CS+CC で確認のためのプロジェクトを作成し，TESSERA 社の RL78/G14 Stick で確認してみました。

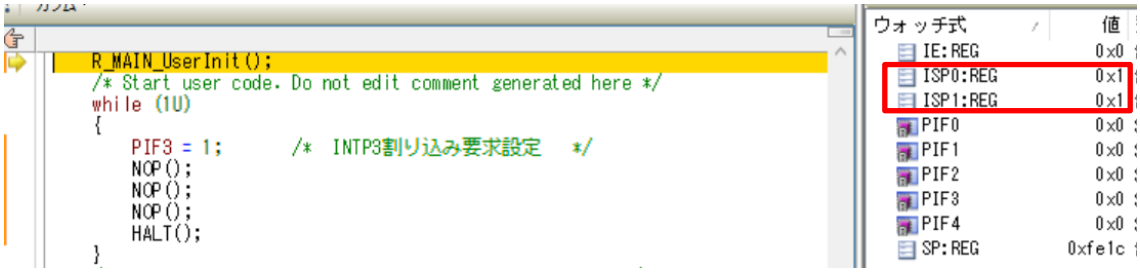
確認のために，INTP0～INTP4 の優先順位を以下のように設定しておきます。

その他の設定はプロジェクトを参照してください。

R_MAIN_UserInit 関数では，INTP0～INTP3 の割り込みマスクを解除しておきます。

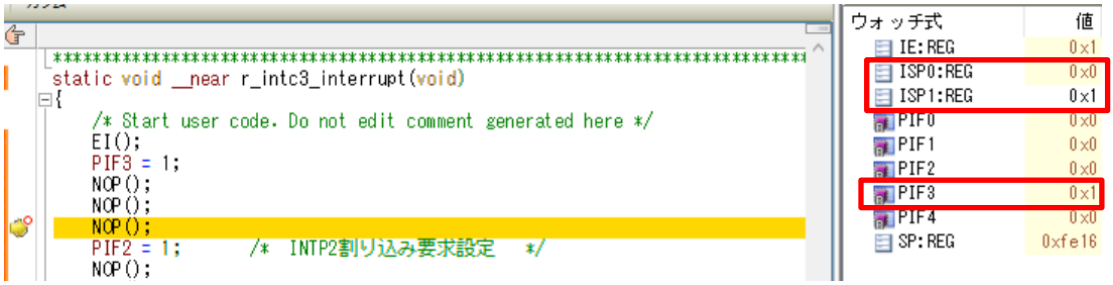
```
/***********************************************************************
void R_MAIN_UserInit(void)
{
    /* Start user code. Do not edit comment generated here */
    PMK0 = 0;           /* 割り込みマスク解除    */
    PMK1 = 0;
    PMK2 = 0;
    PMK3 = 0;
    EI();               /* ベクタ割り込み許可    */
    /* End user code. Do not edit comment generated here */
}
```

その上で，main 関数では INTP3 の割り込み要求をプログラムで設定します。初期状態では，以下のように ISP1,ISP0 は 11 になっています。

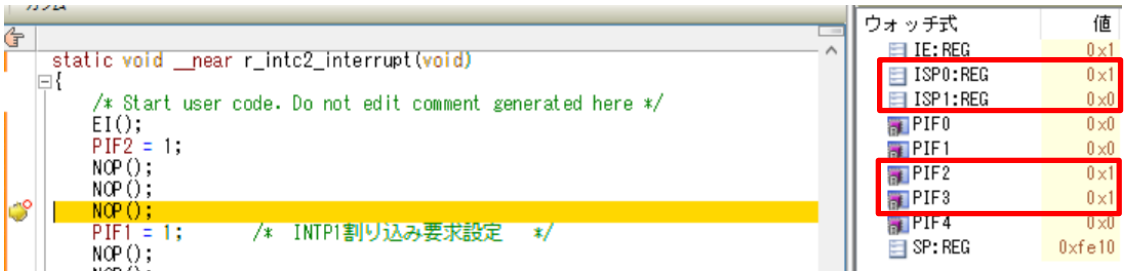


各割り込み処理(r_cg_intc_user.c)の各割り込みにはブレークポイントを設定しておき，プログラムをブレークポイント付きで実行します。すると，当然ながら，INTP3 割り込みの処理部分でブレークします。INTP3 は優先順位が低(レベル 3)なので，ISP1,ISP0 は 10 になります。ここで，多重割り込みのために EI()を実行し，INTP3 と INTP2 の割り込み要求をプログラムで設定します。

