

---

---

## 第 11 章 充電時間測定ユニット (CTMU)

---

---

### ハイライト

本章では、次のトピックについて説明します。

11.1	はじめに .....	11-2
11.2	レジスタ .....	11-3
11.3	CTMU の動作 .....	11-5
11.4	CTMU モジュールの初期化 .....	11-6
11.5	CTMU モジュールの較正 .....	11-7
11.6	CTMU モジュールによる容量の測定 .....	11-12
11.7	CTMU モジュールによる時間の測定 .....	11-14
11.8	CTMU モジュールによる遅延の生成 .....	11-14
11.9	スリープ/アイドルモード中の動作 .....	11-16
11.10	リセットの CTMU への影響 .....	11-16
11.11	レジスタ マップ .....	11-17
11.12	電気的特性 .....	11-18
11.13	関連するアプリケーション ノート .....	11-19
11.14	改版履歴 .....	11-20

## 11.1 はじめに

充電時間測定ユニット (CTMU) は柔軟性を備えるアナログ モジュールで、パルス間の正確な時間差が測定できると同様に、非同期のパルスが生成できます。他のオンチップのアナログ モジュールと併用することで、CTMU は正確な時間測定、容量測定、容量の相対変化測定などや、一定の遅延を持ったパルス出力を生成できます。CTMU は容量ベースのセンサとのインターフェースに理想的です。

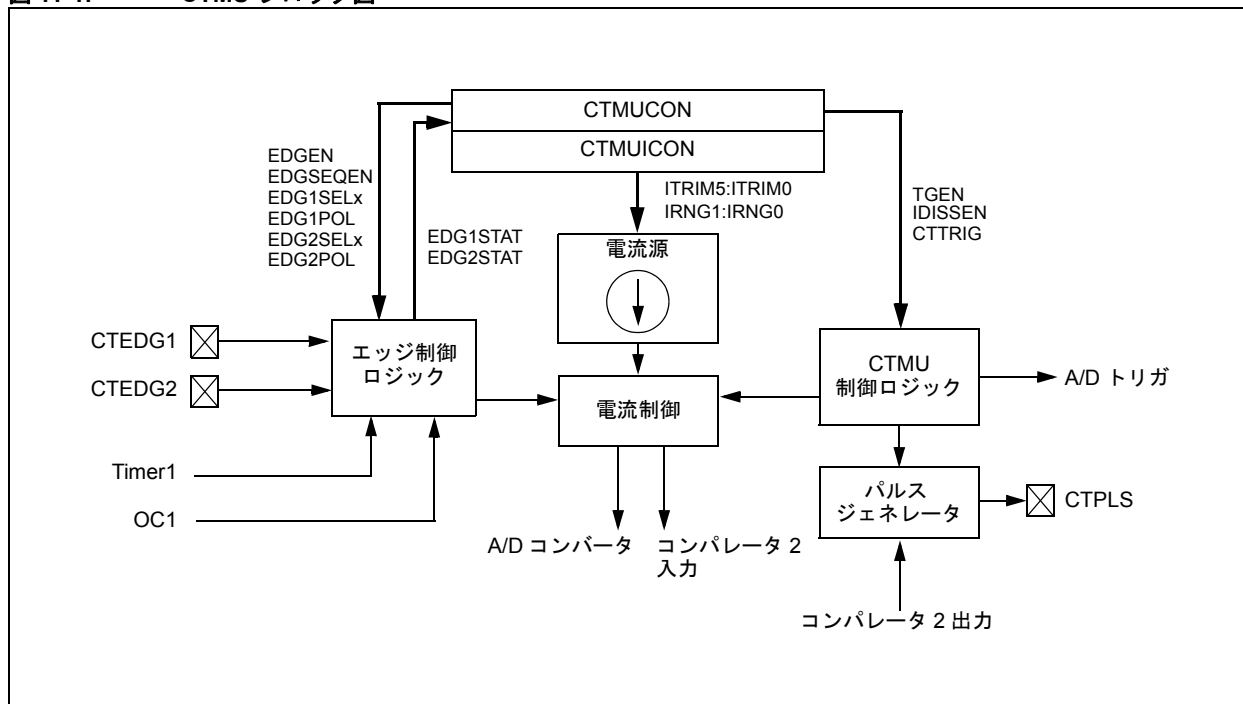
モジュールには、下記の主な特性があります。

- 16 チャンネルまでの容量または時間測定の入力
- オンチップの高精度電流源
- 4 回のエッジ入力トリガ源
- 各エッジ源の極性制御
- エッジシーケンスの制御
- エッジ応答の制御
- 高精度時間測定
- システム クロックに同期した外付けまたは内蔵信号に対する遅延

CTMU は A/D コンバータと併用して、特定のデバイスの A/D チャンネル数が有効であれば、最大 16 チャンネルまでの時間か電荷が測定できます。遅延として構成した場合、CTMU はアナログ コンパレータの 1 つに接続されます。レベルで感応するエッジ源は、2 つの外部入力、タイマ 1 または出力コンペア モジュール 1 の 4 つのいずれかから選択できます。有効な入力源となるデバイス別の、固有の情報については、対応する PIC24F のデータシートを参照してください。

