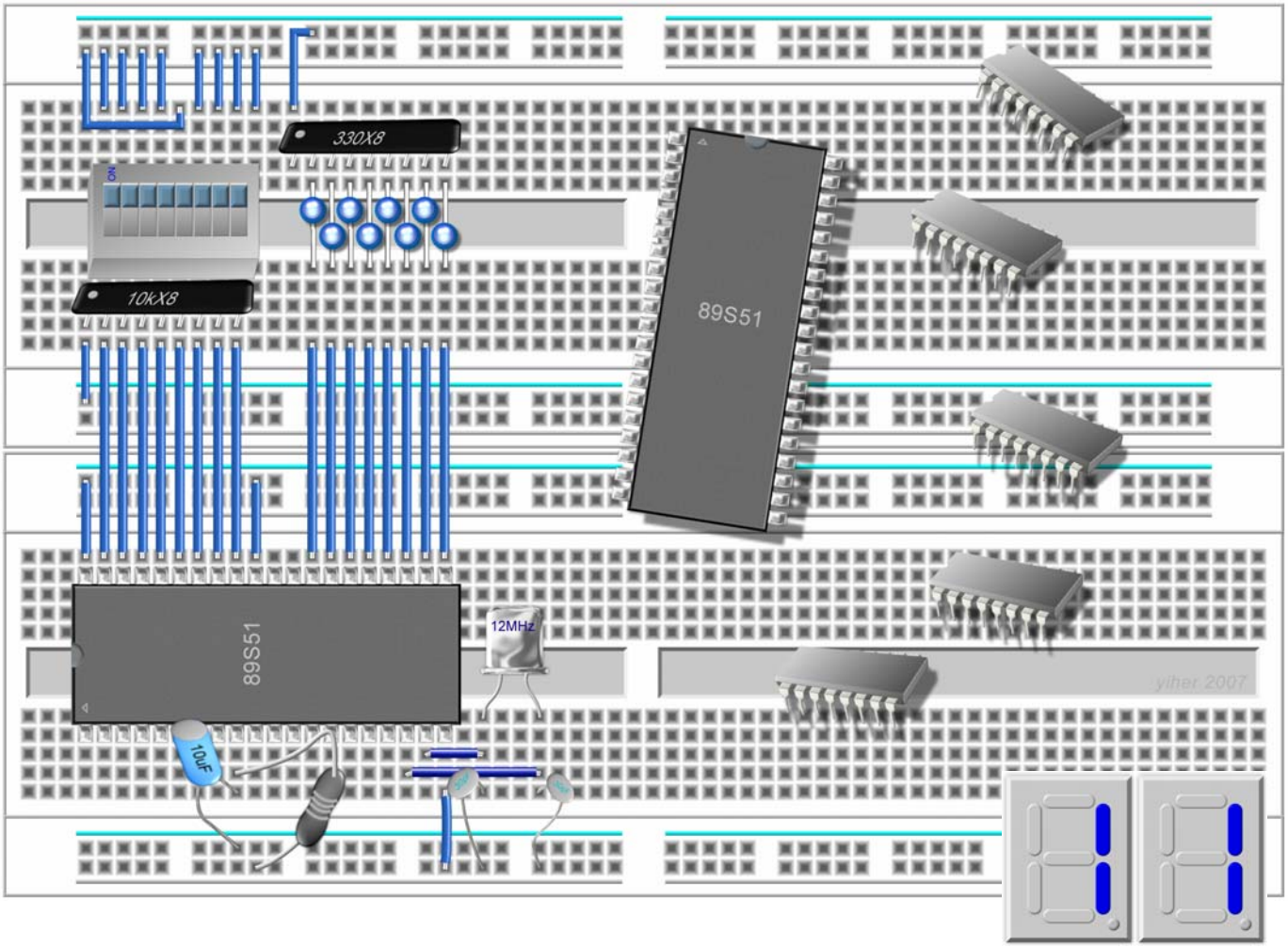


ADC 與 DAC 之應用

本章內容豐富，主要包括兩部分：

硬體部分：

類比-數位轉換原理，數位-類比轉換原理。

類比-數位轉換 IC，及其應用原理。

數位-類比轉換 IC，及其應用原理。

溫度感測 IC-AD590，及其應用原理。

程式與實作部分：

類比-數位轉換應用程式、數位-類比轉換應用程式、配合 AD-590 設計一個數位溫度計及簡易溫控裝置等。

類比(analog)信號是一種連續性的信號，大自然的總總現象(如溫度、溼度、光線等)都屬於這類信號；而數位(digital)信號則是一種非 0 即 1 的非連續性的信號，**通常有 TTL 與 CMOS 兩種準位**。

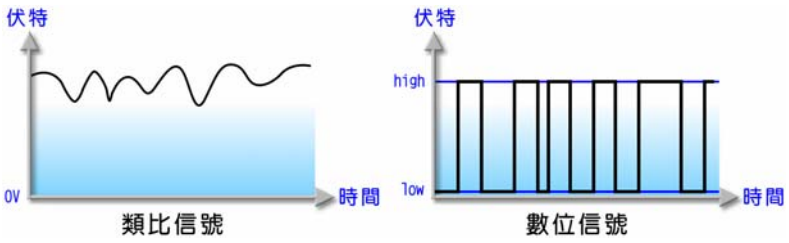


圖1　類比信號與數位信號

人類直接感受的就是類比信號，不過，**類比信號比較不容易儲存、處理與傳輸，且容易失真！**相反的，數位信號就比較容易儲存與處理，且較有效率，在傳輸上，也不易失真，成爲目前信號處理的主流。因此，我們就以感測器測得所要控制的類比信號，經類比-數位轉換器(analog-digital converter，簡稱 **ADC**)將它轉換成數位信號。如此一來，就可進行較高效率的處理、儲存或傳輸。當處理完成後，再經數位-類比轉換器(digital-analog converter，簡稱 **DAC**)將它轉換成類比信號，以驅動控制裝置(如電熱器、電磁閥、馬達等)，如此形成一個閉迴路(closed loop)的控制系統，如下圖所示：

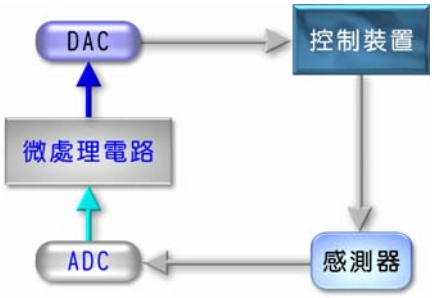


圖2　閉迴路控制系統

在本單元裡將介紹類比-數位轉換、數位-類比轉換，及其應用。

11-1 類比-數位轉換原理

ADC 與 DAC 之應用

類比-數位轉換是將類比信號變成數位信號，而轉換的方式如下說明：

並列式類比-數位轉換

並列式類比-數位轉換是以多個比較器(運算放大器)並列處理，又稱爲**比較器型類比-數位轉換**。此種類比-數位轉換以數個比較器同時偵測輸入的類比信號，然後予以編碼，即可產生數位信號，如下圖所示：

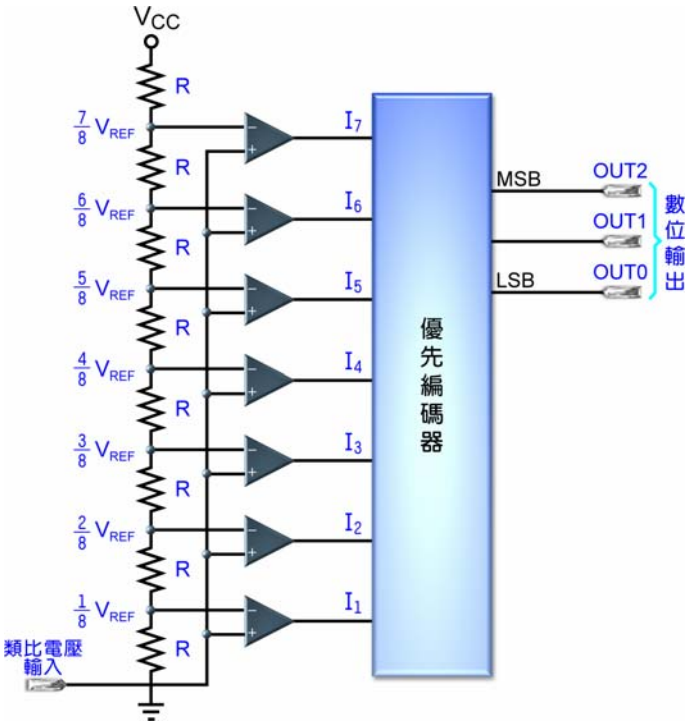


圖3　並列式類比數位轉換

並列式類比-數位轉換的特性如下：

- 轉換速度快。
- 所需要的電路較複雜，以 n 個位元的並列式類比-數位轉換爲例，則需要 2^n 個精密電阻器、2^n-1 個比較器，以及一個 n 位元的優先編碼器。

逐漸接近式類比-數位轉換

逐漸接近式類比-數位轉換器(successive-approximation ADC)採乘 2/除 2 比對、快速接近的方式，將類比信號轉換成數位信號，首先將參考電壓 V_r 與輸入類比信號比較；若輸入類比信號較高，則 V_r 乘以 2，再與輸入類比信號比較；若輸入類比信號還是比較高，則再將 V_r 乘以 2，與輸入類比信號比較…。反之，若輸入類比信號比較低，則將 V_r 除以 2，再與輸入

類比信號比較...，最後即可找到最接近的值。

對於類比電壓而言，乘以 2 或除以 2 都不容易操作，不過，對於數位信號而言，只要將資料左移一位，就是乘以 2；而資料右移一位，就是除以 2。移位之後的數位資料再經過數位-類比轉換，即可產生相對應的類比信號 V_r。就可與輸入的類比電壓 V_a 相比較，以產生左移或右移的控制信號，而控制移位暫存器的動作，如下圖所示：

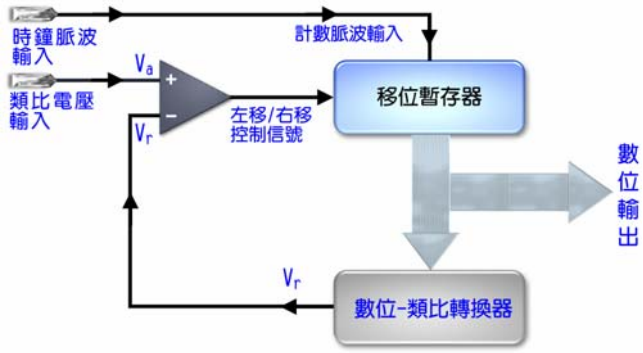


圖4　逐漸接近式類比-數位轉換概念圖

如上圖所示，當 $V_r<V_a$ 時，移位暫存器將左移，而 $V_r>V_a$ 時，移位暫存器將右移。$V_r=V_a$ 時，即可輸出數位信號。逐漸接近式類比-數位轉換的特性如下：

- n 位元之逐漸接近式類比-數位轉換，其轉換時間爲 n 個時鐘脈波，其轉換速度僅次於並列式類比-數位轉換。
- 電路較並列式類比-數位轉換之電路簡單。

連續計數式類比-數位轉換

連續計數式類比-數位轉換器(continuons counting ADC)是利用比較器、上下計數器與數位-類比轉換器，構成一個閉迴路的轉換電路，將輸入的類比信號與輸出端經數位-類比轉換器，回授回來的信號比較，以產生計數器的上數/下數控制信號，以計數外部輸入的時鐘脈波，如下圖所示：

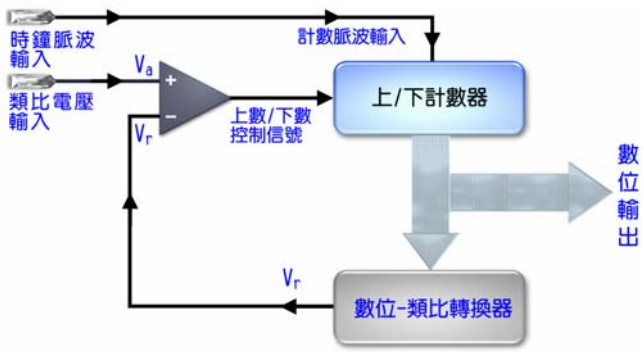


圖5　連續計數式類比-數位轉換概念圖

如上圖所示，當 $V_r<V_a$ 時，計數器將上數，而 $V_r>V_a$ 時，計數器將下數。當 $V_r=V_a$ 時，即停止計數，而輸出數位信號。連續計數式類比-數位轉換的特性如下：

- 轉換速度依輸入類比電壓而不同，類比電壓越高所需轉換時間越長。
- 電路較並列式類比-數位轉換之電路簡單。

雙斜率式類比-數位轉換

雙斜率式類比-數位轉換器(dual slope ADC)屬於**積分式類比-數位轉換器**的一種，這是以**定電流積分器**，先以輸入的類比信號來充電，然後改以固定的參考電壓，予以放電，而放電期間就是計數器計數的時間。放電完畢時，將停止計數，而計數的結果就是所要輸出的數位信號，如下圖所示：

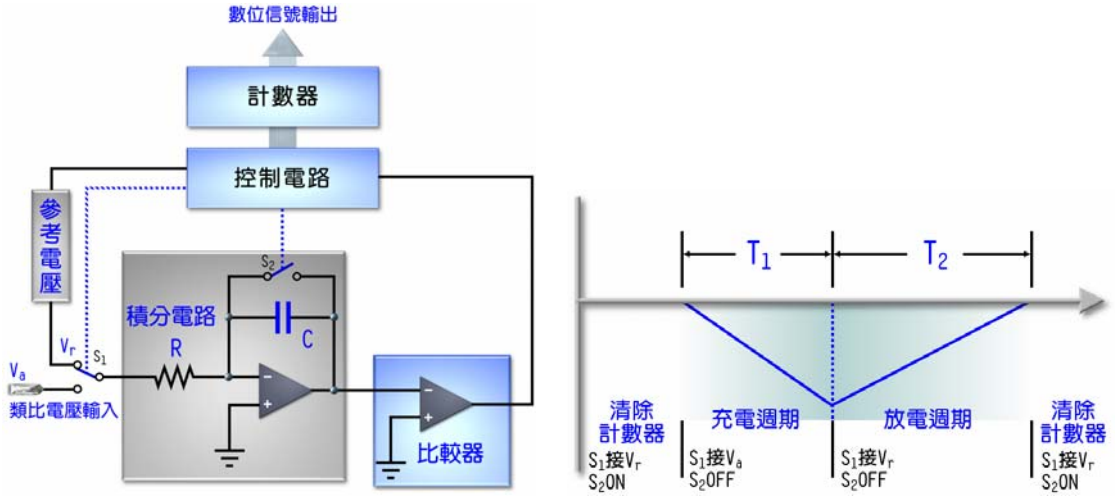


圖6　雙斜率式類比-數位轉換概念圖

如上圖所示，T1 爲輸入類比電壓充電所產生的斜率，T2 則爲連接參考電壓放電的所產生的斜率。雙斜率式類比-數位轉換的特性如下：

- 轉換速度最慢。
- 精密度高，穩定性佳。
- 雜訊免役力良好。

11-2 認識 AD 轉換 IC

ADC 與 DAC 之應用

市面上，類比-數位 IC 很多，而在學校裡以 ADC0804 系列爲主，在此將以 ADC0804 爲例，如下說明：

特性

ADC0804 之特性如下：

- CMOS 的逐漸接近式 AD 轉換器。
- 具有 8 位元解析能力，轉換時間爲 100 微秒，而最大誤差爲 1 個 LSB 值(最小電壓刻度)。
- 採差動式類比電壓輸入，三態式數位輸出。

