お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。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。
3. 当社製品を改造、改変、複製等しないでください。
4. 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お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、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に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   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

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障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。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、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、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害など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、機器またはシステムとしての出荷保証をお願いいたします。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、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、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。お客様が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。
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固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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その総株主の議決権の過半数を直接または間接に保有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。

サンプルプログラム関数仕様(周辺機能使用例)

使用法説明書（78K0/Kx1,78K0/Kx1+)

本ドキュメントは 78K0/Kx1 および 78K0/Kx1+において各周辺機能の動作を実現する

サンプルソフトウェアの動作について説明しています。

ご注意

本ソフトウェアはあくまで参考用のソフトであり、当社がこの動作を保証するものではありません。

本ソフトウェアを使用する場合、お客様のセット上で十分な評価の上ご使用いただきますようお願いいたします。

NEC Electronics Corporation
2003

目 次

１．概要

本書は７８Ｋ０／Ｋｘ１および７８Ｋ０／Ｋｘ１+用に作成した、サンプル・プログラムの説明書です。

２．作業環境

本サンプル・プログラムを作成した作業環境を下記に示します。

２．１　対象デバイス

７８Ｋ０／Ｋｘ１

２．２　サンプル作成環境

・コンパイラ　：CC78K0　Ver.E3.40h
・アセンブラ　：RA78K0　Ver.E3.51c
・シミュレータ：SM78K0　Ver.E2.52a

・デバイス･ファイル
　　：uPD780148 V1.00

３．プログラム説明

３．１ ポート

3.1.1 CMOS

■ ファイル名

CMOS

- PORT_CMOS.asm（アセンブリ言語ソース）
- PORT_CMOS.c（C言語ソース）

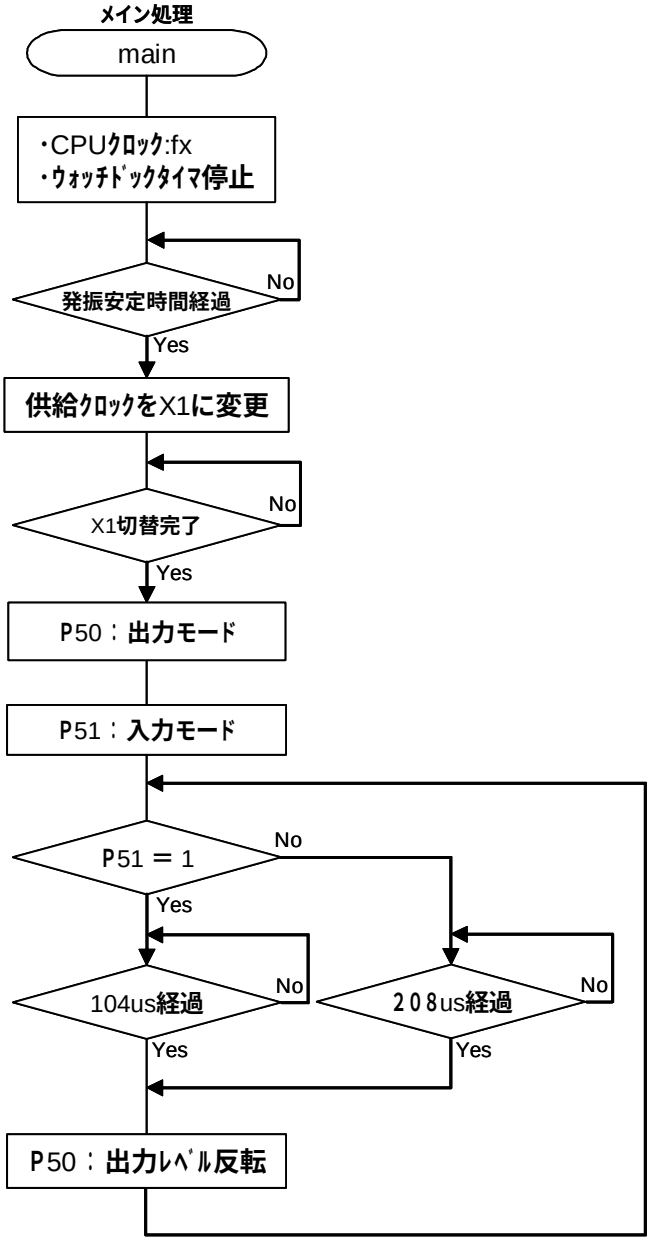
■ 説明

入力および出力の専用ポートと、入出力兼用ポートが用意されています。各ポートは1ビットおよび8ビットの操作ができます。ほとんどのポートは内蔵プルアップ抵抗の使用が可能です。

このプログラムは、メイン処理において P51 が High の時、P50 から 4.88kHz の波形を出力し、P51 が Low の時、P50 から 2.4kHz の波形を出力します。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ（10MHz）を選択します。

プログラムの詳細動作としては、まず X1 入力ｸﾛｯｸ(10MHz)に切り替った事を確認します。その後 P51 端子から入力したレベルを判断します。判断の結果、P51=0 の場合、約 2.4kHz の周波数の波形を出力します。P51=1 の場合、約 4.8kHz の周波数の波形を出力します。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
P50：出力モード
P51：入力モード
P51 ＝ 1
No
Yes
104us経過
No
Yes
208us経過
No
Yes
P50：出力ﾚﾍﾞﾙ反転

■ C言語ソース

```
/*
************************************************************************
*
*  Title: CMOS port control
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - port 5.0 outputs 4.8kHz when port 5.1 = 1
*               - port 5.0 outputs 2.4kHz when port 5.1 = 0
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

/*status list definition*/
#define TRUE            1
#define FALSE           0


/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/

void main( void )
{
    unsigned short  i;

    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0 );
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    PM5.0 = 0;                          /* set P50 output mode */
    PM5.1 = 1;                          /* set P51 input mode */

    while(TRUE)                         /* main loop */
    {
        /* control port level judge */
        if( P5.1 == 1 ){
            for( i=0 ; i<45 ; i++ );    /* approx. delay for 4.8kHz */
        }
        else{
            for( i=0 ; i<92 ; i++ );    /* approx. delay for 2.4kHz */
        }
        P5.0 ^= 1;                      /* Output level reversal */
    }
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: CMOS port control
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - port 5.0 outputs 4.8kHz when port 5.1 = 1
;               - port 5.0 outputs 2.4kHz when port 5.1 = 0
;
;*****************************************************************************


;---------------------------------------------------------------------------
;       Specify Interrupt Vectors
;---------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h   ;On reset, go to Start
                DW      start


;-----------------------------------------------------------------------------
;       Main Program
;-----------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CLR1    PM5.0                   ; set P50 output mode
        SET1    PM5.1

main:
        MOVW    AX, #0000H

        BF      P5.1, $main02

main01:                         ;approx. delay for 4.8kHz
        CMPW    AX, #45
        BNC     $main03
        INCW    AX
        BR      $main01

main02:                         ;approx. delay for 2.4kHz
        CMPW    AX, #92
        BNC     $main03
        INCW    AX
        BR      $main02
main03:

        MOV1    CY, P5.0
        NOT1    CY
```

```
        MOV1    P5.0, CY        ; Output level reversal
        BR      $main

END
```

3.1.2 Ｎｃｈ

■ ファイル名

Nch

- PORT_Nch.asm（アセンブリ言語ソース）
- PORT_Nch.c（Ｃ言語ソース）

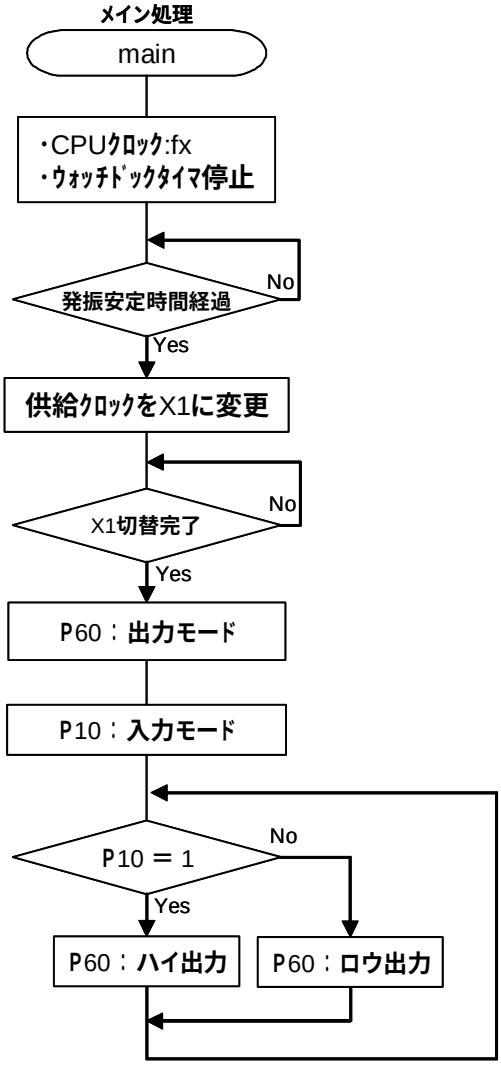
■ 説明

入力および出力の専用ポートと、入出力兼用ポートが用意されています。各ポートは 1 ビットおよび 8 ビットの操作ができます。ほとんどのポートは内蔵プルアップ抵抗の使用が可能です。

このプログラムは、メイン処理において P10 が High の時、P60 に High 出力し、P10 が Low の時、P60 に Low 出力を行います。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ（10MHz）を選択します。

プログラムの詳細動作としては、まず X1 入力ｸﾛｯｸ(10MHz)に切り替った事を確認します。その後 P10 端子から入力したレベルを判断します。判断の結果、P10=0 の場合、P60 から Low レベルを出力します。P10=1 の場合、P60 から High レベルを出力します。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
No
発振安定時間経過
Yes
供給ｸﾛｯｸをX1に変更
No
X1切替完了
Yes
P60：出力モード
P10：入力モード
No
P10 ＝ 1
Yes
P60：ハイ出力
P60：ロウ出力

■ C言語ソース

```
/*
************************************************************************
*
*  Title: Nch port control
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - port 6.0 outputs high level when port 1.0 = 1
*               - port 6.0 outputs low level when port 1.0 = 0
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

/*status list definition*/
#define TRUE            1
#define FALSE           0


/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/

void main( void )
{
    unsigned short  i;

    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    PM6.0 = 0;                          /* set P60 output mode */
    PM1.0 = 1;                          /* set P10 input mode */

    while(TRUE)                         /* main loop */
    {
        /* control port level judge */
        if( P1.0 == 1 ){
            P6.0 = 1;                   /* P60=1 */
        }
        else{
            P6.0 = 0;                   /* P60=0 */
        }
    }
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: Nch port control
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - port 6.0 outputs high level when port 1.0 = 1
;               - port 6.0 outputs low level when port 1.0 = 0
;
;*****************************************************************************


;---------------------------------------------------------------------------
;       Specify Interrupt Vectors
;---------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h   ;On reset, go to Start
                DW      start


;-----------------------------------------------------------------------------
;       Main Program
;-----------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CLR1    PM6.0
        SET1    PM1.0

main:                                   ; main loop
        BF      P1.0, $main01
        SET1    P6.0
        BR      $main

main01:
        CLR1    P6.0
        BR      $main

END
```

３．２ クロック発生

■ ファイル名

- CLK_Generate.asm（アセンブリ言語ソース）
- CLK_Generate.c（C言語ソース）

■ 説明

クロック発生回路は，CPUおよび周辺ハードウエアに供給するクロックを発生する回路です。
システム・クロック発振回路には，次の３種類があります。

（1）X1発振回路

2.0～10.0 MHzのクロックを発振します。STOP命令の実行またはメインOSCコントロール・レジスタ（MOC），プロセッサ・クロック・コントロール・レジスタ（PCC）の設定により，発振を停止することができます。

（2）Ring-OSC発振回路

240 kHz（TYP.）のクロックを発振します。マスク・オプションで「ソフトウエアにより停止可能」に設定し，CPUクロックがX1入力クロックの場合，Ring-OSCモード・レジスタ（RCM）の設定により，発振を停止することができます。

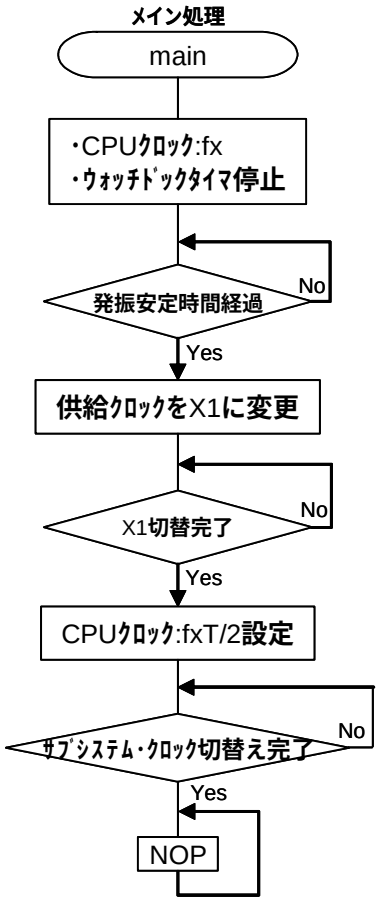
（3）サブシステム・クロック発振回路

32.768 kHzの周波数を発振します。発振の停止はできません。サブシステム・クロック発振回路を使用しないとき，プロセッサ・クロック・コントロール・レジスタ（PCC）により，内蔵フィードバック抵抗を使用しない設定ができます。これによって，STOPモード時の消費電力を低減できます。

このプログラムは、メイン処理において「Ring-OSC」→「メインクロック」→「サブクロック」の順番で動作クロックを変更します（動作クロックは、ステータスを確認してから切り替えます）。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ（10MHz）を選択します。

プログラムの詳細動作としてはリセット解除後、発振安定時間を待つために発振安定時間カウンタ状態レジスタ(OSTC)を確認します（6.55ms 以上経過を待つ）。 時間経過後、CPU への供給クロックを X1 入力クロックに設定します。CPU クロックのステータス(MCS)を確認しながら X1 入力クロックに切り替わるのを待ちます。 X1 入力クロックに切り替わったことを確認後、CPU への供給クロックをサブクロックに設定します。その後 CPU クロックのステータス(CLS)を確認しながらサブクロックに切り替わるのを待ちます。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
No
発振安定時間経過
Yes
供給ｸﾛｯｸをX1に変更
No
X1切替完了
Yes
CPUｸﾛｯｸ:fxT/2設定
No
ｻﾌﾞｼｽﾃﾑ・ｸﾛｯｸ切替え完了
Yes
NOP

■ C言語ソース

```
/*
************************************************************************
*
*  Title: Clock generating
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - subsystem clock 32.768kHz
*               - Ring-OSC clock 240kHz
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

/*status list definition*/
#define TRUE            1
#define FALSE           0


/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/

void main( void )
{

    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0 );
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    CSS = 1;                            /* enable sub clock */
    /* Waiting for sub clock change */
    while( CLS == 0 );

    while(TRUE)                         /* Endless loop */
    {
        NOP();
    }
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: Clock generating
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - subsystem clock 32.768kHz
;               - Ring-OSC clock 240kHz
;
;*****************************************************************************


;---------------------------------------------------------------------------
;       Specify Interrupt Vectors
;---------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ;On reset, go to Start
                DW      start


;-----------------------------------------------------------------------------
;       Main Program
;-----------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        SET1    CSS                     ; enable sub clock
start10:
        BT      CLS, $start11
        BR      $start10
start11:

main:                                   ; Endless loop
        NOP
        BR      $main

END
```

３．３ 16 ビット･タイマ／イベント･カウンタ００

3.3.1 インターバル･タイマ

■ ファイル名

Interval

- TM0_Interval.asm（アセンブリ言語ソース）
- TM0_Interval.c（C言語ソース）

■ 説明

16ビット・タイマ／イベント・カウンタ00，01[注]には，次のような機能があります。

（1）インターバル・タイマ

あらかじめ設定した任意の時間間隔で割込み要求を発生します。

（2）PPG出力

周波数と出力パルス幅を任意に設定できる矩形波を出力できます。

（3）パルス幅測定

外部から入力される信号のパルス幅を測定できます。

（4）外部イベント・カウンタ

外部から入力される信号のパルス数を測定できます。

（5）方形波出力

任意の周波数の方形波出力が可能です。

（6）ワンショット・パルス出力

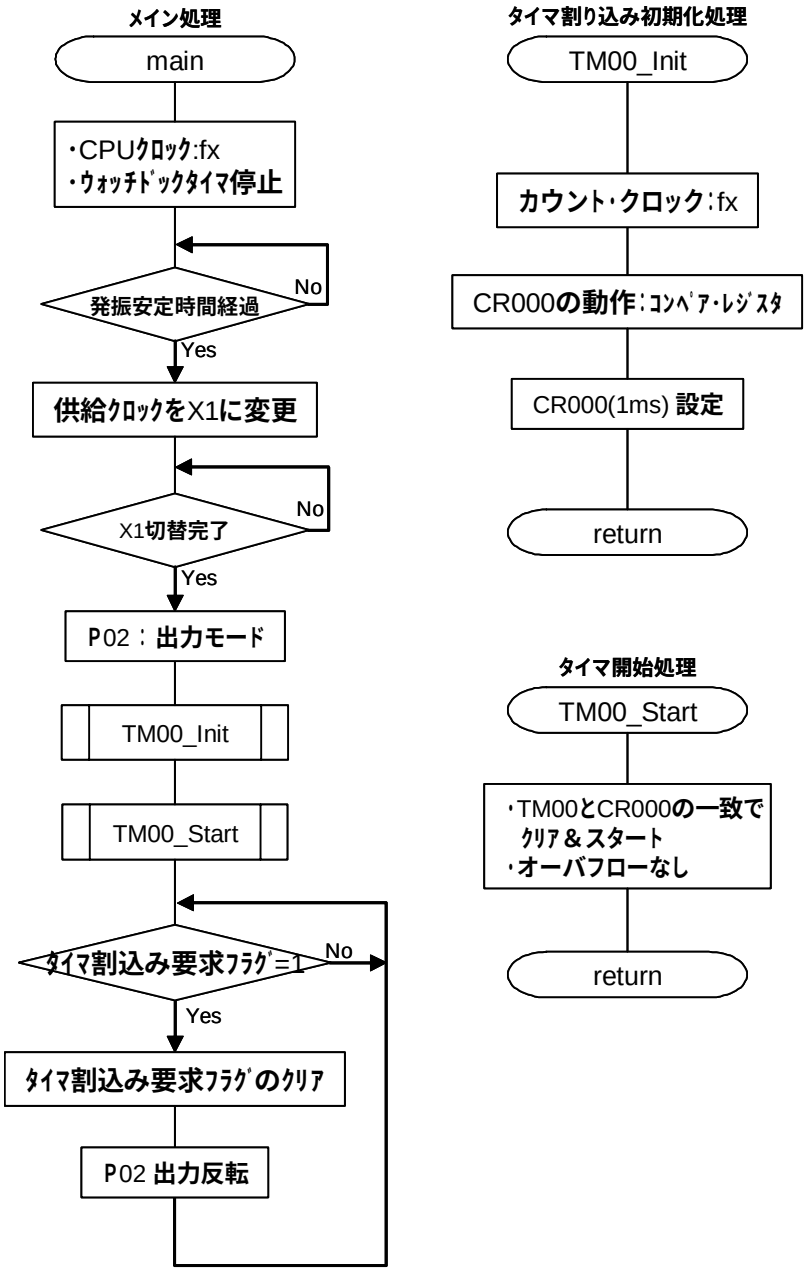
出力パルス幅を任意に設定できるワンショット・パルスを出力できます。

注 μPD780146，780148，78F0148のみ。

このプログラムは、1ms 毎にインターバルモードにより INTTM00 を発生させます。そして、メイン処理にてフラグセンスにより１ms 経過を判断し、P02 を反転出力します。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ（10MHz）を選択します。

初期設定として、タイマのカウントクロックは 10MHz が選択され、かつ 16 ビット・タイマ・キャプチャ／コンペアレジスタ 000(CR000)は 9999 に設定されています。タイマの動作開始後、タイマ・カウンタがコンペアレジスタの値（1ms）と一致した時、タイマ割込みの要求フラグ（TMIF000）がセットされます。メインループは、このタイマ割込みのリクエストフラグがセットされたことを検知すると、P02 のレベルを反転出力させます。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
P02：出力モード
TM00_Init
TM00_Start
ﾀｲﾏ割込み要求ﾌﾗｸﾞ=1
No
Yes
ﾀｲﾏ割込み要求ﾌﾗｸﾞのｸﾘｱ
P02 出力反転
タイマ割り込み初期化処理
TM00_Init
カウント・クロック：fx
CR000の動作：ｺﾝﾍﾟｱ・ﾚｼﾞｽﾀ
CR000(1ms) 設定
return
タイマ開始処理
TM00_Start
・TM00とCR000の一致で
ｸﾘｱ＆スタート
・オーバフローなし
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: Interval timer by TM00
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - timer: 16-bit timer/event counter TM00
*               - timer count clock: fx (10MHz)
*               - compare register value (CR000): 0x270f
*               - interval time: 1.0 ms
*               - interrupt handling: polling of interrupt request flag (TMIF000)
*               - output port: port 0.2 (toggles every 1.0 ms)
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

#define TM_TM00_INTERVALVALUE   0x270f

/*status list definition*/
#define TRUE            1
#define FALSE           0

void TM00_Init(void);
void TM00_Start();


/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/

void main( void )
{

    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    PM0.2 = 0;                          /* P02=0 */

    TM00_Init();                        /* TM00 initialization function */
    TM00_Start();                       /* TM00 start function */


    while(TRUE)                         /* main loop */
    {
        /* Waiting for timer request flag */
        while( TMIF000 == 0 )
        {
            NOP();
```

```
        }
        TMIF000 = 0;                    /* No interrupt request signal is generated */
        P0.2 ^= 1;                      /* P0.2 output reversal */
    }
}


/*
-------------------------------------------------------------------------------
    Functions Group
-------------------------------------------------------------------------------
*/

/*
-------------------------------------------------------------------------------
  Abstract:
    Initiate TM00, select function and input parameter
    count clock selection, INT init
-------------------------------------------------------------------------------
*/
void TM00_Init( void )
{
    PRM00 = 0x00;                   /* internal count clock */

    /*timer00 interval*/
    CRC00 = 0x00;                   /* CR010 operating mode : Operates as compare register */
                                    /* CR000 operating mode : Operates as compare register */
                                    /* CR000 capture trigger: Captures on valid edge of TI010 */

    CR000 = TM_TM00_INTERVALVALUE;
}


/*
-------------------------------------------------------------------------------
  Abstract:
    Start the timer00 counter
-------------------------------------------------------------------------------
*/
void TM00_Start()
{
    TMC00 = 0x0c;                   /* Clear & start occurs on match between TM00 and CR000 */
}
```

■ アセンブリ言語ソース

```
;*****************************************************************************
;
;  Title: Interval timer by TM00
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - timer: 16-bit timer/event counter TM00
;               - timer count clock: fx (10MHz)
;               - compare register value (CR000): 0x270f
;               - interval time: 1.0 ms
;               - interrupt handling: polling of interrupt request flag (TMIF000)
;               - output port: port 0.2 (toggles every 1.0 ms)
;
;*****************************************************************************


;---------------------------------------------------------------------------
;       Macro Define
;---------------------------------------------------------------------------

TM_TM00_INTERVALVALUE   EQU     270fH


;---------------------------------------------------------------------------
;       Specify Interrupt Vectors
;---------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to Start
                DW      start


;---------------------------------------------------------------------------
;       Main Program
;---------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CLR1    PM0.2

        CALL    !TM00_Init              ; Waiting for X1 clock change
        CALL    !TM00_Start             ; TM00 start function

main:                                   ; main loop
        BTCLR   TMIF000, $main00        ; Waiting for timer request flag
        NOP
        BR      $main

main00:
```

```
        MOV1    CY, P0.2
        NOT1    CY
        MOV1    P0.2, CY                ; P0.2 output reversal

        BR      $main


;---------------------------------------------------------------------------
;       Functions Group
;---------------------------------------------------------------------------


;---------------------------------------------------------------------------
;  Abstract:
;       Initiate TM00, select founction and input parameter
;       count clock selection, INT init
;---------------------------------------------------------------------------
TM00_Init:
        MOV     PRM00, #00H             ; internal count clock

        ;Timer00 interval
        MOV     CRC00, #00H
        MOVW    CR000, #TM_TM00_INTERVALVALUE

        RET


;---------------------------------------------------------------------------
;  Abstract:
;       Start the timer00 counter
;---------------------------------------------------------------------------
TM00_Start:
        MOV     TMC00, #0CH             ; Clear & start occurs on match between TM00 and CR000

        RET

END
```

3.3.2 ＰＰＧ出力

■ ファイル名

PPG_out

- TM0_PPGout.asm（アセンブリ言語ソース）
- TM0_PPGout.c（Ｃ言語ソース）

■ 説明

16ビット・タイマ／イベント・カウンタ00，01[注]には，次のような機能があります。

（1）インターバル・タイマ

あらかじめ設定した任意の時間間隔で割込み要求を発生します。

（2）PPG出力

周波数と出力パルス幅を任意に設定できる矩形波を出力できます。

（3）パルス幅測定

外部から入力される信号のパルス幅を測定できます。

（4）外部イベント・カウンタ

外部から入力される信号のパルス数を測定できます。

（5）方形波出力

任意の周波数の方形波出力が可能です。

（6）ワンショット・パルス出力

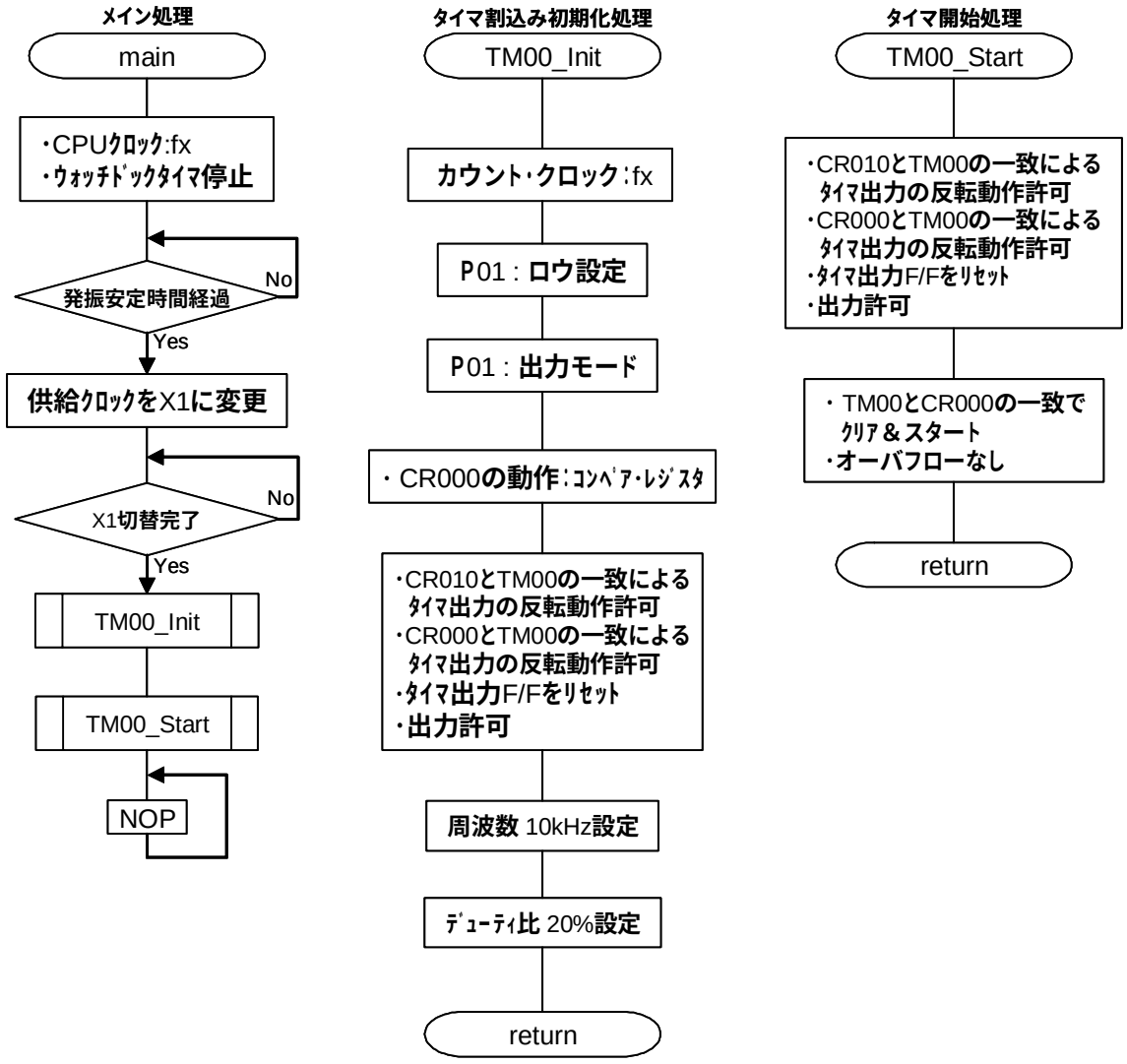
出力パルス幅を任意に設定できるワンショット・パルスを出力できます。

注 μPD780146，780148，78F0148のみ。

このプログラムは、P01/TO00 より PPG モードを使用して、10KHz、ﾃﾞｭｰﾃｨ比 20%(High=80μs, Low=20μs)の波形を出力します。そしてメイン処理は、出力設定後無限ループにします。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ（10MHz）を選択します。

初期設定として、タイマのカウントクロックは 10MHz を選択します。そして、コンペアレジスタ(CR010)は PPG 出力期間を決定し、999 までのカウントで 100μs(10kHz)期間を作成します。コンペアレジスタ(CR000)はパルスの幅を決定し、199 までのカウントで 20μs(ﾃﾞｭｰﾃｨ比 20%)のパルス幅を作成します。タイマの実行開始後、コンペアレジスタとカウント値が一致した時、出力ポート(TO00)の出力レベルを反転します。まず、タイマが０からのスタート後、CR010 と値が一致したとすると出力が逆になります。その後、カウントは継続して進み CR000 と一致した時点で出力を再び逆にした後タイマはクリアされ０から再カウントされます。このオペレーションはプログラムからの介在なしに連続的に行われます。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
TM00_Init
TM00_Start
NOP
タイマ割込み初期化処理
TM00_Init
カウント・クロック：fx
P01：ロウ設定
P01：出力モード
・CR000の動作：ｺﾝﾍﾟｱ・ﾚｼﾞｽﾀ
・CR010とTM00の一致による
ﾀｲﾏ出力の反転動作許可
・CR000とTM00の一致による
ﾀｲﾏ出力の反転動作許可
・ﾀｲﾏ出力F/Fをﾘｾｯﾄ
・出力許可
周波数 10kHz設定
ﾃﾞｭｰﾃｨ比 20%設定
return
タイマ開始処理
TM00_Start
・CR010とTM00の一致による
ﾀｲﾏ出力の反転動作許可
・CR000とTM00の一致による
ﾀｲﾏ出力の反転動作許可
・ﾀｲﾏ出力F/Fをﾘｾｯﾄ
・出力許可
・TM00とCR000の一致で
ｸﾘｱ＆スタート
・オーバフローなし
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: PPG output by TM00
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - timer: 16-bit timer/event counter TM00
*               - timer count clock: fx (10MHz)
*               - compare register CR000 value: 0x03e7
*               - compare register CR010 value: 0x00c7
*               - output port: port 0.1 (P01/TO00)
*               - PPG output frequency: 10kHz (100us period)
*               - PPG duty cycle: 20% (20us pulse)
*
************************************************************************
*/
#pragma sfr
#pragma NOP


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

#define TM_TM00_PPGCYCLE    0x3e7
#define TM_TM00_PPGWIDTH    0xc7

/*status list definition*/
#define TRUE            1
#define FALSE           0

void TM00_Init(void);
void TM00_Start();


/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/

void main( void )
{
    PCC = 0x00;                         /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    TM00_Init();                        /* TM00 initialization function */
    TM00_Start();                       /* TM00 start function */

    while(TRUE)                         /* Endless loop */
    {
        NOP();
    }
}

/*
------------------------------------------------------------------------
    Functions Group
```

```
------------------------------------------------------------------------------
*/

/*
------------------------------------------------------------------------------
  Abstract:
    Initiate TM00, select founction and input parameter
    count clock selection, INT init
------------------------------------------------------------------------------
*/
void TM00_Init( void )
{
    PRM00 = 0x00;                       /* internal count clock */
    /*timer00 PPG output function*/

    P0.1 = 0;                           /* P01=0 */
    PM0.1 = 0;                          /* set P01 is output mode */

    CRC00 = 0x00;                       /* CR010 operating mode : Operates as compare register */
                                        /* CR000 operating mode : Operates as compare register */
                                        /* CR000 capture trigger: Captures on valid edge of TI010 */
    TOC00 = 0x17;                       /* init low */
    CR000 = TM_TM00_PPGCYCLE;
    CR010 = TM_TM00_PPGWIDTH;
}


/*
------------------------------------------------------------------------------
  Abstract:
    Start the timer00 counter
------------------------------------------------------------------------------
*/
void TM00_Start( void )
{
    TOC00=0x17;                         /* init low */
    TMC00=0x0c;                         /* Clear & start occurs on match between TM00 and CR000 */
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;*
;*  Title: PPG output by TM00
;*
;*  Parameters: - fastest CPU clock (fx=10MHz)
;*              - timer: 16-bit timer/event counter TM00
;*              - timer count clock: fx (10MHz)
;*              - compare register CR000 value: 03e7H
;*              - compare register CR010 value: 00c7H
;*              - output port: port 0.1 (P01/TO00)
;*              - PPG output frequency: 10kHz (100us period)
;*              - PPG duty cycle: 20% (20us pulse)
;*
;*****************************************************************************
;



;-----------------------------------------------------------------------------
;       Macro Define
;-----------------------------------------------------------------------------


TM_TM00_PPGCYCLE        EQU     03e7H
TM_TM00_PPGWIDTH        EQU     00c7H


;-----------------------------------------------------------------------------
;       Specify Interrupt Vectors
;-----------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to  Start
        DW      start


;-----------------------------------------------------------------------------
;       Main Program
;-----------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CALL    !TM00_Init              ; TM00 initialization function
        CALL    !TM00_Start             ; TM00 start function

main:                                   ; Endless loop
        NOP
        BR      $main
```

```
;-------------------------------------------------------------------------------
;       Functions Group
;-------------------------------------------------------------------------------


;-------------------------------------------------------------------------
;  Abstract:
;       Initiate TM00, select founction and input parameter
;       count clock selection, INT init
;-------------------------------------------------------------------------
TM00_Init:
        MOV     PRM00, #00H             ; internal count clock
        ;Timer00 PPG output function

        CLR1    P0.1
        CLR1    PM0.1

        MOV     CRC00, #00H
        MOV     TOC00, #17H             ; init low
        MOVW    CR000, #TM_TM00_PPGCYCLE
        MOVW    CR010, #TM_TM00_PPGWIDTH

        RET


;-------------------------------------------------------------------------
;  Abstract:
;       Start the timer00 counter
;-------------------------------------------------------------------------
TM00_Start:
        MOV     TOC00, #17H
        MOV     TMC00, #0CH             ; Clear & start occurs on match between TM00 and CR000

        RET

END
```

3.3.3 ＰＰＧ切替え

■ ファイル名

PPG_change

- TM0_PPGcng.asm（アセンブリ言語ソース）
- TM0_PPGcng.c（Ｃ言語ソース）

■ 説明

16ビット・タイマ／イベント・カウンタ00，01[注]には，次のような機能があります。

（1）インターバル・タイマ

あらかじめ設定した任意の時間間隔で割込み要求を発生します。

（2）PPG出力

周波数と出力パルス幅を任意に設定できる矩形波を出力できます。

（3）パルス幅測定

外部から入力される信号のパルス幅を測定できます。

（4）外部イベント・カウンタ

外部から入力される信号のパルス数を測定できます。

（5）方形波出力

任意の周波数の方形波出力が可能です。

（6）ワンショット・パルス出力

出力パルス幅を任意に設定できるワンショット・パルスを出力できます。

注 μPD780146，780148，78F0148のみ。

このプログラムは、P40 端子の入力状態（P40=0(Low)：20%，P40=1(High)：50%）により、PPG モードで P01/TO00 から出力する１0KHz 出力波形の、ﾃﾞｭｰﾃｨ比を変更します。ただし P40 端子の入力状態は、TM00 の割り込み処理内で判定します。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ（10MHz）を選択します。

