handler
  Interrupt320_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt321_Handler
  Interrupt321_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt322_Handler
  Interrupt322_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt323_Handler
  Interrupt323_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt324_Handler
  Interrupt324_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt325_Handler
  Interrupt325_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt326_Handler
  Interrupt326_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt327_Handler
  Interrupt327_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt328_Handler
  Interrupt328_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt329_Handler
  Interrupt329_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt330_Handler
  Interrupt330_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt331_Handler
  Interrupt331_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt332_Handler
  Interrupt332_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt333_Handler
  Interrupt333_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt334_Handler
```

```
  Interrupt334_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt335_Handler
  Interrupt335_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt336_Handler
  Interrupt336_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt337_Handler
  Interrupt337_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt338_Handler
  Interrupt338_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt339_Handler
  Interrupt339_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt340_Handler
  Interrupt340_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt341_Handler
  Interrupt341_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt342_Handler
  Interrupt342_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt343_Handler
  Interrupt343_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt344_Handler
  Interrupt344_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt345_Handler
  Interrupt345_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt346_Handler
  Interrupt346_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt347_Handler
  Interrupt347_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt348_Handler
  Interrupt348_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt349_Handler
  Interrupt349_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt350_Handler
  Interrupt350_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt351_Handler
  Interrupt351_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt352_Handler
  Interrupt352_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt353_Handler
  Interrupt353_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt354_Handler
  Interrupt354_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt355_Handler
  Interrupt355_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt356_Handler
  Interrupt356_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt357_Handler
  Interrupt357_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt358_Handler
  Interrupt358_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt359_Handler
  Interrupt359_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt360_Handler
  Interrupt360_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt361_Handler
  Interrupt361_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt362_Handler
 Interrupt362_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt363_Handler
 Interrupt363_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt364_Handler
 Interrupt364_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt365_Handler
 Interrupt365_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt366_Handler
 Interrupt366_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt367_Handler
 Interrupt367_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt368_Handler
```

```
  Interrupt368_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt369_Handler
  Interrupt369_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt370_Handler
  Interrupt370_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt371_Handler
  Interrupt371_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt372_Handler
  Interrupt372_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt373_Handler
  Interrupt373_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt374_Handler
  Interrupt374_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt375_Handler
 Interrupt375_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt376_Handler
 Interrupt376_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt377_Handler
 Interrupt377_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt378_Handler
 Interrupt378_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt379_Handler
 Interrupt379_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt380_Handler
 Interrupt380_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt381_Handler
 Interrupt381_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt382_Handler
  Interrupt382_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt383_Handler
  Interrupt383_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt384_Handler
  Interrupt384_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt385_Handler
  Interrupt385_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt386_Handler
  Interrupt386_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt387_Handler
  Interrupt387_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt388_Handler
  Interrupt388_Handler,
#else
```

```
 Dummy_Handler,
#endif
#ifdef Interrupt389_Handler
 Interrupt389_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt390_Handler
 Interrupt390_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt391_Handler
 Interrupt391_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt392_Handler
 Interrupt392_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt393_Handler
 Interrupt393_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt394_Handler
 Interrupt394_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt395_Handler
 Interrupt395_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt396_Handler
  Interrupt396_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt397_Handler
  Interrupt397_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt398_Handler
  Interrupt398_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt399_Handler
  Interrupt399_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt400_Handler
  Interrupt400_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt401_Handler
  Interrupt401_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt402_Handler
```

```
  Interrupt402_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt403_Handler
  Interrupt403_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt404_Handler
  Interrupt404_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt405_Handler
  Interrupt405_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt406_Handler
  Interrupt406_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt407_Handler
  Interrupt407_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt408_Handler
  Interrupt408_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt409_Handler
 Interrupt409_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt410_Handler
 Interrupt410_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt411_Handler
 Interrupt411_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt412_Handler
 Interrupt412_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt413_Handler
 Interrupt413_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt414_Handler
 Interrupt414_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt415_Handler
 Interrupt415_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt416_Handler
  Interrupt416_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt417_Handler
  Interrupt417_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt418_Handler
  Interrupt418_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt419_Handler
  Interrupt419_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt420_Handler
  Interrupt420_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt421_Handler
  Interrupt421_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt422_Handler
  Interrupt422_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt423_Handler
  Interrupt423_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt424_Handler
  Interrupt424_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt425_Handler
  Interrupt425_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt426_Handler
  Interrupt426_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt427_Handler
  Interrupt427_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt428_Handler
  Interrupt428_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt429_Handler
  Interrupt429_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt430_Handler
  Interrupt430_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt431_Handler
  Interrupt431_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt432_Handler
  Interrupt432_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt433_Handler
  Interrupt433_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt434_Handler
  Interrupt434_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt435_Handler
  Interrupt435_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt436_Handler
```

```
  Interrupt436_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt437_Handler
  Interrupt437_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt438_Handler
  Interrupt438_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt439_Handler
  Interrupt439_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt440_Handler
  Interrupt440_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt441_Handler
  Interrupt441_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt442_Handler
  Interrupt442_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt443_Handler
 Interrupt443_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt444_Handler
 Interrupt444_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt445_Handler
 Interrupt445_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt446_Handler
 Interrupt446_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt447_Handler
 Interrupt447_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt448_Handler
 Interrupt448_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt449_Handler
 Interrupt449_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt450_Handler
  Interrupt450_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt451_Handler
  Interrupt451_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt452_Handler
  Interrupt452_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt453_Handler
  Interrupt453_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt454_Handler
  Interrupt454_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt455_Handler
  Interrupt455_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt456_Handler
  Interrupt456_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt457_Handler
  Interrupt457_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt458_Handler
  Interrupt458_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt459_Handler
  Interrupt459_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt460_Handler
  Interrupt460_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt461_Handler
  Interrupt461_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt462_Handler
  Interrupt462_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt463_Handler
  Interrupt463_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt464_Handler
  Interrupt464_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt465_Handler
  Interrupt465_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt466_Handler
  Interrupt466_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt467_Handler
  Interrupt467_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt468_Handler
  Interrupt468_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt469_Handler
  Interrupt469_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt470_Handler
```

```
  Interrupt470_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt471_Handler
  Interrupt471_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt472_Handler
  Interrupt472_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt473_Handler
  Interrupt473_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt474_Handler
  Interrupt474_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt475_Handler
  Interrupt475_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt476_Handler
  Interrupt476_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt477_Handler
  Interrupt477_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt478_Handler
  Interrupt478_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt479_Handler
  Interrupt479_Handler
#else
  Dummy_Handler
#endif
};

#if defined ( __GNUC__ )
#pragma GCC diagnostic pop
#endif
/*----------------------------------------------------------------------------
  Reset Handler called on controller reset
 *----------------------------------------------------------------------------*/
void Reset_Handler(void)
{
  SystemInit();            /* CMSIS System Initialization */
  __main();
}

void SystemInit (void)
{
  SCB_VTOR = (int) &(IntVectors[0]);
  SCB_CPACR |= (0xF << 20);  // Enable CP10, CP11
```

```
}

/*----------------------------------------------------------------------------
  Default Handler for Exceptions / Interrupts
 *----------------------------------------------------------------------------*/
void Dummy_Handler(void)
{
  while(1);
}
/*----------------------------------------------------------------------------
  Default Handler for Exceptions / Interrupts
 *----------------------------------------------------------------------------*/
void Interrupt_Dummy_Handler(void)
{
  while(1);
}

void NMI_Dummy_Handler(void)
{
  while(1);
}

void NMI_Common_Handler(void)
{
  while(1);
}

void HardFault_Dummy_Handler(void)
{
  while(1);
}

void MemManage_Dummy_Handler(void)
{
  while(1);
}

void BusFault_Dummy_Handler(void)
{
  while(1);
}

void UsageFault_Dummy_Handler(void)
{
  while(1);
}

void SecureFault_Dummy_Handler(void)
{
  while(1);
}

void SVC_Dummy_Handler(void)
{
  while(1);
}

void DebugMon_Dummy_Handler(void)
{
  while(1);
}

void PendSV_Dummy_Handler(void)
{
  while(1);
}

void SysTick_Dummy_Handler(void)
```

```
{
  while(1);
}
```

For the U5L1 device uses the following startup.c file:

```
#define SCB_VTOR          (*((volatile int *)(0xE000ED08UL)))   /*< SCB configuration struct */
#define SCB_CPACR          (*((volatile int *)(0xE000ED88UL)))

#define __INITIAL_SP  Image$$ARM_LIB_STACK$$ZI$$Limit
typedef void(*VectorTableType)(void);
/*----------------------------------------------------------------------------
  External References
 *----------------------------------------------------------------------------*/
const VectorTableType IntVectors[496];
/*----------------------------------------------------------------------------
  Internal References
 *----------------------------------------------------------------------------*/
extern int __INITIAL_SP;
extern int main(void);
extern void m33_stl_mem_manage_handler(void);

void Dummy_Handler(void);
void NMI_Dummy_Handler(void);
void NMI_Common_Handler(void);
void HardFault_Dummy_Handler(void);
void MemManage_Dummy_Handler(void);
void BusFault_Dummy_Handler(void);
void UsageFault_Dummy_Handler(void);
void SecureFault_Dummy_Handler(void);
void SVC_Dummy_Handler(void);
void DebugMon_Dummy_Handler(void);
void PendSV_Dummy_Handler(void);
void SysTick_Dummy_Handler(void);
void SystemInit (void);
void Init_gregs_8_12 (void);
void Init_regs_mpu_debug (void);
void Reset_Handler  (void);