接腳

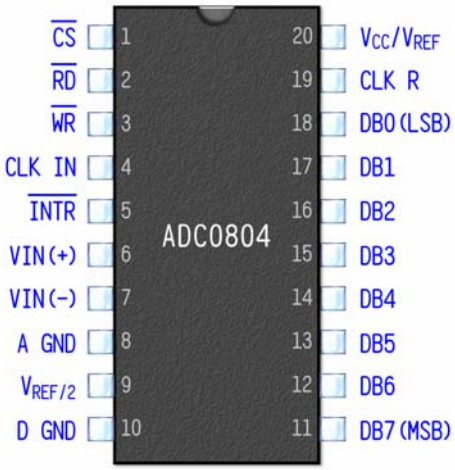


圖7　ADC0804 接腳圖

如上圖所示爲 ADC0804 之接腳圖，其中各腳說明如下：

- $\overline{CS}$：晶片選擇接腳，此爲低態動作接腳，若 $\overline{CS}$=0，則 ADC0804 動作；若 $\overline{CS}$=1，則 ADC0804 不動作，輸出資料接腳 DB0～DB7 呈現高阻抗狀態。
- $\overline{RD}$：資料讀取接腳，此爲低態動作接腳，若 $\overline{CS}$=0、且 $\overline{RD}$=0 時，則可由 DB0～DB7 讀取 ADC0804 的輸出數位資料。
- $\overline{WR}$：開始轉換接腳，此爲低態動作接腳，若 $\overline{WR}$=0，即可使 ADC0804 開始進行類比-數位轉換動作。
- $\overline{INTR}$：完成轉換接腳，此爲低態動作接腳，若 $\overline{INTR}$=0，表示 ADC0804 已完成類比-數位轉換動作，而此信號常被用來通知微處理機，請它中斷而前來提取數位資料。
- **CLK IN**：時鐘脈波輸入接腳，ADC0804 接受 100 到 1460kHz 的時鐘脈波。而我們可配合 CLK R 接腳，以外加的電阻、電容，由內部電路自行產生時鐘脈波，如下圖所示，其頻率爲：

$$f_{CLK} = \frac{1}{1.1RC}$$

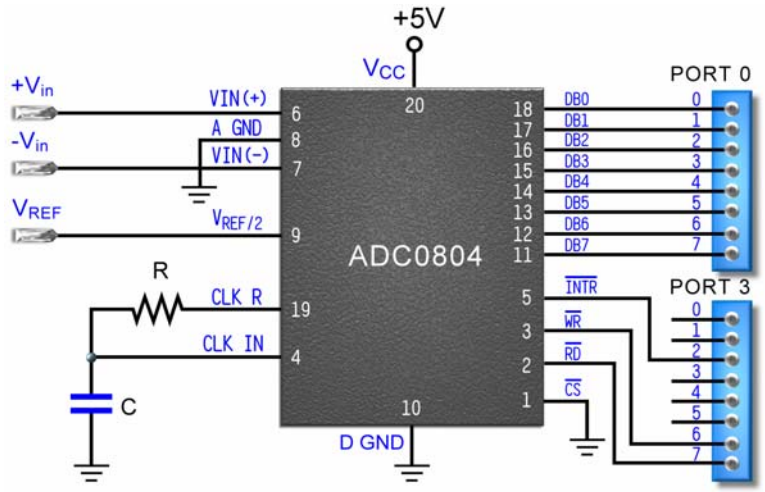


圖8　ADC0804 時鐘脈波電路

- **CLK R**：時鐘脈波輸出接腳，如上圖所示，可連接電阻器，以產生時鐘脈波。
- **$V_{REF/2}$**：參考電壓輸入接腳。通常本接腳所連接的電壓，即輸入類比電壓最大值的一半。

- **V_{IN+}**：類比電壓輸入之正端接腳，所輸入的類比電壓不得超過 $V_{REF/2}$ 接腳的電壓。
- **V_{IN-}**：類比電壓輸入之負端接腳。
- **V_{CC}**：電源接腳或參考電壓接腳，通常是連接+5V，以做爲電源之用。若 $V_{REF/2}$ 接腳沒有連接參考電壓時，則 ADC0804 則以本接腳上的電壓爲參考電壓。
- **D GND**：數位信號接地接腳。
- **A GND**：類比信號接地接腳，通常本接腳都與 D GND 接腳連接後接地，若處理高干擾性的類比信號，本接腳自可單獨接地。
- **DB0～DB7**：數位輸出資料接腳，此八隻接腳爲三態式輸出，可直接連接微處理機的資料匯流排，若此 IC 不輸出時，則這八隻接腳呈現高阻抗狀態。

操作方式

ADC0804 之操作方式說明如下：

- **連續轉換**：ADC0804 最簡單的操作方式，就是讓它不停地進行轉換，如下圖所示，$\overline{CS}$ 與 $\overline{RD}$ 接腳連接到接地端，再將 $\overline{INTR}$ 接腳連接到 $\overline{WR}$ 接腳，如此就可令 $\overline{INTR}$ 接腳輸出的完成轉換信號，成爲 $\overline{WR}$ 接腳的開始轉換信號。而微處理機隨時可讀取這個資料匯流排上的資料。

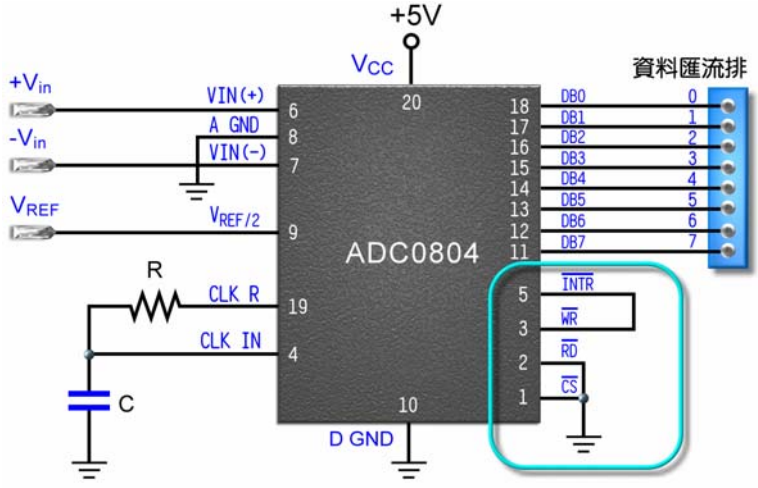


圖9　ADC0804 之連續轉換電路

- **交握式控制**：如下圖所示，將 $\overline{CS}$ 接腳接地、將 $\overline{WR}$ 與 $\overline{RD}$ 接腳連接到微處理機的輸出埠，此信號稱爲 **START** 或 **SOC**(start of convert)，若微處理機透過這個輸出埠輸出一個負脈波，則 ADC0804 即可進行類比-數位轉換。當 ADC0804 完成轉換後，則由 $\overline{INTR}$ 接腳輸出一個低態的脈波，此信號稱爲 IRQ，若將這個信號連接到微處理機的輸入埠，則該微處理機將可以垂詢方式或中斷方式偵測得到，而進行 ADC0804 數位資料的讀取；若將這個信號連接到微處理機的外部中斷接腳，則該微處理機將中斷，以進行 ADC0804 數位資料的讀取。

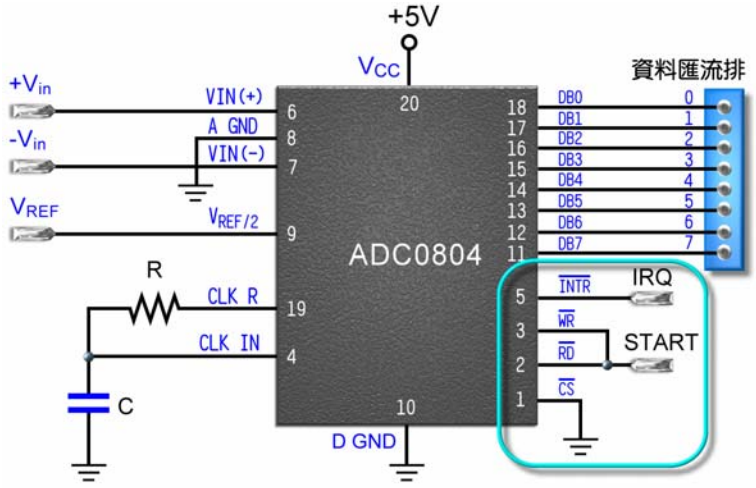


圖10　ADC0804 之交握式控制電路

ADC0804 之操作時序如下圖所示：

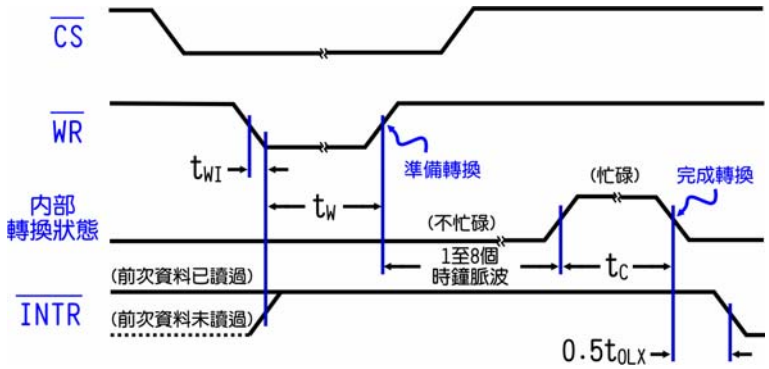


圖11　ADC0804 之轉換時序圖

如上圖所示，當 ADC0804 之 $\overline{CS}$ 接腳爲低態，且 $\overline{WR}$ 接腳也爲低態

時，ADC0804 內部開始進行轉換，轉換期間，$\overline{\text{INTR}}$ 接腳爲高態。如下圖所示，當 ADC0804 內部轉換完成後，$\overline{\text{INTR}}$ 接腳轉爲低態。這時候，若 $\overline{\text{CS}}$ 接腳又變成低態，且 $\overline{\text{RD}}$ 接腳也爲低態，則 ADC0804 轉換的結果，將隨 $\overline{\text{INTR}}$ 接腳轉爲高態時，放入資料匯流排(DB0-DB7)，以供微處理機讀取。

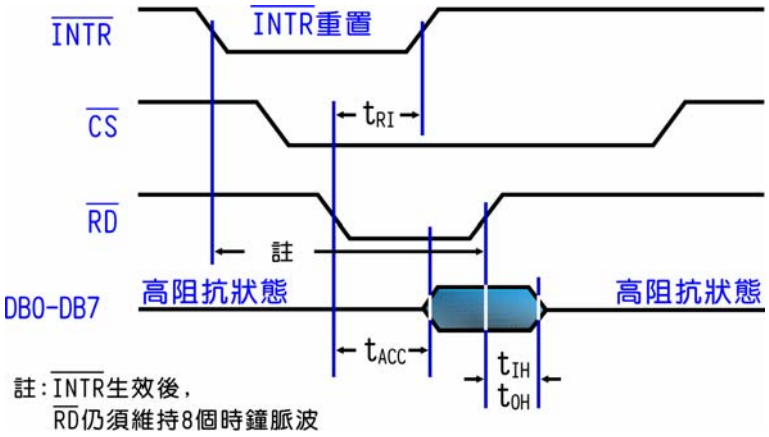


圖12　ADC0804 之時序圖

整個交握的控制，如下圖所示，第一步由微處理機送一個低態的 $\overline{\text{WR}}$ 信號到 ADC0804，以啓動 ADC0804；當 ADC0804 轉換完成後，即進入第二步，ADC0804 送出一個低態的 $\overline{\text{INTR}}$ 信號，請微處理機來提取；第三步，當微處理機要來提取之前，送一個低態的 $\overline{\text{RD}}$ 信號通知 ADC0804；第四步，微處理機即可讀取匯流排的資料。

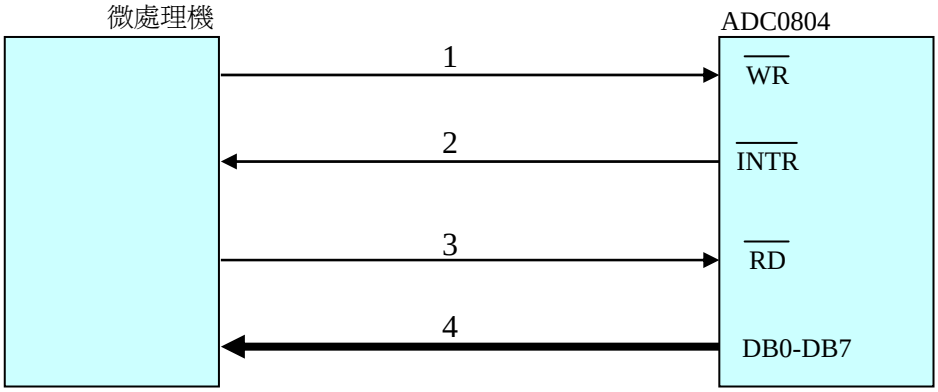


圖13　交握信號

8051 與 ADC0804 之連接

基本上，8051 與 ADC0804 之連接可分爲控制線與資料線，若 ADC0804 採用**連續轉換**的方式(如圖 9 所示)，則不須連接控制線，直接將 DB0～DB7 連接到 8051 的任一個 PORT 即可。在程式方面，也把連接的輸出入埠，當成一般的輸入埠，隨時讀取其中的資料。若 ADC0804 採用交握式控制，則除了資料線外，還須將其 START 及 IRQ 控制線(如圖 10 所示)，各連接到 8051 輸出入埠的任一位元，例如 P2.0、P2.1 等。我們可將 ADC0804 的 $\overline{WR}$ 接腳連接到 8051 的 $\overline{WR}$ 接腳(P3.6)、將 ADC0804 的 $\overline{RD}$ 接腳連接到 8051 的 $\overline{RD}$ 接腳(P3.7)、將 ADC0804 的 $\overline{INTR}$ 接腳連接到 8051 的 INT0 接腳(P3.2)，而 DB0～DB7 連接到 8051 的 PORT 0，如下圖所示：