CTMU のブロック図を図 11-1 に示します。

図 11-1: CTMU ブロック図



## 11.2 レジスタ

CTMU に関連するレジスタとして、CTMUCON と CTMUICON の 2 つがあります。

CTMUCON レジスタ (レジスタ 11-1) には、CTMU モジュールのエッジ源選択、エッジ源極性選択、エッジシーケンス、A/D トリガ、アナログ回路のキャパシタの放電の制御、それに有効化を設定する制御ビットが含まれています。CTMUICON レジスタ (レジスタ 11-2) には、電流源のレンジと電流源の微調用のビットがあります。

レジスタ 11-1: CTMUCON: CTMU 制御レジスタ

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CTMUEN	—	CTMUSIDL	TGEN	EDGEN	EDGSEQEN	IDISSEN	CTTRIG
bit 15						bit 8	

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
EDG2POL	EDG2SEL1 <sup>(1)</sup>	EDG2SEL0 <sup>(1)</sup>	EDG1POL	EDG1SEL1 <sup>(1)</sup>	EDG1SEL0 <sup>(1)</sup>	EDG2STAT	EDG1STAT
bit 7						bit 0	

## 記号の説明:

R = 読み出し可	W = 書き込み可	U = 未実装、読むと「0」
-n = POR 後の値	「1」 = セット	「0」 = クリア
		x = 不定

- ビット 15 **CTMUEN**: CTMU 有効化ビット  
 1 = モジュールを有効とする  
 0 = モジュールを無効とする
- ビット 14 **未実装**: 読むと「0」
- ビット 13 **CTMUSIDL**: アイドル時停止制御ビット  
 1 = デバイスがアイドルモードになったらモジュール動作停止  
 0 = アイドルモード中もモジュール動作継続
- ビット 12 **TGEN**: 時間生成有効化ビット  
 1 = エッジ遅延生成を有効とする  
 0 = エッジ遅延生成を無効とする
- ビット 11 **EDGEN**: エッジ有効化ビット  
 1 = エッジをブロックしない  
 0 = エッジをブロックする
- ビット 10 **EDGSEQEN**: エッジシーケンス有効化ビット  
 1 = エッジ 1 イベントがエッジ 2 のイベントより前に発生していること  
 0 = エッジシーケンスを無効とする
- ビット 9 **IDISSEN**: アナログ電流源制御ビット  
 1 = アナログ電流源出力をグラウンドに接続  
 0 = アナログ電流源出力をグラウンドに接続しない
- ビット 8 **CTTRIG**: トリガ制御ビット  
 1 = トリガ出力を有効とする  
 0 = トリガ出力を無効とする
- ビット 7 **EDG2POL**: エッジ 2 の極性選択ビット  
 1 = エッジ 2 を正のレベル応答とする  
 0 = エッジ 2 を負のレベル応答とする

注 1: 特定のエッジ源のタイプや指定については、特定のデバイスのデータシートを参照してください。

# PIC24F ファミリ リファレンス マニュアル

## レジスタ 11-1: CTMUCON: CTMU 制御レジスタ (続き)

ビット 6-5 **EDG2SEL1:EDG2SEL0:** エッジ 2 源選択ビット (1)

11 = エッジ源 3 を選択  
 10 = エッジ源 2 を選択  
 01 = エッジ源 1 を選択  
 00 = エッジ源 0 を選択

ビット 4 **EDG1POL:** エッジ 1 の極性選択ビット

1 = エッジ 1 を正のレベル応答とする  
 0 = エッジ 1 を負のレベル応答とする

ビット 3-2 **EDG1SEL1:EDG1SEL0:** エッジ 1 源選択ビット (1)

11 = エッジ源 3 を選択  
 10 = エッジ源 2 を選択  
 01 = エッジ源 1 を選択  
 00 = エッジ源 0 を選択

ビット 1 **EDG2STAT:** エッジ 2 状態ビット

1 = エッジ 2 のイベントが発生  
 0 = エッジ 2 のイベントは発生していない

ビット 0 **EDG1STAT:** エッジ 1 状態ビット

1 = エッジ 1 のイベントが発生  
 0 = エッジ 1 のイベントは発生していない

注 1: 特定のエッジ源のタイプや指定については、特定のデバイスのデータシートを参照してください。

## レジスタ 11-2: CTMUICON: CTMU 電流制御レジスタ

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ITRIM5	ITRIM4	ITRIM3	ITRIM2	ITRIM1	ITRIM0	IRNG1	IRNG0
bit 15						bit 8	

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 7						bit 0	

### 記号の説明:

R = 読み出し可                      W = 書き込み可                      U = 未実装、読むと「0」  
 -n = POR 後の値                      「1」 = セット                      「0」 = クリア                      x = 不定

ビット 15-10 **ITRIM5:ITRIM0:** 電流源微調整ビット

011111 = 通常電流から正の最大変更  
 011110

.....  
 100010

100001 = 通常電流から負の最大変更

ビット 9-8 **IRNG1:IRNG0:** 電流源レンジ選択ビット

11 = 100 × 基本電流  
 10 = 10 × 基本電流  
 01 = 基本電流レベル (0.55 μA 標準)  
 00 = 電流源無効

ビット 7-0 **未実装:** 読むと「0」

## 11.3 CTMU の動作

CTMU は回路を充電するために固定の電流源を使用して動作します。回路のタイプは計測方式によって決定されます。電荷計測の場合には、電流は固定で、回路に供給する電流の時間も固定とします。これで、電圧を A/D で読み取ると回路の容量が測定できます。時間測定の場合は、回路の電流および容量を固定とします。これで、A/D で電圧を読み取ると、電流源の回路充電の開始から終了までの時間がわかります。

