RX ファミリ

Trusted Memory 機能の使い方

要旨

Trusted Memory（以下、TM）機能は、コードフラッシュメモリの一部領域に暗号アルゴリズムの処理ソフトウェア、ノウハウを伴う機器制御処理ソフトウェア、有償のミドルウェアなどの秘匿性を高めたいソフトウェアを格納することで、第三者からの不正アクセスおよびソフトウェアの改ざんを防止します。

本アプリケーションノートでは、TM 対象領域にソフトウェアを格納し、TM 対象領域内のソフトウェアを使用する方法などの運用例を説明します。

対象デバイス

- RX64M グループ
- RX71M グループ
- RX65N グループ
- RX651 グループ
- RX66T グループ
- RX72T グループ
- RX72M グループ
- RX72N グループ
- RX66N 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合わせて変更し、十分評価してください。

目次

1.　仕様

本アプリケーションノートでは、TM 機能を有効にする第 1 ユーザと、TM 機能が有効にされたデバイスを使用する第 2 ユーザを想定しています。第 1 ユーザが TM 対象領域にプログラムを書き込み、その後第 2 ユーザが TM 対象領域以外の書き込みを行います。

プログラムの書き込みおよび TM 機能の有効/無効の切り換えには、フラッシュ書き込みツール Renesas Flash Programmer(以下、RFP)を使用します。

表 1.1 に　使用する周辺機能と用途、図 1.1 に概要図を示します。

表 1.1　使用する周辺機能と用途

周辺機能	用途
Trusted Memory 機能	コードフラッシュメモリの TM 対象領域に対する不正リード防止、追加プログラミングの防止

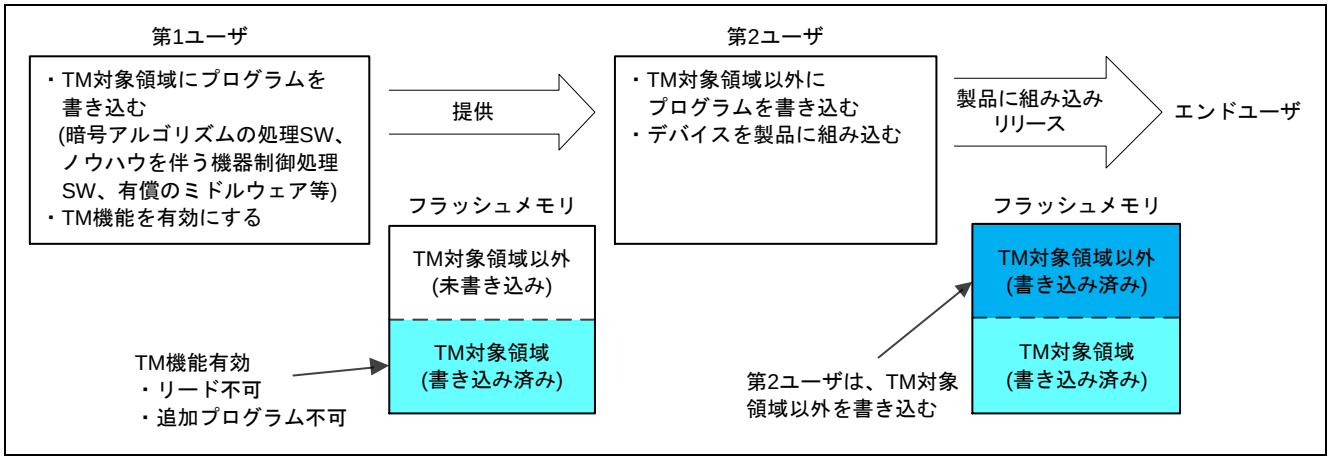


図 1.1　概要図

2. TM 機能の運用例

第 1 ユーザ(TM 対象領域内のプログラム提供者)はアプリケーション開発、TM 対象領域内のプログラムが使用する変数および関数テーブルのライブラリファイル(.lib)を作成します。アプリケーション開発後、RFP を使用し、デバイスの TM 対象領域へのプログラム書き込みと TM 機能を有効にします。

TM 対象領域のプログラム仕様、サンプルコードのセクション情報、作成したライブラリファイル(.lib)を第 2 ユーザへ提供してください。

第 2 ユーザ(アプリケーション開発者)は、提供されたプログラム仕様、セクション情報、ライブラリファイル(.lib)を使用し、アプリケーションを開発します。アプリケーション開発後、e2 studio でアプリケーションのデバッグ、RFP で TM 対象領域以外にプログラムを書き込みます。

プログラムを書き込んだデバイスを製品に組み込みエンドユーザに提供します。

図 2.1 開発フローを示します。

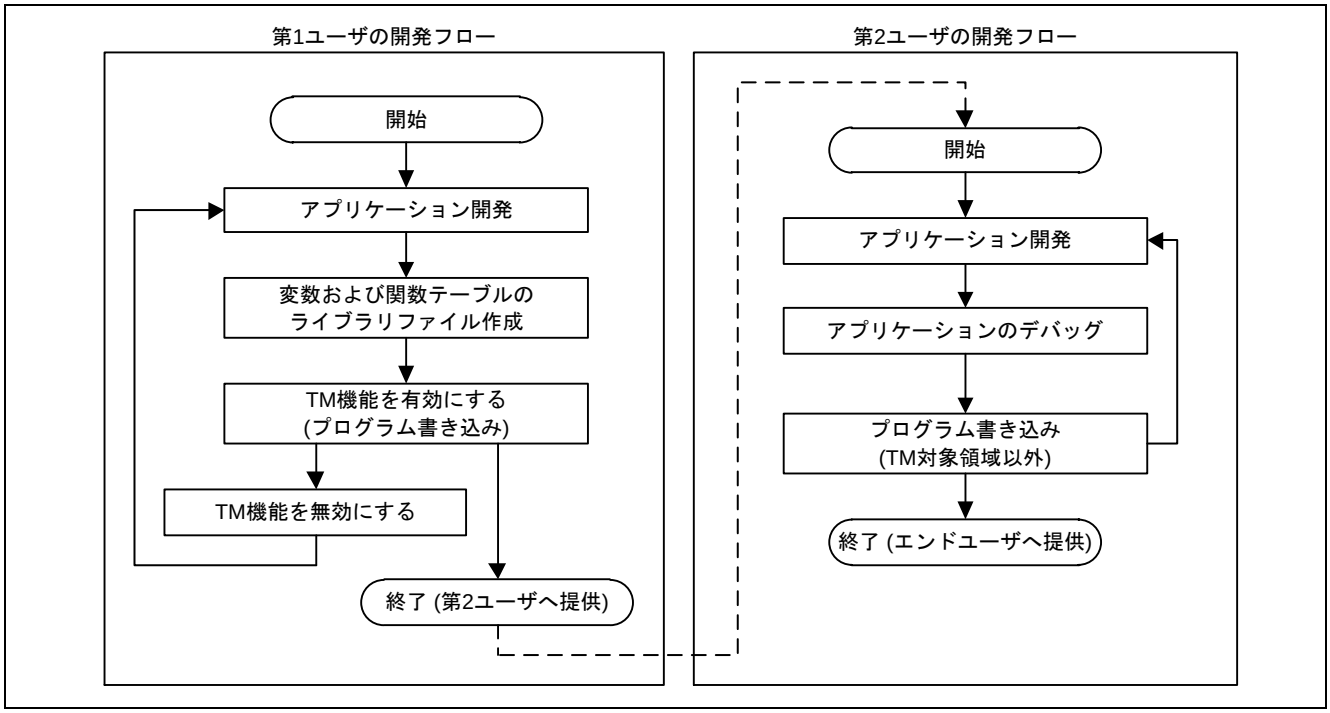


図 2.1 開発フロー

2.1 e2 studio の新規プロジェクト

第 1 ユーザおよび第 2 ユーザは、e2 studio の新規プロジェクト作成の際にセクション設定、ROM オプション設定、セクション初期化を行ってください。

2.1.1 セクション設定

第 1 ユーザ開発と第 2 ユーザ開発で使用する領域を明確にするため、第 1 ユーザの TM 対象領域のプログラムが使用するセクションと第 2 ユーザがアプリケーションを開発するセクションを分けて設定してください。

