

マイコントレーニングボード MT-R300

割り込みプログラムのモニタデバッグ方法

1. 概要

マイコントレーニングボード MT-R300 に搭載しているルネサス テクノロジ社の 16 ビットマイコン HD64F3062BF (以下 H8/3062BF マイコン) では、H8/300H 用モニタプログラム (以下モニタプログラム) を内蔵フラッシュメモリに書込むことで、簡単なプログラムの評価をすることができます。

モニタプログラムは、ユーザプログラムを RAM に書込んで実行します。ユーザプログラムで割り込みの機能を使用している場合は、RAM 上に仮想のベクタテーブルを設定し、本来のフラッシュメモリのベクタテーブルから仮想ベクタテーブルを介して割り込みプログラムを実行します。

本アプリケーションノートでは、例として 16 ビットタイマ チャネル 0 の割り込みを使っているプログラム (LED_PICOPICO_H8_Int) に仮想ベクタテーブルを設定する方法と、HEW での必要な設定について説明します。

※ 本アプリケーションノートで使用するツール類は、MT-R300 に付属する CD-ROM (Ver1.00) に収録されているバージョンのものとします。

※ H8/300H モニタプログラムは、ルネサス テクノロジ社が提供している H8/300H シリーズ用のモニタプログラムです (ルネサス テクノロジ社のサポート対象外)。MT-R300 に付属する CD-ROM (Ver1.00) に収録している H8/300H モニタプログラムは、MT-R300 用にサンハヤト株式会社がモディファイしたものです。

2. 仮想ベクタテーブルのしくみ

モニタプログラムでは、ユーザープログラムの実行中に割り込みが発生すると、本来の割り込みベクタ (フラッシュメモリに配置) が示すモニタプログラムの割り込みプログラムを実行します。モニタプログラムの割り込みプログラムでは、仮想ベクタテーブル (RAM に配置) の該当するベクタが示すアドレス (ユーザープログラムの割り込みプログラム) にジャンプします。

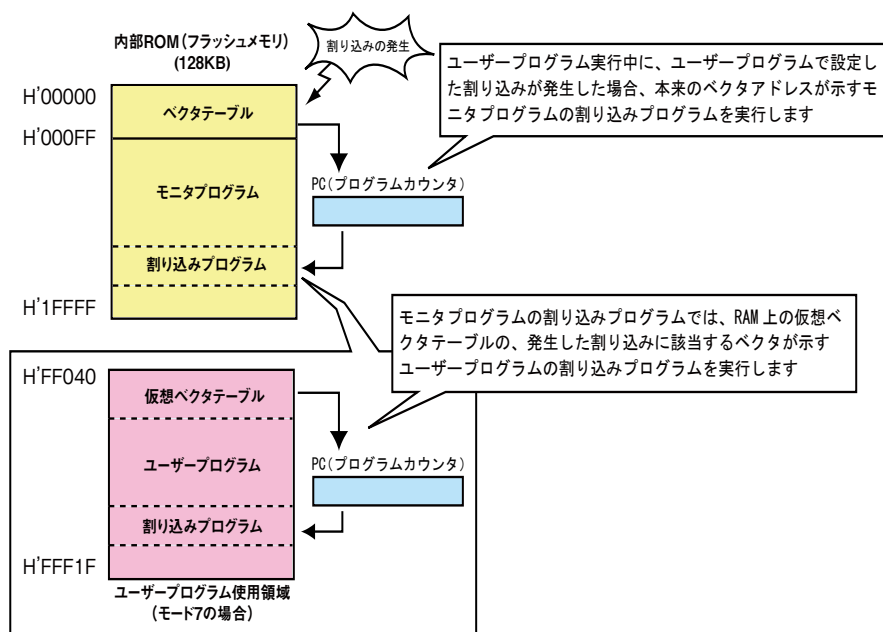


図1 仮想ベクタテーブルのしくみ (モード7の場合、イメージ図)

