

M031/M032類比數位轉換器基礎介紹

Application Note for 32-bit NuMicro® Family

Document Information

Abstract	本文介紹了ADC對連續外部信號進行採樣的概念和用法
Apply to	NuMicro® M031/M032系列

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

*Nuvoton is providing this document only for reference purposes of NuMicro microcontroller based system design.
Nuvoton assumes no responsibility for errors or omissions.*

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

目錄

1	簡介.....	3
1.1	概述	3
2	一般操作	5
2.1	時鐘源選擇	5
2.2	輸入模式選擇	5
2.2.1	單端輸入	5
2.2.2	全差分輸入	6
2.3	外部參考電壓管腳 V_{REF}	7
2.4	轉換	8
2.5	外部等效阻抗	9
2.6	ADC操作模式.....	11
2.6.1	單次觸發模式	11
2.6.2	Burst模式	11
2.6.3	單週期掃描模式	11
2.6.4	連續掃描模式	12
2.7	校準	12
2.8	直接記憶體存取控制器(PDMA)	13
2.9	數位比較器	14
2.9.1	閾值模式	14
2.9.2	區間模式	15
3	範例程式碼.....	16
4	結論.....	22
5	版本歷史	23

1 簡介

M031/M032內建逐次逼近式類比數位控制器(SAR ADC)，支援12位解析度，每秒可實現2百萬次採樣（Msps）。內建類比多工器(Analog Multiplexer)，ADC輸入信號可選擇從外部管腳或者內部電壓。利用PDMA功能，ADC可以在沒有CPU參與的情況下運行，執行最大採樣率，最小化系統電流消耗，或允許CPU執行其它工作項目。

本應用手冊討論了ADC一般操作流程、軟硬體注意事項及PDMA的使用方法。最後範例程序以PDMA和ADC來展示連續採樣外部信號的流程。

1.1 概述

圖 1-1說明輸入信號，參考電壓選擇和寄存器的內部連接方式，並包含以下功能：

- 一組 12 位解析度 SAR ADC 並帶獨立 V_{REF} 管腳
- 多達 16 路的單端輸入通道或 8 組全差分模擬輸入通道
- 1 個內部輸入通道 band-gap voltage(V_{BG})
- A/D 轉換開始條件:
 - 軟體向 ADST 位元寫 1
 - 外部 (STADC)管腳觸發
 - Timer 0~3 溢出脈衝觸發
 - PWM 事件觸發
- 每個通道的轉換結果儲存在相應資料寄存器內，並帶有有效和覆蓋標誌
- 支援 PDMA 傳輸模式，將轉換結果儲存在用戶指定的目標位址

