RX Family

三角関数演算器(TFU)故障診断例

要旨

本アプリケーションノートは、三角関数演算器（TFU）の故障診断例を説明しています。

動作確認デバイス

以下のデバイスがこのソフトウェアでサポートされています。

- RX72M グループ
- RX72T グループ

本アプリケーションノートを他のルネサスマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

目次

1. 概要

本アプリケーションノートは、三角関数演算器（TFU）の故障診断例を説明しています。また、永久故障と一時故障の双方の故障診断に対応しています。

1.1 TFU 故障診断例

このサンプルプログラムは関数として、プロジェクトに組み込んで使用します。そして、故障診断の応用例として使用できます。

本ソフトウェアは機能安全関連の認証（産業向け機能安全規格を含め）を取得してません。お客様が機能安全関連の認証を取得する場合、このサンプルプログラムを参照し TFU の診断処理を実装し、お客様のシステムとしての認証を取得する必要があります。

1.2 関連ドキュメント

[1] RX ファミリ ボードサポートパッケージモジュール Firmware Integration Technology, Rev.1.16, Document No.R01AN1685JJ0520, Apr 08, 2019
[2] Renesas Starter Kit+ for RX72M CPU ボード（試作版），ユーザーズマニュアル、Rev.1.01, Document No. R20UT4383EG0101, Feb 28, 2019
[3] Renesas Starter Kit+ for RX72T CPU ボード（試作版），ユーザーズマニュアル、Rev.1.03, Document No. R20UT4279JG0103, Nov 28, 2018

1.3 ハードウェアの構成

このサンプルプログラムは TFU の診断で CPU、ROM、RAM を使用します。なお、処理時間の測定のみに MTU3 を使用します。

詳細に関しては「RX72M/72T グループユーザーズマニュアル ハードウェア編」を参照ください。

1.4 ソフトウェアの構成

このサンプルプログラムは TFU の組み込み関数を使用して、TFU によるハードウェア演算を実行します。一時故障の診断では、演算結果の期待値生成でコンパイラに組み込まれている標準の数学関数を使用します。処理時間の測定が必要な場合、処理時間測定用のソフトウェアを使用できます。故障検出位置や処理時間等の処理結果は開発統合環境のコンソール [1] に表示されます。

[1] e2 studio 環境では”Renesas Debug Virtual Console”、EWRX 環境では”Terminal I/O”になります。

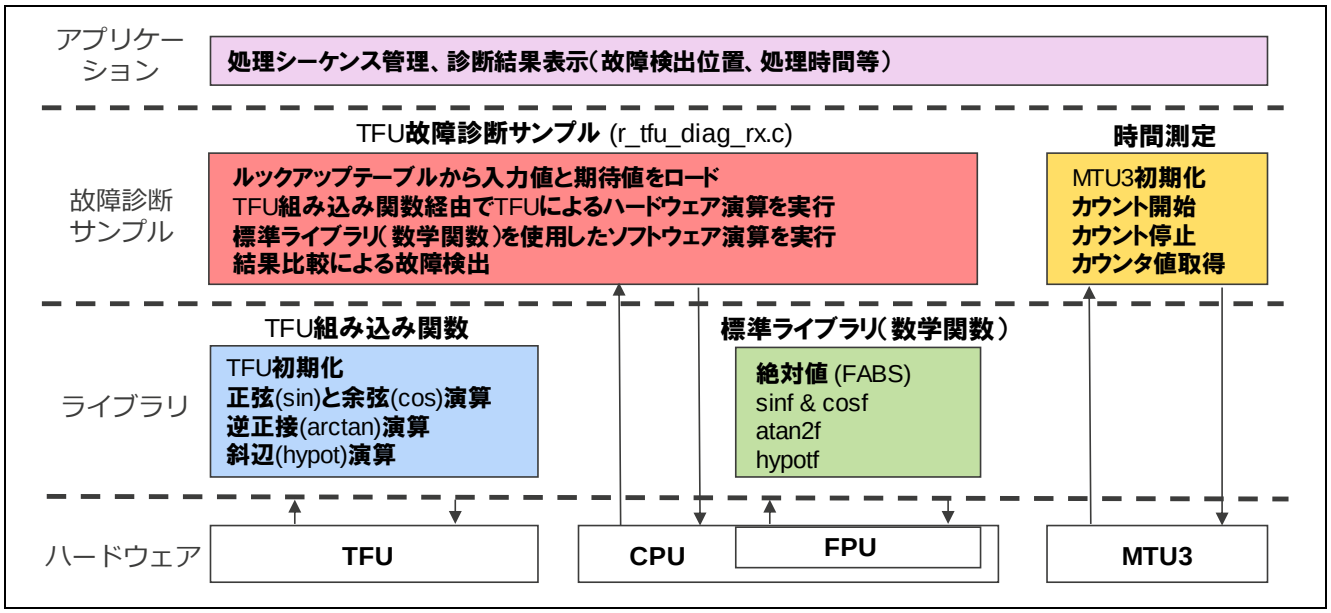

図 1.1 ソフトウェア構成

1.5 ファイル構成

このサンプルプログラムのソースコードは、“src”, “r_config”, “r_tfu_diag_rx”, “r_bsp”の各フォルダとそれ以下の階層のフォルダに格納しています。図 1.2 にソースとヘッダのファイル構成を示します。“src”フォルダはサンプルアプリケーション、コンソール出力処理、時間測定処理のファイルを格納しています。
“r_tfu_diag_rx”フォルダは TFU 故障診断サンプル（以降、TFU 診断ソフト）に関連したファイルを格納しています。このサンプルプログラムの初期設定には、ルネサスボードサポートパッケージ（BSP）[1]を使用しています。

```
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
```

図 1.2　ファイル構成

1.6 関数の概要

サンプルアプリケーションとそれ以下の階層の関数を表 1.1 に、TFU 故障診断に関連する API 関数を表 1.2 にそれぞれ示します。

表 1.1　アプリケーション関連

関数	内容
main()	このサンプルプログラムのメイン処理
start_label()	処理の開始をコンソールに表示
end_label()	処理の終了をコンソールに表示
show_result()	処理に要したサイクル数(ICLK)と処理時間をコンソールに表示
mtu3_dev_start()	MTU3のモジュールストップを解除
mtu3_dev_stop()	MTU3 のモジュールストップを設定
Init_timer()	MTU3 の初期設定
start_eval()	MTU3のカウントアップ開始
stop_eval()	MTU3 のカウントアップ停止
get_eval_cycle()	現在のMTU3のカウンタ値を取得

