R01AN3475JJ0100
Rev.1.00
2017.07.31

RX ファミリ
多機能タイマ故障診断サンプルプログラム

要旨

本アプリケーションノートは、EPTPC FIT (Firmware Integration Technology) モジュール[1]を使用した多機能タイマの故障診断例について記載します。

動作確認デバイス

以下のデバイスがこの実例でサポートされます。

- RX64M グループ
- RX71M グループ

本アプリケーションノートを他の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

RX に搭載するマルチファンクションタイマパルスユニット（MTU3）や汎用 PWM タイマ（GPT）のような多機能タイマは産業機器制御では主要なモジュールとなっています。また、機能安全[2]の観点から多機能タイマの故障診断も重要な処理です。

アプリケーションノートでは、EPTPC FIT モジュール（以下、PTP(Precision Time Protocol)ドライバ）を使用した多機能タイマ（以下、タイマモジュール）の故障診断例について説明します。この例では、タイマモジュールのタイマカウンタと EPTPC のローカルクロックカウンタを比較し、これらの差異が閾値を超えていた場合にタイマモジュールの故障として検出します。閾値は実行済みの故障診断の結果にあらかじめ定義した補正値を加えた値として決めています。比較の基準クロックとして使用される EPTPC のローカルクロックカウンタは IEEE1588 同期イーサネット[3]のネットワークでの外部クロックに同期していること、及び PTP ドライバのイベントリンクコントローラ（ELC）経由でのタイマモジュールとの連係機能[4]により、信頼性が高く正確なタイマモジュールの故障診断の実装が可能になります。

1.1　多機能タイマ故障診断サンプルプログラム

このサンプルプログラムはプロジェクト形式で提供しており、複数の FIT モジュールを使用したタイマモジュールの 故障診断例として使用できます。

1.2　関連ドキュメント

[1] RX ファミリ EPTPC モジュール Firmware Integration Technology, Rev.1.14, Document No. R01AN1943JJ0114, Apr 30, 2017
[2] Functional safety of electrical/electronic/programmable electronic safety-related systems, IEC61508, Edition 2.0, Apr, 2010
[3] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, Revision of IEEE Std 1588-2008, Mar 2008
[4] RX ファミリ EPTPC：PTP タイマ同期開始サンプルプログラム, Rev.1.12, Document No. R01AN1984JJ0112, Mar 31, 2017
[5] RX ファミリ イーサネットモジュール Firmware Integration Technology, Rev.1.12, Document No. R01AN2009JJ0112, Nov 11, 2016
[6] 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.3.50, Document No. R01AN1685JJ0350, May 15, 2017
[7] RX Family Flash Module Using Firmware Integration Technology, Rev.2.10, Document No. R01AN2184EU0210, Dec 20, 2016
[8] Renesas Starter Kit+ for RX64M, ユーザーズマニュアル、Rev.1.20, Document No. R20UT2590JG0102, Jun 25, 2015
[9] Renesas Starter Kit+ for RX71M, ユーザーズマニュアル、Rev.1.00, Document No. R20UT3217JG0100, Jan 23, 2015

1.3　ハードウェアの構成

このサンプルプログラムはMTU3、GPT、EPTPC、ETHERC、EDMAC、ELC、RAM とデータフラッシュを使用します。
MTU3とGPTは故障診断の対象モジュールとして使用します。
RX64M/71M グループのイーサネットモジュールは、EPTPC、PTP 用イーサネットコントローラ用 DMA コントロートローラ（PTPEDMAC）、2 チャネルのイーサネットコントローラ（ETHERC（CH0）、ETHERC（CH1））、および 2 チャネルのイーサネットコントローラ用 DMA コントローラ（EDMAC（CH0）、EDMAC（CH1））で構成しています。EPTPC は、PTP 同期フレーム処理部（CH0）、PTP 同期フレーム処理部（CH1）、パケット中継部（PRC-TC）、および統計的クロック補正部で構成しています。また、EPTPC は、MTU3、GPT、I/O ポートに ELC を介して接続し、時刻同期したイベントを発生させることができます。データフラッシュはイーサネットの MAC アドレスとこれまでの故障診断で使用した閾値を格納します。

詳細に関しては「RX64M/71M 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。

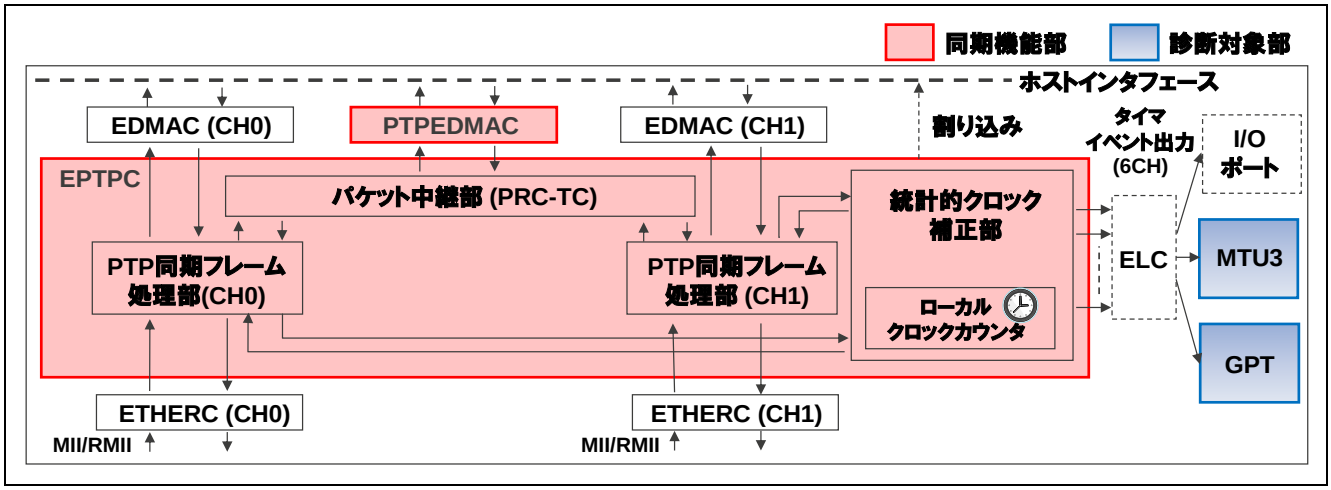

図1.1　ハードウェア構成

1.4　ソフトウェアの構成

このサンプルプログラムは、共通の初期設定をボードサポートパッケージモジュール[5]として提供する複数の FIT モジュールを使用したアプリケーション層での実行例です。

図 1.2にソフトウェアの典型的な構成と機能概要を示します。アプリケーションはタイマモジュールの故障診断、外部クロックとの同期、診断用パラメータの設定等で構成します。タイマモジュールの故障診断は、MTU3/GPT のタイマカウンタと EPTPC のローカルクロックカウンタを同期開始後、種々の誤差を考慮して互いのカウンタを比較することでタイマモジュールの故障を検出します。PTP ドライバは外部デバイスと PTP のプロトコルにより時刻同期します。また、PTP ドライバは標準イーサネットフレームの処理をする Ethernet ドライバ[6]と共に使用する必要があります。TCP/IP ミドルウェアはこの例には含まれていませんので、TCP/IP を使用するには、別途、TCP/IP ミドルウェアを用意する必要があります。ELC ドライバが EPTPC のイベントと MTU3/GPT のイベントを接続することで、MTU3/GPT のタイマカウンタと EPTPC のローカルクロックカウンタを同期開始することを可能にします。フラッシュドライバ（フラッシュ API）[7]は、診断用のパラメータと MAC アドレスのデータフラッシュからの消去とデータフラッシュへの書き込みを行います。