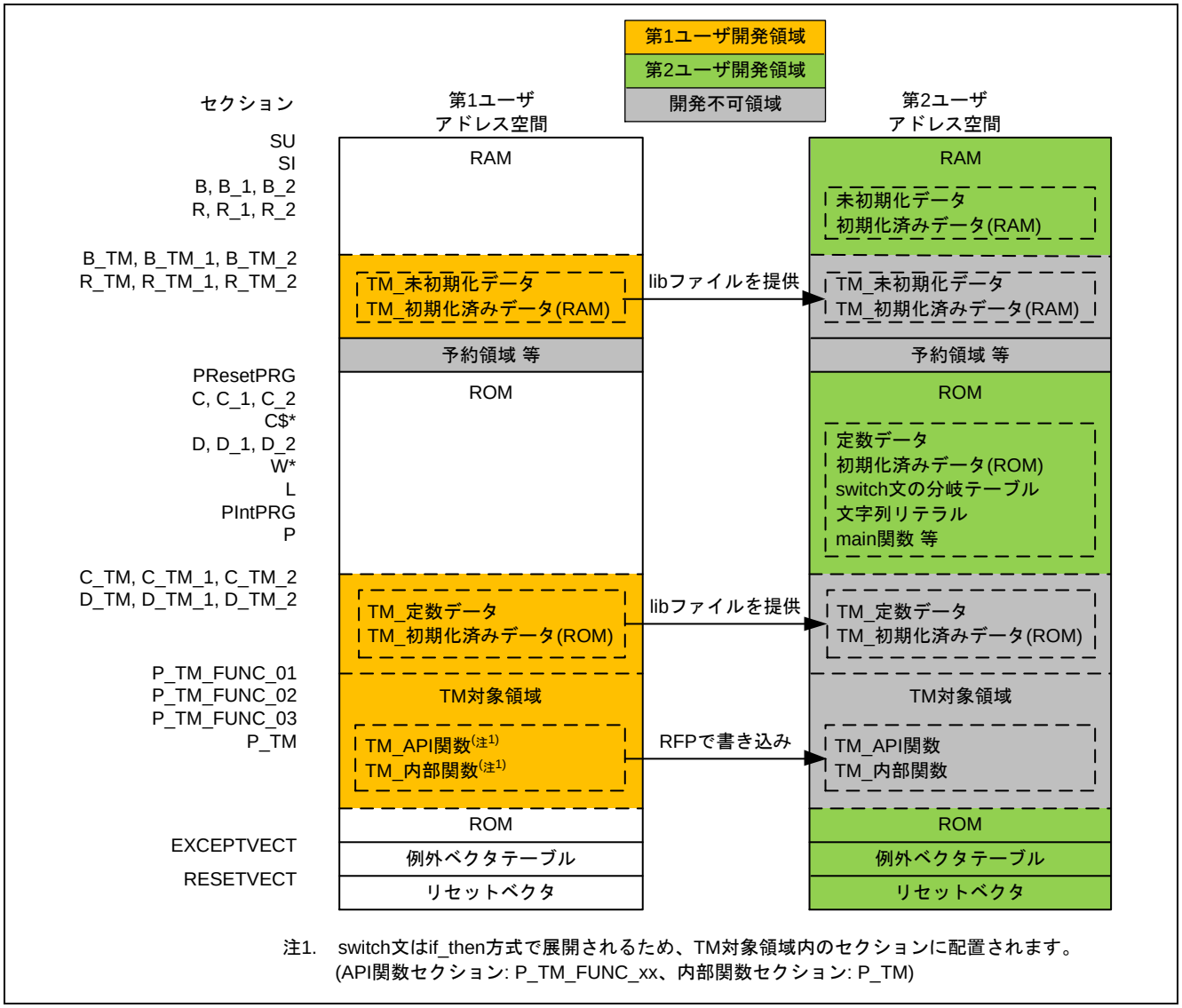
ただし、セクション L、W は#pragma section でセクション名を変更できないため使用しません。switch 文を使用する場合は table 方式(switch 分岐テーブル領域に配置)ではなく、if_then 方式(プログラム領域に配置)で展開する設定にしてください。

この設定をすることで、switch 文はプログラム領域に配置され、デバイスに書き込まれます。

switch 文のコード展開方式の設定は、第 1 ユーザのみが行います。第 1 ユーザは使用セクションの情報を第 2 ユーザへ提供してください。

第 2 ユーザは、提供された第 1 ユーザ使用セクションの設定してください。セクション設定の際にアドレスがずれないように注意してください。また、第 1 ユーザが使用している領域を使用しないように第 2 ユーザ使用セクションを設定してください。

本サンプルコードのセクションは図 2.2 セクション図を参考にしてください。



注1.　switch文はif_then方式で展開されるため、TM対象領域内のセクションに配置されます。
(API関数セクション: P_TM_FUNC_xx、内部関数セクション: P_TM)

図 2.2　セクション図

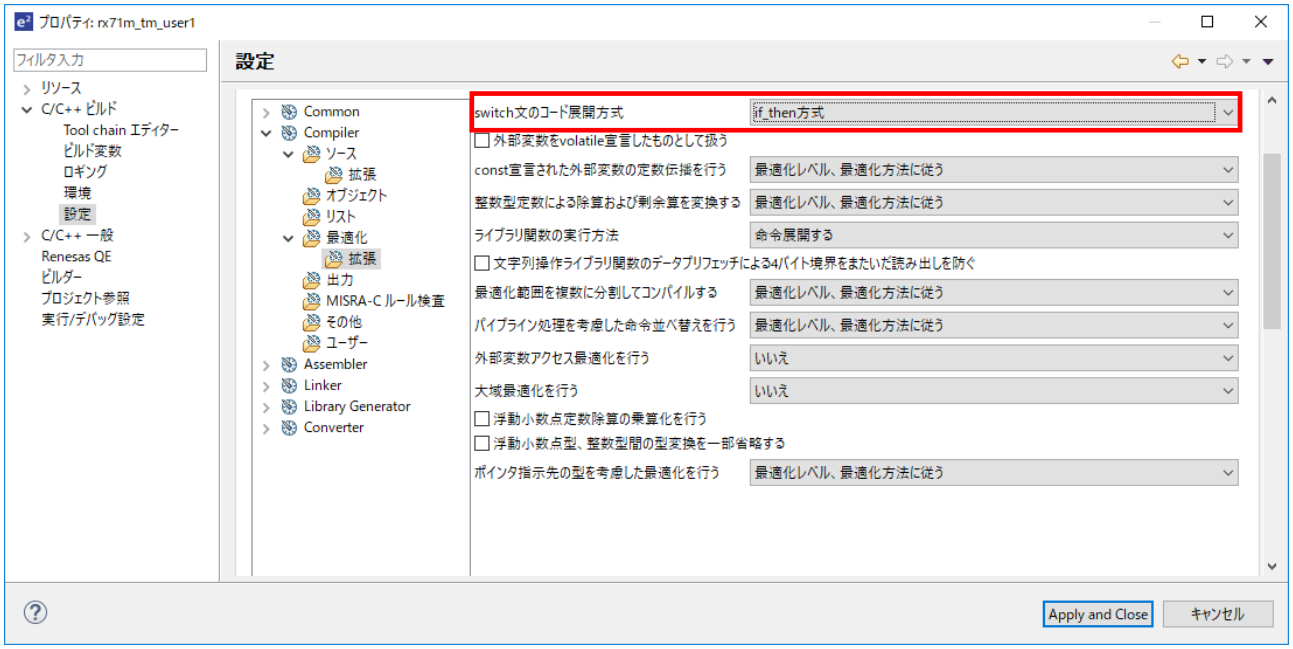
switch 文のコード展開方式の設定方法を以下に示します。

(1)e2 studio で対象プロジェクトを右クリック、「プロパティ(R)」を選択。

(2)「C/C++ビルド > 設定 > ツール設定タブ > Compiler > 最適化 > 拡張」を選択。

(3)switch 文のコード展開方式を if_then 方式に設定。

(4)「適用(L)」をクリック。



2.1.2 ROM オプション設定

ROM オプションは、初期化データ領域の ROM 用、RAM 用領域を確保し、ROM セクション内定義シンボルを RAM セクション内アドレスになるようにリロケーションします。

