NUC140 CLOCK

- 2013/4/4

課程大綱 (Lesson Outline)

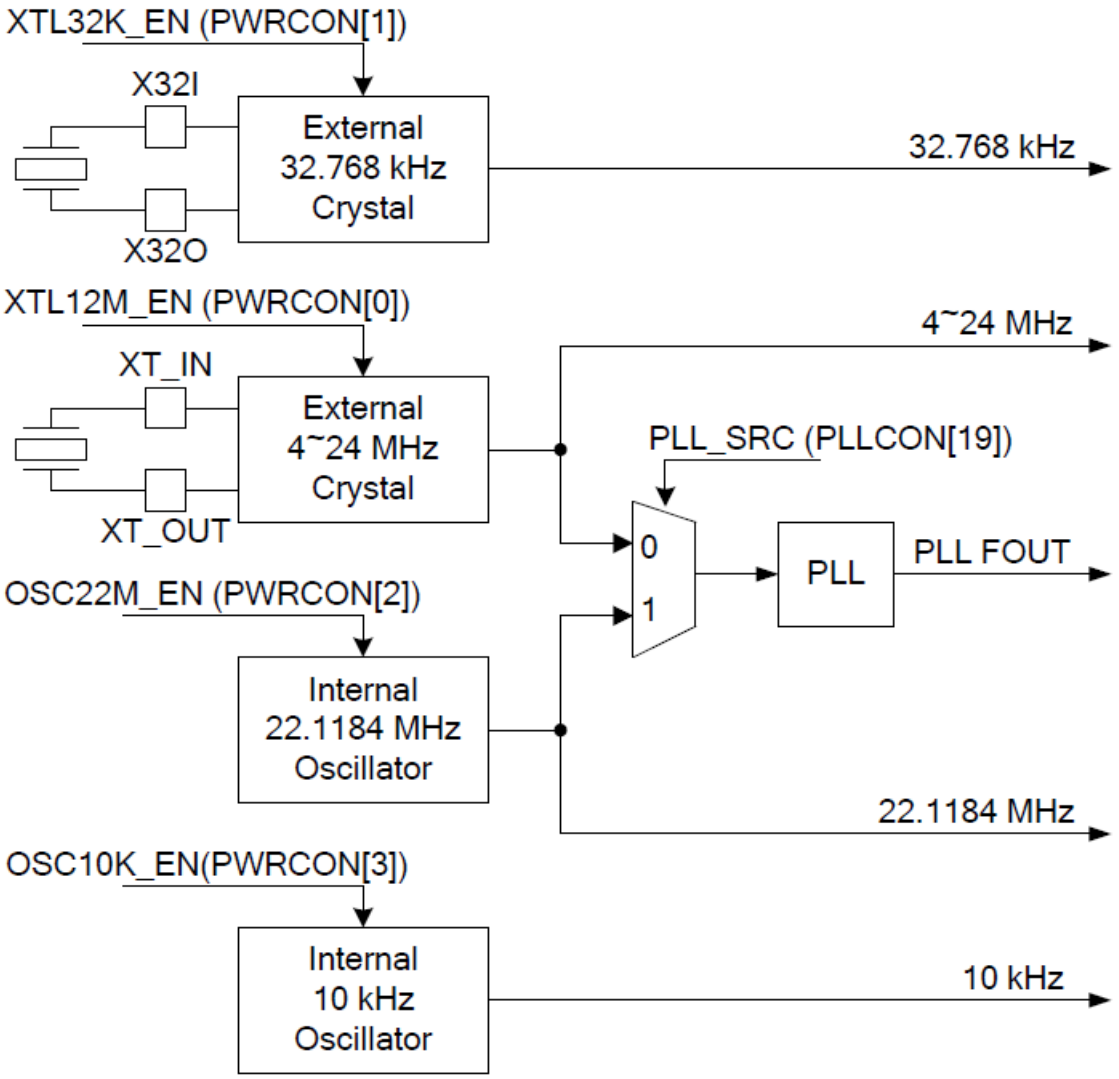
- **Cortex-M0 MCU之時序**

- **實習範例1 :**
- **實習範例 2:**
- **實習範例 3:**
- **實習範例 4:**
- **實習範例 5:**
- **實習範例 6:**