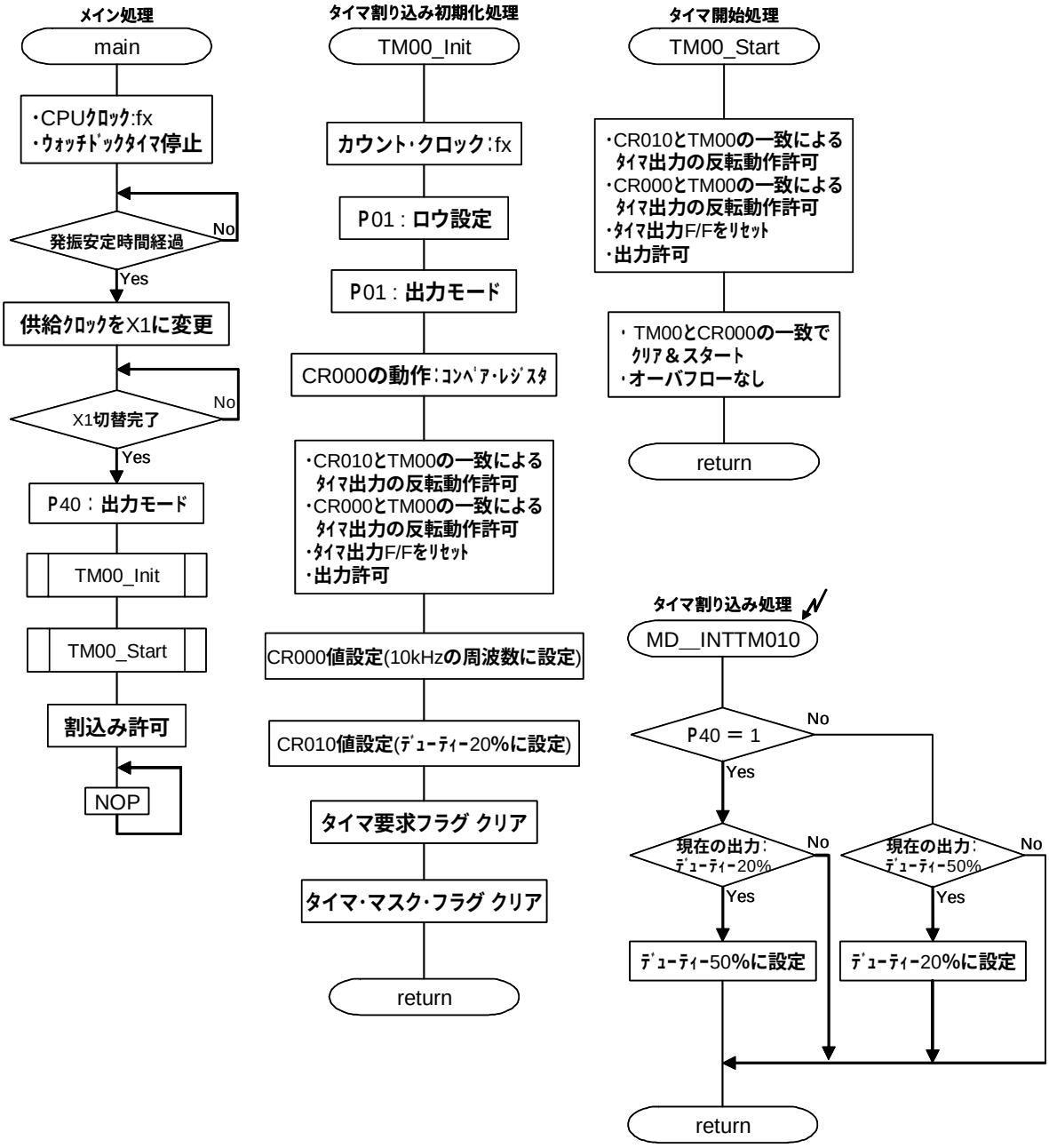
タイマのカウントクロックは 10MHz を選択します。そしてﾃﾞｭｰﾃｨ比の切り替え用の P40 端子の入力レベルを判断し、そのレベルに応じて P01/TO00 から、10kHz の周波数で２種類のﾃﾞｭｰﾃｨ比の波形を切り替えて出力します。ﾃﾞｭｰﾃｨ比切り替えタイミングは、16 ビット・タイマ・キャプチャ／コンペアレジスタ 000(CR000)に設定された値とタイマ・カウンタ 00(TM00)のカウント値が一致した時に割込みを発生させ、割込み処理内にて 16 ビット・タイマ・キャプチャ／コンペアレジスタ 010(CR010)の設定値を変更することで行います。

CR010 の初期値はﾃﾞｭｰﾃｨ比 20%に設定されます。

P40=0 の時、現在の出力設定がﾃﾞｭｰﾃｨ比 50%なら、ﾃﾞｭｰﾃｨ比を 20%に変更します。

P40=1 の時、現在の出力設定がﾃﾞｭｰﾃｨ比 20%なら、ﾃﾞｭｰﾃｨ比を 50%に変更します。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
P40：出力モード
TM00_Init
TM00_Start
割込み許可
NOP
タイマ割り込み初期化処理
TM00_Init
カウント・クロック:fx
P01：ロウ設定
P01：出力モード
CR000の動作:ｺﾝﾍﾟｱ・ﾚｼﾞｽﾀ
・CR010とTM00の一致による
ﾀｲﾏ出力の反転動作許可
・CR000とTM00の一致による
ﾀｲﾏ出力の反転動作許可
・ﾀｲﾏ出力F/Fをﾘｾｯﾄ
・出力許可
CR000値設定(10kHzの周波数に設定)
CR010値設定(ﾃﾞｭｰﾃｨｰ20%に設定)
タイマ要求フラグ クリア
タイマ・マスク・フラグ クリア
return
タイマ開始処理
TM00_Start
・CR010とTM00の一致による
ﾀｲﾏ出力の反転動作許可
・CR000とTM00の一致による
ﾀｲﾏ出力の反転動作許可
・ﾀｲﾏ出力F/Fをﾘｾｯﾄ
・出力許可
・TM00とCR000の一致で
ｸﾘｱ＆スタート
・オーバフローなし
return
タイマ割り込み処理
MD__INTTM010
P40 ＝ 1
No
Yes
現在の出力:
ﾃﾞｭｰﾃｨｰ20%
No
Yes
現在の出力:
ﾃﾞｭｰﾃｨｰ50%
No
Yes
ﾃﾞｭｰﾃｨｰ50%に設定
ﾃﾞｭｰﾃｨｰ20%に設定
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: PPG output waveform change by TM00
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - timer: 16-bit timer/event counter TM00
*               - timer count clock: fx (10MHz)
*               - output port: port 0.1 (P01/TO00)
*               - PPG output 10kHz ( PPG duty cycle 20% ) when port 4.0 = 0
*               - PPG output 10kHz ( PPG duty cycle 50% ) when port 4.0 = 1
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI

/*
------------------------------------------------------------------------------
    Specify Interrupt Vectors
------------------------------------------------------------------------------
*/
#pragma interrupt    INTTM010        MD_INTTM010


/*
------------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------------
*/

#define TM_TM00_PPGCYCLE    0x3e7
#define TM_TM00_PPGWIDTH20  0x0c7
#define TM_TM00_PPGWIDTH50  0x1f3

/*status list definition*/
#define TRUE            1
#define FALSE           0

void TM00_Init(void);
void TM00_Start();


/*
------------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------------
*/

void main( void )
{

    PCC = 0x00;                         /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    PM4.0 = 1;                          /* set P40 is input mode */

    TM00_Init();                        /* TM00 initialization function */
    TM00_Start();                       /* TM00 start function */
```

```
    EI();                               /* Master interruption permission */

    while(TRUE)                         /* Endless loop */
    {
        NOP();
    }

}


/*
--------------------------------------------------------------------------------
    Functions Group
--------------------------------------------------------------------------------
*/

/*
--------------------------------------------------------------------------------
  Abstract:
    Initiate TM00, select founction and input parameter
    count clock selection, INT init
--------------------------------------------------------------------------------
*/
void TM00_Init( void )
{
    PRM00 = 0x00;                       /* internal count clock */
    /*timer00 PPG output function*/

    P0.1 = 0;                           /* P01=0 */
    PM0.1 = 0;                          /* set P01 is output mode */
    CRC00 = 0x00;                       /* CR010 operating mode : Operates as compare register */
                                        /* CR000 operating mode : Operates as compare register */
                                        /* CR000 capture trigger: Captures on valid edge of TI010 */
    TOC00 = 0x17;                       /* init low */
    CR000 = TM_TM00_PPGCYCLE;
    CR010 = TM_TM00_PPGWIDTH20;
    TMIF010 = 0;                        /* No interrupt request signal is generated */
    TMMK010 = 0;                        /* Interrupt servicing enabled */
}


/*
--------------------------------------------------------------------------------
  Abstract:
    Start the timer00 counter
--------------------------------------------------------------------------------
*/
void TM00_Start( void )
{
    TOC00 = 0x17;                       /* init low */
    TMC00 = 0x0c;                       /* Clear & start occurs on match between TM00 and CR000 */
}


/*
--------------------------------------------------------------------------------
  Abstract:
    TM00 INTTM010 interrupt service routine
--------------------------------------------------------------------------------
*/
__interrupt void    MD_INTTM010( void )
{
    /* control port level judge */
    if( P4.0 == 1 )
    {
        if( CR010 == TM_TM00_PPGWIDTH20 )
        {
            TMIF000 = 0;                /* No interrupt request signal is generated */
```

```
            TOC00.4 = 0;                /* Disables inversion operation */
            TMMK010 = 1;                /* Interrupt servicing disabled */
            CR010 = TM_TM00_PPGWIDTH50;

            while( TMIF000 == 0 );      /* waiting for interrupt request */
            TOC00.4 = 1;                /* Enables inversion operation */
            TMIF010 = 0;                /* No interrupt request signal is generated */
            TMMK010 = 0;                /* Interrupt servicing enabled */
        }
    }
    else
    {
        if( CR010 == TM_TM00_PPGWIDTH50 )
        {
            TMIF000 = 0;                /* No interrupt request signal is generated */
            TOC00.4 = 0;                /* Disables inversion operation */
            TMMK010 = 1;                /* Interrupt servicing disabled */
            CR010 = TM_TM00_PPGWIDTH20;

            while( TMIF000 == 0 );      /* waiting for interrupt request */
            TOC00.4 = 1;                /* Enables inversion operation */
            TMIF010 = 0;                /* No interrupt request signal is generated */
            TMMK010 = 0;                /* Interrupt servicing enabled */
        }
    }
}
```

■ アセンブリ言語ソース

```
;
;*******************************************************************************
;
;  Title: PPG output waveform change by TM00
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - timer: 16-bit timer/event counter TM00
;               - timer count clock: fx (10MHz)
;               - output port: port 0.1 (P01/TO00)
;               - PPG output 10kHz ( PPG duty cycle 20% ) when port 4.0 = 0
;               - PPG output 10kHz ( PPG duty cycle 50% ) when port 4.0 = 1
;
;*******************************************************************************



;-------------------------------------------------------------------------------
;       Macro Define
;-------------------------------------------------------------------------------


TM_TM00_PPGCYCLE        EQU     03e7H
TM_TM00_PPGWIDTH20      EQU     00c7H
TM_TM00_PPGWIDTH50      EQU     01f3H


;-------------------------------------------------------------------------------
;       Specify Interrupt Vectors
;-------------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to Start
                DW      start
TM010_VECTOR    CSEG AT 0022h
                DW      MD_INTTM010


;-------------------------------------------------------------------------------
;       Main Program
;-------------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        SET1    PM4.0

        CALL    !TM00_Init              ; TM00 initialization function
        CALL    !TM00_Start             ; TM00 start function

        EI                              ; Master interruption permission
```

```
main:                                   ; Endless loop
        NOP
        BR      $main


;---------------------------------------------------------------------------
;       Functions Group
;---------------------------------------------------------------------------


;---------------------------------------------------------------------------
;  Abstract:
;       Initiate TM00, select founction and input parameter
;       count clock selection, INT init
;---------------------------------------------------------------------------
TM00_Init:
        MOV     PRM00, #00H             ; internal count clock

        ;Timer00 PPG output function
        CLR1    P0.1
        CLR1    PM0.1

        MOV     CRC00, #00H
        MOV     TOC00, #17H             ; init low
        MOVW    CR000, #TM_TM00_PPGCYCLE
        MOVW    CR010, #TM_TM00_PPGWIDTH20

        CLR1    TMIF010
        CLR1    TMMK010                 ; Interrupt servicing enabled

        RET


;---------------------------------------------------------------------------
;  Abstract:
;       Start the timer00 counter
;---------------------------------------------------------------------------
TM00_Start:
        MOV     TOC00, #17H
        MOV     TMC00, #0CH             ; Clear & start occurs on match between TM00 and CR000

        RET


;---------------------------------------------------------------------------
;  Abstract:
;       TM00 INTTM010 interrupt service routine
;---------------------------------------------------------------------------
MD_INTTM010:
        SEL     RB1

        BF      P4.0, $MD_INTTM010_10
        MOVW    AX, CR010
        CMPW    AX, #TM_TM00_PPGWIDTH20
        BNZ     $MD_INTTM010_END

        CLR1    TMIF000
        CLR1    TOC00.4                         ; Disables inversion operation
        SET1    TMMK010
        MOVW    CR010, #TM_TM00_PPGWIDTH50

MD_INTTM010_01:
        BT      TMIF000, $MD_INTTM010_02        ; waiting for interrupt request
        BR      $MD_INTTM010_01
MD_INTTM010_02:

        SET1    TOC00.4                         ; Enables inversion operation
        CLR1    TMIF010
        CLR1    TMMK010
```

```
        BR      MD_INTTM010_END

MD_INTTM010_10:
        MOVW    AX, CR010
        CMPW    AX, #TM_TM00_PPGWIDTH50
        BNZ     $MD_INTTM010_END

        CLR1    TMIF000
        CLR1    TOC00.4                 ; Disables inversion operation
        SET1    TMMK010
        MOVW    CR010, #TM_TM00_PPGWIDTH20

MD_INTTM010_11:
        BT      TMIF000, $MD_INTTM010_12        ; waiting for interrupt request
        BR      $MD_INTTM010_11
MD_INTTM010_12:

        SET1    TOC00.4                 ; Enables inversion operation
        CLR1    TMIF010
        CLR1    TMMK010

MD_INTTM010_END:
        RETI

END
```

3.3.4 パルス幅測定

■ ファイル名

Capture

- TM0_Capture.asm（アセンブリ言語ソース）
- TM0_Capture.c（C言語ソース）

■ 説明

16ビット・タイマ／イベント・カウンタ00，01[注]には，次のような機能があります。

（1）インターバル・タイマ

あらかじめ設定した任意の時間間隔で割込み要求を発生します。

（2）PPG出力

周波数と出力パルス幅を任意に設定できる矩形波を出力できます。

（3）パルス幅測定

外部から入力される信号のパルス幅を測定できます。

（4）外部イベント・カウンタ

外部から入力される信号のパルス数を測定できます。

（5）方形波出力

任意の周波数の方形波出力が可能です。

（6）ワンショット・パルス出力

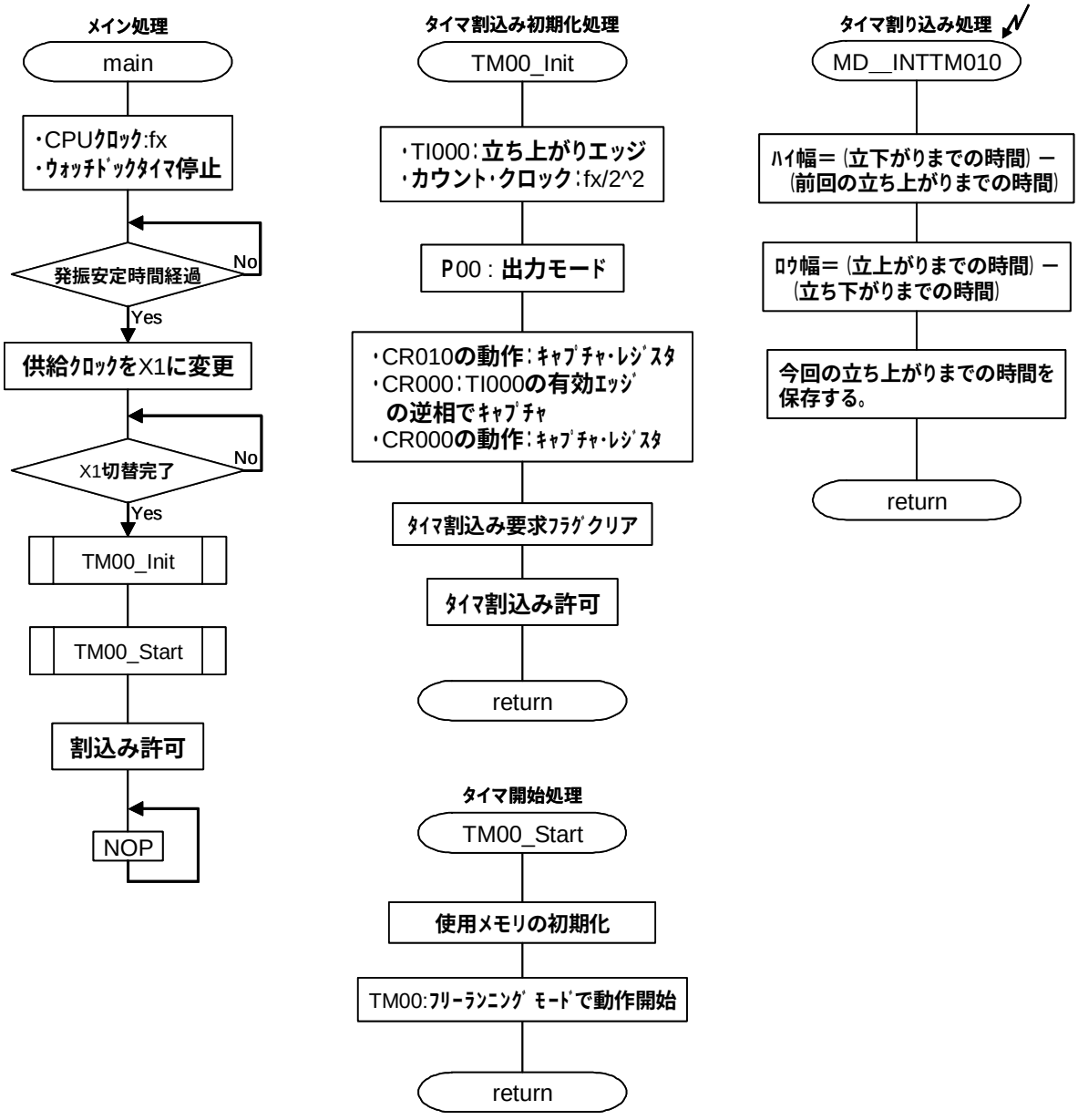
出力パルス幅を任意に設定できるワンショット・パルスを出力できます。

注 μPD780146，780148，78F0148のみ。

このプログラムは、P00/TI00 に入力される波形の各 High, Low 区間の幅を、割込み処理内で測定し、データを格納します。そしてメイン処理は、出力設定後無限ループにします。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

タイマへのカウントクロックは2.5MHz を選択しています。キャプチャ動作の有効エッジはP00/TIO00 の立ち上がりで、INTTM010 割込みを使用しています。タイマの動作モードは、フリーランニング・モードを使用し、キャプチャレジスタは２個使用してパルスの High と Low の期間を計測しています。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
TM00_Init
TM00_Start
割込み許可
NOP
タイマ割込み初期化処理
TM00_Init
・TI000:立ち上がりエッジ
・カウント・クロック:fx/2^2
P00 : 出力モード
・CR010の動作:ｷｬﾌﾟﾁｬ・ﾚｼﾞｽﾀ
・CR000:TI000の有効ｴｯｼﾞ
の逆相でｷｬﾌﾟﾁｬ
・CR000の動作:ｷｬﾌﾟﾁｬ・ﾚｼﾞｽﾀ
ﾀｲﾏ割込み要求ﾌﾗｸﾞｸﾘｱ
ﾀｲﾏ割込み許可
return
タイマ開始処理
TM00_Start
使用メモリの初期化
TM00:ﾌﾘｰﾗﾝﾆﾝｸﾞ ﾓｰﾄﾞで動作開始
return
タイマ割り込み処理
MD__INTTM010
ﾊｲ幅＝(立下がりまでの時間)－
(前回の立ち上がりまでの時間)
ﾛｳ幅＝(立上がりまでの時間)－
(立ち下がりまでの時間)
今回の立ち上がりまでの時間を
保存する。
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: Pulse width measurement by TM00
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - timer: 16-bit timer/event counter TM00
*               - timer count clock: fx/4(2.5MHz)
*               - capture register: 16-bit register CR000
*               - capture trigger: rising edge on port 0.0 (P00/TI000)
*               - interrupt: rising edge on port 0.0 (P00/TI000)
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI

/*
------------------------------------------------------------------------------
    Specify Interrupt Vectors
------------------------------------------------------------------------------
*/
#pragma interrupt    INTTM010        MD_INTTM010


/*
------------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------------
*/

/*status list definition*/
#define TRUE            1
#define FALSE           0

/*Timer00, Timer01 pulse width measure*/
unsigned short tm00_pul_old_1;
unsigned short tm00_pos_width;
unsigned short tm00_neg_width;

void TM00_Init(void);
void TM00_Start( void );


/*
------------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------------
*/

void main( void )
{

    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    TM00_Init();                        /* TM00 initialization function */
    TM00_Start();                       /* TM00 start function */
```

```
    EI();                               /* Master interruption permission */

    while(TRUE)                         /* Endless loop */
    {
        NOP();
    }
}


/*
-------------------------------------------------------------------------------
    Functions Group
-------------------------------------------------------------------------------
*/

/*
-------------------------------------------------------------------------
  Abstract:
    Initiate TM00, select founction and input parameter
    count clock selection, INT init
-------------------------------------------------------------------------
*/
void TM00_Init( void )
{
    /*timer00 pulse measurement function*/

    PRM00 = 0x11;                       /* count clock=systemclock/4 */
                                        /* TI000 low->high edge capture */

    PM0.0 = 1;                          /* set P00 is input mode */

    CRC00 = 0x7;                        /* CR010 operating mode : Operates as capture register */
                                        /* CR000 operating mode : Operates as capture register */
                                        /* CR000 capture trigger: Captures on valid edge of TI000 by reverse phase */
    CR000 = 0x0;
    CR010 = 0x0;

    TMIF010 = 0;                        /* No interrupt request signal is generated */
    TMMK010 = 0;                        /* Interrupt servicing enabled */
}


/*
-------------------------------------------------------------------------
  Abstract:
    Start the timer00 counter
-------------------------------------------------------------------------
*/
void TM00_Start( void )
{
    tm00_pul_old_1 = 0;
    tm00_pos_width = 0;
    tm00_neg_width = 0;

    TMC00 = 0x04;                       /* Free-running mode */
}


/*
-------------------------------------------------------------------------
  Abstract:
    TM00 INTTM010 interrupt service routine
-------------------------------------------------------------------------
*/
__interrupt void    MD_INTTM010( void )
{
    /*pulse width calculation*/
    unsigned short tm00_pul_new_0 = (unsigned short)CR000;
    unsigned short tm00_pul_new_1 = (unsigned short)CR010;
```

```
    unsigned char  stat = TMC00;

    OVF00 = 0;                          /* Overflow flag clear */

    if(stat & 0x1){                     /* overflow generated */
        if(tm00_pul_new_0 < tm00_pul_new_1){
            tm00_pos_width = (unsigned short)(((unsigned long)0x10000
                                    + (unsigned long)tm00_pul_new_0) - tm00_pul_old_1);
            tm00_neg_width = tm00_pul_new_1 - tm00_pul_new_0;
        }
        else{
            tm00_pos_width = tm00_pul_new_0 - tm00_pul_old_1;
            tm00_neg_width = (unsigned short)(((unsigned long)0x10000
                                    + (unsigned long)tm00_pul_new_1) - tm00_pul_new_0);
        }
    }
    else{
        tm00_pos_width = tm00_pul_new_0 - tm00_pul_old_1;
        tm00_neg_width = tm00_pul_new_1 - tm00_pul_new_0;
    }

    tm00_pul_old_1 = tm00_pul_new_1;
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title:  Pulse width measurement by TM00
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - timer: 16-bit timer/event counter TM00
;               - timer count clock: fx/4(2.5MHz)
;               - capture register: 16-bit register CR000
;               - capture trigger: rising edge on port 0.0 (P00/TI000)
;               - interrupt: rising edge on port 0.0 (P00/TI000)
;
;*****************************************************************************


;-----------------------------------------------------------------------------
;       Macro Define
;-----------------------------------------------------------------------------
        DSEG

tm00_pul_old_1:         DS      2
tm00_pos_width:         DS      2
tm00_neg_width:         DS      2

;-----------------------------------------------------------------------------
;       Specify Interrupt Vectors
;-----------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to  Start
                DW      start
TM010_VECTOR    CSEG AT 0022h
                DW      MD_INTTM010


;-----------------------------------------------------------------------------
;       Main Program
;-----------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CALL    !TM00_Init              ; TM00 initialization function
        CALL    !TM00_Start             ; TM00 start function

        EI                              ; Master interruption permission
main:                                   ; Endless loop
        NOP
        BR      $main
```

```
;-------------------------------------------------------------------------------
;       Functions Group
;-------------------------------------------------------------------------------


;-------------------------------------------------------------------------
;  Abstract:
;       Initiate TM00, select founction and input parameter
;       count clock selection, INT init
;-------------------------------------------------------------------------
TM00_Init:

        ;Timer00 pulse measurement function
        MOV     PRM00, #11H             ; count clock=systemclock/4
                                        ; TI000 low->high edge capture

        SET1    PM0.0

        MOV     CRC00, #07H
        MOVW    CR000, #0000H
        MOVW    CR010, #0000H
        CLR1    TMIF010                 ; enable INTTM010 interrupt
        CLR1    TMMK010

        RET


;-------------------------------------------------------------------------
;  Abstract:
;       Start the timer00 counter
;-------------------------------------------------------------------------
TM00_Start:
        MOVW    AX, #0000H
        MOVW    !tm00_pul_old_1, AX
        MOVW    !tm00_pos_width, AX
        MOVW    !tm00_neg_width, AX

        MOV     TMC00, #04H             ; Free-running mode

        RET


;-------------------------------------------------------------------------
;  Abstract:
;       TM00 INTTM010 interrupt service routine
;-------------------------------------------------------------------------
MD_INTTM010:

        SEL     RB1

        MOVW    AX, CR010               ; Rising
        MOVW    DE, AX                  ;
        MOVW    AX, CR000               ; Falling
        MOVW    BC, AX                  ;

        ; calculate pulse active interval: Falling(CR000) - Rising(Old)
        XCH     A, X
        SUB     A, !tm00_pul_old_1+0
        XCH     A, X
        SUBC    A, !tm00_pul_old_1+1
        MOVW    !tm00_pos_width, AX

        MOVW    AX, DE
        MOVW    !tm00_pul_old_1+0, AX

        ; calculate pulse inactive interval: Rising(CR010) - Falling(CR000)
        XCH     A, X
        SUB     A, C
```

```
        XCH     A, X
        SUBC    A, B
        MOVW    !tm00_neg_width, AX

        RETI

END
```

3.3.5 方形波出力

■ ファイル名

Square

- TM0_SquareWave.asm（アセンブリ言語ソース）
- TM0_SquareWave.c（C言語ソース）

■ 説明

16ビット・タイマ／イベント・カウンタ00，01[注]には，次のような機能があります。

（1）インターバル・タイマ

あらかじめ設定した任意の時間間隔で割込み要求を発生します。

（2）PPG出力

周波数と出力パルス幅を任意に設定できる矩形波を出力できます。

（3）パルス幅測定

外部から入力される信号のパルス幅を測定できます。

（4）外部イベント・カウンタ

外部から入力される信号のパルス数を測定できます。

（5）方形波出力

任意の周波数の方形波出力が可能です。

（6）ワンショット・パルス出力

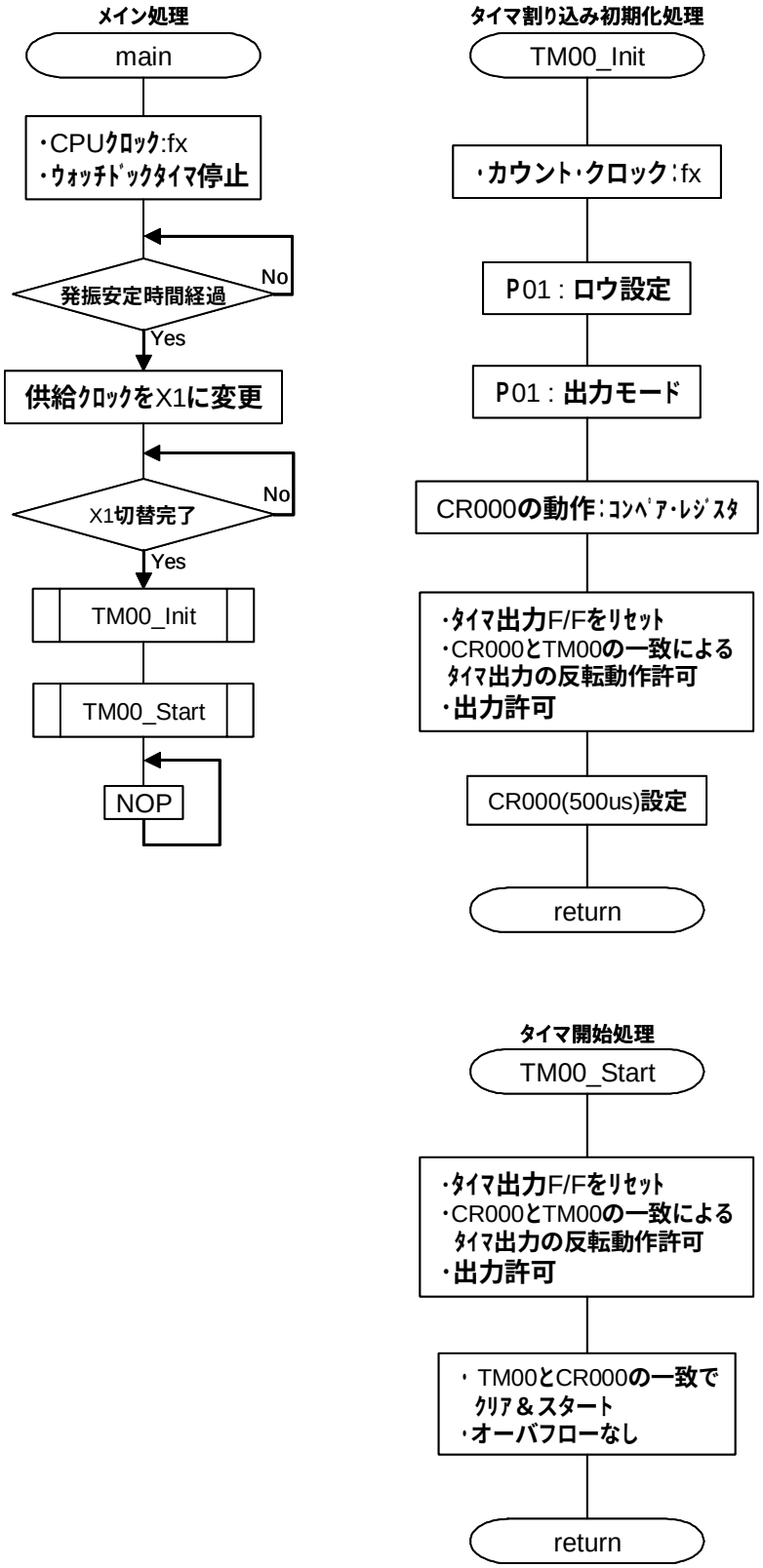
出力パルス幅を任意に設定できるワンショット・パルスを出力できます。

注 μPD780146，780148，78F0148のみ。

このプログラムは、方形波出力モードを使用し 500ms 毎に P01/TO00 からの出力を反転します。そしてメイン処理は、出力設定後無限ループにします。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz) を選択します。

初期設定としてタイマのカウントクロックに 10MHz を選択します。そして 16 ビット・タイマ・キャプチャ／コンペアレジスタ(CR000) は 4999 に設定されています。タイマの実行開始後、タイマカウンタ(TM00) が CR000 の値(500μs) と一致する毎に、P01/TO00 を反転させます。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
TM00_Init
TM00_Start
NOP
タイマ割り込み初期化処理
TM00_Init
・ｶｳﾝﾄ・ｸﾛｯｸ:fx
P01：ロウ設定
P01：出力モード
CR000の動作:ｺﾝﾍﾟｱ・ﾚｼﾞｽﾀ
・ﾀｲﾏ出力F/Fをﾘｾｯﾄ
・CR000とTM00の一致による
ﾀｲﾏ出力の反転動作許可
・出力許可
CR000(500us)設定
return
タイマ開始処理
TM00_Start
・ﾀｲﾏ出力F/Fをﾘｾｯﾄ
・CR000とTM00の一致による
ﾀｲﾏ出力の反転動作許可
・出力許可
・TM00とCR000の一致で
ｸﾘｱ＆スタート
・オーバフローなし
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: Rectangle wave output by TM00
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - timer: 16-bit timer/event counter TM00
*               - timer count clock: fx (10 MHz)
*               - compare register value (CR000): 0x1387
*               - output port: port 0.1 (P01/TO00)
*               - square wave output frequency: 2kHz (500us period)
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

#define TM_TM00_SQUAREWIDTH     0x1387

/*status list definition*/
#define TRUE            1
#define FALSE           0

void TM00_Init(void);
void TM00_Start();


/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/

void main( void )
{

    PCC = 0x00;                         /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    TM00_Init();                        /* TM00 initialization function */
    TM00_Start();                       /* TM00 start function */

    while(TRUE)                         /* Endless loop */
    {
        NOP();
    }
}


/*
------------------------------------------------------------------------
    Functions Group
```

```
---------------------------------------------------------------------------
*/

/*
---------------------------------------------------------------------------
  Abstract:
    Initiate TM00, select founction and input parameter
    count clock selection, INT init
---------------------------------------------------------------------------
*/
void TM00_Init( void )
{
    PRM00 = 0x00;                       /* internal count clock*/
    /*timer00 squarewave output*/

    P0.1 = 0;                           /* P01=0 */
    PM0.1 = 0;                          /* set P01 is output mode */

    CRC00 = 0x00;                       /* CR010 operating mode : Operates as compare register */
                                        /* CR000 operating mode : Operates as compare register */
                                        /* CR000 capture trigger: Captures on valid edge of TI010 */
    TOC00 = 0x7;                        /* init low */
    CR000 = TM_TM00_SQUAREWIDTH;
}


/*
---------------------------------------------------------------------------
  Abstract:
    Start the timer00 counter
---------------------------------------------------------------------------
*/
void TM00_Start()
{
    TOC00 = 0x7;                        /* init low */
    TMC00 = 0x0c;                       /* Clear & start occurs on match between TM00 and CR000 */
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: Rectangle wave output by TM00
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - timer: 16-bit timer/event counter TM00
;               - timer count clock: fx (10 MHz)
;               - compare register value (CR000): 0x1387
;               - output port: port 0.1 (P01/TO00)
;               - square wave output frequency: 2kHz (500us period)
;
;*****************************************************************************


;---------------------------------------------------------------------------
;       Macro Define
;---------------------------------------------------------------------------