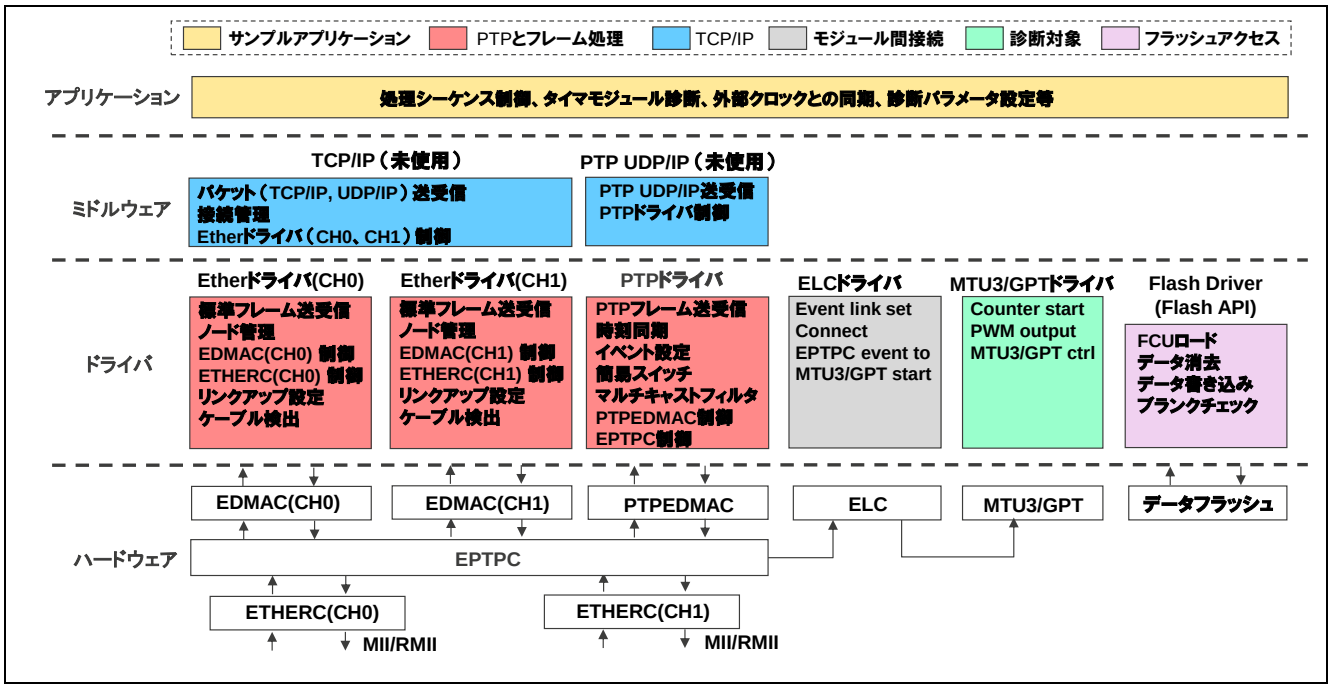

図1.2　ソフトウェア構成

1.5　ファイル構成

このサンプルプログラムのコードは demo_src と下位階層のフォルダに格納されています。ELC、GPT、MTU3 ドライバは private_driver フォルダの下位フォルダにサンプル用ドライバとして、それぞれ格納されています。図 1.3はファイル構成を示しています。r_pincfg フォルダはイーサネット通信用の端子設定をするファイルを格納します。FIT モジュールに関しては、それぞれの FIT モジュール（BSP、イーサネットドライバ、フラッシュドライバ、PTP ドライバ）のドキュメントを参照してください。

```
demo_src: メイン動作と動作構成設定
|   sample_main.c
|   sample_main.h
|
+ --- flash_if: データフラッシュアクセス処理
|       flash_if.c
|       flash_if.h
|
+ --- sync: PTP同期処理
|       sync.c
|       sync.h
|
+ --- usr: LEDの制御
|       led.c
|       led.h

private_drivers: サンプル用ドライバ （FITモジュールとは異なる）
|
+ --- r_elc_rx: ELCドライバフォルダ
|   r_elc_rx_if.h ;ELCドライバヘッダファイル
|
+ --- src:
|       r_elc.c ; ELCドライバソースファイル
|
r_gpt_rx: GPTドライバフォルダ
|   r_gpt_rx_if.h ;GPTドライバヘッダファイル
|
+ --- src:
|       r_gpt.c ; GPTドライバソースファイル
|
r_mtu3_rx: MTU3ドライバフォルダ
|   r_mtu3_rx_if.h ;MTU3ドライバヘッダファイル
|
+ --- src:
|       r_mtu3.c ; MTU3ドライバソースファイル
```

```
r_config: FITモジュールの構成設定
|       r_bsp_config.h
|       r_bsp_interrupt_config.h
|       r_ether_rx_config.h
|       r_flash_rx_config.h
|       r_mcu_config.h
|       r_ptp_rx_config.h

r_bsp: BSP (Board Support Package) FITモジュール

r_ether_rx: イーサネットドライバ FITモジュール

r_flash_rx: フラッシュドライバ（フラッシュAPI） FITモジュール

r_pincfg: 端子設定ファイルフォルダ

r_ptp_rx: PTPドライバ FITモジュール
```

図1.3　ファイル構成

2. 機能情報

本サンプルプログラムは以下の内容で開発しています。

2.1 ハードウェア要件

サンプルプログラムは、使用する MCU が以下の機能をサポートしている必要があります。

- EPTPC
- ETHERC
- EDMAC (PTPEDMAC)
- ELC
- MTU3
- GPT
- データフラッシュ

2.2 ハードウェアリソース要件

サンプルプログラムで使用するドライバ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。特に明記されていない限り、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 EPTPC チャネル

この例では、EPTPC を使用します。EPTPC は外部デバイスと PTP のプロトコルによる時刻同期、及び ELC 経由での MTU3/GPT を起動します。

2.2.2 ETHERC チャネル

この例では、クロック（ノード）の種類に応じて、ETHEC（CH0）、ETHEC（CH1）、またはその両方を使用します。これらの周辺回路はイーサネットの MAC 層の動作をします。

2.2.3 EDMAC チャネル（PTPEDMAC）

この例では、クロック（ノード）の種類に応じて、EDMAC（CH0）、EDMAC（CH1）、またはその両方を使用します。これらの周辺回路は標準イーサネットフレーム（PTP フレーム）処理での CPU ホストインタフェースとして動作します。

2.2.4 ELC

この例では EPTPC と MTU3/GPT の間のイベント接続に ELC を使用します。ELC は MTU3/GPT を同期開始します。

2.2.5 MTU3 チャネル

この例では故障診断の対象として MTU3 チャネルを使用します。MTU3 はカウントアップ動作をします。

2.2.6 GPT チャネル

この例では故障診断の対象として GPT チャネルを使用します。GPT はカウントアップ動作をします。

2.2.7 データフラッシュ

この例では診断用のパラメータ格納にデータフラッシュを使用します。

2.3　ソフトウェアリソース要件

サンプルプログラムは以下の FIT モジュールを使用しています。

- r_bsp
- r_ether_rx
- r_ptp_rx
- r_flash_rx

2.4　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次のツールチェインでテストと動作確認を行っています。

- Renesas RX Toolchain v2.06.00

2.5　ヘッダファイル

すべての関数呼び出しは、サンプルプログラムとともに提供されているヘッダファイル sample_main.h, flash_if.h, sync.h, led.h, r_elc_rx_if.h, r_gpt_rx_if.h, r_mtu3_rx_if.h のうち 1 個のファイルをインクルードすることで行われます。

2.6　整数型

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

2.7　コンパイル時の設定

サンプルプログラムのコンパイル設定は sample_main.h, flash_if.h, r_gpt_rx_if.h, r_mtu3_rx_if.h で設定します。オプション名と設定値を以下の表に記載します。

