Analog-to-Digital Converter (ADC)
類比數位轉換器

- 2013/5/13

課程大綱 (Lesson Outline)

- ADC類比數位轉換器之原理
  - Direct-Conversion直接轉換式
  - Successive Approximation 逐次逼近式
  - Delta-Sigma 三角積分調變式
- NuMicro ADC
  - Block Diagram 方塊圖
  - Programming Example 程式範例
- 週邊與感應器 (Peripheral & Sensors)
  - Thermister 熱敏電阻
  - Fuel Tank Detector 油箱偵測器
- 實習範例 (Exercise Examples)
  - 使用ADC 讀取可變電阻 (學習板上VR1)
  - 使用ADC 讀取熱敏電阻 (外接~1K或10K歐姆)
  - 使用ADC 讀取油箱感測器 (外接~100歐姆)

概觀

- one 12-bit successive approximation analog-to-digital conveters (SAR A/D converter)
- 8 input channels
- Three operation modes: single, single-cycle scan and continuous scan mode
- An A/D conversion can be started by
  - Software write 1 to ADST bit
  - External pin STADC
- Analog input voltage range: 0~V_{ref}

特性

- Analog input voltage range: 0~V_{ref}
- 12-bit resolution and 10-bit accuracy is guaranted
- 8 single-end analog input channels or 4 differential analog input channels
- Maximum ADC clock frequency is 16 MHz
- Up to 700K SPS conversion rate
- Channel 7 supports 3 input sources: external analog voltage, internal bandgap voltage, and internal temperature sensor output
- Support Self-calibration to minimize conversion error

NuMicro ADC pins

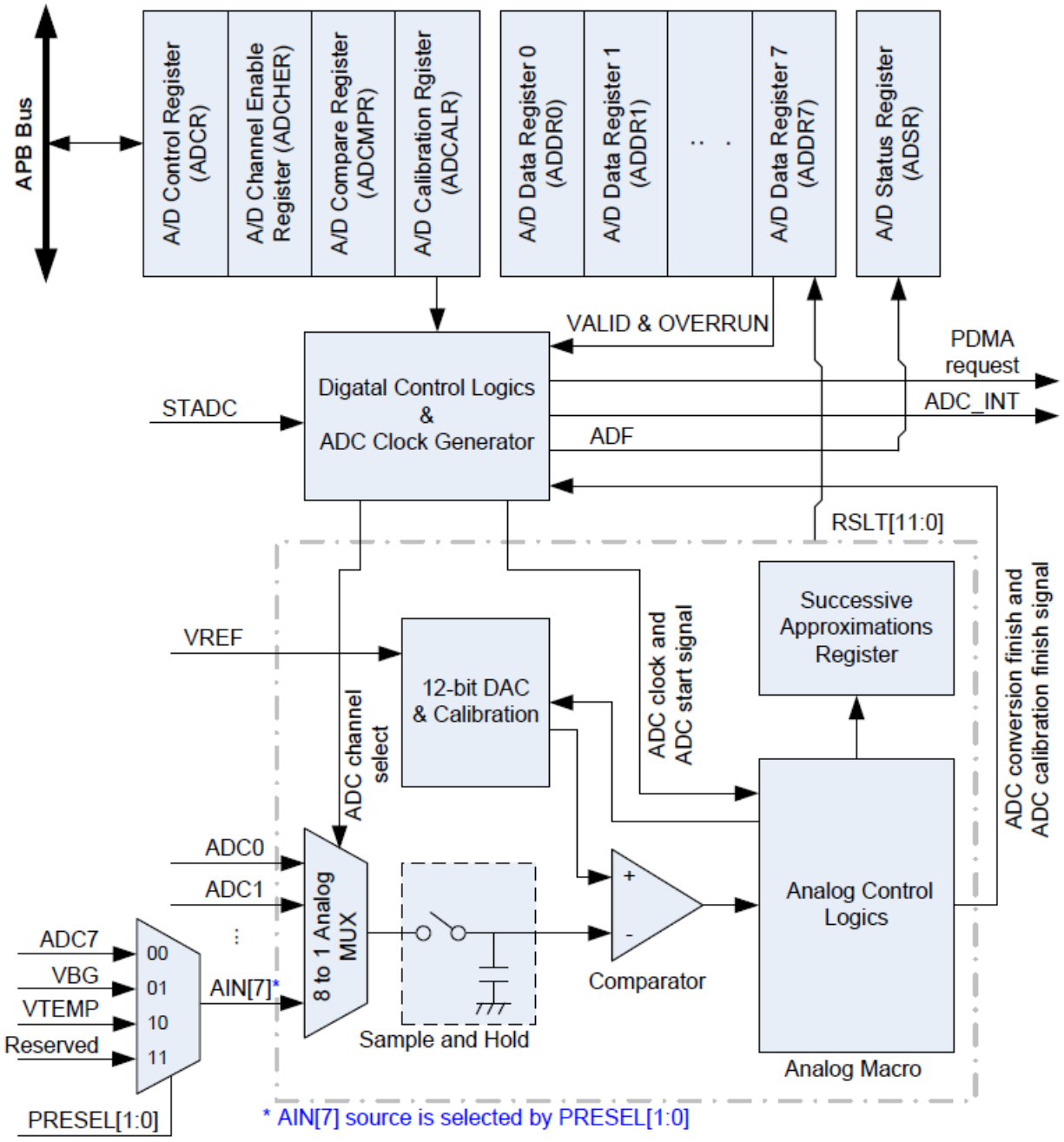
- The ADC input pins share with GPIO port A
- The ADC input pins must be configured in **input type**
- Single-end input mode
  - Channel 0 ~ Channel 7 share with GPA0~GPA7
- Differential input mode
  - Channel 0 ~ Channel 3
  - Differential input channels are the paired channels

Differential input paired channel	Single-end input channel	
0	0	1
1	2	3
2	4	5
3	6	7

特性

- Three operating modes：
- Single mode: A/D conversion is performed one time on a specified channel
- Single-cycle scan mode: A/D conversion is performed one cycle on all specified channels with the sequence from ch0-ch7
- Continuous scan mode: A/D converter continuously performs single-cycle scan mode until software stops A/D conversion

方塊圖

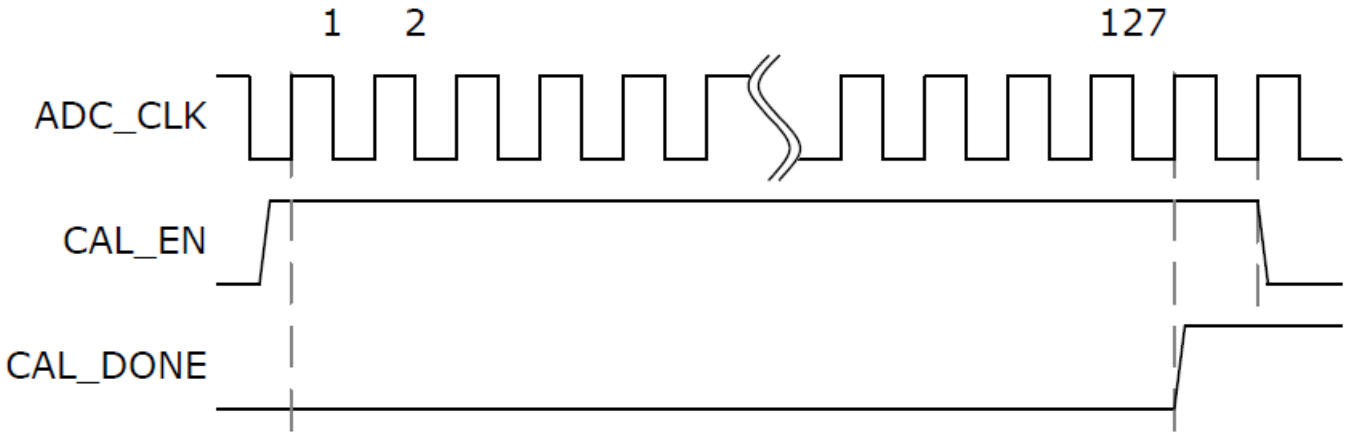

APB Bus
A/D Control Register (ADCR)
A/D Channel Enable Register (ADCHER)
A/D Compare Register (ADCMPR)
A/D Calibration Rgister (ADCALR)
A/D Data Register 0 (ADDR0)
A/D Data Register 1 (ADDR1)
A/D Data Register 7 (ADDR7)
A/D Status Register (ADSR)
VALID & OVERRUN
PDMA request
ADC_INT
ADF
STADC
Digatal Control Logics & ADC Clock Generator
RSLT[11:0]
VREF
12-bit DAC & Calibration
Successive Approximations Register
ADC clock and ADC start signal
ADC channel select
ADC conversion finish and ADC calibration finish signal
ADC0
ADC1
ADC7
VBG
VTEMP
Reserved
00
01
10
11
AIN[7]*
8 to 1 Analog MUX
Sample and Hold
Comparator
Analog Control Logics
Analog Macro
PRESEL[1:0]
* AIN[7] source is selected by PRESEL[1:0]

功能描述

- self calibration function: to minimize conversion error,
  - write 1 to CALEN bit in ADCALR register to enable calibration function,
  - while calibration finished the CAL_DONE bit will be set to 1.
- Change mode or channel: When changing the operating mode or analog input channel,
  - software must clear ADST bit to 0 in ADCR register.

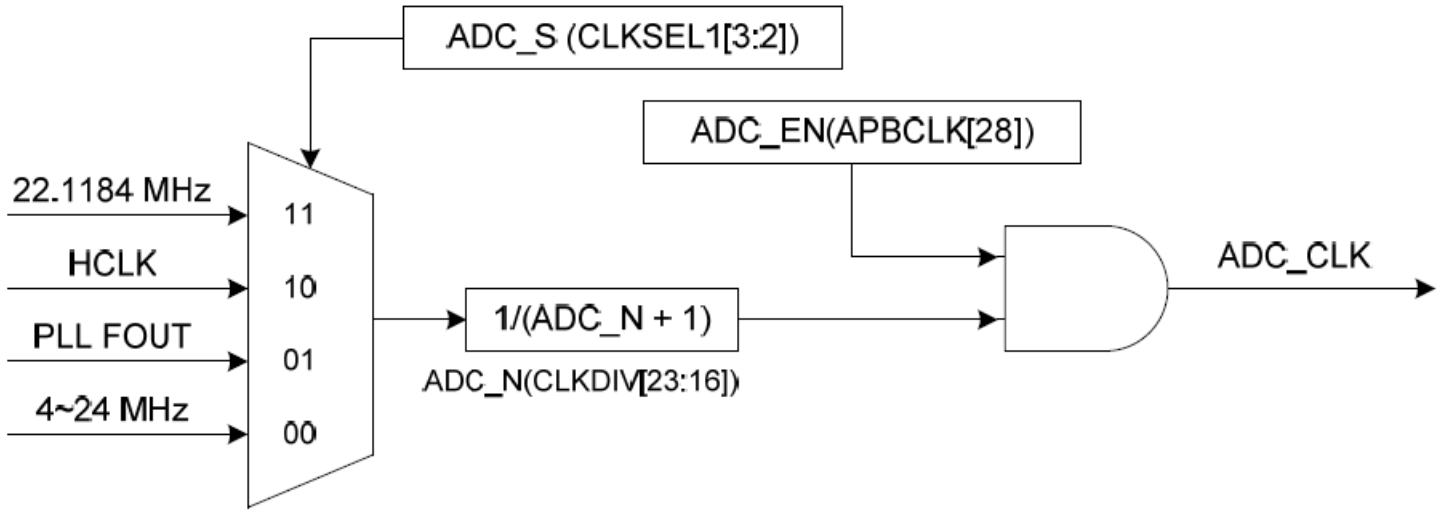
Self-Calibration

- self calibration function: minimize conversion error
- power on or switch ADC input type between single-end input and differential input
- write 1 to CALEN bit in ADCALR register to enable calibration function
- It needs 127 ADC clocks and the CAL_DONE bit will be set to 1