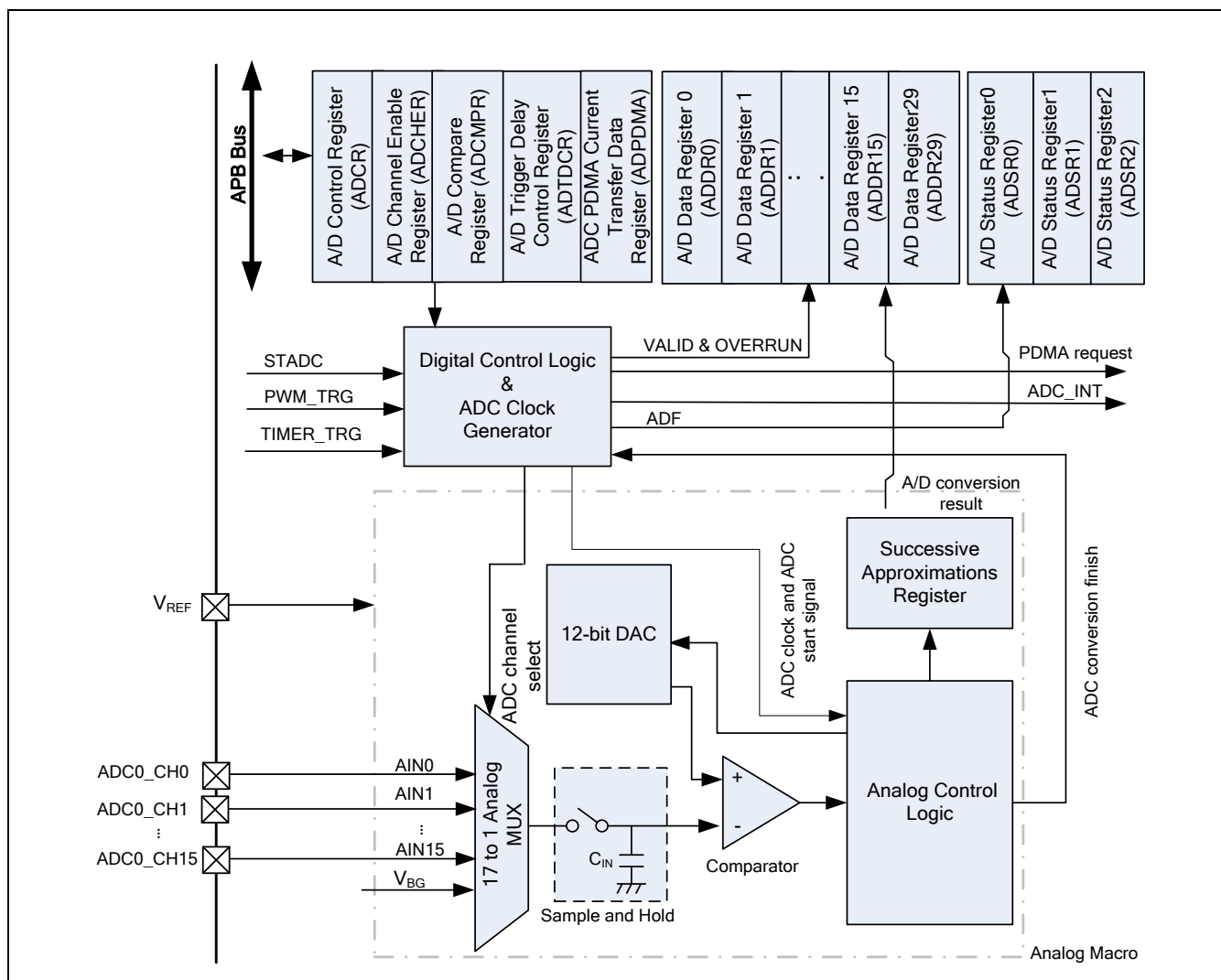


圖 1-1 ADC 控制器框圖

2 一般操作

2.1 時鐘源選擇

ADC具有一組8-bit預分頻器，按照以下列公式得出ADC時鐘頻率(ADCCLK)為：

$$\text{ADC時鐘頻率(ADCCLK)} = (\text{ADC時鐘源頻率}) / (\text{ADCDIV} + 1);$$

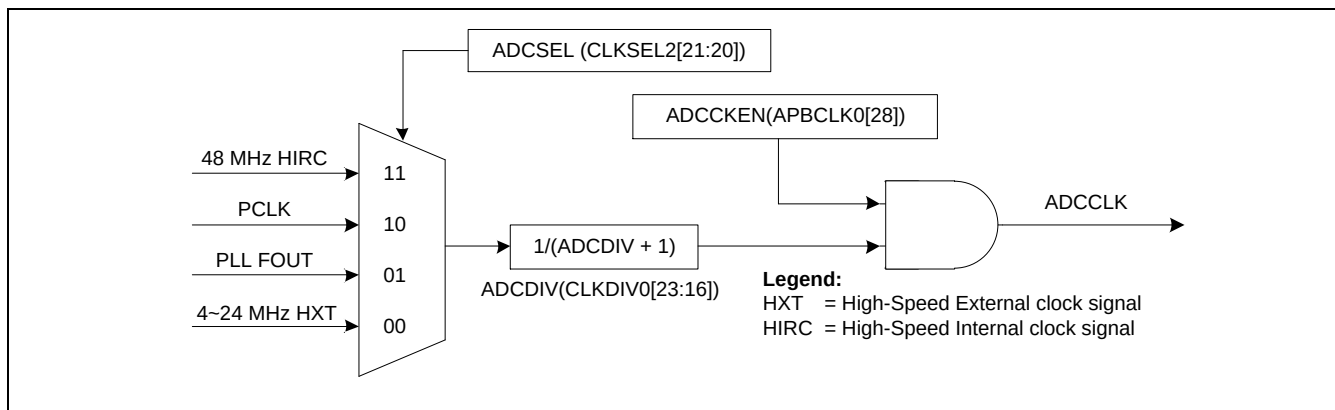


圖 2-1 ADC 時鐘源控制

其中M031/M032 ADC時鐘頻率(ADCCLK)最大值為34 MHz。使用者可以透過PLL倍頻，產生34 MHz或者68 MHz頻率。將PLL設定為ADC時鐘源，並透過ADCDIV除頻產生34 MHz，即可讓ADC運行在最高速度。

2.2 輸入模式選擇

ADC輸入模式可以選擇為單端輸入或全差分輸入。當DIFFEN(ADC_ADCCR [10])為1，ADC為差分輸入。

2.2.1 單端輸入

如圖 2-2所示，在單端輸入模式下， AV_{SS} 或者 V_{SS} 為ADC負端輸入的信號(AIN-)，外部輸入電壓為ADC正端輸入的信號(AIN+)。如圖 2-3所示，在0V和參考電壓(V_{REF})之間，切分出微小電位階。其數位結果以0到4095的無符號數，表示輸入電壓位於0V和參考電壓(V_{REF})之間的位置。

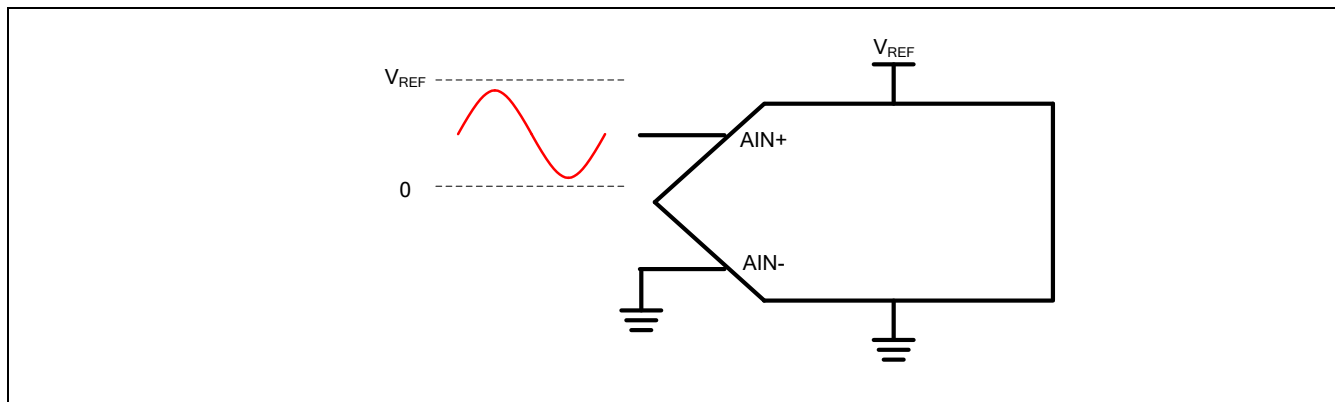


圖 2-2 ADC 單端輸入信號

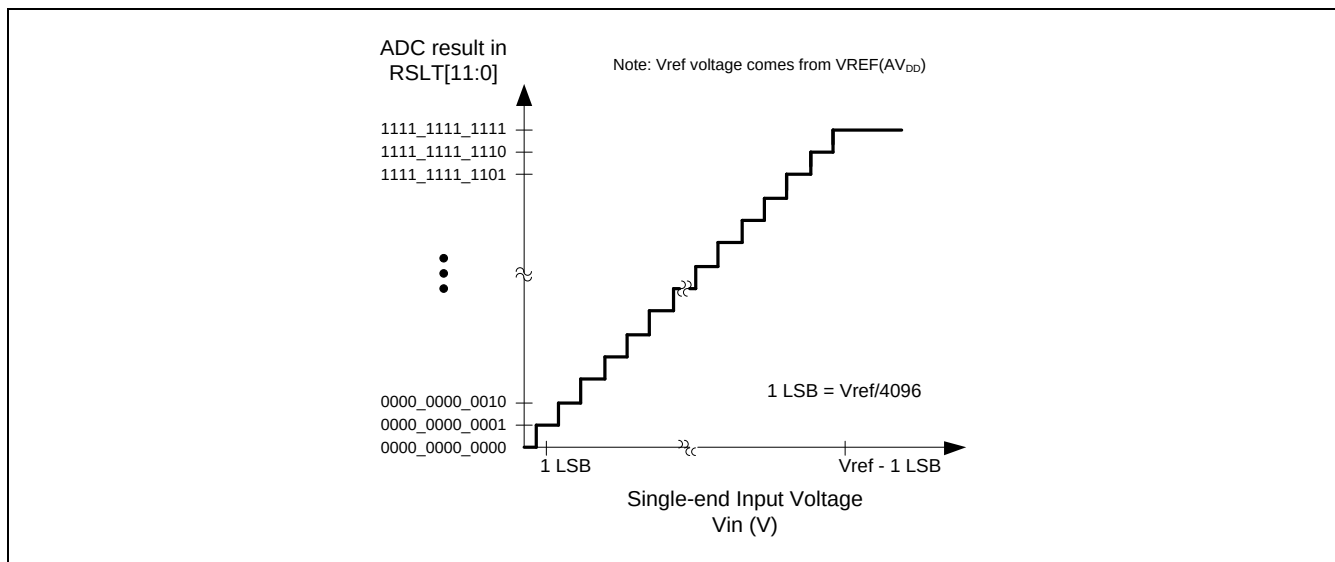


圖 2-3 ADC 單端輸入模式轉換結果映射圖

2.2.2 全差分輸入

如圖 2-4所示，在全差分輸入模式下，測量值是兩個輸入信號之間的差值，兩個輸入信號必須為 180° 反相信號，並固定共模電壓為 $V_{REF}/2$ 。由於一個輸入管腳被定義為正輸入管腳(AIN+)而另一個被定義為負輸入管腳(AIN-)，因此根據哪個輸入信號更高，差值可以是正的或負的。如圖 2-5所示，當DMOF(ADC_ADCR [31])設置為1時，轉換結果是以二的補數碼形式表示的有符號數，或者當DMOF(ADC_ADCR [31])設置為0時，轉換結果是以二進制格式表示的無符號數。

