RX Family

TFU Fault Diagnosis Example

Introduction

This document describes a fault diagnosis example of an Arithmetic Unit for Trigonometric Functions (TFU) peripheral circuit.

Target Device

This example supports the following devices.

- RX72M Group
- RX72T Group

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Contents

1. Overview

This document explains a fault diagnosis example of an Arithmetic Unit for Trigonometric Functions (TFU) peripheral circuit to apply kind of functional safety. The fault diagnosis is not only permanent fault but also transient fault.

1.1 TFU Diagnosis Example

This example is implemented in a project and can be used as the application example of TFU fault diagnosis.

This software does not get any certification including industrial functional safety specification. If a user needs any certification, the user implements the TFU diagnosis operation refer to this example on the user's system and needs to get the certification as the system.

1.2 Related documents

[1] RX Family Board Support Package Module Using Firmware Integration Technology, Rev.5.20, Document No. R01AN1685EJ0520, Apr 08, 2019.

[2] RX72M Group Renesas Starter Kit+ for RX72M CPU Board (Prototype) User's Manual, Rev. 1.00, Document No. R20UT4383EG0100, Jan 31, 2019.

[3] RX72T Group Renesas Starter Kit for RX72T User's Manual, Rev. 1.00, Document No. R20UT4272EG0100, Nov 30, 2018.

1.3 Hardware Structure

This example uses CPU, ROM and RAM to diagnosis TFU. Only when measuring operation time, MTU3 is optionally used.

In detail, please refer to RX72M/72T Group User's Manual: Hardware.

1.4 Software Structure

This example should be used with TFU intrinsic function to execute TFU hardware calculation. Case of transient error diagnosis, the math standard library, which is embedded in compiler, is used to create the expectation values. If performance evaluation needs, time measurement software is available. A diagnosis results such as error detection point, performance, are showed on the console[1].

[1] "Renesas Debug Virtual Console" case of e2 studio environment or "Terminal I/O" case of EWRX.

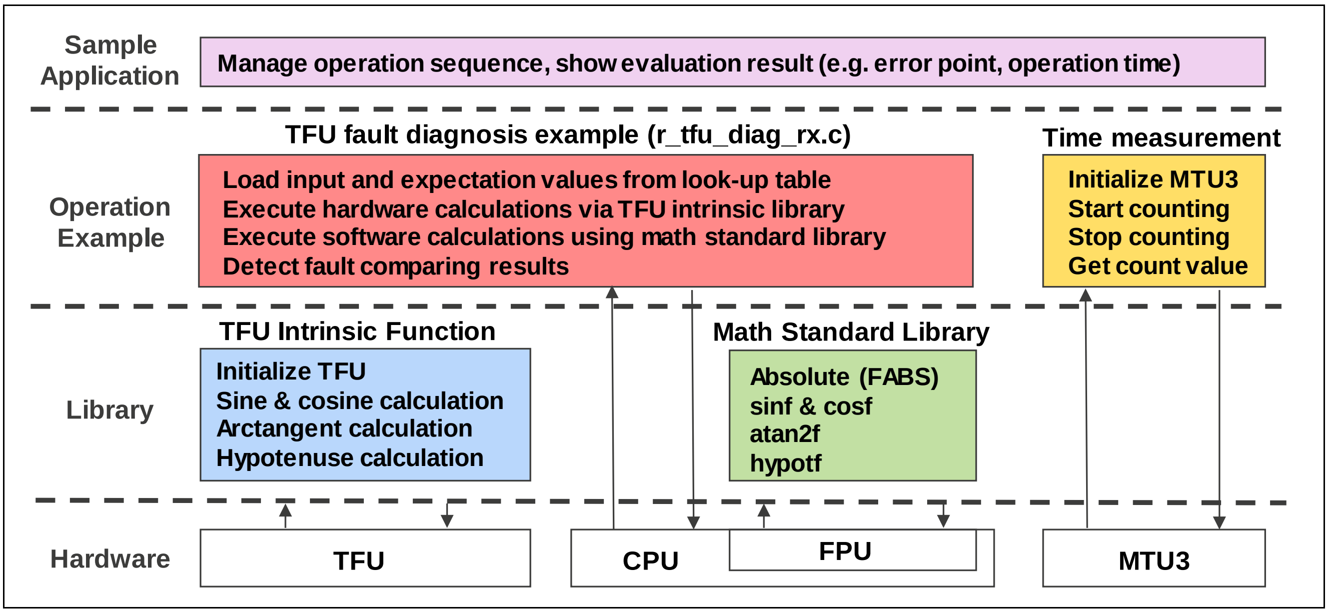

Figure 1.1 Software structure system example

1.5 File Structure

This sample codes are stored the "src", "r_config", "r_tfu_diag_rx", "r_bsp" and lower hierarchical folders. Figure 1.2 shows the source and header file structures of this sample. The "src" folder stores files of sample application, console output operation and time measurement operation. The “r_tfu_diag_rx” folder stores files related to this TFU fault diagnosis example (hereafter, TFU diagnosis software). An initial setting of this example uses the Renesas Board Support Package (BSP) [1].

```
src: sample application (main operation)
|  tfu_test.c
|
+ --- output_if: terminal output operation
|       show_label.c
|       show_label.h
|
+ --- tmr_if: time measurement operation
|       tmr_if.c
|       tmr_if.h

r_config: configuration setting
|       r_bsp_config.h
|       r_bsp_interrupt_config.h
|       r_tfu_diag_rx_config.h

r_tfu_diag_rx; TFU fault diagnosis example
|  r_tfu_diag_rx_if.h
|
+ --- src
|       r_tfu_diag_rx.c
|       r_tfu_diag_rx_private.h

r_bsp: BSP (Board Support Package) FIT module
```

Figure 1.2 File structure of this example

1.6 Outline of Functions

The functions of sample application and the lower layers show Table 1.1 and the API functions related to the diagnosis operation of TFU show Table 1.2 respectively.

Table 1.1 Functions of application and the lower layers

Item	Contents
main()	Main operation of this example.
start_label()	Show operation start indication to console.
end_label()	Show operation end indication to console.
show_result()	Show ICLK cycles and time of operation to console.
mtu3_dev_start()	Clear MTU3 module stop.
mtu3_dev_stop()	Set MTU3 module stop.
Init_timer()	Initial setting of MTU3.
start_eval()	Start counting of MTU3.
stop_eval()	Stop counting of MTU3.
get_eval_cycle()	Get current MTU3 counter value.