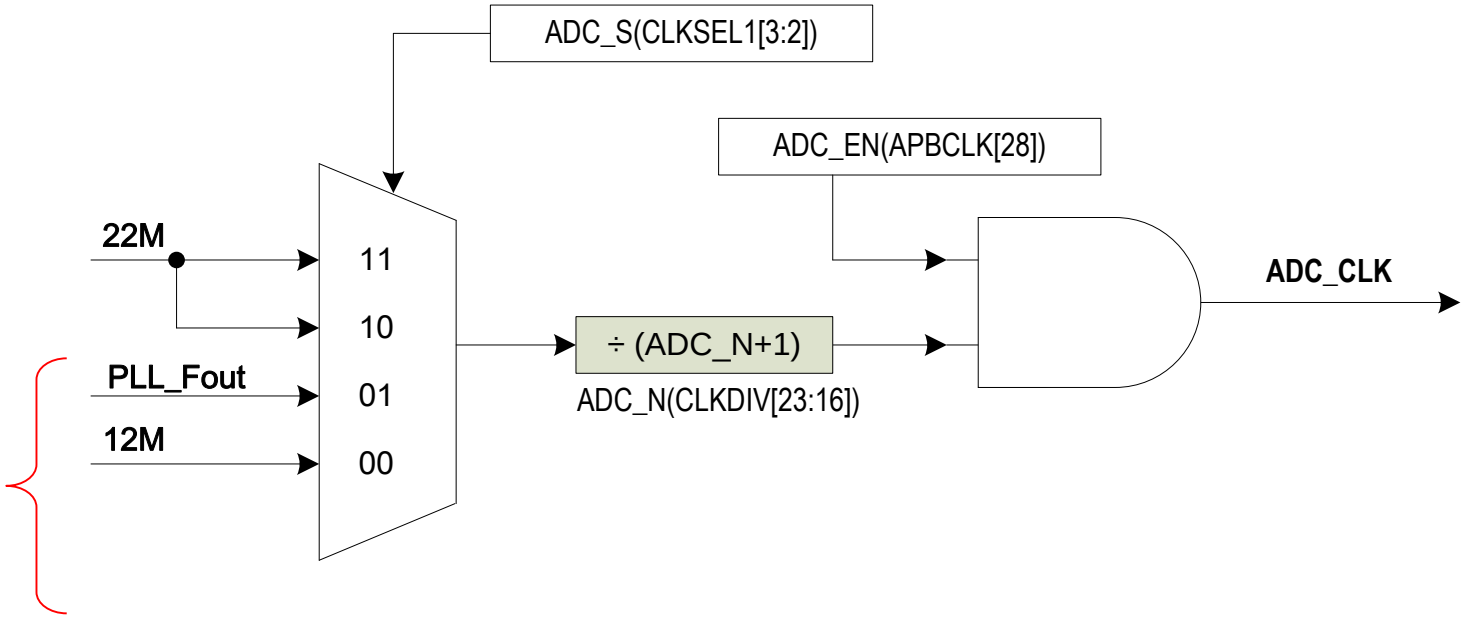


ADC Clock Generator

- The maximum sampling rate is up to 700K SPS.
- select clock sources: ADC_S (CLKSEL1[3:2]),
- ADC clock frequency = (ADC clock source frequency) / (ADC_N+1);
- ADC enable: ADC_EN(APBCLK[28])



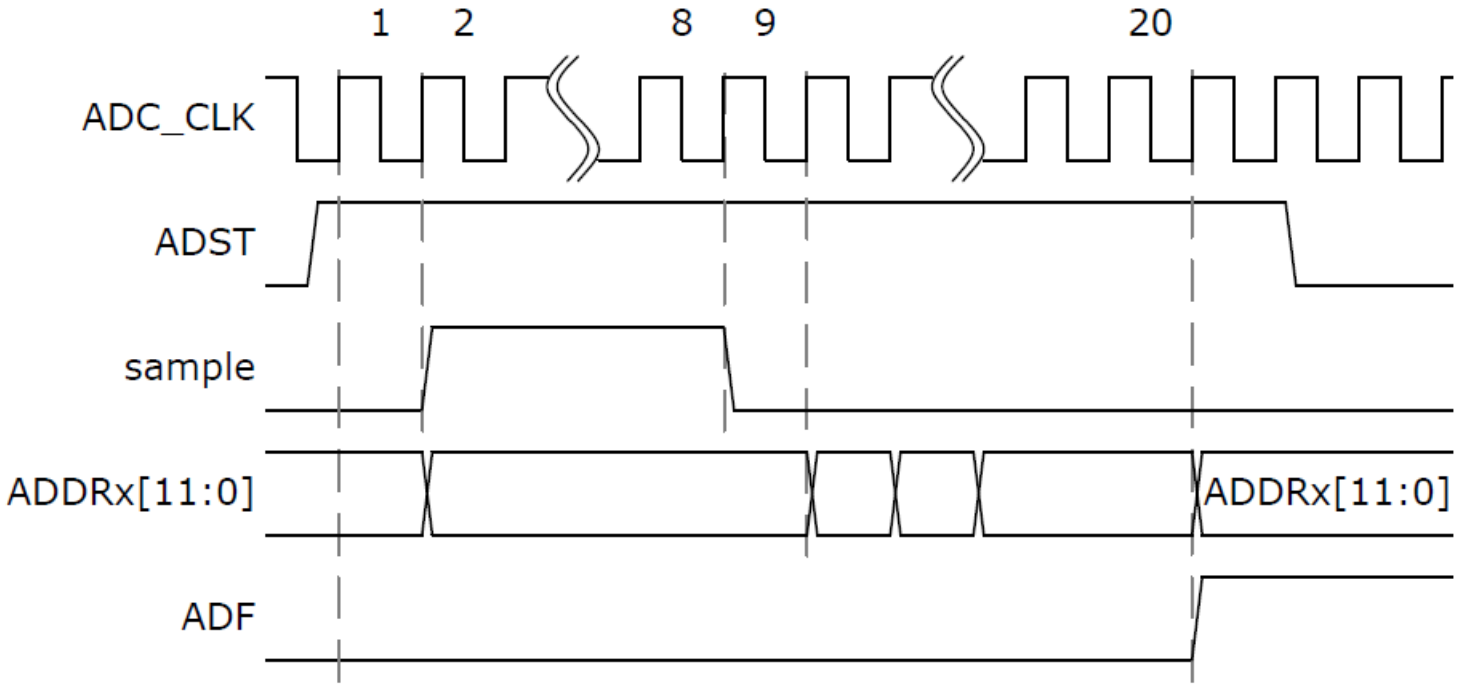
NuMicro ADC clock source



- The maximum ADC clock frequency is 16MHz
- The ADC clock frequency = (ADC clock source) / (ADC_N+1);
  ADC_N is an 8-bit pre-scaler

Single Mode

- 1. A/D started when ADCR.ADST= 1 or external trigger input.
- 2. A/D conversion is finished, result stored in the A/D data register.
- 3. ADSR. ADF=1. If the ADCR.ADIE =1, the ADC interrupt will be asserted.
- 4. ADST=1 until A/D conversion ends, then the ADST=0



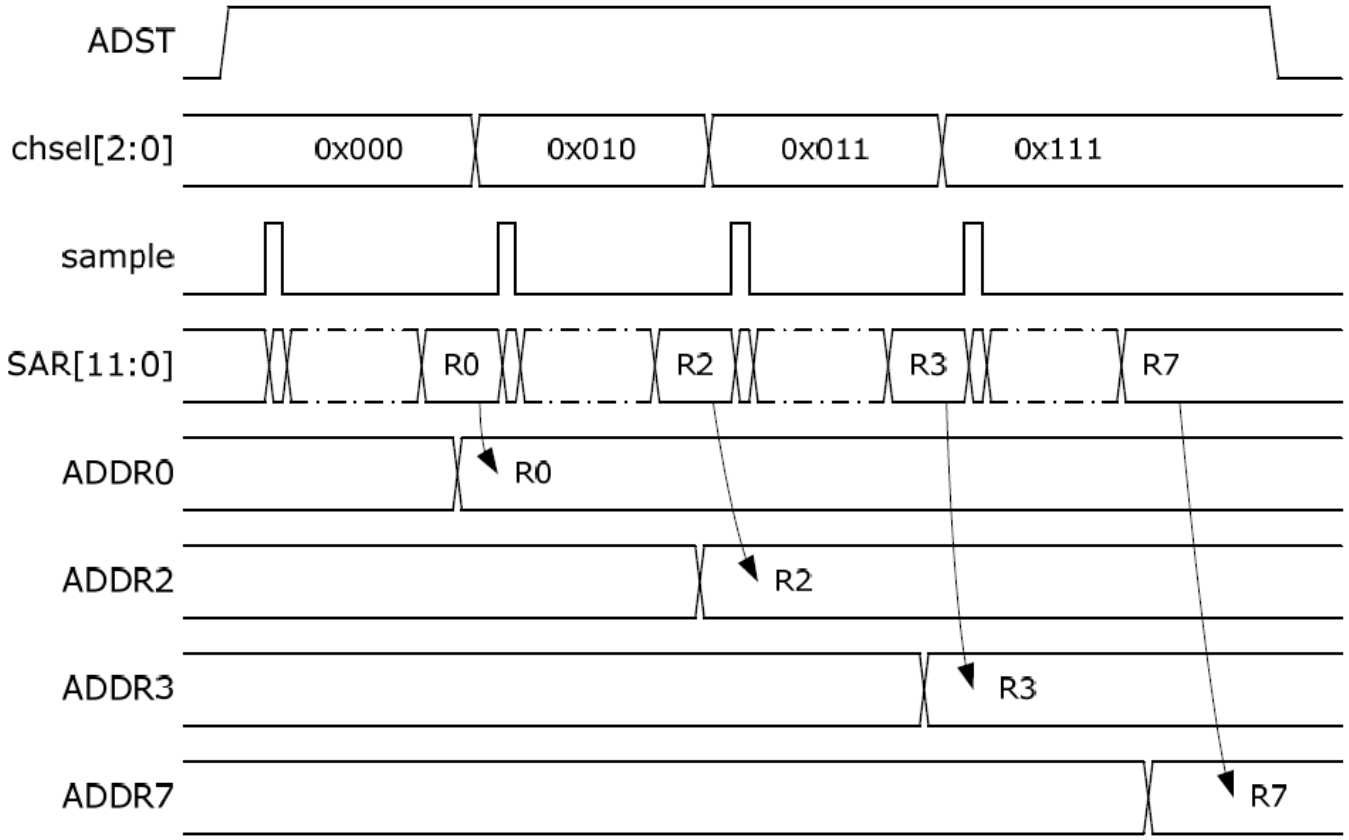
Single-Cycle Scan Mode

- A/D conversion will sample and convert the specified channels once in the sequence from the lowest number enabled channel to the highest number enabled channel.
- 1. A/D started when ADCR.ADST= 1 or external trigger input.
- 2. A/D conversion is finished, result stored in the A/D data register.
- 3. When finish all enabled channels, ADSR. ADF=1.

