

I-8092 運動控制模組使用手冊

(Version 1.7)

應用程式函式庫

WinCon-8000、I-8000、LinCon-8000 系列控制器
(適用於 i8092, i8092F DLL V0.6.0.3)



ICP DAS CO., LTD.

泓格科技股份有限公司

Warranty

All products manufactured by ICPDAS Inc. are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICPDAS Inc. assumes no liability for damages consequent to the use of this product. ICPDAS Inc. reserves the right to change this manual at any time without notice. The information furnished by ICPDAS Inc. is believed to be accurate and reliable. However, no responsibility is assumed by ICPDAS Inc. for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Copyright 1997-2005 by ICPDAS Inc., LTD. All rights reserved worldwide.

Trademark

The names used for identification only maybe registered trademarks of their respective companies.

License

The user can use, modify and backup this software on a single machine. The user may not reproduce, transfer or distribute this software, or any copy, in whole or in part.

目錄

1 前言	7
1.1 手冊的使用	7
1.2 基本函式和巨集函式	7
1.3 函式基本結構說明	7
2.3 巨集函式類別	8
2.4 基本函式類別	8
2 基本設定功能	9
2.1 各軸定義的代碼	9
2.2 註冊軸卡及版本讀取	9
2.3 軸卡重置	11
2.4 軸輸出PULSE模式設定	11
2.5 設定軸速度輸出最大範圍	12
2.6 設定軸前後極限的輸入觸發邏輯	12
2.7 設定碰觸前後極限的處理模式	13
2.8 設定軸近原點輸入觸發邏輯	13
2.9 設定軸原點輸入觸發邏輯	14
2.10 設定軸前後軟體極限,參考來源及取消	14
2.11 設定編碼器輸入參數	15
2.12 伺服開關(Servo_ON/OFF)	16
2.13 設定伺服馬達異常ALARM輸入參數	16
2.14 設定伺服馬達定位完成輸入參數	17
2.15 設定數位輸入雜訊濾波功能	17
2.16 指定軸為圓形運動軸(環狀計數器)	19
2.17 三角形速度曲線的預防	20
2.18 外部輸入驅動	20
2.18.1 手輪脈波驅動	20
2.18.2 固定脈波驅動	21
2.18.3 連續脈波驅動	22
2.18.4 外部輸入關閉	22
3 狀態讀取及設定功能	23
3.1 設定及讀取指令邏輯位置	23

3.2 設定及讀取ENCODER位置	24
3.3 讀取目前速度	25
3.4 讀取目前加速度	25
3.5 讀取目前DI狀態	26
3.6 讀取目前ERROR狀態	28
3.2.7 一般DO 輸出	29
4 FRNET功能(I8092F專用函式)	30
4.1 FRnet DI讀取	30
4.2 FRnet DO寫入	30
5 軸自動歸零	31
5.1 軟體歸零相關函式	31
5.2 軟體自動歸零功能	32
5.3 軟體單步驟歸零功能	34
5.3.1 尋找近原點開關	34
5.3.2 尋找近原點開關	35
5.3.3 尋找伺服馬達Z相信號	36
6 軸控功能	37
6.1 兩軸同時獨立運動	37
6.1.1 設定單軸加減速模式	37
6.1.2 設定軸初始速度	38
6.1.3 設定軸定速度	39
6.1.4 設定軸加速度	39
6.1.5 設定軸減速度	40
6.1.6 設定軸加速度變化率	40
6.1.7 設定軸減速(保留脈波數)	41
6.1.8 固定脈波數輸出	41
6.1.9 連續脈波輸出	42
6.2 補間運動	43
6.2.1 XY軸補間	43
6.2.2 設定補間加減速模式	43
6.2.3 設定軸向量初始速度	45
6.2.4 設定軸向量定速度	46
6.2.5 設定軸向量加速度	46
6.2.6 設定軸向量減速度	47
6.2.7 設定軸向量加速度變化率	47
6.2.8 設定軸向量減速(保留脈波數)	48
6.2.9 二軸直線補間	49

6.2.10 二軸圓弧補間.....	50
6.2.11 二軸圓形補間.....	52
6.3 連續補間運動	53
6.3.1 二軸矩形連續補間	53
6.3.2 二軸直線連續補間	54
6.3.3 二軸多段連續補間	55
6.3.4 二軸比例運動.....	56
6.3.5 二軸混合連續補間	58
6.5 其他功能	60
6.5.1 設定軸暫停	60
6.5.2 設定軸啟動	60
6.5.3 等待完成軸運動	61
6.5.4 設定軸停止	62
6.5.5 清除停止狀態.....	63
6.5.6 補間動作結束(單軸運動或改變座標系)	63
附錄A (I-8092 BASE FUNCTION)	64
A.1 i-8092 運動控制命令集.....	64
A.2 脈波輸出命令	65
A.2.1 連續脈波輸出驅動.....	69
A.2.2 定速驅動	70
A.3 加減速曲線的設定	71
A.3.1 T-曲線加減速驅動 [對稱].....	71
A.3.2 T-曲線加減速驅動 [非對稱].....	72
A.3.3 三角形速度曲線的預防.....	74
A.3.4 S-曲線加減速驅動 [對稱]	75
A.4 二軸補間運動	79
A.4.1 二軸直線補間.....	79
A.4.2 圓弧補間	80
A.4.3 位元補間	83
A.4.4 連續補間	85
A.5 原點返回運動	86
A.6 中斷控制	87
A.6.1 各軸中斷	87
A.6.2 補間中斷	88
A.7 i-8092 功能函式庫.....	89
A.7.1 暫存器管理函式	90
A.7.2 函式初始設定	97
A.7.3 位置控制函式.....	106
A.8.4 基本運動命令函式.....	113
A.8.5 補間函式	125
A.8.6 原點返回運動(軸歸零)函式.....	139

A.8.9 中斷函式	161
A.8.10 FRnet相關函式.....	163
A. 9 i-8092 命令列表	164
A.9.1 資料寫入命令	164
A.9.2 資料讀取命令	165
A.9.3 驅動命令	165
A.9.4 補間命令	166
A.9.5 其他命令	166
附錄B (MCX312 REGISTER)	167
B.1 Command Register: WR0	167
B.2 Mode Register1: WR1	168
B.3 Mode Register2: WR2	170
B.4 Mode Register3: WR3	172
B.5 Output Register: WR4	175
B.7 Data Register: WR6/WR7.....	176
B.8 Main Status Register: RR0	176
B.9 Status Register 1: RR1.....	177
B.10 Status Register 2: RR2.....	179
B.11 Status Register 3: RR3.....	180
B.12 Input Register: RR4 / RR5	180
B.13 Data-Read Register: RR6 / RR7.....	181

1 前言

1.1 手冊的使用

- 使用 i8092 運動控制模組，去設計你的自動化設備時，本手冊提供了完整且詳細的說明，幫助你很快的找到你要的運動控制函式，並配合簡單的範例，迅速開發你的應用程式。
- 手冊分為六大章和附錄，本章是手冊的前言，2、3、4、5、6 五章為巨集函式(MF)的說明，最後一章附錄 A、B，內容為基本函式(BF)及 MCX312 暫存器說明。
- 本手冊需搭配泓格公司所提供的應用程式函式庫(DLL)，它支援各類軟體平台(eVC++、VB.net、C#.net)及作業系統(MiniOS7 / WinCE / Linux)。

1.2 基本函式和巨集函式

- 基本函式適合熟 MCX312 運動控制晶片者使用，它提供了許多直接控制晶片的函式，但是使用難度較高。
- 巨集函式替使用者架構了簡單易用的程式撰寫環境，降低了運動控制高難度的門檻。直覺式的參數設計、客製化巨集的運動函式、連續補間及減速點的自動運算.....，已經符合絕大多數使用者的需求，希望這能提供使用者一個更好的選擇。
- 兩大類函式不要混合使用，因為大部份函式內部參數的定義並無共通性。

1.3 函式基本結構說明

- 函式名稱(參數一, 參數二,)

功能: 函式基本功能說明。

參數: 參數的定義及使用方法。

回應: 函式的回傳值。

範例: 簡單的示範參考程式。(手冊中的範例皆以 C++ code 撰寫)

備註: 備忘註解。

1.3 巨集函式類別

泓格 ICPDAS 針對 eVC++ 提供一 **CI8092MF** 的類別，可以引導及幫助程式撰寫巨集函式，詳細請參考，快速上手第四章。

1.4 基本函式類別

泓格 ICPDAS 針對 eVC++ 提供一 **CI8092** 的類別，可以引導及幫助程式撰寫基本函式，詳細請參考，快速上手第四章。

2 基本設定功能

2.1 各軸定義的代碼

所有功能中有關軸參數，是以 X=1、Y=2 作為代碼，假設我們要指定 XY=3，就是 1+2=3，因此同一功能，可以一次做單軸設定，也可以一次設兩軸相同設定，所有功能中有關軸參數代碼(WORD axis)與意義如下：

對照表(2-1)

軸	X	Y	XY
代碼	0x1	0x2	0x3
變數	AXIS_X	AXIS_Y	AXIS_XY

2.2 註冊軸卡及版本讀取

● BYTE i8092MF_REGISTRATION(BYTE cardNo, BYTE slot)

功能： 註冊軸卡，指定插槽及卡號，使用 i8092 所有功能前，都必須做此註冊。

參數： **cardNo:** 指定卡號
slot: 插槽號碼 → I-8000 : 0~7
→ WinCon-8000 : 1~7

回應： YES: 正常
NO: 異常

範例： **//===== WinCon-8000 =====**
//設定各槽(slot1~slot7)，對應的卡號為 1~7。
BYTE cardNo;
BYTE slot;
int Found = 0;
for (slot = 1; slot < 8; slot++)
{
 cardNo = slot;
 if (i8092MF_REGISTRATION(cardNo, slot) == YES)
 {
 //找到軸卡，註冊。
 i8092MF_RESET_CARD(cardNo);
 Found++;
 }
}

```

if (Found == 0)
{
    //找不到軸卡，異常處理。
    return;
}

//===== I-8000 =====
//設定各槽(slot0~slot7)，對應的卡號為 1~8。
BYTE cardNo;
BYTE slot;
int Found = 0;
for (slot = 0; slot < 8; slot++)
{
    cardNo = slot + 1;
    if (i8092MF_REGISTRATION(cardNo, slot) == YES)
    {
        //找到軸卡，註冊。
        i8092MF_RESET_CARD(cardNo);
        Found++;
    }
}
if (Found == 0)
{
    //找不到軸卡，異常處理。
    return;
}

```

● **WORD i8092MF_GET_VERSION(void)**

功能： 讀取 i8092 運動函式庫之版本。

參數： 無

回應： WORD 版本號碼 0x0000 ~ 0x9999

範例： WORD VER_No;
VER_No = i8092MF_GET_VERSION();
//讀取 i8092.dll 版本號碼。

備註： 以下為讀到的版本release資訊(2005年11月)
i8094MF_GET_VERSION: 0x0511
i8094.dll : 0,5,1,1
0,5 函式庫版本年流水序
1,1 函式庫版本月流水序

2.3 軸卡重置

- **void i8092MF_RESET_CARD(BYTE cardNo)**

功能： I-8092 重設成電源開啟狀態。

參數： **cardNo:** 指定卡號

回應： 無

範例： **i8092MF_RESET_CARD (1);**

//重置第 1 卡。

2.4 軸輸出 PULSE 模式設定

- **void i8092MF_SET_PULSE_MODE(BYTE cardNo, WORD axis, BYTE nMode)**

功能： 設定軸之輸出模式，包含 CW/CCW 或 PULSE/DIR，及正方向定義。

參數： **cardNo:** 指定卡號
axis: 指定軸號碼(參考表 2-1)
nMode: 指定模式(參考表 2-2)

回應： 無

範例： **i8092MF_SET_PULSE_MODE(1, AXIS_XY, 2);**

//指定第 1 卡 XY 軸，脈波輸出模式皆為 2。

脈波輸出模式表(2-2)

形式	模式	方向	脈波信號輸出	
			nPP	nPM
CW / CCW	0		CW(正緣觸發)	CCW(正緣觸發)
	1		CW(負緣觸發)	CCW(負緣觸發)
PULSE / DIR	2	+	PULSE(正緣觸發)	DIR(LOW)
	3		PULSE(負緣觸發)	DIR(LOW)
	4	-	PULSE(正緣觸發)	DIR(HIGH)
	5		PULSE(負緣觸發)	DIR(HIGH)

2.5 設定軸速度輸出最大範圍

- **void i8092MF_SET_MAX_V**(BYTE *cardNo*, WORD *axis*, DWORD *data*)

功能： 設定軸之輸出最高速度 PPS 限制，影響：最高速度越小，速度解析度越高，反之越

大 (速度總共有 8000 段)。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼(參考表 2-1)
data: 指定最高速度，單軸(8,000~4,000,000 PPS)
補間最高速度，Y 軸(8,000~2,828,854 PPS)

回應： 無

範例： `i8092MF_SET_MAX_V(1, AXIS_XY, 200000L);`
//設定第 1 卡 XY 軸，最高速為 200K PPS，每段速度為 $200000 / 8000 = 25$ PPS。

2.6 設定軸前後極限的輸入觸發邏輯

- **void i8092MF_SET_HLMT**(BYTE *cardNo*, WORD *axis*, BYTE *nFLEdge*, BYTE *nRLEdge*)

功能： 設定軸之"前後極限"開關觸發邏輯。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼(參考表 2-1)
nFLEdge: 前極限觸發邏輯: 0=低準位觸發, 1=高準位觸發
nRLEdge: 後極限觸發邏輯: 0=低準位觸發, 1=高準位觸發

回應： 無

範例： `i8092MF_SET_HLMT(1, AXIS_XY, 0, 0);`
//設定第1卡 XY 軸，其"前後極限"觸發邏輯，全部為低準位觸發。

2.7 設定碰觸前後極限的處理模式

● **void i8092MF_LIMITSTOP_MODE** (BYTE *cardNo*, WORD *axis*, BYTE *nMode*)

功能： 設定碰觸"前後極限"處理模式。

參數： **cardNo:** 指定卡號
axis: 指定軸號碼(參考表 2-1)
nMode: 設定處理方法: 0=立即停止,1=減速後停止

回應： 無

範例： **i8092MF_LIMITSTOP_MODE(1, AXIS_X, 0);**
//設定第 1 卡 X 軸，碰觸前後極限後立即停止。

2.8 設定軸近原點輸入觸發邏輯

● **void i8092MF_SET_NHOME**(BYTE *cardNo*, WORD *axis*, BYTE *nNHEdge*)

功能： 設定軸之"近原點"開關觸發邏輯。

參數： **cardNo:** 指定卡號
axis: 指定軸號碼(參考表 2-1)
nNHEdge: "近原點"開關觸發邏輯: 0=低準位觸發, 1=高準位觸發

回應： 無

範例： **i8092MF_SET_NHOME(1, AXIS_XY, 0);**
//設定第 1 卡 XY 軸，其"近原點"開關，觸發邏輯全部為低準位觸發。

2.9 設定軸原點輸入觸發邏輯

- **void i8092MF_SET_HOME_EDGE**(BYTE *cardNo*, WORD *axis*, BYTE *nHEdge*)

功能： 設定軸之"原點"開關觸發邏輯。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼(參考表 2-1)
nHEdge: "原點"開關觸發邏輯: 0=低準位觸發, 1=高準位觸發

回應： 無

範例： `i8092MF_SET_HOME_EDGE(1, AXIS_XY, 1);`
//設定第 1 卡 XY 軸，其"原點"開關，觸發邏輯全部為高準位觸發。

2.10 設定軸前後軟體極限,參考來源及取消

- **void i8092MF_SET_SLMT**(BYTE *cardNo*, WORD *axis*, long *dwFL*, long *dwRL*, BYTE *nType*)

功能： 設定軸之"前後軟體極限"功能。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼(參考表 2-1)
dwFL: 前軟體極限值(-2,147,483,648 ~ +2,147,483,647)
dwRL: 後軟體極限值(-2,147,483,648 ~ +2,147,483,647)
nType: 比較對象: 0=指令輸出位置,1=實際編碼器回饋位置

回應： 無

範例： `i8092MF_SET_SLMT(1, AXIS_XY, 20000, -3000, 0);`
//設定第 1 卡 XY 軸，以指令輸出位置做比較，前軟體極限=20000，後軟體極限=-3000。

- **void i8092MF_CLEAR_SLMT**(BYTE *cardNo*, WORD *axis*)

功能： 取消軸之"前後軟體極限"功能。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼(參考表 2-1)

回應： 無

範例： `i8092MF_CLEAR_SLMT(1, AXIS_XY);`
//取消第 1 卡 XY 軸，前後軟體極限功能。

2.11 設定編碼器輸入參數

● **void i8092MF_SET_ENCODER**(BYTE *cardNo*, WORD *axis*, BYTE *nMode*, BYTE *nDevision*, BYTE *nZEdge*)

功能： 設定軸之編碼器輸入參數。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼(參考表 2-1)
nMode: 編碼器輸入模式: 0=AB 相輸入, 1=上下計數輸入
nDevision: 模式為 AB 相輸入時, 指定除頻: 0=1:1, 1=1:2, 2=1:4
nZEdge: 設定伺服 Z 輸入信號觸發邏輯: 0=低準位觸發, 1=高準位觸發

回應： 無

範例： `i8092MF_SET_ENCODER(1, AXIS_XY, 0, 0, 0);`
//設定第 1 卡 X Y 軸，編碼器輸入為 AB 相，不除頻，Z 輸入信號低準位觸發。

● **void i8092MF_SET_EN_DIR**(BYTE *cardNo*, WORD *axis*, BYTE *nDir*)

功能： 設定軸之編碼器輸入正反向。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼(參考表 2-1)
nDire: 編碼器輸入方向: 0=正向輸入, 1=反向輸入

回應： 無

範例： `i8092MF_SET_EN_DIR (1, AXIS_XY, 1);`
//設定第 1 卡 X Y 軸，編碼器輸入為反向輸入。

2.12 伺服開關(Servo_ON/OFF)

- **void i8092MF_SERVO_ON**(BYTE *cardNo*, WORD *axis*)

功能： 設定軸驅動器伺服啟動。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼(參考表 2-1)

回應： 無

範例： **i8092MF_SERVO_ON(1, AXIS_XY);**
//設定第 1 卡 X Y 軸，啟動驅動器伺服。

- **void i8092MF_SERVO_OFF**(BYTE *cardNo*, WORD *axis*)

功能： 設定軸驅動器伺服關閉。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼(參考表 2-1)

回應： 無

範例： **i8092MF_SERVO_OFF(1, AXIS_XY);**
//設定第 1 卡 X Y 軸，關閉驅動器伺服。

2.13 設定伺服馬達異常 ALARM 輸入參數

- **void i8092MF_SET_ALARM**(BYTE *cardNo*, WORD *axis*, BYTE *nMode*, BYTE *nAEdge*)

功能： 設定軸之驅動器異常(ALARM)輸入參數。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼(參考表 2-1)
nMode: 模式: 0=關閉,1=開啟
nAEdge: 設定異常(ALARM)輸入信號觸發邏輯: 0=低準位觸發, 1=高準位觸發

回應： 無

範例： **i8092MF_SET_ALARM(1, AXIS_XY, 1, 0);**
//設定第 1 卡 X Y 軸，異常(ALARM)輸入為開啟，輸入信號觸發邏輯為低準位觸發。

2.14 設定伺服馬達定位完成輸入參數

● **void i8092MF_SET_INPOS**(BYTE *cardNo*, WORD *axis*, BYTE *nMode*, BYTE *nlEdge*)

功能： 設定軸之驅動器定位完成輸入參數。

參數： **cardNo:** 指定卡號
axis: 指定軸號碼(參考表 2-1)
nMode: 模式: 0=關閉,1=開啟
nlEdge: 設定定位完成輸入信號觸發邏輯: 0=低準位觸發, 1=高準位觸發

回應： 無

範例： **i8092MF_SET_INPOS(1, AXIS_X, 1, 0);**
//設定第 1 卡 X 軸，定位完成輸入為開啟，輸入信號觸發邏輯為低準位觸發。

備註： 請配合硬體接線使用,參考(Fig. 2.12 一般 DI 輸入接線範例)

2.15 設定數位輸入雜訊濾波功能

● **void i8092MF_SET_FILTER**(BYTE *cardNo*, WORD *axis*, WORD *FEn*, WORD *FLn*)

功能： 設定軸之輸入數位濾波項目及濾波時間參數。

參數： **cardNo:** 指定卡號
axis: 指定軸號碼(參考表 2-1)
FEn: 濾波項目:欲開啟項目代號加總值(0~31)如下表:

代號	開啟項目
1	緊急停止,前後極限, 近原點, 原點
2	編碼器 Z-相輸入
4	定位完成,伺服 ALARM
8	nEXPP, nEXPM, EXPLSN
16	輸入信號(IN3)

FLn: 設定濾波時間參數(0~7) 如下表:

代號	可移除最大雜訊寬(width)	輸入延遲時間
0	1.75 μ SEC	2 μ SEC
1	224 μ SEC	256 μ SEC
2	448 μ SEC	512 μ SEC
3	896 μ SEC	1.024mSEC
4	1.792mSEC	2.048mSEC

5	3.584mSEC	4.096mSEC
6	7.168mSEC	8.192mSEC
7	14.336mSEC	16.384mSEC

回應: 無

範例: **i8092MF_SET_FILTER(1, AXIS_XY, 21, 3);**
 //設定第 1 卡 X Y 軸, (21=1+4+16) 1→緊急停止、前後極限、近原點、原點, 4→
 定位完成、伺服 **ALARM**, 16→輸入信號(IN3)輸入濾波為開啟, 濾波時間常數
 =1.024mSEC。

2.16 指定軸為圓形運動軸(環狀計數器)

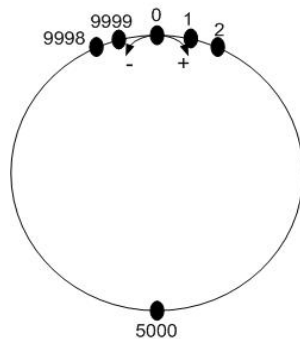
● **void i8092MF_VRING_ENABLE**(BYTE *cardNo*, WORD *axis*, DWORD *nVRing*)

功能： 指定軸啟動為環狀計數器 (如附圖)。

參數： **cardNo:** 指定卡號
axis: 指定軸號碼(參考表 2-1)
nVRing: 環狀計數器值(-2,147,483,648 ~ +2,147,483,647)

回應： 無

範例： **i8092MF_VRING_ENABLE(1, AXIS_X, 9999);**
//設定第 1 卡 X 軸，指定為環狀計數器，一圈為 10000 Pulse。



例如：

我們設計轉一圈為 10000 Pulse, 環狀計數器值設為 9999 正轉到 9999 後下一 Pulse 歸為 0,1..重新計算起

環狀計數器=9999

備註： 1.此功能,會同時使指令位置計數器及實際位置計數器同時有效,不能分別設定
2.此功能啟動後,軟體極限功能將不能使用

● **void i8092MF_VRING_DISABLE**(BYTE *cardNo*, WORD *axis*)

功能： 指定軸關閉環狀計數器功能。

參數： **cardNo:** 指定卡號
axis: 指定軸號碼 (參考表 2-1)

回應： 無

範例： **i8092MF_VRING_DISABLE(1, AXIS_X);**
//設定第 1 卡 X 軸，關閉環狀計數器功能。

2.17 三角形速度曲線的預防

- **void i8092MF_AVTRI_ENABLE**(BYTE *cardNo*, WORD *axis*)

功能： 致能預防三角形速度曲線的產生。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼 (參考表 2-1)

回應： 無

範例： `i8092MF_AVTRI_ENABLE(1, AXIS_X);`
//設定第 1 卡 X 軸，致能預防三角形速度的產生。

- **void i8092MF_AVTRI_DISABLE**(BYTE *cardNo*, WORD *axis*)

功能： 除能預防三角形速度的產生。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼 (參考表 2-1)

回應： 無

範例： `i8092MF_AVTRI_DISABLE(1, AXIS_X);`
//設定第 1 卡 X 軸，除能預防三角形速度的產生。

2.18 外部輸入驅動

2.18.1 手輪脈波驅動

- **void i8092MF_EXD_MP**(BYTE *cardNo*, WORD *axis*, long *data*)

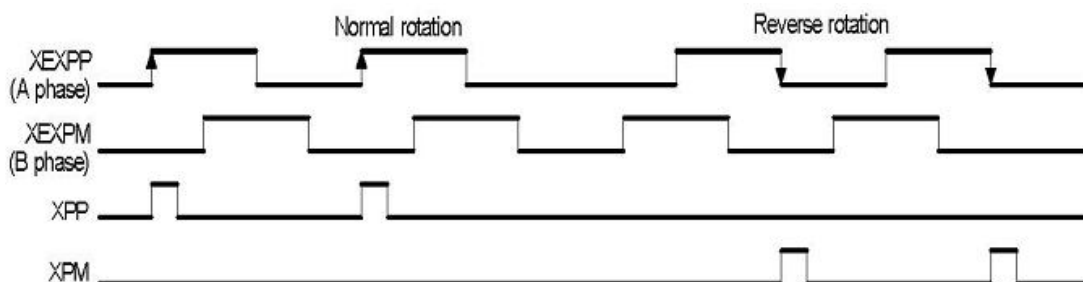
功能： 執行手輪輸入驅動，輸出固定步數。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼 X 或 Y (1 or 2)
data: 指定步數

回應： 無

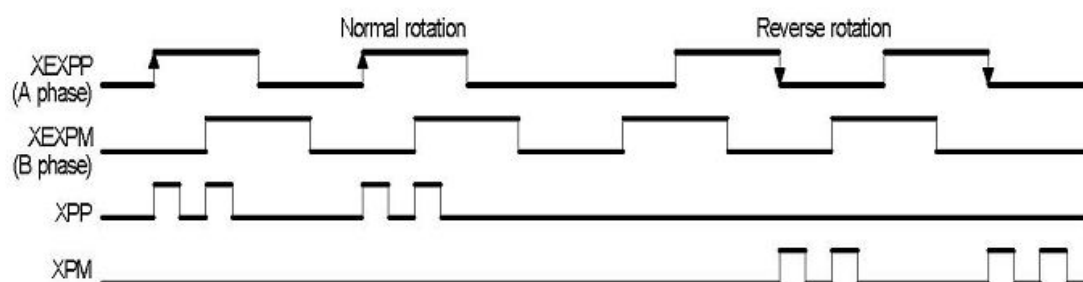
範例： `i8092MF_EXD_MP(1, AXIS_X, 1);`

//第1卡 X 軸，手輪觸發移動1步(Pulse)。



`i8092MF_EXD_MP(1, AXIS_X, 2);`

//第1卡 X 軸，手輪觸發移動2步(Pulse)。



2.18.2 固定脈波驅動

● **void** `i8092MF_EXD_FP`(**BYTE** *cardNo*, **WORD** *axis*, **long** *data*)

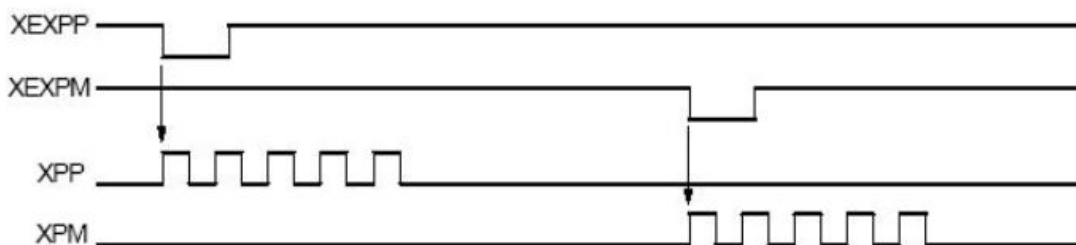
功能： 執行外部輸入驅動，輸出固定步數。

參數： **cardNo**: 指定卡號
axis: 指定軸號碼 X 或 Y (1 or 2)
data: 指定步數

回應： 無

範例： `i8092MF_EXD_FP(1, AXIS_X, 5);`

//第 1 卡 X 軸，外部觸發移動 5 步(Pulse)。



2.18.3 連續脈波驅動

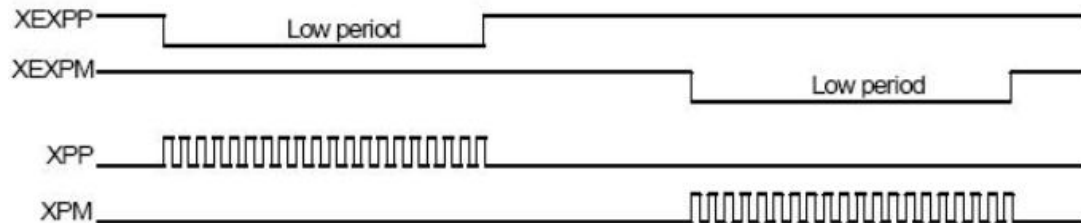
● **void** i8092MF_EXD_CP(BYTE *cardNo*, WORD *axis*, long *data*)

功能： 執行外部輸入驅動固定步數輸出。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼 X 或 Y (1 or 2)
data: 設定速度

回應： 無

範例： i8092MF_EXD_CP(1, AXIS_X, 20);
//第1卡 X 軸，觸發速度20 PPS移動。



2.18.4 外部輸入關閉

● **void** i8092MF_EXD_DISABLE(BYTE *cardNo*, WORD *axis*)

功能： 關閉外部輸入驅動功能。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼 X 或 Y(1 or 2)

回應： 無

範例： i8092MF_EXD_DISABLE(1, AXIS_X);
//關閉第1卡 X 軸，外部輸入驅動功能。

3 狀態讀取及設定功能

3.1 設定及讀取指令邏輯位置

- **void** i8092MF_SET_LP(BYTE *cardNo*, WORD *axis*, **long** *wdata*)

功能： 設定軸之目前指令位置。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼 (參考表 2-1)
wdata: 指令位置(-2,147,483,648 ~ +2,147,483,647)

回應： 無

範例： i8092MF_SET_LP(1, AXIS_XY, 0);
//設定第 1 卡 XY 軸，目前指令位置皆為 0。

- **long** i8092MF_GET_LP(BYTE *cardNo*, WORD *axis*)

功能： 讀取軸目前之指令位置。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼 X 或 Y(1 or 2)

回應： 目前指令位置 (-2,147,483,648 ~ +2,147,483,648)

範例： **long** X_LP;
X_LP = i8092MF_GET_LP(1, AXIS_X);
//讀取第 1 卡 X 軸，目前指令位置值。

3.2 設定及讀取 ENCODER 位置

- **void** i8092MF_SET_EP(BYTE *cardNo*, WORD *axis*, **long** *wdata*)

功能： 設定軸之目前 ENCODER 回授位置。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼(參考表 2-1)
wdata: 指令位置(-2,147,483,648 ~ +2,147,483,647)

回應： 無

範例： i8092MF_SET_EP(1, AXIS_XY, 0);
//設定第 1 卡 XY 軸，目前 ENCODER 回授位置皆為 0。

- **long** i8092MF_GET_EP(BYTE *cardNo*, WORD *axis*)

功能： 讀取軸目前之 ENCODER 回授位置。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼 X 或 Y(1 or 2)

回應： 目前指令位置 (-2,147,483,648 ~ +2,147,483,648)

範例： **long** X_EP;
X_EP = i8092MF_GET_EP(1, AXIS_X);
//讀取第 1 卡 X 軸，目前 ENCODER 回授位置值。

3.3 讀取目前速度

- **DWORD i8092MF_GET_CV(BYTE cardNo, WORD axis)**

功能： 讀取軸目前之運動速度。

參數： **cardNo:** 指定卡號
axis: 指定軸號碼 X 或 Y(1 or 2)

回應： 目前速度(PPS)

範例： **DWORD dwdata;**
dwdata = i8092MF_GET_CV(1, AXIS_X);
//讀取第 1 卡 X 軸，目前之運動速度。

3.4 讀取目前加速度

- **DWORD i8092MF_GET_CA(BYTE cardNo, WORD axis)**

功能： 讀取軸目前之運動加速度 PPS/Sec。

參數： **cardNo:** 指定卡號
axis: 指定軸號碼 X 或 Y(1 or 2)

回應： 目前加速度(PPS/Sec)

範例： **DWORD dwdata;**
dwdata = i8092MF_GET_CA(1, AXIS_X);
//讀取第 1 卡 X 軸，目前之運動加速度。

3.5 讀取目前 DI 狀態

● **BYTE i8092MF_GET_DI(BYTE cardNo, WORD axis, WORD nType)**

功能： 讀取軸之輸入點狀態。

參數：

cardNo:	指定卡號
axis:	指定軸號碼 X 或 Y(1 or 2)
nType:	0 → DRIVING (檢查 i8092 有沒有輸出脈波)
	1 → LIMIT+ (檢查有沒有碰觸前極限)
	2 → LIMIT- (檢查有沒有碰觸後極限)
	3 → EMERGENCY (檢查緊急停止信號)
	4 → ALARM (檢查警報信號)
	5 → HOME (檢查原點信號)
	6 → NEAR HOME (檢查近原點信號)
	7 → IN3 (檢查 IN3 信號)
	8 → INPOS (檢查 INPOS 信號)
	9 → Z PHASE (檢查 INPOS 信號)

回應：

YES:	on
NO:	off

範例：

```
if (i8092MF_GET_DI(1, AXIS_X, 1) == YES)
{
    //讀取第 1 卡 X 軸，前極限信號處理。
}
```

● **WORD i8092MF_GET_DI_ALL(BYTE cardNo, WORD axis)**

功能： 讀取軸所有之輸入點狀態。

參數： **cardNo:** 指定卡號
 axis: 指定軸號碼 X 或 Y(1 or 2)

回應： — 16 bits 值 (0=Low,1=High)

bit 0	NHOME 信號
bit 1	HOME 信號
bit 2	Z-PHASE 信號
bit 3	EMG 信號(Only AXIS_X)
bit 4	EXP+ 信號
bit 5	EXP- 信號
bit 6	READY(INPOS) 信號
bit 7	ALARM 信號
bit 8	N/A
bit 9	N/A
bit 10	N/A
bit 11	IN3 信號
bit 12	N/A
bit 13	N/A
bit 14	LMT+ 信號
bit 15	LMT- 信號

範例： **WORD DI_Flag=i8092MF_GET_DI_ALL(1, AXIS_X) ;**
 //讀取第 1 卡 X 軸 DI 信號。

3.6 讀取目前 ERROR 狀態

- BYTE i8092MF_GET_ERROR(BYTE cardNo)

功能： 讀取軸運動有無錯誤發生。

參數： cardNo: 指定卡號

回應： YES: 有錯誤發生(欲讀錯誤碼請搭配使用
i8092MF_GET_ERROR_CODE)

NO: 沒有錯誤

範例： if (i8092MF_GET_ERROR(1) == YES)

{

//讀取第 1 卡，錯誤停止處理。

WORD ErrorCode_X = i8092MF_GET_ERROR_CODE(1, AXIS_X);

WORD ErrorCode_Y = i8092MF_GET_ERROR_CODE(1, AXIS_Y);

if ((ErrorCode_X & ErrorCode_Y) == 256)

{

//表示使用了 6.5.4 功能，使軸運動停止，請故障排除後，清除停止狀態。

i8092MF_CLEAR_STOP(1);

}

}

● **WORD i8092MF_GET_ERROR_CODE(BYTE cardNo, WORD axis)**

功能： 讀取各軸之錯誤碼。

參數： **cardNo:** 指定卡號
axis: 指定軸號碼 X 或 Y(1 or 2)

回應： **0:** 沒有任何錯誤
 非零值請參考下表,如同時有多個錯誤,會回傳其所有錯誤之錯誤碼總和。

錯誤碼	原因	說明
1	SOFT LIMIT+	碰觸軟體前極限
2	SOFT LIMIT-	碰觸軟體後極限
4	LIMIT+	碰觸前極限
8	LIMIT-	碰觸後極限
16	ALARM	伺服警報
32	EMERGENCY	緊急停止
64	N/A	N/A
128	N/A	N/A
256	請參考 6.5.4	軟體緊急停止