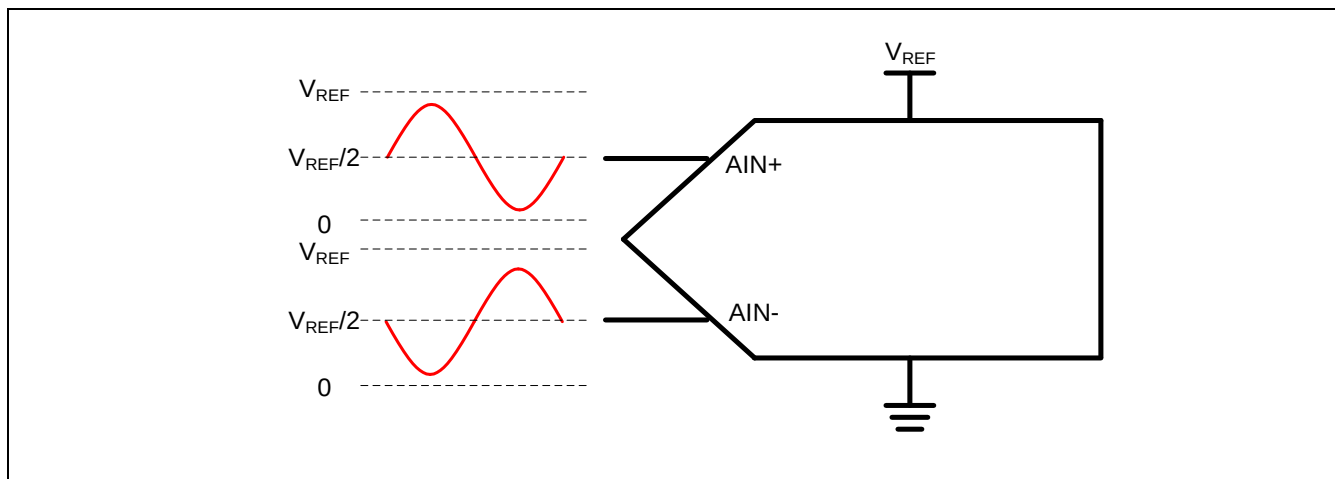


圖 2-4 ADC 全差分輸入信號

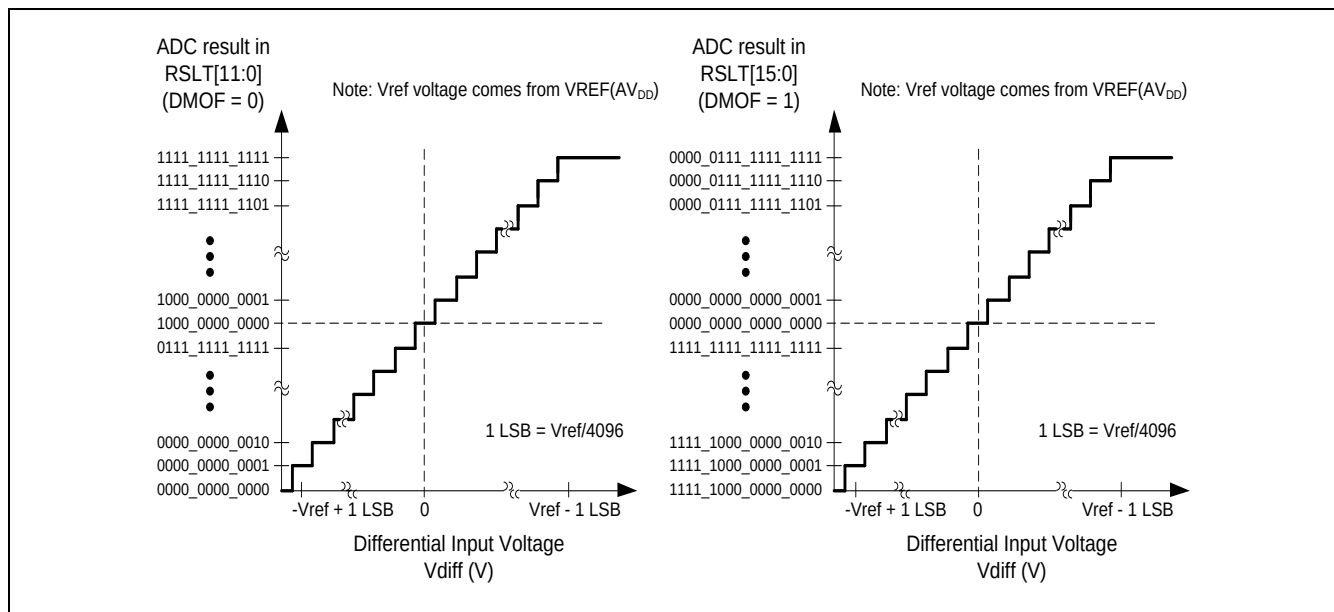


圖 2-5 ADC 全差分輸入模式轉換結果映射圖

2.3 外部參考電壓管腳 V_{REF}

為了將類比電壓轉換為數位元結果，ADC需要一個參考電壓來比較輸入的類比電壓。由於ADC測量的電壓(V_{IN})無法大於參考電壓(V_{REF})，因此參考電壓(V_{REF})應高於最大預期測量電壓且低於 AV_{DD} 電壓。詳細電壓規格列於表 2-1中。

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
AV_{DD}	Analog operating voltage	1.8	-	3.6	V	$V_{DD} = AV_{DD}$
V_{REF}	Reference voltage	1.8	-	AV_{DD}	V	
V_{IN}	ADC channel input voltage	0	-	V_{REF}	V	

表 2-1 ADC 電壓範圍

在硬體規畫中，列出以下三個注意事項：

1. ADC 無法測量負電壓或大於參考電壓的電壓，因此 V_{IN} 信號必須保持在電氣特性範圍內。
2. ADC 沒有使用的通道管腳，也可當作一般的 GPIO，其注入電流也是 ADC 精準度的重要課題。如圖 2-6 所示，假設 ADC0_CH15 被設定為一般 GPIO 管腳，類比多工器的類比開關會處於斷開狀態下。當外部有負電壓或者大於 AV_{DD} 電壓產生，會強制將類比多工器的類比開關變為連接狀態，進而產生注入電流。此時 GPIO 電壓會直接影響 ADC 電容(C_{IN})充放電。設計過程中，應避免在任何 ADC 管腳上有負電壓或者大於 AV_{DD} 電壓產生的注入電流，以保護類比輸入信號轉換的結果。如果有注入電流問題，建議在有注入電流的 ADC 管腳上添加肖特基二極體(管腳對地和管腳對電源)，將注入電流引導到電源端。