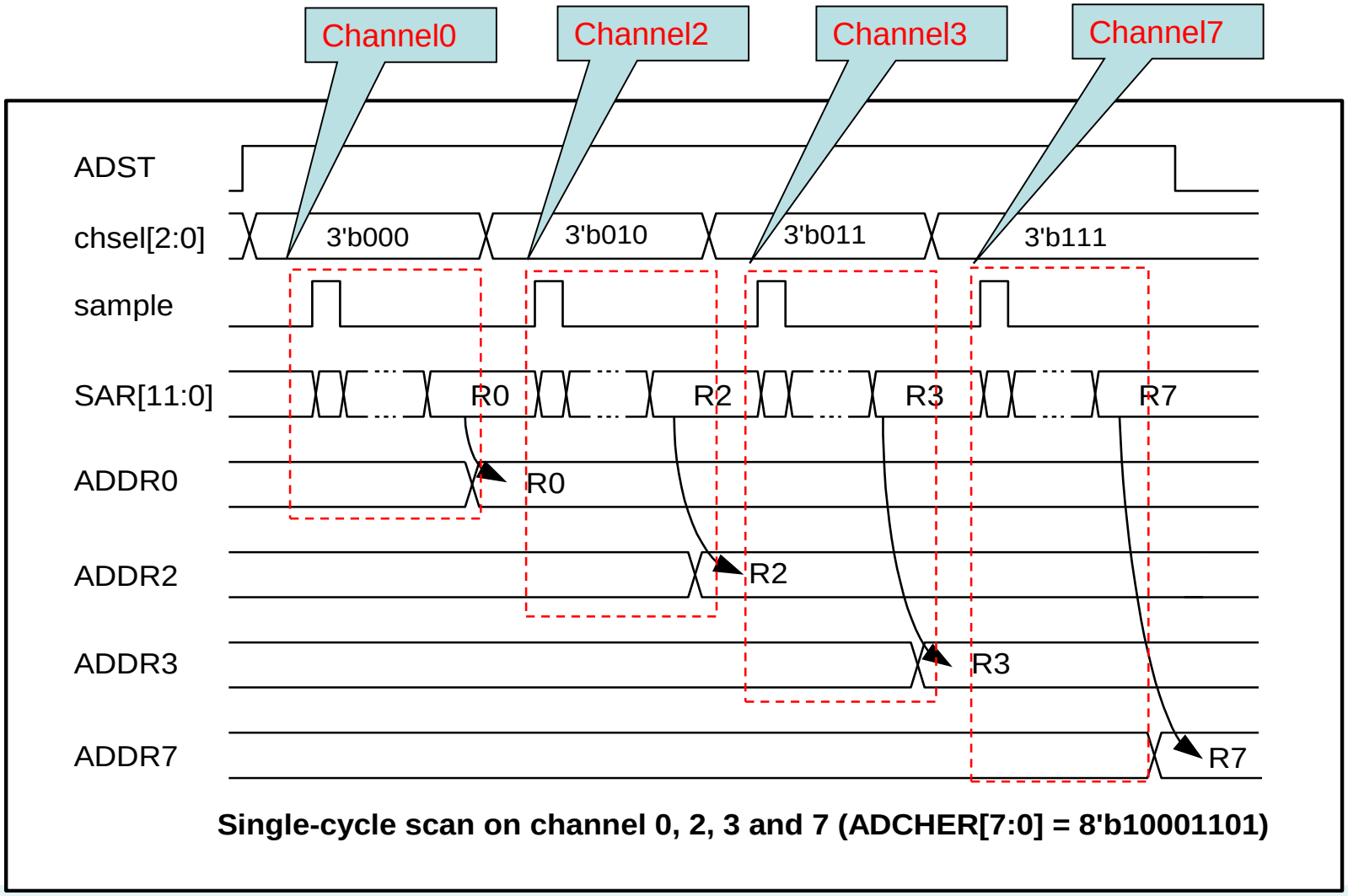
  If the ADCR.ADIE =1, the ADC interrupt will be asserted.
- 4. ADST=1 until A/D conversion ends, then the ADST=0

Single-Cycle Scan Mode

- An example timing diagram for single-cycle scan on enabled channels (0, 2, 3 and 7)



Single scan mode

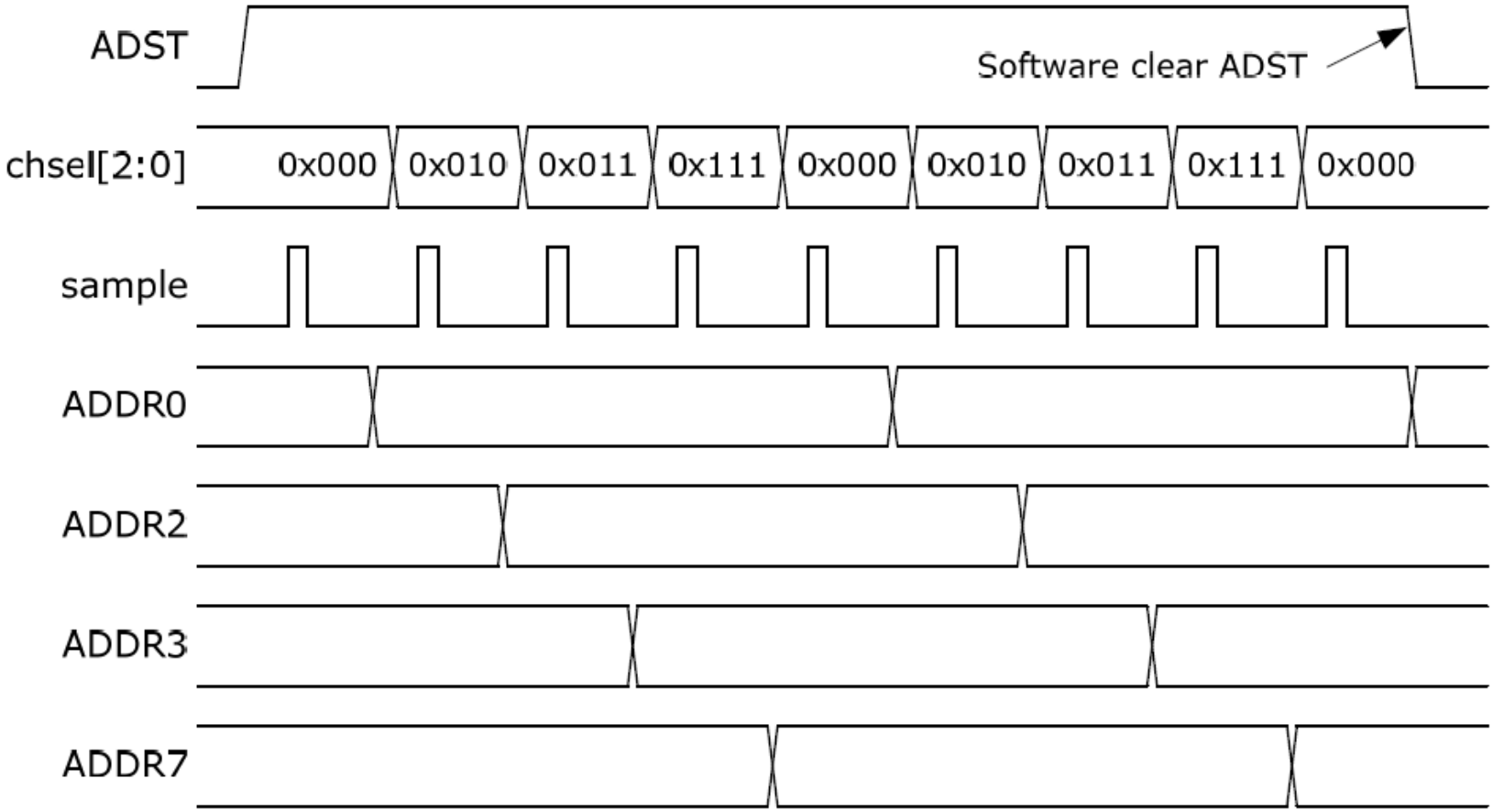


Single-cycle scan on channel 0, 2, 3 and 7 (ADCHER[7:0] = 8'b10001101)

Continuous Scan Mode

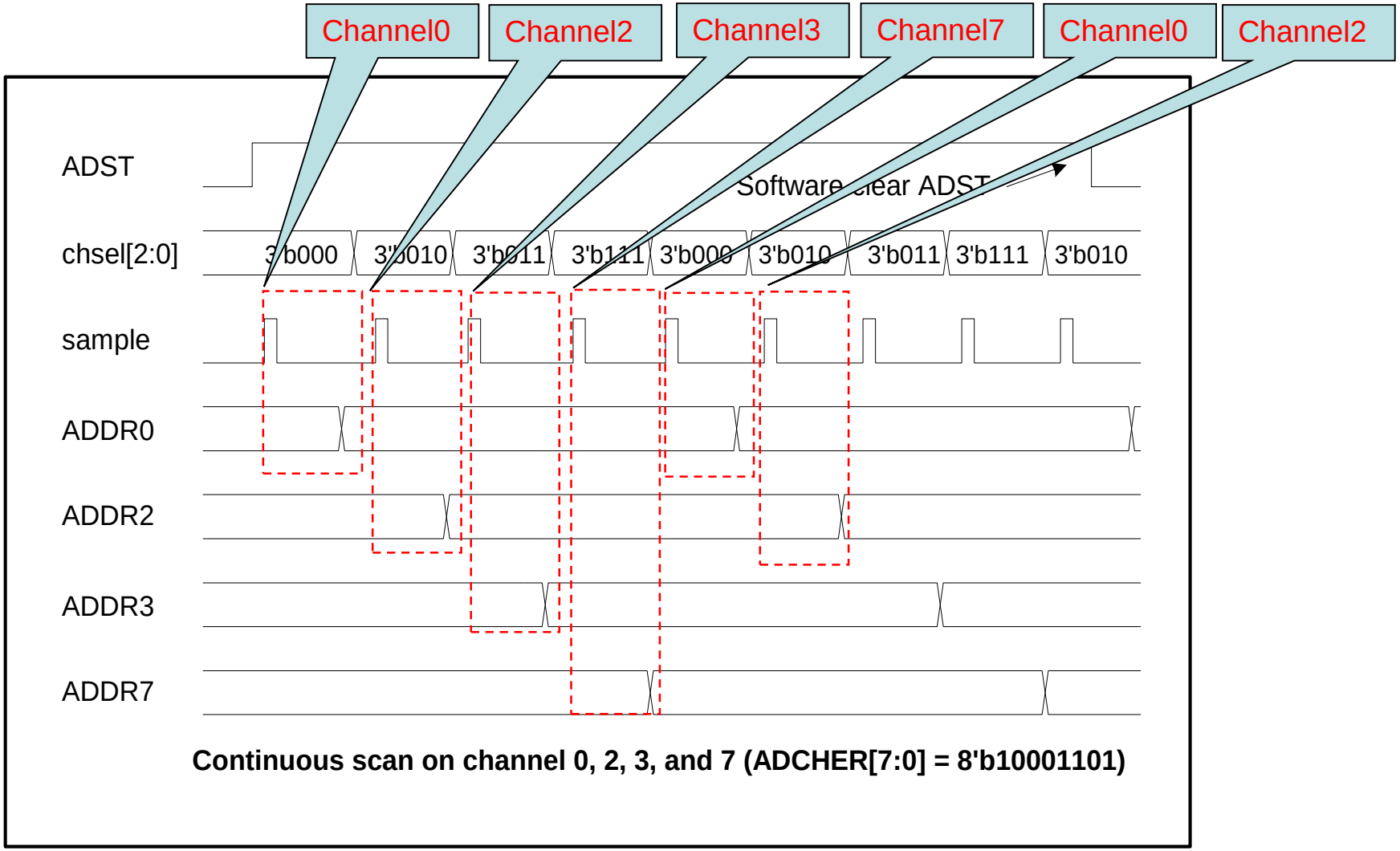
- A/D conversion is performed sequentially on the specified channels that enabled by CHEN bits in ADCHER register
- 1. A/D started when ADCR.ADST= 1 or external trigger input.
- 2. A/D conversion is finished, result stored in the A/D data register.
- 3. When A/DC completes all enabled channels, ADSR.ADF=1.
  If the ADCR.ADIE =1, the ADC interrupt will be asserted.
- 4. if ADST=1, repeat steps 2 and 3.
  when ADST=0, A/DC ends.