デフォルトで ROM オプション設定がされているのは、セクション D、R のみです。新規プロジェクト作成の際は、第 1 ユーザが使用している初期化データ領域(D_TM、R_TM)の ROM オプション設定を行ってください。

ROM オプションの設定方法を以下に示します。

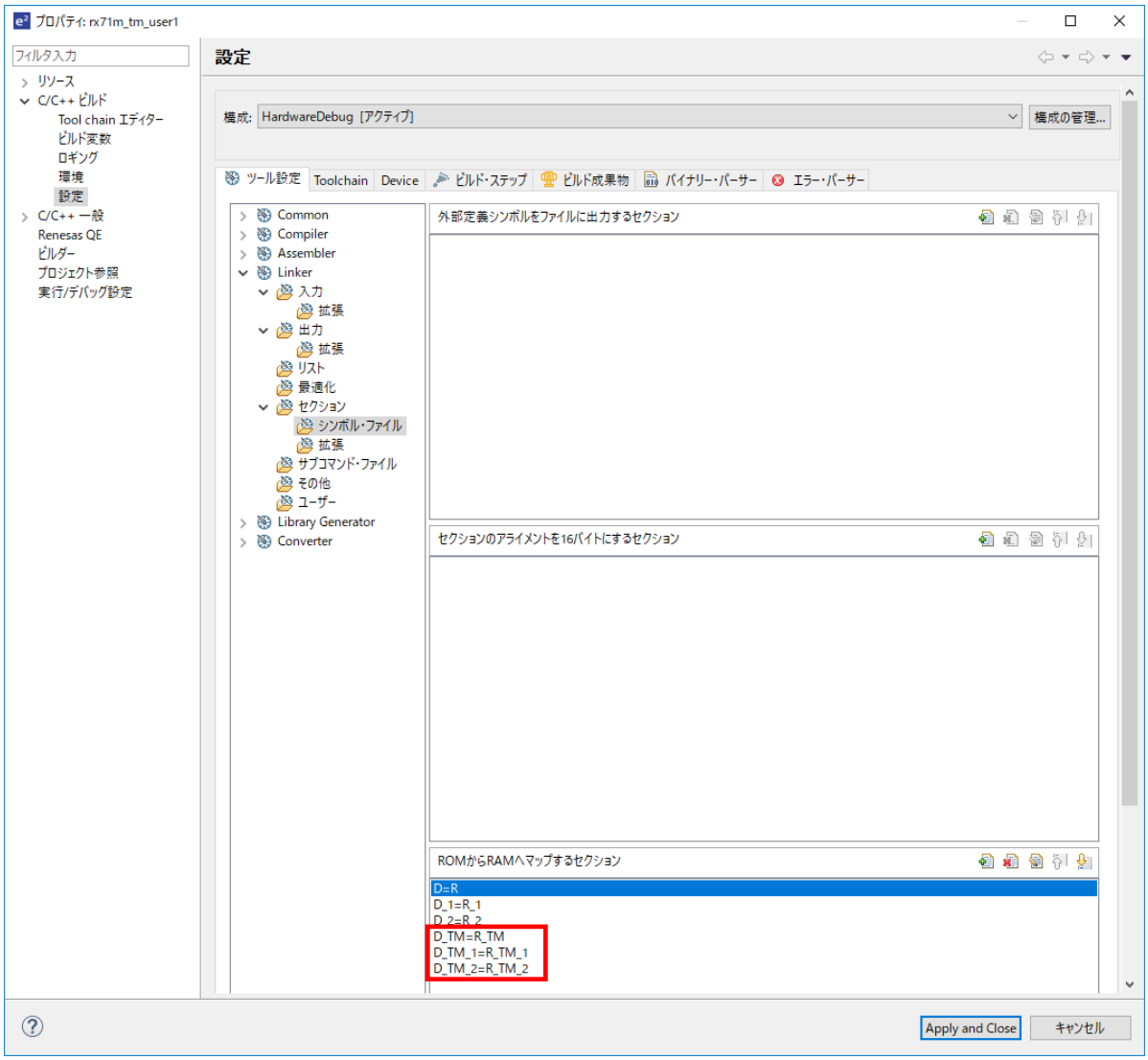
(1)e2 studio で対象プロジェクトを右クリック、「プロパティ(R)」を選択。

(2)「C/C++ビルド > 設定 > ツール設定タブ > Linker > セクション > シンボル・ファイル」を選択。

(3)ROM から RAM にマップするセクションを追加。

(本アプリケーションノートでは、D_TM=R_TM、D_TM_1=R_TM_1、D_TM_2=R_TM_2 を追加)

(4)「適用(L)」をクリック。



ROM オプションの詳細は、最新の RX ファミリ C/C++コンパイラ(CC-RX)ユーザーズマニュアルを参照してください。

2.1.3 セクション初期化

e2 studio の自動生成ファイル”resetprg.c”の_INITSCT 関数が呼び出されると、未初期化データセクションはゼロ初期化、初期化データセクションは、ROM 領域から RAM 領域へコピーされます。

初期化対象のセクションは、ユーザがセクション初期化用テーブル(DTBL、BTBL)へ記述する必要があります。e2 studio の自動生成ファイル”dbsct.c”のセクション初期化用テーブルに以下のプログラムを追加してください。

セクション初期化の詳細は、最新の RX ファミリ C/C++コンパイラ(CC-RX)ユーザーズマニュアルを参照してください。

```
#pragma section C C$DSEC
extern const struct {
    _UBYTE *rom_s;        /* Start address of the initialized data section in ROM */
    _UBYTE *rom_e;        /* End address of the initialized data section in ROM   */
    _UBYTE *ram_s;        /* Start address of the initialized data section in RAM */
}   _DTBL[] = {
    { __sectop("D"), __secend("D"), __sectop("R") },
    { __sectop("D_2"), __secend("D_2"), __sectop("R_2") },
    { __sectop("D_1"), __secend("D_1"), __sectop("R_1") },
    { __sectop("D_TM"), __secend("D_TM"), __sectop("R_TM") },
    { __sectop("D_TM_2"), __secend("D_TM_2"), __sectop("R_TM_2") },
    { __sectop("D_TM_1"), __secend("D_TM_1"), __sectop("R_TM_1") }
};
#pragma section C C$BSEC
extern const struct {
    _UBYTE *b_s;          /* Start address of non-initialized data section */
    _UBYTE *b_e;          /* End address of non-initialized data section */
}   _BTBL[] = {
    { __sectop("B"), __secend("B") },
    { __sectop("B_2"), __secend("B_2") },
    { __sectop("B_1"), __secend("B_1") },
    { __sectop("B_TM"), __secend("B_TM") },
    { __sectop("B_TM_2"), __secend("B_TM_2") },
    { __sectop("B_TM_1"), __secend("B_TM_1") }
};

#pragma section

/*
** CTBL prevents excessive output of L1100 messages when linking.
** Even if CTBL is deleted, the operation of the program does not change.
*/
_UBYTE * const _CTBL[] = {
    __sectop("C_1"), __sectop("C_2"), __sectop("C"),
    __sectop("W_1"), __sectop("W_2"), __sectop("W"),
    __sectop("L"), __sectop("SU"),
    __sectop("C$DSEC"), __sectop("C$BSEC"),
    __sectop("C$INIT"), __sectop("C$VTBL"), __sectop("C$VECT"),
    __sectop("C_TM_1"), __sectop("C_TM_2"), __sectop("C_TM")
};
```