例如 48 表示"伺服警報"及"緊急停止" 同時發生

範例：

```
if (i8092MF_GET_ERROR_CODE(1, AXIS_X) == 10)
{
    //讀取第 1 卡 X 軸，碰觸軟體後極限(2)及碰觸後極限(8) 的錯誤處理。
    //軟體後極限(SOFT LIMIT-)及後極限(LIMIT-)設同一點。
}
```

3.7 一般 DO 輸出

● **void i8092MF_SET_OUT0(BYTE cardNo, WORD axis, WORD nLevel)**

功能： 設定軸之驅動器異常(ALARM)輸入參數。

參數： **cardNo:** 指定卡號
axis: 指定軸號碼(參考表 2-1)
nLevel: DO 輸出: 0=OFF,1=ON

回應： 無

範例：

```
i8092MF_SET_OUT0 (1, AXIS_XY, 1);
//設定第 1 卡 X Y 軸，DO 輸出為 ON。
```

4 FRnet 功能(i8092F 專用函式)

4.1 FRnet DI 讀取

- **WORD i8092MF_FRNET_RA**(BYTE *cardNo*, WORD *wRA*)

功能： 讀取 FRnet 的數位輸入資料。

參數： *cardNo*: 指定卡號
 wRA: 群組範圍(8~15)RA8~RA15

回應： **WORD** 16-位元輸入資料

範例： **WORD IN_Data;**
 IN_Data = i8092MF_FRNET_RA(1, 8);
 //設定第 1 卡，RA 群組 = 8。

4.2 FRnet DO 寫入

- **void i8092MF_FRNET_SA**(BYTE *cardNo*, WORD *wSA*, WORD *data*)

功能： 寫入 FRnet 的數位輸出資料。

參數： *cardNo*: 指定卡號
 wSA: 群組範圍(0~7)SA0~SA7
 dara: 16-位元資料

回應： 無

範例： **i8092MF_FRNET_SA(1, 0, 0xffff);**
 //設定第 1 卡，SA 群組 = 0，16 位元資料為 0xffff。

5 軸自動歸零

i8092 提供軟體自動歸零功能(軟體功能的巨集函式，會耗用系統資源)，只要經適當設定後，即可下指令自動執行，主要步驟如下：

- 以高速尋找近原點開關
- 以低速尋找原點開關
- 以低速尋找伺服馬達 Z 相信號
- 以高速運動到補正值(Offset)位置(程式原點)

設定時，其中步驟可以選擇不執行，以符合客戶實際需求動作，執行時完全自動執行，簡化程式設計。

i8092 亦提供軟體單步驟歸零功能

- 以指定速度尋找近原點開關
- 以指定速度原點開關
- 以指定速度伺服馬達 Z 相信號

5.1 軟體歸零相關函式

請參考相關指令：

- 3.1.6 設定軸前後極限的輸入觸發邏輯
- 3.1.8 設定軸近原點(NHOME)輸入觸發邏輯
- 3.1.9 設定軸原點(HOME)輸入觸發邏輯
- 3.1.10 設定軸前後軟體極限,參考來源及取消
- 4.1.2 設定軸初始速度
- 4.1.4 設定軸加速度

5.2 軟體自動歸零功能

● BYTE i8092MF_AUTO_HOME(BYTE cardNo, WORD axis, WORD nStep1, WORD nStep2, WORD nStep3, WORD nStep4, long IOffset, DWORD dwV, DWORD dwHV, long IBack)

功能： 軟體自動歸零功能。

(軟體功能的巨集函式，會耗用系統資源)

參數：

- cardNo:** 指定卡號
- axis:** 指定軸號碼 AXIS_X, AXIS_Y (參考表 2-1)
- nStep1:** 設定 0=不執行, 1=朝正向尋找, 2=朝負向尋找
- nStep2:** 設定 0=不執行, 1=朝正向尋找, 2=朝負向尋找
- nStep3:** 設定 0=不執行, 1=朝正向尋找, 2=朝負向尋找
- nStep4:** 設定 0=不執行, 1=執行
- IOffset:** 補正值(- 2,147,483,647~+ 2,147,483,647)
- dwV:** 驅動速度
- dwHV:** 歸零速度
- IBack:** 異常反向移出 Pulse 數(正好在 Senser 上時或先碰到極限開關後返回碰觸 Senser 後)

自動歸零步驟(Homing Step)

步驟	動作	運動速度	開關
步驟 1	以高速尋找近原點開關	驅動速度 (dwV)	近原點(ST0)
步驟 2	以歸零速度尋找原點開關	歸 零 速 度 (dwHV)	原點(ST1)
步驟 3	以歸零速度尋找伺服馬達 Z 相信號	歸 零 速 度 (dwHV)	Z 相信號(ST2)
步驟 4	以高速運動到位移值	驅動速度 (dwV)	

回應：

- 0: 正常結束。
- 4: 伺服 ALARM 停止。
- 8: 緊急停止。
- 253: 步驟 3 如果壓在 Z-PHASE(與 HOME 同時 ON)錯誤。
- 254: 找不到 Senser(NHOME, HOME)。
- 255: 指定軸錯誤。

範例：

```

BYTE m_Axis=AXIS_X;
i8092MF_SET_MAX_V(cardNo, m_Axis, 16000);
i8092MF_SET_A(cardNo, m_Axis, 50000);    //set Acc =50000 PPS/S
i8092MF_SET_SV(cardNo, m_Axis, 1000);
BYTE RR=i8092MF_AUTO_HOME(cardNo, m_Axis, 2, 2, 0, 1,
10000, 10000, 2000, 6000);
Sleep(100);
if (RR==0)

```



```

{
    i8092MF_SET_LP(cardNo,m_Axis,0);
    i8092MF_SET_EP(cardNo,m_Axis,0);
}

```

//設定第 1 卡 **AXIS_X** 以下表為執行範例:

	輸入信號	尋找方向	尋找速度
步驟 1	近原點 (ST0) Low active	—	10000 (PPS) (V)
步驟 2	原點 (ST1) Low active	—	2000 (PPS) (HV)
步驟 3	Z相信號 (ST2) High active	+	2000 (PPS) (HV)
步驟 4	+10000 pulse 補正 (offset)		10000 (PPS) (V)

5.3 軟體單步驟歸零功能

5.3.1 尋找近原點開關

● **BYTE i8092MF_SEARCH_NHOME**(BYTE cardNo, WORD axis, WORD nDir, DWORD v,long IBack)

功能： 以指定速度尋找近原點開關。
(軟體功能的巨集函式，會耗用系統資源)

參數： **cardNo:** 指定卡號
axis: 指定軸號碼 **AXIS_X,AXIS_Y** (參考表 2-1)
nDir: 設定 0=朝正向尋找,1=朝負向尋找
v: 驅動速度
dwHV: 歸零速度
IBack: 異常反向移出 Pulse 數
(正好在 **Senser** 上時或先碰到極限開關後返回碰觸 **Senser** 後)

回應： **0:** 正常結束。
4: 伺服 **ALARM** 停止。
8: 緊急停止。
254: 找不到 **Senser(NHOME,HOME)**。
255: 指定軸錯誤。

範例： **BYTE m_Axis=AXIS_X;**
i8092MF_SET_MAX_V(cardNo, m_Axis, 16000);
i8092MF_SET_A(cardNo, m_Axis,50000); //set Acc =50000 PPS/S
i8092MF_SET_SV(cardNo, m_Axis, 1000);
BYTE RR=i8092MF_SEARCH_NHOME(cardNo, m_Axis, 1, 2000,5000);
Sleep(100);
if (RR==0)
{
i8092MF_SET_LP(cardNo,m_Axis,0);
i8092MF_SET_EP(cardNo,m_Axis,0);
}
//設定第 1 卡 **AXIS_X** 執行範例:

5.3.2 尋找近原點開關

● **BYTE i8092MF_SEARCH_HOME**(**BYTE cardNo**, **WORD axis**, **WORD nDir**, **DWORD v**, **long IBack**)

功能： 以指定速度尋找原點開關。
(軟體功能的巨集函式，會耗用系統資源)

參數：

cardNo:	指定卡號
axis:	指定軸號碼 AXIS_X , AXIS_Y (參考表 2-1)
nDir:	設定 0=朝正向尋找, 1=朝負向尋找
v:	驅動速度
dwHV:	歸零速度
IBack:	異常反向移出 Pulse 數 (正好在 Senser 上時或先碰到極限開關後返回碰觸 Senser 後)

回應：

0:	正常結束。
4:	伺服 ALARM 停止。
8:	緊急停止。
254:	找不到 Senser (NHOME , HOME)。
255:	指定軸錯誤。

範例：

```
BYTE m_Axis=AXIS_X;
i8092MF_SET_MAX_V(cardNo, m_Axis, 16000);
i8092MF_SET_A(cardNo, m_Axis, 50000);    //set Acc =50000 PPS/S
i8092MF_SET_SV(cardNo, m_Axis, 1000);
BYTE RR=i8092MF_SEARCH_HOME(cardNo, m_Axis, 1, 2000, 5000);
Sleep(100);
if (RR==0)
{
    i8092MF_SET_LP(cardNo, m_Axis, 0);
    i8092MF_SET_EP(cardNo, m_Axis, 0);
}
//設定第 1 卡 AXIS_X 執行範例:
```

5.3.3 尋找伺服馬達 Z 相信號

● **BYTE i8092MF_SEARCH_ZPHASE(BYTE cardNo, WORD axis, WORD nDir, DWORD v)**

功能： 指定速度尋找伺服馬達 Z 相信號。
(軟體功能的巨集函式，會耗用系統資源)

參數： **cardNo:** 指定卡號
axis: 指定軸號碼 **AXIS_X,AXIS_Y** (參考表 2-1)
nDir: 設定 0=朝正向尋找,1=朝負向尋找
v: 驅動速度

回應： **0:** 正常結束。
4: 伺服 **ALARM** 停止。
8: 緊急停止。
253: 如果開始時已壓在 **Z-PHASE** 上。
255: 指定軸錯誤。

範例： **BYTE m_Axis=AXIS_X;**
i8092MF_SET_MAX_V(cardNo, m_Axis, 16000);
i8092MF_SET_A(cardNo, m_Axis,50000); //set Acc =50000 PPS/S
i8092MF_SET_SV(cardNo, m_Axis, 1000);
BYTE RR=i8092MF_SEARCH_ZPHASE(cardNo, m_Axis, 1, 1000);
Sleep(100);
if (RR==0)
{
i8092MF_SET_LP(cardNo,m_Axis,0);
i8092MF_SET_EP(cardNo,m_Axis,0);
}
//設定第 1 卡 **AXIS_X** 執行範例：

6 軸控功能

6.1 各軸獨立運動

- 單軸運動中，各軸可在任一時間同時運動。
- 各軸下完指令後，完全獨立運作不會互相干擾。
- 可單獨對每一軸下獨立指令，多工運動(各軸不補間)。
- 在運動執行中，我們可以動態改變參數值，包含位移脈波數、速度....等等。
- 也可以中途令其減速停止或立即停止...，以順應我們對運動控制不同的需求。

6.1.1 設定單軸加減速模式

● **void i8092MF_NORMAL_SPEED(BYTE cardNo, WORD axis , WORD nMode)**

功能： 設定速度模式。

參數： **cardNo:** 指定卡號
 axis: 指定軸號碼 (參考表 2-1)
 nMode: 0 → 對稱 T 曲線 (SV、V、A、AO)
 1 → 對稱 S 曲線 (SV、V、K、AO)
 2 → 非對稱 T 曲線 (SV、V、A、D、AO)

回應： 無

範例： **BYTE cardNo=1; //設定第 1 號卡。**

i8092MF_SET_MAX_V(cardNo, AXIS_XY, 20000); //設定 XY 軸最高速 20K PPS。

//=====

i8092MF_NORMAL_SPEED(cardNo, AXIS_XY, 0); //設定 XY 軸為對稱 T 曲線。

i8092MF_SET_V(cardNo, AXIS_XY, 2000); //設定 XY 軸速度=2000 PPS。

i8092MF_SET_A(cardNo, AXIS_XY,1000); //設定 XY 軸加速度=1000 PPS/S。

i8092MF_SET_SV(cardNo, AXIS_XY, 2000); //設定 XY 軸初始速度=2000

PPS。

i8092MF_SET_AO(cardNo, AXIS_XY, 9); //設定 XY 軸減速(保留脈波數)= 9

PPS。

i8092MF_FIXED_MOVE(cardNo, AXIS_XY, 10000); //XY 軸移動 10000

Pulse。

```
//=====
i8092MF_NORMAL_SPEED(cardNo, AXIS_XY,1); //設定 XY 軸為對稱 S 曲線。
i8092MF_SET_V(cardNo, AXIS_XY, 2000); //設定 XY 軸速度=2000 PPS。
i8092MF_SET_K(cardNo, AXIS_XY, 3000); // XY 軸加速度變化率=3000
PPS/S。
i8092MF_SET_SV(cardNo, AXIS_XY, 200); //設定 XY 軸初始速度=200 PPS。
i8092MF_SET_AO(cardNo, AXIS_XY, 9); //設定 XY 軸減速(保留脈波數)= 9
PPS。
i8092MF_FIXED_MOVE(cardNo, AXIS_XY, -10000); //XY 軸移動-10000
Pulse。
```

```
//=====
i8092MF_NORMAL_SPEED(cardNo, AXIS_XY,2); //設定 XY 軸為非對稱 T 曲
線。
i8092MF_SET_V(cardNo, AXIS_XY, 2000); //設定 XY 軸速度=2000 PPS。
i8092MF_SET_A(cardNo, AXIS_XY,1000 ); //設定 XY 軸加速度=1000 PPS/S。
i8092MF_SET_D(cardNo, AXIS_XY, 500); //設定 XY 軸減速度=500 PPS/S。
i8092MF_SET_SV(cardNo, AXIS_XY, 200); //設定 XY 軸初始速度=200 PPS。
i8092MF_SET_AO(cardNo, AXIS_XY, 9); //設定 XY 軸減速(保留脈波數)= 9
PPS。
i8092MF_FIXED_MOVE(cardNo, axis, 10000); //執行 XY 軸移動 10000 Pulse。
```

備註： 請搭配設定相關速度參數……。

6.1.2 設定軸初始速度

● void i8092MF_SET_SV(BYTE cardNo, WORD axis, DWORD data)

功能： 設定軸之初始速度。

參數： **cardNo:** 指定卡號
axis: 指定軸號碼 (參考表 2-1)
data: 設定速度值 (最大值請參考 3.1.3) PPS

回應： 無

範例： i8092MF_SET_SV(1, AXIS_X, 1000);
 //設定第 1 卡 X 軸，初始速度為 1000 PPS。

6.1.3 設定軸定速度

● **void i8092MF_SET_V**(BYTE *cardNo*, WORD *axis*, DWORD *data*)

功能： 設定軸之定速度。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼 (參考表 2-1)
data: 設定速度值 (最大值請參考 3.1.3) PPS

回應： 無

範例： **i8092MF_SET_V(1, AXIS_X, 120000L);**
//設定第 1 卡 X 軸，定速度為 120000 PPS。

6.1.4 設定軸加速度

● **void i8092MF_SET_A**(BYTE *cardNo*, WORD *axis*, DWORD *data*)

功能： 設定軸之加速度。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼 (參考表 2-1)
data: 設定加速度值 (PPS/Sec)

回應： 無

範例： **i8092MF_SET_A (1, AXIS_X, 100000L);**
//設定第 1 卡 X 軸，加速度為 100K PPS/Sec。

6.1.5 設定軸減速度

● **void i8092MF_SET_D**(BYTE *cardNo*, WORD *axis*, DWORD *data*)

功能： 設定軸之減速度。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼 (參考表 2-1)
data: 設定減速度值 (PPS/Sec)

回應： 無

範例： **i8092MF_SET_D(1, AXIS_X, 100000L);**
//設定第 1 卡 X 軸，減速度為 100K PPS/Sec。

6.1.6 設定軸加速度變化率

● **void i8092MF_SET_K**(BYTE *cardNo*, WORD *axis*, DWORD *data*)

功能： 設定軸之輸出加速度變化率。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼 (參考表 2-1)
data: 設定加速度變化率值 (Jerk PPS/ Sec²)

回應： 無

範例： **i8092MF_SET_K(1, AXIS_X, 500000L);**
//設定第 1 卡 X 軸，加速度變化率為 500K PPS/ Sec²。

6.1.7 設定軸減速(保留脈波數)

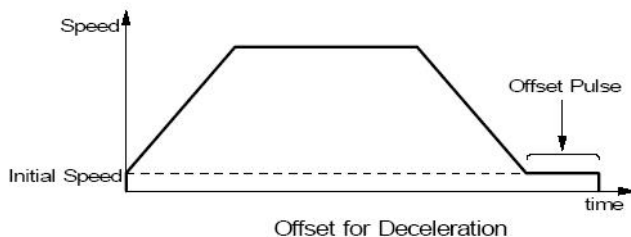
● **void i8092MF_SET_AO**(BYTE *cardNo*, WORD *axis*, short int *data*)

功能： 於固定脈波數運動控制時，至目標前保留低速輸出 Offset Pulse 數，如圖所示 Offset Pulse 位置。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼 (參考表 2-1)
data: 設定 Offset Pulse 值 (-32,768 ~ +32,767)

回應： 無

範例： **i8092MF_SET_AO(1, AXIS_X, 200);**
//設定第 1 卡 X 軸，Offset Pulse 為 200 Pulses。



6.1.8 固定脈波數輸出

● **BYTE i8092MF_FIXED_MOVE**(BYTE *cardNo*, WORD *axis*, long *data*)

功能： 執行單軸固定步數輸出。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼 (參考表 2-1)
data: 指定步數(-268,435,455 ~ + 268,435,455)

回應： YES: 有錯誤發生(欲讀錯誤碼請搭配使用
i8092MF_GET_ERROR_CODE)
NO: 沒有錯誤

範例： **BYTE cardNo=1;** //設定第 1 號卡。
i8092MF_SET_MAX_V(cardNo, AXIS_XY, 20000); //設定 XY 軸最高速 20K PPS。
i8092MF_NORMAL_SPEED(cardNo, AXIS_XY, 0); //設定 XY 軸為對稱 T 曲線。
i8092MF_SET_V(cardNo, AXIS_XY, 2000); //設定 XY 軸速度=2000 PPS。
i8092MF_SET_A(cardNo, AXIS_XY, 1000); //設定 XY 軸加速度=1000 PPS/S。

```

i8092MF_SET_SV(cardNo, AXIS_XY, 2000); //設定 XY 軸初始速度=2000
PPS。
i8092MF_SET_AO(cardNo, AXIS_XY, 9); //設定 XY 軸減速(保留脈波數)= 9
PPS。
i8092MF_FIXED_MOVE(cardNo, AXIS_XY, 10000); //XY 軸移動 10000
Pulse。

```

6.1.9 連續脈波輸出

● **BYTE i8092MF_CONTIUNE_MOVE(BYTE cardNo, WORD axis, long data)**

功能： 執行單軸連續脈波輸出。

參數：

cardNo:	指定卡號
axis:	指定軸號碼 (參考表 2-1)
data:	指定速度: -V(CCW) ~ +V(CW) PPS , V=Vmin~Vmax

回應：

YES:	有錯誤發生(欲讀錯誤碼請搭配使用 i8092MF_GET_ERROR_CODE)
NO:	沒有錯誤

範例：

```

BYTE cardNo=1; //設定第 1 號卡。
i8092MF_SET_MAX_V(cardNo, AXIS_XY, 20000); //設定 XY 軸最高速 20K
PPS。
i8092MF_NORMAL_SPEED(cardNo, AXIS_XY, 0); //設定 XY 軸為對稱 T 曲線。
i8092MF_SET_V(cardNo, AXIS_XY, 2000); //設定 XY 軸速度=2000 PPS。
i8092MF_SET_A(cardNo, AXIS_XY, 1000); //設定 XY 軸加速度=1000 PPS/S。
i8092MF_SET_SV(cardNo, AXIS_XY, 2000); //設定 XY 軸初始速度=2000
PPS。
i8092MF_CONTIUNE_MOVE(cardNo, AXIS_XY, 1000); //1000 PPS 速度連續
移動。

```

6.2 補間運動

6.2.1 XY 軸補間

由於 i8092/F 為兩軸模組,因此其補間第一軸恆為 **AXIS_X** 軸,而第二軸就是 **AXIS_Y** 軸因此其位置參數,固定為 **X(FP1,CP1)**,後 **Y(FP2,CP2)**請您在使用時請注意。

6.2.2 設定補間加減速模式

● **void i8092MF_VECTOR_SPEED(BYTE cardNo, WORD nMode)**

功能： 設定向量加減速模式。

參數： **cardNo:** 指定卡號
nMode: 0 → 二軸(直線&弧&圓)固定向量速度 (VV)
1 → 二軸直線對稱 T 曲線 (VSV、VV、VA、VAO)
2 → 二軸直線對稱 S 曲線 (VSV、VV、VK、VAO)
3 → 二軸直線非對稱 T 曲線 (VSV、VV、VA、VD、VAO)
4 → N/A
5 → 二軸(弧&圓)對稱 T 曲線 (VSV、VV、VA、VAO)
6 → 二軸(弧&圓)非對稱 T 曲線 (VSV、VV、VA、VD、VAO)

回應： 無

範例： **BYTE cardNo=1; //設定第 1 號卡。**
i8092MF_SET_MAX_V(cardNo, AXIS_XY, 20000);
//設定 XY 軸最高速 20K PPS
//=====
i8092MF_VECTOR_SPEED(cardNo, 0);
//二軸(直線&弧&圓)固定向量速度 VSV=VV
i8092MF_SET_VSV(cardNo, 1000); //設定第 1 卡,向量初始速度為 1000 PPS。
i8092MF_SET_VV(cardNo, 1000); //設定第 1 卡,向量定速度為 1000 PPS。
i8092MF_LINE_2D(1, 12000, 10000); //執行向量 2D 補間。

//=====
BYTE axis=AXIS_XY; //for AXIS_XY
i8092MF_SET_MAX_V(cardNo, axis, 20000);
i8092MF_VECTOR_SPEED(cardNo, 1);
//二軸直線對稱 T 曲線 (VSV、VV、VA、VAO)。
i8092MF_SET_VSV(cardNo, 500); //設定向量初始速度為 500 PPS。
i8092MF_SET_VV(cardNo, 2000); //設定向量速度為 2000 PPS。
i8092MF_SET_VA(cardNo, 1000); //設定向量加速速度為 1000 PPS。

```

i8092MF_LINE_2D(cardNo, 20000, 10000); //執行向量 2D 補間。
//=====
BYTE axis=AXIS_XY; //for AXIS_XY
i8092MF_SET_MAX_V(cardNo, axis, 30000);
i8092MF_VECTOR_SPEED(cardNo, 2);
//二軸直線對稱 S 曲線(VSV、VV、VA、VK、AO)。
i8092MF_SET_VSV(cardNo, 200); //設定向量初始速度為 500 PPS。
i8092MF_SET_VV(cardNo, 2000); //設定向量速度為 2000 PPS。
i8092MF_SET_VK(cardNo, 12500); //設定向量加速度變化率為 3000
PPS/s^2。
i8092MF_SET_VAO(cardNo, 20); //設定軸向量減速(保留脈波數)20 Pulse。
i8092MF_LINE_2D(cardNo, 10000, 10000); //執行向量 2D 補間。

```

```

//=====
BYTE axis=AXIS_XY; //for AXIS_XY
i8092MF_SET_MAX_V(cardNo, axis, 20000);
i8092MF_VECTOR_SPEED(cardNo, 3);
//二軸直線非對稱 T 曲線 (VSV、VV、VA、VD、VAO)。
i8092MF_SET_VSV(cardNo, 100); //設定向量初始速度為 500 PPS。
i8092MF_SET_VV(cardNo, 2000); //設定向量速度為 2000 PPS。
i8092MF_SET_VA(cardNo, 1000); //設定向量加速度為 1000 PPS/s。
i8092MF_SET_VD(cardNo, 500); //設定向量減速度為 500 PPS/s。
i8092MF_SET_VAO(cardNo, 20); //設定軸向量減速(保留脈波數)20 Pulse。
i8092MF_LINE_2D(cardNo, 10000, 5000); //執行向量 2D 補間。

```

```

//=====
long fp1=11000;
long fp2=9000;
long c1=10000;
long c2=0;
int sv=100;
int v=3000;
int a=5000;
int d=5000;
i8092_SET_MAX_V(cardNo, AXIS_XY, 8000);
i8092_VECTOR_SPEED(cardNo, 5);
//二軸(弧&圓)對稱 T 曲線 (VSV、VV、VA、VAO)。
i8092_SET_VSV(cardNo, sv); //設定向量初始速度為 sv PPS。
i8092_SET_VV(cardNo, v); //設定向量速度為 v PPS。
i8092_SET_VA(cardNo, a); //設定向量加速度為 a PPS/s。
i8092_SET_VAO(cardNo, 0); //設定軸向量減速(保留脈波數)0 Pulse。

```

```

i8092_ARC_CW(cardNo, c1,c2, fp1, fp2); //執行二軸順時針圓弧補間。

//=====================================================
long c1=300;
long c2=0;
int sv=100;
int v=3000;
int a=125;
int d=12;
i8092_SET_MAX_V(cardNo, AXIS_XY, 8000);
i8092_VECTOR_SPEED(cardNo, 6);
//二軸(弧&圓)對稱 T 曲線 (VSV、VV、VA、VAO)。
i8092_SET_VSV(cardNo, sv); //設定向量初始速度為 sv PPS。
i8092_SET_VV(cardNo, v); //設定向量速度為 v PPS。
i8092_SET_VA(cardNo, a); //設定向量加速度為 a PPS/s。
i8092_SET_VD(cardNo, d); //設定向量減速度為 d PPS/s。
i8092_SET_VAO(cardNo, 0); //設定軸向量減速(保留脈波數)0 Pulse。
i8092_CIRCLE_CW(cardNo, c1, c2); //執行二軸順時針圓形補間。

```

備註： 請搭配設定相關向量速度參數.....。

6.2.3 設定軸向量初始速度

● **void i8092MF_SET_VSV(BYTE cardNo, DWORD data)**

功能： 設定軸之向量初始速度。

參數： **cardNo:** 指定卡號
data: 設定向量速度值 (最大值請參考 3.1.3) PPS

回應： 無

範例： **i8092MF_SET_VSV(1, 1000);**
 //設定第 1 卡，向量初始速度為 1000 PPS。

6.2.4 設定軸向量定速度

● **void i8092MF_SET_VV**(BYTE *cardNo*, DWORD *data*)

功能： 設定軸之向量定速度。

參數： *cardNo*: 指定卡號
data: 設定向量速度值 (最大值請參考 3.1.3) PPS

回應： 無

範例： **i8092MF_SET_VV(1, 120000L);**
//設定第 1 卡，向量定速度為 120000 PPS。

6.2.5 設定軸向量加速度

● **void i8092MF_SET_VA**(BYTE *cardNo*, DWORD *data*)

功能： 設定軸之向量加速度。

參數： *cardNo*: 指定卡號
data: 設定向量加速度值 (PPS/Sec)

回應： 無

範例： **i8092MF_SET_VA (1, 100000L);**
//設定第 1 卡，向量加速度為 100K PPS/Sec。

6.2.6 設定軸向量減速度

● **void i8092MF_SET_VD**(BYTE *cardNo*, DWORD *data*)

功能： 設定軸之向量減速度。

參數： *cardNo*: 指定卡號
data: 設定向量減速度值 (PPS/Sec)

回應： 無

範例： **i8092MF_SET_VD(1, 100000L);**
//設定第 1 卡，向量減速度為 100K PPS/Sec。

6.2.7 設定軸向量加速度變化率

● **void i8092MF_SET_VK**(BYTE *cardNo*, DWORD *data*)

功能： 設定軸之輸出向量加速度變化率。

參數： *cardNo*: 指定卡號
data: 設定向量加速度變化率值 (Jerk PPS/ Sec²)

回應： 無

範例： **i8092MF_SET_VK(1, 500000L);**
//設定第 1 卡，向量加速度變化率為 500K PPS/ Sec²。

6.2.8 設定軸向量減速(保留脈波數)

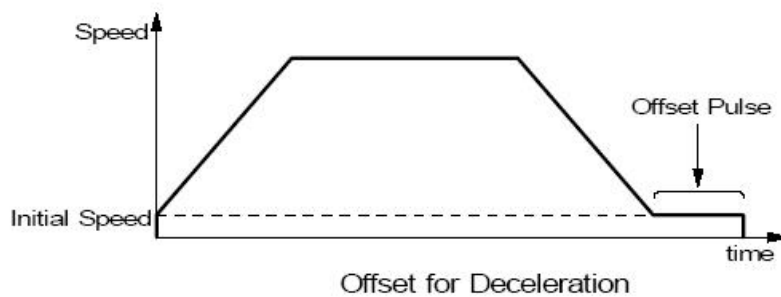
● **void i8092MF_SET_VAO**(BYTE *cardNo*, short int *data*)

功能： 於固定脈波數運動控制時，至目標前保留低速輸出 Offset Pulse 數，如圖所示 Offset Pulse 位置。

參數： *cardNo*: 指定卡號
data: 設定 Offset Pulse 值 (-32,768 ~ +32,767)

回應： 無

範例： **i8092MF_SET_VAO(1, 200);**
//設定第 1 卡補間軸，Offset Pulse 為 200 Pulses。



6.2.9 二軸直線補間

● **BYTE i8092MF_LINE_2D**(**BYTE cardNo**, **long fp1**, **long fp2**)

功能： 執行二軸直線補間。

參數： **cardNo**: 指定卡號
fp1: 指定 X 軸 Pulse 數(-8,388,607 ~ +8,388,607)
fp2: 指定 Y 軸 Pulse 數(-8,388,607 ~ +8,388,607)

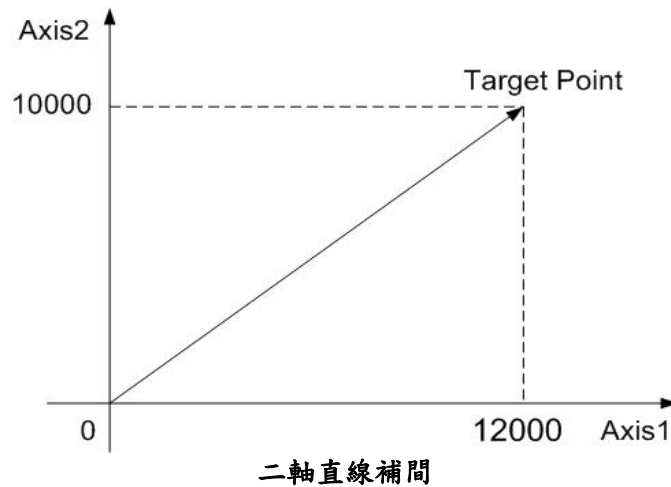
回應： **YES**: 有錯誤發生(欲讀錯誤碼請搭配使用

i8092MF_GET_ERROR_CODE)

NO: 沒有錯誤

範例： **i8092MF_LINE_2D(1, 12000, 10000);**

//設定第 1 卡，執行二軸直線補間。



6.2.10 二軸圓弧補間

● BYTE i8092MF_ARC_CW(BYTE *cardNo*, long *cp1*, long *cp2*, long *fp1*, long *fp2*)

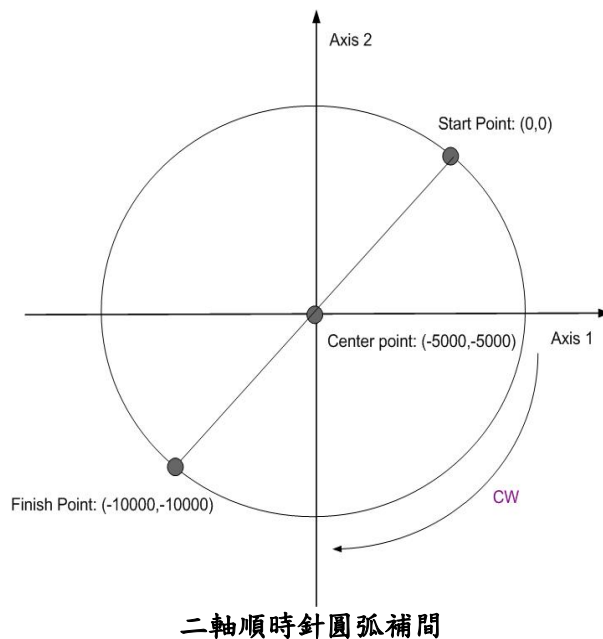
功能： 執行二軸順時針圓弧補間。

參數：

<i>cardNo</i> :	指定卡號
<i>cp1</i> :	指定 X 軸圓弧中心相對位置 (-8,388,607 ~ +8,388,607)
<i>cp2</i> :	指定 Y 軸圓弧中心相對位置 (-8,388,607 ~ +8,388,607)
<i>fp1</i> :	指定 X 軸圓弧終點相對位置 (-8,388,607 ~ +8,388,607)
<i>fp2</i> :	指定 Y 軸圓弧終點相對位置 (-8,388,607 ~ +8,388,607)

回應： YES: 有錯誤發生(欲讀錯誤碼請搭配使用
i8092MF_GET_ERROR_CODE)
NO: 沒有錯誤

範例： i8092MF_ARC_CW(1, -5000, -5000, -10000, -10000);
//設定第 1 卡，執行二軸順時針圓弧補間。



● **BYTE i8092MF_ARC_CCW**(**BYTE cardNo**, **long cp1**, **long cp2**, **long fp1**, **long fp2**)

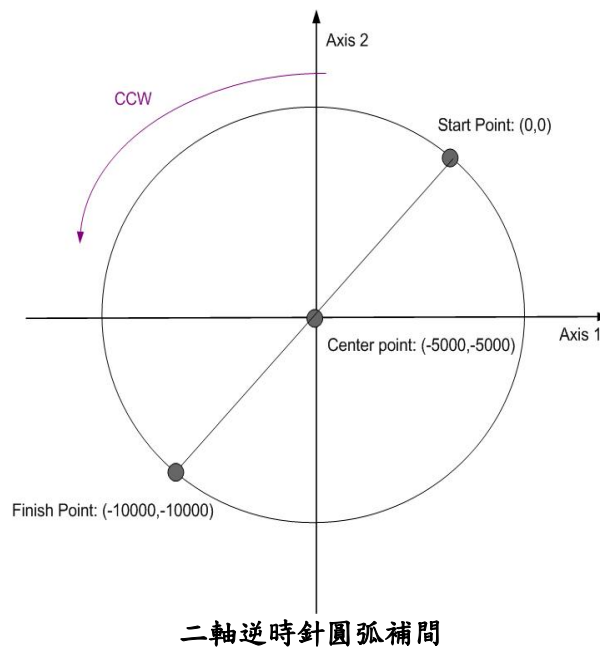
功能： 執行二軸逆時針圓弧補間。

參數：

cardNo:	指定卡號
cp1:	指定 X 軸圓弧中心相對位置 (-8,388,607 ~ +8,388,607)
cp2:	指定 Y 軸圓弧中心相對位置 (-8,388,607 ~ +8,388,607)
fp1:	指定 X 軸圓弧終點相對位置 (-8,388,607 ~ +8,388,607)
fp2:	指定 Y 軸圓弧終點相對位置 (-8,388,607 ~ +8,388,607)

回應： YES: 有錯誤發生(欲讀錯誤碼請搭配使用
i8092MF_GET_ERROR_CODE)
NO: 沒有錯誤

範例： **i8092MF_ARC_CCW(1, -5000, -5000, -10000, -10000);**
//設定第 1 卡，執行二軸逆時針圓弧補間。



6.2.11 二軸圓形補間

● BYTE i8092MF_CIRCLE_CW(BYTE *cardNo*, long *cp1*, long *cp2*)