Continuous Scan Mode



Continuous scan on channel 0, 2, 3 and 7 (ADCHER[7:0] = 0x10001101b)

Continuous scan mode



Continuous scan on channel 0, 2, 3, and 7 (ADCHER[7:0] = 8'b10001101)

External trigger Input Sampling and A/D Conversion Time

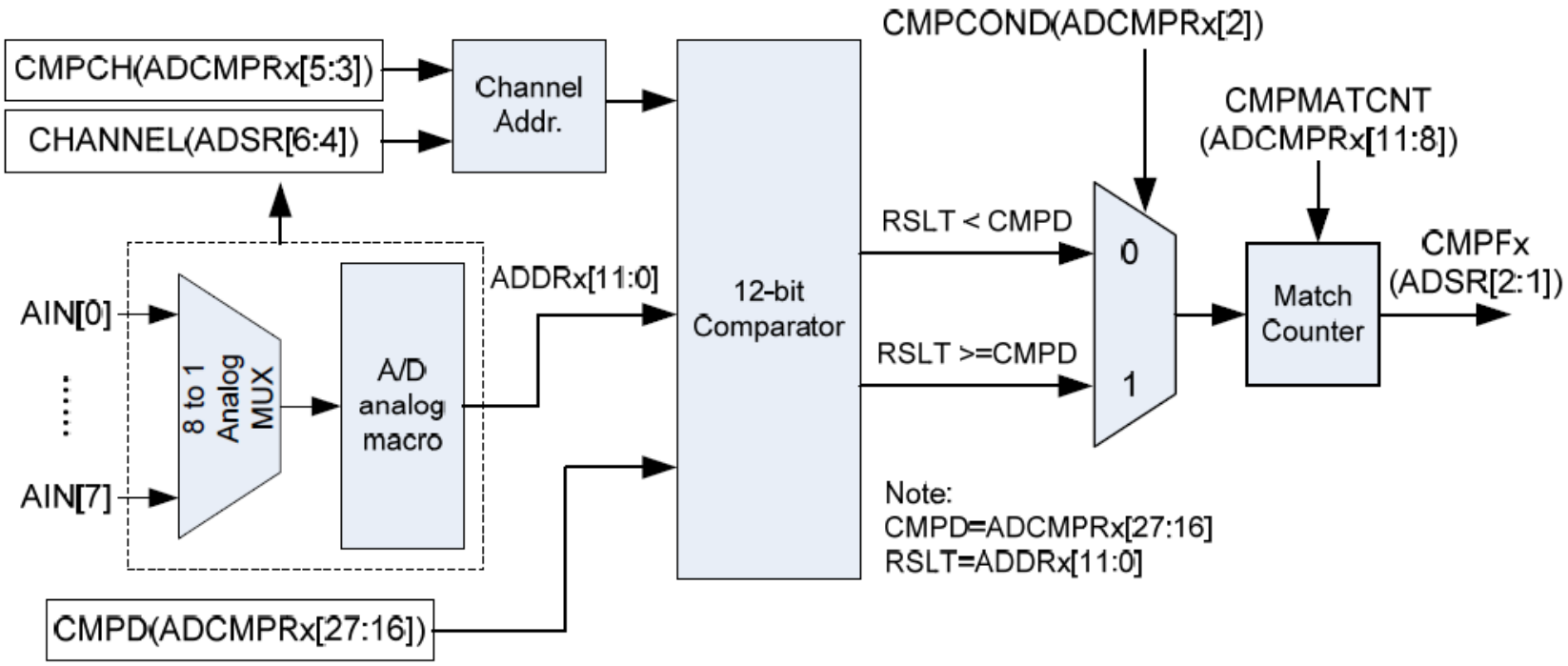
- When the ADCR.TRGEN=1 to enable ADC external trigger function, setting the TRGS[1:0] = 00b is to select external trigger input from the STADC pin.
- TRGCOND[1:0] to select trigger condition is falling/rising edge or low/high level.
- level trigger condition：
  - STADC pin must be kept at defined state at least 8 PCLKs
  - Set ADST=1 at the 9th PCLK and start to conversion
  - ADC continue if external trigger input is kept at active state.
  - ADC stopped only when external trigger condition disappeared
- edge trigger condition: the high and low state must be kept at least 4 PCLKS

Conversion Result Monitor by Compare Function

- ADCMPR0 and ADCMPR1: to monitor two specified channels conversion result from A/D conversion controller
- CMPCH(ADCMPRx[5:0]): channel to be monitored
- CMPCOND(ADCMPRx[2]): check conversion result is less than, greater than (equal to) specified value (CMPD[11:0])
- CMPD(ADCMPRx[27:16]: specified value
- When the compare result meets the setting, compare match counter will increase 1, otherwise, the compare match counter will be clear to 0.
- CMPFx(ADSR[2:1]): When counter value reach the setting of (CMPMATCNT+1), set CMPF=1. if CMPIE=1, then an ADC_INT interrupt request is generated.

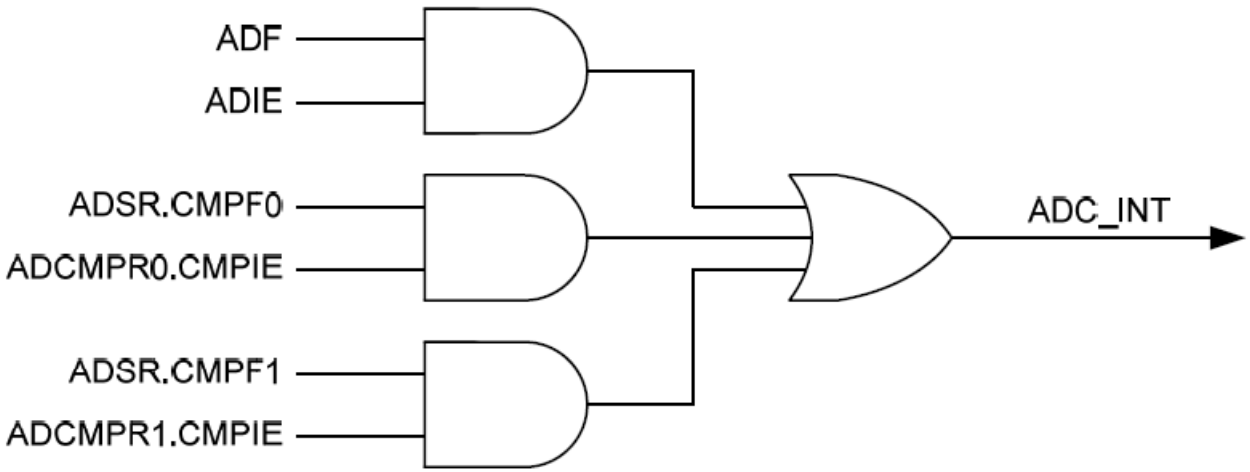
Conversion Result Monitor by Compare Function

- monitor the external analog input pin voltage transition in scan mode without imposing a load on software.



Interrupt Sources

- 1. ADF=1 and ADIE=1.
- 2. ADSR.CMPF0 and ADCMPR0.CMPIE=1.
- 3. ADSR.CMPF1 and ADCMPR1=1
- the ADC interrupt will be asserted.
- Software can clear the flag to revoke the interrupt request.



Control Flow of ADC interrupt

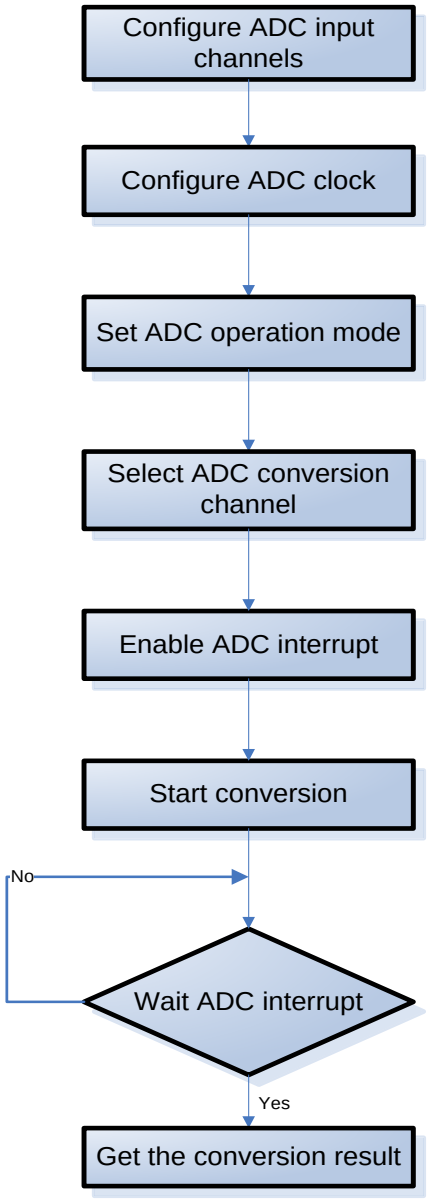
Configure ADC input channels

// Disable IO digital input path (digital input tied to low)

```
GPIOA->OFFD|=0x00800000;
```

// GPA[7] Pin Function Selection

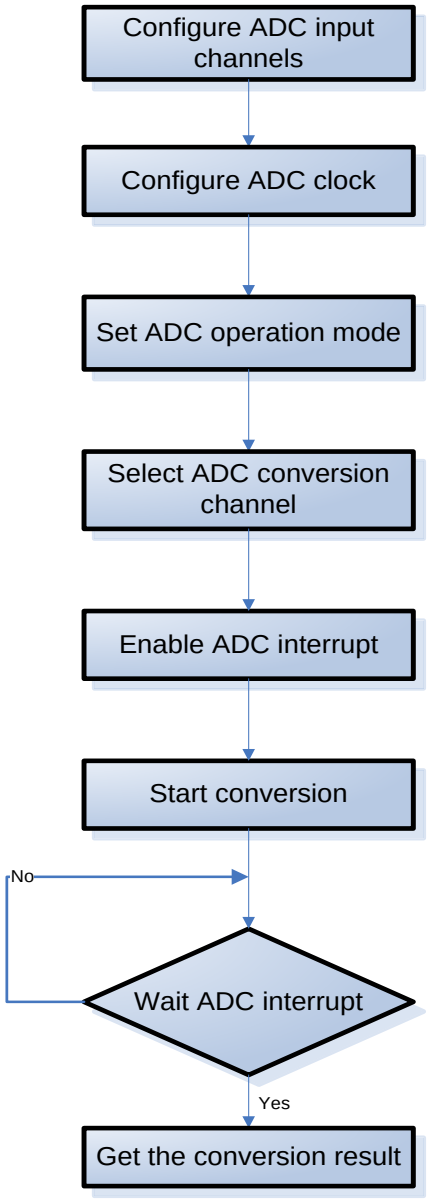
```
SYS->GPAMFP.ADC7_SS21_AD6=1;
```



Control Flow of ADC interrupt

Configure ADC clock

```
// ADC clock source select,00=12M,
01=PLL, 10=HCLK, 11=22M
SYSCLK->CLKSEL1.ADC_S = 0;
// ADC clock divide number
SYSCLK->CLKDIV.ADC_N = 0;
//ADC clock = 12Mhz/(0+1)=11Mhz;
// (ADC) Clock Enable
SYSCLK->APBCLK.ADC_EN = 1;
// A/D Converter Enable
ADC->ADCR.ADEN = 1;
```



Control Flow of ADC interrupt

Set ADC operation mode

// Differential Input Mode Enable

// 0=single end input

ADC->ADCR.DIFFEN = 0;

// A/D Converter Operation Mode,
00=single, 10=single-cycle scan,
11=continuous scan