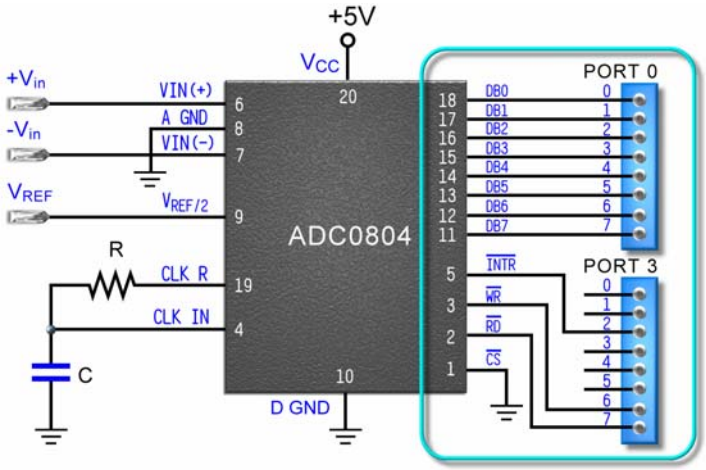


圖14　ADC0804 與 8051 連接

當我們要進行 8051 的外部記憶體操作時，若使用組合語言，則可透過專用的外部記憶體存取指令－MOVX 指令；而在 C 語言裡並沒有獨立的存取外部記憶體的指令，而是以「xdata」記憶體形式做爲操作外部記憶體的依據。只要將某個資料變數宣告爲「xdata」記憶體形式，則該變數將視爲一個外部記憶體，只要動到該資料，就會觸動外部記憶體的操作。例如要宣告一個 8 位元的「xdata」記憶體形式變數 adc，如下：

```
unsigned char xdata    adc;
```

此後，若要將某一資料放入 adc 變數，例如：

```
adc=0xff;
```

則 8051 的 $\overline{WR}$ 接腳送出一個低態信號，同時 Port 0 將輸出 0xff(不重要)。同樣地，若要將 adc 變數存到另一個變數，例如：

```
results=adc;
```

則 8051 的 $\overline{RD}$ 接腳送出一個低態信號，同時，將讀取 Port 0 的資料。

在進行 ADC0804 的控制時，當然先要宣告「xdata」記憶體形式的變數，緊接著的第一步是由 8051 透過 $\overline{WR}$ 接腳，送出一個低態的啓動信號，這時候就可以利用剛才所介紹的「adc=0xff;」指令，以啓動 ADC0804。當 ADC0804 完成轉換後，將透過 $\overline{INTR}$ 接腳通知 8051；而這支接腳連接到 8051 的 INT0 接腳(P3.2)，我們可利用 INT 0 中斷的方式，在中斷副程式裡，以剛才所介紹的「results=adc;」指令，將 ADC0804 轉換的結果，存入 results 變數。

11-3 數位-類比轉換原理

ADC 與 DAC 之應用

基本上，**數位-類比轉換器**(digital-analog converter，簡稱 **DAC**)是由電阻網路所構成，常見的數位-類比轉換電路有加權電阻網路及 R-2R 電阻網路兩種，如下說明：

加權電阻網路

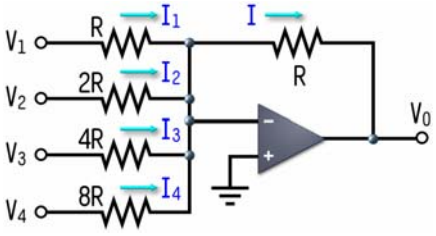


圖15　加權電阻網路

如上圖所示爲加權電阻網路，其中各電流如下：

$$I_1=\frac{V_1}{R}\ 、\ I_2=\frac{V_2}{2R}\ 、\ I_3=\frac{V_3}{4R}\ 、\ I_4=\frac{V_4}{8R}$$

$I=I_1+I_2+I_3+I_4$

$$
\begin{aligned}
V_O &= -IR \\
&= -(I_1 + I_2 + I_3 + I_4)R \\
&= -(\frac{V_1}{R} + \frac{V_2}{2R} + \frac{V_3}{4R} + \frac{V_4}{8R})R \\
&= -(V_1 + \frac{1}{2}V_2 + \frac{1}{4}V_3 + \frac{1}{8}V_4) \\
&= -\frac{1}{8}(8V_1 + 4V_2 + 2V_3 + V_4) \\
&= -\frac{1}{8}(2^3V_1 + 2^2V_2 + 2^1V_3 + 2^0V_4)
\end{aligned}
$$

其中的 V_1、V_2、V_3、V_4 分別爲數位資料的 bit 3、bit 2、bit 1 及 bit 0，其電壓值非 0V 就是 5V，而 $-\frac{1}{8}$ 可由運算放大器的回授電阻來調整其大小。若此電路輸入 1111 數位資料，則輸出電壓 V_O 爲：

$$
\begin{aligned}
V_O &= -(5+\frac{1}{2}\times 5+\frac{1}{4}\times 5+\frac{1}{8}\times 5) \\
&= -(5+2.5+1.25+0.625) \\
&= -9.375
\end{aligned}
$$

同理，若輸入 1110 數位資料，則輸出電壓 V_O 爲：

$$
\begin{aligned}
V_O &= -(5+\frac{1}{2}\times 5+\frac{1}{4}\times 5+\frac{1}{8}\times 0) \\
&= -(5+2.5+1.25+0) \\
&= -8.75
\end{aligned}
$$

如下表所示爲這個網路的輸出入關係：

bit3	bit2	bit1	bit0	V_O	bit3	bit2	bit1	bit0	V_O
0	0	0	0	0	1	0	0	0	-5
0	0	0	1	-0.625	1	0	0	1	-5.625
0	0	1	0	-1.25	1	0	1	0	-6.25
0	0	1	1	-1.875	1	0	1	1	-6.875
0	1	0	0	-2.5	1	1	0	0	-7.5
0	1	0	1	-3.125	1	1	0	1	-8.125
0	1	1	0	-3.75	1	1	1	0	-8.75
0	1	1	1	-4.375	1	1	1	1	-9.375

由上述可得知，加權電阻網路數位-類比轉換的原理，在此將其特性歸納於下列：

- 電路結構簡單，但不容易製作，因爲其中所使用的電阻值，種類太多，差異過大。在 IC 的內部電路裡，很難做出這樣的電路。
- 由於最大與最小的電阻差異太大，非常容易造成誤差，以 8 位元的轉換電路爲例，其中最大電阻爲最小電阻的 256 倍，若電阻的誤差爲 1%，則最大電阻的誤差值就比最小電阻或次小電阻還大了！所以，很難達到較高的精確度。

R-2R 電阻網路

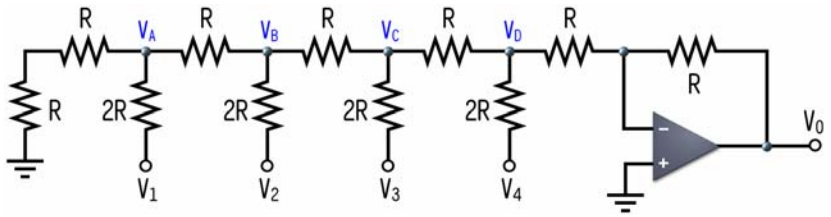
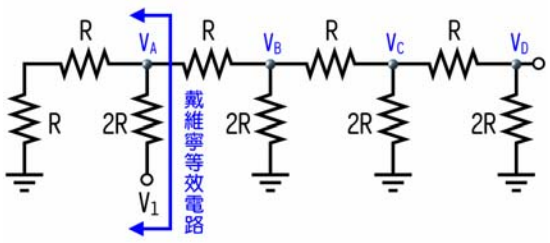


