簡易型微電腦實體模擬系統
EPROM 模擬器及 8749 單晶片模擬器之研製

李 隆 財

摘 要

EPROM 模擬器是一個利用 RS232 介面與 IBM PC個人電腦連線使用的簡易型微電腦實體模擬系統，可模擬 2716、2732、2764、27128、27256 等五種 EPROM，配合 IBM PC各種不同 CPU 的 Cross Assembler,如 Z80, 6502, 8751, 8748, 68000, 8086 等 Cross Assembler 或者 IBM PC 的 MASM, TASM 等 Assembler，在 IBM PC上編輯組譯成機器碼，再轉換成 Intel HEX 格式的機器碼，經由 RS232 介面 DOWN LOAD 到 EPROM 模擬器的模擬 RAM ，再由排線連接到待模擬發展的目標系統的 EPROM 插座上，來做即時實體模擬，正確無誤後，再將程式燒到 EPROM ，插到目標系統的 EPROM 插座上，完成微電腦系統的設計開發工作。

8749單晶片模擬器是一個與 EPROM 模擬器連接使用的單晶片實體模擬系統，可完全模擬 8749單晶片微電腦的所有軟硬體功能，有正常執行和單步執行兩種工作模式。如上所述，單獨使用 EPROM 模擬器，即 8749單晶片的模擬將僅限於外接 ROM 的模擬，如此將減弱單晶片微電腦的 I/O 功能，所以針對不同類型的單晶片微電腦的模擬，必須設計個別的單晶片模擬器，與 EPROM 模擬器配合使用，以構成完整的單晶片微電腦的實體模擬系統。

目 錄

一、緒 言

由於微電腦的快速發展，產業升級以及工業生產自動化的需求，因此微電腦的應用非常普遍；而在微電腦的應用設計中，硬體電路設計所需的時間約占整個系統設計的百分之二十左右，而控制軟體大部分以組合語言設計，需要百分之八十以上的時間來設計、測試、修改軟體程式。因此在微電腦控制系統的設計上，組合語言程式的設計非常重要，而且程式寫好了以後必須做實體模擬，以測試系統的功能是否完全符合需求，最後再把設計好的程式燒到EPROM ，插到系統上，才完成微電腦控制系統的設計工作。

微電腦系統的發展工具有數萬到十幾萬元全友公司出品的MICE系列產品，到其它廠牌的一、二萬元的ICE 系列產品，以及數千元的介面卡型式的簡易模擬系統，不過介面卡型式的模擬器使用起來較不方便，功能也不完整，雖然MICE及ICE 的功能都很強，但是都屬於特定對象，也就是說Z80的MICE-II只能做Z80 CPU 產品的開發，而無法做其它CPU 的模擬。由於近幾年來專科以上學校的教學，特別強調實作的觀念，每一位學生都必須做專題製作，要做專題就必須有發展工具，而這種工具最好是萬用的，可適用於不同CPU的模擬、測試，更重要的是價格要便宜，可以人手一套，攜帶方便；只要有PC，再配合此套發展工具，隨時隨地都可做微電腦控制電路的模擬、測試工作。

針對上述的需求，因此設計出這一套簡易型微電腦實體模擬系統--EPROM模擬器及8749單晶片模擬器，EPROM 模擬器不受CPU 的限制，只要微電腦電路有EPROM 即可應用，8749單晶片模擬器是針對8748、8749單晶片而設計的。這套系統的製作成本低於新台幣參仟元，可說是多用途，低價位的簡易型微電腦發展系統，不僅適合微電腦實習教學，專題製作，更可做爲一般個人或公司做微電腦工業控制設計的發展工具。

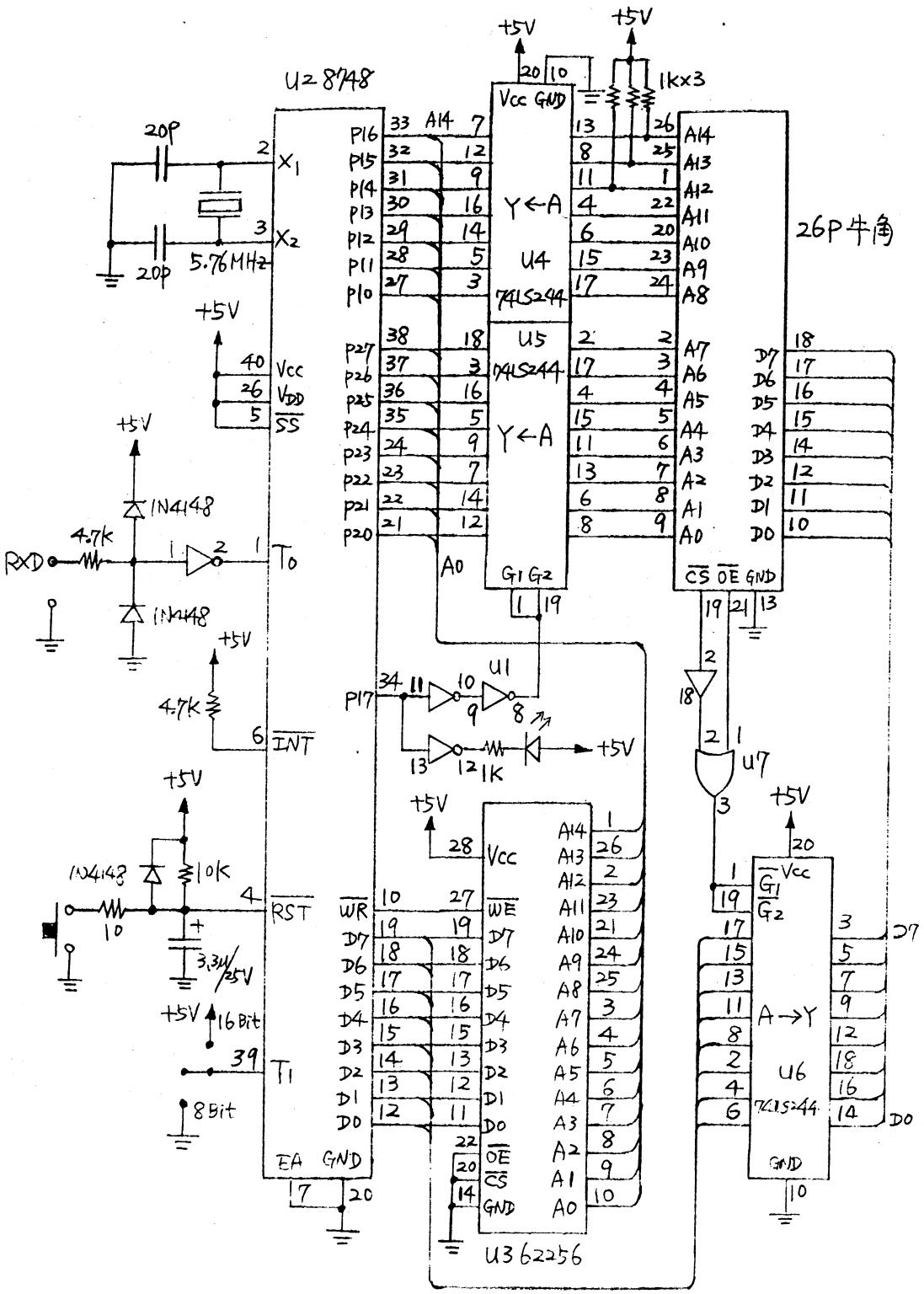


圖 1　EPROM 模擬器電路圖

二、硬體電路工作原理

㈠EPROM 模擬器

硬體電路如圖 1 所示，其工作原理分述如下：

1.石英振盪：必須採用5.76 MHz的石英振盪器，因爲8748單晶片的機器週期爲石英振盪頻率除以15 (5.76 MHz÷15=384 KHz)，所以機器週期爲384 KHz ，如果IBM PC RS232介面的傳輸率(Baud Rate) 爲9600 BPS，則38400÷9600=40 個機器週期，則串列傳輸資料一個位元的時間相當於8748的40個機器週期。因爲8748的 To腳接IBM PC RS232介面的資料傳輸接腳 TXD ，而8748 接收IBM PC RS232 介面傳送過來的串列資料，是以軟體方式處理，也就是在接收到起始位元的低電位後，每隔40個機器週期接收一個位元的串列資料。

2.RS232 介面位階轉換電路：由IBM PC RS232 TXD送出的資料，其電壓準位爲±12V，但是To接腳的輸入準位必須爲TTL準位，因此利用兩個1N4148檢波二極體和一個4.7K 歐姆的電阻，構成箝位電路(Clamping Circuit) 限制其電壓爲－0.6V到+5.7V 。因爲RS232 的輸出是經由MC1488 RS232位階轉換IC反向輸出，因此接收時必須再加上74LS04反向輸出。