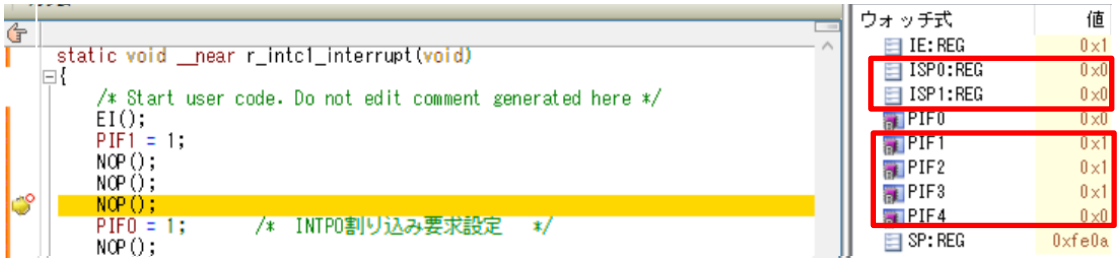
(INTP3 の方を先に設定して，NOP()で割り込みを受け付ける時間を確保しておきます。)



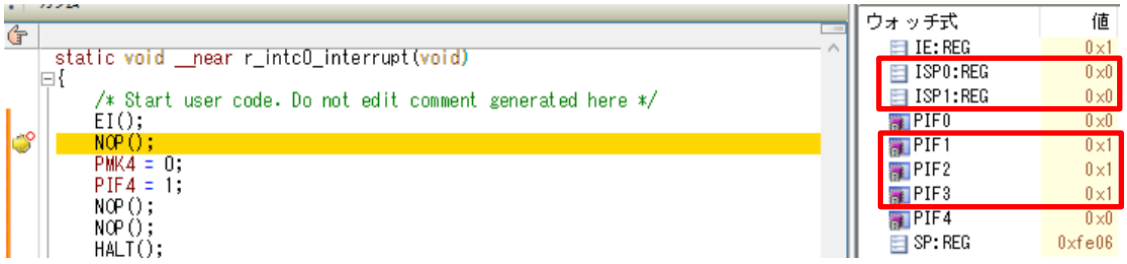
実行を再開すると，INTIP2 割り込み処理に分岐し，ISP1,ISP0 は 01 に変化しています。さらに，PIF3 をセットしていますが，INTP3 処理には分岐していません。



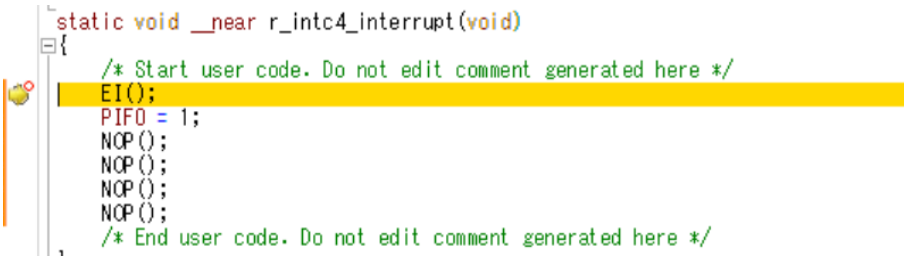
さらに，実行を再開します。今度は INTIP1 割り込み処理に分岐し，ISP1,ISP0 は 00 に変化しています



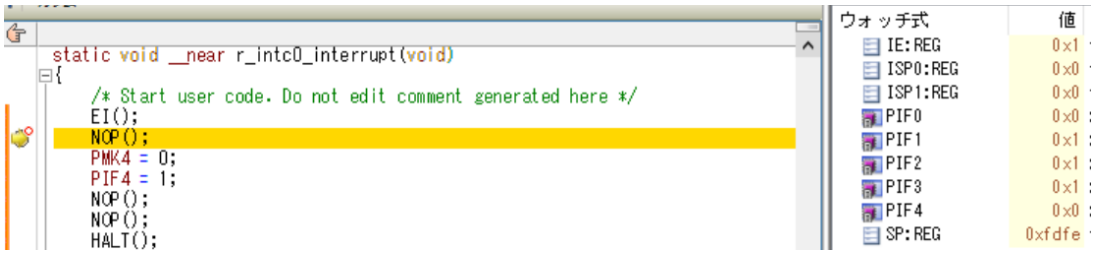
さらに，実行を再開します。今度は INTIP0 割り込み処理に分岐します。ISP1,ISP0 は 00 のままです。



さらに，INTP4 割り込み(優先順位最高)をセットします。これで，実行を再開します。



すると INTP4 の割り込み処理が起動していることが分かります。
ここでは，INTP0 割り込みをセットしているので，実行を再開すると，INTP0 割り込み処理に移るはずです。実行結果を下に示します。このように，優先順位最高の割り込みは EI()を実行するとどんどん割り込みを処理するのが分かります。



ここまでに処理は，すべて多重割り込みで処理しているので，SP の値がどんどん小さくなっているのも確認できます。