TM_TM00_SQUAREWIDTH     EQU     1387h


;---------------------------------------------------------------------------
;       Specify Interrupt Vectors
;---------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to  Start
                DW      start


;---------------------------------------------------------------------------
;       Main Program
;---------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CALL    !TM00_Init              ; TM00 initialization function
        CALL    !TM00_Start             ; TM00 start function

main:                                   ; Endless loop
        NOP
        BR      $main


;---------------------------------------------------------------------------
;       Functions Group
;---------------------------------------------------------------------------
```

```
;-------------------------------------------------------------------------
;  Abstract:
;       Initiate TM00, select founction and input parameter
;       count clock selection, INT init
;-------------------------------------------------------------------------
TM00_Init:
        MOV     PRM00, #00H             ; internal count clock

        ;Timer00 square wave output function
        CLR1    P0.1
        CLR1    PM0.1

        MOV     CRC00, #00H
        MOV     TOC00, #07H             ; init low
        MOVW    CR000, #TM_TM00_SQUAREWIDTH

        RET


;-------------------------------------------------------------------------
;  Abstract:
;       Start the timer00 counter
;-------------------------------------------------------------------------
TM00_Start:
        MOV     TOC00, #07H             ; init low
        MOV     TMC00, #0CH             ; Clear & start occurs on match between TM00 and CR000

        RET

END
```

3.3.6 ワンショット･パルス出力

■ ファイル名

OneShot

- TM0_Oneshot.asm（アセンブリ言語ソース）
- TM0_Oneshot.c（C言語ソース）

■ 説明

16ビット・タイマ／イベント・カウンタ00，01[注]には，次のような機能があります。

（1）インターバル・タイマ

あらかじめ設定した任意の時間間隔で割込み要求を発生します。

（2）PPG出力

周波数と出力パルス幅を任意に設定できる矩形波を出力できます。

（3）パルス幅測定

外部から入力される信号のパルス幅を測定できます。

（4）外部イベント・カウンタ

外部から入力される信号のパルス数を測定できます。

（5）方形波出力

任意の周波数の方形波出力が可能です。

（6）ワンショット・パルス出力

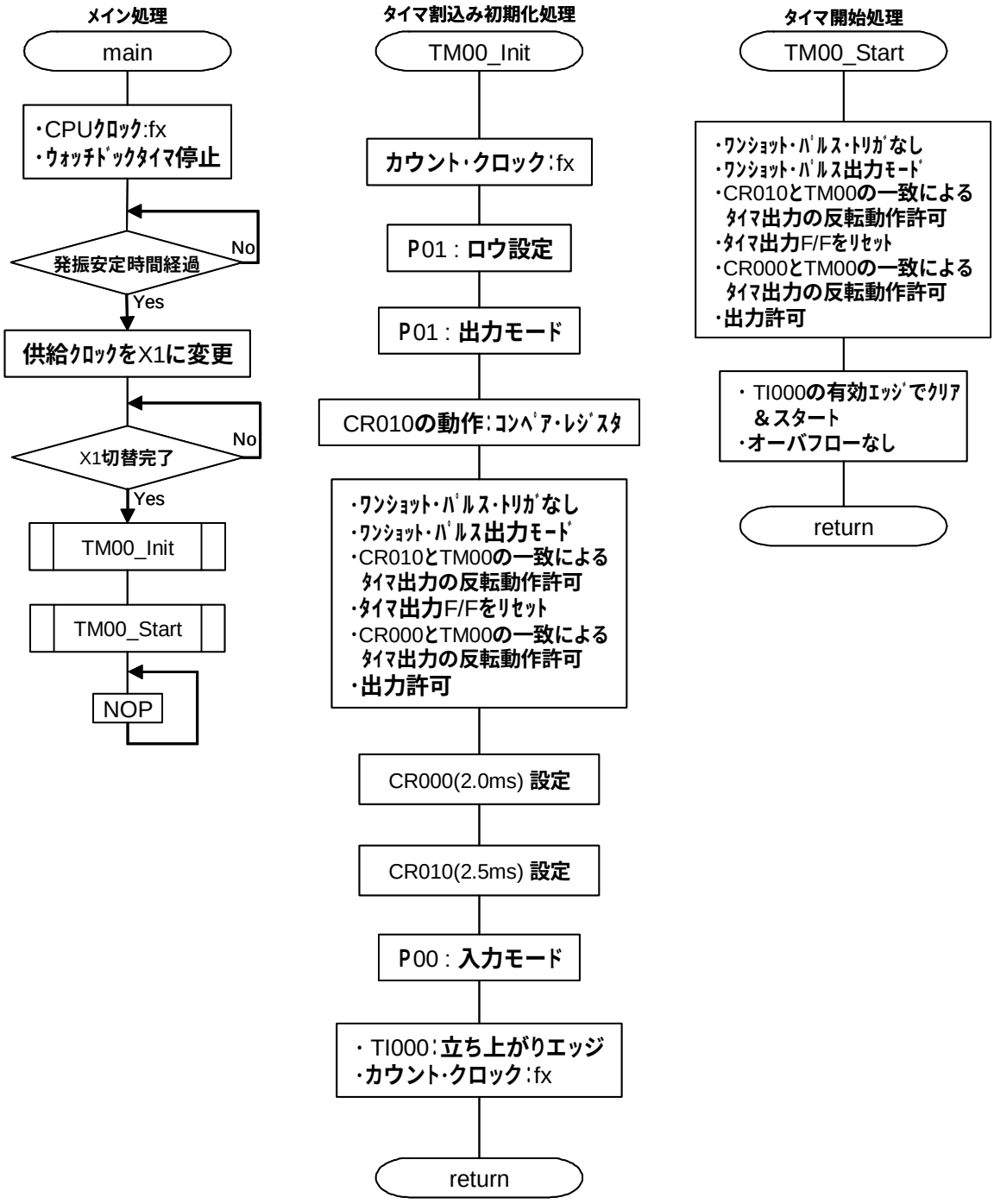
出力パルス幅を任意に設定できるワンショット・パルスを出力できます。

注 μPD780146，780148，78F0148のみ。

このプログラムは、P00/TI00 の立ち上がりエッジ検出から、2ms 経過後に P01/TO00 より 500μs 幅のワンショット･パルスを出力します。そしてメイン処理は、出力設定後 無限ループにします。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

初期設定としてタイマのカウントクロックに 10MHz を選択します。そして 16 ビット・タイマ・キャプチャ／コンペアレジスタ 000(CR000) はパルス開始までの時間を決定し、19999 までのカウントで 2.0ms の期間を作成しています。16 ビット・タイマ・キャプチャ／コンペアレジスタ 010（CR010）はパルスの終了を決定し、24999 までのカウントで 2.5ms の期間を作成しています。タイマは P00/TIO00 入力の立ち上がり検出にて、16 ビット・タイマ・カウンタ 00（TM00）のカウントをクリア&スタートします。そして TM00 と CR000 が一致すると P0.1 はアクティブになります。次に TM00 と CR010 が一致することで P01 はインアクティブになります。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
TM00_Init
TM00_Start
NOP
タイマ割込み初期化処理
TM00_Init
カウント・クロック:fx
P01 : ロウ設定
P01 : 出力モード
CR010の動作:ｺﾝﾍﾟｱ・ﾚｼﾞｽﾀ
・ﾜﾝｼｮｯﾄ・ﾊﾟﾙｽ・ﾄﾘｶﾞなし
・ﾜﾝｼｮｯﾄ・ﾊﾟﾙｽ出力ﾓｰﾄﾞ
・CR010とTM00の一致による
ﾀｲﾏ出力の反転動作許可
・ﾀｲﾏ出力F/Fをﾘｾｯﾄ
・CR000とTM00の一致による
ﾀｲﾏ出力の反転動作許可
・出力許可
CR000(2.0ms) 設定
CR010(2.5ms) 設定
P00 : 入力モード
・TI000:立ち上がりエッジ
・カウント・クロック:fx
return
タイマ開始処理
TM00_Start
・ﾜﾝｼｮｯﾄ・ﾊﾟﾙｽ・ﾄﾘｶﾞなし
・ﾜﾝｼｮｯﾄ・ﾊﾟﾙｽ出力ﾓｰﾄﾞ
・CR010とTM00の一致による
ﾀｲﾏ出力の反転動作許可
・ﾀｲﾏ出力F/Fをﾘｾｯﾄ
・CR000とTM00の一致による
ﾀｲﾏ出力の反転動作許可
・出力許可
・TI000の有効ｴｯｼﾞでｸﾘｱ
＆スタート
・オーバフローなし
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: One shot pulse output by TM00
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - timer: 16-bit timer/event counter TM00
*               - timer count clock: fx (10 MHz)
*               - compare register value (CR010): 0x61a7
*               - compare register value (CR000): 0x4e1f
*               - pulse delay time: 2.0 ms
*               - pulse width: 0.5 ms
*               - output port: port 0.1
*               - input  port: port 0.0 (positive edge)
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI


/*
------------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------------
*/

#define TM_TM00_ONESHOTCYCLE    0x61a7
#define TM_TM00_ONEPULSEDELAY   0x4e1f


/*status list definition*/
#define TRUE            1
#define FALSE           0

void TM00_Init(void);
void TM00_Start();


/*
------------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------------
*/

void main( void )
{

    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    TM00_Init();                        /* TM00 initialization function */
    TM00_Start();                       /* TM00 start function */


    while(TRUE)                         /* Endless loop */
    {
        NOP();
    }
```

```
}


/*
---------------------------------------------------------------------------
    Functions Group
---------------------------------------------------------------------------
*/

/*
---------------------------------------------------------------------------
  Abstract:
    Initiate TM00, select founction and input parameter
    count clock selection, INT init
---------------------------------------------------------------------------
*/
void TM00_Init( void )
{
    PRM00 = 0x00;                       /* internal count clock */
    /*timer00 one shot pulse output*/


    P0.1 = 0;                           /* P01=0 */
    PM0.1 = 0;                          /* set P01 is output mode */
    CRC00 = 0x00;                       /* CR010 operating mode : Operates as compare register */
                                        /* CR000 operating mode : Operates as compare register */
                                        /* CR000 capture trigger: Captures on valid edge of TI010 */
    TOC00 = 0x37;                       /* Timer output F/F reset (0) */
                                        /* Enables inversion operation/output */
                                        /* One-shot pulse output mode */

    CR000 = TM_TM00_ONESHOTCYCLE;
    CR010 = TM_TM00_ONEPULSEDELAY;

    PM0.0 = 1;                          /* set P00 is input mode */
    PRM00 = 0x10;                       /* timer00 input 0 rising edge */
}

/*
---------------------------------------------------------------------------
  Abstract:
    Start the timer00 counter
---------------------------------------------------------------------------
*/
void TM00_Start( void )
{
    TOC00 = 0x37;                       /* Timer output F/F reset (0) */
                                        /* Enables inversion operation/output */
                                        /* One-shot pulse output mode */
    TMC00 = 0x08;                       /* Clear & start occurs on TI001 valid edge */
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: One shot pulse output by TM00
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - timer: 16-bit timer/event counter TM00
;               - timer count clock: fx (10 MHz)
;               - compare register value (CR010): 0x61a7
;               - compare register value (CR000): 0x4e1f
;               - pulse delay time: 2.0 ms
;               - pulse width: 0.5 ms
;               - output port: port 0.1
;               - input  port: port 0.0 (positive edge)
;
;*****************************************************************************



;---------------------------------------------------------------------------
;       Macro Define
;---------------------------------------------------------------------------


TM_TM00_ONESHOTCYCLE    EQU     61a7h
TM_TM00_ONEPULSEDELAY   EQU     4e1fh


;---------------------------------------------------------------------------
;       Specify Interrupt Vectors
;---------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to  Start
                DW      start


;---------------------------------------------------------------------------
;       Main Program
;---------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CALL    !TM00_Init              ; TM00 initialization function
        CALL    !TM00_Start             ; TM00 start function

main:                                   ; Endless loop
        NOP
        BR      $main
```

```
;-------------------------------------------------------------------------------
;       Functions Group
;-------------------------------------------------------------------------------


;-------------------------------------------------------------------------
;  Abstract:
;       Initiate TM00, select founction and input parameter
;       count clock selection, INT init
;-------------------------------------------------------------------------
TM00_Init:
        MOV     PRM00, #00H             ; internal count clock

        ;Timer00 one shot pulse output
        CLR1    P0.1
        CLR1    PM0.1

        MOV     CRC00, #00H
        MOV     TOC00, #37H             ; One-shot pulse output mode

        MOVW    CR000, #TM_TM00_ONESHOTCYCLE
        MOVW    CR010, #TM_TM00_ONEPULSEDELAY

        SET1    PM0.0
        MOV     PRM00, #10H             ; Timer00 input 0 rising edge

        RET


;-------------------------------------------------------------------------
;  Abstract:
;       Start the timer00 counter
;-------------------------------------------------------------------------
TM00_Start:
        MOV     TOC00, #37H
        MOV     TMC00, #08H             ; Clear & start occurs on TI001 valid edge

        RET

END
```

３．４ 8ビット･タイマ／イベント･カウンタ５０

3.4.1 インターバル･タイマ

■ ファイル名

Interval

- TM50_Interval.asm（アセンブリ言語ソース）
- TM50_Interval.c（Ｃ言語ソース）

■ 説明

８ビット・タイマ／イベント・カウンタ50，51は，次のような機能として使用できます。

（1）インターバル・タイマ

あらかじめ設定した任意の時間間隔で割込み要求を発生します。

（2）外部イベント・カウンタ

外部から入力される信号のパルス数を測定できます。

（3）方形波出力

任意の周波数の方形波出力が可能です。

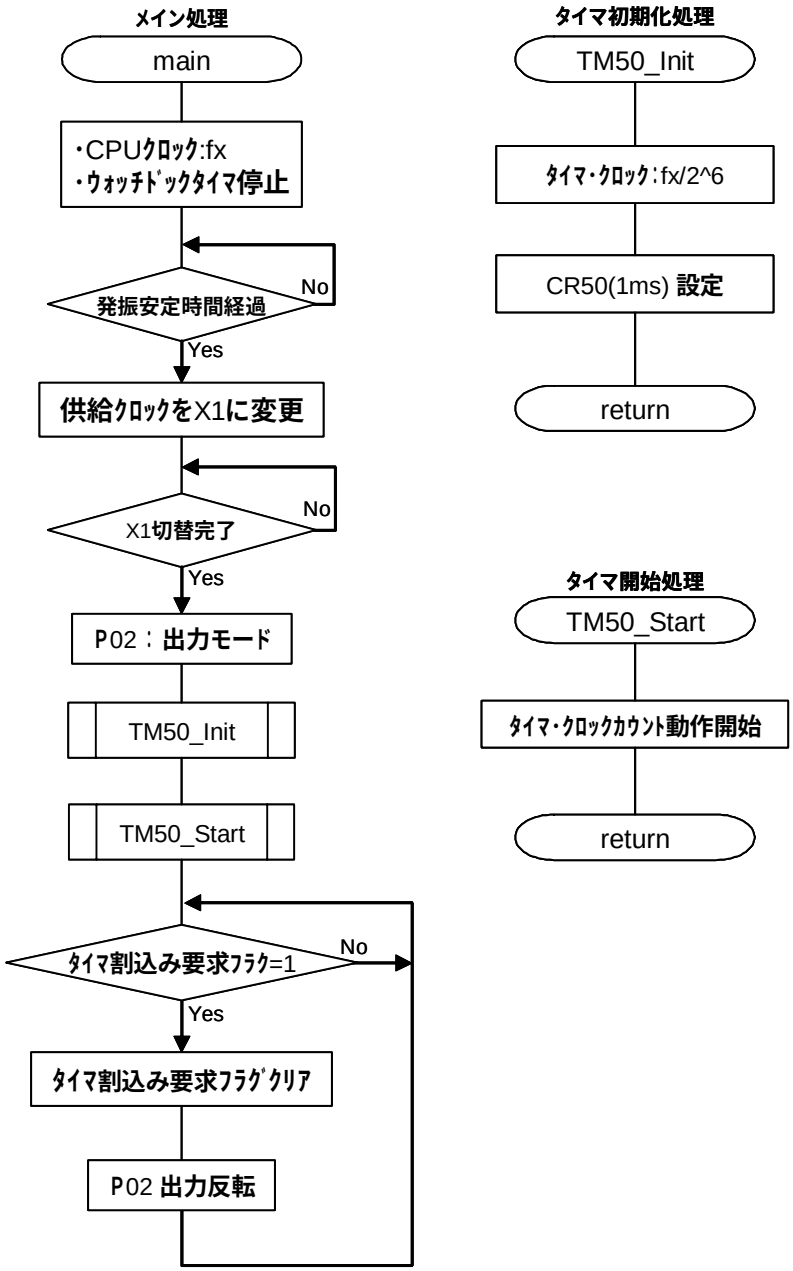
（4）PWM出力

出力パルス幅を任意に設定できる矩形波を出力できます。

このプログラムは、1ms 毎にインターバルモードにより INTTM50 を発生させます。そしてメイン処理にてフラグセンスにより１ms 経過を判断し、P02 を反転出力します。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

初期設定として、タイマのカウントクロックは 156.25kHz が選択され、かつコンペアレジスタ（CR50）は 155 に設定されます。タイマの動作開始後、タイマ・カウンタがコンペアレジスタの値（1ms）と一致した時、タイマ割込みの要求フラグ（TMIF50）がセットされます。メインループは、このタイマ割込みのリクエストフラグがセットされたことを検知すると、P02 のレベルを反転出力させます。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
P02：出力モード
TM50_Init
TM50_Start
ﾀｲﾏ割込み要求ﾌﾗｸ=1
No
Yes
ﾀｲﾏ割込み要求ﾌﾗｸﾞｸﾘｱ
P02 出力反転
タイマ初期化処理
TM50_Init
ﾀｲﾏ・ｸﾛｯｸ：fx/2^6
CR50(1ms) 設定
return
タイマ開始処理
TM50_Start
ﾀｲﾏ・ｸﾛｯｸｶｳﾝﾄ動作開始
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: Interval timer by TM50
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - timer: 8-bit timer/event counter TM50
*               - timer count clock: fx/64(156.25KHz)
*               - compare register value (CR50): 0x9b
*               - interval time: 1.0 ms
*               - interrupt handling: polling of interrupt request flag (TMIF50)
*               - output port: port 0.2 (toggles every 1.0 ms)
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

#define TM_TM50_CLOCK           0x5
#define TM_TM50_INTERVALVALUE   0x9b

/*status list definition*/
#define TRUE            1
#define FALSE           0

void TM50_Init(void);
void TM50_Start();


/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/

void main( void )
{

    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    PM0.2 = 0;                          /* set P02 is output mode */

    TM50_Init();                        /* TM50 initialization function */
    TM50_Start();                       /* TM50 start function */


    while(TRUE)                         /* main loop */
    {
        /* Waiting for timer request flag */
        while( TMIF50 == 0 )
        {
```

```
            NOP();
        }
        TMIF50 = 0;                     /* No interrupt request signal is generated */
        P0.2 ^= 1;                      /* P0.2 output reversal */
    }
}


/*
-------------------------------------------------------------------------------
    Functions Group
-------------------------------------------------------------------------------
*/

/*
-------------------------------------------------------------------------------
  Abstract:
    Initiate TM50, select founction and input parameter
    count clock selection, INT init
-------------------------------------------------------------------------------
*/
void TM50_Init( void )
{
    TCL50 = TM_TM50_CLOCK;              /* internal count clock */

    /*timer50 interval*/

    CR50 = TM_TM50_INTERVALVALUE;
}


/*
-------------------------------------------------------------------------------
  Abstract:
    Start the timer50 counter
-------------------------------------------------------------------------------
*/
void TM50_Start()
{
    /*timer50 interval*/
    TCE50 = 1;                          /* Count operation start */
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: Interval timer by TM50
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - timer: 8-bit timer/event counter TM50
;               - timer count clock: fx/64(156.25KHz)
;               - compare register value (CR50): 0x9b
;               - interval time: 1.0 ms
;               - interrupt handling: polling of interrupt request flag (TMIF50)
;               - output port: port 0.2 (toggles every 1.0 ms)
;
;*****************************************************************************


;---------------------------------------------------------------------------
;       Macro Define
;---------------------------------------------------------------------------

TM_TM50_CLOCK           EQU     05H
TM_TM50_INTERVALVALUE   EQU     9BH


;---------------------------------------------------------------------------
;       Specify Interrupt Vectors
;---------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to Start
                DW      start


;---------------------------------------------------------------------------
;       Main Program
;---------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CLR1    PM0.2

        CALL    !TM50_Init              ; TM50 initialization function
        CALL    !TM50_Start             ; TM50 start function

main:
        BTCLR   TMIF50, $main00
        NOP
        BR      $main
```

```
main00:
        MOV1    CY, P0.2
        NOT1    CY
        MOV1    P0.2, CY                ; P0.2 output reversal

        BR      $main


;---------------------------------------------------------------------------
;       Functions Group
;---------------------------------------------------------------------------


;---------------------------------------------------------------------------
;  Abstract:
;       Initiate TM50, select founction and input parameter
;       count clock selection, INT init
;---------------------------------------------------------------------------
TM50_Init:

        MOV     TCL50, #TM_TM50_CLOCK   ; internal count clock

        ;Timer50 interval timer

        MOV     CR50, #TM_TM50_INTERVALVALUE

        RET


;---------------------------------------------------------------------------
;  Abstract:
;       Start the timer50 counter
;---------------------------------------------------------------------------
TM50_Start:
        SET1    TCE50                   ;  Count operation start

        RET

END
```

3.4.2 方形波出力

■ ファイル名

Square

- TM50_SquareWave.asm（アセンブリ言語ソース）
- TM50_SquareWave.c（C言語ソース）

■ 説明

８ビット・タイマ／イベント・カウンタ50，51は，次のような機能として使用できます。

（1）インターバル・タイマ

あらかじめ設定した任意の時間間隔で割込み要求を発生します。

（2）外部イベント・カウンタ

外部から入力される信号のパルス数を測定できます。

（3）方形波出力

任意の周波数の方形波出力が可能です。

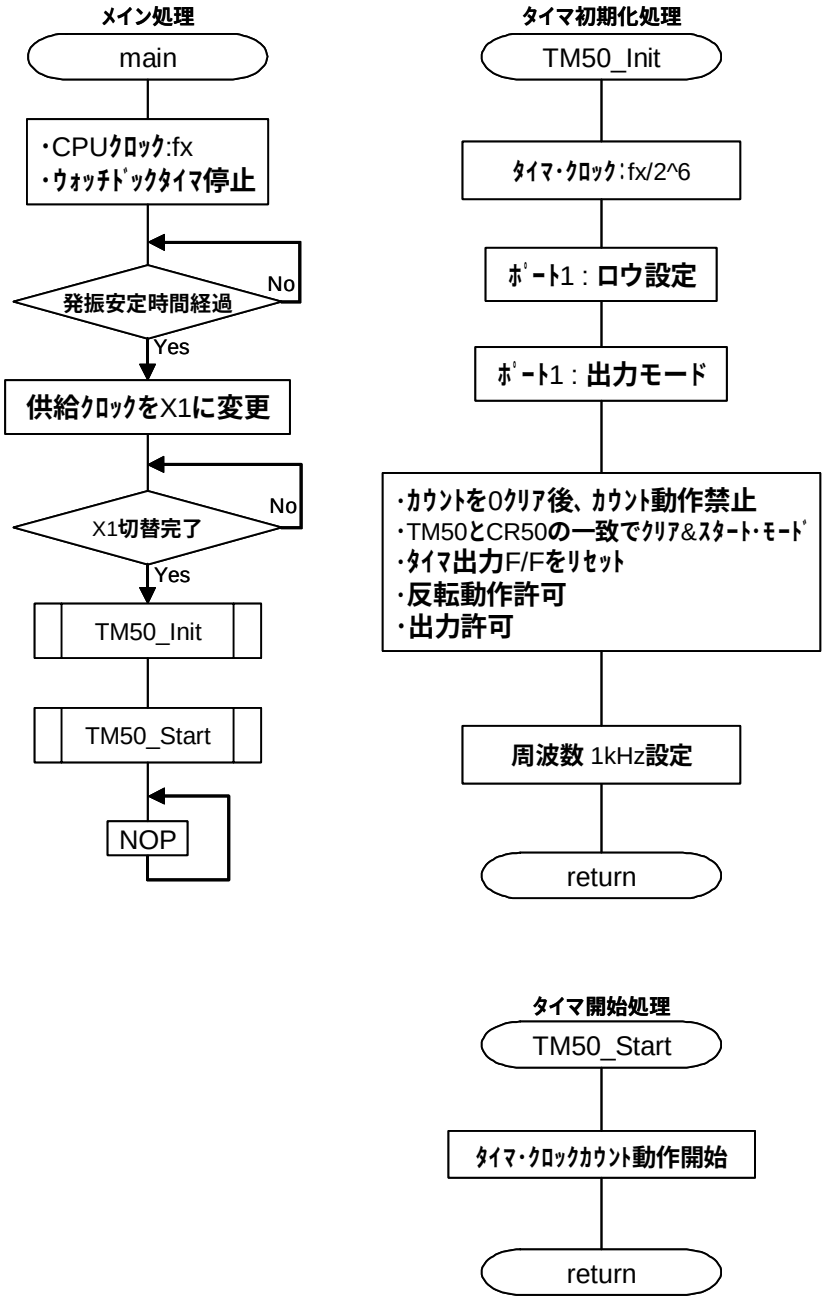
（4）PWM出力

出力パルス幅を任意に設定できる矩形波を出力できます。

このプログラムは、方形波出力モードを使用し 500ms 毎に P17/TO50 からの出力を反転します。そしてメイン処理は、出力設定後 無限ループにします。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

初期設定として、タイマのカウントクロックは 156.25kHz を選択します。そして、コンペアレジスタ（CR50）は 77 に設定します。タイマの実行開始後、タイマカウンタがコンペアレジスタに設定された時間（500μs）と一致した時、P17/TO50 が反転します。このオペレーションはプログラムからの介在なしに連続的に行われます。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
TM50_Init
TM50_Start
NOP
タイマ初期化処理
TM50_Init
ﾀｲﾏ・ｸﾛｯｸ:fx/2^6
ﾎﾟｰﾄ1：ロウ設定
ﾎﾟｰﾄ1：出力モード
・ｶｳﾝﾄを0ｸﾘｱ後、ｶｳﾝﾄ動作禁止
・TM50とCR50の一致でｸﾘｱ&ｽﾀｰﾄ・ﾓｰﾄﾞ
・ﾀｲﾏ出力F/Fをﾘｾｯﾄ
・反転動作許可
・出力許可
周波数 1kHz設定
return
タイマ開始処理
TM50_Start
ﾀｲﾏ・ｸﾛｯｸｶｳﾝﾄ動作開始
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: Rectangle wave output by TM50
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - timer: 8-bit timer/event counter TM50
*               - timer count clock: fx/64(156.25KHz)
*               - compare register value (CR50): 0x4d
*               - output port: port 1.7 (P17/TO50)
*               - square wave output frequency: 1kHz (1ms period)
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI


/*
------------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------------
*/

#define TM_TM50_CLOCK       0x5
#define TM_TM50_SQUAREWIDTH 0x4d

/*status list definition*/
#define TRUE            1
#define FALSE           0

void TM50_Init(void);
void TM50_Start();


/*
------------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------------
*/

void main( void )
{

    PCC = 0x00;                         /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    TM50_Init();                        /* TM50 initialization function */
    TM50_Start();                       /* TM50 start function */


    while(TRUE)                         /* Endless loop */
    {
        NOP();
    }
}


/*
```

```
--------------------------------------------------------------------------------
    Functions Group
--------------------------------------------------------------------------------
*/

/*
--------------------------------------------------------------------------------
  Abstract:
    Initiate TM50, select founction and input parameter
    count clock selection, INT init
--------------------------------------------------------------------------------
*/
void TM50_Init( void )
{
    TCL50=TM_TM50_CLOCK;                /* internal count clock */

    /* timer50 square wave output */

    P1 = 0x00;                          /* P1=0x00 */
    PM1 = 0x00;                         /* TO50 output */
    TMC50 = 0x07;                       /* F/F reset */
                                        /* F/F turn over enable */

    CR50 = TM_TM50_SQUAREWIDTH;
}


/*
--------------------------------------------------------------------------------
  Abstract:
    Start the timer50 counter
--------------------------------------------------------------------------------
*/
void TM50_Start()
{
    /* timer50 square wave output */
    TCE50 = 1;                          /* Count operation start */
}
```

■ アセンブリ言語ソース

```
;
;*******************************************************************************
;
;  Title: Rectangle wave output by TM50
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - timer: 8-bit timer/event counter TM50
;               - timer count clock: fx/64(156.25KHz)
;               - compare register value (CR50): 0x4d
;               - output port: port 1.7 (P17/TO50)
;               - square wave output frequency: 2kHz (500us period)
;
;*******************************************************************************


;-------------------------------------------------------------------------------
;       Macro Define
;-------------------------------------------------------------------------------

TM_TM50_CLOCK           EQU     05H
TM_TM50_SQUAREWIDTH     EQU     4dH


;-------------------------------------------------------------------------------
;       Specify Interrupt Vectors
;-------------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to Start
                DW      start


;-------------------------------------------------------------------------------
;       Main Program
;-------------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; supply clock: X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CALL    !TM50_Init              ; TM50 initialization function
        CALL    !TM50_Start             ; TM50 start function

main:                                   ; Endless loop
        NOP
        BR      $main


;-------------------------------------------------------------------------------
;       Functions Group
```

```
;-------------------------------------------------------------------------------


;-------------------------------------------------------------------------
;  Abstract:
;       Initiate TM50, select founction and input parameter
;       count clock selection, INT init
;-------------------------------------------------------------------------
TM50_Init:

        MOV     TCL50, #TM_TM50_CLOCK   ; internal count clock

        ;Timer50 square wave output function
        MOV     P1, #00H
        MOV     PM1, #00H
        MOV     TMC50, #07H             ; F/F turn over enable

        MOV     CR50, #TM_TM50_SQUAREWIDTH

        RET


;-------------------------------------------------------------------------
;  Abstract:
;       Start the timer50 counter
;-------------------------------------------------------------------------
TM50_Start:
        SET1    TCE50                   ; Count operation start

        RET

END
```

3.4.3 PWM出力

■ ファイル名

PWM

- TM50_PWM.asm（アセンブリ言語ソース）
- TM50_PWM.c（C言語ソース）

■ 説明

８ビット・タイマ／イベント・カウンタ50，51は，次のような機能として使用できます。

（1）インターバル・タイマ

あらかじめ設定した任意の時間間隔で割込み要求を発生します。

（2）外部イベント・カウンタ

外部から入力される信号のパルス数を測定できます。

（3）方形波出力

任意の周波数の方形波出力が可能です。

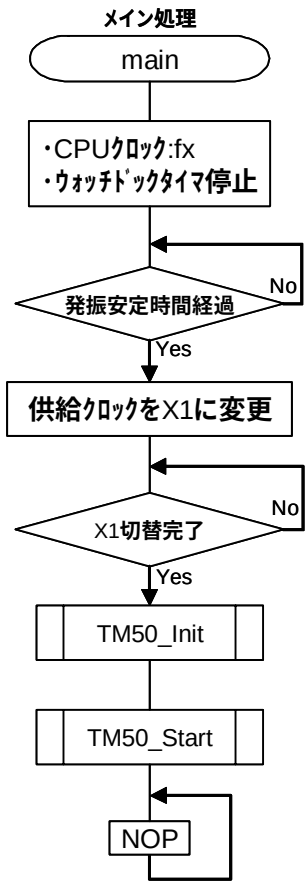
（4）PWM出力

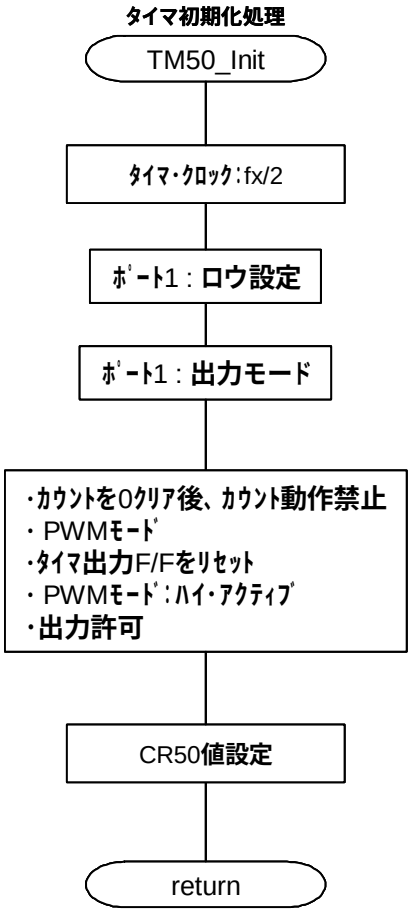
出力パルス幅を任意に設定できる矩形波を出力できます。

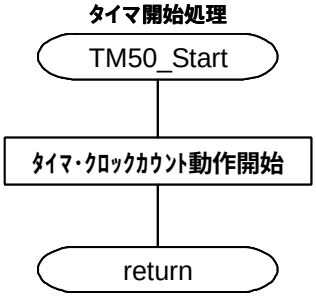
このプログラムは、12.8us の High 期間[ﾃﾞｭｰﾃｨ比 25%]の波形を 51.2us の周期で繰り返し出力します。そしてメイン処理は、出力設定後 無限ループにします。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

初期設定としてタイマのカウントクロックは 5MHz が選択されており 51.2μs の周期になります。そしてコンペアレジスタ（CR50）はﾃﾞｭｰﾃｨ比を決定し、64 までのカウントでﾃﾞｭｰﾃｨ比 25%のパルスを作成しています。タイマが 0 からのスタート後、CR50 と値が一致したとするとインアクティブ・レベルを出力します。その後もカウントは継続して進み、オーバーフローが発生した時点でアクティブレベルを出力します。このオペレーションはプログラムからの介在なしで、連続的に行われます。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
TM50_Init
TM50_Start
NOP


ﾀｲﾏ初期化処理
TM50_Init
ﾀｲﾏ・ｸﾛｯｸ:fx/2
ﾎﾟｰﾄ1：ロウ設定
ﾎﾟｰﾄ1：出力モード
・ｶｳﾝﾄを0ｸﾘｱ後、ｶｳﾝﾄ動作禁止
・PWMﾓｰﾄﾞ
・ﾀｲﾏ出力F/Fをﾘｾｯﾄ
・PWMﾓｰﾄﾞ:ﾊｲ・ｱｸﾃｨﾌﾞ
・出力許可
CR50値設定
return


ﾀｲﾏ開始処理
TM50_Start
ﾀｲﾏ・ｸﾛｯｸｶｳﾝﾄ動作開始
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: PWM output by TM50
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - timer: 8-bit timer/event counter TM50
*               - timer count clock: fx/2(5MHz)
*               - compare register CR50 value: 0x40
*               - PWM period: 51.2us
*               - PWM duty-cycle: 25% (12.8us)
*               - output port: port 1.7 (P17/TO50)
*
************************************************************************
*/
#pragma sfr
#pragma NOP


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

#define TM_TM50_CLOCK   0x3
#define TM_TM50_PWMACTIVEVALUE  0x40

/*status list definition*/
#define TRUE            1
#define FALSE           0

void TM50_Init(void);
void TM50_Start();


/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/

void main( void )
{
    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    TM50_Init();                        /* TM50 initialization function */
    TM50_Start();                       /* TM50 start function */

    while(TRUE)                         /* Endless loop */
    {
        NOP();
    }
}


/*
------------------------------------------------------------------------
    Functions Group
```

```
------------------------------------------------------------------------------
*/

/*
------------------------------------------------------------------------------
  Abstract:
    Initiate TM50, select founction and input parameter
    count clock selection, INT init
------------------------------------------------------------------------------
*/
void TM50_Init( void )
{
    TCL50=TM_TM50_CLOCK;                /* internal count clock */

    /*timer50 PWM output*/

    P1 = 0x00;                          /* P1=0 */
    PM1 = 0x00;                         /* set P1 is output mode */

