逢 甲 大 學

資訊工程系專題報告

使用 8051 製作 LCD 面板之三維動畫

學生：周日新 (87 級丙班)
指導教授：陳啟鏘 老師

中華民國九十一年四月

摘要

這篇報告主要內容包括, 8051 應用系統的開發環境, 8051 的架構, 繪圖型 LCD 模組的架構和基本的三維繪圖方法. 市面上一般書籍中有關繪圖型 LCD 模組的資料極少. 使用者必須花費許多時間來尋找資料, 十分浪費時間. 報告中對於繪圖型 LCD 模組有詳細的說明, 並附有完整的原始程式. 另外對於 8051 應用系統的開發環境也有簡要的介紹, 並且寫下使用上的經驗. 相信能夠讓想要開發 8051 應用系統的人減少許多不必要時間.

目錄

圖表目錄

第一章

第二章

第三章

第四章

第一章 導論

1.1 動機

在大三上的時候, 有修習 MCS-51 的相關課程, 並且做了許多小實驗. 當時就對於硬體或韌體產生興趣. 後來做專題便想說以 8051 為控制器作一個相關應用, 並學習一些新的開發工具. 繪圖型 LCD 模組是 8051 應用中較為複雜的一個應用, 許多相關書籍並沒有詳細說明. 我的專題就是使用 T6963C 為控制器的 LCD 模組作為輸出顯示. 另外, 因為 8051 是 8-bit 的微控制器, 不具有浮點運算功能, 在專題中我使用軟體方式作浮點運算, 並對於這種方式作一個結論.

1.2 目的

做這個專題的主要目的為熟悉 8051 和繪圖型 LCD 模組的架構, 8051 開發工具的使用, 如硬體模擬器(ICE, InCircuit Emulator), 8051 的整合式開發平台(Keil 8051 IDE), 並對熟悉整個開發流程. 另外, 以 C 語言內所提供的浮點運算程式庫作一個基本的 3D 成像引擎. 並繪製一個正立方體於 LCD 面板上, 並讓其產生旋轉的動態效果. 整個專題大部分使用 8051 組合語言, 少部分使用 C 語言完成.

第二章 8051 的架構

2.1 簡介

8051 是 Intel 所開發的一款 8-bits 微控制器(Micro Controller), 也是 MCS-51 族系的始祖. 8051 晶片內部整合了 CPU core, ROM, RAM, I/O units, 屬於單晶片微電腦[1].

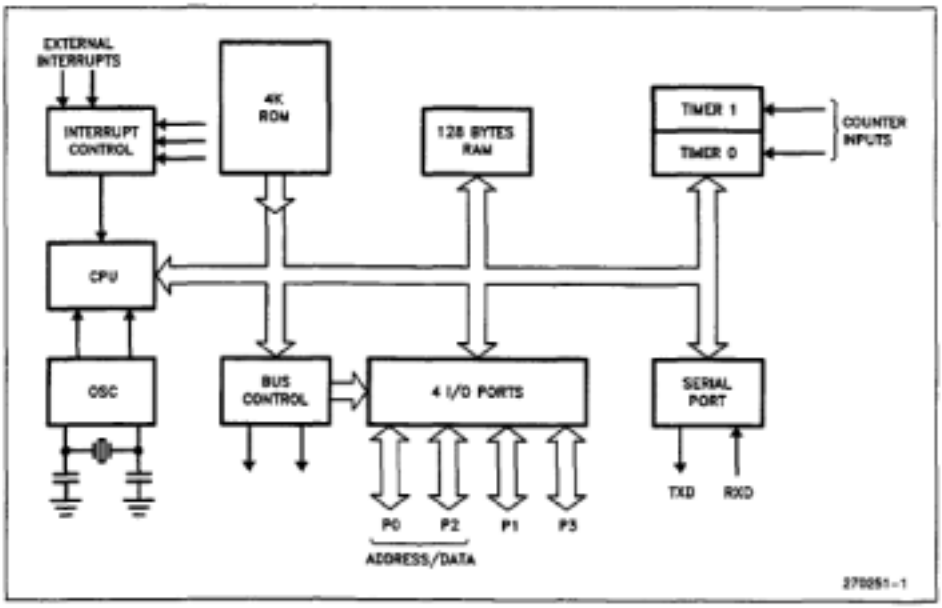

圖 2-1-1 8051 單晶片

2.2 Interface pins

8051 的 IC 封包形式 DIP 和 PLCC 兩種, 其中 DIP 是較為常見的形式. 8051 的對外接腳共有 40 pins. 各 pin 的定義如下表:

pin	代號	用途
40	Vcc	+5V 電源供應
20	Vss	接地
39-32	P0.0-P0.7	I/O Port 0 or A/D Bus
1-8	P1.0-P1.7	I/O Prot 1
21-28	P2.0-P2.7	I/O Port 2 or A/D Bus
10-17	P3.0-P3.7	I/O Port 3
9	RST	Reset
18,19	XTAL1,XTAL2	石英震盪器輸入
29	PSEN	Program Store Enable
30	ALE/PROG	Address Latch Enable
31	EA/VDD	External Access Enable

表 2-2-1　8051 介面接腳

Port 0 和 Port 2 具有多種用途, 這兩個 Port 可以當作一般的 I/O Port 使用, 也可以作為外部記憶體的 Address/Data Bus. Port 3 除了可作為一般的 I/O Port 以外, 也作為外部記憶體存取控制信號, 串列阜, Timer, 和 External Interrupt. 這些用途會在後面 I/O Prot 的小節詳述.

2.3 記憶體組織

8051 具有獨立的 Code Memory 和 Data Memory. 其中 4k 的 Code Memory (ROM)和 128byte 的 Data Memory (SRAM)是 on-chip, 其他的記憶體必須外接. 圖 2-3-1 說明 8051 的 Code Memory 結構, 圖 2-3-2 說明 8051 的 Data Memory 結構[1].

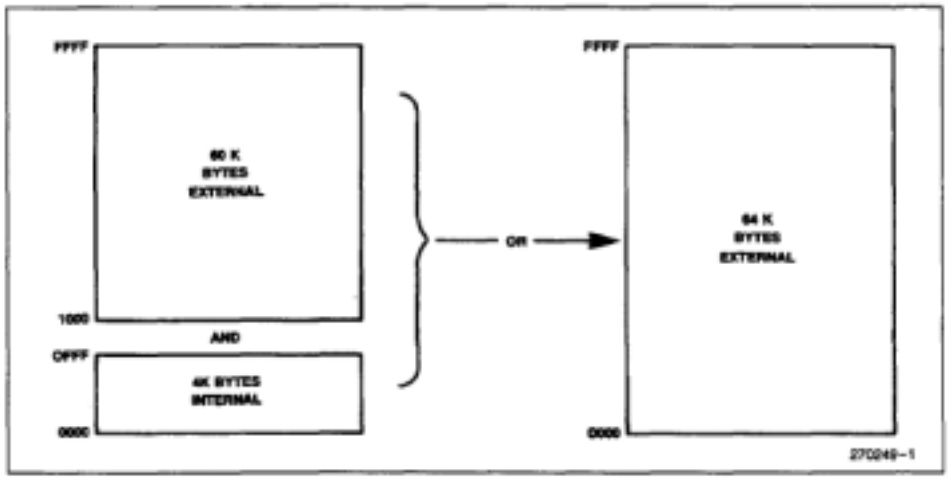

圖 2-3-1 8051 Code Memory

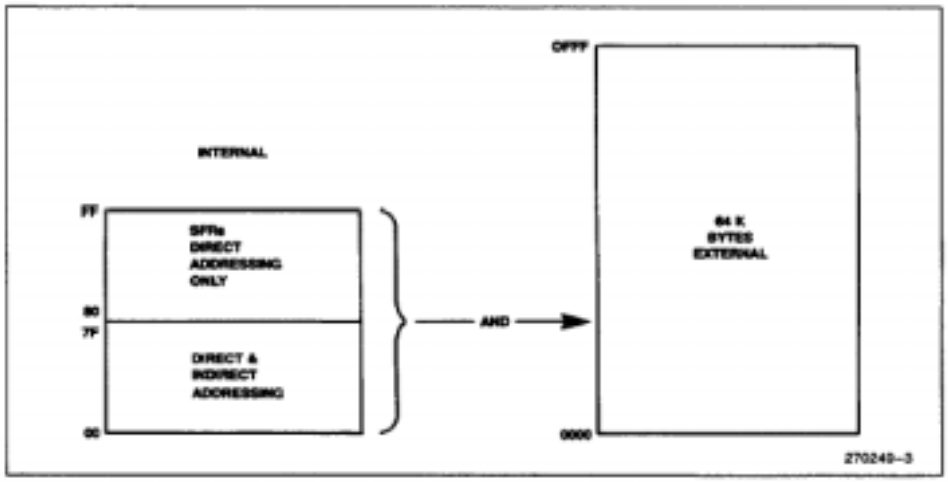

圖 2-3-2 8051 Data Memory

對於 Code Memory 而言, 當位置在 4k 以下時, 8051 會直接由內部的 ROM 抓取資料. 如果位置超過 4k, 則會利用到 Port0 和 Port2 作為 16-bit 的 Data/Address Bus, 並利用 ALE, PSEN 作為控制訊號, 進行外部 Code Memory 讀取.