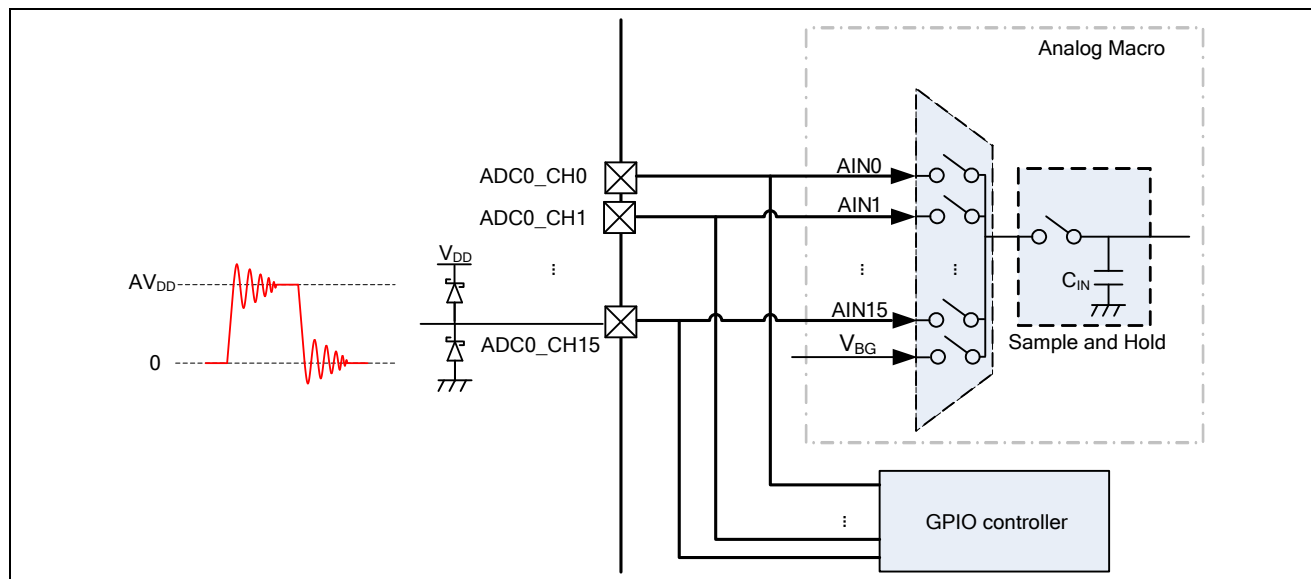


圖 2-6 ADC 輸入保護電路

3. ADC 測量值是類比電壓和 V_{REF} 之間的比率， V_{REF} 上的任何雜訊都會導致轉換結果發生變化。在某些封裝上拿 AV_{DD} 當作 V_{REF} 使用，在此限制下 AV_{DD} 的品質會影響 ADC 精度。

例如，當 V_{REF} 連接到 3.3V，類比輸入為 2 V 時，轉換結果為：

$$\frac{2}{3.3} \times 4095 = 0x9B1$$

如果 V_{REF} 的電壓包含紋波，其值為 100 mV 的峰對峰值，則轉換結果可能會為：

$$\text{當紋波為正 50mV 時，} \frac{2}{3.3+0.1/2} \times 4095 = 0x98C$$

$$\text{兩者的誤差} = 0x9B1 - 0x98C = 37 \text{ LSB}$$

$$\text{當紋波為負 50mV 時，} \frac{2}{3.3-0.1/2} \times 4095 = 0x9D8$$

$$\text{兩者的誤差} = 0x9B1 - 0x9D8 = -39 \text{ LSB}$$

因此，ADC 結果是類比電壓與 V_{REF} 的比值， V_{REF} 電壓對 ADC 精度非常重要。必須依照整體系統的特性，加入適當的電容來減少紋波的影響。

2.4 轉換

轉換過程包括著信號的採樣保持階段(Sample & Hold)。ADC 內部有一個等效電容(C_{IN})，在採樣階段時輸入信號(V_{IN})對著內部等效電容(C_{IN})充放電，使其同電位。接著在保持階段，將內部等效電容(C_{IN})的電壓，利用逐次逼近法，將保持的電壓轉換為數位結果來表示。其中、如表 2-2 所示，採樣時間可以通過設置 ADC 時序的 EXTSMPT(ADC_ESMPCTL[7:0])來配置採樣時間。

當 ADC 以高速運行時，如果外部等效負載太重而無法產生足夠的充放電電流給內部等效電容(C_{IN})，在保持階段將會以錯誤的電位轉換為數位結果。此時應用上就會產生問題。為了避免充放電時間過短。可以利用 EXTSMPT(ADC_ESMPCTL[7:0])來增加充放電時間。其中延長採樣時間範圍為 0~255 ADC 時鐘。

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
$F_{ADC}^{[*1]}$ $1/T_{ADC}$	ADC Clock frequency	4	-	34	MHz	
T_{SMP}	Sampling Time	1	-	256	$1/F_{ADC}$	$T_{SMP} = (EXTSMPT(ADC_ESMPCTL[7:0]) + 1) * T_{ADC}$
T_{CONV}	Conversion time	17	-	272	$1/F_{ADC}$	$T_{CONV} = T_{SMP} + 16 * T_{ADC}$
$F_{SPS}^{[*1]}$	Sampling Rate	0.222	-	2	MSPS	$F_{SPS} = F_{ADC} / T_{CONV}$ $EXTSMPT(ADC_ESMPCTL[7:0]) = 0$

表 2-2 ADC 轉換時序

2.5 外部等效阻抗

在圖 2-7 中，信號源的等效阻抗(R_{EX})會產生電壓差，並在採樣階段中，限制對內部採樣電容(C_{IN})的充放電電流。如果要對內部採樣電容(C_{IN})完全充電，其充電時間要考量到外部等效阻抗(R_{EX})和開關電阻(R_{IN})效應。

C_{IN} 有效充電由($R_{IN} + R_{EX}$)來控制，因此充電時間常數變為 $T_C = (R_{IN} + R_{EX}) \times C_{IN}$ 。如果採樣時間小於完全充電 C_{IN} 所需的時間，則ADC數位結果將不等於實際值。在計算 C_{IN} 完全充電時間，會以誤差為1/4 LSB為條件，求出最大允許的等效阻抗($R_{IN} + R_{EX}$)。

如公式 1所示，當 R_{EX} 小於10分之一的 R_{IN} ， C_{EX} 影響變小，可以假設 C_{EX} 為0。其中 $N = 12$ （基於12位解析度）， k 是採樣時鐘數（ T_{SMP} ）。 C_{EX} 為PCB和焊盤的等效電容，並與 R_{EX} 組合成低通濾波器。

$$\text{公式 1: } R_{EX} < \frac{k}{f_{ADC} \times C_{IN} \times \ln(2^{N+2})} - R_{IN}$$

或者在 R_{EX} 為已知條件下，可以改寫成公式 2，求得需要採樣時鐘數。

公式 2:

$$k > f_{ADC} \times \ln(2^{N+2}) \times C_{IN} \times (R_{EX} + R_{IN}) \cong 9.7 \times f_{ADC} \times C_{IN} \times (R_{EX} + R_{IN})$$

往往 R_{EX} 大於10分之一的 R_{IN} ，需要考慮 C_{EX} 的影響時候，可以簡化成公式 3。

公式 3:

$$k > f_{ADC} \times \ln(2^{N+2}) \times (C_{EX} + C_{IN}) \times (R_{EX} + R_{IN}) \cong 9.7 \times f_{ADC} \times (C_{EX} + C_{IN}) \times (R_{EX} + R_{IN})$$