概論

- 時鐘控制器為整個晶片提供時鐘源，包括系統時鐘和所有週邊設備時鐘。
- 在掉電模式下，時鐘控制器關閉外部 4~24 MHz 晶振和內部 22.1184 MHz 振盪器，以降低整個系統的功耗。
- 時鐘發生器由如下 5 個時鐘源組成：
  - 一個外部 32.768 kHz 晶振
  - 一個外部 4~24 MHz 晶振
  - 一個可程式設計的 PLL FOUT（PLL 由外部 4~24 MHz 晶振和內 部 22.1184 MHz 振盪器提供時鐘源）
  - 一個內部 22.1184 MHz 振盪器
  - 一個內部 10 kHz 振盪器

時鐘發生器


XTL32K_EN (PWRCON[1])
X32I
X32O
External
32.768 kHz
Crystal
32.768 kHz
XTL12M_EN (PWRCON[0])
XT_IN
XT_OUT
External
4~24 MHz
Crystal
4~24 MHz
PLL_SRC (PLLCON[19])
0
1
PLL
PLL FOUT
OSC22M_EN (PWRCON[2])
Internal
22.1184 MHz
Oscillator
22.1184 MHz
OSC10K_EN(PWRCON[3])
Internal
10 kHz
Oscillator
10 kHz

啟用時鐘源

```
UNLOCKREG();
//External 4~24 MHz (write-protection bit)
SYSCLK->PWRCON.XTL12M_EN = 1;
while(SYSCLK->CLKSTATUS.XTL12M_STB == 0);
//External 32.768 kHz (write-protection bit)
SYSCLK->PWRCON.XTL32K_EN=1;
while(SYSCLK->CLKSTATUS.XTL32K_STB == 0);
//Internal 10 kHz (write-protection bit)
SYSCLK->PWRCON.OSC10K_EN=1;
while(SYSCLK->CLKSTATUS.OSC10K_STB == 0);
//Internal 22.1184 MHz (write-protection bit)
SYSCLK->PWRCON.OSC22M_EN=1;
while(SYSCLK->CLKSTATUS.OSC22M_STB == 0);
LOCKREG();
```

PLL時鐘

- 輸出時鐘頻率設置

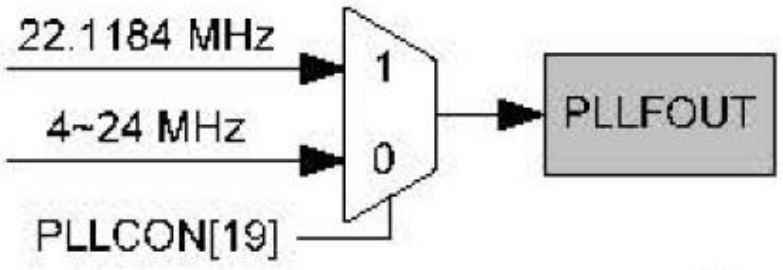
符號	描述
FOUT	輸出時鐘頻率
FIN	輸入（參考）時鐘頻率
NR	輸入分頻 (IN_DV + 2)
NF	回饋分頻 (FB_DV + 2)
NO	OUT_DV = “00” : NO = 1 OUT_DV = “01” : NO = 2 OUT_DV = “10” : NO = 2 OUT_DV = “11” : NO = 4

$$FOUT = FIN \times \frac{NF}{NR} \times \frac{1}{NO}$$

Constraint:

1. $3.2MHz < FIN < 150MHz$
2. $800\,KHz < \frac{FIN}{2 * NR} < 8MHz$

   $100\,MHz < FCO = FIN \times \frac{NF}{NR} < 200\,MHz$
3. $120\,MHz < FCO$ is preferred



PLL時鐘(30M-50M)

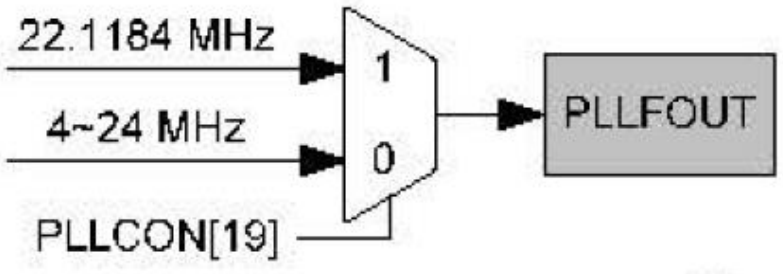
- 輸出時鐘頻率設置
- 取FIN=12M,NR=3,NO=4
- NF決定輸出的頻率
- 滿足條件
- 3.2M<FIN=12M<150M
- 800k<12M/2/3=2M<8M
- 120M<12Mx50/3<200M
- 30<NF<50
- 即FOUT的範圍30M-50M

$$FOUT = FIN \times \frac{NF}{NR} \times \frac{1}{NO}$$

Constraint:

1. $3.2MHz < FIN < 150MHz$
2. $800\,KHz < \frac{FIN}{2 * NR} < 8MHz$

   $100\,MHz < FCO = FIN \times \frac{NF}{NR} < 200\,MHz$
3. $120\,MHz < FCO$ is preferred

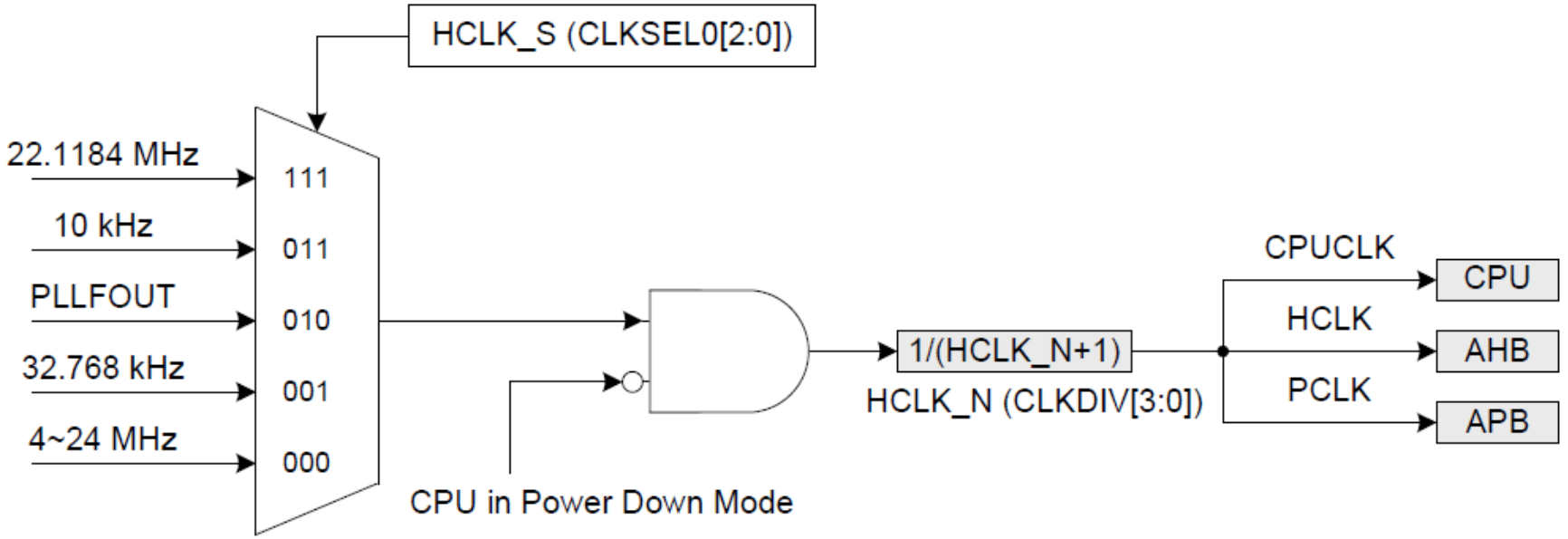


PLL時鐘

```
//DrvSYS_SelectPLLSource, 1=22.1184MHz, 0=12MHz
SYSCLK->PLLCON.PLL_SRC = 0;
// Fout=(Fin)*(FB_DV+2)/NO /(IN_DV+2)
// Fout= (12M)*(FB_DV+2)/3/4
SYSCLK->PLLCON.FB_DV  = 50-2;  //NF
SYSCLK->PLLCON.IN_DV  = 3-2 ;    //NR
SYSCLK->PLLCON.OUT_DV = 3;     //NO:00=1,01=2,10=2,11=4
//: Disable PLL power down mode, PLL operates in normal mode
//PLL OE (FOUT enable) pin Control, 0 = PLL FOUT enable
 SYSCLK->PLLCON.OE = 0;
//Power Down Mode,0 = PLL is in normal mode
SYSCLK->PLLCON.PD = 0;
//Internal PLL Clock Source Stable Flag
while( SYSCLK->CLKSTATUS.PLL_STB == 0);
```