對於 Data Memory 而言, 內部記憶體和外部記憶體是獨立並存的. 8051 經由直接定址模式(direct addressing)可存取內部資料記憶體, 經由間接定址模式(indirect addressing)可存取外部資料記憶體. 對外部資料記憶體存取時會利用到 Port0 和 Port2 作為 16-bit 的 Data/Address Bus, 並利用 ALE, WR, RD 作為控制訊號, 進行外部 Code Memory 讀取或寫入(視 WR, RD 而定).

2.4 暫存器

8051 的暫存器可分為一般用途暫存器(General Purpose Registers, GPRs)和特殊用途暫存器(Special Function Registers, SFRs). 所有的暫存器皆直接對應到內部資料記憶體, 存取暫存器及視同存取內部資料記憶體. 圖 2-4-1 說明 8051 的暫存器位置[1].

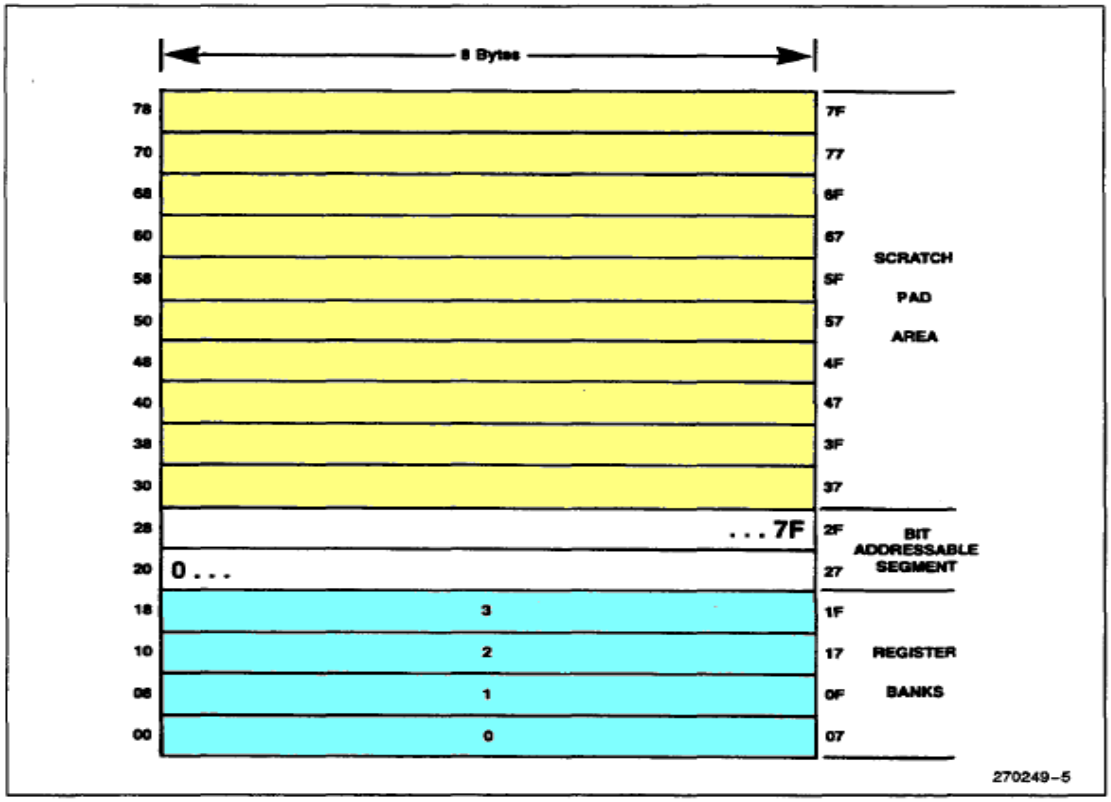

圖 2-4-1　8051 Registers and internal RAM

8051 共有 32 個一般用途暫存器, 這些暫存器被分成 4 個位 Register Bank, 每個 Register Bank 內有 8 個 General Purpose Register (R0-R7), 8051 只能夠存取正在使用的 Register Bank 內的暫存器, 如果要存取其他的一用途暫存器必須先切換 Register Bank.

特殊用途暫存器決定 8051 目前的系統狀態及運作模式, 表 2-4-1 說明這些暫存器的功能及位置.

Symbol	Name	Address
*ACC	Accumulator	0E0H
*B	B Register	0F0H
*PSW	Program Status Word	0D0H
SP	Stack Pointer	81H
DPTR	Data Pointer 2 Bytes	
DPL	Low Byte	82H
DPH	High Byte	83H
*P0	Port 0	80H
*P1	Port 1	90H
*P2	Port 2	0A0H
*P3	Port 3	0B0H
*IP	Interrupt Priority Control	0B8H
*IE	Interrupt Enable Control	0A8H
TMOD	Timer/Counter Mode Control	89H
*TCON	Timer/Counter Control	88H
*+T2CON	Timer/Counter 2 Control	0C8H
TH0	Timer/Counter 0 High Byte	8CH
TL0	Timer/Counter 0 Low Byte	8AH
TH1	Timer/Counter 1 High Byte	8DH
TL1	Timer/Counter 1 Low Byte	8BH
+TH2	Timer/Counter 2 High Byte	0CDH
+TL2	Timer/Counter 2 Low Byte	0CCH
+RCAP2H	T/C 2 Capture Reg. High Byte	0CBH
+RCAP2L	T/C 2 Capture Reg. Low Byte	0CAH
*SCON	Serial Control	98H
SBUF	Serial Data Buffer	99H
PCON	Power Control	87H

表 2-4-1　8051 SFRs

2.5　並列 I/O Port

8051 具有 4 個 8-bit 的雙向 I/O Port (P0-P3), 這些 Port 同時歸屬於 SFRs. 除了 P1 外, 其他的 I/0 Port 都具有多種用途.　圖 2-5-1 說明 8051 I/O Port 的硬體結構[1].