表 1.2　API 関数（TFU 故障診断）

関数	内容
R_TFU_Diag_GetVersion()	TFU 診断ソフトウェアのバージョン番号取得
R_TFU_Diag_Init()	TFU の初期設定
R_TFU_Diag_SinCos()	正弦(sin)と余弦(cos)演算による故障診断
R_TFU_Diag_AtanHypot()	逆正接(arctan)と斜辺(hypot)演算による故障診断

2. 関数情報

このサンプルプログラムは下記の条件で動作を確認しています。

2.1 ハードウェアの要件

ご使用になる MCU が以下の機能をサポートしている必要があります。

- TFU
- MTU3（オプション）

2.2 ハードウェアリソース要件

サンプルプログラムが必要とする周辺回路のハードウェアについて説明します。特に明記しない限り、周辺回路はドライバで制御します。ユーザアプリケーションから直接制御し、使用することはできません。

2.2.1 TFU

故障診断の対象として TFU を使用します。サンプルプログラムが動作中、他のタスクは TFU にアクセスし使用することはできません。

2.2.2 MTU3 チャネル（オプション）

サンプルプログラムは MTU3 の CH1 と CH2 をカスケード接続し使用します。MTU3 を使用した処理時間の測定中、他のタスクは MTU3 の CH1 と CH2 を使用できません。

2.3 ソフトウェア要件

サンプルプログラムは以下のパッケージ（FIT モジュール）に依存しています。

- r_bsp

2.4 制限事項

このソフトウェアは機能安全の認証（産業向け機能安全規格を含む）を取得していません。お客様が機能安全関連の認証を取得する場合、このサンプルプログラムを参照し TFU の診断処理を実装し、お客様のシステムとしての認証を取得する必要があります。

2.5 サポートされているツールチェイン

サンプルプログラムは「5.1 動作確認環境」に示すツールチェーンで動作確認を行っています。

2.6 ヘッダファイル

個々の関数はサンプルプログラムのプロジェクトに含まれる以下のヘッダファイルのいずれか 1 種、または、複数種をインクルードすることで呼び出すことができます。

show_label.h, tmr_if.h, r_tfu_diag_rx_config.h, r_tfu_diag_rx_if.h, r_tfu_diag_rx_private.h をインクルードする。

2.7 整数型

このプロジェクトでは ANSI C99 を使用し、整数型は *stdint.h* で定義しています。

2.8 コンパイル時の設定

サンプルプログラムの構成設定は *r_tfu_diag_rx_config.h* と *tfu_test.c* で行います。オプション名と設定内容を以下の表に記載します。

構成設定	
#define RX_DEVICE_TYPE #define DEVICE_RX72M (0) #define DEVICE_RX72T (1) - Default value = 0	対象デバイスを指定します。 • 対象デバイスがRX72Mの場合、0 を設定してください。 • 対象デバイスがRX72Tの場合、1 を設定してください。 このバージョンでは他のデバイスは対応していません。
#define LUTSinSize - Default value = 256	正弦と余弦演算に使用するルックアップテーブルのサイズを設定します。 このバージョンではデフォルト値を設定してください。
#define LUTAtanSize - Default value = 256	逆正接と斜辺演算に使用するルックアップテーブルのサイズを設定します。 このバージョンではデフォルト値を設定してください。
#define RES_OUT - undefined	故障診断結果をコンソールに出力するか、しないかを選択します。 • 定義した場合、故障診断結果をコンソールに出力します。
#define TMR_CHK - defined	処理時間の測定結果をコンソールに出力するか、しないかを選択します。 定義した場合、処理時間測定結果をコンソールに出力します。
#define HW_DIAG_THRE - Default value = 0.05f	永久故障診断結果の期待値との比較時、演算による相対誤差の閾値を設定します。この値を小さくするほど、故障検出率は向上しますが、演算誤差による誤検出が発生する確率は高くなります。
#define SW_DIAG_THRE - Default value = 0.05f	一時故障診断結果の期待値との比較時、演算による相対誤差の閾値を設定します。この値を小さくするほど、故障検出率は向上しますが、演算誤差による誤検出が発生する確率は高くなります。
#define BT_DIAG_THRE - Default value = 0.05f	永久故障診断と一時故障診断を両方行う場合、故障診断結果の期待値との比較時、演算による相対誤差の閾値を設定します。この値を小さくするほど、故障検出率は向上しますが、演算誤差による誤検出が発生する確率は高くなります。

2.9 データ構造

サンプルプログラムの関数で使用しているデータ構造について説明します。これらの構造体は *r_tfu_diag_rx_if.h* と *r_tfu_diag_rx_private.h* に関数の型宣言と共に定義しています。

```
/* 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 戻り値

サンプルプログラムの関数の戻り値を示します。これらの戻り値は r_tfu_diag_rx_if.h にプロトタイプ宣言と共に定義しています。

```
/* 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 コードサイズ

サンプルプログラムの ROM (コードおよび定数) 、RAM (グローバルデータ) 、使用する最大のスタックのサイズを以下に示します。なお、サイズは”r_tfu_diag_rx.c”に対応する TFU 故障診断部のみを抽出したサイズです。

ROM と RAM のサイズは、ビルド時の「2.8 コンパイル時の設定」のコンフィギュレーションオプションによって決まります。

下表の値は下記条件で確認しています。

プログラムリビジョン: r_tfu_diag_rx.c rev1.00

コンパイラバージョン: Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00
(統合開発環境のデフォルト設定に“-lang = c99”, “-optimize = 0”,
“-tfu = intrinsic”オプションを追加)
GCC for Renesas RX 4.08.04.201902-SP1=GNURX
(統合開発環境のデフォルト設定に”-std=gnu99”と“MTFU = intrinsic”
オプションを追加)
IAR C/C++ Compiler for Renesas RX version 4.12.1
(統合開発環境のデフォルト設定に”no optimization”と“TFU intrinsics”
オプションを追加)

コンフィグレーションオプション: デフォルト設定

ROM、RAM およびスタックのコードサイズ

デバイス	分類	ファイル	使用メモリ		
			Renesas Compiler	GCC	IAR Compiler
RX72M	ROM	r_tfu_diag_rx.c	**8064 バイト**	**8281 バイト**	**7206 バイト**
	RAM	r_tfu_diag_rx.c	**7184 バイト**	**7184 バイト**	**7184** バイト
	STACK	r_tfu_diag_rx.c	**100 バイト**	**-**	**64** バイト

3. サンプルプログラムの仕様

3.1 実行手順

サンプルプログラムの実行には RX72M RSK+ボード [1]、または、RX72T RSK ボード [2] が必要になります。以下に実行手順の概要を記載します。