2.2 第 1 ユーザの開発手順

2.2.1 アプリケーション開発

通常、関数呼び出しにはシンボルを使用しますが、第 2 ユーザには関数のシンボル情報が提供されないため、TM 対象領域の関数にアクセスすることはできません。第 2 ユーザが TM 対象領域にアクセスするためには、関数のアドレス情報が必要となります。

第 1 ユーザは TM 対象領域の関数アドレス情報を格納した関数テーブルを用意します。関数テーブルは定数領域に配置されるため、ライブラリファイル(.lib)で第 2 ユーザに提供します。第 2 ユーザは関数テーブルを使用して、TM 対象領域内の関数を呼び出すことができます。

TM 対象領域の API 関数は個別にセクションと番地の指定をし、関数テーブルには API 関数のセクション先頭アドレスを格納してください。ライブラリファイル作成時、変数と関数テーブル以外を削除するので、関数テーブルに関数シンボルは使用できません。

関数テーブル作成例を以下に示します。

```
void (*const tm_func_table1[])(void) = {
//;R_tm_func_01
    (void (*)())(__sectop("P_TM_FUNC_01")),
//;R_tm_func_02
    (void (*)())(__sectop("P_TM_FUNC_02")),
};
```

TM 機能有効時、TM 対象領域に対するデータアクセスをすると“0”がリードされるため、TM 対象領域にはセクション P のみを配置してください。データアクセスする領域(セクション C など)は、TM 対象領域外に用意してください。

2.2.2 変数および関数テーブルのライブラリファイル作成

第 2 ユーザが TM 対象領域にアクセスするには、関数のアドレス情報が必要です。そのため、第 1 ユーザは、TM 対象領域で使用する変数および関数テーブルのライブラリファイル(.lib)を作成、提供します。

ライブラリファイルは、サンプルコード<第 2 ユーザへ提供する lib ファイル作成プロジェクト>で global 変数、const 変数、TM 対象領域の関数テーブル以外を削除した tm_api.c ファイルをビルドすることで作成します。

2.2.3 TM 機能を有効にする

RFP を使用して TM 機能を有効にします。TM 機能有効と TM 対象領域へのプログラム書き込みは同時に行います(注 1)。

第 1 ユーザは、TM イネーブルフラグレジスタ(TMEF レジスタ)、TM 識別データレジスタ(TMINF レジスタ)、TM 対象領域のコード以外を削除した mot ファイルを作成し、対象デバイスに書き込みます。

TM 機能は、TMEF レジスタの TMEF ビットを 000b に設定することで有効になります(注 2)。TMINF レジスタには TM 対象領域のプログラムを識別できるコードを格納できます。

TMEF レジスタおよび TMINF レジスタを設定する場合、e2 studio の自動生成ファイル”vecttbl.c”の以下を修正してください。

RX64M、RX71M、RX66T、RX72T の場合

```
#pragma address __TMEFreg=0x00120048          // TMEF register
const unsigned long __TMEFreg = 0xf8ffffff;
```

```
#pragma address __TMINFreg=0x00120060         // TMINF register
const unsigned long __TMINFreg = 0xffffffff;
```

RX65N（リニアモード）、RX651（リニアモード）、RX72M（リニアモード）、RX72N（リニアモード）、RX66N（リニアモード）の場合

```
#pragma address __TMEFreg=0xfe7f5d48          // TMEF register
const unsigned long __TMEFreg = 0xf8ffffff;
```

```
#pragma address __TMINFreg=0xfe7f5d10         // TMINF register
const unsigned long __TMINFreg = 0xffffffff;
```

デュアルバンク機能を搭載する製品は、コードフラッシュメモリを 1 つの領域として扱うリニアモードと、2 つのバンク領域として扱うデュアルモードとを切り替えることができます。デュアルバンク機能を搭載する製品を以下に示します。

・RX65N グループ(コードフラッシュメモリ容量が 1.5M バイト以上の製品)

・RX651 グループ(コードフラッシュメモリ容量が 1.5M バイト以上の製品)

・RX72M グループ、RX72N グループ、RX66N グループ

デュアルモードで TM 機能を有効にする場合は 2.2.5(3)「デュアルモードで TM 機能を有効にする方法」を参照してください。

TM 機能を有効する方法の詳細は、使用する製品の ユーザーズマニュアル ハードウェア編 「フラッシュメモリ」章の「TM 機能を有効にする方法」を参照してください。

RFP の詳細は、最新の Renesas Flash Programmer フラッシュ書き込みソフトウェアユーザーズマニュアルを参照してください。

注1. セルフプログラミングを使用し、内部から TM 機能を有効にすることも可能です。

注2. e2 studio 環境(ダウンロード)では、TM 機能を有効にできません。RFP で書き込みを行ってください。

2.2.4 TM 機能を無効にする

RFP を使用すると、TM 対象領域を含むフラッシュメモリを全て消去し、TM 機能を無効にすることができます。

RFP の詳細は、最新の Renesas Flash Programmer フラッシュ書き込みソフトウェアユーザーズマニュアルを参照してください。

2.2.5 デュアルモードで TM 機能を使用する場合

デュアルモードで TM 機能を使用する場合、第 2 ユーザが起動バンク切り替えをすると、バンク 1 の TM 対象領域内の API 関数が呼び出されるため、第 1 ユーザはバンク 0 の TM 対象領域をバンク 1 の TM 対象領域へコピーしてデバイスにプログラムしておく必要があります。また、第 1 ユーザはあらかじめデュアルモードに設定しておく必要があります。