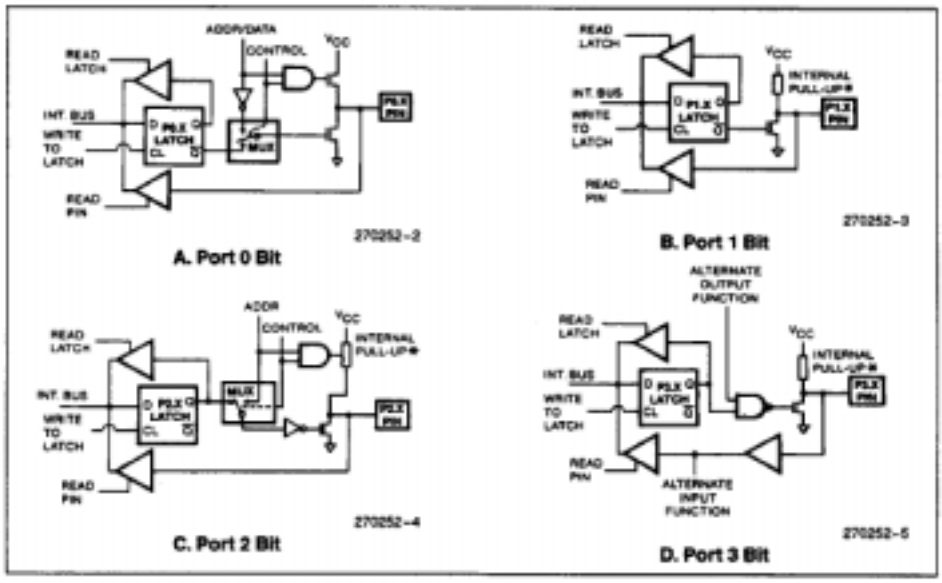

圖 2-5-1　8051 I/O Port

Port0 本身不具有提升電阻, 如果要使用的話, 必須在晶片外面連接一個提升電阻, 以確保訊號的正確性. Port0 當作 Data/Address Bus 使用時, 不需要經過特殊的設定, 只需利用 MOVX 指令就可以在 8051 與外部記憶體之間傳送資料. Port0 當作輸出阜使用時, 則是利用 MOV 指令將資料送到 P0 這個暫存器裡, 外部的接腳就能送出資料. 但如果做為輸入用的話必須先利用指令 MOV P0,#FFH 將 Port 內部的兩個 FET 斷開(open), 然後才能夠正確的接收外部輸入.

Port1 只能夠作為一般的 I/O Port 使用, 內部皆有提升電阻, 所以直接可以做輸入輸出使用. 作為輸出用時只需要利用 MOV 指令即可, 做為輸入使用則需要先送 MOV P2,#FFH 指令將內部的 FET

斷開, 才能夠正確的接收外部輸入.

Port2 本身具有提升電阻, 所以直接可以做輸入輸出使用. Port2 在其他方面和 Port0 是一樣的, 就不重複敘述了.

Port3 本身具有提升電阻, 所以直接可以做輸入輸出使用. Port3 除了當作一般 I/O Port, 還有許多用途, 表 2-5-1 列出這些功能. Port3 作為一般輸入輸出使用時和 Port1 相同.

pin	function
P3.0/RxD	Serial input pin
P3.1/TxD	Serial output pin
P3.2/INT0	External interrupt 0
P3.3/INT1	External interrupt 1
P3.4/T0	Timmer/Counter 0 external input
P3.5/T1	Timmer/Counter 1 external input
P3.6/WR	External memory write strobe
P3.7/RD	External memory read strobe

表 2-5-1 Port3 functions

2.6 內部時序

8051 的機器週期(mechine cycle)可分為 6 個狀態(state), 每個狀態又可分為兩個相位(phase), 每個相位就是一個方波輸入. 所以每個機器週期共有 6 (stage) * 2 (phase) = 12 個方波輸入. 8051 外接一個 12MHz 的時英震盪器作為方波產生器, 所以一個機器週期所消耗的時間為 12*1/12MHz = 1 μs. 8051 的內部時序決定了其 CPU

core 的運算速度和 I/O Port 訊號的更新速度. 因為 I/O Port 連接到 LCD 模組的介面匯流排, 所以在設計 LCD 模組控制介面時, 必須分析 8051 的內部時序. 圖 2-6-1 說明了 8051 的內部時序, 圖 4-3-3 說明了 8051 的 I/O Port 時序[1].

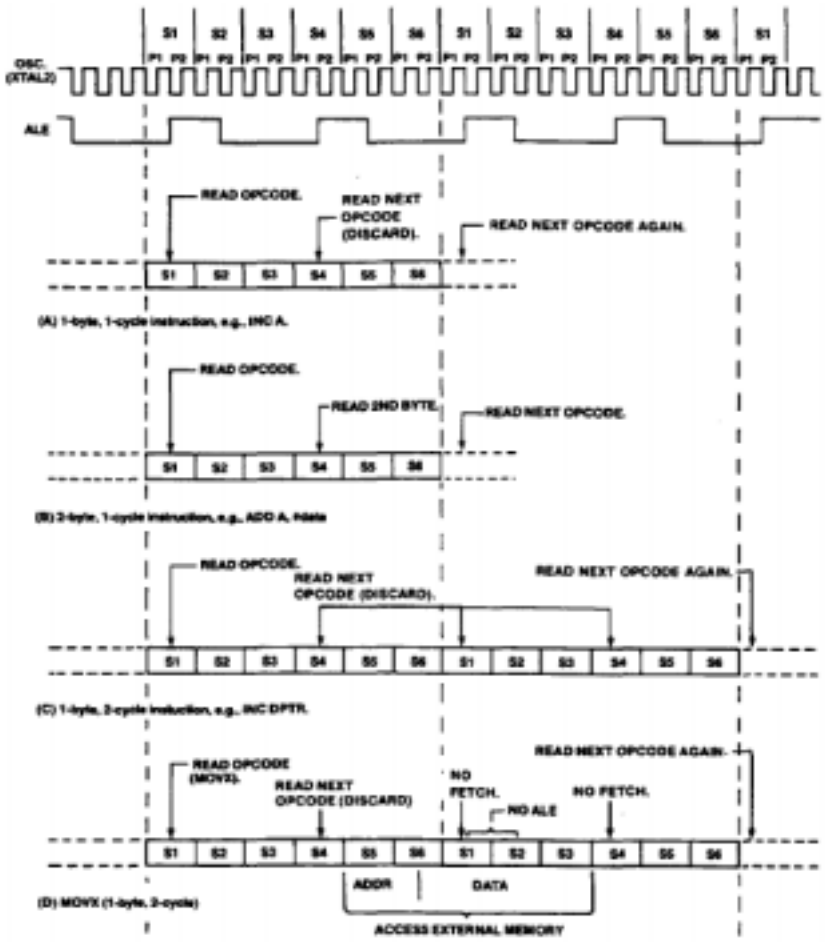

圖 2-6-1　8051 Internal timing

第三章　開發環境簡介

3.1　整體工作流程

早期開發 8051 單晶片系統或其他的嵌入式系統(Enbadded System)時, 因為各種工具與開發環境並不像當今這麼完整. 尤其是對於除錯方面更是欠缺, 所以開發一個系統是十分費時費力的. 現在各種工具與開發環境完整, 幾乎所有的開發工作都可藉由各種工具輔助完成, 不需要真正去燒錄 8051 的 IC, 就能個藉由模擬的方式完成開發的工作. 下圖說明 8051 應用系統的開發流程.

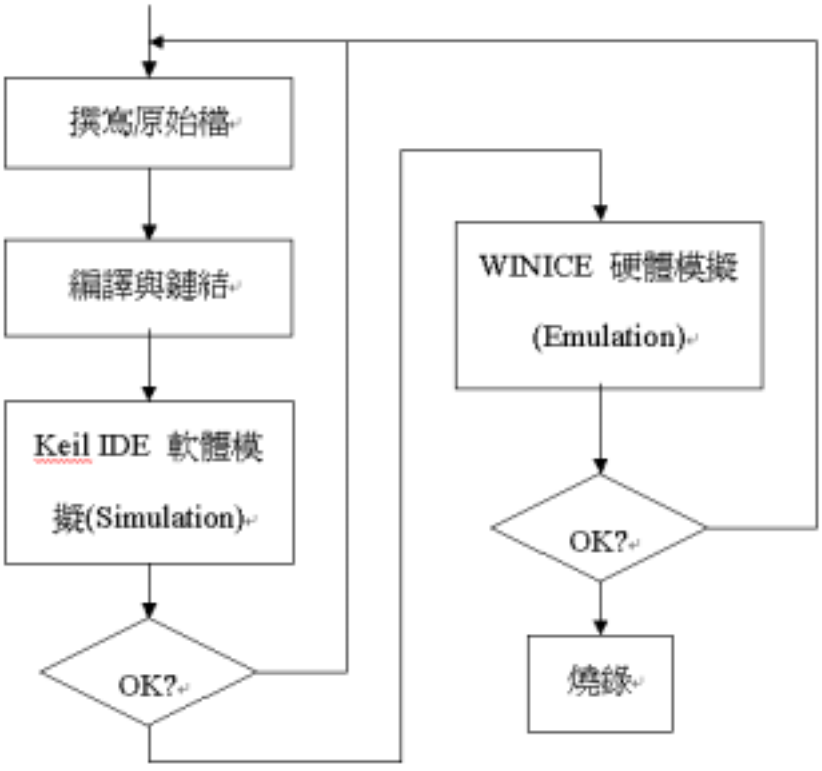

圖 3-1-1　8051 應用系統開發流程

軟體模擬是 Keil μ51 IDE 這套開發工具十分強大的一項功能. 這項功能主要是提供開發者對於整個應用系統作除做與追蹤的工

作. 但只能做到邏輯層面的模擬與除錯, 並不能表現出硬體實際的特性. 例如週邊 I/O 裝置的狀態, 和介面的時序等.