/*----------------------------------------------------------------------------
  Exception / Interrupt Vector table
 *----------------------------------------------------------------------------*/
 #if defined ( __GNUC__ )
#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wpedantic"
#endif

 const VectorTableType IntVectors[496] __attribute__((used, section("RESET"))) = {
  (VectorTableType)(&__INITIAL_SP),      /*    Initial Stack Pointer */
  Reset_Handler,
  NMI_Common_Handler,                              /* -14 NMI Handler */
  HardFault_Dummy_Handler,                          /* -13 Hard Fault Handler */
  m33_stl_mem_manage_handler,                         /* -12 MPU Fault Handler */
  BusFault_Dummy_Handler,                          /* -11 Bus Fault Handler */
  UsageFault_Dummy_Handler,                         /* -10 Usage Fault Handler */
  SecureFault_Dummy_Handler,                        /*  -9 Secure Fault Handler */
  0,                             /*    Reserved */
  0,                             /*    Reserved */
  0,                             /*    Reserved */
  SVC_Dummy_Handler,                            /*  -5 SVCall Handler */
  DebugMon_Dummy_Handler,                           /*  -4 Debug Monitor Handler */
  0,                             /*    Reserved */
  PendSV_Dummy_Handler,                           /*  -2 PendSV Handler */
  SysTick_Dummy_Handler,                         /*  -1 SysTick Handler */

  /* Interrupts IRQn*/
#ifdef Interrupt0_Handler
  Interrupt0_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt1_Handler
  Interrupt1_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt2_Handler
  Interrupt2_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt3_Handler
  Interrupt3_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt4_Handler
  Interrupt4_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt5_Handler
  Interrupt5_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt6_Handler
  Interrupt6_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt7_Handler
  Interrupt7_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt8_Handler
  Interrupt8_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt9_Handler
  Interrupt9_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt10_Handler
  Interrupt10_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt11_Handler
  Interrupt11_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt12_Handler
  Interrupt12_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt13_Handler
  Interrupt13_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt14_Handler
```

```
  Interrupt14_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt15_Handler
  Interrupt15_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt16_Handler
  Interrupt16_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt17_Handler
  Interrupt17_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt18_Handler
  Interrupt18_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt19_Handler
  Interrupt19_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt20_Handler
  Interrupt20_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt21_Handler
  Interrupt21_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt22_Handler
  Interrupt22_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt23_Handler
  Interrupt23_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt24_Handler
  Interrupt24_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt25_Handler
  Interrupt25_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt26_Handler
  Interrupt26_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt27_Handler
  Interrupt27_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt28_Handler
  Interrupt28_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt29_Handler
  Interrupt29_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt30_Handler
  Interrupt30_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt31_Handler
  Interrupt31_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt32_Handler
  Interrupt32_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt33_Handler
  Interrupt33_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt34_Handler
  Interrupt34_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt35_Handler
  Interrupt35_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt36_Handler
  Interrupt36_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt37_Handler
  Interrupt37_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt38_Handler
  Interrupt38_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt39_Handler
  Interrupt39_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt40_Handler
  Interrupt40_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt41_Handler
  Interrupt41_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt42_Handler
  Interrupt42_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt43_Handler
  Interrupt43_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt44_Handler
  Interrupt44_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt45_Handler
  Interrupt45_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt46_Handler
  Interrupt46_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt47_Handler
  Interrupt47_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt48_Handler
```

```
  Interrupt48_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt49_Handler
  Interrupt49_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt50_Handler
  Interrupt50_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt51_Handler
  Interrupt51_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt52_Handler
  Interrupt52_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt53_Handler
  Interrupt53_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt54_Handler
  Interrupt54_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt55_Handler
  Interrupt55_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt56_Handler
  Interrupt56_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt57_Handler
  Interrupt57_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt58_Handler
  Interrupt58_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt59_Handler
  Interrupt59_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt60_Handler
  Interrupt60_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt61_Handler
  Interrupt61_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt62_Handler
  Interrupt62_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt63_Handler
  Interrupt63_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt64_Handler
  Interrupt64_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt65_Handler
  Interrupt65_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt66_Handler
  Interrupt66_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt67_Handler
  Interrupt67_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt68_Handler
  Interrupt68_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt69_Handler
  Interrupt69_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt70_Handler
  Interrupt70_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt71_Handler
  Interrupt71_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt72_Handler
  Interrupt72_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt73_Handler
  Interrupt73_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt74_Handler
  Interrupt74_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt75_Handler
  Interrupt75_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt76_Handler
  Interrupt76_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt77_Handler
  Interrupt77_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt78_Handler
  Interrupt78_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt79_Handler
  Interrupt79_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt80_Handler
  Interrupt80_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt81_Handler
  Interrupt81_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt82_Handler
```

```
  Interrupt82_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt83_Handler
  Interrupt83_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt84_Handler
  Interrupt84_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt85_Handler
  Interrupt85_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt86_Handler
  Interrupt86_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt87_Handler
  Interrupt87_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt88_Handler
  Interrupt88_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt89_Handler
 Interrupt89_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt90_Handler
 Interrupt90_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt91_Handler
 Interrupt91_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt92_Handler
 Interrupt92_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt93_Handler
 Interrupt93_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt94_Handler
 Interrupt94_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt95_Handler
 Interrupt95_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt96_Handler
  Interrupt96_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt97_Handler
  Interrupt97_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt98_Handler
  Interrupt98_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt99_Handler
  Interrupt99_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt100_Handler
  Interrupt100_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt101_Handler
  Interrupt101_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt102_Handler
  Interrupt102_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt103_Handler
  Interrupt103_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt104_Handler
  Interrupt104_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt105_Handler
  Interrupt105_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt106_Handler
  Interrupt106_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt107_Handler
  Interrupt107_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt108_Handler
  Interrupt108_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt109_Handler
  Interrupt109_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt110_Handler
  Interrupt110_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt111_Handler
  Interrupt111_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt112_Handler
  Interrupt112_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt113_Handler
  Interrupt113_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt114_Handler
  Interrupt114_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt115_Handler
  Interrupt115_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt116_Handler
```

```
  Interrupt116_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt117_Handler
  Interrupt117_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt118_Handler
  Interrupt118_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt119_Handler
  Interrupt119_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt120_Handler
  Interrupt120_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt121_Handler
  Interrupt121_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt122_Handler
  Interrupt122_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt123_Handler
 Interrupt123_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt124_Handler
 Interrupt124_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt125_Handler
 Interrupt125_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt126_Handler
 Interrupt126_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt127_Handler
 Interrupt127_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt128_Handler
 Interrupt128_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt129_Handler
 Interrupt129_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt130_Handler
  Interrupt130_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt131_Handler
  Interrupt131_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt132_Handler
  Interrupt132_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt133_Handler
  Interrupt133_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt134_Handler
  Interrupt134_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt135_Handler
  Interrupt135_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt136_Handler
  Interrupt136_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt137_Handler
  Interrupt137_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt138_Handler
  Interrupt138_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt139_Handler
  Interrupt139_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt140_Handler
  Interrupt140_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt141_Handler
  Interrupt141_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt142_Handler
  Interrupt142_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt143_Handler
  Interrupt143_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt144_Handler
 Interrupt144_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt145_Handler
 Interrupt145_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt146_Handler
 Interrupt146_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt147_Handler
 Interrupt147_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt148_Handler
 Interrupt148_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt149_Handler
 Interrupt149_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt150_Handler
```

```
  Interrupt150_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt151_Handler
  Interrupt151_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt152_Handler
  Interrupt152_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt153_Handler
  Interrupt153_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt154_Handler
  Interrupt154_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt155_Handler
  Interrupt155_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt156_Handler
  Interrupt156_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt157_Handler
  Interrupt157_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt158_Handler
  Interrupt158_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt159_Handler
  Interrupt159_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt160_Handler
  Interrupt160_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt161_Handler
  Interrupt161_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt162_Handler
  Interrupt162_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt163_Handler
  Interrupt163_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt164_Handler
  Interrupt164_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt165_Handler
  Interrupt165_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt166_Handler
  Interrupt166_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt167_Handler
  Interrupt167_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt168_Handler
  Interrupt168_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt169_Handler
  Interrupt169_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt170_Handler
  Interrupt170_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt171_Handler
  Interrupt171_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt172_Handler
  Interrupt172_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt173_Handler
  Interrupt173_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt174_Handler
  Interrupt174_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt175_Handler
  Interrupt175_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt176_Handler
  Interrupt176_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt177_Handler
  Interrupt177_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt178_Handler
 Interrupt178_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt179_Handler
 Interrupt179_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt180_Handler
 Interrupt180_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt181_Handler
 Interrupt181_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt182_Handler
 Interrupt182_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt183_Handler
 Interrupt183_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt184_Handler
```