3. 仮想ベクタテーブルの設定

以下に仮想ベクタテーブルの C 言語の記述例を示します。

仮想テーブルを配置する領域として新しいセクション（名前は任意、リストでは "V_Vector"）を設定します。

```
#include <machine.h>
#include "LED_PICOPICO_H8.h"

#pragma interrupt (INT_IMIA0)
void INT_IMIA0(void);

#pragma section V_Vector
void (*const VEC_TBL1[])(void)=
{
    INT_IMIA0
};

#pragma section IntPRG
void INT_IMIA0(void)
{
    static int counter_1sec;          /*1 秒計測用カウンタ */
    ITU.TISRA.BIT.IMFA0 = 0;         /* タイマ割り込み要求フラグをクリアしておく */
    counter_1sec++;                  /*1 秒計測用カウンタをインクリメントする */

    if (counter_1sec>=100)           /*1 秒計測用カウンタが 100 以上か (10ms×100=1 秒) */
    {
        counter_1sec=0;              /*1 秒計測用カウンタをクリアしておく */
        LED1 = ~LED1;                /*LED1 反転 */
        LED2 = ~LED2;                /*LED2 反転 */
    }
}
```

リスト1 intprg_debug.c

4. HEW でのリンカオプションの設定

新しく設定したセクションとユーザープログラムが、RAM の正しいアドレスに配置されるように、リンカオプションを設定します。

モニタプログラム（MT-R300 に付属の CD-ROM に収納）では、ユーザープログラムが使用できる RAM の先頭アドレス（モード 7 の場合は FF040H、モード 5 の場合は 1F8000H）に仮想ベクタテーブルの先頭アドレスを割り当てています。したがって設定する仮想ベクタテーブルのアドレスは、RAM の先頭アドレス+プログラムで使用する割り込みのベクタアドレスのオフセットとなります。

16 ビットタイマのチャンネル 0 コンペアマッチ A0 の割り込みベクタは、アドバンスモードの場合 0060H ~ 0063H なので、オフセットは 60H となります。

表1 各モードの仮想テーブル先頭アドレスと16ビットタイマチャンネル0コンペアマッチA0割り込みのベクタアドレス

	モード7	モード5
仮想テーブル先頭アドレス(リセットベクタ) ~全ての割り込みベクタを含む最終アドレス	FF040H ~ FF13Fh	1F8000H ~ 1F80FFH
16ビットタイマチャンネル0 コンペアマッチ A0 割り込みの仮想テーブル ベクタアドレス	FF0A0H (FF040H + 60H) ~ FF0A3H	1F8060H (1F8000H + 60H) ~ 1F8063H

ユーザープログラムやデータは、設定した仮想ベクタテーブルに重ならないように配置します※。以下に HEW での設定方法を示します。

HEW の「ビルド」メニューより「H8S,H8/300 Standard Toolchain...」をクリックしてください。「最適化リンカ」タブをクリックし、「カテゴリ」を「セクション」に設定します。「編集」ボタンをクリックして、配置するアドレスとセクション名を設定します。