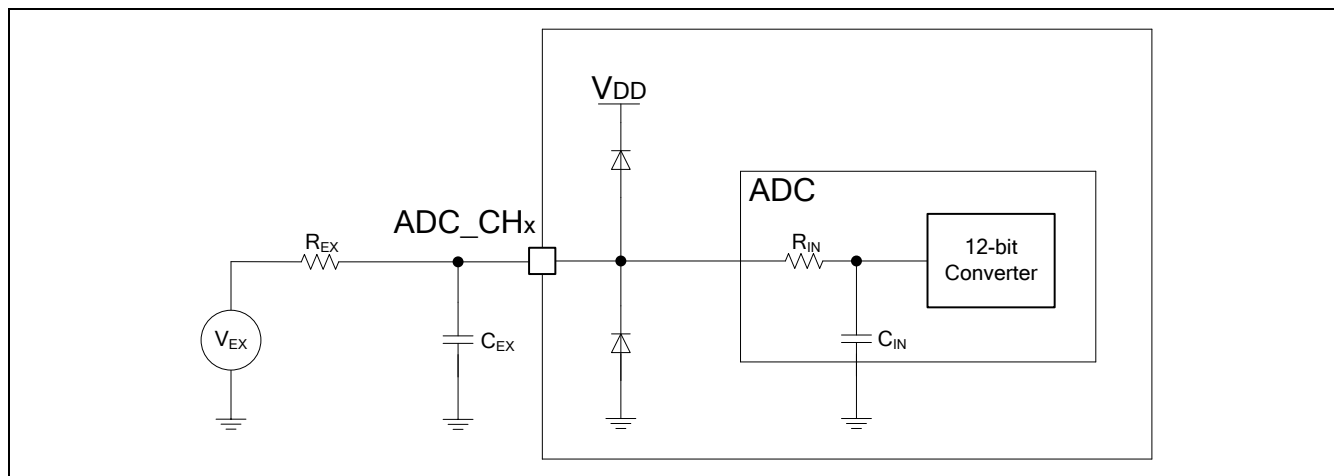


圖 2-7 ADC 等效電路

如表 2-3 所示，假設 C_{IN} 及 R_{IN} 為固定值，分別為 2.9 pF 及 2 k Ω 。當 ADC 時鐘頻率為 34 MHz 及 R_{EX} 為 50 k Ω 條件下，透過公式 2 求得 k 採樣時鐘數必須要大於 49.73384 個 ADC 時鐘週期。 ， T_{SMP} 值設定為 50 個 ADC 時鐘週期。如果考慮公式 3，假設 $C_{EX} = 5$ pF，k 採樣時鐘數必須要大於 135.482 個 ADC 時鐘週期， T_{SMP} 值設定為 136 個 ADC 時鐘週期。

如果要驗證充電時間是否足夠，可以將 T_{SMP} 值設定為最大值(256 個 ADC 時鐘週期)，將 ADC 轉換結果跟公式 2 或者公式 3 的 T_{SMP} 設定來比較。如果 ADC 轉換結果是一樣，就表示公式 2 或者公式 3 的 T_{SMP} 設定可以對內部採樣電容(C_{IN})完全充電。

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
$C_{IN}^{[*1]}$	Internal Capacitance	-	2.9	-	pF	
$R_{IN}^{[*1]}$	Internal Switch Resistance	-	-	2	k Ω	
R_{EX}	External input impedance	-	-	50	k Ω	
Notes: 1. Guaranteed by characterization result, not tested in production.						

表 2-3 ADC 等效阻抗及電容值

2.6 ADC 操作模式

ADC包含四種操作模式：單次觸發模式，Burst模式，單週期掃描模式和連續掃描模式。

2.6.1 單次觸發模式

在單次觸發模式下，ADC轉換僅在一個指定通道上執行一次。操作步驟如下：

1. 由軟件或外部輸入觸發，ADST(ADC_ADCR[11])被設置為 1，ADC 開始轉換。
2. 當 ADC 轉換完成時，轉換值將儲存在對應通道的 ADC 數據寄存器中。
3. ADC 轉換完成後，ADF(ADSR0[0])將被設置為 1。此時如果 ADIE(ADC[1])設置為 1，將產生 ADC 中斷要求。
4. ADC 轉換期間 ADST 保持為 1。當 ADC 轉換結束時，ADST 自動清零，ADC 控制器進入空閒狀態。

注1：在單次觸發模式，如果軟體使能多於一個通道，編號最小的通道將會被轉換，其他通道被忽略。

注2：如果在ADC正在轉換，一ADST被軟件清零，BUSY將立即清零，則ADC無法完成當前轉換，ADC控制器直接進入空閒狀態。

2.6.2 Burst 模式

ADC控制器對一個指定通道進行採樣和轉換，並將結果依次儲存到FIFO中（最多8個採樣）。操作步驟如下：

1. 當通過軟件或外部觸發將 ADST(ADC_ADCR[11])設置為 1 時，開始轉換編號最小的通道。
2. 當指定通道的 ADC 轉換完成後，其轉換結果將按順序傳送到 FIFO 裡，其值只能從 ADC_ADDR0 寄存器訪問。
3. 當在 FIFO 中超過 4 次採樣結果，ADF(ADSR0[0])會被設置為 1。此時如果 ADIE(ADC[1])設置為 1，將產生 ADC 中斷要求。
4. 只要 ADST 保持為 1，將一直重複步驟 2 和 3。當 ADST 位由軟件清零時，ADC 無法完成當前轉換，ADC 控制器直接進入空閒狀態

注1：如果軟件在Burst模式下，設定ADCHER開啟多個通道，則只會選擇編號最小的通道，而忽略其它啟用的通道。

注2：用戶可以通過重複讀取ADC_ADDR0來獲得轉換結果，直到VALIDF(ADSR0[8])變為0。例如，如果FIFO有4個轉換結果，則需要讀取ADC_ADDR0寄存器4次才能得到所有結果。

2.6.3 單週期掃描模式

在單週期掃描模式下，ADC控制器根據該ADCHER寄存器設置，從最小數量啟用通道到最大數量啟用通道，依序對所有指定通道進行採樣和轉換。操作步驟如下：

1. 當通過軟件或外部觸發輸入將 ADST(ADC_ADCR[11])設置為 1 時，在 ADCHER 具有最小編號的使能通道上啟動 ADC 轉換。
2. 當每個使能通道的 ADC 轉換完成時，結果將按順序傳送到與每個通道對應的 ADC 數據寄存器。
3. 當所有使能通道的轉換完成時，ADF(ADSR0[0])設置為 1。此時如果 ADIE(ADC[1])設置為 1，將產生 ADC 中斷要求。
4. ADC 完成一個週期轉換後，ADST 自動清零，ADC 控制器進入空閒狀態。如果在完成所有使能的 ADC 通道轉換之前將 ADST 清為零，則 ADC 無法完成當前轉換而 ADC 控制器直接進入空閒狀態。

2.6.4 連續掃描模式

在連續掃描模式下，在ADCHER寄存器使能的指定通道上依次執行ADC轉換。操作步驟如下：

1. 當通過軟件或外部觸發，將 ADST(ADC_ADCR[11])設置為 1 時，在具有最小編號的使能通道上啟動 ADC 轉換。
2. 當每個使能通道的 ADC 轉換完成時，每個使能通道的結果存儲在與每個使能通道對應的 ADC 數據寄存器中。
3. 當 ADC 控制器按順序完成所有使能通道的轉換時，ADF(ADSR0[0])將被設置為 1。此時如果 ADIE(ADC[1])設置為 1，將產生 ADC 中斷要求。
4. 只要 ADST 保持為 1，就會重複步驟 2~3。當 ADST 由軟件清零時，ADC 無法完成當前的轉換，ADC 控制器直接進入空閒狀態。

2.7 校準

為了獲得最好的轉換結果，可以透過校準功能對偏移及不匹配進行補償。然後，在正常操作模式中，校準值透過控制電容器陣列減少偏移和不匹配的影響，增加ADC精準度。建議在使用ADC之前、參考電壓改變後或者輸入模式切換後，都需要重新校準ADC。

校準操作流程：

1. 將 CALEN(ADC_ADCALR [0])設置為 1，ADC 切換為校準模式。
2. 當 ADST(ADC_ADCR [11])設置為 1 時，校準開始。
3. 校準完成後，旗標 CALIF (ADC_ADCALSTSR [0]) 會被設置為 1，ADC 將自動切換到正常模式。

2.8 直接記憶體存取控制器(PDMA)

如果PTEN(ADCR[9])設置為1，則ADC控制器可在ADC轉換完成時將轉換結果寫入到ADPDMA寄存器，並向PDMA發送請求。PDMA可以將ADC結果傳輸到指定的SRAM存儲空間，而無需CPU的參與。無論選擇哪種通道，PDMA操作的源位址都固定在ADPDMA寄存器中。當PDMA傳輸轉換結果時，如果ADC的工作模式為Burst模式，單掃描模式或連續掃描模式，ADC將繼續轉換下一個選定的通道。如果ADC完成所選通道的轉換並且PDMA尚未傳輸同一通道的最後轉換結果，則相應通道的OVERRUN(ADSR2[31:0])位將置1，最後的ADC轉換結果將為被新的ADC轉換結果覆蓋。PDMA會將所選通道的最新數據傳輸到用戶指定的目標地址。