```
 Interrupt184_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt185_Handler
 Interrupt185_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt186_Handler
 Interrupt186_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt187_Handler
 Interrupt187_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt188_Handler
 Interrupt188_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt189_Handler
 Interrupt189_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt190_Handler
 Interrupt190_Handler,
#else
 Dummy_Handler,
#endif
```

```
#ifdef Interrupt191_Handler
  Interrupt191_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt192_Handler
  Interrupt192_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt193_Handler
  Interrupt193_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt194_Handler
  Interrupt194_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt195_Handler
  Interrupt195_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt196_Handler
  Interrupt196_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt197_Handler
  Interrupt197_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt198_Handler
 Interrupt198_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt199_Handler
 Interrupt199_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt200_Handler
 Interrupt200_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt201_Handler
 Interrupt201_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt202_Handler
 Interrupt202_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt203_Handler
 Interrupt203_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt204_Handler
 Interrupt204_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt205_Handler
  Interrupt205_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt206_Handler
  Interrupt206_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt207_Handler
  Interrupt207_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt208_Handler
  Interrupt208_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt209_Handler
  Interrupt209_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt210_Handler
  Interrupt210_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt211_Handler
  Interrupt211_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt212_Handler
  Interrupt212_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt213_Handler
  Interrupt213_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt214_Handler
  Interrupt214_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt215_Handler
  Interrupt215_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt216_Handler
  Interrupt216_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt217_Handler
  Interrupt217_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt218_Handler
```

```
  Interrupt218_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt219_Handler
  Interrupt219_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt220_Handler
  Interrupt220_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt221_Handler
  Interrupt221_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt222_Handler
  Interrupt222_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt223_Handler
  Interrupt223_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt224_Handler
  Interrupt224_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt225_Handler
  Interrupt225_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt226_Handler
  Interrupt226_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt227_Handler
  Interrupt227_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt228_Handler
  Interrupt228_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt229_Handler
  Interrupt229_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt230_Handler
  Interrupt230_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt231_Handler
  Interrupt231_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt232_Handler
 Interrupt232_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt233_Handler
 Interrupt233_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt234_Handler
 Interrupt234_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt235_Handler
 Interrupt235_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt236_Handler
 Interrupt236_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt237_Handler
 Interrupt237_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt238_Handler
 Interrupt238_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt239_Handler
  Interrupt239_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt240_Handler
  Interrupt240_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt241_Handler
  Interrupt241_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt242_Handler
  Interrupt242_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt243_Handler
  Interrupt243_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt244_Handler
  Interrupt244_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt245_Handler
  Interrupt245_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt246_Handler
  Interrupt246_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt247_Handler
  Interrupt247_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt248_Handler
  Interrupt248_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt249_Handler
  Interrupt249_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt250_Handler
  Interrupt250_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt251_Handler
  Interrupt251_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt252_Handler
```

```
  Interrupt252_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt253_Handler
  Interrupt253_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt254_Handler
  Interrupt254_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt255_Handler
  Interrupt255_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt256_Handler
  Interrupt256_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt257_Handler
  Interrupt257_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt258_Handler
  Interrupt258_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt259_Handler
  Interrupt259_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt260_Handler
  Interrupt260_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt261_Handler
  Interrupt261_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt262_Handler
  Interrupt262_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt263_Handler
  Interrupt263_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt264_Handler
  Interrupt264_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt265_Handler
  Interrupt265_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt266_Handler
  Interrupt266_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt267_Handler
  Interrupt267_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt268_Handler
  Interrupt268_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt269_Handler
  Interrupt269_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt270_Handler
  Interrupt270_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt271_Handler
  Interrupt271_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt272_Handler
  Interrupt272_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt273_Handler
  Interrupt273_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt274_Handler
  Interrupt274_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt275_Handler
  Interrupt275_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt276_Handler
  Interrupt276_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt277_Handler
  Interrupt277_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt278_Handler
  Interrupt278_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt279_Handler
  Interrupt279_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt280_Handler
  Interrupt280_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt281_Handler
  Interrupt281_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt282_Handler
  Interrupt282_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt283_Handler
  Interrupt283_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt284_Handler
  Interrupt284_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt285_Handler
  Interrupt285_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt286_Handler
```

```
  Interrupt286_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt287_Handler
  Interrupt287_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt288_Handler
  Interrupt288_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt289_Handler
  Interrupt289_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt290_Handler
  Interrupt290_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt291_Handler
  Interrupt291_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt292_Handler
  Interrupt292_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt293_Handler
  Interrupt293_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt294_Handler
  Interrupt294_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt295_Handler
  Interrupt295_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt296_Handler
  Interrupt296_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt297_Handler
  Interrupt297_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt298_Handler
  Interrupt298_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt299_Handler
  Interrupt299_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt300_Handler
  Interrupt300_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt301_Handler
  Interrupt301_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt302_Handler
  Interrupt302_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt303_Handler
  Interrupt303_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt304_Handler
  Interrupt304_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt305_Handler
  Interrupt305_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt306_Handler
  Interrupt306_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt307_Handler
  Interrupt307_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt308_Handler
  Interrupt308_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt309_Handler
  Interrupt309_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt310_Handler
  Interrupt310_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt311_Handler
  Interrupt311_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt312_Handler
  Interrupt312_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt313_Handler
  Interrupt313_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt314_Handler
  Interrupt314_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt315_Handler
  Interrupt315_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt316_Handler
  Interrupt316_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt317_Handler
  Interrupt317_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt318_Handler
  Interrupt318_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt319_Handler
  Interrupt319_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt320_Handler
```

```
  Interrupt320_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt321_Handler
  Interrupt321_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt322_Handler
  Interrupt322_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt323_Handler
  Interrupt323_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt324_Handler
  Interrupt324_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt325_Handler
  Interrupt325_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt326_Handler
  Interrupt326_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt327_Handler
 Interrupt327_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt328_Handler
 Interrupt328_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt329_Handler
 Interrupt329_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt330_Handler
 Interrupt330_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt331_Handler
 Interrupt331_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt332_Handler
 Interrupt332_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt333_Handler
 Interrupt333_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt334_Handler
  Interrupt334_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt335_Handler
  Interrupt335_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt336_Handler
  Interrupt336_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt337_Handler
  Interrupt337_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt338_Handler
  Interrupt338_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt339_Handler
  Interrupt339_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt340_Handler
  Interrupt340_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt341_Handler
  Interrupt341_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt342_Handler
  Interrupt342_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt343_Handler
  Interrupt343_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt344_Handler
  Interrupt344_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt345_Handler
  Interrupt345_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt346_Handler
  Interrupt346_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt347_Handler
  Interrupt347_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt348_Handler
  Interrupt348_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt349_Handler
  Interrupt349_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt350_Handler
  Interrupt350_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt351_Handler
  Interrupt351_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt352_Handler
  Interrupt352_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt353_Handler
  Interrupt353_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt354_Handler
```

```
  Interrupt354_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt355_Handler
  Interrupt355_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt356_Handler
  Interrupt356_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt357_Handler
  Interrupt357_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt358_Handler
  Interrupt358_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt359_Handler
  Interrupt359_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt360_Handler
  Interrupt360_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt361_Handler
  Interrupt361_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt362_Handler
  Interrupt362_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt363_Handler
  Interrupt363_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt364_Handler
  Interrupt364_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt365_Handler
  Interrupt365_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt366_Handler
  Interrupt366_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt367_Handler
  Interrupt367_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt368_Handler
  Interrupt368_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt369_Handler
  Interrupt369_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt370_Handler
  Interrupt370_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt371_Handler
  Interrupt371_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt372_Handler
  Interrupt372_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt373_Handler
  Interrupt373_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt374_Handler
  Interrupt374_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt375_Handler
  Interrupt375_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt376_Handler
  Interrupt376_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt377_Handler
  Interrupt377_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt378_Handler
  Interrupt378_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt379_Handler
  Interrupt379_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt380_Handler
  Interrupt380_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt381_Handler
  Interrupt381_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt382_Handler
  Interrupt382_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt383_Handler
  Interrupt383_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt384_Handler
  Interrupt384_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt385_Handler
  Interrupt385_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt386_Handler
  Interrupt386_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt387_Handler
  Interrupt387_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt388_Handler
```

```
  Interrupt388_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt389_Handler
  Interrupt389_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt390_Handler
  Interrupt390_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt391_Handler
  Interrupt391_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt392_Handler
  Interrupt392_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt393_Handler
  Interrupt393_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt394_Handler
  Interrupt394_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt395_Handler
  Interrupt395_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt396_Handler
  Interrupt396_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt397_Handler
  Interrupt397_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt398_Handler
  Interrupt398_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt399_Handler
  Interrupt399_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt400_Handler
  Interrupt400_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt401_Handler
  Interrupt401_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt402_Handler
  Interrupt402_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt403_Handler
  Interrupt403_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt404_Handler
  Interrupt404_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt405_Handler
  Interrupt405_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt406_Handler
  Interrupt406_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt407_Handler
  Interrupt407_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt408_Handler
  Interrupt408_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt409_Handler
  Interrupt409_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt410_Handler
  Interrupt410_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt411_Handler
  Interrupt411_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt412_Handler
  Interrupt412_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt413_Handler
  Interrupt413_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt414_Handler
  Interrupt414_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt415_Handler
  Interrupt415_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt416_Handler
  Interrupt416_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt417_Handler
  Interrupt417_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt418_Handler
  Interrupt418_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt419_Handler
  Interrupt419_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt420_Handler
  Interrupt420_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt421_Handler
  Interrupt421_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt422_Handler
```

