Charge Time Measurement Unit (CTMU) and CTMU Operation with Threshold Detect

HIGHLIGHTS

This section of the manual contains the following major topics:

1.0 INTRODUCTION

The Charge Time Measurement Unit (CTMU) is a flexible analog module that has a configurable current source with a digital circuit built around it. The CTMU can be used for differential time measurement between pulse sources and can be used for generating an asynchronous pulse. By working with other on-chip analog modules, the CTMU can be used for high-resolution time, capacitance, resistance, inductance and temperature measurement. The CTMU can also generate output pulses with a specific time delay. These measurements can be used in applications including capacitive touch, humidity sensing and measuring relative changes in capacitance. The CTMU is ideal for interfacing with capacitive-based sensors.

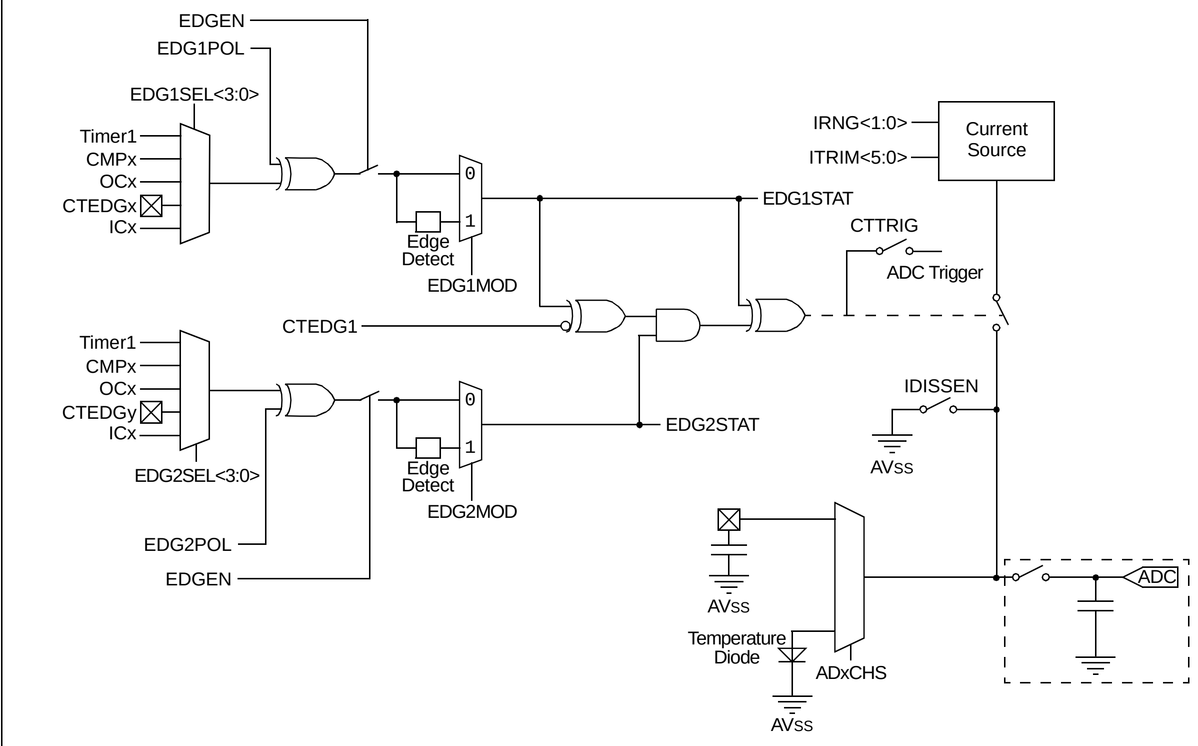
The module includes the following key features:

- On-chip precision current source
- Sixteen-edge input trigger sources
- Selection of edge or level-sensitive inputs
- Polarity control for each edge source
- Control of edge sequence
- Control of response to edges
- High-precision time measurement
- Time delay of external or internal signal asynchronous to system clock
- Integrated temperature sensing diode
- Control of current source during auto-sampling
- Four current source ranges
- Time measurement resolution of one nanosecond or less
- CTMU operation in Sleep mode
- Relative measurements using Analog-to-Digital Converter Threshold Detect

The CTMU works in conjunction with the A/D Converter for time or charge measurement. When configured for time delay, the CTMU is connected to one of the analog comparators. The input edge sources can be selected from up to sixteen sources for each edge. For device-specific information on available input sources, refer to the appropriate device data sheet.

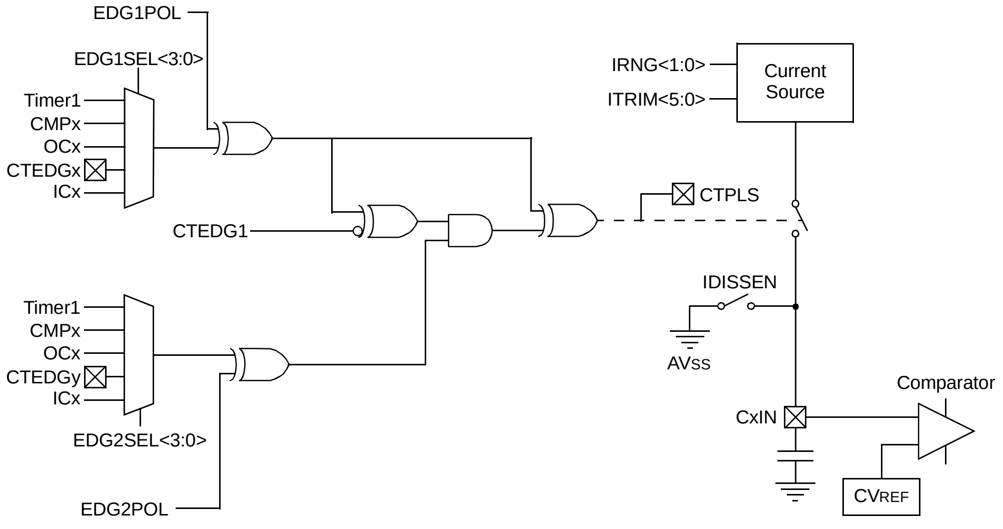
A block diagram of the CTMU in Measurement mode is shown in Figure 1-1 and in Time Generation (TGEN) mode in Figure 1-2.

Figure 1-1: CTMU Measurement Mode Block Diagram



Note 1: Refer to the particular device data sheet for specific edge source types and assignments.

Figure 1-2: CTMU Time Generation Block Diagram



Note 1: Refer to the particular device data sheet for specific edge source types and assignments.

2.0 REGISTER MAPS

Depending on the device variant, there are up to three control registers available for the CTMU: CMTUCON1H, CTMUCON1L and CTMUCON2L.

The CTMUCON1H and CTMUCON1L registers (Register 2-1 and Register 2-2) contain control bits for configuring the CTMU module edge source selection, edge source polarity selection, edge sequencing, ADC trigger, current source range and current source trim. The CTMUCON2L register (Register 2-3) has bits for selecting the Current Source Reset, and analog circuit capacitor discharge and enables.

A summary of the registers associated with the dsPIC33/PIC24 CTMU and CTMU Operation with Threshold Detect is provided in Table 2-1.

Table 2-1: CTMU Register Map

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
CTMUCON1H	EDG1MOD	EDG1POL	EDG1SEL3	EDG1SEL2	EDG1SEL1	EDG1SEL0	EDG2STAT	EDG1STAT	EDG2MOD	EDG2POL	EDG2SEL3	EDG2SEL2	EDG2SEL1	EDG2SEL0	—	IRNGH	`0000`
CTMUCON1L	CTMUEN	—	CTMUSIDL	TGEN	EDGEN	EDGSEQEN	IDISSEN	CTTRIG	ITRIM5	ITRIM4	ITRIM3	ITRIM2	ITRIM1	ITRIM0	IRNG1	IRNG0	`0000`
CTMUCON2L	—	—	—	—	—	—	—	—	—	—	—	IRSTEN	—	DSCHS2	DSCHS1	DSCHS0	`0000`

Legend: — = unimplemented, read as ‘0’. Reset values are shown in hexadecimal.

Register 2-1: CTMUCON1H: CTMU Control Register 1 High

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
EDG1MOD	EDG1POL	EDG1SEL3	EDG1SEL2	EDG1SEL1	EDG1SEL0	EDG2STAT	EDG1STAT
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
EDG2MOD	EDG2POL	EDG2SEL3	EDG2SEL2	EDG2SEL1	EDG2SEL0	—	IRNGH
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15 **EDG1MOD:** Edge 1 Edge-Sensitive Select bit
1 = Input is edge-sensitive
0 = Input is level-sensitive

bit 14 **EDG1POL:** Edge 1 Polarity Select bit
1 = Edge 1 is programmed for a positive edge response
0 = Edge 1 is programmed for a negative edge response

bit 13-10 **EDG1SEL<3:0>:** Edge 1 Source Select bits
1111 = CMP C3OUT
1110 = CMP C2OUT
1101 = CMP C1OUT
1100 = IC3 interrupt
1011 = IC2 interrupt
1010 = IC1 interrupt
1001 = CTED8 pin
1000 = CTED7 pin(1)
0111 = CTED6 pin
0110 = CTED5 pin
0101 = CTED4 pin
0100 = CTED3 pin(1)
0011 = CTED1 pin
0010 = CTED2 pin
0001 = OC1
0000 = Timer1 match

bit 9 **EDG2STAT:** Edge 2 Status bit
Indicates the status of Edge 2 and can be written to control current source.
1 = Edge 2 has occurred
0 = Edge 2 has not occurred

bit 8 **EDG1STAT:** Edge 1 Status bit
Indicates the status of Edge 1 and can be written to control current source.
1 = Edge 1 has occurred
0 = Edge 1 has not occurred

bit 7 **EDG2MOD:** Edge 2 Edge-Sensitive Select bit
1 = Input is edge-sensitive
0 = Input is level-sensitive

bit 6 **EDG2POL:** Edge 2 Polarity Select bit
1 = Edge 2 is programmed for a positive edge response
0 = Edge 2 is programmed for a negative edge response

Note 1: CTED3, CTED7, CTED10 and CTED11 are not available on 64-pin packages.

Register 2-1: CTMUCON1H: CTMU Control Register 1 High (Continued)

bit 5-2	**EDG2SEL<3:0>:** Edge 2 Source Select bits `1111` = CMP C3OUT `1110` = CMP C2OUT `1101` = CMP C1OUT `1100` = Peripheral clock `1011` = IC3 interrupt `1010` = IC2 interrupt `1001` = IC1 interrupt `1000` = CTED13 pin `0111` = CTED12 pin `0110` = CTED11 pin(1) `0101` = CTED10 pin(1) `0100` = CTED9 pin `0011` = CTED1 pin `0010` = CTED2 pin `0001` = OC1 `0000` = Timer1 match
bit 1	**Unimplemented:** Read as ‘`0`’
bit 0	**IRNGH:** High-Current Range Select bit Current output is set by the IRNG<1:0> bits in the CTMUCON1L register. `1` = Uses the higher current ranges (550 µA-2.2 mA) `0` = Uses the lower current ranges (550 nA-50 µA)

Note 1: CTED3, CTED7, CTED10 and CTED11 are not available on 64-pin packages.

Register 2-2: CTMUCON1L: CTMU Control Register 1 Low

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
ITRIM5	ITRIM4	ITRIM3	ITRIM2	ITRIM1	ITRIM0	IRNG1[2]	IRNG0[2]
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15 **CTMUEN:** CTMU Enable bit
1 = Module is enabled
0 = Module is disabled

bit 14 **Unimplemented:** Read as ‘0’

bit 13 **CTMUSIDL:** CTMU Stop in Idle Mode bit
1 = Discontinues module operation when device enters Idle mode
0 = Continues module operation in Idle mode

bit 12 **TGEN:** Time Generation Enable bit
1 = Enables edge delay generation and routes the current source to the comparator pin
0 = Disables edge delay generation and routes the current source to the selected ADC input pin

bit 11 **EDGEN:** Edge Enable bit
1 = Edges are not blocked
0 = Edges are blocked

bit 10 **EDGSEQEN:** Edge Sequence Enable bit
1 = Edge 1 event must occur before Edge 2 event can occur
0 = No edge sequence is needed

bit 9 **IDISSEN:** Analog Current Source Control bit
1 = Analog current source output is grounded
0 = Analog current source output is not grounded

bit 8 **CTTRIG:** CTMU Trigger Control bit
1 = Trigger output is enabled
0 = Trigger output is disabled

Note 1: Step-size is device dependent. Refer to the device data sheet for values.
2: All ranges are not available on all devices. Refer to the particular device data sheet for specific range availability.