    TMC50 = 0x45;                       /* PWM mode */
                                        /* F/F reset */
                                        /* high active */
                                        /* timer output enable */

    CR50 = TM_TM50_PWMACTIVEVALUE;
}


/*
------------------------------------------------------------------------------
  Abstract:
    Start the timer50 counter
------------------------------------------------------------------------------
*/
void TM50_Start( void )
{
    TCE50 = 1;                          /* Count operation start */
}
```

■ アセンブリ言語ソース

```
;
;*******************************************************************************
;
;  Title: Rectangle wave output by TM50
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - timer: 8-bit timer/event counter TM50
;               - timer count clock: fx/2(5MHz)
;               - compare register value (CR50): 0x4d
;               - output port: port 1.7 (P17/TO50)
;               - square wave output frequency: 2kHz (500us period)
;
;*******************************************************************************


;-------------------------------------------------------------------------------
;       Macro Define
;-------------------------------------------------------------------------------

TM_TM50_CLOCK           EQU     05H
TM_TM50_SQUAREWIDTH     EQU     4dH


;-------------------------------------------------------------------------------
;       Specify Interrupt Vectors
;-------------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to  Start
                DW      start


;-------------------------------------------------------------------------------
;       Main Program
;-------------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; supply clock: X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CALL    !TM50_Init              ; TM50 initialization function
        CALL    !TM50_Start             ; TM50 start function

main:                                   ; Endless loop
        NOP
        BR      $main


;-------------------------------------------------------------------------------
;       Functions Group
```

```
;-------------------------------------------------------------------------------


;-------------------------------------------------------------------------
;  Abstract:
;       Initiate TM50, select founction and input parameter
;       count clock selection, INT init
;-------------------------------------------------------------------------
TM50_Init:

        MOV     TCL50, #TM_TM50_CLOCK   ; internal count clock

        ;Timer50 square wave output function
        MOV     P1, #00H
        MOV     PM1, #00H
        MOV     TMC50, #07H             ; F/F turn over enable

        MOV     CR50, #TM_TM50_SQUAREWIDTH

        RET


;-------------------------------------------------------------------------
;  Abstract:
;       Start the timer50 counter
;-------------------------------------------------------------------------
TM50_Start:
        SET1    TCE50                   ; Count operation start

        RET

END
```

３．５ ８ビット･タイマＨ０

3.5.1 PWM出力

■ ファイル名

PWM

- TMH0_PWM.asm（アセンブリ言語ソース）
- TMH0_PWM.c（Ｃ言語ソース）

■ 説明

8ビット・タイマH0，H1には，次のような機能があります。

（1）8ビット精度のインターバル・タイマ

あらかじめ設定した任意の時間間隔で割込み要求を発生します。

（2）8ビット精度のPWMパルス・ジェネレータ・モード

任意のデューティ及び周期が可能なパルスを出力できます。

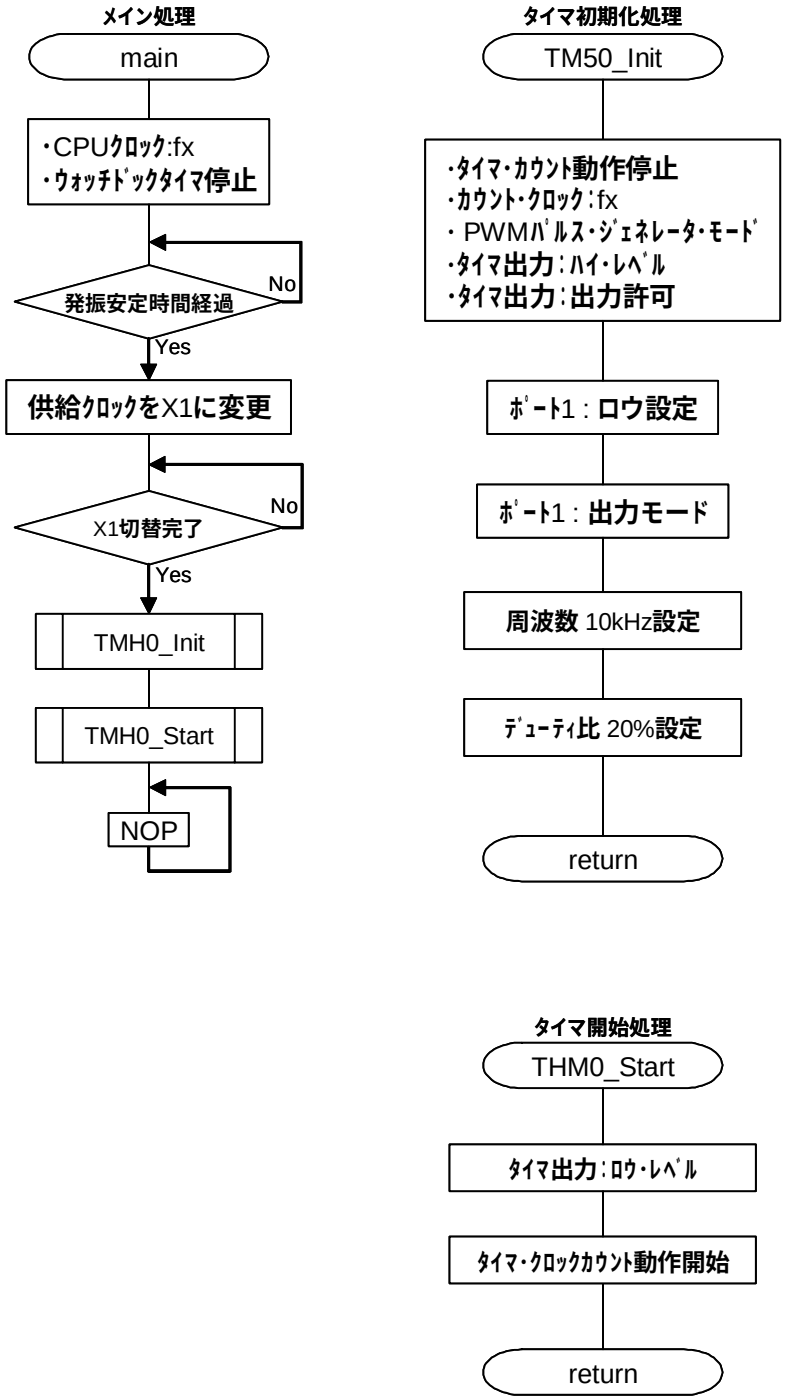
（3）8ビット精度のキャリア・ジェネレータ・モード（８ビット・タイマH1のみ）

8ビット・タイマH1で生成されるキャリア・クロックを，8ビット・タイマ／イベント・カウンタ51で設定した周期で出力します。

このプログラムは、P15/TOH0 より、10ｋHz(ﾃﾞｭｰﾃｨ比 20%)の波形を出力します。そしてメイン処理は、出力設定後 無限ループにします。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ（10MHz）を選択します。

初期設定としてタイマのカウントクロックは2.5MHz を選択します。そしてコンペアレジスタ(CMP00)は周波数を決定し、249 までのカウントで 100μs(10kHz)の時間を作成しています。コンペアレジスタ(CMP10)はデューティ比を決定し、199 までのカウントで 20μs(ﾃﾞｭｰﾃｨ比 20%)のアクティブ幅を作成しています。タイマが 0 からのスタート後、CMP10 と一致するとインアクティブ・レベルを出力します。その後もカウントは継続して進み、CMP00 と一致するとアクティブレベルを出力します。このオペレーションはプログラムからの介在なしに連続的に行われます。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
No
発振安定時間経過
Yes
供給ｸﾛｯｸをX1に変更
No
X1切替完了
Yes
TMH0_Init
TMH0_Start
NOP
タイマ初期化処理
TM50_Init
・ﾀｲﾏ・ｶｳﾝﾄ動作停止
・ｶｳﾝﾄ・ｸﾛｯｸ:fx
・PWMﾊﾟﾙｽ・ｼﾞｪﾈﾚｰﾀ・ﾓｰﾄﾞ
・ﾀｲﾏ出力:ﾊｲ・ﾚﾍﾞﾙ
・ﾀｲﾏ出力:出力許可
ﾎﾟｰﾄ1 : ロウ設定
ﾎﾟｰﾄ1 : 出力モード
周波数 10kHz設定
ﾃﾞｭｰﾃｨ比 20%設定
return
タイマ開始処理
THM0_Start
ﾀｲﾏ出力:ﾛｳ・ﾚﾍﾞﾙ
ﾀｲﾏ・ｸﾛｯｸｶｳﾝﾄ動作開始
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: PWM output by TMH0
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - timer: 8-bit timer/event counter TMH0
*               - timer count clock: fx/4(5MHz)
*               - compare register CMP00 value: 0xf9
*               - compare register CMP10 value: 0xc7
*               - PWM period: 100us
*               - PWM duty-cycle: 20% (20us)
*               - output port: port 1.5 (P15/TOH0)
*
************************************************************************
*/
#pragma sfr
#pragma NOP


/*
------------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------------
*/

#define TM_TMH0_CLOCK       0x2
#define TM_TMH0_PWMCYCLE    0xf9
#define TM_TMH0_PWMDELAY    0xc7

/*status list definition*/
#define TRUE            1
#define FALSE           0

void TMH0_Init(void);
void TMH0_Start();


/*
------------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------------
*/

void main( void )
{
    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    TMH0_Init();                        /* TMH0 initialization function */
    TMH0_Start();                       /* TMH0 start function */

    while(TRUE)                         /* Endless loop */
    {
        NOP();
    }
}

/*
------------------------------------------------------------------------------
```

```
    Functions Group
-------------------------------------------------------------------------------
*/

/*
-------------------------------------------------------------------------------
  Abstract:
    Initiate TMH0, select founction and input parameter
    count clock selection, INT init
-------------------------------------------------------------------------------
*/
void TMH0_Init( void )
{
    TMHMD0 = (TM_TMH0_CLOCK<<4)+0x09;   /* internal count clock */
                                        /* PWM mode, enable output */
    /*timerH0 PWM output*/

    P1 = 0x00;                          /* P1=0 */
    PM1 = 0x00;                         /* set P1 is output mode */

    CMP00 = TM_TMH0_PWMCYCLE;           /* cycle data set */
    CMP10 = TM_TMH0_PWMDELAY;           /* delay data set */
}


/*
-------------------------------------------------------------------------------
  Abstract:
    Start the timerH0 counter
-------------------------------------------------------------------------------
*/
void TMH0_Start( void )
{
    TOLEV0 = 0;                         /* Timer output level: Low */
    TMHE0 = 1;                          /* Enables timer count operation */
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: PWM output by TMH0
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - timer: 8-bit timer/event counter TMH0
;               - timer count clock: fx/4(5MHz)
;               - compare register CMP00 value: 0xf9
;               - compare register CMP10 value: 0xc7
;               - PWM period: 100us
;               - PWM duty-cycle: 20% (20us)
;               - output port: port 1.5 (P15/TOH0)
;
;*****************************************************************************


;---------------------------------------------------------------------------
;       Macro Define
;---------------------------------------------------------------------------

TM_TMH0_CLOCK           EQU      02H
TM_TMH0_PWMCYCLE        EQU     0F9H
TM_TMH0_PWMDELAY        EQU     0C7H


;---------------------------------------------------------------------------
;       Specify Interrupt Vectors
;---------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to  Start
                DW      start


;---------------------------------------------------------------------------
;       Main Program
;---------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CALL    !TMH0_Init              ; TMH0 initialization function
        CALL    !TMH0_Start             ; TMH0 start function

main:                                   ; Endless loop
        NOP
        BR      $main
```

```
;-------------------------------------------------------------------------------
;       Functions Group
;-------------------------------------------------------------------------------


;-------------------------------------------------------------------------
;  Abstract:
;       Initiate TMH0, select founction and input parameter
;       count clock selection, INT init
;-------------------------------------------------------------------------
TMH0_Init:

        MOV     A, #09H
        OR      A, #TM_TMH0_CLOCK SHL 4
        MOV     TMHMD0, A                       ; internal count clock

        ;TimerH0 PWM output function

        MOV     P1 ,#00H
        MOV     PM1 ,#00H

        MOV     CMP00, #TM_TMH0_PWMCYCLE        ; cycle data set
        MOV     CMP10, #TM_TMH0_PWMDELAY        ; delay data set

        RET


;-------------------------------------------------------------------------
;  Abstract:
;       Start the timerH0 counter
;-------------------------------------------------------------------------
TMH0_Start:
        CLR1    TOLEV0                          ; Timer output level: Low
        SET1    TMHE0                           ; Enables timer count operation

        RET

END
```

３．６ 時計用タイマ

■ ファイル名

- WatchTimer.asm（アセンブリ言語ソース）
- WatchTimer.c（C言語ソース）

■ 説明

時計用タイマには，次のような機能があります。

（1）時計用タイマ

X1入力クロックまたはサブシステム・クロックを使用し，一定の時間間隔ごとに，割込み要求（INTWT）を発生します。

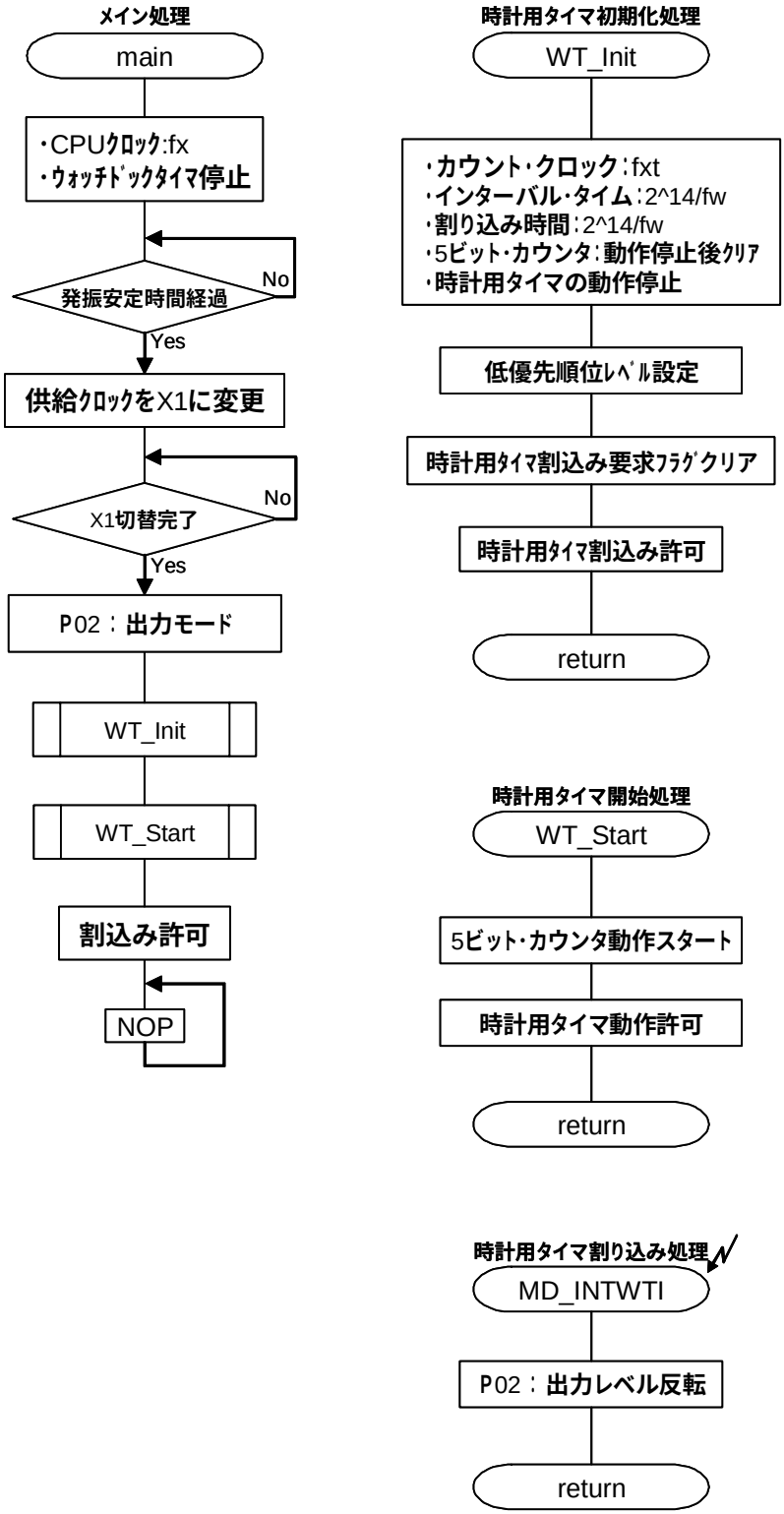
（2）インターバル・タイマ

あらかじめ設定したカウント値をインターバルとし，繰り返し割込み要求（INTWTI）を発生するインターバル・タイマとして動作します。

このプログラムは、0.5s 毎の時計タイマ割り込み処理内で、P02 を反転出力させます。そしてメイン処理は、出力設定後 無限ループにします。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

初期設定としてタイマのカウントクロックには、サブクロックの 32.768kHz を選択します。そして時計用タイマを選択し、割り込み間隔は 0.5s に設定します。時計用タイマ動作開始後 0.5s 毎に発生する時計用タイマ割り込み（INTWT）内にて、P02 の出力レベルを反転させます。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
P02：出力モード
WT_Init
WT_Start
割込み許可
NOP
時計用ﾀｲﾏ初期化処理
WT_Init
・カウント・クロック：fxt
・インターバル・タイム：2^14/fw
・割り込み時間：2^14/fw
・5ビット・カウンタ：動作停止後ｸﾘｱ
・時計用タイマの動作停止
低優先順位ﾚﾍﾞﾙ設定
時計用ﾀｲﾏ割込み要求ﾌﾗｸﾞｸﾘｱ
時計用ﾀｲﾏ割込み許可
return
時計用タイマ開始処理
WT_Start
5ビット・カウンタ動作スタート
時計用タイマ動作許可
return
時計用タイマ割り込み処理
MD_INTWTI
P02：出力レベル反転
return

■ C言語ソース

```
/*
********************************************************************************
*
*  Title: Watch timer
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - subsystem clock 32.768kHz
*               - port 0.2 toggles every 0.5s
*
********************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI

/*
--------------------------------------------------------------------------------
    Specify Interrupt Vectors
--------------------------------------------------------------------------------
*/
#pragma interrupt   INTWT       MD_INTWT


/*
--------------------------------------------------------------------------------
    Constants/Variables
--------------------------------------------------------------------------------
*/

/*status list definition*/
#define TRUE            1
#define FALSE           0

void WT_Init(void);
void WT_Start( void );


/*
--------------------------------------------------------------------------------
    Main Program
--------------------------------------------------------------------------------
*/

void main( void )
{

    PCC = 0x00;                         /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    PM0.2 = 0;                          /* set P02 is output mode */

    WT_Init();                          /* WT initialization function */
    WT_Start();                         /* WT start function */

    EI();                               /* Master interruption permission */

    while(TRUE)                         /* Endless loop */
    {
        NOP();
    }
```

```
}


/*
--------------------------------------------------------------------------------
    Functions Group
--------------------------------------------------------------------------------
*/

/*
--------------------------------------------------------------------------
  Abstract:
    Initiate watch timer, select founction and input parameter
    count clock selection, INT init
--------------------------------------------------------------------------
*/
void WT_Init( void )
{
    WTM = 0x80;                         /* Set watch timer clock:fw=fxt*/
                                        /* watch timer flag setting:2^14/fw(0.5s)*/
                                        /* Interval timer prescaler setting:2^4/fw*/

    /*INTWT enable*/
    WTPR = 1;                           /* Low priority level */
    WTIF = 0;                           /* No interrupt request signal is generated */
    WTMK = 0;                           /* Interrupt servicing enabled */
}

/*
--------------------------------------------------------------------------
  Abstract:
    Start the watch timer counter
--------------------------------------------------------------------------
*/
void WT_Start( void )
{
    /* enable watch timer operation */
    /* start the 5-bit counter */
    WTM |= 0x03;
}

/*
--------------------------------------------------------------------------
  Abstract:
    INTWT interrupt service routine.
--------------------------------------------------------------------------
*/
__interrupt void MD_INTWT( void )
{
    P0.2 ^= 1;                          /* P0.2 output reversal */
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: Watch timer
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - subsystem clock 32.768kHz
;               - port 0.2 toggles every 0.5s
;
;*****************************************************************************



;-----------------------------------------------------------------------------
;       Macro Define
;-----------------------------------------------------------------------------


;-----------------------------------------------------------------------------
;       Specify Interrupt Vectors
;-----------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to  Start
                DW      start
WT_VECTOR       CSEG AT 002Eh
                DW      MD_INTWT


;-----------------------------------------------------------------------------
;       Main Program
;-----------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CLR1    PM0.2

        CALL    !WT_Init                ; WT initialization function
        CALL    !WT_Start               ; WT start function

        EI                              ; Master interruption permission

main:                                   ; Endless loop
        NOP
        BR      $main


;-----------------------------------------------------------------------------
;       Functions Group
;-----------------------------------------------------------------------------
```

```
;-------------------------------------------------------------------------
;  Abstract:
;       Initiate watch timer, select founction and input parameter
;       count clock selection, INT init
;-------------------------------------------------------------------------
WT_Init:
        MOV     WTM, #80H               ; Set watch timer clock:fw=fxt
                                        ; watch timer flag setting:2^14/fw
                                        ; Interval timer prescaler setting:2^4/fw

        ;INTWT enable
        SET1    WTPR                    ; Low priority level
        CLR1    WTIF                    ; No interrupt request signal is generated
        CLR1    WTMK                    ; Interrupt servicing enabled

        RET


;-------------------------------------------------------------------------
;  Abstract:
;       Start the watch timer counter
;-------------------------------------------------------------------------
WT_Start:
        ;enable watch timer operation
        ;start the 5-bit counter
        SET1    WTM1
        SET1    WTM0

        RET


;-------------------------------------------------------------------------
;  Abstract:
;       INTWT interrupt service routine.
;-------------------------------------------------------------------------
MD_INTWT:
        MOV1    CY, P0.2
        NOT1    CY
        MOV1    P0.2, CY                ; P0.2 output reversal

        RETI

END
```

３．７ ウォッチドッグ･タイマ

■ ファイル名

- WatchdogTimer.asm（アセンブリ言語ソース）
- WatchdogTimer.c（Ｃ言語ソース）

■ 説明

プログラムの暴走を検出します。暴走検出時，内部リセット信号（WDTRES）を発生します。
ウォッチドッグ・タイマによるリセットが発生した場合，リセット・コントロール・フラグ・レジスタ（RESF）のビット4（WDTRF）がセット（1）されます。

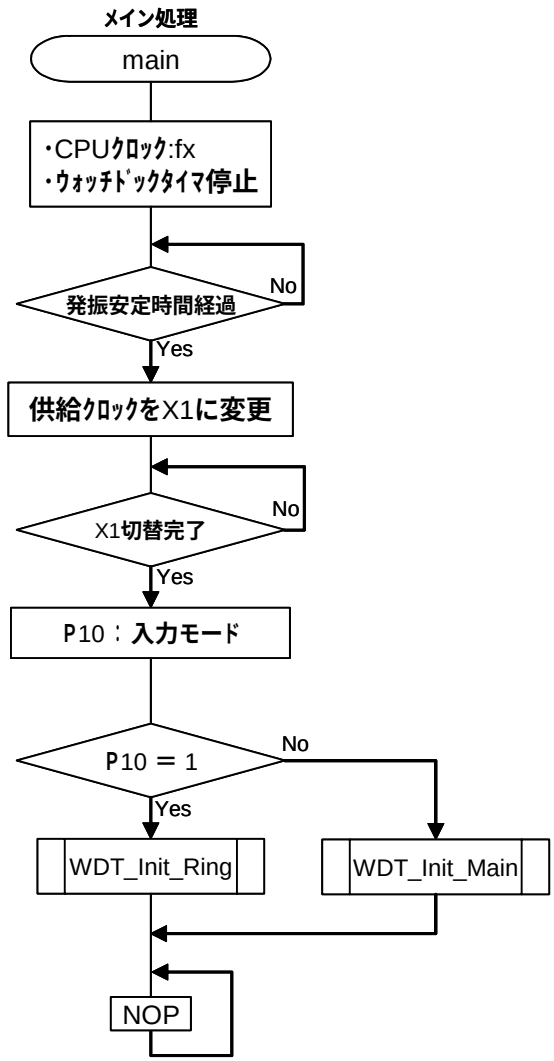
このプログラムは、P10 の状態によって WDT の暴走検出時間およびクロック･ソースを下記に示すとおり切替えます。

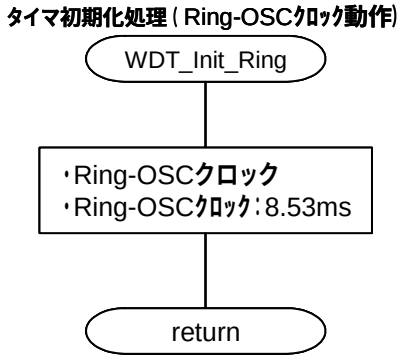
P10=High(1)の時、Ring-OSC 選択で 8.53ms に設定します
P10=Low(0)の時、メインクロック選択で 105ms に設定します

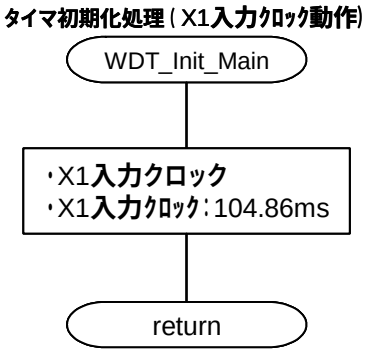
そしてメイン処理は、出力設定後 無限ループにします。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

プログラムの起動前に P10 により予めウォッチドッグ・タイマのクロックを選択しておきます。プログラム起動後は､P10=1 の時は､Ring-OSC クロックにて動作させ暴走検出時間を 8.53ms に設定します。P10=0 の時は、メインクロックにて動作させ暴走検出時間を 104.86ms に設定します。そして暴走検出までの時間が経過すると、内部リセット信号（WDTRES）が発生します。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
P10：入力モード
P10 ＝ 1
No
Yes
WDT_Init_Ring
WDT_Init_Main
NOP


タイマ初期化処理（Ring-OSCｸﾛｯｸ動作）
WDT_Init_Ring
・Ring-OSCクロック
・Ring-OSCｸﾛｯｸ：8.53ms
return


タイマ初期化処理（X1入力ｸﾛｯｸ動作）
WDT_Init_Main
・X1入力クロック
・X1入力ｸﾛｯｸ：104.86ms
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: Watchdog timer
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - Ring-OSC clock 240kHz
*               - port 1.0 input Hi:
*                   non-maskable interrupt is generated every 8.53ms
*               - port 1.0 input Low:
*                   non-maskable interrupt is generated every 104.86ms
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

/*status list definition*/
#define TRUE            1
#define FALSE           0

void WDT_Init_Ring(void);
void WDT_Init_Main( void );


/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/

void main( void )
{

    PCC = 0x00;                         /* CPU clock: fx */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    PM1.0 = 1;                          /* set P10 is input mode */

    /* control port level judge */
    if( P1.0 == 1)
    {
        WDT_Init_Ring();
    }
    else
    {
        WDT_Init_Main();
    }

    while(TRUE)                         /* Endless loop */
    {
        NOP();
    }
```

```
}


/*
-------------------------------------------------------------------------------
    Functions Group
-------------------------------------------------------------------------------
*/

/*
-------------------------------------------------------------------------
  Abstract:
    Initiate watchdog timer, select founction and input parameter
    count clock selection, INT init
-------------------------------------------------------------------------
*/
void WDT_Init_Ring( void )
{
    WDTM = 0x60;            /* Ring-OSC clock 8.53ms */
}

void WDT_Init_Main( void )
{
    WDTM = 0x6f;            /* Main clock 104.86ms */
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: Watchdog timer
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - Ring-OSC clock 240kHz
;               - port 1.0 input Hi:
;                   non-maskable interrupt is generated every 8.53ms
;               - port 1.0 input Low:
;                   non-maskable interrupt is generated every 105ms
;
;*****************************************************************************


;-----------------------------------------------------------------------------
;       Macro Define
;-----------------------------------------------------------------------------


;-----------------------------------------------------------------------------
;       Specify Interrupt Vectors
;-----------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to  Start
                DW      start


;-------------------------------------------------------------------------------
;       Main Program
;-------------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        SET1    PM1.0

        BF      P1.0, $start10
        CALL    !WDT_Init_Ring
        BR      $main

start10:
        CALL    !WDT_Init_Main

main:                                   ; Endless loop
        NOP
        BR      $main


;-------------------------------------------------------------------------------
```

```
;       Functions Group
;-------------------------------------------------------------------------------


;-------------------------------------------------------------------------
;  Abstract:
;       Initiate watchdog timer, select founction and input parameter
;       count clock selection, INT init
;-------------------------------------------------------------------------
WDT_Init_Ring:
        MOV     WDTM, #60H              ; Ring-OSC clock 8.53ms

        RET


WDT_Init_Main:
        MOV     WDTM, #6FH              ; Main clock 104.86ms

        RET

END
```

３．８ クロック出力

3.8.1 クロック出力

■ ファイル名

Clock_out

- CLK_Clock.asm（アセンブリ言語ソース）
- CLK_Clock.c（Ｃ言語ソース）

■ 説明

クロック出力はリモコン送信時のキャリア出力や周辺SIに供給するクロックを出力する機能です。また，ブザー出力はCKSで選択したブザー周波数の方形波を出力する機能です。

このプログラムは、P01 端子の状態により、P140/PCL からの出力を下記に示すように制御します。

P01 が High の時、P140/PCL から f_{XT}(32.768kHz)クロックを出力します。

P01 が Low の時、P140/PCL からのクロック出力を停止させます。

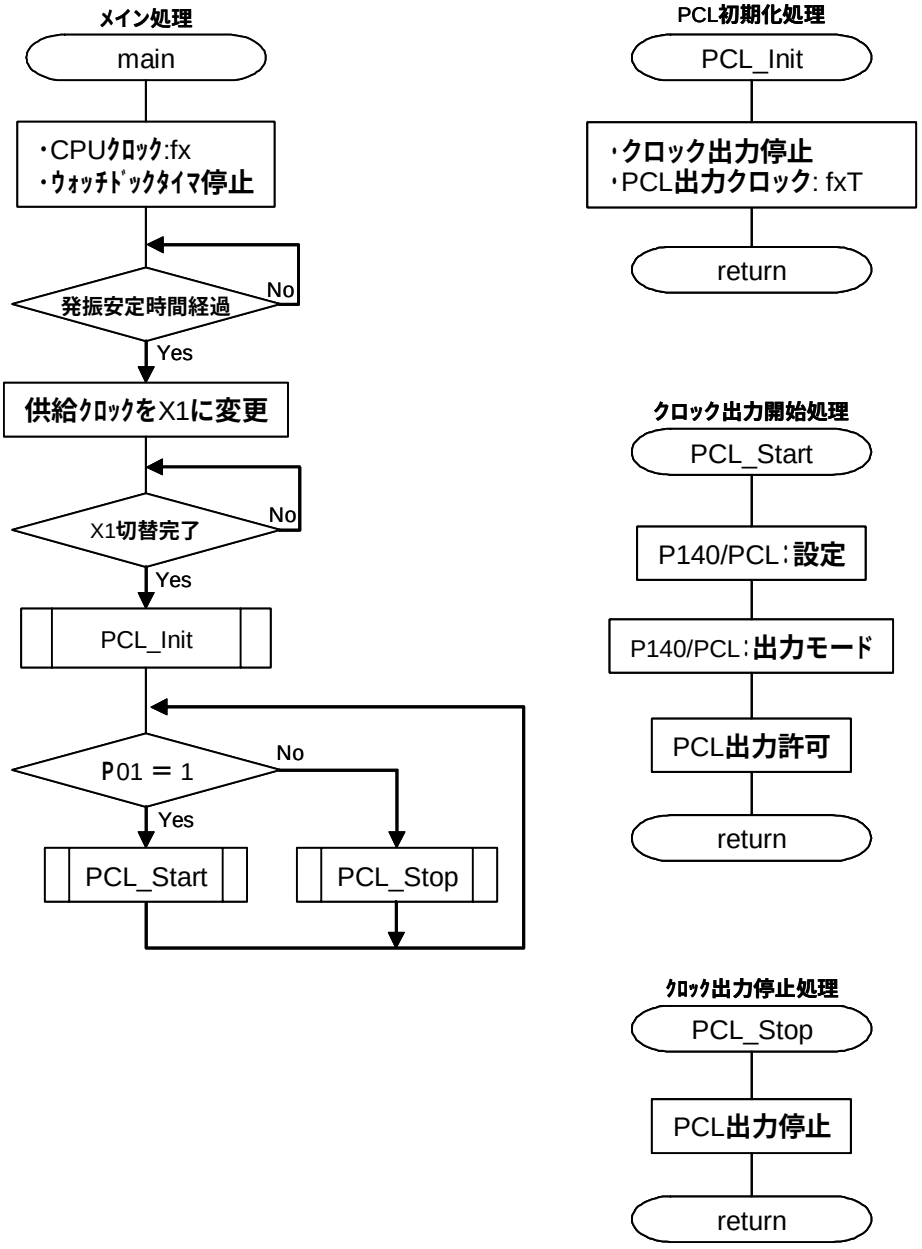
また、CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

クロック出力制御判断用に P01 の入力レベルを使用します。出力クロックはサブクロックを選択しておきます。プログラム動作後、P01 入力レベルにより下記動作をおこないます。

P01=0 の時、PCL 端子からのクロック出力が無効になり、Low 出力を行います。

P01=1 の時、PCL 端子から設定(32.768kHz)したクロックを出力します。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
PCL_Init
P01 ＝ 1
No
Yes
PCL_Start
PCL_Stop
PCL初期化処理
PCL_Init
・クロック出力停止
・PCL出力クロック: fxT
return
クロック出力開始処理
PCL_Start
P140/PCL：設定
P140/PCL：出力モード
PCL出力許可
return
ｸﾛｯｸ出力停止処理
PCL_Stop
PCL出力停止
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: Clock output
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - port 14.0 outputs 32.768kHz subsystem clock when port 0.1 = 1
*               - port 14.0 outputs low level when port 0.1 = 0
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

/*status list definition*/
#define TRUE            1
#define FALSE           0

void PCL_Init( void );
void PCL_Start( void );
void PCL_Stop( void );


/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/
void main( void )
{
    PCC = 0x00;                         /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    PCL_Init();                         /* PCL initialization function */

    while(TRUE)                         /* main loop */
    {
        /* Control port level judge */
        if( P0.1 == 1 ){
            PCL_Start();                /* PCL Start function */
        }
        else{
            PCL_Stop();                 /* PCL Stop function */
        }
    }
}


/*
------------------------------------------------------------------------
    Functions Group
------------------------------------------------------------------------
```

```
*/
/*
---------------------------------------------------------------------------
  Abstract:
    This function initializes the Clock output/Buzzer output controller.
    Initial Clock output function according to the configuration setting .
---------------------------------------------------------------------------
*/
void PCL_Init( void )
{
    CKS = 0x08;                         /* stop PCL */
                                        /* fxt */
}


/*
---------------------------------------------------------------------------
  Abstract:
    This function enable clock output operation .
---------------------------------------------------------------------------
*/
void PCL_Start( void )
{
    P14.0 = 0;                          /* P140=0 */
    PM14.0 = 0;                         /* set p140 is output mode */

    CLOE = 1;                           /* enable PCL */
}


/*
---------------------------------------------------------------------------
  Abstract:
    This function disable clock output operation .
---------------------------------------------------------------------------
*/
void PCL_Stop( void )
{
    CLOE = 0;                           /* disable PCL */
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: Clock output
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - port 14.0 outputs 32.768kHz subsystem clock when port 0.1 = 1
;               - port 14.0 outputs low level when port 0.1 = 0
;
;*****************************************************************************


;---------------------------------------------------------------------------
;       Specify Interrupt Vectors
;---------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ;On reset, go to  Start
                DW      start


;---------------------------------------------------------------------------
;       Main Program
;---------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CALL    !PCL_Init               ; PCL initialization function

main:                                   ; main loop
        BF      P0.1, $main01
        CALL    !PCL_Start              ; PCL Start function
        BR      $main

main01:
        CALL    !PCL_Stop               ; PCL Stop function
        BR      $main


;---------------------------------------------------------------------------
;       Functions Group
;---------------------------------------------------------------------------


;-----------------------------------------------------------------------
;  Abstract:
;       This function initializes the Clock output/Buzzer output controller.
;       Initial Clock output function according to the configuration setting .
;-----------------------------------------------------------------------
PCL_Init:
```

```
        MOV     CKS, #08H               ; stop PCL
                                        ; fxt
        RET


;-------------------------------------------------------------------------
;  Abstract:
;       This function enable clock output operation .
;-------------------------------------------------------------------------
PCL_Start:
        CLR1    P14.0                   ; P140=0
        CLR1    PM14.0                  ; set p140 is output mode
        SET1    CLOE                    ; enable PCL

        RET


;-------------------------------------------------------------------------
;  Abstract:
;       This function disable clock output operation .
;-------------------------------------------------------------------------
PCL_Stop:
        CLR1    CLOE                    ; disable PCL

        RET

END
```