Table 1.2 API functions (TFU fault diagnosis)

Item	Contents
R_TFU_Diag_GetVersion()	Get version number of TFU diagnostic software.
R_TFU_Diag_Init()	Initial setting of TFU.
R_TFU_Diag_SinCos()	Diagnostic operation by sine and cosine numerical calculations.
R_TFU_Diag_AtanHypot()	Diagnostic operation by arctangent and hypotenuse numerical calculations.

2. Functional Information

This example is developed by the following principles.

2.1 Hardware Requirements

This example requires your MCU supports the following feature:

- TFU
- MTU3 (Optional)

2.2 Hardware Resource Requirements

This section details the hardware peripherals that this example requires. Unless explicitly stated, these resources must be reserved for the following driver, and the user cannot use them.

2.2.1 TFU

This example uses the TFU as the diagnosis target module. During operation of this example, the other task cannot access TFU.

2.2.2 MTU3 Channel (optional)

This example uses the MTU3 as the cascade operation connected to CH1 and CH2. If evaluating performance in this example, user cannot use the CH1 and CH2 of MTU3.

2.3 Software Requirements

This example is dependent on the following packages (FIT modules):

- r_bsp

2.4 Limitations

This software does not get any certification including industrial functional safety specification. If a user needs any certification, the user implements the TFU diagnosis operation refer to this example on his/her system and needs to get the certification as the system.

2.5 Supported Toolchains

This example has been confirmed to work with the toolchain listed in 5.1 Confirmed Operation Environment.

2.6 Header Files

Each function call is accessed by including one of or multiple files followed:

show_label.h, tmr_if.h, r_tfu_diag_rx_config.h, r_tfu_diag_rx_if.h and r_tfu_diag_rx_private.h which are supplied with this project code.

2.7 Integer Types

This project uses ANSI C99. These types are defined in *stdint.h*.

2.8 Configuration Overview

The configuration options in this example are specified in *r_tfu_diag_rx_config.h and tfu_test.c.* The option names and setting values are listed in the table below.

Configuration options	
#define RX_DEVICE_TYPE **#define DEVICE_RX72M (0)** **#define DEVICE_RX72T (1)** **- Default value = 0**	Specify a target device. - When the target device is RX72M, please set to 0. - When the target device is RX72T, please set to 1. The other device is not supported in this version.
#define LUTSinSize **- Default value = 256**	Look-up table size for sine and cosine calculation. Please set this value in this version.

Configuration options	
#define LUTAtanSize **- Default value = 256**	Look-up table size for arctangent and hypotenuse calculation. Please set this value in this version.
#define RES_OUT **- undefined**	Selecting output diagnosis result to the console or not? - If defined, output diagnosis result to the console.
#define TMR_CHK **- defined**	Selecting output performance evaluation result to the console or not? - If defined, output performance evaluation result to the console.
#define HW_DIAG_THRE **- Default value = 0.05f**	Define threshold as affordable error value when permanent fault diagnosis executes. If this value set smaller, error detection ratio may improve but the probability of error occurrence due to calculation error may become larger.
#define SW_DIAG_THRE **- Default value = 0.05f**	Define threshold as affordable error value when transient fault diagnosis executes. If this value set smaller, error detection ratio may improve but the probability of error occurrence due to calculation error may become larger.
#define BT_DIAG_THRE **- Default value = 0.05f**	Define threshold as affordable error value when executing both of permanent and transient fault diagnosis. If this value set smaller, error detection ratio may improve but the probability of error occurrence due to calculation error may become larger.

2.9 Data Structures

This section details the data structures that are used with the functions of this example. In this project, those data structures are located in *r_tfu_diag_rx_if.h and r_tfu_diag_rx_private.h* as the prototype declaration.

```
/* Diagnostic mode */
typedef enum
{
    DIAG_HW_ERR = 0x1,  /* Hard error detection (Comparison of table) */
    DIAG_SW_ERR = 0x2,  /* Soft error detection (Comparison of CPU calculation)
*/
    DIAG_BT_ERR = 0x3,  /* Both (hard & soft error detection) */
} DiagMode;
```

```
/* Diagnosis configuration */
typedef struct
{
    uint32_t start; /* Start point of input data */
    uint32_t end;   /* End point of input data */
    DiagMode mode;  /* Diagnosis mode */
    float thresh;   /* Threshold value of deviation (relative error) */
} DiagConf;
```

```
/* Diagnosis result */
typedef struct
{
    uint32_t h_point;    /* Hard error detection point */
    uint32_t s_point;    /* Soft error detection point */
    DiagMode knd;        /* Kind of detected error */
} DiagRes;
```

```
/* LUT sine and cosine diag table type */
typedef struct
{
    const float in;     /* input data */
    const float out[2]; /* expectation value, 0:sine, 1:cosine */
} LUTSinType;
```

```
/* LUT arctan and hypot diag table type */
typedef struct
{
    const float in[2];  /* input data, 0:x, 1:y */
    const float out[2]; /* expectation value, 0:arctan, 1:hypot */
} LUTAtanType;
```

2.10 Return Values

This section describes return values of the functions of this example. This return value is located in *r_tfu_diag_rx_if.h* as the prototype declarations.

```
/* TFU diagnostic software return value */
typedef enum
{
    TFU_ERR_DET = -3,     /* Detected diagnosis error */
    TFU_ERR_PARAM = -2,   /* Parameter error */
    TFU_ERR = -1,         /* General error */
    TFU_OK = 0,
} tfu_return_t;
```

2.11 Code Size

The sizes of ROM (code and constants), RAM (global data) and maximum stack usage associated with this example are listed below. The size listed extracts only TFU fault diagnosis part whose source code corresponds to “r_tfu_diag_rx.c”.

The ROM and RAM sizes are determined by the build-time configuration options described in “2.8 Configuration Overview”.

The values in the table below are confirmed under the following conditions.

Source code Revision: r_tfu_diag_rx rev1.00

Compiler Version: Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00

(The options of “-lang = c99”, “-optimize = 0” and “-tfu = intrinsic” are added to the default settings of the integrated development environment.)