圖16　R-2R 電阻網路

如上圖所示爲 R-2R 電阻網路，電路結構很有規律，在此將以重疊定理與戴維寧等效電路來解析 V_D 的電壓：

1. 只考慮 V_1，則 V_2、V_3、V_4 視爲接地，如下所示：

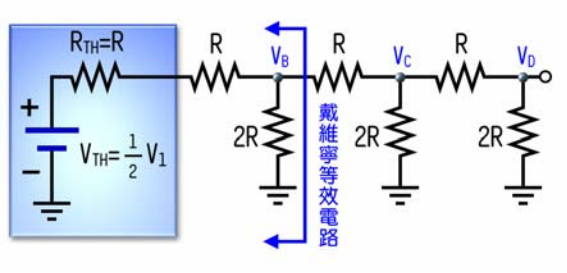


由 V_A 向左看的戴維寧等效電路爲：

$R_{TH}=(R+R)//2R$

$$V_{TH}=V_1\times\frac{2R}{4R}=\frac{1}{2}V_1$$

電路可改爲：

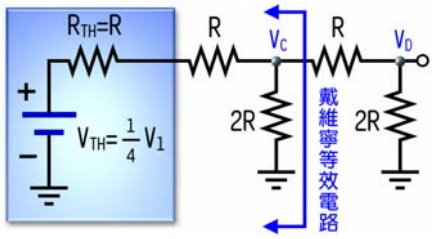


由 V_B 向左看的戴維寧等效電路爲：

$R_{TH}=(R+R)//2R$

$$V_{TH}=\frac{1}{2}V_1\times\frac{2R}{4R}=\frac{1}{4}V_1$$

電路可改爲：

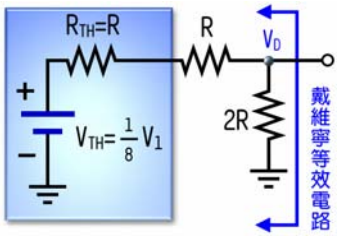


由 V_C 向左看的戴維寧等效電路爲：

$R_{TH}=(R+R)//2R$

$$V_{TH}=\frac{1}{4}V_1\times\frac{2R}{4R}=\frac{1}{8}V_1$$

電路可改爲：

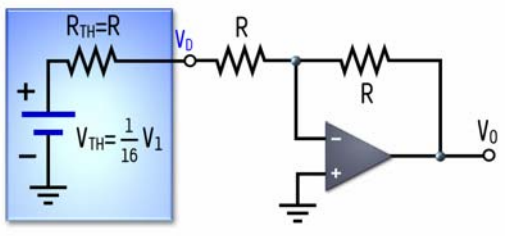


由 V_D 向左看的戴維寧等效電路爲：

$R_{TH}=(R+R)//2R$

$$V_{TH}=\frac{1}{8}V_1\times\frac{2R}{4R}=\frac{1}{16}V_1$$

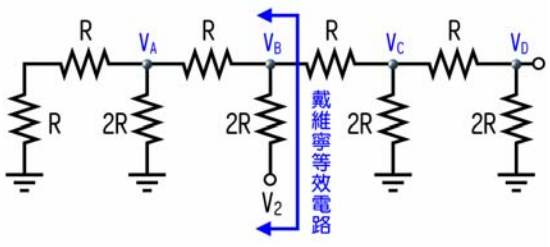
電路可改爲：



所以，

$$V_O' = -\frac{R}{2R} \times \frac{1}{16} V_1 = -\frac{1}{32} V_1$$

2. 只考慮 V_2，則 V_1、V_3、V_4 視爲接地，如下所示：

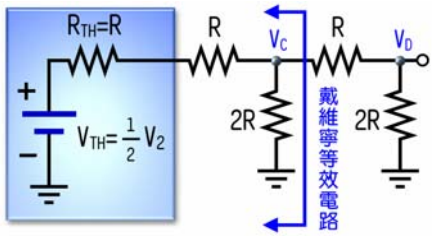


由 V_B 向左看的戴維寧等效電路爲：

$R_{TH}=((R+R)//2R+R)//2R=R$

$$V_{TH} = V_2 \times \frac{2R}{4R} = \frac{1}{2} V_2$$

電路可改爲：

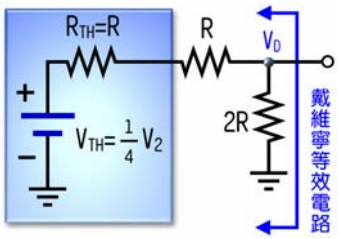


由 V_C 向左看的戴維寧等效電路爲：

$R_{TH}=(R+R)//2R$

$$V_{TH} = \frac{1}{2} V_2 \times \frac{2R}{4R} = \frac{1}{4} V_2$$

電路可改爲：

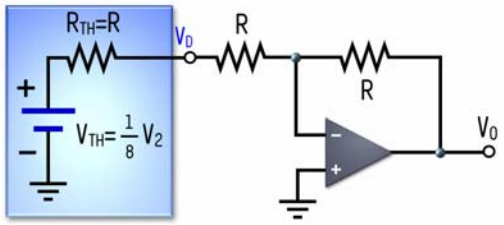


由 V_D 向左看的戴維寧等效電路爲：

$R_{TH}=(R+R)//2R$

$$V_{TH}=\frac{1}{4}V_2\times\frac{2R}{4R}=\frac{1}{8}V_2$$

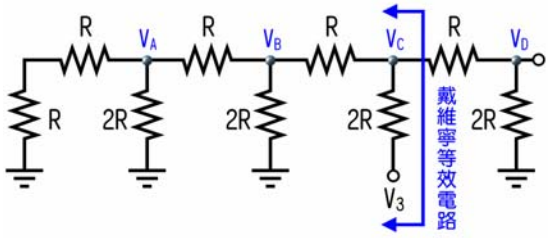
電路可改爲：



所以，

$$V_O''=-\frac{R}{2R}\times\frac{1}{8}V_2=-\frac{1}{16}V_2$$

3. 只考慮 V_3，則 V_1、V_2、V_4 視爲接地，如下所示：

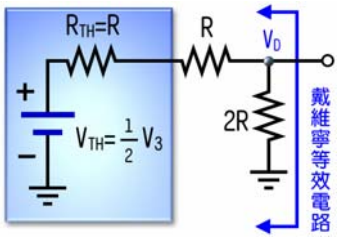


由 V_C 向左看的戴維寧等效電路爲：

$R_{TH}=(((R+R)//2R+R)//2R+R)//2R=R$

$$V_{TH}=V_3\times\frac{2R}{4R}=\frac{1}{2}V_3$$

電路可改爲：

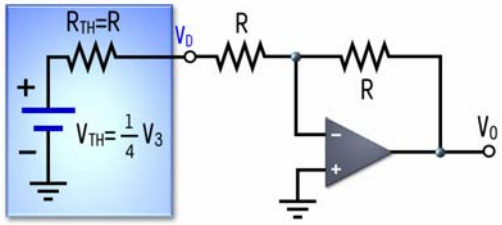


由 V_D 向左看的戴維寧等效電路爲：

$R_{TH}=(R+R)//2R$

$$V_{TH}=\frac{1}{2}V_3\times\frac{2R}{4R}=\frac{1}{4}V_3$$

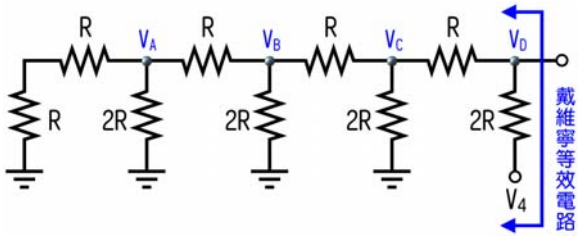
電路可改爲：



所以，

$$V_O'''=-\frac{R}{2R}\times\frac{1}{4}V_3=-\frac{1}{8}V_3$$

4. 只考慮 V_4，則 V_1、V_2、V_3 視爲接地，如下所示：

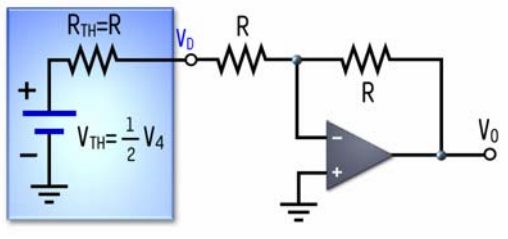


由 V_D 向左看的戴維寧等效電路爲：

$R_{TH}=((((R+R)//2R+R)//2R+R)//2R+R)//2R=R$

$$V_{TH}=V_4\times\frac{2R}{4R}=\frac{1}{2}V_4$$

電路可改爲：



所以，$V_O'''' = -\frac{R}{2R} \times \frac{1}{2} V_4 = -\frac{1}{4} V_4$

綜合前四項

$V_O' = -\frac{R}{2R} \times \frac{1}{16} V_1 = -\frac{1}{32} V_1$

$V_O'' = -\frac{R}{2R} \times \frac{1}{8} V_2 = -\frac{1}{16} V_2$

$V_O''' = -\frac{R}{2R} \times \frac{1}{4} V_3 = -\frac{1}{8} V_3$

$V_O'''' = -\frac{R}{2R} \times \frac{1}{2} V_4 = -\frac{1}{4} V_4$，根據重疊定理可得

$$V_O = V_O' + V_O'' + V_O''' + V_O''''$$
$$= -(\frac{1}{32} V_1 + \frac{1}{16} V_2 + \frac{1}{8} V_3 + \frac{1}{4} V_4)$$
$$= -\frac{1}{32}(V_1 + 2V_2 + 4V_3 + 8V_4)$$
$$= -\frac{1}{32}(2^0 V_1 + 2^1 V_2 + 2^2 V_3 + 2^3 V_4)$$

其中的 V_1、V_2、V_3、V_4 分別爲數位資料的 bit 0、bit 1、bit 2 及 bit 3，其電壓值非 0V 就是 5V，而 $-\frac{1}{32}$ 可由運算放大器的回授電阻來調整其大小。若此電路輸入 1111 數位資料，則輸出電壓 V_O 爲：

$$V_O = -\frac{1}{32}(5 + 10 + 20 + 40)$$
$$= -\frac{75}{32}$$
$$= -2.345375$$

由上述可得知，R-2R 電阻網路數位-類比轉換的原理，在此將其特性歸納於下列：

- 電路結構簡單，其中的電阻值只有兩種，不管是自製電路，或 IC 的內部電路裡，都很容易實現這樣的電路。

- 不管是 ADC 或是 DAC，其電壓解析度 V_{RES}(或 V_{LSB})與其數位的位元數及電壓範圍(參考電壓)有關，如下：

$$V_{RES} = \frac{V_{REF}}{2^n - 1} = \frac{V_{\max} - V_{\min}}{2^n - 1}$$

其中的 V_{REF} 就是參考電壓，V_{max} 爲最大電壓，V_{min} 爲最小電壓，n 爲位元數。

11-4 認識 DA 轉換 IC

ADC 與 DAC 之應用

市面上，數位-類比轉換 IC 不少，而在學校裡以 DAC-08 系列(1408)爲主，在此就以 DAC-08 爲例，如下說明：

特性

DAC08 之特性如下：

- 電流型 R-2R 電阻網路的 DA 轉換器。
- 具有 8 位元解析能力，轉換時間爲 300 奈秒。
- 電源可採用±15V 雙電源，或+5 到+15 單電源。

接腳

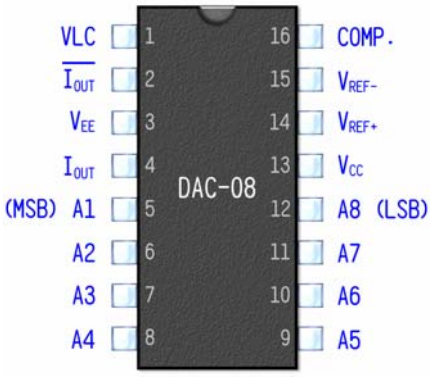


圖17 DAC-08 接腳圖

如上圖所示爲 DAC-08 之接腳圖，其中各接腳說明如下：

- VLC：臨界電壓控制輸入接腳，其功能是設定數位信號準位，接地即可。
- $\overline{I_{OUT}}$：互補類比電流輸出接腳，$\overline{I_{OUT}}$ =I_{FS}-I_{OUT}，其中的 I_{OUT} 爲類比輸出電流，I_{FS} 爲滿刻度電流(約 0.2 毫安到 4 毫安之間)

$$I_{FS} = \frac{V_{REF}}{R_{REF}} \times \frac{255}{256}$$

- V_{EE}：負電源接腳，其電壓範圍爲-4.5 到-18V。
- I_{OUT}：類比電流輸出接腳，而

$$I_{OUT} = \frac{2^{n-1} \cdot D_{n-1} + 2^{n-2} \cdot D_{n-2} + ... + 2^0 \cdot D_0}{2^n} \times I_{REF}$$

其中 $I_{REF} = \frac{V_{REF}}{R_{REF}}$

$$V_O = -I_{OUT} \times R_O = -\frac{2^{n-1} \cdot D_{n-1} + 2^{n-2} \cdot D_{n-2} + ... + 2^0 \cdot D_0}{2^n} \times I_{REF} \times R_O$$

- A1～A8：此八隻接腳爲數位輸出接腳，其中 A1 爲最高位元(MSB)、A8 爲最低位元(LSB)。
- V_{CC}：正電源接腳，其電壓範圍爲+4.5 到+18V。
- V_{REF+}：正參考電壓輸入接腳。
- V_{REF-}：負參考電壓輸入接腳。
- COMP.：補償接腳，外接補償電容器，以避免高頻振盪。

操作方式

如圖 18 所示爲 DAC-08 的基本電路，其中將正電源連接+5V、負電源接地，I_{REF}=V_{REF}/R_{REF}，若希望 I_{REF} 爲 1mA，而 V_{REF} 連接+5V 的話，則 R_{REF} 可採用 5k 歐姆即可。

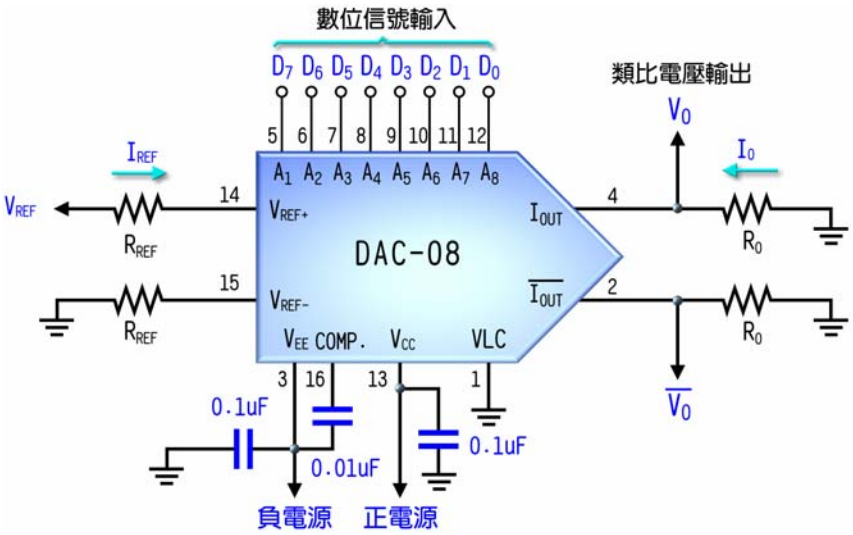


圖18　基本電路

另外，由於

$$I_{OUT}=\frac{2^{n-1}\cdot D_{n-1}+2^{n-2}\cdot D_{n-2}+...+2^{0}\cdot D_{0}}{2^{n}}\times I_{REF}$$ ，所以

$$V_O=-I_{OUT}\times R_O=-\frac{2^{n-1}\cdot D_{n-1}+2^{n-2}\cdot D_{n-2}+...+2^{0}\cdot D_{0}}{2^{n}}\times I_{REF}\times R_O$$ ，

若使 R_O 也爲 5k 歐姆，則

$$I_{OUT}=\frac{2^{7}\cdot D_{7}+2^{6}\cdot D_{6}+...+2^{0}\cdot D_{0}}{256}\times 1m$$

$$V_O=-\frac{2^{7}\cdot D_{7}+2^{6}\cdot D_{6}+...+2^{0}\cdot D_{0}}{256}\times 1m\times 5k$$

$$=-\frac{2^{7}\cdot D_{7}+2^{6}\cdot D_{6}+...+2^{0}\cdot D_{0}}{256}\times 5$$

若數位信號輸入爲 11111111，則類比電壓輸出爲：

$$I_{OUT}=\frac{2^{7}\cdot 1+2^{6}\cdot 1+...+2^{0}\cdot 1}{256}\times 1m=\frac{255}{256}m\cong 0.996m\,A$$

$$V_O=-0.996m\times 5\cong -4.98\,V$$

若數位信號輸入爲 11111110，則類比電壓輸出爲：

$$I_{OUT} = \frac{2^7 \cdot 1 + 2^6 \cdot 1 + ... + 2^0 \cdot 0}{256} \times 1m = \frac{254}{256} m \cong 0.992m\,A$$

$$V_O = -0.992m \times 5 \cong -4.96\,V$$

同理，若數位信號輸入爲 00000001，則類比電壓輸出爲：

$$I_{OUT} = \frac{1}{256} m \cong 0.0039m\,A$$

$$V_O = -0.0039m \times 5 \cong -0.02\,V$$

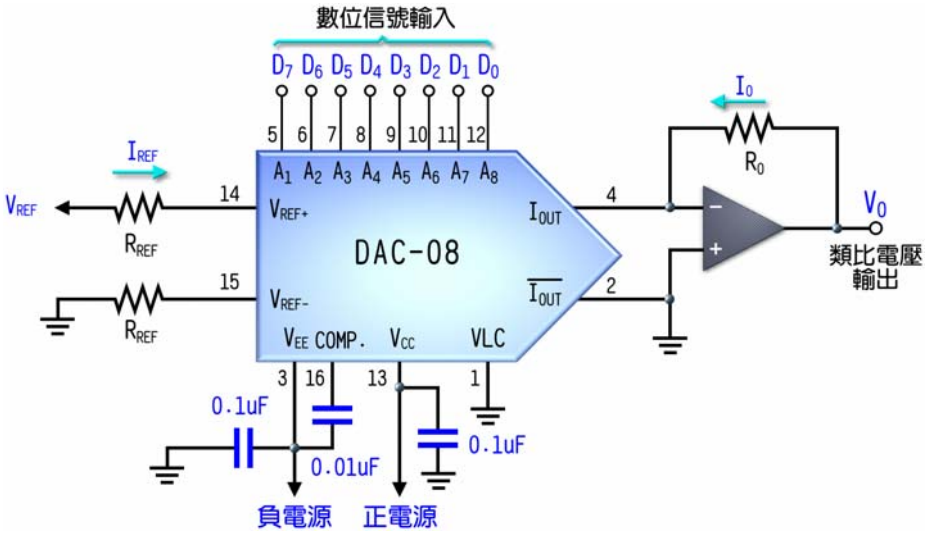


圖19　低阻抗輸出電路

由 DAC-08 直接輸出的話，由於輸出阻抗較高，容易造成負載效應，所以在其輸出端加一個運算放大器，如上圖所示(其中運算放大器的電源可採±12V)，即可得到較佳的輸出結果，而其電壓轉換的結果，除沒有的 $\overline{V_O}$ 輸出外，與原電路相同。若要有 V_O 與 $\overline{V_O}$ 對稱性輸出，則可將電路改如下圖：