- プロジェクトの実行コードを RX72M RSK+ボード、または、RX72T RSK ボード（以下、ボード）に書き込んでください。
- ボードに電源を投入してください。
- コードを実行してください。
- 実行が故障検出なしに終了した場合、図 3.1 に示されるメッセージが、e2 studio 環境ではルネサスデバッグコンソール、EWRX 環境ではターミナル I/O に表示されます。

  このメッセージの表示には”RES_OUT”マクロを定義する必要があります。
- 実行中に故障を検出した場合、図 3.2 に示されるメッセージがコンソールに表示されます。

  このメッセージの表示には”RES_OUT”マクロを定義する必要があります。
- 処理に要した実行サイクルと時間を表示することができます。

  このメッセージの表示には”TMR_CHK”マクロを定義する必要があります。

[1] 製品名は”Renesas Starter Kit+ for RX72M [2]。[2] 製品名は”Renesas Starter Kit for RX72T [3]。

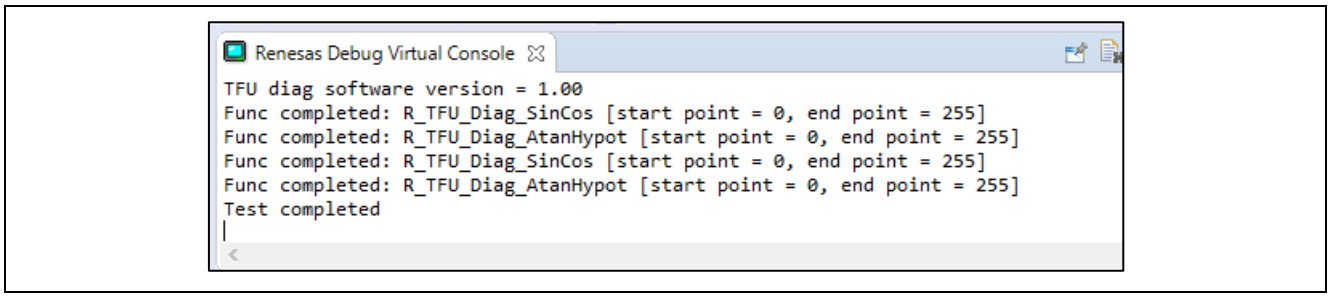

図 3.1 結果メッセージ（故障検出なし）

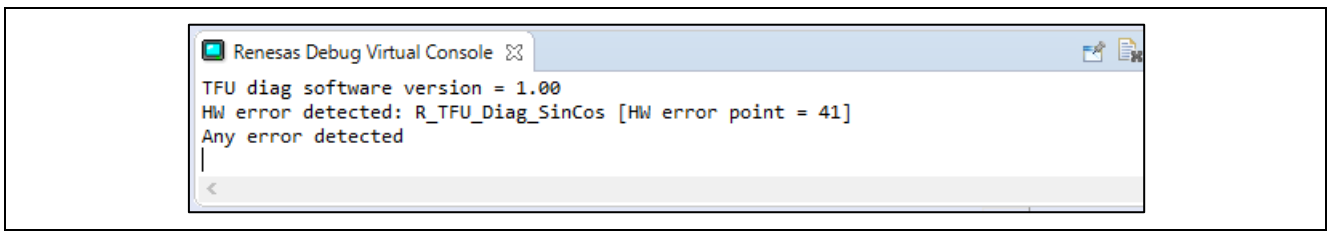

図 3.2 結果メッセージ（故障検出あり）

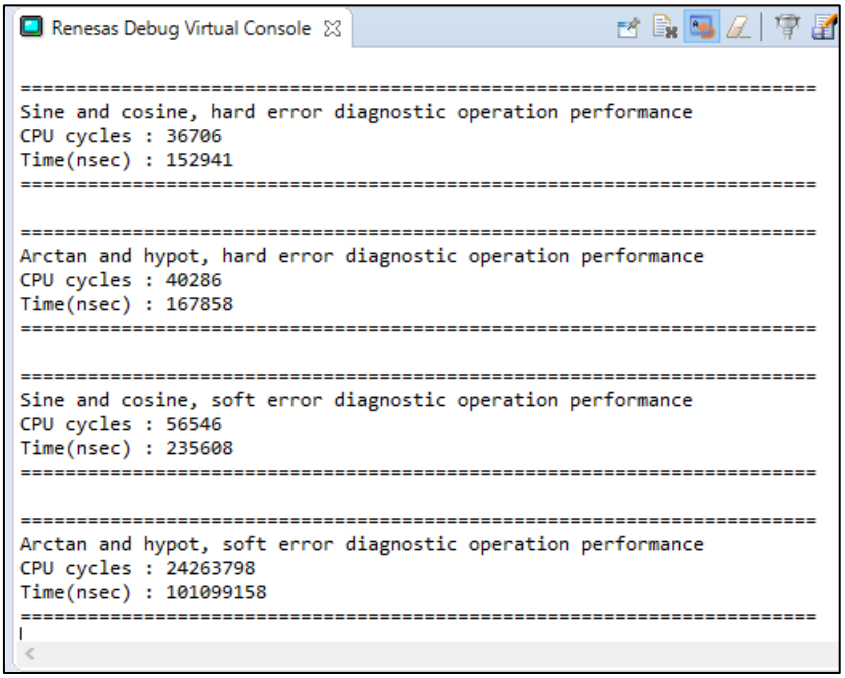

図 3.3　結果メッセージ（処理時間測定時）

3.2　故障診断概要

永久故障と一時故障に対応した故障診断方法の概要を説明します。TFU を使用した数値演算は個々のコンパイラの TFU 用の組み込み関数経由で実行します。TFU 組み込み関数は以下に示す 2 種の演算を提供します。

正弦と余弦演算の同時実行：__sincosf（CC-RX と EWRX）、__builtin_rx_sincosf（GCC）

逆正接と斜辺演算の同時実行：__atan2hypotf（CC-RX と EWRX）、__builtin_rx_atan2hypotf（GCC）

この節では、TFU 組み込み関数として__sincosf と__atan2hypotf を使用した場合を説明します。

3.2.1　永久故障

- 正弦と余弦

(1) 入力θ、正弦の期待値 sin(θ)、余弦の期待値 cos(θ)を格納するルックアップテーブル用のメモリを確保します。そのサイズは 3072 バイトです（256 x 3 x float サイズ）。

(2) ルックアップテーブルからロードしたθ を入力として__sincosf 関数を実行します。

(3) __sincosf 関数から入力された q により、TFU は正弦と余弦演算を同時に実行します。そして、TFU は演算結果である a1 = sin(θ)と a2 = cos(θ)を__sincosf 関数に返します。