GCC for Renesas RX 4.08.04.201902-SP1=GNURX

(The options of “-std = gnu99” and “MTFU = intrinsic” are added to the default settings of the integrated development environment.)

IAR C/C++ Compiler for Renesas RX version 4.12.1

(The options of “no optimization” and “TFU intrinsics” are added to the default settings of the integrated development environment.)

Configuration Options: Default settings.

ROM, RAM and Stack Code Sizes					
Device	Category	File	Memory Used		
			Renesas Compiler	GCC	IAR Compiler
RX72M	ROM	r_tfu_diag_rx.c	**8064 bytes**	**8281 bytes**	**7206 bytes**
	RAM	r_tfu_diag_rx.c	**7184 bytes**	**7184 bytes**	**7184 bytes**
	STACK	r_tfu_diag_rx.c	**100 bytes**	-	**64 bytes**

3. Specification of This Example

3.1 Execution Sequence

Execution of this example needs a RX72M RSK+ board[1] or RX72T RSK board[2].

The outline of the execution is following.

- Write the project execution code to a code Flash of in the RX72M RSK+ board or RX72T RSK board (hereafter boards).
- Power on the board.
- Run the execution code.
- If the execution finished without error, the message showed by Figure 3.1 appears in the "Renesas Debug Console" or "Terminal I/O which are equipped with the e2 studio or EWRX respectively. To output this message, "RES_OUT" macro should be defined.
- If a fault detected during the operation, the message showed by Figure 3.2 appears in the corresponding console. To output this message, "RES_OUT" macro should be defined.
- The operation cycles and time show the console. To output this message, "TMR_CHK" macro should be defined.

[1] Product name is Renesas Starter Kit+ for RX72M [2]. [2] Product name is Renesas Starter Kit for RX72T [3].

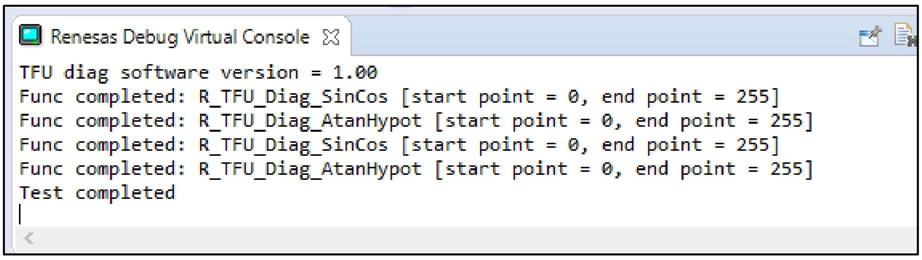

Figure 3.1 Result Message (No fault detection)

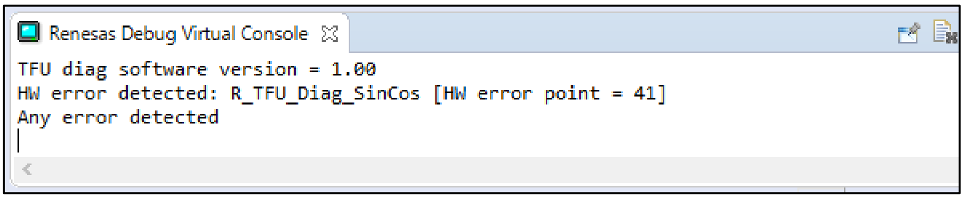

Figure 3.2 Result Message (Fault detection)

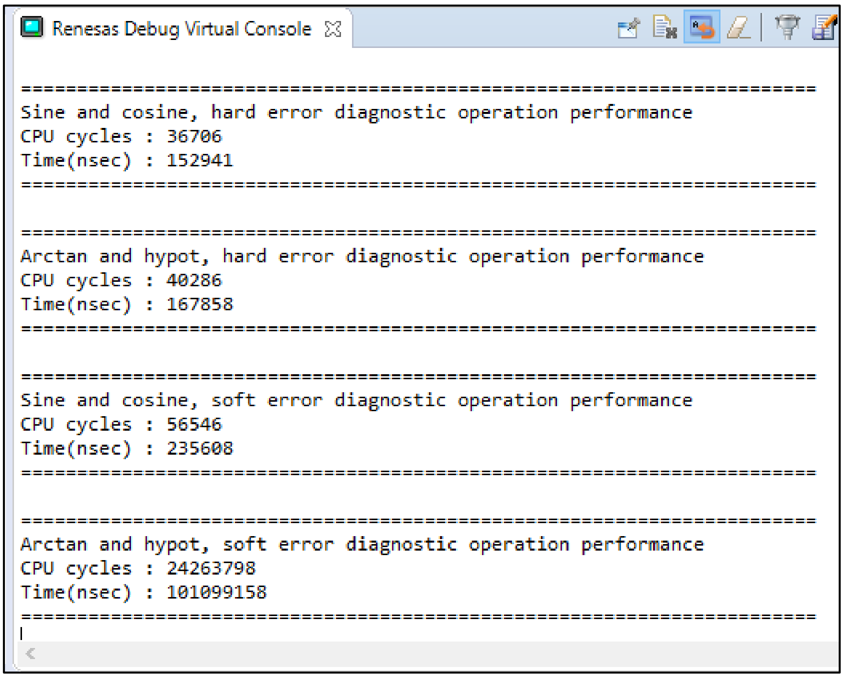

Figure 3.3 Result Message (Performance measurement)

3.2 Diagnosis Methods Overview

Explain overview of the diagnostic method applicable to permanent and transient fault. Calculation using TFU is executed via a TFU intrinsic function which is built in each compiler. The TFU intrinsic function supports the following two kinds of operation.

sine and cosine simultaneous operation: __sincosf in CC-RX, EWRX, __builtin_rx_sincosf in GCC

arctangent and hypotenuse simultaneous operation: __atan2hypotf in CC-RX, EWRX, __builtin_rx_atan2hypotf in GCC

In this section, explain using only __sincosf and __atan2hypotf as the TFU intrinsic function.

3.2.1 Permanent Fault

- Sine and cosine

(1) Allocate look-up table which stores input θ, sine expectation sin(θ) and cosine expectation cos(θ). The size is 3072 bytes (256 x 3 x float size).

(2) Execute __sincosf whose input θ is loaded from look-up table.

