國立交通大學

電機與控制工程學系

碩士論文

使用微控制器 *SN8P1708* 晶片實現三相永磁馬達之控制

Realization for Permanent Magnet Synchronous Motor Control by Micro Controller SN8P1708 Chip

研究生：吳宇中

指導教授：林錫寬 博士

中華民國九十四年六月

誌謝

首先，我想感謝我的指導老師林錫寬教授，在研究所這兩年的生涯中，老師給了我很多的意見與指導。此外，老師在學術研究上所抱持的嚴謹態度、豐富的學識一直是我深感值得效法的典範。

其次，非常感謝陳傳生教授、張文中教授和蔡清元教授，在百忙之中來幫我進行論文口試，也感謝各位老師對本論文的建議與指正，以及對我個人的勉勵。感謝博士班李宗原學長以及謝孟勳、張豪揚、邱俊傑和張維娜幾位學長姐在我研究過程中對我的指導與建議，並感謝我的同窗好友威勳、存堯、啓昌、俊杰，與學弟星宇、典瑋、品齊、匯欽，陪伴我在實驗室做研究的日子中，給我的鼓勵和支持，使得我在研究所這兩年獲益良多。

最後，我更要感謝我的家人，他們在這段時間內不曾間斷的鼓勵和關懷，使得我能順利完成論文的撰寫。在此僅以本份論文的結果獻給我的家人與其他關心、幫助我師長及朋友，非常的感謝你們。

國立交通大學

電機與控制工程學系

碩士論文

使用微控制器 *SN8P1708* 晶片實現三相永磁馬達之控制

Realization for Permanent Magnet Synchronous Motor Control by Micro Controller SN8P1708 Chip

研究生：吳宇中

指導教授：林錫寬 博士

中華民國九十四年六月

使用微控制器 SN8P1708 晶片實現三相永磁馬達之控制

Realization for Permanent Magnet Synchronous Motor Control by Micro Controller SN8P1708 Chip

研究生　：　吳宇中　　Student　：　Yu-Zhong Wu

指導教授　：　林錫寬 博士　　Advisor　：　Dr. Shir-Kuan Lin

國立交通大學

電機與控制工程學系

碩士論文

A Thesis

Submitted to Department of

Electrical and Control Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of Master

in

Electrical and Control Engineering

June 2005

Hsinchu， Taiwan， Republic of China

中華民國九十四年六月

使用微控制器 SN8P1708 晶片實現三相永磁馬達之控制

研究生：吳宇中　　指導教授：林錫寬 博士

國立交通大學電機與控制工程學系

摘要

自從馬達被發明出來，經過不斷的改良與創新，現在馬達在工業界已經是一種不可或缺的重要元件，舉凡辦公室裡的電器用品，幾乎都可以看到馬達的存在，像是印表機，掃瞄器，光碟機，硬碟 ... 等。因此，如何控制馬達便成爲一個重要的學問。

本篇論文主要是使用 Sonix 公司所生產的 SN8P1708 晶片來實現控制三相永磁馬達之旋轉。在韌體方面，利用組合語言來實現控制傳輸，傳送控制三相永磁馬達所需的資料。在硬體方面使用六步變頻器 (six-step inverter) 電路來驅動三相永磁馬達，透過霍爾感測器做閉迴路之控制，使馬達旋轉更加穩定，透過鍵盤輸入馬達旋轉之資料並搭配 LCD display 提示，讓使用者更易於操作此系統。

Realization for Permanent Magnet Synchronous Motor Control by Micro Controller SN8P1708 Chip

Student : Yu-Zhong Wu Advisor : Dr. Shir-Kuan Lin

Department of Electrical and Control Engineering
National Chiao Tung University

ABSTRACT

Since the motor is invented, through people's constant improvement and innovation, the motor becoms a kind of indispensable important component in the industry now. We can nearly all see the existence of the motor in the electric products inside all offices, it is like a printer, scanner, CD-ROM driver, hard disk,etc...So, how to control the motor becomes an important knowledge.

This thesis mainly describes the realization for permanent magnet synchronous motor control by SN8P1708 chip manufactured by Sonix. In firmware, we use assembler codes to relaize control transfer between host and device. In hard ware, we drive permanent magnet synchronous motor by six-step inverter circuit. It's stabilize to motor running by close-loop control used with hall sensor. It is more easy to operate this system by used with keyboard and LCD display.

目錄

圖例目錄

表格目錄

第一章

緒論

1.1 研究動機與目的

微控器應用已經是無所不在。日常生活中的電子產品諸如微波爐、電子鍋、電熱水壺、各項電器的遙控器等；工業產品如馬達控制器汽車引擎的噴油控制器等，都是由微控器來實現的。8 位元控制器是最簡單的微控器，具有價廉的優勢，卻可以勝任大多數的微控器產品。談到 8 位元控制器，大家都會想到 8051 微處理器，其強大的功能及週邊介面的擴充性，使它曾經風光一時。近年來強調整合性晶片，許多廠商紛紛推出內建多種週邊介面的微控器，搶佔了 8051 微處理器的市場。微控器的優勢在於價廉節省電路板空間和韌體程式開發容易等。本論文利用松翰科技公司推出的 SN8P1700 系列微控器其中的一款微控器 SN8P1708 微控器來實現三相永磁馬達 (Permanent Magnet Synchronous Motor) 的正反轉及速度控制，透過霍爾元件的回授訊號來達成閉迴路控制馬達的旋轉，有別於以往開迴路控制馬達容易發生失速的現象，閉迴路控制將使馬達旋轉更加穩定。松翰科技公司所推出的 SN8P1708 微控器有下列幾點特色 :

1. 指令爲 16 位元的機械碼，其內之程式位址碼爲 12 位元，所以 4K 大的程式記憶體都可以使用直接跳躍 (JMP 和 CALL)。某些微控器機械碼僅含 10 位元的位址碼，所以不同程式記憶體頁 (page of ROM) 的跳躍，需要加其他指令來作處理。一個記憶體頁爲 1K 的位址容量。

2. 系統控制器的記憶體有足夠的空間，所以每個系統暫存器只含同性質的參數位元，避免掉二個以上不同性質的參數用一個系統暫存器來管理，如此使用時比較不會混淆。

3. 將系統暫存器與一般暫存器都放在同一個資料記憶體之實體 (a physical RAM)，並且使用相同指令來讀取和寫入。此項特點降低組合語言程式撰寫的難度，和提高組合語言程式閱讀的親和性。傳統的微處理器，讀寫系統暫存器的指令是和讀寫一般暫存器的指令是不同，所以撰寫組合語言程式較爲不便。

4. 看門狗計時器重置，睡眠模式設定都用一般指令來執行，不需要使用特定指令，減少指令集的個數。

5. 多種不同數量腳位的晶片，可依所需輸出輸入腳多寡來選擇不同晶片，輸出輸入腳最多可達 33 支。

6. 一套實體模擬器 (In-Circuit Emulator，ICE) 支援全系列微控器。

7. 8 位元微控制器卻可比擁有 12 位元解析度的類比數位轉換器 (Analog-to-Digital Converter，ADC)，擴大應用領域。

8. 擁有完整的控制週邊。價格卻與市面的一般微控器一樣便宜。

既然 SN8P1708 微控器有如此多的優點，而且又比 8051 微控器更好用，所以我們可以使用它來控制三相永磁馬達。

1.2 論文架構

本篇論文總共分爲六章，其中第一章將說明研究的方向 ; 第二章將介紹本實驗所需要用到的相關知識 ; 第三章將說明硬體的設計方式以及介紹微控器的硬體架構 ; 第四章將說明軟體架構與程式流程 ; 第五章將實驗的結果做一個展示 ; 第六章將做一個討論與總結。 本論文架構可分爲六大章節，分述如下 :

第一章 緒論。 先對研究背景及動機進行說明。

第二章 基本知識介紹。 說明三相永磁馬達的內部構造與旋轉原理。

第三章 硬體架構。 實做電路之設計與微控器之硬體結構說明。

第四章 軟體架構。 逐一說明程式流程與軟體開發視窗。

第五章 實驗結果。 展示硬體電路以及實驗結果。

第六章 結論與展望。 將討論本實驗之改善，與其他微控器做三相永磁馬達控制之比較，並提出未來的改進方向及建議。

第二章

基本知識介紹

爲了使讀者對本論文的實驗有更深入的了解，在此將對三相永磁式馬達的結構、驅動方式以及一些相關原理及知識做基本的介紹，讓讀者能清楚了解之後的理論推導與實做的方法。

2.1 一般光碟機主軸馬達之驅動方法

一般的光碟機主軸馬達驅動方法爲透過主軸馬達驅動 IC(例如 BA6849 晶片) 來切換變頻器 (Inverter)，藉以送出六步方波驅動主軸馬達，其驅動方式可分爲有感測器與無感測器兩種方式。有感測器之方式爲在無刷馬達內部安裝霍爾 (Hall) 感測器作換相之用，馬達才能正常旋轉，而無感測器 (sensorless) 驅動方法是參考馬達開路相感應電動勢 (back-emf) 的越零點 (zero-crossing) 作換相之用，馬達內不需要再裝上霍爾 (Hall) 感測器。有感測器之控制理論較無感測器簡單，但無感測器因爲不需要裝上霍爾 (Hall) 感測器，所以較省面積。

2.2　三相永磁馬達結構

我們選用的三相永磁馬達 (如圖 2.1所示) 是 9 槽 12 極的三相永磁馬達 (內部定子有九個槽，外部轉子有 12 極)；外部轉子每極所佔的角度是 30 度，由永久磁鐵所構成；內部的定子則是鐵心繞上線圈，鐵心兩邊的線圈各佔 5 度，因此每槽所佔的機械角度爲 30 度，槽與槽的中心點相距 40 度，其中的空隙爲纏繞線圈的空間；透過霍爾元件感測轉子與定子的相對位置，改變線圈電流的流動方向，產生一正向的扭矩，推動外部轉子，讓三相永磁馬達能夠朝同一方向旋轉。三相永磁馬達內部的接線圖如圖 2.2，細部結構如圖 2.3，其內部各元件的名稱如表 2.1。

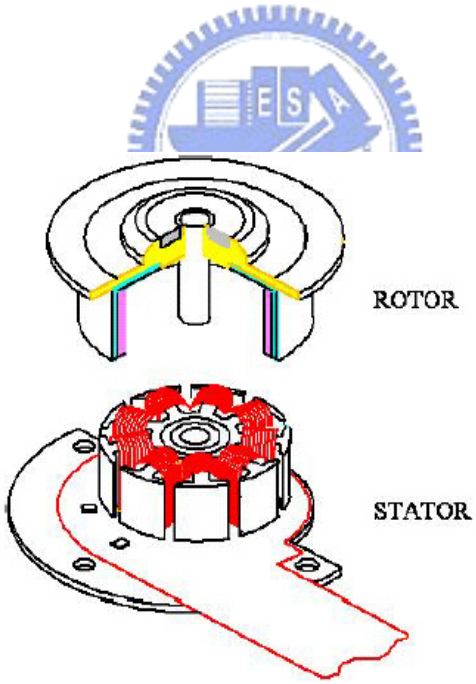
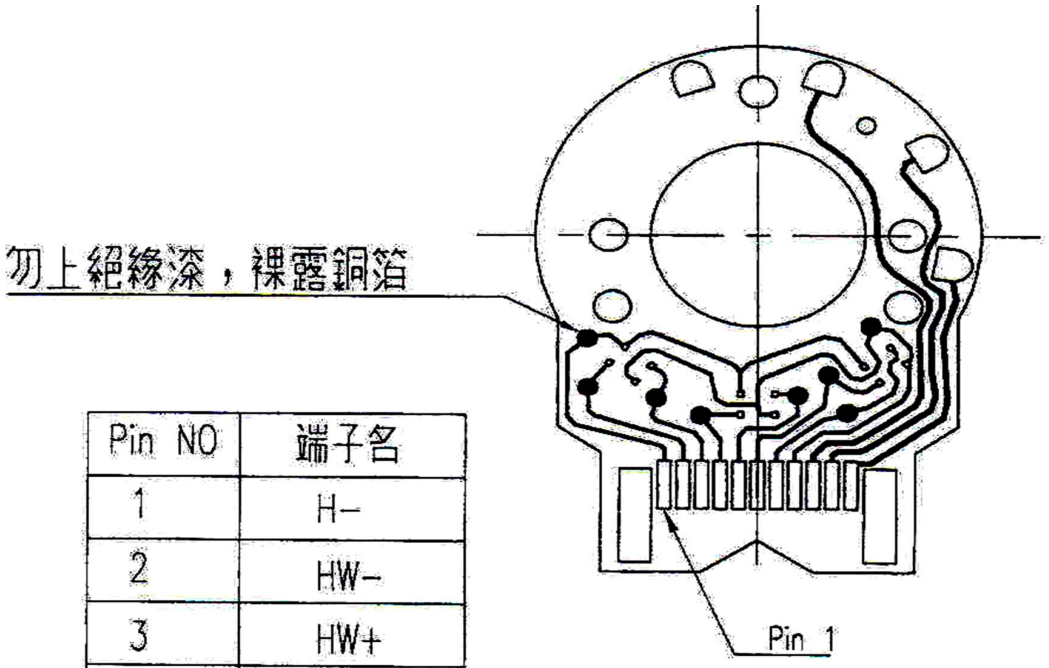


圖 2.1: 三相永磁馬達的轉子 (ROTOR) 與定子 (STATOR)[2]



Pin NO	端子名
1	H-
2	HW-
3	HW+
4	HV-
5	HV+
6	HU-
7	HU+
8	H+
9	U
10	V
11	W

Pin9--11:耐電流1.3A

圖 2.2: 三相永磁馬達內部接線圖 [2]

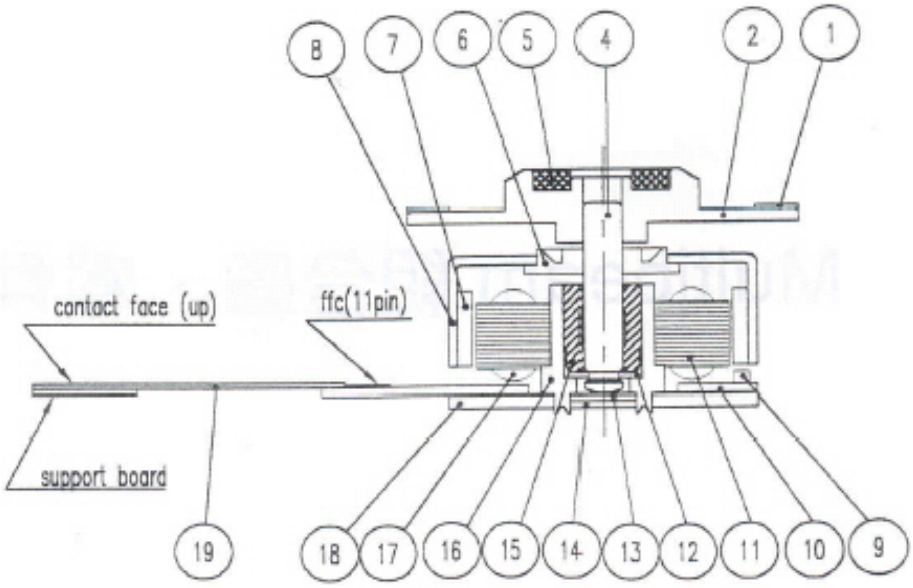


圖 2.3: 三相永磁馬達結構圖 [2]

NO.	物件名稱	QTY	NO.	物件名稱	QTY
1	止滑片	1	11	鐵心	1
2	承座基座	1	12	防拔扣	1
4	主軸	1	13	摩擦片	1
5	Hub 磁石	1	14	止推片	1
6	軛鐵套	1	15	含油軸承	1
7	磁石	1	16	軸承套	1
8	軛鐵	1	17	漆包線	1
9	霍爾元件	3	18	底板	1
10	PCB 板	1	19	FFC	1

表 2.1: 三相永磁馬達結構說明表

2.3 三相永磁馬達旋轉原理

三相永磁馬達外部轉子是由永久帶磁物質所構成，若欲使轉子轉動則必須要使內部定子的磁場依照轉子的位置改變。藉由在內部的定子上繞線圈，用外界的輸入電流造成電磁場的改變，因此而產生轉矩，但內部定子已經固定住，，因此產生了一個反作用力推動外部轉子，所以才能使三相永磁馬達旋轉。在此先複習一下基本觀念，在往後談到相關的知識更能駕輕就熟：

2.3.1 扭矩常數 K_t

我們定義轉子與定子的位置如圖 2.4

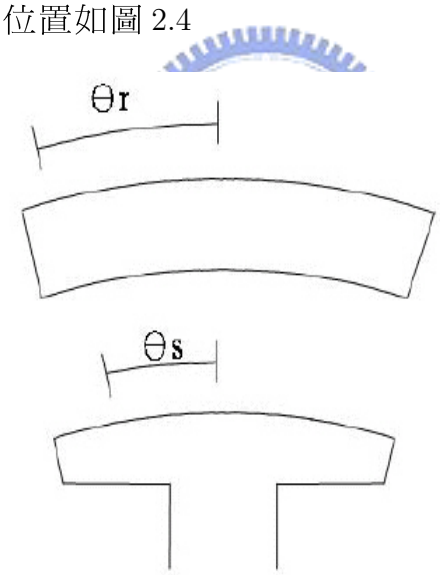


圖 2.4: θ_r 與 θ_s 示意圖

根據弗來明左手定律

$F = I \cdot L \times B$ $F = N \cdot I \cdot L \times B$(若有 N 匝) 其中

I 定義爲線圈上的電流

L 定義爲線圈上受磁場感應的有效長度

B 定義爲磁通密度

三相永磁馬達扭矩則爲 $T = F \cdot r = N \cdot I \cdot L \times B \cdot r = K_t \cdot I$ 其中

K_t 定義爲扭矩常數

r 定義爲力臂

因爲 B 與 θ_r,θ_s 的角度有關 $B = B(\theta_r, \theta_s) = B_{max} \cdot \cos[P/2(\theta_r - \theta_s)]$ 其中

P 定義爲轉子上永久磁鐵的磁極數

θ_r 定義爲轉子上對定子中點的角度差

θ_s 定義爲定子上某一點對定子中點的角度差

經由推算以後，我們得知三相永磁馬達的扭矩常數 K_t 會隨著轉子轉動而成 sin 波變化。 $K_t = 6 \cdot N \cdot B_{max} \cdot L \cdot r \cdot \sin(P\theta_r/2)$