為了達到ADC 2Msps採樣率，必須借助PDMA，將ADC轉換結果傳輸到指定的SRAM存儲空間。以下四點需要特別注意：

1. PDMA的時鐘源為HCLK，HCLK必須要大於等於ADC時鐘頻率，讓PDMA收到ADC請求後，可以立即傳輸資料。
2. ADC會依照VALID位，向PDMA發送請求。在啟動PDMA前，需要將每個ADC通道的VALID位清為0，避免SRAM存儲空間的資料產生問題。
3. PDMA支持多組資料傳輸通道，避免PDMA被其它外設占用，ADC資料傳輸通道必須為最高優先權。
4. 在PDMA資料傳輸的過程中，如果OVERRUN位從0變成1。表示PDMA來不及服務ADC資料傳輸通道。此時建議使用ADC Burst模式，一旦PDMA被其它外設占用，ADC可以預先將資料存在FIFO裡，避免資料被覆蓋掉，產生問題。

2.9 數位比較器

ADC支援數位比較器，其控制流程跟類比較器一樣，拿來監控類比電壓信號。以溫控器為例，溫度變化時ADC轉換結果也會跟著改變。利用數位比較器，程序不需要判斷每次ADC轉換結果。一旦ADC轉換結果符合比較條件，利用數位比較器中斷，程序再去處理溫度過高或者過低的流程，減少CPU處理時間。

操作模式可配置為閾值模式或區間模式，以監控指定通道的ADC轉換結果。圖 2-8所示，閾值模式和區間模式的比較範圍，一旦ADC結果落在白色區塊，就會產生數位比較器中斷旗標。此時如果CMP1E(ADCMPRx[1])設置為1，將產生ADC 中斷要求。

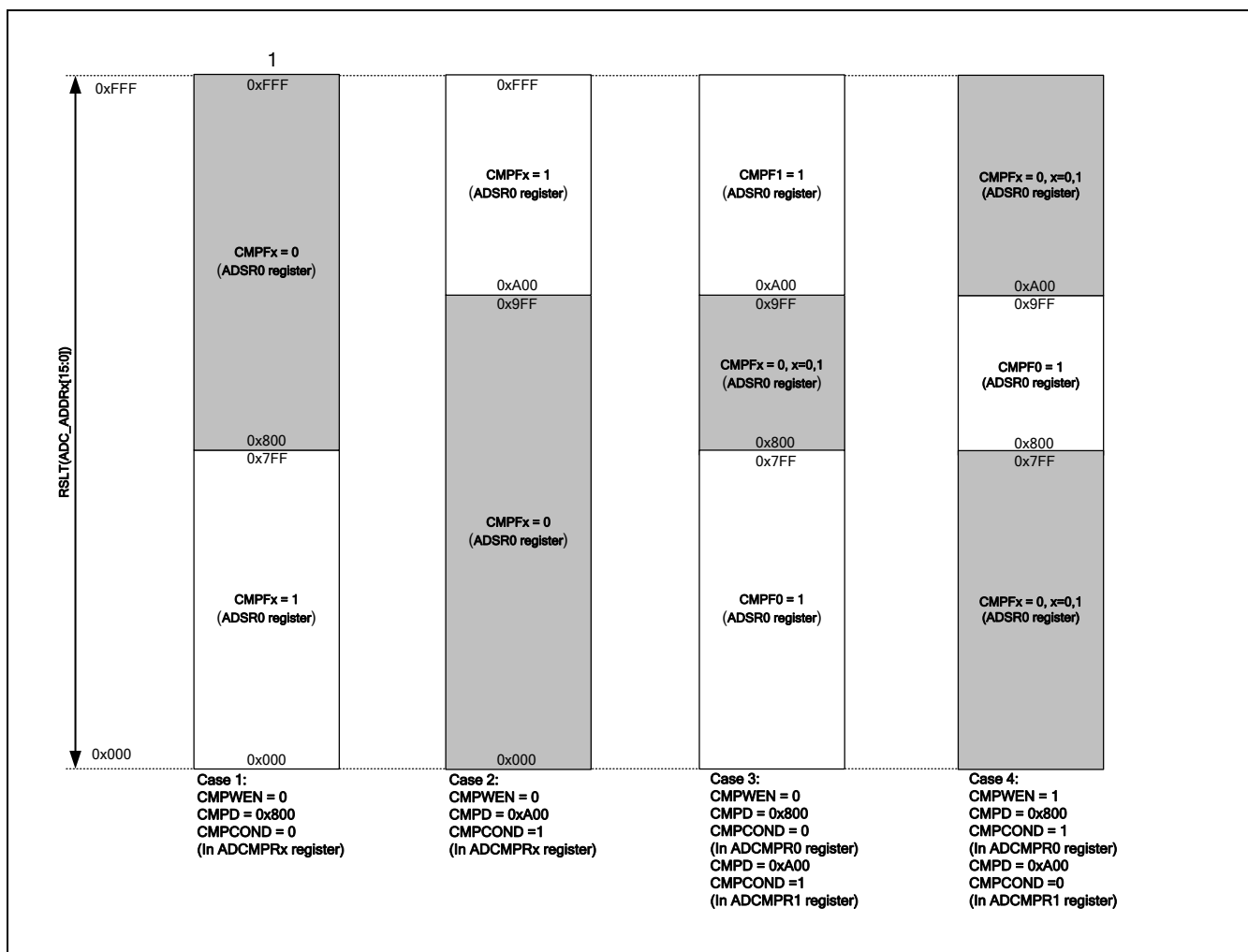


圖 2-8 A/D 數字比較器比較範圍

2.9.1 閾值模式

如圖 2-9所示，有兩個比較寄存器ADCMPRx(x=0,1)，用於監視兩個指定通道的ADC結果。CMPCH(ADCMPRx[7:3])用於選擇監控通道。CMPCOND(ADCMPRx[2])用於確定比較條件。如果CMPCOND清為零，當轉換結果小於CMPD(ADCMPRx[27:16])中指定的閾值時，內部匹配計數器將增加 1；如果 CMPCOND 位設置為 1，則當轉換結果大於或等於

CMPD(ADCMPRx[27:16])中指定的閾值時，內部匹配計數器將增加1。

當CMPCH指定的通道轉換(ADCMPRx[7:3])完成後，比較操作將自動觸發一次。當比較結果滿足設置時，比較匹配計數器將增加1，否則，比較匹配計數器將被清除為0。一旦匹配計數器達到設置(CMPMATCNT+1)，中斷旗標CMPF0/1(ADSR0[1]/[2])將被設置為1，如果CMPIE(ADCMPRx[1])為1，則會產生ADC中斷請求。