Register 2-2: CTMUCON1L: CTMU Control Register 1 Low (Continued)

bit 7-2	**ITRIM<5:0>:** Current Source Trim bits
	`011111` = Maximum positive change from nominal current (+ 62% nominal)[1]
	`011110`
	•
	•
	•
	`000001` = Minimum positive change from nominal current (+ 2% nominal)
	`000000` = Nominal current output specified by IRNG<1:0>
	`111111` = Minimum negative change from nominal current (– 2% nominal)
	•
	•
	•
	`100010`
	`100001` = Maximum negative change from nominal current (– 64% nominal)[1]

bit 1-0	**IRNG<1:0>:** Current Source Range Select bits[2] If IRNGH = `0`: `11` = 55 µA range `10` = 5.5 µA range `01` = 550 nA range `00` = 550 µA range If IRNGH = `1`: `11` = Reserved; do not use `10` = Reserved; do not use `01` = 2.2 mA range `00` = 550 µA range

Note 1: Step-size is device dependent. Refer to the device data sheet for values.

2: All ranges are not available on all devices. Refer to the particular device data sheet for specific range availability.

Register 2-3: CTMUCON2L: CTMU Control Register 2 Low

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	IRSTEN(1)	—	DSCHS2(1,2)	DSCHS1(1,2)	DSCHS0(1,2)
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-5 **Unimplemented:** Read as ‘0’

bit 4 **IRSTEN:** CTMU Current Source Reset Enable bit(1)

1 = Signal selected by the DSCHS<2:0> bits or the IDISSEN control bit will reset CTMU edge detect logic
0 = CTMU edge detect logic must be reset by software

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **DSCHS<2:0>:** Discharge Source Select bits(1,2)

111 = CLC2 out
110 = CLC1 out
101 = Disabled
100 = ADC end of conversion
011 = MCCP3 auxiliary output
010 = MCCP2 auxiliary output
001 = MCCP1 auxiliary output
000 = Disabled

Note 1: This feature is not available on all devices. Refer to the device data sheet for availability.
2: Discharge sources are device-dependent. Refer to the device data sheet for available sources.

3.0 CTMU OPERATION

The CTMU works by using a fixed current source to charge a circuit. The type of circuit depends on the type of measurement being made. In the case of charge capacitive measurement, the current is fixed and the amount of time the current is applied to the circuit is fixed. The voltage level read by the ADC is then a measurement of the capacitance of the circuit. In the case of time measurement, the current, as well as the capacitance of the circuit, is fixed. The voltage read by the ADC is then representative of the amount of time elapsed from the time the current source starts and stops charging the circuit.

If the CTMU is being used as a time delay, both capacitance and current source are fixed, as well as the reference voltage supplied to the comparator circuit. The delay of a signal is determined by the amount of time it takes to charge the circuit to the comparator threshold voltage.

3.1 Theory of Operation

The operation of the CTMU is based on the equation for charge, as shown in Equation 3-1.

Equation 3-1:

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

More simply, the amount of charge measured in coulombs in a circuit is defined as current in amperes (I) multiplied by the amount of time in seconds that the current flows (t). Charge is also defined as the capacitance in farads (C) multiplied by the voltage of the circuit (V), as shown in Equation 3-2.

Equation 3-2:

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

The CTMU module provides a constant, known current source. The A/D Converter is used to measure (V) in the equation, leaving two unknowns: capacitance (C) and time (t). Equation 3-2 can be used to calculate capacitance or time, by either the relationship shown in Equation 3-3 and using the known fixed capacitance of the circuit, or by Equation 3-4 using a fixed time that the current source is applied to the circuit.

Equation 3-3:

$$t = \frac{(C \cdot V)}{I}$$

Equation 3-4:

$$C = \frac{(I \cdot t)}{V}$$

3.2 Current Source

At the heart of the CTMU is a precision calibratable current source, designed to provide a constant reference for measurements. The amount of current is user-selectable across four ranges, or a total of three orders of magnitude, with the ability to trim the output in ±2% increments (nominal). The current range is selected by the IRNG<1:0> bits (CTMUCON1L<1:0>).

Current trim is provided by the 2's complement signed ITRIM<5:0> bits (CTMUCON1L<7:2>). These six bits allow trimming of the current source in steps of approximately 2% per step. A value of '`000000`' is the middle value (no change). A value of '`100000`' is the maximum negative adjustment (approximately -64%) and '`011111`' is the maximum positive adjustment (approximately +62%).

3.3 Edge/Level Selection and Control

CTMU measurements are controlled by the edge or level events occurring on the module's two input channels. Each channel, referred to as Edge 1 and Edge 2, can be configured to receive input pulses from one of the sixteen edge input pins. The inputs are selected using the EDG1SEL<3:0> and EDG2SEL<3:0> bit pairs (CTMUCON1H<13:10> and <5:2>). Further, the mode of the input sources to the Edge 1 and Edge 2 can either be level-sensitive or edge-sensitive, which is selected using the EDG1MOD bit (CTMUCON1H<15>).

In addition to source, each channel can be configured for event polarity using the EDG1POL and EDG2POL bits (CTMUCON1H<14,6>). The input channels can also be filtered for an edge event sequence (Edge 1 occurring before Edge 2) by setting the EDGSEQEN bit (CTMUCON1L<10>).

3.4 Edge Status

The CTMUCON1H register also contains two status bits: EDG2STAT and EDG1STAT (CTMUCON1H<9:8>). Their primary function is to show if an edge response has occurred on the corresponding channel. The CTMU automatically sets a particular bit when an edge response is detected on its channel. The level-sensitive, or edge-sensitive, nature of the input channels also means that the status bits become set immediately if the channel's configuration is changed and is the same as the channel's current state.

The module uses the edge status bits to control the current source output to external analog modules (such as the A/D Converter). Current is only supplied to external modules when EDG1STAT is not equal to EDG2STAT and shuts current off when EDG1STAT is equal to EDG2STAT. This allows the CTMU to measure current only during the interval between edges. After both status bits are set, it is necessary to clear them before another measurement is taken. Both bits should be cleared simultaneously to avoid re-enabling the CTMU current source.

In addition to being set by the CTMU hardware, the edge status bits can also be set by software. This allows the user's application to manually enable or disable the current source. Setting either one (but not both) of the bits enables the current source. Setting or clearing both bits at once disables the source.

3.5 Interrupts

The CTMU sets its interrupt flag whenever the current source is enabled and disabled. The CTMU interrupt enable bit must also be set to generate interrupts. If edge sequencing is not enabled, Edge 1 must occur before Edge 2 to generate an interrupt. If edge sequencing is not enabled, it is necessary to monitor the edge status bits and determine which edge occurred last and caused the interrupt.

4.0 CALIBRATING THE CTMU MODULE

The CTMU requires calibration for precise measurements of capacitance and time, as well as for generating an accurate time delay. If the application only requires measurement of a relative change in resistance, capacitance or time, calibration is usually not necessary. These applications include capacitive touch switching, in which the touch circuit has a baseline capacitance and the added capacitance of the human body changes the overall capacitance of a circuit. If actual resistance, capacitance or time measurement is required, calibration is required.

4.1 Current Source Calibration

This calibration consists of measuring the CTMU current source generated voltage drop across a known value resistor (RCAL) using the ADC. The CTMU trim is adjusted until the ADC measured value is within the desired tolerance. The value for RCAL depends on the current source range (refer to Table 4-1 for suggested RCAL values). The RCAL value can be also be calculated using Equation 4-1. After the RCAL value has been determined, the expected ADC counts can be calculated using Equation 4-2. An example calculation is provided in Example 4-1.

There are two current calibrations; the appropriate one depends on the current range used in the application. For current source ranges of 550 µA or 5.5 µA, the Low-Current mode calibration is appropriate; it can typically be performed on any analog input pin. For the higher CTMU ranges, the internal resistances have a significant effect on the calibration, therefore, the High-Current mode calibration is appropriate. The High-Current mode calibration is typically limited to a single device pin with a dedicated connection to the CTMU (refer to the specific device data sheet for more information).

Equation 4-1: Current Source RCAL Value Calculation

$$R_{CAL} = \frac{ADCV_{REF} * 70\%^{(1)}}{CTMU\ Range}$$

Note 1: Refer to **Section 4.3 "Calibration and Measurement Considerations"**.

Equation 4-2: Current Source Expected ADC Counts Calibration

$$Expected\ Counts = \frac{R_{CAL} * CTMU\ Range * (2\text{^}ADC\ Bits)}{ADCV_{REF}}$$

Example 4-1: Current Source Calibration

ADCVREF = 3.3V, 10-Bit ADC Mode, CTMU Range is 0.550 µA:

$$R_{CAL} = \frac{3.3V * 70\%}{0.550\ \mu A} = 4.22\ MOhms \quad Common\ 1\%\ Resistor\ 4.22\ MOhms$$

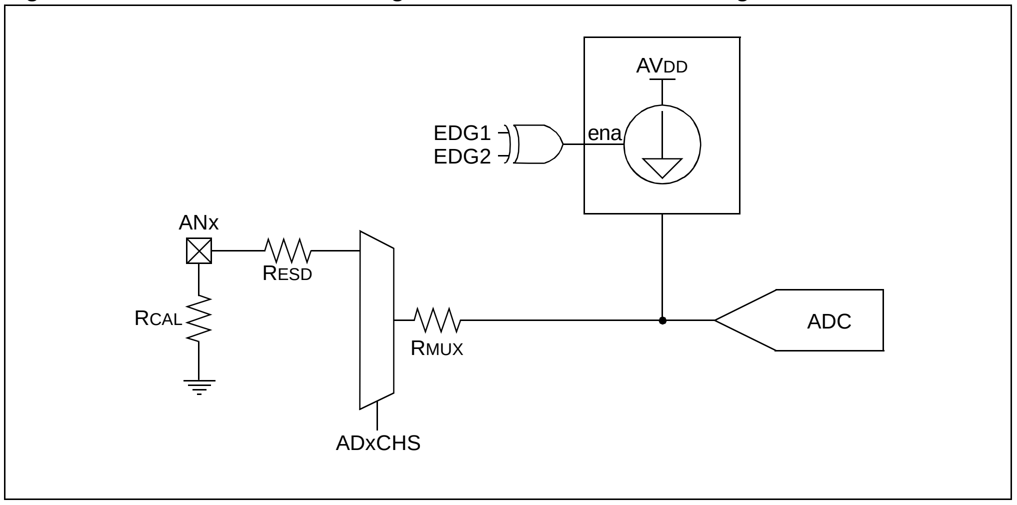
$$Expected\ Counts = \frac{4.22\ MOhms * 0.550\ \mu A * 1023}{3.3V} = 720\ Counts$$

4.1.1 LOW-CURRENT RANGE MODE CALIBRATION

This calibration is only suitable for the 550 µA and 5.5 µA ranges. The large ratio of RCAL to RESD and RMUX allows the RESD and RMUX values to be ignored to simplify the calculations (refer to Figure 4-1). The calibration process consists of enabling the current source and sampling the voltage generated across RCAL. This voltage is then divided by the known resistor value to calculate the CTMU current. The trim is then iteratively adjusted until the desired calibration is achieved. To configure the device for current calibration, perform the following steps (refer to Example 4-2).

1. Configure the CTMU for current generation (TGEN = 0).
2. Configure the I/O port pin as an input and enable Analog mode.
3. Configure the ADC for Manual mode.
4. Enable the CTMU.
5. Enable the current source by setting the EDG1STAT bit.
6. Start sampling.
7. Wait approximately 1500 µs for the sample capacitor to charge.
8. Convert the analog sample.
9. Disable the CTMU.
10. Repeat ‘n’ times, accumulating the values and averaging the result.
11. Calculate the current.
12. Iteratively adjust the trim value until the result is the desired CTMU current.