構成設定	
#define DEVICE_ID - Default value = 0	デバイスを特定する番号を設定します。 • 0～2 を設定してください。 このサンプルでは他の値はサポートしません。
#define MODE_PORT - Default value = 1	クロックの種類を設定します。 • 0 を設定すると、OC port0 を使用します。 • 1 を設定すると、OC port1 を使用します。 この例では BC と TC は対応していません。 OC port0 を選択した場合、r_ether_rx_config.h の ETHER_CFG_CH0/1_PHY_ACCESS を 0 に設定してください。 OC port1 を選択した場合、r_ether_rx_config.h の ETHER_CFG_CH0/1_PHY_ACCESS を 1 に設定してください。
#define MS_PORT0/1 - Default value = 0	port0/port1 のマスタまたはスレーブを設定します。 • 0 を設定すると、クロックはマスタです。 • 1 を設定すると、クロックはスレーブです。
#define SYNC_PORT0/1 - Default value = 1	port0/port1 の遅延メカニズム（P2P/E2E）を設定します。 • 0 を設定すると、遅延メカニズムは P2P になります。 • 1 を設定すると、遅延メカニズムは E2E になります。
#define SAMPLE_VID - Default value = 0x00749050	MAC アドレスのベンダ ID を設定します。 デフォルト値はルネサスのベンダ ID（74-90-50）です。 お客様のシステムに適用する場合、必ずこの値を変更してください。

構成設定	
#define MAC_ADDR_1H/2H - Default value = 0x00749050 (port0) - Default value = 0x00749050 (port1)	port0/port1 の MAC アドレスの上位 24 ビットを設定します。 下位 24 ビットはベンダ ID フィールドで、デフォルト値はルネサスベンダ ID（74-90-50）を設定しています。 上位 8 ビットは予約で 00 を設定してください。 お客様のシステムに適用する場合、必ずこの値を変更してください。
#define MAC_ADDR_1L/2L Case of device id = 0, - Default value = 0x0000795F (port0) - Default value = 0x00007960 (port1) Case of device id = 1, - Default value = 0x00007961 (port0) - Default value = 0x00007962 (port1) Case of device id = 2, - Default value = 0x00007963 (port0) - Default value = 0x00007964 (port1)	port0/port1 の MAC アドレスの下位 24 ビットを設定します。 下位 24 ビットはプロダクト ID フィールドで、デフォルト値はこのサンプルプログラム固有の値を設定しています。 上位 8 ビットは予約で 00 を設定してください。 お客様のシステムに適用する場合、必ずこの値を変更してください。
#define IP_ADDR_1/2 Case of device id = 0, - Default value = 0x0A0B0C0D (port0) - Default value = 0x1A1B1C1D (port1) Case of device id = 1, - Default value = 0x2A2B2C2D (port0) - Default value = 0x3A3B3C3D (port1) Case of device id = 2, - Default value = 0x4A4B4C4D (port0) - Default value = 0x5A5B5C5D (port1)	port0/port1 の IP（IPv4）アドレスを設定します。 お客様のシステムに適用する場合、必ずこの値を変更してください。
#define DIAG_START_SEC - Default value = 15	診断開始時刻の秒単位を設定します。 • 0～0xFFFFFFFF(= 4,294,967,295)を設定してください。
#define DIAG_START_NSEC - Default value = 0	診断開始時刻のナノ秒単位を設定します。 • 0～999,999,999 を設定してください。
#define STCA_TMR_CH - Default value = 0	STCA のパルス出力タイマを選択します。 • 0～5 を設定してください。
#define STCA_TMR_EDGE - Default value = 0	STCA の ELC イベントトリガの立ち上がりエッジ、または立ち下がりエッジを選択します。 • 0 を設定すると、立ち上がりエッジになります。 • 1 を設定すると、立ち下がりエッジになります。 このサンプルでは 0（立ち上がりエッジ）を選択してください。
#define DIAG_TMR_KND - Default value = 0	診断対象の多機能タイマの種類を選択します。 • 0 を設定すると、診断対象タイマが MTU3 になります。 • 1 を設定すると、診断対象タイマが GPT になります。
#define DIAG_TMR_CH - Default value = 0	診断対象の多機能タイマのチャネルを設定します。 MTU3 の場合、0:CH0, 3:CH3, 4:CH4 から選択してください。 GPT の場合、0:CH0, 1:CH1, 2:CH2, 3:CH3 から選択してください。 このサンプルでは他の値はサポートしません。
#define DIAG_TMR_FREQ - Default value = 1	多機能タイマのカウントアップ時の周波数を選択します。 • 0 を設定すると、周波数は 120MHz(=PCLKA/1)になります。 • 1 を設定すると、周波数は 60MHz(=PCLKA/2)になります。 • 2 を設定すると、周波数は 30MHz(=PCLKA/4)になります。 • 3 を設定すると、周波数は 15MHz(=PCLKA/8)になります。 このサンプルでは他の値はサポートしません。

構成設定	
#define INIT_TMR_THRESH - Default value = 1000000	タイマカウンタ比較に使用する閾値の初期値を設定します。 設定値はナノ秒単位です。
#define COR_TMR_THRESH - Default value = 1000	タイマカウンタ比較に使用する閾値を補正する値（先回までに行った診断でのカウンタ差異の絶対値の平均値に加える値）を設定します。 設定値はナノ秒単位です。
#define DIAG_PERIOD - Default value = 300000000	診断を行う間隔を設定します。 設定値はナノ秒単位です。
#define FORCE_ERASE_PRM - undefined	動作完了後、データフラッシュに書き込まれた診断パラメータを消去するか、残すかを選択します。 定義した場合、データフラッシュからパラメータを消去します。
#define USE_HYST_THRESH - undefined	閾値を診断履歴から設定するか、初期値（=INIT_TMR_THRESH）を設定するかを選択します。 定義した場合、閾値を診断履歴を基に設定します。
#define NUM_DEV - Default value = 3	割り当て可能なデバイス番号の数を設定します。 このサンプルでは他の値はサポートしてません。
#define GPT_CFG_INTERRUPT_LEVEL - Default value = 3	GPT からの割り込み優先度レベルを設定します。 • 1～15 を設定してください。
#define MTU3_CFG_INTERRUPT_LEVEL - Default value = 3	MTU3 からの割り込み優先度レベルを設定します。 • 1～15 を設定してください。

2.8　データ構造

サンプルプログラムで使用するデータ構造について説明します。これらの構造体は sample_main.h と flash_if.c にプロトタイプ宣言と共に定義しています。

```
/* Communication parameter */
typedef struct
{
     uint32_t vd; /* MAC address vendor ID field */
     uint32_t pd; /* MAC address product ID field */
} CommParm;
```

```
/* Timer diagnosis result hysteresis */
typedef struct
{
     uint32_t num;  /* Number of sampling */
     uint32_t max;  /* Maximum absolute difference */
     uint32_t min;  /* Minimum absolute difference */
     uint32_t mean; /* Mean absolute difference */
} DiagHyst;
```

```
/* Communication parameter segment address */
const flash_block_address_t comm_addr[NUM_DEV] =
{ /* Refer to "r_flash_rx/src/targets/rx64m/r_flash_rx64m.h or
rx71m/r_flash_rx71m.h". */
     FLASH_DF_BLOCK_256, /* ID = 0: 0x00104000 to 0x0010403F (64B) */
     FLASH_DF_BLOCK_257, /* ID = 1: 0x00104040 to 0x0010407F (64B) */
     FLASH_DF_BLOCK_258, /* ID = 2: 0x00104080 to 0x001040BF (64B) */
};
```

```
 /* Timer diagnosis result hysteresis address */
 const flash_block_address_t diag_addr[NUM_DEV] =
 { /* Refer to "r_flash_rx/src/targets/rx64m/r_flash_rx64m.h or
rx71m/r_flash_rx71m.h". */
      FLASH_DF_BLOCK_320, /* 0x00105000 to 0x0010503F (64B) */
      FLASH_DF_BLOCK_321, /* 0x00105040 to 0x0010507F (64B) */
      FLASH_DF_BLOCK_322, /* 0x00105080 to 0x001050BF (64B) */
 };
```

2.9　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 flash_if.h, r_elc_rx_if.h, r_mtu3_rx_if.h、r_gpt_rx_if.h にプロトタイプ宣言と共に定義しています。

```
 /* Data flash access return value */
 typedef enum
 {
      FLSIF_ERR_ERASE = -5,  /* Flash erase error */
      FLSIF_ERR_WRITE = -4,  /* Flash write error */
      FLSIF_ERR_VERIFY = -3, /* Flash verify error */
      FLSIF_ERR_PARAM = -2,  /* Parameter error */
      FLSIF_ERR = -1,       /* General error */
      FLSIF_OK = 0,
      FLSIF_BLANK = 1,      /* Flash was blank */
 } flshif_t;
```

```
 /* ELC driver return value */
 ELC_OK (0) /* No error */
 ELC_ERROR (-1) /* General error */
```

```
 /* MTU3 driver return value */
 MTU3_OK (0) /* No error */
 MTU3_ERROR (-1) /* General error */
```

```
 /* GPT driver return value */
 GPT_OK (0) /* No error */
 GPT_ERROR (-1) /* General error */
```