CTMU を遅延用を使用する場合には、容量と電流両方固定とし、コンパレータ回路に供給する電圧も同様に固定とします。遅延時間は、充電電圧がコンパレータのスレッシュホールド電圧になるまでの時間として決定されます。

### 11.3.1 動作理論

CTMU の動作は充電に関する下記式に基づいています。

$$C = I \cdot \frac{dV}{dT}$$

簡単に言い換えると、回路の電荷量をクーロンで測定すると、電流のアンペア ( $I$ ) と電流が流れた時間の秒数 ( $t$ ) の積で表されます。また、電荷量は容量ファラッド ( $C$ ) と回路電圧 ( $V$ ) の積で表されます。したがって、下記式が成り立ちます。

$$I \cdot t = C \cdot V$$

CTMU モジュールは一定の既知の電流源を提供します。A/D コンバータで式の ( $V$ ) を測定し、残りの 2 つ、容量 ( $C$ ) と時間 ( $t$ ) は未知数となります。上記式から、下記いずれかの式が容量または時間の計算に使用できます。

$$t = (C \cdot V) / I$$

これは回路の容量が固定で既知の場合に用います。

$$C = (I \cdot t) / V$$

これは電流源が回路に適用される時間が固定の場合に用います。

### 11.3.2 電流源

CTMU の中心部は高精度の電流源で、測定用の一定のリファレンスを提供します。電流レベルは、3 つのレンジまたは 2 桁の大きさからユーザが選択可能で、出力を  $\pm 2\%$  (標準) だけ微調整できます。電流レンジは IRNG1:IRNG0 (CTMUICON<9:8>) ビットで選択でき、値「00」が最低レンジとなります。

電流微調整は、ITRIM5:ITRIM0 (CTMUICON<15:10>) ビットで行われます。これら 6 ビットが電流源を約 2% ステップで微調整します。レンジの半分は電流を増加させる調整で、残り半分が電流源を減少させます。値「000000」が中央値です (変化なし)。値「100000」が負の最大調整値 (約 -62%) で、「011111」が正の最大調整値 (約 +62%) です。

### 11.3.3 エッジ選択と制御

CTMU 計測は、モジュールの 2 つの入力チャンネルで起きるエッジ イベントで制御されます。チャンネルごとに、エッジ 1 とエッジ 2 として区別し、エッジ入力ピン (CTEDG1 と CTEDG2)、タイマ 1 または出力コンペア モジュール 1 のいずれかからパルスを入力するように設定できます。入力チャンネルはレベル感応ですが、レベル変化より、チャンネルの瞬時のレベルに対してより反応します。入力は EDG1SEL と EDG2SEL のビット ペア (CTMUCON<3:2 と 6:5>) で選択されます。

さらに、チャンネルごとに EDGE2POL と EDGE1POL ビット (CTMUCON<7,4>) でイベントの極性が設定できます。また、入力チャンネルは、EDGSEQEN ビット (CTMUCON<10>) によってエッジ イベント シーケンス (エッジ 1 がエッジ 2 より先) でフィルタもできます。

## 11.3.4 エッジ状態

CTMUCON レジスタには2つの状態ビット、EDG2STAT と EDG1STAT (CTMUCON<1:0>)があります。これらの主要な機能はエッジ応答が対応するチャンネルで起きたかどうかを示すことです。CTMU は、対応するチャンネルでエッジ応答が検出されると自動的にこのビットをセットします。入力チャンネルはレベル感応であるため、チャンネル設定が変更されたり、チャンネルの電流状態が変化する状態ビットにとすぐセットされます。

モジュールは、エッジ状態ビットを使用して外部のアナログモジュール (A/D コンバータなど) への電流源の出力を制御します。外部モジュールへの電流は、状態ビットの一方 (両方ではなく) がセットされたときのみ供給し、両方の状態ビットがセットまたはクリアされると電流を遮断します。これで、CTMU がエッジ間の電流のみ測定できるようにしています。両方の状態ビットをセット後、次の測定を実行するには両ビットをクリアする必要があります。CTMU 電流源が再度有効化されないように、できるだけ両方のビットを同時にクリアしてください。

エッジ状態ビットは、CTMU がハードウェアでセットする以外に、ソフトウェアでもセットできます。つまり、ユーザアプリケーションで手動で電流源を有効あるいは無効にできます。いずれか一方 (両方ではなく) のセットで電流源が有効化され、両方の同時セットまたはクリアで電流源が無効化されます。

## 11.3.5 割り込み

CTMU は電流源が有効化あるいは無効化されるごとに割り込みフラグ (IFS4<13>) をセットします。割り込みは、対応する割り込み許可ビット (IEC4<13>) のセット時のみ生成されます。エッジシーケンス (つまりエッジ1がエッジ2より先に発生) が有効でなければ、どちらのエッジによって割り込みが発生したかを決定するには、エッジ状態ビットをモニタする必要があります。