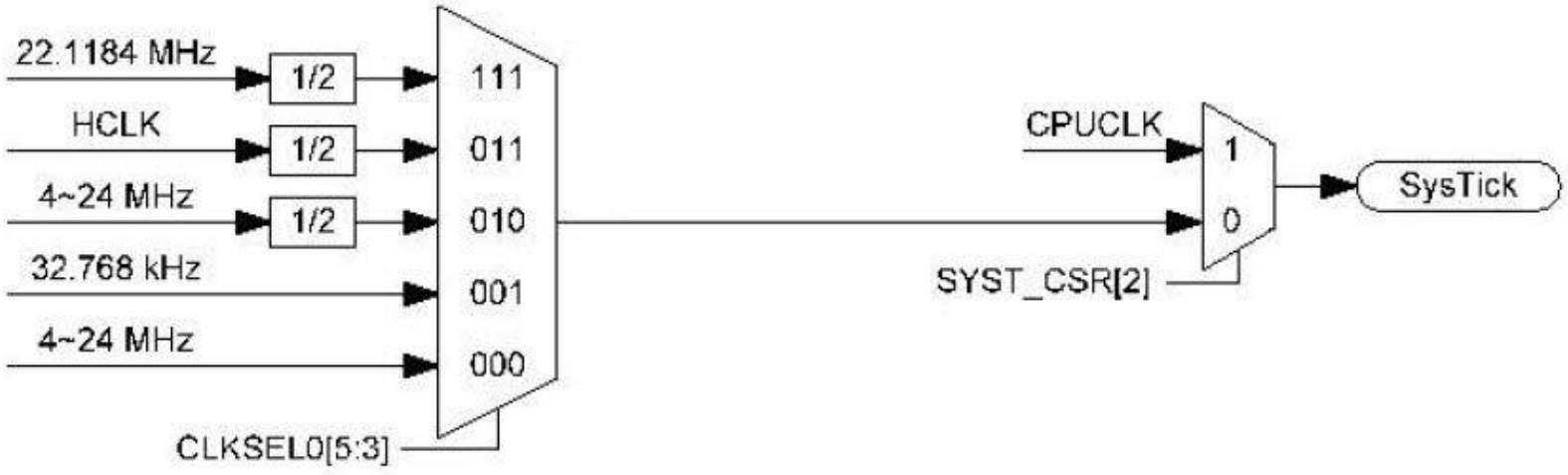
系統時鐘

```
//HCLK clock source select (write-protection bits)
SYSCLK->CLKSEL0.HCLK_S = 0;
//HCLK clock frequency = (HCLK clock source / (HCLK_N + 1)
SYSCLK->CLKDIV.HCLK_N=0;
```