3.8.2 **ブザー出力**

■ ファイル名

Buzzer_out

- CLK_Buzzer.asm（アセンブリ言語ソース）
- CLK_Buzzer.c（C言語ソース）

■ 説明

クロック出力はリモコン送信時のキャリア出力や周辺LSIに供給するクロックを出力する機能です。また，ブザー出力はCKSで選択したブザー周波数の方形波を出力する機能です。

このプログラムは、P01 端子の状態により、P141/BUZ からの出力を下記に示すように制御します。

P01 が High の時、P141/BUZ から f_X(2.44kHz)のクロックを出力します。

P01 が Low の時、P141/BUZ からのクロック出力を停止させます。

また、CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

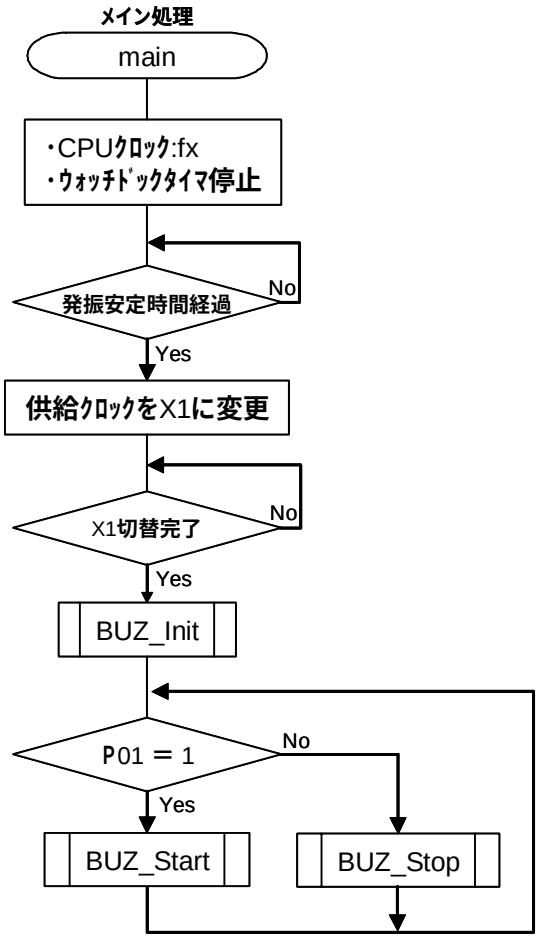
ブザー出力制御判断用に P01 の入力レベルを使用します。出力クロックは 2.44kHz を選択しておきます。

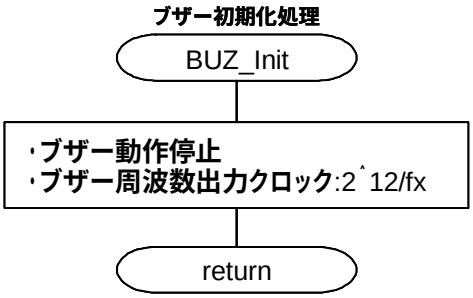
プログラム動作後、P01 入力レベルにより下記動作をおこないます。

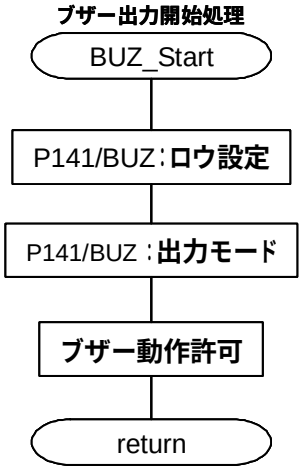
P01=0 の時、BUZ 端子からのクロック出力が無効になり、Low 出力を行います。

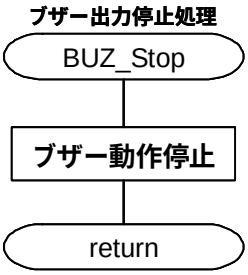
P01=1 の時、BUZ 端子から設定(2.44kHz)したクロックを出力します。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
BUZ_Init
P01 ＝ 1
No
Yes
BUZ_Start
BUZ_Stop


ブザー初期化処理
BUZ_Init
・ブザー動作停止
・ブザー周波数出力クロック:2^12/fx
return


ブザー出力開始処理
BUZ_Start
P141/BUZ:ロウ設定
P141/BUZ:出力モード
ブザー動作許可
return


ブザー出力停止処理
BUZ_Stop
ブザー動作停止
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: Buzzer output
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - port 14.1 outputs 2.44kHz subsystem clock when port 0.1 = 1
*               - port 14.1 outputs low level when port 0.1 = 0
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

/*status list definition*/
#define TRUE            1
#define FALSE           0

void BUZ_Init( void );
void BUZ_Start( void );
void BUZ_Stop( void );

/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/

void main( void )
{

    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    BUZ_Init();                         /* Buzzer initialization function */

    while(TRUE)                         /* main loop */
    {
        /* control port level judge */
        if( P0.1 == 1 ){
            BUZ_Start();                /* Buzzer Start function */
        }
        else{
            BUZ_Stop();                 /* Buzzer Stop function */
        }
    }
}


/*
------------------------------------------------------------------------
    Functions Group
```

```
------------------------------------------------------------------------------
*/

/*
------------------------------------------------------------------------------
  Abstract:
    This function initializes the Clock output/Buzzer output controller.
    Initial BUZ function according to the configuration setting .
------------------------------------------------------------------------------
*/
void BUZ_Init( void )
{
    CKS = 0x40;                         /* stop buzzer */
                                        /* fx/2^12 */
}

/*
------------------------------------------------------------------------------
  Abstract:
    This function enable Buzzer operation .
------------------------------------------------------------------------------
*/
void BUZ_Start( void )
{
    P14.1 = 0;                          /* P141=0 */
    PM14.1 = 0;                         /* set p141 is output mode */
    BZOE = 1;                           /* enable buzzer */
}


/*
------------------------------------------------------------------------------
  Abstract:
    This function disable Buzzer operation .
------------------------------------------------------------------------------
*/
void BUZ_Stop( void )
{
    BZOE = 0;                           /* disable buzzer */
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: Buzzer output
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - port 14.1 outputs 2.44kHz subsystem clock when port 0.1 = 1
;               - port 14.1 outputs low level when port 0.1 = 0
;
;*****************************************************************************


;-----------------------------------------------------------------------------
;       Specify Interrupt Vectors
;-----------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ;On reset, go to  Start
                DW      start


;-----------------------------------------------------------------------------
;       Main Program
;-----------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CALL    !BUZ_Init               ; Buzzer initialization function

main:                                   ; main loop
        BF      P0.1, $main01           ; control port level judge
        CALL    !BUZ_Start              ; Buzzer Start
        BR      $main

main01:
        CALL    !BUZ_Stop               ; Buzzer Stop
        BR      $main


;-----------------------------------------------------------------------------
;       Functions Group
;-----------------------------------------------------------------------------


;-----------------------------------------------------------------------
;  Abstract:
;       This function initializes the Clock output/Buzzer output controller.
;       Initial BUZ function according to the configuration setting .
;-----------------------------------------------------------------------
BUZ_Init:
```

```
        MOV     CKS, #40H               ;stop buzzer

        RET


;---------------------------------------------------------------------------
;  Abstract:
;       This function enable Buzzer operation .
;---------------------------------------------------------------------------
BUZ_Start:
        CLR1    P14.1
        CLR1    PM14.1
        SET1    BZOE                    ;enable buzzer

        RET


;---------------------------------------------------------------------------
;  Abstract:
;       This function disable Buzzer operation .
;---------------------------------------------------------------------------
BUZ_Stop:
        CLR1    BZOE                    ;disable buzzer

        RET

END
```

３．９ A／Dコンバータ

3.9.1 ソフト･トリガ

■ ファイル名

Soft

- AD_Soft.asm（アセンブリ言語ソース）
- AD_Soft.c（C言語ソース）

■ 説明

A/Dコンバータは，アナログ入力をディジタル値に変換する10ビット分解能のコンバータで，最大８チャネル（ANI0-ANI7）のアナログ入力を制御できる構成になっています。

A/Dコンバータには，次のような機能があります。

（1）10ビット分解能A/D変換

アナログ入力をANI0-ANI7から１チャネル選択し，10ビット分解能のA/D変換動作を繰り返します。A/D変換を１回終了するたびに，割込み要求（INTAD）を発生します。

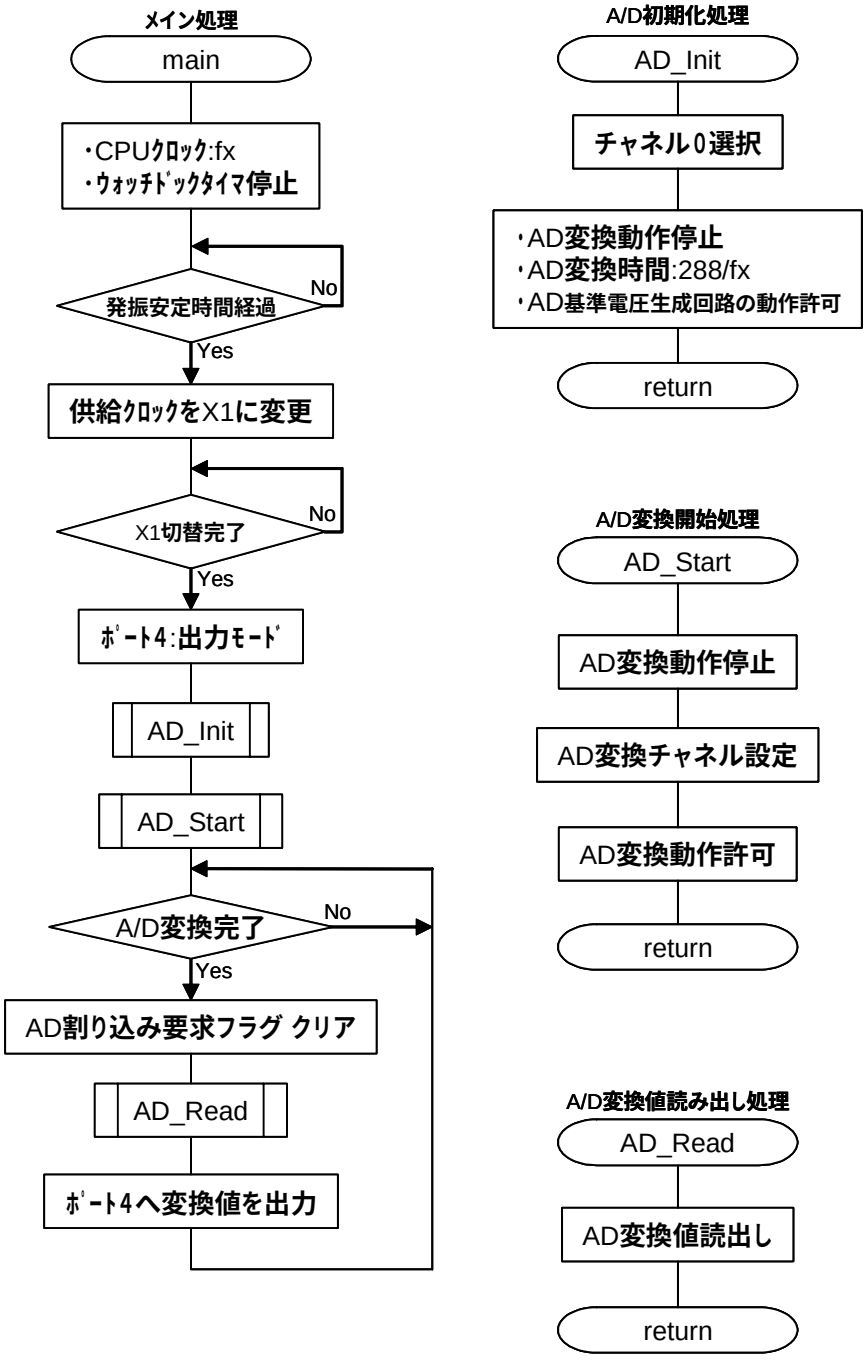
（2）パワーフェイル検出機能

バッテリ電圧低下を検出するための機能です。A/D変換結果（ADCRレジスタ値）とパワーフェイル比較しきい値レジスタ（PFT）の値との大小比較を行い，比較条件に合致した場合のみINTADを発生します。

このプログラムは、メイン処理において P20/ANI0 端子を A/D 変換し、変換結果をポート 4(下位 2bit を切り捨て 8bit)に出力し、この処理を無限に繰り返します。変換時間は 288/fx(57.6μs)を選択します。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

初期設定として、チャンネル 0（P20/ANI0）をアナログ入力用に選択します。そして変換時間を 28.8μs に設定します。メインループにて A/D 変換を繰り返し開始させます。変換の終了は A/D 変換終了フラグ（ADIF）がセットされるのを検知します。変換完了後、 変換結果の下位 2 ビットを切り捨てた上位 8 ビットのデータを、ポート 4 から出力します。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
ﾎﾟｰﾄ4:出力ﾓｰﾄﾞ
AD_Init
AD_Start
A/D変換完了
No
Yes
AD割り込み要求フラグ クリア
AD_Read
ﾎﾟｰﾄ4へ変換値を出力
A/D初期化処理
AD_Init
チャネル0選択
・AD変換動作停止
・AD変換時間:288/fx
・AD基準電圧生成回路の動作許可
return
A/D変換開始処理
AD_Start
AD変換動作停止
AD変換チャネル設定
AD変換動作許可
return
A/D変換値読み出し処理
AD_Read
AD変換値読出し
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: A/D conversion by the soft trigger
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - A/D conversion time: 28.8us
*               - A/D resolution: 10 bits
*               - A/D start: Setting of ADCS bit
*               - A/D stop: none (left operating continuously)
*               - A/D channel: channel 0 (ANI0)
*               - A/D input pin: port 2.0 (P20/ANI0)
*               - A/D output: port 4 (8 bits)
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

/* Channel number */
enum ANChannel{Channel0, Channel1, Channel2, Channel3, Channel4, Channel5, Channel6, Channel7};

/*status list definition*/
#define TRUE            1
#define FALSE           0


void AD_Init(void);
void AD_Start(enum ANChannel channel);
void AD_Read( unsigned short* buffer );


/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/

void main( void )
{
    unsigned short  usAD_DATA;

    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    PM4 = 0x00;                         /* set P4 is output mode */
    AD_Init();                          /* A/D initialization function */
    AD_Start( Channel0 );               /* A/D start function */

    while(TRUE)                         /* main loop */
    {
        /* Waiting for the completion of A/D conversion */
```

```
        while( ADIF == 0 );
        ADIF = 0;

        AD_Read( &usAD_DATA );          /* A/D Read function */
        P4 = (unsigned char)( usAD_DATA >> 8 ); /* conversion result output */
    }
}


/*
--------------------------------------------------------------------------------
    Functions Group
--------------------------------------------------------------------------------
*/

/*
--------------------------------------------------------------------------
  Abstract:
    This function initializes the A/D converter. Conversion is stopped
    and the interrupt is disabled before the configuration is made.
--------------------------------------------------------------------------
*/
void AD_Init( void )
{
    ADS = 0x00;                         /* Channel:ANI0 */
    ADM = 0x01;                         /* Conversion time: 288/fx(28.8us) */
                                        /* standard voltage generation circuit: stop */
                                        /* conversion operation: stop */
}

/*
--------------------------------------------------------------------------
  Abstract:
    This function starts the A/D converter .
--------------------------------------------------------------------------
*/
void AD_Start(enum ANChannel channel)
{
    ADCS = 0;                           /* Conversion operation stop */

    ADS = channel;                      /* Channel set */
    ADCS = 1;                           /* Conversion operation permission */
}

/*
--------------------------------------------------------------------------
  Abstract:
    This function can be called after an A/D conversion is completed,
    and returns the conversion result(s) in the buffer.
--------------------------------------------------------------------------
*/
void AD_Read( unsigned short* buffer )
{
    *buffer = ADCR;
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: A/D conversion by the soft trigger
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - A/D conversion time: 28.8us
;               - A/D resolution: 10 bits
;               - A/D start: Setting of ADCS bit
;               - A/D stop: none (left operating continuously)
;               - A/D channel: channel 0 (ANI0)
;               - A/D input pin: port 2.0 (P20/ANI0)
;               - A/D output: port 4 (8 bits)
;
;*****************************************************************************


;---------------------------------------------------------------------------
;       Specify Interrupt Vectors
;---------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ;On reset, go to  Start
                DW      start


;---------------------------------------------------------------------------
;       Main Program
;---------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; supply clock: X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        MOV     PM4, #00H               ; set P4 is output mode

        CALL    !AD_Init                ; A/D initialization function
        MOV     A, #00H
        CALL    !AD_Start               ; A/D start function

main:
        BTCLR   ADIF, $main01           ; Waiting for the completion of A/D conversion
        BR      $main

main01:
        CALL    !AD_Read                ; A/D Read function
        MOV     P4, A                   ; conversion result output

        BR      $main


;---------------------------------------------------------------------------
```

```
;       Functions Group
;-------------------------------------------------------------------------------


;-------------------------------------------------------------------------
;  Abstract:
;       This function initializes the A/D converter. Conversion is stopped
;       and the interrupt is disabled before the configuration is made.
;-------------------------------------------------------------------------
AD_Init:

;--- Channel 0 Selected ---
        MOV     ADS, #00H

;--- Selection of conversion time=288/fx
        MOV     ADM, #01H

        RET

;-------------------------------------------------------------------------
;  Abstract:
;       This function starts the A/D converter .
;-------------------------------------------------------------------------
AD_Start:
;--- Stop the A/D converter ---
        CLR1    ADCS
        MOV     ADS, A
        SET1    ADCS

        RET


;-------------------------------------------------------------------------
;  Abstract:
;      This function can be called after an A/D conversion is completed,
;      and returns the conversion result(s) in the buffer.
;-------------------------------------------------------------------------
AD_Read:

;--- Read result after conversion ---
        MOVW    AX, ADCR

        RET

END
```

3.9.2 ハード･トリガ

■ ファイル名

Hard

- AD_Hard.asm（アセンブリ言語ソース）
- AD_Hard.c（C言語ソース）

■ 説明

A/Dコンバータは，アナログ入力をディジタル値に変換する10ビット分解能のコンバータで，最大８チャネル（ANI0-ANI7）のアナログ入力を制御できる構成になっています。

A/Dコンバータには，次のような機能があります。

（1）10ビット分解能A/D変換

アナログ入力をANI0-ANI7から１チャネル選択し，10ビット分解能のA/D変換動作を繰り返します。A/D変換を１回終了するたびに，割込み要求（INTAD）を発生します。

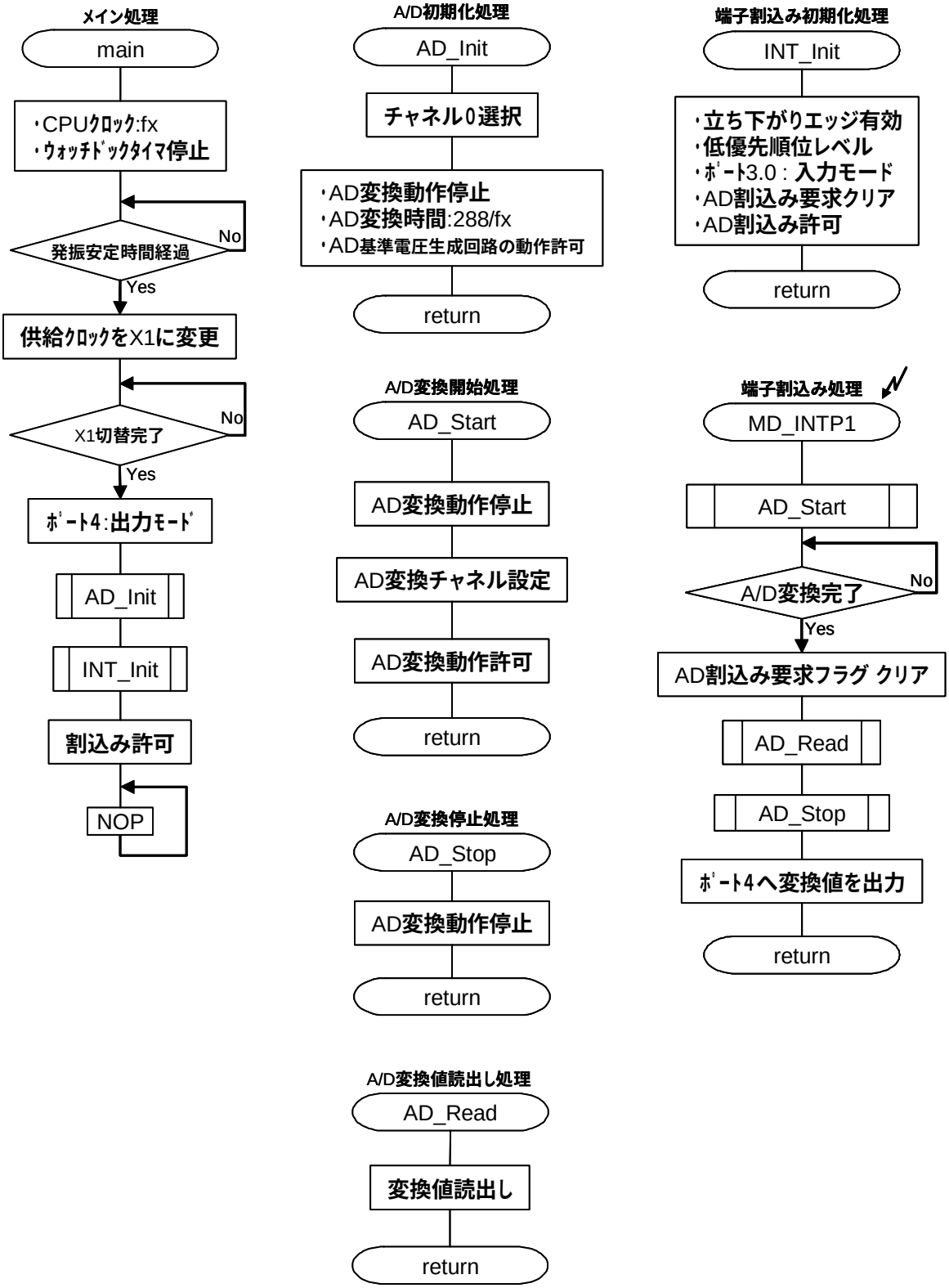
（2）パワーフェイル検出機能

バッテリ電圧低下を検出するための機能です。A/D変換結果（ADCRレジスタ値）とパワーフェイル比較しきい値レジスタ（PFT）の値との大小比較を行い，比較条件に合致した場合のみINTADを発生します。

このプログラムは、P30/INTP1 の立下りエッジ毎に P20/ANI0 を A/D 変換し、変換結果をポート 4（下位 2bit を切り捨て 8bit）に出力する事を繰り返します。そしてメイン処理は、出力設定後無限ループにします。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

初期設定として、チャンネル 0（P20/ANI0）をアナログ入力用に選択します。そして変換時間を 28.8μs に設定します。P30/INTP1 の立ち下がりエッジの検出毎に割込みを発生させ、割込み内にて A/D 変換を開始させます。変換の終了は A/D 変換終了フラグ（ADIF）がセットされるのを検知します。変換結果は下位 2 ビットを切り捨てた上位 8 ビットのデータをポート 4 から出力します。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
ﾎﾟｰﾄ4:出力ﾓｰﾄﾞ
AD_Init
INT_Init
割込み許可
NOP
A/D初期化処理
AD_Init
チャネル0選択
・AD変換動作停止
・AD変換時間:288/fx
・AD基準電圧生成回路の動作許可
return
端子割込み初期化処理
INT_Init
・立ち下がりエッジ有効
・低優先順位レベル
・ﾎﾟｰﾄ3.0：入力モード
・AD割込み要求クリア
・AD割込み許可
return
A/D変換開始処理
AD_Start
AD変換動作停止
AD変換チャネル設定
AD変換動作許可
return
端子割込み処理
MD_INTP1
AD_Start
A/D変換完了
No
Yes
AD割込み要求フラグ クリア
AD_Read
AD_Stop
ﾎﾟｰﾄ4へ変換値を出力
return
A/D変換停止処理
AD_Stop
AD変換動作停止
return
A/D変換値読出し処理
AD_Read
変換値読出し
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: A/D conversion by the hard trigger
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - A/D conversion time: 28.8us
*               - A/D resolution: 10 bits
*               - A/D start: falling edge on port 3.0 (P30/INTP1)
*               - A/D channel: channel 0 (ANI0)
*               - A/D input pin: port 2.0 (P20/ANI0)
*               - A/D output: port 4 (8 bits)
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI

/*
------------------------------------------------------------------------------
    Specify Interrupt Vectors
------------------------------------------------------------------------------
*/
#pragma interrupt   INTP1       MD_INTP1


/*
------------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------------
*/

/* Channel number */
enum ANChannel{Channel0, Channel1, Channel2, Channel3, Channel4, Channel5, Channel6, Channel7};

/*status list definition*/
#define TRUE            1
#define FALSE           0


void AD_Init(void);
void AD_Start(enum ANChannel channel);
void AD_Stop( void );
void AD_Read( unsigned short* buffer );
void INT_Init( void );


/*
------------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------------
*/

void main( void )
{

    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0 );
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );
```

```
    PM4 = 0x00;                         /* set P4 is output mode */
    AD_Init();                          /* A/D initialization function */
    INT_Init();                         /* INT initialization function */

    EI();                               /* Master interruption permission */

    while(TRUE)                         /* Endless loop */
    {
        NOP();
    }
}


/*
--------------------------------------------------------------------------------
    Functions Group
--------------------------------------------------------------------------------
*/

/*
--------------------------------------------------------------------------------
  Abstract:
    This function initializes the A/D converter. Conversion is stopped
    and the interrupt is disabled before the configuration is made.
--------------------------------------------------------------------------------
*/
void AD_Init( void )
{
    ADS = 0x00;                         /* Channel:ANI0 */
    ADM = 0x01;                         /* Conversion time: 288/fx(28.8us) */
                                        /* standard voltage generation circuit: stop */
                                        /* conversion operation: stop */
}

/*
--------------------------------------------------------------------------------
  Abstract:
    This function starts the A/D converter.
--------------------------------------------------------------------------------
*/
void    AD_Start(enum ANChannel channel)
{
    ADCS = 0;                           /* Conversion operation stop */

    ADS = channel;                      /* Channel set */
    ADCS = 1;                           /* Conversion operation permission */
}

/*
--------------------------------------------------------------------------------
  Abstract:
    This function stops the A/D converter.
--------------------------------------------------------------------------------
*/
void AD_Stop( void )
{
    ADCS = 0;                           /* Conversion operation stop */
}


/*
--------------------------------------------------------------------------------
  Abstract:
    This function can be called after an A/D conversion is completed,
    and returns the conversion result(s) in the buffer.
--------------------------------------------------------------------------------
*/
void AD_Read( unsigned short* buffer )
{
```

```
    *buffer = ADCR;
}

/*
---------------------------------------------------------------------------
  Abstract:
    Init the external INT, including enable or disable,
    priority setting
---------------------------------------------------------------------------
*/
void  INT_Init( void )
{
    EGP.1 = 0;                          /* Falling edge */
    EGN.1 = 1;

    PPR1 = 1;                           /* Low priority */
    PM3.0 = 1;                          /* set P30 input mode */
    PIF1 = 0;                           /* Interruption factor flag clear */
    PMK1 = 0;                           /* enable INTP1 */
}

/*
---------------------------------------------------------------------------
  Abstract:
    INTP1 Interrupt service routine.
---------------------------------------------------------------------------
*/
__interrupt void MD_INTP1( void )
{
    unsigned short usAD_DATA;

    AD_Start( Channel0 );               /* A/D start function */

    /* Waiting for the completion of A/D conversion */
    while( ADIF == 0 );
    ADIF = 0;

    AD_Read( &usAD_DATA );              /* A/D Read function */
    AD_Stop();                          /* A/D stop function */
    P4 = (unsigned char)( usAD_DATA >> 8 ); /* conversion result output */
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: A/D conversion by the hard trigger
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - A/D conversion time: 28.8us
;               - A/D resolution: 10 bits
;               - A/D start: falling edge on port 3.0 (P30/INTP1)
;               - A/D channel: channel 0 (ANI0)
;               - A/D input pin: port 2.0 (P20/ANI0)
;               - A/D output: port 4 (8 bits)
;
;*****************************************************************************



;---------------------------------------------------------------------------
;       Specify Interrupt Vectors
;---------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ;On reset, go to  Start
                DW      start
INTP1_VECTOR    CSEG AT 0008h
                DW      MD_INTP1


;---------------------------------------------------------------------------
;       Main Program
;---------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; supply clock: X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        MOV     PM4, #00H               ; set P4 is output mode

        CALL    !AD_Init                ; A/D initialization function
        CALL    !INT_Init               ; INT initialization function

        EI                              ; Master interruption permission

main:                                   ; Endless loop
        NOP
        BR      $main


;---------------------------------------------------------------------------
;       Functions Group
;---------------------------------------------------------------------------
```

```
;---------------------------------------------------------------------------
;  Abstract:
;       This function initializes the A/D converter. Conversion is stopped
;       and the interrupt is disabled before the configuration is made.
;---------------------------------------------------------------------------
AD_Init:

;--- Channel 0 Selected ---
        MOV     ADS, #00H

;--- Selection of conversion time=288/fx
        MOV     ADM, #01H

        RET


;---------------------------------------------------------------------------
;  Abstract:
;       This function starts the A/D converter .
;---------------------------------------------------------------------------
AD_Start:
;--- Stop the A/D converter ---
        CLR1    ADCS                    ; Conversion operation stop
        MOV     ADS, A
        SET1    ADCS                    ; Conversion operation permission

        RET


;---------------------------------------------------------------------------
;  Abstract:
;       This function stops the A/D converter.
;---------------------------------------------------------------------------
AD_Stop:
;--- Stop the A/D converter ---
        CLR1    ADCS                    ; Conversion operation stop

        RET


;---------------------------------------------------------------------------
;  Abstract:
;      This function can be called after an A/D conversion is completed,
;      and returns the conversion result(s) in the buffer.
;---------------------------------------------------------------------------
AD_Read:

;--- Read result after conversion ---
        MOVW    AX, ADCR

        RET


;---------------------------------------------------------------------------
;  Abstract:
;       Init the external INT, including enable or disable,
;       priority setting
;---------------------------------------------------------------------------
INT_Init:
;INTP1 valid edge and  priority selection
        CLR1    EGP.1                   ; Falling edge
        SET1    EGN.1
        SET1    PPR1                    ; Low priority
        SET1    PM3.0                   ; set P30 input mode
        CLR1    PIF1                    ; Interruption factor flag clear
        CLR1    PMK1                    ; enable INTP1

        RET
```

```
;-------------------------------------------------------------------------
;  Abstract:
;       INTP1 Interrupt service routine.
;-------------------------------------------------------------------------
MD_INTP1:

        SEL     RB1
        MOV     A, #00H
        CALL    !AD_Start               ; A/D start function

; Waiting for the completion of A/D conversion
MD_INTP1_01:
        BTCLR   ADIF, $MD_INTP1_02
        BR      MD_INTP1_01
MD_INTP1_02:

        CALL    !AD_Read                ; A/D Read function
        CALL    !AD_Stop                ; A/D stop function