Figure 4-1: Low-Current Range Mode Calibration Block Diagram



Example 4-2: Low-Current Range Mode Calibration

```
#include   "p24FJ1024GB610.h"
#define    CTMU_MODE_EDGE 0
#define    RANGE_0_550uA 1    // .550uA
#define    RCAL 4.22e6        // R value is 4.22M
#define    ADSCALE 1023       //for 10-bit ADC
#define    ADREF 3.3          //Vdd connected to ADC Vref+

unsigned   int CtmuCurrentCalConfig(unsigned int mode, unsigned int range, signed int trim)
{
    unsigned   int result, x;
    // Step 1 Configure the CTMU
    CTMUCON1L = 0x0000;                  // Disable CTMU
    CTMUCON1Lbits.TGEN = mode;           // Enable/Disable Time Generation mode
    CTMUCON1Lbits.EDGEN = 0;             // Edges are disabled
    CTMUCON1Lbits.IDISSEN = 0;           // Current source is not grounded
    CTMUCON1Lbits.ITRIM = trim;          // Set trim
    CTMUCON1Lbits.CTTRIG = 0;            // Trigger output disabled
    CTMUCON1Lbits.IRNG = (range & 3);    // Set range
    CTMUCON1H = 0;                       // Edges are disabled, edge controls ignored
    // Next line does not apply to all devices
    CTMUCON1Hbits.IRNGH = (range>>2);    // set high bit of range
    CTMUCON2Lbits.IRSTEN = 0;            // Current source reset disabled
    CTMUCON2Lbits.DSCHS = 0;             // Discharge source disabled

    // Step 2 Configure the GPIO Port
    TRISB = TRISB | (1<<2);              // Set channel 2
    ANSBbits.ANSB2 = 1;                  // Make AN2 as analog (Resistor is connected to this pin)

    // Step 3 configure the ADC
    AD1CHSbits.CH0SA = 2;                // Select the analog channel 2
    AD1CON1 = 0x8000;                    // Turn On A/D Converter,
    // Unsigned fractional format, Clear SAMP bit to
    // start conversion, Sample when SAMP bit is set
    AD1CON2 = 0x0000;                    // VR+ = AVDD, V- = AVSS, Don't scan,
    AD1CON3 = 0x0000;                    // ADC uses system clock
    AD1CON3bits.ADCS = 0;                // conversion clock = 1xTcy
    AD1CON5 = 0x0000;                    // Auto-Scan disabled

    //  Step 4 - 6 Enable the current source and start sampling
    CTMUCON1Lbits.CTMUEN = 1;            // Enable the CTMU
    CTMUCON1Hbits.EDG1STAT = 1;          // Enable current source
    AD1CON1bits.SAMP = 1;                // Manual sampling start

    // step 7  ~3000 us delay to charge sample cap
    for (x = 0; x < 2000; x++);          // ~6 cycles * 2000 ,Fcy = 4Mhz

    // step 8 Convert the sample
    AD1CON1bits.SAMP = 0;                // Begin A/D conversion
    while(AD1CON1bits.DONE == 0);        // Wait for A/D convert complete

    // Step 9 Disable the CTMU
    CTMUCON1Hbits.EDG1STAT = 0;          // Disable current source
    IFS0bits.AD1IF = 0;                  // Clear ADC interrupt flag
    CTMUCON1Lbits.CTMUEN = 0;            // Disable the CTMU
    result = ADC1BUF0;
    return (result);                     // return accumulated result
}
```

Example 4-2: Low-Current Range Mode Calibration (Continued)

```
#define    ITT 10                                                      // 10 iterations

int        main(void)
{
    float cntsAvg, vCal, cntsTot = 0, ctmuISrc = 0, result = 0;

    // Step 10 perform calibration 10 times
    for(x = 0; x < ITT; x++)
    {
      result = (float)(CtmuCurrentCalConfig(CTMU_MODE_EDGE, RANGE_0_550uA, 0));// 0.550uA , no trim
      cntsTot +=  (float)result;
    }

    // Step 11 calculate the result
    cntsAvg = (cntsTot / ITT);                                         // Average of 10 readings
    vCal = (cntsAvg / ADSCALE * ADREF);
    ctmuISrc = vCal / RCAL;                                            // CTMU current in uA

    // step 12
    // user code to perform iteration for calibration

    while(1);
}
```

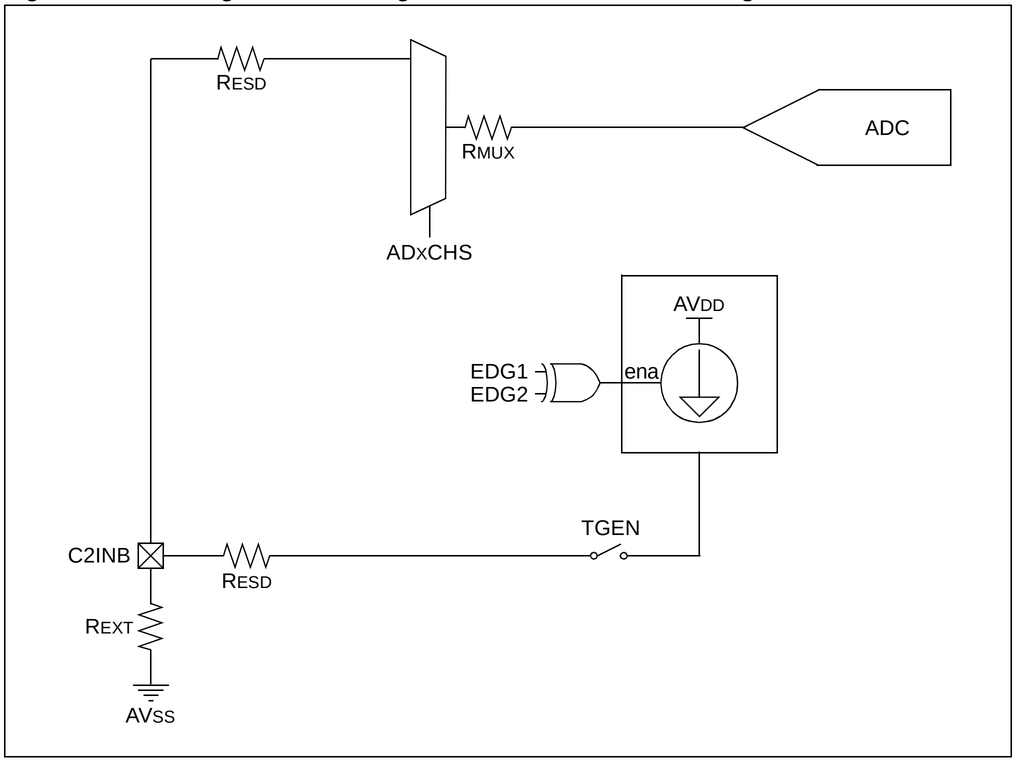
4.1.2 HIGH-CURRENT MODE CALIBRATION

High-Current mode calibration uses a dedicated connection to a special device pin to remove the RMUX voltage offset seen in the Low-Current mode calibration method. These offsets become a significant source of error for the higher current ranges. While this method is intended for the high-current ranges, it can be used to improve the calibration accuracy on the low-current ranges. This method typically achieves a calibration tolerance of less than 3% on all ranges (refer to Figure 4-2). The calibration process consists of enabling the current source and sampling the voltage generated across RCAL. This voltage is then divided by the known resistor value to calculate the CTMU current. The trim is then iteratively adjusted until the desired calibration is achieved. To configure the device for current calibration, perform the following steps (refer to Example 4-3).

1. Configure the CTMU for Time Generation mode (TGEN = 1).
2. Configure the I/O port pin as an input and enable Analog mode.
3. Configure the ADC for Manual mode.
4. Enable the CTMU.
5. Enable the current source by setting the EDG1STAT bit.
6. Start sampling.
7. Wait approximately 1500 µs for the sample capacitor to charge.
8. Convert the analog sample.
9. Disable the CTMU.
10. Repeat ‘n’ times, accumulating the values and averaging the result.
11. Calculate the current.
12. Iteratively adjust the trim value until the result is the desired CTMU current.

Refer to Example 4-3 and Equation 4-1.

Figure 4-2: High-Current Range Mode Calibration Block Diagram



Example 4-3: High-Current Range Mode Calibration

```
#include   "p24FJ1024GB610.h"
#define    _MODE_TGEN 1
#define    RANGE_550uA   0    // 550uA
#define    RCAL 4.22e6        // R value is 4200000 (4.22M)
#define    ADSCALE 1023       //for 10-bit ADC
#define    ADREF 3.3          //Vdd connected to ADC Vr+

unsigned   int CtmuCurrentCalConfig(unsigned int mode, unsigned int range, signed int trim)
{
    unsigned int     result, x;
    // Step 1 Configure the CTMU
    CTMUCON1L = 0x0000;                   // Disable CTMU
    CTMUCON1Lbits.TGEN = mode;            // Enable/Disable Time Generation mode
    CTMUCON1Lbits.EDGEN = 0;              // Edges are disabled
    CTMUCON1Lbits.IDISSEN = 0;            // Current source is not grounded
    CTMUCON1Lbits.ITRIM = trim;           // Set trim
    CTMUCON1Lbits.CTTRIG = 0;             // Trigger output disabled
    CTMUCON1Lbits.IRNG = (range & 3);     // Set range

    CTMUCON1H = 0;                        // Edges are disabled, edge controls ignored
    // Next line does not apply to all devices
    CTMUCON1Hbits.IRNGH = (range>>2);     // set high bit of range

    CTMUCON2Lbits.IRSTEN = 0;             // Current source reset disabled
    CTMUCON2Lbits.DSCHS = 0;              // Discharge source disabled

    // Step 2 Configure the GPIO Port
    TRISB = TRISB | (1<<2);               // Set channel 2
    ANSBbits.ANSB14 = 1;                  // Make AN2 as analog (Resistor is connected to this pin)

    // Step 3 configure the ADC
    AD1CHSbits.CH0SA = 2;                 // Select the analog channel(2)
    AD1CON1 = 0x8000;                     // Turn On A/D Converter,
    // Unsigned fractional format, Clear SAMP bit to
    // start conversion, Sample when SAMP bit is set
    AD1CON2 = 0x0000;                     // VR+ = AVDD, V- = AVSS, Don't scan,
    AD1CON3 = 0x0000;                     // ADC uses system clock
    AD1CON3bits.ADCS = 0;                 // conversion clock = 1xTcy
    AD1CON5 = 0x0000;                     // Auto-Scan disabled

    //  Step 4 - 6 Enable the current source and start sampling
    CTMUCON1Lbits.CTMUEN = 1;             // Enable the CTMU
    CTMUCON1Hbits.EDG1STAT = 1;           // Enable current source
    AD1CON1bits.SAMP = 1;                 // Manual sampling start

    /// step 7  ~3000 us delay to charge sample cap
    for (x = 0; x < 2000; x++);           // ~6 cycles * 2000 ,Fcy = 4Mhz

    // step 8 Convert the sample
    AD1CON1bits.SAMP = 0;                 // Begin A/D conversion
    while(AD1CON1bits.DONE == 0);         // Wait for A/D convert complete

    // Step 9 Disable the CTMU
    CTMUCON1Hbits.EDG1STAT = 0;           // Disable current source
    IFS0bits.AD1IF = 0;                   // Clear ADC interrupt flag
    CTMUCON1Lbits.CTMUEN = 0;             // Disable the CTMU
    result = ADC1BUF0;
    return (result);                      // return accumulated result
}
```

Example 4-3: High-Current Range Mode Calibration (Continued)

```
#define ITT 10                                                              // 10 iterations

int     main(void)
{
    float cntsAvg, vCal, cntsTot = 0, ctmuISrc = 0, result;

    // Step 10 perform calibration 10 times
    for(x = 0; x < ITT; x++)
    {
        result = (float)(CtmuCurrentCalConfig(_MODE_TGEN, RANGE_550uA, 0));  // 550uA, no trim

        cntsTot += result;
    }
    // Step 11 calculate the result
    cntsAvg = (cntsTot / ITT);                                              // Average of 10 readings
    vCal = (cntsAvg / ADSCALE * ADREF);
    ctmuISrc = vCal / RCAL;                                                 // CTMU current in uA

    // step 12
    // user code to perform iteration for calibration

    while(1);
}
```

Table 4-1: Suggested Calibration (RCAL) Values

Range	RCAL Value (Ohms)
0.550 µA	4.22 MOhms
5.5 µA	422 kOhms
55 µA	42.2 kOhms
550 µA	4.2 kOhms
2.2 mA	1 kOhms

4.2 Capacitance Calibration

The capacitance calibration consists of measuring the system capacitance without the load to be measured. The capacitance is calculated using the charge time, current and resulting voltage; where current is known from the current source measurement step, t is a known fixed delay and V is measured by performing an ADC conversion. This measured value, the offset, can then be stored and subtracted from calculations of time measurement or capacitance measurement. For calibration, the time delay can be approximated if the approximate values of C_{STRAY} and C_{ADC} are known (refer to Equation 4-3 and Equation 4-4). Please refer to the respective device data sheet for the value of C_{ADC}.

For capacitance calibration, the current source is first calibrated using a procedure from **Section 4.1 "Current Source Calibration"**. The calibration resistor is then removed or another analog channel is used for the capacitance calibration. The calibrated current source is then enabled, while the ADC is sampling, to charge the ADC sample capacitor and the system capacitance. The ADC conversion is started after a precise delay. The ADC conversion result and the delay period are then used to calculate the system capacitance (refer to Equation 4-5 and Example 4-4). This value can then be subtracted from future measurements to account for the system capacitance. To minimize the effect of noise and use the linear operating range of the current source, the delay period should be chosen or adjusted such that the measured voltage is approximately 70% of AV_{DD}. Refer to **Section 4.3 "Calibration and Measurement Considerations"**.