(3) Using the input θ inputted from __sincosf, TFU calculates sine and cosine simultaneously. And then, TFU returns those results a1 = sin(θ) and a2 = cos(θ) to __sincosf.

(4) TFU diagnosis software compares TFU calculation results which are a1 and a2 with look-up table expectation values which are a1' and a2'. The difference is defined as the relative error |a1 – a1'| / |a1'| or |a2 – a2'| / |a2'|. The comparison is also considering calculation error. If the comparison results do not match even if considering the calculation error, it is judged by detecting a permanent error.

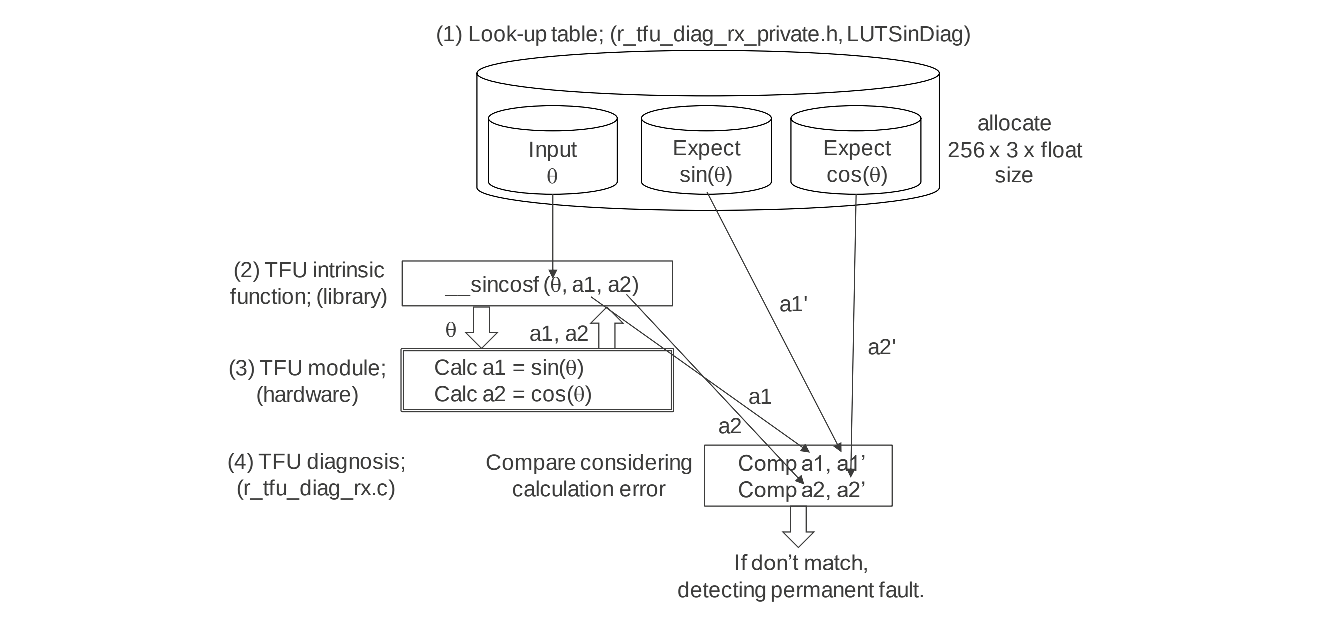

Figure 3.4 Permanent Fault (sine & cosine)

- Arctangent and hypotenuse

(1) Allocate look-up table which stores base input x, height input y, arctangent expectation arctan(y/x) and hypotenuse expectation $(x^2 + y^2)^{1/2}$. The size is 4096 bytes (256 x 4 x float size).

(2) Execute __atan2hypotf whose base input x and height input y are loaded from look-up table.

(3) Using the base input x and height input y inputted from __atan2hypotf, TFU calculates arctangent and hypotenuse simultaneously. And then, TFU returns those results a1 = arctan(y/x) and a2 = $(x^2 + y^2)^{1/2}$ to __ atan2hypotf.

(4) TFU diagnosis software compares TFU calculation results which are a1 and a2 with look-up table expectation values which are a1' and a2'. The difference is defined as the relative error |a1 – a1'| / |a1'| or |a2 – a2'| / |a2'|. The comparison is also considering calculation error. If the comparison results do not match even if considering the error, it is judged by detecting a permanent error.

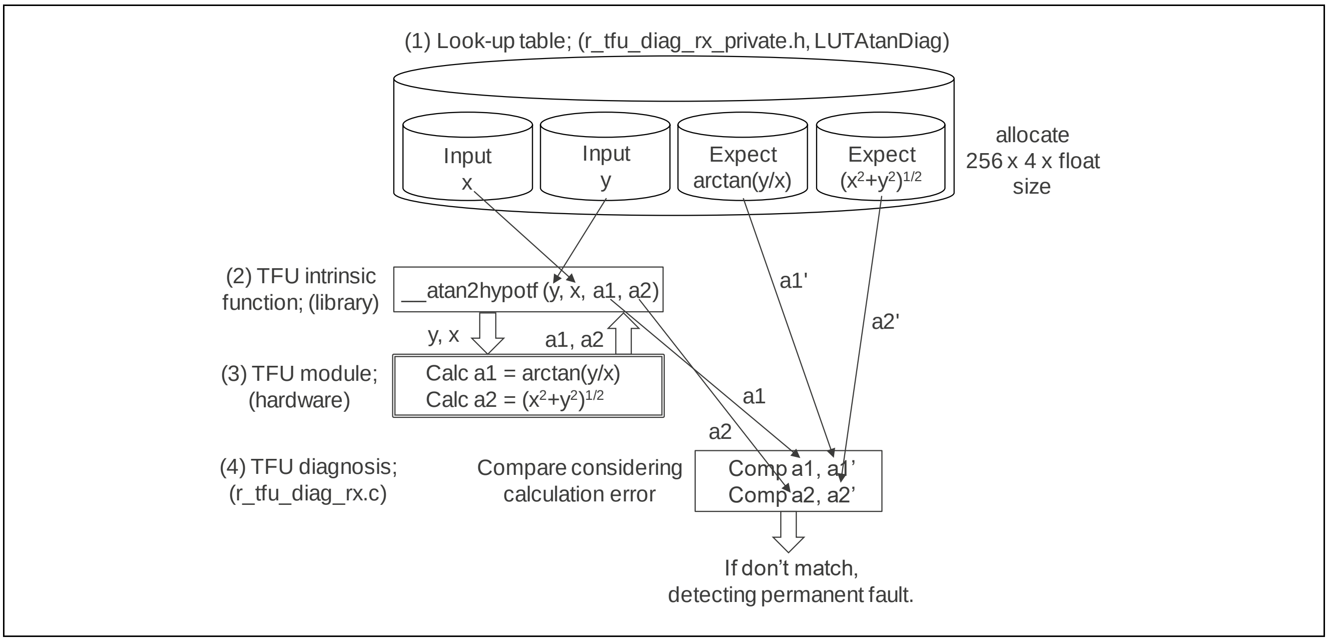

Figure 3.5 Permanent Fault (arctangent & hypotenuse)