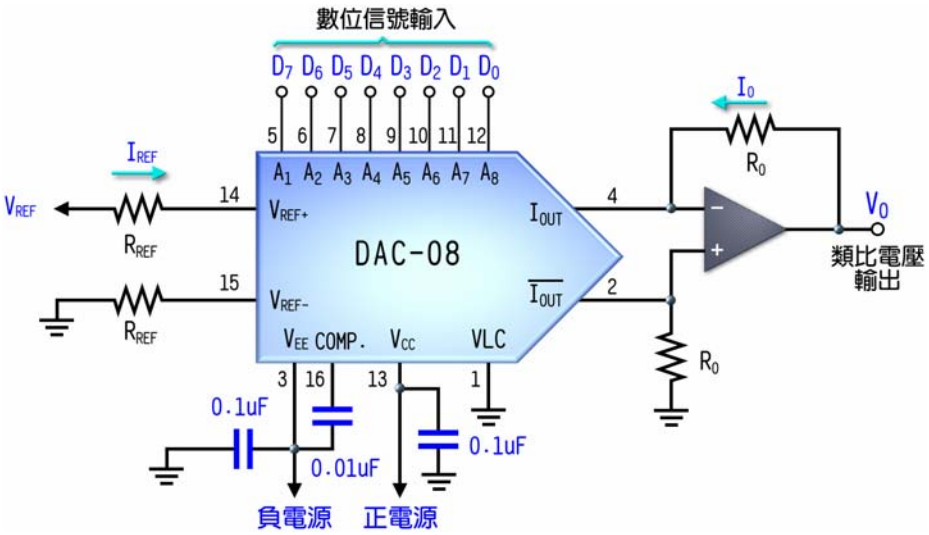


圖20　對稱性輸出電路

8051 與 DAC-08 之連接

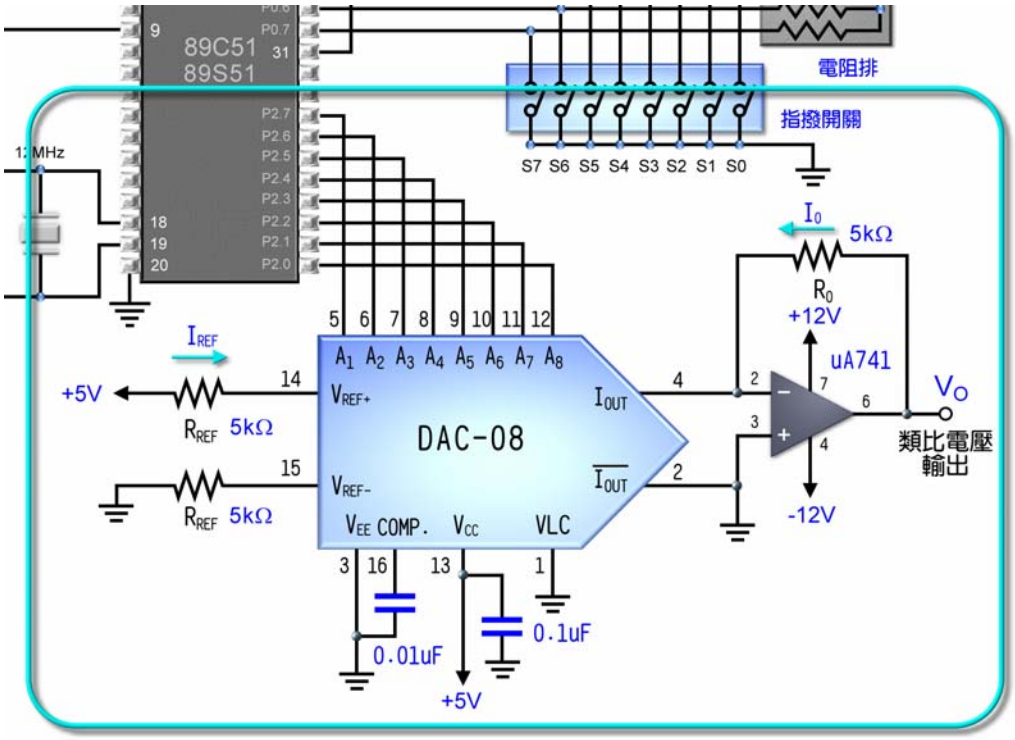


圖21　DAC-08 與 8051 連接

如上圖所示，8051 與 DAC-08 之連接只是簡單的匯流排連接而已，例如把 DAC-08 的 A1～A8 連接到 8051 的 Port 2(或其它輸出入埠)，就可把 8051 由 Port 2 輸出的數位資料轉換成類比電壓。

11-5 內建 ADC 之 51 系列

ADC 與 DAC 之應用

大部分提供 51 系列的半導體廠商都有提供內建 ADC 的 51 單晶片，以 Atmel 公司爲例，其內建 ADC 的 51 單晶片有 AT89C5115、AT89C51AC2 及 AT89C51AC3 等，而這三個單晶片除內建 10 位元 ADC 外，其內部結構也比標準的 89C51 強很多！簡介如下：

AT89C5115

- RAM：256 bytes RAM、256 bytes XRAM。
- ROM：16k bytes Flash ROM。
- 14 個中斷源。
- 3 個 16 位元計時計數器。
- 1 個全雙工 UART。
- 最高工作頻率 40M Hz。
- 輸出入埠：16 或 20 條數位 I/O，視包裝而定。
- 雙通道 16 位元的 PCA，可做爲 8 位元脈波寬度調變(Pulse Width Modulation，簡稱 **PWM**)。
- 兩組資料指標暫存器。
- 21 位元看門狗計時器。
- 10 位元 ADC。
- 提供 Power-Down 及 Idle 等兩種省電模式。
- 電源範圍：3V 至 5.5V。
- 零件包裝：SOIC28、SOIC24、PLCC28、VQFP32，在學校或訓練單位可採用 PLCC28，配合 PLCC 腳座，以方便練習。

AT89C51AC2

- RAM：256 bytes RAM、1k bytes XRAM。
- ROM：32k bytes Flash ROM。
- 14 個中斷源。
- 3 個 16 位元計時計數器。
- 1 個全雙工 UART。
- 最高工作頻率 40M Hz。
- 輸出入埠：34 條數位 I/O，視包裝而定。

- 雙通道 16 位元的 PCA，可做爲 8 位元脈波寬度調變(Pulse Width Modulation，簡稱 **PWM**)。
- 兩組資料指標暫存器。
- 21 位元看門狗計時器。
- 10 位元 ADC。
- 晶片內置模擬器邏輯，即 On-chip Emulator Logic。
- 提供 Power-Down 及 Idle 等兩種省電模式。
- 電源範圍：3V 至 5.5V。
- 零件包裝：PLCC44、VQFP44，在學校或訓練單位可採用 PLCC44，配合 PLCC 腳座，以方便練習。

AT89C51AC3

- RAM：256 bytes RAM、2k bytes ERAM。
- ROM：32k bytes Flash ROM。
- 14 個中斷源。
- 3 個 16 位元計時計數器。
- 1 個全雙工 UART。
- 最高工作頻率 60M Hz。
- 輸出入埠：36 條數位 I/O，視包裝而定。
- 雙通道 16 位元的 PCA，可做爲 8 位元脈波寬度調變(Pulse Width Modulation，簡稱 **PWM**)。
- 兩組資料指標暫存器。
- 21 位元看門狗計時器。
- 10 位元 ADC。
- SPI 介面(SPI 爲 Serial Peripheral Interface 之簡稱)是連接 Motorola 公司所發展之**同步串列資料連接標準**(synchronous serial data link standard)，用以控制**串列週邊裝置介面匯流排**(即 Serial Peripheral Interface Bus)之控制器，如下圖所示爲 SPI 匯流排系統：

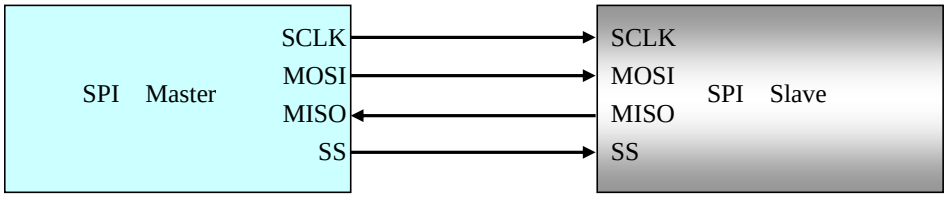


圖22 SPI 匯流排系統

- 晶片內置模擬器邏輯，即 On-chip Emulator Logic。
- 提供 Power-Down 及 Idle 等兩種省電模式。
- 電源範圍：3V 至 5.5V。
- 零件包裝：PLCC44、VQFP44、VQFP64、PLCC52，在學校或訓練單位可採用 PLCC44 或 PLCC52，配合 PLCC 腳座，以方便練習。

11-6 認識溫度感測器

ADC 與 DAC 之應用

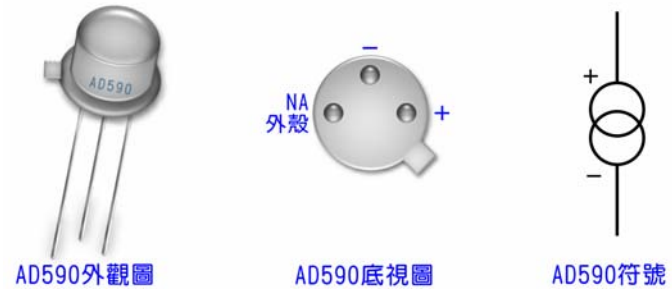


圖23 AD590 之外觀、底部接腳圖與符號

美國 Analog Device 公司所開發的 AD590 是體積小、使用方便的溫度感測器，如上圖所示，AD590 就像一般小型金屬殼包裝的電晶體，同樣是三隻接腳，很容易被誤以爲是電晶體，實際上，它是一個溫度感測 IC，而且是不便宜的小零件。雖然 AD590 有三隻接腳，通常只使用其中兩隻接腳，其特性如下說明：

- 其輸出電流與凱氏溫度成正比，凱氏溫度 0 度時輸出 0A，凱氏溫度每上升 1 度電流增加 1 微安(即 1μA/K)。其中的**凱氏溫度**(Kelvin temperature scale)，又稱爲**絕對溫度**(absolute temperature scale)，而凱氏溫度與**攝氏溫度**(Celsius temperature scale)之關係爲凱氏溫度等於攝氏溫度加上 273。換言之，攝氏溫度每上升 1 度 AD590 電流增加 1 微安。
- 有效溫度感測範圍爲-55℃到 150℃。
- 可採用的電源範圍爲 4V 到 30V。

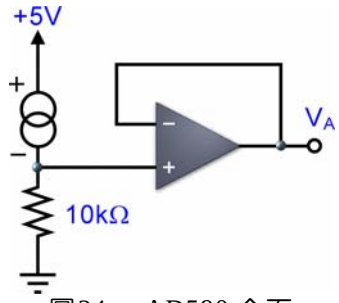


圖24　AD590 介面

如上圖所示，最簡單的 AD590 介面是串接一個 10kΩ電阻再接地，即可產生 10×(273.2+T℃)毫伏特，這個電壓先經一個運算放大器所組成的緩衝器，以避免負載效應。當 0℃時，V_A=10×273.2mV=2.732V、100℃時，V_A=10×373.2mV=3.732V，不是很人性化，如果將 V_A 減去 2.732，則 0℃時 V_A=0V、100℃時 V_A=1V，每增加 1℃，V_A 增加 0.01V(即 10mV)，這樣比較容易被接受！在此利用一個運算放大器，以進行減法功能，如下圖所示：

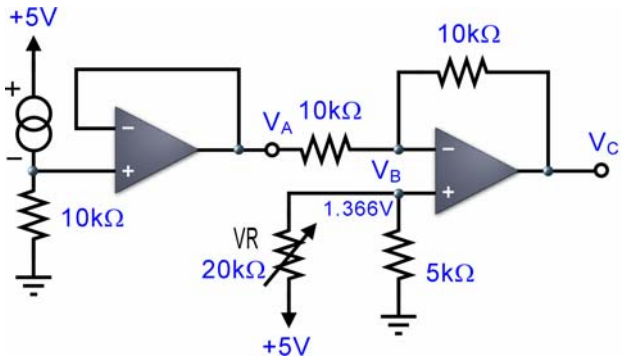


圖25　減去 2.732V

如上圖所示，請以數字電表量測 V_B，調整 VR 半固定電阻，讓 V_B 爲 1.366V，則 V_C=-V_A+2V_B=-(V_A-2.732)。

若要使用前述之 ADC0804 將此電壓轉換成數位信號，而 ADC0804 所採用的參考電壓 V_{REF} 爲 2.5V 的話，其 V_{LSB} 約爲 0.0196V(接近 0.02V)。則還需將圖 25 中的 V_C 再放大-2 倍，使溫度增加 1℃時，V_C 增加 0.02V，如圖 26 所示。

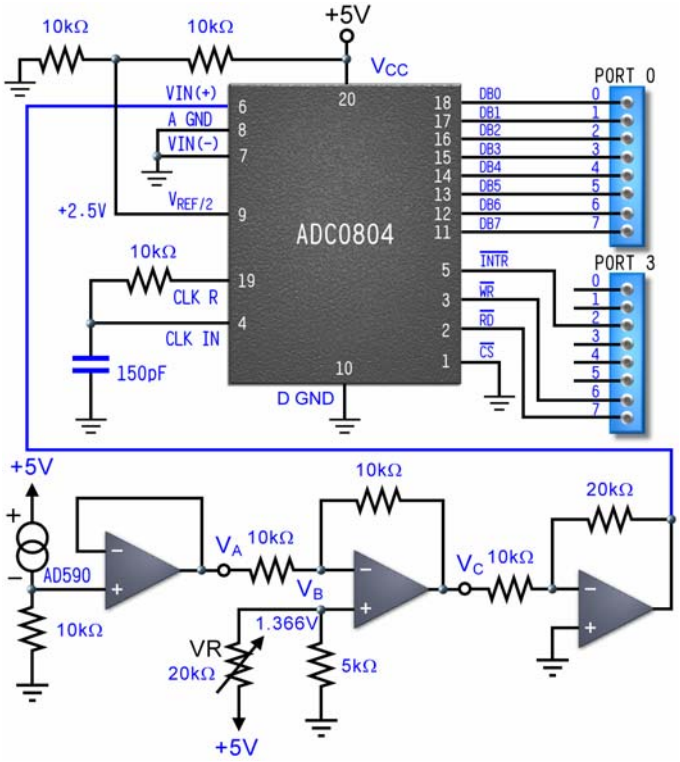


圖26 AD590 與 ADC0804 之介面電路

11-7 實例演練

ADC 與 DAC 之應用

在本單元裡提供四個範例，如下所示：

11-7-1 ADC 連續轉換實例演練

實驗要點

如圖 27 所示，ADC0804 連接到 8051 的 Port 2，而 ADC0804 之 $\overline{WR}$ 接腳與 $\overline{INTR}$ 接腳相連接，而 $\overline{RD}$ 與 $\overline{CS}$ 接腳都接地，如此就能使 ADC0804 不斷地進行轉換。轉換的結果也隨時放在匯流排上，8051 可從 Port 2 讀取之。ADC0804 的 $+V_{in}$ 接腳連接到 10kΩ可變電阻器，調整該可變電阻器，即可改變其電壓值，可從 0V 調整到 5V。$V_{REF/2}$ 接腳連接到一個分壓電路，以取得 2.5V 的參考電壓，如此將決定所要量測的電壓最大值不得超過 5V，即 $V_{REF/2}\times 2$。