Equation 4-3: Delay Time Calculation

$$Time = \frac{C_{TOTAL} * V_{DESIRED}}{I}$$

Equation 4-4: Delay Calculation with Known C_{STRAY} and C_{ADC} Values

$$(4\ pF + 11\ pF) * 2.31V/0.55\ \mu A = 63\ \mu s\ Delay$$

Equation 4-5: System Capacitance Calculations

$$C_{OFFSET} = C_{SYSTEM} = C_{ADC} + C_{STRAY} = (I * t)/V$$

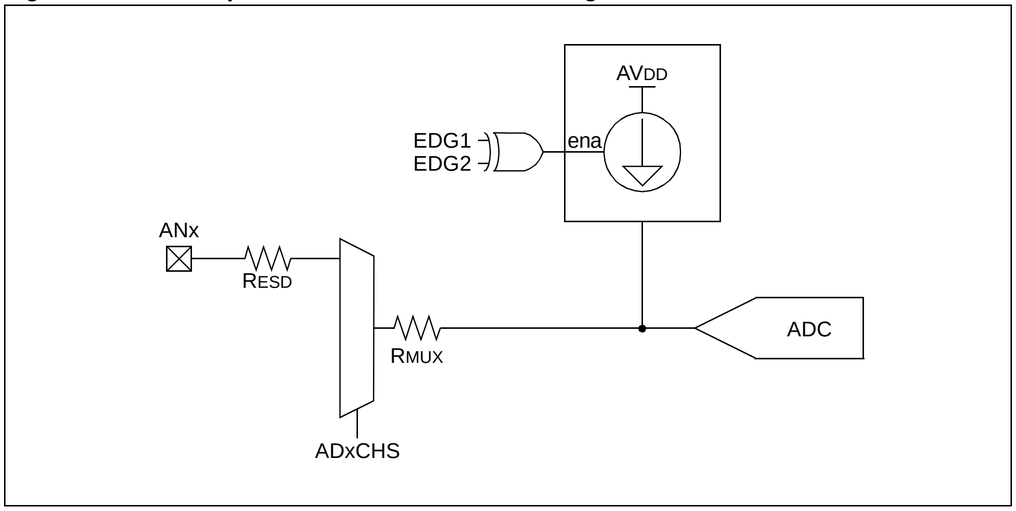
Example 4-4: Calculating the System Capacitance

Delay = 2.25 ms
CTMU Range = 0.55 µA
ADC Sample = 670 Counts (cnts) (ADC in 10-bit mode)
ADC Reference Voltage = 3.3V

Using Equation 4-5:

$$C = \frac{Current * Delay}{Voltage} = \frac{0.55\ \mu A * 2.25\ ms}{3.3V * (670\ cnts/1023\ cnts)} = 573\ pF$$

Figure 4-3: Capacitance Calibration Block Diagram



Capacitance calibration can be performed using the configuration from the Current mode calibration with the RCAL resistor disconnected or using a different pin for calibrations and measurements. To configure the device for capacitance calibration, perform the following steps:

1. Perform current calibration (refer to **Section 4.1 "Current Source Calibration"**).
2. Configure the CTMU for current generation (TGEN = `0`).
3. Configure the I/O port pin as an input and enable Analog mode.
4. Configure the ADC for Manual mode.
5. Enable the CTMU.
6. Enable the current source by setting the EDG1STAT bit.
7. Start sampling.
8. Discharge the internal charge by shorting the current source to ground.
9. Delay approximately 1200 µs to allow the internal circuit to discharge.
10. Disconnect the current source from ground.
11. Wait a predetermined time for the sample capacitor to charge.
12. Convert the analog sample.
13. Disable the CTMU.
14. Repeat 10 times, accumulating the values and averaging the result.
15. Calculate the capacitance using Equation 4-2.

Example 4-5: Absolute Capacitance Calibration and Measurement

```
#include    "p24FJ1024GB610.h"
#define     CTMU_MODE_EDGE 0
#define     RANGE_0_550uA 1    // .550uA
#define     ADSCALE 1023       //for 10-bit ADC
#define     ADREF 3.3          //Vdd connected to ADC Vr+
#define     DELAY_LOOPS 150    // delay = 6 cycles * DELAY_LOOPS * (1/4 MHz)  Fosc = 8MHz

unsigned    int CtmuCapMeasureConfig(unsigned int mode, unsigned int range, signed int trim)
{
    unsigned int result, x;
    // Step 1 Configure the CTMU
    CTMUCON1L = 0x0000;                    // Disable CTMU
    CTMUCON1Lbits.TGEN = mode;             // Enable/Disable Time Generation mode
    CTMUCON1Lbits.EDGEN = 0;               // Edges are disabled
    CTMUCON1Lbits.IDISSEN = 1;             // Current source is grounded (discharge enabled)
    CTMUCON1Lbits.ITRIM = trim;            // Set trim
    CTMUCON1Lbits.CTTRIG = 0;              // Trigger output disabled
    CTMUCON1Lbits.IRNG = (range & 3);      // Set range

    CTMUCON1H = 0;                         // Edges are disabled, edge controls ignored
    // Next line does not apply to all devices
    CTMUCON1Hbits.IRNGH = (range>>2);      // set high bit of range

    CTMUCON2Lbits.IRSTEN = 0;              // Current source reset disabled
    CTMUCON2Lbits.DSCHS = 1;               // Discharge source enabled

    // Step 2 Configure the port Ports
    TRISB = TRISB | (1<<2);                // Set channel 2
    ANSBbits.ANSB14 = 1;                   // Make AN2 as analog

    // Step 3 configure the ADC
    AD1CHSbits.CH0SA = 2;

    AD1CON1 = 0x8000;                      // Turn On A/D Converter,
    // Unsigned fractional format, Clear SAMP bit to
    // start conversion, Sample when SAMP bit is set
    AD1CON2 = 0x0000;                      // VR+ = AVDD, V- = AVSS, Don't scan,
    AD1CON3 = 0x0000;                      // ADC uses system clock
    AD1CON3bits.ADCS = 0;                  // conversion clock = 1xTcy
    AD1CON5 = 0x0000;                      // Auto-Scan disabled

    //  Step 4 - 6 Enable the current source and start sampling
    CTMUCON1Lbits.CTMUEN = 1;              // Enable the CTMU
    CTMUCON1Hbits.EDG1STAT = 1;            // Enable current source
    AD1CON1bits.SAMP = 1;                  // Manual sampling start

    // step 7  1500us delay to discharge sample cap
    for (x = 0; x < 5000; x++);            // ~6 cycles * 5000

    // step 9 disable discharge
    CTMUCON1Lbits.IDISSEN = 0;             // Discharge disabled

    // step 10 delay to charge sample cap
    for (x = 0; x < DELAY_LOOPS; x++);     // 6 clocks per loop iteration

    // step 11 convert the sample
    AD1CON1bits.SAMP = 0;                  // Begin A/D conversion
    while(AD1CON1bits.DONE == 0);          // Wait for A/D convert complete
```

Example 4-5: Absolute Capacitance Calibration and Measurement (Continued)

```
    // Step 12 disable the CTMU
    CTMUCON1Hbits.EDG1STAT = 0;           // Disable current source
    IFS0bits.AD1IF = 0;                   // Clear ADC interrupt flag
    CTMUCON1Lbits.CTMUEN = 0;             // Disable the CTMU
    result = ADC1BUF0;

    return (result);                      // return accumulated result
}

#define ITT 10                            // 10 iterations

int     main(void)
{
    float cntsAvg, cntsTot = 0, c;
    unsigned int x;

    // Step 10 perform measurement 10 times
    for(x = 0; x < ITT; x++)
    {
    cntsTot += (float)(CtmuCapMeasureConfig(CTMU_MODE_EDGE, RANGE_0_550uA, 0));    // .550uA, no trim
     }
    // Step 11 calculate the result
    cntsAvg = (cntsTot / ITT);            // Average of 10 readings

    c = (.550e-6 * DELAY_LOOPS * 6 * (1 / 4e6)) / (3.3 * cntsAvg / ADSCALE);     // capacitance in Farads
    // step 12
    // user code to perform iteration for calibration

    while(1);
}
```

4.3 Calibration and Measurement Considerations

The following should be considered when performing calibrations:

1. The actual value of the calibration resistor is not important, but it should meet the following requirements:
   - The generated voltage across the resistor should be between ½ and ¾ of the ADC full-scale value to minimize the noise effect on the measurement, and to stay in the linear range of the current source. The current source operates in a linear fashion when the developed load voltage is less than (AV_{DD} – 0.7V, typical).
   - The tolerance of the resistor is not important if the actual value of the resistor is known and that value is used in the calculations.
   - The chosen resistor value should be at least 100 times the internal resistance to minimize the offsets generated by the internal resistors.
   - For capacitance calibrations, the delay time should be chosen such that the ADC sample capacitor charges to between ½ and ¾ of the ADC range to maximize resolution and reduce the impact of noise.
2. The delay time for the current calibration should be sufficient to allow the ADC sample capacitor to charge to a steady-state value.
3. Measurement resolution may be increased by using an external ADC reference that is approximately ¾ of AV_{DD}.

4.4 Measuring Internal Resistance

The value of the internal resistances can be measured and used in calculations to improve the accuracy of calibrations and current measurements. The current source must first be calibrated. The device is then configured for current measurement, but the I/O pin is driven low. When the measurement is taken, the measured voltage is the drop across the internal resistance. This value can then be subtracted from future measurements to improve accuracy.

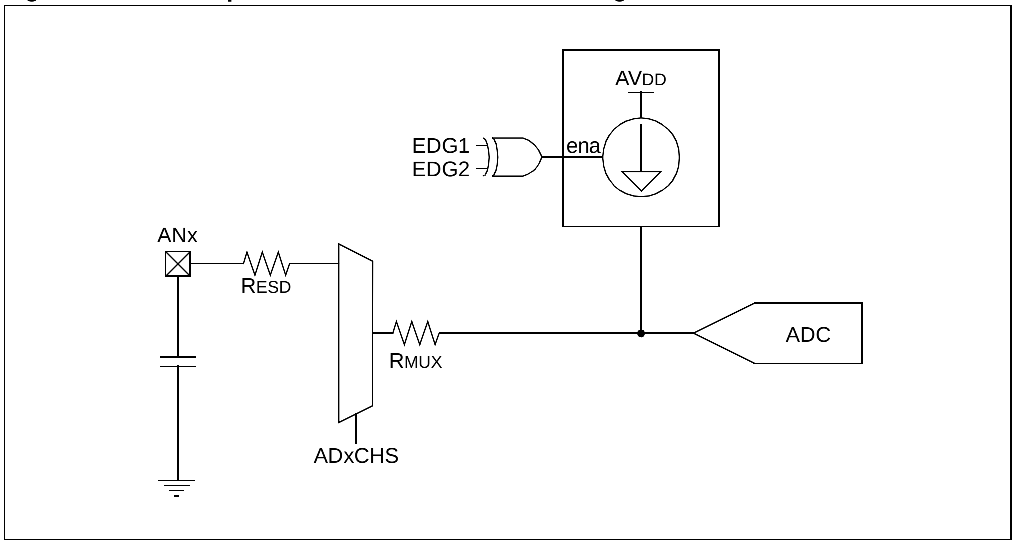
5.0 MEASURING CAPACITANCE WITH THE CTMU

There are two methods of measuring capacitance with the CTMU. The first is the absolute method, in which the actual capacitance value is desired. The second is the relative method, in which the actual capacitance value is not required, rather a detection of change in capacitance is desired.

5.1 Absolute Capacitance Measurement

For absolute capacitance measurements, both the current and capacitance calibration steps found in **Section 4.0 "Calibrating the CTMU Module"** should be followed. The process for absolute measurement is the same as the capacitance calibration process, but the load to be measured is connected during measurement. The capacitance value from the calibration step is then subtracted from the measured result to get the capacitance value. Refer to Example 4-5 for example code. If the capacitance to be measured is significantly larger than the system capacitance measured in the calibration step, the delay to charge the sample capacitor may have to be adjusted to allow the capacitor to charge to a measurable value or to provide the desired resolution in the result. The selected delay should not allow the resulting voltage to exceed 75% of the ADC reference voltage (refer to **Section 4.3 "Calibration and Measurement Considerations"**).

Figure 5-1: Capacitance Measurement Block Diagram



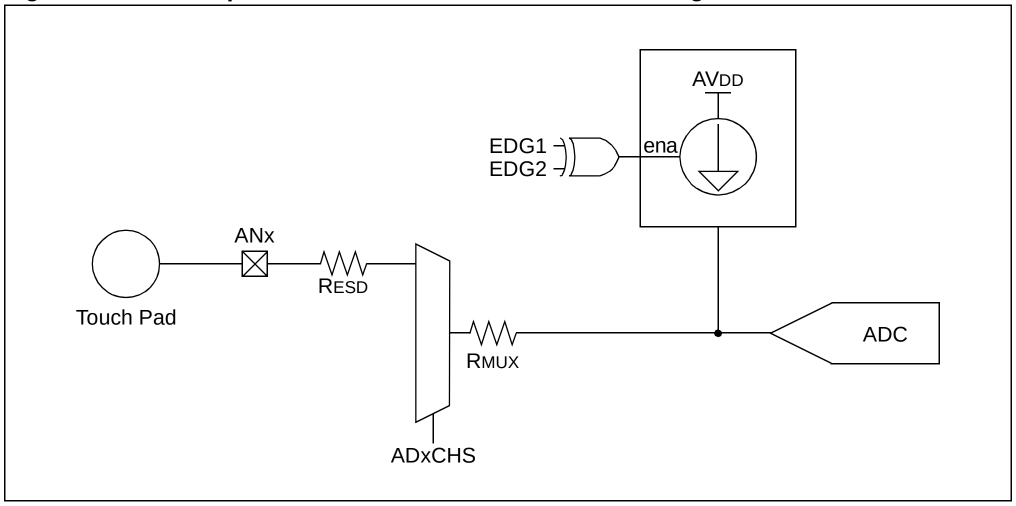
5.2 Relative Charge Measurement and Capacitive Touch Sense

Relative measurements may not require precise capacitance measurements. For example, when detecting a valid press of a capacitance-based switch, detecting a relative change of capacitance is of interest. In this type of application, when the button is not touched, the total capacitance is system capacitance (the PCB traces, the ADC, etc.). A larger voltage will be measured by the ADC due to the relatively smaller capacitance. When the button is pressed, the total capacitance is larger due to the addition of the capacitance of the human body to the above listed capacitances, and a smaller voltage will be measured by the ADC. Software then compares the non-touched values to the current ADC result to determine if a button has been touched. The threshold values used to determine a touch event are dependent on the system and the environment, and therefore, should be determined experimentally. Typical implementations do not require calibration to operate. ESD protection must be provided on capacitive touch pins (refer to **Section 5.4 "Electrostatic Discharge (ESD) Protection"**).