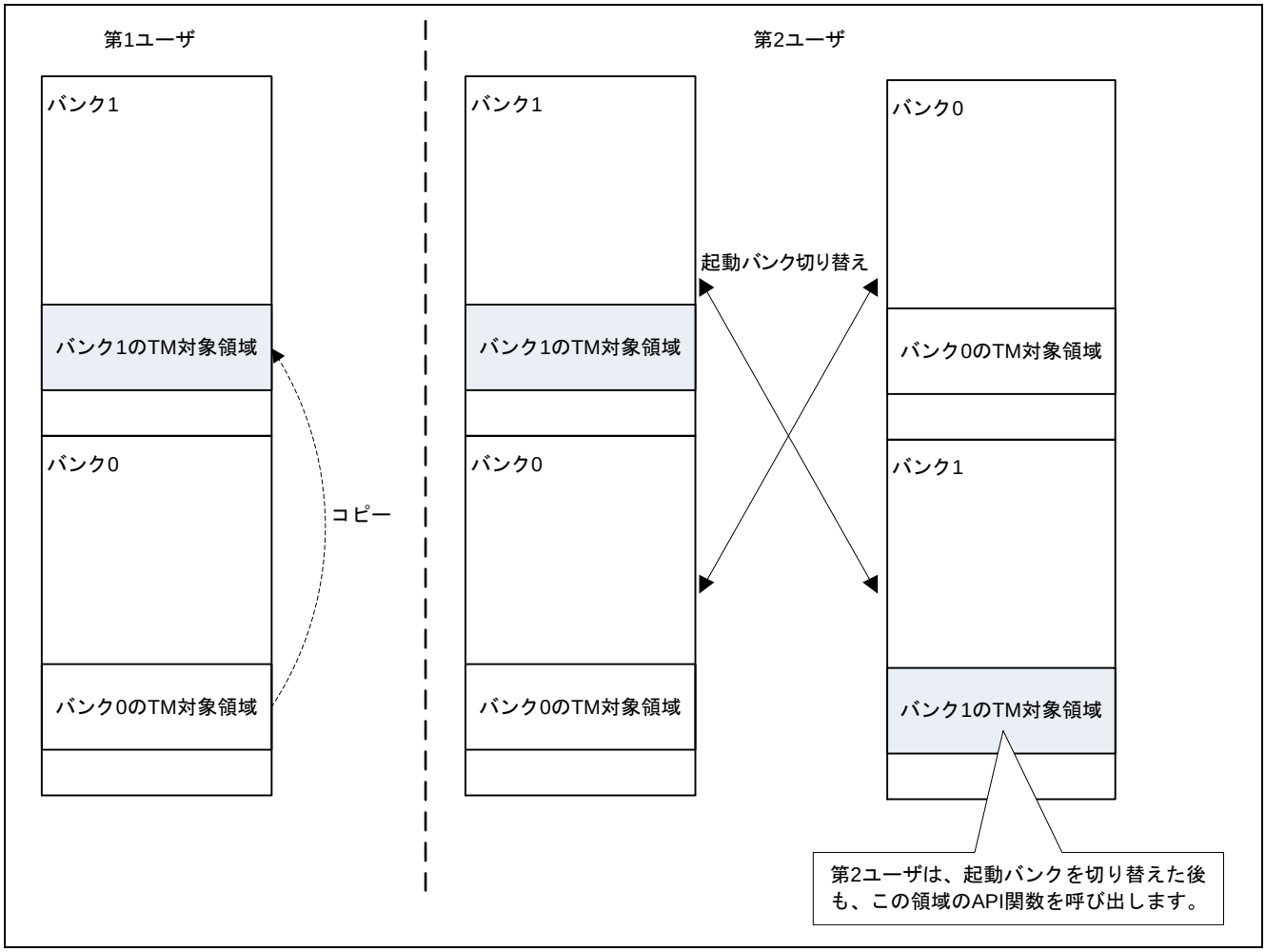


図 2.3 バンク 0 の TM 対象領域とバンク 1 の TM 対象領域

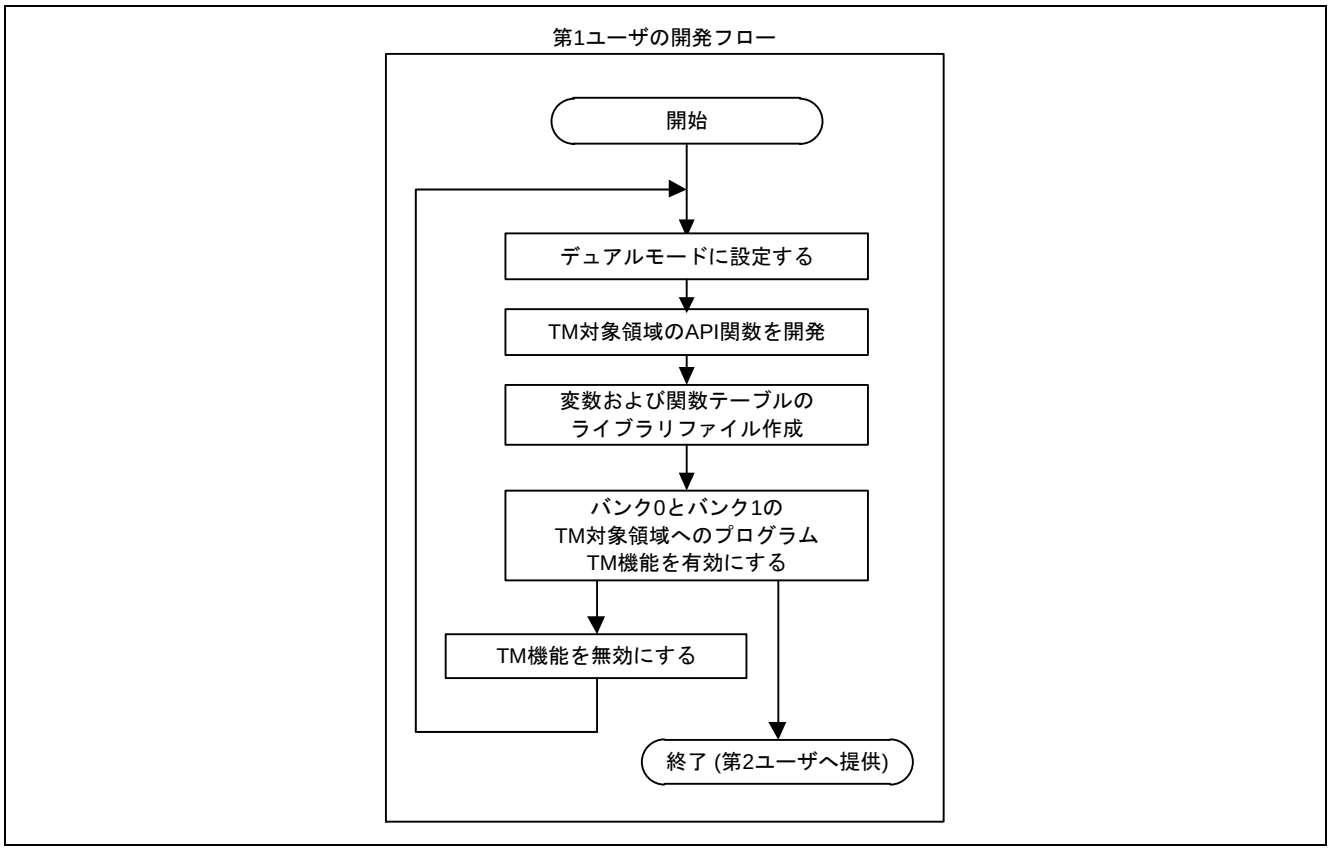


図 2.4 デュアルモードで TM 機能を使用する場合の開発フロー

(1) デュアルモードへの設定

第 1 ユーザは、オプション設定メモリの MDE レジスタの BANKMD ビットを 000b に設定した mot ファイルを作成し、RFP を使用してプログラム後、リセットすることで、デバイスをデュアルモードに設定します。

TM 対象領域へのプログラムと TM 機能の有効化の前にデュアルモードに設定してください。

MDE レジスタの BANKMD ビットを設定するためには、自動生成ファイル”vecttbl.c”を以下のように修正し、RFP を使用してオプション設定メモリへプログラムしてください。

```
#pragma address __MDEreg=0xfe7f5d00           // MDE register (Single Chip Mode)
#ifdef __BIG
    const unsigned long __MDEreg = 0xffffff88;    // big
#else
    const unsigned long __MDEreg = 0xffffff8f;    // little
#endif
```

(2) TM 対象領域のコピー

第 1 ユーザはバンク 0 の TM 対象領域をバンク 1 の TM 対象領域へコピーした mot ファイルを作成し、RFP を使用してバンク 1 の TM 対象領域へプログラムします。