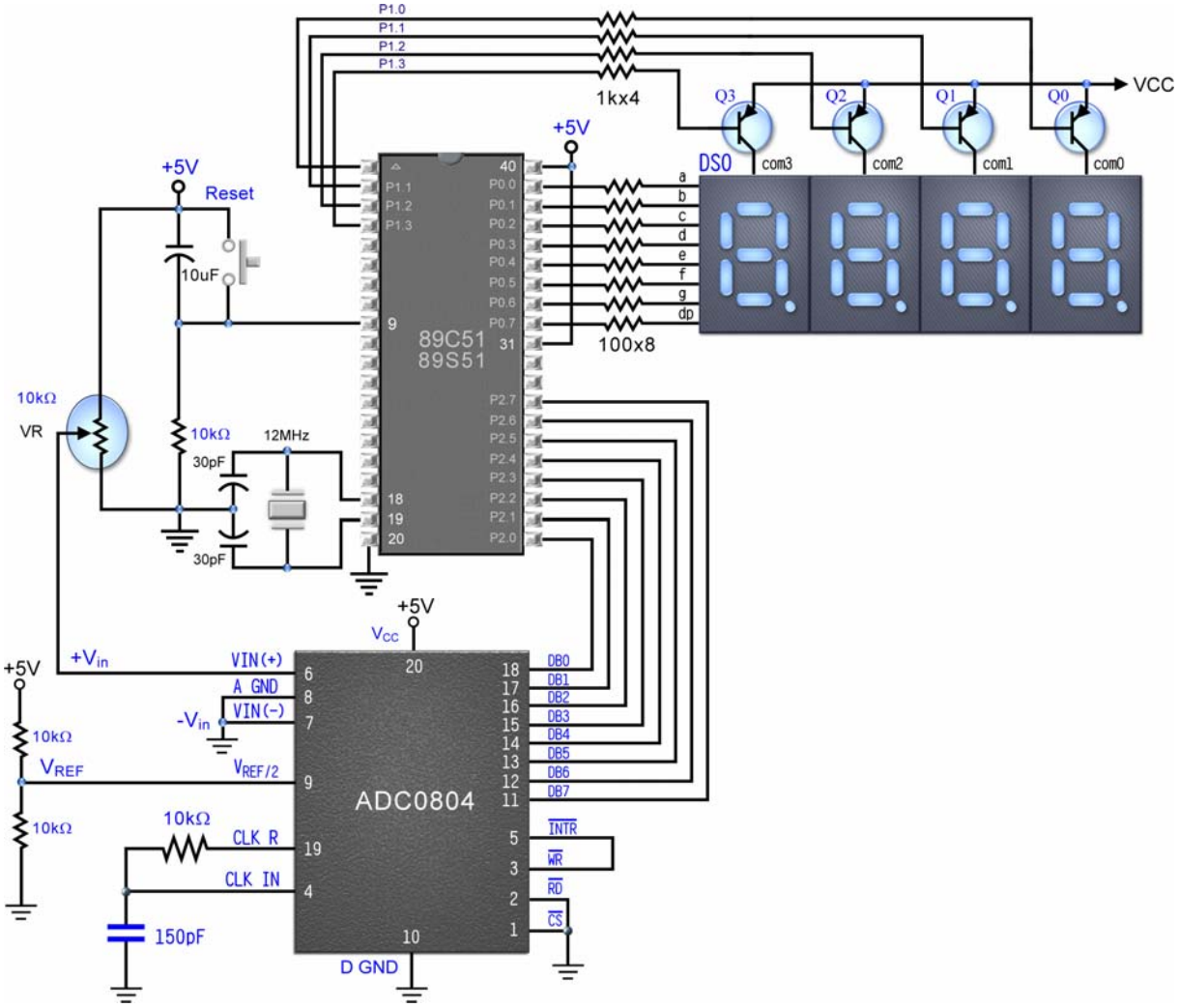


圖27　ADC 連續轉換電路圖

8051 除了讀取 ADC0804 的轉換結果外，也將其值做適當處理後，顯示在四數位七節顯示器模組，如此即爲一個簡易的數位電壓表。當 ADC0804 的 $+V_{in}$ 接腳連接到 5V 時，DB7－DB0 將輸出 11111111，即 0xff；若 $+V_{in}$ 接腳連接到 0V 時，DB7－DB0 將輸出 00000000，即 0x00，其解析度爲 $\frac{5-0}{255}=\frac{1}{51}\cong 0.0196\,\text{V}$。

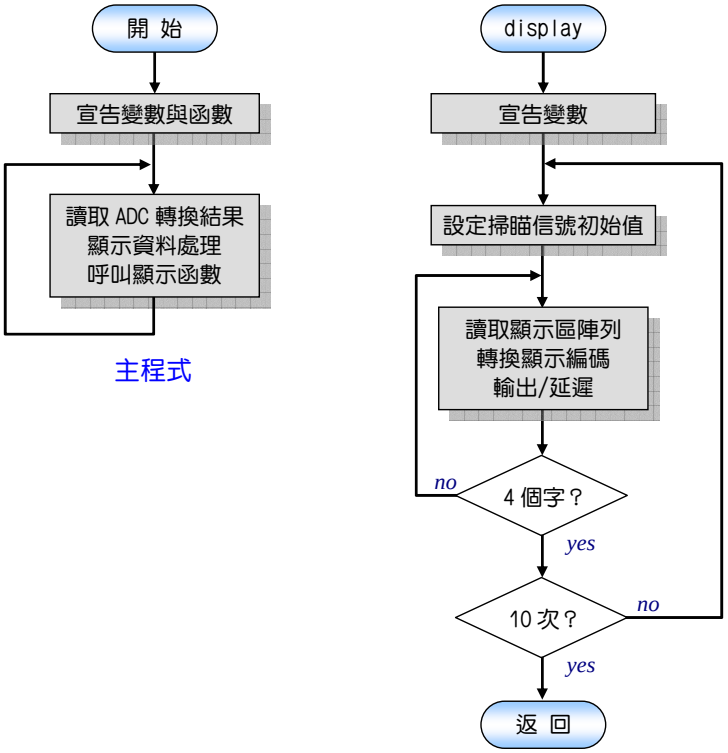
當量測到 5V 時，ADC0804 轉換的結果爲 11111111(即 255)，我們希望七節顯示器上顯示 5000，所以要把 255×19.6，才會得到 5000；在此將乘以 196，而非 19.6，如此將可得到 10 倍大的值，即 50000。將此值(results)除以 10000，所得的商數輸出到七節顯示器的千位

數；將 results 除以 1000，其商數再除以 10，所得的餘數輸出到七節顯示器的百位數；將 results 除以 100，其商數再除以 10，所得的餘數輸出到七節顯示器的十位數；將 results 除以 10，所得的餘數除以 10，其商數輸出到七節顯示器的個位數，如下：

```
results= adc*196;                    // 乘以 196 倍
disp[3]=results/10000;               // 取得千位數
disp[2]=(results/1000)%10;           // 取得百位數
disp[1]=(results/100)%10;            // 取得十位數
disp[0]=(results/10)%10;             // 取得個位數
```

流程圖與程式設計

流程圖與整個程式設計，如下：




```
/* ADC 連續轉換實例演練(ch11-1.c) */
#include <reg51.h>
/*宣告驅動信號陣列*/
char TAB[10]={  0xc0, 0xf9, 0xa4, 0xb0, 0x99,
                0x92, 0x83, 0xf8, 0x80, 0x98 };
//=========================================
unsigned char disp[4]={0, 0, 0, 0};      // 宣告顯示區陣列
void display(void);                      // 宣告顯示函數
```

```
void delay1ms(char);                    // 宣告延遲函數
main()                                  // 主程式
{    int results;                       // 宣告變數*/
     while(1)                           // while 開始*/
     {    P2=0xff;
          results= P2*196;              // 讀取 ADC0804 轉換結果乘以 196 倍
          disp[3]=results/10000;        // 取得千位數
          disp[2]=(results/1000)%10;    // 取得百位數
          disp[1]=(results/100)%10;     // 取得十位數
          disp[0]=(results/10)%10;      // 取得個位數
          display();                    // 呼叫顯示函數
     }                                  // while 結束
}                                       // 主程式結束
//====顯示函數====
void display(void)
{    char j,scan;                       // 宣告變數
     char i=10;                         // 掃瞄 10 次
     while (--i>=0)                     // while 迴圈開始
     {    scan=1;                       // 初始掃瞄信號
          for(j=0;j<4;j++)              // for 敘述開始
          {    P0=0xff;                 // 關 7 段顯示器
               P1=~scan;                // 輸出掃瞄信號
               P0=TAB[disp[j]];         // 轉換成驅動信號，並輸出到 P0
               delay1ms(4);             // 延遲 4ms
               scan<<=1;                // 下一個掃瞄信號
          }                             // 結束 for 敘述
     }                                  // 結束 while 敘述
}                                       // display 函數結束
//====延遲函數====
void delay1ms(char x)
{    int i,j;                           // 宣告變數
     for(i=0;i<x;i++)                   // 外迴圈
          for(j=0;j<120;j++);           // 內迴圈
}                                       // 延遲函數結束
```

ADC 連續轉換實例演練(ch11-1.c)

操作

1. 依功能需求與電路結構撰寫程式，然後將該程式編譯與連結，以產生*.HEX 檔。
2. 在 Keil C 裡進行軟體除錯/模擬，看看其功能是否正常？若有錯誤或非預期的狀況，則檢視原始程式，看看哪裡出問題？並將它記錄在實驗報告裡。
3. 若軟體除錯/模擬功能正常，再按圖 27 連接線路，使用實體模擬

器，載入新的程式(*.HEX)，以模擬該電路的動作。調整其中 10kΩ 可變電阻器，看看七節顯示器上有無變化？若沒有變化，請檢查線路有無錯誤？程式有無問題？並將它記錄在實驗報告裡。

4. 若七節顯示器上所指示的值會隨可變電阻器的調整而變動，再利用數字電表量測「$+V_{in}$」接腳對地電壓，並與七節顯示器上所指示的值比較，以找出誤差，並記錄在實驗報告裡。
5. 若實體模擬功能正常，請將程式燒錄到 89C51/89S51，再把該 89C51/89S51 放入實體電路，以取代剛才的實體模擬器，然後直接送電，看看是否正常？
6. 撰寫實驗報告。

11-7-2　ADC 交握式轉換實例演練之一

實驗要點

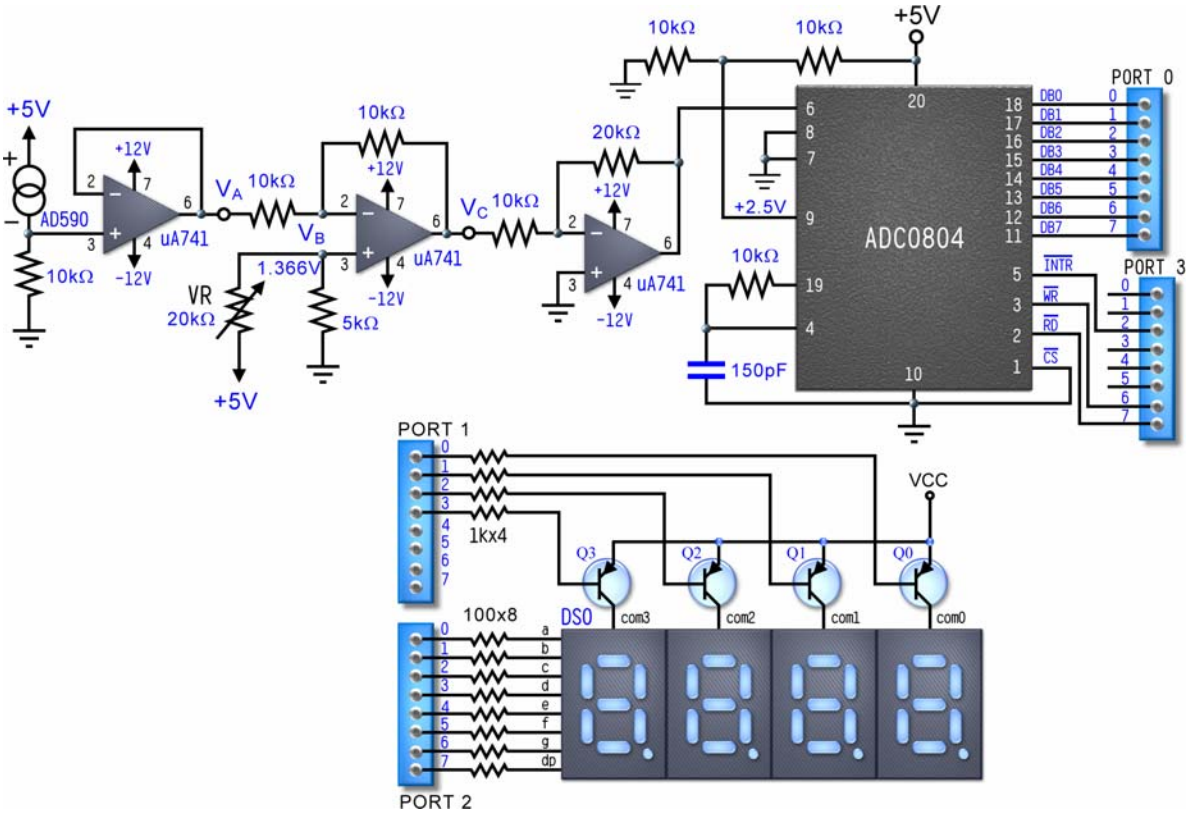
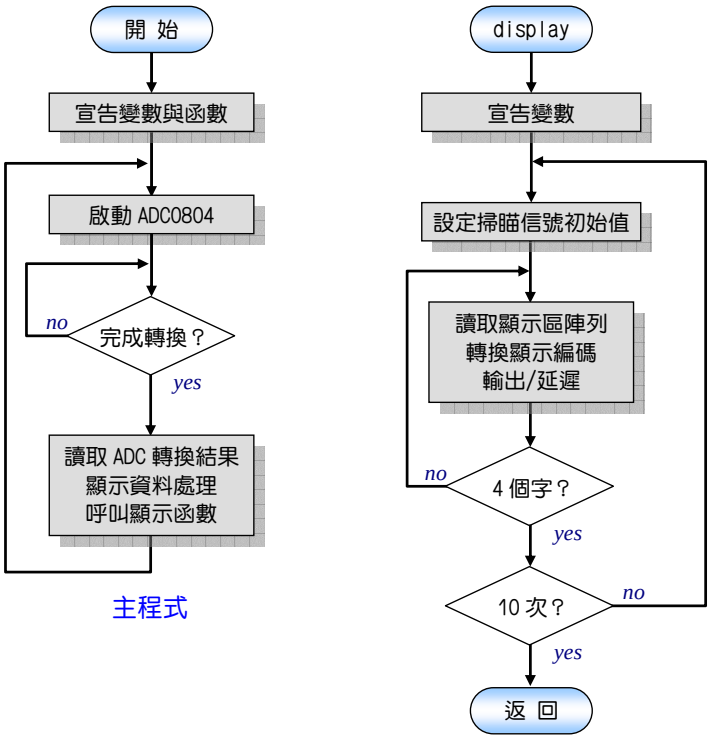


圖28　交握式傳輸電路圖

如上圖所示，ADC0804 連接到 8051 的 PORT 0，而 ADC0804 之 $\overline{WR}$ 接腳連接到 8051 之 $\overline{WR}$ 接腳、ADC0804 之 $\overline{RD}$ 接腳連接到 8051 之 $\overline{RD}$ 接腳，讓 ADC0804 變成是 8051 的「**外部記憶體**」，而 ADC0804 之 $\overline{INTR}$ 接腳連接到 8051 之 P3.2 接腳，可當成一般輸入埠，以垂詢方式偵測 ADC0804 是否完成轉換；也可以中斷方式處理。將轉換後的數位信號，輸出到 PORT 2 所連接的四數位七節顯示器，而其掃瞄信號是透過 Port 1 送出去的。