Detecting capacitance changes is accomplished with the following steps:

1. Configure the CTMU for current generation (TGEN = `0`).
2. Configure the I/O port pin as an input and enable Analog mode.
3. Configure the ADC for Manual mode.
4. Enable the CTMU.
5. Enable the current source by setting the EDG1STAT bit.
6. Start sampling.
7. Enable discharge circuit.
8. Delay to allow the internal circuit to discharge.
9. Disable the discharge circuit.
10. Wait a predetermined time for the sample capacitor to charge (delay).
11. Convert the analog sample.
12. Disable the current source.
13. Calculate the total capacitance using Equation 4-5. Refer to Example 5-1.
14. Subtract the system capacitance from the result (optional).
15. Compare the result to the system capacitance.
16. Repeat Steps 5-15 if averaging is required.

Figure 5-2: Capacitance Touch Measurement Block Diagram



Example 5-1: Capacitance Touch

```
#include   "p24FJ1024GB610.h"
#define    CTMU_MODE_EDGE 0
#define    RANGE_0_550uA 1    // .550uA
#define    CTMU_TOUCH_THRESHHOLD_OFFSET 100

void   CtmuCapTouchConfig(unsigned int mode, unsigned int range, signed int trim)
{
    // step 1 Configure the CTMU
    CTMUCON1L = 0x0000;                   // Disable CTMU
    CTMUCON1Lbits.TGEN = mode;            // Enable/Disable Time Generation mode
    CTMUCON1Lbits.EDGEN = 0;              // Edges are disabled
    CTMUCON1Lbits.ITRIM = trim;           // Set trim
    CTMUCON1Lbits.CTTRIG = 0;             // Trigger output disabled
    CTMUCON1Lbits.IRNG = (range & 3);     // Set range

    CTMUCON1H = 0;                        // Edges are disabled, edge controls ignored
    // This line does not apply to all devices
    CTMUCON1Hbits.IRNGH = (range>>2);     // set high bit of range

    CTMUCON2Lbits.IRSTEN = 0;             // Current source reset disabled
    CTMUCON2Lbits.DSCHS = 0;              // Discharge source disabled

    // Step 2 Configure the port Ports
    TRISB = TRISB | (1<<2);               // Set channel 2
    ANSBbits.ANSB2 = 1;                   // Make AN2 as analog

    // Step 3 configure the ADC
    AD1CHSbits.CH0SA = 2;                 // Select the analog channel(2)

    AD1CON1 = 0x8000;                     // Turn On A/D Converter,
    // Unsigned fractional format, Clear SAMP bit to
    // start conversion, Sample when SAMP bit is set
    AD1CON2 = 0x0000;                     // VR+ = AVDD, V- = AVSS, Don't scan,
    AD1CON3 = 0x0000;                     // ADC uses system clock
    AD1CON3bits.ADCS = 0;                 // conversion clock = 1xTcy
    AD1CON5 = 0x0000;                     // Auto-Scan disabled
    }


    unsigned   int    CtmuReturnSample(void)
{
        unsigned   int    result, x;

    //  Step 4 - 7 Enable the current source and start sampling
    CTMUCON1Lbits.CTMUEN = 1;             // Enable the CTMU
    CTMUCON1Hbits.EDG1STAT = 1;           // Enable current source
    CTMUCON1Lbits.IDISSEN = 1;            // Enable discharge
    AD1CON1bits.SAMP = 1;                 // Manual sampling start

    // step 8  1500us delay to discharge sample cap
    for (x = 0; x < 2000; x++);           // ~6 cycles * 2000

    // step 9 Disable the discharge circuit
    CTMUCON1Lbits.IDISSEN = 0;            // Disable discharge (start charging)

    // step 10 allow the sample cap to partially charge
    for (x = 0; x < 250; x++);            // ~6 cycles * 250 ~ 670 cnts

    // step 11 Convert the analog sample
    AD1CON1bits.SAMP = 0;                 // Begin A/D conversion
    while(AD1CON1bits.DONE == 0);         // Wait for A/D convert complete
```

Example 5-1: Capacitance Touch (Continued)

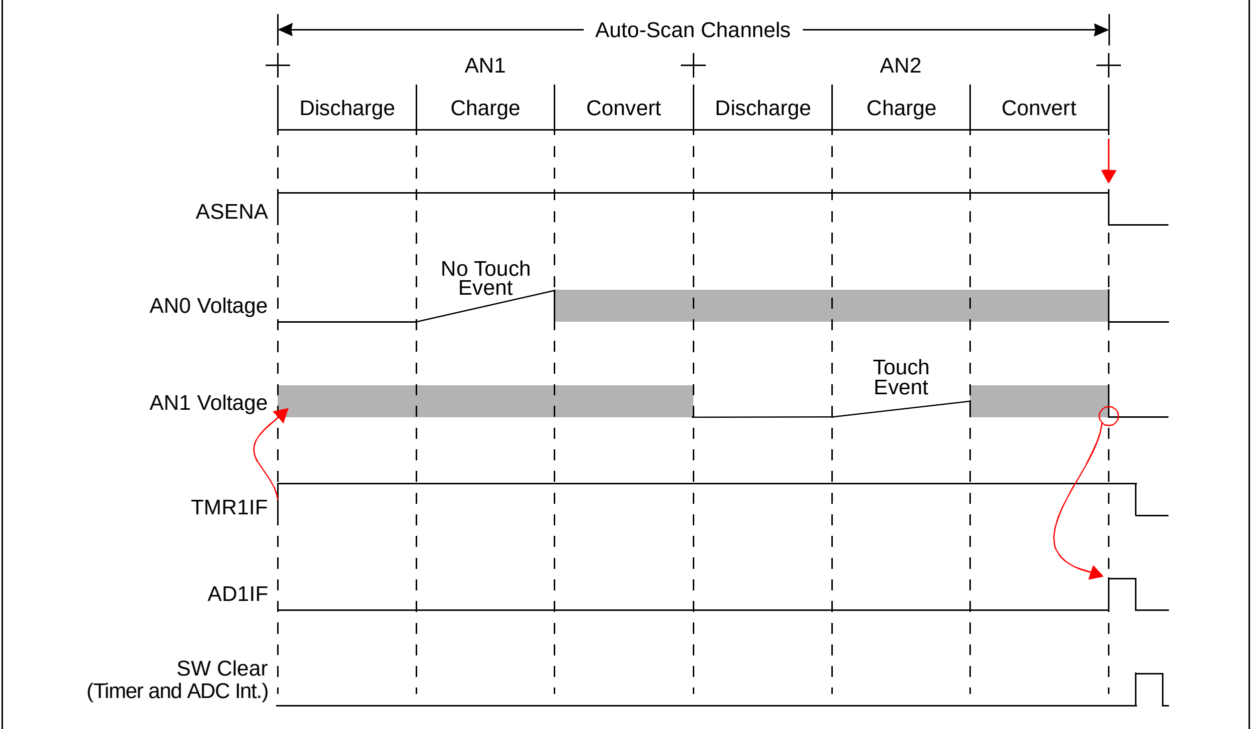
```
    // Step 12 Disable the current source
    CTMUCON1Hbits.EDG1STAT = 0;           // Disable current source
    IFS0bits.AD1IF = 0;                   // Clear ADC interrupt flag
    CTMUCON1Lbits.CTMUEN = 0;             // Disable the CTMU
    result = ADC1BUF0;

    return (result);
}

int     main(void)
{
    unsigned   int     untouched, sample;

    CtmuCapTouchConfig(CTMU_MODE_EDGE, RANGE_0_550uA, 0);

    untouched = CtmuReturnSample();       // get reference value

    while(1)
    {
      sample = CtmuReturnSample();

      // step 14-15 subtract the threshold and test
      if (sample < untouched - CTMU_TOUCH_THRESHHOLD_OFFSET)
      {
        // button was pressed
      }
      // user code
    }
}
```

5.3 Capacitance Touch Sense with Auto-Threshold Detect ADCs

The CTMU, when combined with a trigger source and an ADC with Auto-Threshold Detect, form a semi-autonomous relative charge detect sub-system that can be used for capacitive touch sense. The trigger source, such as an OCMP module, enables the CTMU current source for a predetermined period, charging the capacitor. The resulting voltage is determined by the capacitance of the system and additional capacitance of a touch. The ADC is configured to automatically convert a list of channels corresponding to the touch sensors. The Threshold Detect is configured with a predefined level, and therefore, the ADC only generates an interrupt when that threshold is met (a button is touched). See Figure 5-2. Figure 5-3 illustrates the timing relationships for 2 scanned channels. The ADC interrupt occurs after the lower voltage is detected on Channel 2. User code then clears the ADC interrupt. User code must also clear the timer interrupt. Calibration is not typically required with relative sense applications. ESD protection must be provided on capacitive touch pins (refer to **Section 5.4 "Electrostatic Discharge (ESD) Protection"**).

1. Configure the GPIO ports.
2. Configure the CTMU for External Edge Trigger Reset and ADC control of discharge.
3. Configure the ADC for Threshold Detect and auto-scan.
4. Configure a scan trigger (Timer1).
5. Wait for an interrupt (this can be implemented as an ISR).
6. Clear the interrupt flag.
7. Determine the channel that caused the interrupt.
8. Clear the appropriate Channel Hit flag.

Figure 5-3: Capacitance Touch Sense with Auto-Threshold Detect ADCs Timing Diagram



Example 5-2: Capacitance Touch Sense with Auto-Threshold Detect ADC

```
#define TOUCHED 100        // expected maximum ADC counts on a touch event

void   CtmuCapTouchThreshConfig(unsigned int range, signed int trim)
{
    // Step 1 Configure the port Ports
    TRISB = TRISB | (6);                  // Configure AN1 and AN2 as inputs
    ANSBbits.ANSB2 = 1;                   // Configure AN! and AN2 as analog
    // Step 2 configure the CTMU
    CTMUCON1L = 0x0000;                   // Disable CTMU
    CTMUCON1Lbits.TGEN = 0;               // Disable Time Generation mode
    CTMUCON1Lbits.EDGEN = 0;              // Edges are enabled???
    CTMUCON1Lbits.ITRIM = trim;           // Set trim
    CTMUCON1Lbits.CTTRIG = 1;             // Trigger output enabled
    CTMUCON1Lbits.IRNG = (range & 3);     // Set range
    // Next line does not apply to all devices
    CTMUCON1Hbits.IRNGH = (range>>2);     // set high bit of range

    CTMUCON2Lbits.IRSTEN = 1;             // enable CTMU status reset
    CTMUCON2Lbits.DSCHS = 4;              // end of ADC conversion resets CTMU status

    CTMUCON1Lbits.CTMUEN = 1;             // enable the CTMU
    CTMUCON1Hbits.EDG1STAT = 1;           // enable the current source

    // Step 3 configure the ADC
    IFS0bits.AD1IF = 0;                   // make sure ADC Int not set
    AD1CON1 = 0;                          // turn off ADC
    AD1CON1bits.SSRC = 5;                 // Timer1 IF starts autoscan sequence
    AD1CON1bits.ASAM = 1;                 // sampling begins automatically after last conversion
    AD1CON2bits.CSCNA = 1;                // enable scan mode
    AD1CON2bits.SMPI = 0;                 // interrupt after 1st event
    AD1CON3bits.ADCS = 2;                 // ADC clock is 1/2 sys clock  ???
    AD1CON5bits.ASEN = 1;                 // enable autoscan
    AD1CON3bits.SAMC = 31;                // set sample time in TADs
    AD1CON5bits.CTMREQ = 1;               // request the CTMU
    AD1CON5bits.BGREQ = 1;
    AD1CON5bits.ASINT = 3;                // interrupt after a threshold event compare
    AD1CON5bits.WM = 2;                   // do not write ADC result to buffer
    AD1CON5bits.CM = 0;                   // compare mode: less than threshold value
    AD1CTMENL = 4;                        // Connect CTMU current source to analog channel
    AD1CSSL = 6;                          // enable scan of AN2 & AN1

    ADC1BUF1 = TOUCHED;                   // Threshold value for A12
    ADC1BUF2 = TOUCHED;                   // Threshold value for AN2

    AD1CON1bits.ADON = 1;                 // Turn on the ADC

    // Step 4 configure the timer
    PR1 = 0xA000;                         // set rollover rate ~10ms to trigger ADC scan
    T1CONbits.TON = 1;                    // enable timer
}


#define RANGE_5_5uA   2    // 5.5uA
```