3.2.2 Transient Fault

- Sine and cosine

(1) Allocate look-up table which stores input θ, sine expectation sin(θ) and cosine expectation cos(θ). The size is 3072 bytes (256 x 3 x float size). However, sine expectation sin(θ) and cosine expectation cos(θ) are not used for transient fault diagnosis.

(2) Execute __sincosf whose input θ is loaded from look-up table.

(3) Using the input θ inputted from __sincosf, TFU calculates sine and cosine simultaneously. And then, TFU returns those results a1 = sin(θ) and a2 = cos(θ) to __sincosf.

(4) To create results by another calculation method as the expectation values, calculating sine and cosine using mathematic float type standard library sinf and cosf respectively.

(5) TFU diagnosis software compares TFU calculation results which are a1 and a2 with the standard library calculation results which are a1’ = sinf(θ) and a2’ = cosf(θ). The difference is defined as the relative error |a1 – a1’| / |a1’| or |a2 – a2’| / |a2’|. The comparison is also considering calculation error. If the comparison results do not match even if considering the error, it is judged by detecting a transient error.

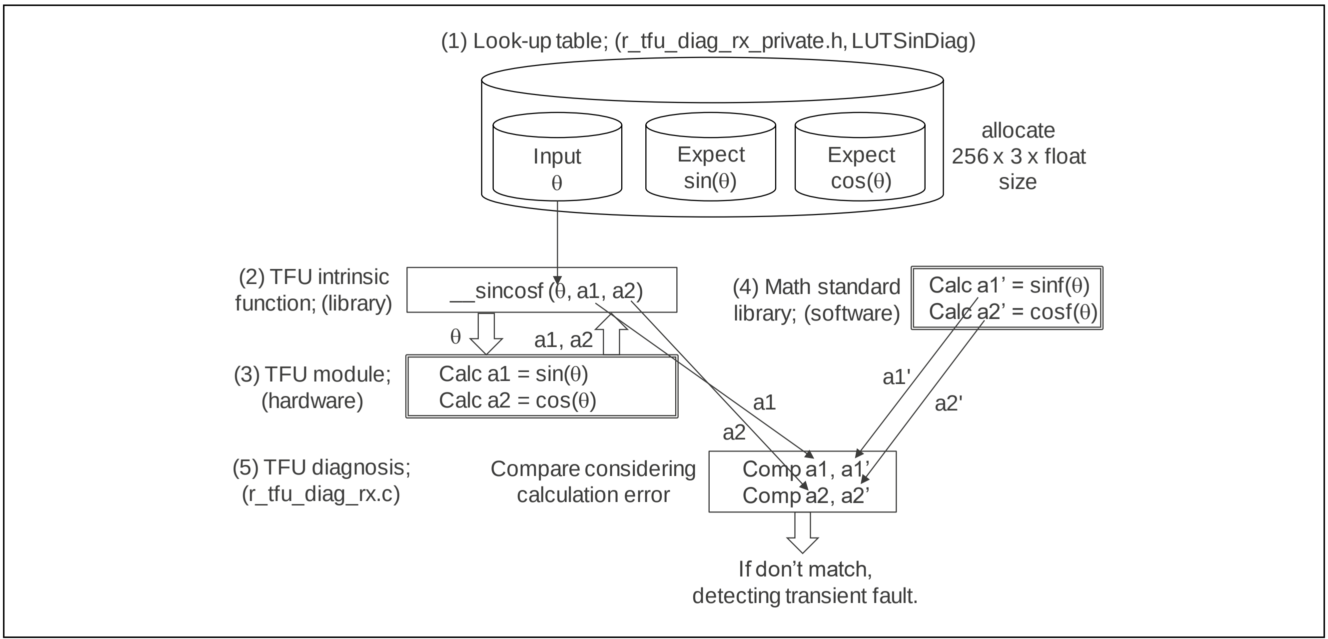

Figure 3.6 Transient Fault (sine & cosine)

- Arctangent and hypotenuse

(1) Allocate look-up table which stores base input x, height input y, arctangent expectation arctan(y/x) and hypotenuse expectation $(x^2 + y^2)^{1/2}$. The size is 4096 bytes (256 x 4 x float size). However, arctangent expectation artan2f(y, x) and hypotenuse expectation $(x^2 + y^2)^{1/2}$ are not used for transient fault diagnosis.

(2) Execute __atan2hypotf whose base input x and height input y are loaded from look-up table.

(3) Using the base input x and height input y inputted from __atan2hypotf, TFU calculates arctangent and hypotenuse simultaneously. And then, TFU returns those results a1 = arctan(y/x) and a2 = $(x^2 + y^2)^{1/2}$ to __ atan2hypotf.

(4) To create results by another calculation method, calculating arctangent and hypotenuse using mathematic float type standard library atan2f and hypotf respectively.

(5) TFU diagnosis software compares TFU calculation results which are a1 and a2 with library calculation results which are a1’ = artan2f(y, x) and a2’ = $(x^2 + y^2)^{1/2}$. The difference is defined as the relative error |a1 – a1’| / |a1’| or |a2 – a2’| / |a2’|. The comparison is also considering calculation error. If the comparison results do not match even if considering the error, it is judged by detecting a transient error.