3. サンプルプログラムの仕様

3.1 環境とプログラムの実行

サンプルプログラムは、2台以上のRX64M/71M RSKボード[1]（マスタノードと1個以上のスレーブノード）、イーサネットハブ（以後、ハブ）、イーサネットケーブルを使用します。

実行手順の概要を以下に説明します。

- 全ての RX64M /71M RSK ボード（以後、RSK ボード）のコードフラッシュに実行コードを書き込んでください。
- 全 RSK ボードを互いにイーサネットケーブルによりハブ経由で接続してください。
- 全 RSK ボードに電源を投入してください。
- RSK ボードは EPTPC ドライバ、Ethernet ドライバ、フラッシュドライバ、ELC ドライバ、MTU3 ドライバ、GPT ドライバの初期化が完了すると、ユーザ LED が全て点灯します（LED3: オン、LED2: オン、LED1: オン、LED0: オン）。
- MAC アドレス[2] は CH0 に”74-90-5-00-79-5F”、CH1 に”74-90-5-00-79-60”があらかじめソースファイルで設定しています。
- ユーザスイッチの”SW1”を押すと、RSK ボードは EPTPC ドライバと Ethernet ドライバを開始します。そして、同期を開始するとユーザ LED が”0x1”パターンに点灯します（LED3: オフ、LED2: オフ、LED1: オフ、LED0: オン）。
- RSK ボードは同期を開始した時刻より 30 秒進んだ時刻を診断開始時刻として設定します。
- 診断開始時刻を過ぎると、RSK ボードは多機能タイマの診断を開始します。
- ユーザスイッチの”SW2”を押すと、多機能タイマの診断を終了します。そして、ユーザ LED が偶数パターンに点灯します（LED3: オフ、LED2: オン、LED1: オフ、LED0: オン）。
- 処理中に故障または他のエラーを検出した場合、ユーザ LED が奇数パターンに点灯します（LED3: オン、LED2: オフ、LED1: オン、LED0: オフ）。

[1] 製品名は Renesas Starter Kit+ for RX64M [8] または Renesas Starter Kit+ for RX71M [9]です。
[2] お客様のシステムに適用する場合、必ずこの値を変更してください。

3.2　動作シーケンス

この例の動作シーケンスを説明します。図3.1にマスタとスレーブが各1台で多機能タイマとしてMTU3を選択し、時刻”t(N) (i = 1, 2, 3,,,　N)”で故障を検出する例を示します。ここで、”t(N)”はn回目のEPTPCのPTPタイマ開始イベントが発生する時刻、”δ”は”t(N)”からの微小な遅延時間とします(δ <<　t(N) – t(N-1))。

(1) t(0)以前：マスタとスレーブで PTP プロトコルによる同期を開始する。

(2) 時刻 t(0)：PTP タイマ開始イベントの割り込みを設定する。

(3) 時刻 t(1):PTP タイマ開始イベントで MTU3 のカウントアップを開始するように ELC の接続設定をする。

(4) 時刻 t(2)：PTP タイマ開始イベントで MTU3 のカウントアップを開始する。

(5) 時刻 t(3)：MTU3 のカウントアップを停止する。

(6) 時刻 t(3) + δ:MTU3 のカウンタと前回の割り込み（時刻 t(2)で発生）から進んだ EPTPC のローカルクロックカウンタ値の差異の絶対値を比較する。そして、その差異が閾値以下であれば、故障は検出されなかったとする。

(7) 時刻 t(4)：PTP タイマ開始イベントで MTU3 のカウントアップを開始する。

(8) 時刻 t(5)：MTU3 のカウントアップを停止する。

(9) 時刻 t(5) + δ:MTU3 のカウンタと前回の割り込み（時刻 t(4)で発生）から進んだ EPTPC のローカルクロックカウンタ値の差異の絶対値を比較する。そして、その差異が閾値以下であれば、故障は検出されなかったとする。

(10) 時刻 t(N-1)：PTP タイマ開始イベントで MTU3 のカウントアップを開始する。

(11) 時刻 t(N)：MTU3 のカウントアップを停止する。

(12)時刻 t(N) + δ：MTU3 のカウンタと前回の割り込み（時刻 t(N-1)で発生）から進んだ EPTPC のローカルクロックカウンタ値の差異の絶対値を比較する。そして、その差異が閾値を越えていれば、故障として検出する。

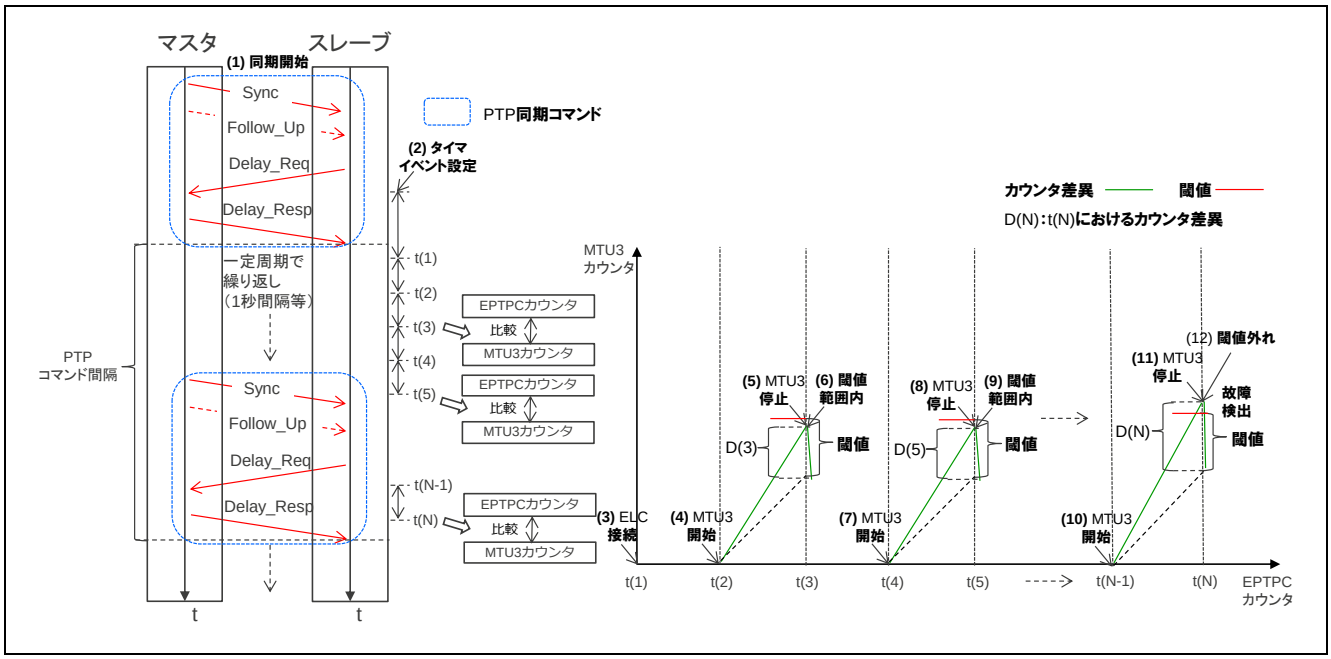

図3.1　処理概要

3.3　ソフトウェア動作フロー

この節では、サンプルプログラムのソフトウェア動作フローを説明します。
図 3.2と図 3.3はそれぞれドライバの初期設定と時刻同期の初期設定を示します。図 3.4は定常ルーチンの処理で、主に多機能タイマの診断を行います。図 3.5は多機能タイマのカウンタと EPTPC のローカルクロックカウンタを比較し、閾値から故障を判定する処理を示します。図 3.6は診断結果から診断の履歴を計算する処理を示します。図 3.7と図 3.8はそれぞれパラメータのデータフラッシュからのロード処理とデータフラッシュへの更新処理を示します。