(4) サンプルプログラムの故障診断部は TFU の演算結果である a1、a2 とルックアップテーブルの期待値である a1’と a2’の差異を比較します。差異は|a1 – a1’| / |a1’| or |a2 – a2’| / |a2’|で定義される相対誤差です。比較は数値演算の誤差も考慮して行います。比較の結果、数値演算誤差を考慮しても一致しない場合、永久故障と判断します。

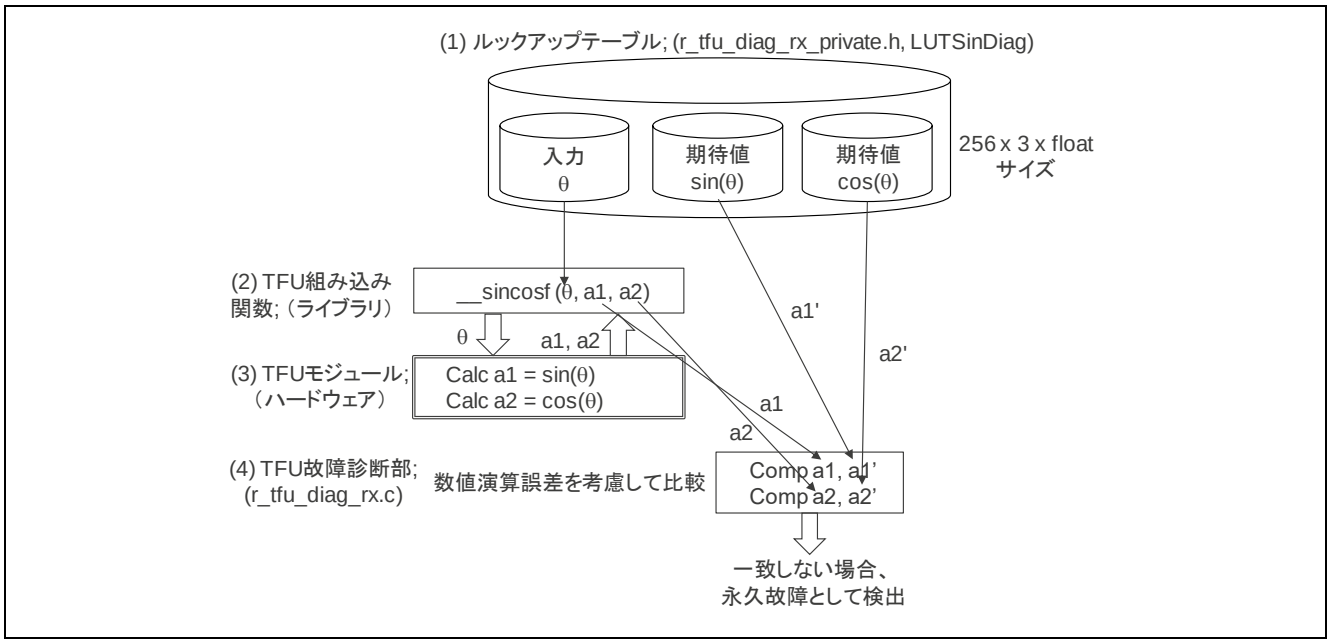

図 3.4　永久故障（正弦と余弦）

- 逆正接と斜辺

(1) 底辺 x、高さ y、逆正接の期待値 arctan(y/x)、斜辺の期待値$(x^2 + y^2)^{1/2}$ を格納するルックアップテーブル用のメモリを確保します。そのサイズは 4096 バイトです（256 x 4 x float サイズ）。

(2) ルックアップテーブルからロードした底辺 x と高さ y を入力として__atan2hypotf 関数を実行します。

(3) __atan2hypotf 関数から入力された底辺 x と高さ y により、TFU は逆正接と斜辺演算を同時に実行します。そして、TFU は演算結果である a1 = arctan(y/x)と $a2 = (x^2 + y^2)^{1/2}$ を__atan2hypotf 関数に返します。

(4) サンプルプログラムの故障診断部は TFU の演算結果である a1、a2 とルックアップテーブルの期待値である a1'と a2'の差異を比較します。差異は|a1 – a1'| / |a1'| or |a2 – a2'| / |a2'|で定義される相対誤差です。比較は数値演算の誤差も考慮して行います。比較の結果、数値演算誤差を考慮しても一致しない場合、永久故障と判断します。

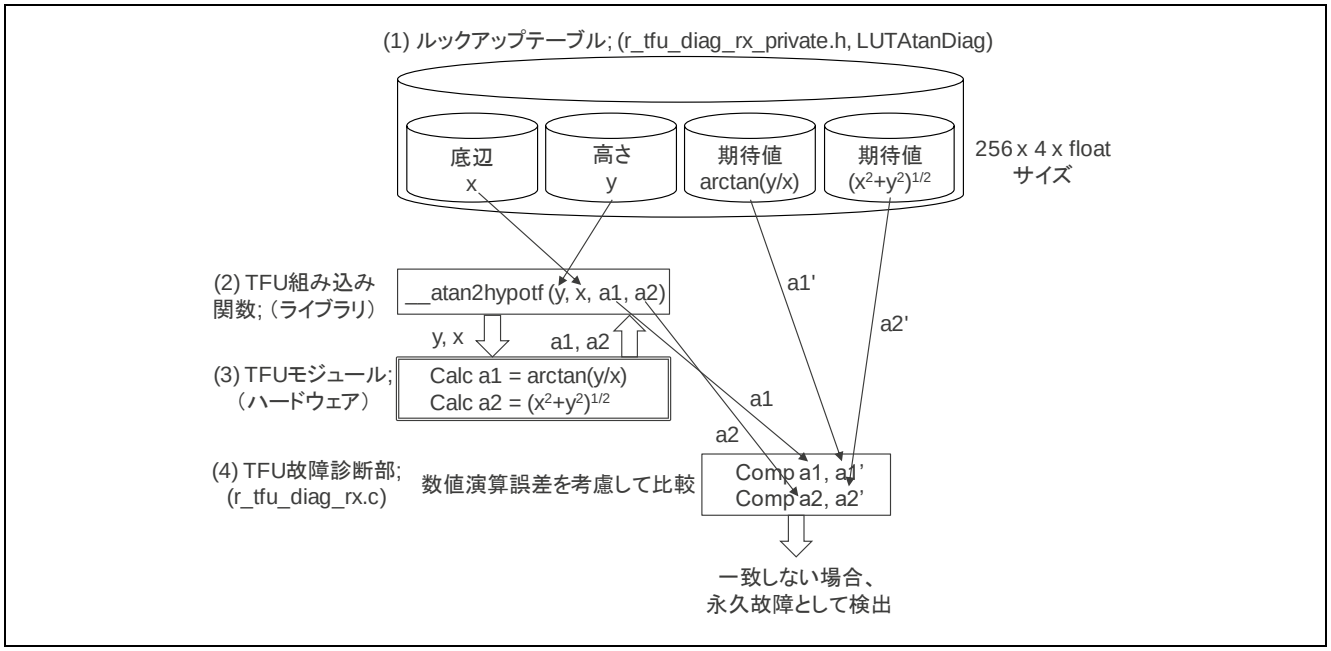

図 3.5　永久故障（逆正接と斜辺）

3.2.2 一時故障

- 正弦と余弦

(1) 入力θ、正弦の期待値 sin(θ)、余弦の期待値 cos(θ)を格納するルックアップテーブル用のメモリを確保します。そのサイズは 3072 バイトです（256 x 3 x float サイズ）。ただし、正弦の期待値 sin(θ)と余弦の期待値 cos(θ)は一時故障の診断では使用しません。