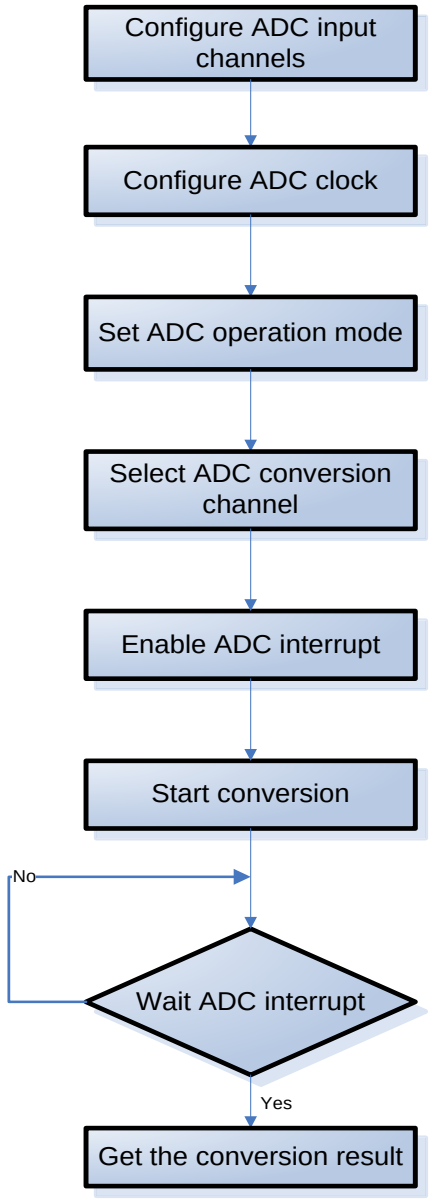
ADC->ADCR.ADMD = 0;

Select ADC channel

// Analog Input Channel Enable

//Channel 7 Enable

ADC->ADCHER.CHEN = 0x80;



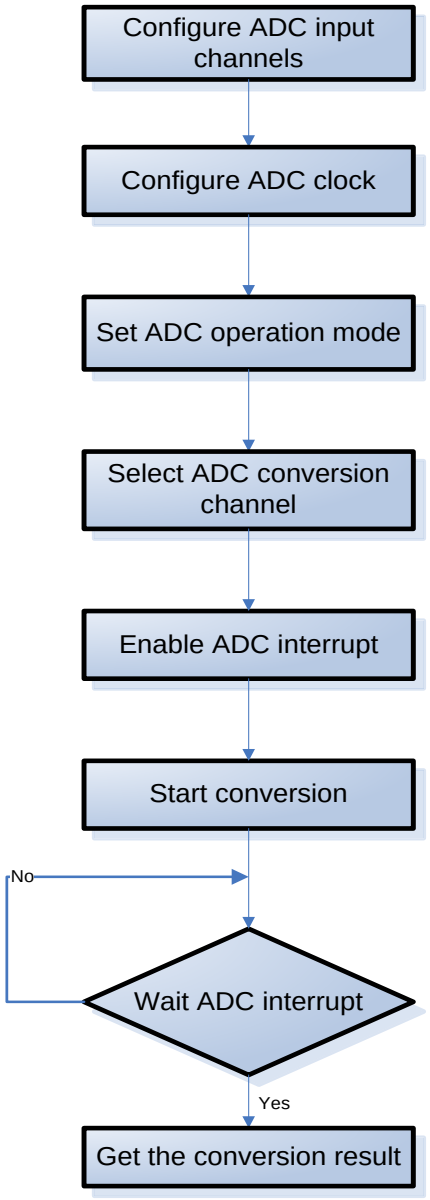
Control Flow of ADC interrupt

Enable ADC interrupt

```
//clear the A/D interrupt flags for safe
ADC->ADSR.ADF =1;
//Enable A/D interrupt function
ADC->ADCR.ADIE = 1;
//
NVIC_SetPriority(ADC_IRQn,
(1<<__NVIC_PRIO_BITS) - 2);
//
NVIC_EnableIRQ(ADC_IRQn);
```

Start Conversion

```
ADC->ADCR.ADST=1;
```



Peripheral DMA Request

- PDMA transfer the latest data of selected channels to the user-specified destination address.
- When A/D conversion is finished, the conversion result will be loaded into ADDR register and set VALID=1.
- If the ADCR.PTEN=1, ADC controller will generate a request to PDMA.
- User can use PDMA to transfer the conversion results to a user-specified memory space without CPU's intervention.
- The source address of PDMA operation is fixed at ADPDMA
- When PDMA transfer ADC result, ADC will continue converting the next selected channel with scan mode.
- ADPDMA register: monitor current PDMA transfer data.

Peripheral DMA Request

Register	Description
ADDR0-7	A/D Data Register 0-7
ADCR	A/D Control Register
ADCHER	A/D Channel Enable Register
ADCMPR0	A/D Compare Register 0
ADCMPR1	A/D Compare Register 1
ADSR	A/D Status Register
ADCALR	A/D Calibration Register
ADPDMA	ADC PDMA current transfer data

ADC 函數

Function	Description
DrvADC_Open	Enable the ADC function and configure the related settings.
DrvADC_Close	Close ADC functions. Disable ADC, ADC engine clock and ADC interrupt.
DrvADC_SetADCChannel	Select ADC input channels.
DrvADC_ConfigADCChannel7	Select the input signal source of ADC channel 7.
DrvADC_SetADCInputMode	Set the ADC input mode.
DrvADC_SetADCOperationMode	Set the ADC operation mode.
DrvADC_SetADCClkSrc	Select the ADC clock source.

ADC 函數

Function	Description
DrvADC_SetADCDivisor	Set the divisor value of ADC clock to determine the ADC clock frequency.
DrvADC_EnableADCInt	Enable ADC interrupt and setup the callback function.
DrvADC_DisableADCInt	Disable the ADC interrupt.
DrvADC_EnableADCCmp0Int	Enable the ADC comparator 0 interrupt and setup callback function.
DrvADC_DisableADCCmp0Int	Disable the ADC comparator 0 interrupt.
DrvADC_EnableADCCmp1Int	Enable the ADC comparator 1 interrupt and setup callback function.
DrvADC_DisableADCCmp1Int	Disable the ADC comparator 1 interrupt.

ADC 函數

Function	Description
DrvADC_GetConversionRate	Get the A/D conversion rate. The ADC takes about 27 ADC clock cycles for converting one sample.
DrvADC_EnableExtTrigger	Allow the external trigger pin (PB8) to be the trigger source of ADC.
DrvADC_DisableExtTrigger	Prohibit the external ADC trigger.
DrvADC_StartConvert	Clear the ADC interrupt flag (ADF) and start A/D converting.
DrvADC_StopConvert	Stop A/D converting.
DrvADC_IsConversionDone	Check whether the conversion action is finished or not.
DrvADC_GetConversionData	Get the conversion result of the specified ADC channel.

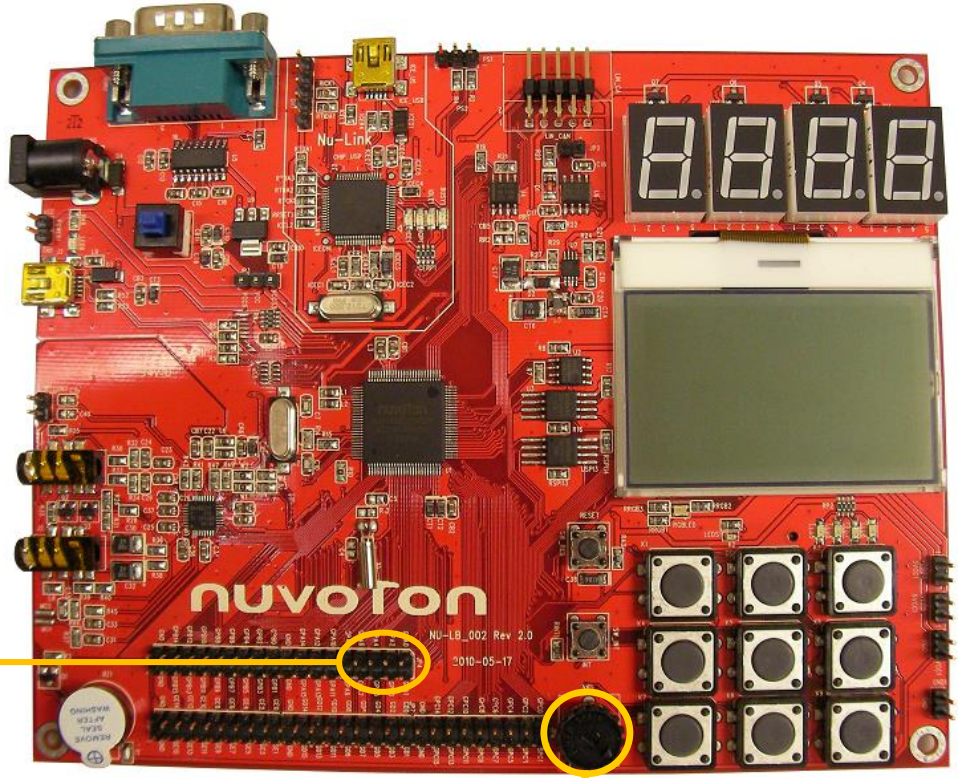
ADC 函數

Function	Description
DrvADC_EnablePDMA	Enable PDMA transfer.
DrvADC_DisablePDMA	Disable PDMA transfer
DrvADC_IsDataValid	Check whether the conversion data is valid or not.
DrvADC_IsDataOverrun	Check whether the conversion data is overrun or not.
DrvADC_EnableADCCmp0	Enable the ADC comparator 0 and configure the necessary settings.
DrvADC_DisableADCCmp0	Disable the ADC comparator 0.
DrvADC_EnableADCCmp1	Enable the ADC comparator 1 and configure the necessary settings.

ADC 函數

Function	Description
DrvADC_DisableADCCmp1	Disable the ADC comparator 1.
DrvADC_EnableSelfCalibration	Enable the self calibration function for minimizing the A/D conversion error.
DrvADC_IsCalibrationDone	Check whether the self calibration action is finished or not.
DrvADC_DisableSelfCalibration	Disable the self calibration function.
DrvADC_DiffModeOutputFormat	Select the output format of differential input mode
DrvADC_GetVersion	Return the current version number of ADC driver.

NuMicro Cortex-M0學習板



ADC0~7(GPA0~7)

VR1可變電阻

範例: Smpl_LCD_VR1 or Smpl_ADC_PWM

RTC01： Test_ADC

從ADC7輸入一個類比信號，轉換成數位信號，將之顯示在LCD。

1. 觸發ADC轉換
2. 檢查ADF旗標，等待轉換完成
3. 將12bits的數位資料，以10進制顯示在LCD

RTC01： Test_ADC 1/x

```
int32_t main (void)
{
        char adc_value[15]="ADC Value:";
        uint16_t ADC_result;

        //Initial 12M and set HCLK=12MHz
        InitHCLK12M();

        // initial LCD pannel function
        Initial_pannel();  //(LCD_Driver.c)
```

RTC01： Test_ADC 2/x

```
// clear LCD pannel
clr_all_pannal(); //(LCD_Driver.c)

//Initial ADC
InitADC();

// show title on row 0
print_lcd(0, adc_value);//(LCD_Driver.c

while(1)
{
```

RTC01： Test_ADC 3/x

```
// A/D Conversion Start
ADC->ADCR.ADST=1;  //1 = Conversion start

// wait for ADC complete
while(ADC->ADSR.ADF==0);  //1=A/D Conversion End Flag

// clear A/D Conversion End Flag
ADC->ADSR.ADF=1;  //0=clear A/D Conversion End Flag

//A/D Conversion Result
ADC_result=ADC->ADDR[7].RSLT;// A/D Conversion Result
```

RTC01： Test_ADC 4/x

```
        // convert 16 bits number to ASCII char
        uint16_ascii(ADC_result,adc_value);

        // display string from 0,10
        show_string(0,10,adc_value);

        //Delay(20000);
        DrvSYS_Delay(500000);  //delay 500,000us=500ms

        }
}
```

RTC01： Test_ADC 5/x

```
void InitADC(void)
{
        // GPIOA Pin[7] Digital Input Path Disable Control
        GPIOA->OFFD|=0x00800000;
        // GPA[7] Pin Function Selection
        SYS->GPAMFP.ADC7_SS21_AD6=1;
        // ADC clock source select,00=12M, 01=PLL, 10=HCLK,
11=22M
        SYSCLK->CLKSEL1.ADC_S = 0;
        // ADC clock divide number from ADC clock source
        SYSCLK->CLKDIV.ADC_N = 0;
```