        MOV     P4, A                   ; conversion result output

        RETI

END
```

３．１０ シリアル･インタフェースＵＡＲＴ０

■ ファイル名

Uart0

- UART0_trans.asm（アセンブリ言語ソース）
- UART0_trans.c（Ｃ言語ソース）

■ 説明

シリアル・インタフェースUART0には，次の2種類のモードがあります。

（1）動作停止モード

シリアル転送を行わないときに使用するモードです。消費電力を低減できます。

（2）アシンクロナス・シリアル・インタフェース（UART）モード

機能の概要を次に示します。

・2端子構成　TxD0：送信データの出力端子
　　RxB0：受信データの入力端子

・転送データのデータ長は7ビット／8ビット可変

・専用の5ビット・ボー・レート・ジェネレータを内蔵していることにより，任意のボー・レートが設定可能

・送信動作と受信動作は独立して動作することが可能

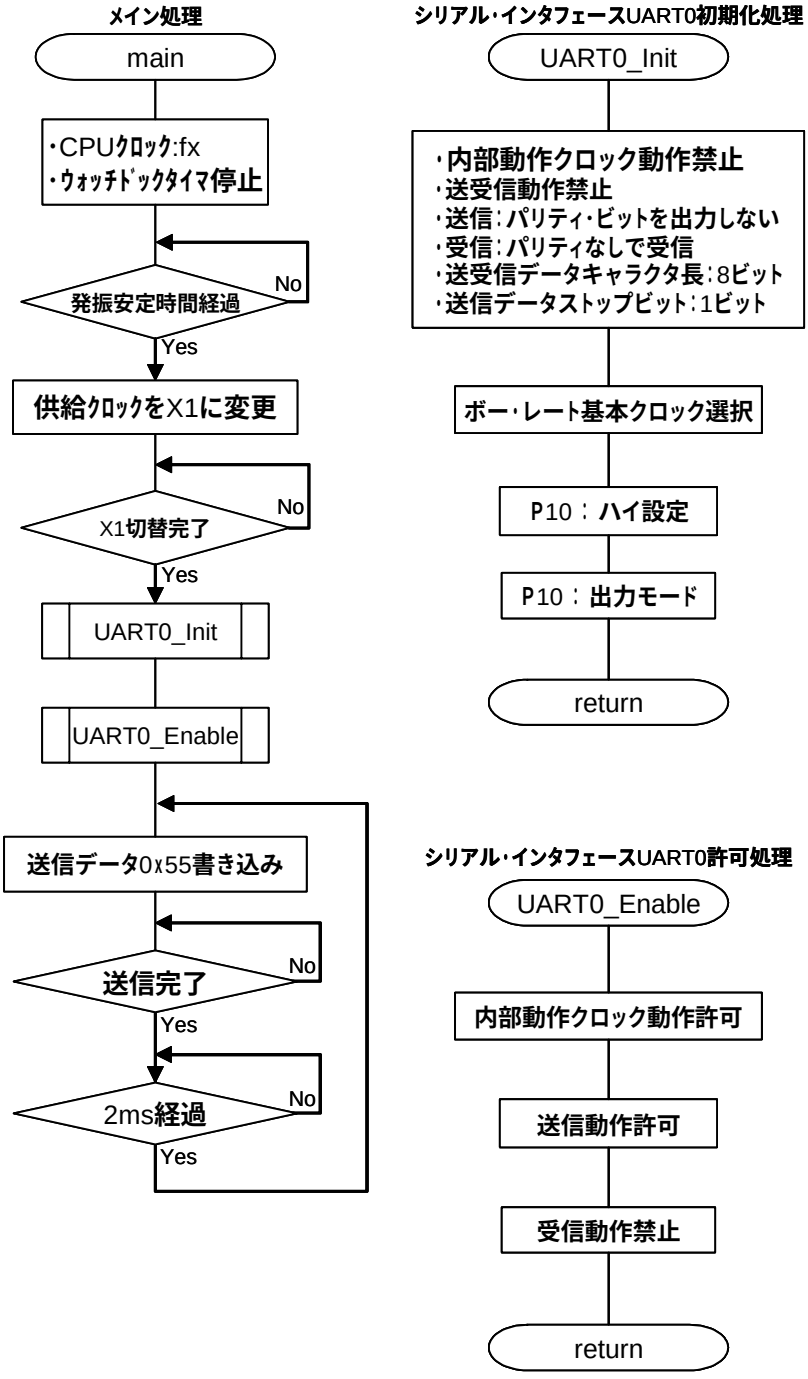
・動作クロックは，4本のクロック入力選択可能

・転送するデータの先頭ビットは，LSB固定

このプログラムは､9600bps の転送速度にて､2ms の間隔を空けて 0x55 のデータ送信を繰り返します。そしてメイン処理では２ms の時間をループ処理で実現し、経過毎に開始設定を行います。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

ボー・レート・ジェネレータ・コントロール・レジスタ 0(BRGC0) によりボー・レートを 9600 に設定します。送信フォーマットは、1bit のスタートビット、8bit のデータ、パリティビットなし、1bit のストップビットとなっています。送信データの登録は、送信シフトレジスタ(TXS0) にて行われます。メインループでは、TXS0 に 0x55 の値を書込むことで送信を繰り返し行っています。送信完了は、送信完了のリクエストフラグ(DUALIF0) にて確認します。そして送信完了後は、約 2ms の wait を設定しています。そのため、送信データは約 2ms 間隔で出力されます。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
No
発振安定時間経過
Yes
供給ｸﾛｯｸをX1に変更
No
X1切替完了
Yes
UART0_Init
UART0_Enable
送信データ0x55書き込み
No
送信完了
Yes
No
2ms経過
Yes
シリアル・インタフェースUART0初期化処理
UART0_Init
・内部動作クロック動作禁止
・送受信動作禁止
・送信：パリティ・ビットを出力しない
・受信：パリティなしで受信
・送受信データキャラクタ長：8ビット
・送信データストップビット：1ビット
ボー・レート基本クロック選択
P10：ハイ設定
P10：出力モード
return
シリアル・インタフェースUART0許可処理
UART0_Enable
内部動作クロック動作許可
送信動作許可
受信動作禁止
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: UART0 transmitting control
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - Serial Interface: UART0
*               - Serial data port: Port 1.0 (P10/TxD0)
*               - Baud rate: 9600
*               - Data format: 1 start bit, 8 data bits, no parity, 1 stop bits
*               - Data byte continuously transmitted: 55 hex
*               - Delay between each data byte transmitted: approx. 2ms
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

#define UART_BAUDRATE_M0    0x3
#define UART_BAUDRATE_K0    0x10

/*status list definition*/
#define TRUE            1
#define FALSE           0

void UART0_Init( void );
void UART0_Enable( void );

/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/

void main( void )
{
    unsigned short i;

    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    UART0_Init();                       /* UART0 initialization function */
    UART0_Enable();                     /* UART0 enable function */

    while(TRUE)                         /* main loop */
    {
        TXS0 = 0x55;
        /* Waiting for the completion of transmitting */
        while( DUALIF0 == 0 );
        DUALIF0 = 0;

        for( i=0 ; i<1000 ; i++ );      /* wait 2 ms */
```

```
    }
}


/*
-------------------------------------------------------------------------------
    Functions Group
-------------------------------------------------------------------------------
*/

/*
-------------------------------------------------------------------------
  Abstract:
    Initializes UART0 interface for each channel; the application is responsible for
    set work mode, transfer speed, data bit length, stop bit length, parity type  setting.
-------------------------------------------------------------------------
*/
void UART0_Init( void )
{
    ASIM0 = 0x05;                       /* Data length is 8 bits */
                                        /* Stop length is 1 bits */
                                        /* Parity mode is None parity */
    BRGC0 = UART_BAUDRATE_K0 |
          (UART_BAUDRATE_M0 << 6);      /* Baud rate selection */
    P1.0 = 1;                           /* P10=1 */
    PM1.0 = 0;                          /* Set UART0 work in transmit(output) mode */
}


/*
-------------------------------------------------------------------------
  Abstract:
    This function is responsible for supplying clock to UART0 interface.
-------------------------------------------------------------------------
*/
void UART0_Enable( void )
{
    POWER0 = 1;                         /* Permission of an internal operation clock of operation */
    TXE0 = 1;                           /* Transmitting operation permission */
    RXE0 = 0;                           /* Prohibition of reception operation */
}
```

■ アセンブリ言語ソース

```
;
;*******************************************************************************
;
;  Title: UART0 transmitting control
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - Serial Interface: UART0
;               - Serial data port: Port 1.0 (P10/TxD0)
;               - Baud rate: 9600
;               - Data format: 1 start bit, 8 data bits, no parity, 1 stop bits
;               - Data byte continuously transmitted: 55 hex
;               - Delay between each data byte transmitted: approx. 2ms
;
;*******************************************************************************



;-------------------------------------------------------------------------------
;       Macro Define
;-------------------------------------------------------------------------------


UART_BAUDRATE_M0        EQU     03H
UART_BAUDRATE_K0        EQU     10H


;-------------------------------------------------------------------------------
;       Specify Interrupt Vectors
;-------------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to  Start
        DW      start



;-------------------------------------------------------------------------------
;       Main Program
;-------------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CALL    !UART0_Init
        CALL    !UART0_Enable

main:                                   ; main loop
        MOV     TXS0, #55H
main01:
        BTCLR   DUALIF0, $main02
        BR      $main01
```

```
main02:

        MOVW    AX, #0000H
main03:
        CMPW    AX, #1000
        BNC     $main

        INCW    AX
        BR      $main03


;-------------------------------------------------------------------------------
;       Functions Group
;-------------------------------------------------------------------------------


;-------------------------------------------------------------------------
;  Abstract:
;       Initializes UART0 interface for each channel; the application is responsible for
;       set work mode, transfer speed, data bit length, stop bit length, parity type setting.
;-------------------------------------------------------------------------
UART0_Init:
        MOV     ASIM0, #05H             ; Data length is 8 bits
                                        ; Stop length is 1 bits
                                        ; Parity mode is None parity

        MOV     BRGC0, #UART_BAUDRATE_M0 SHL 6 OR UART_BAUDRATE_K0      ; Baud rate selection
        SET1    P1.0                    ; Set UART0 work in transmit mode
        CLR1    PM1.0

        RET


;-------------------------------------------------------------------------
;  Abstract:
;     This function is responsible for supplying clock to UART0 interface.
;-------------------------------------------------------------------------
UART0_Enable:
        SET1    POWER0                  ; Permission of an internal operation clock of operation
        SET1    TXE0                    ; Transmitting operation permission
        CLR1    RXE0                    ; Prohibition of reception operation

        RET

END
```

３．１１ シリアル･インタフェースＣＳＩ１０

■ ファイル名

Csi0

- CSI0_3wire.asm（アセンブリ言語ソース）
- CSI0_3wire.c（Ｃ言語ソース）

■ 説明

シリアル・インタフェースCSI10，CSI11[注]には，次の２種類のモードがあります。

（1）動作停止モード

シリアル転送を行わないときに使用するモードです。消費電力を低減することができます。

（2）３線式シリアルI/0モード（MSB/LSB先頭切り替え可能）

シリアル・クロック（SCK1n）とシリアル・データ（SI1n，SO1n）の３本のラインにより，８ビット・データ転送を行うモードです。

３線式シリアルI/0モードは同時送受信動作が可能なので，データ転送の処理時間が短くなります。

シリアル転送する８ビット･データの先頭ビットをMSBか，またはLSBかに切り替えることができますので，いずれの先頭ビットのデバイスとも接続できます。

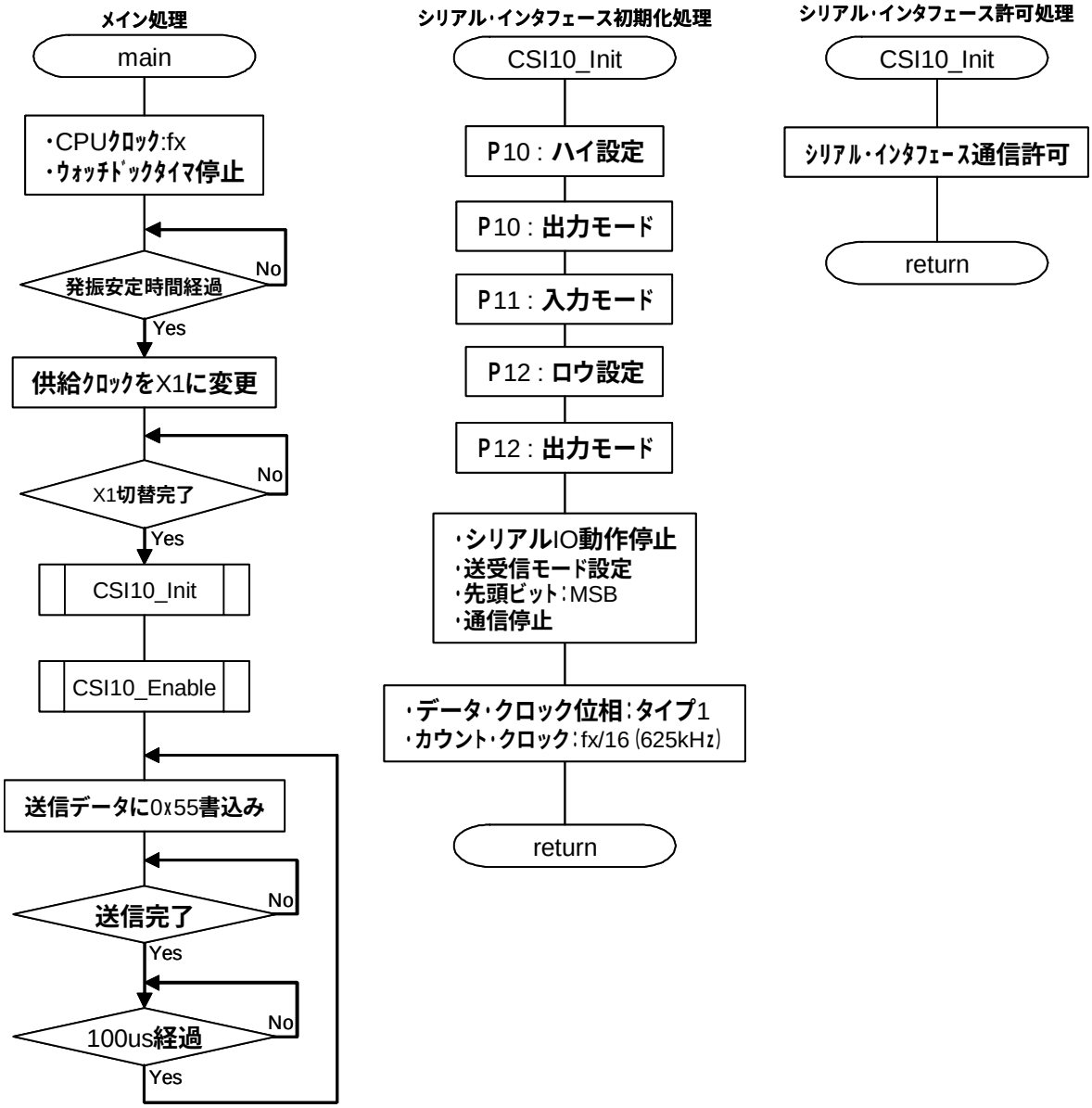
３線式シリアルI/0モードは，75XLシリーズ，78Kシリーズ，17Kシリーズなど従来のクロック同期式シリアル・インタフェースを内蔵する周辺I/0や表示コントローラなどを接続するときに有効です。

注 μPD780146，780148，78F0148のみ。

このプログラムは、625kHz のクロックにて、100us の間隔を空けて 0x55 のデータ送信を繰り返します。そして、メイン処理では 100us の時間をループ処理で実現し、経過毎に開始設定を行います。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz) を選択します。

通信速度は、シリアル・クロック選択レジスタ(CSIC10) により通信速度を 625kHz に設定します。送信データの登録は、送信バッファ・レジスタ(SOTB10) に書込むことで行われます。メインループでは、SOTB10 に 0x55 の値を設定し送信を繰り返し行っています。送信完了は、送信完了のリクエストフラグ(DUALIF0) にて確認しています。送信完了後は、約 100μs の wait を設定しています。そのため、送信データは約 100μs 間隔で出力されます。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
CSI10_Init
CSI10_Enable
送信データに0x55書込み
送信完了
No
Yes
100us経過
No
Yes
シリアル・インタフェース初期化処理
CSI10_Init
P10：ハイ設定
P10：出力モード
P11：入力モード
P12：ロウ設定
P12：出力モード
・シリアルIO動作停止
・送受信モード設定
・先頭ビット：MSB
・通信停止
・データ・クロック位相：タイプ1
・カウント・クロック：fx/16（625kHz）
return
シリアル・インタフェース許可処理
CSI10_Init
ｼﾘｱﾙ・ｲﾝﾀﾌｪｰｽ通信許可
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: CSI0 3wire control
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - Serial Interface: CSI10
*               - Serial clock port: Port 1.0 (P10/SCK10)
*               - Serial  data port: Port 1.2 (P12/SO10)
*               - Serial clock frequency: 625kHz (fx/8)
*               - Data byte continuously transmitted: 55 hex
*               - Delay between each data byte transmitted: approx. 100us
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI


/*
------------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------------
*/

/*status list definition*/
#define TRUE            1
#define FALSE           0

void CSI10_Init( void );
void CSI10_Enable( void );


/*
------------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------------
*/

void main( void )
{
    unsigned char i;

    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    CSI10_Init();                       /* CSI10 initialization function */
    CSI10_Enable();                     /* CSI10 enable function */

    while(TRUE)                         /* main loop */
    {
        SOTB10 = 0x55;
        while( DUALIF0 == 0 );
        DUALIF0 = 0;
        for( i=0 ; i<50 ; i++ );        /* wait 100us */
    }

}
```

```
/*
--------------------------------------------------------------------------------
    Functions Group
--------------------------------------------------------------------------------
*/

/*
--------------------------------------------------------------------------------
  Abstract:
    CSI10 interface initialization, the application is responsible for
    set work mode, transfer speed, data bit length, data direction setting,
    automatic transfer mode, clock and data phase setting.
--------------------------------------------------------------------------------
*/
void CSI10_Init( void )
{
    P1.0 = 1;                           /* P10=1 */
    PM1.0 = 0;                          /* set P10 is output mode(Internal clock output) */
    PM1.1 = 1;                          /* set P11 is input mode */
    P1.2 = 0;                           /* P12=0 */
    PM1.2 = 0;                          /* set P12 is output mode */

    CSIM10 = 0x40;                      /* CSI10 work in half-duplex mode */
                                        /* Set data direction is MSB */
    CSIC10 = 0x03;                      /* Clock data phase1 */
                                        /* fxx/16 */

}

/*
--------------------------------------------------------------------------------
  Abstract:
    This function is responsible for supplying clock to CSI10 interface.
--------------------------------------------------------------------------------
*/
void CSI10_Enable( void )
{
    CSIE10 = 1;                         /* Permission of operation */
}
```

■ アセンブリ言語ソース

```
;
;*******************************************************************************
;
;  Title: CSI0 3wire control
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - Serial Interface: CSI10
;               - Serial clock port: Port 1.0 (P10/SCK10)
;               - Serial  data port: Port 1.2 (P12/SO10)
;               - Serial clock frequency: 625kHz (fx/8)
;               - Data byte continuously transmitted: 55 hex
;               - Delay between each data byte transmitted: approx. 100us
;
;*******************************************************************************



;-------------------------------------------------------------------------
;       Specify Interrupt Vectors
;-------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to  Start
        DW      start



;-------------------------------------------------------------------------------
;       Main Program
;-------------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CALL    !CSI10_Init
        CALL    !CSI10_Enable

main:                                   ; main loop
        MOV     SOTB10, #55H
main01:
        BTCLR   DUALIF0, $main02
        BR      $main01
main02:

        MOV     A, #00H
main03:
        CMP     A, #50                  ; wait 100us
        BNC     $main

        INC     A
        BR      $main03
```

```
;-------------------------------------------------------------------------------
;       Functions Group
;-------------------------------------------------------------------------------


;-------------------------------------------------------------------------
;  Abstract:
;       CSI10 interface initialization, the application is responsible for
;       set work mode, transfer speed, data bit length, data direction setting,
;       automatic transfer mode, clock and data phase setting.
;-------------------------------------------------------------------------
CSI10_Init:

        SET1    P1.0
        CLR1    PM1.0
        SET1    PM1.1
        CLR1    P1.2
        CLR1    PM1.2

        SET1    CSIM10.6                ;CSI10 work in half-duplex mode
        CLR1    CSIM10.4                ;Set data direction is MSB
        CLR1    CSIC10.4                ;Clock data phase1
        CLR1    CSIC10.3
        CLR1    CSIC10.2                ;fxx/16
        SET1    CSIC10.1
        SET1    CSIC10.0

        RET


;-------------------------------------------------------------------------
;  Abstract:
;       This function is responsible for supplying clock to CSI10 interface.
;-------------------------------------------------------------------------
CSI10_Enable:
        SET1    CSIE10                  ; Permission of operation

        RET

END
```

３．１２ シリアル･インタフェースＣＳＩＡ０

■ ファイル名

Csia0

- CSIA0_3wire.asm（アセンブリ言語ソース）
- CSIA0_3wire.c（Ｃ言語ソース）

■ 説明

シリアル・インタフェースCSIA0には，次の３種類のモードがあります。

（1）動作停止モード

シリアル転送を行わないときに使用するモードです。消費電力を低減することができます。

（2）３線式シリアルI/Oモード（MSB/LSB先頭切り替え可能）

シリアル・クロック（SCKA0）とシリアル・データ（SIA0，SOA0）の３本のラインにより，８ビット・データ転送を行うモードです。

３線式シリアルI/Oモードは同時送受信動作が可能なので，データ転送の処理時間が短くなります。

シリアル転送する８ビット･データの先頭ビットをMSBか，またはLSBかに切り替えることができますので，いずれの先頭ビットのデバイスとも接続できます。

（3）自動送受信機能付き３線式シリアルI/Oモード（MSB/LSB先頭切り替え可能）

シリアル・クロック（SCKA0）とシリアル・データ（SIA0，SOA0）の３本のラインにより，８ビット・データ転送を行うモードです。

３線式シリアルI/Oモードは同時送受信動作が可能なので，データ転送の処理時間が短くなります。

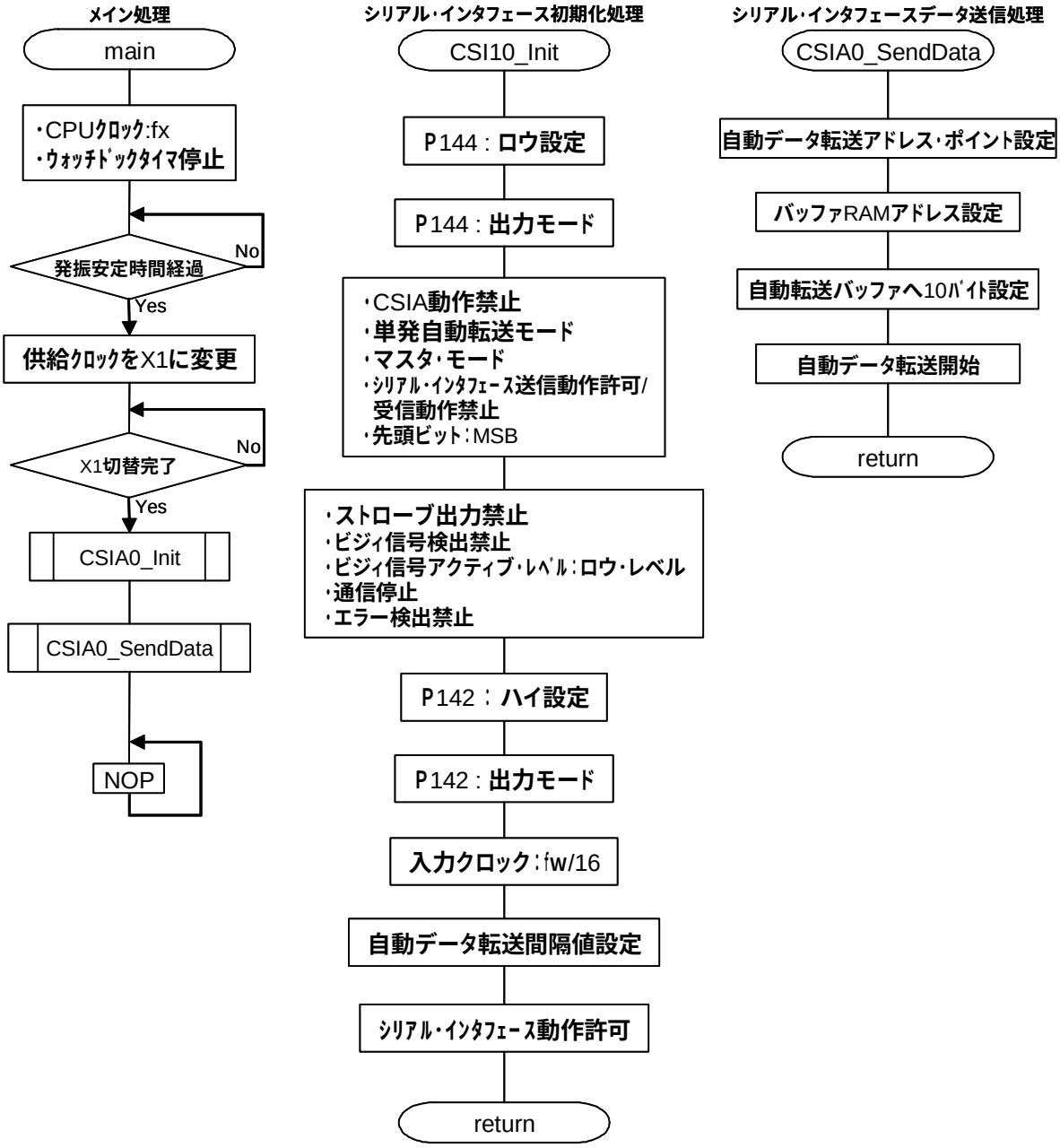
シリアル転送する８ビット･データの先頭ビットをMSBか，またはLSBかに切り替えることができますので，いずれの先頭ビットのデバイスとも接続できます。

転送バッファRAMを32バイト内蔵しているので，ソフトウエアを介さずに表示ドライバなどとデータ転送可能です。またハンドシェーク端子（STB0，BUSY0）をサポートしており，容易に周辺LSIと接続することができます。

このプログラムは、625kHz のクロックにて、自動送信機能を使用し 0x01～0x0A までの 10byte を送信します。そしてメイン処理では、出力設定後 無限ループにします。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

初期設定として、分周値選択レジスタ 0（BRGCA0）に通信速度として 625kHz を設定し、送信データは、0x01 から 0x0A までの 10byte をバッファ RAM に設定します。また、自動データ転送間隔指定レジスタ(ADTI0)により 3 クロック分のインターバル時間を設定します。その後、自動転送機能を起動させ送信します。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
No
発振安定時間経過
Yes
供給ｸﾛｯｸをX1に変更
No
X1切替完了
Yes
CSIA0_Init
CSIA0_SendData
NOP
シリアル・インタフェース初期化処理
CSI10_Init
P144：ロウ設定
P144：出力モード
・CSIA動作禁止
・単発自動転送モード
・マスタ・モード
・ｼﾘｱﾙ・ｲﾝﾀﾌｪｰｽ送信動作許可/受信動作禁止
・先頭ビット：MSB
・ストローブ出力禁止
・ビジィ信号検出禁止
・ビジィ信号アクティブ・ﾚﾍﾞﾙ：ロウ・レベル
・通信停止
・エラー検出禁止
P142：ハイ設定
P142：出力モード
入力クロック：fw/16
自動データ転送間隔値設定
ｼﾘｱﾙ・ｲﾝﾀﾌｪｰｽ動作許可
return
シリアル・インタフェースデータ送信処理
CSIA0_SendData
自動データ転送アドレス・ポイント設定
バッファRAMアドレス設定
自動転送バッファへ10ﾊﾞｲﾄ設定
自動データ転送開始
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: CSIA0 3wire control
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - Serial Interface: CSIA0
*               - Serial clock port: Port 14.2 (P142/SCKA0)
*               - Serial  data port: Port 14.4 (P144/SOA0)
*               - Serial clock frequency: 625kHz (fx/16)
*               - Data byte continuously transmitted: 01 - 0A HEX
*               - Delay between each data byte transmitted: approx. 3 clock
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

#define ADR_CSIA0B0     0xfa00      /*CSIA0 automatic transfer RAM address*/
#define CSIA_AUTORAMSIZE    32      /*CSIA automatic transfer RAM size*/
#define CSIA0_INTERVALVALUE 0x3

/*status list definition*/
#define TRUE            1
#define FALSE           0


const   unsigned char ucSendData[10] =
{
     0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x0A
};

void CSIA0_Init( void );
void CSIA0_SendData(unsigned char* txbuf, unsigned char txnum);


/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/

void main( void )
{

    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    CSIA0_Init();                       /* CSIA0 initialization function */
    CSIA0_SendData( &ucSendData[0] , 10 );  /* CSIA0 transmitting function */

    while(TRUE)                         /* Endless loop */
```

```
    {
        NOP();
    }

}


/*
-------------------------------------------------------------------------------
    Functions Group
-------------------------------------------------------------------------------
*/

/*
-------------------------------------------------------------------------
  Abstract:
    CSIA0 interface initialization, the application is responsible for set
    work mode, transfer speed, automatic transfer mode , data direction
    setting, automatic transfer byte number,internale time setting.
-------------------------------------------------------------------------
*/
void CSIA0_Init( void )
{
    P14.4 = 0;                          /* P144=0 */
    PM14.4 = 0;                         /* set P144 is output mode */

    CSIMA0 = 0x58;                      /* CSIA0 work in transmit mode */
                                        /* Internal clock(master mode) */
                                        /* CSIA0 work in automatic transfer mode */
                                        /* CSIA0 work in single transfer mode */
                                        /* The data direction is MSB */

    CSIS0 = 0x00;                       /* Strobe output Disabled */
                                        /* Busy signal detection Disabled */
                                        /* Busy signal active level setting Low level */
                                        /* Bit error detection disabled */

    P14.2 = 1;                          /* P142=1 */
    PM14.2 = 0;                         /* set P142 is output mode */

    BRGCA0 = 0x02;                      /* fxx/16 */

    ADTI0 = CSIA0_INTERVALVALUE;        /* master mode insert interval */
    CSIAE0 = 1;                         /* CSIA0 operation permission */
}


/*
-------------------------------------------------------------------------
  Abstract:
    This function responsible for transferring data for CSIA0 interface and
    a call back function is provided to high level user.
-------------------------------------------------------------------------
*/
void    CSIA0_SendData(unsigned char* txbuf, unsigned char txnum)
{
    unsigned char i;
    unsigned char * p_buf;

    ADTP0 = txnum - 1;                  /* setup of transmitting bytes */

    p_buf = (unsigned char*)ADR_CSIA0B0;/* CSIA0 buffer RAM address */
    for (i=0; i<txnum; i++){
        *p_buf++ = *txbuf++;
    }
    ATSTA0 = 1;                         /* Normal mode */
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: CSIA0 3wire control
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - Serial Interface: CSIA0
;               - Serial clock port: Port 14.2 (P142/SCKA0)
;               - Serial  data port: Port 14.4 (P144/SOA0)
;               - Serial clock frequency: 625kHz (fx/16)
;               - Data byte continuously transmitted: 01 - 0A hex
;               - Delay between each data byte transmitted: approx. 3 clock
;
;*****************************************************************************


;-----------------------------------------------------------------------------
;       Macro Define
;-----------------------------------------------------------------------------


ADR_CSIA0B0             EQU     0FA00H  ; CSIA0 automatic transfer RAM address
CSIA_AUTORAMSIZE        EQU     32D     ; CSIA automatic transfer RAM size
CSIA0_INTERVALVALUE     EQU     03H


                CSEG
ucSendData:             DB      01H
                        DB      02H
                        DB      03H
                        DB      04H
                        DB      05H
                        DB      06H
                        DB      07H
                        DB      08H
                        DB      09H
                        DB      0AH

                DSEG
CSIA0_TX_ADDRESS:       DS      2

;-----------------------------------------------------------------------------
;       Specify Interrupt Vectors
;-----------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to  Start
        DW      start



;-----------------------------------------------------------------------------
;       Main Program
;-----------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
```

```
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CALL    !CSIA0_Init

        MOVW    DE, #ucSendData
        MOV     B, #10
        CALL    !CSIA0_SendData

main:                                   ; Endless loop
        NOP
        BR      $main


;---------------------------------------------------------------------------
;       Functions Group
;---------------------------------------------------------------------------


;---------------------------------------------------------------------------
;  Abstract:
;       CSIA0 interface initialization, the application is responsible for set
;       work mode, transfer speed, automatic transfer mode , data direction
;       setting, automatic transfer byte number,internale time setting.
;---------------------------------------------------------------------------
CSIA0_Init:

        CLR1    P14.4
        CLR1    PM14.4

        MOV     CSIMA0, #58H            ; CSIA0 work in transmit mode
                                        ; Internal clock(master mode)
                                        ; CSIA0 work in automatic transfer mode
                                        ; CSIA0 work in single transfer mode
                                        ; The data direction is MSB

        MOV     CSIS0, #00H             ; Strobe output Disabled
                                        ; Busy signal detection Disabled
                                        ; Busy signal active level setting Low level

        SET1    P14.2
        CLR1    PM14.2

        MOV     BRGCA0, #02H            ; fw/16
        MOV     ADTI0, #CSIA0_INTERVALVALUE     ; master mode insert interval

        SET1    CSIAE0                  ; CSIA0 operation permission

        RET


;---------------------------------------------------------------------------
;  Abstract:
;       This function responsible for transferring data for CSIA0 interface and
;       a call back function is provided to high level user.
;---------------------------------------------------------------------------
CSIA0_SendData:

        MOV     A, B
        DEC     A
        MOV     ADTP0, A

        MOVW    HL, #ADR_CSIA0B0        ; CSIA0 buffer RAM address
```

```
CSIA0_SetData1:
        MOV     A, [DE]
        MOV     [HL], A
        INCW    DE
        INCW    HL
        DBNZ    B, $CSIA0_SetData1

        SET1    ATSTA0

        RET

END
```

３．１３ 乗算器

■ ファイル名

- Hard_Mult.asm（アセンブリ言語ソース）
- Hard_Mult.c（C言語ソース）

■ 説明

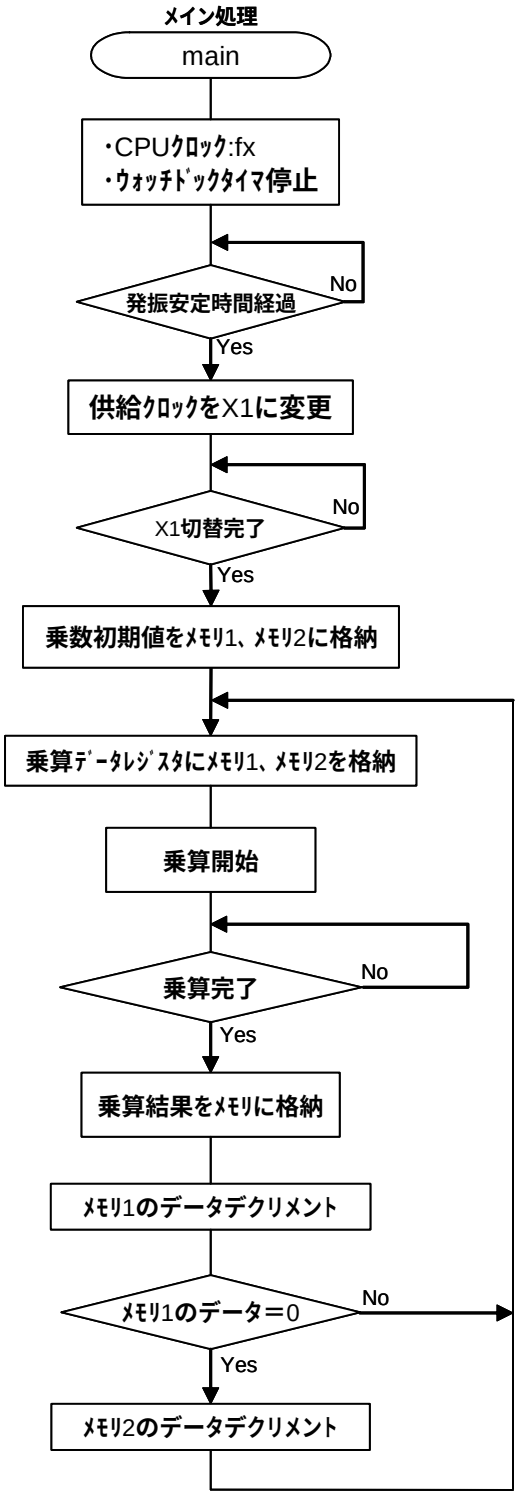
乗除算器には，次のような機能を持ちます。

・16ビット×16ビット= 32ビット（乗算）
・32ビット÷16ビット= 32ビット　剰余16ビット（除算）

このプログラムは、0～255 までの２つの値の乗算を繰り返し RAM に格納します。そしてメイン処理は、出力設定後無限ループにします。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ（10MHz）を選択します。

メインループ内にて、８ビット×８ビットの掛け算を繰り返します。計算する値を、MDA0L と MDB0 に代入した後、DMUC0.7 をセットします。 計算が終了すると DMUE がセットされるので、MDA0L より計算結果を読み込みます。計算の順序は、255×255=65025 から始まり、254×255=64770、253×255=64515、……、200×100=20000、199×100=19900、……、100×100=10000、0×9=0、255×9=2295、……、1×1=1、0×0=0、255×0=0……、0×255=0 の順番で、これを繰り返します。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
No
発振安定時間経過
Yes
供給ｸﾛｯｸをX1に変更
No
X1切替完了
Yes
乗数初期値をﾒﾓﾘ1、ﾒﾓﾘ2に格納
乗算ﾃﾞｰﾀﾚｼﾞｽﾀにﾒﾓﾘ1、ﾒﾓﾘ2を格納
乗算開始
No
乗算完了
Yes
乗算結果をﾒﾓﾘに格納
ﾒﾓﾘ1のデータデクリメント
No
ﾒﾓﾘ1のデータ＝0
Yes
ﾒﾓﾘ2のデータデクリメント

■ C言語ソース

```
/*
************************************************************************
*
*  Title: Hardware multiplier
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - multiplier control and status register: DMUC0
*               - 16-bit multiplicand registers: MDA0 and MDB0
*               - 16-bit multiplication result register: MDA0
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

/*status list definition*/
#define TRUE            1
#define FALSE           0

/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/

void main( void )
{
    unsigned char   ucMult1,ucMult2;
    unsigned short  usResult;

    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    /* Initial multiplier value set */
    ucMult1 = 255;
    ucMult2 = 255;

    while(TRUE)                         /* main loop */
    {
        MDA0L = ucMult1;                /* Move 1st multiplicand into MDA0 */
        MDB0  = ucMult2;                /* Move 2nd multiplicand into MDB0 */

        DMUC0 = 0x81;                   /* Start multiply operation */
        while( DMUE == 1 )              /* Wait until operation complete */
        {
            NOP();
        }
        usResult = MDA0L;               /* Store 16-bit result */
        NOP();                          /* Break here to examine result */
        if(--ucMult1 == 0) ucMult2--;   /* Cycle through multiplicand values */
    };
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: Hardware multiplier
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - multiplier control and status register: DMUC0
;               - 16-bit multiplicand registers: MDA0 and MDB0
;               - 16-bit multiplication result register: MDA0
;
;*****************************************************************************


;---------------------------------------------------------------------------
;       Specify Interrupt Vectors
;---------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to  Start
                DW      start


;-----------------------------------------------------------------------------
;       Main Program
;-----------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        MOV     B, #255
        MOV     C, #255

main:                                   ; main loop
        MOV     X, #00H
        MOV     A, B
        XCH     A, X
        MOVW    MDA0L, AX               ;Move 1st multiplicand into MDA0
        MOV     X, #00H
        MOV     A, C
        XCH     A, X
        MOVW    MDB0, AX                ;Move 2nd multiplicand into MDB0

        MOV     DMUC0, #81H             ;Start multiply operation
main01:
        BF      DMUE, $main02           ;Wait until operation complete
        NOP
        BR      $main01

main02:
        MOVW    AX, MDA0L
        MOVW    DE, AX                  ; Store 16-bit result
```

```
        DEC     B
        BNZ     $main03
        DEC     C

main03:
        BR      $main

END
```