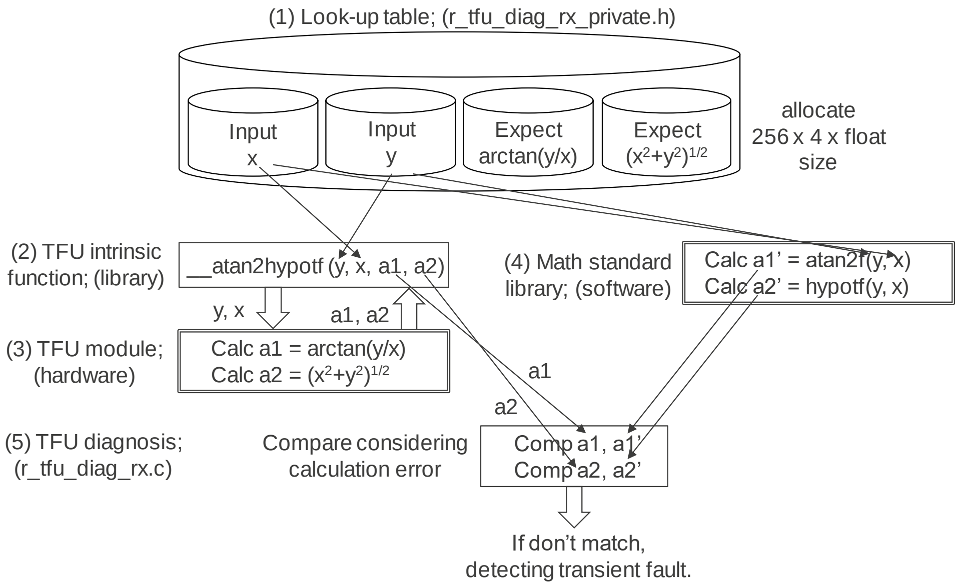

Figure 3.7 Transient Fault (arctangent & hypotenuse)

3.3 Operation Flow Example

In this section, describes the software operation flow of this sample.
Figure 3.8 shows the operation flow of initial setting of TFU and diagnosis software. Figure 3.9 shows the operation flow of diagnostics for permanent fault and transient fault.