2.3.2 霍爾效應

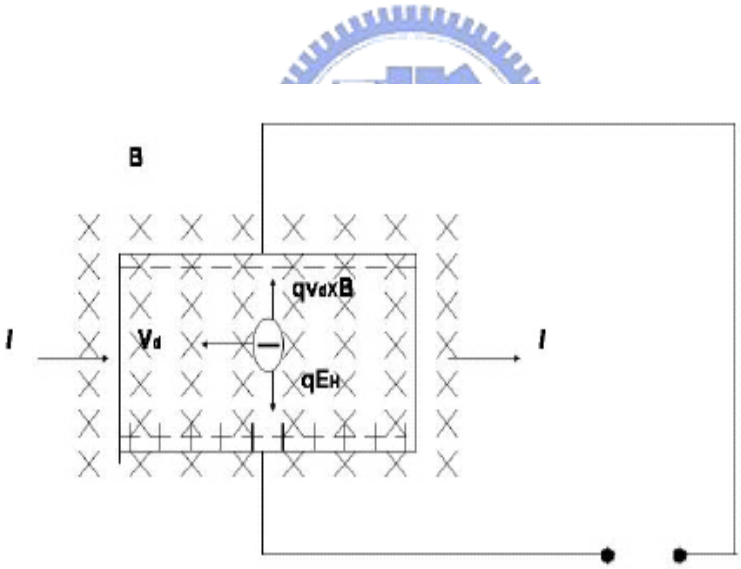


圖 2.5: 霍爾效應示意圖 [2]

圖 2.5爲說明霍爾效應的原理，當導體在磁場的附近感應到電位差時，因爲電子的流動與磁場的感應會造成橫向的電動勢，此電動勢與磁場的大小變化有關。霍爾效應說明如下：

通電流經一導電材料，施加一和電流垂直之均勻磁場，在電流兩側和磁場垂直方向可量得一電位差，此效應稱爲霍爾效應。(如圖 2.5)

2.3.3 驅動方式

對於三相永磁馬達，通常採用的驅動方式有 120 度方波驅動以及 180 度方波驅動。在此我們只對 120 度驅動方式詳加說明。

[120 度六步方波的驅動原理]

我們定義三相永磁馬達的轉子與定子的初始位置如圖 2.6

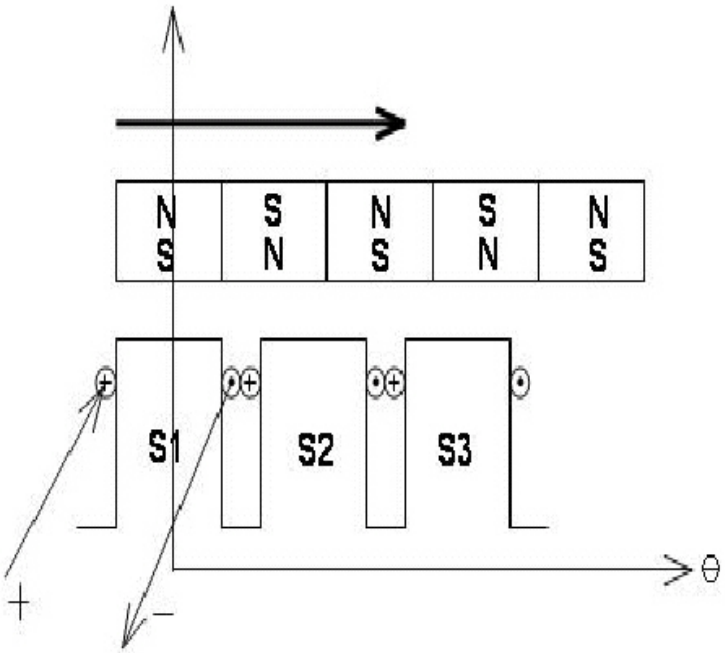


圖 2.6: 三相永磁馬達中轉子與定子的展開圖 [2]

圖 2.7爲當三相永磁馬達旋轉時，分別在 S1，S2，S3 三個定子的位置，所量到 K_t 隨著電氣角與機械角變化的波形圖 (電氣角 / 機械角 =pole 的數目 /2=6，所以電氣角跑了 360 度，機械角只跑了 60 度，也就是說馬達只轉了 60 度)

圖 2.8爲 S1，S2，S3 三個定子任意通兩相電流的情況下 (一個爲輸入端，一個爲輸出端，另一端爲開路)，K_t 隨著電氣角與機械角變化的波形圖。

若要使三相永磁馬達的輸出轉矩維持在最大值，由公式 $T = K_t \cdot I$ 知道，我們必須要將 K_t) 維持在最大值，因此藉由圖 2.8的波形圖我們可以整理切換電流的時間點如表 2.2

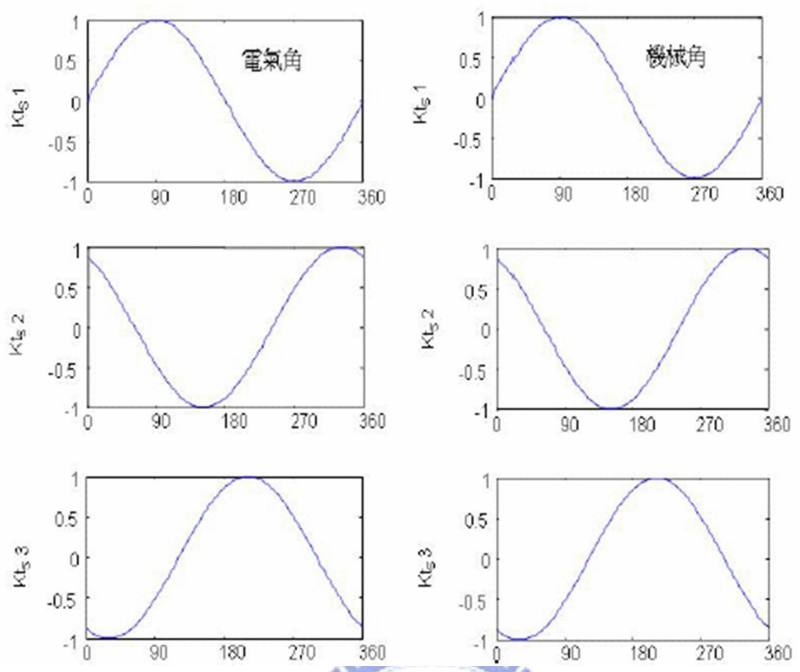


圖 2.7: 單相 K_t 圖 (橫軸單位 :degree，縱軸單位 :g*cm/A)

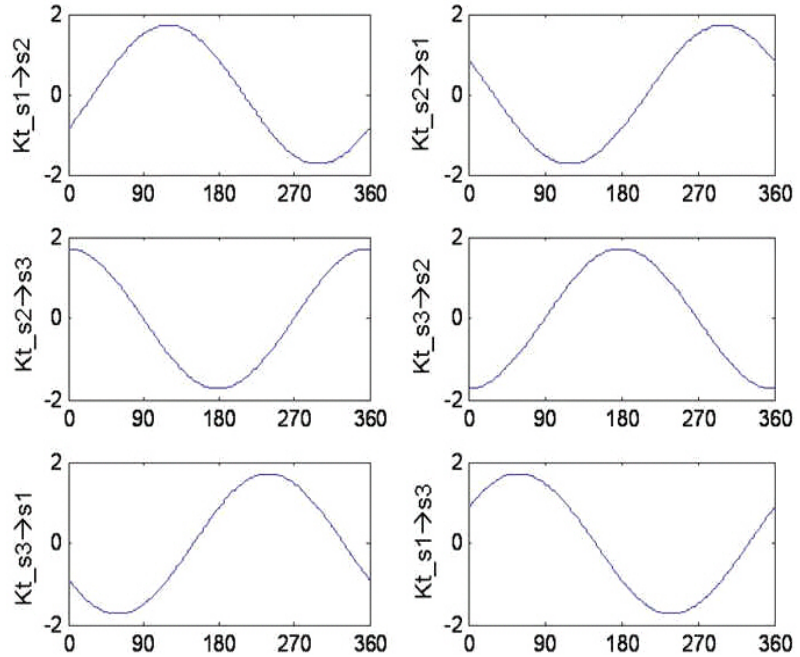


圖 2.8: 任意通兩相電流的 K_t 圖 (橫軸單位 :degree，縱軸單位 :g*cm/A)[2]

狀態	通以電流方向	選擇電氣角度
狀態一	$S2 \rightarrow S3$	[-30,30]
狀態二	$S1 \rightarrow S3$	[30,90]
狀態三	$S1 \rightarrow S2$	[90,150]
狀態四	$S3 \rightarrow S2$	[150,210]
狀態五	$S3 \rightarrow S1$	[210,270]
狀態六	$S2 \rightarrow S1$	[270,330]
狀態一	略	略
關鍵切換點	-30,30,90	150,210,270

表 2.2: 電流切換點

例如在電氣角 [-30，30] 間要使 K_t 在最大值，因此選擇通電流方向爲 $S2 \rightarrow S3$

在電氣角 [30，90] 間要使 K_t 在最大值，因此選擇通電流方向爲 $S1 \rightarrow S3$

我們透過霍爾感測器來偵測電氣角的位置，此感測器最重要的功能是能在無接觸的狀況下感應到磁場的變化，進而確定轉子與定子的相對位置；此霍爾感測器乃是利用著名的霍爾效應（Hall effect）將磁場變化產生的感應電流透過電壓的形式表現出來。在推算出關鍵切換點後，我們就可以將霍爾感測器埋藏在切換點 (如圖 2.9所示)，再利用霍爾感測器得到的邏輯訊號，判斷三相永磁馬達的轉子位在哪一個狀態點，然後在改變三相的電壓值。

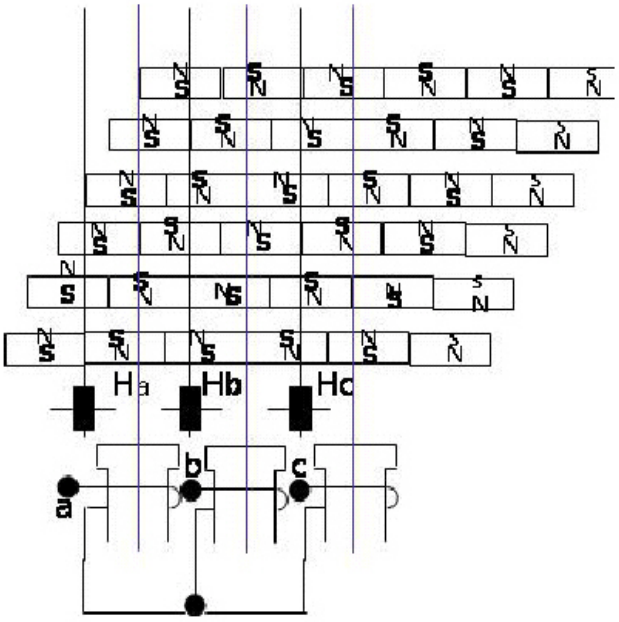


圖 2.9: 霍爾感測器埋設點 [2]

經過觀察，我們發現將霍爾感測器埋藏在各個定子的中間點，剛好和切換的時間點重合。可以經由霍爾感測器的換相得知正確的切換時間，以達到最大的扭矩輸出。圖 2.10是分別是由埋藏好的 Ha，Hb 以及 Hc 所量測到的霍爾感測器輸出訊號圖。

藉由霍爾感測器的輸出訊號圖，可以整理出如表 2.3的 6 個狀態。

我們可以透過霍爾感測器知道三相永磁馬達在哪一個狀態點後，再利用電流的切換，便可以達到最大轉矩。 我們可以透過霍爾感測器知道三相永磁馬達在哪一個狀態點後，再利用電流的切換，便可以達到最大轉矩。

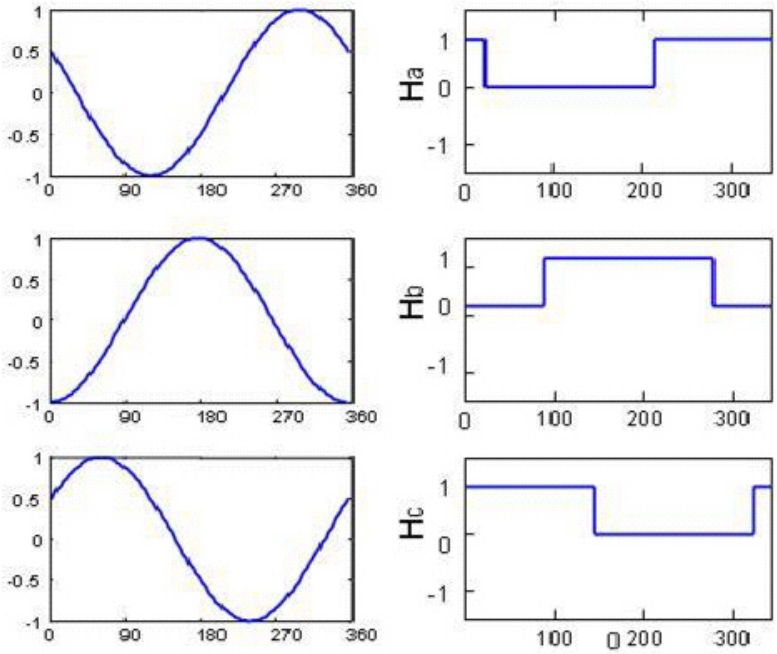


圖 2.10: 霍爾感測器輸出訊號圖 [2]

狀態	S1	S2	S3	Ha	Hb	Hc
狀態一	open	+	-	H	L	H
狀態二	+	open	-	L	L	H
狀態三	+	-	open	L	H	H
狀態四	open	-	+	L	H	L
狀態五	-	open	+	H	H	L
狀態六	-	+	open	H	L	L

表 2.3: 霍爾感測器的邏輯表

2.4 計算馬達轉速之公式推導

一般來說可使用轉速計來量測得馬達之轉速，但量測誤差頗大，得量測多次再取之平均值，但使用 FG(Frequency Generator) 的頻率來計算馬達之轉速較爲便利，誤差也較小。方法爲在每次送出一六步方波至馬達時，做出一方波訊號，當再送下一六步方波至馬達時，將此方波訊號做反向 (Not)，整個 FG 的波形如下之示意圖 (圖 2.11):

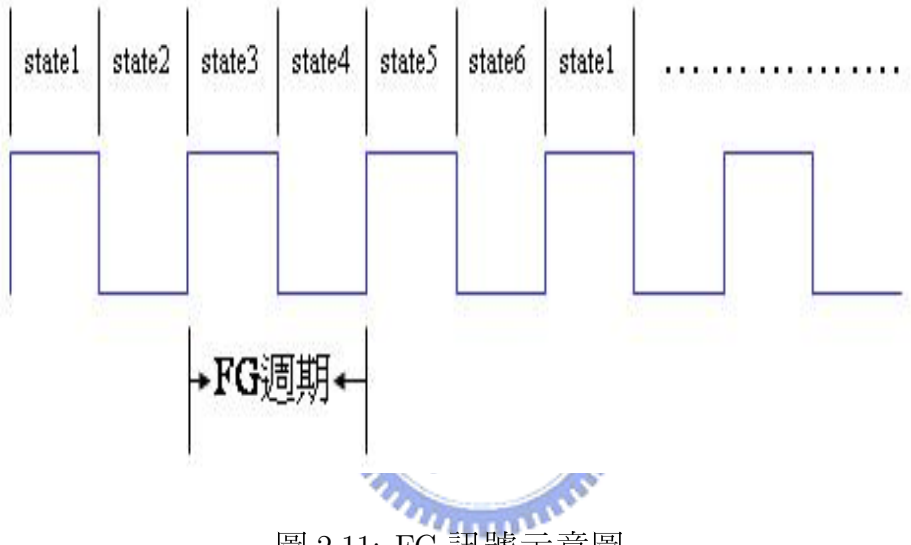


圖 2.11: FG 訊號示意圖

由此可知送出一完整之六步方波訊號內有三個 FG 週期，又一完整之六步方波只轉了機械角 60 度，也就是 1/6 轉，故馬達 1 轉總共有 6 個完整之六步方波，也就是 18 個 FG 週期，假設 FG 之頻率爲 f，所以一分鐘有 60 × f 個 FG，故

FG 訊號計算馬達轉速之公式 = 60 × f/18 (轉 / 分鐘)= 60 × f/18 (rpm)。

第三章

硬體架構

3.1 概述

發展此系統的硬體部份，是採用松翰科技公司所製造的實體模擬器 ICE 透過 RS-232C 介面傳輸，與本人所製作之受控端電路與三相永磁馬達以及個人電腦所組成之硬體架構，此章會依序介紹發展工具的硬體部份以及模擬電路板與受控端電路。

3.2 微控器架構

本論文所使用之微控器架構如圖 3.1所示，它是一顆非常完整的微控器。算術邏輯運算器 ALU 爲微控器的中樞，依指令暫存器 (instruction register) 內的機械碼來執行算術或邏輯運算，而程式的機械碼是燒錄在唯讀記憶體 ROM 中，由程式計數器 PC 負責記載著下一個要執行指令在 ROM 中的位址。每當執行完一指令，就從 PC 所指的 ROM 位置處取出指令存到指令暫存器，然後 PC 就加 1，以指到下一個指令。如果指令是執行算術或邏輯運算，需要兩個運算元，其中一個必須是累積器 ACC 的內容，另外一個可以是可讀寫記憶體 RAM 中某位址之內容或是在機械碼中給予直接數值。運算結果會產生一些狀態旗標，例如進位、零結果等，就存於暫存器 Flags。爲了方便撰寫

程式和加快運算速度，微處理器都會在 ALU 旁加上除了 ACC 與 Flags 外數個特殊暫存器，這些暫存器在 SN8P1700 系列微控器中稱作工作暫存器。ALU 執行指令的速度完全由振盪器的時脈控制，準確的時脈都是由外部的石英振盪器提供，雖然微控器內部也提供 RC 電路型的振盪器，振盪器的時脈經過一個預除頻器後才形成 CPU 時脈。不同的指令需要的執行時間不同，最新的微控器都設計儘可能在一個 CPU 時脈週期內完成一個指令，以方便估算程式執行時間。SN8P1708 微控器的大部份指令的執行時間都是一個 CPU 時脈週期，只有少數的跳躍指令需要 2 個 CPU 時脈週期。一般微處理器除了運算功能外，還要具備中斷控制功能 (interrupt control)，就是可以打斷程式的執行程序，先處理一些緊急而臨時的程式後，再回過來處理先前的程式。

SN8P1708 微控器的週邊介面也示於圖 3.1中。這系列的微控器最多有五組輸出入埠 :Port0 ～ Port2，Port4 和 Port5。除了 Port0 僅能當輸入腳外，其他四組都可以隨意規劃爲輸入腳和輸出腳。計時器 (timer) 可以利用 CPU 時脈來計算時間，用以產生計時中斷、輸出脈寬調變訊號 PWM、週期性方波輸出。SN8P1708 擁有 2 組 PWM 產生器 PWM0 和 PWM1，但是訊號輸出腳和 Port5 的 2 支腳共用。這 2 組 PWM 產生器也具有產生週期性方波的功能，所以二種訊號共用一個介面。

SN8P1708 微控器提供 8 通道的類比轉數位的轉換器 (ADC)，類比訊號可以經由 Port4 的八支腳輸入，但是 ADC 每次僅能處理一個通道的類比訊號，以程式方式來切換輸入通道。一般 8 位元的微控器之 ADC 儘能提供 8 位元的解析度，但是 SN8P1708 微控器卻可以用程式來選擇 8 位元 12 位元的解析度，擴大了應用範圍。數位轉類比的轉換器 DAC 爲 ADC 之反相，是將數位訊號轉成類比訊號輸出，但是要注意，SN8P1708 微控器的 DAC 輸出爲電流訊號，這是專門給電流控制的裝置使用。