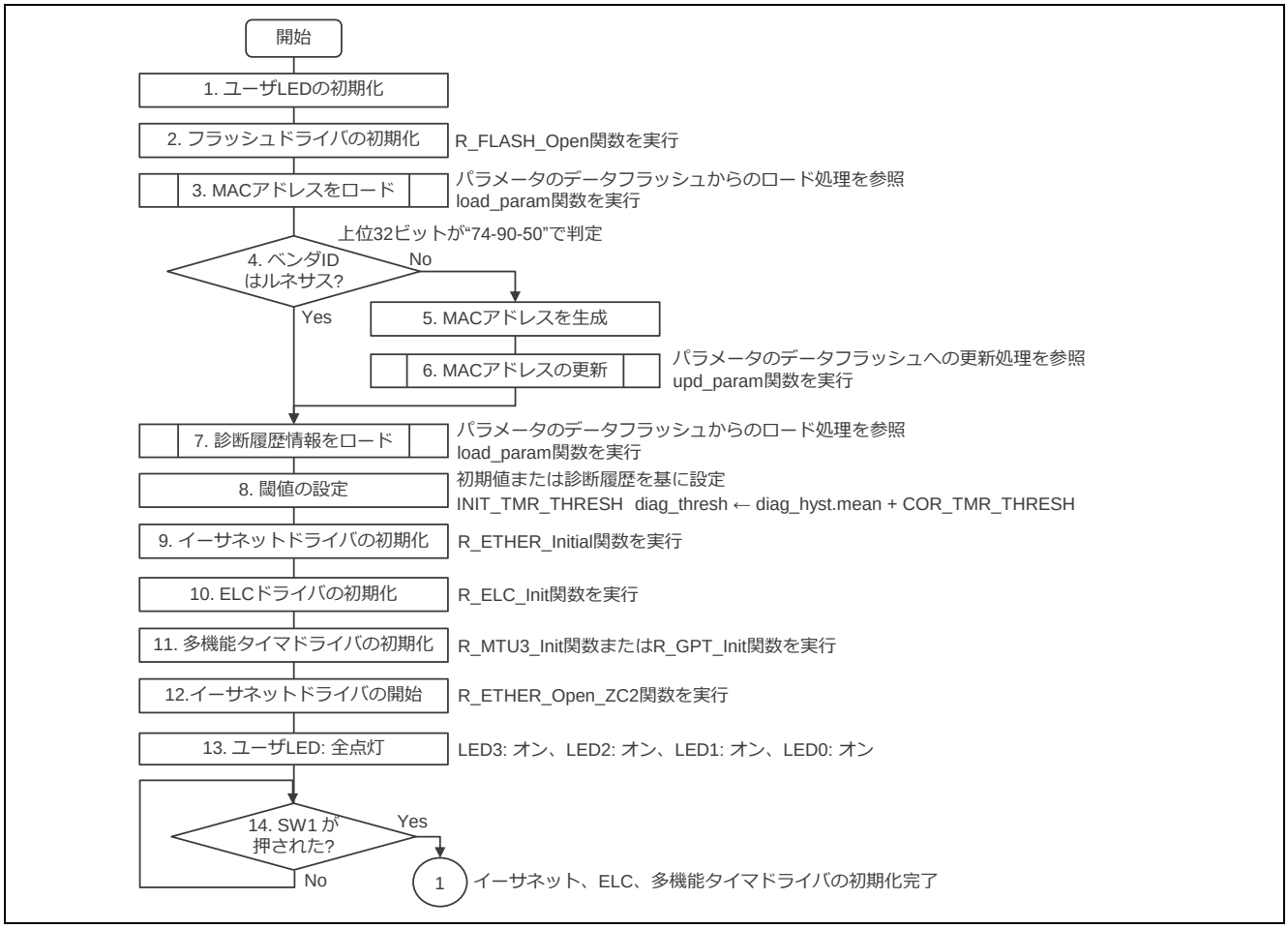

図3.2　ドライバの初期設定

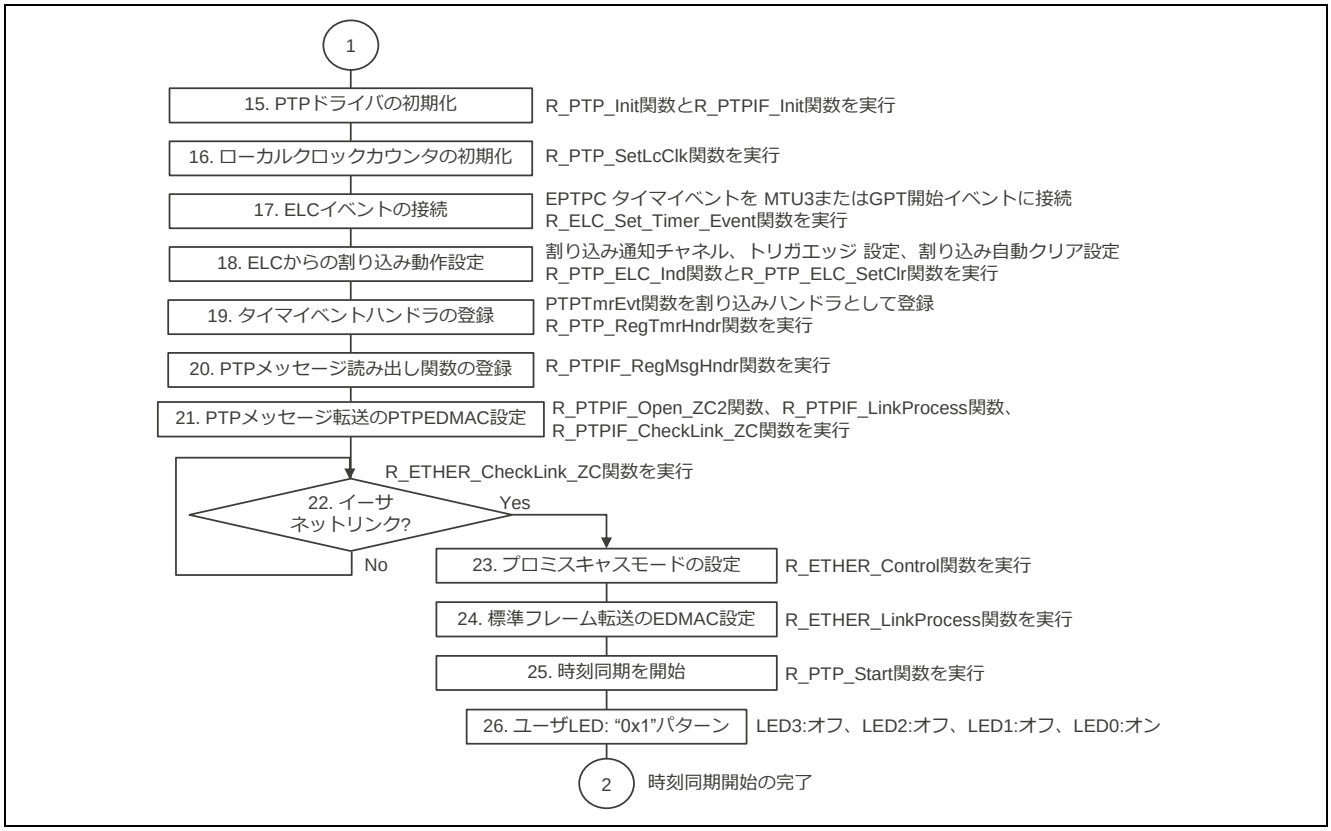

図3.3　時刻同期の初期設定

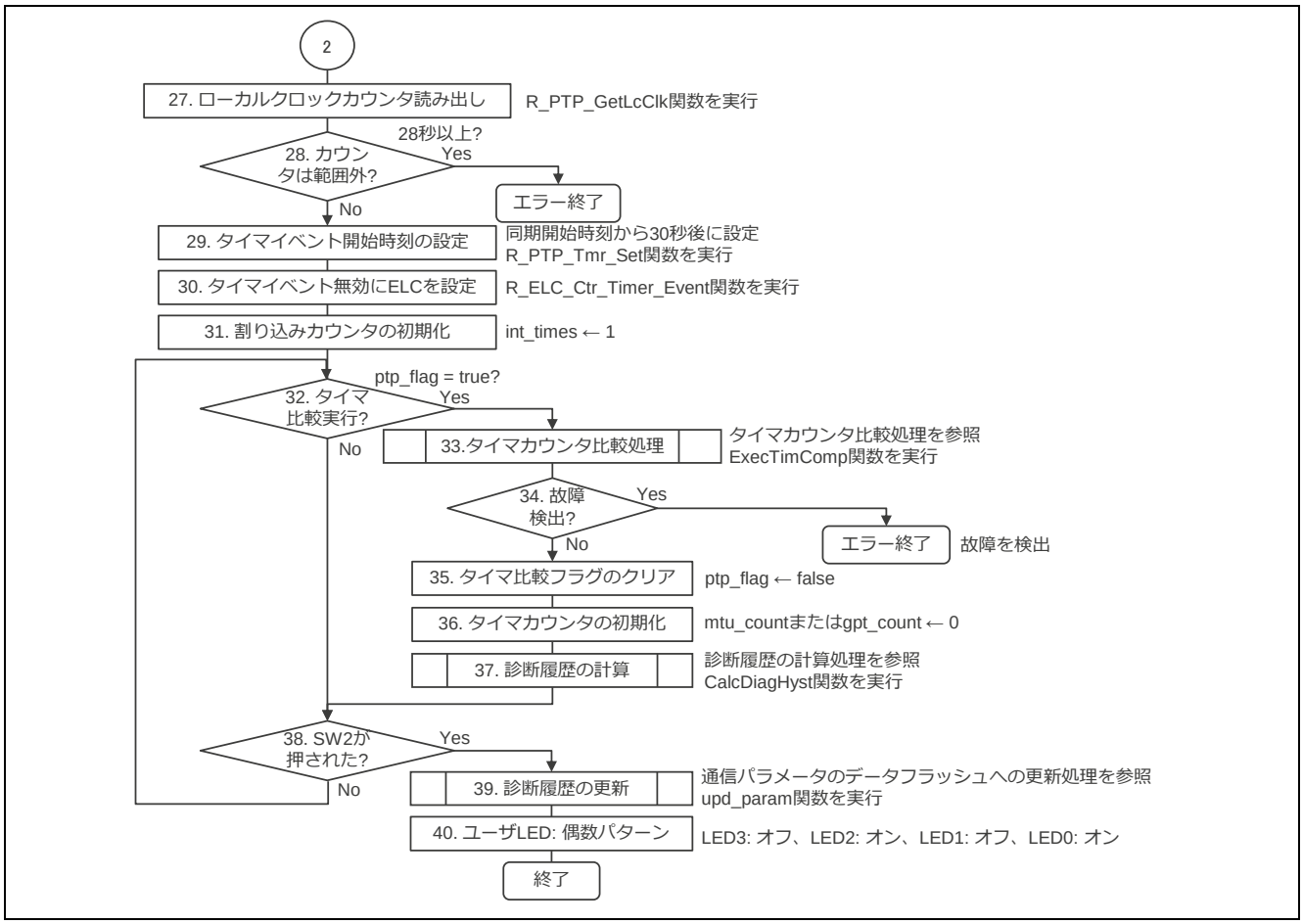

図3.4　多機能タイマ診断処理