３．１４ 割込み機能

■ ファイル名

- Interrupt.asm（アセンブリ言語ソース）
- Interrupt.c（C言語ソース）

■ 説明

割込み機能には，次の２種類があります。

（1）マスカブル割り込み

マスク制御を受ける割込みです。優先順位指定フラグ・レジスタ（PR0L，PR0H，PR1L，PR1H）の設定により，割込み優先順位を高い優先順位のグループと低い優先順位のグループに分けることができます。

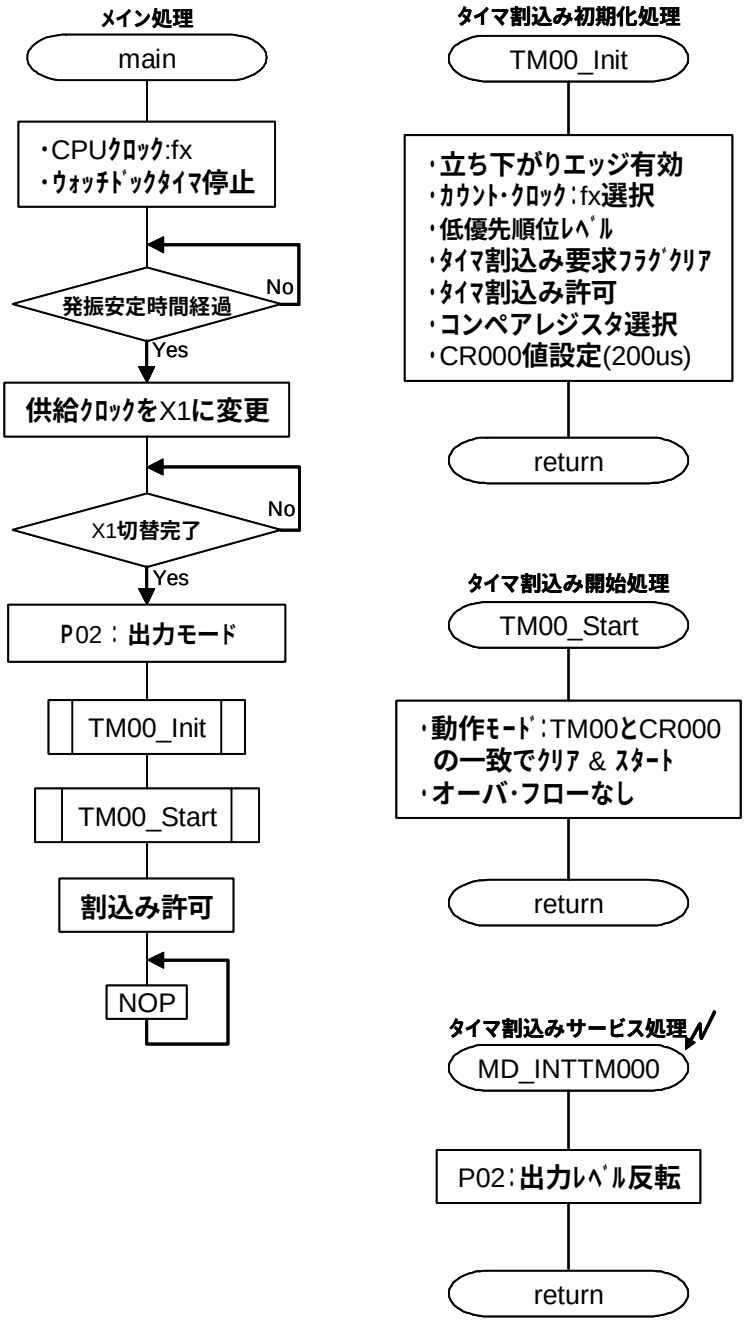
（2）ソフトウエア割り込み

BRK命令の実行によって発生するベクタ割込みです。割込み禁止状態でも受け付けられます。また，割込み優先順位制御の対象になりません。

このプログラムは、タイマ 00 を使用して 200us 毎の割込み処理内で、P02 を反転出力させます。そしてメイン処理は、出力設定後無限ループにします。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

16 ビット・タイマ／イベント・カウンタ 00 を使用して、タイマ割込みを発生させます。カウントクロックは 10MHz を選択します。16 ビット・タイマ・キャプチャ／コンペアレジスタ 000(CR000）は 1999 に設定されています。タイマ動作開始後タイマカウンタがコンペアレジスタの設定時間(1999)と一致した時、タイマ割込みが発生します。割込み処理内では P02 の出力レベルを反転させます。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
No
発振安定時間経過
Yes
供給ｸﾛｯｸをX1に変更
No
X1切替完了
Yes
P02：出力モード
TM00_Init
TM00_Start
割込み許可
NOP
タイマ割込み初期化処理
TM00_Init
・立ち下がりエッジ有効
・ｶｳﾝﾄ･ｸﾛｯｸ：fx選択
・低優先順位ﾚﾍﾞﾙ
・ﾀｲﾏ割込み要求ﾌﾗｸﾞｸﾘｱ
・ﾀｲﾏ割込み許可
・コンペアレジスタ選択
・CR000値設定(200us)
return
タイマ割込み開始処理
TM00_Start
・動作ﾓｰﾄﾞ：TM00とCR000
の一致でｸﾘｱ & ｽﾀｰﾄ
・オーバ･フローなし
return
タイマ割込みサービス処理
MD_INTTM000
P02：出力ﾚﾍﾞﾙ反転
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: Interrupt
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - timer: 16-bit timer/event counter TM00
*               - timer count clock: fx (10MHz)
*               - compare register value (CR000): 0x7cf
*               - interval time: 200us
*               - interrupt: INTTM0
*               - output port: port 0.2 (toggles every 200us)
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI

/*
------------------------------------------------------------------------------
    Specify Interrupt Vectors
------------------------------------------------------------------------------
*/
#pragma interrupt    INTTM000        MD_INTTM000


/*
------------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------------
*/

#define TM_TM00_INTERVALVALUE   0x7cf

/*status list definition*/
#define TRUE            1
#define FALSE           0

void TM00_Init( void );
void TM00_Start( void );


/*
------------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------------
*/

void main( void )
{

    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    PM0.2 = 0;                          /* set P02 is output mode */

    TM00_Init();                        /* TM00 initialization function */
    TM00_Start();                       /* TM00 start function */
```

```
    EI();                               /* Master interruption permission */

    while(TRUE)                         /* Endless loop */
    {
        NOP();
    }
}


/*
-------------------------------------------------------------------------------
    Functions Group
-------------------------------------------------------------------------------
*/

/*
-------------------------------------------------------------------------------
  Abstract:
    Initiate TM00, select founction and input parameter
    count clock selection, INT init
-------------------------------------------------------------------------------
*/
void TM00_Init( void )
{
    PRM00 = 0x00;                       /* internal count clock */
    /*timer00 interval*/

    TMPR000 = 1;                        /* low priority level */
    TMIF000 = 0;                        /* request flag clear */
    TMMK000 = 0;                        /* INTTM000 enable */

    CRC00 = 0x00;                       /* Operates as compare register */
                                        /* Captures on valid edge of TI010 */
                                        /* Operates as compare register */
    CR000 = TM_TM00_INTERVALVALUE;
}

/*
-------------------------------------------------------------------------------
  Abstract:
    Start the timer00 counter
-------------------------------------------------------------------------------
*/
void TM00_Start( void )
{
    TMC00 = 0x0c;                       /* Clear & start occurs on match between TM00 and CR000 */
}


/*
-------------------------------------------------------------------------------
  Abstract:
    INTTM00 interrupt service routine.
-------------------------------------------------------------------------------
*/
__interrupt void MD_INTTM000( void )
{
    P0.2 ^= 1;                          /* Output level reversal */
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: Interrupt
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - timer: 16-bit timer/event counter TM00
;               - timer count clock: fx (10MHz)
;               - compare register value (CR000): 0x7cf
;               - interval time: 200us
;               - interrupt: INTTM0
;               - output port: port 0.2 (toggles every 200us)
;
;*****************************************************************************


;---------------------------------------------------------------------------
;       Macro Define
;---------------------------------------------------------------------------

TM_TM00_INTERVALVALUE   EQU     07CFH

;---------------------------------------------------------------------------
;       Specify Interrupt Vectors
;---------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to  Start
                DW      start
INTTM000_VECTOR CSEG AT 0020h
                DW      MD_INTTM000


;---------------------------------------------------------------------------
;       Main Program
;---------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CLR1    PM0.2
        CALL    !TM00_Init
        CALL    !TM00_Start
        EI

main:                                   ; Endless loop
        NOP
        BR      $main


;---------------------------------------------------------------------------
```

```
;       Functions Group
;-------------------------------------------------------------------------------


;-------------------------------------------------------------------------
;  Abstract:
;       Initiate TM00, select founction and input parameter
;       count clock selection, INT init
;-------------------------------------------------------------------------
TM00_Init:

        MOV     PRM00, #00

        ; timer00 interval

        SET1    TMPR000
        CLR1    TMIF000
        CLR1    TMMK000

        MOV     CRC00, #00
        MOVW    CR000, #TM_TM00_INTERVALVALUE

        RET


;-------------------------------------------------------------------------
;  Abstract:
;       Start the timer00 counter
;-------------------------------------------------------------------------
TM00_Start:
        MOV     TMC00, #0CH

        RET


;-------------------------------------------------------------------------
;  Abstract:
;  INTTM00 interrupts service routine.
;-------------------------------------------------------------------------
MD_INTTM000:

        MOV1    CY, P0.2
        NOT1    CY
        MOV1    P0.2, CY

        RETI

END
```

３．１５ キー割込み機能

■ ファイル名

- KEY_Return.asm（アセンブリ言語ソース）
- KEY_Return.c（C言語ソース）

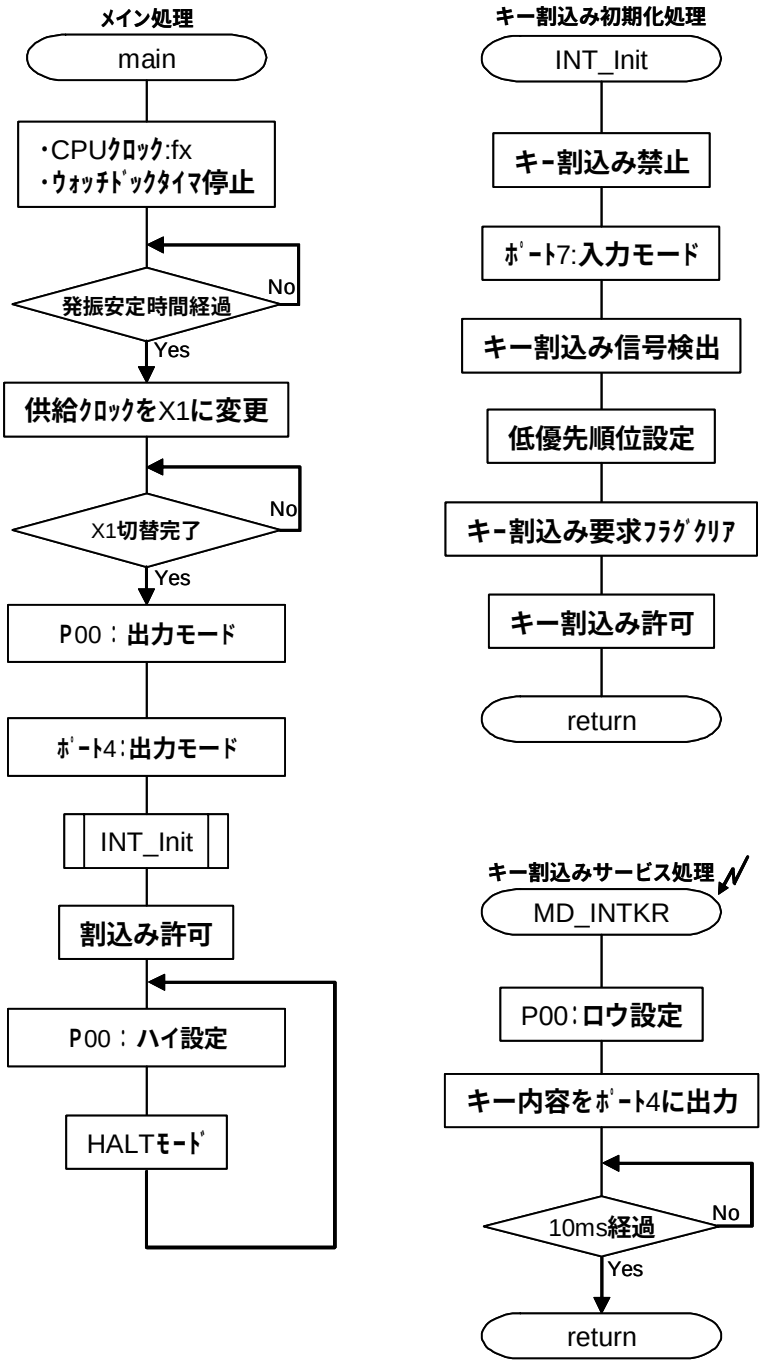
■ 説明

キー・リターン・モード・レジスタ（KRM）の設定により，キー割込み入力端子（KR0-KR7）に立ち下がりエッジを入力することによって，キー割込み（INTKR）を発生させることができます。

このプログラムは、P70/KR0～P77/KR7 の 8 本をキーリターン入力用に設定し、HALT モードの解除として用います。キーリターンを検出とたら、HALT を解除し割込み処理内で P00 を 10ms 間 Low にします。またキーの内容を P40～P47 に出力します。時間経過後メイン処理に復帰して P00 を High にし HALT モードに再び遷移させます。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ（10MHz）を選択します。

初期設定として、キー割込み端子として P70/KR0～P77/KR7 の 8 本を指定します。またキーの入力には内蔵プルアップ抵抗オプションを使用します。メインループでは、P00 を High にしてから HALT モードに遷移するように設定されており、HALT モードの解除は、P70/KR0～P77/KR7 の KEY 入力を使用しています。INTKR の割込み処理内は P00 を Low に設定しポート 7 の 8 ビットの入力データをポート 4 より出力させた後、約 10ms の wait を行っています。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
P00：出力モード
ﾎﾟｰﾄ4:出力モード
INT_Init
割込み許可
P00：ハイ設定
HALTﾓｰﾄﾞ
キー割込み初期化処理
INT_Init
キー割込み禁止
ﾎﾟｰﾄ7:入力モード
キー割込み信号検出
低優先順位設定
キー割込み要求ﾌﾗｸﾞｸﾘｱ
キー割込み許可
return
キー割込みサービス処理
MD_INTKR
P00:ロウ設定
キー内容をﾎﾟｰﾄ4に出力
10ms経過
No
Yes
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: Key interruption
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - external interrupt: INTKR (falling edge)
*                 input port: port7.0 (P70/KR0)
*                 input port: port7.1 (P71/KR1)
*                 input port: port7.2 (P72/KR2)
*                 input port: port7.3 (P73/KR3)
*                 input port: port7.4 (P74/KR4)
*                 input port: port7.5 (P75/KR5)
*                 input port: port7.6 (P76/KR6)
*                 input port: port7.7 (P77/KR7)
*               - interrupt service routine delay: approximately 10ms
*               - output port: port0.0 (P00/TI000)
*               - output port: port4 (8bit)
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI
#pragma HALT

/*
------------------------------------------------------------------------
    Specify Interrupt Vectors
------------------------------------------------------------------------
*/
#pragma interrupt   INTKR   MD_INTKR


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

/*status list definition*/
#define TRUE            1
#define FALSE           0

void INT_Init( void );

/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/

void main( void )
{

    PCC = 0x00;                         /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    PM0.0 = 0;                          /* set P00 output mode */
    PM4 = 0;                            /* set P4 output mode */
```

```
    INT_Init();                         /* INT initialization function */
    EI();                               /* Master interruption permission */

    while(TRUE)                         /* main loop */
    {
        P0.0 = 1;                       /* P00=1 */
        NOP();
        HALT();                         /* HALT mode */
        NOP();
    }
}


/*
-------------------------------------------------------------------------------
    Functions Group
-------------------------------------------------------------------------------
*/

/*
-------------------------------------------------------------------------------
  Abstract:
    Init the external INT, including enable or disable, priority setting.
-------------------------------------------------------------------------------
*/
void  INT_Init( void )
{
    KRMK = 1;                           /* disable INTKR */

    PU7 = 0x0FF;                        /* P7 pull up */
    PM7 = 0x0FF;                        /* P7 input mode */

    KRM = 0x0FF;

    KRPR = 1;                           /* priority :Low */
    KRIF = 0;                           /* clear interrupt flag */
    KRMK = 0;                           /* enable INTKR */
}


/*
-------------------------------------------------------------------------------
  Abstract:
    INTKR Interrupt service routine.
-------------------------------------------------------------------------------
*/
__interrupt void MD_INTKR( void )
{
    unsigned short  i;

    P0.0 = 0;                           /* P00=0 */
    P4 = P7;
    for(i=0 ; i<4600 ; i++);            /* wait 10 ms */
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: Key interruption
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - external interrupt: INTKR (falling edge)
;                 input port: port7.0 (P70/KR0)
;                 input port: port7.1 (P71/KR1)
;                 input port: port7.2 (P72/KR2)
;                 input port: port7.3 (P73/KR3)
;                 input port: port7.4 (P74/KR4)
;                 input port: port7.5 (P75/KR5)
;                 input port: port7.6 (P76/KR6)
;                 input port: port7.7 (P77/KR7)
;               - interrupt service routine delay: approximately 10ms
;               - output port: port0.0 (P00/TI000)
;               - output port: port4 (8bit)
;
;*****************************************************************************


;-----------------------------------------------------------------------------
;       Specify Interrupt Vectors
;-----------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to  Start
                DW      start
INTKR_VECTOR    CSEG AT 002ch
                DW      MD_INTKR


;-------------------------------------------------------------------------------
;       Main Program
;-------------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CLR1    PM0.0
        MOV     PM4, #00H
        CALL    !INT_Init
        EI                              ; Enable interrupts

main:                                   ; main loop
        SET1    P0.0
        NOP
        HALT
        NOP
        BR      $main
```

```
;-------------------------------------------------------------------------------
;       Functions Group
;-------------------------------------------------------------------------------


;-------------------------------------------------------------------------
;  Abstract:
;       Init the external INT, including enable or disable, priority setting.
;-------------------------------------------------------------------------
INT_Init:
        SET1    KRMK                    ; disable INTKR
        MOV     PU7, #0FFH              ; P7 pull up
        MOV     PM7, #0FFH
        MOV     KRM, #0FFH

        SET1    KRPR
        CLR1    KRIF
        CLR1    KRMK                    ; enable INTKR

        RET


;-------------------------------------------------------------------------
;  Abstract:
;       INTKR Interrupt service routine.
;-------------------------------------------------------------------------
MD_INTKR:
        SEL     RB1

        CLR1    P0.0
        MOV     A, P7
        MOV     P4, A

        MOVW    AX, #0000H
MD_INTKR_01:
        CMPW    AX, #4600               ; wait 10ms
        BNC     $MD_INTKR_02
        INCW    AX
        BR      $MD_INTKR_01
MD_INTKR_02:

        RETI

END
```

３．１６ スタンバイ機能

3.16.1 ＨＡＬＴ－タイマ割込みによる復帰

■ ファイル名

HALT_timer

- HALT_timer.asm（アセンブリ言語ソース）
- HALT_timer.c（Ｃ言語ソース）

■ 説明

スタンバイ機能は，システムの消費電力をより低減するための機能で，次の2種類のモードがあります。

（1）HALTモード

HALTモードは，CPUの動作クロックを停止させるモードです。システム・クロック発振回路の発振は継続します。このモードでは，STOPモードほどの消費電流の低減はできませんが，割込み要求により，すぐに処理を再開したい場合や，間欠動作をさせたい場合に有効です。

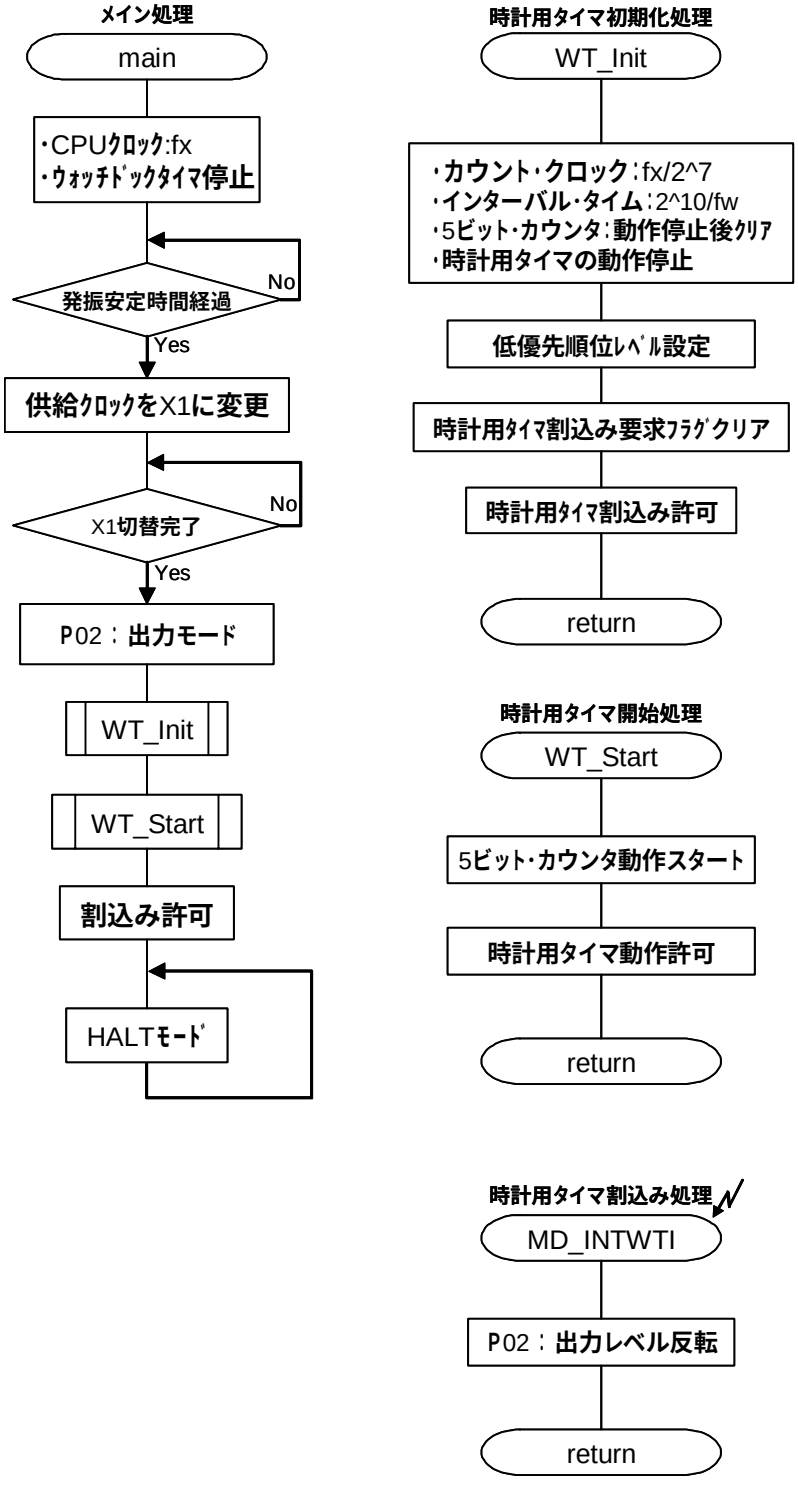
（2）STOPモード

STOPモードは，X1入力クロック発振回路を停止させ，システム全体が停止するモードです。CPUの消費電流を，かなり低減することができます。さらに，割込み要求によって解除できるため，間欠動作も可能です。ただし，STOPモード解除時に発振安定時間確保のためのウエイト時間がとられるため，割込み要求によって，すぐに処理を開始しなければならない場合にはHALTモードを選択してください。

このプログラムは、13ms 毎に HALT から復帰し、P02 を反転出力させます。（時計タイマのインターバルモードを使用）CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

初期設定で 13.1ms 毎に時計タイマ割込みを発生させるように、メインクロックを使用して時計タイマを設定します。メインループでは、HALT モードへの遷移が行われ、時計タイマ割込みにより解除します。割込み処理内にて P02 の出力レベルを反転させます。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
P02：出力モード
WT_Init
WT_Start
割込み許可
HALTﾓｰﾄﾞ
時計用タイマ初期化処理
WT_Init
・カウント・クロック：fx/2^7
・インターバル・タイム：2^10/fw
・5ビット・カウンタ：動作停止後ｸﾘｱ
・時計用タイマの動作停止
低優先順位ﾚﾍﾞﾙ設定
時計用ﾀｲﾏ割込み要求ﾌﾗｸﾞｸﾘｱ
時計用ﾀｲﾏ割込み許可
return
時計用タイマ開始処理
WT_Start
5ビット・カウンタ動作スタート
時計用タイマ動作許可
return
時計用タイマ割込み処理
MD_INTWTI
P02：出力レベル反転
return

■ C言語ソース

```
/*
********************************************************************************
*
*  Title: The return from HALT by timer interruption
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - watch timer count clock: main clock
*               - watch timer overflow time: 13.1ms
*               - output port: port 0.2 (toggles every 13.1ms)
*
********************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma HALT
#pragma EI

/*
--------------------------------------------------------------------------------
    Specify Interrupt Vectors
--------------------------------------------------------------------------------
*/
#pragma interrupt   INTWTI      MD_INTWTI


/*
--------------------------------------------------------------------------------
    Constants/Variables
--------------------------------------------------------------------------------
*/

/*status list definition*/
#define TRUE            1
#define FALSE           0

void WT_Init( void );
void WT_Start( void );


/*
--------------------------------------------------------------------------------
    Main Program
--------------------------------------------------------------------------------
*/

void main( void )
{

    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    PM0.2 = 0;                          /* set P02 is output mode */

    WT_Init();                          /* WT initialization function */
    WT_Start();                         /* WT start function */

    EI();                               /* Master interruption permission */

    while(TRUE)                         /* main loop */
    {
```

```
        NOP();
        HALT();                         /* HALT mode */
        NOP();
    }
}


/*
--------------------------------------------------------------------------------
    Functions Group
--------------------------------------------------------------------------------
*/

/*
--------------------------------------------------------------------------
  Abstract:
    Initiate the watch timer, select its working mode, count clock,
    enable/disable interrupt etc through Configuration.
--------------------------------------------------------------------------
*/
void WT_Init( void )
{
    WTM = 0x60;                         /* Disable Watch timer before change the setting */
                                        /* Set watch timer clock:fw=FX/2^7 */
                                        /* watch timer flag setting:2^14/fw(0.5s) */
                                        /* Interval timer prescaler setting:2^10/fw */

    WTIPR = 1;                          /* priority : Low */
    WTIIF = 0;                          /* request flag clear */
    WTIMK = 0;                          /* INTWTI enable */

}


/*
--------------------------------------------------------------------------
  Abstract:
    Restart the watch timer after stopping.
--------------------------------------------------------------------------
*/
void WT_Start( void )
{
    /*enable watch timer operation*/
    /*start the 5-bit counter*/
    WTM.1 = 1;
    WTM.0 = 1;
}

/*
--------------------------------------------------------------------------
  Abstract:
    INTTM00 interrupt service routine.
--------------------------------------------------------------------------
*/
__interrupt void MD_INTWTI( void )
{
    P0.2 ^= 1;                          /* Output level reversal */
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: The return from HALT by timer interruption
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - watch timer count clock: main clock
;               - watch timer overflow time: 13.1ms
;               - output port: port 0.2 (toggles every 13.1ms)
;
;*****************************************************************************


;---------------------------------------------------------------------------
;       Specify Interrupt Vectors
;---------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to  Start
                DW      start
INTWTI_VECTOR   CSEG AT 0028h
                DW      MD_INTWTI


;---------------------------------------------------------------------------
;       Main Program
;---------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CLR1    PM0.2
        CALL    !WT_Init
        CALL    !WT_Start
        EI

main:                                   ; main loop
        NOP
        HALT
        NOP
        BR      $main


;---------------------------------------------------------------------------
;       Functions Group
;---------------------------------------------------------------------------


;---------------------------------------------------------------------
;  Abstract:
```

```
;       Initiate the watch timer, select its working mode, count clock,
;       enable/disable interrupt etc through Configuration.
;-------------------------------------------------------------------------
WT_Init:

        MOV     WTM, #60H               ; Disable Watch timer before change the setting
                                        ; Set watch timer clock:fw=FX/2^7
                                        ; watch timer flag setting:2^14/fw(0.5s)
                                        ; Interval timer prescaler setting:2^10/fw

        SET1    WTIPR                   ; priority : Low
        CLR1    WTIIF                   ; INTWTI enable
        CLR1    WTIMK

        RET


;-------------------------------------------------------------------------
;  Abstract:
;       Restart the watch timer after stopping.
;-------------------------------------------------------------------------
WT_Start:
        ; enable watch timer operation
        ; start the 5-bit counter
        SET1    WTM.1
        SET1    WTM.0

        RET


;-------------------------------------------------------------------------
;  Abstract:
;       INTTM00 interrupts service routine.
;-------------------------------------------------------------------------
MD_INTWTI:
        MOV1    CY, P0.2
        NOT1    CY
        MOV1    P0.2, CY                ; Output level reversal

        RETI

END
```

3.16.2 ＳＴＯＰ－タイマ割込みによる復帰

■ ファイル名

STOP_timer

- STOP_timer.asm（アセンブリ言語ソース）
- STOP_timer.c（Ｃ言語ソース）

■ 説明

スタンバイ機能は，システムの消費電力をより低減するための機能で，次の2種類のモードがあります。

（1）HALTモード

HALTモードは，CPUの動作クロックを停止させるモードです。システム・クロック発振回路の発振は継続します。このモードでは，STOPモードほどの消費電流の低減はできませんが，割込み要求により，すぐに処理を再開したい場合や，間欠動作をさせたい場合に有効です。

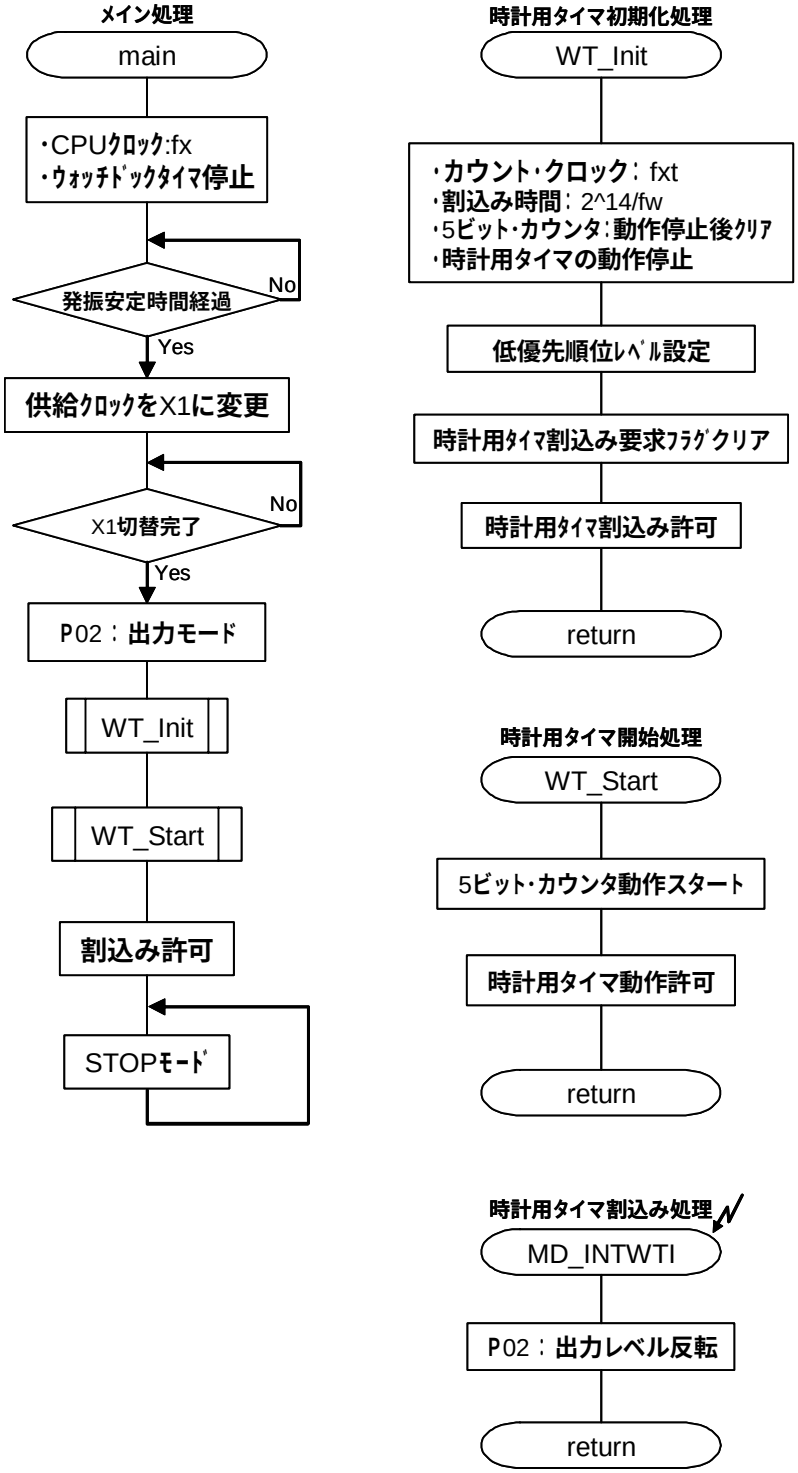
（2）STOPモード

STOPモードは，X1入力クロック発振回路を停止させ，システム全体が停止するモードです。CPUの消費電流を，かなり低減することができます。さらに，割込み要求によって解除できるため，間欠動作も可能です。ただし，STOPモード解除時に発振安定時間確保のためのウエイト時間がとられるため，割込み要求によって，すぐに処理を開始しなければならない場合にはHALTモードを選択してください。

このプログラムは、0.5us 毎の時計タイマ割込みで STOP から復帰し、割込み処理内で P02 を反転出力させます。(時計用タイマは SubCLK で動作させます)そして、メイン処理は、出力設定後無限ループにします。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

初期設定で 0.5μs 毎に時計タイマ割込みを発生させるように、サブクロックを使用して時計タイマを設定します。メインループでは、STOP モードへの遷移が行われ、時計タイマ割込みにより解除します。割込み処理内にて P02 の出力レベルを反転させます。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
P02：出力モード
WT_Init
WT_Start
割込み許可
STOPﾓｰﾄﾞ
時計用タイマ初期化処理
WT_Init
・カウント・クロック：fxt
・割込み時間：2^14/fw
・5ビット・カウンタ：動作停止後ｸﾘｱ
・時計用タイマの動作停止
低優先順位ﾚﾍﾞﾙ設定
時計用ﾀｲﾏ割込み要求ﾌﾗｸﾞクリア
時計用ﾀｲﾏ割込み許可
return
時計用タイマ開始処理
WT_Start
5ビット・カウンタ動作スタート
時計用タイマ動作許可
return
時計用タイマ割込み処理
MD_INTWTI
P02：出力レベル反転
return