串列式通信的重要性與日俱增。2 個以上的晶片相互傳輸資料的必要性已經出現在許多產品上。而晶片間的資料傳輸以串列式傳輸爲主流，因爲可以節省輸出入腳位。SN8P1708 微控器內含一個串列式通信介面 SIO，這個介

面引擎與串列周邊介面 SPI(Serial Peripheral Interface) 的規格相容，也就是可以直接使用 SIO 來和具有 SPI 的晶片作串列式資料傳輸。其他常用的串列式傳輸爲 RS232 和 I2C 規格，必要時 SN8P1708 微控器僅能以韌體程式來作這些串列式傳輸。

爲了提高微控器的可靠度，低電位偵測器 (low voltage detector) 和看門狗計時器 (watchdog timer) 這二項功能也成爲 SN8P1078 必備的功能。當電源的工作電壓過低時，低電位偵測器會啓動重置動作，以避免微控器因電壓不足而作出誤動作，誤動作嚴重時會損壞週邊電路。看門狗計時器本身如同一般計時器一樣，會依據 CPU 時脈來計時，但是它的功用卻不同。在看門狗計時器計時到規定的逾時時間，就會觸發系統重置的動作。所以在正常程式中，每隔適當位置，就要作一次看門狗計時器的歸零動作，以避免發生逾時的問題。在無法預期的環境或狀況下，程式可能會進入無窮迴圈，這時就會發生看門狗計時器逾時，而觸發重置，讓系統重新執行。由於看門狗計時器的逾時時間都短於 0.3 秒，所以短時間的程式錯誤還不至於造成系統毀損。

其他關於此微控器的腳位說明與設定, 還有程式記憶體 ROM 與資料記憶體 RAM 之架構, 可參考林錫寬教授所著掌握控制器原理與技術使用 SN8P1700 系列晶片 [1] 一書, 將有詳細之介紹。

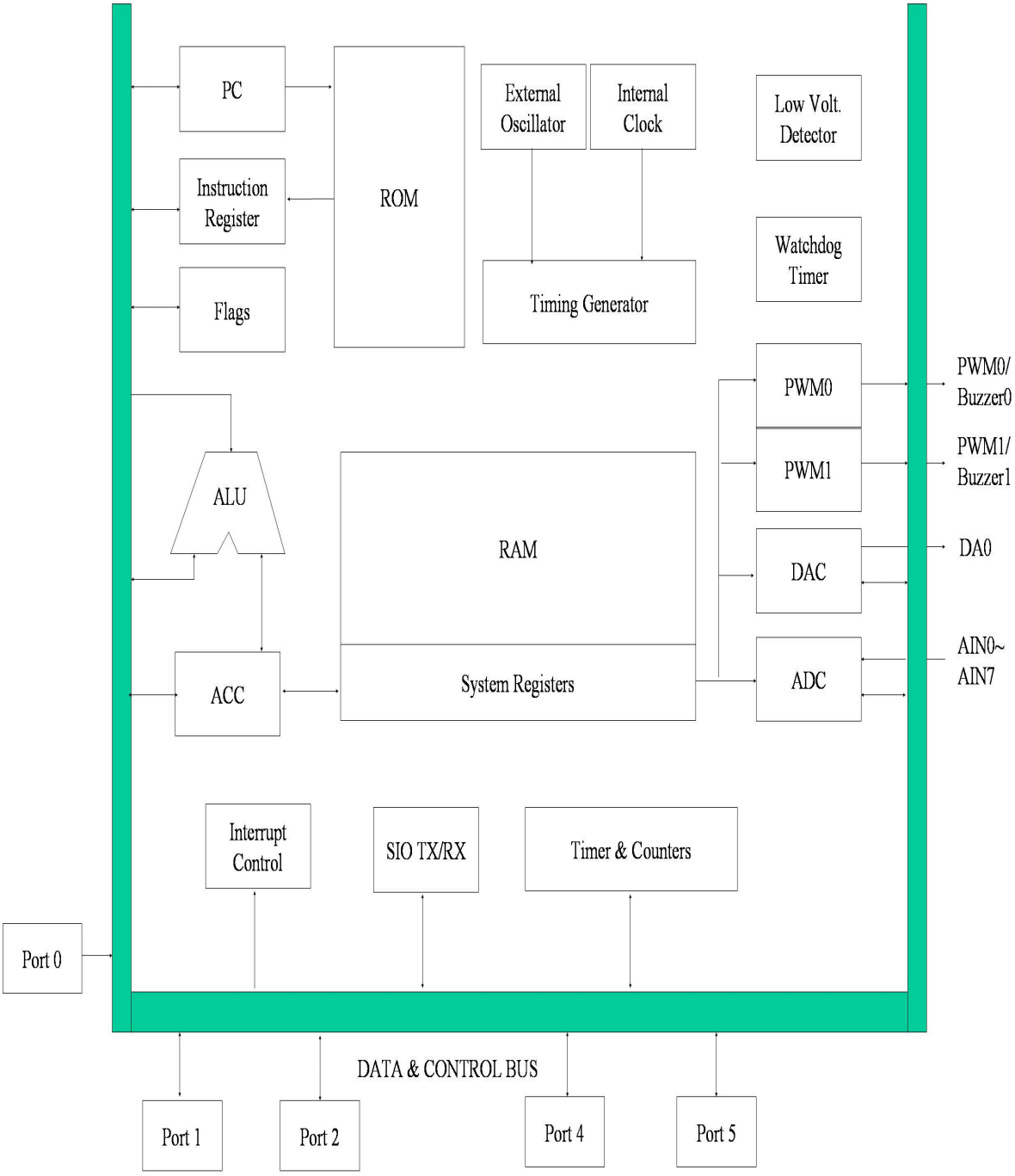


圖 3.1: SN8P1708 晶片系統方塊圖 [1]

3.3 軟體掃描鍵盤

最常見的按鍵有機械式與薄膜式，機械式利用彈簧來回彈。而薄膜式乃藉橡膠薄膜黏著石墨來運作，壓下時因石墨接觸電路而導電，鬆開時橡膠彈力使石墨離開電路。無論何種按鍵都有彈跳現象，即按下和放開時都會有維持 3ms ～ 10ms 的接觸分離交替不穩定現象，雖然可以用硬體線路來除彈跳，但是爲了節省成本，都是用軟體延遲程式來除彈跳。基於人的反應速度，按下按鍵至放開的最短時間大於 40ms，我們可以設計一個延遲程式，每 10ms 檢查一次按鍵狀態，若有狀態改變時，待新狀態連續相同達 2 次以上才確認爲改變狀態，對於確認放開按鍵也採取相同之方式。

以一個 4 × 4 鍵盤爲例，如圖 3.2所示，輸出埠爲低電位時致能掃描。輸入埠需選用內部高接型 (pull-high)，每一列接於一支輸入腳，輸入腳選擇腳位爲 P1.0 ～ P1.3; 而每一行接於一條掃瞄線，經過一顆 74138 解碼器 (圖 3.3所示) 才接於微控器的 P5.5 ～ P5.7 腳。74138 解碼器的輸出腳 (Y0 ～ Y7) 平常爲高電位，一次可以選擇一支腳爲低電位輸出，輸出腳致能低電位由輸入腳 A、B 與 C 的電位來控制，進一步規定，DisplayBit=0 時爲 Y0 低電位，DisplayBit=1 時爲 Y1 低電位，DisplayBit=2 時爲 Y2 低電位，DisplayBit=3 時爲 Y3 低電位。也就是當 DisplayBit=0 時致能連接按鈕 0，1，2，3 的掃瞄線 ; 如果此時這四鍵中某一鍵被按下，則高接型的輸入腳將經此掃瞄線連到 74138 的低電位 (即接地)，結果輸入腳的值就從平常爲 1(高電位) 變成 0(低電位)。所以輸入腳 P1.0 ～ P1.3 平時爲高電位，一但有一支腳爲低電位，就表示有按鍵被壓下，再根據壓下時的掃瞄線腳位，可以換算出被壓下鍵的號碼。

編碼之公式 : A = 4 × Column No. + Row No.

另外，當 A、B 與 C 輸入腳的電位均爲低電位時致能 Y7 掃瞄線，此時若有任一鍵被壓下則執行外部中斷 INT0 之功能，鍵盤掃描之副程式將在第四章有所介紹。

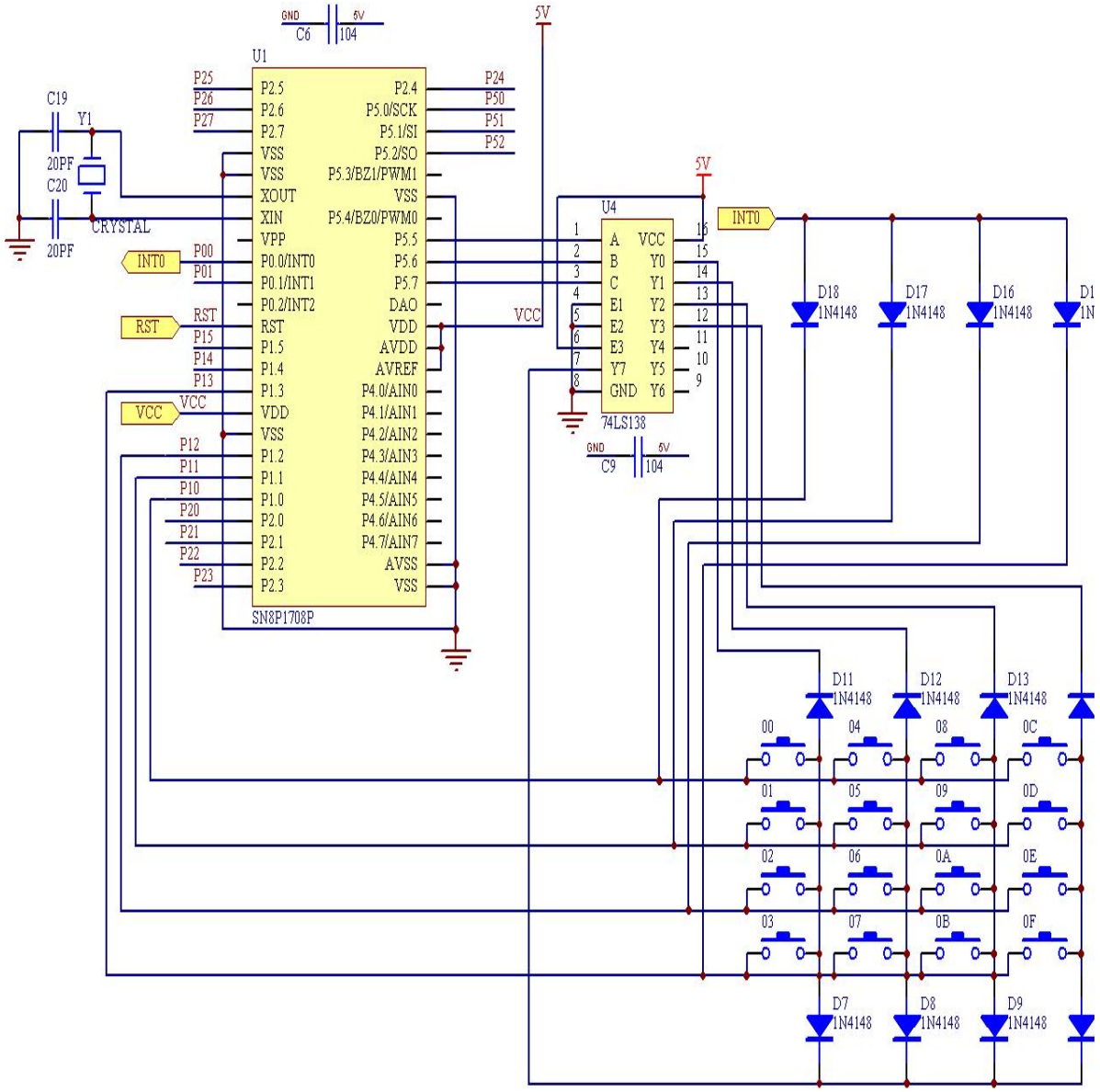
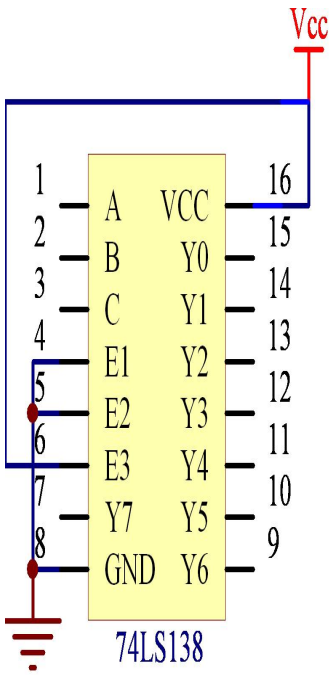


圖 3.2: 軟體鍵盤掃描之電路圖



輸　入					輸　出							
G1	G2A/B	C	B	A	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
---	H	---	---	---	H	H	H	H	H	H	H	H
L	---	---	---	---	H	H	H	H	H	H	H	H
H	L	L	L	L	H	H	H	H	H	H	H	L
H	L	L	L	H	H	H	H	H	H	H	L	H
H	L	L	H	L	H	H	H	H	H	L	H	H
H	L	L	H	H	H	H	H	H	L	H	H	H
H	L	H	L	L	H	H	H	L	H	H	H	H
H	L	H	L	H	H	H	L	H	H	H	H	H
H	L	H	H	L	H	L	H	H	H	H	H	H
H	L	H	H	H	L	H	H	H	H	H	H	H

圖 3.3: 74138 解碼器

3.4　霍爾回授電路

霍爾回授電路如圖 3.4所示，由埋藏在三相永磁馬達內的霍爾元件所感應出之正弦波電壓波形，經由比較器 IC LM358(圖 3.5) 產生高準位與低準位電壓經由二極體輸入至微控器的 P2.0、P2.1 與 P1.4 腳位，來判斷馬達轉子之位置。利用其二極體單相導通之特性，避免過大之比較器輸出電流流至單晶片內使之燒燬。IC LM358 內有二個比較器電路，其中每一個比較器電路如圖 3.6所示。

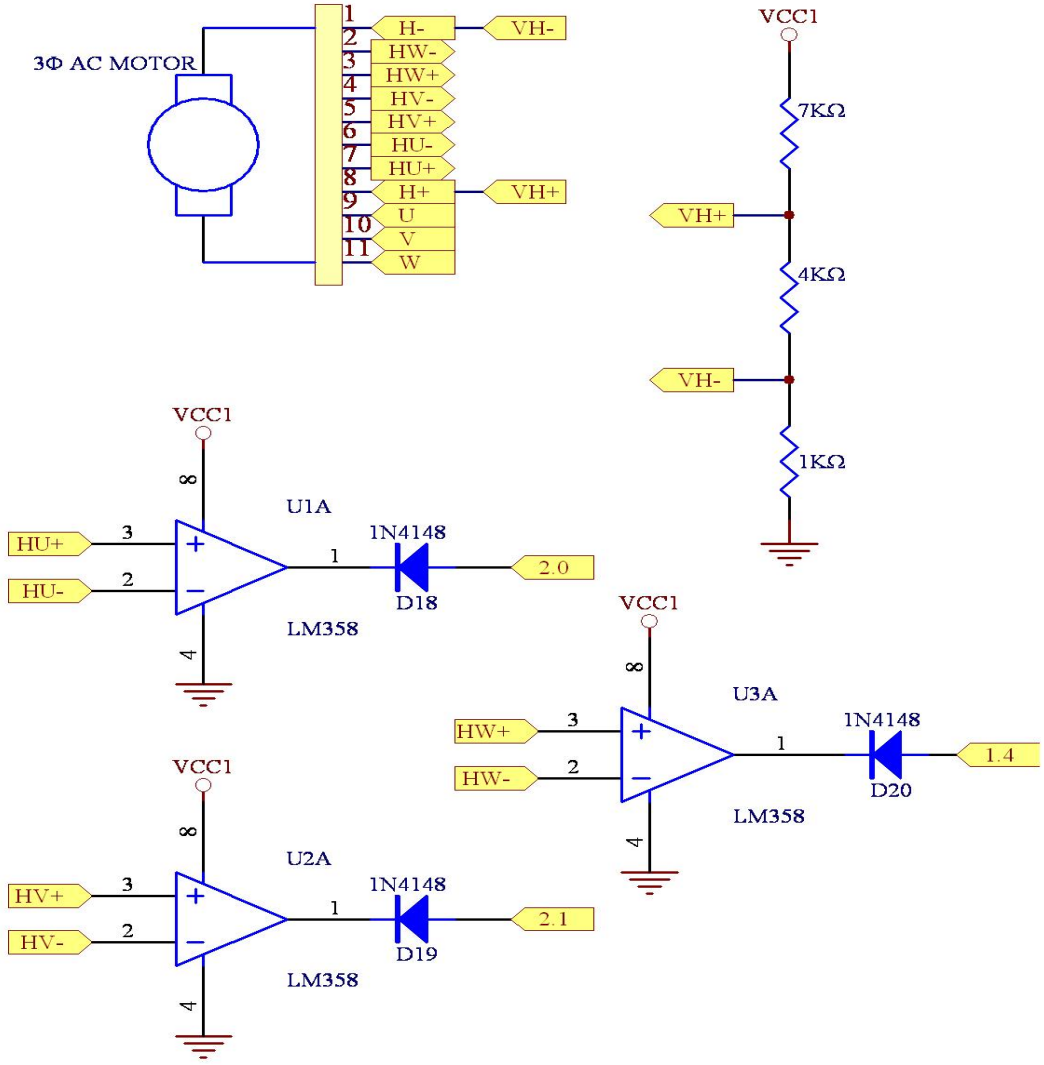


圖 3.4: 霍爾回授電路

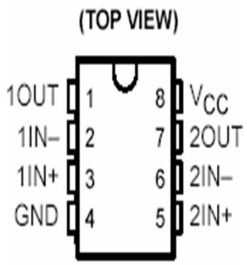


圖 3.5: IC LM358

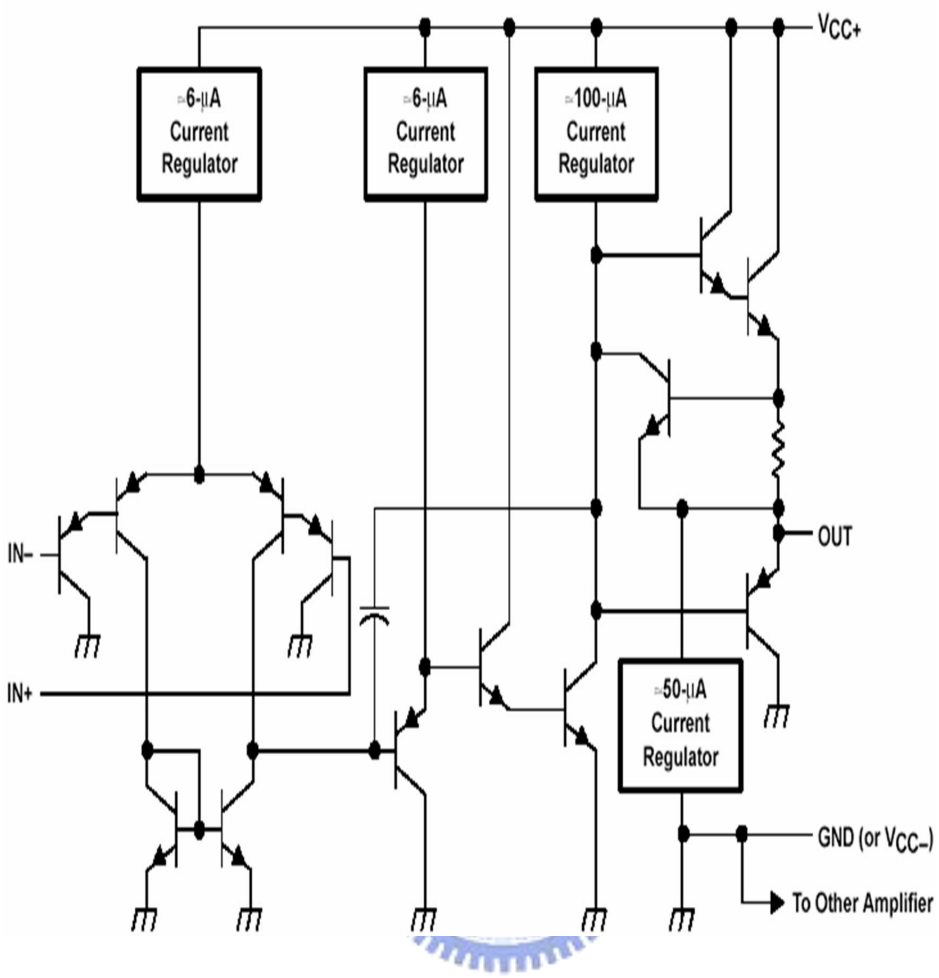


圖 3.6: 單一比較器電路

3.5 六步變頻器電路

透過六步變頻器 (six-step inverter) 電路 (圖 3.7所示) 的時序切換，所產生之類似正弦波之六部方波可以用來驅動三相永磁馬達之旋轉。其中使用光耦合 IC TLP250(圖 3.8) 來區隔 MOS 驅動馬達電路與微控器輸出腳，避免萬一 MOS 較大之電流逆流回微控器造成晶片之燒毀。