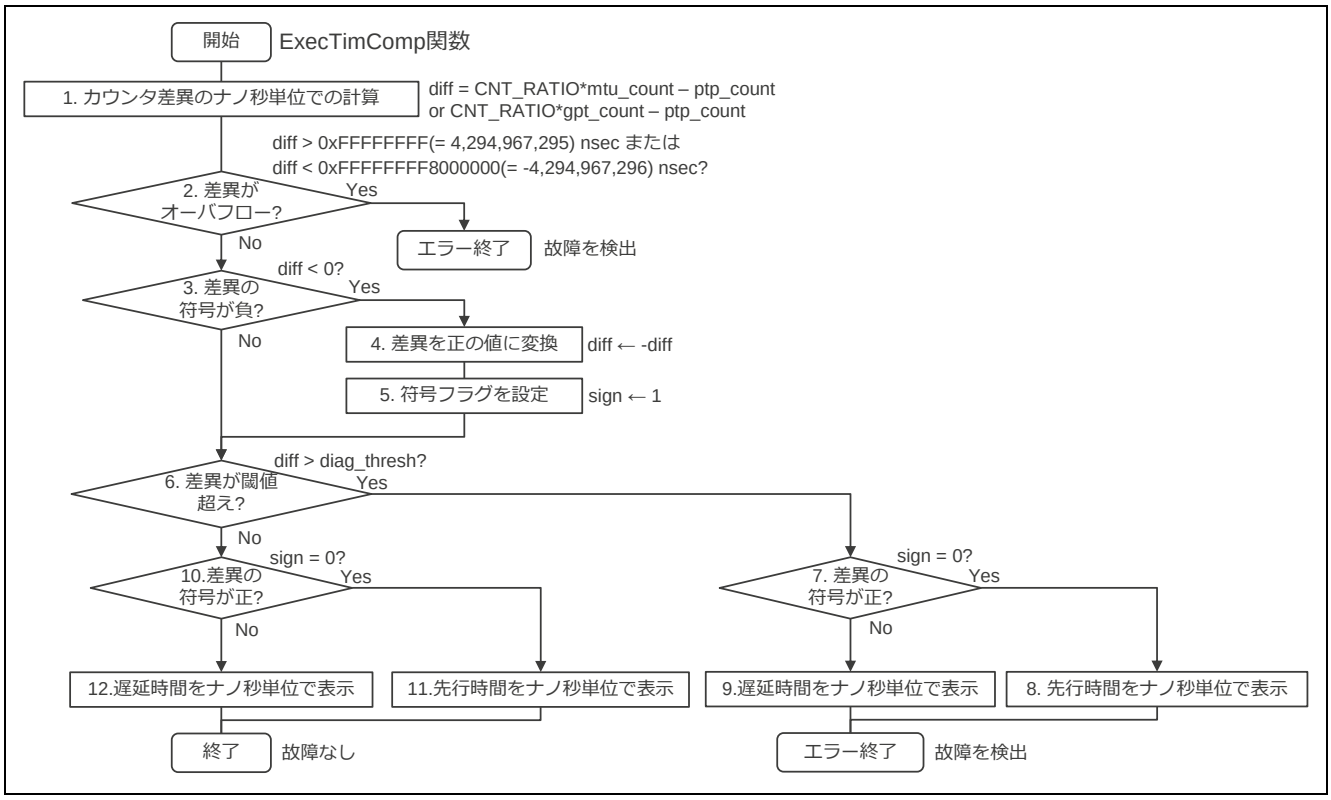

図3.5　多機能タイマ比較処理

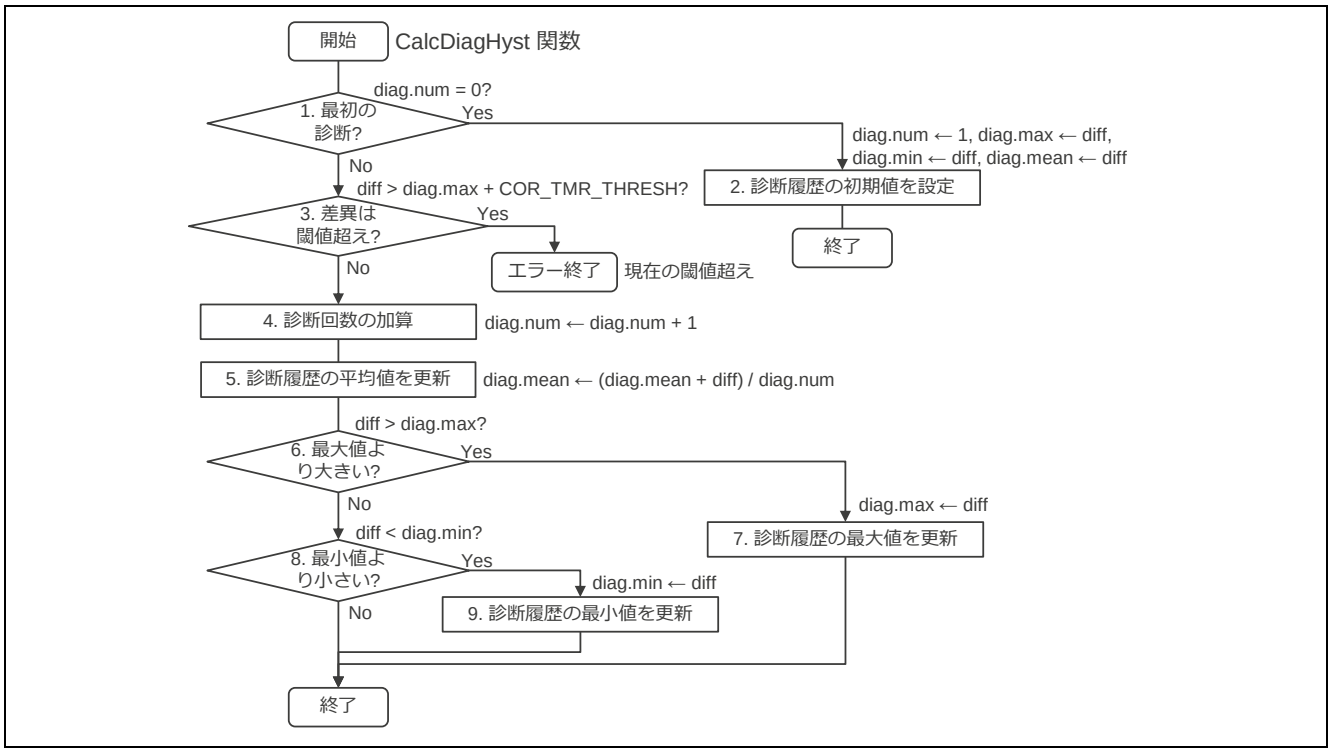

図3.6　診断履歴の計算

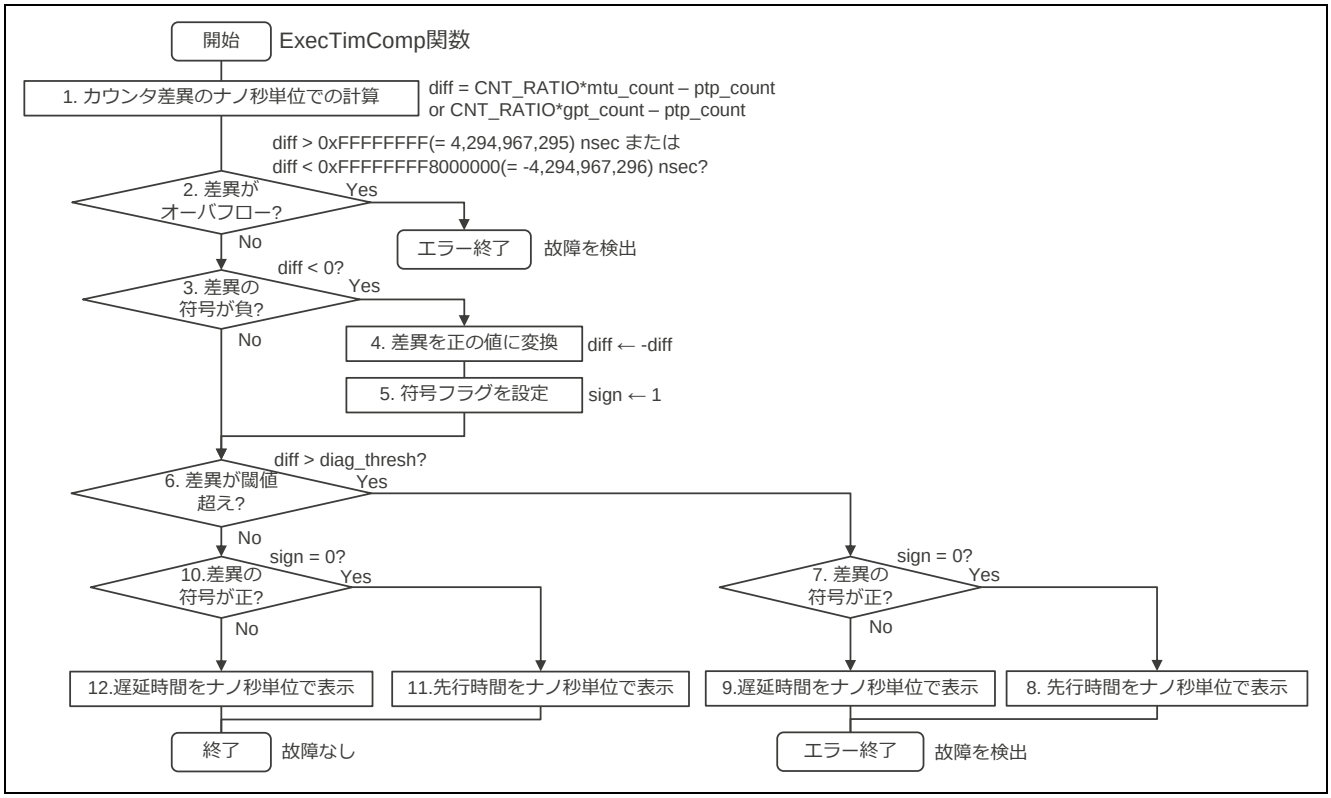

図3.7　パラメータロード処理