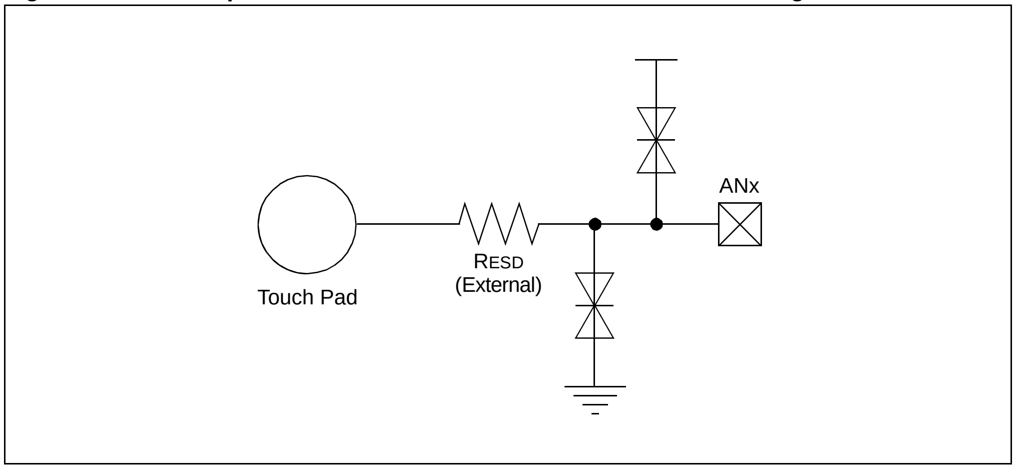
Example 5-2: Capacitance Touch Sense with Auto-Threshold Detect ADC (Continued)

```
int     main(void)
{
    CtmuCapTouchThreshConfig(RANGE_5_5uA, 15);

    TRISDbits.TRISD1 = 0;
    RPOR12 = 13;                        // map OCMP1 output to pin 76
    TRISDbits.TRISD1 = 0;               // make pin an output

    while(1)
    {
        if (IFS0bits.T1IF == 1)
        {
            IFS0bits.T1IF = 0;
        }

        LATDbits.LATD1 = 0;

        // step 5-8 wait for an ADC interrupt and service the event
        if (IFS0bits.AD1IF)             // Wait for ADC threshold match (this can an ISR)
        {
            LATDbits.LATD1 = 1;         // signal threshold event

            IFS0bits.AD1IF = 0;         // clear int
            //  Touch event handler
            if (AD1CHITL & 4)           // test to determine which ANx had a touch event
            {
                AD1CHITL &= ~4;         // clear the event
                // user code to handle touch event
            }
        }
    }
}
```

5.4 Electrostatic Discharge (ESD) Protection

When the CTMU is used for capacitive touch applications, ESD protection must be provided on the touch input pins to protect the microcontroller and supporting circuitry. The minimum ESD protection is a series resistor, typically 1-10 kOhms (see Figure 5-4). Additional protection in the form of TVS diodes to power and ground is recommended. The capacitance of the TVS diodes adds to the system capacitance for the analog channel being protected, therefore, low capacitance TVS diodes are recommended. The device power and ground traces should be sized to accommodate the additional current from an ESD event (refer to **Section 13.0 "Related Application Notes"**).

Figure 5-4: Capacitive Touch External ESD Protection Block Diagram



6.0 MEASURING TIME WITH THE CTMU MODULE

The CTMU, in conjunction with the ADC, can be used to precisely measure the time between 2 events. The events, internal or external, enable and disable the current source charging the ADC sample capacitor (refer to Figure 6-1 and Figure 6-2). The ADC then converts this voltage. An external capacitor can be used if capacitance greater than the ADC sample capacitor is required. If an external capacitor is used, firmware may need to discharge the capacitor prior to the next measurement. The CTMU current source and capacitance calibrations should be performed prior to measurements (refer to **Section 4.1 "Current Source Calibration"** and **Section 4.2 "Capacitance Calibration"**).

The time measured must be such that the current source charging the capacitance, C_{SYSTEM}, generates a voltage that is within the linear range of the valid current source. For the smallest time measurement, use the lowest current range and set the ADC Channel Select register (ADxCHS) to an unused ADC channel whose corresponding pin is not connected to any circuit board trace. This minimizes added stray capacitance, keeping the total circuit capacitance close to that of the A/D Converter itself. If the measured delay is too long to measure with this method, an external capacitor may be connected to an ADC channel and this channel is selected during the measurement. If an external capacitor is used for the measurement, it should be connected during the capacitance calibration step.

Time measurement is accomplished with the following steps:

1. Configure the CTMU for Edge mode (TGEN = `0`).
2. Configure the I/O port pin as an input and enable Analog mode.
3. Configure the ADC for auto-sample and auto-convert.
4. Clear the edge status.
5. Enable the CTMU.
6. Manually discharge the capacitor (may be required for external capacitor).
7. Wait for an ADC interrupt (charging will be stopped by Edge 1 and Edge 2).
8. Clear the ADC interrupt.
9. Read the ADC result.
10. Manually discharge the capacitor (may be required for external capacitor).
11. Clear the edge status.

Figure 6-1: Time Measurement Block Diagram

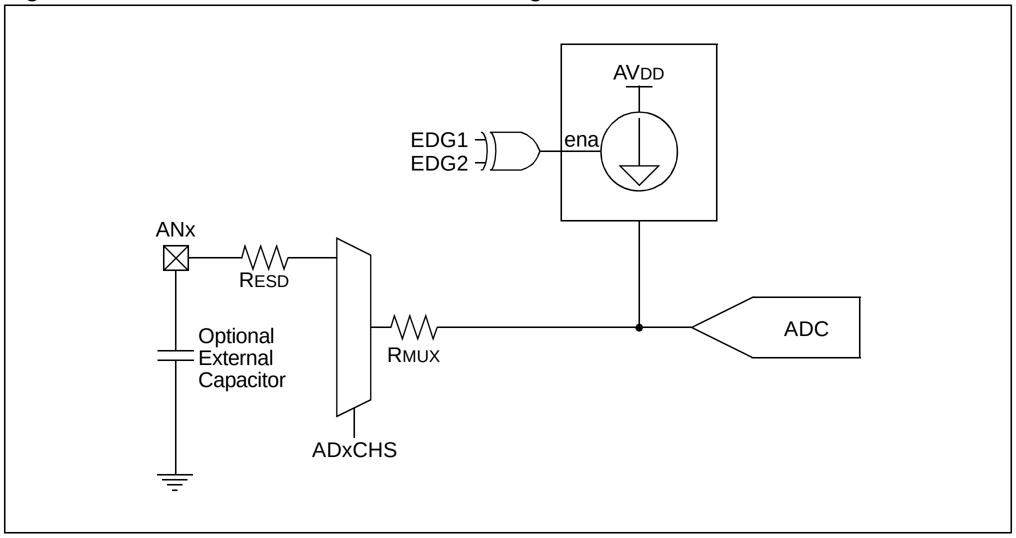
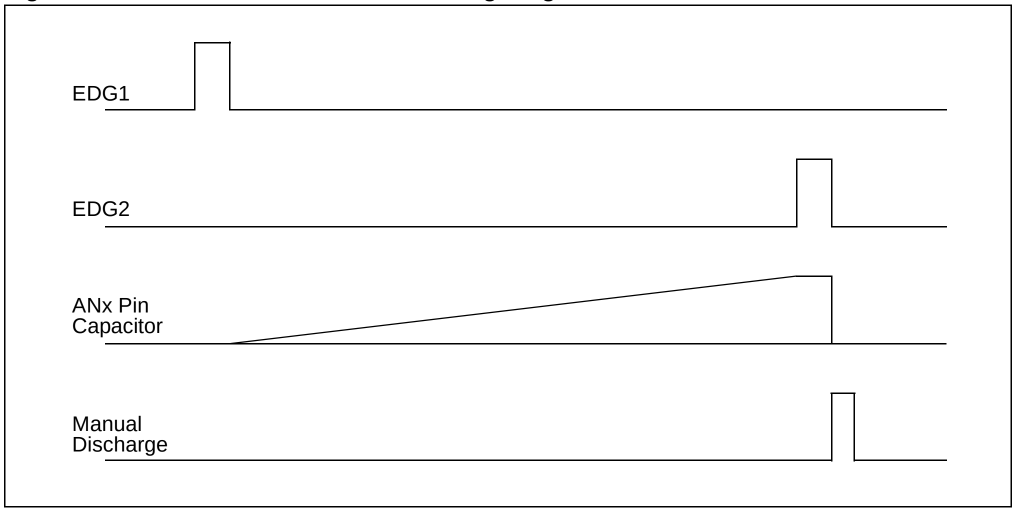


Figure 6-2: Time Measurement Timing Diagram



Example 6-1: Time Measurement

```
#define RANGE_0_550uA 1    // .550uA

void   CtmuTimeConfig(unsigned int range, signed int trim)
{
    // Step 1 Configure the CTMU
    CTMUCON1L = 0x0000;                  // Disable CTMU
    CTMUCON1Lbits.TGEN = 0;              // Disable Time Generation mode
    CTMUCON1Lbits.EDGEN = 1;             // Edges are enabled
    CTMUCON1Lbits.ITRIM = trim;          // Set trim
    CTMUCON1Lbits.CTTRIG = 1;            // Trigger output enabled
    CTMUCON1Lbits.IRNG = (range & 3);    // Set range
    // This line does not apply to all devices
    CTMUCON1Hbits.IRNGH = (range>>2);    // set high bit of range

    CTMUCON1Hbits.EDG1MOD = 1;           // Edge mode
    CTMUCON1Hbits.EDG1POL = 1;           // rising edge
    CTMUCON1Hbits.EDG1SEL = 4;           // CTED3 pin 8
    CTMUCON1Hbits.EDG2POL = 1;           // polarity
    CTMUCON1Hbits.EDG2MOD = 0;           // level sensitive
    CTMUCON1Hbits.EDG2SEL = 3;           // CTED1 pin 42

    CTMUCON2Lbits.IRSTEN = 1;            // enable reset by external trigger
    CTMUCON2Lbits.DSCHS = 4;             // ADC end of conversion


    // Step 2 Configure the port Ports
    TRISBbits.TRISB12 = 1;               // Configure RB12 as a input CTED2
    ANSBbits.ANSB12 = 0;                 // disable analog on RB12

    TRISBbits.TRISB13 = 1;               // Configure RB13 as a input CTED1
    ANSBbits.ANSB13 = 0;                 // disable analog on RB13

    TRISBbits.TRISB2 = 1;                // Configure RB2 as a input
    ANSBbits.ANSB2 = 1;                  // Configure AN2 as analog

    // Step 3 configure the ADC
    AD1CON1 = 0x0000;                    // Turn off ADC
    AD1CON1bits.SSRC = 4;                // CTMU is the conversion trigger source
    AD1CON2 = 0x0000;                    // VR+ = AVDD, V- = AVSS, Don't scan,
    AD1CON3 = 0x0000;                    // ADC uses system clock
    AD1CON3bits.ADCS = 8;                // conversion clock = 1xTcy
    AD1CON5 = 0x0000;                    // Auto-Scan disabled
    AD1CON1bits.ADON = 1;
    AD1CON1bits.ASAM = 1;                // Auto-sample
    AD1CHSbits.CH0SA = 2;                // Select AN2

    //  Step 4 - 6 Enable the current source and stop manual discharge
    CTMUCON1H &= ~0x0300;                // clear the edge status bits
    CTMUCON1Lbits.CTMUEN = 1;            // Enable the CTMU
    CTMUCON1Lbits.IDISSEN = 1;           // Enable Discharge
    asm("NOP");                          // may be required for external caps
    asm("NOP");
    asm("NOP");
    asm("NOP");
    CTMUCON1Lbits.IDISSEN = 0;           // stop discharge
}

int    main(void)
{
```

Example 6-1: Time Measurement (Continued)

```
    unsigned   int result;

    CtmuTimeConfig(RANGE_0_550uA, 5);     // .550uA

    while(1)
    {
        // Step 7: Wait for ADC interrupt
        while(IFS0bits.AD1IF == 0);
        {
            // Steps 8-11
            IFS0bits.AD1IF = 0;           // clear the interrupt
            result = ADC1BUF0;            // read ADC result

            CTMUCON1Lbits.IDISSEN = 1;    // begin manual discharge of cap
            asm("NOP");                   // may be required for external caps
            asm("NOP");
            asm("NOP");
            asm("NOP");
            CTMUCON1Lbits.IDISSEN = 0;    // stop discharge of cap

            CTMUCON1H &= ~0x0300;         // clear the edge status bits
        }

        // user code
    }
}
```

7.0 GENERATING DELAYS WITH THE CTMU MODULE

The CTMU module can be used to create delays independent of the system clock. This is accomplished by using the CTMU in Time Generation mode to charge a capacitance, internal or external, and using Comparator 2 to generate an EDG2 event when the capacitor voltage reaches the desired value, as set by the CVREF module. The pulse is started by an EDG1, internal or external, event. In this mode, the CTPLS GPIO pin is controlled by the CTMU module and is used to generate a pulse whose width is the delay time. If desired, the Comparator 2 interrupt can be enabled to inform software that a pulse was generated. For proper operation, the EDG1 event should be inactive before the EDG2 event occurs. Refer to Figure 7-1 and Figure 7-2.