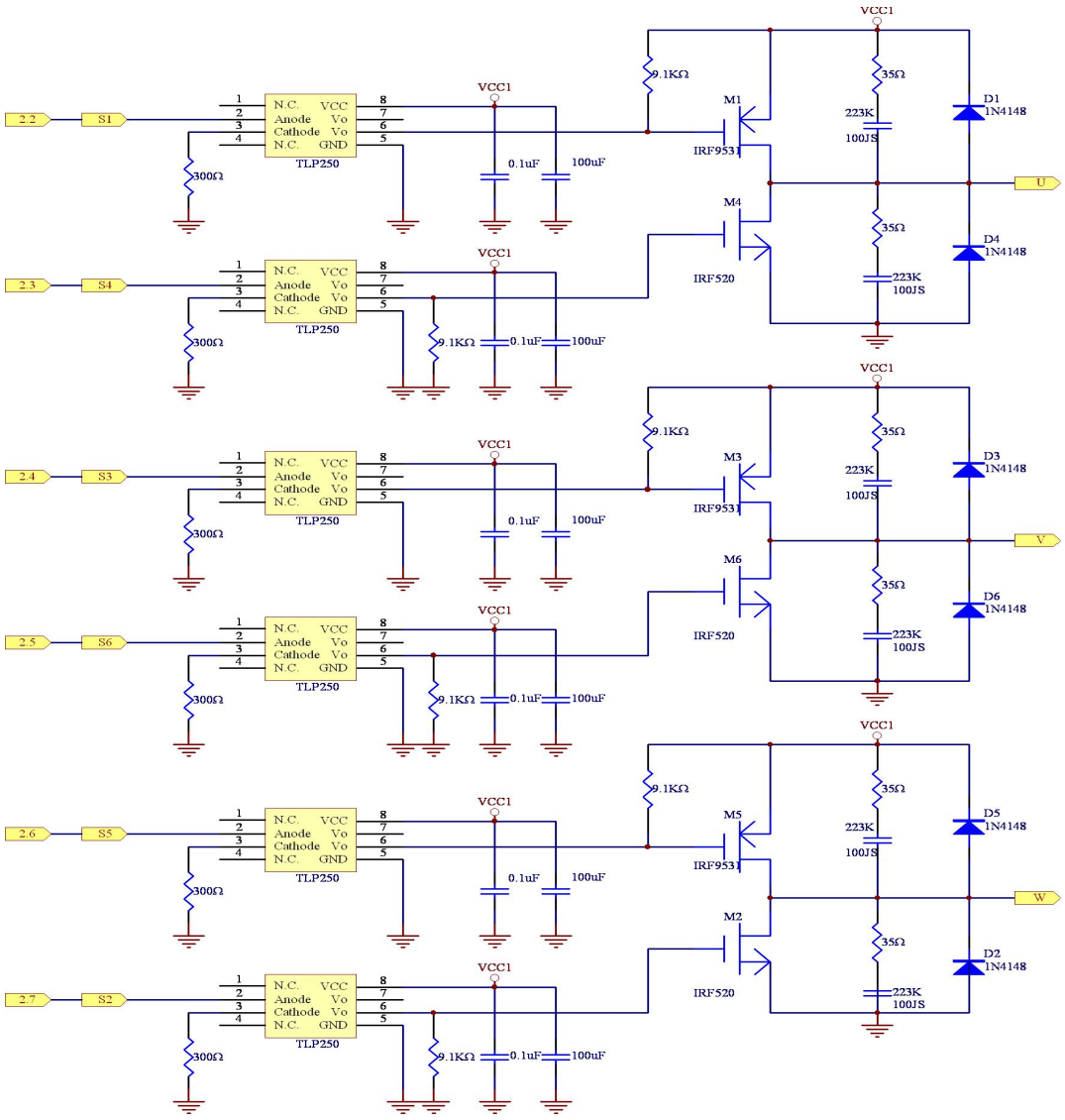


圖 3.7: 六步變頻器電路

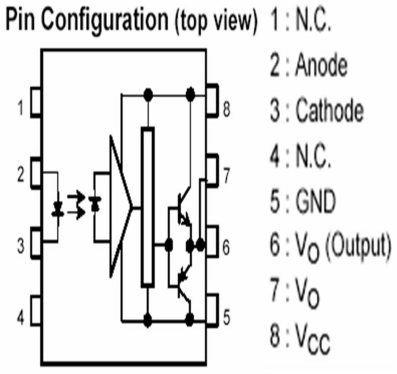


圖 3.8: TLP250

3.6 LCD Display

一般的 LCD Display 爲 LCD+ 控制電路 + 驅動電路所組成的 Module(簡稱 LCM)，有文字型與繪圖型兩種，本實驗使用的爲 20 字× 2 列的文字型 LCM，腳位定義如下圖 (3.9) 所示 :

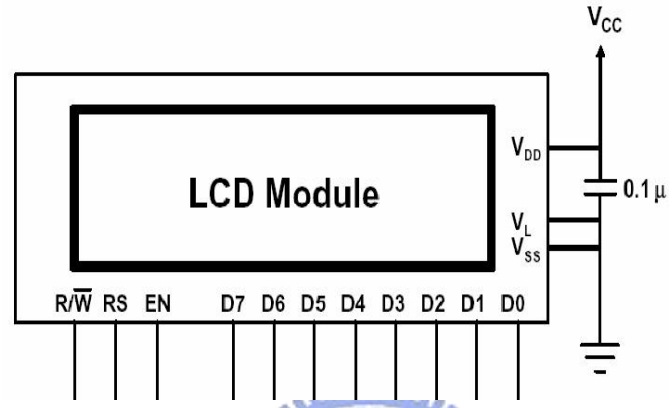


圖 3.9: LCM 腳位圖

共有 14 支 pin，其中 Pin1 爲 VSS 電源接地 ;Pin2 爲 VDD;Pin3 爲 VO 腳用來調整 LCD 亮度 ;Pin4 爲 RS，功能爲 Register Select，用來選擇 Instruction Register 與 Data Register;Pin5 爲 R/W，用來選擇 Read 或 Write;Pin6 爲 E(Enable) 用來致能寫入或讀取 LCD;Pin7 ～ Pin14 爲 DB0 ～ DB7 雙向資料匯流排 (Data Bus)。

3.6.1 LCM 結構與功能

LCM 有一專用控制晶片 :HD44780，主要特性如下：

- 80 bytes Display Data RAM (DD RAM)
- 內建 192 個 5 × 7 點矩陣字型的 Character Generator ROM (CG ROM)
- 64 bytes Character Generator RAM(CG RAM)

  (*1*) 每個字型 (character) 用 8 bytes 組成

  (*2*) 使用者可自行設計 8 個 characters

內部有二個暫存器 (Register) 分別爲指令暫存器 (Instruction Register) 和資料暫存器 (Data Register)，其中使用者下的指令存到指令暫存器，資料則透過資料暫存器傳送到 DD RAM 或 CG RAM，操作方式如下表格 (表 3.1) 所示 :

E	RS	R/W	Operation
0	X	X	Read/Write Disable
1	0	0	Write to Instruction Register
1	0	1	Read Busy flag and Address Counter
1	1	0	Write to Data Register
1	1	1	Read from Data Register

表 3.1: LCD 模式選擇表

LCD 內部有一 Busy Flag (BF)，當 BF=1 時 LCM busy，無法接收 Data Bus 來的指令，當 BF=0 時 LCM 可接收來自 Data Bus 的指令。

DD RAM 共有 80 個 bytes，並非全部都有使用到，其 Address 的分配爲 : 第一列 00h ～ 27h(00 ～ 39); 第二列爲 40h ～ 67h，當字數小於 40 字時各取前面的 Address。

CG ROM 內建 192 個 5 × 7 的點矩陣字型，其 Character Code 爲 20H ～ FFH(ASCII Codes);CG RAM 共有 64 bytes，例如一個 5 × 7 的點矩陣，字型或圖形部分需用 7 列 (7 bytes)，另外保留一列給游標 (1 byte)，造字型時，必須先設定 CG RAM Address。所以自行設計一字型需要 8 bytes CG RAM，最多可自行設計 8 個字型或圖形。

要讀取或寫入 DD RAM 或 CG RAM 時，必須設定 AC(Address Counter) 來指定位址。其設定 AC 的步驟爲 :

1. 將位址 (Address) 寫入 IR(Instruction Register)。

2. IR 自動將寫入的資料傳送至 AC。

3. 做讀取或寫入 DD RAM 或 CG RAM 的動作。

4. AC 會自動加 1 或減 1。

3.7　硬體電路圖

如圖 3.10所示的系統架構圖，以 PC 爲主的界面撰寫韌體程式碼透過實體模擬器 ICE，連接到受控端系統，並藉由電源供應器來驅動馬達。受控端電路圖如 3.11所示，其中包含 LCD Display 用來顯示馬達資訊，霍爾回授電路用來偵測馬達轉子的位置，六步變頻器電路用來驅動馬達旋轉，鍵盤掃描電路用來輸入資料給微控器。

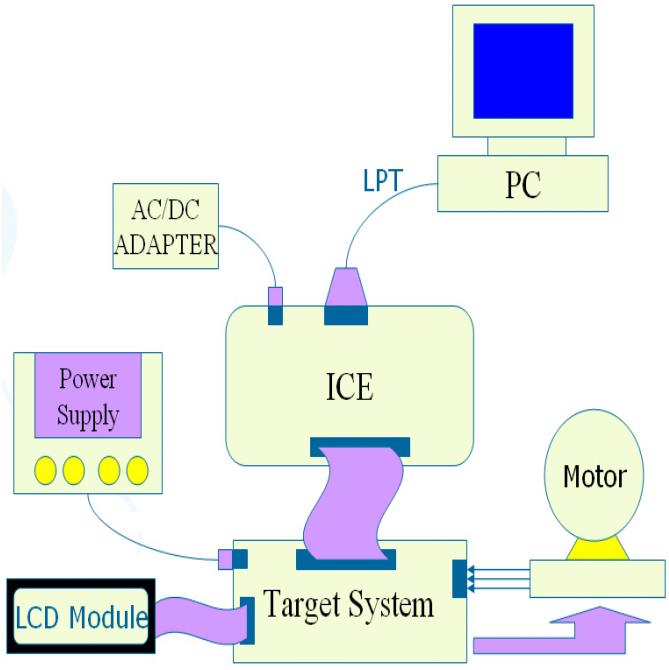


圖 3.10: 系統架構圖

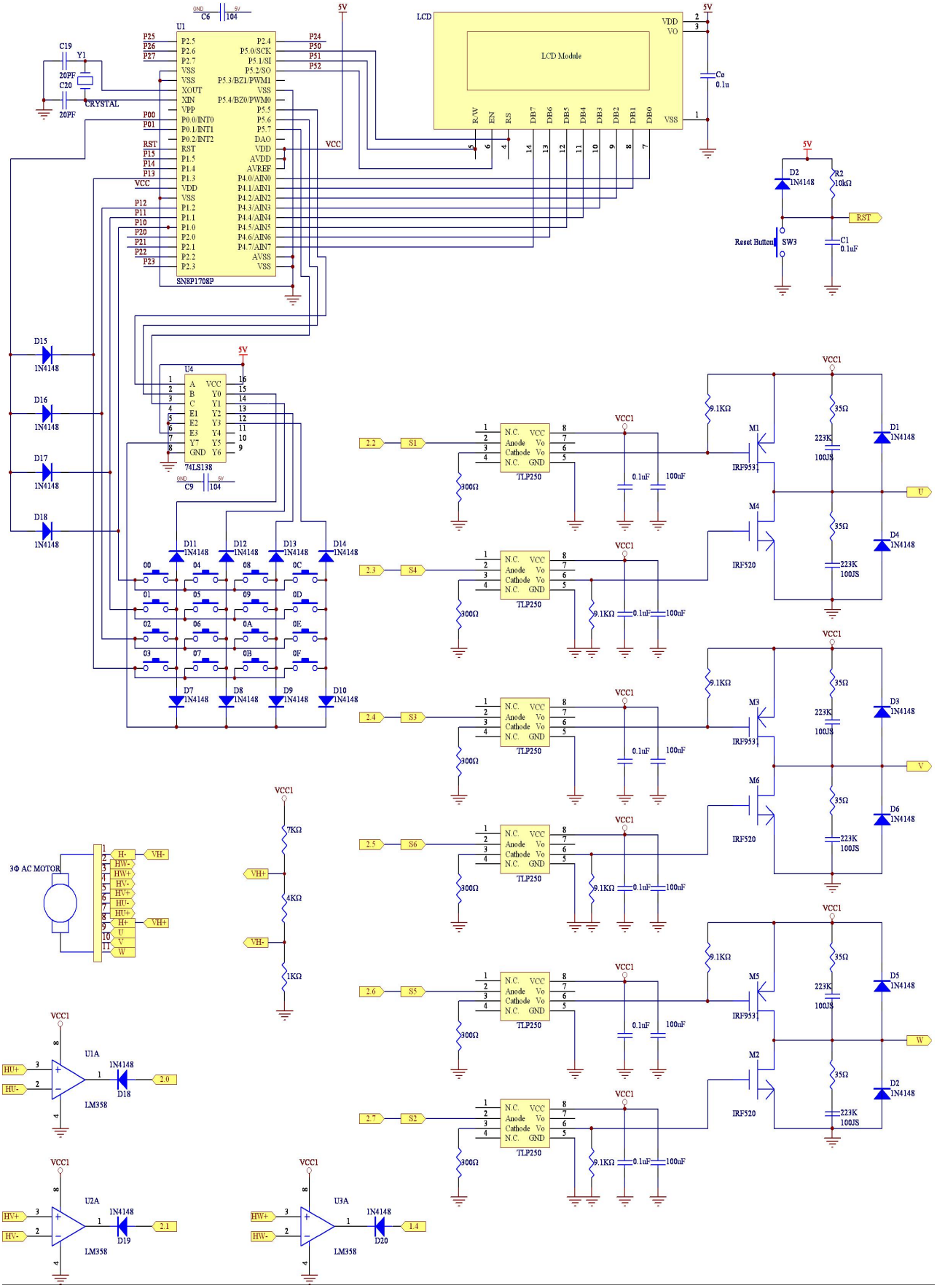

U1
SN8P1708P
LCD
LCD Module
U4
74LS138
D15
D16
D17
D18
D11
D12
D13
D14
D7
D8
D9
D10
S1
S4
S3
S6
S5
S2
TLP250
IRF953
IRF520
M1
M4
M3
M6
M5
M2
D1
D4
D3
D6
D5
D2
U
V
W
3Φ AC MOTOR
U1A
U2A
U3A
LM358
D18
D19
D20
Reset Button
SW3
RST
CRYSTAL
VCC1
5V

圖 3.11: 受控端電路圖

第四章

軟體架構

4.1 概述

本章將介紹系統的軟體部分。第二節將介紹發展此系統的軟體介面。第三節將依序說明本實驗的韌體程式碼，包括主程式及相關副程式。

4.2 軟體發展系統

執行 SASM199A 後，就會出現如圖 4.1之視窗。工具列 (toolbar) 和狀態列 (statusbar) 會主動出現，但是記憶體 (memory) 和系統暫存區 (register) 需要在下拉選單 [View] 中點選才會出現。在狀態列處會顯現目前爲模擬模式 (simu) 或是使用實體模擬器 (ICE)，其由軟體自行判斷後產生。在模擬模式下，可以組譯程式和執行程式如同 ICE 模式的情況，只是無法作硬體驗證。下拉選單的 [File] 和 [Edit] 都與微軟的軟體 WORD 中的功能一樣。只有 [View] 和 [Tools] 需要加以說明，其他部份若不是常見用法，就是很少用到。[View] 的下拉選單顯示在圖 4.1中，用來勾選顯示區。 顯示區 Toolbar 爲圖示工具按鍵的區域。顯示區 register 爲一些重要系統暫存器的顯示區，包含了所有工作暫存器，累積器 ACC，程式計數器 PCHL，所有的輸出入埠，和堆疊緩衝區 STACK。顯示區 Memory 爲資料記憶體 RAM 的顯示區，通常在單步執行時選用 [Auto] 來看即時變化的變數 ; 而選 [Mem Name] 可以看所有定義