(2) ルックアップテーブルからロードしたθ を入力として__sincosf 関数を実行します。

(3) __sincosf 関数から入力された q により、TFU は正弦と余弦演算を同時に実行します。そして、TFU は演算結果である a1 = sin(θ)と a2 = cos(θ)を__sincosf 関数に返します。

(4) 他の計算方法による結果を期待値とする目的で、正弦と余弦を標準ライブラリの float 型の数学関数である sinf 関数と cosf 関数により、それぞれ計算します。

(5) サンプルプログラムの故障診断部は TFU の演算結果である a1、a2 と標準ライブラリの演算結果である a1’ = sinf(θ)と a2’ = cosf(θ)の差異を比較します。差異は|a1 – a1’| / |a1’| or |a2 – a2’| / |a2’|で定義される相対誤差です。比較は数値演算の誤差も考慮して行います。比較の結果、数値演算誤差を考慮しても一致しない場合、一時故障と判断します。

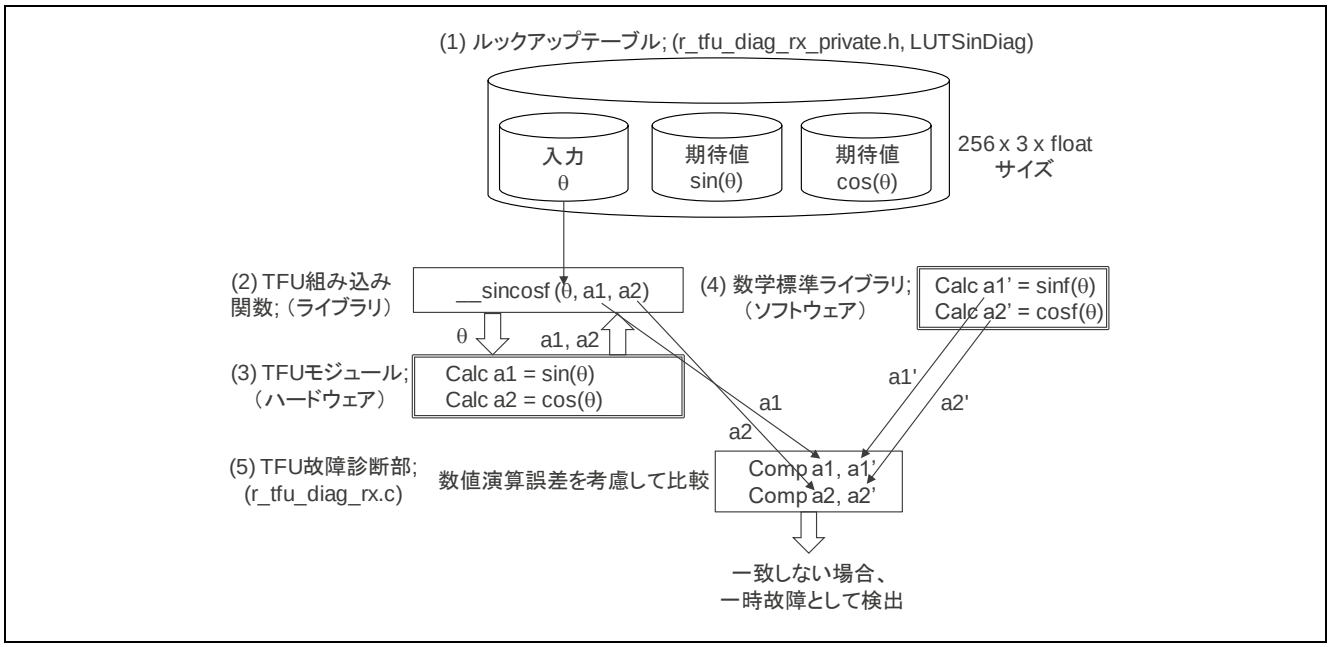

図 3.6 一時故障（正弦と余弦）

- 逆正接と斜辺

(1) 底辺 x、高さ y、逆正接の期待値 arctan(y/x)、斜辺の期待値$(x^2 + y^2)^{1/2}$ を格納するルックアップテーブル用のメモリを確保します。そのサイズは 4096 バイトです（256 x 4 x float サイズ）。ただし、逆正接の期待値 arctan(y/x)と斜辺の期待値$(x^2 + y^2)^{1/2}$ は一時故障の診断では使用しません。

(2) ルックアップテーブルからロードした底辺 x と高さ y を入力として__atan2hypotf 関数を実行します。

(3) __atan2hypotf 関数から入力された底辺 x と高さ y により、TFU は逆正接と斜辺演算を同時に実行します。そして、TFU は演算結果である a1 = arctan(y/x)と a2 = $(x^2 + y^2)^{1/2}$ を__atan2hypotf 関数に返します。

(4) 他の計算方法による結果を期待値とする目的で、逆正接と斜辺を標準ライブラリの float 型の数学関数である atan2f 関数と hypotf 関数により、それぞれ計算します。

(5) サンプルプログラムの故障診断部は TFU の演算結果である a1、a2 と標準ライブラリの演算結果である a1’ = artan2f(y, x) and a2’ = $(x^2 + y^2)^{1/2}$ の差異を比較します。差異は|a1 – a1’| / |a1’| or |a2 – a2’| / |a2’|で定義される相対誤差です。比較は数値演算の誤差も考慮して行います。比較の結果、数値演算誤差を考慮しても一致しない場合、一時故障と判断します。