硬體模擬的原理是利用一個仿真的 8051 電路再外加一些具有除錯功能的電路, 並藉由此模擬真正的 8051 和支援額外的除錯功能. 利用 WINICE 作除錯可以完整的模擬出 8051 的硬體特性. 一般而言, 程式編譯完成時都會先用 Keil μ51 IDE 內附的模擬功能作高階除錯的工作, 先將程式中邏輯性的錯誤除去, 在轉換成燒錄檔(HEX)並載入 WINICE 內進行整個系統的模擬和除錯, 以修正和硬體有關的錯誤.

3.2 Keil μ51 IDE 使用方法

Keil μ51 IDE 是一套 8051 應用系統的整合是開發工具, 主要功能包括專案管理, 編譯, 軟體模擬除錯等. 下面介紹使用方法. Keil 這套工具是以專案(project)為編譯和鏈結的單位, 如果不使用專案來管理原始檔的話, 編譯出來的程式就無法正確的執行. 建立新的專案時需要將目的裝置(target device)這定在 8051AH, 如下圖.

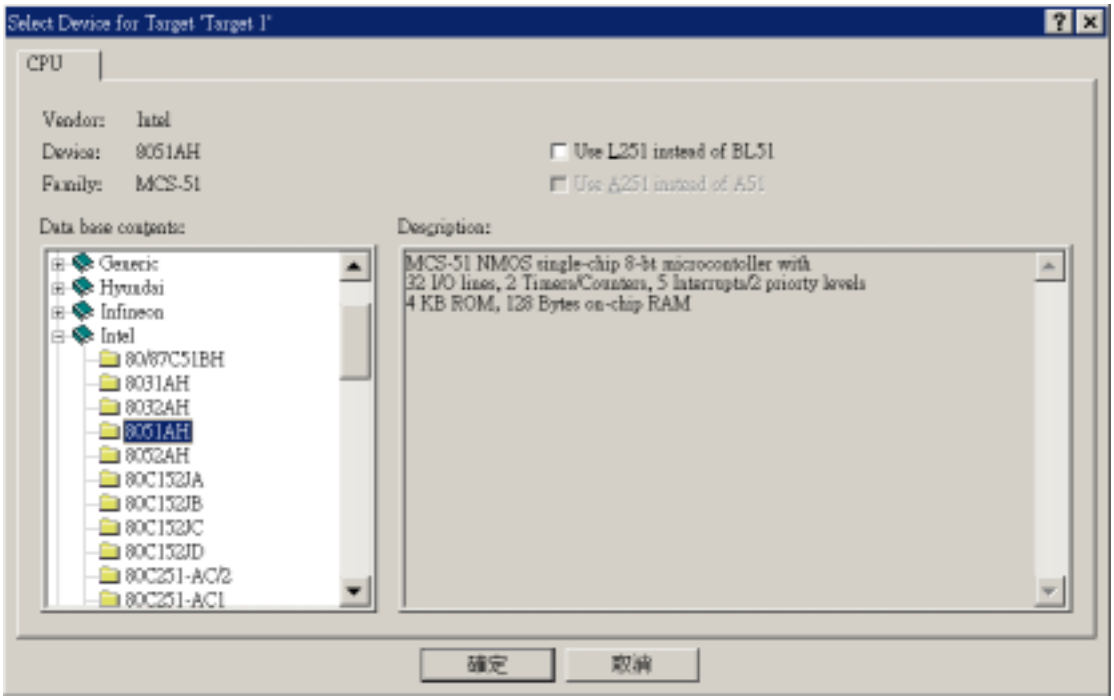

圖 3-2-1　設定目的裝置

將目的裝置設定好之後就可以進行撰寫程式碼的工作, 等到編譯完成時, 就要進行軟體模擬除錯.

進行模擬除錯時的畫面如圖 3-2-2. I/O Port 的內容並不是真正經由外部裝置傳送到 8051 的資料. 使用者必須以手動方式設定 I/O Port 的值來觀察程式是否能正確的處理. Timer 也是必須靠手動設定才能測試程式是否正確. 如果是開發中斷服務程式(Interrupt Serves Routine, ISR)的話, 也是必須利用手動設定來對中斷服務程式來除錯. 圖上並沒有將中斷的功能打開, 所以看不到.

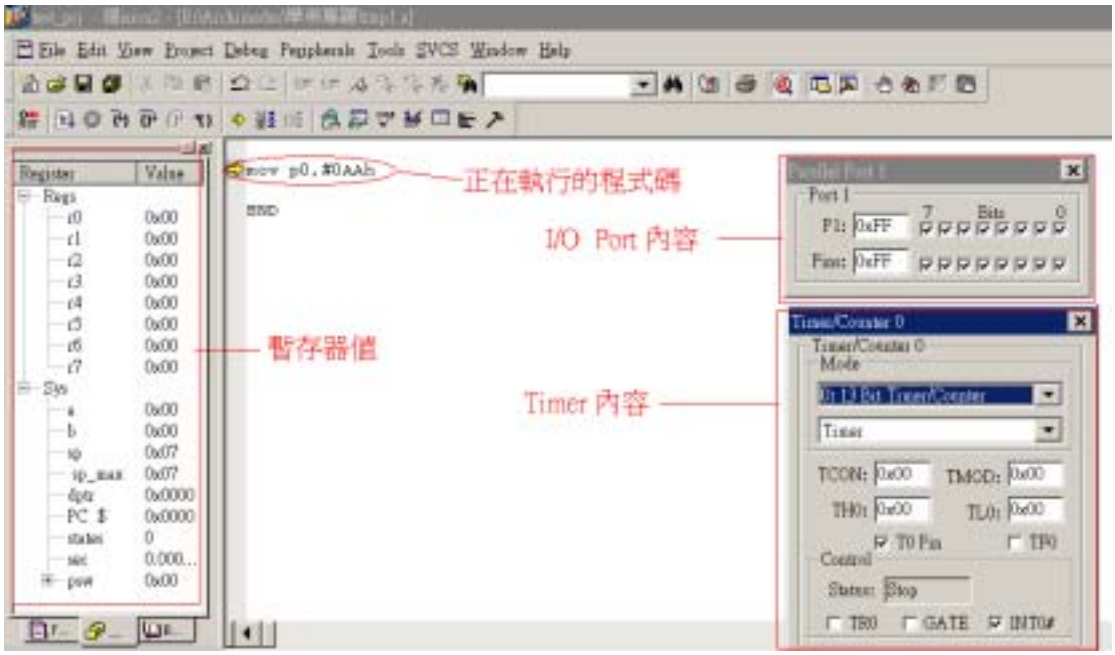

圖 3-2-2　Keil IDE 軟體模擬功能

3.3　WinICE 8051 模擬器使用方法

WinICE 是一套 8051 的硬體模擬器(In Circuit Emulator, ICE), 在使用上可以完全的視做真正的 8051 單晶片. 下圖為 WinICE 進行模擬時的連接方式. 其中 40pin 排線插座可以更換為 IC 燒錄用的活動插槽, WinICE 就可以進行 8051 IC 燒錄的工作. 另外 WinICE 有兩個除錯用的連線接頭(圖中沒畫出)可以視情況使用.