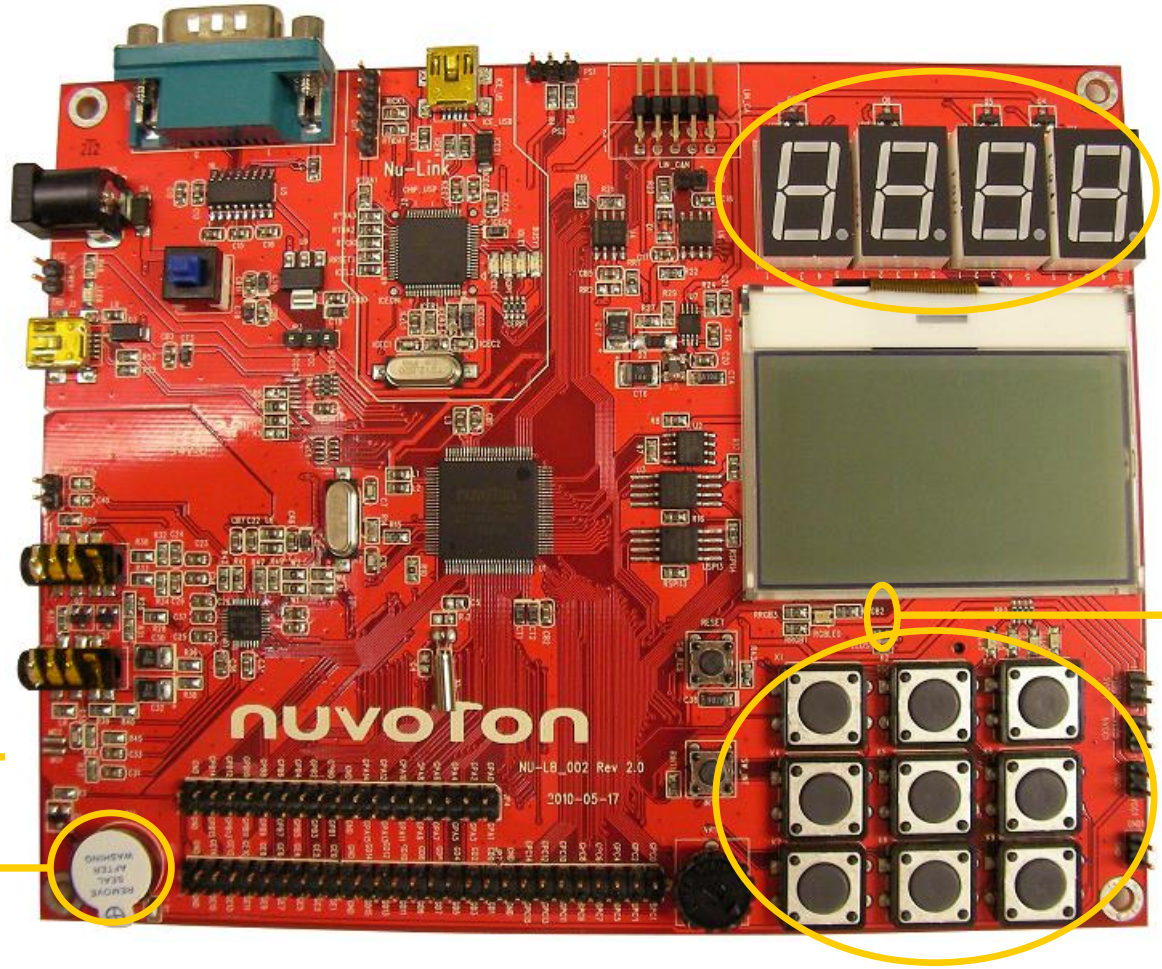


SysTick時鐘

- Cortex-M0 內核的 SysTick 時鐘源可以選擇 CPU 時鐘或外部時鐘 (SYST_CSR[2])。
- SysTick->CTRL &= ~(1<<2) //other clock
- SysTick->CTRL |= (1<<2) //Core clock
- //Cortex_M0 SysTick clock source select (write-protection bits)
- //0=12M,1=32.768k,2=12M/2,3=HCLK/2,7=22.1184M/2
- SYSCLK->CLKSEL0.STCLK_S =0;



NU-LB-NUC140 開發板



7 segment LEDs

RGB LED
GPA12,13,14

Keypad

SW Int
GPB15

Buzzer
GPB11

使用SysTick來計算時間

- 設定SysTick的clock來源為12M
- SysTick的計數器為24位元，從16777215倒數到0，則COUNTFLAG=1
- 在測試的指令之前設定SysTick，指令結束後，檢查SYST_CVR的值，就可計算出花費的時間

設定和啟動SysTick

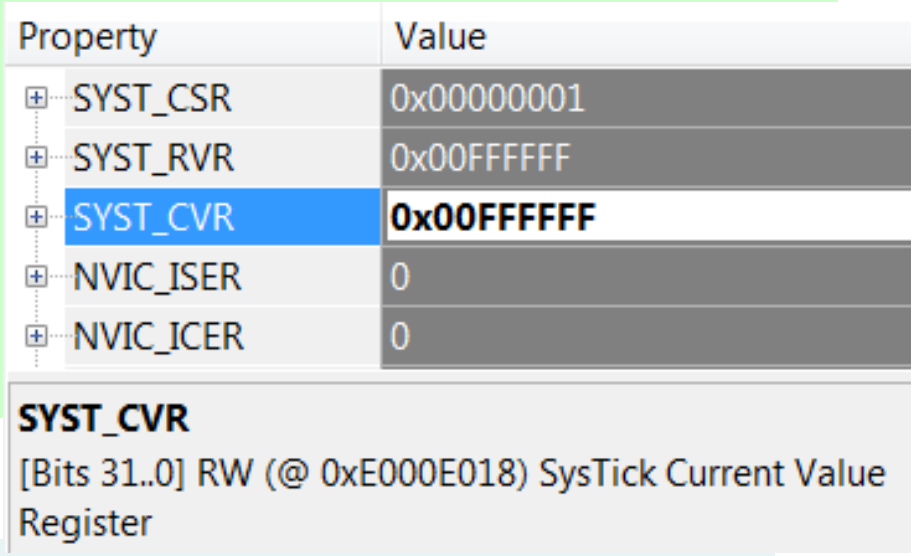
```
//: SYST_RVR=SysTick reload value=2^24-1
SysTick->LOAD = 16777215;
//: SYST_CVR=SysTick current value
SysTick->VAL  = 0;
// SysTick =core clock, and  start
SysTick->CTRL = (0<<2) | (1<<0);
```