3.EPROM 模擬器與IBM PC RS232 埠DB25接頭的接線如圖 2 所示。

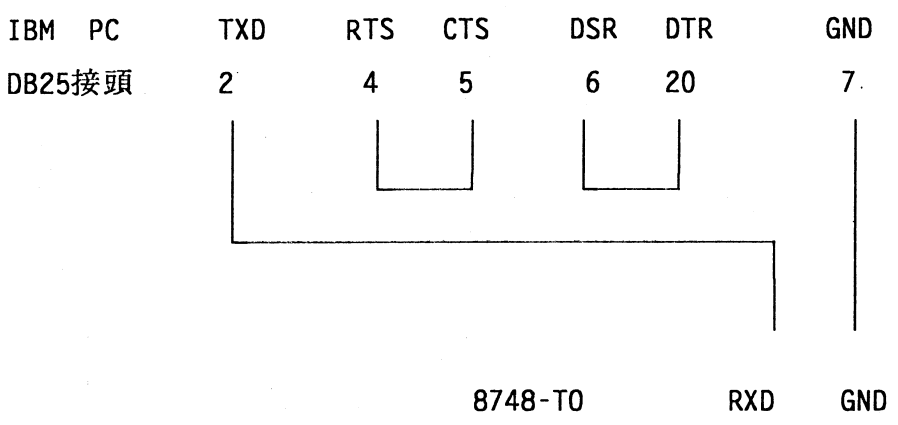
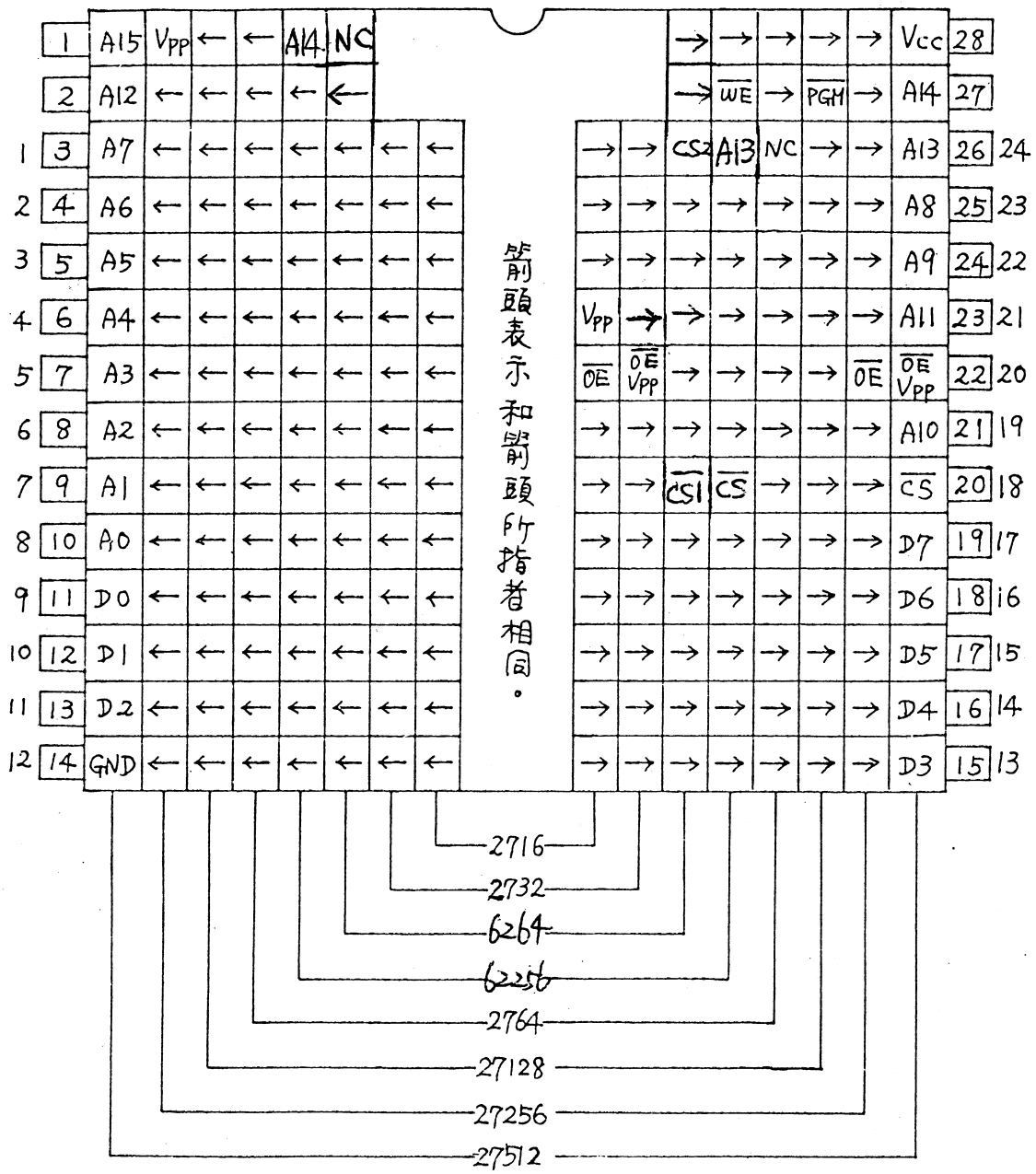


圖 2　RS232 DB25 接頭接線圖



註：外側之數字是 24 支腳 IC 之接腳編號

圖 3　EPROM／SRAM 接腳對照圖

4.T1接腳做爲16位元EPROM 模擬器之切換控制，T1接+5V 表示16位元，T1接地表示8 位元。如果是32位元，則可再利用INT 接腳加以控制選擇。

5.P17 的功能爲控制送到模擬RAM的位址，由8748 P20～P27，P10～P16所定址，還是由目標系統的EPROM 經由排線傳送過來的位址信號所定址。當P17=0 ，模擬RAM 由排線所連接的目標系統的EPROM 所定址，代表EPROM 模擬器是處於ON LINE 狀態；當P17=1，由8748控制將資料寫入P20～P27及P10～P16 所定址的模擬PAM 內，此時代表EPROM 模擬器處於OFF LINE狀態。當P17 要由 1 變爲 0 時，必須先將P20～P27，P10～P16之位階變成高電位，才不致於影響由目標系統所送出位址信號的位階變化。兩個74LS244 位址緩衝器由P17 控制，並配合8748 P1，P2 的I/O特性，不用二選一多工器(74157)，就可隔離兩種位址信號，不致發生衝突。

6.8748的DATA BUS與模擬RAM 的DATA BUS連接，$\overline{WR}$信號接到模擬RAM 的$\overline{WE}$輸入，當8748執行到MOVX @R0，A時，8748 $\overline{WR}$ 信號會產生一個$\overline{WR}$的負脈波，將累積器A 的資料送到DATA BUS，寫入P20～P27及P10～P16所定址的模擬RAM 內。

7.U6 74LS244當做資料緩衝器，當待模擬目標系統執行到模擬器所連接的EPROM 的程式時，此時EPROM 之$\overline{CS}$及$\overline{OE}$同時變成低電位，而將其所定址之模擬RAM 的資料由74LS244 緩衝器送出。

8.EPROM 模擬器爲了達到全自動智慧型，也就是在全部操作過程中，不必撥動任何開關，完全由軟體控制，參考圖 3 EPROM/SRAM接腳對照圖可知：

(1)2716的21腳爲Vpp ，應接+5V ，而其它4 種EPROM 均爲A11 。

(2) 5 種EPROM 的$\overline{CS}$（18或20腳），$\overline{OE}$（20或22腳）均相同。

(3)這些EPROM 在當作ROM 使用時，Vpp及$\overline{PGM}$均接高電位，還有一些是空腳，故將28支腳的2，26，27 三支接腳各接一個1K提升電阻到+5V ，也就是將27256 的A12，A13，A14 均接1K提升電阻至+5V 高電位。

(4) 5 種EPROM 模擬時，所使用的模擬RAM 區如圖 4 所示。

9.EPROM模擬器的零件配置圖如圖 5 所示，其中 U3的模擬 RAM，如果爲6264，則此EPROM 模擬器可模擬2716、2732、2764等三種EPROM ，如果改用62256則可模擬2716、2732、2764、27128、27256等5 種EPROM，U3模擬 RAM

的IC 座，可插6264或62256 ，不必做任何跳線變更，參考圖3 EPROM/SRAM接腳對照圖。

EPROM 編 號	模擬 RAM 位址
27256	0000
27128	4000
2764	6000
2732	7000
2716	7800

圖 4　EPROM 與模擬 RAM 區對照表

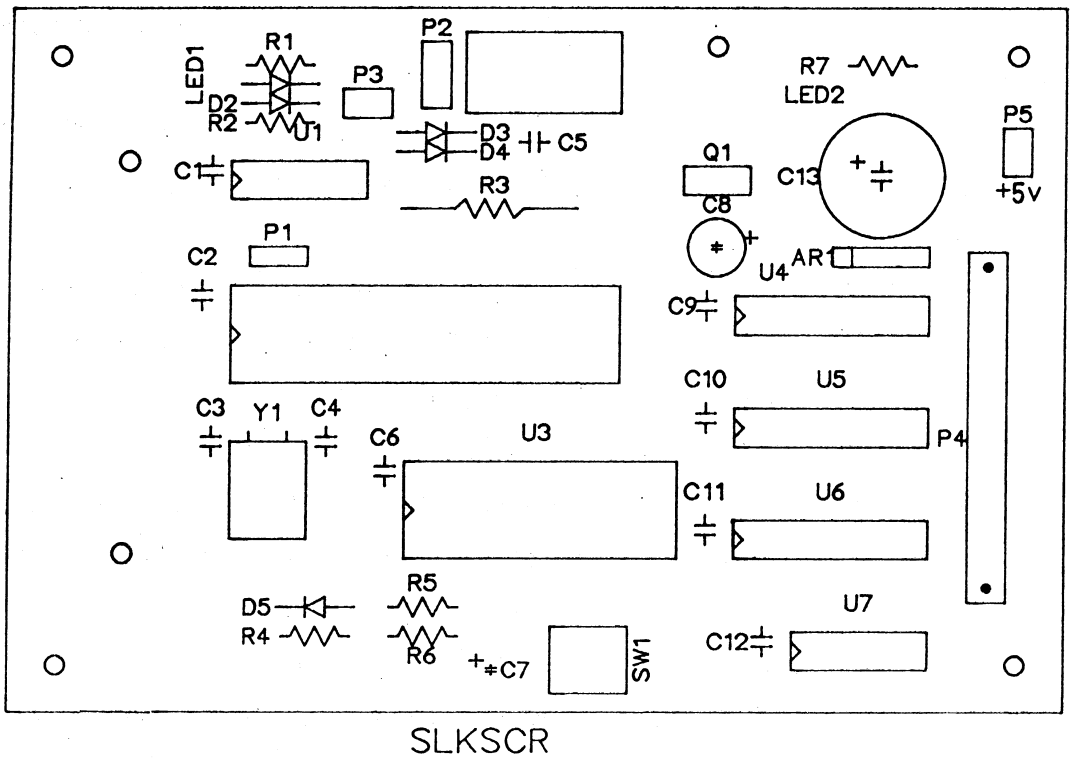


圖 5　EPROM 模擬器零件配置圖

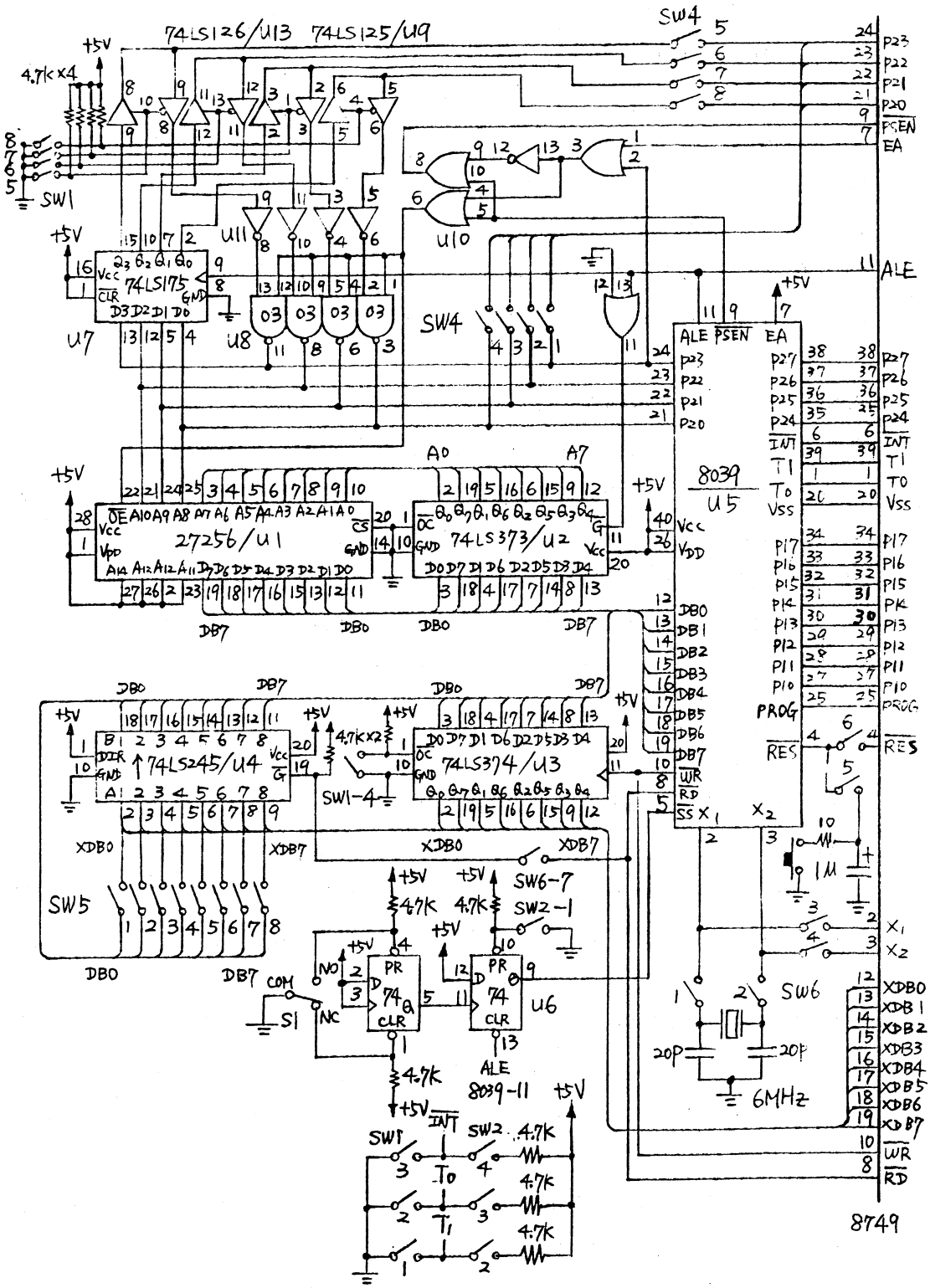


圖 6　8749 單晶體模擬器電路圖

㈡8749單晶片模擬器

8749單晶片模擬器的硬體電路如圖6 所示，主要是依據Intel 公司1978年版MCS-48 Micro computer User's Manual 中的8049 Emulator 電路改良而成，原設計圖請參考附錄 1 。經大幅度改良的8749單晶片模擬器的電路工作原理分別敍述如下：