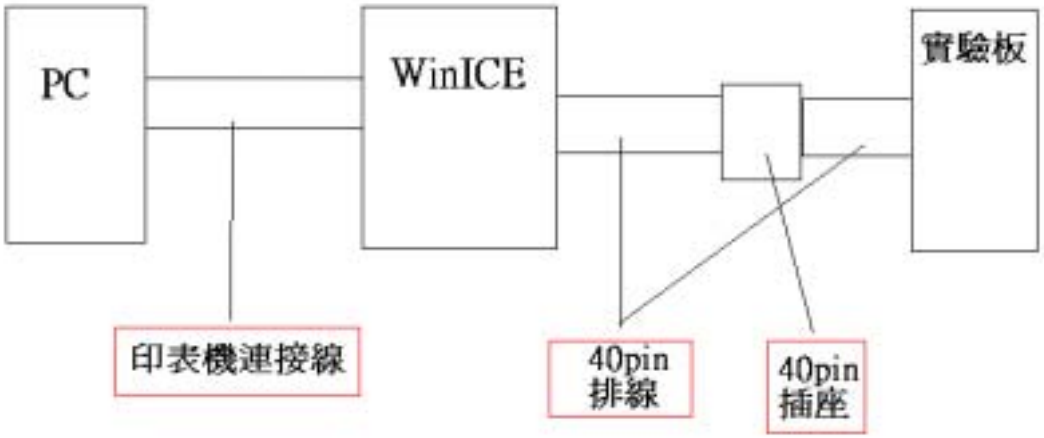

圖 3-3-1　WinICE 的連接方式

WinICE 內部具有 64K 的外部資料記憶體和 64K 的外部程式記憶體, 所以在使用時不可以再外接任何的記憶體模組. 另外 Port0 和 Port2 因同時作為外部記憶體的資料/位置匯流排. 所以在使用這兩個 Port 時必須考慮 WinICE 會隱含性的使用這兩個 Port, 不然會使外部記憶體無法正常運作.

第四章 繪圖型 LCD 模組

4.1 簡介

LCD 模組可分為繪圖型和文字型兩種, 文字型 LCD 的結構較為簡單, 面板較小, 功能非常有限. 繪圖型 LCD 一般而言都能夠同時顯示文字和圖形, 功能較文字型 LCD 強大許多. 市面上能夠買到的 LCD 模組通常是使用 T6963C LCD controller. 我的專題是以 240*128 點的繪圖型 LCD 模組(WG240128B)作為顯示輸出裝置.

我所使用的 LCD 模組的基本構造如下圖[2].

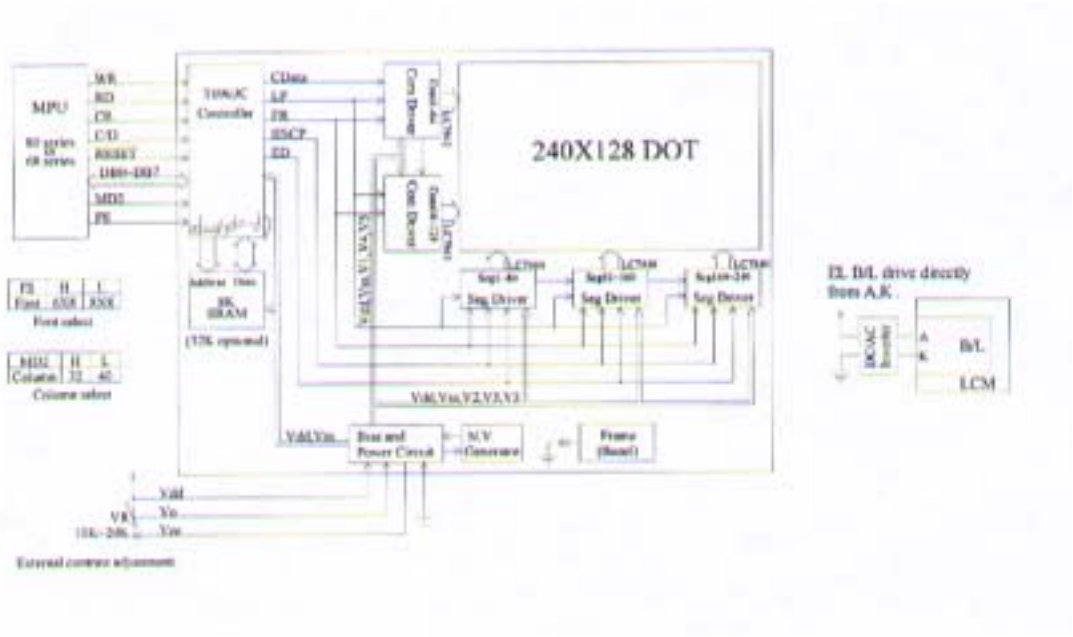

圖 4-4-1 LCD 模組架構

LCD 模組的基本工作原理為, 利用 T6963C 作為控制晶片, 根據 LCD 模組上顯示記憶體的內容, 不斷的將資料掃描顯示到 LCD 面板上. T6963C 同時能夠和外部的控制器如 8051 之間溝通, 或傳

送資料.

4.2 LCD 模組介面接腳

表 4-2-1 說明了我所使用的繪圖型 LCD 模組的介面接腳定義[2].

Pin NO	Symbol	Function
1	Vss	Power supply (GND)
2	Vdd	Power supply (+5V)
3	Vo	Power supply for LCD driving
4	C/D	Command/data read/write
5	$\overline{RD}$	Data read
6	$\overline{WR}$	Data write
7	DB0	Data bus line
8	DB1	Data bus line
9	DB2	Data bus line
10	DB3	Data bus line
11	DB4	Data bus line
12	DB5	Data bus line
13	DB6	Data bus line
14	DB7	Data bus line
15	$\overline{CE}$	Chip enable
16	$\overline{RESET}$	Reset signal
17	Vee	Negative Voltage
18	MD2	Control signal
19	FS1	Font selection
20	NC	No connection

表 4-2-1 LCD 模組 Interface pins

4.3 LCD 模組介面時序

LCD 模組和 8051 之間要能夠正確的溝通, 必須遵守一定的時序, 8051 的內部時序和 LCD 模組的介面時序都要考慮. 撰寫 LCD 模組 driver 時必須詳細考慮上述兩點, 不然無法正確的控制 LCD 模

組, 而且會造成 debug 的困難. 圖 4-3-1 和表 4-3-1 說明了 LCD 模組的介面時序[2].

- Switching Characteristics (2)

Bus Timing

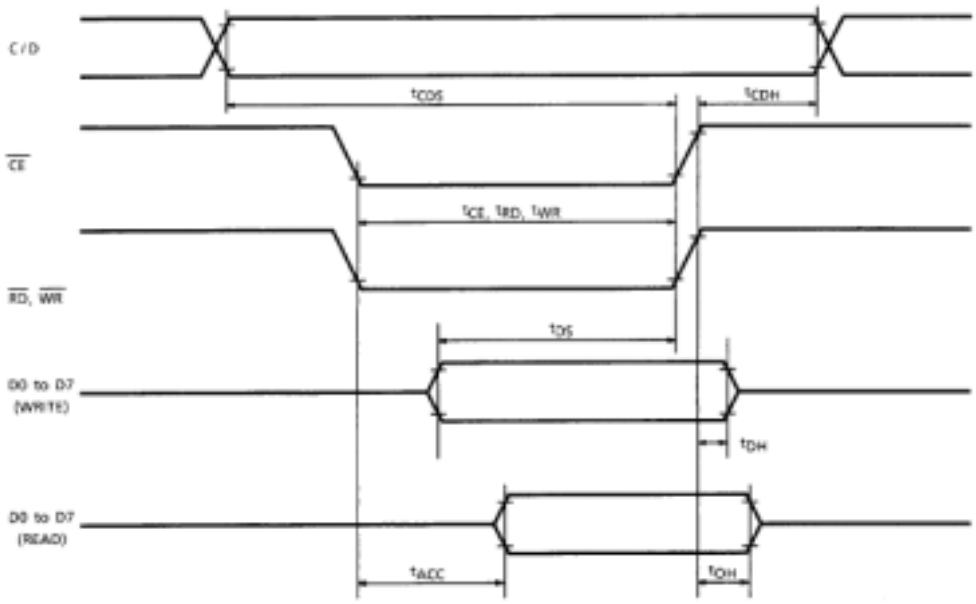

圖 4-3-1　LCD 模組介面時序圖

TEST CONDITIONS (Unless otherwise noted, V_{DD} = 5.0V ± 10%, V_{SS} = 0V, Ta = − 20 to 75°C)