## 11.4 CTMU モジュールの初期化

次のシーケンスが CTMU モジュールの初期化の標準的なガイドラインです。

1. 電流レンジ選択を IRNG ビット (CTMUICON<9:8>) で行う
2. 電流微調整を ITRIM ビット (CTMUICON<15:10>) で行う
3. エッジ1とエッジ2のエッジ入力源設定をEDG1SELとEDG2SELビット(CTMUICON<3:2と6:5>)をセットして行う
4. エッジ入力の入力極性設定を EDG1POL と EDG2POL ビット (CTMUICON<4,7>) で行う。デフォルト設定は負極性 (High から Low への遷移)
5. エッジシーケンスの有効化を EDGSEQEN ビット (CTMUICON<10>) で行う。デフォルトではエッジシーケンスは無効
6. 動作モード選択 (計測か遅延) を TGEN ビットで行う。デフォルトモードは時間 / 容量の測定
7. 2回目のエッジイベント発生でモジュールが A/D コンバータを自動トリガするように CTRIG ビット (CTMUICON<8>) で設定する。変換トリガはデフォルトでは無効
8. 接続回路の放電を IDISSEN ビット (CTMUICON<9>) をセットして行う。回路の放電に十分な時間だけ待機した後、IDISSEN をクリアする
9. CTMUEN ビット (CTMUICON<15>) ビットをクリアして CTMU を無効化する
10. エッジ状態ビット EDG2STAT と EDG1STAT (CTMUICON<1:0>) をクリアする
11. EDGEN ビット (CTMUICON<11>) をセットして両方のエッジ入力を有効化する
12. CTMUEN ビットをセットして CTMU を有効化する

計測かパルス生成をさせるかのタイプにより、1 つ以上の追加モジュールの初期化と、CTMU モジュールの設定が必要です。

- エッジ源生成の場合 : 外部エッジ入力ピンに加えて、タイマ1と出力コンペア /PWM1 モジュール両方を CTMU のエッジ源として使用できる
- 容量または時間計測の場合 : CTMU モジュールは、A/D コンバータをアナログ入力チャンネルの1つに接続されたコンデンサの電圧計測に使用する
- パルス生成の場合 : システムクロックとは独立した出力パルスを生成する場合には、CTMU モジュールはコンパレータ2とコンパレータに関連する電圧リファレンスを使用する

これらのモジュールの初期化に関する情報は、PIC24F ファミリ リファレンスの対応するモジュールの章を参照してください。

## 11.5 CTMU モジュールの較正

CTMU は容量や時間の高精度な測定、また正確な遅延にも較正を必要とします。アプリケーションで容量か時間の相対変化の測定のみが必要な場合には、較正は通常必要ありません。このようなアプリケーションの例には容量性タッチスイッチがあり、ここではタッチ回路が基本の容量となり、人体による容量変化が回路の全体容量に変化を加えます。

実際の容量や時間の計測が必要な場合には、2つのハードウェアによる較正が必要です。1つは正確な電流設定に必要な電流源の較正で、もう1つは被測定回路の測定される対象以外の全容量の測定較正です。

### 11.5.1 電流源の較正

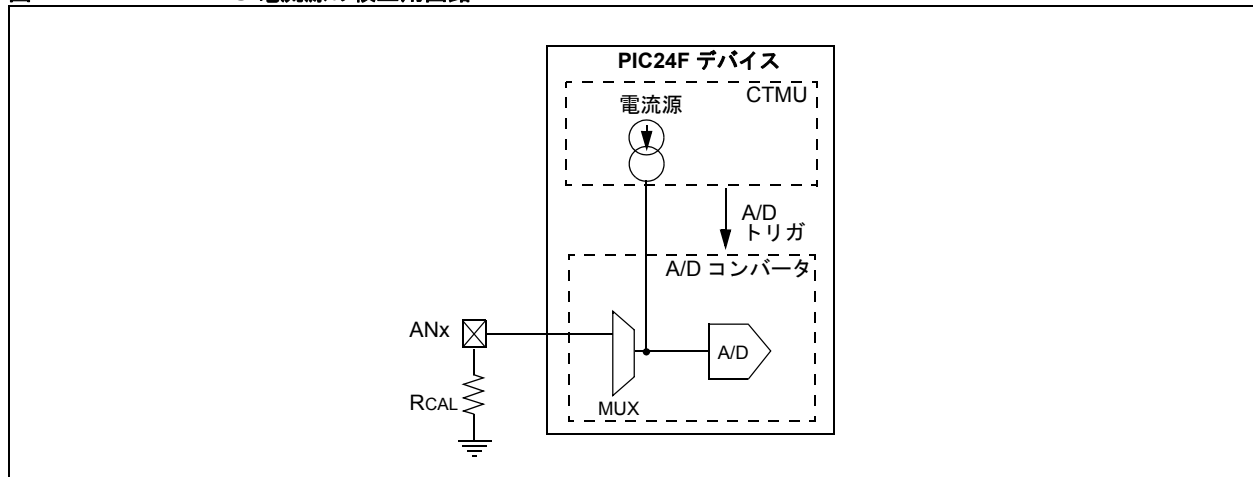
CTMU モジュール内蔵の電流源は、3種の電流レンジごとに  $\pm 60\%$  標準のレンジを有しています。したがって、高精度な測定には未使用のアナログチャンネルに高精度抵抗  $R_{CAL}$  を接続することで、この電流源を測定および調整できます。図 11-2 に回路例を示します。電流源の計測は下記手順で行います。

1. A/D コンバータを初期化する
2. CTMU を初期化する
3. EDG1STAT (CTMUCON<0>) をセットして電流源を有効とする
4. セトリングの時間待機する
5. A/D 変換を実行する
6.  $I = V/R_{CAL}$  で電流源の電流を計算する。ここで  $R_{CAL}$  は高精度抵抗で、 $V$  は A/D 変換による計測値

CTMU 電流源は、CTMUICON の微調整ビットで微調整を繰り返すことで正確な目標値に達することができます。また、調整なしの標準値も使用できます。その値をソフトウェアで格納し、すべての容量と時間測定に用います。