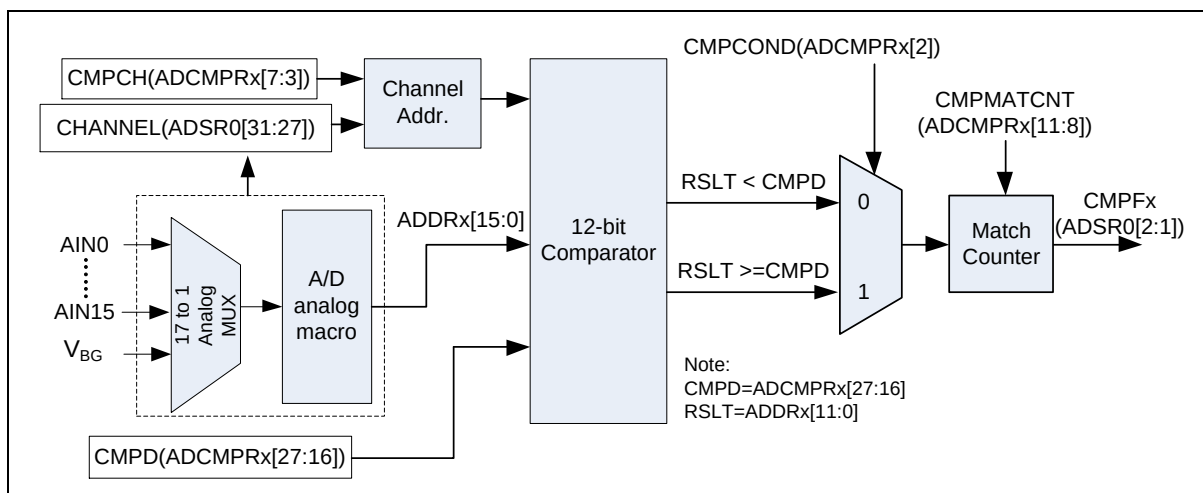


圖 2-9 ADC 數位比較器框圖

如圖 2-8範例3所示，兩個比較寄存器也可以設定監視相同ADC的通道，但兩者偵測的範圍不能重複。以水壓感測器為例，一旦待測物輸出的水壓不在偵測的範圍內，也就表示水壓處於正常狀態。一旦水壓過高或者過低，對應的中斷旗標CMPF0/1(ADSR0[1]/[2])會被設置為1。此時進入中斷程序，藉由中斷旗標可以快速判斷水壓狀態。

2.9.2 區間模式

如圖 2-8範例4所示，當CMPWEN(ADCMPR0[15])位設置為1時，此時為區間模式比較器，需要設定二個比較條件，決定比較區間的範圍在CMPD(ADCMPR0[27:16])和CMPD(ADCMPR1[27:16])之間。一旦同時滿足時，CMPF0(ADSR0[1])才會置1。

以麥克風感測器應用為例，區間模式可以用來偵測靜音狀態。當麥克風感測器接收的聲音變小時，感測器輸出的電壓振幅同時間也會變小。一旦電壓振幅落在區間模式比較器的範圍內，就表示聲音過小，由此就可以判定目前的聲音是處於靜音狀態。

3 範例程式碼

本範例程式碼包含在M031/M032的M031_Series_BSP內，可以透過以下路徑到官方網頁下載：

https://www.nuvoton.com/hq/support/tool-and-software/software/dirver/?_locale=zh_TW

在使用此範例程式前，請先將M031_Series_BSP解壓縮，並按照以下路徑開啟範例程式。

\M031_Series_BSP\CMSIS_V3.00.000\SampleCode\StdDriver\PDMA_ADC_1882ksps_ContinuousScanMode\KEIL

範例程式可以在M031/M032開發板上運行，會以連續掃描模式對PB.0，PB.1，PB.2或PB.3進行採樣，並且透過PDMA將ADC轉換結果，搬到SRAM存儲空間。

```
#include <stdio.h>
#include "NuMicro.h"

/*-----*/
/* Define global variables and constants */
/*-----*/

#define ADCDatalenght 16
#define ADCextendSampling 0
#define PDMAchannel 1

volatile uint32_t g_u32ADCounter = 0;
volatile uint32_t g_u32IsTestOver = 0;
//PDMA software flag - 0: Busy/Idle, 1 = Done ,2 = Abort
volatile uint32_t g_u32ResultSum = 0;
volatile uint32_t g_u32ADChannel = 0;
int16_t g_i32ConversionData[ADCDatalenght+1] = {0};

void SYS_Init(void)
{
    /* Unlock protected registers */
    SYS_UnlockReg();
    /* Set XT1_OUT(PF.2) and XT1_IN(PF.3) to input mode */
    GPIO_SetMode(PF, BIT2 | BIT3, GPIO_MODE_INPUT);
    /* Enable HXT (4~32 MHz) */
    CLK_EnableXtalRC(CLK_PWRCTL_HXTEN_Msk);
    /* Waiting for 32MHz clock ready */
    CLK_WaitClockReady(CLK_STATUS_HXTSTB_Msk);
    /* Select HCLK clock source as HIRC and HCLK source divider as 1 */
    CLK_SetHCLK(CLK_CLKSEL0_HCLKSEL_HIRC, CLK_CLKDIV0_HCLK(1));
    /* Enable UART module clock */
```



```

CLK_EnableModuleClock(UART0_MODULE);
/* Switch UART0 clock source to XTAL */
CLK_SetModuleClock(UART0_MODULE, CLK_CLKSEL1_UART0SEL_HIRC, CLK_CLKDIV0_UART0(1));
/* Enable ADC module clock */
CLK_EnableModuleClock(ADC_MODULE);
/* ADC clock source is HXT 12MHz, set divider to 8, ADC clock is 12/8 MHz */
CLK_SetModuleClock(ADC_MODULE, CLK_CLKSEL2_ADCSEL_HXT, CLK_CLKDIV0_ADC(1));
/* Enable PDMA clock source */
CLK_EnableModuleClock(PDMA_MODULE);
/* Update System Core Clock */
/* User can use SystemCoreClockUpdate() to calculate PllClock, SystemCoreClock and
CyclesPerUs automatically. */
SystemCoreClockUpdate();

/*-----*/
/* Init I/O Multi-function */
/*-----*/

/* Set GPB multi-function pins for UART0 RXD and TXD */
SYS->GPB_MFPH = (SYS->GPB_MFPH & ~(SYS_GPB_MFPH_PB12MFP_Msk | SYS_GPB_MFPH_PB13MFP_Msk)) \
    | (SYS_GPB_MFPH_PB12MFP_UART0_RXD | SYS_GPB_MFPH_PB13MFP_UART0_TXD);
/* Configure the GPB0 - GPB3 ADC analog input pins. */
SYS->GPB_MFPL = (SYS->GPB_MFPL & ~(SYS_GPB_MFPL_PB0MFP_Msk | SYS_GPB_MFPL_PB1MFP_Msk |
    SYS_GPB_MFPL_PB2MFP_Msk | SYS_GPB_MFPL_PB3MFP_Msk)) | (SYS_GPB_MFPL_PB0MFP_ADC_CH0 |
    SYS_GPB_MFPL_PB1MFP_ADC_CH1) | (SYS_GPB_MFPL_PB2MFP_ADC_CH2 |
    SYS_GPB_MFPL_PB3MFP_ADC_CH3);
/* Set PB.0 ~ PB.3 to input mode */
GPIO_SetMode(PB, BIT0|BIT1|BIT2|BIT3, GPIO_MODE_INPUT);
/* Disable the PB0 ~ PB3 digital input path to avoid the leakage current. */
GPIO_DISABLE_DIGITAL_PATH(PB, BIT0|BIT1|BIT2|BIT3);
/* Lock protected registers */
SYS_LockReg();
}

void PDMA_Init()
{
    /* Configure PDMA peripheral mode form ADC to memory */
    /* Open PDMA Channel 1 based on PDMA channel setting*/
    PDMA_Open(PDMA, 1 << PDMAchannel);
    /* transfer width is half word(16 bit) and transfer count is ADCDataLength+1 */
    PDMA_SetTransferCnt(PDMA, PDMAchannel, PDMA_WIDTH_16, ADCDataLength+1);
    /* Set source address as ADC data register (no increment) and destination address as
    g_i32ConversionData array (increment) */