ITEM	SYMBOL	TEST CONDITIONS	MIN	MAX	UNIT
C/D Set-up Time	t_{CDS}	—	100	—	ns
C/D Hold Time	t_{CDH}	—	10	—	ns
CE, RD, WR Pulse Width	t_{CE}, t_{RD}, t_{WR}	—	80	—	ns
Data Set-up Time	t_{DS}	—	80	—	ns
Data Hold Time	t_{DH}	—	40	—	ns
Access Time	t_{ACC}	—	—	150	ns
Output Hold Time	t_{OH}	—	10	50	ns

表 4-3-1　LCD 模組介面時序參數

從上面圖表可以得知, LCD 模組在讀取 LCD 模組內的記憶體時需要存取時間(Access Time,tACC)為 150 ns 和改變讀寫訊號 RD/WR 時, 正確資料保持在 BUS 上的時間(Output Hold Time, tOH)

為 50 ns. 其他的訊號只需要保持足夠的時間就可以正確的運作. 例如 CE 訊號最少需持續 80 ns, C/D 訊號最少需持續 110 ns (tCDS+tCDH).

8051 執行一道指令需要 1μs 或 2μs, 所以在撰寫 LCD 模組 Driver 時不需要用到等待指令(NOP), 只要遵守 LCD 介面訊號的順序就可以了. 值得一提的是, 如果使用較快速的系統例如 PCI BUS, 或更快的 CPU 來控制 LCD 模組, 則必須適當的插入等待指令(NOP)來調整介面訊號的時序.

下面列出 8051 和 LCD 模組寫入資料(Data Write)與命令(Command Write)的程式片段, 讀取資料(Data Read)和寫入資料類似, 就不重複說明了. 送出 I/O 訊號的順序請參考圖 3-3-1, 訊號內容請參考表 3-2-1.

```
WRITE_COMMAND:
        LCALL   SUB_CHECK_COMMAND_EXECUTION    ;
                檢查 T6963C 是否 Ready?
    SETB    CD          ; 設定 Data/Command bit
    SETB    _RD         ; 設定 Read/Write bit0
        CLR     _CE     ; Enable T6963C data bus
    CLR     _WR         ; 清除 Read/Write bit1
    MOV     _DATA,A     ; 將欲送出的資料放在 I/O Port 上
    SETB    _CE         ; Disable T6963C data bus
    SETB    _WR         ; 設定 Read/Write bit1
RET

WRITE_DATA:
```

```
        LCALL  SUB_CHECK_COMMAND_EXECUTION    ;
                                ; 檢查 T6963C 是否 Ready?

        CLR     _CD             ; 清除 Data/Command bit
        SETB    _RD             ; 設定 Read/Write bit0
        CLR     _CE             ; Enable T6963C data bus
        CLR     _WR             ; 清除 Read/Write bit1
        MOV     _DATA,A         ; 將欲送出的指令放在 I/O Port 上
        SETB    _CE             ; Disable T6963C data bus
        SETB    _WR             ; 設定 Read/Write bit1
    RET
```

註: SUB_CHECK_COMMAND_EXECUTION 為 LCD 模組狀態檢查的副程式, 請參考 4.4 節 .

4.4 LCD 狀態檢查

在對 LCD 傳送資料或指令之前, 必須先檢查 T6963C 的狀態是否就緒. 檢查狀態的時必須先將 RD 設為 0(低電位),WR 設為 1(高電位), 再將 CE 設為 0, 最後將 C/D 設為 1. 下一個 8051 的指令週期(instruction cycle)就可以由 LCD 的 Data Bus 讀取到 T6963C 的狀態. 讀取到 T6963C 的狀態之後, 必須檢查其中幾個 bit 來判斷其狀態. 圖 4-4-2 說明各個 bit 所代表的意義. 一般模式下, 只會檢查 bit0(STA0)和 bit1(STA1). 如果這兩個 bit 都為 1, 則代表 T6963C 已經就緒準備傳送資料. 除此之外, 必須不斷的檢查其狀態, 也就是做忙碌等待(busy waiting)的動作. 圖 4-4-3 為狀態檢查的流程圖[2].

The T6963C status word format is as follows:

MSB							LSB
STA7 D7	STA6 D6	STA5 D5	STA4 D4	STA3 D3	STA2 D2	STA1 D1	STA0 D0

STA0	Check command execution capability	0 : Disable 1 : Enable
STA1	Check data read / write capability	0 : Disable 1 : Enable
STA2	Check Auto mode data read capability	0 : Disable 1 : Enable
STA3	Check Auto mode data write capability	0 : Disable 1 : Enable
STA4	Not used	
STA5	Check controller operation capability	0 : Disable 1 : Enable
STA6	Error flag. Used for Screen Peek and Screen copy commands.	0 : No error 1 : Error
STA7	Check the blink condition	0 : Display off 1 : Normal display

(Note 1) It is necessary to check STA0 and STA1 at the same time. There is a possibility of erroneous operation due to a hardware interrupt.
(Note 2) For most modes STA0 / STA1 are used as a status check.
(Note 3) STA2 and STA3 are valid in Auto mode; STA0 and STA1 are invalid.

圖 4-4-2　T6963C 狀態定義

Status checking flow

a)

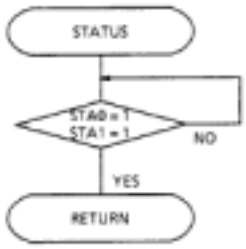

圖 4-4-3　狀態檢查流程圖

這一段程式是做 LCD 模組狀態檢查的程式:

SUB_CHECK_COMMAND_EXECUTION:

LOOP_BUSY_WAITING:

```
MOV     R0,A
SETB    _WR             ;
SETB    _CD             ;
MOV     _DATA,#0FFH     ; 設定 I/O Port 為 Read
CLR     _CE             ; Enable T6963C
CLR     _RD             ;
MOV     A,_DATA         ; 將 LCD 模組狀態搬到 A
SETB    _CE             ; Disable T6963C
ANL     A,#03H          ; 檢查第 0,1 個 bit
SUBB    A,#03H          ; 判斷狀態是否 Ready?
JC   LOOP_BUSY_WAITING
MOV     A,R0

RET
```

4.5 LCD 指令集

8051 必須使用 T6963C 的指令集來操作 LCD 模組的運作. T6963C 的指令格式分為兩類, 一種是具有 1byte 資料的指令, 另一種是具有 2byte 資料的指令. 在送資料之前必須先檢查 T6963C 的狀態(check statu)是否就緒, 然後再傳送資料. 最後再傳送指令(command). 圖 4-5-1 說明這兩種形式的指令格式 [2].

圖 4-5-1　LCD 模組指令格式

依據指令的功能, 可將 LCD 指令集分為三大類, 即暫存器設定指令, 顯示記憶體讀寫指令, 和 LCD 自動讀寫模式(Auto read/write mode)設定指令. 表 4-5-1 列出所有的 LCD 模組指令[2].