$R_{CAL}$  値の計算には、標準電流値を用いて抵抗値を計算してください。例えば、A/D コンバータのリファレンスが 3.3V の場合、そのフルスケールの 70% か、2.31V を A/D コンバータで読み取るときの目標電圧値として使用します。CTMU 電流源のレンジが  $0.55 \mu\text{A}$  と選択されていれば、必要な抵抗値は  $R_{CAL} = 2.31\text{V}/0.55 \mu\text{A}$  で計算され、 $4.2 \text{M}\Omega$  となります。同様に、電流源が  $5.5 \mu\text{A}$  と選択されていれば、 $R_{CAL}$  は  $420,000\Omega$ 、 $55 \mu\text{A}$  として設定されていれば  $42,000\Omega$  となります。

図 11-2: CTMU 電流源の較正用回路



フルスケール電圧の 70% という値は、A/D コンバータがノイズ レベルよりも確実に十分高いレンジであるように選択されています。正確な電流とするには、CTMUICON からのビットで微調整が必要で、それに対応した RCAL の抵抗値が必要であることを留意してください。また、RCAL が有効な抵抗値となるように調整する必要があります。CTMU を計測に使用する際の回路を高精度にするには、RCAL には最も高精度な値としてください。推奨の最低値は 0.1% 精度です。

次の例に、CTMU の電流較正を実行する標準的な方法を示します。例 11-1 に A/D コンバータと CTMU をどのように初期化するかを示します。このルーチンは両モジュールを使用する場合の標準です。例 11-2 には、実際の較正ルーチンを示します。この方法では A/D コンバータを手動でトリガし、これは全段階の手順をデモしています。また、CTMU の CTTRIG ビット (CTMUCON<8>) をセットすることで変換を自動的にトリガできます。

## 例 11-1: CTMU 較正ルーチンの設定

```
#include "p24Fxxxx.h"

/*****
/*Setup CTMU
*****/

void setup(void)
{ //CTMUCON - CTMU Control register
  CTMUCON = 0x0090; //make sure CTMU is disabled
  //CTMU continues to run when emulator is stopped,CTMU continues
  //to run in idle mode,Time Generation mode disabled, Edges are blocked
  //No edge sequence order, Analog current source not grounded, trigger
  //output disabled, Edge2 polarity = positive level, Edge2 source =
  //source 0, Edg1 polarity = positive level, Edg1 source = source 0,
  // Set Edge status bits to zero

  //CTMUICON - CTMU Current Control Register
  CTMUICON = 0x0001; //0.55uA, Nominal - No Adjustment
/*****
//setupAD converter;
*****/
  TRISB=0x0004; //set channel 2 as an input
  AD1PCFG=0x0004; //
  AD1CHS=0x002; //select the analog channel(2)
  AD1CSSL=0x0000; //

  AD1CON1 = 0x8000; //Turn On A/D Converter, continue in
  // idle mode, Unsigned fractional format, Clear
  Samp bit to start
  //conversion, Sample when SAMP bit is set,
  sampling on hold

  AD1CON2 = 0x0000; //VR+ = AVDD, V- = AVSS, Don't scan,
  //always use MUX A inputs

  AD1CON3 = 0x0000; //A/D uses system clock, conversion
  //clock = 1xTcy
}
```

例 11-2: 電流較正ルーチン

```

#include "p24Fxxxx.h"

#define COUNT 500 //@ 8MHz = 125uS.
#define DELAY for(i=0;i<COUNT;i++)
#define RCAL .027 //R value is 4200000 (4.2M)
//scaled so that result is in
//1/100th of uA
#define ADSCALE 1023 //for unsigned conversion 10 sig bits
#define ADREF 3.3 //Vdd connected to A/D Vr+

int main(void)
{
    int i;
    int j = 0; //index for loop
    unsigned int Vread = 0;
    double VTot = 0;
    float Vavg=0, Vcal=0, CTMUISrc = 0; //float values stored for calcs

    //assume CTMU and A/D have been setup correctly
    //see Example 11-1 for CTMU & A/D setup
    setup();

    CTMUCONbits.CTMUEN = 1; //Enable the CTMU

    for(j=0;j<10;j++)
    {
        AD1CON1bits.SAMP = 1; //Manual sampling start
        CTMUCONbits.IDISSEN = 1; //drain charge on the circuit
        DELAY; //wait 125us
        CTMUCONbits.IDISSEN = 0; //end drain of circuit

        CTMUCONbits.EDG1STAT = 1; //Begin charging the circuit
        //using CTMU current source
        DELAY; //wait for 125us
        CTMUCONbits.EDG1STAT = 0; //Stop charging circuit

        IFS0bits.AD1IF = 0; //make sure A/D Int not set
        AD1CON1bits.SAMP = 0; //and begin A/D conv.
        while(!IFS0bits.AD1IF); //Wait for A/D convert complete
        AD1CON1bits.DONE = 0;
        Vread = ADC1BUF0; //Get the value from the A/D
        IFS0bits.AD1IF = 0; //Clear A/D Interrupt Flag
        VTot += Vread; //Add the reading to the total
    }

    Vavg = (float)(VTot/10.000); //Average of 10 readings
    Vcal = (float)(Vavg/ADSCALE*ADREF);
    CTMUISrc = Vcal/RCAL; //CTMUISrc is in 1/100ths of uA
}