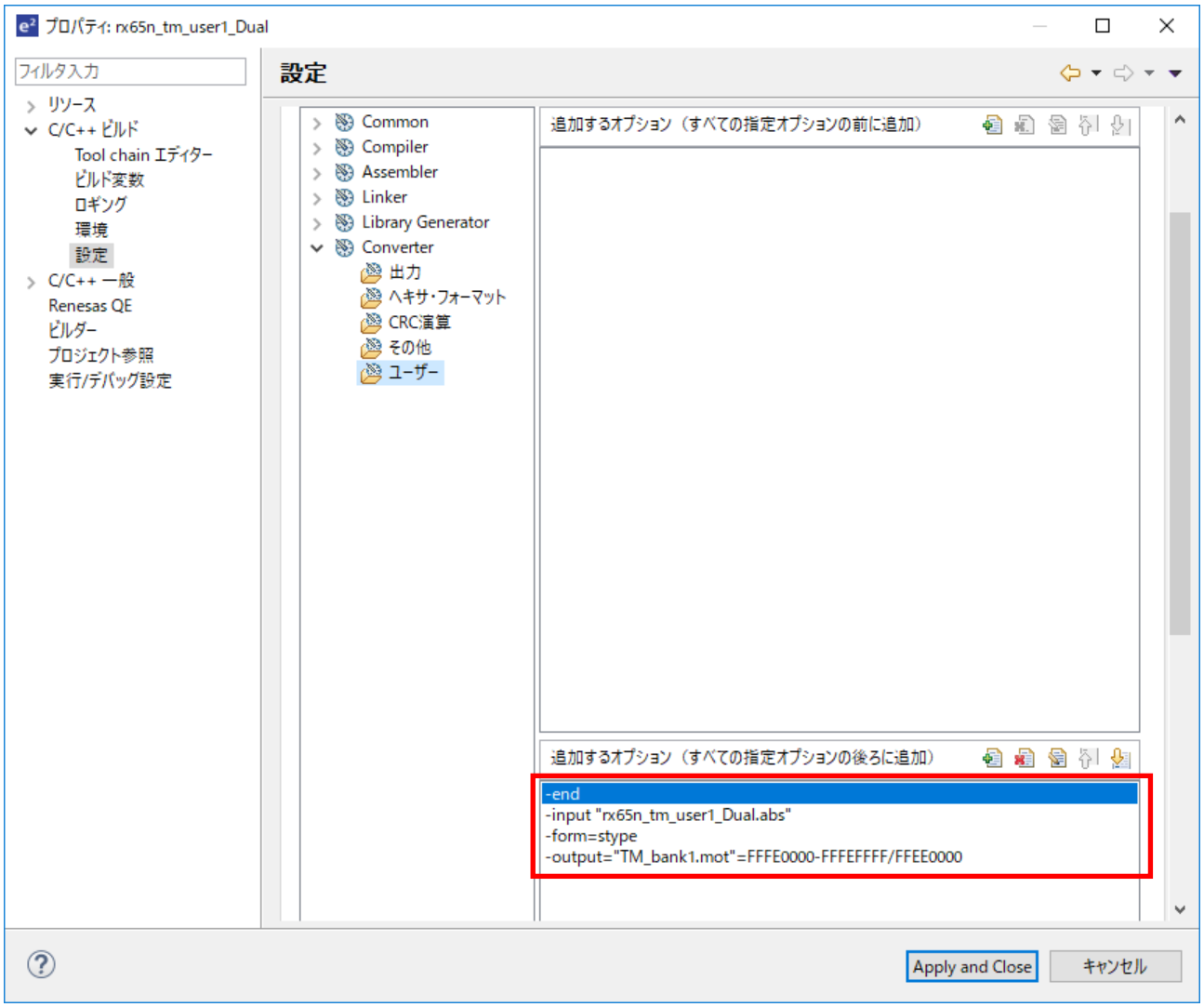
以下に、TM 対象領域のコピーと mot ファイルへ出力する設定を追加する方法を示します。

(1)e2 studio で対象プロジェクトを右クリック、「プロパティ(R)」を選択。

(2)「C/C++ビルド > 設定 > ツール設定タブ > Converter > ユーザー」を選択。

(3)オプションを追加。

(4)「適用」をクリック。



以下に追加するオプションの内容を示します。

表 2.1 追加するオプションの内容

追加するオプション	内容
-end	-end より前に設定されたリンクオプション列を実行します。
-input "rx65n_tm_user1_Dual.abs"	入力ファイルを指定します。この例ではサンプルコードの rx65n_tm_user1_Dual が出力した "rx65n_tm_user1_Dual.abs"を入力ファイルとしています。
-form=stype	出力形式をモトローラ S 形式ファイル(mot ファイル)としています。
-output="TM_bank1.mot"=FFFE0000-FFFEFFFF/FFEE0000	出力範囲をバンク 0 の TM 対象領域である FFFE 0000h – FFFE FFFFh とし、mot ファイル上の先頭ロードアドレス(注 1)をバンク 1 の TM 対象領域の先頭アドレスである FFEE 0000h に変更し、ファイル"TM_bank1.mot"に出力します。TM 対象領域のアドレスは製品によって異なりますので、使用する製品に合わせて設定値を変更してください。

注1. ロードアドレスの指定は、V3.00.00 以降の RX ファミリ C/C++コンパイラを使用してください。ロードアドレスの指定、またその他オプションの詳細は最新の CC-RX コンパイラユーザーズマニュアル（R20UT3248）を参照してください。

(3) デュアルモードで TM 機能を有効にする方法

デュアルモードで TM 機能を有効にするためには TMEF レジスタの TMEF ビットと、TMEFDB ビットの両方を設定します。また、TMINF レジスタには TM 対象領域のプログラムを識別できるコードを格納できます。

TMEF レジスタの TMEF ビットと、TMEFDB ビット、TMINF レジスタを設定するためには、自動生成ファイル”vecttbl.c”を以下のように修正し、RFP を使用してオプション設定メモリへプログラムしてください。

```
#pragma address __TMEFreg=0xfe7f5d48          // TMEF register
const unsigned long __TMEFreg = 0x88ffffff;
```

```
#pragma address __TMINFreg=0xfe7f5d10         // TMINF register
const unsigned long __TMINFreg = 0xffffffff;
```