1.CLOCK 電路：由DIP Switch SW6控制。SW6-12 ON, SW6-34 OFF 時，選擇模擬器內部6MHz之振盪頻率。SW6-12 OFF, SW6-34 ON 時，選擇外部目標板上的CLOCK 。

2.RESET 電路：SW6-56 ON 以模擬器之RESET 鍵控制RESET 。SW6-5 OFF, SW6-6 ON 時，以外部目標系統上的RESET 鍵控制RESET 。

3.單步執行電路：當SW2-1 ON時，程式正常執行；SW2-1 OFF 時，由微動開關S1控制做單步執行，每按一次微動開關並放鬆，就執行一個指令。

4.T1, TO, INT 控制：當SW2-2 ON; SW1-1 ON, 則T1=0, SW1-1 OFF,則T1=1 。當SW2-2 及 SW1-1 OFF時，T1由外界目標系統控制。同理To由SW2-3 及 SW1-2 控制，INT 由SW2-4及SW1-3所控制。

5.P20, P21, P22, P23的控制：

(1)SW4-1234 ON, SW4-5678 OFF 時，可控制外接8243做I/O 擴展。

(2)SW4-1234 OFF, SW4-5678 ON 時，可控制P20, P21, P22, P23當I/O 使用，而由SW1-5678控制其輸入輸出方向，當SW1-8 ON時控制P20 爲輸入，SW1-8 OFF 時，控制P20 爲輸出，同理SW1-7 控制P21 的輸入輸出方向，SW1-6 控制P22 的輸入輸出方向，SW1-5 控制P23 的輸入輸出方向。

(3)當SW1-5678全部OFF 時，74LS125 的四組輸出均爲三態電路的高阻抗輸出，經過74LS04反向變成低電位輸出，因此74LS03四組開集極式輸出的電晶體均爲截止狀態，不會影響8039 P20, P21, P22, P23 的輸出電位。

(4)當8039讀取外接ROM 之資料時，$\overline{PSEN}$變爲低電位，同時配合目標板之$\overline{EA}$腳接地，P23=0, 74LS32 第 6 腳輸出爲低電位，控制74SL03 開集極式輸出電晶體爲截止狀態，以及控制外接ROM 的$\overline{OE}$腳爲低電位。

(5)當目標板上又附加一個2K的擴充記憶體(EPROM) 時，也就是記憶體位址

的800H～FFFH時，此時P23=1 ，配合8039 $\overline{PSEN}$ 爲低電位，則74LS32的第8 腳輸出爲低電位，接到目標板另外再附加的2K擴展ROM 的$\overline{OE}$腳，來控制讀取此2K Byte 程式記憶體的資料。

6.模擬器上2K擴展ROM 電路：此EPROM 接腳是以27256 接腳爲準連接而成，事實上只用到其中2K Byte 而已。爲了與EPROM 模擬器配合使用，因此將27256的A14，A13，A12，A11 四支接腳均接到高電位。當利用EPROM 模擬器模擬完成後，也可以燒一個EPROM 插到這個27256 的IC座上，不過資料要燒在EPROM 的最後2K Byte 的位址內，才能正常執行。

7.DATA BUS的處理電路：8749單晶片的 DATA BUS 可當作資料匯流排，也可當作輸入或輸出來使用。其中74LS374 配合8039的$\overline{WR}$信號控制DATA BUS做輸出，74LS245配合8039的$\overline{RD}$信號控制DATA BUS做輸入。當SW5全部 OFF，SW6-7 ON 時，由 SW1-4 控制 DATA BUS的輸入輸出方向，SW1-4 ON 時，DATA BUS 指定爲輸出用途；SW1-4 OFF時，DATA BUS規劃爲輸入。當SW5全部ON，SW6-7 OFF及SW1-4 OFF時，DATA BUS就是普通的資料匯流排，可外接擴展RAM 或其它 INTEL 系列的介面IC，如8255，8253，8155，8279等。

8.爲了實習方便起見，電源供應器有兩組+5V 的穩壓輸出，一組供應單晶片模擬器使用，一組供給目標系統使用，大約400 mA左右，如果目標系統的消耗+5V 電源的電流大於400 mA，則建議另外使用外加電源來供應目標系統使用，以免負載過大，而使得模擬器的電源不穩定，影響到實體模擬的效果。