```

## 11.5.2 容量の較正

正確な容量測定に影響するものとして、A/D コンバータの内部サンプル容量と、回路ボードのパターンやパッドに浮遊容量によるわずかな容量があります。浮遊容量は計測目的とする容量を取り除くことで確実に測定できます。計測は下記手順で行います。

1. A/D コンバータと CTMU を初期化する
2. EGD1STAT を 1 にセットする
3. 固定遅延時間  $t$  だけ待機する
4. EDG1STAT をクリアする
5. A/D 変換を実行する
6. 浮遊容量と A/D コンバータのサンプル容量を下記で計算する

$$C_{\text{OFFSET}} = C_{\text{STRAY}} + C_{\text{AD}} = (I \cdot t) / V$$

ここで、 $I$  は電流源測定ステップで計測される値で、 $t$  は固定の遅延時間、 $V$  は A/D 変換で得られた電圧値とします。

較正の際には、 $C_{\text{STRAY}} + C_{\text{AD}}$  の容量値はおよその値でもわかっていることが必要です。 $C_{\text{AD}}$  はおよそ 4 pF です。

回路が A/D コンバータで読み取るのに適当な電圧まで充電するまでの時間  $t$  は、何回か測定を繰り返して調整する必要があります。 $t$  の値は、 $C_{\text{OFFSET}}$  に理論値を設定して求めることで決定できます。例えば、 $C_{\text{STRAY}}$  の理論値が 11 pF とし、 $V$  が VDD の 70% か 2.31V とすれば、 $t$  は下記で計算できます。

$$(4 \text{ pF} + 11 \text{ pF}) \cdot 2.31\text{V} / 0.55 \mu\text{A}$$

つまり 63  $\mu\text{s}$  となります。

標準的な CTMU の容量較正のルーチンを例 11-3 に示します。

## 例 11-3: 容量校正のルーチン

```

#include "p24Fxxxx.h"

#define COUNT 25          //@ 8MHz INTFRC = 62.5 us.
#define ETIME COUNT*2.5  //time in us
#define DELAY for(i=0;i<COUNT;i++)
#define ADSCALE 1023     //for unsigned conversion 10 sig bits
#define ADREF 3.3        //Vdd connected to A/D Vr+

int main(void)
{
    int i;
    int j = 0;            //index for loop
    unsigned int Vread = 0;
    float CTMUISrc, CTMUCap, Vavg, VTot, Vcal;

                                //assume CTMU and A/D have been setup correctly
                                //see Example 11-1 for CTMU & A/D setup
    setup();

    CTMUCONbits.CTMUEN = 1;//Enable the CTMU

    for(j=0;j<10;j++)
    {
        AD1CON1bits.SAMP = 1;    //Manual sampling start
        CTMUCONbits.IDISSEN= 1;  //drain any charge on the circuit
        DELAY;                   //wait 62.5 us
        CTMUCONbits.IDISSEN = 0; //end drain of circuit
        CTMUCONbits.EDG1STAT = 1; //Begin charging the circuit
                                //using the CTMU current source
        DELAY;                   //wait for 62.5 us for circuit to charge
        CTMUCONbits.EDG1STAT = 0; //Stop charging circuit and begin A/D conv.
        AD1CON1bits.SAMP = 0;
        while(!IFS0bits.AD1IF); //Wait for A/D conversion to complete
        Vread = ADC1BUF0;        //Get the value from the A/D converter
        IFS0bits.AD1IF = 0;     //Clear AD1IF
        VTot += Vread;          //Add the reading to the total
    }

    Vavg = (VTot/10); //Average of 10 readings
    Vcal = (Vavg/ADSCALE*ADREF);
    CTMUCap = (CTMUISrc*ETIME/Vcal)/100;
    //CTMUISrc is in 1/100ths of uA,
    //calculated in Example 1-2
    //time is in us
    //CTMUCap is in pF
}

```

## 11.6 CTMU モジュールによる容量の測定

CTMU による容量測定には 2 つの方法があります。1 つは絶対的方法で、実際の容量値が欲しい場合に使用します。もう 1 つは相対的方法で、実際の容量値は必要なく、容量変化の区別が必要な場合に使用します。

### 11.6.1 絶対容量の測定

容量の絶対値の計測の場合には、11.5 項「CTMU モジュールの較正」に示した電流と容量の較正が必要です。その後容量の測定は次の手順で実行します。

1. A/D コンバータを初期化する
2. CTMU を初期化する
3. EDG1STAT をセットする
4. 固定遅延時間  $T$  だけ待機する
5. EDG1STAT をクリアする
6. A/D 変換を実行する
7. 合計容量を  $C_{TOTAL} = (I * T)/V$  で計算する。ここで  $I$  は電流源測定のステップ (11.5.1 項「電流源の較正」) で求められる。 $T$  は固定の遅延で、 $V$  は A/D 変換による計測値
8.  $C_{TOTAL}$  から浮遊容量と A/D 容量 (COFFSET は 11.5.2 項「容量の較正」で求める) を引き算して計測容量を決定する

### 11.6.2 相対容量の測定

正確な容量計測が必要ないアプリケーションがあります (アプリケーションによっては、正確な容量計測は必要ありません)。例えば、容量方式のスイッチの押下を検出する場合には、容量の相対変化のみが必要となります。このようなタイプのアプリケーションでは、スイッチがオフ (つまりタッチしていない) ときには、合計容量は基板のパターンと A/D コンバータなどの結合容量となります。A/D コンバータでの計測は高めの電圧となります。スイッチがオン (タッチしたとき) は、人体による容量が上記容量に加わるため合計容量が大きくなり、A/D コンバータの計測値は低めの電圧となります。