流程圖與程式設計



依功能需求與電路結構得知，當執行「**儲存外部記憶體**」時，8051 將透過 $\overline{WR}$ 接腳通知 ADC0804 開始進行轉換。緊接著，程式只要等待 ADC0804 由 $\overline{INTR}$ 接腳送來的完成轉換信號，即可進行外部資料的讀取。再將讀取到的資料，進行資料處理後，送到 PORT 1 即可顯示該數值。

```
/* ADC 交握式轉換實例演練之一(ch11-2.c) */
#include <reg51.h>
/*宣告驅動信號陣列*/
char TAB[10]={  0xc0, 0xf9, 0xa4, 0xb0, 0x99,
                0x92, 0x83, 0xf8, 0x80, 0x98 };
```

```
//=======================================
unsigned char disp[4]={0, 0, 0, 0};      // 宣告顯示區陣列
unsigned char xdata    adc;              // 宣告 xdata 記憶體形式變數
sbit    INTR=P3^2;                       // 宣告 INTR 變數
void display(void);                      // 宣告顯示函數
void delay1ms(char);                     // 宣告延遲函數
main()                                   // 主程式
{     int results;                       // 宣告變數
      while(1)                           // while 開始
      {     adc=0xff;                    // 執行寫入外部記憶體的操作
                                         // ADC0804 開始進行轉換
            if (INTR==1)                 // 判斷是否完成轉換
            {     results=adc;           // 讀取轉換結果
                  disp[3]=results/10000; // 取得千位數
                  disp[2]=(results/1000)%10;// 取得百位數
                  disp[1]=(results/100)%10; // 取得十位數
                  disp[0]=(results/10)%10;  // 取得個位數
            }
            display();                   // 呼叫顯示函數
      }                                  // while 結束
}                                        // 主程式結束
//====顯示函數====
void display(void)
{     char j,scan;                       // 宣告變數
      char i=10;                         // 掃瞄 10 次
      while (--i>=0)                     // if 迴圈開始
      {     scan=1;                      // 初始掃瞄信號
            for(j=0;j<4;j++)             // for 敘述開始
            {     P2=0xff;               // 關 7 段顯示器
                  P1=~scan;              // 輸出掃瞄信號
                  P2=TAB[disp[j]];       // 轉換成驅動信號，並輸出到 P2
                  delay1ms(4);           // 延遲 4ms
                  scan<<=1;              // 下一個掃瞄信號
            }                            // 結束 for 敘述
      }                                  // 結束 if 敘述
}                                        // display 函數結束
//====延遲函數====
void delay1ms(char x)
{     char i,j;                          // 宣告變數
      for(i=0;i<x;i++)                   // 外迴圈
            for(j=0;j<120;j++);          // 內迴圈
}                                        // 延遲函數結束
```

ADC 交握式轉換實例演練之一(ch11-2.c)

操作

1. 依功能需求與電路結構撰寫程式，然後將該程式編譯與連結，以

產生*.HEX 檔。

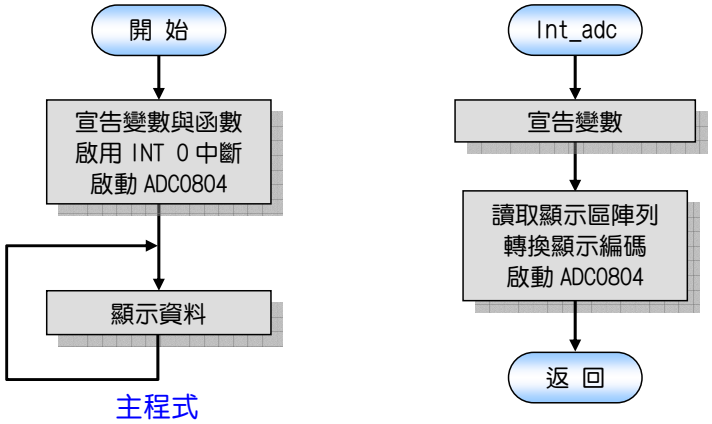
2. 在 Keil C 裡進行軟體除錯/模擬，看看其功能是否正常？若有錯誤或非預期的狀況，則檢視原始程式，看看哪裡出問題？並將它記錄在實驗報告裡。
3. 若軟體除錯/模擬功能正常，再按圖 28 連接線路，使用實體模擬器，載入新的程式(*.HEX)，以模擬該電路的動作。七節顯示器顯示爲何？若將加熱後的烙鐵靠近 AD590，七節顯示器有無反應？若無反應或非預期的狀況，則檢視線路的連接狀況，看看哪裡出問題？並將它記錄在實驗報告裡。
4. 若實體模擬功能正常，請將程式燒錄到 89C51/89S51，再把該 89C51/89S51 放入實體電路，以取代剛才的實體模擬器，然後直接送電，看看是否正常？
5. 撰寫實驗報告。

11-7-3 ADC 交握式轉換實例演練之二

實驗要點

本實驗之功能與 11-7-1 節完全相同，唯在本實驗裡，將採中斷方式。

流程圖與程式設計

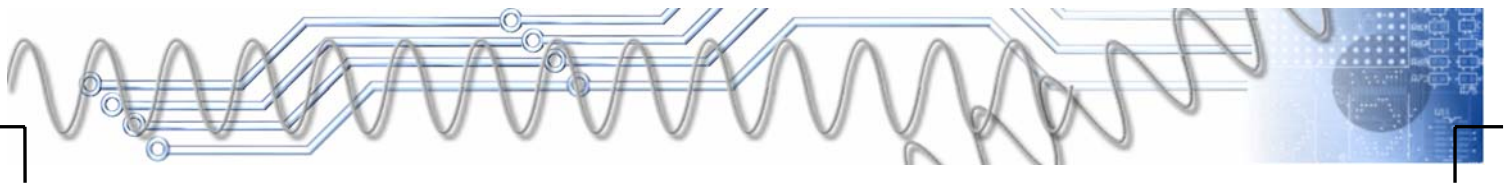


首先宣告變數與函數，再設定中斷，然後執行「**儲存外部記憶體**」時，以通知 ADC0804 開始進行轉換，而主程式就持續執行掃瞄顯示。在中斷副程式裡，首先讀取外部資料的讀取，將它進行顯示資

料處理後，然後放入顯示區陣列，重新啓動 ADC0804 即可。

```
/* ADC 交握式轉換實例演練之二(ch11-3.c) */
#include <reg51.h>
/*宣告驅動信號陣列*/
char TAB[10]={  0xc0, 0xf9, 0xa4, 0xb0, 0x99,
                0x92, 0x83, 0xf8, 0x80, 0x98 };
//=========================================
unsigned char disp[4]={0, 0, 0, 0};      // 宣告顯示區陣列
unsigned char xdata    adc;              // 宣告 xdata 記憶體形式變數
unsigned char results;                   // 宣告變數
void delay1ms(char);                     // 宣告延遲函數
//====主程式====
main()                                   // 主程式
{     char i,scan;                       // 宣告變數
      IE=0x81;                           // 啟用 INT0 中斷
      TCON=0x01;                         // 採負緣觸發
      adc=0xff;                          // 採啟動 ADC0804
      while(1)                           // while 開始
      {     scan=1;                      // 初始掃瞄信號
            for(i=0;i<4;i++)             // for 敘述開始
            {     P2=0xff;               // 關 7 段顯示器
                  P1=~scan;              // 輸出掃瞄信號
                  P2=TAB[disp[i]];       // 轉換成驅動信號，並輸出到 P2
                  delay1ms(4);           // 延遲 4ms
                  scan<<=1;              // 下一個掃瞄信號
            }                            // 結束 for 敘述
      }                                  // 結束 while 敘述
}
//====中斷函數====
void int_adc(void) interrupt 0
{     results= adc;                      // 讀取轉換結果
      disp[3]=results/10000;             // 取得千位數
      disp[2]=(results/1000)%10;         // 取得百位數
      disp[1]=(results/100)%10;          // 取得十位數
      disp[0]=(results/10)%10;           // 取得個位數
      adc=0xff;                          // 採啟動 ADC0804
}                                        // 中斷程式結束
//====延遲函數====
void delay1ms(char x)
{     char i,j;                          // 宣告變數
      for(i=0;i<x;i++)                   // 外迴圈
            for(j=0;j<120;j++);          // 內迴圈
}                                        // 延遲函數結束
```

ADC 交握式轉換實例演練之二(ch11-3.c)



操作

1. 依功能需求與電路結構撰寫程式，然後將該程式編譯與連結，以產生*.HEX 檔。
2. 在 Keil C 裡進行軟體除錯/模擬，看看其功能是否正常？若有錯誤或非預期的狀況，則檢視原始程式，看看哪裡出問題？並將它記錄在實驗報告裡。
3. 若軟體除錯/模擬功能正常，再按圖 28 連接線路，使用實體模擬器，載入新的程式(*.HEX)，以模擬該電路的動作。七節顯示器顯示爲何？若將加熱後的烙鐵靠進 AD590，七節顯示器有無反應？若無反應或非預期的狀況，則檢視線路的連接狀況，看看哪裡出問題？並將它記錄在實驗報告裡。
4. 若實體模擬功能正常，請將程式燒錄到 89C51/89S51，再把該 89C51/89S51 放入實體電路，以取代剛才的實體模擬器，然後直接送電，看看是否正常？
5. 撰寫實驗報告。

11-7-4 ADC 之溫控實例演練

實驗要點

在本單元裡所使用的電路與前面的單元類似，只是多出一個繼電器的控制，如圖 29 所示，由 P3.3 輸出到電晶體，以驅動繼電器。當溫度超過 35 度時，關閉繼電器、溫度低於 20 度時，啓動繼電器。

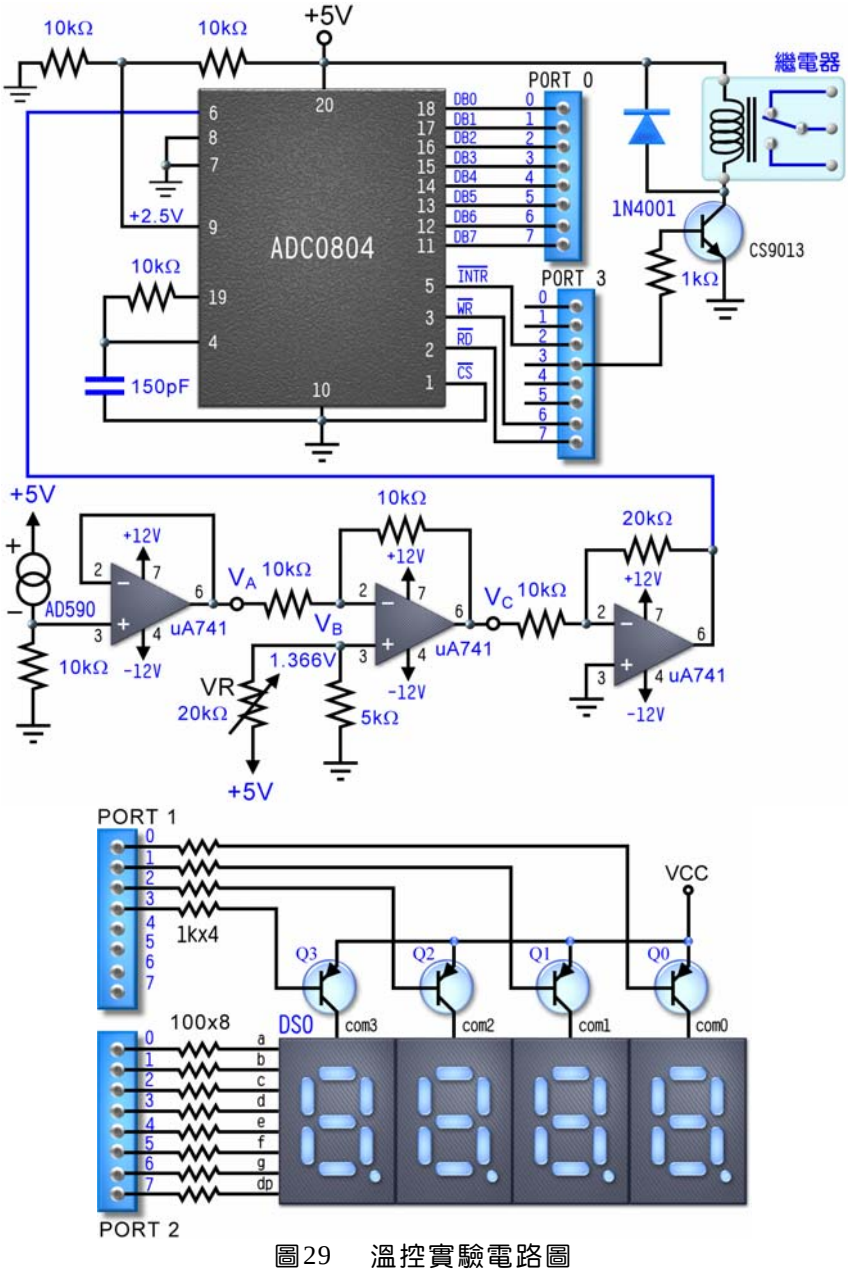
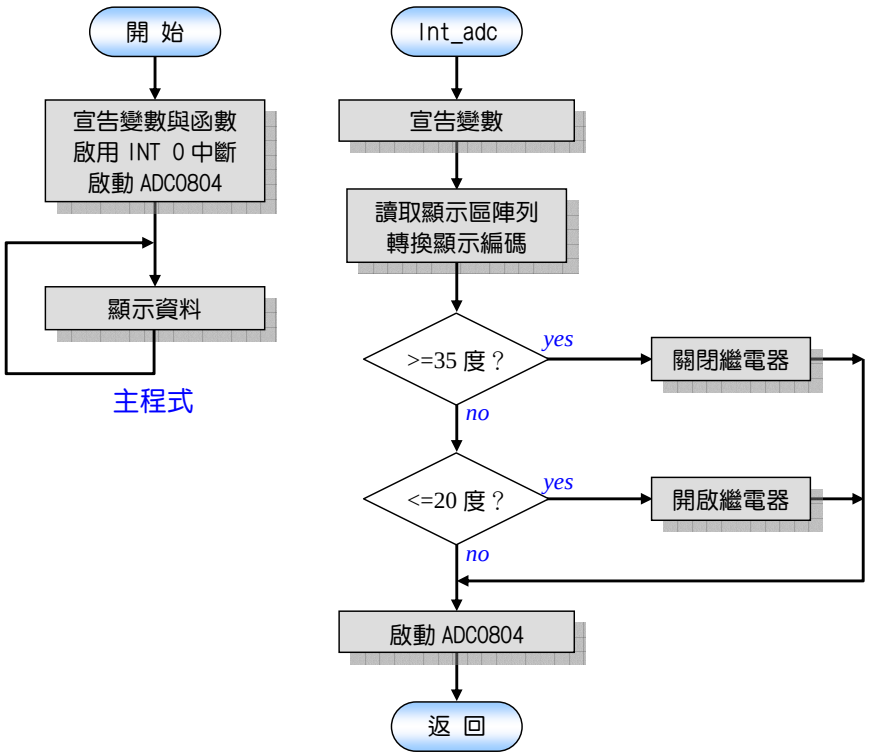


