```
/************************************************************************/
/************************************************************************/
/***                                                                  ***/
/***         コンペアマッチタイマCMT0～CMT3による周期割り込みの発生          ***/
/***                                                                  ***/
/***                                                  by:  S.Suzuki   ***/
/************************************************************************/
/************************************************************************/
/* 注意 : ・PCLK=20MHzであること。                                      */
/*      : ・RX220の64ピンパッケージであること。                         */

#include    <machine.h>
#include    "typedefine.h"
#include    "iodefine.h"


#pragma section
#pragma bit_order right
/************************************************************************/
/*                      コンペアマッチタイマ初期化                      */
/************************************************************************/
/* 注意 : CMTのコンペアマッチ割り込みCMIはCMCNT=CMCOR（CMCORは0x0000になる）  */
/*      : となった次のクロックで発生するため、CMCORで指定したカウント値より  */
/*      : １カウント分多い時間になってしまう。従って、CMCORには希望のカウント */
/*      : 値―１の値を設定する。                                        */
/* 注意 : コンペアマッチが発生するとカウンタCMCNTは0x0000になる。          */

#define CONST500U   (1250)          /* PCLK/8(=2.5MHz)時の500uS発生用コンスタント値    */
#define CONST01M    (2500)          /* PCLK/8(=2.5MHz)時の 1mS発生用コンスタント値     */
#define CONST10M    (25000)         /* PCLK/8(=2.5MHz)時の10mS発生用コンスタント値     */
#define CONST50M    (7812)          /* PCLK/128(=156.25KHz)時の 50mS発生用コンスタント値 */
#define CONST100M   (15625)         /* PCLK/128(=156.25KHz)時の100mS発生用コンスタント値 */

void Init_PeriodTimer(void)
{
    /* 動作モード、消費電力低減機能、ソフトウェアリセット関連レジスタへの書き込みを許可する */
    SYSTEM.PRCR.WORD = 0xA502;
    /* CMT0,CMT1のモジュールストップ解除 */
    MSTP(CMT0) = 0;                 /* SYSTEM.MSTPCRA.BIT.MSTPA15 = 0;（CMT0～CMT1共通） */
    /* CMT2,CMT3のモジュールストップ解除 */
    MSTP(CMT2) = 0;                 /* SYSTEM.MSTPCRA.BIT.MSTPA14 = 0;（CMT2～CMT3共通） */
    /* 動作モード、消費電力低減機能、ソフトウェアリセット関連レジスタへの書き込みを禁止する */
    SYSTEM.PRCR.WORD = 0xA500;

    /* コンペアマッチタイマCMT0,CMT1,CMT2,CMT3停止 */
    CMT.CMSTR0.WORD = 0x0000;
    CMT.CMSTR1.WORD = 0x0000;

    /* コンペアマッチタイマCMT0初期化 */
    CMT0.CMCR.WORD = 0x0040;        /* クロック選択(PCLK/8→2.5MHz),コンペアマッチ割り込み（CMI0）を許可 */
    CMT0.CMCNT = 0x0000;            /* コンペアマッチタイマカウンタクリア */
    CMT0.CMCOR = CONST01M-1;        /* コンペアマッチタイマコンスタントレジスタセット(1mS) */
    /* コンペアマッチタイマCMT1初期化 */
    CMT1.CMCR.WORD = 0x0040;        /* クロック選択(PCLK/8→2.5MHz),コンペアマッチ割り込み（CMI1）を許可 */
    CMT1.CMCNT = 0x0000;            /* コンペアマッチタイマカウンタクリア */
    CMT1.CMCOR = CONST10M-1;        /* コンペアマッチタイマコンスタントレジスタセット(10mS) */
    /* コンペアマッチタイマCMT2初期化 */
    CMT2.CMCR.WORD = 0x0042;        /* クロック選択(PCLK/128→156.25KHz),コンペアマッチ割り込み（CMI2）を許可 */
    CMT2.CMCNT = 0x0000;            /* コンペアマッチタイマカウンタクリア */
    CMT2.CMCOR = CONST50M-1;        /* コンペアマッチタイマコンスタントレジスタセット(50mS) */
    /* コンペアマッチタイマCMT3初期化 */
    CMT3.CMCR.WORD = 0x0042;        /* クロック選択(PCLK/128→156.25KHz),コンペアマッチ割り込み（CMI3）を許可 */
    CMT3.CMCNT = CONST100M/3;       /* コンペアマッチタイマカウンタセット（CMT2と同期しないよう初期値をずらす） */
    CMT3.CMCOR = CONST100M-1;       /* コンペアマッチタイマコンスタントレジスタセット(100mS) */

    /* 割り込み優先順位セット */
    IPR(CMT0, ) = 6;
    IPR(CMT1, ) = 6;
    IPR(CMT2, ) = 6;
    IPR(CMT3, ) = 6;
    /* 割り込み要求ステータスフラグクリア */
    IR(CMT0, CMI0) = 0;
    IR(CMT1, CMI1) = 0;
```

```
    IR(CMT2, CMI2) = 0;
    IR(CMT3, CMI3) = 0;
    /* コンペアマッチ割り込み許可 */
    IEN(CMT0, CMI0) = 1;
    IEN(CMT1, CMI1) = 1;
    IEN(CMT2, CMI2) = 1;
    IEN(CMT3, CMI3) = 1;

    /* コンペアマッチタイマCMT0,CMT1をスタート */
    CMT.CMSTR0.WORD = 0x0003;
    /* コンペアマッチタイマCMT2,CMT3をスタート */
    CMT.CMSTR1.WORD = 0x0003;
}


/************************************************************************/
/*                  CMT0コンペアマッチ割り込み(CMI0)の処理                */
/************************************************************************/
/* ※1mS周期                                                            */
/* ※CMInはエッジ検知割り込みのため、IRフラグは自動的にクリアされる。          */

#pragma interrupt (Int_CMT0_CMI0)
void Int_CMT0_CMI0(void)
{
    /* ここに1mS周期で実行する処理を記述 */

}


/************************************************************************/
/*                  CMT1コンペアマッチ割り込み(CMI1)の処理                */
/************************************************************************/
/* ※10mS周期                                                           */
/* ※CMInはエッジ検知割り込みのため、IRフラグは自動的にクリアされる。          */

#pragma interrupt (Int_CMT1_CMI1)
void Int_CMT1_CMI1(void)
{
    /* ここに10mS周期で実行する処理を記述 */

}

/************************************************************************/
/*                  CMT2コンペアマッチ割り込み(CMI2)の処理                */
/************************************************************************/
/* ※50mS周期                                                           */
/* ※CMInはエッジ検知割り込みのため、IRフラグは自動的にクリアされる。          */

#pragma interrupt (Int_CMT2_CMI2)
void Int_CMT2_CMI2(void)
{
    /* ここに50mS周期で実行する処理を記述 */

}

/************************************************************************/
/*                  CMT3コンペアマッチ割り込み(CMI3)の処理                */
/************************************************************************/
/* ※100mS周期　→ 分周して１秒周期を作成                                */
/* ※CMInはエッジ検知割り込みのため、IRフラグは自動的にクリアされる。          */
static int  st_nDivSecCnt;

#pragma interrupt (Int_CMT3_CMI3)
void Int_CMT3_CMI3(void)
{
    /* ここに100mS周期で実行する処理を記述 */

    st_nDivSecCnt++;
    if(10 <= st_nDivSecCnt){
        st_nDivSecCnt = 0;
        /* ここに1sec周期で実行する処理を記述 */

    }
}

/* End of File */
```