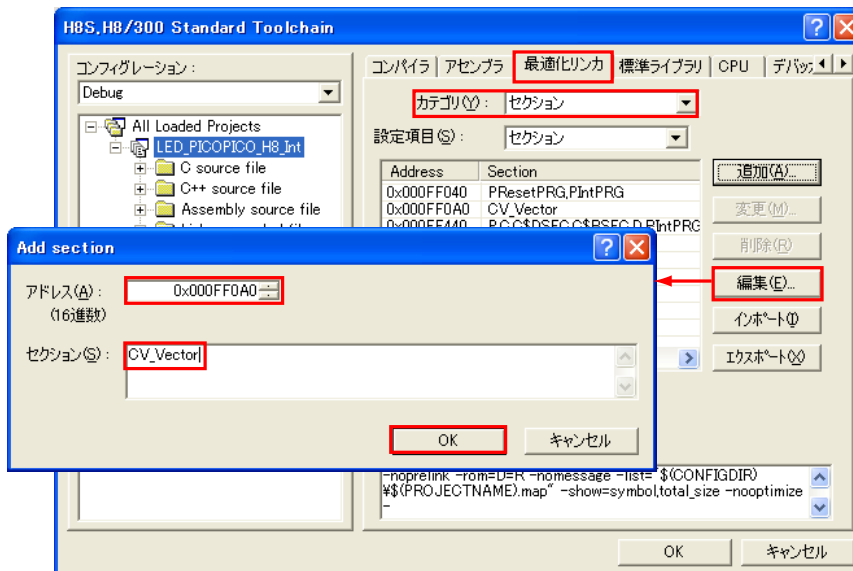


図2 HEWによるセクションの追加設定（モード7の場合）

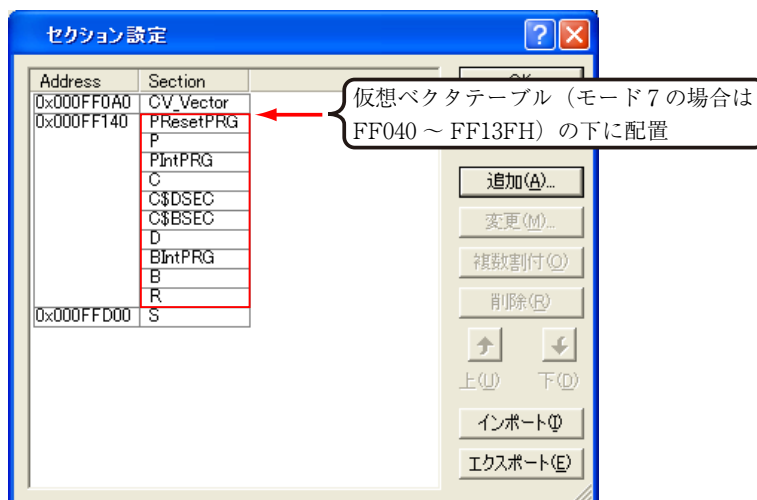
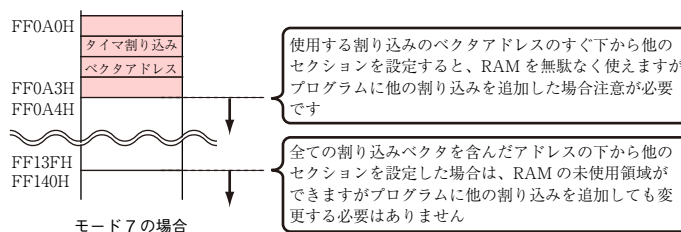


図3 セクションの設定例（モード7の場合）

※ 上記の設定例では、リザーブ領域を含めた全ての割り込みベクタを含めた領域（モード7の最終アドレスはFF13F、モード5では1F80FFH）を仮想テーブルのセクションとして設定しています。プログラムでタイマ割り込みのみを使っている場合、ベクタアドレスはFF0A0H～FF0A3Hに設定されるので（アドバンスモードの場合はアドレスは4バイトで表現します）、CV_Vector以降の各セクションをFF0A4Hから割り当てることもできます。この設定はタイマ割り込みのみを使用する場となりますので、タイマ割り込み以外の割り込みをプログラムに追加した場合は、各セクションのアドレスを追加した割り込みの仮想テーブルのセクションに重ならないように考慮する必要があります。



5. 複数の割り込みを使用している場合

ユーザープログラムで複数の割り込みを使用している場合は、割り込みごとに新しいセクションとテーブルを設定するか、該当するベクタテーブルのアドレスが近い場合はダミーアドレスを挿入して設定します。以下に6ビットタイマのチャンネル0とチャンネル1の割り込みを使う場合の設定例を示します。(ヘッダファイルのインクルード、割り込み関数の拡張機能の設定とプロトタイプ宣言は省略しています。)

```
#pragma section V_Vector
void (*const VEC_TBL1[])(void)=
{
    INT_IMIA0,
    dummy,          /* IMIB0 */
    dummy,          /* OVIO */
    dummy,          /* Reserved */
    INT_IMIA1
};

#pragma section IntPRG
void INT_IMIA0(void)
{
    省略
}

void INT_IMIA1(void)
{
    省略
}

void dummy(void)
{
    ;
}
```