9.8749單晶片模擬器的零件配置圖如圖7 所示。

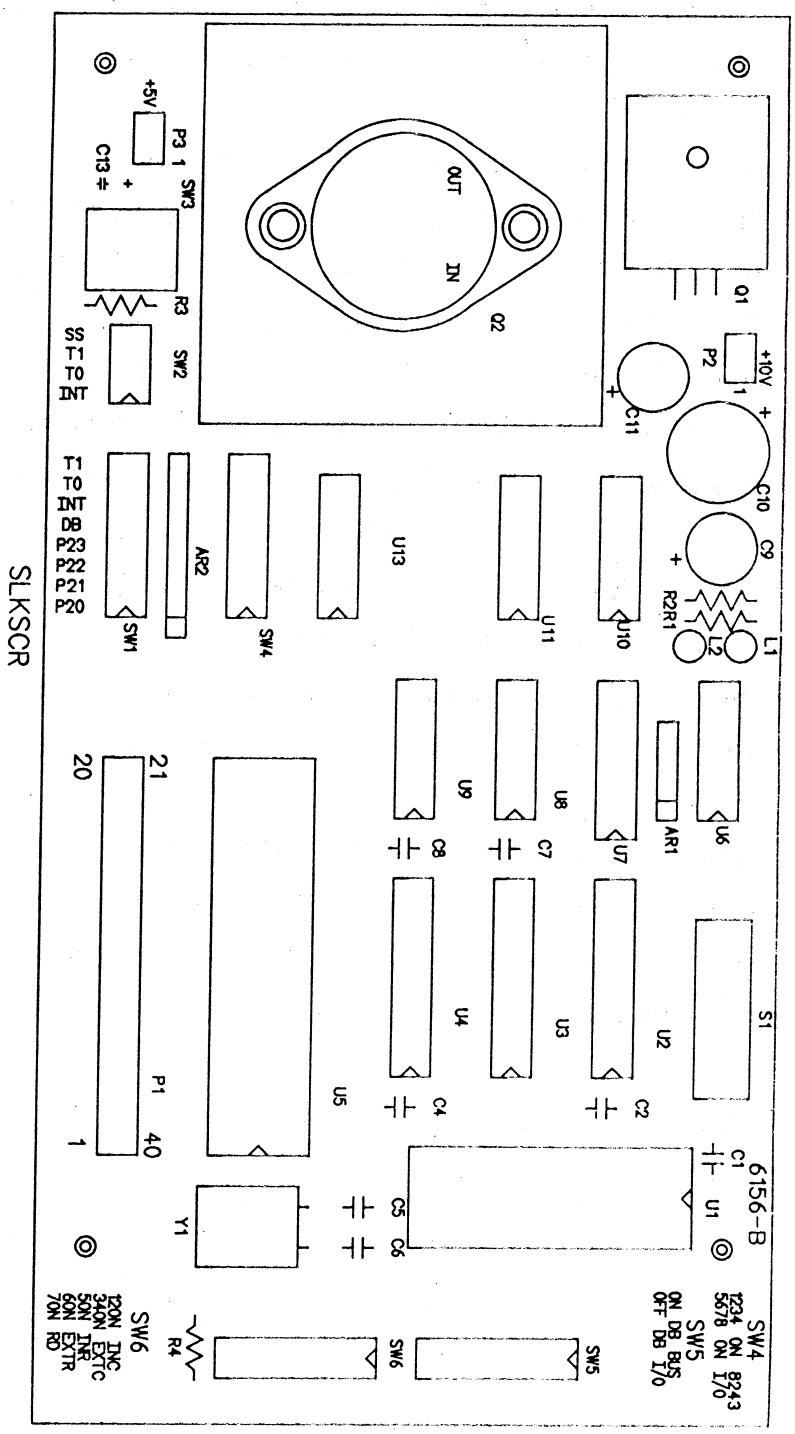


圖 7　8749 單晶片模擬器零件配置圖

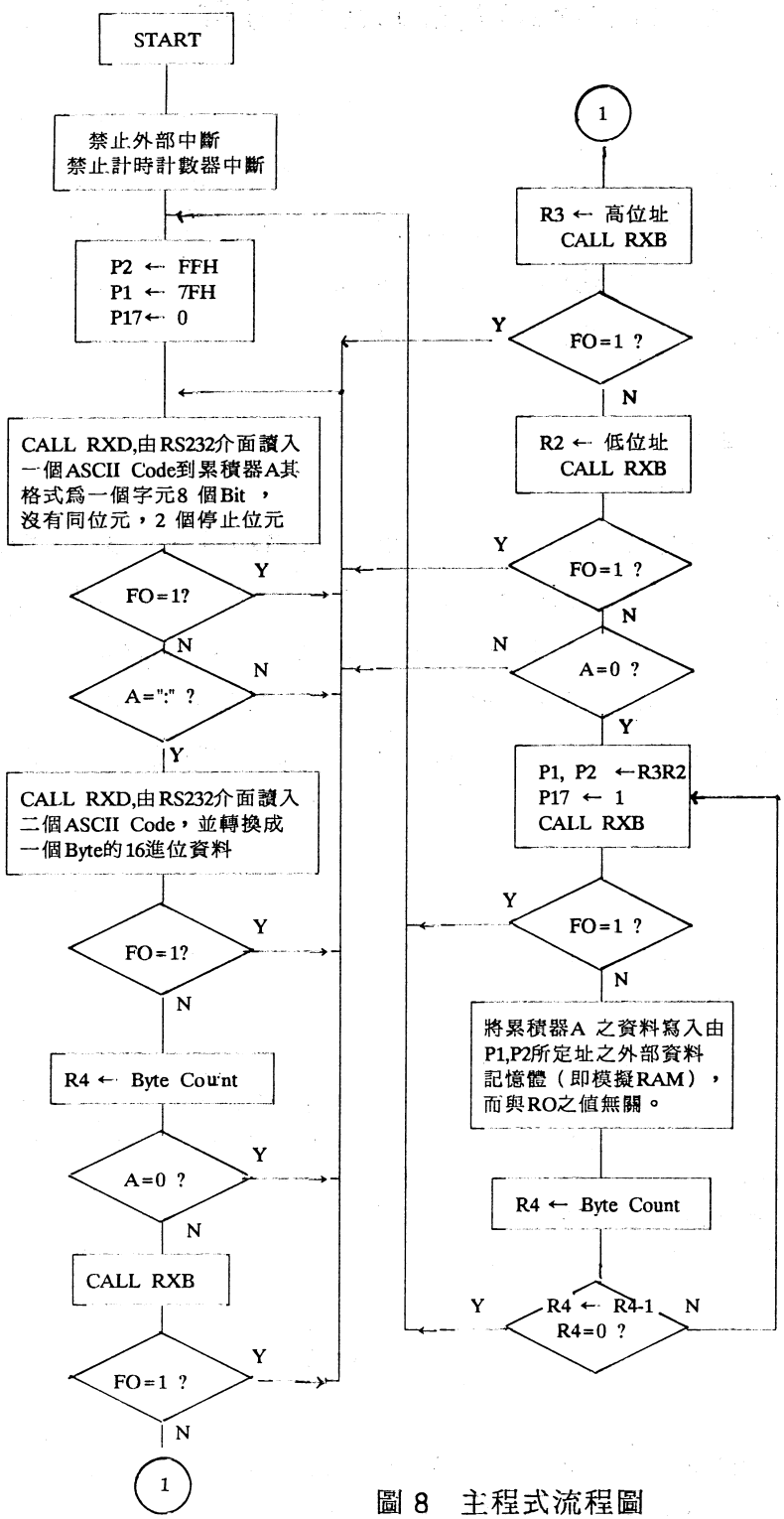

START
禁止外部中斷
禁止計時計數器中斷
P2 ← FFH
P1 ← 7FH
P17← 0
CALL RXD,由RS232介面讀入一個ASCII Code到累積器A其格式爲一個字元8 個Bit ，沒有同位元，2 個停止位元
FO=1?
A=":" ?
CALL RXD,由RS232介面讀入二個ASCII Code，並轉換成一個Byte的16進位資料
FO=1?
R4 ← Byte Count
A=0 ?
CALL RXB
FO=1 ?
1
1
R3 ← 高位址
CALL RXB
FO=1 ?
R2 ← 低位址
CALL RXB
FO=1 ?
A=0 ?
P1, P2 ←R3R2
P17 ← 1
CALL RXB
FO=1 ?
將累積器A 之資料寫入由P1,P2所定址之外部資料記憶體（即模擬RAM），而與RO之值無關。
R4 ← Byte Count
R4 ← R4-1
R4=0 ?
Y
N

圖 8　主程式流程圖

三、軟體程式流程圖與程式分析

EPROM模擬器的控制程式可分爲三部分，第一部分爲主程式，由8748 T0腳讀入Intel HEX格式之機器碼，寫入P20～P27，P10～P16所定址的模擬RAM內。第二部分爲RXB 副程式，將由T0讀入之ASCII Code轉換成16進位碼，一共讀入2個ASCII Code，並轉換成一個Byte的16進位碼。第三部分爲RXD 副程式，可接收傳輸率爲9600 BPS的異步串列傳輸資料。

㈠主程式

參考圖8 流程圖及附錄2程式列表說明如下：

1. 5-6 行，禁止外部中斷及計時計數器中斷。
2. 7-9 行，P20～P27及P10～P16輸出爲高阻抗(50KΩ)高電位輸出，P17=0時，模擬RAM 的位址由目標系統的EPROM 來定址，模擬RAM 此時處於僅能讀出的模擬狀態，也就是與目標系統EPROM ON LINE 的狀態。
3. 10-13 行，由8748 T0 腳讀入一個ASCII Code，判斷是否爲Intel HEX格式機器碼的前置碼":" (ASCII Code 3A)，Intel HEX檔的格式如圖9所示。

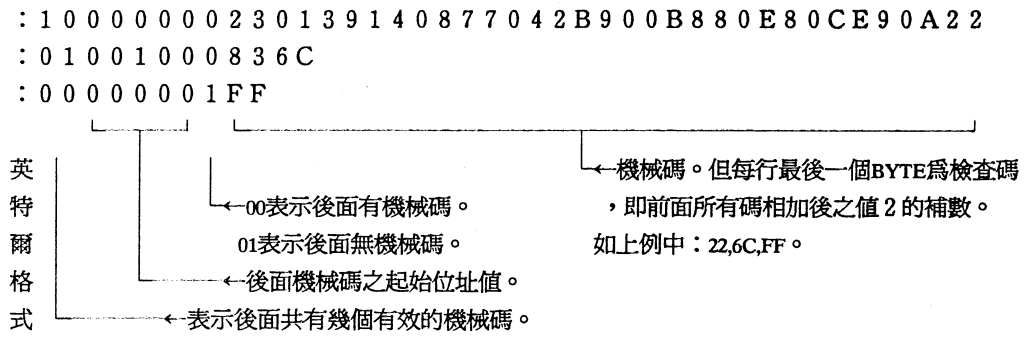


圖 9　Intel HEX 檔格式

4. 14-16 行，由8748 T0 腳讀入兩個ASCII Code，轉換成一個Byte的16進位碼，存入R4，R4所存的是後面機器碼的長度。

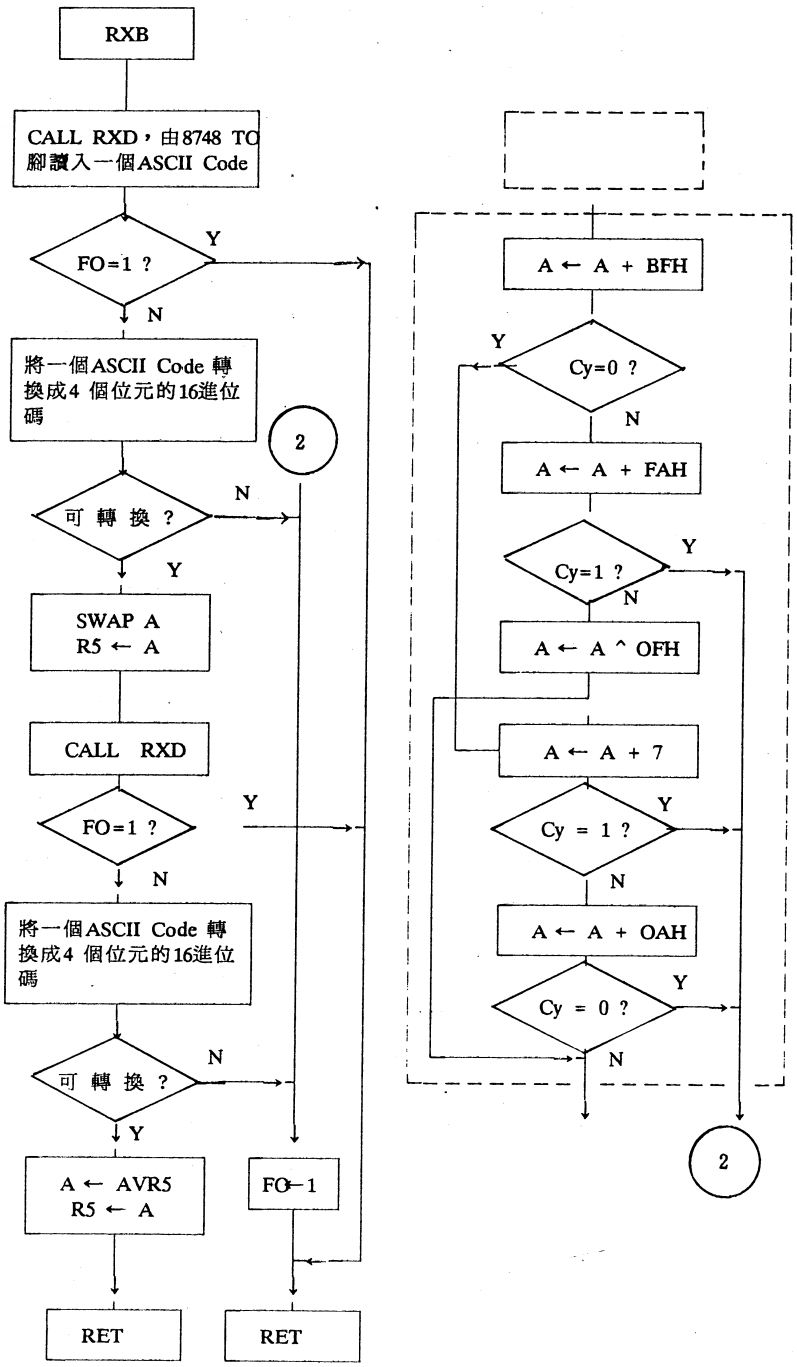


圖 10　RXB 副程式流程圖

5.17行，如果機器碼的長度等於0 ，則回到第10行，重新再判斷下一個ASCII Code是否爲":" ，如果不是則捨棄不要。

6.18-20 行，由8748 TO 腳讀入兩個ASCII Code，轉換成一個Byte 的16進位碼，存入R3，R3所存的是後面機器碼的起始位址的高位址。

7.21-23 行，由8748 TO 腳讀入兩個ASCII Code，轉換成一個Byte的16進位碼，存入R2，R2所存的是後面機器碼的起始位址的低位址。

8.24-26 行，由8748 TO 腳讀入兩個ASCII Code，轉換成一個Byte的16進位碼，判斷是否等於00；如果等於00，表示後面有機器碼，繼續執行下面第27行的指令；如果等於01，則表示後面無機器碼，重新回到第10行執行CALL RXD的指令。

9.27-31行，將機器碼的位址，R3R2送到P16～P10及P27～P20，同時將P17定爲1 ，兩個74SL244 位址緩衝器的輸出均變成高阻抗輸出，模擬RAM的位址變成由8748的P16～P10及P27～P20來定址，此時模擬RAM 處於可由8748 $\overline{WR}$ 控制寫入資料的狀態，也就是與目標系統EPROM OFF LINE的狀態。

10.32-34 行，由8748 TO腳讀入兩個ASCII Code ，轉換成一個Byte的16進位碼，這一個Byte的16進位碼就是機器碼，存入累積器A ；執行MOVX @R0，A 這一個指令，可將累積器的資料寫入由P16～P10及P27～P20所定址的模擬RAM 內，因爲執行MOVX @R0，A，8748 $\overline{WR}$可產生負脈波，而依據圖1 硬體電路可知，機器碼所寫入的位址與R0無關。

11.35-38 行，R3R2←R3R2+1，即機器碼的位址加1 。

12.39-40 行，R4←R4-1，機器碼的長度計數器減1 ，判斷是否等於0 ；如果不等於0 ，則回到第27行執行，將下個機器碼寫入模擬RAM 的下一個位址；如果等於0 ，則回到第7 行執行，將P16～P10及P27～P20變成高阻抗高電位輸出，P17=0 ，模擬RAM 此時處於僅能讀出的模擬狀態，也就是與目標系統EPROM ON LINE 的狀態，再準備讀取下一行Intel HEX格式的機器碼，再寫入對應的模擬RAM 的位址。如果這個Intel HEX 格式的機器碼已寫入完畢，則8748將執行第10行CALL RXD的指令，等候讀取下一個Intel HEX 格式的機器碼檔。

