TMS320F28x MCUシリーズ

HRPWMのSFO()関数による長期の割り込み禁止期間の対策

富士エレクトロニクス株式会社

SFO()関数と割り込み禁止期間

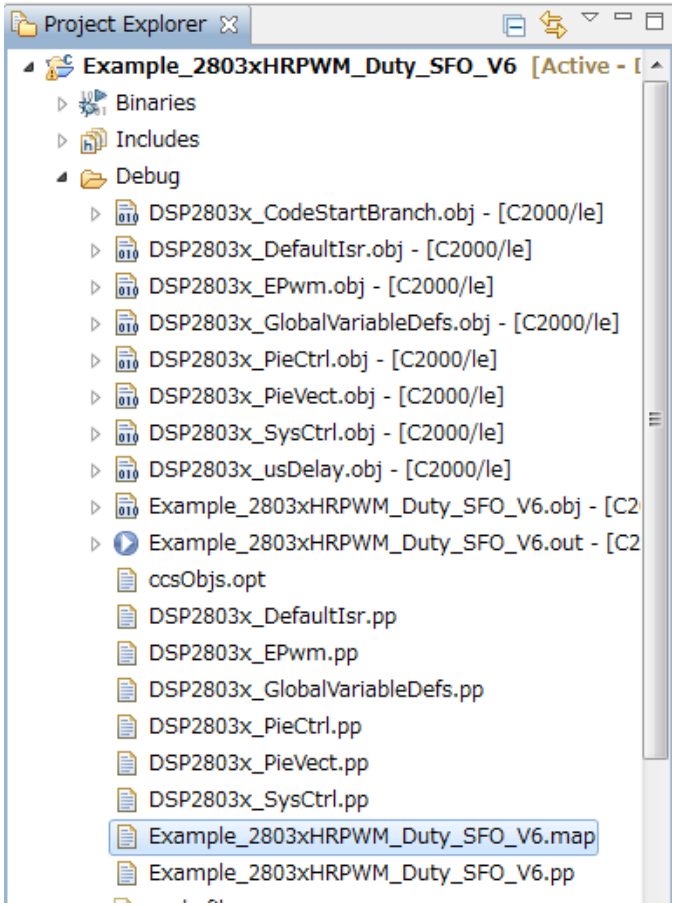
HRPWMを使用する時は、SFO()関数によって、キャリブレーションを行います。
しかしながら、SFO()関数には長期の割り込み禁止期間が存在し、デジタル制御電源のように、高速かつ確実な割り込み応答が求められるアプリケーションにおいては、それが問題になります。

SFO()関数は、浮動小数点演算を行っています。F2802x/F2803xのようにFPUがないデバイスの場合は、浮動小数点演算はランタイム・ライブラリの浮動小数点演算関数がコールされます。この関数に、RPT命令(次の行を指定した回数繰り返す命令)が使われています。RPT中は、割り込みがはいらないため、これが、長期の割り込み禁止期間となってしまいます。

これを回避するためには、ランタイム・ライブラリを変更します。

Step 1. 使用されているランタイムライブラリを確認

- まずランタイムライブラリ中の何の浮動小数点関数が使用されているかを調査します。



- プロジェクトにあるmapファイルを開いてください。.mapファイルは通常Debugフォルダの中にあります。
- .mapファイルは、正しくビルドが完了しないとできません。
- もし、.mapファイルが生成されていない時は、プロジェクトのPropertiesにて、リンカの設定を確認し、-mオプションにファイル名の指定があるか確認して下さい。