2.3 第 2 ユーザの開発手順

2.3.1 アプリケーション開発

提供された変数および関数テーブルのライブラリファイル(.lib)を使用し、アプリケーションを開発します(注 1)。

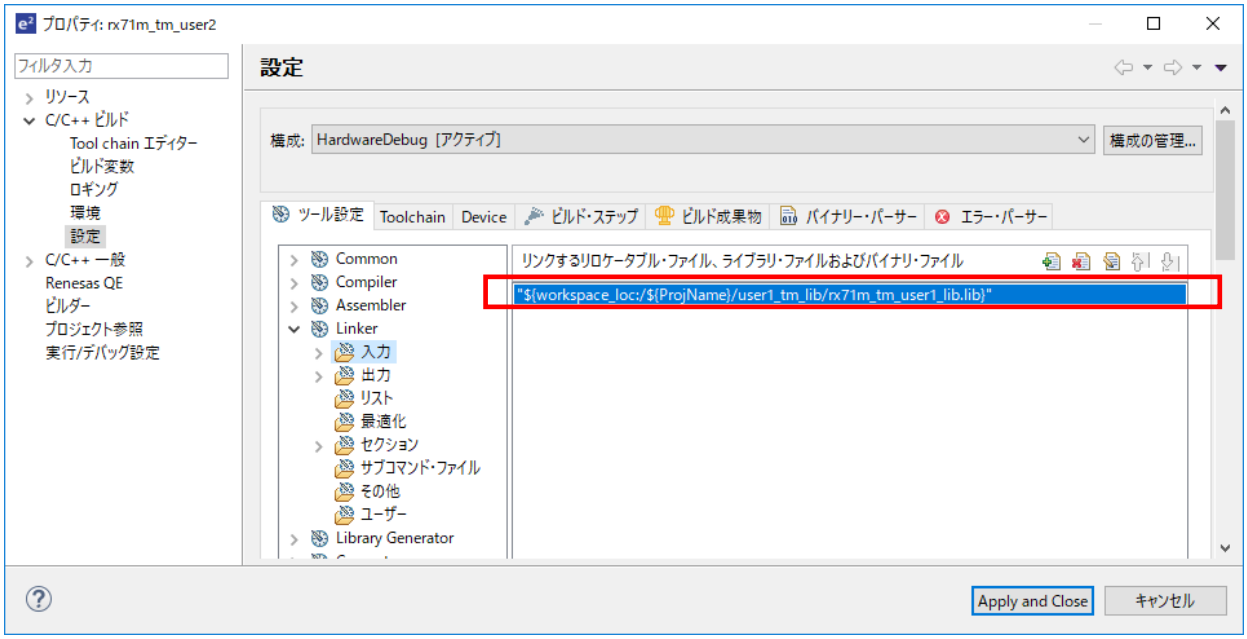
ライブラリファイル(.lib)のリンク方法を以下に示します。

(1)e2 studio で対象プロジェクトを右クリック、「プロパティ(R)」を選択。

(2)「C/C++ビルド > 設定 > ツール設定タブ > Linker > 入力」を選択。

(3)提供された lib ファイルを追加。

(4)「適用(L)」をクリック。



第 2 ユーザは、提供されたライブラリファイル(.lib)の関数テーブルを使用して、TM 対象領域の API 関数を呼び出してください。

関数テーブルを使用した API 関数の呼び出し例を以下に示します。

```
/* ==== Initialize LED ports ==== */
tm_func_table1[R_TM_PORT_INIT]();

/* ==== LEDs ON ==== */
tm_func_table2[R_TM_SET_LED](*table);

/* ==== LEDs OFF ==== */
tm_func_table1[R_TM_LED_OFF]();
```

注1. アプリケーション開発の際に、TM 対象領域にプログラムを配置しないでください。TM 機能有効のため、書き込み時にエラーが発生します。

2.3.2 開発したアプリケーションをデバッグ

統合開発環境 e2 studio で TM 対象領域以外にプログラムをダウンロードすることでアプリケーションのデバッグが可能です。

統合開発環境 e2 studio のプログラムダウンロードは、ブロックごとに「消去せずに上書き」または「消去してから上書き」を選択できます。デフォルトでは「消去してから上書き」に設定されていますが、TM 機能有効の場合、TM 対象領域のプログラムは消去されません。

3. サンプルコード

本アプリケーションノートの 2 章で説明している、セクションの設定方法や、API 関数の関数テーブルの作成方法、API 関数の呼び出し方法などをサンプルコードにして提供していますので参考にしてください。

サンプルコードは、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。

3.1 サンプルコードの構成

表 3.1 に RX71M 用のサンプルコード、表 3.2 に RX65N デュアルモード用のサンプルコードを示します。

表 3.1 RX71M 用のサンプルコード*1

ファイル名	概要
rx71m_tm_user1	第 1 ユーザの開発プロジェクト
rx71m_tm_user1_lib	第 2 ユーザへ提供する lib ファイル作成プロジェクト
rx71m_tm_user2	第 2 ユーザの開発プロジェクト

注1. RX65N、RX651、RX72M、RX72N、RX66N のリニアモード、および RX64M、RX66T、RX72T を使用する場合は本サンプルコードを参考にしてください。

表 3.2 RX65N デュアルモード用のサンプルコード*1

ファイル名	概要
DualMode.mot	第 1 ユーザがデバイスをデュアルモードに設定するための mot ファイル
rx65n_tm_user1_Dual	第 1 ユーザの開発プロジェクト(デュアルモード用)
rx65n_tm_user1_Dual_lib	第 2 ユーザへ提供する lib ファイル作成プロジェクト
rx65n_tm_user2_Dual	第 2 ユーザの開発プロジェクト

注1. RX651、RX72M、RX72N、RX66N のデュアルモードを使用する場合は本サンプルコードを参考にしてください。

3.2 サンプルコードの動作環境

サンプルコードは、表 3.3、表 3.4 の条件で動作を確認しています。

表 3.3 サンプルコードの動作確認環境 1

項目	内容
動作確認ファイル	rx71m_tm_user1 rx71m_tm_user1_lib rx71m_tm_user2
使用マイコン	R5F571MLCDFC (RX71M グループ)
ユーザ領域の容量	4M バイト
動作周波数	• メインクロック: 24MHz • PLL: 240MHz (メインクロック 1 分周 10 逓倍) • システムクロック(ICLK): 120MHz (PLL 2 分周) • 周辺モジュールクロック A (PCLKA): 120MHz (PLL 2 分周) • 周辺モジュールクロック B (PCLKB): 60MHz (PLL 4 分周) • FlashIF クロック(FCLK): 60MHz (PLL 4 分周)
動作電圧	3.3V
統合開発環境	ルネサスエレクトロニクス製 e2 studio Version: 7.6.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V.3.01.00
iodefine.h のバージョン	V1.0A
エンディアン	リトルエンディアン
動作モード	シングルチップモード
プロセッサモード	スーパバイザモード
サンプルコードのバージョン	rx71m_tm_user1：Version 1.10 rx71m_tm_user1_lib：Version 1.10 rx71m_tm_user2：Version 1.11
使用ボード	Renesas Starter Kit+ for RX71M
使用ツール	Renesas Flash Programmer V3.06.01

表 3.4　サンプルコードの動作確認環境 2

項目	内容
動作確認プロジェクト	DualMode.mot rx65n_tm_user1_Dual rx65n_tm_user1_Dual_lib rx65n_tm_user2_Dual
使用マイコン	R5F565NEDDFC (RX65N グループ)
ユーザ領域の容量	2M バイト
動作周波数	• メインクロック: 24MHz • PLL: 240MHz (メインクロック 1 分周 10 逓倍) • システムクロック(ICLK): 120MHz (PLL 2 分周) • 周辺モジュールクロック A (PCLKA): 120MHz (PLL 2 分周) • 周辺モジュールクロック B (PCLKB): 60MHz (PLL 4 分周) • FlashIF クロック(FCLK): 60MHz (PLL 4 分周)
動作電圧	3.3V
統合開発環境	ルネサスエレクトロニクス製 e2 studio Version: 7.6.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V.3.01.00
iodefine.h のバージョン	V2.2
エンディアン	リトルエンディアン
動作モード	シングルチップモード
プロセッサモード	スーパバイザモード
サンプルコードのバージョン	rx65n_tm_user1_Dual：Version 1.10 rx65n_tm_user1_Dual_lib：Version 1.10 rx65n_tm_user2_Dual：Version 1.10
使用ボード	Renesas Starter Kit+ for RX65N-2MB
使用ツール	Renesas Flash Programmer V3.06.01

3.3 サンプルコードの内容

表 3.5 に各サンプルコードの内容を示します。

表 3.5 各サンプルコードの内容

サンプルコード	内容
rx71m_tm_user1	・TM 対象領域に Renesas Starter Kit+ に搭載されている LED を点灯する API 関数を格納 ・TM 対象領域外に、API 関数のテーブル、API 関数が使用する変数、定数を格納 （詳細は 2.1.1「セクション設定」、2.2.1「アプリケーション開発」を参照してください） ・TM 機能の有効化 （詳細は 2.2.3「TM 機能を有効にする」を参照してください）
rx71m_tm_user1_lib	・TM 対象領域で使用する定数、および API 関数のテーブルをライブラリファイル(.lib)として出力 （詳細は 2.2.2「変数および関数テーブルのライブラリファイル作成」を参照してください）
rx71m_tm_user2	・ライブラリファイルのリンク （詳細は 2.3.1「アプリケーション開発」を参照してください） ・TM 対象領域内の API 関数を呼び出します （詳細は 2.3.1「アプリケーション開発」を参照してください） RX64M、RX71M、RX66T、RX72T を使用する場合、TMEF=111b の書き込み後のベリファイはエラーとなります。TMEF ビットを 000b に設定したベリファイ用の mot ファイルを作成してベリファイしてください。
DualMode.mot	・デュアルモードに設定するための mot ファイル （詳細は 2.2.5(1)「デュアルモードへの設定」を参照してください）
rx65n_tm_user1_Dual	・TM 対象領域に Renesas Starter Kit+ に搭載されている LED を点灯する API 関数を格納 ・TM 対象領域外に、API 関数のテーブル、API 関数が使用する変数、定数を格納 （詳細は 2.1.1「セクション設定」、2.2.1「アプリケーション開発」を参照してください） ・TM 機能の有効化 （詳細は 2.2.3「TM 機能を有効にする」を参照してください） ・バンク 0 の TM 対象領域をバンク 1 の TM 対象領域へコピー （詳細は 2.2.5(2)「TM 対象領域のコピー」を参照してください）
rx65n_tm_user1_Dual_lib	・TM 対象領域で使用する定数、および API 関数のテーブルをライブラリファイル(.lib)として出力 （詳細は 2.2.2「変数および関数テーブルのライブラリファイル作成」を参照してください）
rx65n_tm_user2_Dual	・ライブラリファイルのリンク （詳細は 2.3.1「アプリケーション開発」を参照してください） ・TM 対象領域内の API 関数を呼び出します （詳細は 2.3.1「アプリケーション開発」を参照してください）