㈡CALL RXB副程式

參考圖10流程圖及附錄 2 程式列表說明如下：

1.CALL RXB副程式是由42-55 行，以及56-69 行兩段幾乎完全相同的程式組合而成，每一段都是將一個ASCII Code轉換成4 個位元的16進位碼。

2.42行，由8748 T0腳讀入一個Byte 的ASCII Code。

3.43-55 行，參考圖11判斷轉換過程圖，判斷所讀入的一個Byte的ASCII Code是否爲30～39，41～46，16進位數的0～F。如果不是，則執行71-72行的指令將錯誤旗號F0變成 1 ，回到主程式。如果是，則將一個Byte的ASCII Code轉換成4 個位元的16進位碼，因爲第一個 Byte的 ASCII Code爲16進位碼的高位數，因此執行SWAP A指令，將轉換後的16進位碼，換到累積器的高位數，而低位數等於0 ，同時暫時存入R5暫存器。

```
          40  41  42  43  44  45  46  47
    +  ,  BF  BF  BF  BF  BF  BF  BF  BF
 ------------------------------------------
         0FF 100 101 102 103 104 105 106
    +         FA  FA  FA  FA  FA  FA  FA
 ------------------------------------------
              FA  FB  FC  FD  FE  FF 100
    ∧         0F  0F  0F  0F  0F  0F
 ------------------------------------------
   A =        0A  0B  0C  0D  0E  0F


    2F  30  31  32  33  34  35  36  37  38  39  3A
 +  BF  BF  BF  BF  BF  BF  BF  BF  BF  BF  BF  BF
-----------------------------------------------------
    EE  EF  F0  F1  F2  F3  F4  F5  F6  F7  F8  F9
 +  07  07  07  07  07  07  07  07  07  07  07  07
-----------------------------------------------------
    F5  F6  F7  F8  F9  FA  FB  FC  FD  FE  FF 100
 +  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A
-----------------------------------------------------
    FF 100 101 102 103 104 105 106 107 108 109
-----------------------------------------------------
A =     00  01  02  03  04  05  06  07  08  09
```

圖 11 ASCII Code 轉換成 16 進位碼的轉換過程圖

4.56行，由8748 T0 腳讀入第二個Byte的ASCII Code。

5.57-72 行，參考圖11判斷轉換過程圖，判斷所讀入的ASCII Code是否爲30～39，41～46，16進位數的0～F。如果不是，則執行71-72行的指令，

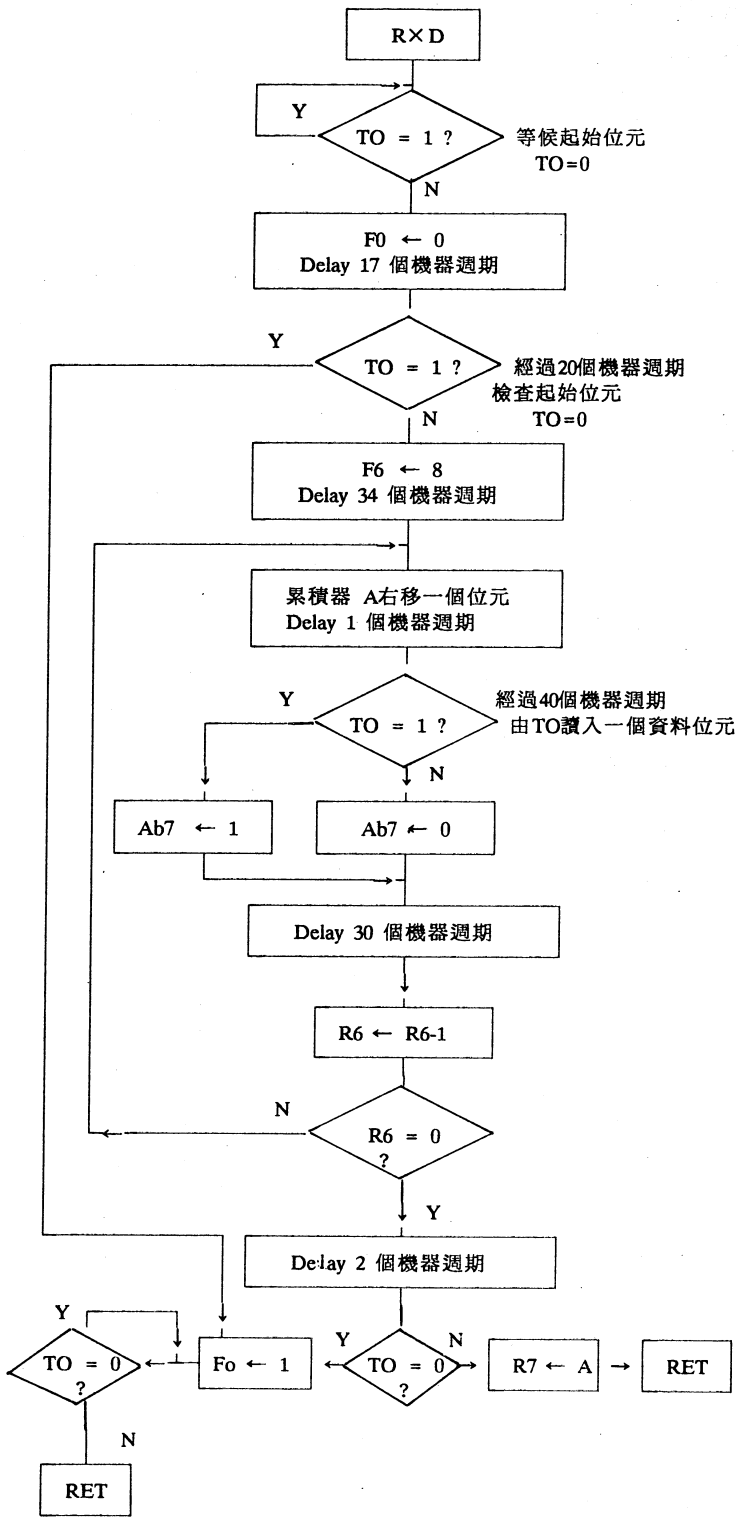


圖 12　RXD 副程式流程圖

將F0變成1 ，回到主程式；如果是，則將此ASCII Code轉換成4 個位元的16進位碼，再與R5 OR在一起，則R5及累積器均存有一個Byte 的16進位機器碼，回到主程式。

㈢CALL RXD 副程式

參考圖12流程圖及附錄 2 程式列表說明如下：

1.RS232 異步串列傳輸碼的格式如圖13所示，包含一個低電位的起始位元，8 個位元的資料位元，2 個位元高電位的停止位元。

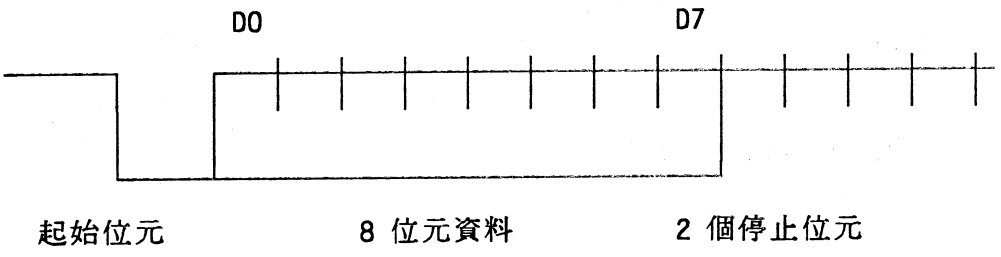


圖 13 RS232 異步串列傳輸碼格式

2.RXD 副程式是一段以軟體方法來處理RS232 串列傳輸料接收工作的程式。配合IBM PC串列資料傳輸率爲9600 BPS的傳輸速度，在8748的石英振盪部分，選擇一個最適當的振盪頻率，可以準確的接收IBM PC傳送過來的串列傳輸資料，經實際計算比較，結果以5.76 MHZ的石英振盪器最適當，而且軟體程式的處理速度也能配合。如果串列資料的傳輸率爲9600 BPS ，則8748由T0腳偵測到異步串列傳輸資料的低電位起始位元後，每隔40個機器週期，由T0腳讀入一個位元的資料，一共讀入8 個位元的資料和檢查一個位元的停止位元（高電位）後，才完成一個異步串列傳輸碼的接收工作。

3.74行，RXD JT0 RXD 等候異步串列傳輸碼低電位的起始位元，如果T0一直保持高電位，則8748一直執行這一個指令，也就是當IBM PC已將一個

Intel HEX 檔傳送完畢或沒有資料傳送時8748一直執行這一個指令，等候接收下一個資料檔。

4.75-79 行，將F0旗號清除爲0 ，因爲F0爲一個錯誤旗號，當執行完RXD副程式後，如果F0變成1 ，則表示接收資料有錯誤，因此每次要開始接收一個串列傳輸資料時，先將F0清除爲0 。其它三個指令只是延遲的作用，因爲從T0腳偵測到起始位元的低電位開始，經過20個機器週期，也就是起始位元的中點，再從T0腳讀進一次，如果仍然是低電位，則表示是起始位元，而不是雜訊，請參考附錄2 程式列表中間的統計數字。

5.80-85行，R6=8，因爲一個字元有8個位元，81及82行兩個指令爲延遲作用。RR A指令將累積器的資料右旋一個位元，因爲每次由T0腳讀入的一個位元的資料都存在累積器的最高位元，第一次執行RR A並無作用。其它7 次執行RR A指令，則配合將串列傳輸資料轉換成並列資料，因爲字元資料的最低位元D0先傳送，所以RR A指令配合做串列資料變並列資料的轉換。

6.83-92 行，每隔40個機器週期，由T0腳讀入一個位元的資料，也就是在每個位元資料的中點做資料取樣工作。如果T0=0，表示對應的資料位元等於0 ，則將累積器的最高位元定爲0 ；如果T0=1，表示對應的資料位元等於1 ，則將累積器的最高位元定爲1 。每次由T0讀入一個資料位元，都先放在累積器的最高位元，然後將R6計數器減1 ，R6若不等於0 ，則將累積器的資料右旋，再由T0讀入下一個資料位元，一直到第8 個資料位元，也就是D7讀入後，R6等於0 ，此時8 個位元的資料；已完全依序存入累積器。

7.93-100行，同樣的經過40個機器週期，由T0讀入一個位元的資料，這個資料應該是高電位的停止位元，如果是，則將累積器的資料存入R7，完成一個串列資料的接收工作，如果不是，則先將F0旗號變成1 ，表示傳輸資料格式不正確。99行指令L4 JNT0 L4的作用，主要作用爲假如有錯誤時，如果T0等於0 ，則要等到T0變成1 ，也就是資料傳輸線回到高電位狀態時，才令其回到主程式，準備接收下一個串列傳輸資料。