■ C言語ソース

```
/*
********************************************************************************
*
*  Title: The return from STOP by timer interruption
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - watch timer count clock: subsystem clock
*               - watch timer overflow time: 0.5s
*               - output port: port 0.2 (toggles every 0.5s)
*
********************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma STOP
#pragma EI

/*
--------------------------------------------------------------------------------
    Specify Interrupt Vectors
--------------------------------------------------------------------------------
*/
#pragma interrupt   INTWT       MD_INTWT


/*
--------------------------------------------------------------------------------
    Constants/Variables
--------------------------------------------------------------------------------
*/

/*status list definition*/
#define TRUE            1
#define FALSE           0

void WT_Init( void );
void WT_Start( void );


/*
--------------------------------------------------------------------------------
    Main Program
--------------------------------------------------------------------------------
*/

void main( void )
{

    PCC = 0x00;                         /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0 );
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    PM0.2 = 0;                          /* set P02 is output mode */

    WT_Init();                          /* WT initialization function */
    WT_Start();                         /* WT start function */

    EI();                               /* Master interruption permission */

    while(TRUE)                         /* main loop */
    {
```

```
        NOP();
        STOP();                         /* stop mode */
        NOP();
    }
}


/*
-------------------------------------------------------------------------------
    Functions Group
-------------------------------------------------------------------------------
*/

/*
-------------------------------------------------------------------------------
  Abstract:
    Initiate the watch timer, select its working mode, count clock,
    enable/disable interrupt etc through Configuration.
-------------------------------------------------------------------------------
*/
void WT_Init( void )
{
    WTM = 0x80;                         /* watch timer flag setting:2^14/fw(0.5s) */
                                        /* Interval timer prescaler setting:2^4/fw */
                                        /* Set watch timer clock:fw=fxt */

    WTPR=1;                             /* Low priority level */
    WTIF=0;                             /* No interrupt request signal is generated */
    WTMK=0;                             /* Interrupt servicing enabled */
}


/*
-------------------------------------------------------------------------------
  Abstract:
    Restart the watch timer after stopping.
-------------------------------------------------------------------------------
*/
void WT_Start( void )
{
    /*enable watch timer operation*/
    /*start the 5-bit counter*/
    WTM.1=1;
    WTM.0=1;
}

/*
-------------------------------------------------------------------------------
  Abstract:
    INTTM00 interrupt service routine.
-------------------------------------------------------------------------------
*/
__interrupt void MD_INTWT( void )
{
    P0.2 ^= 1;                          /* Output level reversal */
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: The return from STOP by timer interruption
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - watch timer count clock: subsystem clock
;               - watch timer overflow time: 0.5s
;               - output port: port 0.2 (toggles every 0.5s)
;
;*****************************************************************************
;



;---------------------------------------------------------------------------
;       Specify Interrupt Vectors
;---------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to  Start
                DW      start
INTWT_VECTOR    CSEG AT 002eh
                DW      MD_INTWT


;---------------------------------------------------------------------------
;       Main Program
;---------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CLR1    PM0.2
        CALL    !WT_Init
        CALL    !WT_Start
        EI                              ; Master interruption permission

main:                                   ; main loop
        NOP
        STOP                            ; stop mode
        NOP
        BR      $main


;---------------------------------------------------------------------------
;       Functions Group
;---------------------------------------------------------------------------


;---------------------------------------------------------------------
;  Abstract:
```

```
;       Initiate the watch timer, select its working mode, count clock,
;       enable/disable interrupt etc through Configuration.
;-------------------------------------------------------------------------
WT_Init:

        MOV     WTM, #80H       ; Set watch timer clock:fw=fxt
                                ; watch timer flag setting:2^14/fw(0.5s)
                                ; Interval timer prescaler setting:2^4/fw

        ; INTWT enable
        SET1    WTPR
        CLR1    WTIF
        CLR1    WTMK

        RET

;-------------------------------------------------------------------------
;  Abstract:
;       Restart the watch timer after stopping.
;-------------------------------------------------------------------------
WT_Start:
        ; enable watch timer operation
        ; start the 5-bit counter
        SET1    WTM.1
        SET1    WTM.0

        RET

;-------------------------------------------------------------------------
;  Abstract:
;  INTTM00 interrupts service routine.
;-------------------------------------------------------------------------
MD_INTWT:
        MOV1    CY, P0.2
        NOT1    CY
        MOV1    P0.2, CY        ; Output level reversal

        RETI

END
```

3.16.3 ＳＴＯＰ－端子割込みによる復帰

■ ファイル名

STOP_pin

- STOP_pin.asm（アセンブリ言語ソース）
- STOP_pin.c（Ｃ言語ソース）

■ 説明

スタンバイ機能は，システムの消費電力をより低減するための機能で，次の2種類のモードがあります。

（1）HALTモード

HALTモードは，CPUの動作クロックを停止させるモードです。システム・クロック発振回路の発振は継続します。このモードでは，STOPモードほどの消費電流の低減はできませんが，割込み要求により，すぐに処理を再開したい場合や，間欠動作をさせたい場合に有効です。

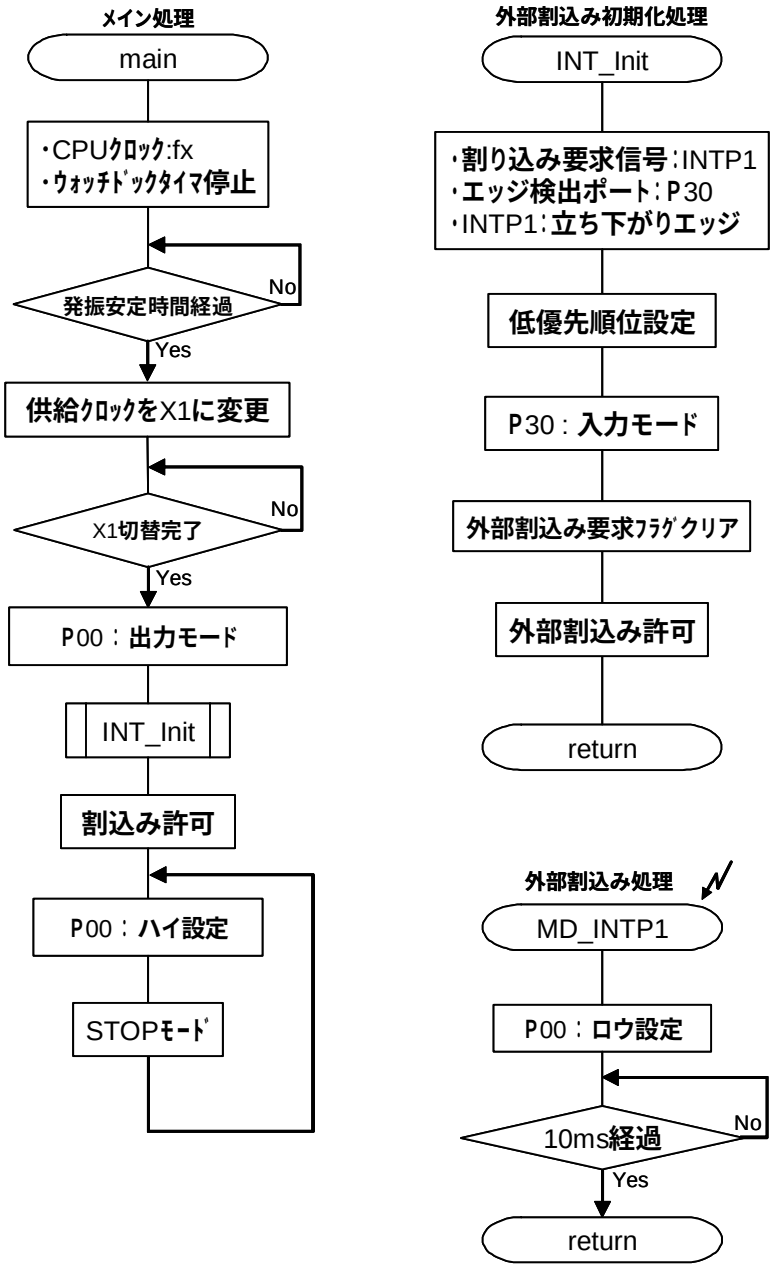
（2）STOPモード

STOPモードは，X1入力クロック発振回路を停止させ，システム全体が停止するモードです。CPUの消費電流を，かなり低減することができます。さらに，割込み要求によって解除できるため，間欠動作も可能です。ただし，STOPモード解除時に発振安定時間確保のためのウエイト時間がとられるため，割込み要求によって，すぐに処理を開始しなければならない場合にはHALTモードを選択してください。

このプログラムは、INTP1 の立下りにて、STOP モードを解除させ P00 を 10ms の間 Low にします。時間経過後、P00 を High にして STOP モードへ遷移させます。そしてメイン処理は、出力設定後 無限ループにします。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

初期設定で INTP1 の立ち下がりで端子割込みを発生させるように設定します。メインループでは、P00 を High にして STOP モードへの遷移が行われ、端子割込みにより解除します。端子割込み処理内では P00 を Low に設定した後、約 10ms の wait を行っています。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
P00：出力モード
INT_Init
割込み許可
P00：ハイ設定
STOPﾓｰﾄﾞ
外部割込み初期化処理
INT_Init
・割り込み要求信号：INTP1
・エッジ検出ポート：P30
・INTP1：立ち下がりエッジ
低優先順位設定
P30：入力モード
外部割込み要求ﾌﾗｸﾞクリア
外部割込み許可
return
外部割込み処理
MD_INTP1
P00：ロウ設定
10ms経過
No
Yes
return

■ C言語ソース

```
/*
********************************************************************************
*
*  Title: The return from STOP by terminal interruption
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - external interrupt: INTP1 (falling edge)
*               - interrupt service routine delay: approximately 10ms
*               - output port: port0.0 (P00/TI000)
*
********************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI
#pragma STOP

/*
--------------------------------------------------------------------------------
    Specify Interrupt Vectors
--------------------------------------------------------------------------------
*/
#pragma interrupt   INTP1   MD_INTP1


/*
--------------------------------------------------------------------------------
    Constants/Variables
--------------------------------------------------------------------------------
*/

/*status list definition*/
#define TRUE            1
#define FALSE           0

void INT_Init( void );

/*
--------------------------------------------------------------------------------
    Main Program
--------------------------------------------------------------------------------
*/

void main( void )
{

    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    PM0.0 = 0;                          /* set P00 is output mode */
    INT_Init();                         /* INT initialization function */
    EI();                               /* Master interruption permission */

    while(TRUE)                         /* main loop */
    {
        P0.0 = 1;                       /* P00=1 */
        NOP();
        STOP();                         /* stop mode */
        NOP();
    }
```

```
}


/*
--------------------------------------------------------------------------------
    Functions Group
--------------------------------------------------------------------------------
*/

/*
--------------------------------------------------------------------------
  Abstract:
    Init the external INT, including enable or disable, priority setting.
--------------------------------------------------------------------------
*/
void  INT_Init( void )
{
    EGP.1 = 0;                          /* Falling edge set */
    EGN.1 = 1;

    PPR1 = 1;                           /* Low priority level */
    PM3.0 = 1;                          /* set P30 input mode */
    PIF1 = 0;                           /* No interrupt request signal is generated */
    PMK1 = 0;                           /* Interrupt servicing enabled */
}


/*
--------------------------------------------------------------------------
  Abstract:
    INTP1 Interrupt service routine.
--------------------------------------------------------------------------
*/
__interrupt void MD_INTP1( void )
{
    unsigned short  i;

    P0.0 = 0;                           /* P00=0 */
    for(i=0 ; i<4600 ; i++);            /* wait 10 ms */
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: The return from STOP by terminal interruption
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - external interrupt: INTP1 (falling edge)
;               - interrupt service routine delay: approximately 10ms
;               - output port: port0.0 (P00/TI000)
;
;*****************************************************************************
;



;---------------------------------------------------------------------------
;       Specify Interrupt Vectors
;---------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to  Start
                DW      start
INTP1_VECTOR    CSEG AT 0008h
                DW      MD_INTP1


;---------------------------------------------------------------------------
;       Main Program
;---------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CLR1    PM0.0
        CALL    !INT_Init
        EI                              ; Enable interrupts

main:
        SET1    P0.0
        NOP
        STOP
        NOP
        BR      $main


;---------------------------------------------------------------------------
;       Functions Group
;---------------------------------------------------------------------------


;---------------------------------------------------------------------
;  Abstract:
```

```
;       Init the external INT, including enable or disable, priority setting.
;-------------------------------------------------------------------------
INT_Init:

        CLR1    EGP.1
        SET1    EGN.1
        SET1    PPR1
        SET1    PM3.0
        CLR1    PIF1
        CLR1    PMK1                    ; enable INTP1

        RET


;-------------------------------------------------------------------------
;  Abstract:
;       INTP1 Interrupt service routine.
;-------------------------------------------------------------------------
MD_INTP1:
        SEL     RB1

        CLR1    P0.0

        MOVW    AX, #0000H
MD_INTP1_01:
        CMPW    AX, #4600               ; wait 10ms
        BNC     $MD_INTP1_02
        INCW    AX
        BR      $MD_INTP1_01
MD_INTP1_02:

        RETI

END
```

３．１７ クロック･モニタ

■ ファイル名

- CLK_Monitor.asm（アセンブリ言語ソース）
- CLK_Monitor.c（C言語ソース）

■ 説明

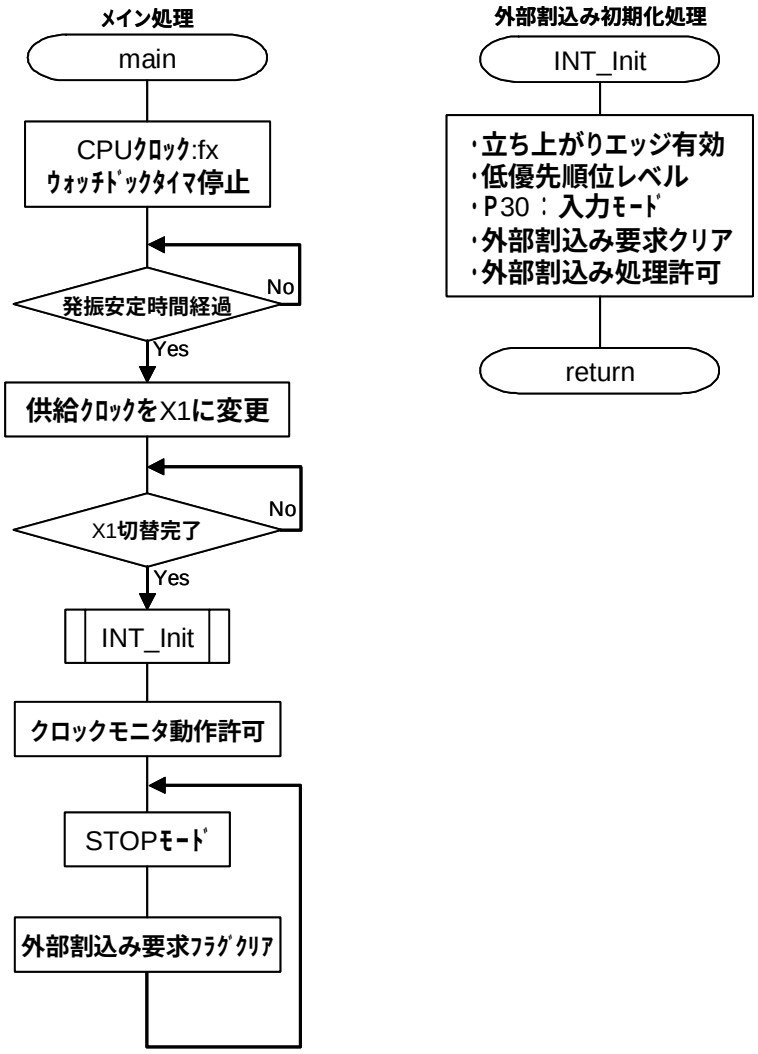
内蔵のRing-OSCにて，X1入力クロックのサンプリングを行い，X1入力クロックの発振停止時に，内部リセット信号を発生する，という機能を持ちます。
次の条件のとき，クロック・モニタは自動的に停止します。

・STOPモード時～発振安定時間
・ソフトウエアによりX1入力クロック停止時（MSTOP=1またはMCC = 1のとき）
・リセット解除後の発振安定時間
・Ring-OSCクロック停止時

このプログラムは､CLK モニタ機能を設定した状態で STOP モードにし､INTP1 の立ち上がりで STOP モードを解除するのを繰り返します。そしてメイン処理は、出力設定後 無限ループにします。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

初期設定にて CLME をセットし、CLK モニタ機能を設定します。メインループにて、STOP モードへ遷移させます。 STOP モードの解除は、INTP1 の立ち上がりエッジで行いますが、割込みは使用せずにメインループを継続して実行させます。

■ フローチャート


メイン処理
main
CPUｸﾛｯｸ:fx
ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
INT_Init
クロックモニタ動作許可
STOPﾓｰﾄﾞ
外部割込み要求ﾌﾗｸﾞｸﾘｱ
外部割込み初期化処理
INT_Init
・立ち上がりエッジ有効
・低優先順位レベル
・P30：入力ﾓｰﾄﾞ
・外部割込み要求クリア
・外部割込み処理許可
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: Clock Monitor
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - external interrupt: INTP1 (Rising edge)
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI
#pragma STOP


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

/*status list definition*/
#define TRUE            1
#define FALSE           0

void INT_Init( void );

/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/

void main( void )
{

    PCC = 0x00;                         /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    INT_Init();                         /* INT initialization function */
    CLME = 1;                           /* clock monitor enable */

    while(TRUE)                         /* main loop */
    {
        NOP();
        STOP();                         /* Stop mode */
        NOP();
        PIF1 = 0;                       /* request flag clear */
    }
}


/*
------------------------------------------------------------------------
    Functions Group
------------------------------------------------------------------------
*/

/*
```

```
--------------------------------------------------------------------------
  Abstract:
    Init the external INT, including enable or disable, priority setting.
--------------------------------------------------------------------------
*/
void  INT_Init( void )
{
    /* INTP1 select rising edge */
    EGP = 0x02;
    EGN = 0x00;

    PPR1 = 1;                           /* priority: Low */
    PM3.0 = 1;                          /* set P30 input mode */
    PIF1 = 0;                           /* request flag clear */
    PMK1 = 0;                           /* enable INTP1 */
}
```

■ アセンブリ言語ソース

```
;
;*******************************************************************************
;
;  Title: Clock Monitor
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - external interrupt: INTP1 (Rising edge)
;
;*******************************************************************************


;-------------------------------------------------------------------------
;       Specify Interrupt Vectors
;-------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to Start
                DW      start


;-------------------------------------------------------------------------------
;       Main Program
;-------------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        CALL    !INT_Init
        SET1    CLME                    ; clock monitor enable

main:                                   ; main loop
        NOP
        STOP                            ; Stop mode
        NOP
        CLR1    PIF1                    ; request flag clear
        BR      $main


;-------------------------------------------------------------------------------
;       Functions Group
;-------------------------------------------------------------------------------


;-------------------------------------------------------------------------
;  Abstract:
;       Init the external INT, including enable or disable, priority setting.
;-------------------------------------------------------------------------
INT_Init:

        MOV     EGP, #02H
        MOV     EGN, #00H
```

```
        SET1    PPR1                    ; priority: Low
        SET1    PM3.0                   ; set P30 input mode
        CLR1    PIF1
        CLR1    PMK1                    ; enable INTP1

        RET

END
```

３．１８ 低電圧検出回路

■ ファイル名

- Lvi.asm（アセンブリ言語ソース）
- Lvi.c（C言語ソース）

■ 説明

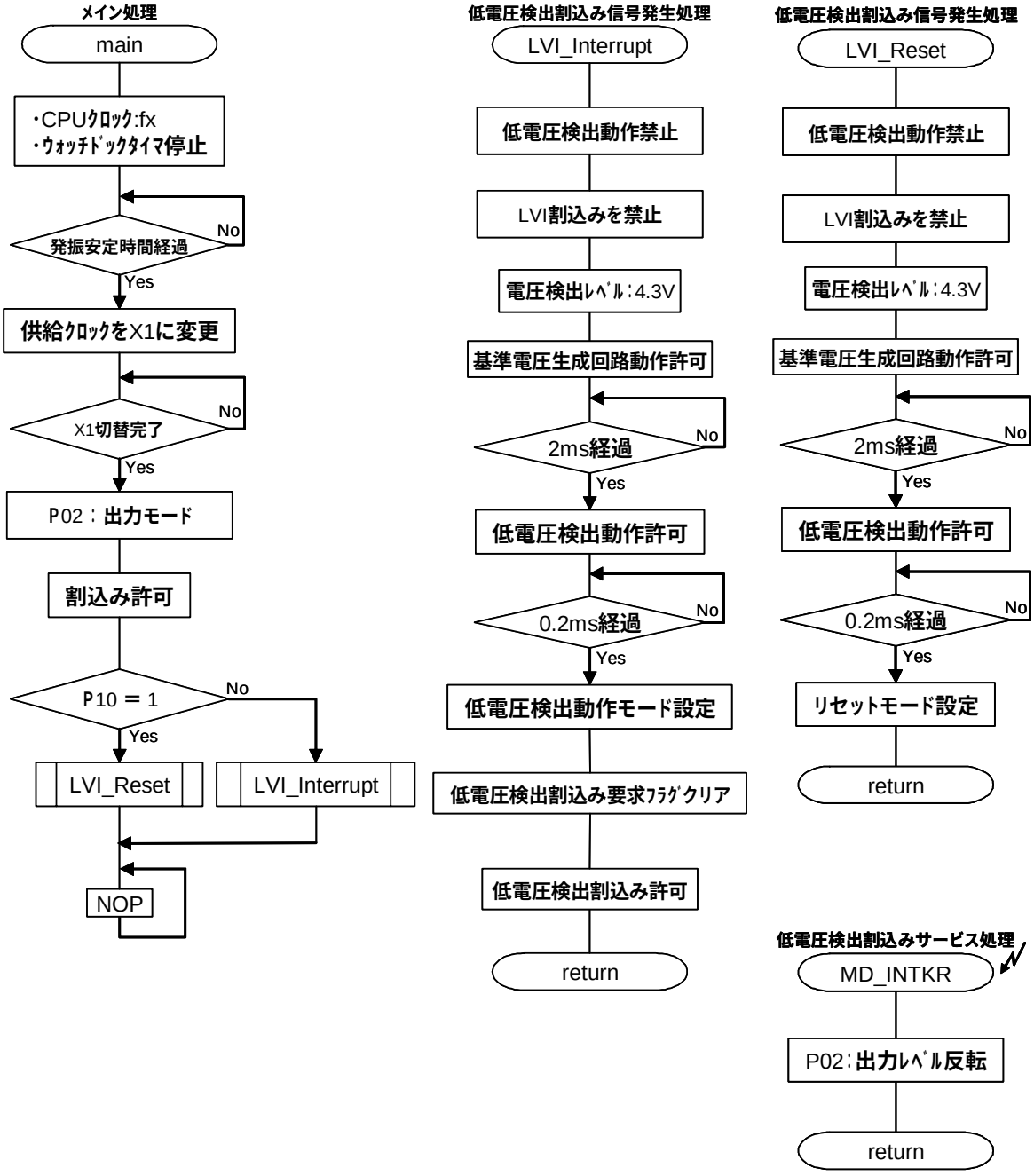
低電圧検出（LVI）回路は次のような機能を持ちます。

・電源電圧（VDD）と検出電圧（VLVI）を比較し，VDD＜VLVIになったとき，内部割込み信号もしくは内部リセット信号を発生します。
・電源電圧の検出レベル（5段階）をソフトウエアにて変更できます。
・割込み／リセットをソフトウエアにて選択できます。
・STOPモード時において動作可能です。

このプログラムは、メイン処理において P10 が High の時、内部リセット信号モード、P10 が Low の時、割込み信号発生モードを選択します。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

P10 の入力レベルを判断して内部リセット信号モードと割込み信号発生モードを切り替えます。P10=1 の時、内部リセット信号モードを選択し、検出電圧が 4.3V 未満になった時に内部リセットを発生させます。P10=0 の時、割込み信号発生モードを選択し、検出電圧が 4.3V 未満になった時に低電圧検出割込みを発生させます。割込み処理内で P02 の出力レベルを反転させます。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
発振安定時間経過
No
Yes
供給ｸﾛｯｸをX1に変更
X1切替完了
No
Yes
P02：出力モード
割込み許可
P10 ＝ 1
No
Yes
LVI_Reset
LVI_Interrupt
NOP
低電圧検出割込み信号発生処理
LVI_Interrupt
低電圧検出動作禁止
LVI割込みを禁止
電圧検出ﾚﾍﾞﾙ：4.3V
基準電圧生成回路動作許可
2ms経過
No
Yes
低電圧検出動作許可
0.2ms経過
No
Yes
低電圧検出動作モード設定
低電圧検出割込み要求ﾌﾗｸﾞクリア
低電圧検出割込み許可
return
低電圧検出割込み信号発生処理
LVI_Reset
低電圧検出動作禁止
LVI割込みを禁止
電圧検出ﾚﾍﾞﾙ：4.3V
基準電圧生成回路動作許可
2ms経過
No
Yes
低電圧検出動作許可
0.2ms経過
No
Yes
リセットモード設定
return
低電圧検出割込みサービス処理
MD_INTKR
P02：出力ﾚﾍﾞﾙ反転
return

■ C言語ソース

```
/*
************************************************************************
*
*  Title: Low voltage detection circuit
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - reset mode when port 1.0 = 1
*               - interrupt mode when port 1.0 = 0
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI

/*
------------------------------------------------------------------------
    Specify Interrupt Vectors
------------------------------------------------------------------------
*/
#pragma interrupt   INTLVI      MD_INTLVI


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

/*status list definition*/
#define TRUE            1
#define FALSE           0

void LVI_Reset( void );
void LVI_Interrupt( void );


/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/

void main( void )
{

    PCC = 0x00;                         /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0);
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    PM0.2 = 0;                          /* set P02 output mode */
    EI();                               /* Master interruption permission */

    /* control port level judge */
    if( P1.0 == 1 ){
        LVI_Reset();                    /* LVI reset function */
    }
    else{
        LVI_Interrupt();                /* LVI interrupt function */
    }
```

```
    while(TRUE)                         /* Endless loop */
    {
        NOP();
    }
}


/*
-------------------------------------------------------------------------------
    Functions Group
-------------------------------------------------------------------------------
*/

/*
-------------------------------------------------------------------------
  Abstract:
    This function initializes the Low-Voltage Detector.
    Initial LVI function according to the configuration setting.
-------------------------------------------------------------------------
*/

void  LVI_Interrupt( void )
{
    unsigned short i;

    LVIM = 0x00;                        /* stop/init LVI */
    LVIMK = 1;                          /* mask LVI interrupt */

    LVIS = 0x00;                        /* compare with 4.3V */

    LVIE = 1;
    for( i=0 ; i<=1000 ; i++ );         /* wait 2 ms */

    LVION = 1;                          /* enable LVI operation */
    for( i=0 ; i<=100 ; i++ );          /* wait 0.2 ms */

    LVIMD = 0;                          /* internal interrupt mode */

    LVIPR = 1;                          /* Low priority level */
    LVIIF = 0;                          /* No interrupt request signal is generated */
    LVIMK = 0;                          /* Interrupt servicing enabled */

}


/*
-------------------------------------------------------------------------
  Abstract:
    This function initializes the Low-Voltage Detector.
    Initial LVI function according to the configuration setting.
-------------------------------------------------------------------------
*/
void LVI_Reset( void )
{
    unsigned short i;

    LVIM = 0x00;                        /* stop/init LVI */
    LVIMK = 1;                          /* Interrupt servicing disabled */

    LVIS = 0x00;                        /* compare with 4.3V */

    LVIE = 1;
    for( i=0 ; i<=1000 ; i++ );         /* wait 2 ms */

    LVION = 1;                          /* enable LVI operation */
    for( i=0 ; i<=100 ; i++ );          /* wait 0.2 ms */

    LVIMD = 1;                          /* reset mode */
}
```

```
/*
--------------------------------------------------------------------------
  Abstract:
    INTLVI interrupt service routine.
--------------------------------------------------------------------------
*/
__interrupt void MD_INTLVI( void )
{
    P0.2 ^= 1;                          /* Output level reversal */
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: Low voltage detection circuit
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - reset mode when port 1.0 = 1
;               - interrupt mode when port 1.0 = 0
;
;*****************************************************************************


;-----------------------------------------------------------------------------
;       Specify Interrupt Vectors
;-----------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h   ;On reset, go to Start
                DW      start
INTLVI_VECTOR   CSEG AT 0004h
                DW      MD_INTLVI

;-----------------------------------------------------------------------------------
;       Main Program
;-----------------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

main:
        CLR1    PM0.2
        EI

        BF      P1.0, $main01
        CALL    !LVI_Reset
        BR      $main02

main01:
        CALL    !LVI_Interrupt

main02:                                 ; Endless loop
        NOP
        BR      $main02


;-----------------------------------------------------------------------------------
;       Functions Group
;-----------------------------------------------------------------------------------


;-----------------------------------------------------------------------
```

```
;  Abstract:
;       This function initializes the Low-Voltage Detector.
;       Initial LVI function according to the configuration setting.
;-------------------------------------------------------------------------
LVI_Interrupt:
        MOV     LVIM, #00H              ;stop/init LVI
        SET1    LVIMK                   ;mask LVI interrupt
        MOV     LVIS, #00H              ;compare with 4.3V
        SET1    LVIE

        MOVW    AX, #0000H
LVI_Interrupt01:
        CMPW    AX, #1000               ;wait 2 ms
        BNC     $LVI_Interrupt02
        INCW    AX
        BR      $LVI_Interrupt01

LVI_Interrupt02:
        SET1    LVION                   ;enable LVI operation

        MOVW    AX, #0000H
LVI_Interrupt03:
        CMPW    AX, #100                ;wait 0.2 ms
        BNC     $LVI_Interrupt04
        INCW    AX
        BR      $LVI_Interrupt03

LVI_Interrupt04:
        CLR1    LVIMD                   ;internal interrupt mode
        SET1    LVIPR
        CLR1    LVIIF
        CLR1    LVIMK

        RET


;-------------------------------------------------------------------------
;  Abstract:
;       This function initializes the Low-Voltage Detector.
;       Initial LVI function according to the configuration setting.
;-------------------------------------------------------------------------
LVI_Reset:
        MOV     LVIM, #00H              ;stop/init LVI
        SET1    LVIMK                   ;mask LVI interrupt
        MOV     LVIS, #00H              ;compare with 4.3V
        SET1    LVIE

        MOVW    AX, #0000H
LVI_Reset01:
        CMPW    AX, #1000               ;wait 2 ms
        BNC     $LVI_Reset02
        INCW    AX
        BR      $LVI_Reset01

LVI_Reset02:
        SET1    LVION                   ;enable LVI operation

        MOVW    AX, #0000H
LVI_Reset03:
        CMPW    AX, #100                ;wait 0.2 ms
        BNC     $LVI_Reset04
        INCW    AX
        BR      $LVI_Reset03

LVI_Reset04:
        SET1    LVIMD                   ;reset mode

        RET
```

```
;-------------------------------------------------------------------------
;  Abstract:
;  INTLVI interrupts service routine.
;-------------------------------------------------------------------------
MD_INTLVI:
        MOV1    CY, P0.2
        NOT1    CY
        MOV1    P0.2, CY                ; Output level reversal

        RETI

END
```

３．１９ リセット･アウト

■ ファイル名

- Reset_Out.asm（アセンブリ言語ソース）
- Reset_Out.c（Ｃ言語ソース）

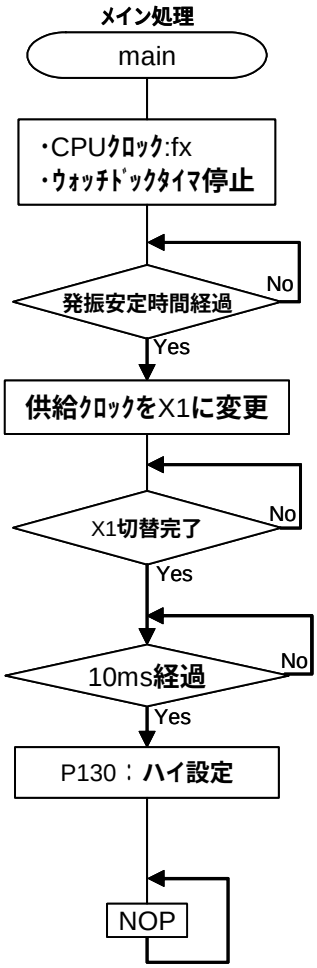
■ 説明

出力専用ポートのＰ１３０(ﾘｾｯﾄ値：0)を使用し、リセット時の外部へのリセット信号出力動作を行う。

このプログラムは、リセット解除後、P130 から 10ms 間 Low の波形を一回出力します。そしてメイン処理は、出力設定後無限ループにします。CPU 供給ｸﾛｯｸは、X1 入力ｸﾛｯｸ(10MHz)を選択します。

リセット解除後、動作クロックをメインクロック(X1)に切り替えます。その後、約 10ms の wait 処理を行います。そして時間経過後に P130 の出力レベルを High にする。

■ フローチャート


メイン処理
main
・CPUｸﾛｯｸ:fx
・ｳｫｯﾁﾄﾞｯｸﾀｲﾏ停止
No
発振安定時間経過
Yes
供給ｸﾛｯｸをX1に変更
No
X1切替完了
Yes
No
10ms経過
Yes
P130：ハイ設定
NOP

■ C言語ソース

```
/*
************************************************************************
*
*  Title: Reset signal output
*
*  Parameters:  - fastest CPU clock (fx=10MHz)
*               - port 13.0 outputs
*
************************************************************************
*/
#pragma sfr
#pragma NOP
#pragma EI


/*
------------------------------------------------------------------------
    Constants/Variables
------------------------------------------------------------------------
*/

/*status list definition*/
#define TRUE            1
#define FALSE           0


/*
------------------------------------------------------------------------
    Main Program
------------------------------------------------------------------------
*/

void main( void )
{
    unsigned short  i;

    PCC  = 0x00;                        /* CPU clock: fx */
    WDTM = 0x77;                        /* Watchdog Timer Stop */

    /* Waiting for oscillation stable time */
    while( OSTC.0 == 0 );
    MCM0 = 1;                           /* supply clock: X1 */
    /* Waiting for X1 clock change */
    while( MCS == 0 );

    for(i=0 ; i<4600 ; i++);            /* wait 10ms */
    P13.0 = 1;                          /* P130=1 */

    while(TRUE)                         /* Endless loop */
    {
        NOP();
    }
}
```

■ アセンブリ言語ソース

```
;
;*****************************************************************************
;
;  Title: Reset signal output
;
;  Parameters:  - fastest CPU clock (fx=10MHz)
;               - port 13.0 outputs
;
;*****************************************************************************


;---------------------------------------------------------------------------
;       Specify Interrupt Vectors
;---------------------------------------------------------------------------

RESET_VECTOR    CSEG AT 0000h           ; On reset, go to Start
                DW      start


;-----------------------------------------------------------------------------
;       Main Program
;-----------------------------------------------------------------------------
START           CSEG

start:
        SEL     RB0
        MOVW    SP, #0FE20h
        MOV     PCC, #00H               ; CPU clock: fx
        MOV     WDTM, #77H              ; Watchdog Timer Stop

; Waiting for oscillation stable time
start01:
        BT      OSTC.0, $start02
        BR      $start01
start02:
        SET1    MCM0                    ; enable X1
; Waiting for X1 clock change
start03:
        BT      MCS, $start04
        BR      $start03
start04:

        MOVW    AX, #0000H
start05:
        CMPW    AX, #4600               ; wait 10ms
        BNC     $start06
        INCW    AX
        BR      $start05
start06:
        SET1    P13.0

main:                                   ; Endless loop
        NOP
        BR      $main

END
```