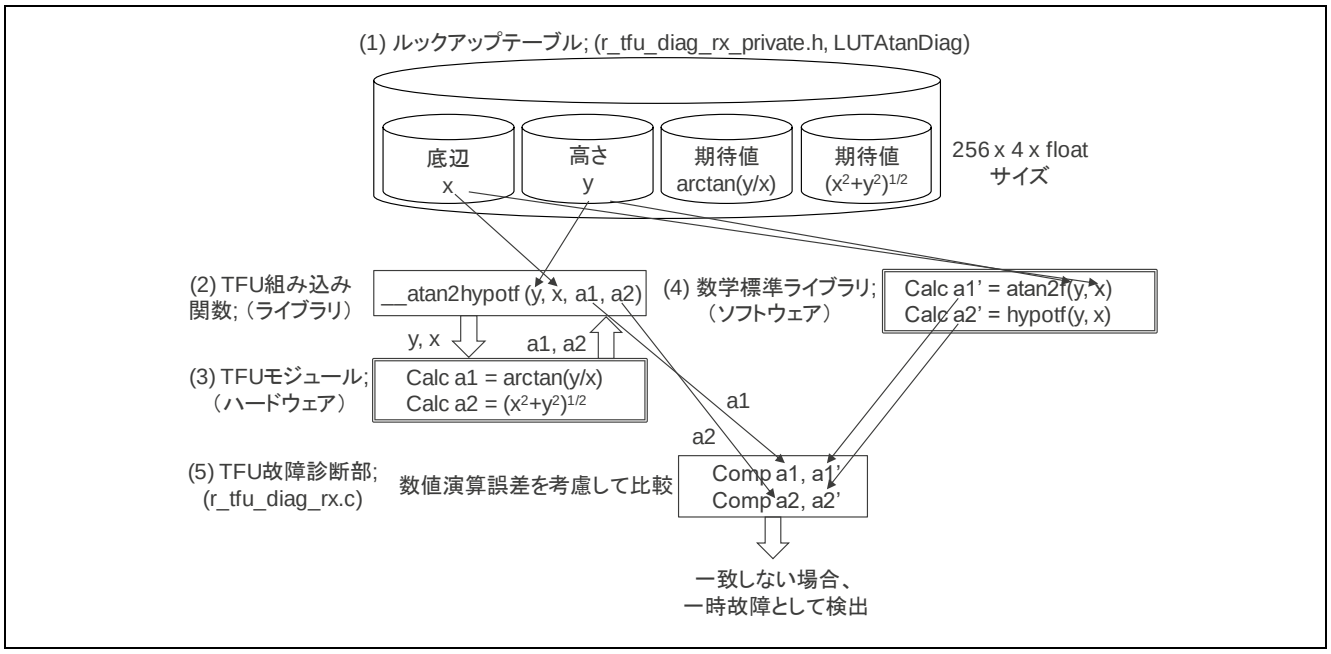

図 3.7　一時故障（逆正接と斜辺）

3.3　動作例

永久故障と一時故障に対応した故障診断方法の概要を説明します。TFU を使用した数値演算は個々のコンパイラの TFU 用の組み込み関数経由で実行します。TFU 組み込み関数は以下に示す 2 種の演算を提供します。