功能： 執行二軸順時針圓形補間。

參數： *cardNo*: 指定卡號
cp1: 指定 X 軸圓弧中心相對位置
(-8,388,607 ~ +8,388,607)
cp2: 指定 Y 軸圓弧中心相對位置
(-8,388,607 ~ +8,388,607)

回應： YES: 有錯誤發生(欲讀錯誤碼請搭配使用
i8092MF_GET_ERROR_CODE)
NO: 沒有錯誤

範例： i8092MF_CIRCLE_CW(1, 0, 10000);
//設定第 1 卡，執行二軸順時針圓形補間。

● BYTE i8092MF_CIRCLE_CCW(BYTE *cardNo*, long *cp1*, long *cp2*)

功能： 執行二軸逆時針圓形補間。

參數： *cardNo*: 指定卡號
cp1: 指定 X 軸圓弧中心相對位置
(-8,388,6078 ~ +8,388,6078)
cp2: 指定 Y 軸圓弧中心相對位置
(-8,388,6078 ~ +8,388,6078)

回應： YES: 有錯誤發生(欲讀錯誤碼請搭配使用
i8092MF_GET_ERROR_CODE)
NO: 沒有錯誤

範例： i8092MF_CIRCLE_CCW(1, 0, 10000);
//設定第 1 卡，執行二軸逆時針圓形補間。

6.3 連續補間運動

連續補間運動如被異常中斷而停止，請參考 6.5.5 排除！

6.3.1 二軸矩形連續補間

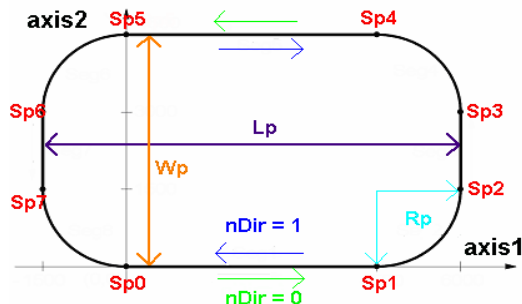
- BYTE i8092MF_RECTANGLE(
 BYTE cardNo, WORD nAcc, WORD Sp, WORD nDir, long Lp, long
 Wp, long Rp, DWORD RSV, DWORD RV, DWORD RA, DWORD RD)

功能： 執行二軸矩形補間。
(軟體功能的巨集函式，會耗用系統資源)

參數：
 cardNo: 指定卡號
 nAcc: 0 → 定速度補間
 1 → 對稱 T 曲線加減速補間
 Sp: 設定起點 0 ~ 7 (Sp0 ~ Sp7 如下圖所示)
 nDir: 設定方向 0、1 (CCW or CW)
 Lp: 設定長度 Pulse 數(1 ~ 8,388,607)
 Wp: 設定寬度 Pulse 數(1 ~ 8,388,607)
 Rp: 設定圓弧半徑 Pulse 數(1 ~ 8,388,607)
 RSV: 設定矩形補間向量起始速度(PPS)
 RV: 設定矩形補間向量速度(PPS)
 RA: 設定矩形補間向量加速度(PPS/Sec)
 RD: 設定矩形末段補間向量減速度(PPS/Sec)

回應： YES: 有錯誤發生(欲讀錯誤碼請搭配使用
i8092MF_GET_ERROR_CODE)
 NO: 沒有錯誤

範例： BYTE cardNo=1; //設定第 1 號卡。
int sv=1000; //設定向量初始速度為 1000 PPS。
int v=10000; //設定向量速度為 10000 PPS。
int a=5000; //設定向量加速度為 5000 PPS/s。
int d=5000; //設定向量減速度為 5000 PPS/s。
i8092MF_SET_MAX_V(cardNo, AXIS_XY, 16000); //最高速度為 16000 PPS。
i8092MF_RECTANGLE(
 cardNo, 1, 0, 0, 20000, 10000, 1000, sv, v, a, d);
//設定第 1 卡，執行二軸矩形連續補間，減速點會自動運算。



6.3.2 二軸直線連續補間

- **void i8092MF_LINE_2D_INITIAL**(BYTE *cardNo*, DWORD *VSV*, DWORD *VV*, DWORD *VA*)

功能： 二軸直線連續補間初始設定(對稱 T 曲線加減速)。

參數： *cardNo*: 指定卡號
VSV: 設定向量初始速度(PPS)
VV: 設定向量速度(PPS)
VA: 設定加速度(PPS/Sec)

回應： 無

範例： 和 **void i8092MF_LINE_2D_CONTINUE**(BYTE *cardNo*, WORD *nType*, long *fp1*, float *fp2*) 並用。

- **BYTE i8092MF_LINE_2D_CONTINUE**(BYTE *cardNo*, WORD *nType*, long *fp1*, long *fp2*)

功能： 執行二軸直線連續補間。
(軟體功能的巨集函式，會耗用系統資源)

參數： *cardNo*: 指定卡號
nType: 0 → 二軸直線連續補間
1 → 二軸直線連續補間結束
fp1: 指定 X 軸 Pulse 數(-8,388,607 ~ +8,388,607)
fp2: 指定 Y 軸 Pulse 數(-8,388,607 ~ +8,388,607)

回應： YES: 有錯誤發生(欲讀錯誤碼請搭配使用
i8092MF_GET_ERROR_CODE)
NO: 沒有錯誤

範例： **BYTE cardNo=1;** //設定第 1 號卡。
int sv=300; //設定向量初始速度為 PPS。
int v=18000; //設定向量速度為 PPS。
long a=500000; //設定向量加速度為 PPS/s。
int loop1;
i8092MF_SET_MAX_V(cardNo, AXIS_XY,160000L);
i8092MF_LINE_2D_INITIAL(cardNo, AXIS_X, AXIS_Y, sv, v, a);
for (loop1 = 0; loop1 < 10000; loop1++)
{
 i8092MF_LINE_2D_CONTINUE (cardNo, 0, 100, 100);
 i8092MF_LINE_2D_CONTINUE (cardNo, 0, -100, -100);
}
i8092MF_LINE_2D_CONTINUE (cardNo, 1, 100, 100);
//設定第 1 卡，執行 X、Y 兩軸直線連續補間運動。

6.3.3 二軸多段連續補間

● BYTE i8092MF_CONTINUE_INTP(BYTE cardNo, WORD nAcc, DWORD VSV, DWORD VV, DWORD VA, DWORD VD, BYTE nType[], long cp1[], long cp2[], long fp1[], long fp2[])

功能： 執行多點連續補間(對稱 T 曲線)。
(軟體功能的巨集函式，會耗用系統資源)

參數：

cardNo:	指定卡號
nAcc:	0 → 定速度補間 (vv) 1 → 對稱 T 曲線加減速補間 (vsv、vv、va、vd)
VSV:	設定補間向量起始速度(PPS)
VV:	設定補間向量速度(PPS)
VA:	設定補間向量加速度(PPS/Sec)
VD:	設定末段補間向量減速度(PPS/Sec)
nType[]:	連續補間點最大: 1024 點(0 ~ 1023) 1 → i8092MF_LINE_2D(BYTE cardNo, long fp1, long fp2) 2 → i8092MF_ARC_CW(BYTE cardNo, long cp1, long cp2, long fp1, long fp2) 3 → i8092MF_ARC_CCW(BYTE cardNo, long cp1, long cp2, long fp1, long fp2) 4 → i8092MF_CIRCLE_CW(BYTE cardNo, long cp1, long cp2) 5 → i8092MF_CIRCLE_CCW(BYTE cardNo, long cp1, long cp2) 7 → 連續補間結束
cp1[]:	指定 X 軸圓、弧中心相對位置(-8,388,607 ~ +8,388,607)
cp2[]:	指定 Y 軸圓、弧中心相對位置(-8,388,607 ~ +8,388,607)
fp1[]:	指定 X 軸 Pulse 數(-8,388,607 ~ +8,388,607) 指定 X 軸圓弧終點相對位置
fp2[]:	指定 Y 軸 Pulse 數(-8,388,607 ~ +8,388,607) 指定 Y 軸圓弧終點相對位置

回應： YES: 有錯誤發生(欲讀錯誤碼請搭配使用
i8092MF_GET_ERROR_CODE)
NO: 沒有錯誤

範例：

```
BYTE cardNo=1; //設定第 1 號卡。
int sv=100; //設定向量初始速度為 100 PPS。
int v=3000; //設定向量速度為 3000 PPS。
int a=2000; //設定向量加速度為 2000 PPS/s。
int d=2000; //設定向量減速度為 2000 PPS/s。
i8092MF_SET_MAX_V(cardNo, AXIS_XY, 20000); //設定各軸最高速度為 20K PPS。
BYTE nType[10]= { 1, 2, 1, 2, 1,7,0,0,0,0};
long cp1[10]= { 0, 10000, 0, 0, 0,0,0,0,0,0};
long cp2[10]= { 0, 0, 0,-10000, 0,0,0,0,0,0};
```

```

long fp1[10]=    { 10000, 10000,  1000, 10000,-31000,0,0,0,0,0};
long fp2[10]=    { 10000, 10000,    0,-10000,-10000,0,0,0,0,0};
i8092MF_CONTINUE_INTP(
cardNo, 1, sv, v, a, d, nType,cp1, cp2, fp1, fp2,fp3);
//設定第 1 卡，執行多點連續補間而減速點會自動運算。
//此範例主要以兩軸補間，直線搭配圓弧的運動，起點運動後最終將回到起點位置。

```

6.3.4 二軸比例運動

- **void** i8092MF_RATIO_INITIAL(BYTE *cardNo*,DWORD SV, DWORD V, DWORD A, **float** *ratio*)

功能： 比例運動初始設定(對稱 T 曲線加減速)。

參數：

cardNo:	指定卡號
SV:	設定比例運動初始速度(PPS)
V:	設定比例運動速度(PPS)
A:	設定比例運動加速度(PPS/Sec)
ratio:	設定兩軸的比例

回應： 無

範例： 和 **void** i8092MF_RATIO_2D(BYTE *cardNo*, WORD *nType*, **long** *data*, WORD *nDir*) 並用。

● **BYTE i8092MF_RATIO_2D**(**BYTE cardNo**, **WORD nType**, **long data**, **WORD nDir**)

功能： 執行比例連續運動。

(軟體功能的巨集函式，會耗用系統資源)

參數： **cardNo**: 指定卡號
nType: 0 → 比例連續運動
 1 → 比例運動結束
data: 比例運動 X 軸 Pulse 數(-8,388,607 ~ +8,388,607)
nDir: 比例運動 Y 軸方向:
 0 → 正轉 CW
 1 → 反轉 CCW

回應： YES: 有錯誤發生(欲讀錯誤碼請搭配使用

i8092MF_GET_ERROR_CODE)

NO: 沒有錯誤

範例： **BYTE cardNo=1**; //設定第 1 號卡。
int sv=300; //設定初始速度為 PPS。
int v=18000; //設定速度為 PPS。
long a=500000; //設定加速度為 PPS/s。
int loop1;
int loop2;
i8092MF_SET_MAX_V(cardNo, **AXIS_XY**,160000L);
i8092MF_RATIO_INITIAL(cardNo, **AXIS_U**, **AXIS_X**, sv, v, a, 0.36f);
for (loop2 = 0; loop2 < 5; loop2++)
{
 for (loop1 = 0; loop1 < 5; loop1++)
 {
 i8092MF_RATIO_2D(cardNo, 0, 3600, 0);
 i8092MF_RATIO_2D(cardNo, 0, 3600, 1);
 }
 i8092MF_RATIO_2D(cardNo, 0, 7200, 0);
 i8092MF_RATIO_2D(cardNo, 0, 3600, 1);
}
i8092MF_RATIO_2D(cardNo, 1, 7200, 0);
//設定第 1 卡，執行 X、Y 兩軸比例運動。

6.3.5 二軸混合連續補間

- **void i8092MF_MIX_2D_INITIAL**(BYTE *cardNo*, WORD *nAcc*, DWORD *VSV*, DWORD *VV*, DWORD *VA*)

功能： 二軸直線和圓弧連續補間初始設定。

參數： **cardNo:** 指定卡號
nAcc: 0 → 定速度補間 (vv)
1 → 對稱 T 曲線加減速補間 (vsv、vv、va)
VSV: 設定向量初始速度(PPS)
VV: 設定向量速度(PPS)
VA: 設定加速度(PPS/Sec)

回應： 無

範例： 和 **void i8092MF_MIX_2D_CONTINUE**(
BYTE *cardNo*, WORD *nAcc*, WORD *nType*, long *cp1*, long *cp2*, long *fp1*, long *fp2*)並用。

- **BYTE i8092MF_MIX_2D_CONTINUE**(BYTE *cardNo*, WORD *nAcc*,
WORD *nType*, long *cp1*, long *cp2*, long *fp1*, long *fp2*)

功能： 執行二軸直線和圓弧連續補間(可連續，無段數限制)。

參數： **cardNo:** 指定卡號
nAcc: 0 → 連續補間
1 → 結束連續補間減速停止(定速度不需減速)
nType: 1 → i8092MF_LINE_2D(BYTE *cardNo*, long *fp1*, long *fp2*)
2 → i8092MF_ARC_CW(BYTE *cardNo*, long *cp1*, long *cp2*, long *fp1*, long
fp2)
3 → i8092MF_ARC_CCW(BYTE *cardNo*, long *cp1*, long *cp2*, long *fp1*, long
fp2)
4 → i8092MF_CIRCLE_CW(BYTE *cardNo*, long *cp1*, long *cp2*)
5 → i8092MF_CIRCLE_CCW(BYTE *cardNo*, long *cp1*, long *cp2*)
cp1: 指定 X 軸圓、弧中心相對位置(-8,388,607 ~ +8,388,607)
cp2: 指定 Y 軸圓、弧中心相對位置(-8,388,607 ~ +8,388,607)
fp1: 指定 X 軸 Pulse 數(-8,388,607 ~ +8,388,607)
fp2: 指定 Y 軸 Pulse 數(-8,388,607 ~ +8,388,607)

回應： YES: 有錯誤發生(欲讀錯誤碼請搭配使用
i8092MF_GET_ERROR_CODE)
NO: 沒有錯誤

範例: BYTE cardNo=1; //設定第 1 號卡。

unsigned short sv=300; //設定向量初始速度為 PPS。

unsigned short v=8000; //設定向量速度為 PPS。

unsigned long a=50000; //設定向量加速度為 PPS/s。

unsigned short loop1;

i8092MF_SET_MAX_V(cardNo, AXIS_XY,16000L);

i8092MF_MIX_2D_INITIAL(cardNo, 1, sv, v, a);

i8092MF_MIX_2D_CONTINUE(cardNo, 0, 1, 0, 0, 50000, 0);

//1 segment

i8092MF_MIX_2D_CONTINUE(cardNo, 0, 2, 0,-20000, 0,-40000);

//2 segment

i8092MF_MIX_2D_CONTINUE(cardNo, 0, 3, 0,-10000, 0,-20000);

//3 segment

i8092MF_MIX_2D_CONTINUE(cardNo, 0, 3, 0, 10000, 0, 20000);

//4 segment

i8092MF_MIX_2D_CONTINUE(cardNo, 0, 1, 0, 0,-50000, 0);

//5 segment

i8092MF_MIX_2D_CONTINUE(cardNo, 0, 4, 0, 10000, 0, 0);

//6 segment

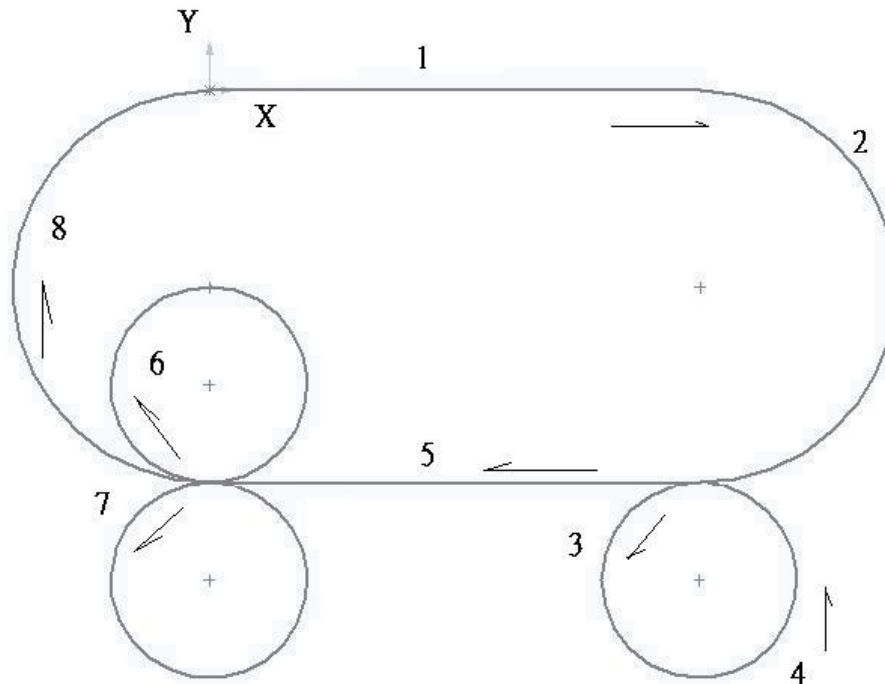
i8092MF_MIX_2D_CONTINUE(cardNo, 0, 5, 0,-10000, 0, 0);

//7 segment

i8092MF_MIX_2D_CONTINUE(cardNo, 1, 2, 0, 20000, 0, 40000);

//8 segment and end the Interpolation

//設定第 1 卡，執行 X、Y 兩軸連續補間運動。



6.5 其他功能

6.5.1 設定軸暫停

- **void i8092MF_DRV_HOLD(BYTE cardNo, WORD axis)**

功能： 指定軸運動暫停。

參數： **cardNo:** 指定卡號
axis: 指定軸號碼 (參考表 2-1)

回應： 無

範例： 請參考 6.5.2

6.5.2 設定軸啟動

- **void i8092MF_DRV_START(BYTE cardNo, WORD axis)**

功能： 指定軸開始動作。

參數： **cardNo:** 指定卡號
axis: 指定軸號碼 (參考表 2-1)

回應： 無

範例：

BYTE cardNo=1; //設定第 1 號卡。

i8092MF_DRV_HOLD(cardNo, AXIS_XY); //設定 XY 兩軸暫停移動。

i8092MF_SET_MAX_V(cardNo, AXIS_XY, 8000); //設定 XY 軸最高速 8K PPS。

i8092MF_NORMAL_SPEED(cardNo, AXIS_XY, 0); //設定 XY 軸對稱 T 曲線。

i8092MF_SET_V(cardNo, AXIS_X, 2000); //設定 X 軸速度=2,000 PPS。

i8092MF_SET_A(cardNo, AXIS_X, 10000); //設定 X 軸加速度 10,000 PPS/S。

i8092MF_SET_SV(cardNo, AXIS_X, 1000); //設定 X 初始速度 1,000 PPS。

i8092MF_SET_V(cardNo, AXIS_Y, 4000); //設定 Y 軸速度=4,000 PPS。

i8092MF_SET_A(cardNo, AXIS_Y, 10000); //設定 Y 軸加速度 10,000 PPS/S。

i8092MF_SET_SV(cardNo, AXIS_Y, 500); //設定 Y 初始速度 500 PPS。

i8092MF_FIXED_MOVE(cardNo, AXIS_X, 5000); //X 移動 5,000 Pulse。

i8092MF_FIXED_MOVE(cardNo, AXIS_Y, 8000); //Y 移動 8,000 Pulse。

i8092MF_DRV_START(cardNo, AXIS_XY); //開始 XY 兩軸同時移動。

6.5.3 等待完成軸運動

- **BYTE i8092MF_STOP_WAIT(BYTE cardNo, WORD axis)**

功能： 等待軸完成停止。

參數： **cardNo:** 指定卡號
axis: 指定軸號碼 (參考表 2-1)

回應： **YES** 完成
NO 未完

範例： **BYTE cardNo=1; //設定第 1 號卡。**
i8092MF_SET_MAX_V(cardNo, AXIS_XY, 20000); //設定 XY 軸最高速 20K
PPS。
i8092MF_NORMAL_SPEED(cardNo, AXIS_XY, 0); //設定 XY 軸為對稱 T 曲線。
i8092MF_SET_V(cardNo, AXIS_XY, 2000); //設定 XY 軸速度=2000 PPS。
i8092MF_SET_A(cardNo, AXIS_XY, 1000); //設定 XY 軸加速度=1000 PPS/S。
i8092MF_SET_SV(cardNo, AXIS_XY, 2000); //設定 XY 軸初始速度=2000
PPS。
i8092MF_SET_AO(cardNo, AXIS_XY, 9); //設定 XY 軸減速(保留脈波數)= 9
PPS。
i8092MF_FIXED_MOVE(cardNo, AXIS_XY, 10000); //XY 軸移動 10000
Pulse。

if (i8092MF_STOP_WAIT(cardNo, AXIS_X) == NO)
{
//第 cardNo 卡 X 軸運動尚未停止，處理程序。
}

6.5.4 設定軸停止

- **void i8092MF_STOP_SLOWLY**(BYTE *cardNo*, WORD *axis*)

功能： 指定軸之輸出減速停止。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼 (參考表 2-1)

回應： 無

範例： `i8092MF_STOP_SLOWLY(1, AXIS_XY);`
//設定第 1 卡 XY 軸，減速停止。

- **void i8092MF_STOP_SUDDENLY**(BYTE *cardNo*, WORD *axis*)

功能： 指定軸之輸出立即(緊急)停止。

參數： *cardNo*: 指定卡號
axis: 指定軸號碼 (參考表 2-1)

回應： 無

範例： `i8092MF_STOP_SUDDENLY(1, AXIS_XY);`
//設定第 1 卡 XY 軸，立即(緊急)停止。

- **void i8092MF_VSTOP_SLOWLY**(BYTE *cardNo*)

功能： 指定補間軸之輸出減速停止。

參數： *cardNo*: 指定卡號

回應： 無

範例： `i8092MF_VSTOP_SLOWLY(1);`
//設定第 1 卡補間軸，減速停止。

- **void i8092MF_VSTOP_SUDDENLY**(BYTE *cardNo*)

功能： 指定補間軸之輸出立即(緊急)停止。

參數： *cardNo*: 指定卡號

回應： 無

範例： `i8092MF_VSTOP_SUDDENLY(1);`
//設定第 1 卡補間軸，立即(緊急)停止。

6.5.5 清除停止狀態

- **void** i8092MF_CLEAR_STOP(BYTE cardNo)

功能： 軸運動發生錯誤時，或使用(i8092MF_STOP_SLOWLY，
i8092MF_STOP_SUDDENLY)，使軸運動停止，請故障排除後，清除錯誤狀態。

參數： **cardNo:** 指定卡號

回應： 無

範例： i8092MF_CLEAR_STOP(1);
 //清除第 1 卡錯誤狀態。

6.5.6 補間動作結束(單軸運動或改變座標系)

- **void** i8092MF_INTP_END(BYTE cardNo)

功能： 補間動作結束。

參數： **cardNo:** 指定卡號

回應： 無

範例： i8092MF_INTP_END(1);
 //設定第 1 卡補間，動作結束。

附錄 A (i-8092 Base Function)

A.1 i-8092 運動控制命令集

Table0-1 I8092 運動控制命令分類表

函式分類	說明
基本暫存器處理函式	這些函式包括設定命令暫存器 (WR0)、模式暫存器 (WR1~WR3)、輸出暫存器 (WR4)、補間模式暫存器 (WR5)、狀態讀取暫存器 (RR0~RR5)。
初始設定函式	這些函式能設定系統註冊的初始狀態，設定卡名、設定輸出脈波模式、設定硬體極限的信號準位，和是否使用軟體極限的設定。
基本運動命令函式	這些函式包括設定 (T/S)-曲線、加減速(同步/非同步)，4 軸中任一軸的運動模式設定。
補間函式	這些函式包括 2 軸直線補間，2 軸圓或圓弧補間，2 軸的位元補間。
原點返回函式	這些函式提供自動尋找原點的函式呼叫，硬體信號的設定，外部模式函式的設定。
中斷控制函式	這些函式能偵 MCX312 運動控制晶片的中斷，並且提供符合(ISR)的運動例行中斷服務。
軸 I/O 信號函式	這些函式處理包括 Alarm、Servo Ready、外部信號輸入和 Servo Enable 的輸出信號。
讀取運動參數函式	這些函式包括設定或取得，邏輯位置或編碼器位置的計數值，和現在的速度或加速度值。

A.2 脈波輸出命令

i-8092 脈波輸出有兩個模式：一是固定脈波輸出，為了從一個位置到另一個位置，二是連續脈波輸出，能提供連續運動的速度命令。下面有各種模式可供選擇，差動信號的硬體介面，可由 **Jumper** 去選擇。

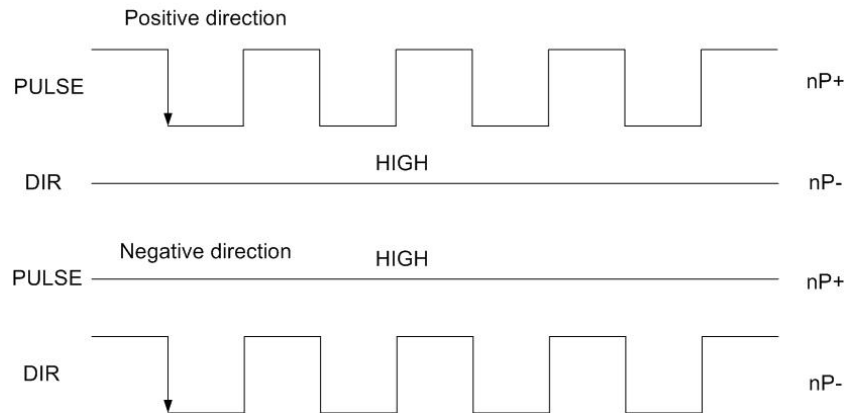


Fig.0-1 CW/CCW 輸入模式 1

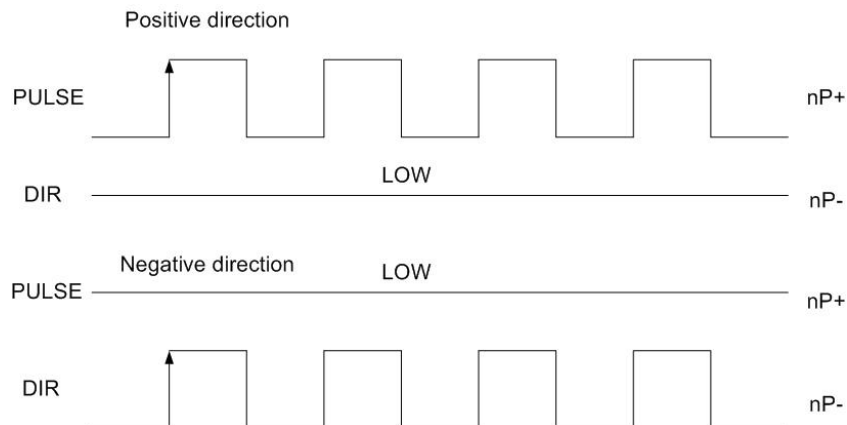


Fig.0-2 CW/CCW 輸入模式 2

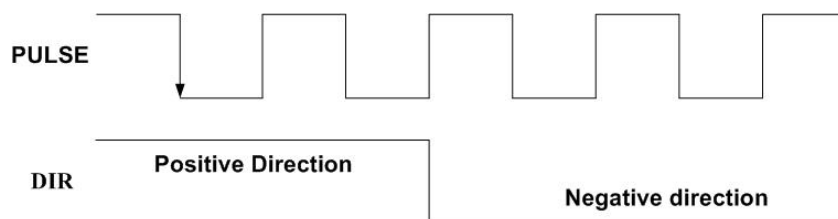


Fig.0-3 Pulse / Direction 輸入模式 1

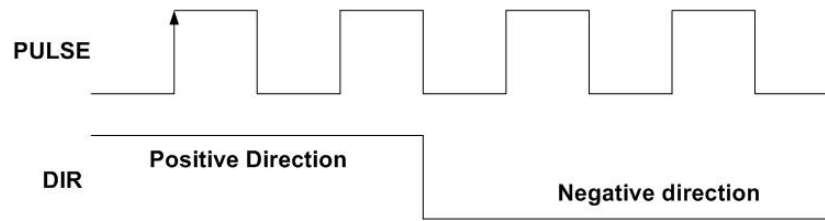


Fig.0-4 Pulse / Direction 輸入模式 2

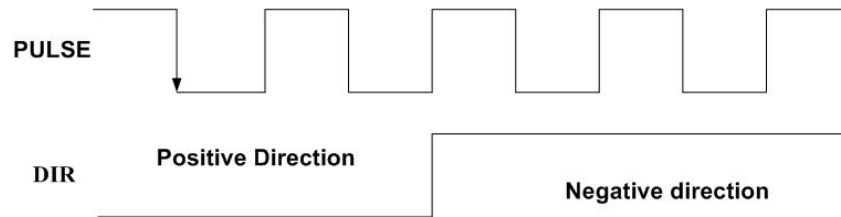


Fig.0-5 Pulse / Direction 輸入模式 3

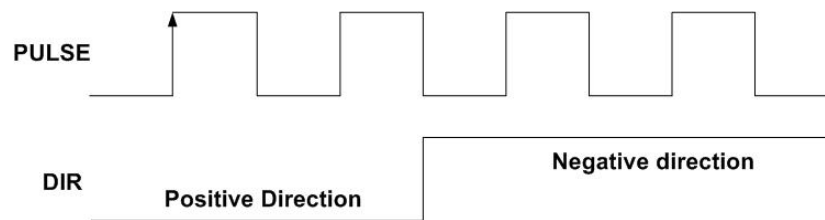


Fig.0-6 Pulse / Direction 輸入模式 4

● 固定脈波驅動

當主控端對i-8092下固定脈波驅動數，並且設定加減速。當輸出脈波和命令脈波數相等時i-8092就會停止脈波輸出。當固定脈波輸出虛設加減速時，必須在啟動前先設定一些參數：

- 範圍: R
 - 啟動速度: SV (PPS)
 - 驅動速度: V (PPS)
 - 加速: A (PPS/Sec)
 - 減速: D (PPS/Sec)
 - 輸出脈波數: P
- 更多的資訊請參考範例程式

● 在運轉時改變輸出脈波數

固定脈波輸出數可以在運轉時改變，如果新的命令增加輸出脈波數，可以參考Fig. 0-8 或 0-9，當新的命令減少輸出脈波數，如果沒有剩餘脈波數，通常他停止會取決於新的命令，可以參考Fig.0-10。而且如果是S-曲線加減速驅動模式，輸出脈波數的改變，將有可能會發生S-曲線減速不完全的情況。

● 加減速驅動的剩餘脈波數設定

於固定脈波數運動控制時，至目標前保留低速輸出 Offset Pulse 數，如圖 Fig.0-11 所示 Offset Pulse 位置。

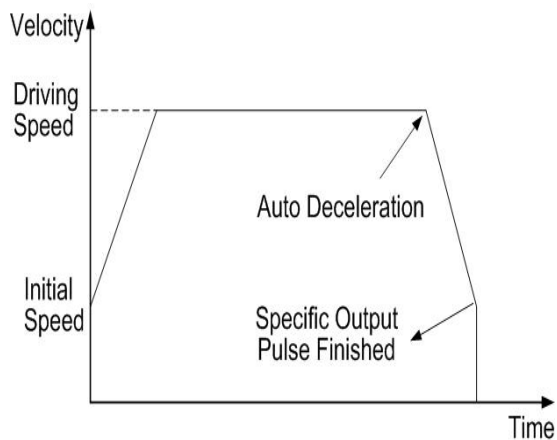


Fig.0-7 固定脈波驅動

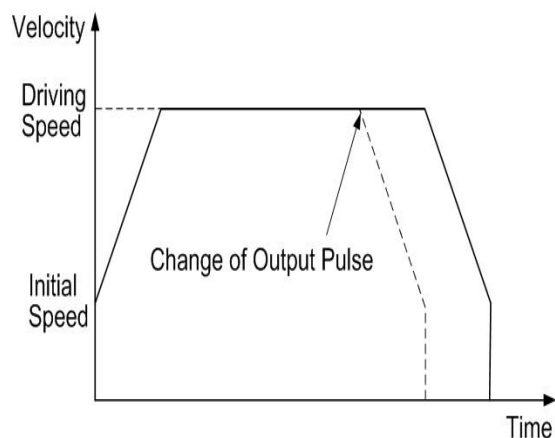


Fig.0-8 在運轉時改變輸出脈波數

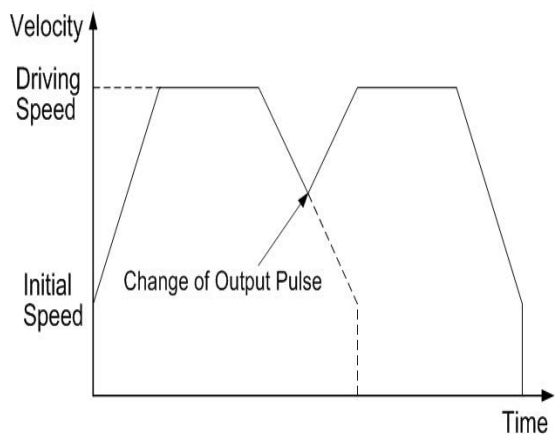


Fig.0-9 在減速期間改變命令

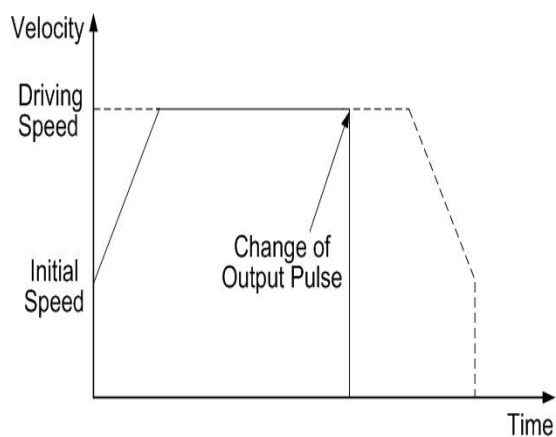


Fig.0-10 改變比輸出脈波數要少的脈波數

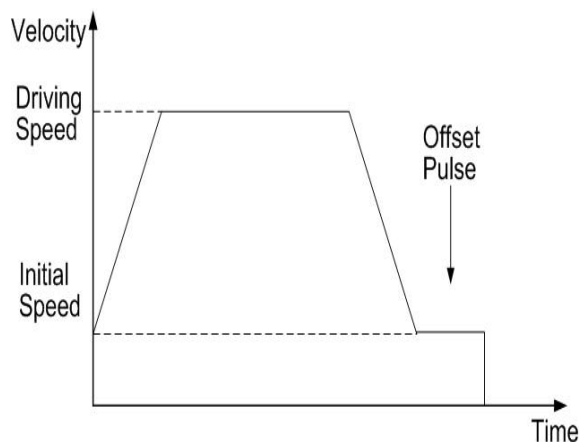


Fig.0-11 固定脈波驅動的剩餘脈波數

A.2.1 連續脈波輸出驅動

當連續驅動的執行，將以定義的速度輸出脈波，一直到停止命令或外部停止信號發生。典型的應用是：在歸原點的教導並且以定速控制，有兩個命令可以去停止連續驅動，一個是減速停止，另一個是緊急停止。每一軸有四個輸入點ST0、ST1、ST2和IN3，可以連接外部數位輸入信號，能控制減速停止或緊急停止。每一個信號準位和模式都能由函式去設定，例如ST0是定義近原點(NHOME)感測器輸入，ST1是定義原點(HOME)感測器輸入，ST2是定義編碼器Z相輸入，IN3是給使用者自定義的輸入點，他們都必須在啟動前先設定一些參數：

- 範圍: R
- 啟動速度: SV (PPS)
- 驅動速度: V (PPS)
- 加速: A (PPS/Sec)

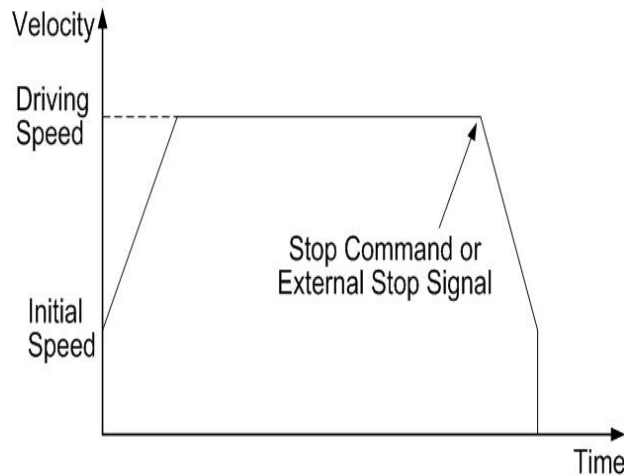


Fig.0-12 連續驅動

A.2.2 定速驅動

當i-8092設定的驅動速度命令比啟動速度低時，加減速將不會執行，而以定速驅動啟動，如果使用者當碰到原點或編碼器Z相信號而緊急停止，必需命令加減速驅動，最好的解決方法就是從開始時以低速定速驅動，在執行前必須先設定一些參數：

- 範圍: R
- 啟動速度: SV (PPS)
- 驅動速度: V (必需設定一個有效的SV和V的值)
- 輸出脈波數: P (僅適用於固定脈波驅動)

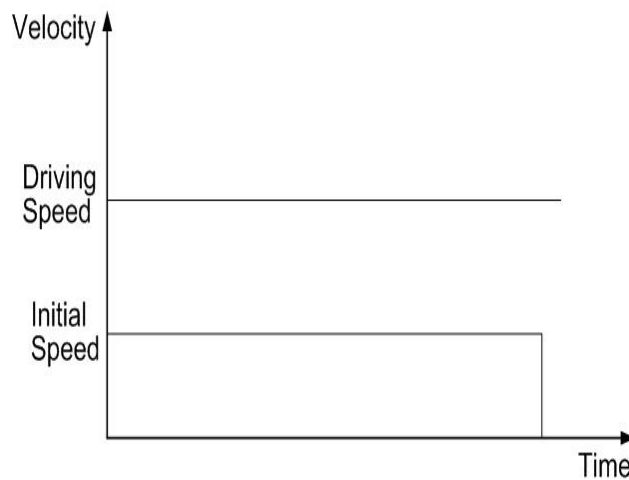


Fig.0-13 定速驅動

A.3 加減速曲線的設定

根據不同的運動控制方式，就有不同的加減速設定，使馬達驅動平滑並且減少定位錯誤的發生。在i-8092可以設定T-曲線或S-曲線加減速。

A.3.1 T-曲線加減速驅動 [對稱]

● T-曲線說明 Description

這個直線加減速驅動也能使用 T 形驅動，相關參數說明：位移總數 S、啟動速度 SV、驅動速度 V、加速度 A。

加速方程式：

$$V = SV + A \times TA \quad (1-1)$$

定速度結束的時間：

$$TM = \frac{S}{V} \quad (1-2)$$

定速度開始的時間：

$$A = \frac{V - SV}{TA} \quad (1-3)$$

T-曲線加減速可以參考 Fig.0-14.