```
  Interrupt422_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt423_Handler
  Interrupt423_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt424_Handler
  Interrupt424_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt425_Handler
  Interrupt425_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt426_Handler
  Interrupt426_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt427_Handler
  Interrupt427_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt428_Handler
  Interrupt428_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt429_Handler
 Interrupt429_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt430_Handler
 Interrupt430_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt431_Handler
 Interrupt431_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt432_Handler
 Interrupt432_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt433_Handler
 Interrupt433_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt434_Handler
 Interrupt434_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt435_Handler
 Interrupt435_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt436_Handler
  Interrupt436_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt437_Handler
  Interrupt437_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt438_Handler
  Interrupt438_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt439_Handler
  Interrupt439_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt440_Handler
  Interrupt440_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt441_Handler
  Interrupt441_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt442_Handler
  Interrupt442_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt443_Handler
  Interrupt443_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt444_Handler
  Interrupt444_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt445_Handler
  Interrupt445_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt446_Handler
  Interrupt446_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt447_Handler
  Interrupt447_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt448_Handler
  Interrupt448_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt449_Handler
  Interrupt449_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt450_Handler
 Interrupt450_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt451_Handler
 Interrupt451_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt452_Handler
 Interrupt452_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt453_Handler
 Interrupt453_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt454_Handler
 Interrupt454_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt455_Handler
 Interrupt455_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt456_Handler
```

```
  Interrupt456_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt457_Handler
  Interrupt457_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt458_Handler
  Interrupt458_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt459_Handler
  Interrupt459_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt460_Handler
  Interrupt460_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt461_Handler
  Interrupt461_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt462_Handler
  Interrupt462_Handler,
#else
  Dummy_Handler,
#endif
```

```
#ifdef Interrupt463_Handler
 Interrupt463_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt464_Handler
 Interrupt464_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt465_Handler
 Interrupt465_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt466_Handler
 Interrupt466_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt467_Handler
 Interrupt467_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt468_Handler
 Interrupt468_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt469_Handler
 Interrupt469_Handler,
#else
 Dummy_Handler,
#endif
#ifdef Interrupt470_Handler
  Interrupt470_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt471_Handler
  Interrupt471_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt472_Handler
  Interrupt472_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt473_Handler
  Interrupt473_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt474_Handler
  Interrupt474_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt475_Handler
  Interrupt475_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt476_Handler
  Interrupt476_Handler,
#else
```

```
  Dummy_Handler,
#endif
#ifdef Interrupt477_Handler
  Interrupt477_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt478_Handler
  Interrupt478_Handler,
#else
  Dummy_Handler,
#endif
#ifdef Interrupt479_Handler
  Interrupt479_Handler
#else
  Dummy_Handler
#endif
};

#if defined ( __GNUC__ )
#pragma GCC diagnostic pop
#endif
/*----------------------------------------------------------------------------
  Reset Handler called on controller reset
 *----------------------------------------------------------------------------*/
void Reset_Handler(void)
{
  SystemInit();           /* CMSIS System Initialization */
  main();
}

void SystemInit (void)
{
  SCB_VTOR = (int) &(IntVectors[0]);
  SCB_CPACR |= (0xF << 20);  // Enable CP10, CP11

}

/*----------------------------------------------------------------------------
  Default Handler for Exceptions / Interrupts
 *----------------------------------------------------------------------------*/
void Dummy_Handler(void)
{
  while(1);
}
/*----------------------------------------------------------------------------
  Default Handler for Exceptions / Interrupts
 *----------------------------------------------------------------------------*/
void Interrupt_Dummy_Handler(void)
{
  while(1);
}