COMMAND DEFINITIONS

COMMAND	CODE	D1	D2	FUNCTION
REGISTERS SETTING	00100001	X address	Y address	Set Cursor Pointer
	00100010	Data	00H	Set Offset Register
	00100100	Low address	High address	Set Address Pointer
SET CONTROL WORD	01000000	Low address	High address	Set Text Home Address
	01000001	Columns	00H	Set Text Area
	01000010	Low address	High address	Set Graphic Home Address
	01000011	Columns	00H	Set Graphic Area
MODE SET	1000X000	—	—	OR mode
	1000X001	—	—	EXOR mode
	1000X011	—	—	AND mode
	1000X100	—	—	Text Attribute mode
	10000XXX	—	—	Internal CG ROM mode
	10001XXX	—	—	External CG RAM mode
DISPLAY MODE	10010000	—	—	Display off
	1001XX10	—	—	Cursor on, blink off
	1001XX11	—	—	Cursor on, blink on
	100101XX	—	—	Text on, graphic off
	100110XX	—	—	Text off, graphic on
	100111XX	—	—	Text on, graphic on
CURSOR PATTERN SELECT	10100000	—	—	1-line cursor
	10100001	—	—	2-line cursor
	10100010	—	—	3-line cursor
	10100011	—	—	4-line cursor
	10100100	—	—	5-line cursor
	10100101	—	—	6-line cursor
	10100110	—	—	7-line cursor
	10100111	—	—	8-line cursor
DATA AUTO READ / WRITE	10110000	—	—	Set Data Auto Write
	10110001	—	—	Set Data Auto Read
	10110010	—	—	Auto Reset
DATA AUTO READ / WRITE	10110000	—	—	Set Data Auto Write
	10110001	—	—	Set Data Auto Read
	10110010	—	—	Auto Reset
DATA READ / WRITE	11000000	Data	—	Data Write and Increment ADP
	11000001	—	—	Data Read and Increment ADP
	11000010	Data	—	Data Write and Decrement ADP
	11000011	—	—	Data Read and Decrement ADP
	11000100	Data	—	Data Write and Nonvariable ADP
	11000101	—	—	Data Read and Nonvariable ADP
SCREEN PEEK	11100000	—	—	Screen Peek
SCREEN COPY	11101000			Screen Copy
BIT SET / RESET	11110XXX	—	—	Bit Reset
	11111XXX	—	—	Bit Set
	1111X000	—	—	Bit 0 (LSB)
	1111X001	—	—	Bit 1
	1111X010	—	—	Bit 2
	1111X011	—	—	Bit 3
	1111X100	—	—	Bit 4
	1111X101	—	—	Bit 5
	1111X110	—	—	Bit 6
	1111X111	—	—	Bit 7 (MSB)

X : invalid

表 4-5-1　LCD 模組指令集

4.6　顯示記憶體

再使用 LCD 模組做顯示之前, 必須先將模組上的顯示記憶

體規劃好, 並設定相關的暫存器. LCD 面板上的輸出資料是儲存在一個 8K 的靜態隨機存取記憶體作(SRAM)中. 這 8K 的 SRAM 也稱為 LCD 模組的顯示記憶體, 記憶體空間可由使用者自由規劃. 一般建議的規劃方式如下圖[2].

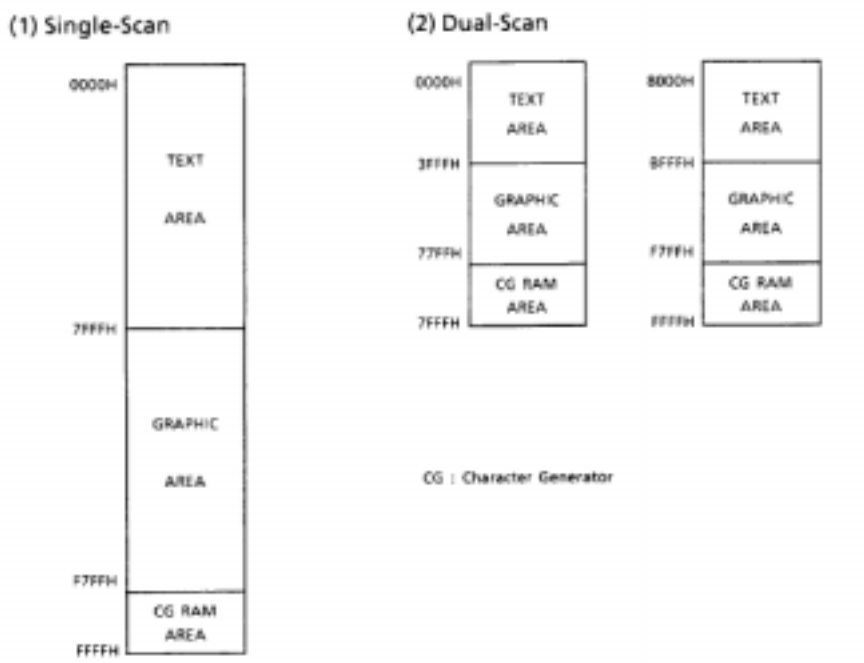

圖 4-6-1　LCD 顯示記憶體

下圖說明, 在文字模式與繪圖模式下, 顯示記憶體與 LCD 面板的對應方式

The relationship between external display RAM address and display position

TH		TH + CL
TH + TA		TH + TA + CL
(TH + TA) + TA		TH + 2TA + CL
(TH + 2TA) + TA		TH + 3TA + CL
TH + (n-1) TA		TH + (n-1) TA + CL

TH : Text home address
TA : Text area number (columns)
CL : Columns are fixed by hardware (pin-programmable).

圖 4-6-2　文字模式下對應方式

The relationship between external display RAM address and display position

GH		GH + CL
GH + GA		GH + GA + CL
(GH + GA) + GA		GH + 2GA + CL
(GH + 2GA) + GA		GH + 3GA + CL
GH + (n-1) GA		GH + (n-1) GA + CL

GH : Graphic home address
GA : Graphic area number (columns)
CL : Columns are fixed by hardware (pin-programmable).

圖 4-6-3　繪圖模式下對應方式

存取顯示記憶體必須透過 LCD 指令集中的 Data Data 和 Data Write 指令. 在使用這兩個指令前必須設定 T6963C 的 Address Pointer. Data Read 和 Data Write 指令會根據 Address Pointer 所指向的位址做記憶體存取的動作. 圖 4-6-4 說明寫入顯示記憶體的流程[2], 讀取的流程與寫入類似.

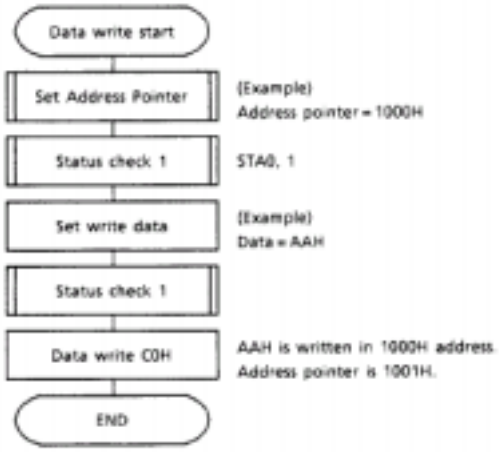

圖 4-6-4　顯示記憶體寫入流程

下面是寫入顯示記憶體的程式片段:

```
SUB_DATA_WRITE_LCD:          ; 寫入顯示記憶體副程式
      CALL    WRITE_DATA     ; 將 A 的內容寫入顯示記憶體
      MOV     A,#0C0H        ; 送 Data Write 指令
      LCALL   WRITE_COMMAND  ;
RET
```

註: WRITE_DATA 和 WRITE_COMMAND 為寫入資料和指令的副程式, 請參考 4.3 節 .

第五章　繪圖原理

5.1　繪圖流程

繪圖的基本流程就是, 先用繪圖引擎產生一個畫面(frame)儲存於 8051 的外部資料記憶體中, 再將這個畫面的資料全部搬移到 LCD 模組上的會顯示記憶體中. 不斷的重複這個步驟, 就會產生動態的效果. 圖 5-1-1 詳述這個流程.

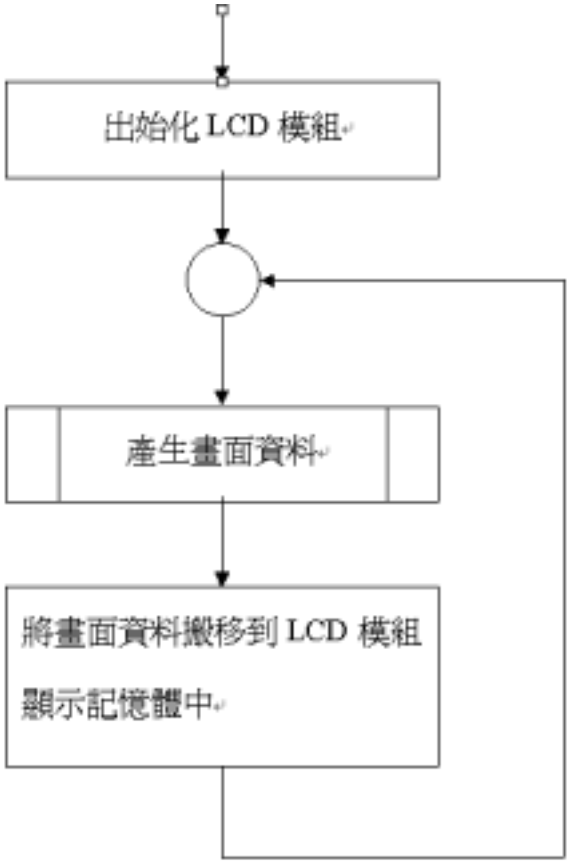

圖 5-5-1　　繪圖流程

產生畫面資料主要包括兩個過程, 首先必須利用3D投影原理進行座標的轉換, 即將 3 為座標轉換為 2 為座標. 接著要在

記憶體中產生畫面(frame), 也就是 在記憶體中繪製多邊形.