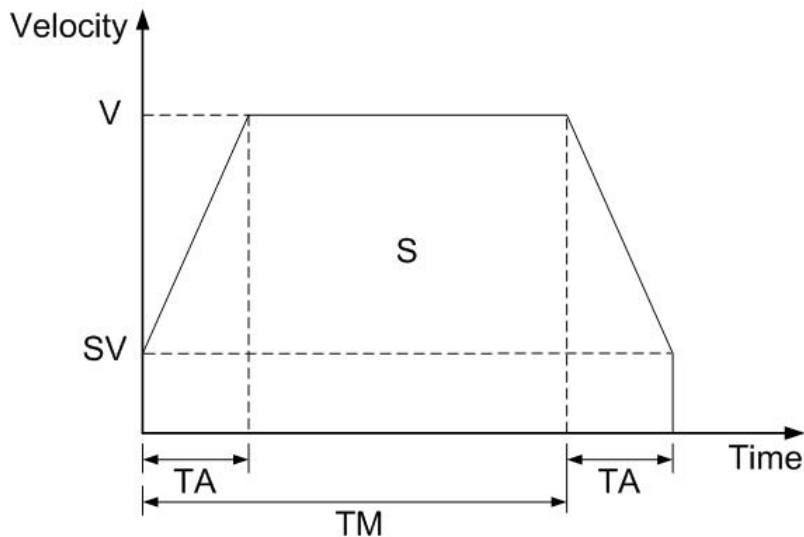


Fig.0-14 對稱 T-曲線加減速

T形驅動從起始速度啟動，並且加速到指定速度，在這個區段脈波增加的期間將被計數，並且自動減速到剩餘脈波數後，絕不再增加脈波。他們都必須在啟動前先設定一些參數：

- 範圍: R
- 啟動速度: SV (PPS)
- 驅動速度: V (PPS)
- 加速: A (PPS/Sec)
- 輸出脈波數: P

A.3.2 T-曲線加減速驅動 [非對稱]

i-8092 在固定脈波非對稱直線加速驅動，執行自動減速，在加速和減速那裡是不同的。他不用事先計算減速點， Fig.0-15是加速度大於減速度，而Fig.0-16是減速度大於加速度。在非對稱的直線加速度亦是如此，這減速起始點是經過晶片計算的。

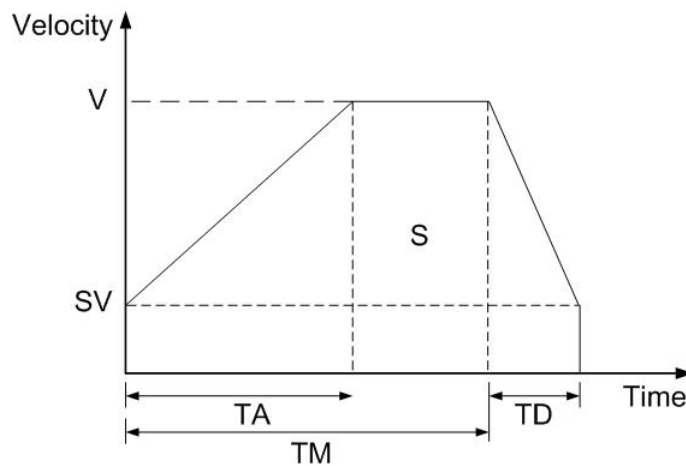


Fig.0-15 非對稱 T-曲線加減速 (A<D)

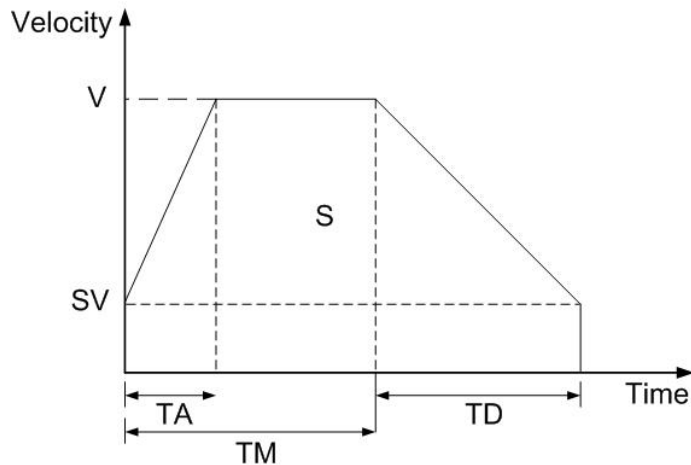


Fig.0-16 非對稱 T-曲線加減速 (A>D)

當執行非對稱T形驅動，必須先設定一些進階的參數：

- 範圍: R
- 啟動速度: SV (PPS)
- 驅動速度: V (PPS)
- 加速: A (PPS/Sec)
- 減速: D (PPS/Sec)
- 輸出脈波數: P

Note:

- 決定加減速的定義如下。 $D > A \times \frac{V}{4 \times 10^6}$ ，CLK=16 MHz，引證如果驅動速度 V = 100kps，減速度 D 必需比加速 A 的 1/40 還要大。

在這個 A>D 的例子，大者緩慢增加脈波時(當 $A/D = 10$ 時，大約最大 10 pulse)，如何預防這個問題的發生，可以增加初始速度，或設定一個負值補償加速度。

- 非對稱加減速曲線只有 T-Curve 加減速，i-8092 不支援非對稱 S-Curve 加減速

A.3.3 三角形速度曲線的預防

在直線加速固定脈波驅動時，如果輸出脈波數過低，當在輸出脈波加速期間，加減速利用的脈波數超過 1/2 總脈波數，i-8092 就會停止加速，並且進入定速模式。這個預防三角形速度的函式，在復歸時是無效的，必需先設定 **Command (60h)** 為外部模式，再將 **WR6/D3 (AVTRI)** 位元設為 1，就能使這個功能有效。

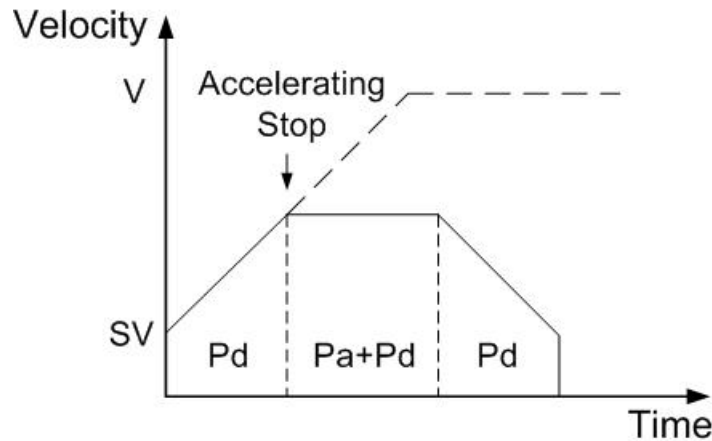


Fig.0-17 定脈波驅動三角形速度曲線的預防

$$P=2 \times (Pa+Pd)$$

P: 輸出脈波數

Pa: 加速使用的脈波數

Pd: 減速使用的脈波數

A.3.4 S-曲線加減速驅動 [對稱]

- 完整的 S-曲線

完整的 S-曲線由兩個拋物形速度曲線組成，在 Fig. 0-18 TA 是加速時間。

(1)速度方程式部份

$$V(t) = Ct^2, \quad t < TA/2 \quad (1-4)$$

(2)速度方程式部份

$$V(t) = V - C(TA - t)^2, \quad t > TA/2 \quad (1-5)$$

條件的分野如下

$$V(0) = 0, \quad V(TA/2) = V/2, \quad V'(0) = 0 \quad (1-6)$$

我們能找到這些方程式

$$C = \frac{2V}{(TA)^2} \quad (1-7)$$

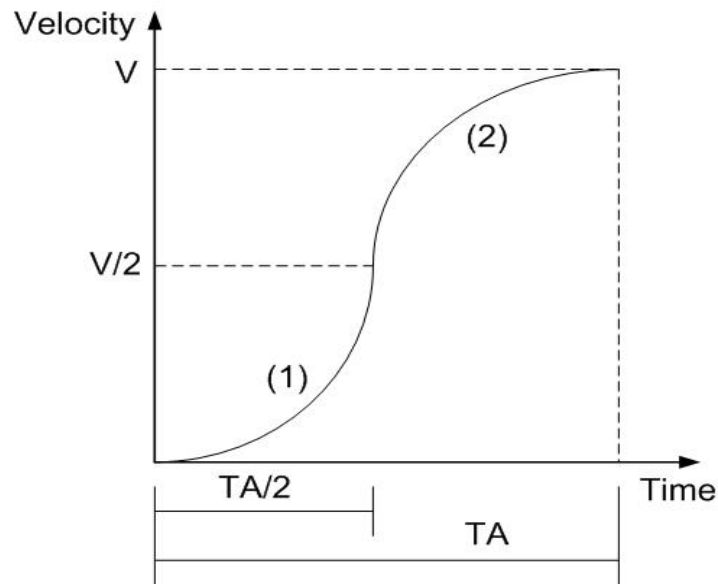


Fig.0-18 完整的 S-曲線加減速

● 局部的S-曲線

局部的 S-曲線由三個部份組成((1), (2), (3) in Fig. 0-19): 一個直線和兩個 S-曲線，曲線(1) 和(3)部份是 S-曲線加速，(2)是直線加速是直線加速部份，整個運動的時間定義為 TA，而直線加速的時間是 $TA - (2 \times TS)$ 。

(1)速度方程式部份

$$V(t) = C_1 t^2, \quad t < TS \quad (1-8)$$

(2)速度方程式部份

$$V(t) = C_2 t, \quad TS < t < TA - TS \quad (1-9)$$

(3)速度方程式部份

$$V(t) = V - C_1(TA - t)^2, \quad TA - TS < t < TA \quad (1-10)$$

C_2 是定值、是直線斜坡段

$$C_2 = \frac{V - 2VS}{TA - 2TS} \quad (1-11)$$

決定(1)、(2)連接的分野

$$V'(TS) = C_2 \quad (1-12)$$

因此

$$C_1 = \frac{V}{2[TS^2 + (TS \times (TA - 2TS))]} \quad (1-13)$$

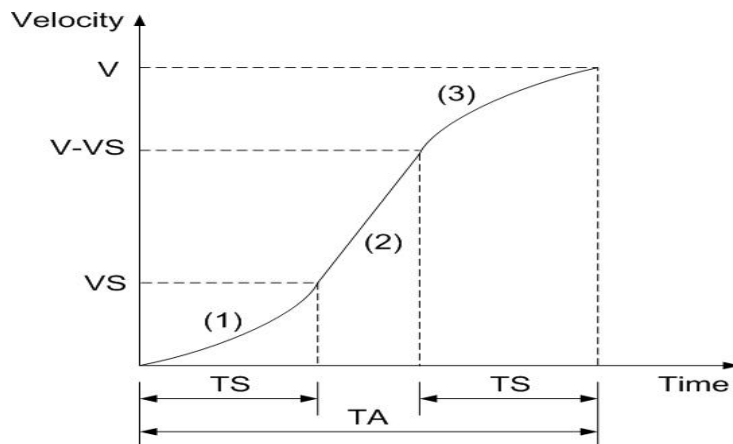


Fig.0-19 局部的 S-曲線加減速

S-曲線加減速驅動，這加速波形不是線性，這加減速波形是不規則的，如Fig. 0-20 所示。在加速有三個區段各有不同的加速值，在開始時加速度以線性地增加，從0 到定義的A(加速度值)，有一個特定的加速度率值K。並且在驅動速度增加，有一個

拋物線段的區域"a"，在第二個區段"b"，驅動速度的增加是固定的加速度。在區段"c"，加速度線性減少到0，也有一個加速度率值K。所以S-曲線加速包括a、b、c三個區域。而減速度的驅動速度改變也類似如此，可以觀察e、f、g三個區域。

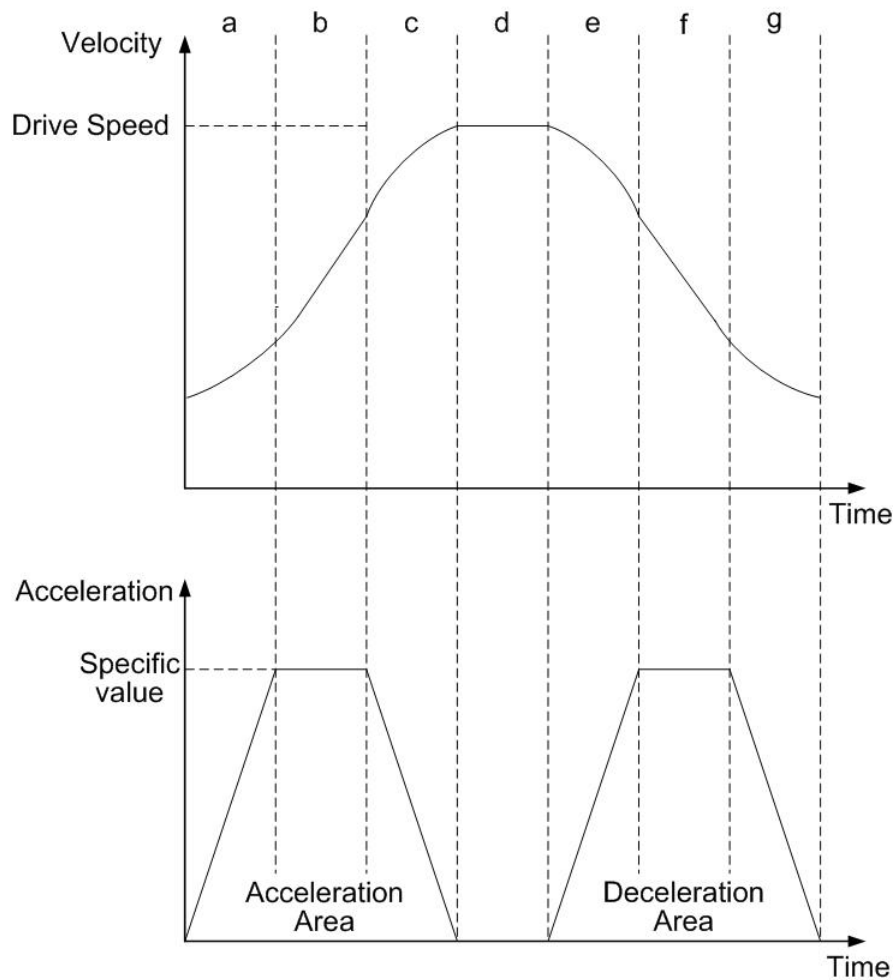


Fig.0-20 S-曲線加減速驅動

S-曲線對稱加減速固定脈波驅動，主要應用於平滑速度曲線，當脈波輸出還沒加速到驅動速度時使用，或者應用在加速期間減速停止。

如果起始速度是0，加速度是a，在t時間加速度區段的速度如下。

$$V(t) = at^2 \quad (1-14)$$

因此總脈波輸出數p(t)，從時間 0 到 t，速度是一致的

$$p(t) = \int V(t) dt = \frac{1}{3}at^3 \quad (1-15)$$

總脈波數是

$$(1/3 + 2/3 + 1 + 2/3 + 1 + 1/3) \times at^3 = 4at^3 \quad (1-16)$$

從(1-15)、(1-16)方程式，當輸出脈波在S-曲線加速段，比輸出脈波總數的1/12要多時，他將停止加速度的增加，並且開始減少加速度值。

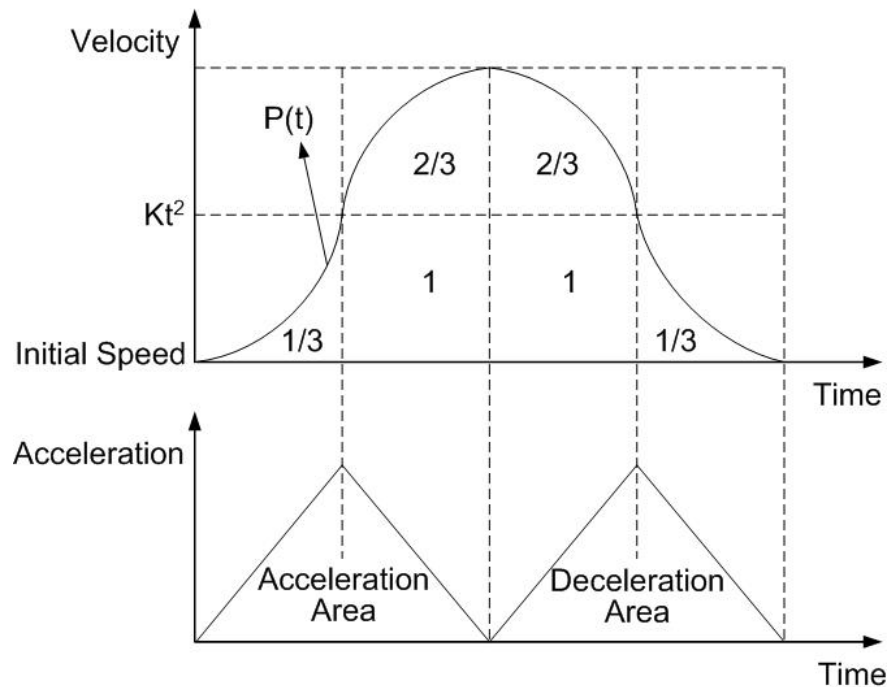


Fig.0-21 拋物線加減速的 1/12 規則

A.4 二軸補間運動

根據不同的運動控制方式，就有不同的加減速設定，使馬達驅動平滑並且減少定位錯誤的發生。在i-8092可以設定T-曲線或S-曲線加減速。

A.4.1 二軸直線補間

二軸直線補間去依據目前的位置去設置輸出脈波數。直線補間命令如Fig.0-23所示。為了個別軸的控制，命令脈波數是不帶符號的，他是被"+ direction"或"-direction"命令所控制。這個命令脈波數他的線性精度誤差 ± 0.5 LSB，我們定義長距離的移動補間為"長軸"，兩軸的短軸驅動脈波定義和長軸是相關聯的，每一軸都有24位元的計數器，範圍從 -2^{23} 到 $+2^{23}$ 。

當執行直線補間，需預先設定如下參數：

- 範圍: R
- 啟動速度: SV (PPS)
- 驅動速度: V (PPS)
- 加速度: A (PPS/Sec) (T-曲線和S-曲線加速度模式需要)
- 加速度率: K (PPS/ Sec²) (S-曲線加速度模式需要)
- 手動減速點: DP (S-曲線加速度模式需要)
- 結束位置: FP

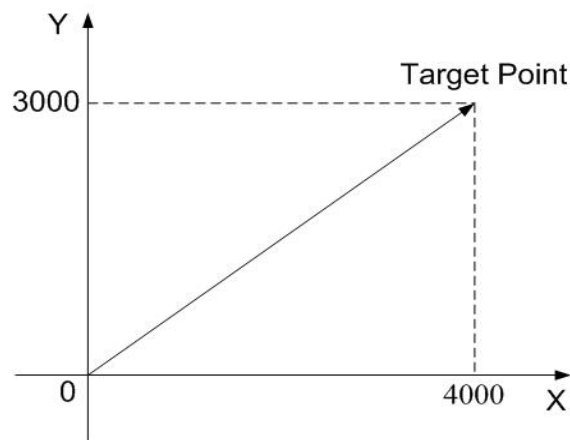


Fig.0-22 二軸直線補間

A.4.2 圓弧補間

圓形補間從現在位置(起點)開始，之後設定圓心，最後設定結束點並且定義正轉(CW)或反轉(CCW)。使用者能參考Note如何開始圓形補間：這設定的值是參考起點的相對位置，在Fig.0-24他解釋CW和CCW的定義，CW圓形補間是從起點到終點位置以順時針方向，而CCW是以反時針方向。在圓形補間他假定一開始的起點為(0, 0)，之後設定圓心，半徑就能確定，當終點也設(0, 0)，那麼個完整的圓將產生。當執行圓形補間，需預先設定如下參數：

- 範圍: R
- 啟動速度: SV (PPS)
- 驅動速度: V (PPS)
- 加速度: A (PPS/Sec) (T-曲線模式需要)
- 手動減速點: DP (T-曲線加速度模式需要)
- 結束位置: FP
- 圓心位置: C

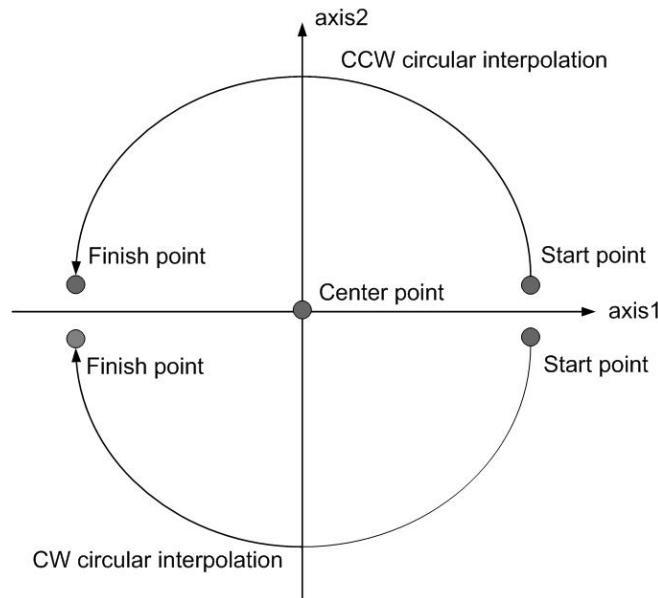


Fig.0-23 圓弧補間

在 Fig. 0.25，他解釋長軸和短軸，首先我們在 X-Y 平面上定義了八個 90° 的圓弧，並且編號為 0~7，我們發現在 0、3、4、7 的圓弧，ax1 的絕對值總是比 ax2 的值大，所以我們稱 ax1 是長軸(ax2 是短軸)，而在 1、2、5、6 的圓弧，ax2 是長軸(ax1 是短軸)。短軸將有規律的輸出脈波，並且長軸將依據補間計算的結果輸出脈波。

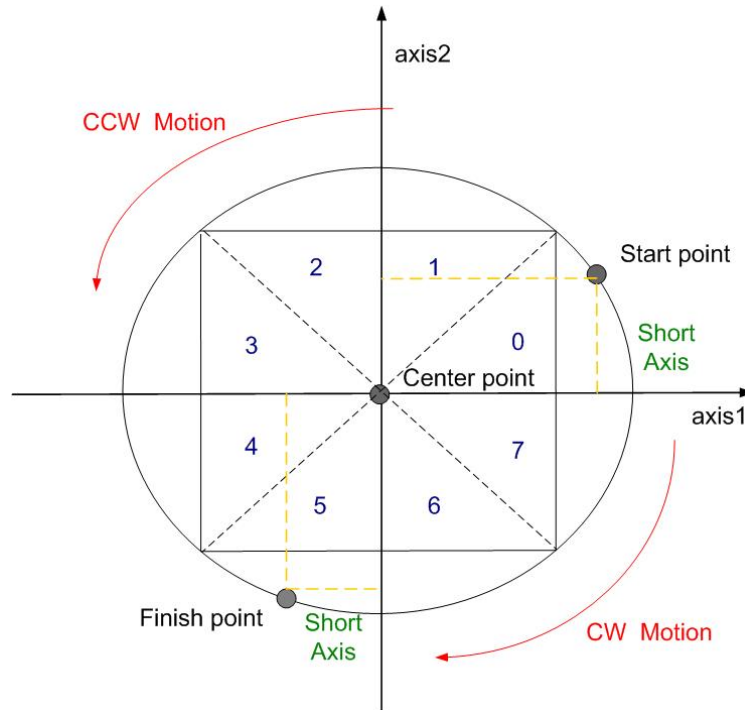


Fig.0-24 在圓弧補間的計算分為 0~7，八個 90° 的圓弧

Note: 圓弧補間在T-曲線加減速的手動減速距離

1. 首先判斷起點和終點位於哪個區域(上圖0~7)

2. 計算在這個區域起點和終點的脈波數

(注意有不同的運動方向(CW/CCW))

3. 計算通過的圓弧區段的全部數量

4. 總脈波數=短軸長度×90°圓弧的號碼

5. 如果倍率=M、並且真正的起始速度: $RSV = SV \times M$ 、真正的驅動速度: $RV = V \times M$ 、真正的加速度: $RA = A \times 125 \times M$

公式為: $RV = RSV + RA \times TA$ ，並且你能求得加速時間 TA 和加速區域的脈波數量。

6. 如果這脈波數是在減速段，他和加速段的計算方法是一樣簡單的:

手動減速點=總脈波數-減速期間輸出的脈波

Fig.0.26是逆時針補間的範例，起點(0, 0)、圓心(-200, 500)、終點為(-702, 299)，終點在第4弧段，ax2是短軸，所以補間結束點在ax2軸是299.

- Note:**
1. 不使用定速驅動
 2. 在圓弧補間使用手動減速，T-曲線驅動能自動減速而S-曲線驅動不能

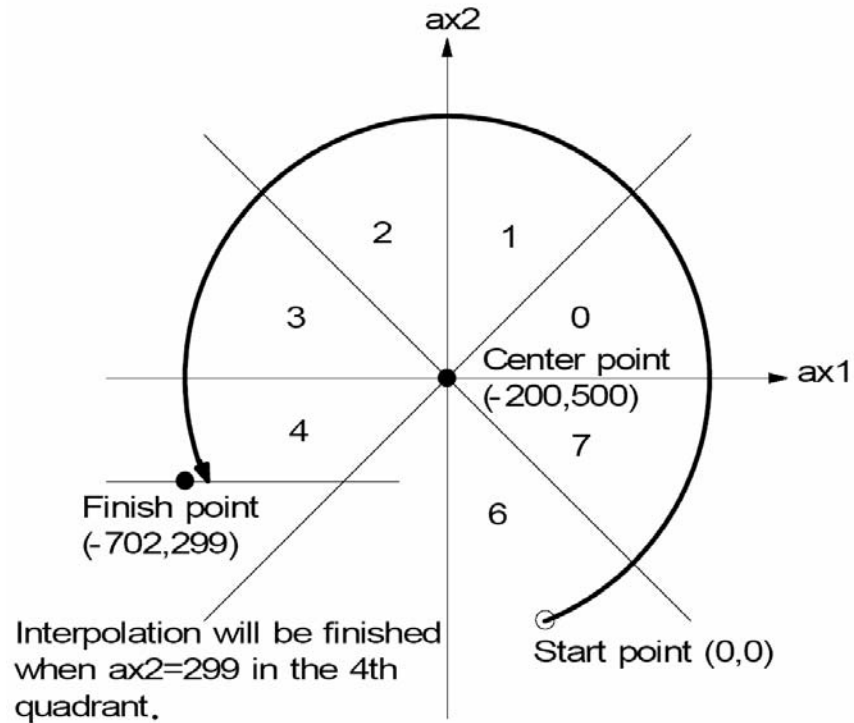


Fig.0-25 計算圓弧補間的手動減速點

A.4.3 位元補間

位元補間驅動所接收的補間資料，是來自上層CPU所傳送的一個確定大小區塊的位元圖案，並以指定的驅動速度連續輸出補間脈波。每一軸有兩個位元的緩衝器給主CPU：一個是正方向，另一個是負方向。當執行位元圖案補間，主CPU將寫入指定的補間資料到i-8092的2或3軸，如果一個從CPU傳過來的位元圖案資料是“1”，i-8092將在這單位時間輸出一個脈波，如果是“0”，i-8092將在這單位時間不輸出任何脈波。如下範例，使用者如要產生X-Y輪廓(參看Fig.0-28)，主CPU必需寫入設定的圖案到那些特定的暫存器 → XPP: 這是X軸正方向的暫存器，XPM: 這是X軸負方向的暫存器，YPP和YPM: 這是Y軸正和負方向的暫存器。在這段時間，i-8092將檢查一次暫存器，並且不依靠位元補間而自決輸出脈波。

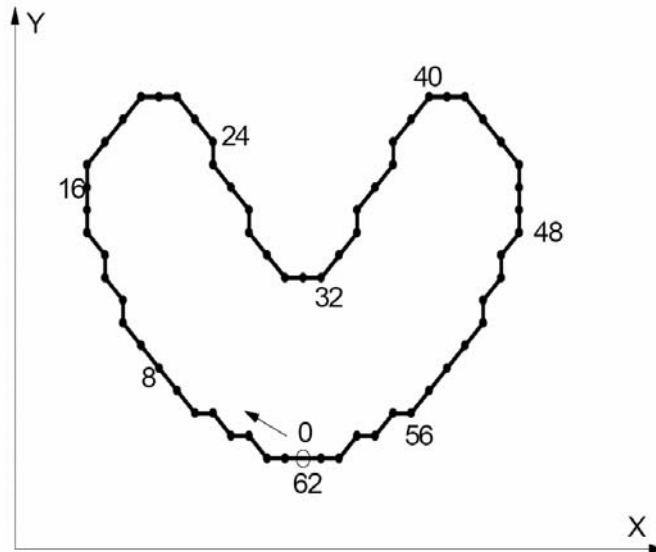


Fig.0-267 X-Y 輪廓圖

```

← 56 ← 48 ← 40 ← 32 ← 24 ← 16 ← 8 ← 0
01000000 00000000 00011111 11011011 11110110 11111110 00000000 00000000 :XPP(X+direction)
01111111 11110101 00000000 00000000 00000000 00000000 00101011 11111111 :XPM(X-direction)
00000000 00000000 00000000 11111111 00000000 00001111 11111111 11010100 :YPP(Y+direction)
00001010 11111111 11111100 00000000 00111111 11000000 00000000 00000000 :YPM(Y-direction)
```

Fig.0-27 X-Y 輪廓的位元圖案資料

堆疊計數器(SC)是一個兩位元的計數器，他的值介於0~3，能從RR0暫存器的D14、D13讀到他們，SC將決定哪個暫存器從主CPU接收資料，SC的初始值是0，所以當主CPU寫入位元圖案資料到BP1P或BP1M，這些資料將被存在SREG，並且SC將向上計數到1，而下一個資料會從主CPU寫到REG1。順便一提，當SC=2變成REG2這個暫存器，當SC=3主CPU將無法寫入任何位元圖案資料到MCX312。

位元補間脈波正在輸出時，D0在SREG(堆疊暫存器)將第一個位移輸出，然後依序為D1、D.....，當所有的SREG(堆疊暫存器)已經位移輸出後，在REG1的資料將被移到SREG，而在REG2的資料將被移到REG1，並且SC將向下計數到2，然後主CPU就能一直把新的資料寫到i-8092。

依序使i-8092持續不斷的輸出位元資料，這個主CPU應該在SC下數到0之前，把資料寫入到i-8092。當SC計數從2到1時，i-8092將輸出中斷信號到主CPU。

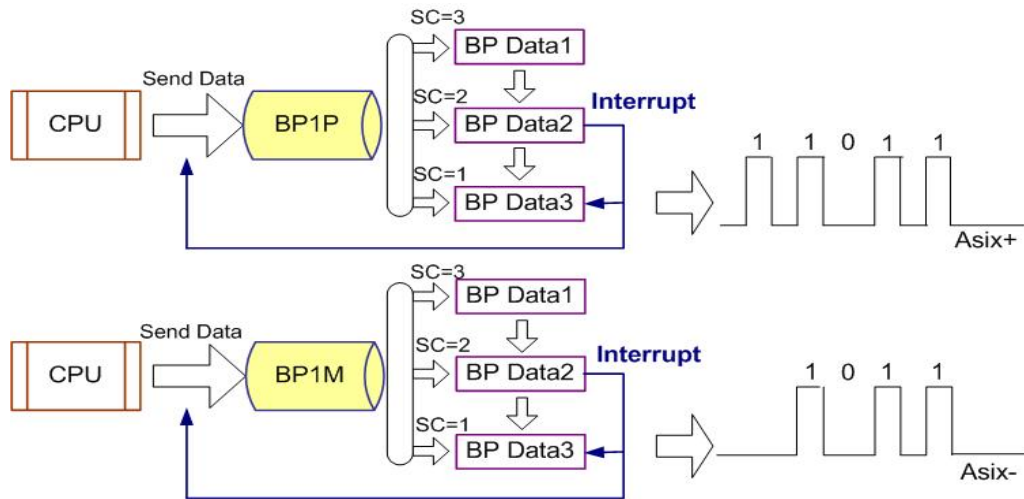


Fig.0-28 位元圖案資料堆疊

■ 位元圖案補間驅動速度的限制

位元補間模式最大輸出速度是4MHz，然而這個最大速度將取決於主CPU資料的更新率，比如位元資料是多於48bits。如範例X和Y軸的位元圖案補間，如果主CPU需要100uSec去更新X和Y軸的16-bit資料，那麼最大速將是：
 $16/100\mu\text{Sec}=160\text{KPPS}$ 。