void NMI_Dummy_Handler(void)
{
  while(1);
}

void NMI_Common_Handler(void)
{
  while(1);
}

void HardFault_Dummy_Handler(void)
{
  while(1);
}
```

```
void MemManage_Dummy_Handler(void)
{
  while(1);
}

void BusFault_Dummy_Handler(void)
{
  while(1);
}

void UsageFault_Dummy_Handler(void)
{
  while(1);
}

void SecureFault_Dummy_Handler(void)
{
  while(1);
}

void SVC_Dummy_Handler(void)
{
  while(1);
}

void DebugMon_Dummy_Handler(void)
{
  while(1);
}

void PendSV_Dummy_Handler(void)
{
  while(1);
}
```

```
void SysTick_Dummy_Handler(void)
{
  while(1);
}
```

Arm Keil Compiler – C Code – armclang command

- -mcpu=cortex-m33
- --target=arm-arm-none-eabi
- -O2
- -fshort-enums
- -mfpu=fpv5-d16
- -mthumb
- -mfloat-abi=softfp
- -fno-common
- -g
- -MMD

Arm Keil Compiler – Assembly Code – armclang command

- -mcpu=cortex-m33
- -mfpu=fpv5-d16
- -mthumb
- -mfloat-abi=softfp
- -x assembler-with-cpp
- --target=arm-arm-none-eabi
- -DGCC_KEIL_HIGHTEC_CMP
- -D__ASSEMBLER__

For these 2 steps, the following directories are included in the command options, as follows:

- -I<Project_folder>/rcar_u5lx/Keil/../STL/diagnostic/common/inc
- -I<Project_folder>/rcar_u5lx/Keil/../STL/diagnostic/scheduler/inc
- -I<Project_folder>/rcar_u5lx/Keil/../STL/diagnostic/tests/cpu/inc
- -I<Project_folder>/rcar_u5lx/Keil/../STL/diagnostic/tests/mpu/inc
- -I<Project_folder>/rcar_u5lx/Keil/../STL/diagnostic/tests/nvic/inc

Arm Keil Linker – armlink command

- --cpu=Cortex-M33
- --fpu=FPv5_D16
- --remove
- --scatter=$(PROJ_DIR)/<DEVICE>/ARMCM33_ac6.sct
- --entry=Reset_Handler

Where $(PROJ_DIR) is the <Project_folder>/rcar_u5lx/Keil/ folder

3.6.3.5 Tasking compiler

To compile and run the M33 STL code using the Tasking compiler, the file "Makefile" in "<Project_folder>\rcar_u5lx\Tasking" shall be used. This makefile has been prepared to execute the proper sequence of commands required to compile and link the M33 STL. In particular, it performs the following steps:

1. Building C files, contained in STL/diagnostic/scheduler and STL/diagnostic/common folders
2. Preprocessing Assembly files, creating a temporary .src file used as a starting point for assembly building
3. Building the assembly files using the .src files created at step 2.
4. Linking the outcomes of the previous steps (for each step, a set of .obj files is created) and producing a .elf file to be used for running the M33 STL

The "Makefile" shall be called by the user from the folder "<Project_folder>\rcar_u5lx\Tasking" with the command make all DEVICE=<device_name>, where device_name can be one of the following:

- U5L1
- U5L2
- U5L4 (default if the DEVICE parameter is not specified)

This command performs the following operations:

1. It deletes the outcomes of the previous building process if present it removes the "Debug", the "asm", and "objects" folders and their content)
2. It executes the building and linking process:
   a. If everything ends with success, in the "build" folder, a file named "*M33_STL.elf*" is created (it is the binary to be used for running the M33 STL). The addresses of the different sections are aligned to what has been specified in the "cm33.lsl" (contained in "<Project_folder>\rcar_u5lx\Tasking\<DEVICE>" folder)

b. If there is an error during the building/linking process, no .elf file is generated in the "build" folder

The "Makefile" can also be used to delete the objects, Debug, and asm folders and their content, without starting the compilation, using the command "make clean".

The compiler and linking flags are specified in the "config.mk" file (contained in "<Project_folder>\rcar_u5lx\Tasking" folder). In particular, the following options are used to build and link the M33 STL Code:

Tasking Compiler – C Code – ccarm command

- --cpu=Cortex-M33-SP
- -O2
- --thumb
- --fpu=VFPv5-sp
- -g3
- --dep-file
- -OF

Tasking Preprocessing – Assembly Code – carm command

- -E
- -DTASKING
- -D__ASSEMBLER__

Tasking Compiler – Assembly Code – ccarm command

- --cpu=Cortex-M33-SP
- --thumb
- --fpu=VFPv5-sp
- -g3

For these 3 steps, the following directories are included in the command options, as follows:

- -I<Project_folder>/rcar_u5lx/Tasking/../STL/diagnostic/common/inc
- -I<Project_folder>/rcar_u5lx/Tasking/../STL/diagnostic/scheduler/inc
- -I<Project_folder>/rcar_u5lx/Tasking/../STL/diagnostic/tests/cpu/inc
- -I<Project_folder>/rcar_u5lx/Tasking/../STL/diagnostic/tests/mpu/inc

- -I<Project_folder>/rcar_u5lx/Tasking/../STL/diagnostic/tests/nvic/inc

Tasking Linker – ccarm command

- --cpu=Cortex-M33-SP
- --fpu=VFPv5-sp
- --thumb
- -Wl-O2
- -d$(PROJ_DIR)/<DEVICE>/cm33.lsl

Where $(PROJ_DIR) is the <Project_folder>/rcar_u5lx/Keil/ folder

3.6.3.6 Windriver Compiler

To compile and run the M33 STL code using the Windriver Diab Compiler, the file "Makefile" in "<Project_folder>\rcar_u5lx\Windriver" shall be used. This makefile has been prepared to execute the proper sequence of commands required to compile and link the M33 STL. In particular, it performs the following steps:

1. Building C files, contained in STL/diagnostic/scheduler and STL/diagnostic/common folders
2. Building the assembly files, contained in STL/diagnostic/common, STL/diagnostic/tests/cpu, STL/diagnostic/tests/mpu, and STL/diagnostic/tests/nvic folders
3. Linking the outcomes of the previous steps (for each step, a set of .o files is created) and producing a .out file to be used for running the M33 STL

The "Makefile" shall be called by the user from the folder "<Project_folder>\rcar_u5lx\Windriver" with the command make all DEVICE=<device_name>, where device_name can be one of the following:

- U5L1
- U5L2
- U5L4 (default if the DEVICE parameter is not specified)

This command performs the following operations:

1. It deletes the outcomes of the previous building process if present (it removes the "Debug" and "objects" folders and their content)
2. It executes the building and linking process:
    a. If everything ends with success, in the "Debug" folder, a file named "M*33_STL.out"* is created (it is the binary to be used for running the M33 STL). The addresses of the different sections are aligned to what has been specified in the "cm33.dld" file (contained in "<Project_folder>\rcar_u5lx\Windriver\<DEVICE>" folder)

b. If there is an error during the building/linking process, no .out file is generated in the "Debug" folder

The "Makefile" can also be used to delete the objects and Debug folders and their content, without starting the compilation, using the command "make clean".

The compiler and linking flags are specified in the "config.mk" file (contained in "<Project_folder>\rcar_u5lx\Windriver" folder). In particular, the following options are used to build and link the M33 STL Code:

Windriver Compiler – C Code – dcc command

- -tARMCORTEXM33MF:simple
- -gdwarf-4
- -MMD
- -O2
- -D__WINDRIVER__

Windriver Compiler – Assembly Code – dcc command

- -tARMCORTEXM33MF:simple
- -W:as:.asm
- -gdwarf-4
- -O2
- -Xpreprocess-assembly
- -xassembler-with-cpp
- -D__WINDRIVER__
- -D__ASSEMBLER__

For these 2 steps, the following directories are included in the command options, as follows:

- -I<Project_folder>/rcar_u5lx/Windriver/../STL/diagnostic/common/inc
- -I<Project_folder>/rcar_u5lx/Windriver/../STL/diagnostic/scheduler/inc
- -I<Project_folder>/rcar_u5lx/Windriver/../STL/diagnostic/tests/cpu/inc
- -I<Project_folder>/rcar_u5lx/Windriver/../STL/diagnostic/tests/mpu/inc
- -I<Project_folder>/rcar_u5lx/Windriver/../STL/diagnostic/tests/nvic/inc

Windriver Linker – dld command

- -tARMCORTEXM33MF:simple
- -lc
- -limpl
- -m6
- -e Reset_Handler
- -Xremove-unused-sections
- $(PROJ_DIR)/<DEVICE>/cm33.dld

Where $(PROJ_DIR) is the <Project_folder>/rcar_u5lx/Windriver/ folder

3.6.3.7 Hightec Compiler

To compile and run the M33 STL code using HighTec ARM Development Platform, the file "Makefile" in "<Project_folder>\rcar_u5lx\Hightec" shall be used. This makefile has been prepared to execute the proper sequence of commands required to compile and link the M33 STL. In particular, it performs the following steps:

1. Building C files, contained in STL/diagnostic/scheduler and STL/diagnostic/common folders
2. Building the assembly files, contained in STL/diagnostic/common, STL/diagnostic/tests/cpu, STL/diagnostic/tests/mpu, and STL/diagnostic/tests/nvic folders
3. Linking the outcomes of the previous steps (for each step, a set of .o files is created) and producing a .elf file to be used for running the M33 STL

The "Makefile" shall be called by the user from the folder "<Project_folder>\rcar_u5lx\Hightec" with the command make all DEVICE=<device_name>, where device_name can be one of the following:

- U5L1
- U5L2
- U5L4 (default if the DEVICE parameter is not specified)

This command performs the following operations:

3. It deletes the outcomes of the previous building process if present (it removes the "Debug" and "objects" folders and their content)
4. It executes the building and linking process:
   a. If everything ends with success, in the "Debug" folder, a file named "M*33_STL.elf"* is created (it is the binary to be used for running the M33 STL). The addresses of the

different sections are aligned to what has been specified in the “section.ld” file (contained in “<Project_folder>\rcar_u5lx\Hightec\<DEVICE>” folder)

b. If there is an error during the building/linking process, no .out file is generated in the “Debug” folder

The “Makefile” can also be used to delete the objects and Debug folders and their content, without starting the compilation, using the command “make clean”.

The compiler and linking flags are specified in the “config.mk” file (contained in “<Project_folder>\rcar_u5lx\Hightec” folder). In particular, the following options are used to build and link the M33 STL Code:

Hightec Compiler – C Code – clang command

- --target=arm-none-eabi
- -mcpu=cortex-m33
- -mthumb
- -mfloat-abi=softfp
- -O2
- -fno-short-enums
- -ffunction-sections
- -mfpu=fpv5-d16
- -fno-common
- -fdata-sections
- -gdwarf-4
- -MMD

Hightec Compiler – Assembly Code – clang command

- --target=arm-none-eabi
- -x assembler-with-cpp
- -mcpu=cortex-m33
- -mthumb
- -mfloat-abi=softfp
- -mfpu=fpv5-d16
- -O2
- -fno-common

- -gdwarf-4
- -fno-short-enums
- -MMD
- -DGCC_KEIL_HIGHTEC_CMP
- -D__ASSEMBLER__

For these 2 steps, the following directories are included in the command options, as follows:

- -I<Project_folder>/rcar_u5lx/Hightec/../STL/diagnostic/common/inc
- -I<Project_folder>/rcar_u5lx/Hightec/../STL/diagnostic/scheduler/inc
- -I<Project_folder>/rcar_u5lx/Hightec/../STL/diagnostic/tests/cpu/inc
- -I<Project_folder>/rcar_u5lx/Hightec/../STL/diagnostic/tests/mpu/inc
- -I<Project_folder>/rcar_u5lx/Hightec/../STL/diagnostic/tests/nvic/inc

Hightec Linker – clang command

- -mcpu=cortex-m33
- -mthumb
- -mfloat-abi=softfp
- -mfpu=fpv5-d16
- -Wl,--entry=Reset_Handler
- -Wl,--discard-all
- -Wl,--gc-sections
- -Wl,--check-sections
- -Wl,-Map=$(OUT_DIR)/M33_STL.map
- -T $(PROJ_DIR)/<DEVICE>/section.ld

Where $(PROJ_DIR) is the <Project_folder>/rcar_u5lx/Hightec/ folder

Where $(OUT_DIR) is the <Project_folder>/rcar_u5lx/Hightec/Debug folder

A REGISTER INFO

This section contains information about the arrays (m33_stl_nvicRegisterInfo and m33_stl_mpuRegisterInfo) used to store the data of the tested registers. These arrays are defined as follows:

```
typedef struct {
   const uint32_t regAddress;
   const uint32_t maskVal;
   uint32_t expectedValue;
} m33_stl_registerInfo_t;
```

Where:

- regAddres: the address of the tested register
- maskValue: the value of the mask used to determine the fields to be tested
- expectedValue: the expected value of the register, to be compared to the read-back value

The following subsections contain details about the tested registers, both for NVIC and MPU tests

A.1 m33_stl_nvicRegisterInfo

In Table 11, the information of the m33_stl_nvicRegisterInfo array is reported

Array Index	Register	Mask
0	SYST_CSR	0x00000007U
1	SYST_CALIB	0x80000000U
2	NVIC_ISER0	0xFFFFFFFFU
3	NVIC_ISER1	0xFFFFFFFFU
4	NVIC_ISER2	0xFFFFFFFFU
5	NVIC_ISER3	0xFFFFFFFFU
6	NVIC_ISER4	0xFFFFFFFFU
7	NVIC_ISER5	0xFFFFFFFFU
8	NVIC_ISER6	0xFFFFFFFFU
9	NVIC_ISER7	0xFFFFFFFFU
10	NVIC_ISER8	0xFFFFFFFFU
11	NVIC_ISER9	0xFFFFFFFFU
12	NVIC_ISER10	0xFFFFFFFFU
13	NVIC_ISER11	0xFFFFFFFFU
14	NVIC_ISER12	0xFFFFFFFFU
15	NVIC_ISER13	0xFFFFFFFFU
16	NVIC_ISER14	0xFFFFFFFFU
17	NVIC_ISER15	0xFFFFFFFFU
18	NVIC_ICER0	0xFFFFFFFFU
19	NVIC_ICER1	0xFFFFFFFFU
20	NVIC_ICER2	0xFFFFFFFFU

21	NVIC_ICER3	0xFFFFFFFFU
22	NVIC_ICER4	0xFFFFFFFFU
23	NVIC_ICER5	0xFFFFFFFFU
24	NVIC_ICER6	0xFFFFFFFFU
25	NVIC_ICER7	0xFFFFFFFFU
26	NVIC_ICER8	0xFFFFFFFFU
27	NVIC_ICER9	0xFFFFFFFFU
28	NVIC_ICER10	0xFFFFFFFFU
29	NVIC_ICER11	0xFFFFFFFFU
30	NVIC_ICER12	0xFFFFFFFFU
31	NVIC_ICER13	0xFFFFFFFFU
32	NVIC_ICER14	0xFFFFFFFFU
33	NVIC_ICER15	0xFFFFFFFFU
34	NVIC_IABR0	0xFFFFFFFFU
35	NVIC_IABR1	0xFFFFFFFFU
36	NVIC_IABR2	0xFFFFFFFFU
37	NVIC_IABR3	0xFFFFFFFFU
38	NVIC_IABR4	0xFFFFFFFFU
39	NVIC_IABR5	0xFFFFFFFFU
40	NVIC_IABR6	0xFFFFFFFFU
41	NVIC_IABR7	0xFFFFFFFFU
42	NVIC_IABR8	0xFFFFFFFFU

43	NVIC_IABR9	0xFFFFFFFFU
44	NVIC_IABR10	0xFFFFFFFFU
45	NVIC_IABR11	0xFFFFFFFFU
46	NVIC_IABR12	0xFFFFFFFFU
47	NVIC_IABR13	0xFFFFFFFFU
48	NVIC_IABR14	0xFFFFFFFFU
49	NVIC_IABR15	0xFFFFFFFFU
50	NVIC_ITNS0	0xFFFFFFFFU
51	NVIC_ITNS1	0xFFFFFFFFU
52	NVIC_ITNS2	0xFFFFFFFFU
53	NVIC_ITNS3	0xFFFFFFFFU
54	NVIC_ITNS4	0xFFFFFFFFU
55	NVIC_ITNS5	0xFFFFFFFFU
56	NVIC_ITNS6	0xFFFFFFFFU
57	NVIC_ITNS7	0xFFFFFFFFU
58	NVIC_ITNS8	0xFFFFFFFFU
59	NVIC_ITNS9	0xFFFFFFFFU
60	NVIC_ITNS10	0xFFFFFFFFU
61	NVIC_ITNS11	0xFFFFFFFFU
62	NVIC_ITNS12	0xFFFFFFFFU
63	NVIC_ITNS13	0xFFFFFFFFU
64	NVIC_ITNS14	0xFFFFFFFFU

65	NVIC_ITNS15	0xFFFFFFFFU
66	NVIC_IPR0	0xFFFFFFFFU
67	NVIC_IPR1	0xFFFFFFFFU
68	NVIC_IPR2	0xFFFFFFFFU
69	NVIC_IPR3	0xFFFFFFFFU
70	NVIC_IPR4	0xFFFFFFFFU
71	NVIC_IPR5	0xFFFFFFFFU
72	NVIC_IPR6	0xFFFFFFFFU
73	NVIC_IPR7	0xFFFFFFFFU
74	NVIC_IPR8	0xFFFFFFFFU
75	NVIC_IPR9	0xFFFFFFFFU
76	NVIC_IPR10	0xFFFFFFFFU
77	NVIC_IPR11	0xFFFFFFFFU
78	NVIC_IPR12	0xFFFFFFFFU
79	NVIC_IPR13	0xFFFFFFFFU
80	NVIC_IPR14	0xFFFFFFFFU
81	NVIC_IPR15	0xFFFFFFFFU
82	NVIC_IPR16	0xFFFFFFFFU
83	NVIC_IPR17	0xFFFFFFFFU
84	NVIC_IPR18	0xFFFFFFFFU
85	NVIC_IPR19	0xFFFFFFFFU
86	NVIC_IPR20	0xFFFFFFFFU

87	NVIC_IPR21	0xFFFFFFFFU
88	NVIC_IPR22	0xFFFFFFFFU
89	NVIC_IPR23	0xFFFFFFFFU
90	NVIC_IPR24	0xFFFFFFFFU
91	NVIC_IPR25	0xFFFFFFFFU
92	NVIC_IPR26	0xFFFFFFFFU
93	NVIC_IPR27	0xFFFFFFFFU
94	NVIC_IPR28	0xFFFFFFFFU
95	NVIC_IPR29	0xFFFFFFFFU
96	NVIC_IPR30	0xFFFFFFFFU
97	NVIC_IPR31	0xFFFFFFFFU
98	NVIC_IPR32	0xFFFFFFFFU
99	NVIC_IPR33	0xFFFFFFFFU
100	NVIC_IPR34	0xFFFFFFFFU
101	NVIC_IPR35	0xFFFFFFFFU
102	NVIC_IPR36	0xFFFFFFFFU
103	NVIC_IPR37	0xFFFFFFFFU
104	NVIC_IPR38	0xFFFFFFFFU
105	NVIC_IPR39	0xFFFFFFFFU
106	NVIC_IPR40	0xFFFFFFFFU
107	NVIC_IPR41	0xFFFFFFFFU
108	NVIC_IPR42	0xFFFFFFFFU

109	NVIC_IPR43	0xFFFFFFFFU
110	NVIC_IPR44	0xFFFFFFFFU
111	NVIC_IPR45	0xFFFFFFFFU
112	NVIC_IPR46	0xFFFFFFFFU
113	NVIC_IPR47	0xFFFFFFFFU
114	NVIC_IPR48	0xFFFFFFFFU
115	NVIC_IPR49	0xFFFFFFFFU
116	NVIC_IPR50	0xFFFFFFFFU
117	NVIC_IPR51	0xFFFFFFFFU
118	NVIC_IPR52	0xFFFFFFFFU
119	NVIC_IPR53	0xFFFFFFFFU
120	NVIC_IPR54	0xFFFFFFFFU
121	NVIC_IPR55	0xFFFFFFFFU
122	NVIC_IPR56	0xFFFFFFFFU
123	NVIC_IPR57	0xFFFFFFFFU
124	NVIC_IPR58	0xFFFFFFFFU
125	NVIC_IPR59	0xFFFFFFFFU
126	NVIC_IPR60	0xFFFFFFFFU
127	NVIC_IPR61	0xFFFFFFFFU
128	NVIC_IPR62	0xFFFFFFFFU
129	NVIC_IPR63	0xFFFFFFFFU
130	NVIC_IPR64	0xFFFFFFFFU

131	NVIC_IPR65	0xFFFFFFFFU
132	NVIC_IPR66	0xFFFFFFFFU
133	NVIC_IPR67	0xFFFFFFFFU
134	NVIC_IPR68	0xFFFFFFFFU
135	NVIC_IPR69	0xFFFFFFFFU
136	NVIC_IPR70	0xFFFFFFFFU
137	NVIC_IPR71	0xFFFFFFFFU
138	NVIC_IPR72	0xFFFFFFFFU
139	NVIC_IPR73	0xFFFFFFFFU
140	NVIC_IPR74	0xFFFFFFFFU
141	NVIC_IPR75	0xFFFFFFFFU
142	NVIC_IPR76	0xFFFFFFFFU
143	NVIC_IPR77	0xFFFFFFFFU
144	NVIC_IPR78	0xFFFFFFFFU
145	NVIC_IPR79	0xFFFFFFFFU
146	NVIC_IPR80	0xFFFFFFFFU
147	NVIC_IPR81	0xFFFFFFFFU
148	NVIC_IPR82	0xFFFFFFFFU
149	NVIC_IPR83	0xFFFFFFFFU
150	NVIC_IPR84	0xFFFFFFFFU
151	NVIC_IPR85	0xFFFFFFFFU
152	NVIC_IPR86	0xFFFFFFFFU

153	NVIC_IPR87	0xFFFFFFFFU
154	NVIC_IPR88	0xFFFFFFFFU
155	NVIC_IPR89	0xFFFFFFFFU
156	NVIC_IPR90	0xFFFFFFFFU
157	NVIC_IPR91	0xFFFFFFFFU
158	NVIC_IPR92	0xFFFFFFFFU
159	NVIC_IPR93	0xFFFFFFFFU
160	NVIC_IPR94	0xFFFFFFFFU
161	NVIC_IPR95	0xFFFFFFFFU
162	NVIC_IPR96	0xFFFFFFFFU
163	NVIC_IPR97	0xFFFFFFFFU
164	NVIC_IPR98	0xFFFFFFFFU
165	NVIC_IPR99	0xFFFFFFFFU
166	NVIC_IPR100	0xFFFFFFFFU
167	NVIC_IPR101	0xFFFFFFFFU
168	NVIC_IPR102	0xFFFFFFFFU
169	NVIC_IPR103	0xFFFFFFFFU
170	NVIC_IPR104	0xFFFFFFFFU
171	NVIC_IPR105	0xFFFFFFFFU
172	NVIC_IPR106	0xFFFFFFFFU
173	NVIC_IPR107	0xFFFFFFFFU
174	NVIC_IPR108	0xFFFFFFFFU

175	NVIC_IPR109	0xFFFFFFFFU
176	NVIC_IPR110	0xFFFFFFFFU
177	NVIC_IPR111	0xFFFFFFFFU
178	NVIC_IPR112	0xFFFFFFFFU
179	NVIC_IPR113	0xFFFFFFFFU
180	NVIC_IPR114	0xFFFFFFFFU
181	NVIC_IPR115	0xFFFFFFFFU
182	NVIC_IPR116	0xFFFFFFFFU
183	NVIC_IPR117	0xFFFFFFFFU
184	NVIC_IPR118	0xFFFFFFFFU
185	NVIC_IPR119	0xFFFFFFFFU

Table 11 – m33_stl_nvicRegisterInfo array

A.2 m33_stl_mpuRegisterInfo

In Table 12, the information of the m33_stl_mpuRegisterInfo array is reported

Array Index	Register	Mask
0	MPU_CTRL	0x00000007U
1	MPU_RBAR	0xFFFFFFFFU
2	MPU_RBAR_A1	0xFFFFFFFFU
3	MPU_RBAR_A2	0xFFFFFFFFU
4	MPU_RBAR_A3	0xFFFFFFFFU
5	MPU_RLAR	0xFFFFFFFFU
6	MPU_RLAR_A1	0xFFFFFFFFU
7	MPU_RLAR_A2	0xFFFFFFFFU
8	MPU_RLAR_A3	0xFFFFFFFFU
9	MPU_TYPE	0x0000FF01U
10	MPU_RNR	0x000000FFU
11	MPU_MAIR0	0xFFFFFFFFU
12	MPU_MAIR1	0xFFFFFFFFU

Table 12 – m33_stl_mpuRegisterInfo array

B STL Test Content Details

This section details the content of each STL Test, providing information about the tested instructions and registers.

STL Test Name	STL Test Content Details