使用debug，點選SCS。檢視

SYST_CSR: control

SYST_RVR: reload value

SYST_CVR: current value



計算PWM一個週期的執行時間

設定core clock=12M

SysTick的clock來源使用core clock

設定PWM的clock來源使用12M，輸出為6M。

設定PWM的週期為60000clock，每秒有100個PWM週期。

1.使用SysTick計算一個PWM週期的clock數目

2.使用SysTick計算一秒內PWM週期的clock數目

計算PWM一個週期的執行時間

```
//: SYST_RVR=SysTick reload value=2^24-1
SysTick->LOAD = 16777215;
//: SYST_CVR=SysTick current value
  SysTick->VAL  = (0x00);
//start SysTick
SysTick->CTRL = (0<<2) | (1<<0);
//PWM-Timer 0 Enable (PWM timer 0/A and 4/ B)
PWMA->PCR.CH0EN=1;
//PWM function->0:Disable, 1:Enable
//=1 when PWM0 down counter reaches zero
while(PWMA->PIIR.PWMIF0 == 0); //120032clock
SysTick->CTRL=(0<<0); //stop SysTick
```

計算PWM一秒執行的次數

```
counti=0;
//: SYST_RVR=SysTick reload value=2^24-1
SysTick->LOAD = 12000000;
//: SYST_CVR=SysTick current value
SysTick->VAL  = (0x00);
//start SysTick
SysTick->CTRL = (0<<2) | (1<<0);
//PWM-Timer 0 Enable (PWM timer 0/ A and 4/ B
//PWM function->0:Disable, 1:Enable
PWMA->PCR.CH0EN=1;
```

計算PWM一秒執行的次數

```
//wait for SysTick dount down to 0
while ((SysTick->CTRL & 1<<16) == 0)
{
        //=1 when PWM0 down counter reaches zero
        if(PWMA->PIIR.PWMIF0 == 1)
        {
                PWMA->PIIR.PWMIF0=1;
                counti++;          //=101
        }
}
SysTick->CTRL=(0<<0); //stop SysTick
```

計算ADC一個週期的執行時間

設定core clock=12M

SysTick的clock來源使用core clock

設定ADC的clock來源使用12M，輸出為12M。

1.使用SysTick計算一個ADC週期的clock數目

2.使用SysTick計算一秒內ADC的clock數目

計算ADC一個週期的執行時間

```
//: SYST_RVR=SysTick reload value=2^24-1
SysTick->LOAD = 16777215;
//: SYST_CVR=SysTick current value
  SysTick->VAL  = (0x00);
//start SysTick
SysTick->CTRL = (0<<2) | (1<<0);
// A/D Conversion Start
ADC->ADCR.ADST=1;  //1 = Conversion start
// wait for ADC complete, 1=A/D Conversion End Flag
while(ADC->ADSR.ADF == 0); //60clock
// clear A/D Conversion End Flag
ADC->ADSR.ADF=1;
SysTick->CTRL=(0<<0); //stop SysTick
```

計算ADC一秒執行的次數

```
counti=0;
//: SYST_RVR=SysTick reload value=2^24-1
SysTick->LOAD = 12000000;
//: SYST_CVR=SysTick current value
SysTick->VAL  = (0x00);
//start SysTick
SysTick->CTRL = (0<<2) | (1<<0);
// A/D Conversion Start
ADC->ADCR.ADST=1;  //1 = Conversion start
```

計算ADC一秒執行的次數

```
//wait for SysTick dount down to 0
while ((SysTick->CTRL & 1<<16) == 0)
{       if(ADC->ADSR.ADF == 1) //A/D Conversion End Flag
        {           //A/D Conversion Result
                    ADC_result=ADC->ADDR[7].RSLT;
                    // clear A/D Conversion End Flag
                    ADC->ADSR.ADF=1;  //clear flag
                    // A/D Conversion Start
                    ADC->ADCR.ADST=1;  //1 = Conversion start
                    counti++;
        }
}
SysTick->CTRL=(0<<0); //stop SysTick
```

~ END ~