■ 位元補間的結束

有兩個方法能夠終結位元補間：

(1) 寫入結束碼到ax1的暫存緩衝器，這個位元圖案補間模式將被結束並停止。如果主CPU

寫“1”到正和負方向的暫存緩衝器，當結束碼被執行時，SC將自動地變成0。

(2) 主CPU停止寫入任何命令到I8092，而SC=0並且沒有任何資料在更新，I8092將停

止輸出脈波，然後位元補間將結束。

■ 利用補間的停止和暫停命令

如果有緊急停止或減速停止命令被寫入主軸ax1，這補間驅動將被暫停。如果主CPU再一次致能位元補間，i-8092將繼續補間動作。如果主CPU想要在寫入停止命令後停止補間，必需清除所有正在使用的BP暫存器的補間位元資料。

A.4.4 連續補間

這個連續補間是執行一連串的補間程序，像是直線補間+圓形補間+直線補間+.....，在連續補間期間驅動將不會停止，脈波將會持續不段的輸出。當執行連續補間時，再先前補間命令結束前，主CPU將寫入下一個補間命令到i-8092。

■ 輪詢

這個輪詢的方法是呼叫i8092_NEXT_WAIT函式去檢查RR0暫存器的D9位元，如果D9=1，i-8092將接受下一個補間命令。所以連續補間的標準程序，是寫入並致能補間資料和命令，然後檢查RR0暫存器的D9位元是1或0，然後重複寫命令和檢查D9。如下系統流程圖：

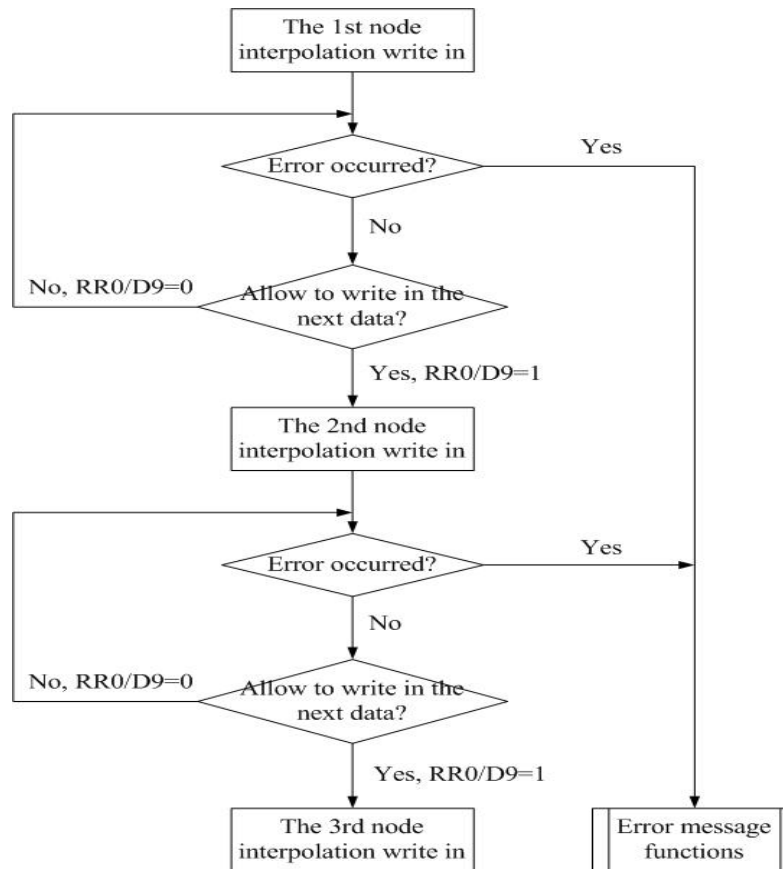


Fig.0-29 連續補間的輪詢方法

■ 中斷

使用i-8092函式庫中斷遮罩函式，在連續補間期間去致能或除能中斷。在連續補間運動時，使用者能設定不同的中斷因子，配合各種補間運動，例如混合速度控制、位元補間，都能很容易的使用。請參考A.6章節。

A.5 原點返回運動

原點返回常常使用，當機器開機時，或系統發生警報，或信號錯誤時。上述情況使用者都能使用歸原點，讓機器回到原先的工作點。

i-8092 提供了 8 種原點返回的功能，例如高速尋找近原點→低速歸原點→編碼器 Z-相尋找，使用者處理的狀況應該和下圖所示類似。這個範例是單軸驅動系統，兩軸軸也能以相同方法處理。

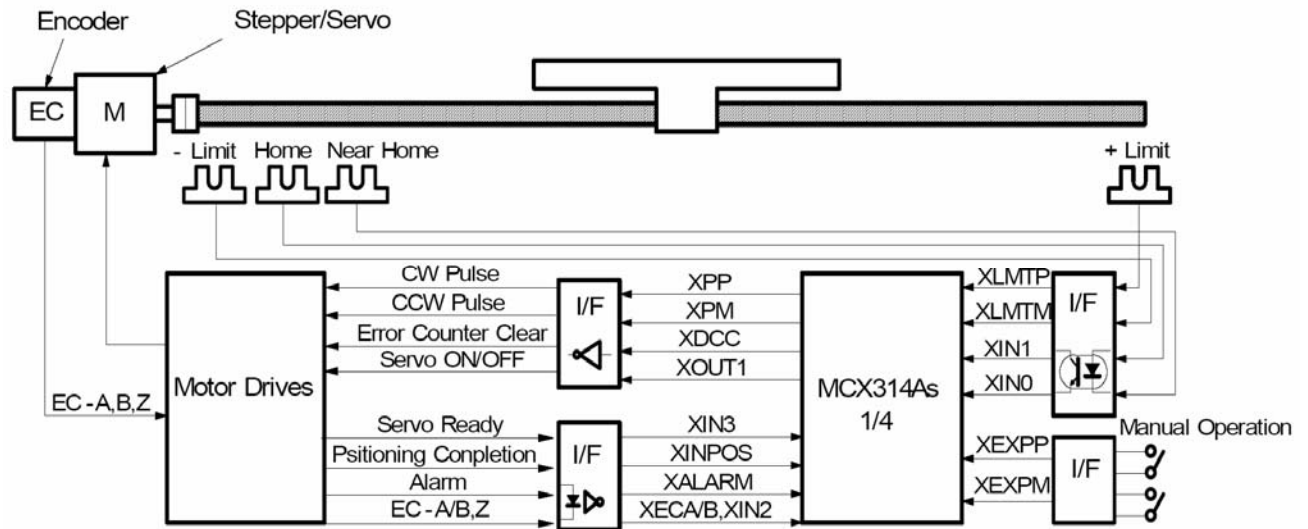


Fig.0-30 X-軸硬體信號狀況

A.6 中斷控制

X、Y軸中斷的產生，有位元補間或連續補間。每一個中斷都能被單獨地致能或除能，在電源復歸期間，所有中斷信號是被除能。

A.6.1 各軸中斷

下表將展示中斷產生的要素和中斷遮罩呼叫的函式

呼叫函式範例 (致能 / 除能)	nRR3 Register	中斷發生的要素
i8092_INTFACTOR_ENABLE(1, 0x3, 1) i8092_INTFACTOR_DISABLE(1, 0x3, 1)	D1(P ≥ C-) (1 / 0)	邏輯或真實位置計數器的值大於或等於 COMP-暫存器的值。
i8092_INTFACTOR_ENABLE(1, 0x3, 2) i8092_INTFACTOR_DISABLE(1, 0x3, 2)	D2(P < C-) (1 / 0)	邏輯或真實位置計數器的值小於 COMP-暫存器的值。
i8092_INTFACTOR_ENABLE(1, 0x3, 3) i8092_INTFACTOR_DISABLE(1, 0x3, 3)	D3(P < C+) (1 / 0)	邏輯或真實位置計數器的值小於 COMP+暫存器的值。
i8092_INTFACTOR_ENABLE(1, 0x3, 4) i8092_INTFACTOR_DISABLE(1, 0x3, 4)	D4(P ≤ C+) (1 / 0)	邏輯或真實位置計數器的值大於或等於 COMP+暫存器的值。
i8092_INTFACTOR_ENABLE(1, 0x3, 5) i8092_INTFACTOR_DISABLE(1, 0x3, 5)	D5(C-END) (1 / 0)	在加減速驅動時，當驅動改變從定速區域進入減速區域。
i8092_INTFACTOR_ENABLE(1, 0x3, 6) i8092_INTFACTOR_DISABLE(1, 0x3, 6)	D6(C-STA) (1 / 0)	在加減速驅動時，當驅動改變從加速區域進入定速區域。
i8092_INTFACTOR_ENABLE(1, 0x3, 7) i8092_INTFACTOR_DISABLE(1, 0x3, 7)	D7(D-END) (1 / 0)	當驅動結束。

A.6.2 補間中斷

呼叫函式範例 (致能 / 除能)	中斷發生的要素
i8092_CIINT_ENABLE i8092_CIINT_DISABLE	在連續補間，當MCX312下一個節點的補間資料是可用的... ★下一個補間命令被寫入後，這個中斷將被清除。
i8092_BPINT_ENABLE i8092_BPINT_DISABLE	在位元補間，當堆疊連結器(SC)的值從2變成1，並且下一個BP命令可以被寫入堆疊... ★BP 命令被寫入堆疊之後，這個中斷將被清除。

Note: 當一個中斷發生在補間期間，經由寫入中斷清除命令(3Dh) INTN，這個中斷將被清除，並且在一次補間結束，自動回復到原先高組抗狀態。

A.7 i-8092 功能函式庫

我們使用 MCX312 的資料暫存器和特殊的命令，發展簡單且強而有力的高階程式去設定應用程式介面。這些函式庫由運動、補間命令、狀態展示和 I/O 信號組成，使編譯控制程式時更容易。最後，只要呼叫動態或靜態連結函式庫，所有的設定及運動控制程式都會容易的執行。

這個軟體函式庫的發展程序是由 Visual C++ 所寫成。我們的包裝成使用者不需設計複雜的路徑計畫，和編碼驅動程式去控制多軸運動的優點。最後，函式的能力和正確性都已經在 2 軸伺服馬達精密機械測試過。

函式格式

在 i-8092 的函式庫，幾乎所有的函式都有以下相同的格式：

i8092_FUNCTION_NAME(cardNo, axis, parameter1, parameter2)

1. cardNo i8092 模組的板號。

2. axis 如下圖 Table0-3 是指定軸編號的編碼。軸編號的指定是在 WR0 暫存器的 D8~D11 位元。當軸的位元數設為 1，則軸就被指定。這指定的動作並非限制只能對單軸作設定，而是可以多軸同時進行的。

Table0-2 軸碼/名指定

WR0 Register						
D15	D14~D12	D11	D10	D9	D8	D7~D0
RESET				Y	X	

Axis	X	Y	XY
Code	0x1	0x2	0x3
Name	AXIS_X	AXIS_Y	AXIS_XY

A.7.1 暫存器管理函式

只有對進階的使用者，他們可以靠著使用 Table0-4 的函式對特殊或進階的使用發展自己的程式。WRn、RRn 暫存器的定義在附錄中詳細說明。一般使用者可以省略這部分。

Table0-3 暫存器管理函式

函式名稱	描述
i8092_SET_COMMAND	設定 4 軸的命令暫存器(WR0)
i8092_SET_WR1	設定 4 軸的模式暫存器(WR1)
i8092_SET_WR2	設定 4 軸的模式暫存器(WR2)
i8092_SET_WR3	設定 4 軸的命令暫存器(WR3)
i8092_SET_WR4	設定輸出暫存器(WR4)
i8092_SET_WR5	設定補間暫存器(WR5)
i8092_GET_RR0	讀回主要狀態暫存器(RR0)
i8092_GET_RR1	讀回狀態暫存器 1 (RR1)
i8092_GET_RR2	讀回狀態暫存器 2 (RR2)
i8092_GET_RR3	讀回狀態暫存器 3 (RR3)
i8092_GET_RR4	讀回輸入暫存器(RR4)
i8092_GET_RR5	讀回輸入暫存器(RR5)

i8092_COMMAND

Format: **void** i8092_COMMAND(BYTE *cardNo*, WORD *axis*, WORD *cmd*)

Function: 設定 4 軸的命令暫存器(WR0)。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考 Table 0-3)。
 cmd 設定在暫存器 WR0 的命令碼。

Example: *//對 XY 軸設定切換功能*
 i8092_COMMAND(1, 0x3, 0xf);
 //對 XY 軸設定意外停止功能
 i8092_COMMAND(1, 0x3, 0x27);

i8092_SET_WR1

Format: `void i8092_SET_WR1(BYTE cardNo, WORD axis, WORD data)`

Function: 設定 2 軸的模式暫存器(WR1)。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考 Table 0-3)。
data 設定在暫存器 WR1 的 16 進制 32 位元值

Example: `//設定 X 軸的 ST0 信號致能並設為高位準致動
i8092_SET_WR1(1, 0x1, 0x0003);`

i8092_SET_WR2

Format: `void i8092_SET_WR2(BYTE cardNo, WORD axis, WORD data)`

Function: 設定 2 軸的模式暫存器(WR2)。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考 Table 0-3)。
data 設定在暫存器 WR2 的 16 進制 32 位元值

Example: `//當和真實位置比較時，設定 XY 軸的軟體極限致能
i8092_SET_WR2(1, 0x3, 0x0023);`

i8092_SET_WR3

Format: `void i8092_SET_WR3(BYTE cardNo, WORD axis, WORD data)`

Function: 設定 2 軸的模式暫存器(WR3)。

Parameters: *cardNo* 是板號。
axis 是指定軸號碼(參考 Table 0-3)。
data 設定在暫存器 WR3 的 16 進制 32 位元值

Example: `//設定 XY 軸非對稱 T 曲線模式
i8092_SET_WR3(1, 0x3, 0x0003);`

i8092_SET_WR4

Format: `void i8092_SET_WR4(BYTE cardNo, WORD data)`

Function: 設定 2 軸的模式暫存器(WR4)。

Parameters: *cardNo* 卡號。
data 設定在暫存器 WR4 的 16 進制 32 位元值

Example: `//設定 2 軸的 OUT1 信號為高準位致動
i8092_SET_WR4(1, 0x0202);`

i8092_SET_WR5

Format: `void i8092_SET_WR5(BYTE cardNo, WORD data)`

Function: 設定 2 軸的模式暫存器(WR5)。

Parameters: *cardNo* 卡號。
data 設定在暫存器 WR5 的 16 進制 32 位元值

Example: *//設定 X、Y 軸的常數向量速度模式*
`i8092_SET_WR5(1, 0x0300);`

i8092_GET_RR0

Format: `void i8092_GET_RR0(BYTE cardNo)`

Function: 讀回主要狀態暫存器(RR0)。

Parameters: *cardNo* 卡號。

Example: *//取回 1 卡的主要狀態暫存器(RR0)的值*
`i8092_GET_RR0(1);`

i8092_GET_RR1

Format: `void i8092_GET_RR1(BYTE cardNo, WORD axis)`

Function: 讀回主要狀態暫存器(RR1)。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考 Table 0-3)。

Example: `//讀回 X 軸的主要狀態暫存器(RR1)的值
i8092_GET_RR1(1, 0x1);`

i8092_GET_RR2

Format: `void i8092_GET_RR2(BYTE cardNo, WORD axis)`

Function: 讀回主要狀態暫存器(RR2)。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考 Table 0-3)。

Example: `//讀回 X 軸的主要狀態暫存器(RR2)的值
i8092_GET_RR2(1, 0x1);`

i8092_GET_RR3

Format: **void** i8092_GET_RR3(BYTE *cardNo*, WORD *axis*)

Function: 讀回主要狀態暫存器(RR3)。

Parameters: *cardNo* 卡號。
 axis 是指定軸號碼(參考 Table 0-3)。

Example: *//取回 X 軸的主要狀態暫存器(RR3)的值*
 i8092_GET_RR3(1, 0x1);

i8092_GET_RR4

Format: **void** i8092_GET_RR4(BYTE *cardNo*)

Function: 讀回主要狀態暫存器(RR4)。

Parameters: *cardNo* 卡號。

Example: *//讀回 1 卡的主要狀態暫存器(RR4)的值*
 i8092_GET_RR4(1);

i8092_GET_RR5

Format: `void i8092_GET_RR5(BYTE cardNo)`

Function: 讀回主要狀態暫存器(RR5)。

Parameters: *cardNo* 卡號。

Example: *//取回 1 卡的主要狀態暫存器(RR5)的值*
 `i8092_GET_RR5(1);`

A.7.2 函式初始設定

在 i8092.h，我們對 I8092 運動控制卡定義一些常數及結構。

板號及槽位定義

#define	CARD1	1
#define	CARD2	2
#define	MAX_SLOT_NO	8

判斷常數定義

#define	YES	1
#define	NO	0
#define	ON	1
#define	OFF	0
#define	SERVO_ON_flag	1
#define	SERVO_OFF_flag	0

動作型式定義

#define	ACCMODE	0
#define	CONST2	1
#define	CONST3	3

中斷條件因子

#define	D_END	0x8000
#define	C_STA	0x4000
#define	C_END	0x2000
#define	CP_GE	0x1000
#define	CP_L	0x0800
#define	CM_L	0x0400
#define	CM_GE	0x0200
#define	CI_INT	0x4001
#define	BP_INT	0x8001
#define	IDLE	0x0000

資料型態轉換

#define	BYTE	unsigned char
#define	WORD	unsigned short int
#define	DWORD	unsigned long int

命令暫存器

#define	wr0	0x0
#define	wr1	0x2
#define	wr2	0x4
#define	wr3	0x6
#define	wr4	0x8
#define	wr5	0xa
#define	wr6	0xc
#define	wr7	0xe

狀態暫存器

#define	rr0	0x0
#define	rr1	0x2
#define	rr2	0x4
#define	rr3	0x6
#define	rr4	0x8
#define	rr5	0xa
#define	rr6	0xc
#define	rr7	0xe

軸定義

#define	AXIS_X	0x1
#define	AXIS_Y	0x2
#define	AXIS_XY	0x3

驅動型式定義

#define	PFD	0x20
#define	NFD	0x21
#define	PCD	0x22
#define	NCD	0x23

Table0-4 函式初始設定

函式名稱	描述
i8092_PULSE_MODE	設定輸入脈衝模式。
i8092_SET_R	設定範圍。
i8092_GET_R	讀取範圍。
i8092_HLMTP_LEVEL	硬體正極限切換信號邏輯準位設定。
i8092_HLMTM_LEVEL	硬體負極限切換信號邏輯準位設定。
i8092_SLMTP_MODE	軟體正極限切換信號模式設定。
i8092_SLMTM_MODE	軟體負極限切換信號模式設定。
i8092_COMPARE_LP	這個函式可以設定 COMP+/-暫存器和邏輯位置計數器比較。
i8092_COMPARE_EP	這個函式可以設定 COMP+/-暫存器和實際位置計數器比較。
i8092_RESET_CARD	重設成電源開啟狀態。

i8092_PULSE_MODE

Format: **void** i8092_PULSE_MODE(BYTE *cardNo*, WORD *axis*, WORD *nMode*)

Function: 設定輸出脈衝模式。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考 Table 0-3)。
 nMode 如表 Table0-6

Table0-5 脈衝輸出模式表

Mode	Value	Direction	Waveform of input pulse	
			nPP / PULSE	nPM / DIR
CW / CCW	0	+	PULSE	LOW
	1	—	LOW	PULSE
PULSE / DIR	2	+	PULSE+	LOW
	3	—	PULSE+	HIGH
	4	+	PULSE-	LOW
	5	—	PULSE	HIGH

Example: //將所有選擇軸設定為 CW/CCW (Dir +)模式
 i8092_PULSE_MODE(1, 0xf, 2);

i8092_SET_R

Format: `void i8092_SET_R(BYTE cardNo, WORD axis, DWORD data)`

Function: "R"指“範圍”，是決定多種驅動速度，加減速度的參數。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考 Table 0-3)。

Example: `// 設至倍率為10倍
i8092_SET_R(1, 0xf, 800000);`

註：如果驅動速度參數的最大值設定為8000，而且驅動速度設為40KPPS。則使用者可以設定V=8000、R=1600000。因為40K是8000的5倍，所以設定R=8000000/5。

i8092_GET_R

Format: `DWORD i8092_GET_R(BYTE cardNo, WORD axis)`

Function: 從全域變數讀回範圍值。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考 Table 0-3)。

Example: `//讀回 X 軸範圍值
i8092_GET_R(1, 0x1);`

i8092_HLMTP_LEVEL i8092_HLMTM_LEVEL

Format: **void** i8092_HLMTP_LEVEL(BYTE *cardNo*, WORD *axis*, WORD
 nLevel)
 void i8092_HLMTM_LEVEL(BYTE *cardNo*, WORD *axis*, WORD
 nLevel)

Function: 設定正負硬體極限輸入信號的邏輯準位。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考 Table 0-3)。
 nLevel nLevel=0：低有效準位，nLevel =1：高有效準位。

Example: *//設定4軸的正方向硬體極限為低有效準位*
 i8092_HLMTP(1, 0xf, 0);

i8092_SLMTP_MODE i8092_SLMTM_MODE

Format: **void** i8092_SLMTP_MODE(BYTE *cardNo*, WORD *axis*, WORD
 nMode)

void i8092_SLMTM_MODE(BYTE *cardNo*, WORD *axis*, WORD
 nMode)

Function: 致能/除能正或負方向的軟體極限。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考 Table 0-3)。
 nMode nMode=0：致能，nMode =1：除能。

Example: *//致能所有軸正方向的軟體極限*
 i8092_SLMTP_MODE(1, 0x3, 0);

i8092_COMPARE_LP

Format: **void** i8092_COMPARE_LP(BYTE *cardNo*, WORD *axis*)

Function: 這個函式可以設定COMP+/-暫存器和邏輯位置計數器比較。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考Table 0-3)。

Example: *//將XY軸的比較對象設為邏輯位置*
 i8092_COMPARE_LP(1, 0x3);

i8092_COMPARE_EP

Format: `void i8092_COMPARE_EP(BYTE cardNo, WORD axis)`

Function: 這個函式可以設定COMP+/-暫存器和實際位置計數器比較。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考Table 0-3)。

Example: *//將XY軸的比較對象設為實際位置*
`i8092_COMPARE_EP(1, 0x3);`

i8092_RESET_CARD

Format: `void i8092_RESET_CARD(BYTE cardNo)`

Function: 重設成電源開啟狀態。

Parameters: *cardNo* 卡號。

Example: `i8092_RESET_CARD(1);`
//重置第1卡。

註: 當這個位元(WR0/D15)是設成1，但是其他是0，在這命令寫入後PS400將被重置。

A.7.3 位置控制函式

邏輯位置計數器是計算在MCX312的驅動脈波數。當正向脈波輸出時，計數器加1；當一個反向脈波輸出時，計數器減1。真實位置計數器將從外部編碼器計算輸入脈波數。輸入脈波的種類可以分成A/B相方波和上/下(順時針/逆時針)脈波。任何時間主機的中央處理器可以讀寫這兩種計數器。計數器是正負32位元，計算範圍在 $-2^{31} \sim +2^{31}$ 之間。

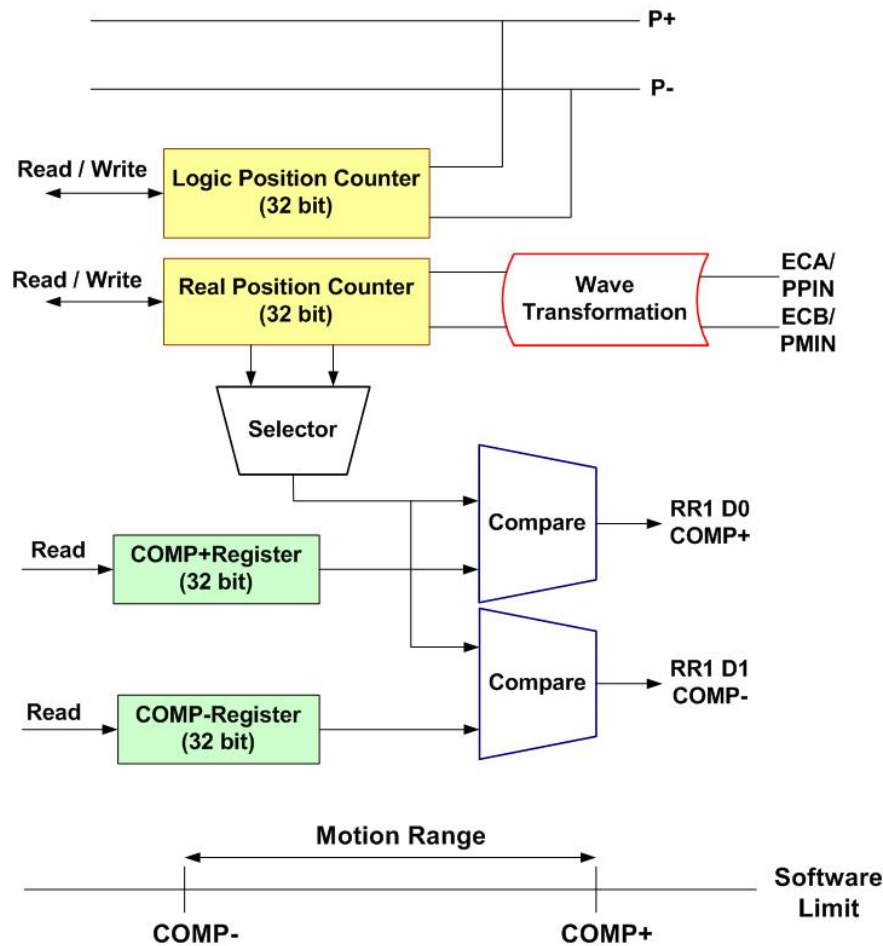


Fig.0-31 位置暫存器及軟體極限的管理

Table0-6 位置控制函數

函式名稱	描述
i8092_SET_LP	邏輯位置計數器設定
i8092_SET_EP	實際位置計數器設定
i8092_GET_LP	讀取邏輯位置計數器
i8092_GET_EP	讀取實際位置計數器
i8092_GET_CV	讀取目前驅動速度
i8092_GET_CA	讀取目前加(減)速度
i8092_SET_CP	設定正向軟體極限
i8092_SET_CM	設定反向軟體極限
i8092_VRING_ENABLE	設定位置計數器為環狀計數
i8092_VRING_DISABLE	取消位置計數器為環狀計數
i8092_AVTRI_ENABLE	致能預防三角形速度曲線的產生
i8092_AVTRI_DISABLE	除能預防三角形速度曲線的產生

i8092_SET_LP

Format: **void i8092_SET_LP**(BYTE *cardNo*, WORD *axis*, long *dwdata*)

Function: 邏輯位置計數器設定，將之設為 0 可以重置計數器值。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考 Table 0-3)。
 dwdata 邏輯位置計數器輸入值，資料範圍： $-2^{31} \sim +2^{31}$ 。

Example: //清除 XY 軸邏輯位置計數器值
 i8092_SET_LP(1, 0x1, 0);
 i8092_SET_LP(1, 0x2, 0);

i8092_SET_EP

Format: `void i8092_SET_EP(BYTE cardNo, WORD axis, long dwdata)`

Function: 實際位置計數器設定，將之設為 0 可以重置計數器值。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考 Table 0-3)。
dwdata 實際位置計數器輸入值，資料範圍： $-2^{31} \sim +2^{31}$ 。

Example: `//清除實際位置計數器值
i8092_SET_EP(1, 0x1, 0);
i8092_SET_EP(1, 0x2, 0);`

i8092_GET_LP

Format: `long i8092_GET_LP(BYTE cardNo, WORD axis)`

Function: 讀取當時邏輯位置計數器的值，而且將被設定在讀取暫存器RR6和RR7。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考 Table 0-3)。

Example: `//讀取 X、Y 軸的邏輯位置計數器
i8092_GET_LP(1, 0x1);
i8092_GET_LP(1, 0x2);`

i8092_GET_EP

Format: `long i8092_GET_EP(BYTE cardNo, WORD axis)`

Function: 讀取當時實際位置計數器的值，而且將被設定在讀取暫存器RR6和RR7。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考 Table 0-3)。

Example: `//讀取 X、Y 軸的實際位置計數器
i8092_GET_EP(1, 0x1);
i8092_GET_EP(1, 0x2);`

i8092_GET_CV

Format: `WORD i8092_GET_CV(BYTE cardNo, WORD axis)`

Function: 讀取當時驅動速度而且可以被設定在讀取暫存器RR6和RR7。當停止驅動時，
此讀取值變為0。當速度從起始速度(SV)到達驅動速度(V)資料值將會增加。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考 Table 0-3)。

Example: `//讀取 X、Y 軸當時速度
i8092_GET_CV(1, 0x1);
i8092_GET_CV(1, 0x2);`

i8092_GET_CA

Format: WORD i8092_GET_CA(BYTE *cardNo*, WORD *axis*)

Function: 讀取當時驅動速度而且可以被設定在讀取暫存器RR6和RR7。當停止驅動時，
此讀取值變為0。當加速度從0到設定值(A)資料值將會增加。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考 Table 0-3)。

Example: *//讀取 X、Y 軸當時加速度*
i8092_READ_CA(1, 0x1);
i8092_READ_CA(1, 0x2);

i8092_SET_CP i8092_SET_CM

Format: void i8092_SET_CP(BYTE *cardNo*, WORD *axis*, long *dwdata*)
void i8092_SET_CM(BYTE *cardNo*, WORD *axis*, long *dwdata*)

Function: 將暫存器COMP+/-的值設定成為正方向軟體極限。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考 Table 0-3)。
dwdata 暫存器 COMP+ 的值，資料範圍： $-2^{31} \sim +2^{31}$ 。

Example: *//將 X、Y 軸的正方向軟體極限設為 100000*
i8092_SET_COMPP(1, 0x3, 100000);
//將 X、Y 軸的負方向軟體極限設為 100000
i8092_SET_COMPM(1, 0x3, 100000);

i8092_VRING_ENABLE i8092_VRING_DISABLE

Format: **void** i8092_VRING_ENABLE(BYTE *cardNo*, WORD *axis*,
 DWORD *nVRing*)
 void i8092_VRING_DISABLE(BYTE *cardNo*, WORD *axis*)

Function: 將最大值的設定致能/除能。相較於直線運動，對轉了一圈會回到原點的圓周運動的軸位置管理，這個函式是非常有益的。

Parameter: *cardNo* 卡號。
 axis 指定軸號碼(參考 Table 0-3)。
 nVRing 暫存器 COMP+/COMP- 的值，資料範圍： $-2^{31} \sim +2^{31}$ 。

Example: //舉例來說，設定一個一圈 10000 個派波的轉動軸。為了致能環狀位置變化方
 程式，將 COMP+/-暫存器設定 9999 當做邏輯位置計數器的最大值。
 i8092_VRING_ENABLE(1, 0xf, 1, 9999);

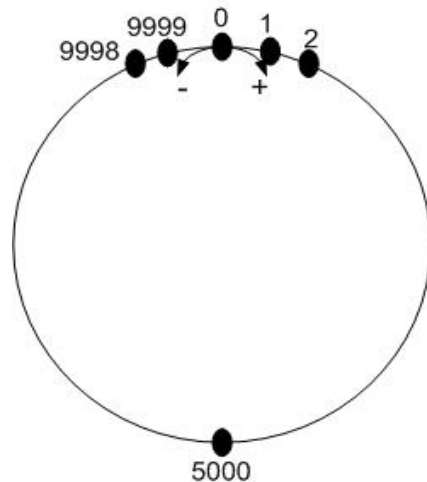


Fig.0-32 環狀位置計數器最大值 9999 的操作

- 註：
1. 環狀變化函式的致能除能設定是對每一軸，然而，邏輯位置計數器和實際位置計數器是不能獨立致能或除能的。
 2. 如果環狀變化函式被致能，那麼軟體極限函式就不能被使用。

i8092_AVTRI_ENABLE

Format: **void** i8092_AVTRI_ENABLE(BYTE *cardNo*, WORD *axis*)

Function: 致能預防三角形速度曲線的產生。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考 Table 0-3)。

Example: i8092_AVTRI_ENABLE(1, AXIS_X);
 //設定第 1 卡 X 軸，致能預防三角形速度的產生。

i8092_AVTRI_DISABLE

Format: **void** i8092_AVTRI_DISABLE(BYTE *cardNo*, WORD *axis*)

Function: 除能預防三角形速度曲線的產生。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考 Table 0-3)。

Example: i8092_AVTRI_ENABLE(1, AXIS_X);
 //設定第 1 卡 X 軸，除能預防三角形速度的產生。

A.8.4 基本運動命令函式

i-8092 的基本運動命令都列在 Table0-8。這些函式的設定包含範圍(R)、倍數(M)、起始速度(SV)、驅動速度(V)、加速度(A)、減速度(D)、加速度比(K)、輸出脈波數(P)、T 曲線加減速和 S 曲線加減速。整個命令程序應該要和初始命令暫存器一起設定。在設定完相關參數之後，CPU 透過 MCX312 送出資料和命令；最後，命令進入邏輯位置計數器，然後會送到驅動器去控制馬達。

Table0-7 基本運動命令函式

函數名稱	描述
i8092_SET_SV	設定初始速度
i8092_SET_V	設定驅動速度
i8092_SET_A	設定加速度
i8092_SET_D	設定減速度
i8092_SET_K	設定加速比
i8092_SET_L	設定減速比
i8092_SET_PULSE	設定輸出脈波數
i8092_SET_AO	設定加速計數補償
i8092_SET_TCURVE	T 曲線加減速模式致能
i8092_SET_SCURVE	S 曲線加減速模式致能
i8092_SET_AUTODEC	設定自動減速
i8092_SET_MANDEC	設定手動減速
i8092_DRV_FDRIVE	設定固定脈波驅動模式
i8092_DRV_CDRIVE	設定連續驅動模式
i8092_SET_SYMMETRY	設定對稱性 T/S 曲線加減速模式
i8092_SET_ASYMMETRY	設定非對稱性 T 曲線加減速模式
i8092_STOP_WAIT	驅動和等待停止
i8092_STOP_SLOWLY	遲緩的停止
i8092_STOP_SUDDENLY	緊急停止
i8092_DRV_HOLD	保持驅動
i8092_DRV_START	驅動狀態保持釋放/完成狀態清除

i8092_SET_SV

Format: `void i8092_SET_SV(BYTE cardNo, WORD axis, WORD data)`

Function: 此函數可以設定起始速度。如果停止類型是緩停止，則運動曲線將減速到起始速度然後停止。設定起始速度是 **SV**，倍數是 **M**，則驅動速度是：
 $\text{起始速度} = \text{SV} \times \text{M}(\text{PPS})$ 。

Parameters: **cardNo** 卡號。
axis 指定軸號碼(參考 Table 0-3)。
data **SV** 的值，資料範圍 1~8000，其他值無效。

Example: `//設定 X 軸的起始速度 500 (PPS)`
`i8092_SET_SV(1, 0x1, 500);`

i8092_SET_V

Format: `void i8092_SET_V(BYTE cardNo, WORD axis, WORD data)`

Function: 此函式用來設定T形驅動時，常數速度週期的速度。在常數速度驅動時，驅動速度等於初始速度。驅動速度計算公式如下： $\text{驅動速度} = \text{V} \times \text{M}(\text{PPS})$ 。

Parameters: **cardNo** 卡號。
axis 指定軸號碼(參考 Table 0-3)。
data **V** 的值，資料範圍 1~8000，其他值無效。

Example: `//設定 X 軸的驅動速度 1000 (PPS)`
`i8092_SET_V(1, 0x1, 1000);`

註： 1.如果驅動速度的設定比初始速度低，則加減速將不會被執行，而且將會是常數速度驅動。在Z相編碼搜尋期間(低速驅動)，一旦Z相被偵測到，使用者要執行緊急停止，所以驅動速度應該設比初始速度低。驅動期間驅動速度可以被改變。當下一個常數速度週期的驅動速度設定好，加減速將被執行去達到新設定的驅動速度，然後一個常數速度驅動開始。