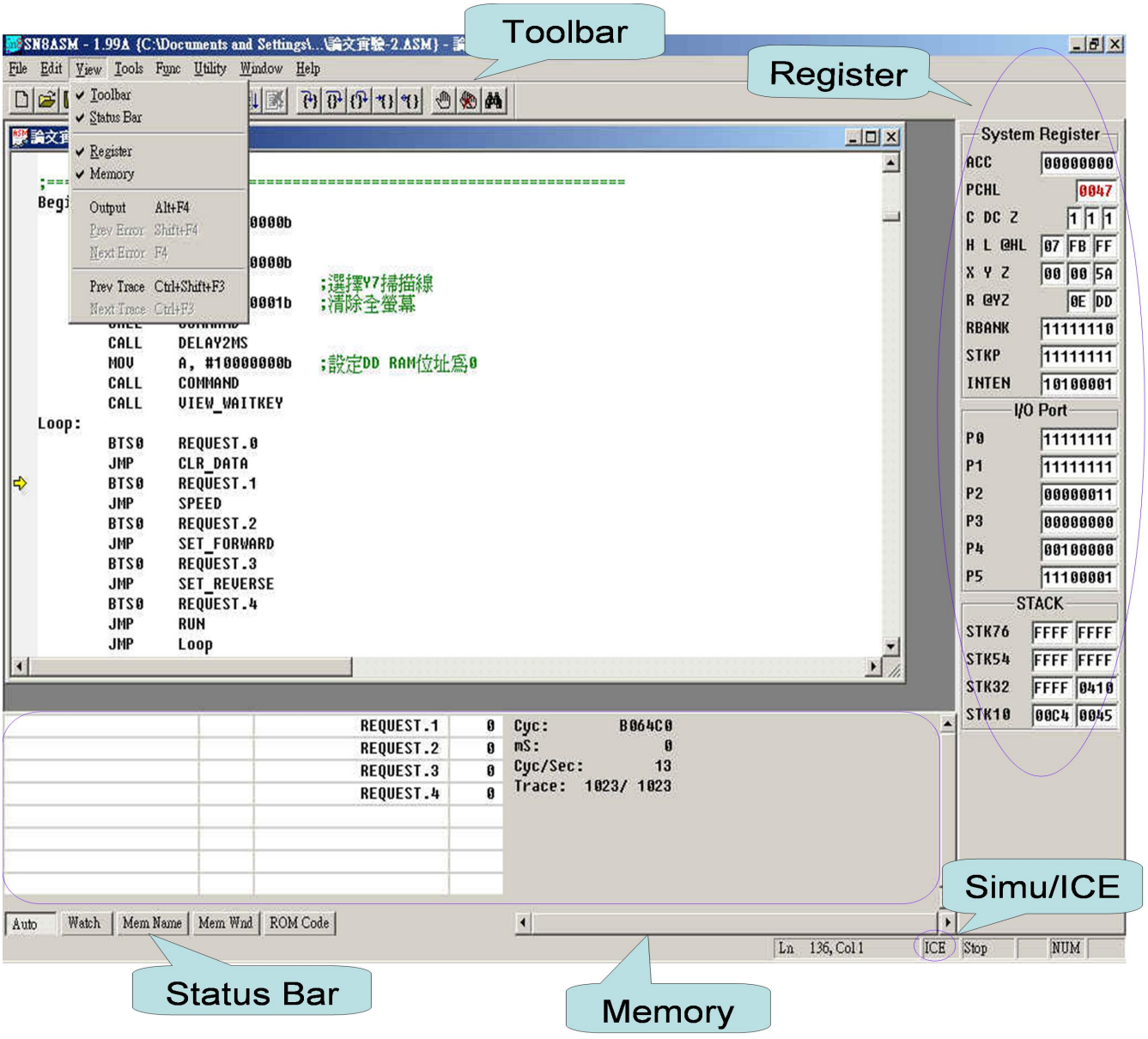


圖 4.1: SN8P1708 晶片之實體模擬器的程式發展軟體 [1]

變數的值 ; 選 [Mem wnd] 可以看所有 RAM 的值 ; 若是選 [ROM Code] 則是看程式碼。

其它關於 SN8P1708 晶片之實體模擬器程式發展軟體視窗詳細操作之介紹 , 可參考林錫寬教授所著掌握微控制器原理與技術使用 SN8P1700 系列晶片 [1] 一書。

4.3 利用 Sonix 之 SN8P1708 來實現三相永磁馬達之控制

此端是由 Sonix 公司所出產的 SN8P1708 來實現，最主要由撰寫韌體程式來實現，我們的目的是以 PC 撰寫好的 Assembler Code 經由 ICE 連接到受控電路，經由軟體鍵盤掃描來控制馬達的整反轉及調節速度，並配合 LCD 來顯示旋轉資訊。

其程式之撰寫流程有一定之步驟，第一步爲指定晶片之名稱，第二步爲撰寫中斷之巨集程式 .(Note: 巨集定義必須放在使用該巨集之前，即主程式之前 .)，第三步驟爲宣告所有使用到的變數所需要用到的記憶體位置與定義常數 . 第四步爲跳躍至主程式處 (也就是位址 0x10 處)，此處才真正爲主程式開始處，其詳細之程式碼可參考附錄 C.1 節。

以下各小節將一一介紹主程式與副程式撰寫之流程與方法 :

4.3.1 主程式架構

韌體端組語的主程式流程圖如圖 4.2所示，程式流程之步驟第一步爲使系統初使化，設定輸入與輸出腳位，且清除所有輸出資料緩衝暫存器。第二步爲清除所有資料暫存器。第三步爲致能外部中斷 INT0 與計時器 TC0 中斷，並開啓中斷總開關。第四步爲設定 LCD 的顯示格式爲 8 位元資料存取 / 雙列字 /5*10 點矩陣字型 / 顯示器 ON/ 游標被顯示在位址計數器。第五步爲要求馬達停止，使六步變頻器開關上下臂的 PMOS 與 NMOS 同時關閉。第六步爲設定顯示字元起始位址爲 00h，並在 LCD 上顯示 ”Wait for Command”。第七步爲致能 Y7 掃描線後，進入 Loop 迴圈，等待命令要求。此時按下任何按鍵則進入 INT0 中斷做軟體掃描鍵盤。其詳細之程式可參考附錄之 C.2 節。

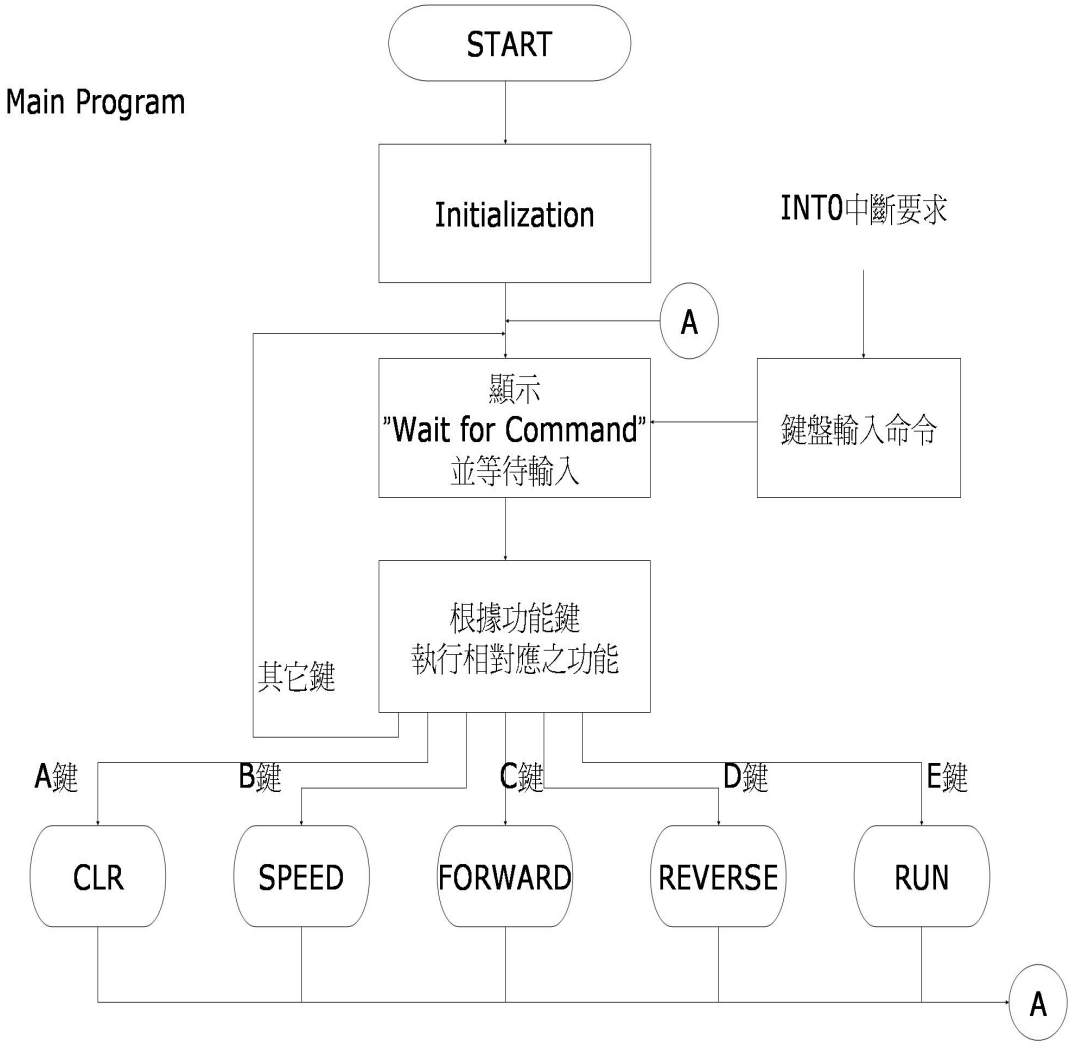

Main Program
START
Initialization
INT0中斷要求
A
顯示
"Wait for Command"
並等待輸入
鍵盤輸入命令
根據功能鍵
執行相對應之功能
其它鍵
A鍵
B鍵
C鍵
D鍵
E鍵
CLR
SPEED
FORWARD
REVERSE
RUN
A

圖 4.2: 韌體程式流程圖

主程式在標記 loop 處會一直偵測 Request 之要求，關於 Request 說明如下表 (表 4.1) 所示，當變數 Request 的位元 0 被設定爲 1 時 (對應按鍵 A)，則跳至 CLR 副程式做清除資料之動作 ;Request.1=1 時 (對應按鍵 B)，則跳至 SPEED 副程式做設定速度之動作 ; 當 Request.2=1(對應按鍵 C)，時則跳至 FORWARD 副程式設定馬達爲正轉 ; 當 Request.3=1 時 (對應按鍵 D) 則跳至 REVERSE 副程式設定馬達爲反轉 ; 當 Request.4=1 時 (對應按鍵 E) 則跳至 RUN 副程式，根據前面所設定之資料做馬達運轉之動作。

bit	7	6	5	4	3	2	1	0
功能			ENTER	RUN	REVERSE	FORWARD	SPEED	CLR
按鍵			F	E	D	C	B	A

表 4.1: Request 說明表

4.3.2 副程式 CLR 架構

副程式 CLR 最主要是做資料清除之動作 , 其程式流程步驟分別爲第一步是清除 Request.0 爲 0，且 LCD 顯示 ”SURE CLEAR DATA?”. 第二步爲選擇 Y7 掃描線等待輸入。最後一步爲判斷若按下 F 鍵則清除所有馬達資料，LCD 顯示 ”CLEAR OK” 並回到 Begin 處，若爲其他功能鍵則返回 Begin 處去執行相對應之動作。其動作流程圖與程式碼可參考圖 5.5 與附錄 C.3 節。

4.3.3 副程式 SPEED 架構

副程式 SPEED 之動作爲設定馬達之旋轉速度，其動作流程圖與程式碼可參考圖 5.8 與附錄 C.4。程式流程步驟分別爲第一步是清除 Request.1 爲 0, 且 LCD 顯示 ”PLEASE INPUT SPEED :”。第二步爲選擇 Y7 掃描線且把 KEYTYPE.0 設爲 1 做數字鍵盤掃描，等待輸入速度，連續輸入四位數字視爲有效速度輸入，若爲其他功能鍵則視爲無效，並返回 Loop 處作其相對應之功能。第三步爲當速度資料輸入完成後則呼叫 UPDATESPPED 副程式做速度更新之動作 . 第四步爲當速度設定完成後，LCD 顯示 ”Speed Set OK”，並把

KEYTYPE.0 清除爲 0 後返回主程式 Begin 處。副程式 UPDATESPEED 之程式碼請參考附錄 C.4.1 節。

副程式 UPDATESPEED 可分成三個步驟 : 第一步爲把 BUFFER 的值存入擺放速度資料的暫存器 SBUFFER，然後再透過副程式 BIN2BCDtoHEX 把 10 進制的速度資料轉換成 16 進制之速度資料，方法爲先將 BUFFER[3] 做減 1 之動作，每減一次 1 就加 1000 到 SPEEDBUFFER 內，當減至 0 時再減 1 發生借位，則跳到 DO_HEX_100 做加 100 至 SPEEDBUFFER 之動作，方法與 DO_HEX_1000 相同。HEX_10 內的做法爲將 BUFFER[1] 內的值乘 10 加至 SPEEDBUFFER 內，HEX_1 內的做法爲直接將 BUFFER[0] 內的值加至 SPEEDBUFFER 內。

本實驗把速度規劃爲 2048 個單位，利用多功能計時器 TC0(表 4.2) 來做速度設定之動作，利用其預除值與初始值自動載入之功能，可設定中斷週期時間藉以控制馬達旋轉速度。

程式流程之第二步爲判斷速度是否大於 2047，若大於 2047 則視爲 2047，因爲速度選擇範圍爲 0 ～ 2047; 第三步驟爲判斷在哪個速度區間，然後根據結果去設定 TC0M 暫存器，其設定速度可參考下表 (表 4.3)，把判斷出的預除值存入 TC0M，且把 SPEEDBUFFER 存入 TC0R。

4.3.4 副程式 FORWARD 架構

副程式 FORWARD 之動作爲設定馬達之轉向爲順時針方向，其程式流程步驟分別爲第一步是清除 Request.2 爲 0，且 LCD 顯示 ”SURE FORWARD ?”。第二步爲選擇 Y7 掃描線等待輸入。第三步爲假若按下 F 鍵則馬達設定爲正轉，LCD 顯示 ”FORWARD OK” 並回到 Begin 處，若爲其他功能鍵則返回 Begin 處去執行相對應之動作。其動作之流程圖與程式碼可參考圖 5.11 與附錄 C.5 節。

4.3.5 副程式 REVERSE 架構

TC0M
(0XXX XXXX)

7	6	5	4	3	2	1	0
TC0ENB	TC0rate2	TC0rate1	TC0rate0	0	ALOAD0	TC0OUT	PWM0OUT

PWM0OUT: 0表『禁能』PWM輸出。
1表『致能』PWM輸出。

TC0OUT: 0表『禁能』週期性方波輸出。
1表『致能』週期性方波輸出。並『禁能』PWM輸出。

ALOAD0: 0表『禁能』初始值自動載入。
1表『致能』初始值自動載入。

TC0ENB: 0表『禁能』計時器。
1表『致能』計時器。

(TC0rate2, TC0rate1, TC0rate0)用來表示計時器TC0計數時脈頻率的預除值d_{TC0}，但是$d_{TC0}=2^{(8-rC0)}$，其中r_{C0}=(TC0rate2, TC0rate1, TC0rate0)。

(TC0rate2, TC0rate1, TC0rate0)	預除值d_{TC0}
000	256
001	128
010	64
011	32
100	16
101	8
110	4
111	2

表 4.2: 多功能計時器 TC0 說明表 [1]

副程式 REVERSE 之動作爲設定馬達之轉向爲逆時針方向，其程式流程步驟分別爲第一步是清除 Request.3 爲 0，且 LCD 顯示 ”SURE REVERSE ?”。第二步爲選擇 Y7 掃描線等待輸入。第三步爲假若按下 F 鍵則馬達設定爲反轉，LCD 顯示 ”REVERSE OK” 並回到 Begin 處，若爲其他功能鍵則返回 Begin 處去執行相對應之動作。其動作之流程圖與程式碼可參考圖 5.14 與附錄 C.6 節。

區間	TC0rate2,TC0rate1,TC0rate0	速度
Interval0	000	0 ～ 255
Interval1	001	256 ～ 511
Interval2	010	512 ～ 767
Interval3	011	768 ～ 1023
Interval4	100	1024 ～ 1279
Interval5	101	1280 ～ 1535
Interval6	110	1536 ～ 1791
Interval7	111	1792 ～ 2047

表 4.3: TC0M 速度設定表

4.3.6 副程式 RUN 架構

副程式 RUN 之功能爲執行馬達運轉之動作，使用 TC0 來做計時中斷，設定計時器中斷之週期，每發生一次計時中斷時就送出一六步方波中一狀態至馬達，故可由設定中斷之週期來調整馬達之轉速，其程式步驟流程的第一部爲清除 Request.4 爲 0，然後判斷馬達爲正轉或反轉。第二步爲假若爲正轉就跳至副程式 FORWARDSTEP 執行，若爲反轉則跳至 BACKWARDSTEP 執行，且 LCD 會在第二列顯示速度資料。第三步爲選擇 Y7 掃描線並且開始計時 TC0 計時器，若按下 F 鍵則停止計時計時器 TC0 並返回 Begin 處，若爲其他鍵則繼續計時 TC0。關於 TC0 中斷時所執行之副程式將在下節所介紹。其詳細之動作流程圖與程式碼可參考圖 5.18 與附錄 C.7 節。

4.3.7 中斷副程式架構

當系統接受到中斷源信號時，立即要執行中斷動作 : 就是跳到 ROM 位址爲 0008h 處，並立即將 GIE(中斷總開關) 設爲 0，不讓後續的中斷源信號影響目前的中斷副程式。所以當中斷副程式執行完後，除了返回原程式外，還

要致能系統中斷功能，所以要執行 GIE=1 的設定。其相關步驟如下所示 :

```
;************************************
; Interrupt Vector on ROM address 0x8
;************************************
ORG 0x8                          ; 中斷位址保留區
Begin_Interrupt                  ; 中斷開始巨集程式
BTS0 INTRQ.0                     ; 判斷是否爲 INT0 外部中斷
JMP INT0_IRS
BTS0 INTRQ.5                     ; 判斷是否爲 TC0 計時器中斷
JMP TC0_IRS
```

當發生中斷時程式會跳至位址 0008h 處，然後依序判斷爲何種類之中斷要求，確認爲此中斷時則跳至相對應之中斷副程式。

此實驗有 INT0 與 TC0 二種中斷，所做的動作分別爲鍵盤掃描與送出六步方波至馬達，首先就 INT0 中斷副程式之步驟說明如下 :