m33_stl_cpu_n000	Tested instructions: • ADC • ADD (immediate) • ADD (large immediate) • ADD (register) • ADD (high register) • AND • TST
m33_stl_cpu_n001	Tested instructions: • ASR (immediate) • ASR (register) • BIC • EOR • ORR
m33_stl_cpu_n002	Tested instructions: • LSL(immediate) • LSL(register) • LSR(immediate) • LSR(register)
m33_stl_cpu_n003	Tested instructions: • MOV(immediate) • MOV(low register) • MOV(high register) • MVN • MOVT • MOVW
m33_stl_cpu_n004	Tested instructions: • MUL • RSBS • ROR(register) • SBC(register) • SUB(immediate) • SUB(large immediate) • SUB(register)
m33_stl_cpu_n005	Tested instructions: • LDR and STR • LDRB and LDRSB • LDRH and LDRSH • STRB and STRH • LDR and ADR with label

Commented [NL54]: Have the instructions CLREX, LDAEX, LDR{type}T, LDRD, LDREX{type}, PLD, POP, PUSH and STLEX been tested?
If not, what is the rationale behind not testing them?

Commented [LM55R54]: Exclusive Load and Store instructions have not been tested, since they may lead to synchronization issues in case of interrupts, and the expected result may be altered, causing a fault detection when instead a fault is not present (false positive).
Other instructions (e.g., POP and PUSH) are implicitly tested since used for saving and restoring GPRs.
Anyway, generally, if an instruction is not tested, it is a development decision:

- based on similarity with other instructions that share the same HW logic
- due to potential problems in the testing procedure (as the exclusive load and store instructions)
- Already extensively used in the STL code (e.g, PUSH, POP)

m33_stl_cpu_n006	Tested instructions: - B (cond) - BL - CBZ - CBZN
m33_stl_cpu_n007	Tested instructions: - SDIV - UDIV
m33_stl_cpu_n008	Tested instructions: - CMP - CMN - CMP(immediate)
m33_stl_cpu_n009	Tested instructions: - Test FP Single Precision registers (s0-s31)
m33_stl_cpu_n010	Tested instructions: - LDA and STL - LDAB - LDAH - STLB and STLH - LDM and STM
m33_stl_cpu_n011	Tested instructions: - SXTH - SXTB - UXTH - UXTB - REV - REV16 - REVSH
m33_stl_cpu_n012	Tested instructions: - Test General Purpose registers (from R0 to R12)
m33_stl_cpu_n013	Tested instructions: - VFMA.F32 with RM mode
m33_stl_cpu_n014	Tested instructions: - VCVTR.S32.F32 and VCVTR.U32.F32 with RZ mode
m33_stl_cpu_n015	Tested instructions: - VCVTR.S32.F32 and VCVTR.U32.F32 with RM mode
m33_stl_cpu_n016	Tested instructions: - VCVTR.S32.F32 and VCVTR.U32.F32 with RP mode
m33_stl_cpu_n017	Tested instructions: - VCVTR.S32.F32 and VCVTR.U32.F32 with RN mode
m33_stl_cpu_n018	Tested instructions: - VCVT.S32.F32 and VCVT.U32.F32 - VCVTA.S32.F32 and VCVTA.U32.F32 - VCVTM.S32.F32 and VCVTM.U32.F32 - VCVTN.S32.F32 and VCVTN.U32.F32 - VCVTP.S32.F32 and VCVTP.U32.F32

Commented [NL56]: Have the instructions BL, BLX, BLXNS, BX, BXNS, CBNZ, CBZ, IT, TBB, and TBH been tested?
If not, what is the rationale behind not testing them?

Commented [LM57R56]: BL, CBZ and CBNZ instructions have been added since missing in the list.
Regarding the other instructions, BLX, BLXNS, BX, BXNS have not been tested since they cause a security exchange, while IT, TBH and TBB are already covered by branch and jump instructions

m33_stl_cpu_n019	Tested instructions: • VFMS.F32 instruction with RN mode
m33_stl_cpu_n020	Tested instructions: • ADC • ADD (immediate) • ADD (Register) • SBC • SUB (register) • SUB (immediate)
m33_stl_cpu_n021	Tested instructions: • LDR and ADR with label • LDMIA, LDMFD, LDMDB and LDMEA • LDR • LDR and SRT • STRD • LDRB, LDRH, LDRSB and LDRSH with label
m33_stl_cpu_n022	Tested instructions: • CMN • CMP
m33_stl_cpu_n023	Tested instructions: • ASR • ASRS (Register) • LSL (Immediate) • LSL (Register) • LSLS • LSR (Immediate) • LSR (Register) • LSRS
m33_stl_cpu_n024	Tested instructions: • AND • EOR • ORN • ORR
m33_stl_cpu_n025	Tested instructions: • BFC • BFI • BIC (immediate) • BIC (register) • CLZ • SBFX • SEL • UBFX
m33_stl_cpu_n026	Tested instructions: • MLA • MLS • MUL
m33_stl_cpu_n027	Tested instructions: • MOV • MOVS

	• MVN • MVNS
m33_stl_cpu_n028	Tested instructions: • PKHBT • PKHTB
m33_stl_cpu_n029	Tested instructions: • QADD • QADD16 • QADD8 • QASX • QSAX • QDADD • QDSUB • QSUB • QSUB16 • QSUB8 • SSAT • SSAT16
m33_stl_cpu_n030	Tested instructions: • RBIT • REV.W • REV16.W • REVSH.W • ROR (immediate) • ROR (Register) • RORS (immediate) • RORS (Register) • RRX • RRXS • RSB
m33_stl_cpu_n031	Tested instructions: • SADD16 • SADD8 • SASX • SSAX • SHADD16 • SHADD8 • SHASX • SHSAX • SHSUB16 • SHSUB8 • SSUB16 • SSUB8
m33_stl_cpu_n032	Tested instructions: • SMLABB, SMLABT, SMLATB and SMLATT • SMLAD, SMLADX, SMLAWB and SMLAWT • SMLAL • SMLALBB, SMLALBT, SMLALTB, SMLALTT, SMLALD and SMLALDX

	• SMLSD, SMLSDX, SMLSLD and SMLSLDX • SMMLA, SMMLAR, SMMLS, SMMLSR, SMMUL and SMMULR • SMUAD, SMUADX, SMUSD and SMUSDX • SMULBB, SMULBT, SMULTB, SMULTT, SMULWB and SMULWT • SMULL
m33_stl_cpu_n033	Tested instructions: • SXTAB • SXTAB16 • SXTAH • SXTB • SXTB16 • SXTH
m33_stl_cpu_n034	• TEQ(Immediate) • TEQ(Register) • TST(Immediate) • TST(Register)
m33_stl_cpu_n035	Tested instructions: • UADD16 and UADD8 • UASX and USAX • UHADD16 and UHADD8 • UHASX and UHSAX • UHSUB16 and UHSUB8 • USAD8 and USADA8 • USUB16 and USUB8
m33_stl_cpu_n036	Tested instructions: • UMAAL • UMLAL • UMULL
m33_stl_cpu_n037	Tested instructions: • UQADD16 • UQADD8 • UQASX and UQSAX • UQSUB16 • UQSUB8 • USAT • USAT16
m33_stl_cpu_n038	Tested instructions: • UXTAB • UXTAB16 • UXTAH • UXTB • UXTB16 • UXTH
m33_stl_cpu_n039	Tested instructions: • VADD.F32 instruction with RM mode • VDIV.F32 instruction with RM mode • VMUL.F32 instruction with RM mode • VSQRT.F32 instruction with RM mode

Commented [NL58]: Has the SMULL instruction been tested?
If not, what is the rationale behind not testing it?

Commented [LM59R58]: SMULL instruction has been added to the list since it was missing

	• VSUB.F32 instruction with RM mode
m33_stl_cpu_n040	Tested instructions: • VADD.F32 instruction with RN mode • VDIV.F32 instruction with RN mode • VMUL.F32 instruction with RN mode • VSQRT.F32 instruction with RN mode • VSUB.F32 instruction with RN mode
m33_stl_cpu_n041	Tested instructions: • VADD.F32 instruction with RP mode • VDIV.F32 instruction with RP mode • VMUL.F32 instruction with RP mode • VSQRT.F32 instruction with RP mode • VSUB.F32 instruction with RP mode