註： 2.固定脈波的S曲線加減速驅動模式中，驅動期間的驅動速度無法改變。連續S曲線加減速驅動模式中，如果在加減速期間速度改變，S曲線的特性無法精確的遵循軌跡。最好的方法是在常數速度期間才改變驅動速度。

i8092_SET_A

Format: `void i8092_SET_A(BYTE cardNo, WORD axis, WORD data)`

Function: 此函式是設定T-曲線驅動時的加減速，對S-曲線的加減速來說，將會以線性加

速直到給一個指定值(A)去驅動。加速計算公式如下：
加速速度 = $A \times 125 \times M$ (PPS/Sec)。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考 Table 0-3)。
data A 加速度的值，資料範圍 1~8000，其他值無效。

Example: *//設定 X 軸的加速度為 80 (PPS/Sec)。*
`i8092_SET_A(1, 0x1, 80);`

i8092_SET_D

Format: `void i8092_SET_D(BYTE cardNo, WORD axis, WORD data)`

Function: 此函式使用在當加速度/減速被分別確定時，"D" 是決定T-曲線驅動的減速度

參數。對S曲線加減速而言，被指定的減速度可以設定直到指定值(D)驅動。

減速度計算公式如下：減速度 = $D \times 125 \times M$ (PPS/Sec)。

Parameters: *cardNo* 卡號。
axis 是指定軸號碼(參考 Table 0-3)。
data D 減速度的值，資料範圍 1~8000，其他值無效。

Example: *//設定 X 軸的減速度為 80 (PPS/Sec)*
`i8092_SET_D(1, 0x1, 80);`

i8092_SET_K

Format: `void i8092_SET_K(BYTE cardNo, WORD axis, WORD data)`

Function: 此函式設定在S曲線加減速時一個時間單位內的加速度比。
加速度比計算公式如下： $\text{Jerk} = 62.5 \times 10^6 / K \times M (\text{PPS}/\text{Sec}^2)$ 。

Parameters: *cardNo* 為卡號。
axis 是指定軸號碼(參考 Table 0-3)。
data K 加速度率的值，資料範圍 1~65535，其他值無效。

Example: `//設定 X 軸的加速度比為 625 (PPS/Sec2)
i8092_SET_K(1, 0x1, 625);`

註: 由於 $K=65535$ to 1 。當乘數 = 1 ， $954 \text{ PPS}/\text{SEC}^2 \sim 62.5 \times 106 \text{ PPS}/\text{SEC}^2$ ；
當倍數 = 500 ， $477 \times 103 \text{ PPS}/\text{SEC}^2 \sim 31.25 \text{ } 109 \text{ PPS}/\text{SEC}^2$ 。在這手冊中，jerk 被定義成單位時間內加速度的加速度比。然而，jerk 應該包括加速度的減少率和減速度的增加率。

i8092_SET_PULSE

Format: `void i8092_SET_PULSE(BYTE cardNo, WORD axis, DWORD data)`

Function: 此函式設定在固定脈波驅動時的全部輸出脈波數。這個值是絕對的正數值，
在驅動期間輸出脈波數可以被改變。

Parameters: *cardNo* 為卡號。
axis 是指定軸號碼(參考 Table 0-3)。
data 脈波數範圍 0~268435455，其他值無效。

Example: `//設定X軸的驅動脈波數為10000
i8092_SET_PULSE(1, 0x1, 10000);`

i8092_SET_AO

Format: `void i8092_SET_AO(BYTE cardNo, WORD axis, short int data)`

Function: 此函式執行加速計數器的補償。當機器使用步進馬達時此函式時常會用到，
高速減速時此函式可以避免減速過量。

Parameters: *cardNo* 是板號。
axis 是指定軸號碼(參考 Table 0-3)。
data 減速補償值，範圍 -32768~+32767，其他值無效。

Example: `//設置緩衝脈波數為 200
i8092_SET_AO(1, 0x1, 200);`

i8092_SET_AUTODEC

Format: `void i8092_SET_AUTODEC(BYTE cardNo, WORD axis)`

Function: 自動減速設定

Parameters: *cardNo* 板號。
axis 指定軸號碼(參考 Table 0-3)。

Example: `//啟動 4 軸自動減速
i8092_SET_AUTODEC(1, 0xf);`

<p>註： 此函式在 T 曲線作環狀補間時是無效的。</p>

i8092_SET_MANDEC

Format: `void i8092_SET_MANDEC(BYTE cardNo, WORD axis, WORD dp)`

Function: 當忙於手動減速模式時，在固定脈波加速/減速驅動或補間運動中設定手動減速

點。在手動減速模式，使用者可以將WR3的D0位元設定為1。
減速點計算如下：手動減速點 = 輸出脈波數 — 減速脈波數。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考 Table 0-3)。
dp 減速值。

Example: `//設定 XY 軸運動的手動減速點 8000`
`i8092_SET_MANDEC(1, 0x3, 8000);`

<p>註： 設定手動減速點適當的時間</p> <ol style="list-style-type: none">1. 非對稱 S 曲線加減速2. 環狀補間
--

i8092_DRV_FDRIVE

Format: `void i8092_DRV_FDRIVE(BYTE cardNo, WORD axis, WORD nDir)`

Function: 設定固定脈波驅動。

Parameters: *cardNo* 是板號。
axis 是指定軸號碼(參考 Table 0-3)。
nDir nDir = 0: 正方向，nDir = 1: 負方向。

Example: `//設定負方向固定脈波驅動。`
`i8092_DRV_FDRIVE(1, 0x3, 1);`

i8092_DRV_CDRIVE

Format: **void** i8092_DRV_CDRIVE(BYTE *cardNo*, WORD *axis*, WORD *nDir*)

Function: 設定連續驅動。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考 Table 0-3)。
 nDir nDir = 0: 正方向，nDir = 1: 負方向。

Example: *//設定正方向連續驅動*
 i8092_DRV_CDRIVE(1, 0x3, 0);

i8092_SET_SYMMETRY

Format: **void** i8092_SET_SYMMETRY(BYTE *cardNo*, WORD *axis*)

Function: 設定對稱加減速。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考 Table 0-3)。

Example: *//設定 4 軸對稱加減速運動*
 i8092_SET_SYMMETRY(1, 0xf);

i8092_SET_ASYMMETRY

Format: **void** i8092_SET_ASYMMETRY(BYTE *cardNo*, WORD *axis*)

Function: 設定非對稱加減速。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考 Table 0-3)。

Example: *//設定 4 軸非對稱加減速運動*
 i8092_SET_ASYMMETRY(1, 0xf);

i8092_STOP_SLOWLY

Format: **void** i8092_STOP_SLOWLY(BYTE *cardNo*, WORD *axis*)

Function: 減速停止命令。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考 Table 0-3)。

Example: *//對 4 軸下減速停止命令*
 i8092_STOP_SLOWLY(1, 0xf);

i8092_STOP_SUDDENLY

Format: **void** i8092_STOP_SUDDENLY(BYTE *cardNo*, WORD *axis*)

Function: 緊急停止命令。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考 Table 0-3)。

Example: //對 4 軸下緊急停止命令
 i8092_STOP_SUDDENLY(1, 0xf);

i8092_DRV_HOLD

Format: **void** i8092_DRV_HOLD(BYTE *cardNo*, WORD *axis*)

Function: 對開始驅動設定保持。

Parameters: *cardNo* 是板號。
 axis 是指定軸號碼(參考 Table 0-3)。

Example: i8092_DRV_HOLD(1, 0xf);

i8092_DRV_START

Format: **void i8092_DRV_START(BYTE cardNo, WORD axis)**

Function: 驅動狀態保持釋放，完成狀態時清除設定。

Parameters: **cardNo** 卡號號。
 axis 指定軸號碼(參考 Table 0-3)。

Example: **i8092_DRV_SATRT(1, 0xf);**

Demo Program: T/S 曲線加速/減速運動[對稱]

Parameters: cardNo=1, motion axes=0x3 (AXIS_XY)

```
PS400_SET_R(cardNo, 0x3, 800000);    // 倍率=10
PS400_SET_TCURVE(cardNo, 0x3);       // T-Curve 模式
PS400_SET_SYMMETRY(cardNo, 0x3);    // 對稱模式
PS400_SET_SV(cardNo, 0x3, 100);      // 初始速度=1000 (PPS)
PS400_SET_V(cardNo, 0x3, 1000);     // 驅動速度=10000 (PPS)
PS400_SET_A(cardNo, 0x3, 80);        // 加速度=10K (PPS/Sec)
PS400_SET_PULSE(cardNo, 0x3, 25000); // 驅動脈波數=25000
PS400_DRV_HOLD(card, 0x3);           // 等待驅動
PS400_DRV_FDRIVE(cardNo, 0x3, 0);    // 定量驅動
PS400_DRV_START(card, 0x3);          // 等待驅動解除
```

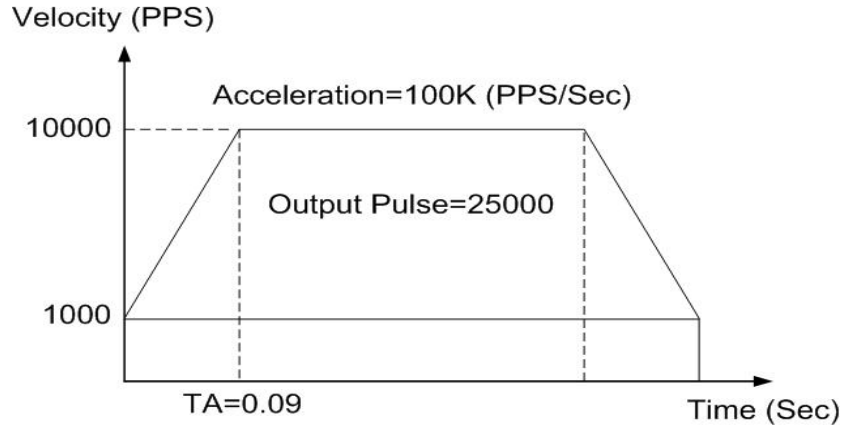


Fig.0-33 對稱性 T 曲線加/減速

Demo Program: T 曲線加速/減速運動[非對稱]

Parameters: cardNo=1, motion axes=0x3 (AXIS_XY)

```

PS400_SET_R(cardNo, 0x3, 8000000); // 倍率=10
PS400_SET_TCURVE(cardNo, 0x3); // S-Curve 模式
PS400_SET_ASYMMETRY(cardNo, 0x3); // 非對稱模式
PS400_SET_SV(cardNo, 0x3, 100); // 初始速度=1000 (PPS)
PS400_SET_V(cardNo, 0x3, 1000); // 驅動速度=10000 (PPS)
PS400_SET_A(cardNo, 0x3, 800); // 加速度=1000K (PPS/Sec)
PS400_SET_D(cardNo, 0x3, 80); // 減速度=100K (PPS/Sec)
PS400_SET_K(cardNo, 0x3, 1250); // 加速度變化率=500K (PPS/Sec2)
PS400_SET_L(cardNo, 0x3, 125); // 減速度變化率=50K (PPS/Sec2)
PS400_SET_PULSE(cardNo, 0x3, 50000);
PS400_DRV_HOLD(card, 0x3); // 等待驅動
PS400_DRV_FDRIVE(cardNo, 0x1, 1); // X 軸負方向定量驅動
PS400_DRV_FDRIVE(cardNo, 0x2, 0); // Y 軸正方向定量驅動
PS400_DRV_START(card, 0x3); // 等待驅動解除

```

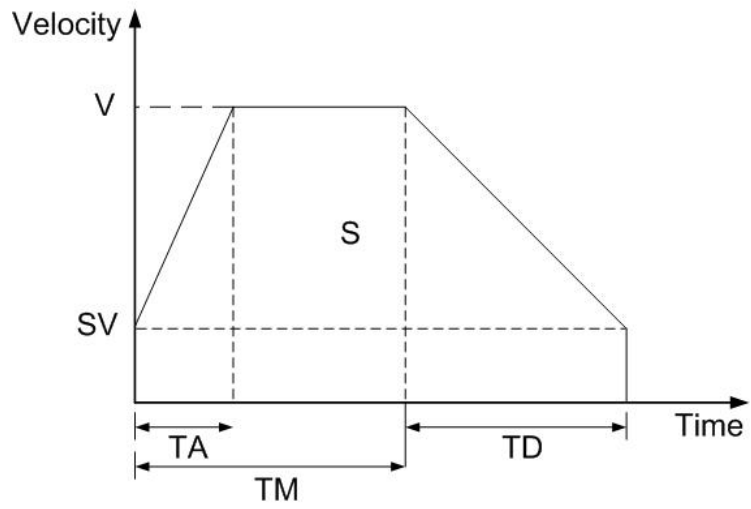


Fig.0-34 非對稱性 T 曲線加/減速

A.8.5 補間函式

下圖是 MCX312 的補間函式方塊圖。由相同的 X、Y 軸控制部分和補間計算部分組成。

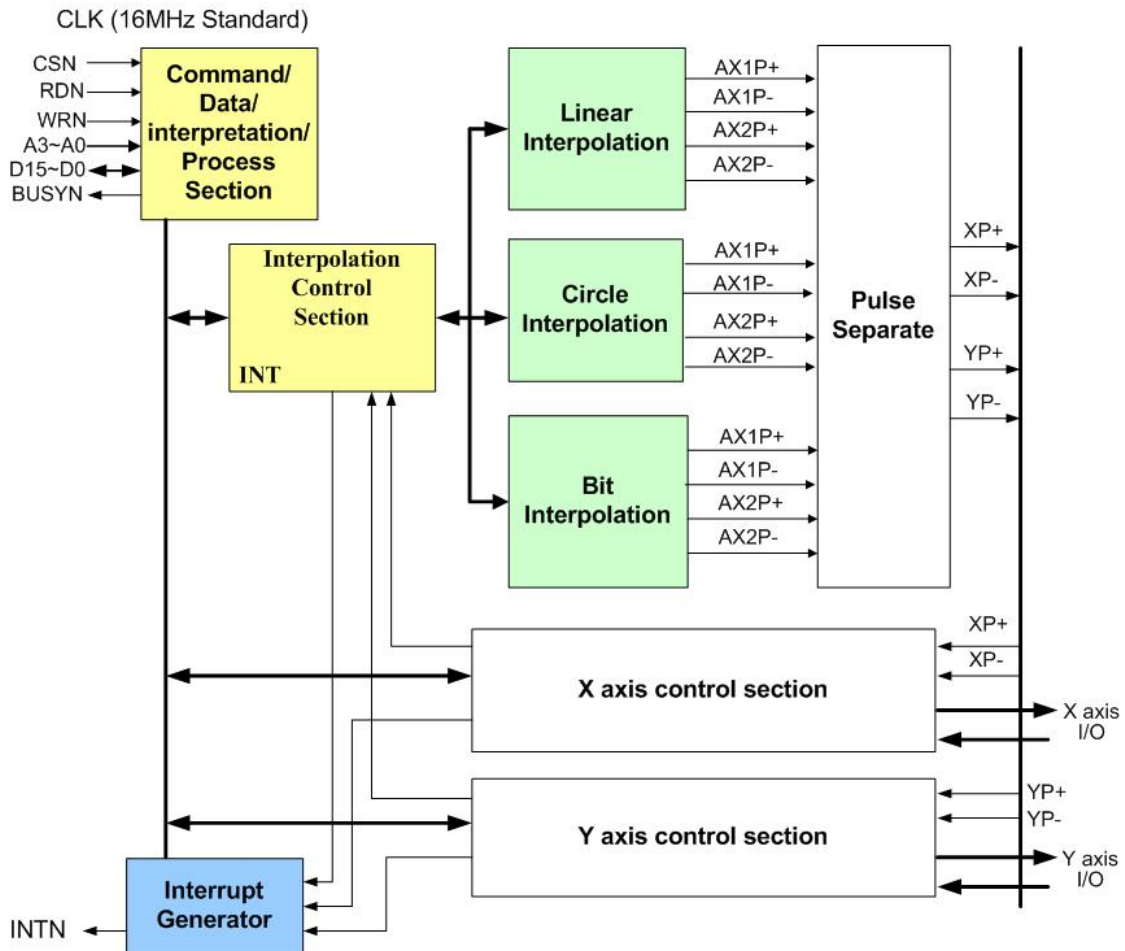


Fig.0-35 MCX312 函式方塊圖

Table0-8 補間函數

函式名稱	描述
i8092_MOTION_TYPE	2 軸補間常數向量速度設定
i8092_SET_FINISH	補間終點設定
i8092_LINE2D	2 軸線性補間模式
i8092_SET_CENTER	環狀補間中心設定
i8092_ARC_CW	順時針方向弧形補間模式
i8092_ARC_CCW	逆時針方向弧形補間模式
i8092_CIRCLE_CW	順時針方向圓形補間模式
i8092_CIRCLE_CCW	逆時針方向圓形補間模式
i8092_NEXT_WAIT	等待下一個補間命令
i8092_BP_ENABLE	致能位元補間
i8092_BP_DISABLE	除能位元補間
i8092_BP_CLEAR	清除位元圖案補間
i8092_BP_STACK	位元補間資料堆疊
i8092_BP_WAIT	等待位元補間資料寫入

i8092_MOTION_TYPE

Format: **void** i8092_MOTION_TYPE(BYTE *cardNo*, WORD *type*)

Function: 設定 2 軸向量速度模式。

Parameter: *cardNo* 卡號。
 type 向量速度模式的參數設定：
 type = 0，常數向量速度模式無效。
 type = 1、2 軸常數向量速度模式。

Example: *//設定 2 軸向量速度模式*
 i8092_MOTION_TYPE(1, 1);

直線補間函式

關於線性補間函式分成 3 種模式：常(正切)速，T 曲線(正切)加速/減速，S 曲線(正切)加速/減速。使用者必須設定下列參數：

- 範圍： R
- 起始速度： SV (PPS)
- 驅動速度： V (PPS)
- 加速度： A (PPS/Sec)
- 終點： FP

i8092_LINE_2D

Format: **void** i8092_LINE_2D(BYTE *cardNo*, long *fp1*, long *fp2*)

Function: 2 軸直線補間。

Parameter: *cardNo* 卡號。
fp1 X 軸終點，資料範圍 -8388608~8388607。
fp2 Y 軸終點，資料範圍 -8388608~8388607。

Example: i8092_LINE_2D(1, 12000, 10000);

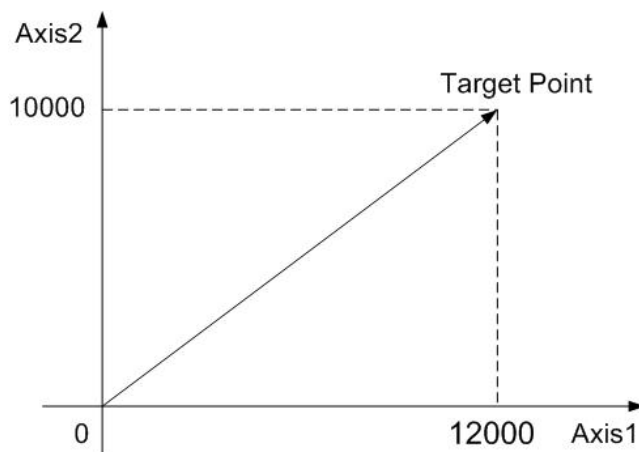


Fig.0-36 2 軸直線補間

Demo Program: 2 軸直線補間

Parameters: cardNo=1

// 2 軸直線補間

```
i8092_MOTION_TYPE(cardNo, ACCMODE);
```

// 設定 2 軸向量加速模式

```
i8092_SET_R(CardNo, 0x1, 8000000);
```

// 倍率設定

```
i8092_SET_R(cardNo, 0x2, 8000000*1414L/1000L);
```

```
i8092_SET_SV(cardNo, 0x1, 100)
```

// 初速度設定

```
i8092_SET_A(cardNo, 0x1, 500);
```

// 加速度設定

```
i8092_SET_V(cardNo, 0x1, 5000);
```

// 驅動速度設定

```
i8092_LINE_2D(cardNo, 3000, 4000);
```

// 2 軸補間

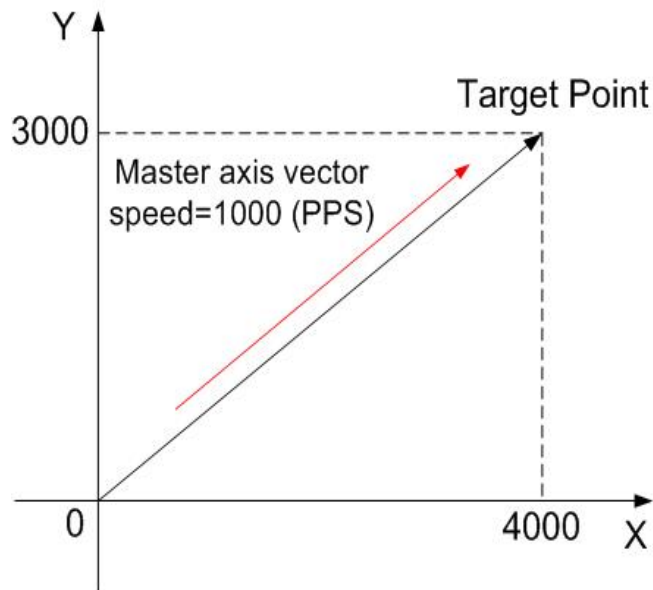


Fig.0-37 2 軸直線補間

圓弧補間

i8092_ARC_CW

Format: `void i8092_ARC_CW(BYTE cardNo, long cp1, long cp2, long fp1, long fp2)`

Function: 順時針方向圓弧補間。

Parameters: *cardNo* 卡號。
 cp1 X 軸圓心。
 cp2 Y 軸圓心。
 fp1 X 軸終點。
 fp2 Y 軸終點。

Example: `i8092_ARC_CW(1, -5000, -5000, -10000, -10000);`

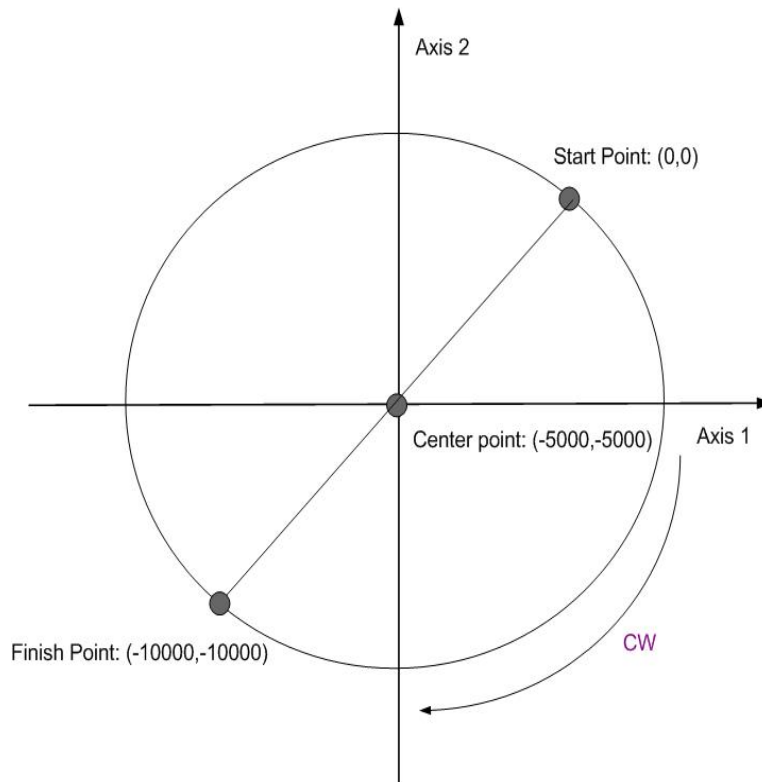


Fig.0-38 順時針圓形補間

i8092_ARC_CCW

Format: `void i8092_ARC_CCW(BYTE cardNo, long cp1, long cp2, long fp1, long fp2);`

Function: 逆時針方向圓弧補間。

Parameters:

- cardNo* 卡號。
- cp1* X 軸圓心。
- cp2* Y 軸圓心。
- fp1* X 軸終點。
- fp2* Y 軸終點。

Example: `i8092_ARC_CCW(1, -5000, -5000, -10000, -10000);`

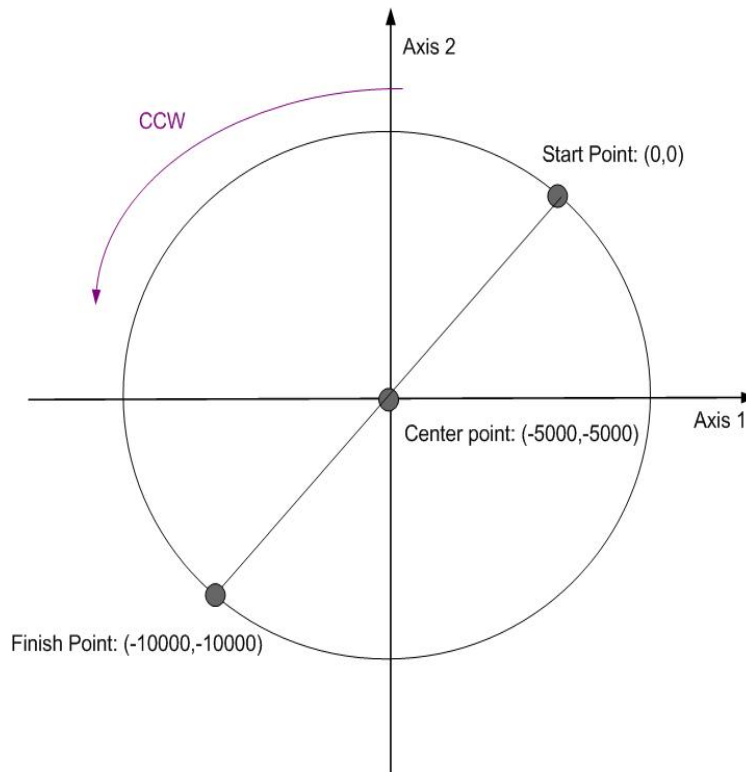


Fig.0-39 逆時針圓弧補間

i8092_CIRCLE_CW

Format: `void i8092_CIRCLE_CW(BYTE cardNo, long cp1, long cp2)`

Function: 以順時針方向圓弧補間實現一個圓。

Parameters: *cardNo* 卡號。
cp1 X 軸圓心。
cp2 Y 軸圓心。

Example: `i8092_CIRCLE_CW(1, 0, 10000);`

i8092_CIRCLE_CCW

Format: `void i8092_CIRCLE_CCW(BYTE cardNo, long cp1, long cp2)`

Function: 以逆時針方向圓弧補間實現一個圓。

Parameters: *cardNo* 卡號。
cp1 X 軸圓心。
cp2 Y 軸圓心。

Example: `i8092_CIRCLE_CCW(1, 0, 10000);`

Demo Program: 圓弧補間

Parameters: cardNo=1, master axis=0x1 (AXIS_X), 2nd axis=0x2 (AXIS_Y)

// T 曲線加速的順時針圓弧補間(See Fig.2-8)

```
PS400_MOTION_TYPE(cardNo, ACCMODE);           // 設定 2 軸加速模式
PS400_SET_R(CardNo, Card[cardNo].ax1, 8000000);
PS400_SET_R(cardNo, Card[cardNo].ax2, 8000000*1414L/1000L);
PS400_SET_SV(cardNo, Card[cardNo].ax1, 200);
PS400_SET_V(cardNo, Card[cardNo].ax1, 3000);
PS400_SET_A(cardNo, Card[cardNo].ax1, 625);
PS400_ARC_CW(1, -5000, -5000, -10000, -10000); // 順時針弧形命令
```

// T 曲線加速的逆時針圓弧補間(See Fig.2-9)

```
PS400_MOTION_TYPE(cardNo, ACCMODE);           // 設定 2 軸加速模式
PS400_SET_R(cardNo, Card[cardNo].ax1, 8000000);
PS400_SET_R(cardNo, Card[cardNo].ax2, 8000000*1414L/1000L);
PS400_SET_SV(cardNo, Card[cardNo].ax1, 200);
PS400_SET_V(cardNo, Card[cardNo].ax1, 3000);
PS400_SET_A(cardNo, Card[cardNo].ax1, 625);
PS400_ARC_CCW(1, -5000, -5000, -10000, -10000); // 逆時針弧形命令
```

連續補間

使用者可以使用線性和圓弧補間去組合特殊的曲線。有 2 種方式可以實現：輪詢和中斷。Fig. 0-50 顯示連續執行從(0,0)開始的 8 個區塊範例。在區塊 1、3、5 和 7 會執行線性補間；在區塊 2、4、6 和 8 會執行圓形補間，而且軌跡是擁有半徑 1500 的四分之一圓。補間驅動以常數向量進行：1500 PPS。

i8092_NEXT_WAIT

Format: `void i8092_NEXT_WAIT(BYTE cardNo)`

Function: 等待下一區塊的命令。

Parameters: *cardNo* 卡號。

Example: `i8092_NEXT_WAIT(1);`

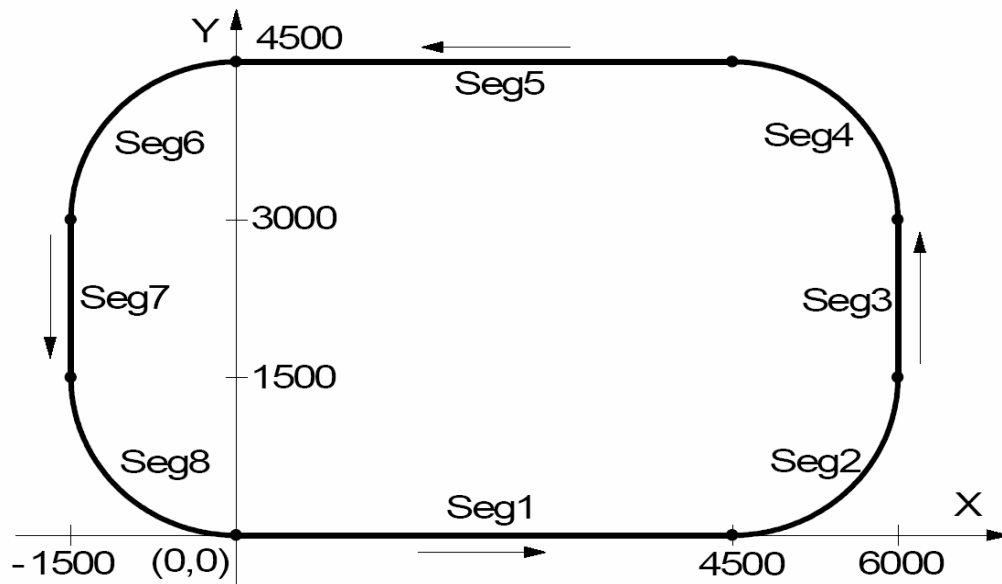


Fig.0-40 連續補間

位元補間函式

i8092_BP_ENABLE i8092_BP_DISABLE

Format: **void** i8092_BP_ENABLE(BYTE *cardNo*)
 void i8092_BP_DISABLE(BYTE *cardNo*)

Function: 開啟/關閉位元補間資料堆疊。

Parameters: *cardNo* 卡號。

Example: i8092_BP_ENABLE(1);
 i8092_BP_DISABLE(1);

i8092_BP_STACK i8092_BP_CLEAR

Format: **void** i8092_BP_STACK(BYTE *cardNo*)
 void i8092_BP_CLEAR(BYTE *cardNo*)

Function: 堆疊/清除位元補間資料。

Parameters: *cardNo* 卡號。

Example: i8092_BP_STACK(1);
 i8092_BP_CLEAR(1);

i8092_BP_WAIT

Format: **void** i8092_BP_WAIT(BYTE *cardNo*)

Function: 等待位元補間資料輸出。

Parameters: *cardNo* 卡號。

Example: i8092_BP_WAIT(1);

依照下面流程圖，使用者可以使用直線補間 DDA 演算法去產生 BP 的資料。然而 $L = \sqrt{P_1^2 + P_2^2}$ ， P_1 、 P_2 是每軸的脈波數。

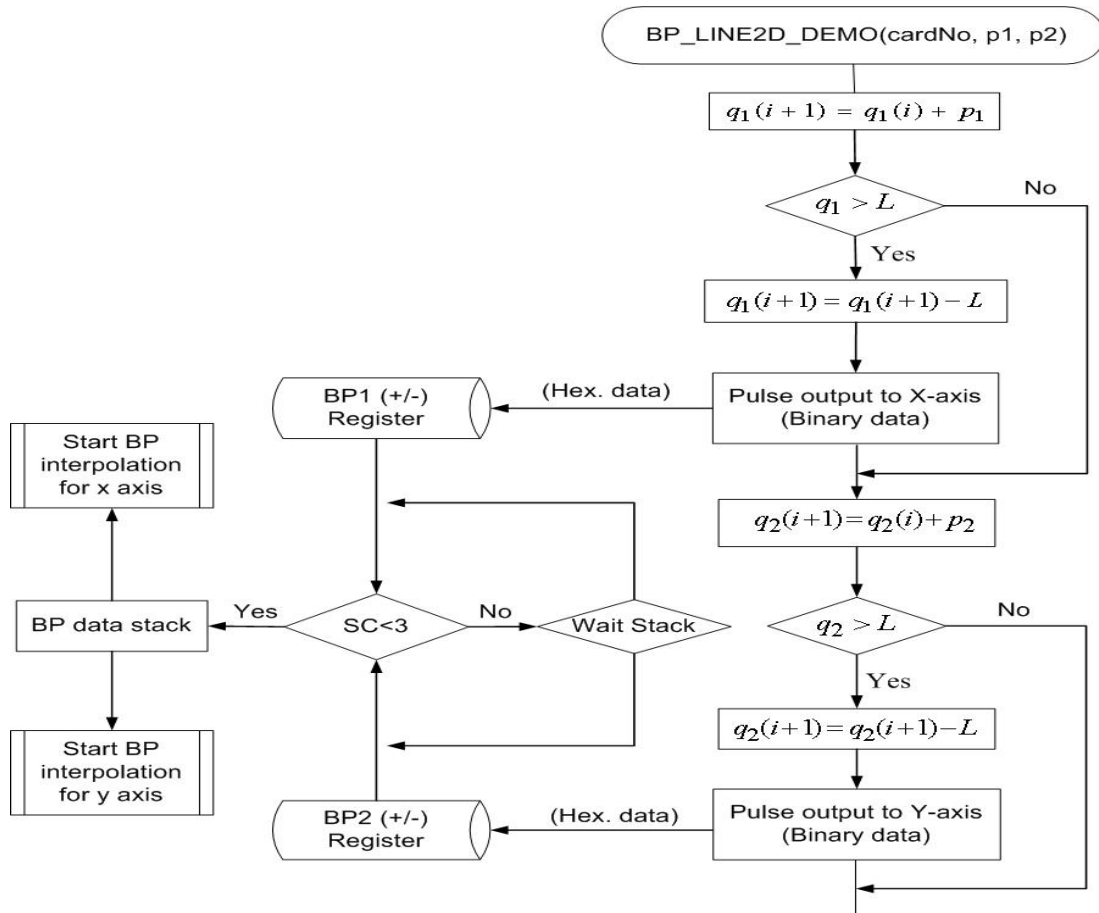


Fig.0-41 使用線性 DDA 方法的位元圖補間

註： 使用者呼叫 PS400_BP_LINE2_DEMO 函式使用輪詢法和呼叫 PS400_BP_LINE_DEMO_INT 函式使用中斷法去實現位元圖。這兩個函式只有在 I-8000 是有效的。