使っていないベクタに dummy アドレスを設定する

割り込みプログラム本体

リスト2 複数の割り込みベクタの設定例 (1つのセクション、テーブルに設定する例)

```
#pragma section V_Vector
void (*const VEC_TBL1[])(void)=
{
    INT_IMIA0
};

#pragma section V_Vector_02
void (*const VEC_TBL2[])(void)=
{
    INT_IMIA1
};

#pragma section IntPRG
void INT_IMIA0(void)
{
    省略
}

void INT_IMIA1(void)
{
    省略
}
```

ベクタごとにセクションを設定する
※リンカオプションで2つのセクションを配置する必要があります

割り込みプログラム本体

リスト3 複数の割り込みベクタの設定例 (セクション別に設定する例)

6. HEW でのコンフィギュレーションごとの設定例

MT-R300 で割り込みプログラムをモニタプログラムでデバッグする場合と、フラッシュメモリに書込む場合とでは、仮想ベクタテーブルの設定等内容が異なります。HEW ではビルド対象となるファイルを、コンフィギュレーション※ごとに分けて設定することができます。

ここでは、デバッグ時の設定を Debug コンフィギュレーション、リリース時（フラッシュメモリに書込む）の設定を Release コンフィギュレーションに分けて保存しておくこととします。

条件コンパイルで分ける方法とビルド対象の設定で分ける方法の 2 つを紹介します。

■ 条件コンパイルでソースを分ける方法

条件コンパイル（`#ifdef ~ #else ~ #endif`）でコンフィギュレーションによってコンパイルするソースを分けます。

以下にソース例を示します。この例ではマクロ名「debug」が定義されていたら `#ifdef` 以下がコンパイルされます。「debug」の定義は、コンパイルオプション「`-DEFine`」で設定できます。Debug コンフィギュレーションでのみ設定します。

```

#include <machine.h>
#include "LED_PICOPICO_H8.h"

#ifdef debug
    #pragma interrupt (INT_IMIA0)
    void INT_IMIA0(void);

    #pragma section V_Vector

    void (*const VEC_TBL1[])(void)=
    {
        INT_IMIA0
    };
#endif

#pragma section IntPRG

#ifdef debug
    void INT_IMIA0(void)
    {
#else
    __interrupt(vect=24) void INT_IMIA0(void)
    {
#endif

        static int counter_1sec;          /*1 秒計測用カウンタ*/

        ITU.TISRA.BIT.IMFA0 = 0;         /* タイマ割り込み要求フラグをクリアしておく*/
        counter_1sec++;                  /*1 秒計測用カウンタをインクリメントする*/

        if (counter_1sec>=100)           /*1 秒計測用カウンタが 100 以上か (10ms×100=1 秒)*/
        {
            counter_1sec=0;              /*1 秒計測用カウンタをクリアしておく*/
            LED1 = ~LED1;                 /*LED1 反転*/
            LED2 = ~LED2;                 /*LED2 反転*/
        }
    }

```

“debug” が定義されていたら仮想ベクタテーブルに割り込みプログラムの先頭アドレスを設定

“debug” が定義されている場合の関数の始まり

“debug” が定義されていない場合は本来のベクタテーブルに割り込み関数の先頭アドレスを設定する

割り込みプログラム本体

リスト4 条件コンパイルで分ける例

※ HEW のコンフィギュレーションは、ビルドを行う際のフェーズ（アセンブラ、コンパイラ、リンカ）の構成や、各フェーズでのオプション設定を保存しています。コンフィギュレーションを切り替えることによって、異なるオプションの組み合わせを効率良く切り替えてビルドすることができます。

以下に HEW でのマクロ名の定義の設定方法を示します。

HEW の「ビルド」メニューより「H8S,H8/300 Standard Toolchain...」をクリックしてください。「コンパイラ」タブをクリックし、「カテゴリ」を「その他」に設定します。設定するコンフィギュレーションが「Debug」になっていることを確認して、「ユーザ指定オプション」のウィンドウに「-DEFine=debug」と設定してください。