RTC01： Test_ADC 6/x

```
// Analog-Digital-Converter (ADC) Clock Enable
SYSCLK->APBCLK.ADC_EN = 1;
// A/D Converter Enable
ADC->ADCR.ADEN = 1;
// Differential Input Mode Enable
ADC->ADCR.DIFFEN = 0;
// A/D Converter Operation Mode, 00=single, 10=single-cycle scan, 11=continuous scan
ADC->ADCR.ADMD   = 0;
// Analog Input Channel Enable
ADC->ADCHER.CHEN = 0x80;
```

RTC01： Test_ADC 7/x

```
        // A/D Conversion End Flag
        ADC->ADSR.ADF =1;
        // A/D Interrupt Enable
        ADC->ADCR.ADIE = 1;
}
```

RTC01： Test_ADC 8/x

```
void InitHCLK12M(void)
{
        UNLOCKREG();
        //External 4~24 MHz High Speed Crystal Enable (write-protection bit)
        SYSCLK->PWRCON.XTL12M_EN = 1;
        //HCLK clock source select (write-protection bits)
        //000 = Clock source from external 12 MHz
        SYSCLK->CLKSEL0.HCLK_S = 0;
        LOCKREG();
}
```

RTC01： Test_ADC 9/x

```
// Analog-Digital-Converter (ADC) Clock Enable
SYSCLK->APBCLK.ADC_EN = 1;
// A/D Converter Enable
ADC->ADCR.ADEN = 1;
// Differential Input Mode Enable
ADC->ADCR.DIFFEN = 0;
// A/D Converter Operation Mode, 00=single, 10=single-cycle scan, 11=continuous scan
ADC->ADCR.ADMD   = 0;
// Analog Input Channel Enable
ADC->ADCHER.CHEN = 0x80;
```

RTC01： Test_ADC 10/x

```
void uint16_ascii(uint16_t value,char text[])
{       uint8_t d1,d0;
        int8_t ia=5;
        for (ia=4; ia>0; ia--)
        {
                d1=value/10;     //value=0-65535
                d0=value-d1*10;
                text[ia]=d0+'0';  //get [4],[3],[2],[1]
                value=d1;         //next value
        }
                text[0]=value+'0'; //get [0]
                text[5]=0;          //string delimiter, /0
}
```

RTC01： Test_ADC 11/x

```
void show_string(unsigned char x, unsigned char y, char *str)
{
        int i=y;
        do{
                Show_Word(x,i,*str++);  //display a character at (x,i)
                i++;                    // next character
                if(i>15)  break;        // max 16 character
        }  while(*str!='\0');           //
}
```

RTC02： Test_ADCint_LCD

從ADC7輸入一個類比信號，轉換成數位信號，將之顯示在LCD。

1. 觸發ADC轉換
2. 使用 中斷處理ADC轉換完成
3. 將12bits的數位資料，以10進制顯示在LCD

RTC02 : Test_ADCint_LCD 1/x

```
int32_t main (void)
{
        char adc_value[15]="ADC Value:";
        //Initial 12M and set HCLK=12MHz
        InitHCLK12M();
        // initial LCD pannel function
        Initial_pannel();  //(LCD_Driver.c)
        // clear LCD pannel
        clr_all_pannal(); //(LCD_Driver.c)
        //Initial ADC
        InitADC();
```

RTC02 : Test_ADCint_LCD 2/x

```
        // A/D Conversion Start
        ADC->ADCR.ADST=1;  //1 = Conversion start
        // show title on row 0
        print_lcd(0, adc_value);//(LCD_Driver.c
        while(1)
        {
                // convert 16 bits number to ASCII char
                uint16_ascii(ADC_result,adc_value);
                // display string from 0,10
                show_string(0,10,adc_value);
                //DrvSYS_Delay(500000);
        }
}
```

RTC02 : Test_ADCint_LCD 3/x

```
// Initial ADC
void InitADC(void)
{
    . . .
    //clear the A/D interrupt flags for safe
    ADC->ADSR.ADF =1;
    // Enable A/D interrupt function
    ADC->ADCR.ADIE = 1;
    NVIC_SetPriority(ADC_IRQn, (1<<__NVIC_PRIO_BITS) - 2);
    NVIC_EnableIRQ(ADC_IRQn); //(core_cm0.h)
    . . .
}
```

RTC02 : Test_ADCint_LCD 4/x

```
// ADC interrupt callback function
void ADC_IRQHandler(void)
{
        // clear A/D Conversion End Flag
        ADC->ADSR.ADF=1;
        //0=clear A/D Conversion End Flag
        //A/D Conversion Result
        ADC_result=ADC->ADDR[7].RSLT;
        // A/D Conversion Start
        ADC->ADCR.ADST=1;  //1 = Conversion start
}
```

RTC03： Test_ADC_7SEG

從ADC7輸入一個類比信號，轉換成數位信號，將之顯示在7 段顯示器。

1. 觸發ADC轉換
2. 檢查ADF旗標，等待轉換完成
3. 將12bits的數位資料，以10進制顯示在7 段顯示器

RTC03 : Test_ADC_7SEG 1/x

```
int32_t main (void)
{
        int32_t ADC_result;
        //Initial 12M and set HCLK=12MHz
        InitHCLK12M();
        // Initial TIMER
        InitTimer0(1000); //T=1/1000=1ms
        //Initial ADC
        InitADC();
        while(1)
        {
```

RTC03 : Test_ADC_7SEG 2/x

```
        // A/D Conversion Start
        ADC->ADCR.ADST=1;
        // wait for ADC complete
        while(ADC->ADSR.ADF==0);
        // clear A/D Conversion End Flag
        ADC->ADSR.ADF=1;
        //A/D Conversion Result
        ADC_result=ADC->ADDR[7].RSLT;
        // display value to 7-segment display
        seg_data(ADC_result);
}
```

RTC03 : Test_ADC_7SEG 3/x

```
void seg_data(int16_t value)
{
        // seperate number to single digit
        digit_7seg[3] = value / 1000;
        value = value - digit_7seg[3] * 1000;
        digit_7seg[2] = value / 100;
        value = value - digit_7seg[2] * 100;
        digit_7seg[1] = value / 10;
        value = value - digit_7seg[1] * 10;
        digit_7seg[0] = value; //4 states
}
```

RTC03 : Test_ADC_7SEG 4/x

```
// Timer 0 interrupt routine
void TMR_Callback(void) // Timer0 interrupt subroutine
{

        // change lighted LED
        seg_display();
}
```

RTC03 : Test_ADC_7SEG 5/x

```
void seg_display(void)
{
        //clear GPIOC bit4-7
        close_seven_segment();
        //show a digit on 7 segment display
        show_seven_segment(light_7seg,digit_7seg[light_7seg]);
        // point to next digit
        if(!light_7seg--)light_7seg=3;
}
```

RTC04： Test_ADCint_7SEG

從ADC7輸入一個類比信號，轉換成數位信號，將之顯示在7 段顯示器。

1. 觸發ADC轉換
2. 使用 中斷處理ADC轉換完成
3. 將12bits的數位資料，以10進制顯示在7 段顯示器

RTC04 : Test_ADCint_7SEG 1/x

```
int32_t main (void)
{
        //Initial 12M and set HCLK=12MHz
        InitHCLK12M();
        // Initial TIMER
        InitTimer0(1000); //T=1/1000=1ms
        //Initial ADC
        InitADC();
        // A/D Conversion Start
        ADC->ADCR.ADST=1;  //1 = Conversion start
        while(1);
}
```

RTC04 : Test_ADCint_7SEG 2/x

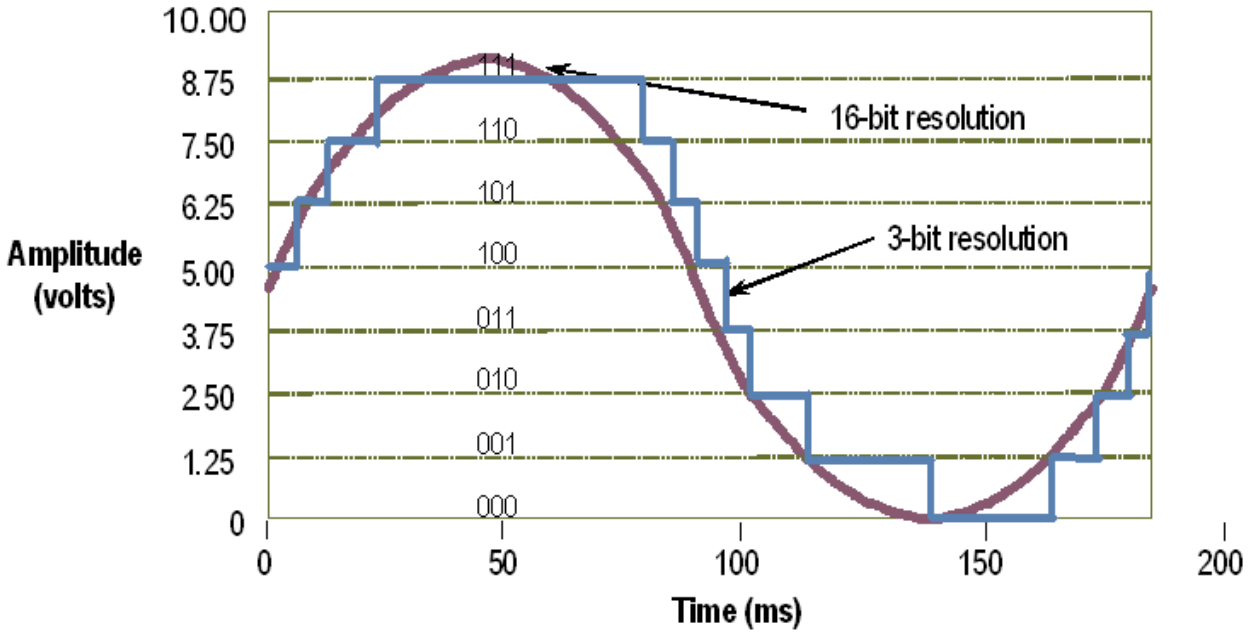
```
// Initial ADC
void InitADC(void)
{
        . . .
        //clear the A/D interrupt flags for safe
        ADC->ADSR.ADF =1;
        //1 = Enable A/D interrupt function
        ADC->ADCR.ADIE = 1;
        NVIC_SetPriority(ADC_IRQn, (1<<__NVIC_PRIO_BITS) - 2);
        NVIC_EnableIRQ(ADC_IRQn); //(core_cm0.h)
        . . .
}
```

RTC04 : Test_ADCint_7SEG 3/x

```
// ADC interrupt callback function
void ADC_IRQHandler(void)
{
        // clear A/D Conversion End Flag
        ADC->ADSR.ADF=1;
        //A/D Conversion Result
        ADC_result=ADC->ADDR[7].RSLT;
        // A/D Conversion Start
        ADC->ADCR.ADST=1;
         // display value to 7-segment display
        seg_data(ADC_result);
}
```

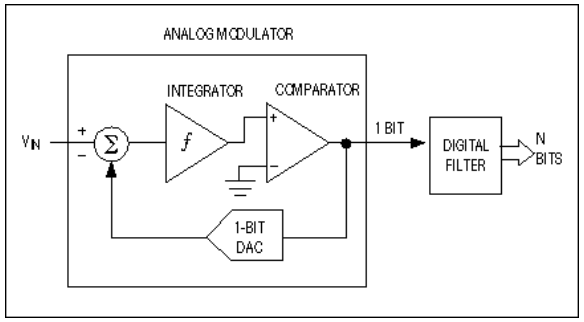
Analog-to-Digital 類比轉數位之應用

- NUC140 : 8-channel 12-bit 800KSPS ADC
- 智能電表(smart meter) 採用16/24-bit ADC
- 電子秤採用16/24-bit ADC
- Audio : 96KHz, 24-bit Stereo ADC