Demo Program: 使用線性 DDA 資料的位元圖補間

Parameters: cardNo=1, master axis=0x1 (AXIS_X), 2nd axis=0x2 (AXIS_Y), P1=30, P2=40

```
PS400_MOTION_TYPE(cardNo, ACCMODE );  
PS400_SET_AUTODEC(cardNo, 0x3);    // 開啟自動補間  
PS400_SET_TCURVE(cardNo, 0x3);    // 設定 T 曲線模式  
PS400_SET_R(cardNo, 0x3, 8000000); // 倍數=1  
PS400_SET_SV(cardNo, 0x3, 50);  
PS400_SET_V(cardNo, 0x3, 500);  
PS400_SET_A(cardNo, 0x3, 80);  
PS400_BP_ENABLE(cardNo);           //開啟 BP 補間  
PS400_BP_LINE2_DEMO(cardNo, 30, 40); // 線性 DDA 資料堆疊  
PS400_BP_DISABLE(cardNo);         //關閉 BP 補間
```

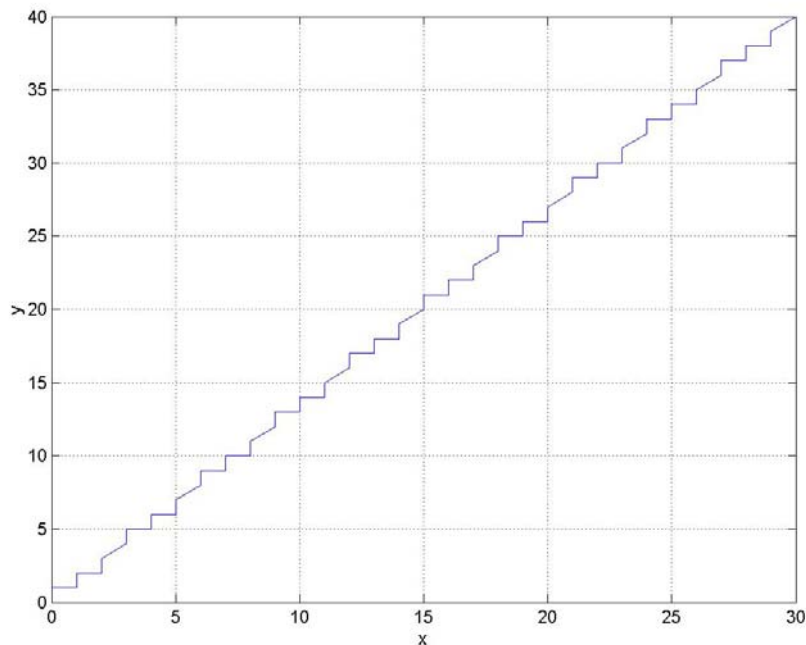


Fig.0-42 使用線性 DDA 方式的位元圖補間

使用直線 DDA 方式，設定 P1=X 軸=30, P2=Y 軸=40，線性 DDA 資料如下：

Table0-9 位元圖補間的 DDA data

Index	Z1	Hex1	Z2	Hex2	X	Y	Index	Z1	Hex1	Z2	Hex2	X	Y
1	0	0x6b5a	0	0x7bde	0	0	26	0	0x5ad6	0	0xdef7	15	20
2	1		1		1	1	27	1		1		16	21
3	0		1		1	2	28	0		1		16	22
4	1		1		2	3	29	1		1		17	23
5	1		1		3	4	30	1		1		18	24
6	0		0		3	4	31	0		0		18	25
7	1		1		4	5	32	1		1		19	25
8	0		1		4	6	33	0		1		19	26
9	1		1		5	7	34	1		1		20	27
10	1		1		6	8	35	1		1		21	28
11	0		0		6	8	36	0		0		21	29
12	1		1		7	9	37	1		1		22	29
13	0		1		7	10	38	0		1		22	30
14	1		1		8	11	39	1		1		23	31
15	1		1		9	12	40	1		1		24	32
16	0		0		9	12	41	0		0		24	32
17	1	0xb5ad	1	0xbdf7	10	13	42	1	0x3	1	0x3	25	33
18	0		1		10	14	43	0		1		25	34
19	1		1		11	15	44	1		1		26	35
20	1		1		12	16	45	1		1		27	36
21	0		0		12	16	46	0		0		27	36
22	1		1		13	17	47	1		1		28	37
23	0		1		13	18	48	0		1		28	38
24	1		1		14	19	49	1		1		29	39
25	1		1		15	20	50	1		1		30	40

註： Z1, Z2 是每個間隔的輸出脈波，X是Z1的總合，Y是Z2的總合，16進制一是由16個0或1(Z1)組成的16位元字元，而16進制二是由16個0或1(Z2)組成的16位元字元。

A.8.6 原點返回運動(軸歸零)函式

Table0-10 自動歸原點函式

函式名稱	描述
i8092_ST0_LEVEL	設定(ST0)近原點觸發準位。
i8092_ST1_LEVEL	設定(ST1)原點觸發準位。
i8092_ST2_LEVEL	設定(ST2)編碼器 Z-相觸發準位。
i8092_STn_ENABLE	致能停止訊號(ST0~ST2)
i8092_STn_DISABLE	除能停止訊號(ST0~ST2)
i8092_HOME1	設定原點返回運動模式 1。
i8092_HOME2	設定原點返回運動模式 2。
i8092_HOME3	設定原點返回運動模式 3。
i8092_HOME4	設定原點返回運動模式 4。
i8092_HOME5	設定原點返回運動模式 5。
i8092_HOME6	設定原點返回運動模式 6。
i8092_HOME7	設定原點返回運動模式 7。
i8092_HOME8	設定原點返回運動模式 8。

i8092_ST0_LEVEL

Format: **void** i8092_ST0_LEVEL(BYTE *cardNo*, WORD *axis*, WORD *nLevel*)

Function: 設定近原點(IN0)信號的邏輯準位。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考表 2-1)。
 nLevel nLevel = 0: 低準位觸發，nLevel = 1: 高準位觸發。

Example: i8092_ST0_LEVEL(1, 0x3, 0);

i8092_ST1_LEVEL

Format: **void** i8092_ST1_LEVEL(BYTE *cardNo*, WORD *axis*, WORD *nLevel*)

Function: 設定原點(ST1)信號的邏輯準位。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考表 2-1)。
 nLevel nLevel = 0: 低準位觸發，nLevel = 1: 高準位觸發。

Example: i8092_ST1_LEVEL(1, 0x3, 0);

i8092_ST2_LEVEL

Format: **void** i8092_ST2_LEVEL(BYTE *cardNo*, WORD *axis*, WORD *nLevel*)

Function: 設定編碼器Z相(IN2)信號的邏輯準位。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考表 2-1)。
 nLevel nLevel = 0: 低準位觸發，nLevel = 1: 高準位觸發。

Example: i8092_ST2_LEVEL(1, 0x3, 0);

i8092_STn_ENABLE i8092_STn_DISABLE

Format: **void** i8092_STn_ENABLE(BYTE *cardNo*, WORD *axis*, WORD *INn*)
 void i8092_STn_DISABLE(BYTE *cardNo*, WORD *axis*, WORD
INn)

Function: 致能/除能停止信號 IN0~3。

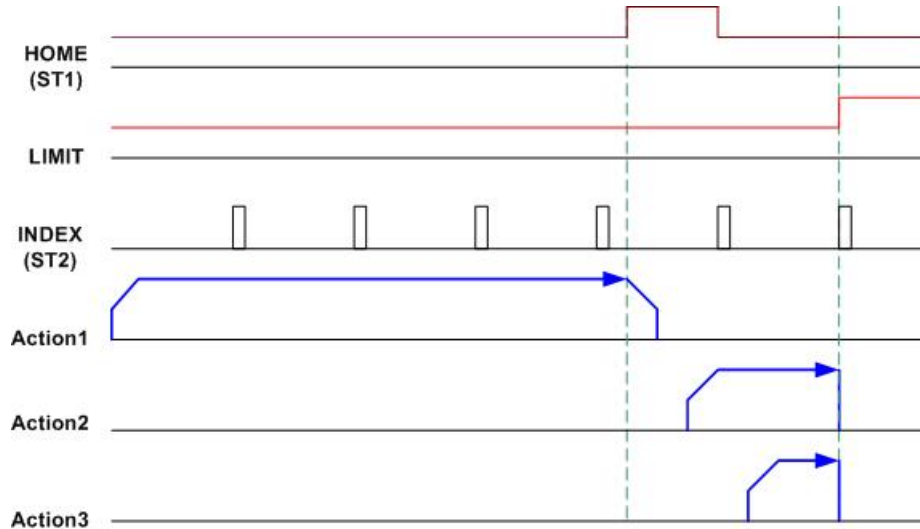
Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考 Table 0-3)。
 INn 0~3: IN0~IN3

Example: //對 XY 軸設定 IN0、IN1 信號致能
 i8092_STn_ENABLE(1, 0x3, 0);
 i8092_STn_ENABLE(1, 0x3, 1);

i8092_HOME1

Format: `void i8092_HOME1(BYTE cardNo, WORD axis, WORD nDir, WORD v)`

Description: 原點返回運動模式 1: 原點訊號(ST1)→減速停止。



Parameters:

- cardNo:** 卡號
- axis:** 指定軸號碼(參考 Table 0-3)。
- nDir:** 0: 正向原點返回運動
1: 負向原點返回運動
- v:** 驅動速度

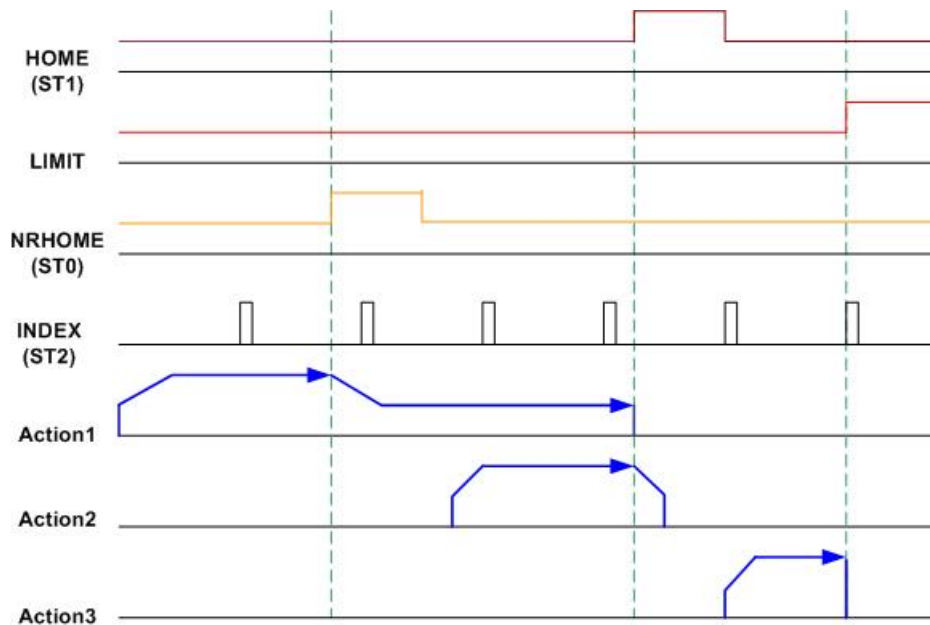
Return: None

Example: `//模式 1: XY 軸正方向尋原, 速度為 5000(PPS)`
`i8092_HOME1(1, AXIS_XY, 0, 5000);`

i8092_HOME2

Format: **void** i8092_HOME2(BYTE *cardNo*, WORD *axis*, WORD *nDir*, WORD *v*, WORD *hv*)

Description: 原點返回模式 2: 近原點訊號(ST0)→減速→較低的等速度→原點訊號(ST1)→停止.



Parameters:

- cardNo:** 卡號
- axis:** 指定軸號碼(參考 Table 0-3)。
- nDir:** 0: 正向原點返回運動
1: 負向原點返回運動
- v:** 驅動速度

Return: None

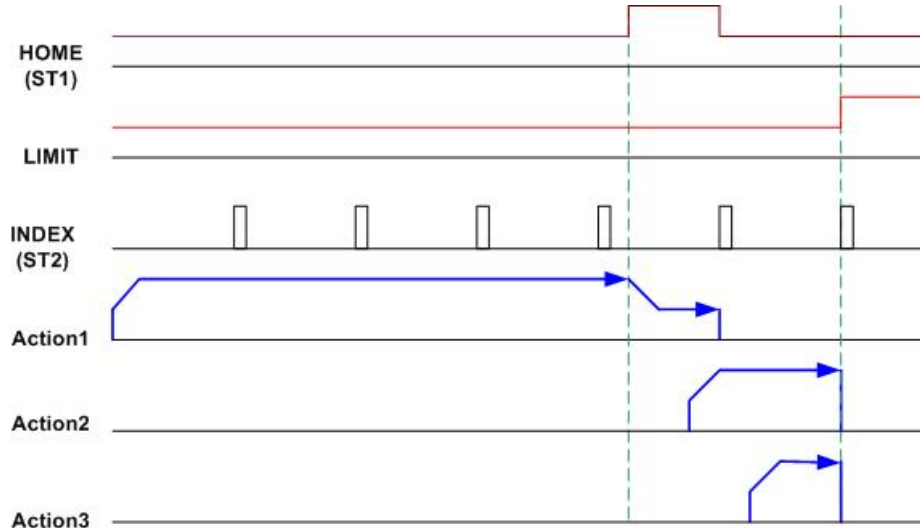
Example: //模式 2: XY 軸正方向尋原, 速度為 5000(PPS), 尋原速度為 1000(PPS)

```
i8092_HOME1(1, AXIS_XY, 0, 5000, 1000);
```

i8092_HOME3

Format: **void i8092_HOME3**(**BYTE cardNo**, **WORD axis**, **WORD nDir**,
WORD v, **WORD hv**)

Description: 原點返回運動模式 3: 原點訊號(ST1)→減速→較低的等速度→編碼器 Z 相訊號(ST2)→停止.



Parameters:

- cardNo:** 卡號
- axis:** 指定軸號碼(參考 Table 0-3)。
- nDir:** 0: 正向原點返回運動
1: 負向原點返回運動
- v:** 驅動速度

Return: None

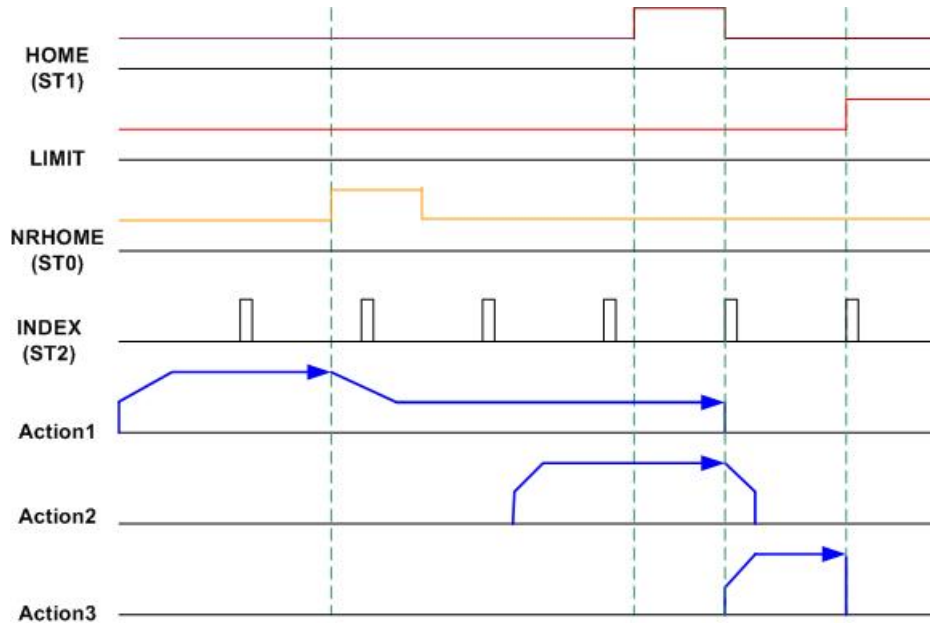
Example: //模式 3: XY 軸正方向尋原, 速度為 5000(PPS), 尋原速度為 1000(PPS)

```
i8092_HOME3(1, AXIS_XY, 0, 5000, 1000);
```


i8092_HOME4

Format: `void i8092_HOME4(BYTE cardNo, WORD axis, WORD nDir, WORD v, WORD hv)`

Description: 原點返回運動模式 4: 近原點(ST0) → 減速 → 較低的等速度 → 原點訊號(ST1) → 等速度 → 編碼器 Z 相訊號(ST2) → 停止。



Parameters:

- cardNo:** 卡號
- axis:** 指定軸號碼(參考 Table 0-3)。
- nDir:** 0: 正向原點返回運動
1: 負向原點返回運動
- v:** 驅動速度
- hv:** 尋原速度

Return: None

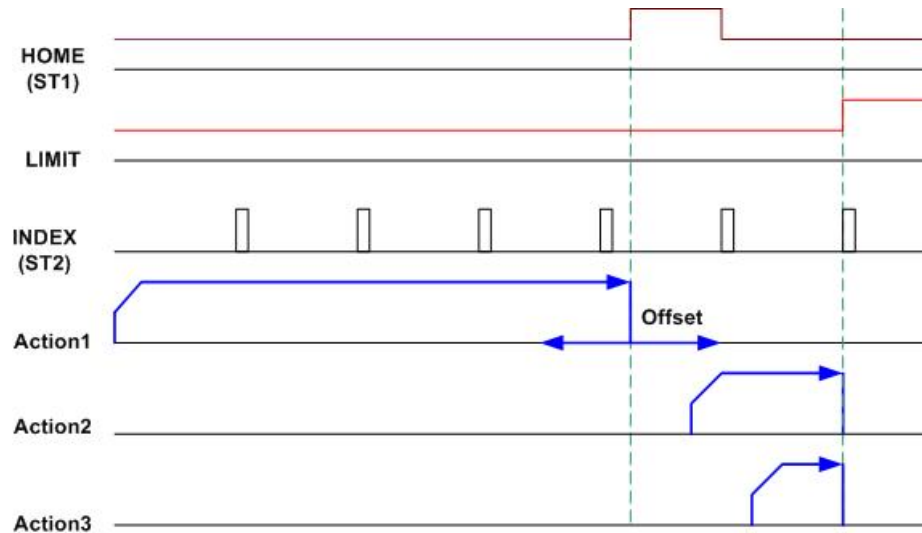
Example: `//模式 4: XY 軸正方向尋原, 速度為 5000(PPS), 尋原速度為 1000(PPS)`

`i8092_HOME4(1, AXIS_XY, 0, 5000, 1000);`

i8092_HOME5

Format: **void** i8092_HOME5(BYTE *cardNo*, WORD *axis*, WORD *nDir*, long *offset*)

Description: 原點返回運動模式 5: 原點訊號(ST1)→ 等速度→ 正負向偏移→ 停止.



Parameters:

- cardNo:** 卡號
- axis:** 指定軸號碼(參考 Table 0-3)。
- nDir:** 0: 正向原點返回運動
1: 負向原點返回運動
- v:** 驅動速度
- offset** 偏移脈波

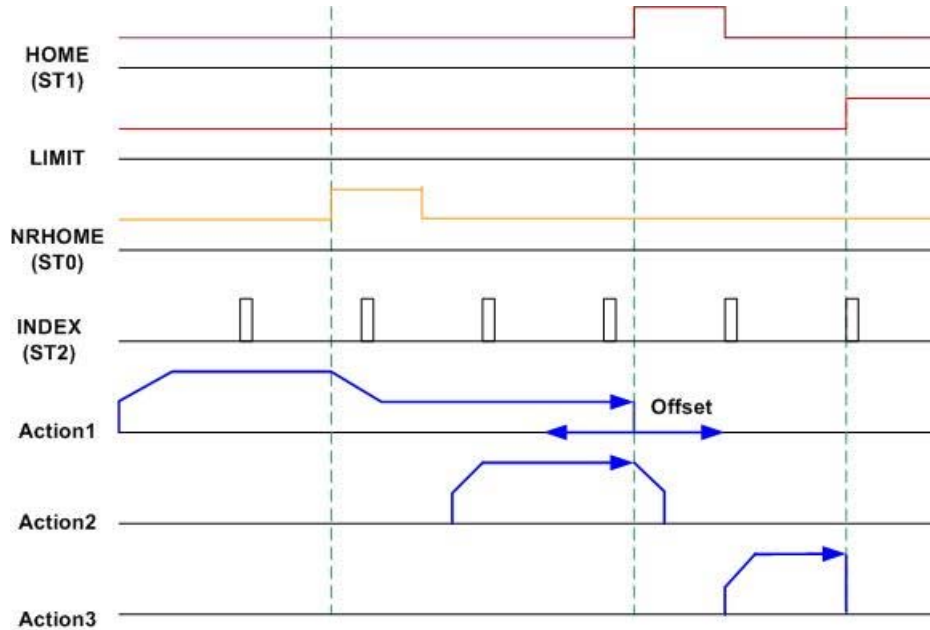
Return: None

Example: //模式 5: XY 軸正方向尋原, 速度為 5000(PPS), 偏移脈波數為 5000
i8092_HOME5(1, AXIS_XY, 0, 5000, 5000);

i8092_HOME6

Format: `void i8092_HOME6(BYTE cardNo, WORD axis, WORD nDir, long offset)`

Description: 原點返回運動模式 6: 近原點訊號(ST0) → 減速 → 較低的等速度 → 原點訊號(ST1) → 等速度 → 正負向偏移 → 停止。



Parameters:

- cardNo:** 卡號
- axis:** 指定軸號碼(參考 Table 0-3)。
- nDir:** 0: 正向原點返回運動
1: 負向原點返回運動
- v:** 驅動速度
- hv** 尋原速度
- offset** 偏移脈波數

Return: None

Example: //模式 6: XY 軸正方向尋原, 速度為 5000(PPS), 尋原速度為 1000(PPS)

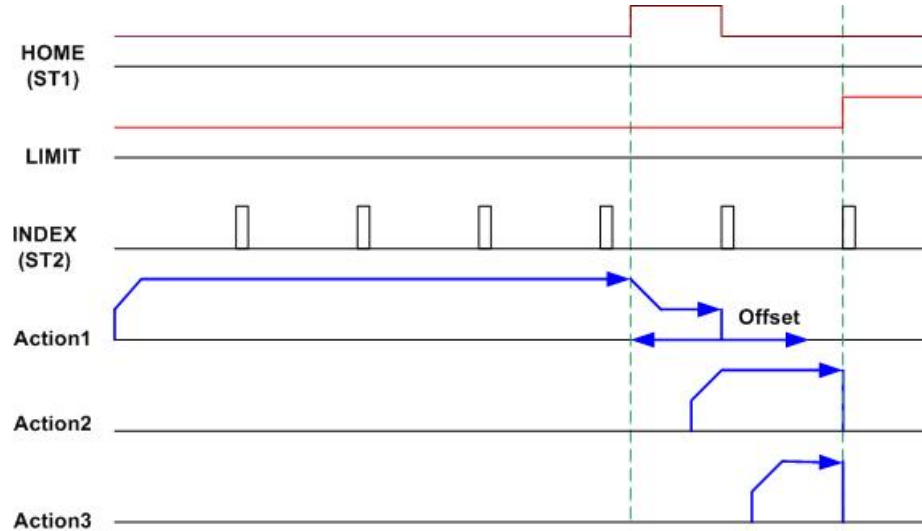
偏移脈波數為 5000

```
i8092_HOME6(1, AXIS_XY, 0, 5000, 1000, 5000);
```

i8092_HOME7

Format: `void i8092_HOME7(BYTE cardNo, WORD axis, WORD nDir, long offset)`

Description: 原點返回運動模式7:原點訊號(ST1)→ 減速→較低的等速度→ 編碼器 Z 相訊號(ST2) → 正負向偏移→ 停止.



Parameters:

- cardNo:** 卡號
- axis:** 指定軸號碼(參考 Table 0-3)。
- nDir:** 0: 正向原點返回運動
1: 正向原點返回運動
- v:** 驅動速度
- hv** 尋原速度
- offset** 偏移脈波數

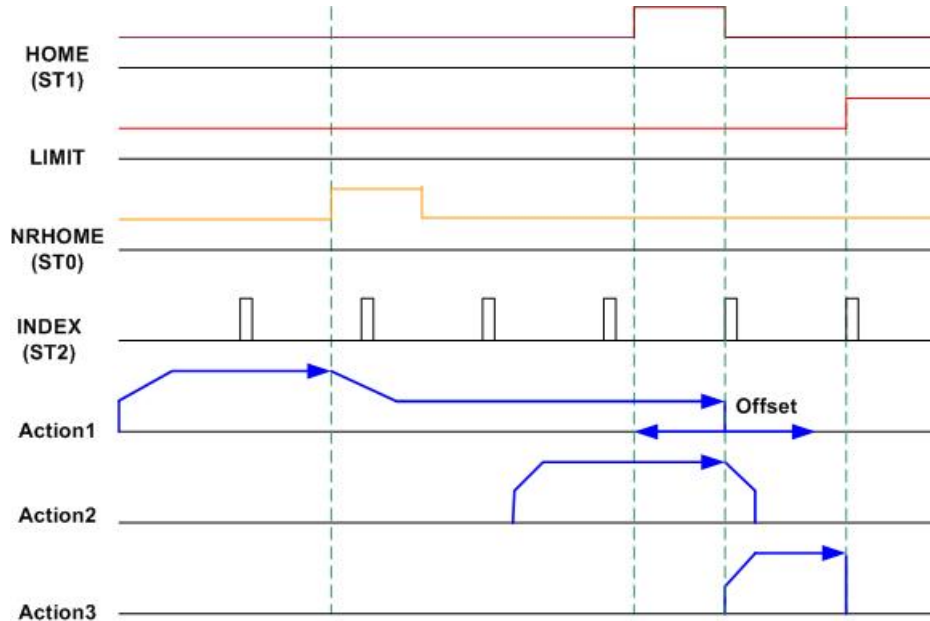
Return: None

Example: `//模式 7: XY 軸正方向尋原, 速度為 5000(PPS), 尋原速度為 1000(PPS) q 偏移脈波數為 5000
i8092_HOME7(1, AXIS_XY, 0, 5000, 1000, 5000);`

i8092_HOME8

Format: `void i8092_HOME8(BYTE cardNo, WORD axis, WORD nDir, long offset)`

Description: 原點返回運動模式 8: 近原點訊號(ST0)→ 減速→較低的等速度→ 原點訊號(ST1)→ 等速度→編碼器 Z 相訊號(ST2) → 正負向偏移→ 停止.



Parameters:

- cardNo:** 卡號
- axis:** 指定軸號碼(參考 Table 0-3)。
- nDir:** 0: 正向原點返回運動
1: 正向原點返回運動
- v:** 驅動速度
- hv:** 尋原速度
- offset:** 偏移脈波數

Return: None

Example: `//模式 8: XY 軸正方向尋原, 速度為 5000(PPS), 尋原速度為 1000(PPS), 移脈波數為 5000
i8092_HOME8(1, AXIS_XY, 0, 5000, 1000, 5000);`

A.8.8 I/O 信號函式

Table0-11 I/O 信號函式

函式名稱	敘述
i8092_LIMITSTOP_MODE	設定碰觸硬體極限信號停止模式
i8092_ENCODER_MODE	設定編碼器信號模式 (PULSE/DIR 或 CW/CCW)
i8092_ENCODER_DEVISION	設定 A/B 相編碼器輸入除頻
i8092_ALARM_ENABLE	致能警報信號
i8092_ALARM_DISABLE	除能警報信號
i8092_INPOS_ENABLE	致能伺服定位輸入信號
i8092_INPOS_DISABLE	除能伺服定位輸入信號
i8092_ALARM_LEVEL	設定警報信號邏輯觸發準位
i8092_INPOS_LEVEL	設定伺服定位輸入信號邏輯準位
i8092_INnSTOP_ENABLE	致能輸入驅動停止信號
i8092_INnSTOP_DISABLE	除能輸入驅動停止信號
i8092_IN3_LEVEL	設定 IN3 輸入信號，觸發準位是高或低
i8092_EXTDRV_DISABLE	設定外部信號除能
i8092_EXTDRV_CDRIVE	設定外部信號連續脈波驅動模式
i8092_EXTDRV_FDRIVE	設定外部信號固定脈波驅動模式
i8092_EXTDRV_MANUAL	設定外部信號手動脈波驅動模式
i8092_SET_OUT1	設定外部 Output 1 狀態
i8092_DIGITAL_FILTER	設定數位濾波
i8092_SERVO_ON	設定 Servo ON
i8092_SERVO_OFF	設定 Servo OFF

i8092_LIMITSTOP_MODE

Format: **void** i8092_LIMITSTOP_MODE(BYTE *cardNo*, WORD *axis*, WORD *nType*)

Function: 是控制碰觸硬體極限信號停止模式。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考表 2-1)。
 nType *nType* = 0: 緊急停止，*nType* = 1: 減速停止。

Example: *//設定碰觸硬體極限信號減速停止*
 i8092_LIMITSTOP_MODE(1, 0x3, 1);

i8092_ENCODER_MODE

Format: **void** i8092_ENCODER_MODE(BYTE *cardNo*, WORD *axis*, WORD *nMode*)

Function: 設定編碼器輸入信號模式(*nECA*/*PPIN* 和 *nECB*/*PMIN*)。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考表 2-1)。
 nMode *nMode* = 0: *CW/CCW*，*nMode* = 1: *Pulse/Direction*。

Example: *//設定所有軸編碼器輸入信號模式為 CW/CCW*
 i8092_ENCODER_MODE(1, 0xf, 0);

i8092_ENCODER_DEVISION

Format: `void i8092_ENCODER_DEVISION(BYTE cardNo, WORD axis, WORD nMode)`

Function: 設定A/B相編碼器輸入除頻，Up / down 計數器不能使用。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考表 2-1)。
nMode nMode = 0: 1/1，nMode = 1: 1/2，
nMode = 2: 1/4，nMode = 3: 沒作用。

Example: `//設定所有軸編碼器輸入除頻為 1/1`
`i8092_ENCODER_DEVISION(1, 0xf, 0);`

i8092_ALARM_ENABLE i8092_ALARM_DISABLE

Format: `void i8092_ALARM_ENABLE(BYTE cardNo, WORD axis)`
`void i8092_ALARM_DISABLE(BYTE cardNo, WORD axis)`

Function: 致能/除能伺服警報輸入信號。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考表 2-1)。

Example: `//設定所有軸伺服警報信號致能`
`i8092_ALM_ENABLE(1, 0xf);`

i8092_INPOS_ENABLE i8092_INPOS_DISABLE

Format: **void** i8092_INPOS_ENABLE(BYTE *cardNo*, WORD *axis*)
 void i8092_INPOS_DISABLE(BYTE *cardNo*, WORD *axis*)

Function: 致能/除能伺服定位輸入信號。

Parameters: *cardNo* 卡號。
 axis 是指定軸號碼(參考表 2-1)。

Example: *//設定所有軸伺服定位信號致能*
 i8092_INPOS_ENABLE(1, 0xf, 1);

i8092_ALARM_LEVEL

Format: **void** i8092_ALARM_LEVEL(BYTE *cardNo*, WORD *axis*,
 WORD *nLevel*)

Function: 設定輸入警報信號觸發準位。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考表 2-1)。
 nLevel nLevel = 0: 低準位觸發，nLevel = 1: 高準位觸發。

Example: *//設定所有軸警報信號是高準位觸發*
 i8092_ALARM_LEVEL(1, 0x3, 1);

i8092_INPOS_LEVEL

Format: **void** i8092_INPOS_LEVEL(BYTE *cardNo*, WORD *axis*, WORD *nLevel*)

Function: 設定伺服定位輸入信號邏輯準位。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考表 2-1)。
 nLevel *nLevel* = 0: 低準位觸發，*nLevel* = 1: 高準位觸發。

Example: *//設定 XY 軸伺服定位輸入信號是高準位觸發*
 i8092_INPOS_LEVEL(1, 0x3, 1);

i8092_INnSTOP_ENABLE i8092_INnSTOP_DISABLE

Format: **void** i8092_INnSTOP_ENABLE(BYTE *cardNo*, WORD *axis*, WORD *nINE*)
 void i8092_INnSTOP_DISABLE(BYTE *cardNo*, WORD *axis*, WORD *nINE*)

Function: 設定輸入驅動停止信號是致能或除能。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考表 2-1)。
 nINE nINE = 0: IN0 , nINE = 1: IN1 , nINE = 2: IN2 , nINE = 3:
 IN3 。
 (WR1/D1 、 D3 、 D5 、 D7)

Example: i8092_INnSTOP_ENABLE(1, 0x3, 1);

i8092_IN3_LEVEL

Format: **void** i8092_IN3_LEVEL(BYTE *cardNo*, WORD *axis*, WORD *nLevel*)

Function: 設定IN3信號的邏輯準位。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考表 2-1)。
 nLevel nLevel = 0: 低準位觸發 , nLevel = 1: 高準位觸發。

Example: i8092_IN3_LEVEL(1, 0x3, 0);

i8092_EXTDRV_DISABLE

Format: **void** i8092_EXTDRV_DISABLE(BYTE *cardNo*, WORD *axis*)

Function: 設定外部驅動信號除能。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考表 2-1)。

Example: *//設定所有軸外部驅動信號除能*
 i8092_EXTDRV_DISABLE(1, 0xf);

i8092_EXTDRV_CDRIVE

Format: **void** i8092_EXTDRV_CDRIVE(BYTE *cardNo*, WORD *axis*)

Function: 設定外部驅動信號為連續脈波驅動模式。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考表 2-1)。

Example: *//設定 X、Y 軸外部驅動信號為連續脈波驅動模式*
 i8092_EXTDRV_CDRIVE(1, 0x3)

i8092_EXTDRV_FDRIVE

Format: **void** i8092_EXTDRV_FDRIVE(BYTE *cardNo*, WORD *axis*)

Function: 設定外部驅動信號為固定脈波驅動模式。

Parameters: *cardNo* 是板號。
 axis 是指定軸號碼(參考表 2-1)。

Example: //設定 X、Y 軸外部驅動信號為固定脈波驅動模式
 i8092_EXTDRV_FDRV(1, 0x3)

i8092_EXDRV_MANUAL

Format: **void** i8092_EXDRIVING_MANUAL(BYTE *cardNo*, WORD *axis*)

Function: 設定外部驅動信號為手動脈波驅動模式。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考表 2-1)。

Example: //設定所有軸外部驅動信號為手動脈波驅動模式
 i8092_EXTDRV_MANUAL(1, 0xf)

註： 當使用 i8092_EXDRV_MANUAL 函式，手動脈波驅動模式變成有效的。使用者能使用手輪接收 A/B 相信號。

i8092_SET_OUT0

Format: `void i8092_SET_OUT0(BYTE cardNo, WORD axis, WORD nLevel)`

Function: 設定 OUTPUT 0 信號輸出。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考表 2-1)。
nLevel *nLevel* = 0:Off, *nLevel* = 1:On。

Example: `//設定 X 軸 OUTPUT0 信號為 On
i8092_SET_OUT0(1, 0x1, 1);`

i8092_SET_OUT1

Format: `void i8092_SET_OUT1(BYTE cardNo, WORD axis, WORD nLevel)`

Function: 設定 OUTPUT 1 信號的觸發準位。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考表 2-1)。
nLevel *nLevel* = 0:Off, *nLevel* = 1:On。

Example: `//設定 X 軸 OUTPUT1 信號為 On
i8092_SET_OUT1(1, 0x1, 1);`

i8092_DIGITAL_FILTER

Format: `void i8092_DIGITAL_FILTER(BYTE cardNo, WORD axis, WORD FEn, WORD FLn)`

Function: 設定輸入信號數位濾波。

Parameters: *cardNo* 卡號。
axis 指定軸號碼(參考表 2-1)。
FEn 致能信號濾波暫存器

Table0-12 致能信號濾波暫存器

定義位元	致能信號濾波
WR6/D8 (FE0)	EMGN, nLMTP, nLMTM, nINO, nIN1
WR6/D9 (FE1)	nIN2
WR6/D10 (FE2)	nINPOS, nALARM
WR6/D11 (FE3)	nEXPP, nEXPM, EXPLSN
WR6/D12 (FE4)	nIN3