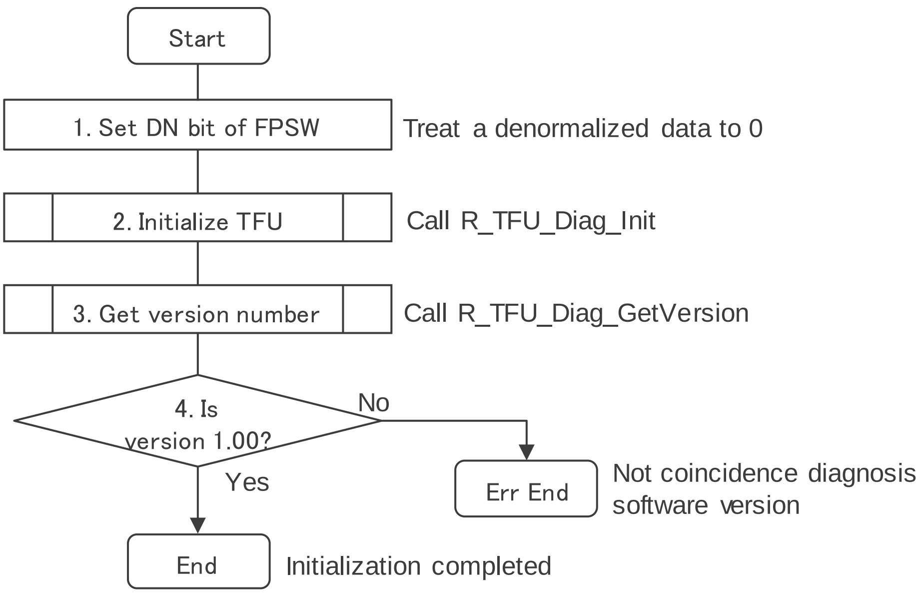

Figure 3.8 Initialization Flow Example

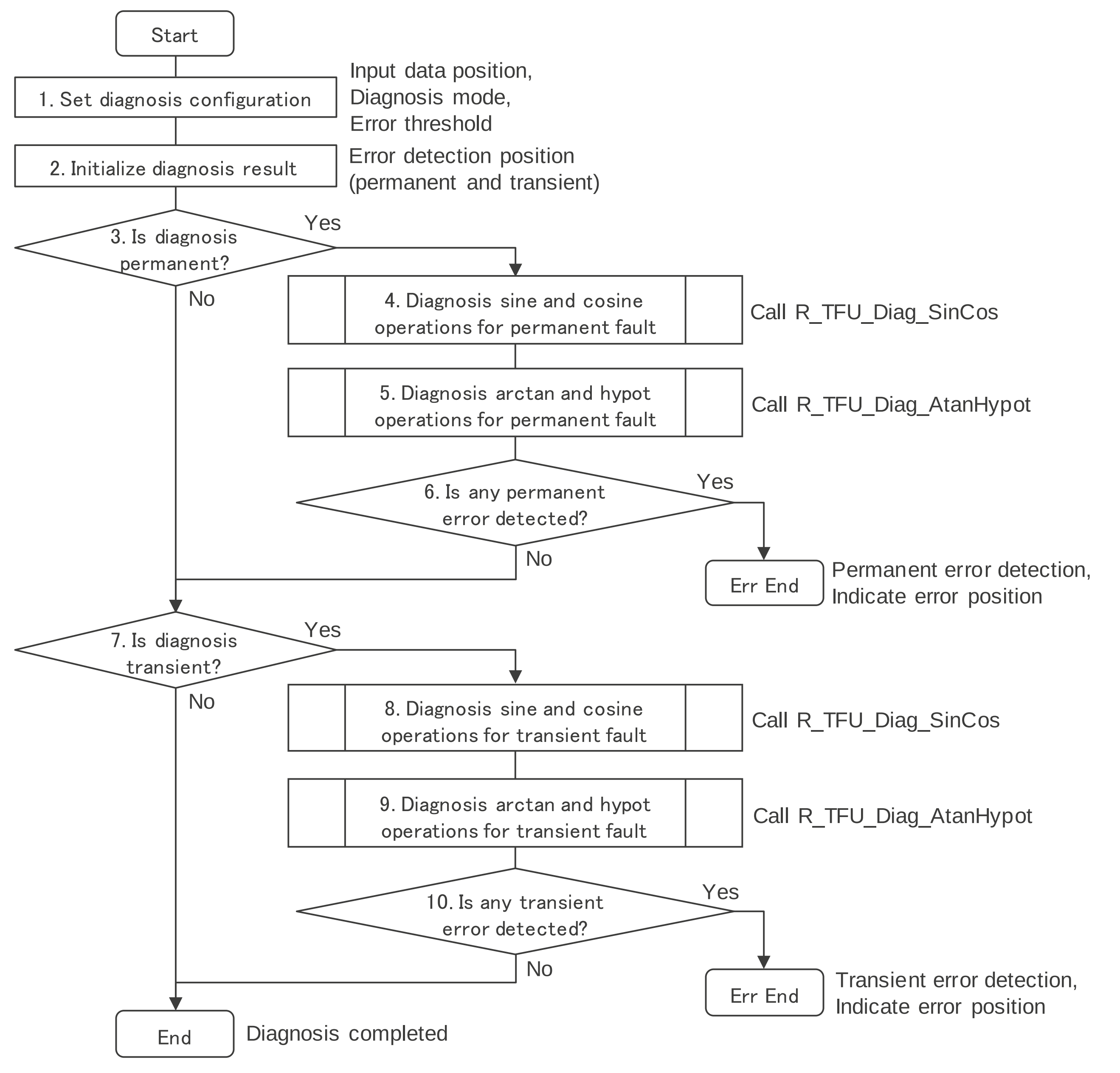

Figure 3.9 Diagnosis Flow Example

3.4 Performance of Diagnosis Operation (Measurement Example)

Table 3.1 and Table 3.2 show the diagnosis operation time of RX72M and RX72T respectively as the performance reference. Those operation times are measured by the execution interval for each kind of operation.

Please keep your mind those results are depend on the condition and environment.

Table 3.1 RX72M operation time

- RX72M Measurement conditions

Board	CPU frequency	Code area	Data area	Endian	Compiler setting
RX72M RSK+ board (RTK5572MNHC00000BJ)	240MHz	Internal ROM Read: 1cycles, if hit cache. 2-3 cycles, if no hit cache.	Internal RAM Read/Write: 1cycle	Little	Safety purpose setting More detail, refer to Table.5.1.

- Operation performances (μs unit)

Kind of fault	Operation	CC-RX V3.01.00	EWRX V4.12.1	GCC V4.08.04
Permanent	Sine and cosine	153	114	193
	Arctangent and hypotenuse	168	123	205
Transient	Sine and cosine	236	237	379
	Arctangent and hypotenuse	101,714	330,770	101,485

Table 3.2 RX72T operation time

- RX72T Measurement conditions

Board	CPU frequency	Code area	Data area	Endian	Compiler setting
RX72T RSK board (RTK5572TKCS00000BE)	200MHz	Internal ROM Read: 1cycles, if hit cache. 2-3 cycles, if no hit cache.	Internal RAM Read/Write: 1cycle	Little	Safety purpose setting More detail, refer to Table.5.1.

- Operation performances (μs unit)

Kind of fault	Operation	CC-RX V3.01.00	EWRX V4.12.1	GCC V4.08.04
Permanent	Sine and cosine	181	135	230
	Arctangent and hypotenuse	199	145	244
Transient	Sine and cosine	277	269	451
	Arctangent and hypotenuse	100,336	284,940	100,417

4. API Functions

TFU diagnosis software (r_tfu_diag_rx.c) has four API functions. Describing their specification in this section.

4.1 R_TFU_Diag_GetVersion ()

This function returns the version number of TFU diagnosis software.

Format

uint32_t R_TFU_Diag_GetVersion(void);

Parameters

None

Return Values

TFU_DIAG_VERSION_MAJOR (upper 16bit): Major version number
TFU_DIAG_VERSION_MINOR (lower 16bit): Minor version number

Properties

Prototyped in “r_tfu_diag_rx_if.h”.

Description

Return major and minor version number of TFU diagnosis software.

This software version number is “1.00”.

- Upper 16 bit indicates major version number
  TFU_DIAG_VERSION_MAJOR: current value = H’1.

- Lower 16 bit indicates minor version number
  TFU_DIAG_VERSION_MINOR: current value = H’00.

Reentrant

Function is reentrant.

Example

Example showing this function being used.

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

   uint32_t version;

   version = R_TFU_Diag_GetVersion();

   printf("TFU diag software version = %d.%02d\n", ver >> 16u, ver & 0xFFFF);
```

Special Notes

Return value itself will be change depend on the diagnosis software version.

4.2 R_TFU_Diag_Init ()

This function initializes TFU.

Format

void R_TFU_Diag_Init(void);

Parameters

None

Return Values

None

Properties

Prototyped in “r_tfu_diag_rx_if.h ".

Description

This function initializes TFU by calling __init_tfu() intrinsic function.

TFU initialization is relevant, only if CC-RX or GCC compiler is used.

No operation executes on this function, If EWRX compiler is used.

Reentrant

Function is not reentrant.

Example

Example showing this function being used.

```
#include <math.h>
#if defined(__CCRX__)
#include <mathf.h>
#endif
#include <stdio.h>
#include "r_tfu_diag_rx_if.h"

   tfu_return_t ret;
   DiagConf conf;
   DiagRes  res;

   /* Initialize TFU */
   R_TFU_Diag_Init();

   printf("TFU is initialized\n");

   /* ==== Hard error diagnosis ==== */
   /* Dignosis sine and cosine operations for hard error */
   conf.start = 0;
   conf.end = (LUTSinSize - 1);
   conf.mode = DIAG_HW_ERR;
   conf.thresh = HW_DIAG_THRE;
   res.h_point = conf.start;
   res.s_point = conf.start;

   ret = R_TFU_Diag_SinCos(&conf, &res);
   if (TFU_OK == ret)
   {
       printf("Func completed: R_TFU_Diag_SinCos [start point = %d, end point
= %d] \n", conf.start, conf.end);
   }
   else if (TFU_ERR_DET == ret)
   {
```

```
        printf("HW error detected: R_TFU_Diag_SinCos [HW error point = %d] \n",
res.h_point);
        goto err_end;
    }
    else
    {
        printf("unknown error: R_TFU_Diag_SinCos [error code = %d] \n", ret);
        goto err_end;
    }