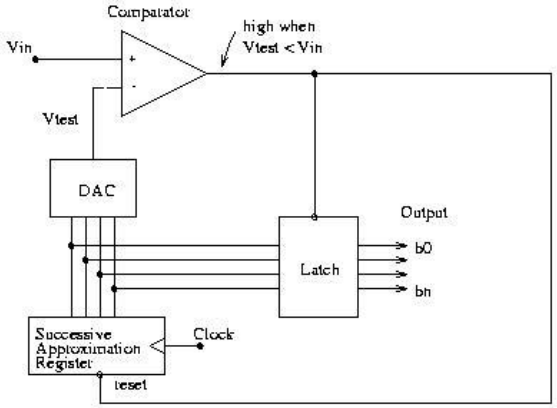


ADC Design Architectures
類比數位轉換器之架構類型

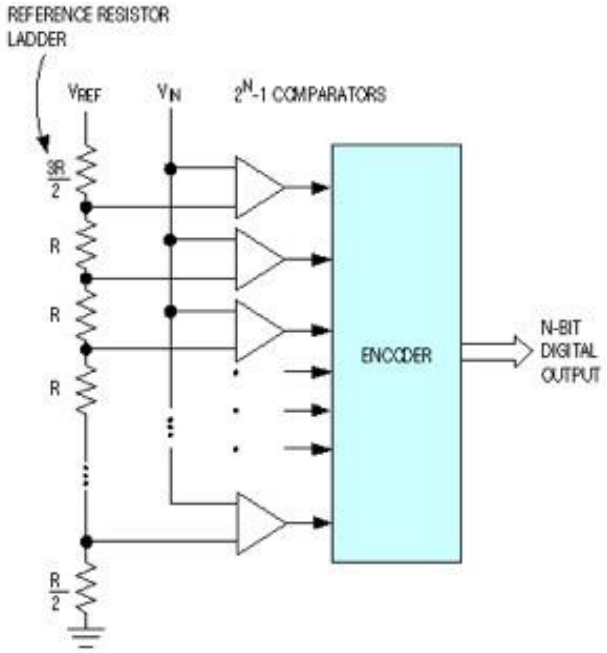
- 直接轉換式：Direct-Conversion ADC
- 逐次逼近式：SAR (Successive Approximation Register) ADC
- 三角積分式：ΔΣ ADC (Sigma-Delta ADC)



Sigma-Delta ADC

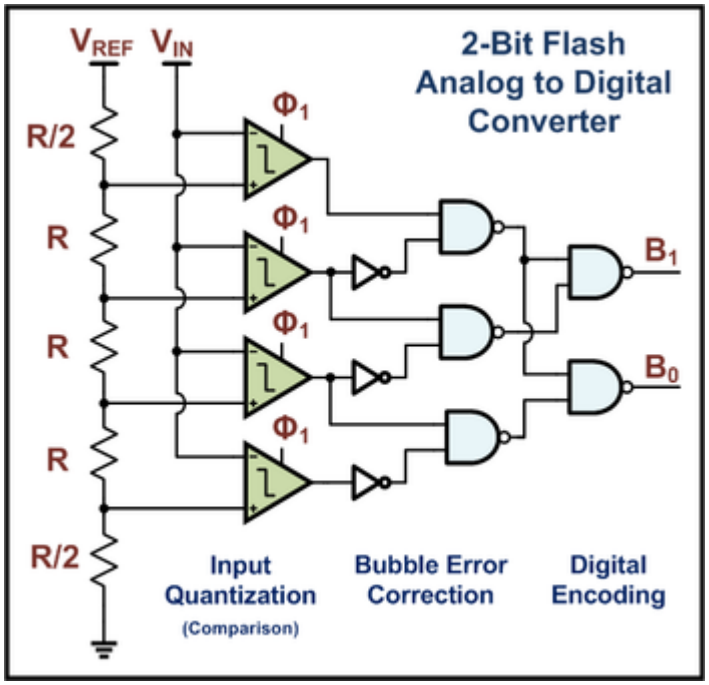


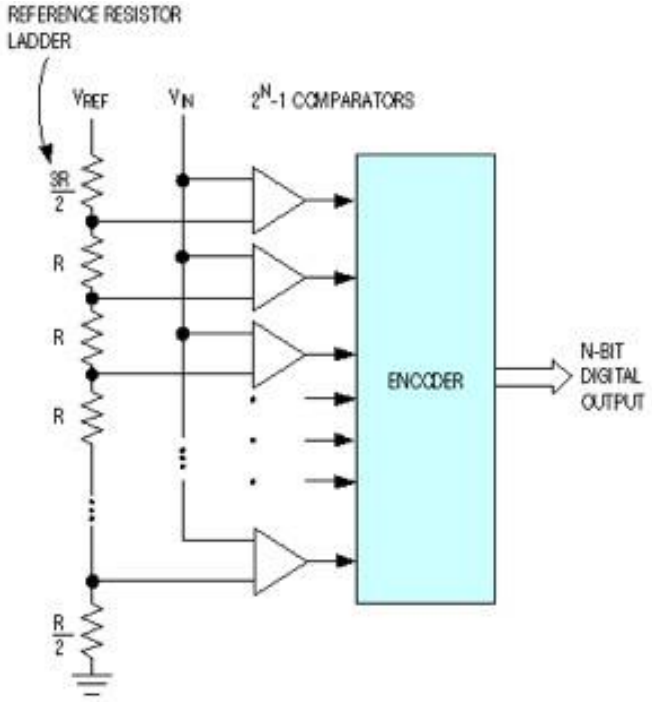
SAR ADC
Successive Approximation Register



Direct-Conversion ADC

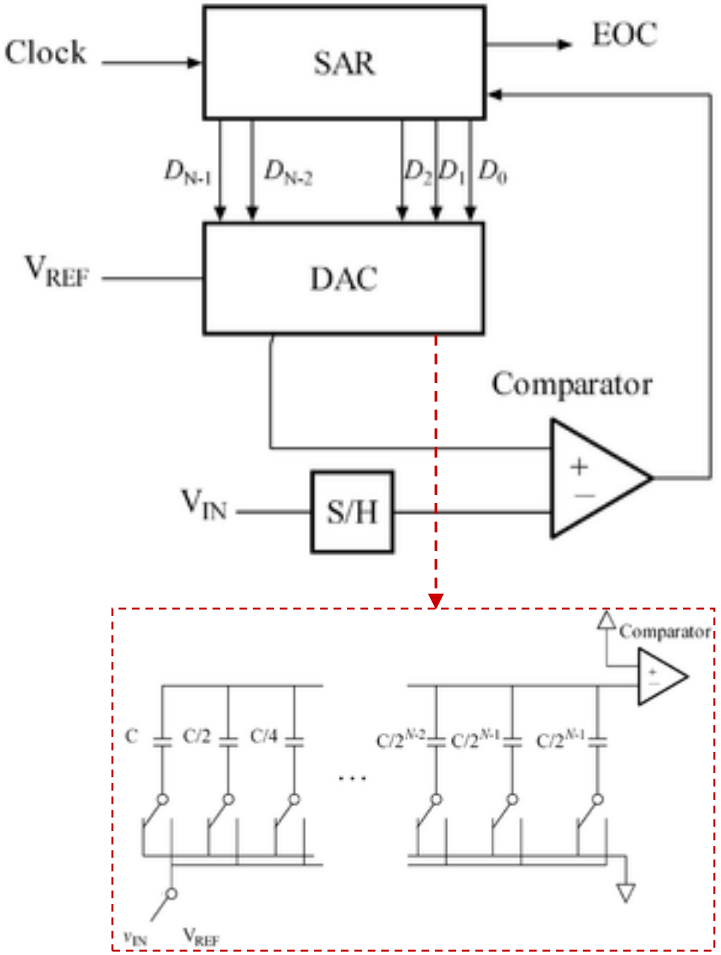
Direct-Conversion ADC
直接轉換式



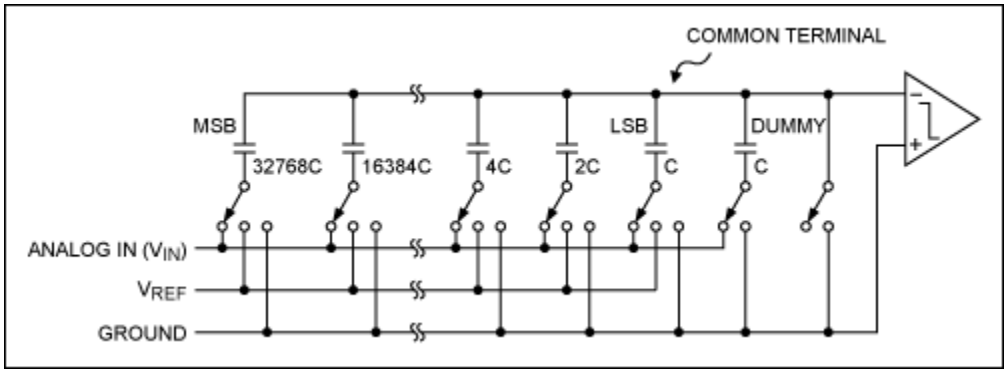


SAR (Successive Approximation Register) ADC
逐次逼近式

- DAC = Digital-to-Analog converter
- EOC = end of conversion
- SAR = successive approximation register
- S/H = sample and hold circuit
- V_{in} = input voltage
- V_{ref} = reference voltage

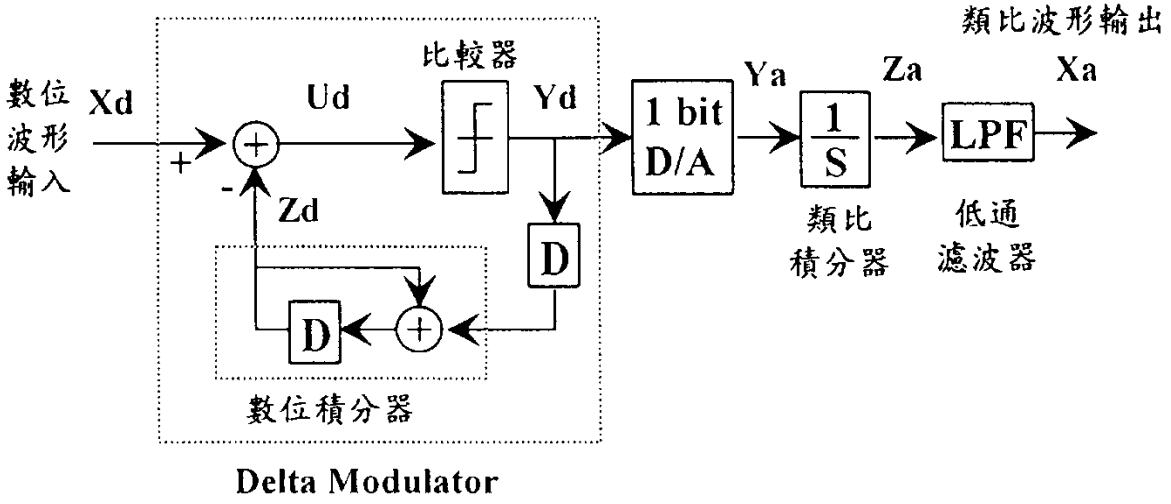


Charge-redistribution successive approximation ADC
分散電量逐次逼近式

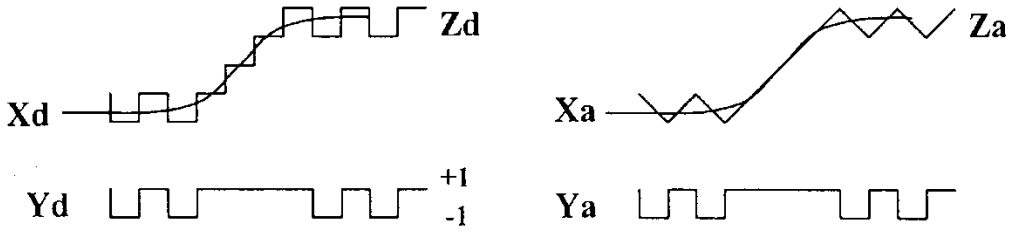


- As the first step in the binary search algorithm, the bottom plate of the MSB capacitor is disconnected from ground and connected to V_{REF}. This drives the common terminal in the positive direction by an amount equal to ½ V_{REF}. Therefore, $V_{COMMON} = -V_{IN} + ½ \times V_{REF}$
- The comparator output yields a logic 1 if $V_{COMMON} < 0$ (i.e., $V_{IN} > ½ \times V_{REF}$). The comparator output yields logic 0 if $V_{IN} < ½ \times V_{REF}$.
- If the comparator output is logic 1, then the bottom plate of the MSB capacitor stays connected to V_{REF}. Otherwise, the bottom plate of the MSB capacitor is connected back to ground.
- This continues until all the bits have been determined.
- $V_{COMMON} = -V_{IN} + B_{N-1} \times V_{REF}/2 + B_{N-2} \times V_{REF}/4 + B_{N-1} \times V_{REF}/8 + ... + B0 \times V_{REF}/2^{N-1}$