圖29　溫控實驗電路圖

流程圖與程式設計

依功能需求與電路結構得知，首先由 8051 送一個開始轉換的信號給 ADC0804，以啓動 ADC0804，再等待 ADC0804 傳回已完成轉換的信號，即可讀入溫度資料。若繼電器 On，則判斷溫度是否達到上限，若是則關閉繼電器，然後顯示溫度、若繼電器 On，而溫度未達上限，則直接顯示溫度。若繼電器 Off，則判斷溫度是否達到下限，若是

則開啓繼電器，然後顯示溫度、若繼電器 Off，而溫度未達下限，則直接顯示溫度。顯示溫度之後，再重新開始。




```
/* ADC 之溫控實驗(ch11-4.c) */
#include <reg51.h>
#define off    35
#define on     20
/*宣告驅動信號陣列*/
char TAB[10]={  0xc0, 0xf9, 0xa4, 0xb0, 0x99,
                0x92, 0x83, 0xf8, 0x80, 0x98 };
//=========================================
unsigned char disp[4]={0, 0, 0, 0};      // 宣告顯示區陣列
unsigned char xdata    adc;              // 宣告 xdata 記憶體形式變數
sbit relay = P3^3;                       // 宣告繼電器位置
unsigned char results;                   // 宣告變數
void delay1ms(char);                     // 宣告延遲函數
//====主程式====
main()                                   // 主程式
{     char i, scan;                      // 宣告變數
      IE=0x81;                           // 啟用 INT0 中斷
      TCON=0x01;                         // 採負緣觸發
      adc=0xff;                          // 採啟動 ADC0804
      relay=0;                           // 關閉繼電器
      while(1)                           // while 開始
      {     scan=1;                      // 初始掃瞄信號
```

```
            for(i=0;i<4;i++)                // for 敘述開始
            {   P2=0xff;                    // 關 7 段顯示器
                P1=~scan;                   // 輸出掃瞄信號
                P2=TAB[disp[i]];            // 轉換成驅動信號，並輸出到 P2
                delay1ms(4);                // 延遲 4ms
                scan<<=1;                   // 下一個掃瞄信號
            }                               // 結束 for 敘述
        }                                   // 結束 while 敘述
}
//====中斷函數====
void int_adc(void) interrupt 0
{   results= adc;                           // 讀取轉換結果
    disp[3]=results/10000;                  // 取得千位數
    disp[2]=(results/1000)%10;              // 取得百位數
    disp[1]=(results/100)%10;               // 取得十位數
    disp[0]=(results/100)%10;               // 取得個位數
    if (results >=off) relay=0;
    /*若溫度高於或等於 35 度，則關閉繼電器*/
    else if (results <=on) relay=1;
    /*若溫度低於或等於 20 度，則開啟繼電器*/
    adc=0xff;                               // 採啟動 ADC0804
}                                           // 中斷程式結束
//====延遲函數====
void delay1ms(char x)
{   char i,j;                               // 宣告變數
    for(i=0;i<x;i++)                        // 外迴圈
        for(j=0;j<120;j++);                 // 內迴圈
}                                           // 延遲函數結束
```

ADC 之溫控實驗(ch11-4.c)

操作

1. 依功能需求與電路結構撰寫程式，然後將該程式編譯與連結，以產生*.HEX 檔。
2. 在 Keil C 裡進行軟體除錯/模擬，看看其功能是否正常？若有錯誤或非預期的狀況，則檢視原始程式，看看哪裡出問題？並將它記錄在實驗報告裡。
3. 若軟體除錯/模擬功能正常，再按圖 29 連接線路，使用實體模擬器，載入新的程式(*.HEX)，以模擬該電路的動作。若有非預期的狀況，則檢視線路的連接狀況，看看哪裡出問題？並將它記錄在實驗報告裡。
4. 若實體模擬功能正常，請將程式燒錄到 89C51/89S51，再把該

89C51/89S51 放入實體電路，以取代剛才的實體模擬器，然後直接送電，看看是否正常？

5. 撰寫實驗報告。

11-7-5 DAC 實例演練

實驗要點

如圖 30 所示，指撥開關的狀態由 P0 輸入，而開關狀態，經 P2 連接到 DAC-08，將它轉換成類比電壓。請以數字電表量測電路中的 V_O，觀察其輸出的類比電壓，是否符合指撥開關的狀態？

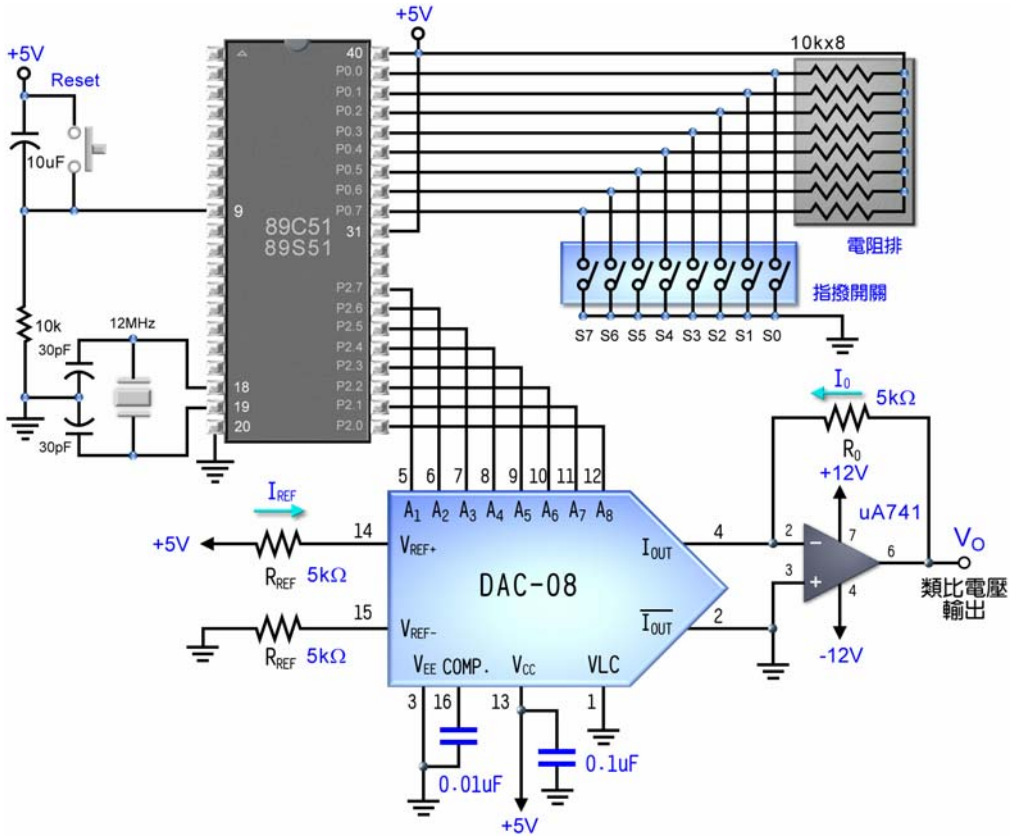
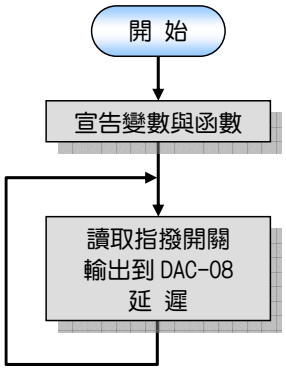


圖30 DAC 實驗電路圖

流程圖與程式設計

依功能需求與電路結構得知，首先將 P0 規劃成輸入埠，再讀取指撥開關的狀態，然後將它輸出到 P2，讓 DAC-08 將它轉換成類比電壓。

經過一段時間延遲(任意時間)後，再重複進行上述操作。



主程式

```
/* DAC 實驗(ch11-5.c) */
#include <reg51.h>
void delay1ms(char);                    // 宣告延遲函數
main()                                  // 主程式
{     P0=0xff;                          // 規劃輸入
      while(1)                          // while 開始
      {                                 // 讀取指撥開關
           P2=~P0;                      // 輸出到 DAC
           delay1ms(1);                 // 呼叫延遲函數
      }                                 // while 結束
}                                       // 主程式結束
//====延遲函數====
void delay1ms(char x)
{     char i,j;                         // 宣告變數
      for(i=0;i<x;j++)                  // 外迴圈
           for(j=0;j<120;j++);          // 內迴圈
}                                       // 延遲函數結束
```

DAC 實驗(ch11-5.c)

操作

1. 依功能需求與電路結構撰寫程式，然後將該程式編譯與連結，以產生*.HEX 檔。
2. 在 Keil C 裡進行軟體除錯/模擬，看看其功能是否正常？若有錯誤或非預期的狀況，則檢視原始程式，看看哪裡出問題？並將它記錄在實驗報告裡。
3. 若軟體除錯/模擬功能正常，再按圖 30 連接線路，使用實體模擬器，載入新的程式(*.HEX)，以模擬該電路的動作。若有非預期的狀況，則檢視線路的連接狀況，看看哪裡出問題？並將它記錄

在實驗報告裡。

4. 若實體模擬功能正常，請將程式燒錄到 89C51/89S51，再把該 89C51/89S51 放入實體電路，以取代剛才的實體模擬器，然後直接送電，再以數位電表量測 V_O 的電壓，記錄每個指撥開關狀態下的 V_O，並記錄在實驗報告裡。
5. 撰寫實驗報告。

11-8 即時練習 @ ADC 與 DAC 之應用

在本章裡探討 ADC 的原理與實用 IC、DAC 的原理與實用 IC，還有簡易的溫控 IC，可說是 8051 應用系統中，相當重要的一部分。在此請試著回答下列問題，以確認對於此部分的認識程度。

選擇題

(　)1. 下列哪種 AD 轉換器的轉換速度比較快？
(A) 雙斜率型 AD 轉換器　(B) 比較型 AD 轉換器　(C) 連續計數式 AD 轉換器　(D) 逐漸接近式 AD 轉換器。

(　)2. 下列哪種 AD 轉換器的精密度比較高？ (A) 雙斜率型 AD 轉換器　(B) 比較型 AD 轉換器　(C) 連續計數式 AD 轉換器　(D) 逐漸接近式 AD 轉換器。

(　)3. ADC0804 具有什麼功能？ (A) 8 位元類比-數位轉換器　(B) 11 位元類比-數位轉換器　(C) 8 位元數位-類比轉換器　(D) 11 位元數位-類比轉換器。

(　)4. 若要啓動 ADC0804，使之進行轉換，應如何處理？ (A) 加高態信號到 $\overline{CS}$ 接腳　(B) 加高態信號到 $\overline{WR}$ 接腳　(C) 加低態信號到 $\overline{CS}$ 接腳　(D) 加低態信號到 $\overline{WR}$ 接腳。

(　)5. 當 ADC0804 完成轉換後，將會如何？ (A) $\overline{CS}$ 接腳轉爲低態 (B) $\overline{CS}$ 接腳轉爲高態　(C) $\overline{INTR}$ 接腳轉爲低態　(D) $\overline{INTR}$ 接腳轉爲高態　。

( )6. 下列哪個 IC 具有溫度感測功能？ (A) DAC-08 (B) AD590 (C) uA741 (D) NE555 。

( )7. 下列哪種數爲信號轉換類比信號的方式，比較實際？ (A) R-2R 電阻網路 (B) 加權電阻網路 (C) 雙 Y 型電阻網路 (D) 三角型電阻網路 。

( )8. 當溫度每上升 1℃時，AD590 會有什麼變化？ (A) 電壓上升 1 毫伏 (B) 電壓下降 1 毫伏 (C) 電流上升 1 微安 (D) 電流下降 1 微安。

( )9. 若要讓 ADC0804 進行連續轉換，應如何連接？ (A) $\overline{CS}$ 接腳與 $\overline{INTR}$ 接腳連接、$\overline{WR}$ 接腳與 $\overline{RD}$ 接腳接地 (B) $\overline{CS}$ 接腳與 $\overline{WR}$ 接腳連接、$\overline{INTR}$ 接腳與 $\overline{RD}$ 接腳接地 (C) $\overline{WR}$ 接腳與 $\overline{INTR}$ 接腳連接、$\overline{CS}$ 接腳與 $\overline{RD}$ 接腳接地 (D) $\overline{RD}$ 接腳與 $\overline{INTR}$ 接腳連接、$\overline{WR}$ 接腳與 $\overline{CS}$ 接腳接地。

( )10. 若要 ADC080 與 8051 採交握式信號傳輸，則應如何？ (A) 8051 將 ADC0804 視爲外部記憶體 (B) 8051 透過 Port 0 連接 ADC0804 之資料匯流排 (C) 8051 之 $\overline{RD}$ 接腳與 ADC0804 之 $\overline{RD}$ 接腳相連接、8051 之 $\overline{WR}$ 接腳與 ADC0804 之 $\overline{WR}$ 接腳相連接 (D) 以上皆是 。

問答題

1. 試述類比信號與數位信號的特性？
2. 簡述並列式 ADC 的原理及其特性？
3. 簡述逐漸接近式 ADC 的原理及其特性？
4. 簡述連續計數式 ADC 的原理及其特性？
5. 簡述雙斜率式 ADC 的原理及其特性？
6. 試述 ADC0804 的特性？
7. 試述 ADC0804 所能接受的時鐘脈波頻率範圍爲何？如何利用其內部振盪電路產生時鐘脈波？

8. 若要將 ADC0804 視爲外部記憶體，在 C 語言程式裡，應如何宣告？而 ADC0804 的 $\overline{WR}$ 接腳、 $\overline{RD}$ 接腳與 $\overline{INTR}$ 接腳應如何連接？
9. 試述 AD590 的用途與特性？
10. 試設計一個 AD590 與 ADC0804 之介面，使溫度變化 1℃，在 ADC0804 的輸出數位信號就增減 1？