8.當執行完95行 JNT0 RXERR 指令，檢查確定第一個高位元的停止位元後，如果只有一個停止位元，則再經過20個機器週期後，馬上可能就會有

下一個串列資料的低電位起始位元送到傳輸線上。根據附錄 2 程式列表上不同情況下的指令執行機器週期數計算，最壞的情況爲44個機器週期，也就是執行33-40行，再到7-10行執行時，所需的時間爲3+18+11+12=44。因此如果只有一個停止位元，則第二個串列傳輸資料將無法正確接收到，因此IBM PC RS232介面所傳送的串列傳輸資料必須含有2 個停止位元，如此才不會產生錯誤。

㈣將兩個EPROM 模擬器合併使用，可構成一個16位元的EPROM 模擬器，此時8748的T1腳必須接到+5V 高電位，而兩個EPROM 模擬器的控制程式也要稍加修改，加上測試T1接腳狀態的指令，來分辨是要做16位元模擬，還是8位元的模擬。兩個EPROM 模擬器的控制程式也要分爲奇數Byte的控制程式和偶數Byte的控制程式，奇數Byte的控制程式參考附錄3 EPROM 模擬器程式列表 (8/16位元) 及圖14部分新增程式流程圖，只要將第50行指令改爲JNC RECE，就變成偶數Byte的控制程式。

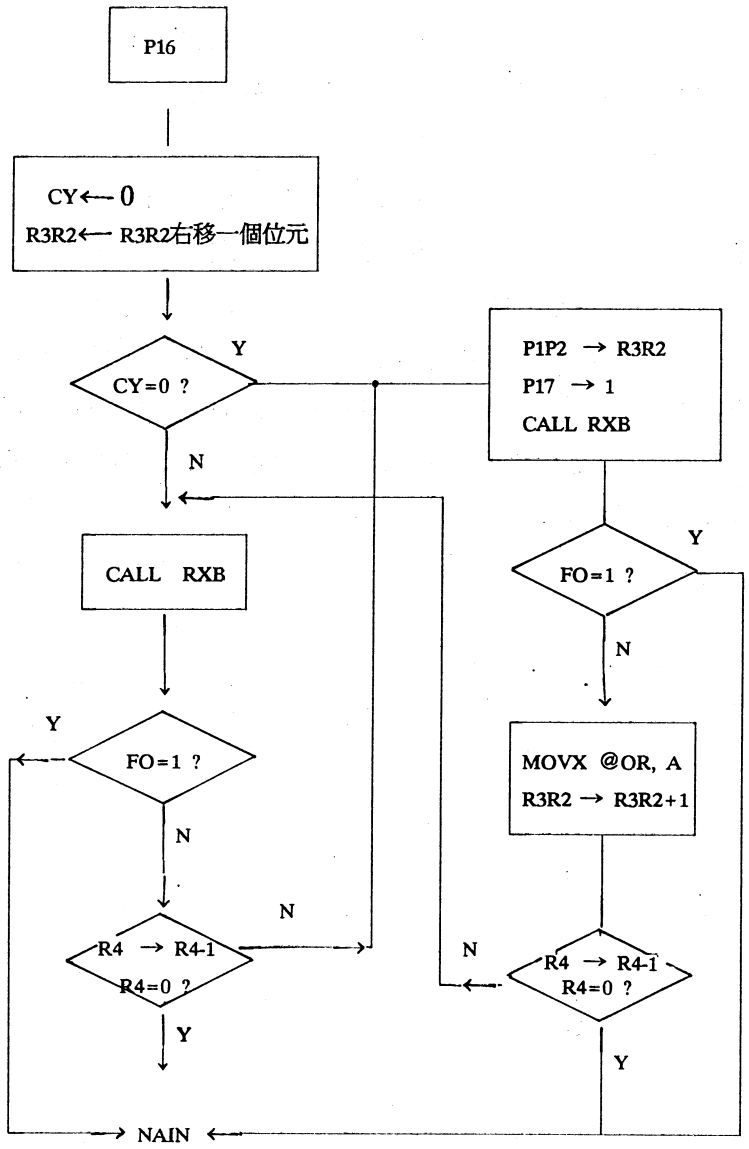


圖 14　部分新增程式流程圖

四、系統功能與使用說明

㈠EPROM 模擬器

EPROM 模擬器是一個以RS232 介面與IBM PC連線的微電腦實體模擬系統，利用PC之PE2 等編輯程式，編輯所欲模擬目標系統CPU 的組合語言原始程式，再經由Cross Assembler 組譯，連結(LINK)產生Intel HEX 之機器碼檔，並加上OFFSET Code ，再將此Intel HEX 格式之機器碼檔，透過IBM PC的RS232 介面DOWN LOAD到EPROM模擬器的模擬RAM 上，即可開始做實體模擬工作。如果結果不正確，再將程式修改、模擬，一直到滿意爲止，最後將模擬好的程式燒到EPROM ，插到目標系統EPROM 插座上，即可完成目標系統軟體的設計工作。

EPROM 模擬器爲了達到完全智慧型，除了硬體和軟體的配合之外，還必須在操作方法上加以配合，也就是在產生Intel HEX 格式的機器碼檔時，配合所欲模擬的EPROM 編號加上OFFSET Code ，其對照表如圖15所示。

OFFSET CODE / EPROM 編號	8 位元	16 位元
2716	7800	F000
2732	7000	E000
2764	6000	C000
27128	4000	8000
27256	0000	0000

圖 15　EPROM 與 OFFSET CODE 對照表

EPROM 模擬器的操作方法說明如下：

1.工作磁片上必須有MODE.COM, HEX.EXE, DEBUG.COM等檔。

建一批次檔 DL256.BAT

其內容爲： MODE COM%2 :96,N,8,2

COPY %1. HEX COM%2

2.此模擬器只能 DOWNLOAD INTEL HEX 格式之檔案即XXX.HEX.

若接RS232 COM1, 則鍵入DL256 XXX 1

若接RS232 COM2, 則鍵入DL256 XXX 2

3.SRAM若爲6264，則可模擬2716, 2732, 2764.

SRAM若爲62256 ，則可模擬2716, 2732, 2764, 27128, 27256.

4.使用8748, Z80, 6502, 68000等Cross Assembler 時。

(1)以PE2 等編輯程式編輯成XXX.ASM。

(2)經Cross Assembler 組譯成XXX.OBJ。

(3)再經LINK產生XXX.TSK.(OFFSET Code=0)

(4)利用HEX 檔將XXX.TSK 轉換成XXX.HEX ，並加上OFFSET Code。

EPROM 編號與OFFSET Code 對照如下：

2716:7800, 2732:7000, 2764:6000,

27128:4000, 27256:0000.

5.使用8088 MASM 或TASM時。

(1)利用PE2 等編輯程式，編輯XXX.ASM 原始程式。

(2)將其組譯並連結產生XXX.EXE。

(3)利用DEBUG 將XXX.EXE 轉換成XXX.TSK ，操作方法如下：

A>DEBUG

-N 檔名.EXE

-L

-D CS:ORG 位址(IP) 假定8000。

-R 看 CS=? IP=?

-RCS 如果 CS=123C 則

CS:1A2C

-N 檔名.TSK

-RCX

CX：長度

-W （將CS:100 起之機器碼，長度BX:CX 寫入檔名.TSK）

-L 8000:0（觀察是否正確）

-Q

A>

(4)再利用HEX 檔將XXX.TSK 轉換成XXX.HEX ，並根據欲模擬之EPROM 編號加上OFFSET Code。

㈡8749單晶片模擬器

8749單晶片模擬器單獨使用時，必須在程式設計好之後，將程式燒到EPROM 上，再將EPROM 插到單晶片模擬器的EPROM 插座上，才可做實體模擬的工作。如果與EPROM模擬器配合使用，則程式設計好之後，只要DOWN LOAD到EPROM 模擬器，利用模擬器的模擬功能，即可做單晶片的實體模擬，而不必將程式燒到EPROM 或單晶片內，既方便、快速又實用。8749單晶片模擬器可完全取代一顆8748或8749，它可模擬8748、8749硬體的所有功能，軟體方面，不可執行的指令有ANL BUS,#N及ORL BUS,#N兩個指令，這兩個指令在世界上一般MICE及ICE 也大都不能執行。

8749單晶片模擬器的操作方法說明如下：

1.8748/8749 之第7 腳EA接腳必須接地。

2.利用8748 Cross Assembler時。

(1)以PE2 等編輯程式編輯成XXX.ASM。

(2)經Cross Assmbler組譯成XXX.OBJ。

(3)再經LINK產生XXX.TSK (OFFSET Code=0)

(4)利用HEX 檔將XXX.TSK 轉換成XXX.HEX ，並加上OFFEST Code 7800.(2716)。

3.利用FORCE-48時

(1)利用FORCE-48本身之編輯程式編輯原始程式XXX.48P，經組譯成XXX.HEX。

(2)利用DEBUG 將XXX.HEX 換成XXX.TSK。

```
A>DEBUG
-N XXX.HEX
-L 8000:0
-RCS
 CS:7FF0
-RCX
 CX:400(1K)
-N XXX.TSK
-W
-Q
```

(3)再利用HEX 檔將XXX.TSK 轉換成XXX.HEX 並加上OFFSET Code 7800。

*(4)利用CONV.BAT將XXX.HEX 直接轉換成 XXX.HEX (自動加上OFFSET Code 7800) 操作方法 A>CONV XXX (僅限於1K)

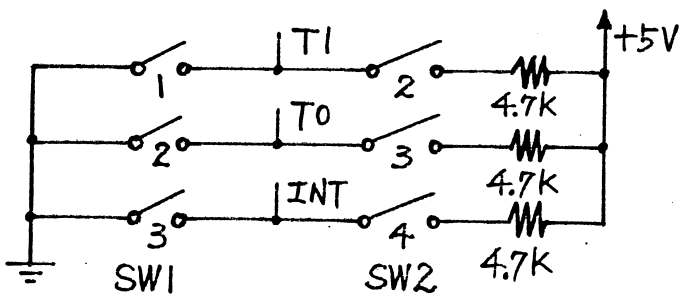
4.SW1 8 Bit DIP Switch切換功能

1	2	3	4	5	6	7	8
T1	TO	INT	DB	P23	P22	P21	P20
ON:LOW			DB	ON:OUTPUT,	OFF:INPUT.		
OFF:HI			P20,P21,P22,P23	ON:INPUT,	OFF:OUTPUT.		