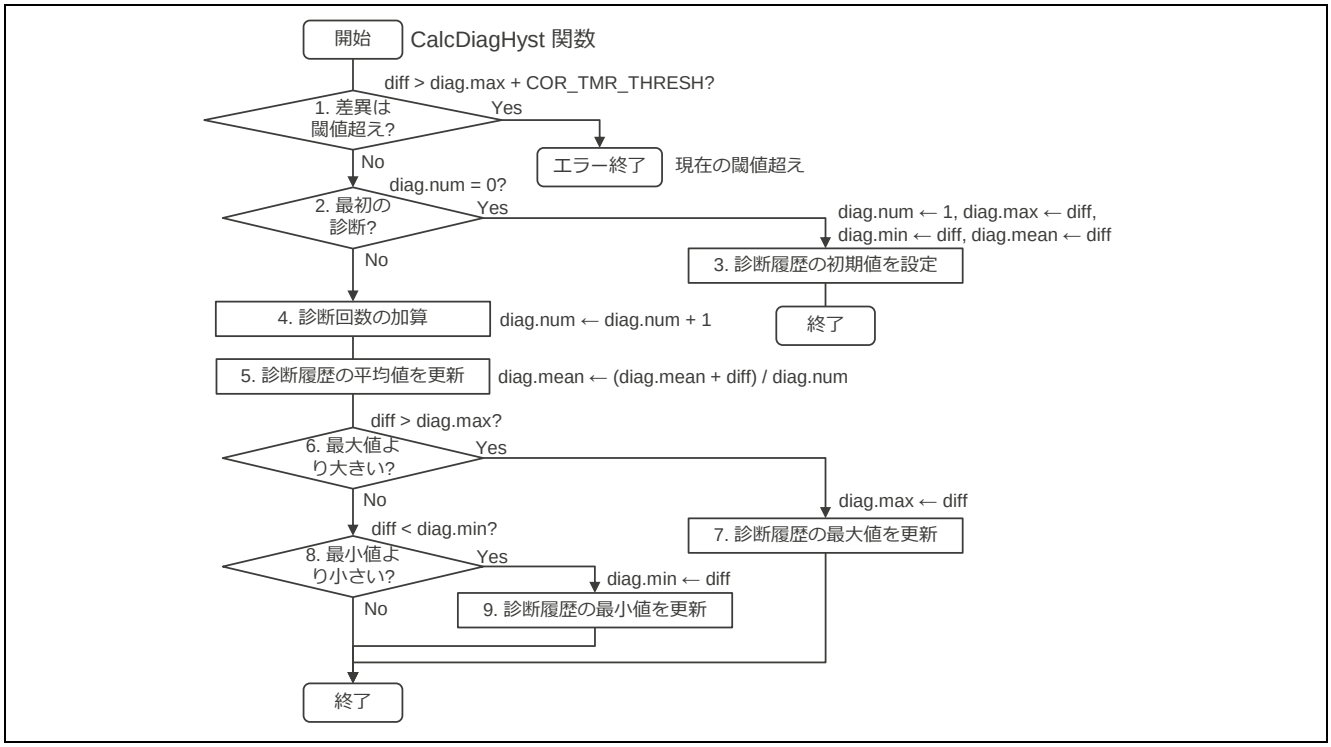

図3.8　パラメータ更新処理

3.4　結果表示例

故障を検出することなく動作を終了した場合、及び、故障を検出し動作を終了した場合の結果の出力を、それぞれ、図 3.9と図 3.10に示します。結果出力は、e2 studio の「Renesas デバッグ仮想コンソール」で参照できます。