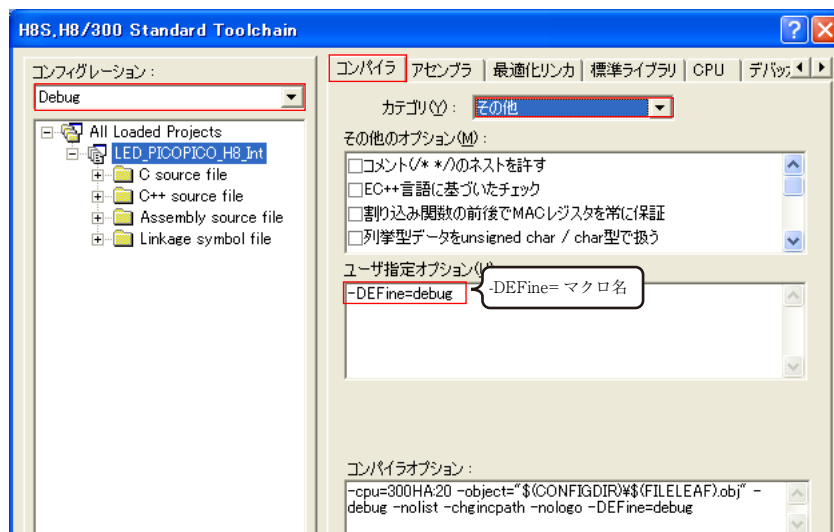


図4 コンパイラオプション (-DEFineオプションによるマクロ名の設定)

■ ビルド対象のファイルをコンフィギュレーションごとに設定する方法

フラッシュメモリに書込む割り込みプログラムとモニタプログラムでデバッグする割り込みプログラムを別のファイルにしておき、コンフィギュレーションごとにビルド対象を変えます。フラッシュメモリに書込む割り込みプログラムを「intprg.c」ファイル、モニタプログラムでデバッグする割り込みプログラムを「intprg_debug.c」ファイルとしています。

Debug コンフィギュレーションでは「intprg.c」をビルド対象から除外し、Release コンフィギュレーションでは「intprg_debug.c」をビルド対象から除外します。