5.SW2 4 Bit DIP Switch 切換功能。

1	2	3	4
SS	T1	TO	INT
SS	ON：正常執行。		OFF：單步執行。

6.SW1 8 Bit 與SW2 4 Bit DIP Switch 關係圖。



7.如上所示，本模擬器之P20--P23可由DIP Switch切換IN/OUT方向。
DB (DATA BUS) 可由DB Bit 切換IN/OUT 方向。

8.P20--P23當I/O 時

SW4 1234 OFF
5678 ON

接 8243

SW4 1234 ON
5678 OFF

9.DATA BUS可接外接RAM 使用或其它介面IC，此時SW5 8Bit DIP Switch 全部ON, SW6 Bit7 OFF, SW1 Bit4 OFF 。DATA BUS 當I/O 時，SW5 全部 OFF, SW6 Bit7 ON。

10.SW6 12 ON, 34 OFF振盪頻率爲模擬器之6MHz。

SW6 12 OFF, 34 ON 使用外界振盪頻率。
SW6 5 OFF, 6 ON 由外界之RESET 鍵控制RESET。
SW6 5 ON, 6 ON 由模擬器之RESET 鍵控制RESET。

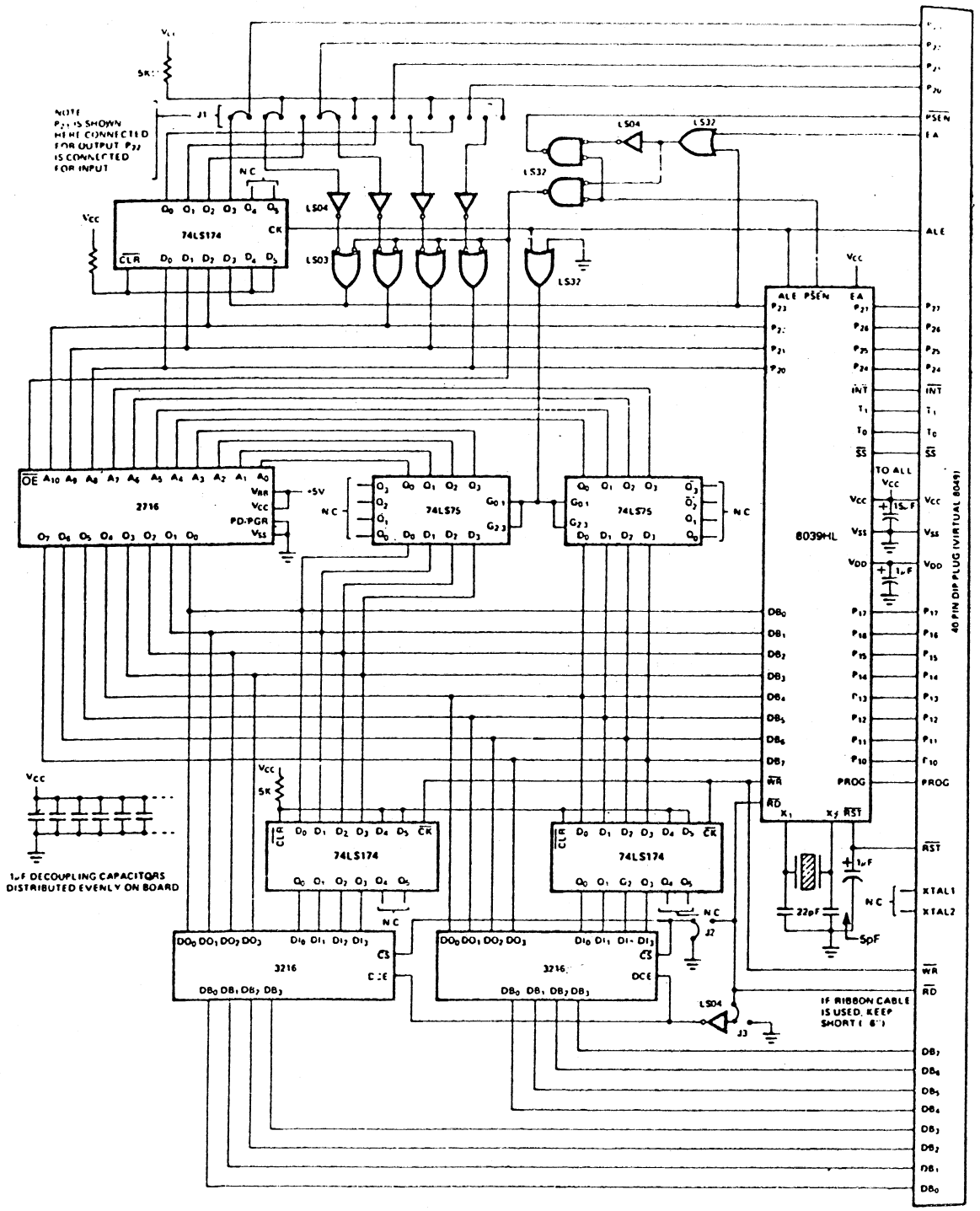
五、結 論

EPROM 模擬器及8749單晶片模擬器已裝設於本校電子科微電腦實習室，8 位元介面技術實習室等三間實習室及教師研究室， 共76組，其中Z80 微電腦實習室配合Z80 CPU 系統的模擬發展，只裝了EPROM 模擬器。希望能對電子科在微電腦系統實習、介面控制、單晶片實作、工業控制、專題製作等等課程教學有所幫助，讓學生有更多實作的機會，以培養更多、更好的微電腦控制的設計人才。

六、參考資料

1.Intel, MCS-48 Microcomputer User's Manual, 1978。
2.張容豪編譯，8048/8049 微電腦，全華科技圖書公司。
3.TI, TTL DATA BOOK, 1981, 宏碁股份有限公司。
4.李隆財，微電腦發展工具EPROM/SRAM模擬器之研究製作，民國75年12月10日，勤益學報第四期。
5.鍾富昭、彭鴻洲編著，8048體系軟硬體專題製作，全華科技圖書公司。
6.鍾富昭編著，8048系列產品設計與開發工具之應用，全華科技圖書公司。
7.施威銘著，IBM MSDOS 磁碟作業系統實務手册，波前圖書公司。
8.謝聿婷編著，IBM PC介面技術與週邊設備實習，全華科技圖書公司。
9.Motorola, Linear Interface Integrated Circuits, 1979。
10.FORCE 48 user's Manual, 協華科技股份有限公司。
11.2500 A. D. Z80 CROSS Assembler Version 4.00。
12.2500 A. D. 6502 Cross Assembler Version 4.00。
13.2500 A. D. 8748 Cross Assembler Version 4.01。
14.2500 A. D. 8051 Cross Assembler Version 4.02。
15.2500 A. D. 8086 Cross Assembler Version 4.01。
16.2500 A. D. 68000 Cross Assembler Version 3.41。

附錄 1　8049 Emulator 電路圖及資料



8049 EMULATOR CIRCUIT DESCRIPTION-6 MHZ

The following is an explanation of a circuit which emulates the operation of an Intel® 8049 using a standard EPROM for program storage.

With the 8049, software may be developed by running external program memory, but doing so requires the use of the bus and P_{23}-P_{20} to access this memory.

The circuit shown may be used to restore the normal functioning of these twelve I/O pins. The circuit consists of an 8039 CPU, 2716 EPROM, two 8216 bi-directional bus drivers, and eight other 7400 Series Low-Power Schottky TTL packages. The whole assembly can be built on a 2-3/4" x 4" board.

A cable coming off the board can be terminated by a forty-pin plug which may be inserted directly into the CPU socket intended for the 8049 in a system undergoing design or testing. Alternatively, a pattern of forty pins extending below the board can be used to plug the board directly into the system undergoing testing, "piggy-back" fashion. The emulator board may be configured in various ways so that the 40 pin plug is the logical equivalent of an 8049 in every legal operating mode. (In the following explanation of the operation of the circuit, an asterisk appearing before a signal or pin number — as in *$\overline{PSEN}$ — refers to that pin on the "virtual 8049" represented by the forty-pin plug).

Since the CPU is identical with the 8049 in all respects other than its lack of program memory, most of the pins of the 8039 are simply connected directly to the corresponding pins of the forty-pin plug. These include all of Port 1, the high order bits of Port 2, the test pins, etc. Signals which are emulated with additional logic include the rest of Port 2, DB_7-DB_0, *$\overline{PSEN}$, etc. $\overline{RD}$, $\overline{WR}$, ALE, and $\overline{PSEN}$ are obtained from the 8039, but are also used by the emulation circuitry.

The EA input of the 8039 is hard-wired high so all instruction fetches are made from the 2716. Two 74LS75 four-bit latches gated by the buffered ALE signal are used to hold the lower eight bits of address from the time-multiplexed data bus. Since the Bus is being used for fetching instructions, data latched to the Bus will be lost on the next instruction fetch. Two 74LS174 latches are used to retain the output data when a bus write is executed. These latches are triggered by the trailing edge of the $\overline{WR}$ pulse, so their outputs are glitch free. Since logical operations to the bus do not generate a WR strobe, the "OUTL BUS, #data" "ANL BUS,#" and "ORL BUS,#" instructions may not be used, though they do function properly with the other ports.

The two 8216 bi-directional bus drivers normally buffer the latched bus contents to the DB pins of the virtual 8049. When an "INS A,BUS" instruction is executed, they buffer the input signals on to the emulator data bus. Thus, the circuit is designed to use the Bus for both latched output and strobed input. If $DB_7$9DB_0 of the 8049 are to be used solely for input data, J2 and J3 may be changed from what is shown in the Figure, so that DB_7-DB_0 act as high impedance inputs and the 8216s are enabled only when the read operation is performed. If the bus is to be used only for latched output, the 8216s can be omitted entirely.

Bi-directional data transfers which require the transfer of address information as well as data, such as to and from external data memory, require removal of the 8216s and replacement with 16-pin jumper blocks on which the DB_x pins are connected with the respective DO_x pins.

The lower four bits of Port 2 are also used in fetching instructions from the 2716, in addition to their use as input or output pins in the user's system. In configuring the emulator for a particular application, the user must dedicate each of these as either an input or output pin and connect jumper set J1 accordingly. Any mix of input and output pins is allowed. At the beginning of each instruction fetch, the last data written to P2 will be present on P_{23}-P_{20} at the rising edge of ALE and will be latched by a 74LS174. The latched data may be connected through the jumpers to those pins which will be used as outputs on the 8049. Emulator pins used as inputs should be pulled above 2.0V for a logic "one". If this is not the case, i.e., if switches to Ground are to be read, 50K pullup resistors should be added to the circuit on each input. They were omitted from the diagram to minimize input loading.

Pins which will be used as inputs may be connected to the input of an OR gate formed of inverters and open-collector NAND gates. The input signals will be relayed directly to the 8039 and will be read by an "IN A,P2" instruction. But when $\overline{PSEN}$ is low, the NAND outputs are forced off, allowing the 8039 pins to be used for high-order program adressing. Open-collector outputs are needed to prevent line contention when $\overline{PSEN}$ is not low.

If 8243s will be be used in the final system, the low order pins of Port 2 must be connected directly to the plug. This may be done by replacing the Port 2 latch with four jumpers connecting the inputs to the outputs. The NANDs should be removed or disabled by grounding the common NAND inputs.

The cluster of three OR gates is used to enable the on-board 2716 and generate the *$\overline{PSEN}$ signal, each of which is a function of $\overline{PSEN}$, *EA, and the high order bit of the program counter. Thus *$\overline{PSEN}$ is generated, forcing an off-board read, only when a jump has been made to Memory Bank 1 or when *EA is brought high. If this feature is to be used to address off-board memory, DB_7-DB_0 may not be used for normal I/O. The 8216s and 74LS174 must be replaced with jumper blocks and the open collector NAND gates disabled, as explained above. The same changes are required to operate the board in single step mode.