The combination of the comparator reference selection, the current range and the capacitance value are used to create the desired delay period. The delay pulse width is calculated by, $T = (C_{SYSTEM}/I) * V$, where I is known from the current source calibration (see **Section 4.1 "Current Source Calibration"**) and V is the comparator reference input voltage.

Time measurement is accomplished with the following steps:

1. Configure the CTMU for Time Generation mode (TGEN = 1).
2. Configure the comparator reference.
3. Configure the comparator.
4. Configure the I/O port pin as an input and enable Analog mode.
5. Enable the CTMU.

Figure 7-1: Delay Generation Block Diagram

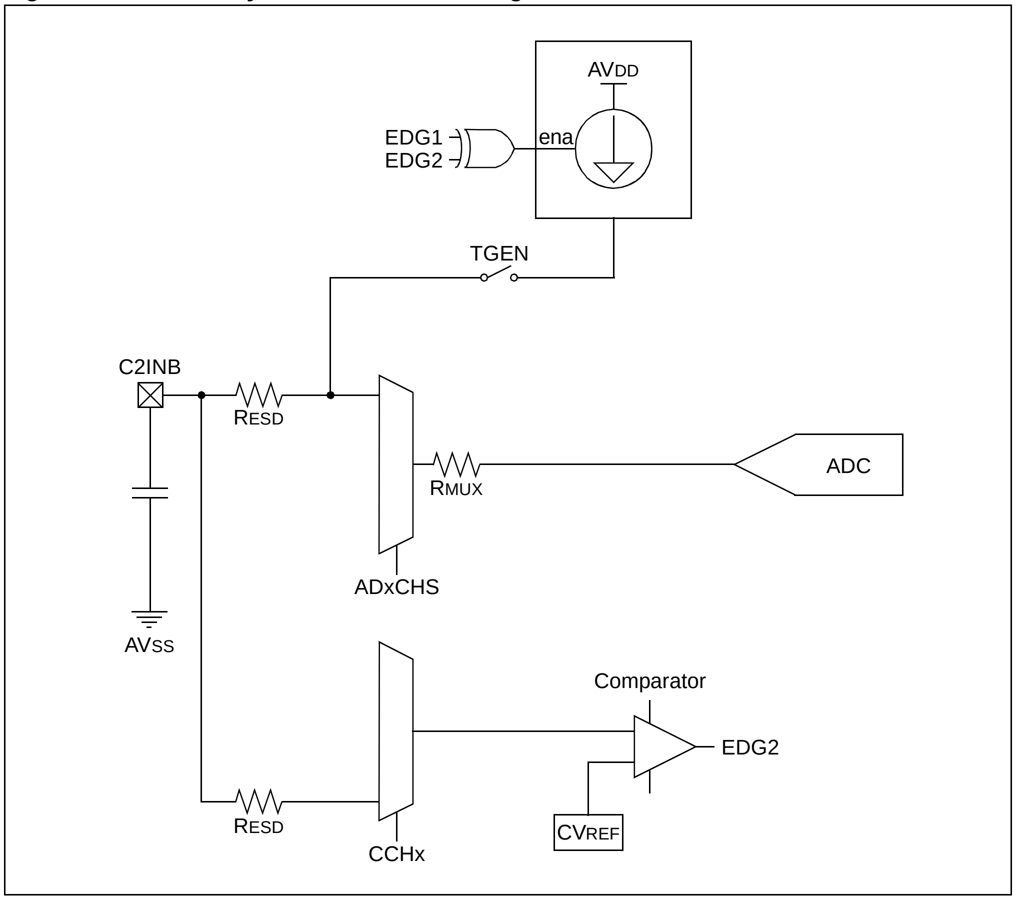
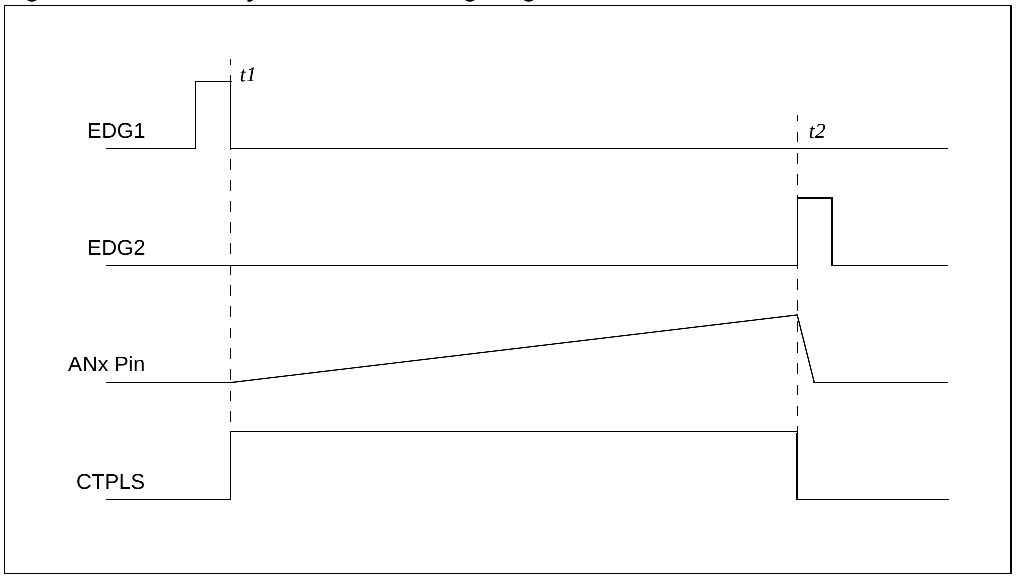


Figure 7-2: Delay Generation Timing Diagram



Example 7-1: Delay Generation

```
#include "p24FJ1024GB610.h"
#define RANGE_55uA 3                      // 55uA

void   CtmuDelayConfig(unsigned int range, signed int trim)
{
    // Step 1 Configure the CTMU
    CTMUCON1L = 0x0000;                   // Disable CTMU
    CTMUCON1Lbits.TGEN = 1;               // Enable/Disable Time Generation mode
    CTMUCON1Lbits.IDISSEN = 1;            // Current source is  grounded
    CTMUCON1Lbits.ITRIM = trim;           // Set trim
    CTMUCON1Lbits.CTTRIG = 0;             // Trigger output disabled
    CTMUCON1Lbits.IRNG = (range & 3);     // Set range
    CTMUCON1Lbits.EDGEN = 1;              // Edges are enabled
    CTMUCON2Lbits.IRSTEN = 0;             // Current source reset disabled
    CTMUCON2Lbits.DSCHS = 0;              // Discharge source disabled
    // This line does not apply to all devices
    CTMUCON1Hbits.IRNGH = (range>>2);     // set high bit of range
    CTMUCON1Hbits.EDG1POL = 1;            // negative polarity
    CTMUCON1Hbits.EDG1MOD = 1;            // edge sensitve
    CTMUCON1Hbits.EDG1SEL = 4;            // CTED3 pin 8
    CTMUCON1Hbits.EDG2POL = 1;            // negative polarity
    CTMUCON1Hbits.EDG2MOD = 0;            // level sensitive
    CTMUCON1Hbits.EDG2SEL = 14;           // CMP2 out

    // Step 2 Configure Comparator Voltage Reference
    CVRCONbits.CVREFP = 0;                // use DAC as module output
    CVRCONbits.CVROE = 1;                 // DAC voltage is output on a pin
    CVRCONbits.CVRSS = 0;                 // DAC references are AVDD/AVSS
    CVRCONbits.CVR = 14;                  // midscale output
    CVRCONbits.CVREN = 1;                 // enable module

    // Step 3 Configure Comparator
    CM2CONbits.COE = 1;                   // comparator output is present on C2OUT pin
    CM2CONbits.CPOL = 1;                  // output is inverted
    CM2CONbits.CREF = 1;                  // non-inverting input is CVREF output
    CM2CONbits.CCH = 0;                   // inverting input is C2INB pin
    CM2CONbits.CEN = 1;                   // enable the comparator

    // Step 2 Configure the port Ports
    TRISBbits.TRISB2 = 1;                 // Configure RB2 as a input
    ANSBbits.ANSB2 = 1;                   // Configure AN2 as analog
    // (Capacitor to AVSS is connected to this pin)

    TRISGbits.TRISG15 = 1;                // Configure RG15 (CTED3 as an input)

    CTMUCON1Lbits.CTMUEN = 1;             // Enable the CTMU
}

int    main(void)
{
       CtmuDelayConfig(RANGE_55uA, 6);

       // external event triggers EDG1
       // user code

    while(1);
}
```

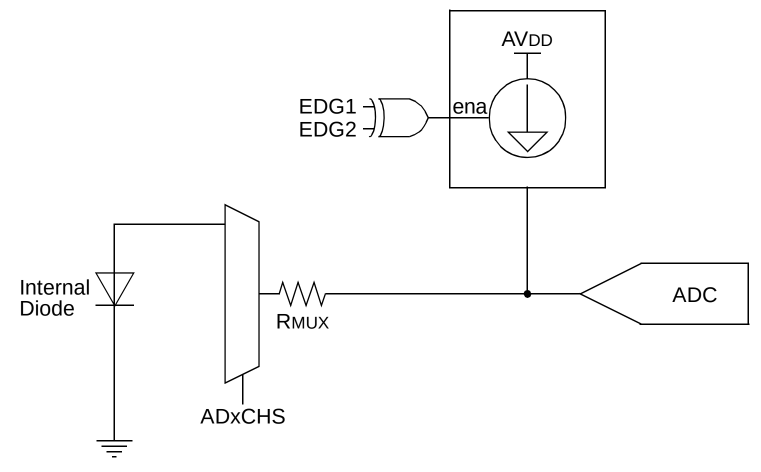
8.0 MEASURING TEMPERATURE WITH THE CTMU

The CTMU module can be used to measure the internal temperature of the device through a dedicated internal temperature diode. When configured for temperature measurement, the CTMU current flows through the diode. The resulting voltage drop across the diode can then be measured with the ADC and the temperature calculated.

The Forward Voltage (Vf) of a P-N diode changes with temperature. The Forward Voltage (Vf) of a diode is negatively proportional to the temperature. In other words, Vf increases with a decrease in temperature. Temperature measurements should only be performed with the CTMU at the 5.5 µA or 55 µA current ranges.

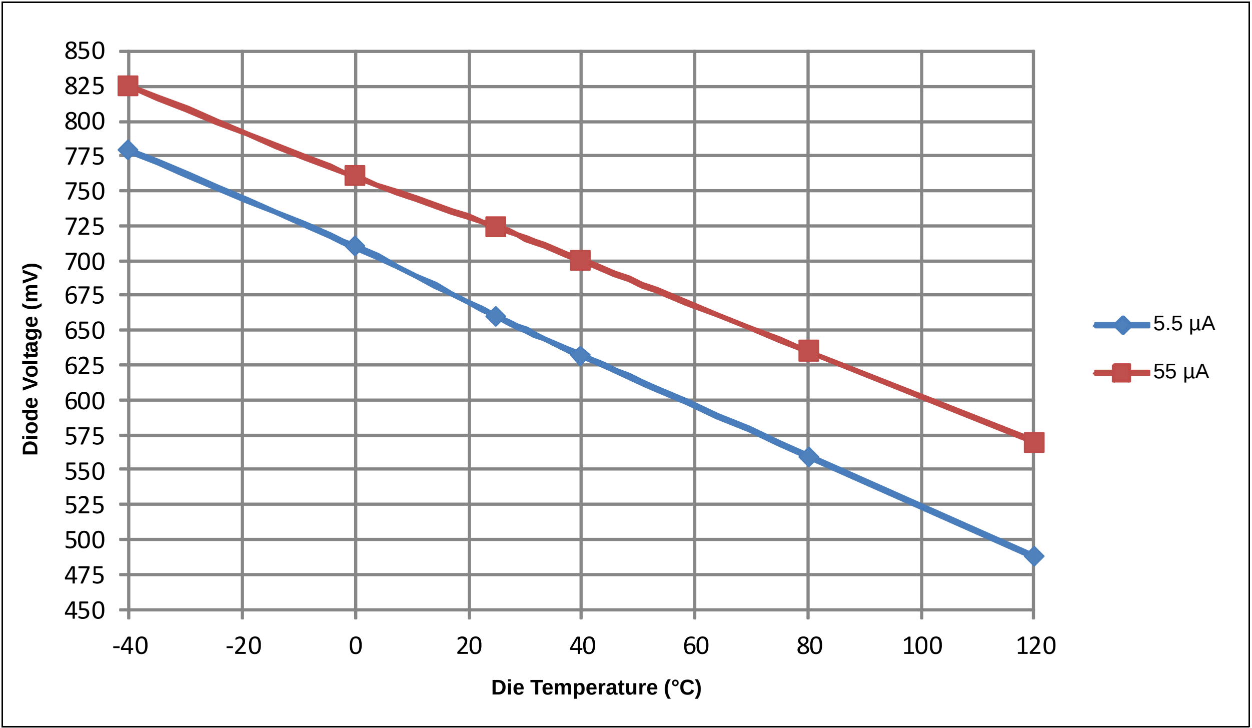
Figure 8-1 shows how this module can be used for temperature measurement. As the temperature rises, the voltage across the diode will drop approximately 1.8 mV/°C (for the 5.5 µA range) over the device operating range (refer to Figure 8-2, Equation 8-1 and Example 8-2). Using the 55 µA current range increases the voltage offset, providing an improvement in signal to noise ratio. The voltage output of the diode is nearly linear across the device specified operating range for both ranges.