5.2 3D 投影與旋轉

在 3 維空間中, 每個點的座標可以用(x,y,z)向量來表示. 但電腦螢幕是 2 維的平面, 如果要將 3 維的物體以 2 維平面表現就必須透過投影的方式表現出 3 維物的真實感. 投影的方式可分為有深度的投影和無深度的投影. 有深度的投影考慮了距離感, 而無深度投影無法表現物體的距離感. 我實作時使用無深度投影法.

這裡先說明無深度投影的原理, 假設空間中有一個點 S(x,y,z), 和任意平面 P, L 為空間中通過 S 且垂直於平面 P 的直線, L 與平面 P 的交點就是點 S 在平面 P 的投影點. 在無深度投影中, 移動投影平面 P 就相當於旋轉被投影的物體. 為了方便起見, 我們固定平面 P 為 xy 平面, 則點 S(x,y,z)的投影為(x,y).

當物體旋轉時, 其座標也會隨著改變. 假設空間中任意點 S(x,y,z), 當點 S 以 x 軸作旋轉角度 θ 時, 選轉後的的座標 $(x' \quad y' \quad z')$可以用以下的式子表示.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$

以 y 軸作旋轉角度 θ 時, 選轉後的的座標$(x' \quad y' \quad z')$可以用以下的式子表示.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \begin{pmatrix} 0 & \cos\theta & \sin\theta \\ 0 & 1 & 0 \\ 0 & -\sin\theta & \cos\theta \end{pmatrix}$$

以 z 軸作旋轉角度 θ 時, 選轉後的的座標$(x' \quad y' \quad z')$可以用以下的式子表示.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

有了對各軸旋轉的公式後, 就能夠讓物體以任意的角度頭在 xy 平面上, 也就是說能夠從任何角度觀看物體.

5.3 產生畫面

產生畫面最基本的動作就是畫點了, 能夠畫點之後畫線段就容易了. 空間中任意點 S(x,y,z)經過投影後會得到(x,y). 下面為在畫面(frame)中畫點(x,y)的程式碼.

```
DRAW_PIXEL:
    MOV     R2,DPH ; X    (x,y)  由暫存器  R2 和 R3 傳入
    MOV     R3,DPL ; Y

    MOV     A,R2      ;  這一段是在計算需  (x,y)  對應到 LCD
    MOV     B,#08H    ;  面板上的行列位置
    DIV AB            ; A=商  B=餘數
    MOV     R4,A      ; Q
    MOV     R5,B      ; R

    MOV     A,R3      ;  這一段在計算 frame 中需要被更新的位置
    MOV     B,#30     ;
    MUL     AB        ; DPTR =  行數  / 8 +  列數  * 30
    ADDC    A,R4      ;
    JC   CARRY
    JMP     NO_CARRY
CARRY:
    INC B
NO_CARRY:
    MOV     DPL,A
    MOV     DPH,B

    MOVX    A,@DPTR     ;  讀取 frame 上原來的資料
    MOV     R6,A      ;

    MOV     A,#07     ;  產生"點"的影像
    SUBB    A,R5
    MOV     R5,A

    MOV     A,#01H

LOOP_SHIFT:
    RL   A
    DJNZ    R5,LOOP_SHIFT

    ORL     A,R6      ;  以 OR 模式更新 frame 資料

    MOVX    @DPTR,A
```

RET

畫線段的動作是由一連串的畫點動作構成的, 我實作時所使用的方法如下. 如果由(x1,y1)畫一線段到(x2,y2), 則先計算出此線的斜率 m, 再依據螢幕的解析度(LCD 面板的解析度), 不斷的由點(x1,y1)靠近(x2,y2), 並畫出路線上所有的點. 下面無畫出線段的 C 程式碼.

```
void draw_line(int xb,int yb,int xe,int ye)
{
    int i;
    int cx,cy;

    x1=xb;
    y1=yb;
    x2=xe;
    y2=ye;
    m=(x2-x1)/(y2-y1);          // 計算斜率   m 為浮點數型態
    p=240.0;
    q=0.0;

    for(i=0;i<Max;i++)          // Max 為螢幕的最大解析度, 在此
                                為 240
    {
        draw_pest(cx,cy);       // 畫點
        q=q+1.0;
        cx=x1+(x2-x1)*q/p;      // 算出下一點的 x 位置
        cy=y1+(y2-y1)*q/p;      // 算出下一點的 y 位置
    }
}
```

5.4　繪製立方體

圖形資料的格式是以點的方式儲存於記憶體中, 每個點都對應著一個 3 維的座標(x,y,z), 3 為空間中的任何物體都能使用點和線段來表示. 專題實作中 LCD 面板上顯示的是一個正立方體. 這個正立方體是以 8 點連接出了 12 條線段來表示的. 下圖正立方體的座標系統和位置.

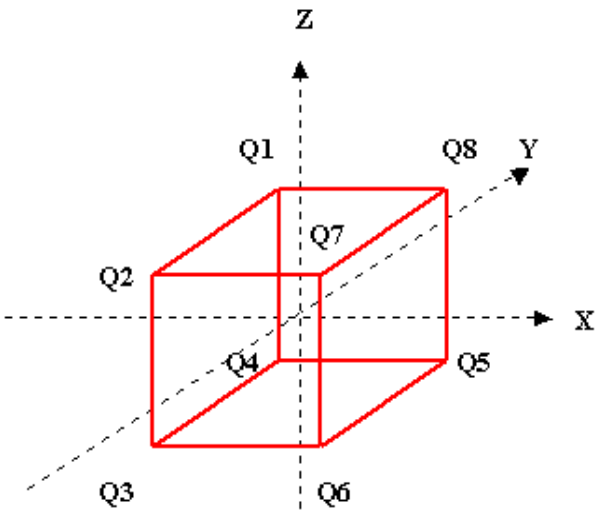

圖 5-4-1　正立方體座標資料

繪製出正立方體其實很容易, 只要畫出這 12 條線段就可以了.

第六章 總結

6.1 問題與改進

這一小節我將開發過程中所遭遇的一些問題和一些改進的想法提出來討論.

開發硬體實最麻煩的就是除錯的問題, 因為不容易分辨是硬體的錯誤還是韌體方面的錯誤. 一開始實作時, 常常就為了找出一點小錯誤而耗費許多時間. 後來我發現了一個不錯的方法可以使除錯的過程更容易, 就是整個程式編譯完後先使用 Keil 的工具作模擬和除錯, 確定沒有問題後再使用 8051 ICE 模擬, 如果有錯, 那大部分都是硬體方面的錯誤. 這樣做的話, 除錯的過程就會比較有效率.

整個專題還有許多可改進的地方, 最主要因為 8051 不具有浮點運算的功能, 所以計算浮點數時非常耗費時間. 如果能夠使用一顆具有浮點運算能力的高速微處理器來做繪圖應用的話, 效能應該會大幅提升. 其他像是 LCD 面板的解析度不夠, 使圖形看起來不夠細膩. 投影可使用較複雜的方式以產生更好的立體效果. 都是值得改進的地方.

附錄

A. 參考資料

[1] MCS 51 Microcontroller Family User's Manual, Intel.

[2] T6963C Manual, TOSHIBA.

[3] MCS 51 單晶片原理與 I/O 應用, 劉銘中, 林進誠.

[4] Using the MCS 51 Microcontroller, Han-Way Huang.

B. 程式列表