容量変化は CTMU により下記手順で容易に検出できます。

1. A/D コンバータと CTMU を初期化する
2. EDG1STAT をセットする
3. 固定遅延時間、待機する
4. EDG1STAT をクリアする
5. A/D 変換を実行する

A/D 変換により計測される電圧は、相対容量を示します。この場合、電流源や容量測定の較正が必要ないことに留意してください。容量式タッチ スイッチのソフトウェア ルーチン例を例 11-4 に示します。

例 11-4: 容量式タッチスイッチのルーチン

```

#include "p24Fxxxx.h"

#define COUNT 500 // @ 8MHz = 125us.
#define DELAY for(i=0;i<COUNT;i++)
#define OPENSW 1000 // Unpressed switch value
#define TRIP 300 // Difference between pressed
// and unpressed switch
#define HYST 65 // amount to change
// from pressed to unpressed

#define PRESSED 1
#define UNPRESSED 0

int main(void)
{
    unsigned int Vread; // storage for reading
    unsigned int switchState;
    int i;

    // assume CTMU and A/D have been setup correctly
    // see Example 11-1 for CTMU & A/D setup

    setup();

    CTMUCONbits.CTMUEN = 1; // Enable the CTMU

    AD1CON1bits.SAMP = 1; // Manual sampling start
    CTMUCONbits.IDISSEN = 1; // drain charge on the circuit
    DELAY; // wait 125us
    CTMUCONbits.IDISSEN = 0; // end drain of circuit

    CTMUCONbits.EDG1STAT = 1; // Begin charging the circuit
    // using CTMU current source
    DELAY; // wait for 125us
    CTMUCONbits.EDG1STAT = 0; // Stop charging circuit

    IFS0bits.AD1IF = 0; // make sure A/D Int not set
    AD1CON1bits.SAMP = 0; // and begin A/D conv.
    while(!IFS0bits.AD1IF); // Wait for A/D convert complete
    AD1CON1bits.DONE = 0;
    Vread = ADC1BUF0; // Get the value from the A/D
    if(Vread < OPENSW - TRIP)
    {
        switchState = PRESSED;
    }
    else if(Vread > OPENSW - TRIP + HYST)
    {
        switchState = UNPRESSED;
    }
}

```

## 11.7 CTMU モジュールによる時間の測定

電流と容量を較正後の比 ( $C/I$ ) から正確な時間を計測するには下記手順で行います。

1. A/D コンバータと CTMU を初期化する
2. EDG1STAT をセットする
3. EDG2STAT をセットする
4. A/D 変換を実行する
5. エッジ間の時間を  $T = (C/I) * V$  で計算する。ここで  $I$  は電流較正 (11.5.1 項「電流源の較正」) で計算した値で、 $C$  は容量較正 (11.5.2 項「容量の較正」) で計算した値、 $V$  は A/D 変換で計測した電圧です。

ここで、時間測定では、容量 COFFSET が A/D 変換した電圧に与える影響は十分小さいものと仮定しています。最小時間測定の場合には、常に A/D チャンネル選択レジスタ (AD1CHS) を使用していない A/D チャンネル、つまりいかなる回路にも接続されていないピンに接続するものとします。これにより浮遊容量を最小化し、回路の合計容量を A/D コンバータそのものの容量 (4-5 pF) に近づけます。長めのインターバル時間測定の場合には、外付けのキャパシタを A/D チャンネルに接続し、時間測定の場合にはこのチャンネルを選択します。

## 11.8 CTMU モジュールによる遅延の生成

内蔵 CTMU モジュール独自の機能として、外付けキャパシタの値に基づいて、システムクロックに対して独立した出力パルスが生成できます。これは、内蔵コンパレータ用電圧リファレンスモジュールとコンパレータ2の入力ピンに外付けキャパシタを付けて使用することで実現できます。パルスは CTPLS ピンに出力されます。このモードを有効にするには TGEN ビットをセットしてください。

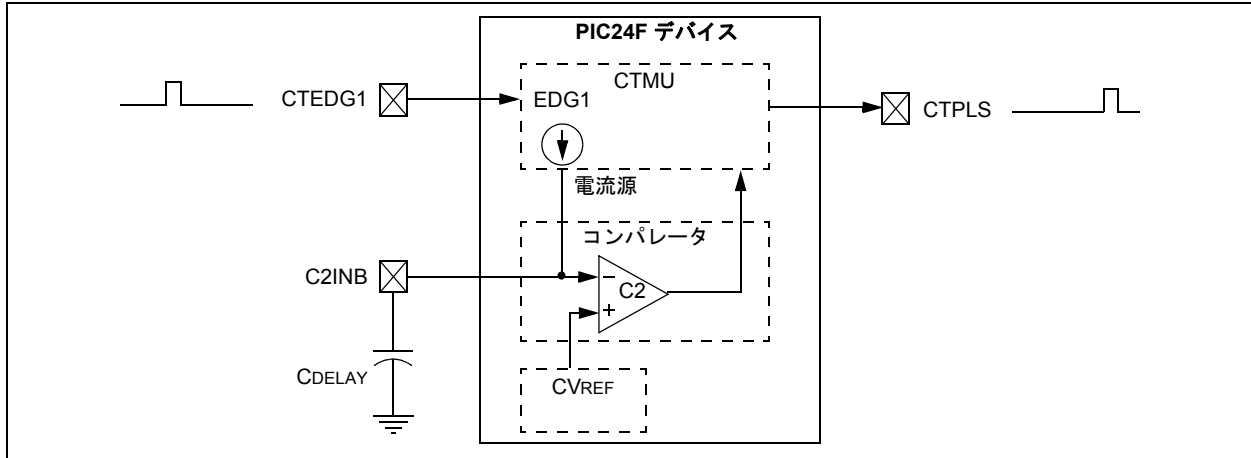
回路例を図 11-3 に示します。CPULSE には CTPLS ピンの出力パルス幅により決定される値を選択します。パルス幅は  $T = (CPULSE/I) * V$  で計算されます。ここで  $I$  は電流源測定ステップ (11.5.1 項「電流源の較正」) で求めた値で、 $V$  は内部リファレンス電圧 (CVREF) です。