Step 1. 使用されているランタイムライブラリを確認

mapファイルにて、.text欄を見てください。
下記の関数が該当する浮動小数点演算のランタイム・ライブラリです。

```
.text      0     00008000    000008d0
                 00008000     000001b5     Example_2803xHRPWM_Duty_SFO_V6.obj (.text)
                 000081b5     0000017c     DSP2803x_DefaultIsr.obj (.text:retain)
                 00008331     00000144     SFO_TI_Build_V6b.lib : SFO_V6b.obj (.text)
                 00008475     00000139     DSP2803x_SysCtrl.obj (.text)
                 000085ae     000000c4     DSP2803x_EPwm.obj (.text)
                 00008672     00000083     rts2800_ml.lib : fs_div.obj (.text)
                 000086f5     00000078                    : fs_add.obj (.text)
                 0000876d     0000005a                    : fs_mpy.obj (.text)
                 000087c7     00000044                    : boot.obj (.text)
                 0000880b     00000028     DSP2803x_PieCtrl.obj (.text)
                 00008833     00000025     DSP2803x_PieVect.obj (.text)
                 00008858     00000025     rts2800_ml.lib : fs_toi.obj (.text)
                 0000887d     00000019                    : args_main.obj (.text)
                 00008896     00000019                    : exit.obj (.text)
                 000088af     00000010                    : u_tofs.obj (.text)
                 000088bf     00000009                    : _lock.obj (.text)
                 000088c8     00000008     DSP2803x_CodeStartBranch.obj (.text)
```

これです。.objになっていますが、そのソースコードは、.asmです。

Step 2. 対象となるファイルをProjectに追加

本来は、ランタイム・ライブラリのソースコードを変更して、ライブラリを再構築するのが正しい方法ですが、これは少し手間になりますので、別の方法を提案します。

変更したいランタイム・ライブラリのファイルをProjectに追加します。
そして、ファイルを変更して、ビルドします。

ここで、Projectにあるファイルと、ランタイム・ライブラリに、同じ関数が2つ存在する事になり、二重定義のエラーが発生するのではと思われるかもしれません。

しかし、リンク時には、Projectにあるファイルが優先してリンクされます。ライブラリのリンクは、シンボルが解決しない関数のみリンクされるためです。

Step 2. 対象となるファイルをProjectに追加

それでは、Projectに対象となるランタイム・ライブラリ・ソース・ファイルを追加します。ファイルの追加は、Project→Add Filesで、ファイルを選択して、Copy Filesを選択して下さい。

fs_div.asm, fs_div28.inc
fs_add.asm, fs_add28.inc
fs_mpy.asm, fs_mpy28.inc
fs_toi.asm, fs_toi28.inc
u_tofs.asm, u_tofs28.inc
c2000asm.inc

をProjectに追加して下さい。ランタイム・ライブラリのソースファイルは、コンパイラのディレクトリ¥lib¥rts.zip
(例えば。C:¥TI¥ccsv8¥tools¥compiler¥c2000_6.1.1¥lib¥rtssrc.zip)
にありますので、これを解凍して下さい。使っているコンパイラと同じバージョンのランタイム・ライブラリを必ず使ってください。

Step 3. 対象となる関数のソースコードを変更する

それでは、ソースコードを変更します。.asmファイルは、.incファイルをインクルードしているだけですので、.incファイルが実際のソースコードです。
ファイルを開くと、.incファイルをCCSのエディタがアセンブラファイルと認識せずに、変な表示になりますが、気にしないで下さい。
fs_add28.incを開くと、以下のような行があります。これが、RPTです。

```
RPT#31
    || NORMACC, XOP1_SE--  ; normalize result and adjust exponent
```

RPT命令の書式は、

```
RPT   #繰り返す回数-1
||   繰り返したい命令
```

です。

そのため、以下は、

```
RPT#31
    || NORMACC, XOP1_SE--  ; normalize result and adjust exponent
```

NORMACC, XOP1_SE—
を32回繰り返すという意味になりますので、
この2行を消して、
NORMACC, XOP1_SE—
を32個書きます。

このようにして、RPTを全対象ファイルに対して、削除していきます。

全てのRPT命令を削除後、再ビルドすることで、SFO関数の割込み禁止区間は解消されます。

Thank you for your Attention

FUJIELE

Value Innovation for the Future.

より良い未来へ、新しい価値を創造する。