Figure 8-1: Temperature Measurement Block Diagram



Note: Refer to the specific device data sheet for the actual channel number associated with the temperature sensing diode.

Figure 8-2: Temperature Curves



8.1 Operation

When the current source is enabled, TGEN = `0`, and the temperature diode is selected as the ADC analog input, the CTMU current source output is connected to the temperature sensing diode. The voltage developed across the diode is available as an input to the ADC module through a dedicated analog input channel, which is selected using ADxCHS register (refer to Example 8-1).

The following outlines the steps required to perform a temperature measurement:

1. Configure and enable the CTMU.
2. Configure the ADC.
3. Start manual sampling.
4. Wait for sample capacitor to charge.
5. Convert sample.
6. Convert counts to temperature (refer to Equation 8-1).

Example 8-1: Temperature Measurement

```
#define    RANGE_5_5uA   2       // 5.5uA

unsigned   int CtmuReadTemperatureCounts(unsigned int range, signed int trim)
{
    unsigned   int x, result;

    // Step 1 Configure the CTMU
    CTMUCON1L = 0x0000;                   // Disable CTMU
    CTMUCON1Lbits.ITRIM = trim;           // Set trim
    CTMUCON1Lbits.IRNG = (range & 3);     // Set range

    CTMUCON1Lbits.CTMUEN = 1;             // Enable the CTMU
    CTMUCON1Hbits.EDG1STAT = 1;           // Enable current source
    CTMUCON1Hbits.EDG2STAT = 1;

    // Step 2 Configure the ADC
    AD1CHSbits.CH0SA = 0x18;              // Select temp sensor
    AD1CON1 = 0x8000;                     // Turn On A/D Converter,
    // Unsigned fractional format, Clear SAMP bit to
    // start conversion, Sample when SAMP bit is set
    AD1CON2 = 0x0000;                     // VR+ = AVDD, V- = AVSS, Don't scan,
    AD1CON3 = 0x0000;                     // ADC uses system clock
    AD1CON3bits.ADCS = 0;                 // conversion clock = 1xTcy
    AD1CON5 = 0x0000;                     // Auto-Scan disabled
    AD1CTMENHbits.CTMEN24 = 1;            // CTMU connected to channel during conversion

    // Step 3 start sampling
    AD1CON1bits.SAMP = 1;                 // Manual sampling start

    // Step 4 delay
    for (x = 0; x < 2000; x++);           // Delay ~xxx us to charge sample cap

    // step 5 convert sample
    AD1CON1bits.SAMP = 0;                 // Begin A/D conversion
    while(AD1CON1bits.DONE == 0);         // Wait for A/D convert complete

    result = ADC1BUF0;

    return (result);
}


int     main(void)
{
    unsigned   int result;

    result = CtmuReadTemperatureCounts( RANGE_5_5uA, 6);

    while(1);
}
```

Equation 8-1: CTMU Temperature Measurement Calculations

For 5.5 µA Current Source Range:

$$T_{INTERNAL} = \frac{710\ mV - V_{MEASURED}}{1.8}$$

For 55 µA Current Source Range:

$$T_{INTERNAL} = \frac{760\ mV - V_{MEASURED}}{1.55}$$

$$V_{MEASURED} = ADC\ Counts * ADC\ Reference/ADC\ Max\ Counts$$

Example 8-2: CTMU Temperature Measurement Calculations

ADC in 10-Bit Mode, ADC Reference 3.3V
ADC Measured Counts = 0xCC = 204 Decimal
CTMU Configured for 5.5 µA Range

$$V_{MEASURED} = ADC\ Counts * ADC\ Reference/ADC\ Max\ Counts$$

$$V_{MEASURED} = 204c * 3.3V/1023 = 657\ mV$$

$$T_{INTERNAL} = \frac{710\ mV - 658\ mV}{1.8} = 28.9C$$

8.2 Calibration

The accuracy of the temperature measurement can be improved with a calibration process.

For narrow temperature measurement ranges, calculating the theoretical voltage, and comparing it to the measured voltage at a known temperature, will give a calibration offset that can be used for future measurements.

For wide temperature measurement ranges, a measurement is required at two known temperatures to more accurately calculate an offset. Ideally, these measurements are near the extremes of the desired measurement range. The results of these two measurements are first used to calculate the temperature curve slope. The resulting slope is then used in the temperature equation to calculate an offset (refer to Equation 8-2). This offset then can be added or subtracted from future measurements.

The temperature calibration compensates for the CTMU current, therefore, current source calibration is not required for temperature calibration. If the current source calibration is required for other applications, the temperature calibration should be performed after the current source is calibrated.

Equation 8-2: Temperature Slope Calculations

$$Slope = \frac{(Temperature1 - Temperature2)}{Voltage\ at\ t1 - Voltage\ at\ t2}$$

$$Offset\ in\ Volts = Temperature1 - (Slope * Voltage\ at\ t1)$$

$$Volts/ADC\ Count = \frac{ADC\ Reference\ Voltage}{(2\text{^}ADC\ Bits)}$$

$$Temperature = (Voltage/Slope) + Offset$$

Example 8-3: Temperature Calibration

t1 = 40°C, 242 ADC Counts
t2 = 120°C, 150 ADC Counts
CTMU Configured for 5.5 µA Range

$$ADCres = \frac{3.3V}{1023\ Counts} = 3.22\ mV/Count = 0.00322$$

$$Slope = \frac{(242c - 150c) * 0.00322}{-40 - 120} = \frac{0.296}{-160} = 0.00186 = 1.86\ mV/°C$$

$$Offset = -40°C - \frac{(0.00322 * 242)}{-0.00186} = -378.8$$

Therefore, for a Measurement of 195 Counts:

$$T = \frac{(195 * 0.00322)}{0.00186} - 378.8 = 40.5°C$$

9.0 OPERATION DURING SLEEP/IDLE MODES

9.1 Sleep Mode and Deep Sleep Modes with the CTMREQ Bit Disabled and Devices without a CTMREQ Bit

When the device enters any Sleep mode, the CTMU module current source is always disabled. If the CTMU is performing an operation that depends on the current source when Sleep mode is invoked, the operation may not terminate correctly. Capacitance and time measurements may return erroneous values.

9.2 Sleep Mode and Deep Sleep Modes with the CTMREQ Bit Enabled

When the device enters Sleep mode with the CTMREQ bit (ADxCON5<13>) set, the CTMU will continue operation in Sleep. The CTMU can be triggered from the ADC, which can perform the conversion in Sleep mode when the A/D clock source is set to the internal A/D RC oscillator (ADRC = 1). This will allow the CPU to remain in the inactive state for a longer period of time, and at the same time, it can perform the conversion of the selected CTMU channel.

When the A/D interrupt (AD1IE) is enabled, the device will wake-up from Sleep as soon as the A/D interrupt occurs. This will help to reduce the power consumed by the CPU, which is the prerequisite for many low-power applications.

9.3 Idle Mode

The behavior of the CTMU in Idle mode is determined by the CTMUSIDL bit (CTMUCON1L<13>). If CTMUSIDL is cleared, the module will continue to operate in Idle mode. If CTMUSIDL is set, the module’s current source is disabled when the device enters Idle mode. If the module is performing an operation when Idle mode is invoked, in this case, the results will be similar to those with Sleep mode.

10.0 EFFECTS OF A RESET ON CTMU

Upon Reset, all registers of the CTMU are cleared. This leaves the CTMU module disabled, its current source is turned off and all configuration options return to their default settings. The module needs to be re-initialized following any Reset.

If the CTMU is in the process of taking a measurement at the time of Reset, the measurement will be lost.

11.0 LOW-POWER APPLICATIONS FOR DEVICES WITH A CTMREQ BIT

The CTMU module, along with the ADC with Auto-Threshold Detection technique, can be used for implementing low-power based applications. The low-power functionality can be achieved by alternatively shifting between the Normal mode and the Sleep mode, which is based on the need of the application. The CTMU module will be active in Sleep mode by enabling the CTMREQ bit (ADxCON5<13>). This logic will significantly reduce the power consumed by the system, which is the common requirement in many of the applications.

12.0 ELECTRICAL SPECIFICATIONS

Refer to the device data sheet **"Electrical Characteristics"** section for the current ranges for your device.

13.0 RELATED APPLICATION NOTES

This section lists application notes and technical briefs that are related to this section of the manual. These application notes and technical briefs may not be written specifically for the dsPIC33/PIC24 device families, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes and technical briefs related to the Charge Time Measurement Unit (CTMU) and CTMU Operation with Threshold Detect module are:

Title	**Application Note #**
See What You Can do with the CTMU	AN1375
Using the PIC® MCU CTMU for Temperature Measurement	TB3016
Microchip CTMU for Capacitive Touch Applications	AN1250
ESD and EOS Causes, Differences and Prevention	AN1785
Techniques for Robust Touch Sensing Design	AN1334
mTouch™ Conducted Noise Immunity Techniques for the CTMU	AN1317

Source Code for PIC24F MCU using CTMU Sensing Method – *PIC24F source code and projects for touch sensing, in conducted noise environment, using the CTMU and ADC for sensing.*

> **Note:** Please visit the Microchip web site (www.microchip.com) for additional application notes and code examples for the dsPIC33/PIC24 families of devices.

14.0 REVISION HISTORY

Revision A (March 2011)

This is the initial released revision of this document.

Revision B (April 2016)

The document was updated to include the dsPIC33 families of devices and the title was changed to CTMU and CTMU Operation with Threshold Detect. Updated the Family Reference Manual name to dsPIC33/PIC24 Family Reference Manual. Removed Section 53 and tabs as this family reference manual does not use overall section numbers. Headings were renumbered appropriately.

- Equations:
  - Added Equation 4-1, Equation 4-2, Equation 4-3, Equation 4-4, Equation 4-5, Equation 8-1 and Equation 8-2
- Examples:
  - Added Example 4-1, Example 4-2, Example 4-3, Example 4-4, Example 4-5, Example 5-1, Example 5-2, Example 6-1, Example 7-1, Example 8-1, Example 8-2 and Example 8-3
- Figures:
  - Updated Figure 1-1
  - Added Figure 1-2, Figure 4-1, Figure 4-2, Figure 4-3, Figure 5-1, Figure 5-2, Figure 5-3, Figure 5-4, Figure 6-1, Figure 6-2, Figure 7-1, Figure 7-2, Figure 8-1 and Figure 8-2
- Tables:
  - Removed previous Table 53-2
  - Updated Table 2-1
  - Added Table 4-1
- Registers:
  - Updated Register 2-1, Register 2-2 and Register 2-3
- Sections:
  - Removed previous Sections 53.4 through 53-9
  - Updated **Section 1.0 "Introduction"**, **Section 2.0 "Register Maps"**, **Section 9.0 "Operation During Sleep/Idle Modes"**, **Section 10.0 "Effects of a Reset on CTMU"**, **Section 11.0 "Low-Power Applications for Devices with a CTMREQ Bit"**, **Section 12.0 "Electrical Specifications"** and **Section 13.0 "Related Application Notes"**
  - Added completely new **Section 3.0 "CTMU Operation"** through **Section 8.0 "Measuring Temperature with the CTMU"**
- Additional minor corrections such as language and formatting updates were incorporated throughout the document

NOTES:

ISBN: 978-1-5224-0493-4

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110

Canada - Toronto
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon

Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Dongguan
Tel: 86-769-8702-9880

China - Hangzhou
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

ASIA/PACIFIC

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7828

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Dusseldorf
Tel: 49-2129-3766400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Venice
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Poland - Warsaw
Tel: 48-22-3325737

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

07/14/15