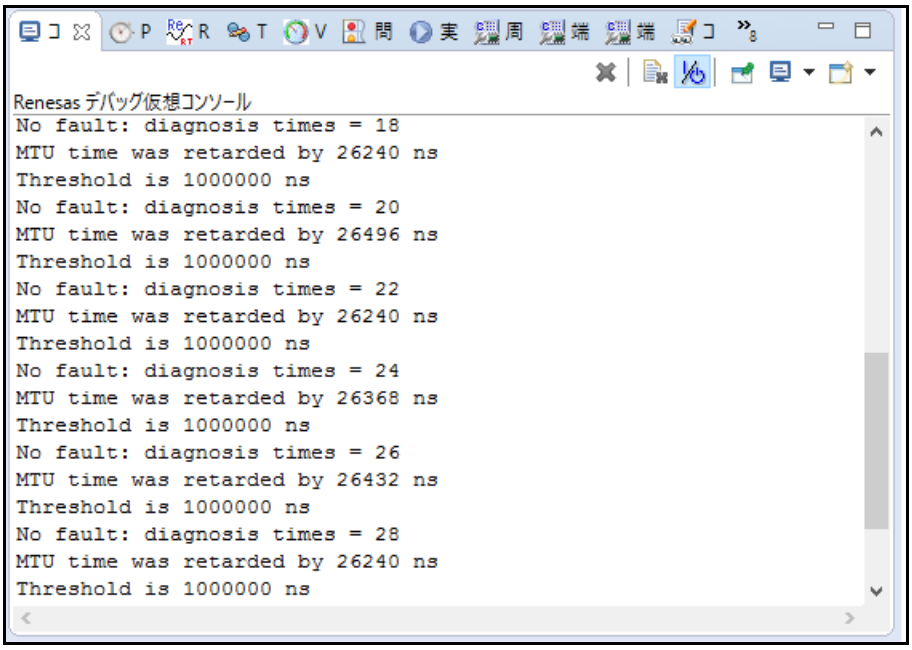

図3.9　結果出力（故障未検出）

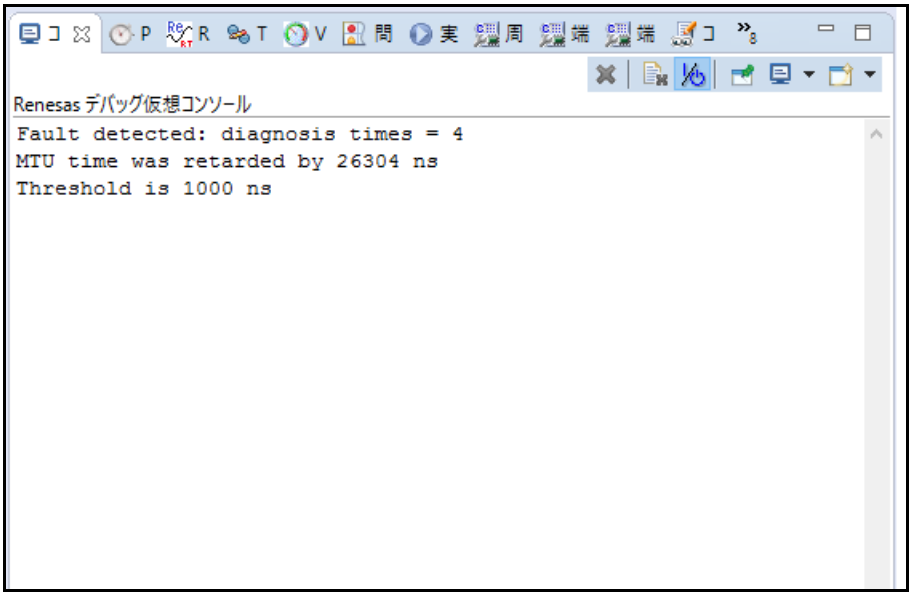

図3.10　結果出力（故障検出）

3.5　ボードの設定

RX64M/71M RSK ボードのジャンパを ETHERC から PHY にアクセスするチャネルにより、デフォルト設定から変更する必要があります。RX64M/71M RSK ボードの型名が R0K50564MC001BR または R0K5RX71MC010BR の場合、ジャンパ設定を図 3.11に示します。また、RX71M RSK ボード型名が R0K50571MC000BR の場合、ジャンパ設定を図 3.12に示します。

ジャンパ	LINK_CH = 1[1]（デフォルト設定）	LINK_CH = 0[2]	使用機能
J3	2-3	1-2	ETHERC ET0MDIO or ET1MDIO
J4	2-3	1-2	ETHERC ET0MDC or ET1MDC

[1] MODE_PORT = 1の場合、LINK_CH = 1 が選択されます。
[2] MODE_PORT = 0の場合、LINK_CH = 0 が選択されます。

図3.11　ジャンパ設定

ジャンパ	LINK_CH = 1[1]（デフォルト設定）	LINK_CH = 0[2]	使用機能
J13	2-3	1-2	ETHERC ET0MDIO or ET1MDIO
J9	2-3	1-2	ETHERC ET0MDC or ET1MDC

[1] MODE_PORT = 1の場合、LINK_CH = 1 が選択されます。
[2] MODE_PORT = 0の場合、LINK_CH = 0 が選択されます。

図3.12　ジャンパ設定

4. 参考資料

ユーザーズマニュアル: ハードウェア

RX64M グループユーザーズマニュアル ハードウェア編 Rev.1.10（R01UH0377JJ）
RX71M グループユーザーズマニュアル ハードウェア編 Rev.1.00（R01UH0493JJ）
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。

ユーザーズマニュアル: ソフトウェア

RX ファミリ RXv2 命令セットアーキテクチャ ユーザーズマニュアル ソフトウェア編（R01US0071JJ）
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。

技術情報/ニュース

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。

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2017.07.31	—	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. 未使用端子の処理
   【注意】未使用端子は、本文の「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。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】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. リザーブアドレス（予約領域）のアクセス禁止
   【注意】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、将来の機能拡張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。
4. クロックについて
   【注意】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。
5. 製品間の相違について
   【注意】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、内部ROM、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、ノイズ輻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、その他の不適切に使用しないでください。かかる改造、改変、複製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、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、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できません。たとえ、意図し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。また、当社製品および技術を、(1)核兵器、化学兵器、生物兵器等の大量破壊兵器およびこれらを運搬することができるミサイル（無人航空機を含みます。）の開発、設計、製造、使用もしくは貯蔵等の目的、(2)通常兵器の開発、設計、製造または使用の目的、または(3)その他の国際的な平和および安全の維持の妨げとなる目的で、自ら使用せず、かつ、第三者に使用、販売、譲渡、輸出、賃貸もしくは使用許諾しないでください。

   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. お客様の転売、貸与等により、本書（本ご注意書きを含みます。）記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は一切その責任を負わず、お客様にかかる使用に基づく当社への請求につき当社を免責いただきます。
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. 本資料に記載された情報または当社製品に関し、ご不明点がある場合には、当社営業にお問い合わせください。

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その総株主の議決権の過半数を直接または間接に保有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.3.0-1　2016.11)

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口　http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/