```

```

    PDMA_SetTransferAddr(PDMA, PDMAchannel, (uint32_t)&ADC->ADPDMA, PDMA_SAR_FIX,
    (uint32_t)g_i32ConversionData, PDMA_DAR_INC);
    /* Select PDMA request source as ADC RX */
    PDMA_SetTransferMode(PDMA, PDMAchannel, PDMA_ADC_RX, FALSE, 0);
    /* Set PDMA as single request type for ADC */
    PDMA_SetBurstType(PDMA, PDMAchannel, PDMA_REQ_SINGLE, PDMA_BURST_4);
    /* Enable PDMA interrupt */
    PDMA_EnableInt(PDMA, PDMAchannel, PDMA_INT_TRANS_DONE);
    NVIC_EnableIRQ(PDMA_IRQn);
}

void ReloadPDMA()
{
    /* transfer width is half word(16 bit) and transfer count is ADCDataLength+1 */
    PDMA_SetTransferCnt(PDMA, PDMAchannel, PDMA_WIDTH_16, ADCDataLength+1);
    /* Select PDMA request source as ADC RX */
    PDMA_SetTransferMode(PDMA, PDMAchannel, PDMA_ADC_RX, FALSE, NULL);
}

void ADC_FunctionTest()
{
    uint8_t u8Option;
    printf("\n");
    printf("+-----+\n");
    printf("|    High speed ADC conversion rate with PDMA transfer    |\n");
    printf("|    ADC clock source -> HXT = 32MHz                      |\n");
    printf("|    ADC extended sampling time = 0                        |\n");
    printf("|    ADC conversion time = 17 + ADC extended sampling time = 17 |\n");
    printf("|    ADC conversion rate = 32 MHz / 17 = 1.882MSPS         |\n");
    printf("+-----+\n");
    printf("\nIn this test, software will get 16 conversion results from the specified\n");
    printf("channel.\n");
    printf("\nEnable ADC Power and then calibrate ADC.\n");
    /* Enable ADC converter */
    ADC_POWER_ON(ADC);
    /*Wait for ADC internal power ready*/
    CLK_SysTickDelay(10000);

    /* Set input mode as single-end, and Single mode*/
    ADC_Open(ADC, ADC_ADCR_DIFFEN_SINGLE_END, ADC_ADCR_ADMD_CONTINUOUS, NULL);
}

```

```

/* Set extend sampling time based on external resistor value.*/
ADC_SetExtendSampleTime(ADC, NULL, ADCExtendSampling);

while(1)
{
    /* reload PDMA configuration for next transmission */
    ReloadPDMA();

    printf("\nSelect input ADC channel from 0 to 3:\n");
    printf(" [0] Single end input channel 0\n");
    printf(" [1] Single end input channel 1\n");
    printf(" [2] Single end input channel 2\n");
    printf(" [3] Single end input channel 3\n");
    printf(" Other keys: exit ADC + PDMA test\n");
    printf(" Please input key.\n");
    u8Option = getchar();
    if(u8Option == '0')
        g_u32ADChannel = 0;
    else if(u8Option == '1')
        g_u32ADChannel = 1;
    else if(u8Option == '2')
        g_u32ADChannel = 2;
    else if(u8Option == '3')
        g_u32ADChannel = 3;
    else
        return ;

    /* Select ADC input channel */
    ADC_SET_INPUT_CHANNEL(ADC, 0x1 << g_u32ADChannel);
    /* ADC enable PDMA transfer */
    ADC_ENABLE_PDMA(ADC);
    /* Start ADC conversion */
    ADC_START_CONV(ADC);
    /* Wait PDMA interrupt (g_u32IsTestOver will be set at IRQ_Handler function) */
    while(g_u32IsTestOver == 0);
    /* Check transfer result */
    if (g_u32IsTestOver == 1)
        printf("PDMA trasnfer done...\n");
    else if (g_u32IsTestOver == 2)
        printf("PDMA trasnfer abort...\n");
    /* Clear g_u32IsTestOver software flag */

```

```
g_u32IsTestOver = 0;
/* Stop ADC conversion */
ADC_STOP_CONV(ADC);
/* Disable PDMA function of ADC */
ADC_DISABLE_PDMA(ADC);

/* Calculate average and print ADC result except first sampling data result that
belongs to previous channel*/
g_u32ResultSum = 0;
printf("\nConversion result of channel %d:\n",g_u32ADChannel);
for(g_u32ADCounter = 1; (g_u32ADCounter) < (ADCDataLength+1); g_u32ADCounter++)
{
    g_u32ResultSum += g_i32ConversionData[g_u32ADCounter];
    printf("%2d:0x%X (%d)\n", g_u32ADCounter, g_i32ConversionData[g_u32ADCounter],
g_i32ConversionData[g_u32ADCounter]);
}
g_u32ResultSum /= (ADCDataLength);
printf("Average Result : 0x%X (%d)\n",g_u32ResultSum, g_u32ResultSum);
}
}

void PDMA_IRQHandler(void)
{
    uint32_t status = PDMA_GET_INT_STATUS(PDMA);
    if(status & PDMA_INTSTS_ABTFIF_Msk)    /* abort */
    {
        if(PDMA_GET_ABORT_STS(PDMA) & PDMA_ABTFSTS_ABTFIF1_Msk)
            g_u32IsTestOver = 2;
        PDMA_CLR_ABORT_FLAG(PDMA, PDMA_ABTFSTS_ABTFIF1_Msk);
    }
    else if(status & PDMA_INTSTS_TDIF_Msk)    /* done */
    {
        if(PDMA_GET_TD_STS(PDMA) & PDMA_TDSTS_TDIF1_Msk)
            g_u32IsTestOver = 1;
        PDMA_CLR_TD_FLAG(PDMA, PDMA_TDSTS_TDIF1_Msk);
    }
    else
        printf("unknown PDMA interrupt !!\n");
}

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

```

/* Init UART0 */
/*-----*/
void UART0_Init(void)
{
    /* Reset UART0 */
    SYS_ResetModule(UART0_RST);
    /* Configure UART0 and set UART0 baud rate */
    UART_Open(UART0, 115200);
}

int32_t main(void)
{
    /* Init System, IP clock and multi-function I/O. */
    SYS_Init();
    /* Init UART0 for printf */
    UART0_Init();
    /* Init PDMA for ADC */
    PDMA_Init();
    printf("\nSystem clock rate: %d Hz", SystemCoreClock);
    /* ADC function test */
    ADC_FunctionTest();
    /* Disable ADC IP clock */
    CLK_DisableModuleClock(ADC_MODULE);
    /* Disable PDMA IP clock */
    CLK_DisableModuleClock(PDMA_MODULE);
    /* Disable PDMA Interrupt */
    NVIC_DisableIRQ(PDMA_IRQn);
    printf("Exit ADC sample code\n");
    while(1);
}

```

4 結論

本文基於應用角度，介紹類比數位轉換器的規格和特性。然後說明如何透過外部參考電壓管腳 V_{REF} 、 T_{SMP} 設定及校準，最小化ADC誤差並獲得最佳ADC精準度的方法和應用設計規則。一旦使用者掌握了這良好工作知識，就可以依照應用要求，基於速度、精準度及計算能力選擇最適合軟硬體設計參數。另外在ADC運行過程中，介紹利用PDMA快速地搬移資料或者利用數位比較器功能來偵測溫度感測器、壓力感測器及聲音感測器，減少進出ADC中斷次數，降低CPU負擔，讓CPU可以處理更多任務。

5 版本歷史

日期	版本	描述
2018.11.26	1.00	1. 初次發佈

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

*Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.*