程式流程可參加附錄之 C.8 節。其程式步驟之第一步爲把 LCD 的游標顯示設定爲第二列，清除 SET_OK.0 爲 0，標籤 FUNCTION_KEY 處爲令掃描線從 Y3 開始掃描。第二步爲呼叫副程式 SCAN_DISP 與 KEY_SCAN 爲做切換掃描線與做軟體掃描鍵盤，每做完一次掃描線的鍵盤掃描會把掃描線的值減 1 以切換掃描線，當掃描線的值爲 0 時 (也就是 Y0 掃描線) 減 1 發生借位，則跳回標籤 FUNCTION_KEY 處，繼續從 Y3 掃描線開始的下一輪的掃描。第三步爲判斷壓下按鍵爲何鍵，若爲 A 鍵則把 REQUEST.0 設爲 1; 若爲 B 鍵則把 REQUEST.1 設爲 1; 若爲 C 鍵則把 REQUEST.2 設爲 1; 若爲 D 鍵則把 REQUEST.3 設爲 1; 若爲 E 鍵則把 REQUEST.4 設爲 1; 若爲 F 鍵則除能 TC0 計時器計時。設定相對應功能後，然後跳至標籤 KEYEND 處，清除 INT0 中斷要求，然後執行結束中斷巨集程式。

假若爲副程式 SPEED 時的數字鍵盤掃描，LCD 螢幕的第二列會即時地顯示所輸入之數字鍵。可參考附錄 C.8.1 節之程式。

此實驗 TC0 多功能計時器的功用爲當計時器發生溢位中斷時，執行送出六步波方至馬達之動作，將在下列程式加以說明 :

程式流程步驟之第一步爲當發生 TC0 中斷時會呼叫副程式 STEPMOVE 做送出六步方波至馬達之動作，當執行完 STEPMOVE 副程式時會清除 TC0 的中斷要求然後執行結束中斷巨集程式。第二步爲執行副程式 STEPMOVE，有關副程式 STEPMOVE 之程式碼可參考附錄之 C.8.2 節。

副程式 STEPMOVE 程式之步驟的第一步爲先判斷方向位元，第二步爲判斷霍爾回授的電壓準位，藉以判斷馬達轉子的位置。第三步爲根據所判斷得馬達轉子位置，參考表 2.3 之表格，送出正確之六步方波，使馬達之合成之轉矩漣波爲最大。

第五章

實驗結果

本系統主要是做三相永磁馬達之正反轉及速度控制，下圖 5.1爲整個系統的外觀。

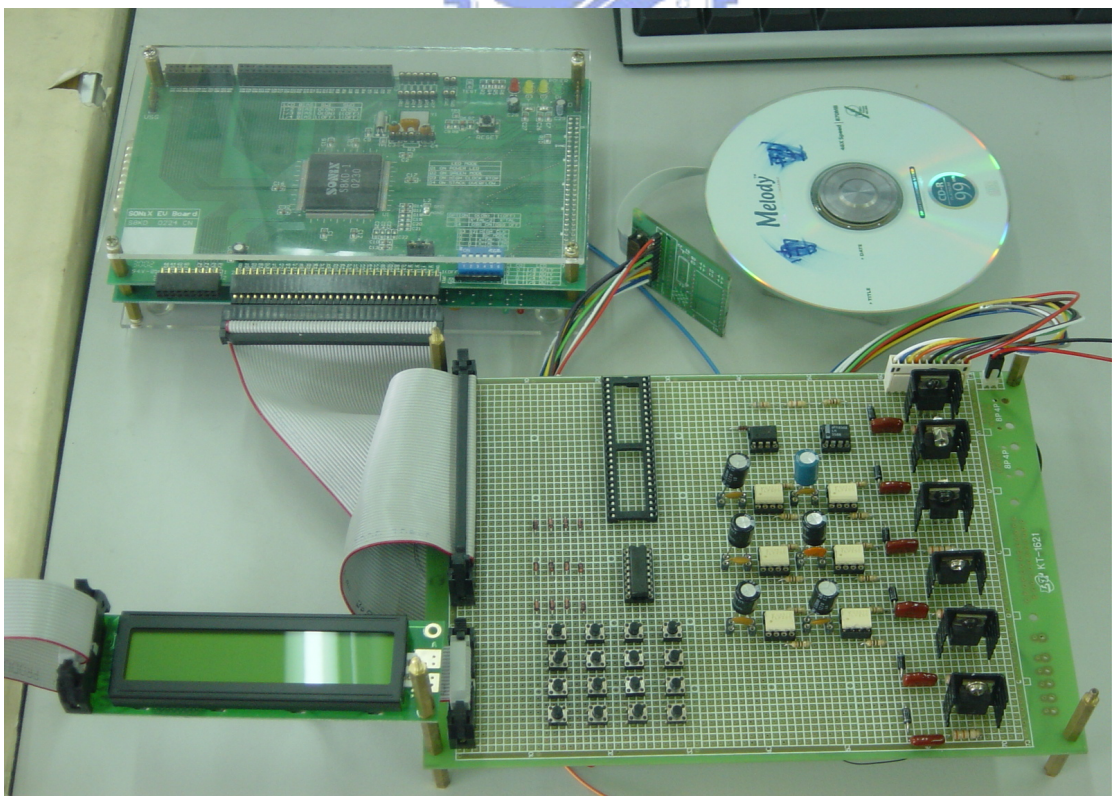


圖 5.1: 系統外觀

操作的步驟如 5.2之流程圖所示，開機後程式初始化，然後 LCD Display 顯示 "Wait for Command :"(圖 5.3) 時等待命令輸入，此時可以按下 A、B、C、D、E 之功能鍵做清除資料、設定速度、設定正轉、設定反轉與執行運轉之動作，按其它鍵則視爲無效繼續等待命令輸入。

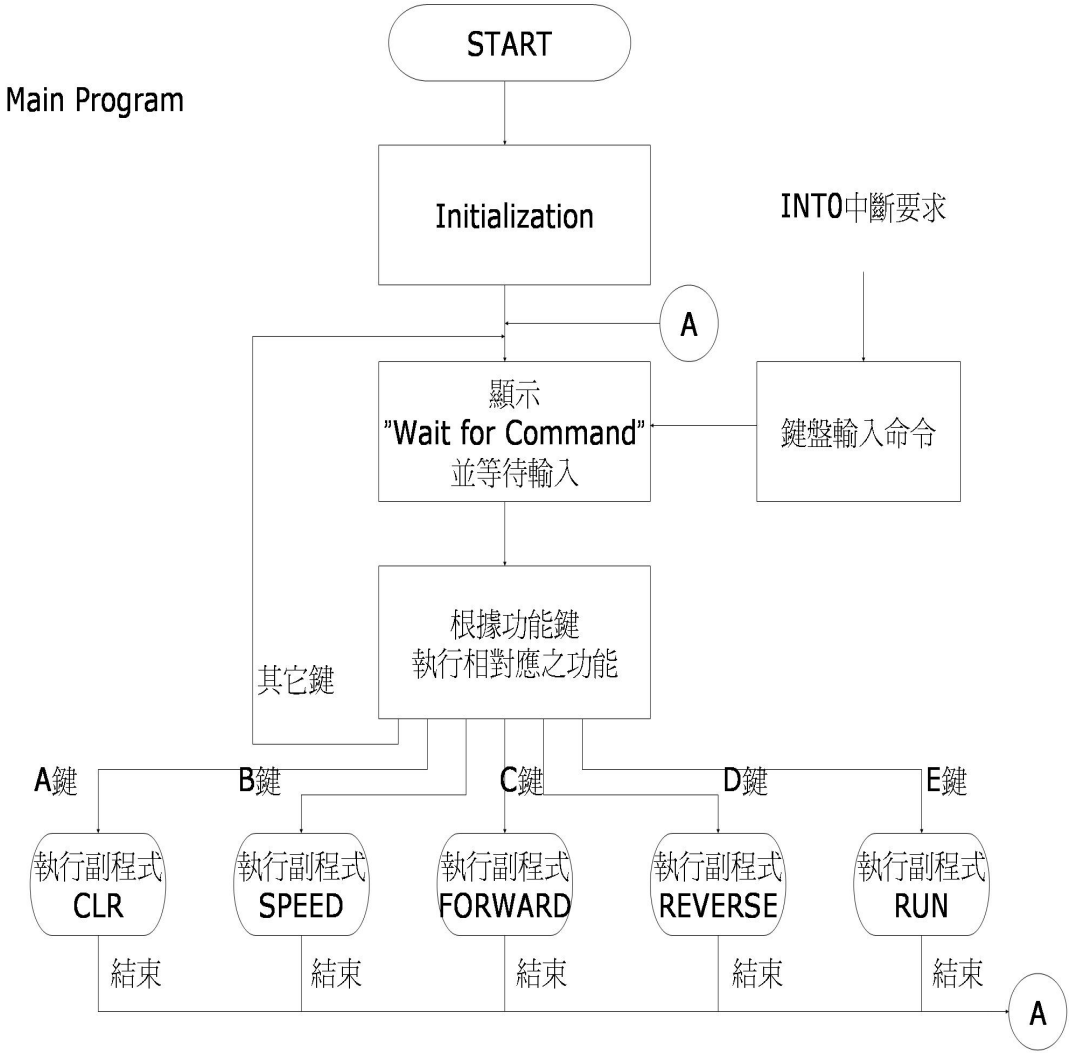


圖 5.2: 主程式流程圖

圖 5.3: LCD Displayer 顯示 "Wait for Command :"

當按下 A 鍵時此時 LCD Display 顯示 "SURE CLEAR DATA?"(圖 5.4)，A 鍵清除資料之動作流程如下所示 (圖 5.5):

圖 5.4: LCD Displayer 顯示 "SURE CLEAR DATA?"

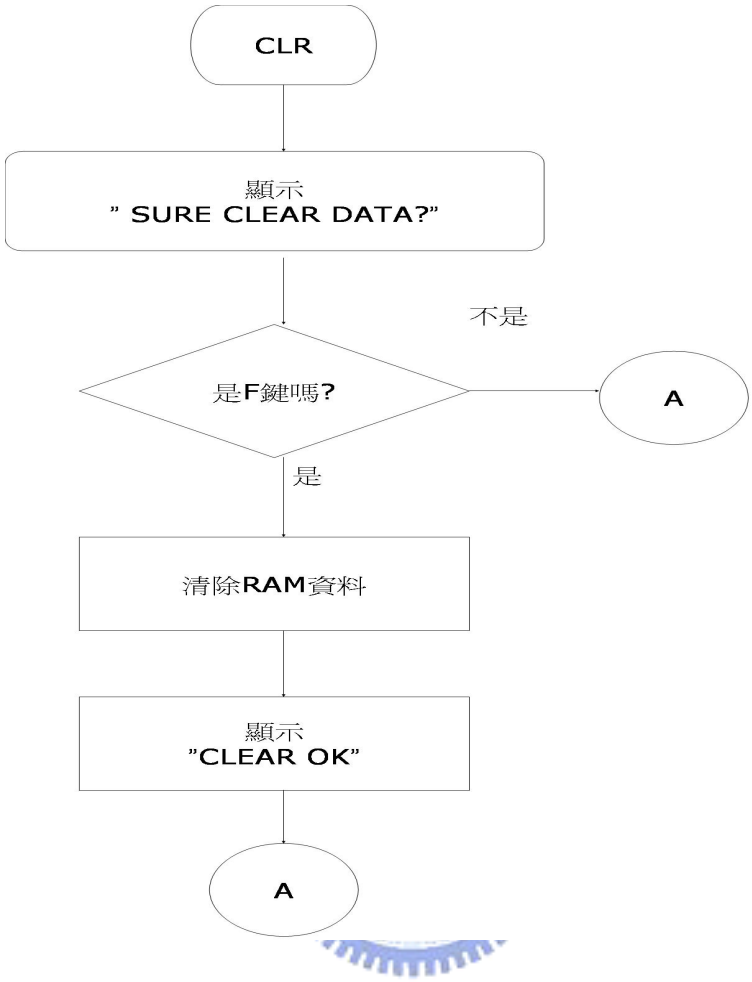


圖 5.5: 副程式 CLR 之流程圖

此時若按下 F 鍵則爲確認清除馬達運轉資料並顯示 "CLEAR OK"(圖 5.6) 兩秒後返回流程圖 A 處，若爲其它功能鍵則返回流程圖 A 處去執行相對應之功能，若爲數字鍵則輸入無效。

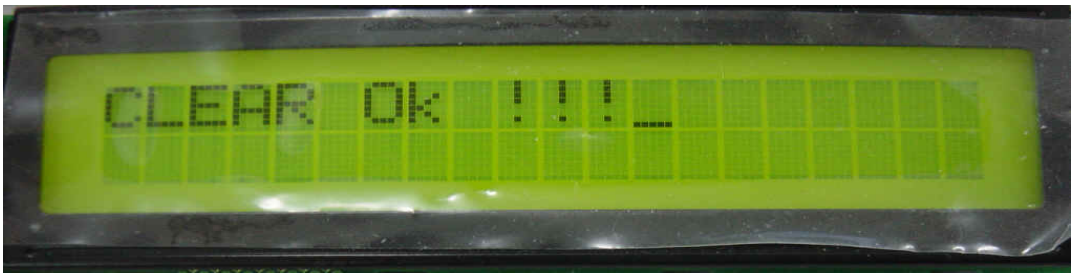


圖 5.6: LCD Display 顯示 "CLEAR OK !!!"

當按下 B 鍵時此時 LCD Display 顯示 "Please Input Speed :"(圖 5.7)，設定速度之動作流程圖如下所示 (圖 5.8):

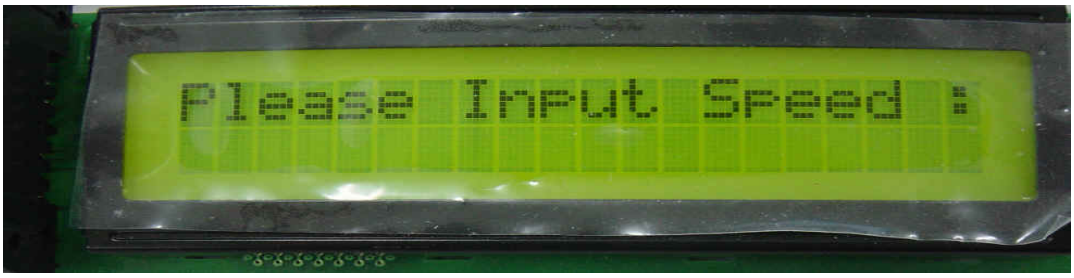


圖 5.7: LCD Display 顯示 "Please Input Speed :"

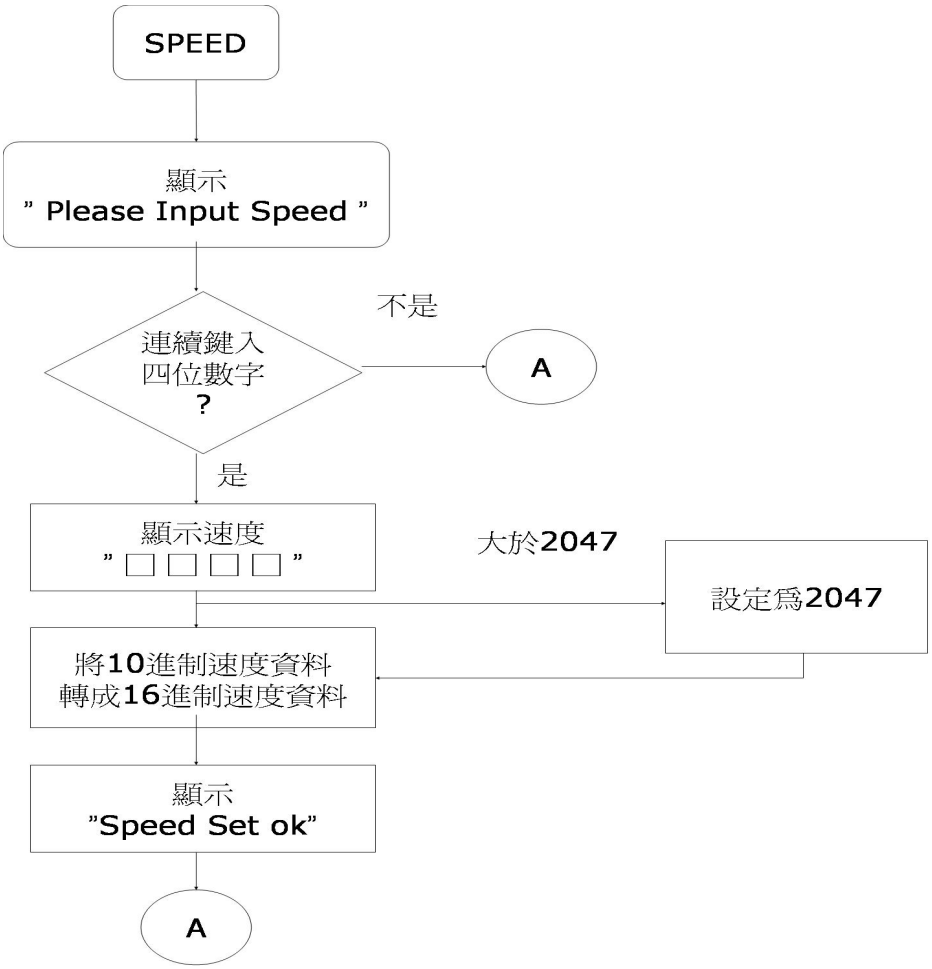


圖 5.8: 副程式 SPEED 之流程圖

此時若連續輸入四位數字輸入則視爲有效輸入速度同時把所輸入之數字顯示於 LCD Display 上，並把所輸入之 10 進制速度資料轉換成 16 進制速度資料，最後顯示 "Speed Set ok!!!"(圖 5.9) 兩秒後回到流程圖 A 處，若爲其它功能鍵則返回流程圖 A 處去執行相對應之功能。

圖 5.9: LCD Display 顯示 "Speed Set ok !!!"

當按下 C 鍵時此時 LCD Display 顯示 "Sure Forward ?"(圖 5.10)，設定正轉之動作流程圖如下所示 (圖 5.11):

圖 5.10: LCD Display 顯示 "Sure Forward ?"