FLn 固定的濾波時間

Table0-13 濾波時間選擇

WR6/D15~13 (FL2~0)	可濾除最大雜訊頻寬	輸入信號延遲時間
0	1.75 μ SEC	2 μ SEC
1	224 μ SEC	256 μ SEC
2	448 μ EC	512 μ SEC
3	896 μ SEC	1.024mSEC
4	1.792mSEC	2.048mSEC
5	3.584mSEC	4.096mSEC
6	7.168mSEC	8.192mSEC
7	14.336mSEC	16.384mSEC

Example: `//設定 IN2 輸入信號延遲時間 = 256 μ Sec
i8092_DIGITAL_FILTER(1, 0xf, 0x0200, 0x4000);`

i8092_SERVO_ON

Format: **void** i8092_SERVO_ON(BYTE *cardNo*, WORD *axis*)

Function: 此函式是使用 I8092_SET_OUTPUT1 函式，去設定伺服 On 的狀態。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考表 2-1)。

Example: *// 設定 XY 軸 Servo-On*
 i8092_SERVO_ON(1, 0x3)

i8092_SERVO_OFF

Format: **void** i8092_SERVO_OFF(BYTE *cardNo*, WORD *axis*)

Function: 這個函式是使用 I8092_SET_OUTPUT1 函式，去設定伺服 Off 的狀態。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考表 2-1)。

Example: *// 設定 XY 軸 Servo-Off*
 i8092_SERVO_OFF(1, 0x3)

A.8.9 中斷函式

Table0-14 中斷函式表

函式名稱	敘述
i8092_BPINT_ENABLE	致能位元補間中斷。
i8092_BPINT_DISABLE	除能位元補間中斷。
i8092_CIINT_ENABLE	致能連續補間中斷。
i8092_CIINT_DISABLE	除能連續補間中斷。
i8092_INTFACTOR_ENABLE	致能各中斷條件因子。
i8092_INTFACTOR_DISABLE	除能各中斷條件因子。

i8092_BPINT_ENABLE
i8092_BPINT_DISABLE

Format: void i8092_BPINT_ENABLE(BYTE *cardNo*)
 void i8092_BPINT_DISABLE(BYTE *cardNo*)

Function: 致能/除能位元補間中斷。

Parameters: *cardNo* 卡號。

Example: i8092_BPINT_DISABLE(1);

i8092_CIINT_ENABLE i8092_CIINT_DISABLE

Format: void i8092_CIINT_ENABLE(BYTE *cardNo*)
 void i8092_CIINT_DISABLE(BYTE *cardNo*)

Function: 致能/除能連續補間中斷。

Parameters: *cardNo* 卡號。

Example: i8092_CIINT_ENABLE(1);

i8092_INTFACTOR_ENABLE i8092_INTFACTOR_DISABLE

Format: void i8092_INTFACTOR_ENABLE(BYTE *cardNo*, WORD *axis*,
 WORD *nINT*)
 void i8092_INTFACTOR_DISABLE(BYTE *cardNo*, WORD *axis*,
 WORD *nINT*)

Function: 致能/除能各中斷條件因子。

Parameters: *cardNo* 卡號。
 axis 指定軸號碼(參考表 2-1)。
 nINT 是中斷條件因子之編號，如下表所示

nINT	代號	說明
1	P>=C-	中斷發生於位置計數器大於等於負方向比較暫存器的設定值
2	P<C-	中斷發生於位置計數器小於負方向比較暫存器的設定值
3	P>=C+	中斷發生於位置計數器大於等於正方向比較暫存器的設定值
4	P<C+	中斷發生於位置計數器小於正方向比較暫存器的設定值
5	C-END	中斷發生於等速段結束，意即減速段開始時
6	C-STA	中斷發生於等速段開始時，意即加速段開始時
7	D-END	中斷發生於減速段結束時，可視為驅動結束時

Example: i8092_INTFACTOR_ENABLE(1, 0x3);

A.8.10 FRnet 相關函式

Table 0-15 FRnet 函式表

函式名稱	敘述
i8092_FRNET_SA	寫入 FRnet 的數位輸出資料
i8092_FRNET_RA	讀取 FRnet 的數位輸入資料。

i8092_FRNET_SA

Format: **void** i8092_FRNET_SA(BYTE *cardNo*, WORD *wRA*, WORD *data*)

Function: 讀取 FRnet 的數位輸入資料。

Parameters: *cardNo*: 卡號
 wSA: 群組範圍 SA0~SA7
 dara: 16-位元資料

Return: 無

Example: //設定第 1 卡，SA 群組 = 0，16 位元資料為 0xffff。
 i8092MF_FRNET_SA(1, 0, 0xffff);

i8092_FRNET_RA

Format: WORD i8092MF_FRNET_RA(BYTE *cardNo*, WORD *wRA*)

Parameters: *cardNo*: 卡號
 wRA: 群組範圍 RA8~RA15

Return: WORD 16-位元輸入資料

Example: WORD IN_Data;
 IN_Data = i8092_FRNET_RA(1, 8);
 //設定第 1 卡，RA 群組 = 8。

A. 9 i-8092 命令列表

為了進階的使用者，參考如下的命令表，能幫助去開發自己的函式庫，或定義使用在 I8092 的函式。使用者能參考附錄 B 或 MCX312 使用者手冊，得到更詳細的資訊。

A.9.1 資料寫入命令

Table0-16 資料寫入命令

符號	代碼 (cmd)	命令	資料範圍	資料 長度 (Byte)
R	00h	設定 Range	8,000,000~16,000	4
K	01h	設定加速度率 Jerk	0 ~ 65,535	2
A	02h	設定加速度	1 ~ 8,000	2
D	03h	設定減速度	1 ~ 8,000	2
SV	04h	設定初始速度	1 ~ 8,000	2
V	05h	設定驅動速度	1 ~ 8,000	2
P	06h	設定輸出脈波	0~+2 ²⁸	4
FP	06h	設定補間的終點	-2 ³¹ ~+2 ³¹	4
DP	07h	設定手動減速點	0 ~ 65,535	2
C	08h	設定圓心	-2 ³¹ ~+2 ³¹	4
LP	09h	設定邏輯位置計數器	-2 ³¹ ~+2 ³¹	4
EP	0Ah	設定編碼器位置計數器	-2 ³¹ ~+2 ³¹	4
CP	0Bh	設定 COMP+暫存器	-2 ³¹ ~+2 ³¹	4
CM	0Ch	設定 COMP-暫存器	-2 ³¹ ~+2 ³¹	4
AO	0Dh	設定補正脈波	0~65535	2

A.9.2 資料讀取命令

資料讀取命令

符號	代碼 (cmd)	命令	資料範圍	資料 長度 (Byte)
LP	10h	讀取邏輯位置計數器	$-2^{31} \sim +2^{31}$	4
EP	11h	讀取編碼器位置計數器	$-2^{31} \sim +2^{31}$	4
CV	12h	讀取目前驅動速度	1 ~ 8,000	2
CA	13h	讀取目前加減速	1 ~ 8,000	2

A.9.3 驅動命令

驅動命令

代碼 (cmd)	命令
20h	固定脈波正方向驅動
21h	固定脈波負方向驅動
22h	連續脈波正方向驅動
23h	連續脈波負方向驅動
24h	驅動暫停
25h	驅動暫停解除
26h	減速停止
27h	緊急停止

A.9.4 補間命令

補間命令

代碼 (cmd)	命令
30h	兩軸直線補間
32h	CW 圓形補間
33h	CCW 圓形補間
34h	兩軸位元圖案補間
36h	BP 暫存器寫入致能
37h	BP 暫存器寫入致能
38h	BP 資料堆疊
39h	BP 資料清除
3Ah	單步補間
3Bh	自動減速致能
3Ch	自動減速除能
3Dh	補間中斷清除

A.9.5 其他命令

其他命令

代碼 (cmd)	命令
62h	執行自動歸原點
65h	同步運動作用
0Fh	各軸選擇開關

附錄 B (MCX312 Register)

附錄 B 內容節錄自 "MCX312 User's Manual.pdf" NOVA electronics 。

B.1 Command Register: WR0

Command register is used for the axis assignment and command registration for each axis in MCX312. The register consists of the bit for axis assignment, bit for setting command code, and bit for command resetting.

After the axis assignment and command code have been written to the register, this command will be executed immediately. The data such as drive speed setting and data writing command must be written to registers WR6 and WR7 first. Otherwise, when the reading command is engaged, the data will be written and set, through IC internal circuit, to registers RR6 and RR7.

When using the 8-bit data bus, the user should write data into the high word byte (H), then low word byte (L).

It requires 250 nSEC (maximum) to access the command code when CLK=16MHz. The input signal BUSYN is on the Low level at this moment. Please don't write the next command into WR0 before BUSYN return to the Hi level.

WR0

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
RESET	0	0	0	0	0	Y	X	0	0						

Axis Assignment Command Code

D5 ~ 0 Command code setting Please refer to chapter 5 and the chapters following for further description of command codes.

D9 ~ 8 Axis assignment When the bits of the axis are set to 1, the axis is assigned. The assignment is not limited only for one axis, but for multi-axes simultaneously. It is possible to write the same parameters also. However, the data reading is only for one assigned axis. Whenever the interpolation is commanded, the bits of the assigned axis (axes) should be set 0.

D15 RESET IC command resetting When this bit is set to 1, but others are 0, the IC will be reset after command writing. After command writing, the BUSYN signal will be on the Low level within 875 nSEC (When CLK=16 MHz) maximum.

When 8-bit data bus is used, the reset is activated when the command (80h) is written to register WR0H.

RESET bit should be set to 0 when the other commands are written.

B.2 Mode Register1: WR1

Each axis is with mode register WR1. The axis specified by NOP command or the condition before decides which axis' s register will be written. The register consists of the bit for setting enable / disable and enable logical levels of input signal STOP2~STOP0 (decelerating stop / sudden stop during the driving) and bit for occurring the interrupt enable / disable. Once SP2~SP0 are active, when the fixed / continuous driving starts, and also when STOP signal becomes the setting logical level, the decelerating stop will be performed during the acceleration / deceleration driving and the sudden stop will be performed during the constant speed driving.

WR1

D15	D14	D13	D12 ^H	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
D-END	C-STA	C-END	P \geq C+	P<C+	P<C-	P \geq C-	SMOD	EPINV	EPCLR	SP2-E	SP2-L	SP1-E	SP1-L	SP0-E	SP0-L
Interrupt Enable/Disable								Driving Stop Input Signal							

- D5,3,1 SPm-E** The bit for setting enable / disable of driving stop input signal STOPm 0: disable, 1: enable
- D4,2,0 SPm-L** The bit for setting enable logical levels for input signal STOPm 0: stop on the Low level, 1:stop on the Hi level
- D6 EPCLR** When driving stops triggered by the nSTOP2 signal, the real position counter is cleared. When the nSTOP2 signal is changed to the Active level while this bit is set to 1, the driving stops and the real position counter (EP) is cleared. The WR1/D5(SP2-E) bit must be set to 1 and the Enable level must be set in the WR1/D4(SP2-L) bit.
- D7 EPINV** Reverse increase / decrease of real position counter.

D7 (EPINV)	Input pulse mode	Increase / Decrease of real position counter
0	A / B -phase mode	Count up when A -phase is advancing Count down when B -phase is advancing
	Up-Down pulse mode	Count up when PPIN pulse input Count down when PMIN pulse input
1	A / B -phase mode	Count up when B -phase is advancing Count down when A -phase is advancing
	Up-Down pulse mode	Count up when PMIN pulse input Count down when PPIN pulse input

D8 SM0D Setting for prioritizing to reach specified drive speed during S curve acceleration / deceleration driving. 1: enable
For the following bits, the interrupt is set: 1: enable, 0: disable

- D9 P ≥ C-** Interrupt occurs when the value of logical / real position counter is larger than or equal to that of COMP- register
- D10 P < C-** Interrupt occurs when the value of logical / real position counter is smaller than that of COMP- register
- D11 P < C+** Interrupt occurs when the value of logical / real position counter is smaller than that of COMP+ register
- D12 P ≥ C+** Interrupt occurs when the value of logical / real position counter is larger than or equal to that of COMP+ register
- D13 C-END** Interrupt occurs at the start of the constant speed drive during an acceleration / deceleration driving
- D14 C-STA** Interrupt occurs at the end of the constant speed drive during an acceleration / deceleration driving
- D15 D-END** Interrupt occurs when the driving is finished

D15~D0 will be set to 0 while resetting.

B.3 Mode Register2: WR2

Each axis is with mode register WR2. The axis specified by NOP command or the condition before decides which axis' s register will be written.

WR2 can be used for setting: (1). external limit inputs, (2). driving pulse types, (3). encoder signal types, and (4). the feedback signals from servo drivers.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
INP-E	INP-L	ALM-E	ALM-L	PIND1	PIND0	PINMD	DIR-L	PLS-L	PLSMD	CMPSL	HLMT-	HLMT+	LMTMD	SLMT-	SLMT+

- D0 SLMT+** Enable / disable setting for COMP+ register which is used as the + direction software limit 1: enable, 0: disable
Once it is enabled during the + direction driving, if the value of logical / real position counter is larger than that of COMP+, the decelerating stop will be performed. The D0 (SLMT+) bit of register RR2 will become 1. Under this situation, further written + direction driving commands will not be executed.
Note: When a position counter variable ring is used, a software over run limit cannot be used.
- D1 SLMT-** Enable / disable setting for COMP- register which is used as the - direction software limit 1: enable, 0: disable
Once it is enabled during the - direction driving, if the value of logical / real position counter is smaller than that of COMP-, the decelerating stop will be performed. The D1 (SLMT-) bit of register RR2 will become 1. Under this situation, further written - direction driving commands will not be executed.
- D2 LMTMD** The bit for controlling stop type when the hardware limits are active
(nLMTP and nLMTM input signals)
0: sudden stop, 1: decelerating stop
- D3 HLMT+** Setting the logical level of + direction limit input signal (nLMTP) 0: active on the Low level, 1: active on the Hi level
- D4 HLMT-** Setting the logical level of - direction limit input signal (nLMTM) 0: active on the Low level, 1: active on the Hi level
- D5 CMPSL** Setting if real position counter or logical position counter is going to be compared with COMP +/- register
0: logical position counter, 1 : real position counter
- D6 PLSMD** Setting output pulse type 0: independent 2-pulse type, 1: 1-pulse 1-direction type

When independent 2-pulse type is engaged, + direction pulses are output through the output signal nPP/PLS, and – direction pulses through nPM/DIR.
 When 1-pulse 1-direction type is engaged, + and – directions pulses are output through the output signal nPP/PLS, and nPM/DIR is for direction signals.
 [Note] Please refer to Chapter 13.2 and 13.3 for the output timing of pulse signal (nPLS) and direction signal (nDIR) when 1-pulse 1-direction type is engaged.

D7 PLS-L Setting logical level of driving pulses 0: positive logical level, 1: negative logical level

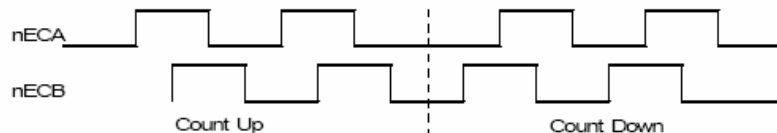


D8 DIR-L Setting logical level of the direction (nPM/DIR) output signal for 1-pulse mode DIR-L

D8 (DIR-L)	+ direction	– direction
0	Low	Hi
1	Hi	Low

D9 PINMD Setting the type of encoder input signals (nECA/PPIN and nECB/PMIN)

0: quadrature pulse input type 1: Up / Down pulse input type
 Real position counter will count up or down when encoder input signal is triggered. When quadrature pulse input type is engaged, the “count up” will happen if the positive logical level pulses are input to phase A; the “count down” will happen if the positive logical level pulses are input to phase B. So, it will count up and down when these 2 signals go up (↑) and down (↓).



When Up / Down pulse input type is engaged, nECA/PPIN is for “count up” input, and nECB/PMIN is for “count down” input. So, it will count up when the positive pulses go up (↑).

D11,10 PIND1,0 The division setting for quadrature encoder input.

D11	D10	Division
0	0	1/1
0	1	1/2
1	0	1/4

1	1	Invalid
---	---	---------

Up / down pulse input is not available.

- D12** **ALM-L** Setting active level of input signal nALARM 0: active on the Low level, 1: active on the Hi level
- D13** **ALM-E** Setting enable / disable of servo alarm input signal nALARM 0: disable, 1: enable
When it is enabled, MCX312 will check the input signal. If it is active, D14 (ALARM) bit of RR2 register will become 1. The driving stops.
- D14** **INP-L** Setting logical level of nINPOS input signal 0: active on the Low level, 1: active on the Hi level
- D15** **INP-E** Setting enable/disable of in-position input signal nINPOS from servo driver 0: disable, 1: enable
When it is enabled, bit n-DRV of RR0 (main status) register does not return to 0 until nINPOS signal is active after the driving is finished.

D15~D0 will be set to 0 while resetting.

B.4 Mode Register3: WR3

Each axis is with mode register WR3. The axis specified by NOP command or the condition before decides which axis' s register will be written. WR3 can be used for manual deceleration, individual deceleration, S-curve acceleration / deceleration, the setting of external operation mode, the setting of input signal filter, and so on.

WR3

D15	D14	D13	D12 ^H	D11	D10	D9	D8	D7	D6	D5	D4 ^L	D3	D2	D1	D0
FL2	FL1	FL0	FE4	FE3	FE2	FE1	FE0	OUTSL	VRING	AVTRI	EXOP1	EXOP0	SACC	DSNDE	MANLD

D0 MANLD Setting manual / automatic deceleration for the fixed acceleration / deceleration driving

0: automatic deceleration, 1: manual deceleration The decelerating point should be set if the manual deceleration mode is engaged.

D1 DSNDE Setting decelerating rate which is in accordance with the rate of the acceleration or an individual decelerating rate

0: acceleration, 1: deceleration

When 0 is set, acceleration value is used as acceleration / deceleration during acceleration / deceleration driving. When 1 is set, acceleration value is used during acceleration driving and deceleration value is used during deceleration driving. 1 should be set for non-symmetrical trapezoidal acceleration / deceleration driving.

D2 SACC Setting trapezoidal driving / S-curve driving

0: trapezoidal driving, 1: S-curve driving

Before S-curve driving is engaged, jerk (K) should be set.

D4,3 EXOP1,0 Setting the external input signals (nEXPP, nEXPM) for driving

D4 (EXOP1)	D3 (EXOP0)	
0	0	external signals disabled
0	1	continuous driving mode
1	0	fixed driving mode
1	1	manual pulsar mode

When the continuous driving mode is engaged, the + direction drive pulses will be output continuously once the nEXPP signal is on the Low level; the – direction pulses will be output continuously once the nEXPM signal is on the Low level. When the fixed driving mode is engaged, the + direction fixed driving starts once the nEXPP signal is falling to the Low level from the Hi level; the – direction fixed driving starts once the nEXPM signal is falling to the Low level from the Hi level. In manual pulsar mode, fixed driving in the + direction is activated at ↑ of the nEXPP signal when the nEXPM signal is at the Low level. The fixed driving is activated at ↓ of the nEXPP signal when the nEXPM signal is at the Low level.

- D5 AVTRI Prevent triangle waveforms during fixed driving at the trapezoidal acceleration / deceleration. 0: disable, 1: enable.
[Note] WR3/D5 bit should be reset to 0 when continuous driving is performed after fixed driving.
- D6 VRING Enable the variable ring function of logical position and real position counter. 0: disable, 1: enable.
- D7 OUTSL Driving status outputting or used as general purpose output signals (nOUT7~0)

0: nOUT7~0: general purpose output

The setting of each bit in WR4 register will be output to nOUT7~0.

1: nOUT7~0: driving status output (see the table below)

Signal Name	Output Description
nOUT0/ACASND	When acceleration or deceleration of S curve acceleration / deceleration increases, the level becomes Hi.
nOUT1/ACDSND	When acceleration or deceleration of S curve acceleration / deceleration decreases, the level becomes Hi.
nOUT2/CMPP	Hi: if logical / real position counter \geq COMP+ register Low : if logical / real position counter < COMP+ register
nOUT3/CPMP	Hi: if logical / real position counter < COMP- register Low: if logical / real position counter \geq COMP- register
nOUT4/DRIVE	When drive pulse is outputting, the level becomes Hi.
nOUT5/ASND	When the driving command is engaged, the level becomes Hi once the driving status is in acceleration.
nOUT6/CNST	When the driving command is engaged, the level becomes Hi once the driving status is in constant speed driving.
nOUT7/DSND	When the driving command is engaged, the level becomes Hi once the driving status is in deceleration.

D12~8 FE4~0 Set whether the input signal filter function enables or signal passes through. 0: through, 1: enable.

Specification bit	Filter Enable signal
D8 FE0	EMGN ^{*2} , nLMTP, nLMTM, nSTOP0, nSTOP1
D9 FE1	nSTOP2
D10 FE2	nINPOS, nALARM
D11 FE3	nEXPP, nEXPM
D12 FE4	nIN0, nIN1, nIN2, nIN3, nIN4, nIN5

*2: The EMGN signal is set using the D8 bit of the WR3 register of the X axis.

D15~13 FL2~0 Set a time constant of the filter.

FL2 ~ 0	Removable maximum noise width	Input signal delay time
0	1.75 μ SEC	2 μ SEC
1	224 μ SEC	256 μ SEC
2	448 μ SEC	512 μ SEC
3	896 μ SEC	1.024mSEC
4	1.792mSEC	2.048mSEC
5	3.584mSEC	4.096mSEC
6	7.168mSEC	8.192mSEC
7	14.336mSEC	16.384mSEC

D15~D0 will be set to 0 while resetting.

B.5 Output Register: WR4

This register is used for setting the general purpose output signals nOUT7~0. This 16-bit register locates 8 output signals of each axis. It can be also used as a 16-bit general purpose output. It is Low level output when the bit is set 0, and Hi level output when the bit is set 1.

D15	D14	D13	D12 ^H	D11	D10	D9	D8	D7	D6	D5	D4 ^L	D3	D2	D1	D0
YOUT7	YOUT6	YOUT5	YOUT4	YOUT3	YOUT2	YOUT1	YOUT0	XOUT7	XOUT6	XOUT5	XOUT4	XOUT3	XOUT2	XOUT1	XOUT0

D15~D0 will be set to 0 while resetting, and nOUT7~0 signals become Low level.

4.8 Interpolation Mode Register: WR5

This register is used for setting constant vector speed mode, multichip interpolation mode, 1-step interpolation mode and interrupt during the interpolation.

D15	D14	D13	D12 ^H	D11	D10	D9	D8	D7	D6	D5	D4 ^L	D3	D2	D1	D0
BPINT	CIINT	CMPLS	EXPLS	MLT1	MLT0	LSPD1	LSPD0	0	0	0	0	0	0	0	0
Interrupt		Step		multichip		Constant Vector Speed									

D9,8 LSPD1,0 Constant vector speed mode setting of interpolation driving
D15~D0 will be set to 0 while resetting.

D9	D8	Code (Binary)
0	0	constant vector speed invalid
0	1	2-axis constant vector speed
1	0	(setting not available)
1	1	(setting not available)

When 2-axis constant vector speed mode is engaged, the user should set the range (R) of the Y axis to be 1.414 times of the range (R) of the X axis.

D11,10
MLT1,0

Multichip interpolation mode setting

D11	D10	Setting
0 0 1	0 1	disable multichip interpolation main chip sub chip X,
1	0 1	Y (Both X and Y axes use) sub chip (Only X axis)

- D12 EXPLS** When it is 1, the external (MPLS) controlled single step interpolation mode is engaged.
- D13 CMPLS** When it is 1, the command controlled single step interpolation mode is engaged.
- D14 CIINT** Interrupt enable / disable setting during interpolation 0: disable
1: enable
- D15 BPINT** Interrupt enable / disable setting during bit-pattern interpolation
0: disable 1: enable

B.7 Data Register: WR6/WR7

Data registers are used for setting the written command data. The low-word data-writing 16-bit (WD15~WD0) is for register RR6 setting, and the high-word data-writing 16-bit (WD31~WD16) is for register RR7 setting.

WR6

D15	D14	D13	D12 ^H	D11	D10	D9	D8	D7	D6	D5	D4 ^L	D3	D2	D1	D0
WD15	WD14	WD13	WD12	WD11	WD10	WD9	WD8	WD7	WD6	WD5	WD4	WD3	WD2	WD1	WD0

WR7

D15	D14	D13	D12 ^H	D11	D10	D9	D8	D7	D6	D5	D4 ^L	D3	D2	D1	D0
WD31	WD30	WD29	WD28	WD27	WD26	WD25	WD24	WD23	WD22	WD21	WD20	WD19	WD18	WD17	WD16

The user can write command data with a designated data length into the write register. It does not matter to write WR6 or WR7 first (when 8-bit data bus is used, the registers are WR6L, WR6H, WR7L and WR7H).

The written data is binary formatted; 2' complement is for negatives. For command data, the user should use designated data length. For instance, the circular interpolation of the finish point should be set by a signed 32-bit format with the data length of 4 bytes, although its calculatable data range is from -8,388,608 to +8,388,607 signed 24-bit format.

The contents of WR6 and WR7 are unknown while resetting.

B.8 Main Status Register: RR0

This register is used for displaying the driving and error status of each axis. It also displays interpolation driving, ready signal for continuous interpolation, quadrant of circular interpolation and stack counter of bit pattern interpolation.

D15	D14	D13	D12 ^H	D11	D10	D9	D8	D7	D6	D5	D4 ^L	D3	D2	D1	D0
-	BPSC1	BPSC0	ZONE2	ZONE1	ZONE0	CNEXT	I-DRV	0	0	Y-ERR	X-ERR	0	0	Y-DRV	X-DRV

Error Status of Each Axis Driving Status of Each Axis

D1, 0 n-DRV Displaying driving status of each axis When the bit is 1, the axis is an outputting drive pulse. ; when the bit is 0, the driving of the axis is finished. Once the in-position input signal nINPOS for servomotor is active, nINPOS will return to 0 after the drive pulse output is finished.

D5, 4 n-ERR Displaying error status of each axis If any of the error bits (D6~D0) of each axis's RR2 register and any of the error-finish bits (D15~D12) of each axis' s RR1 register becomes 1, this bit will become 1.

D8 I-DRV Displaying interpolation driving status While the interpolation drive pulses are outputting, the bit is 1.

D9 CNEXT Displaying the possibility of continuous interpolation data writing When the bit is 1, it is ready for inputting parameters for next node and also ready for writing interpolation command data.

D12 ~ 10 ZONEm Displaying the quadrant of the current position in circular interpolation

D12	D11	D10	Quadrant
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

D14,13 BPSC1,0 In bit pattern interpolation driving, it displays the value of the stack counter (SC).

D14	D13	Stack Counter (SC) Value
0	0	0
0	1	1
1	0	2
1	1	3

In bit pattern interpolation driving, when SC = 3, it shows the stack is full. When SC = 2, there is one word (16-bit) space for each axis. When SC = 1, there is a 2-word (16-bit × 2) for each axis. When SC = 0, it shows all the stacks are empty, and the bit-pattern interpolation is finished.

B.9 Status Register 1: RR1

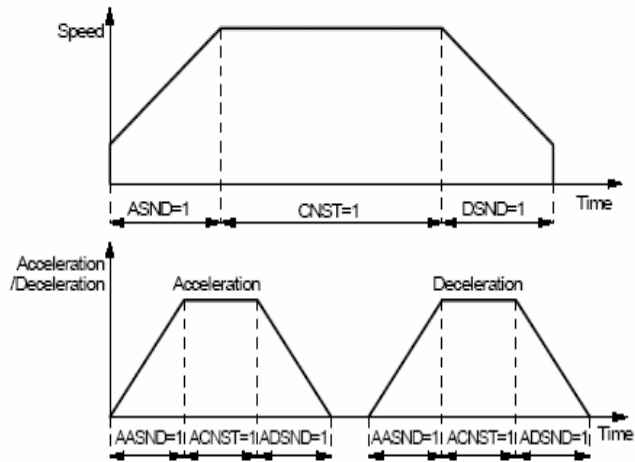
Each axis is with status register 1. The axis specified by NOP command or the condition before decides which axis' s register will be read.

The register can display the comparison result between logical / real position counter and COMP +/- , the acceleration status of acceleration / deceleration driving, jerk of S-curve acceleration / deceleration and the status of driving finishing.

D15	D14	D13	D12	H				D11	D10	D9	D8	L				D7	D6	D5	D4	D3	D2	D1	D0
EMG	ALARM	LMT-	LMT+	—	STOP2	STOP1	STOP0	ADSND	ACNST	AASND	DSND	CNST	ASND	CMP-	CMP+								

Status of Driving Finishing

- D0 CMP+** Displaying the comparison result between logical / real position counter and COMP+ register
1: logical / real position counter \geq COMP+ register
0: logical / real position counter $<$ COMP+ register
- D1 CMP-** Displaying the comparison result between logical / real position counter and COMP- register
1: logical / real position counter $<$ COMP- register
0: logical / real position counter \geq COMP- register
- D2 ASND** It becomes 1 when in acceleration.
- D3 CNST** It becomes 1 when in constant speed driving.
- D4 DSND** It becomes 1 when in deceleration.
- D5 AASND** In S-curve, it becomes 1 when acceleration / deceleration increases.
- D6 ACNST** In S-curve, it becomes 1 when acceleration / deceleration keeps constant speed.
- D7 ADSND** In S-curve, it becomes 1 when acceleration / deceleration decreases.
- D10~8 STOP2~0** If the driving is stopped by one of external decelerating stop signals (nSTOP2 ~ 0), it will become 1.
- D12 LMT+** If the driving is stopped by +direction limit signal (nLMTP), it will become 1.
- D13 LMT-** If the driving is stopped by -direction limit signal (nLMTM), it will become 1.
- D14 ALARM** If the driving is stopped by nALARM from servo drivers, it will become 1.
- D15 EMG** If the driving is stopped by external emergency signal (EMGN), it will become 1.



The Status Bits of Driving Finishing

These bits are keeping the factor information of driving finishing. The factors for driving finishing in fixed driving and continuous driving are shown as follows:

- a. when all the drive pulses are output in fixed driving,
- b. when deceleration stop or sudden stop command is written,
- c. when software limit is enabled, and is active,
- d. when external deceleration signal is enabled, and active,
- e. when external limit switch signals (nLMTP, nLMTM) become active,
- f. when nALARM signal is enabled, and active, and
- g. when EMGN signal is on the Low level.

Above factors “a.” and “b.” can be controlled by the host CPU, and factor “c.” can be confirmed by register RR2 even the driving is finished. As for factors “d.” ~ “g.”, the error status is latched in RR2 until next driving command or a clear command (25h) is written.

After the driving is finished, if the error factor bits D15~D12 become 1, n-ERR bit of main status register RRO will become 1.

Status bit of driving finishing can be cleared when next driving command is written, or when the finishing status clear command (25h) is used.

B.10 Status Register 2: RR2

Each axis is with status register 2. The axis specified by NOP command or the condition before decides which axis' s register will be read.

This register is for reflecting the error information. When an error occurs, the error information bit is set to 1. When one or more of D6 to D0 bits of RR2 register are 1, n-ERR bits of main status register RR0 become 1.

RR2

D15	D14	D13	D12 ^H	D11	D10	D9	D8	D7	D6	D5	D4 ^L	D3	D2	D1	D0
—	—	—	—	—	—	—	—	—	MULT	EMG	ALARM	HLMT ⁻	HLMT ⁺	SLMT ⁻	SLMT ⁺

D0 SLMT+ During the + direction driving, when logical / real position counter \geq COMP+ (COMP+ enabled, and used as software limit)

D1 SLMT- During the - direction driving, when logical / real position counter \leq COMP- (COMP- enabled, and used as software limit)

D2 HLMT+ When external +direction limit signal (nLMTP) is on its active level

D3 HLMT- When external -direction limit signal (nLMTM) is on its active level

D4 ALARM When the alarm signal (nALARM) for servo motor is on its active level

D5 EMG When emergency stop signal (EMGN) becomes Low level.

D6 MULT This bit is only for the X axis of main chip at the multichip interpolation. When an error occurs in any axis of sub chip during multichip interpolation, it will become 1.

In driving, when hardware / software limit is active, the decelerating

stop or sudden stop will be executed. Bit SLMT+ / - will not become 1 during the reverse direction driving.

B.11 Status Register 3: RR3

Each axis is with status register 3. The axis specified by NOP command or the condition before decides which axis' s register will be read.

This register is for reflecting the interrupt factor. When interrupt happens, the bit with the interrupt factor becomes 1. The user should set the interrupt factor through register WR1 to perform the interrupt.

To generate an interrupt, interrupt enable must be set for each factor in the WR1 register.

RR3

H								L							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
—	—	—	—	—	—	—	—	D-END	C-STA	C-END	P \geq C+	P<C+	P<C-	P \geq C-	—

D1 P \geq C- Once the value of logical / real position counter is larger than that of COMP- register D2 P < C- Once the value of logical / real position counter is smaller than that of COMP- register D3 P < C+ Once the value of logical / real position counter is smaller than that of COMP+ register D4 P \geq C+ Once the value of logical / real position counter is larger than that of COMP+ register D5 C-END When the pulse output is finished in the constant speed drive during an acceleration / deceleration driving D6 C-STA When the pulse output is started in the constant speed drive during an acceleration / deceleration driving D7 D-END When the driving is finished When one of the interrupt factors occurs an interrupt, the bit of the register becomes 1, and the interrupt output signal (INTN) will become the Low level. The host CPU will read register RR3 of the interrupted axis, the bit of RR3 will be cleared to 0, and the interrupt signal will return to the non-active level. For a 8-bit data bus, all the bits are cleared when the RR3L register is read.

B.12 Input Register: RR4 / RR5

RR4 and RR5 are used for displaying the input signal status. The bit is 0 if the input is on the Low level; the bit is 1 if the input is on the Hi level. These input signals can be used as general input signal when they are not used as function except for nLMTP/M signal.

RR4

H								L							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X-LM	X-LM+	X-IN5	X-IN4	X-IN3	X-IN2	X-IN1	X-IN0	X-ALM	X-INP	X-EX-	X-EX+	ENG	X-ST2	X-ST1	X-ST0

RR5

D15	D14	D13	D12 ^H	D11	D10	D9	D8	D7	D6	D5	D4 ^L	D3	D2	D1	D0
Y-LM	Y-LM+	Y-IN5	Y-IN4	Y-IN3	Y-IN2	Y-IN1	Y-IN0	Y-ALM	Y-INP	Y-EX-	Y-EX+	—	Y-ST2	Y-ST1	Y-ST0
Bit Name				Input Signal				Bit Name				Input Signal			
n-ST0				n-STOP0				n-IN0				nIN0			
n-ST1				n-STOP1				n-IN1				nIN1			
n-ST2				n-STOP2				n-IN2				nIN2			
EMG				EMGN				n-IN3				nIN3			
n-EX+				nEXPP				n-IN4				nIN4			
n-EX-				nEXPM				n-IN5				nIN5			
n-INP				nINPOS				n-LM+				nLMTP			
n-ALM				nALARM				n-LM-				nLMTM			

B.13 Data-Read Register: RR6 / RR7

According to the data-read command, the data of internal registers will be set into registers RR6 and RR7. The low word 16 bits (D15 ~ D0) is set in RR6 register, and the high word 16 bits (D31 ~ D16) is set in RR7 register for data reading.

RR6

D15	D14	D13	D12 ^H	D11	D10	D9	D8	D7	D6	D5	D4 ^L	D3	D2	D1	D0
RD15	RD14	RD13	RD12	RD11	RD10	RD9	RD8	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0

RR7

D15	D14	D13	D12 ^H	D11	D10	D9	D8	D7	D6	D5	D4 ^L	D3	D2	D1	D0
RD31	RD30	RD29	RD28	RD27	RD26	RD25	RD24	RD23	RD22	RD21	RD20	RD19	RD18	RD17	RD16

The data is binary formatted; 2' s complement is for negatives.