NOTE: FOR EMULATION AT 11 MHZ
1. Substitute a 2716-1
2. Delete 74LS03 package (leave lines open). Elimination of 74LS03 precludes use of P20-P23 as inputs

附錄 2　EPROM 模擬器程式列表（8 位元）

```
 3                          ;DL256 FOR 8748
 4   0000                          ORG     0
 5   0000   15                     DIS     I
 6   0001   35                     DIS     TCNTI
 7   0002   89 FF    MAIN:    ORL     P1,#FFH
 8   0004   8A FF             ORL     P2,#FFH
 9   0006   99 7F             ANL     P1,#7FH
10   0008   14 72    MAIN1:   CALL    RXD
11   000A   B6 08             JF0     MAIN1
12   000C   D3 3A             XRL     A,#3AH   ;":"
13   000E   96 08             JNZ     MAIN1
14   0010   14 3B             CALL    RXB
15   0012   B6 08             JF0     MAIN1
16   0014   AC                MOV     R4,A     ;BYTE COUNT
17   0015   C6 08             JZ      MAIN1
18   0017   14 3B             CALL    RXB
19   0019   B6 08             JF0     MAIN1
20   001B   AB                MOV     R3,A     ;HI ADDRESS
21   001C   14 3B             CALL    RXB
22   001E   B6 08             JF0     MAIN1
23   0020   AA                MOV     R2,A     ;LOW ADDRESS
24   0021   14 3B             CALL    RXB
25   0023   B6 08             JF0  2  MAIN1
26   0025   96 08             JNZ  2  MAIN1
27   0027   FB        MAIN2:  MOV  1  A,R3
28   0028   43 80             ORL  2  A,#80H
29   002A   39                OUTL 2  P1,A
30   002B   FA                MOV  1  A,R2
31   002C   3A                OUTL 2  P2,A
32   002D   14 3B             CALL 2  RXB
33   002F   B6 02             JF0  2  MAIN
34   0031   90                MOVX 2  @R0,A
35   0032   1A                INC  1  R2
36   0033   FA                MOV  1  A,R2
37   0034   96 37             JNZ  2  MAIN3
38   0036   1B                INC  1  R3
39   0037   EC 27    MAIN3:   DJNZ 2  R4,MAIN2
40   0039   04 02             JMP  2  MAIN
41                         ;
42   003B   14 72    RXB:     CALL 2  RXD
43   003D   B6 71             JF0  2  RXBX
44   003F   03 BF             ADD  2  A,#BFH
45   0041   E6 4B             JNC  2  RXB1     ;<41H
46   0043   03 FA             ADD  2  A,#FAH
47   0045   F6 70             JC   2  RXBX1    ;>46H
48   0047   53 0F             ANL  2  A,#0FH
49   0049   04 53             JMP  2  RXB2
50   004B   03 07    RXB1:    ADD  2  A,#7
51   004D   F6 70             JC   2  RXBX1    ;>39H
```

```
 52   004F   03 0A          ADD   2  A,#0AH
 53   0051   E6 70          JNC   2  RXBX1    ;<30H
 54   0053   47       RXB2: SWAP  2  A
 55   0054   AD             MOV   1  R5,A
 56   0055   14 72          CALL  2  RXD
 57   0057   B6 71          JF0   2  RXBX
 58   0059   03 BF          ADD   2  A,#BFH
 59   005B   E6 65          JNC   2  RXB3     ;<41H
 60   005D   03 FA          ADD   2  A,#FAH
 61   005F   F6 70          JC    2  RXBX1    ;>46H
 62   0061   53 0F          ANL   2  A,#0FH
 63   0063   04 6D          JMP   2  RXB4
 64   0065   03 07    RXB3: ADD   2  A,#7
 65   0067   F6 70          JC    2  RXBX1    ;>39H
 66   0069   03 0A          ADD   2  A,#0AH
 67   006B   E6 70          JNC   2  RXBX1    ;<30H
 68   006D   4D       RXB4: ORL   1  A,R5
 69   006E   AD             MOV   1  R5,A
 70   006F   83             RET   2
 71   0070   95       RXBX1:CPL   2  F0
 72   0071   83       RXBX: RET   2
 73                   ;
 74   0072   36 72    RXD:  JT0   2  RXD      ;WAIT START BIT
 75   0074   85             CLR   1  F0
 76   0075   00             NOP   1
 77   0076   BF 07          MOV   2  R7,#7
 78   0078   EF 78    L1:   DJNZ  2  R7,L1     ;2*7
 79   007A   36 9A          JT0   2  RXERR
 80   007C   BE 08          MOV   2  R6,#8
 81   007E   BF 10          MOV   2  R7,#10H
 82   0080   EF 80    L2:   DJNZ  2  R7,L2     ;2*16
 83   0082   77       RX0:  RR    1  A
 84   0083   00             NOP   1
 85   0084   36 8A          JT0   2  RX1
 86   0086   53 7F          ANL   2  A,#7FH   ;T0 = 0
 87   0088   04 8E          JMP   2  RX2
 88   008A   43 80    RX1:  ORL   2  A,#80H   ;T0 = 1
 89   008C   04 8E          JMP   2  RX2
 90   008E   BF 0E    RX2:  MOV   2  R7,#0EH
 91   0090   EF 90    L3    DJNZ  2  R7,L3     ;2*14
 92   0092   EE 82          DJNZ  2  R6,RX0
 93   0094   00             NOP   1
 94   0095   00             NOP   1
 95   0096   26 9A          JNT0  2  RXERR    ;CHECK STOP BIT
 96   0098   AF             MOV   1  R7,A
 97   0099   83             RET   2
 98   009A   95       RXERR:CPL   1  F0
 99   009B   26 9B    L4:   JNT0  2  L4
100   009D   83             RET   2
102   009E                  END
```

附錄 3　EPROM 模擬器程式列表（8/16位元）

```
 1                          ;
 2                          ; DL256P16 FOR 8748
 3                          ;
 4   0000                           ORG     0
 5   0000   15                      DIS     I
 6   0001   35                      DIS     TCNTI
 7   0002   89 FF    MAIN:   ORL     P1,#FFH
 8   0004   8A FF            ORL     P2,#FFH
 9   0006   99 7F            ANL     P1,#7FH
10   0008   14 99    MAIN1:  CALL    RXD
11   000A   B6 08            JF0     MAIN1
12   000C   D3 3A            XRL     A,#3AH  ;":"
13   000E   96 08            JNZ     MAIN1
14   0010   14 62            CALL    RXB
15   0012   B6 08            JF0     MAIN1
16   0014   AC                      MOV     R4,A    ;BYTE COUNT
17   0015   C6 08            JZ      MAIN1
18   0017   14 62            CALL    RXB
19   0019   B6 08            JF0     MAIN1
20   001B   AB                      MOV     R3,A    ;HI ADDRESS
21   001C   14 62            CALL    RXB
22   001E   B6 08            JF0     MAIN1
23   0020   AA                      MOV     R2,A    ;LOW ADDRESS
24   0021   14 62            CALL    RXB
25   0023   B6 08            JF0     MAIN1
26   0025   96 08            JNZ     MAIN1
27   0027   56 3D            JT1     P16     ;16BIT
28   0029   FB      MAIN2:  MOV     A,R3
29   002A   43 80            ORL     A,#80H
30   002C   39                      OUTL    P1,A
31   002D   FA                      MOV     A,R2
32   002E   3A                      OUTL    P2,A
33   002F   14 62            CALL    RXB
34   0031   B6 02            JF0     MAIN
35   0033   90                      MOVX    @R0,A
36   0034   1A                      INC     R2
37   0035   FA                      MOV     A,R2
38   0036   96 39            JNZ     MAIN3
39   0038   1B                      INC     R3
40   0039   EC 29    MAIN3:  DJNZ    R4,MAIN2
41   003B   04 02            JMP     MAIN
42                          ;
43   003D   97        P16:   CLR     C
44   003E   FB              MOV     A,R3
45   003F   67              RRC     A
46   0040   AB              MOV     R3,A
47   0041   FA              MOV     A,R2
```

```
48    0042   67          RRC     A
49    0043   AA          MOV     R2,A
50    0044   F6 4E       JC      RECE
51    0046   14 62  DELE: CALL    RXB
52    0048   B6 02       JF0     MAIN
53    004A   EC 4E       DJNZ    R4,RECE
54    004C   04 02       JMP     MAIN
55    004E   FB     RECE: MOV     A,R3
56    004F   43 80       ORL     A,#80H
57    0051   39          OUTL    P1,A
58    0052   FA          MOV     A,R2
59    0053   3A          OUTL    P2,A
60    0054   14 62       CALL    RXB
61    0056   B6 02       JF0     MAIN
62    0058   90          MOVX    @R0,A
63    0059   1A          INC     R2
64    005A   FA          MOV     A,R2
65    005B   96 5E       JNZ     MAIN4
66    005D   1B          INC     R3
67    005E   EC 46 MAIN4: DJNZ    R4,DELE
68    0060   04 02       JMP     MAIN
69                       ;
70    0062   14 99  RXB:  CALL    RXD
71    0064   B6 98       JF0     RXBX
72    0066   03 BF       ADD     A,#BFH
73    0068   E6 72       JNC     RXB1    ;<41H
74    006A   03 FA       ADD     A,#FAH
75    006C   F6 97       JC      RXBX1   ;>46H
76    006E   53 0F       ANL     A,#0FH
77    0070   04 7A       JMP     RXB2
78    0072   03 07  RXB1: ADD     A,#7
79    0074   F6 97       JC      RXBX1   ;>39H
80    0076   03 0A       ADD     A,#0AH
81    0078   E6 97       JNC     RXBX1   ;<30H
82    007A   47      RXB2: SWAP    A
83    007B   AD          MOV     R5,A
84    007C   14 99       CALL    RXD
85    007E   B6 98       JF0     RXBX
86    0080   03 BF       ADD     A,#BFH
87    0082   E6 8C       JNC     RXB3    ;<41H
88    0084   03 FA       ADD     A,#FAH
89    0086   F6 97       JC      RXBX1   ;>46H
90    0088   53 0F       ANL     A,#0FH
91    008A   04 94       JMP     RXB4
92    008C   03 07  RXB3: ADD     A,#7
93    008E   F6 97       JC      RXBX1   ;>39H
94    0090   03 0A       ADD     A,#0AH
```

```
 95    0092    E6 97           JNC     RXBX1   ;<30H
 96    0094    4D      RXB4:   ORL     A,R5
 97    0095    AD              MOV     R5,A
 98    0096    83              RET
 99    0097    95      RXBX1:  CPL     F0
100    0098    83      RXBX:   RET
101                    ;
102    0099    36 99   RXD:    JT0     RXD     ;WAIT START BIT
103    009B    85              CLR     F0
104    009C    00              NOP
105    009D    BF 07           MOV     R7,#7
106    009F    EF 9F   L1:     DJNZ    R7,L1
107    00A1    36 C1           JT0     RXERR
108    00A3    BE 08           MOV     R6,#8
109    00A5    BF 10           MOV     R7,#10H
110    00A7    EF A7   L2:     DJNZ    R7,L2
111    00A9    77      RXC:    RR      A
112    00AA    00              NOP
113    00AB    36 B1           JT0     RX1
114    00AD    53 7F           ANL     A,#7FH  ;T0 = 0
115    00AF    04 B5           JMP     RX2
116    00B1    43 80   RX1:    ORL     A,#80H  ;T0 = 1
117    00B3    04 B5           JMP     RX2
118    00B5    BF 0E   RX2:    MOV     R7,#0EH
119    00B7    EF B7   L3:     DJNZ    R7,L3
120    00B9    EE A9           DJNZ    R6,RX0
121    00BB    00              NOP
122    00BC    00              NOP
123    00BD    26 C1           JNT0    RXERR   ;CHECK STOP BIT
124    00BF    AF              MOV     R7,A
125    00C0    83              RET
126    00C1    95      RXERR:  CPL     F0
127    00C2    26 C2   L4:     JNT0    L4
128    00C4    83              RET
129                    ;
130    00C5            END
```