此時若按下 F 鍵則爲確認設定馬達正轉並顯示 "Forward ok !!!"(圖 5.12) 兩秒後返回流程圖 A 處，若爲其它功能鍵則返回流程圖 A 處去執行相對應之功能，若爲數字鍵則輸入無效。

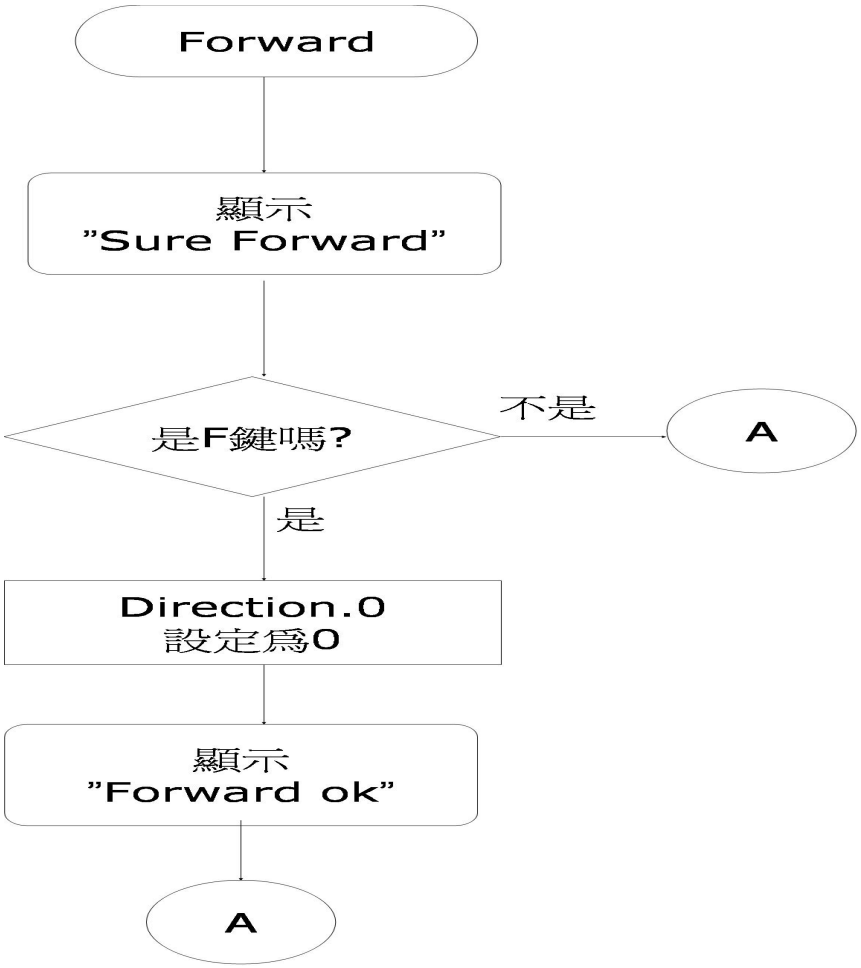


圖 5.11: 副程式 Forward 之流程圖

圖 5.12: LCD Display 顯示 ”Forward ok !!!”

當按下 D 鍵時此時 LCD Display 顯示 "Sure Reverse ?"(圖 5.13)，設定反轉之動作流程圖如下所示 (圖 5.14):

圖 5.13: LCD Display 顯示 "Sure Reverse ?"

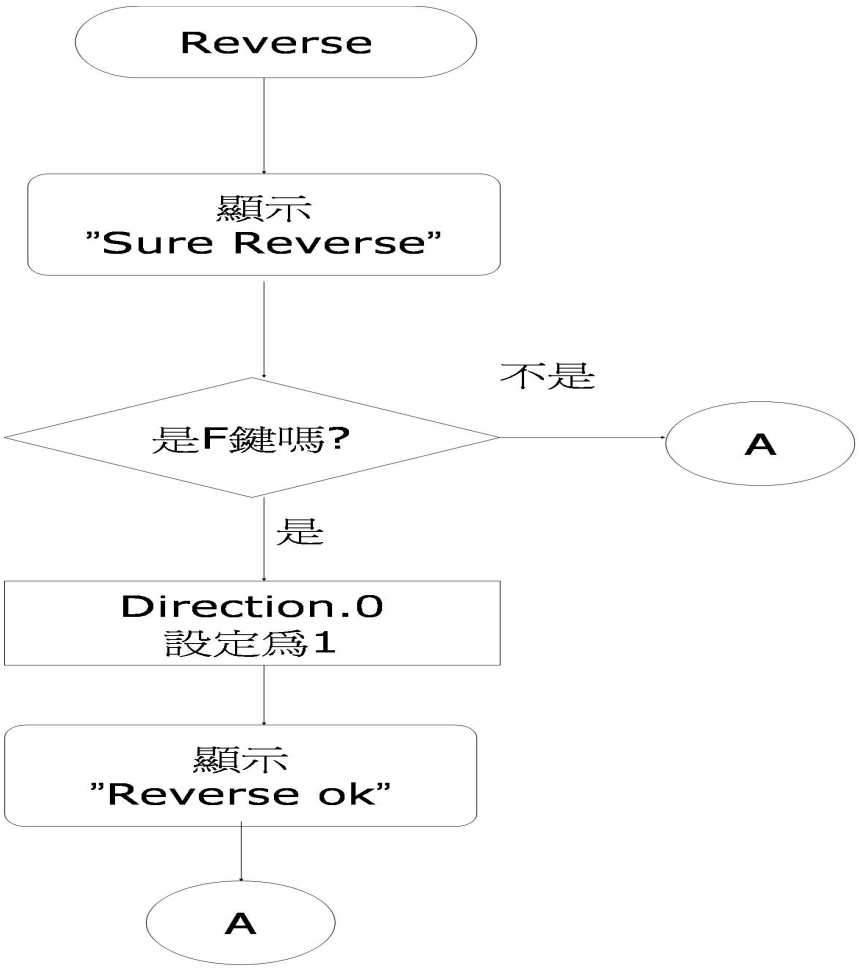


圖 5.14: 副程式 Reverse 之流程圖

此時若按下 F 鍵則爲確認設定馬達反轉並顯示 "Reverse ok !!!"(圖 5.15) 兩秒後返回流程圖 A 處，若爲其它功能鍵則返回流程圖 A 處去執行相對應之功能，若爲數字鍵則輸入無效。

圖 5.15: LCD Display 顯示 "Reverse ok !!!"

當按下 E 鍵時此時 LCD Display 會根據之前設定的資料顯示顯示 "Forward Speed : □□□□ "(圖 5.16) 執行正轉或 "Reverse Speed : □□□□ "(圖 5.17) 執行反轉，執行運轉之動作流程圖如下所示 (圖 5.18): 此時若按下 F 鍵則馬達

圖 5.16: LCD Display 顯示 "Forward Speed : □□□□ "

圖 5.17: LCD Display 顯示 "Reverse Speed : □□□□ "

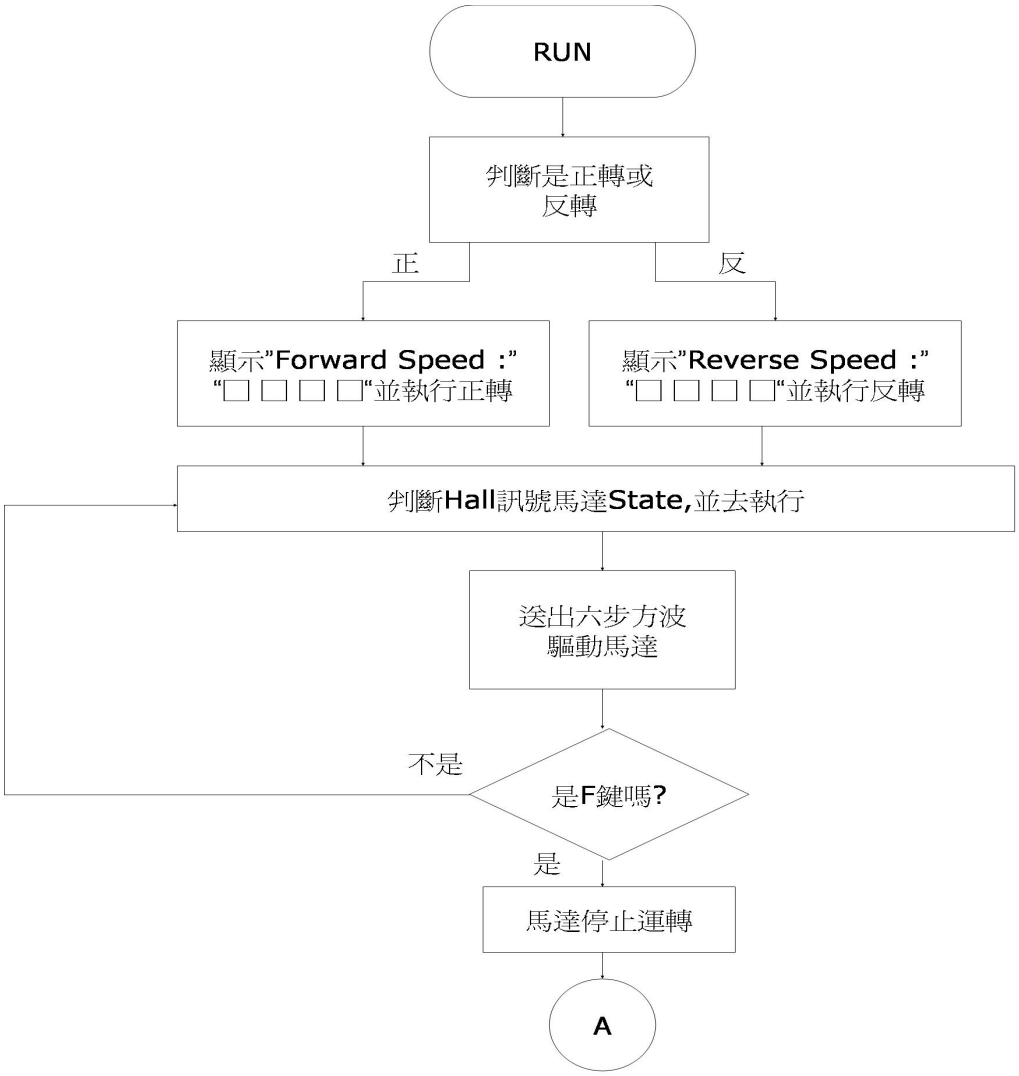


圖 5.18: 副程式 RUN 之流程圖

停止運轉並跳回流程圖 A 處，若爲其它鍵則無效，馬達繼續運轉。

第六章

結論與未來展望

6.1 結論

實現三相永磁馬達之控制需要多方面之配合。除了硬體與韌體外，對交流馬達之驅動方式與理論也要有一定之認識，各部分相互配合才能使馬達正常運作。

本文是從 SN8P1708 晶片的介紹開始，藉由使用閉迴路的方式 (Close Loop Control) 控制三相永磁馬達，在硬體方面，透過馬達驅動週邊介面電路的設計，搭配交流馬達驅動之六步變頻器電路、電源供應器…等，完成三相永磁馬達控制系統之建構。韌體方面則利用組合語言來實現三相永磁馬達之控制法則。在操作介面上，利用鍵盤掃描輸入，搭配 LCD DISPLAY 提示操作，方便使用者更容易於系統之操作。

6.2 未來展望

本論文所提出的三相永磁馬達控制系統 , 功能尚有發展的空間。未來發展方向可分爲兩大重點 :

1. 透過實驗方式量測得實際馬達轉速，固定供應電壓，建立一轉速對計時器 TC0M 設定之表格，將可準確控制馬達之轉速。

2. 可實際用來驅動光碟機或硬碟之馬達。

3. 可加入脈波寬度調變 (PWM) 之功能來控制馬達轉速。

附錄 A

SN8P1708 微控器的系統暫存器 [1]

以下的系統暫存器配置表示將低位址列於左邊，這樣可以和微控器的發展系統軟體中的記憶體內容查詢的表格一致。

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	L	H	R	Z	Y	X	PFLAG	RBANK	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B	DAM	ADM	ADB	ADR	SIOM	SIOR	SIOB	-	-	-	-	-	-	-	-	-
C	P1W	P1M	P2M	-	P4M	P5M	-	-	INTRQ	INTEN	OSCM	-	-	TC0R	PCL	PCH
D	P0	P1	P2	-	P4	P5	-	-	T0M	T0C	TC0M	TC0C	TC1M	TC1C	TC1R	STKP
E	-	-	-	-	-	-	@HL	@YZ	-	-	-	-	-	-	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

表 A.1: 系統暫存器配置表 [1]

附錄 B

SN8P1708 微控器的指令集

註 : 指令週期爲 "1/2" 表示平常爲 1，當跳躍時爲 2。

指　令	說　明	C	DC	Z	週期
NOP	無動作	--	--	--	1
MOV A, M	A←M	--	--	√	1
MOV M, A	M←A	--	--	--	1
MOV A, I	A←I	--	--	√	1
B0MOV M, I	M(工作暫存器)←I	--	--	--	1
MOVC	(R:A)←ROM[X:Y:Z]	--	--	--	2
XCH A, M	A←→M	--	--	--	1
B0MOV A, M	A←M (bank 0)	--	--	√	1
B0MOV M, A	M (bank 0)←A	--	--	--	1
B0XCH A, M	A←→M (bank0)	--	--	--	1
AND A, M	A←A & M	--	--	√	1
AND M, A	M←A & M	--	--	√	1
AND A, I	A←A & I	--	--	√	1

表 B.1: 指令集 [1]

指　令	說　明	C	DC	Z	週期
OR A, M	A←A | M	--	--	√	1
OR M, A	M←A | M	--	--	√	1
OR A, I	A←A | I	--	--	√	1
XOR A, M	A←A ^ M	--	--	√	1
XOR M, A	M←A ^ M	--	--	√	1
XOR A, I	A←A ^ I	--	--	√	1
CLR M	M←0	--	--	--	1
BCLR M.b	M.b←0	--	--	--	1
BSET M.b	M.b←1	--	--	--	1
B0BCLR M.b	M.b (bank 0)←0	--	--	--	1
B0BSET M.b	M.b (bank 0)←1	--	--	--	1
RLCM M	M←(M.6:C), C←M.7	√	--	--	1
RLC M	A←(M.6:C), C←M.7	√	--	--	1
RRCM M	M←(C:M.7.1), C←M.0	√	--	--	1
RRC M	A←(C:M.7.1), C←M.0	√	--	--	1
SWAPM M	(M.3.0)←→(M.7.4)	--	--	--	1
SWAP M	A←(M.3.0:M.7.4)	--	--	--	1
ADD A, M	A←A + M	√	√	√	1
ADD M, A	M←A + M	√	√	√	1
ADD A, I	A←A + I	√	√	√	1
SUB A, I	A←A − M	√	√	√	1
SUB M, A	M←A − M	√	√	√	1

表 B.2: 指令集 (續)

指　令	說　明	C	DC	Z	週期
SUB A, I	A←A − I	√	√	√	1
ADC A, M	A←A + M + C	√	√	√	1
ADC M, A	M←A + M + C	√	√	√	1
SBC A, M	A←A − M − not(C)	√	√	√	1
SBC M, A	M←A − M − not(C)	√	√	√	1
MUL A, M	(R:A)= A * M	--	--	√	2
B0ADD M, A	M (bank 0)←A + M (bank 0)	√	√	√	1
DAA	BCD調整	√	--	--	1
CALL d	Stack←PC, PC← d	--	--	--	2
JMP d	PC←d	--	--	--	2
RET	PC←Stack	--	--	--	2
RETI	PC ← Stack，致能主中斷	--	--	--	2
BTS0 M.b	若 M.b = 0 則跳一行	--	--	--	1 / 2
BTS1 M.b	若 M.b = 1 則跳一行	--	--	--	1 / 2
INCS M	A←M + 1，若 A = 0 則跳一行	--	--	--	1 / 2
INCMS M	M←M + 1，若 M = 0 則跳一行	--	--	--	1 / 2
DECS M	A←M − A，若 A = 0 則跳一行	--	--	--	1 / 2
DECMS M	M←M − 1，若 M = 0 則跳一行	--	--	--	1 / 2
CMPRS A, I	若 A = I 則跳一行	√	--	√	1 / 2
CMPRS A, M	若 A = M 則跳一行	√	--	√	1 / 2
B0BTS0 M.b	若 M.b = 0 (bank 0) 則跳一行	--	--	--	1 / 2
B0BTS1 M.b	若 M.b = 1 (bank 0) 則跳一行	--	--	--	1 / 2
PUSH	儲存工作暫存器	--	--	--	1
POP	取回工作暫存器	--	--	--	1

表 B.3: 指令集 (續)

附錄 C

系統之程式碼

C.1 韌體程式開始之設定流程

```
;step1
chip SN8P1708OTP


      ;step2
Begin_Interrupt macro              ; 開始中斷巨集
.data
stack_A ds 1                       ; stack for ACC
.code
PUSH                               ; 將所有的工作暫存器存入系統隱藏緩衝區
MOV stack_A, A                     ; 把 ACC 存入 stack_A
endm

End_Interrupt macro                ; 結束中斷巨集
MOV A, stack_A                     ; 把 ACC 的值取回
POP                                ; 從系統隱藏緩衝區取出所有工作暫存器
```

```
RETI
endm


      ;step3
;************** VARIABLES address*******************************
.data
SBUFFER ds 4
Buffer ds 4
DisplayBit ds 1                    ; (P57, P56, P55)
Timer ds 1
LastKey ds 1
KeyIn ds 1
ReleaseHold ds 1
KeyBuffer ds 1
R1 ds 1
Delta1 ds 1
Delta2 ds 1
Delta3 ds 1
DIRECTION ds 1
SPEEDBUFFER ds 2
KEYTYPE ds 1
SET_OK ds 1
REQUEST ds 1
.Code
;************** CONSTANTS ****************************
TimeC2 = 1                         ; constant for Delta2 in subprogam DELAY1MS
TimeC3 = 1                         ; constant for Delta3 in subprogam DELAY3


      ;step4
JMP Initialization
```

C.2　主程式

```
    ;step1
ORG 0x10
Initialization:
MOV A, #00000111b
OR P0UR, A              ; pin P0.0 ~ P0.2 as PULL-HIGH
MOV A, #00011111b
OR P1UR, A              ; pin P1.0 ~ P1.4 as PULL-HIGH (for OTP chip)
MOV A, #11100000b
MOV P1M, A              ; pin P1.0 ~ P1.4 as inputs , P1.4 as Hallw
MOV A, #00000011b
OR P2UR, A
MOV A, #11111100b
MOV P2M, A              ; pin P2.0 ~ P2.1 as inputs, P2.0 as Hallu,
                          P2.1 as Hallv, pin P2.2 ~ P2.7 as outputs
MOV A, #11100111b
MOV P5M, A              ; pin P5.0 ~ P5.2, P5.5 ~ P5.7 as outputs,
                          5.2=E , 5.1=RW,5.0=RS
MOV A, #11111111b
MOV P4M, A              ; pin P4.0 ~ P4.7 as outputs,
                          as LCD DB0 ~ DB7
BSET OSCM.6             ; reset Watch Dog Timer, WDT=0
CLR P5
CLR P4
CLR P2
CLR P1
```

```
    ;step2
CLR TC0C
CLR TC0R
CLR KEYTYPE
CLR REQUEST
CLR BUFFER
CLR BUFFER+1
CLR BUFFER+2
CLR BUFFER+3
CLR DisplayBit
B0MOV X, #0
CLR ReleaseHold
BSET ReleaseHold.1
MOV A, #0xFF
MOV LastKey, A

    ;step3
BSET INTEN.0                ; enable INT0 interrupt
BSET INTEN.5                ; enable TC0 interrupt
BSET STKP.7                 ; enable Global Interrupt Handler

    ;step4
CALL DELAY5MS               ; Wait for LCD Power-on Ready
MOV A, #00111111b           ; 採用 8 位元資料存取 / 雙列字 /5*10 點矩
                              陣字型
CALL COMMAND
MOV A, #00001110b           ; 顯示器 ON/ 游標被顯示在位址計數器
CALL COMMAND

    ;step5
```

```
MOV A, #01010100b
XOR A, P2
AND A, #11111100b
XOR P2, A                       ;for motor stop


    ;step6
MOV A, #00000001b               ; 清除全螢幕
CALL COMMAND
CALL DELAY2MS
MOV A, #10000000b               ; 設定 DD RAM 位址爲 0
CALL COMMAND
CALL VIEW_WAITKEY
;================================================
VIEW_WAITKEY:
CLR Y
CLR Z
B0MOV Y, #WAITKEY$M
B0MOV Z, #WAITKEY$L             ; 載入存放在 WAITKEY 位址的值
Begin1:
MOV A, #0
MOVC
CMPRS A, #0
JMP @F
RET
@@:
CALL SDATA
XCH A, R
CALL SDATA
INCMS Z
JMP Begin1
```

```
;================================================
WAITKEY:
DB "Wait for Command : "
RET


    ;step7
MOV A, #11100000b
XOR A, P5
AND A, #11100000b
XOR P5, A                    ; 選擇 Y7 掃描線
Loop:
BTS0 REQUEST.0
JMP CLR_DATA
BTS0 REQUEST.1
JMP SPEED
BTS0 REQUEST.2
JMP SET_FORWARD
BTS0 REQUEST.3
JMP SET_REVERSE
BTS0 REQUEST.4
JMP RUN
JMP Loop
```