この機能の使用例は、湿度センサのような可変容量方式センサとのインターフェースです。湿度変化により、CTPLS のパルス幅出力が変化します。アプリケーションでは、CTPLS 出力ピンを入力キャプチャピンに接続し、可変パルス幅を計測して湿度を求めます。

この機能は下記手順で使用します。

1. コンパレータ 2 を初期化する
2. コンパレータ用電圧リファレンスを初期化する
3. CTMU を初期化し、TGEN ビットをセットして遅延時間生成を有効化する
4. EDG1STAT をセットする
5. CPULSE が電圧リファレンスのトリップポイントまで充電されると、出力パルスが CTPLS に生成される

図 11-3: パルス遅延生成用の標準的な接続と内部設定



## 11.9 スリープ/アイドルモード中の動作

### 11.9.1 スリープモードとディープスリープモード

デバイスがいずれかのスリープモードになると、CTMU モジュールの電流源は常に無効となります。スリープモードになるときに CTMU が電流源に依存する動作中の場合、モジュールは正常に終了せず、容量あるいは時間測定値は誤った値となります。

### 11.9.2 アイドルモード

アイドルモード中の CTMU の動作は、CTMUSIDL ビット (CTMUCON<13>) で決定されます。CTMUSIDL がクリアされている場合、モジュールはアイドルモード中でも動作を継続します。CTMUSIDL がセットされている場合は、デバイスがアイドルモードに入ると、モジュールの電流源は無効化されます。アイドルモードに入るときにモジュールが動作中の場合はスリープモードと同様になります。

## 11.10 リセットの CTMU への影響

リセットにより、CTMU のすべてのレジスタがクリアされます。これで CTMU モジュールは無効化され、電流源はオフとなって、全オプション設定がデフォルト設定に戻ります。モジュールはリセット後再初期化する必要があります。

リセット時に CTMU が計測実行中の場合には、計測は行われません。計測中の回路には電荷の一部が残るため、CTMU で続けて計測をする場合には、その前に適切に放電させてください。回路の放電は、A/D コンバータが適切なチャンネルに接続されている場合には、IDISSEN ビット (CTMUCON<9>) をセットしてからクリアすることで行われます。

## 11.11 レジスタ マップ

PIC24F CTMU に関連するレジスタのまとめを表 11-1 に示します。

表 11-1: CTMU レジスタ マップ

File Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
CTMUCON	CTMUEN	—	CTMUSIDL	TGEN	EDGEN	EDGSEQEN	IDISSEN	CTTRIG	EDG2POL	EDG2SEL1	EDG2SEL0	EDG1POL	EDG1SEL1	EDG1SEL0	EDG2STAT	EDG1STAT	0000
CTMUICON	ITRIM5	ITRIM4	ITRIM3	ITRIM2	ITRIM1	ITRIM0	IRNG1	IRNG0	—	—	—	—	—	—	—	—	0000

記号の説明：— = 未実装、読むと「0」。リセット時の値は 16 進数で示す。

## 11.12 電気的特性

表 11-2: CTMU 電流源の仕様

DC 特性			標準動作条件: 2.0V ~ 3.6V (記載のない限り) 動作温度 $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ (工業用)				
パラメータ No	記号	特性	Min	Typ <sup>(1)</sup>	Max	単位	条件
	IOUT1	CTMU 電流源、 基本レンジ	—	550	—	nA	CTMUICON<1:0> = 01
	IOUT2	CTMU 電流源、 10x レンジ	—	5.5	—	$\mu\text{A}$	CTMUICON<1:0> = 10
	IOUT3	CTMU 電流源、 100x レンジ	—	55	—	$\mu\text{A}$	CTMUICON<1:0> = 11

注 1: 電流微調整レンジの中央付近での標準値 (CTMUICON<7:2> = 000000)

### 11.13 関連するアプリケーションノート

ここでは、マニュアルのこの章に関連するアプリケーション ノートをリストアップします。これらのアプリケーションノートは、特に PIC24F デバイス ファミリ専用ではありませんが、その概念は共通であり、変更、あるいは制限事項を考慮に入れて使用できます。現在、CTMU モジュールに関連するアプリケーション ノートは以下のとおりです。

タイトル	アプリケーション ノート #
現在関連するアプリケーション ノートはありません。	

**注：** PIC24F ファミリ デバイスに関するその他のアプリケーション ノートやコード例については、マイクロチップのウェブサイト([www.microchip.com](http://www.microchip.com))をご覧ください。

## 11.14 改版履歴

### リビジョン A (2008 年 3 月)

本文書の初版リリース。