ΔΣ modulation (三角積分調變)



▲圖二：Δ調變之1-bit DAC



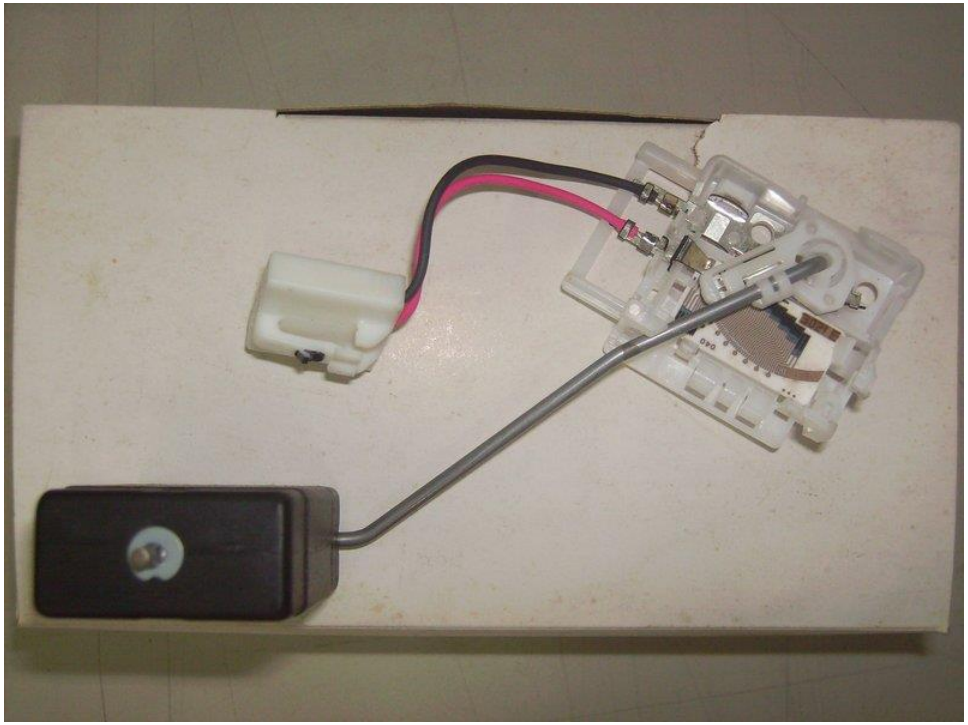
▲圖三：Δ調變1-bit DAC之工作原理

三角積分調變原理

- 圖二是一個八調變之1Bit DAC。Xd代表數位波形輸入，就數位音響而言，Xd可能是18bit，至於尾巴的d代表digital之意。
- Yd為△調變之1Bit輸出，值為正1或負1。△調變之觀念很簡單，就是要使Yd之積分波形愈接近Xd愈好，如圖三所示。每當Yd之積分值(即Zd)超過Xd，下一個Yd值就設為負1。如果Yd之積分值Zd低於Xd，下一個Yd值就設為正1。
- 圖二的減法器就是要看看Xd和Zd誰大誰小，Ud=Xd-Zd，若Ud大於零，比較器輸出(即Yd)就為正1，若Ud小於零，比較器輸出為負1。如此一來Yd不斷的修正使得Yd之積分後波形Zd如影隨形般的和Xd同上同下。現在要做的就是把Zd以類比的方式重現出來。
- 很容易的，首先利用1Bit的DAC將數位的Yd轉成類比的對等信號Ya，(其中a代表analog之意)，然後再用類比積分器將Ya作積分而產生Za。於是Za和Zd兩者之波形是一樣的，只不過Zd是數位而Za是類比。
- 但是由於1Bit DAC，Za會有些不平滑的轉折點，所以最後還需要一個類比低通濾波器以產生平滑的Xa，Xa就是Xd的類比重現

Fuel Level Sensor（油箱浮筒）

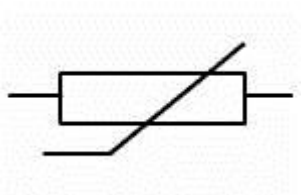
機車油箱浮筒



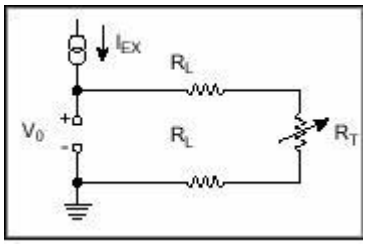
汽車油箱浮筒

可變電阻值: ~100 ohm

Thermistor（熱敏電阻）



敏熱電阻的通用符號



2線式連結

敏熱電阻

車用溫度傳感器

可變電阻值：~500,1K,5K,10K ohm

範例: 熱敏電阻偵測溫度 Smpl_ADC_Thermister

- 熱敏電阻(~10Kohm, 也有1K or 500 ohm的)
- 熱敏電阻電路接法
  - 一端接Vcc (3.3V)
  - 另一端接ADC6(GPA6) 及與熱敏電阻阻值接進之 電阻(如10Kohm)
  - 固定電阻之另一端則接到GND
- 熱敏電阻量測原理
  - 一般室溫25˚C下，熱敏電阻阻抗約與所加分流電阻相同，故ADC讀取值約為4096 (12-bit ADC)之半= 2048
  - 元件之規格書有提供溫度vs阻抗之對照表
  - 0~50˚C約為線性分佈 (溫度軸取log)

範例: Smpl_ADC_Thermister

- 熱敏電阻(~10Kohm)
  - 一端接Vcc (3.3V)
  - 另一端接ADC6(GPA6) 及10Kohm 電阻
  - 10Kohm電阻之另一端則接到GND

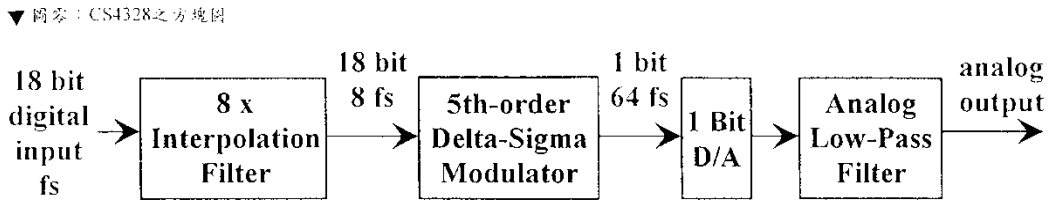
- 以ADC6 (GPA6) 讀取熱敏電阻

References 參考資料

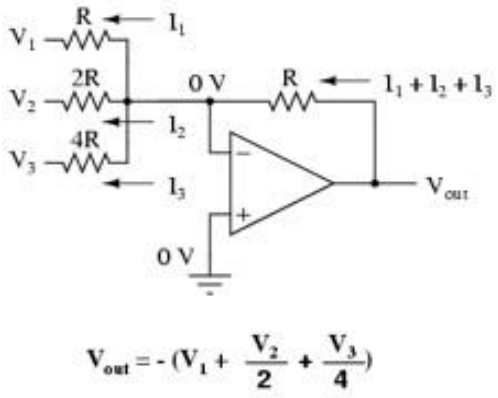
- *Analog*-to-*digital converter* - Wikipedia, the free encyclopedia
- Flash *ADC* - Wikipedia, the free encyclopedia
- Understanding *SAR ADCs*: Their Architecture and Comparison with **...**
- *淺談Delta-Sigma之工作原理*
- TI High Speed *Analog* to *Digital Converter* Basics

DAC (Digital-to-Analog Converter)
數位類比轉換器

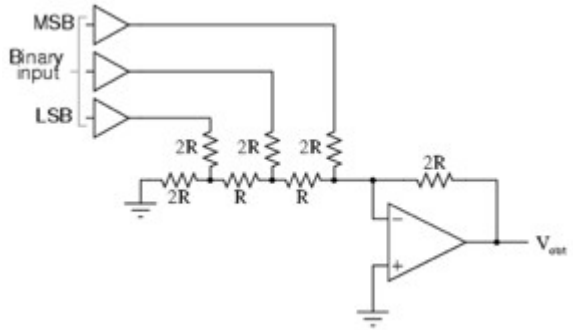
- Binary-Weighted Resistor DAC
- R-2R Ladder DAC
- ΔΣ DAC (Sigma-Delta DAC)



Sigma-Delta DAC



Binary-Weighted Resistor DAC



R-2R DAC

Vout = - (Vmsb + Vn + Vlsb) = - (Vref + Vref/2 + Vref/ 4)

General Disclaimer

The Lecture is strictly used for educational purpose.

MAKES NO GUARANTEE OF VALIDITY

- **The lecture cannot guarantee the validity of the information found here.** The lecture may recently have been changed, vandalized or altered by someone whose opinion does not correspond with the state of knowledge in the relevant fields. Note that most other encyclopedias and reference works also have similar disclaimers.

No formal peer review

- The lecture is not uniformly peer reviewed; while readers may correct errors or engage in casual peer review, they have no legal duty to do so and thus all information read here is without any implied warranty of fitness for any purpose or use whatsoever. Even articles that have been vetted by informal peer review or *featured article* processes may later have been edited inappropriately, just before you view them.

No contract; limited license

- Please make sure that you understand that the information provided here is being provided freely, and that no kind of agreement or contract is created between you and the owners or users of this site, the owners of the servers upon which it is housed, the individual Wikipedia contributors, any project administrators, sysops or anyone else who is in *any way connected* with this project or sister projects subject to your claims against them directly. You are being granted a limited license to copy anything from this site; it does not create or imply any contractual or extracontractual liability on the part of Wikipedia or any of its agents, members, organizers or other users.
- There is **no agreement or understanding between you and the content provider** regarding your use or modification of this information beyond the Creative Commons Attribution-Sharealike 3.0 Unported License (CC-BY-SA) and the GNU Free Documentation License (GFDL);

General Disclaimer

Trademarks

- Any of the trademarks, service marks, collective marks, design rights or similar rights that are mentioned, used or cited in the lectures are the property of their respective owners. Their use here does not imply that you may use them for any purpose other than for the same or a similar informational use as contemplated by the original authors under the CC-BY-SA and GFDL licensing schemes. Unless otherwise stated , we are neither endorsed by nor affiliated with any of the holders of any such rights and as such we cannot grant any rights to use any otherwise protected materials. Your use of any such or similar incorporeal property is at your own risk.

Personality rights

- The lecture may portray an identifiable person who is alive or deceased recently. The use of images of living or recently deceased individuals is, in some jurisdictions, restricted by laws pertaining to personality rights, independent from their copyright status. Before using these types of content, please ensure that you have the right to use it under the laws which apply in the circumstances of your intended use. *You are solely responsible for ensuring that you do not infringe someone else's personality rights.*