C.3　副程式 CLR 程式碼

```
    ;step1
CLR_DATA:
BCLR REQUEST.0
BCLR INTEN.0
```

```
MOV A, #00000001b                ; 清除全螢幕
CALL COMMAND
CALL DELAY2MS
MOV A, #10000000b                ; 設定 DD RAM 位址爲 0
CALL COMMAND
CALL VIEW_CLR_DATA
;================================================
VIEW_CLR_DATA:
CLR Y
CLR Z
B0MOV Y, #CLRDATA$M
B0MOV Z, #CLRDATA$L
MARK1:
MOV A, #0
MOVC                             ;SHOW "SURE CLEAR DATA ?"
CMPRS A, #0
JMP @F
RET
@@:
CALL SDATA
XCH A, R
CALL SDATA
INCMS Z
JMP MARK1
;================================================
CLRDATA:
DB "SURE CLEAR DATA?"
RET


    ;step2
```

```
MOV A, #11100000b
XOR A, P5
AND A, #11100000b
XOR P5, A                          ; 選擇 Y7 掃描線
MOV A, #0xAA
MOV BUFFER, A
NOP
BSET INTEN.0
MOV A, BUFFER
CMPRS A, #0xAA                     ; WAIT FOR INPUT "F" OR OTHERS
JMP @F
JMP $-3


    ;step3
@@:
CMPRS A, #0x0F
JMP Begin
CLR BUFFER
CLR BUFFER+1
CLR BUFFER+2
CLR BUFFER+3
CLR SBUFFER
CLR SBUFFER+1
CLR SBUFFER+2
CLR SBUFFER+3
CLR SPEEDBUFFER
CLR SPEEDBUFFER+1
MOV A, #4
MOV TC0M, A
CLR TC0R
```

```
CLR DIRECTION
CALL CLEAR_OK
JMP Begin
;================================================
CLEAR_OK:
MOV A, #00000001b                ; 清除全螢幕
CALL COMMAND
CALL DELAY2MS
MOV A, #10000000b                ; 設定 DD RAM 位址爲 0
CALL COMMAND
B0MOV Y, #CLROK$M
B0MOV Z, #CLROK$L
MARK2:
MOV A, #0
MOVC                             ; SHOW "CLEAR OK"
CMPRS A, #0
JMP @F
CALL DELAY2S
RET
@@:
CALL SDATA
XCH A, R
CALL SDATA
MOV A, #1
ADD Z, A
JMP MARK2
;================================================
CLROK:
DB "CLEAR Ok !!!"
RET
```

C.4　副程式 SPEED 程式碼

```
    ;step1
SPEED:
BCLR REQUEST.1
BCLR INTEN.0
MOV A, #00000001b              ; 清除全螢幕
CALL COMMAND
CALL DELAY2MS
MOV A, #10000000b              ; 設定 DD RAM 位址爲 0
CALL COMMAND
CALL VIEW_SPEED
;=================================================
VIEW_SPEED:
CLR Y
CLR Z
B0MOV Y, #SET_SPEED$M
B0MOV Z, #SET_SPEED$L
MARK7:
MOV A, #0
MOVC                           ;SHOW "PLEASE INPUT SPEED :"
CMPRS A, #0
JMP @F
RET
@@:
CALL SDATA
XCH A, R
CALL SDATA
INCMS Z
JMP MARK7
;=================================================
```

```
SET_SPEED:
DB "Please Input Speed :"
RET

    ;step2
MOV A, #11100000b
XOR A, P5
AND A, #11100000b
XOR P5, A                          ; 選擇 Y7 掃描線
BSET KEYTYPE.0                     ; 選擇數字 KEY
MOV A, #0xAA
MOV BUFFER+3, A
NOP
BSET INTEN.0
MOV A, BUFFER+3
CMPRS A, #0xAA                     ; WAIT FOR INPUT NUMBERS
JMP @F
JMP $-3

    ;step3
BIN2BCDtoHEX:
BCLR PFLAG.2
CLR R
CLR SPEEDBUFFER
CLR SPEEDBUFFER+1
DO_HEX_1000:
MOV A, BUFFER+3
CMPRS A, #0
JMP $+2
JMP DO_HEX_100
```

```
MOV R, A
HEX_1000:
DECMS  R
JMP @F
JMP $+7
@@:
MOV A, #11101000b
ADD SPEEDBUFFER, A
MOV A, #00000011b
ADC SPEEDBUFFER+1, A
BCLR PFLAG.2
JMP HEX_1000
MOV A, #11101000b
ADD SPEEDBUFFER, A
MOV A, #00000011b
ADC SPEEDBUFFER+1, A
BCLR PFLAG.2
DO_HEX_100:
MOV A, BUFFER+2
CMPRS A, #0
JMP $+2
JMP HEX_10
MOV R, A
HEX_100:
DECMS  R
JMP @F
JMP $+7
@@:
MOV A, #01100100b
ADD SPEEDBUFFER, A
MOV A, #0
```

```
ADC SPEEDBUFFER+1, A
BCLR PFLAG.2
JMP HEX_100
MOV A, #01100100b
ADD SPEEDBUFFER, A
MOV A, #0
ADC SPEEDBUFFER+1, A
BCLR PFLAG.2
HEX_10:
MOV A, #10
MUL A, BUFFER+1
ADD SPEEDBUFFER, A
BTS0 PFLAG.2                    ; C=1 , IF CARRY
INCMS SPEEDBUFFER+1
BCLR PFLAG.2
HEX_1:
MOV A, BUFFER
ADD SPEEDBUFFER, A
BTS0 PFLAG.2                    ; C=1 , IF CARRY
INCMS SPEEDBUFFER+1
BCLR PFLAG.2
RET

@@:
CALL UPDATESPEED
;step4
CALL SPEED_SETOK                ;LCD 顯示 "Speed Set ok"
BCLR KEYTYPE.0
JMP Begin
```

C.4.1　副程式 UPDATESPEED 程式碼

```
    UPDATESPEED:
;step1
MOV A, BUFFER
MOV SBUFFER, A
MOV A, BUFFER+1
MOV SBUFFER+1, A
MOV A, BUFFER+2
MOV SBUFFER+2, A
MOV A, BUFFER+3
MOV SBUFFER+3, A
CALL BIN2BCDtoHEX
;step2
MOV A, #11111111b              ; 是否大於 2047, A=2047-SPEED
SUB A, SPEEDBUFFER
MOV A, #00000111b
SBC A, SPEEDBUFFER+1
BTS1 PFLAG.2                   ; C=0, IF 借位
CALL MORETHAN2047
;step3
BCLR PFLAG.2
CALL JUDGE_INTERVAL
BCLR PFLAG.2
MOV TC0M, A
MOV A, SPEEDBUFFER
MOV TC0R, A
RET
```

C.5　副程式 FOREARD 程式碼

```
    ;step1
SET_FORWARD:
BCLR REQUEST.2
BCLR INTEN.0
MOV A, #00000001b              ; 清除全螢幕
CALL COMMAND
CALL DELAY2MS
MOV A, #10000000b              ; 設定 DD RAM 位址爲 0
CALL COMMAND
CALL VIEW_SETFORWARD

    ;step2
MOV A, #11100000b
XOR A, P5
AND A, #11100000b
XOR P5, A                      ; 選擇 Y7 掃描線
MOV A, #0xAA
MOV BUFFER, A
NOP
BSET INTEN.0
MOV A, BUFFER
CMPRS A, #0xAA                 ; WAIT FOR INPUT KEY
JMP @F
JMP $-3

    ;step3
@@:
CMPRS A, #0x0F
```

```
JMP Begin
BCLR DIRECTION.0                ;DIRECTION.0=0 正轉 ;1 反轉
CALL FORWARD_OK                 ;LCD 顯示 "FORWARD OK"
JMP Begin
```

C.6　副程式 REVERSE 程式碼

```
    ;step1
SET_REVERSE:
BCLR REQUEST.3
BCLR INTEN.0
MOV A, #00000001b               ; 清除全螢幕
CALL COMMAND
CALL DELAY2MS
MOV A, #10000000b               ; 設定 DD RAM 位址爲 0
CALL COMMAND
CALL VIEW_SET_REVERSE           ;SHOW "REVERSE OK"

    ;step2
MOV A, #11100000b
XOR A, P5
AND A, #11100000b
XOR P5, A                       ; 選擇 Y7 掃描線
MOV A, #0xAA
MOV BUFFER, A
NOP
BSET INTEN.0
MOV A, BUFFER
CMPRS A, #0xAA                  ; WAIT FOR INPUT KEY
```

```
JMP @F JMP $-3


    ;step3
@@:
CMPRS A, #0x0F
JMP Begin
BSET DIRECTION.0
CALL REVERSE_OK
JMP Begin
```

C.7　副程式 RUN 程式碼

```
    ;step1
RUN:
BCLR REQUEST.4
BCLR INTEN.0
MOV A, #00000001b              ; 清除全螢幕
CALL COMMAND
CALL DELAY2MS
MOV A, #10000000b              ; 設定 DD RAM 位址爲 0
CALL COMMAND
BTS0 DIRECTION.0               ; 方向判斷
JMP BACKWARDSTEP


    ;step2
FORWARDSTEP:
CALL VIEW_FORWARDSTEP
MOV A, #11000000b              ; 清除全螢幕
CALL COMMAND
```

```
CALL VIEW_SPEED_DATA            ;LCD 第二列顯示速度
JMP RUNF
BACKWARDSTEP:
CALL VIEW_BACKWARDSTEP
MOV A, #11000000b               ; 清除全螢幕
CALL COMMAND
CALL VIEW_SPEED_DATA            ;LCD 第二列顯示速度
JMP RUNR


    ;step3
RUN1:
MOV A, #11100000b
XOR A, P5
AND A, #11100000b
XOR P5, A                       ; 選擇 Y7 掃描線
MOV A, SPEEDBUFFER
MOV TC0R, A
BSET TC0M.7                     ; enable TC0 , start running TC0
BSET INTEN.0
MOV A, BUFFER
CMPRS A, #0x0F                  ; 若按下 F 鍵 , 則馬達停止旋轉
JMP CALL VIEW_SPEED_DATA        ;LCD 第二列顯示速度
JMP Begin
```

C.8　中斷副程式程式碼

```
    ;step1
;******* Interrupt Service Subprograms ***************
INT0_IRS:
```

```
MOV A, #11000000b                ;LCD 顯示在第二列
CALL COMMAND
BCLR SET_OK.0
MOV A, #0xFF
MOV BUFFER, A
MOV BUFFER+1, A
MOV BUFFER+2, A
MOV BUFFER+3, A
FUNCTION_KEY:
MOV A, #3
MOV DisplayBit, A                ; DisplayBit=3
B0MOV Z, #BUFFER+3

    ;step2
KEYLOOP:
MOV A, @YZ
CALL SCAN_DISP
CALL KEY_SCAN
@@:
BTS0 SET_OK.0
JMP KEYEND
MOV A, #256-1
ADD DisplayBit, A                ; DisplayBit=DisplayBit-1
BTS1 PFLAG.2                     ; if flag C=1 無借位 , skip
JMP FUNCTION_KEY                 ; if DisplayBit =0-1, go to FUNCTION_KEY
ADD Z, A
JMP KEYLOOP

    ;step3
MOV A, BUFFER
```

```
CMPRS A, #0x0A
JMP @F
BSET REQUEST.0
JMP KEYEND
@@:
CMPRS A, #0x0B
JMP @F
BSET REQUEST.1
JMP KEYEND
@@:
CMPRS A, #0x0C
JMP @F
BSET REQUEST.2
JMP KEYEND
@@:
CMPRS A, #0x0D
JMP @F
BSET REQUEST.3
JMP KEYEND
@@:
CMPRS A, #0x0E
JMP @F
BSET REQUEST.4
JMP KEYEND
@@:
CMPRS A, #0x0F                  ; IF KEY=ENTER THAN END KEYSCAN
JMP @F
BCLR TC0M.7
JMP KEYEND
KEYEND:
BCLR INTRQ.0
```

```
End_Interrupt                    ; 中斷結束巨集程式
;******* Interrupt Service Subprograms ***************
TC0_IRS:
CALL STEPMOVE
BCLR INTRQ.5
End_Interrupt                    ; 中斷結束巨集程式
```

C.8.1　副程式 SCAN_LCD_DISP 程式碼

```
    SCAN_LCD_DISP:               ;subprogram
MOV A, LASTKEY
SUB A, #0x0A
BTS0 PFLAG.2                     ; C=0, IF 借位
JMP CLRSECONDROW
MOV A, LASTKEY
SCAN_LCD_DISP1:
CMPRS A, #0
JMP @F
MOV A, #0x30
JMP MARK0
@@:
CMPRS A, #1
JMP @F
MOV A, #0x31
JMP MARK0
@@:
CMPRS A, #2
JMP @F
MOV A, #0x32
JMP MARK0
```

```
@@:
CMPRS A, #3
JMP @F
MOV A, #0x33
JMP MARK0
@@:
CMPRS A, #4
JMP @F
MOV A, #0x34
JMP MARK0
@@:
CMPRS A, #5
JMP @F
MOV A, #0x35
JMP MARK0
@@:
CMPRS A, #6
JMP @F
MOV A, #0x36
JMP MARK0
@@:
CMPRS A, #7
JMP @F
MOV A, #0x37
JMP MARK0
@@:
CMPRS A, #8
JMP @F
MOV A, #0x38
JMP MARK0
@@:
```

```
CMPRS A, #9
JMP @F
MOV A, #0x39
JMP MARK0
@@:
JMP $+2
MARK0:
CALL SDATA
RET
```

C.8.2　副程式 STEPMOVE 程式碼

```
    ;step1
STEPMOVE:
BTS0 DIRECTION.0              ; 方向判斷
JMP BACKWARD
;step2
FORWARD:
BTS1 P2.0                     ; 判斷 HALLu 準位
JMP @F
JMP CASE1
@@:
JMP CASE2
CASE1:
BTS1 P2.1                     ; 判斷 HALLv 準位
JMP @F
JMP STATE5TO6
@@:
JMP JUDGEF
CASE2:
```

```
BTS1 P2.1                    ; 判斷 HALLv 準位
JMP @F
JMP CASE3
@@:
JMP STATE2to3
CASE3:
BTS1 P1.4                    ; 判斷 HALLw 準位
JMP @F
JMP STATE3to4
@@:
JMP STATE4to5
JUDGEF:
BTS1 P1.4                    ; 判斷 HALLw 準位
JMP @F
JMP STATE1TO2
@@:
JMP STATE6TO1
;-------------------------------------------------
BACKWARD:
BTS1 P2.0                    ; 判斷 HALLu 準位
JMP @F
JMP CASE4
@@:
JMP CASE5
CASE4:
BTS1 P2.1                    ; 判斷 HALLv 準位
JMP @F
JMP STATE5TO4
@@:
JMP JUDGEB
CASE5:
```

```
BTS1 P2.1                    ; 判斷 HALLv 準位
JMP @F
JMP CASE6
@@:
JMP STATE2to1
CASE6:
BTS1 P1.4                    ; 判斷 HALLw 準位
JMP @F
JMP STATE3to2
@@:
JMP STATE4to3
JUDGEB:
BTS1 P1.4                    ; 判斷 HALLw 準位
JMP @F
JMP STATE1TO6
@@:
JMP STATE6TO5


      ;step3
; 正轉 STATE1to2:
MOV A, #11000100b
XOR A, P2
AND A, #11111100b
XOR P2, A
BCLR P1.5
RET
;=================================
STATE2to3:
MOV A, #11010000b
XOR A, P2
```

```
AND A, #11111100b
XOR P2, A
BSET P1.5
RET
;==================================
STATE3to4:
MOV A, #01110000b
XOR A, P2
AND A, #11111100b
XOR P2, A
BCLR P1.5
RET
;==================================
STATE4to5:
MOV A, #00110100b
XOR A, P2
AND A, #11111100b
XOR P2, A
BSET P1.5
RET
;==================================
STATE5to6:
MOV A, #00011100b
XOR A, P2
AND A, #11111100b
XOR P2, A
BCLR P1.5
RET
;==================================
STATE6to1:
MOV A, #01001100b
```

```
XOR A, P2
AND A, #11111100b
XOR P2, A
BSET P1.5
RET
;===================================
; 反轉 STATE1to6:
MOV A, #11010000b
XOR A, P2
AND A, #11111100b
XOR P2, A
BSET P1.5
RET
;===================================
STATE6to5:
MOV A, #11000100b
XOR A, P2
AND A, #11111100b
XOR P2, A
BCLR P1.5
RET
;===================================
STATE5to4:
MOV A, #01001100b
XOR A, P2
AND A, #11111100b
XOR P2, A
BSET P1.5
RET
;===================================
STATE4to3:
```

```
MOV A, #00011100b
XOR A, P2
AND A, #11111100b
XOR P2, A
BCLR P1.5
RET
;==================================
STATE3to2:
MOV A, #00110100b
XOR A, P2
AND A, #11111100b
XOR P2, A
BSET P1.5
RET
;==================================
STATE2to1:
MOV A, #01110000b
XOR A, P2
AND A, #11111100b
XOR P2, A
BCLR P1.5
RET
```

參考文獻

[1] 林錫寬，掌握微控制器原理與技術，使用 SN8P1700 系列晶片 . 全華科技， 2003.

[2] 謝孟勳，三相永磁馬達感應電動勢常數的新型鑑別方法 . 交通大學電機與控制研究所碩士論文， 2004.

[3] 石富元，以 USB 實現步進馬達控制 . 交通大學電機與控制研究所碩士論文， 2003.

[4] 蔡朝洋，單晶片微電腦 8048/8748 應用實務 . 全華科技， 1993.

[5] 鍾富昭， 8048/8049 高級專題製作 . 全華科技， 1991.

[6] 鄧錦成， 8051 單晶片專題製作 . 益眾資訊， 1991.

[7] 張義和，主流電腦輔助電路設計 :Protel 99 SE. 全華科技， 2002.

[8] 羅永昌，電動機控制 . 高立圖書， 1999.

[9] SN8P1700 Series USER'S MANUAL

[10] http://www.sonix.com.tw