3.4 サンプルコードのセクション情報

表 3.6 にサンプルコードのセクション情報を示します。

セクションの追加／変更および削除の方法は、最新の RX ファミリ C/C++コンパイラ(CC-RX) ユーザーズマニュアルを参照してください。

表 3.6　サンプルコードのセクション情報

セクション名	領域	第 1 ユーザ	第 2 ユーザ	内容
SU	RAM	—	—	ユーザスタック領域
SI		—	—	割り込みスタック領域
B_1		—	—	未初期化データ領域 初期化データ領域
R_1		—	—	
B_2		—	—	
R_2		—	—	
B		—	—	
R		—	—	
B_TM_1	0000 F000	追加	追加	TM 対象領域で使用 – 未初期化データ領域 – 初期化データ領域
R_TM_1		追加	追加	
B_TM_2		追加	追加	
R_TM_2		追加	追加	
B_TM		追加	追加	
R_TM		追加	追加	
PResetPRG	ユーザ領域	—	—	パワーオンリセット PC
C_1		—	—	定数領域
C_2		—	—	
C		—	—	
C$*		—	—	
D_1		追加	追加	初期化データ領域 デフォルトから変更 (D* ⇒ D)
D_2		追加	追加	
D		変更	変更	
W*		—	—	switch 文分岐テーブル領域
L		—	—	リテラル領域
PIntPRG		—	—	割り込みベクタテーブル
P		—	—	プログラム領域
C_TM_1	FFFD F000	追加	追加	TM 対象領域で使用 – 定数領域
C_TM_2		追加	追加	
C_TM		追加	追加	
D_TM_1		追加	追加	TM 対象領域で使用 – 初期化データ領域(ROM)
D_TM_2		追加	追加	
D_TM		追加	追加	
P_TM_FUNC_01	FFFE 0000	追加	追加	TM 対象領域 API 関数 01
P_TM_FUNC_02	FFFE 0100	追加	追加	TM 対象領域 API 関数 02
P_TM_FUNC_03	FFFE 0200	追加	追加	TM 対象領域 API 関数 03
P_TM	FFFE 0300	追加	不要	TM 対象領域 内部関数
EXCEPTVECT	ユーザ領域	—	—	例外ベクタテーブル
RESETVECT		—	—	リセットベクタ

—：デフォルト

3.5 サンプルコードが使用しているアプリケーションノート

本サンプルコードに関連するアプリケーションノートを以下に示します。併せて参照してください。

- RX71M グループ 初期設定例 (R01AN2459)
- RX65N グループ、RX651 グループ 初期設定例 (R01AN3034)

上記アプリケーションノートの初期設定関数を、本サンプルコードで使用しています。
最新版がある場合、最新版に差し替えて使用してください。最新版はルネサスエレクトロニクスホームページで確認および入手してください。

本アプリケーションノートでは、アプリケーションノート「RX71M グループ 初期設定例」の設定を一部変更しています。最新版に差し替える場合、必要に応じて設定を変更してください。

初期設定例の変更箇所を表 3.7 に示します。

表 3.7 サンプルコードで変更する初期設定例の定数(r_init_clock.h)

定数名	変更前の設定値	変更後の設定値	内容
REG_SCKCR	20C2 1222h (PLL 選択時)	21C2 1222h (PLL 選択時)	内部クロック分周比 (SCKCR レジスタの設定値) 変更前 : ICLK = 240MHz 変更後 : ICLK = 120MHz
REG_MEMWAIT	MEMWAIT_1WAIT	MEMWAIT_0WAIT	メモリウェイトサイクルの選択 変更前 : 1 ウェイト 変更後 : 0 ウェイト

4. 注意事項

第 1 ユーザはセクション情報をセクション情報ファイル(.esi)で第 2 ユーザに提供することもできます。

第 1 ユーザは、e2 studio でセクション情報ファイル(.esi)をエクスポートしてください。セクション情報ファイル(.esi)の作成手順を以下に示します。

(1)e2 studio で対象プロジェクトを右クリック、「プロパティ(R)」を選択。

(2)「C/C++ビルド > 設定 > ツール設定タブ > Linker > セクション」を選択。

(3)エクスポートをクリック。

(4)保存先を選択。

第 2 ユーザは、e2 studio でセクション情報ファイル(.esi)をインポートしてください(注 1)。セクション情報ファイル(.esi)のインポート手順を以下に示します。

(1)e2 studio で対象プロジェクトを右クリック、「プロパティ(R)」を選択。

(2)「C/C++ビルド > 設定 > ツール設定タブ > Linker > セクション」を選択。

(3)インポートをクリック。

(4)「セクション情報ファイル(.esi)」を開く。

注1. セクション情報ファイル(.esi)をインポートすると、既に設定しているセクションがすべて消去、上書きされるので注意してください。

5. 参考ドキュメント

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 (R01UH0377)
RX71M グループ ユーザーズマニュアル ハードウェア編 (R01UH0493)
RX65N グループ、RX651 グループ ユーザーズマニュアル ハードウェア編 (R01UH0590)
RX66T グループ ユーザーズマニュアル ハードウェア編 (R01UH0749)
RX72T グループ ユーザーズマニュアル ハードウェア編 (R01UH0803)
RX72M グループ ユーザーズマニュアル ハードウェア編 (R01UH0804)
RX72N グループ ユーザーズマニュアル ハードウェア編 (R01UH0824)
RX66N グループ ユーザーズマニュアル ハードウェア編 (R01UH0825)
(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)

ユーザーズマニュアル：フラッシュメモリ
RX64M グループ フラッシュメモリ ユーザーズマニュアル ハードウェア インタフェース編 (R01UH0435)
RX71M グループ フラッシュメモリ ユーザーズマニュアル ハードウェア インタフェース編 (R01UH0435)
RX65N グループ、RX651 グループ フラッシュメモリ ユーザーズマニュアル ハードウェア インタフェース編 (R01UH0602)
(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)

ユーザーズマニュアル：Renesas Flash Programmer
Renesas Flash Programmer フラッシュ書き込みソフトウェア ユーザーズマニュアル (R20UT4307)
(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)

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 C/C++ コンパイラ(CC-RX)ユーザーズマニュアル(R20UT3248)
(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	Apl.1.4.15	—	初版発行
1.01	Apl.28.15	5	表 2.1 動作確認条件 統合開発環境 変更
		19	4.4.2 開発したアプリケーションのデバッグ 変更
		—	7.2 デバッグについて 削除
2.00	Jun.06.19	—	対象デバイスの追加 ·RX65N グループ ·RX651 グループ ·RX66T グループ ·RX72T グループ ·RX72M グループ
3.00	Jan.20.20	—	対象デバイスの追加 ·RX72N グループ ·RX66N グループ

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。