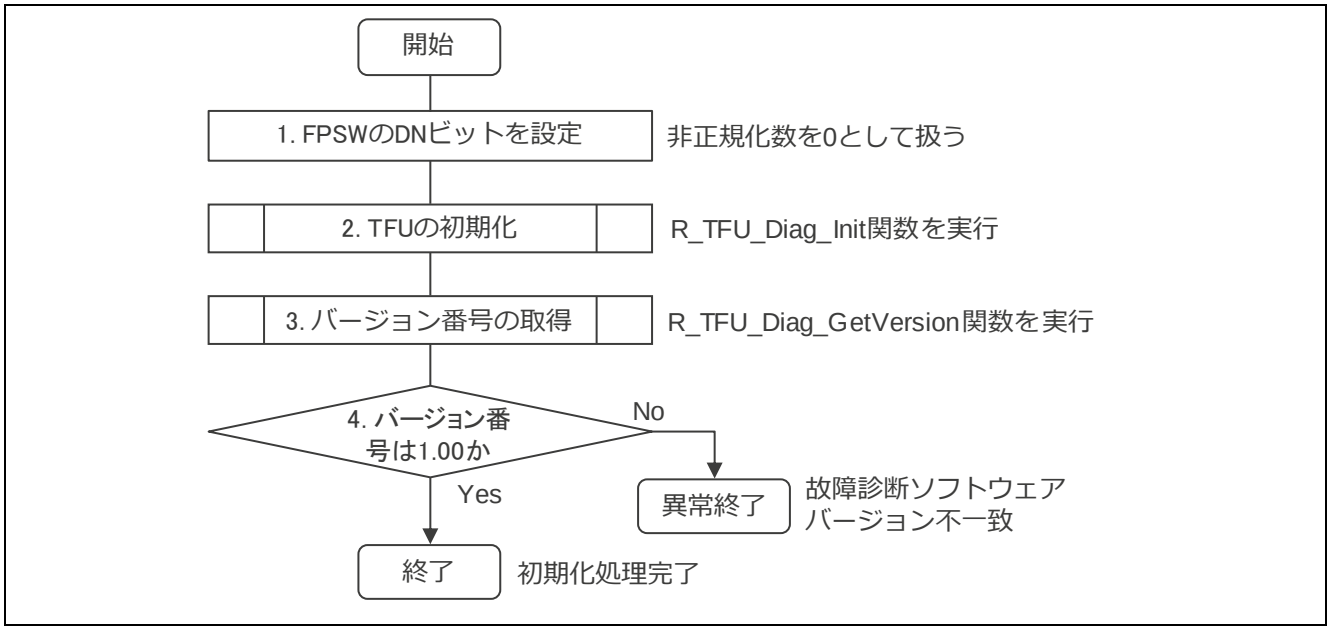

図 3.8　初期化動作例

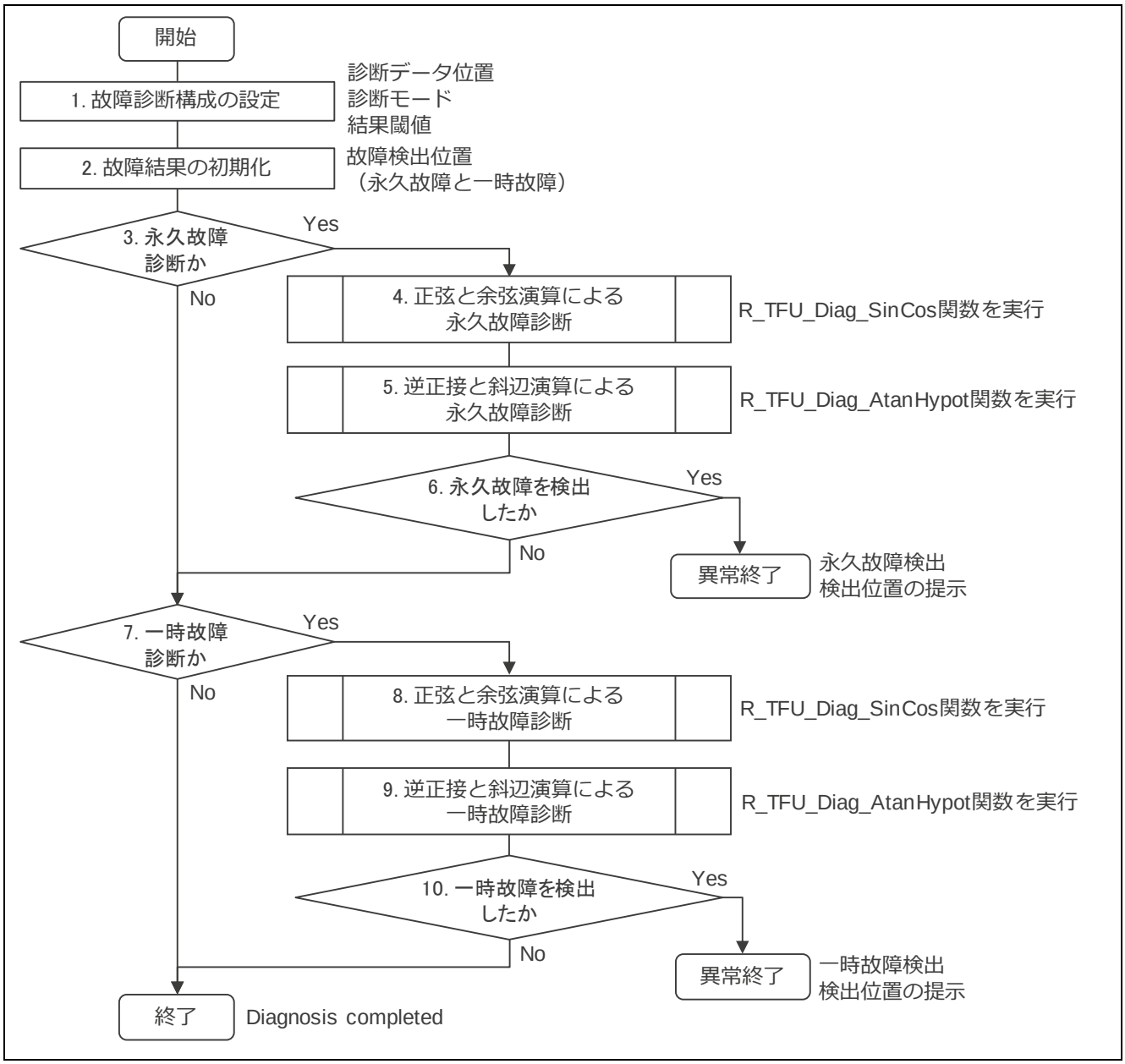
開始
1. 故障診断構成の設定
診断データ位置
診断モード
結果閾値
2. 故障結果の初期化
故障検出位置
（永久故障と一時故障）
3. 永久故障診断か
Yes
No
4. 正弦と余弦演算による永久故障診断
R_TFU_Diag_SinCos関数を実行
5. 逆正接と斜辺演算による永久故障診断
R_TFU_Diag_AtanHypot関数を実行
6. 永久故障を検出したか
Yes
No
異常終了
永久故障検出
検出位置の提示
7. 一時故障診断か
Yes
No
8. 正弦と余弦演算による一時故障診断
R_TFU_Diag_SinCos関数を実行
9. 逆正接と斜辺演算による一時故障診断
R_TFU_Diag_AtanHypot関数を実行
10. 一時故障を検出したか
Yes
No
異常終了
一時故障検出
検出位置の提示
終了
Diagnosis completed

図 3.9　故障診断動作例

3.4 処理時間（測定例）

RX72M と RX72T の故障診断での処理時間の測定結果を参考値として、それぞれ表 3.1 と表 3.2 に示します。これらは個々の処理に要する時間を測定したものです。

表 3.1 RX72M 処理時間

- RX72M 測定条件

ボード	動作周波数	コード領域	データ領域	エンディアン	最適化設定
RX72M RSK+ ボード (RTK5572MNHC00000BJ)	240MHz	内蔵ROM Read: 1サイクル（キャッシュヒット時） 2-3サイクル（ミスヒット時）	内蔵RAM Read/Write: 1サイクル	リトル	機能安全向け設定 詳細は表5.1参照

- 処理時間 (µs単位)

故障の種類	演算の種類	CC-RX V3.01.00	EWRX V4.12.1	GCC V4.08.04
永久故障	正弦と余弦	153	114	193
	逆正接と斜辺	168	123	205
一時故障	正弦と余弦	236	237	379
	逆正接と斜辺	101,714	330,770	101,485

表 3.2 RX72T 処理時間

- RX72T 測定条件

ボード	動作周波数	コード領域	データ領域	エンディアン	最適化設定
RX72T RSK board (RTK5572TKCS00000BE)	200MHz	内蔵ROM Read: 1サイクル（キャッシュヒット時） 2-3サイクル（ミスヒット時）	内蔵RAM Read/Write: 1サイクル	リトル	機能安全向け設定 詳細は表5.1参照

- 処理時間 (µs単位)

故障の種類	演算の種類	CC-RX V3.01.00	EWRX V4.12.1	GCC V4.08.04
永久故障	正弦と余弦	181	135	230
	逆正接と斜辺	199	145	244
一時故障	正弦と余弦	277	269	451
	逆正接と斜辺	100,336	284,940	100,417

4. API 関数

サンプルプログラムの TFU 故障診断部（r_tfu_diag_rx.c）には 4 種の API 関数があります。この節では API 関数の仕様を説明します。

4.1 R_TFU_Diag_GetVersion ()

この関数は TFU 故障診断ソフトウェアのバージョン番号を返します。

フォーマット

uint32_t R_TFU_Diag_GetVersion(void);

パラメータ

なし

戻り値

TFU_DIAG_VERSION_MAJOR（上位 16 ビット）：メジャーバージョン番号

TFU_DIAG_VERSION_MINOR（下位 16 ビット）：マイナーバージョン番号

プロパティ

r_tfu_diag_rx_if.h にプロトタイプ宣言しています。

説明

TFU 故障診断ソフトウェアのメジャーバージョン番号とマイナーバージョン番号を返します。

バージョン番号は「1.00」です。

- 上位 16 ビットはメジャーバージョン番号を示します。
  TFU_DIAG_VERSION_MAJOR: 値 = H’1.
- 下位 16 ビットはマイナーバージョン番号を示します。
  TFU_DIAG_VERSION_MINOR: 値 = H’00.