    /* ==== Soft error diagnosis ==== */
    /* Dignosis arctan and hypot operations for soft error */
    conf.start = 0;
    conf.end = (LUTSinSize - 1);
    conf.mode = DIAG_SW_ERR;
    conf.thresh = SW_DIAG_THRE;
    res.h_point = conf.start;
    res.s_point = conf.start;

    ret = R_TFU_Diag_AtanHypot(&conf, &res);
    if (TFU_OK == ret)
    {
        printf("Func completed: R_TFU_Diag_AtanHypot [start point = %d, end point
= %d] \n", conf.start, conf.end);
    }
    else if (TFU_ERR_DET == ret)
    {
        printf("SW error detected: R_TFU_Diag_AtanHypot [SW error point = %d]
\n", res.s_point);
        goto err_end;
    }
    else
    {
        printf("unknown error: R_TFU_Diag_AtanHypot [error code = %d] \n", ret);
        goto err_end;
    }

    printf("Test completed\n");
    while(1);

err_end:
    printf("Any error detected\n");
    while(1);
```

Special Notes

No operation executes on this function, If EWRX compiler is used.

4.3 R_TFU_Diag_SinCos ()

This function does sine and cosine diagnostic operations.

Format

tfu_return_t R_TFU_Diag_SinCos(DiagConf *conf, DiagRes *res);

Parameters

conf - Diagnostic configuration.

res - Diagnostic result.

Return Values

TFU_OK: Processing completed successfully
TFU_ERR_PARAM: Parameter error
TFU_ERR_DET: Detected any fault

Properties

Prototyped in “r_tfu_diag_rx_if.h".

Description

This function does sine and cosine diagnostic operations. The following operations are executed.

- Verify input configuration parameter
  If table index or diagnosis mode is illegal, returns error(=TFU_ERR_PARAM).

- Load input θ from look-up table.

- Execute sine and cosine calculation using TFU.
  This is executed by calling TFU intrinsic function. When CC-RX or EWRX compiler is used, the TFU intrinsic function is __sincosf(). When GCC compiler is used, it is __builtin_rx_sincosf().

- If the diagnosis mode is permanent fault, executing permanent fault detect operations.
  Load sine and cosine expectation values from look-up table.
  Evaluate the relative error between the calculation result of TFU to the expectation value with numerical calculation error. If the relative error exceeds the threshold of numerical calculation error, judging a permanent fault is detected and returns the detection point of permanent fault.

- If the diagnosis mode is transient fault, executing transient fault detect operations.
  Do the calculation of sine and cosine using mathematic float type standard library sinf and cosf respectively.
  Evaluate the relative error between the calculation result of TFU to the calculation value using sinf and cosf with numerical calculation error. If the relative error exceeds the threshold of numerical calculation error, judging a permanent fault is detected and returns the detection point of permanent fault.

Reentrant

Function is not reentrant.

Example

Example is same as “4.2 R_TFU_Diag_Init”.

Special Notes

This function needs the mathematic float type standard library build in each compiler.

4.4 R_TFU_Diag_AtanHypot ()

This function does arctangent and hypotenuse diagnostic operations.

Format

tfu_return_t R_TFU_Diag_AtanHypot(DiagConf *conf, DiagRes *res);

Parameters

conf - Diagnostic configuration.

res - Diagnostic result.

Return Values

TFU_OK: Processing completed successfully
TFU_ERR_PARAM: Parameter error
TFU_ERR_DET: Detected diagnosis error

Properties

Prototyped in “r_tfu_diag_rx_if.h".

Description

This function does arctangent and hypotenuse diagnostic operations. The following operations are executed.

- Verify input configuration parameter
  If table index or diagnosis mode is illegal, returns error(=TFU_ERR_PARAM).

- Load input base x and height y from look-up table.

- Execute arctangent and hypotenuse calculation using TFU.
  This is executed by calling TFU intrinsic function. When CC-RX or EWRX compiler is used, the TFU intrinsic function is __atan2hypotf(). When GCC compiler is used, it is __builtin_rx_atan2hypotf().

- If the diagnosis mode is permanent fault, executing permanent fault detect operations,
  Load arctangent and hypotenuse expectation values from look-up table.
  Evaluate the relative error between the calculation result of TFU to the expectation value with numerical calculation error. If the relative error exceeds the threshold of numerical calculation error, judging a permanent fault is detected and returns the detection point of permanent fault.

- If the diagnosis mode is transient fault, executing transient fault detect operations,
  Do the calculation of arctangent and hypotenuse using mathematic float type standard library atan2f and hypotf respectively.
  Evaluate the relative error between the calculation result of TFU to the calculation value using atan2f and hypotf with numerical calculation error. If the relative error exceeds the threshold of numerical calculation error, judging a permanent fault is detected and returns the detection point of permanent fault.

Reentrant

Function is not reentrant.

Example

Example is same as “4.2 R_TFU_Diag_Init”.

Special Notes

This function needs the mathematic float type standard library build in each compiler.

5. Appendices

5.1 Confirmed Operation Environment

This section describes confirmed operation environment of this example.

Table 5.1 Confirmed Operation Environment

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version V7.5.0 IAR Embedded Workbench for Renesas RX 4.12.1
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99, -optimize = 0 (level 0: no optimization), -tfu = Intrinsic (use trigonometric function)
	GCC for Renesas RX 4.08.04.201902-SP1-GNURX Compiler option: The following option is added to the default settings of the integrated development environment. -std = gnu99, MTFU = intrinsic
	IAR C/C++ Compiler for Renesas RX version 4.12.1 Compiler option: The following option is added to the default settings of the integrated development environment. no optimization, TFU intrinsics
Endian	Big endian/little endian
Revision of the module	Rev.1.00
Board used	Renesas Starter Kit+ for RX72M Renesas Starter Kit+ for RX72T

6. Provided Modules

The module provided can be downloaded from the Renesas Electronics website.

7. Reference Documents

User's Manual: Hardware

RX72M Group User's Manual: Hardware Rev.1.00 (R01UH0804EJ)
RX72T Group User's Manual: Hardware Rev.1.00 (R01UH0803EJ)
The latest versions can be downloaded from the Renesas Electronics website.

Technical Update/Technical News

The latest information can be downloaded from the Renesas Electronics website.

User's Manual: Development Tools

RX Family CC-RX Compiler User's Manual Rev.1.08 (R20UT3248EJ)
The latest information can be downloaded from the Renesas Electronics website.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Aug 31, 2019	—	First edition issued.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.