m33_stl_cpu_n042	Tested instructions: • VADD.F32 instruction with RZ mode • VDIV.F32 instruction with RZ mode • VMUL.F32 instruction with RZ mode • VSQRT.F32 instruction with RZ mode • VSUB.F32 instruction with RZ mode
m33_stl_cpu_n043	Tested instructions: • VABS.F32 • VNEG.F32
m33_stl_cpu_n044	Tested instructions: • VCMP.F32, VCMPE.F32 and VSEL • VMAXNM.F32 • VMINNM.F32
m33_stl_cpu_n045	Tested instructions: • VRINTA.F32 • VRINTM.F32 • VRINTN.F32 • VRINTP.F32
m33_stl_cpu_n046	Tested instructions: • VRINTR.F32 • VRINTX.F32 • VRINTZ.F32
m33_stl_cpu_n047	Tested instructions: • VCVTB.F16.F32with RN mode and AHP enabled • VCVTT.F16.F32with RN mode and AHP enabled
m33_stl_cpu_n048	Tested instructions: • VCVTB with RN mode and DN enabled • VCVTT with RN mode and DN enabled
m33_stl_cpu_n049	Tested instructions: • VCVT.<dt>.F32
m33_stl_cpu_n050	Tested instructions: • VCVTB.F16.F32 • VCVTT.F16.F32
m33_stl_cpu_n051	Tested instructions: • VCVT.F32.<dt> • VCVTB.F32.F16

	• VCVTT.F32.F16
m33_stl_cpu_n052	Tested instructions: • VMOV
m33_stl_cpu_n053	Tested instructions: • VNMLA.F32 instruction with RM mode
m33_stl_cpu_n054	Tested instructions: • VCVT.F32.S32 and VCVT.F32.U32 with RN mode
m33_stl_cpu_n055	Tested instructions: • VNMLS.F32 instruction with RM mode
m33_stl_cpu_n056	Tested instructions: • VMLS.F32 instruction with RM mode
m33_stl_cpu_n057	Tested instructions: • VCVT.F32.S32 and VCVT.F32.U32 with RP mode
m33_stl_cpu_n058	Tested instructions: • VMLA.F32 instruction with RM mode • VNMUL.F32 instruction with RM mode
m33_stl_cpu_n059	Tested instructions: • VLDM and VSTM • VLDR and VSTR
m33_stl_cpu_n060	Tested instructions: • VCVT.F32.S32 and VCVT.F32.U32 with RM mode
m33_stl_cpu_n061	Tested instructions: • VFNMS.F32 instruction with RZ mode
m33_stl_cpu_n062	Tested instructions: • VFNMA.F32 instruction with RP mode
m33_stl_cpu_n063	Tested instructions: • VCVT.F32.S32 and VCVT.F32.U32 with RZ mode
m33_stl_cpu_n064	Tested registers: • SHPR1 • SHPR2 • SHPR3 • VTOR • SHCSR • BASEPRI
m33_stl_cpu_n065	Tested registers: • CCSIDR • CLIDR • CTR • ICTR • ID_PFR0 • ID_ISAR4 • ID_MMFR2 • ID_MMFR0 • MVFR2 • MVFR0 • MVFR1

Commented [NL60]: Have the instructions ACTLR, CPUID, ICSR, AIRCR, SCR, CCR, CFSR, MMFSR, BFSR, UFSR, HFSR, PIDR3, PIDR5, PIDR6, PIDR7, CIDR0-3, PIDR0-4, DEVARCH, FAULTMASK, PRIMASK and BFAR been tested? If not, what is the rationale behind not testing them?

	• ID_DFR0 • ID_PFR1
m33_stl_cpu_n066	Tested registers: • FPCCR • FPCAR • FPDSCR • FPSCR
m33_stl_cpu_n067	Tested instructions: • Dual-issue strategy (ALU / Load and Store) • Dual-issue strategy (ALU / Shift) • Dual-issue strategy (ALU / MAC)
m33_stl_cpu_n068	Tested instructions: • Dual-issue strategy (ALU / FPU) • Dual-issue strategy (FPU / Load and Store) • Dual-issue strategy (FPU / FPU Load and Store)
m33_stl_cpu_n069	Tested instructions: • VADD.F32 • VCMP.F32 and VCMPE.F32 • VMUL.F32
m33_stl_cpu_n070	Tested instructions: • CLZ
m33_stl_cpu_n071	Tested sequence of the following integer instructions: • Divide and Multiply • Saturation • Logical • Packing • Arithmetic • Shift and Rotation
m33_stl_cpu_n072	Tested sequence of the following floating-point instructions: • FP Multiply and Divide • FP Addition
m33_stl_mpu_n000	Tested functionality of the MPU module: • Write operation in a not permitted area • Branch to a region with the execute never attribute enabled
m33_stl_mpu_n001	Control Registers of the MPU module. The list of tested registers can be found in Table 12
m33_stl_nvic_n000	Control Registers of the NVIC module. The list of tested registers can be found in Table 11

Table 13 – STL Tests Content Details

Commented [LM61R60]: ACTLR, CPUID, SCR, MMFSR, BFSR, UFSR registers have not been tested since during the contact phase, it was understood that these registers were not required by Renesas
ICSR, AIRCR, CCR and BFAR registers cannot be tested since modifying them may lead to unexpected behavior of the CPU (the testing approach is write/read-back)
CFSR and HFSR cannot be tested using the write/read-back approach since they are 32-bit read/write-one-to-clear registers, where a known value cannot be written
PIDR3, PIDR5, PIDR6, PIDR7, CIDR0-3, PIDR0-4, DEVARCH registers belong to CoreSight module which is a debug module and it is not in scope for the STL
FAULTMASK and PRIMASK registers are implicitly tested since they are used to disable/enable interrupts in the STL

Commented [NL62]: Have the instructions FLDMDBX, FLDMIAX, FSTMDBX, FSTMIAX, VMRS, VMSR, VPOP, VPUSH been tested?
If not, what is the rationale behind not testing them?

Commented [LM63R62]: FLDMDBX, FLDMIAX, FSTMDBX, FSTMIAX instructions have not been tested since, in the Armv8-M manual, it is stated that such instructions are deprecated by Arm
VMRS, VMSR, VPOP, VPUSH instructions are implicitly tested when FPU control registers are accessed and when floating point registers are saved and restored for the context switch

Commented [NL64]: Have the registers SAU_CTRL, SAU_TYPE, SAU_RNR, SAU_RBAR, SAU_RLAR, SFSR, SFAR been tested?
If not, what is the rationale behind not testing them?

Commented [LM65R64]: SAU is not present in the M33 configuration used in U5L1, U5L2 and U5L4

Commented [NL66]: Have the registers SYST_RVR, SYST_CVR, NVIC_ISPR0-NVIC_ISPR15, NVIC_ICPR0-NVIC_ICPR15 been tested?
If not, what is the rationale behind not testing them?

Commented [LM67R66]: SYST_RVR, SYST_CVR are timer registers and are not in scope since their value is not predictable by the STL, hence not usable for comparison with known values. On the contrary, they cannot be tested using a write/read-back approach, since write operations on timer may be intrusive on the SW application.
NVIC_ISPR0-NVIC_ISPR15, NVIC_ICPR0-NVIC_ICPR15 are set and clear pending registers. As agreed in the email thread "**[Renesas-Resiltech] Finding for M0b release packages** ", NVIC_ISPRX and NVIC_ICPRX are not suitable to be tested using the agreed approach (i.e., comparison against the register golden value) since they are not static registers, as their value changes every time there is a pending interrupt. This means that, for each register, the STL integrator should update the golden copy with the new value of the register to avoid the STL notifying a mismatch between the read register and its golden value