リエントラント

関数はリエントラントです。

使用例

この関数の使用例を以下に示します。

```
#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);
```

注意

ソフトウェアのバージョンによって、戻り値は異なります。

4.2 R_TFU_Diag_Init ()

この関数は TFU を初期化します。

フォーマット

void R_TFU_Diag_Init(void);

パラメータ

なし

戻り値

なし

プロパティ

r_tfu_diag_rx_if.h にプロトタイプ宣言しています。

説明

この関数は__init_tfu()組み込み関数を実行することで TFU を初期化します。

TFU の初期化が行われるのは CC-RX または GCC コンパイラを使用した環境のみです。

EWRX コンパイラを使用した環境では、この関数は何も処理を行いません。

リエントラント

この関数はリエントラントです。

使用例

この関数の使用例を以下に示します。

```
#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);
```

注意

EWRX コンパイラを使用した環境では、この関数は何も処理を行いません。

4.3 R_TFU_Diag_SinCos ()

この関数は正弦と余弦演算による故障診断を行います。

フォーマット

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

パラメータ

conf – 診断構成情報

res – 診断結果情報

戻り値

TFU_OK: 処理は正常に終了しました。

TFU_ERR_PARAM: パラメータに誤りがありました。

TFU_ERR_DET: 故障を検出しました。

プロパティ

r_tfu_diag_rx_if.h にプロトタイプ宣言しています。

説明

この関数は正弦と余弦演算による故障診断を行います。以下の動作を実行します。

- 入力された診断構成情報の内容を確認します。
  数値と診断モードが範囲外であれば、エラー（TFU_ERR_PARAM）を返します。
- 入力θをルックアップテーブルからロードします。
- TFU により正弦と余弦演算を実行します。
  TFU 組み込み関数を呼ぶことで実行します。CC-RX または EWRX 環境では、TFU 組み込み関数は __sincosf()です。GCC 環境では、__builtin_rx_sincosf()です。
- 診断モードが永久故障の場合、永久故障診断を行います。
  ルックアップテーブルから正弦と余弦の期待値をロードし、TFU での演算結果と期待値の相対誤差を数値演算の誤差を考慮して比較することで評価します。そして、相対誤差が数値演算の誤差として定めた閾値を超えていた場合、永久故障を検出したと判定し、故障検出位置を返します。
- 診断モードが一時故障の場合、一時故障診断を行います。
  標準ライブラリの float 型の数学関数である sinf 関数と cosf 関数により、正弦と余弦をそれぞれ計算し、その結果を期待値とします。TFU での演算結果と期待値の相対誤差を数値演算の誤差を考慮して比較することで評価します。そして、相対誤差が数値演算の誤差として定めた閾値を超えていた場合、一時故障を検出したと判定し、故障検出位置を返します。

リエントラント

関数はリエントラントです。

使用例

この関数の使用例は「4.2 R_TFU_Diag_Init ()」に説明しています。

注意

この関数は個々のコンパイラの float 型の数学関数を必要とします。

4.4 R_TFU_ Diag_AtanHypot ()

この関数は逆正接と斜辺演算による故障診断を行います。

フォーマット

tfu_return_t R_TFU_Diag_ AtanHypot(DiagConf *conf, DiagRes *res);

パラメータ

conf – 診断構成情報

res – 診断結果情報

戻り値

TFU_OK: 処理は正常に終了しました。

TFU_ERR_PARAM: パラメータに誤りがありました。

TFU_ERR_DET: 故障を検出しました。

プロパティ

r_tfu_diag_rx_if.h にプロトタイプ宣言しています。

説明

この関数は正弦と余弦演算による故障診断を行います。以下の動作を実行します。

- 入力された診断構成情報の内容を確認します。
  数値と診断モードが範囲外であれば、エラー（TFU_ERR_PARAM）を返します。
- 入力θをルックアップテーブルからロードします。
- TFU により正弦と余弦演算を実行します。
  TFU 組み込み関数を呼ぶことで実行します。CC-RX または EWRX 環境では、TFU 組み込み関数は __atan2hypotf()です。GCC 環境では、__builtin_rx_atan2hypotf()です。
- 診断モードが永久故障の場合、永久故障診断を行います。
  ルックアップテーブルから逆正接と斜辺の期待値をロードし、TFU での演算結果と期待値の相対誤差を数値演算の誤差を考慮して比較することで評価します。そして、相対誤差が数値演算の誤差として定めた閾値を超えていた場合、永久故障を検出したと判定し、故障検出位置を返します。
- 診断モードが一時故障の場合、一時故障診断を行います。
  標準ライブラリの float 型の数学関数である atan2f 関数と hypotf 関数により、逆正接と斜辺をそれぞれ計算し、その結果を期待値とします。TFU での演算結果と期待値の相対誤差を数値演算の誤差を考慮して比較することで評価します。そして、相対誤差が数値演算の誤差として定めた閾値を超えていた場合、一時故障を検出したと判定し、故障検出位置を返します。

リエントラント

関数はリエントラントです。

使用例

この関数の使用例は「4.2 R_TFU_Diag_Init ()」に説明しています。

注意

この関数は個々のコンパイラの float 型の数学関数を必要とします。

5. 付録

5.1 動作確認環境

サンプルプログラムの動作確認環境を以下に示します。

表 5.1 動作確認環境

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V7.5.0 IAR Embedded Workbench for Renesas RX 4.12.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99, -optimize = 0 (level 0: no optimization), -tfu = Intrinsic (use trigonometric function)
	GCC for Renesas RX 4.08.04. 201902-SP1-GNURX コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99, MTFU = intrinsic
	IAR C/C++ Compiler for Renesas RX version 4.12.1 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 no optimization, TFU intrinsics
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev.1.00
使用ボード	Renesas Starter Kit+ for RX72M Renesas Starter Kit+ for RX72T

6. 提供されているモジュール

提供されているモジュールは、ルネサス エレクトロニクスのウェブサイトからダウンロードすることができます。

7. 参考ドキュメント

ユーザーズマニュアル: ハードウェア

RX72M グループユーザーズマニュアル ハードウェア編 Rev.1.00（R01UH0804JJ）

RX72T グループユーザーズマニュアル ハードウェア編 Rev.1.00（R01UH0803JJ）

最新版はルネサス エレクトロニクスのウェブサイトからダウンロードできます。

テクニカルアップデート／テクニカルニュース

（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

ユーザーズマニュアル：開発環境

RX ファミリ CC-RX コンパイラ ユーザーズマニュアル Rev.1.08（R20UT3248JJ）

（最新版をルネサス エレクトロニクスホームページから入手してください。）

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2019.08.31	—	初版発行

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.4.0-1　2017.11)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。