ビルド対象から外すファイルを右クリックして「ビルドから除外」をクリックしてください。ファイル上に赤矢印が表示され、ビルド対象から外されます。

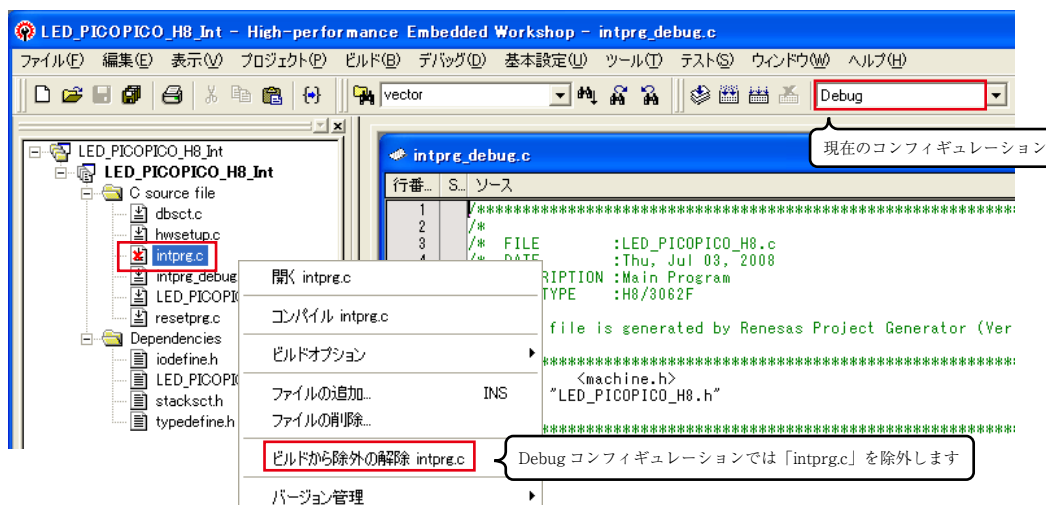


図5 ビルド除外の設定

以下に二つのプログラム例を示します。

intprg_debug.c ← Debug コンフィギュレーションでビルド (Release コンフィギュレーションでビルド除外)

```
#include <machine.h>
#include "LED_PICOPICO_H8.h"

#pragma interrupt (INT_IMIA0)
void INT_IMIA0(void);

#pragma section V_Vector

void (*const VEC_TBL1[])(void)=
{
    INT_IMIA0
};

#pragma section IntPRG

void INT_IMIA0(void)
{
    static int counter_1sec;          /*1 秒計測用カウンタ*/

    ITU.TISRA.BIT.IMFA0 = 0;         /* タイマ割り込み要求フラグをクリアしておく*/
    counter_1sec++;                  /*1 秒計測用カウンタをインクリメントする*/

    if (counter_1sec>=100)           /*1 秒計測用カウンタが 100 以上か (10ms×100=1 秒)*/
    {
        counter_1sec=0;              /*1 秒計測用カウンタをクリアしておく*/
        LED1 = ~LED1;                /*LED1 反転*/
        LED2 = ~LED2;                /*LED2 反転*/
    }
}
```

intprg.c ← Release コンフィギュレーションでビルド (Debug コンフィギュレーションでビルド除外)

```
#include <machine.h>
#include "LED_PICOPICO_H8.h"

#pragma section IntPRG

__interrupt(vect=24) void INT_IMIA0(void)
{
    static int counter_1sec;          /*1 秒計測用カウンタ*/

    ITU.TISRA.BIT.IMFA0 = 0;         /* タイマ割り込み要求フラグをクリアしておく*/
    counter_1sec++;                  /*1 秒計測用カウンタをインクリメントする*/

    if (counter_1sec>=100)           /*1 秒計測用カウンタが 100 以上か (10ms×100=1 秒)*/
    {
        counter_1sec=0;              /*1 秒計測用カウンタをクリアしておく*/
        LED1 = ~LED1;                /*LED1 反転*/
        LED2 = ~LED2;                /*LED2 反転*/
    }
}
```

リスト5 ビルド対象を設定することで分ける例

※ 上記の例では、同じ割り込みプログラムの本体が二つのファイルに存在することになるので、割り込みプログラムの内容を変更する場合は、必ず二つのファイルの割り込みプログラムの内容を一致させるようにしてください。

本資料について

- 本資料は、電子工作や電子回路、パーソナルコンピュータの操作について一般的な知識をお持ちの方を対象にしています。
- 本資料を元に操作するには、ルネサス テクノロジ社製 H8/300H シリーズマイコンについての知識や開発環境などが必要です。
- Microsoft®、Windows® は米国 Microsoft 社の米国およびその他の国における登録商標です。
- その他、記載されている製品名は各社の商標または登録商標です。

本資料のご利用にあたって

- 本資料に掲載している内容は、お客様が用途に応じた適切な製品をご購入頂くことを目的としています。その使用により当社及び第三者の知的財産権その他の権利に対する保証、又は実施権の許諾を意味するものではありません。また、権利の侵害に関して当社は責任を負いません。
- 本資料に記載した情報を流用する場合は、お客様のシステム全体で充分評価し適用可能かご判断願います。当社では適用可能判断についての責任を負いません。

- 本資料に記載してある内容は、一般的な電子機器（学習教材、事務機器、計測機器、パーソナル機器、コンピュータ機器など）に使用されることを目的としています。高い品質や信頼性が要求され、故障や誤作動が直接人命を脅かしたり人体に危害を及ぼす恐れのある、医療、軍事、航空宇宙、原子力制御、運輸、移動体、各種安全装置などの機器への使用は意図も保証もしていません。
- 本資料の一部、又は全部を当社の承諾なしで、いかなる形でも転載又は複製されることは堅くお断りします。
- 全ての情報は本資料発行時点のものであり、当社は予告なしに本資料に記載した内容を変更することがあります。
- 本資料の内容は慎重に制作しておりますが、万一記述誤りによってお客様に損害が生じても当社はその責任を負いません。
- 本資料に関してのお問合せ、その他お気づきの点がございましたら、当社ホームページのお問い合わせページ (<https://www.sunhayato.co.jp/inquiry/>) よりお問合せください。
- 本資料に関する最新の情報はサンハヤト株式会社ホームページ (<http://www.sunhayato.co.jp/>) に掲載しております。

