EzCore API使用手冊

(Version 4.3)

EzProg

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

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.

技術支援

如果您有任何使用泓格產品問題，請用以下電子郵件信箱聯繫:

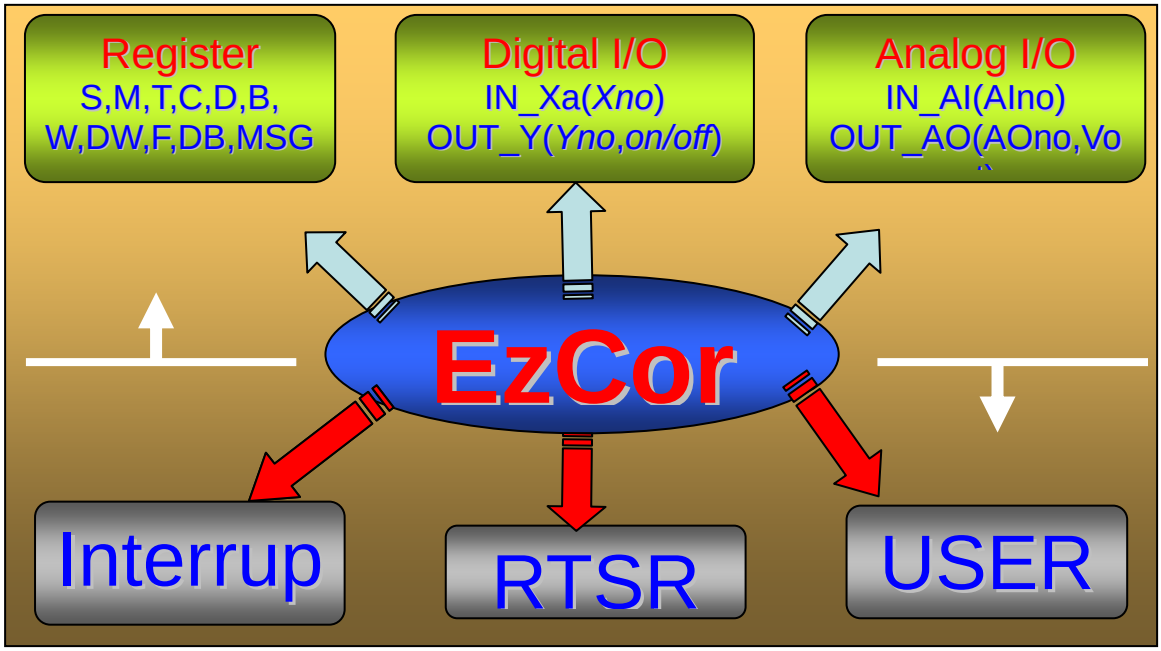
Service@icpdas.com

目錄

1 前言

1.1 使用 EzCore API

EzProg-I 架構出應用平台及 EzCore 提供相關 API(應用程序介面函式)，其主要的架構如下圖:



本手冊針對 EzProg-I 架構出應用平台及 EzCore 提供相關 API(應用程序介面函式)，作詳細使用說明，以提供程式設計人員詳細之應用資訊及使用者方法。

1.2 目前 EzCore 支援模組

目前 EzCore 支援模組如下:
I8K 系列 IO 控制模組
馬達運動控制模組
詳情請參考各控制器的 quick_Start 說明

1.3 EzProg-I IO 與暫存器列表

EzProg-I IO 與暫存器 API 的使用詳細說明，請參考第 2.3 章節。

目前 EzProg-定義的暫存器範圍如下表:

Specification		Register	Register number		Data type	Size	Range
Slot Device Register	Digital Input	X	Local DI:	0 ~ 777	bit	1 bit	true / false
			FRNet DI:	1000 ~ 7777			
	Digital Output	Y	Local DO:	0 ~ 777	bit	1 bit	true / false
			FRNet DO:	1000 ~ 7777			
	Analog Output	AO	Local AO:	0 ~ 511	float	4 bytes	3.4E +/- 38
	Analog Input	AI	Local AI:	0 ~ 511	float	4 bytes	3.4E +/- 38
Software Register	Timer	T	None Retain:	1 ~ 299	bit	1 bit	true / false
	Counter	C	None Retain:	1 ~ 511	bit	1 bit	true / false
			Retain:	512 ~ 1023			
	Flag	M	None Retain:	1 ~ 6999	bit	1 bit	true / false
			Retain:	8192 ~ 15999			
	Step	S	None Retain:	1 ~ 8191	bit	1 bit	true / false
	long integer	D	None Retain:	1 ~ 3599	long integer	4 bytes	–2,147,483,648 to 2,147,483,647
			Retain:	4096 ~ 7999			
	BYTE	B	None Retain:	1 ~ 699	unsigned char	1 byte	0 to 255
			Retain:	1024 ~ 2047			
	WORD	W	None Retain:	1 ~ 1023	unsigned short	2 bytes	0 to 65,535
			Retain:	1024 ~ 1999			
	DWORD	DW	None Retain:	1 ~ 4095	unsigned long	4 bytes	0 to 4,294,967,295
			Retain:	4096 ~ 8191			
	Float	F	None Retain:	1 ~ 1899	float	4 bytes	3.4E +/- 38
			Retain:	2048 ~ 3999			
	Special Type	DB	None Retain:	1 ~ 49			
	Message	MSG	Retain:	1 ~ 249	30 wchar_t	60 bytes	30 unicode char

下表是 EzProg 中已指定用途的暫存器，請按照用途使用:

Register Number	Note
D8000	Muti-language
M16000	HMI:ColorEdit Input Control

2 設定 EzCore 引擎

2.1 初始設定 EzCore 引擎

2.1.1 硬體設備初始化

● **long DEVICE_INITIAL(WORD** *Para1*, **WORD** *Para2*, **WORD** *RunMode*)

功能: 載入 IO 規劃資訊，與輸出預設值，及系統模式規劃。

參數:	***Para1***	0：	**OUTPUT_Clear** 清除所有 DO 與 AO
		1：	**OUTPUT_Now** 載入規劃資料後，立即將 DO 與 AO 的預設值輸出
	Para2	0：	**IO_MODE_Direct** 硬體 DIO 與 AIO 指令立即作動
		1：	**IO_MODE_AutoScan** 硬體 DIO 與 AIO 指令由掃瞄引擎定時更新實體 IO 動作(此模式須先執行 SCAN_ENGINE_START 後 DIO/AIO 才會有反映)
	RunMode	1：	**RUN_PRG_MODE** 以程序模式運行
回應:	**0**		執行成功
	其他值		請參閱附錄一錯誤碼表

範例: 詳情請參考範例程式

```
long ret;
ret=DEVICE_INITIAL(OUTPUT_Now,0,RUN_PRG_MODE);
if (ret == _NO_ERROR)
{
    ret=SCAN_ENGINE_START();
    if (ret == _NO_ERROR)
    {
        SET_M(200,true); //System initial ok
    }
    else
    MessageBox( TEXT("Start ENGINE NG   \n Please check "), TEXT("EzCore
    Engine"), MB_OK|MB_ICONERROR);
}
else
{
    MessageBox( TEXT("Load ALL Device Data NG   \n Please check \n Please
    Rescan IO "), TEXT("EzCore Engine"), MB_OK|MB_ICONERROR);
}
```

2.2 啟動與關閉 EzCore 引擎

2.2.1 控制引擎啟動

- **long SCAN_ENGINE_START()**

功能: 軟體引擎控制程序啟動。

參數: 無

回應:

0	執行成功
其他值	請參閱附錄一錯誤碼表

範例: 請參考 2.1.1 範例程式
詳情請參考範例程式

2.2.2 控制引擎停止

- **long SCAN_ENGINE_STOP()**

功能: 軟體引擎控制程序停止。

參數: 無

回應:

0	執行成功
其他值	請參閱附錄一錯誤碼表

範例: 詳情請參考範例程式

2.3 使用 IO 與暫存器的應用程式庫

EzProg-I 除內建的功能外，還可以依客戶需求選用各種型式的 i8K IO 模組，如數位輸入(DI)、數位輸出(DO)、類比輸入(AI)、類比輸出(AO)、多軸運動控制(Motion)、高速分散式 DIO(FRNET)，而這些模組經適當規劃後(EzConfig 自動規劃)，在 EzCore 中我們就可以用統一的方法，輕易的使用它們，縮短專案規劃的時間，亦可達到設計標準化的目的。

2.3.1 數位及類比控制 DI/O AI/O

2.3.1.1 輸出 DO

- **void OUT_Y(WORD** *DOno*, **bool** *Flag*)

功能: DO Y 點輸出。

參數:

	DOno	**DO 號碼: 0000 ~ 7777(8 進位)**
	Flag	**true** 輸出 ON，**false** 輸出 OFF

回應: 無

範例:

2.3.1.2 讀回 DO (a 接點)狀態

- **bool GET_Ya(WORD** *DOno*)

功能: 取回 DO 輸出 Ya 接點值。

參數:

	DOno	**DO 號碼: 0000 ~ 7777(8 進位)**

回應:

true	**Ya 輸出狀態為 ON**
false	**Ya 輸出狀態為 OFF**

範例:

2.3.1.3 讀回 DO (b 接點)狀態

- **bool GET_Yb(WORD** *DOno*)

功能: 取回 DO 輸出 Yb 接點值。

參數:

	DOno	**DO 號碼: 0000 ~ 7777(8 進位)**

回應:

true	**Yb 狀態為 ON**
false	**Yb 狀態為 OFF**

範例:

2.3.1.4 讀回 DI (a 接點)狀態

- **bool IN_Xa(WORD *DIno*)**

功能: 取回 DI X 輸入 a 接點值。

參數:	*DIno*	DI 號碼: 0000 ~ 7777(8 進位)
回應:	true	Xa 狀態為 ON
	false	Xa 狀態為 OFF

範例:

2.3.1.5 讀回 DI (b 接點)狀態

- **bool IN_Xb(WORD *DIno*)**

功能: 取回 DI X 輸入 b 接點值。

參數:	*DIno*	DI 號碼: 0000 ~ 7777(8 進位)
回應:	true	Xb 狀態為 ON
	false	Xb 狀態為 OFF

範例:

2.3.1.6 輸出 AO

- **void OUT_AO(WORD** *AOno*, **float** *Vout*)

功能: **AO 點輸出。**

參數: ***AOno*** **AO 號碼: 0 ~ 511**
Vout **輸出值**

回應: **無**

範例:

2.3.1.7 讀回 AO 輸出值

- **float GET_AO(WORD** *AOno*)

功能: **取回 AO 輸出值。**

參數: ***AOno*** **AO 號碼: 0 ~ 511**

回應: **輸出值**

範例:

2.3.1.8 讀回 AI 輸入值

- **float IN_AI(WORD** *AIno***)**

功能: 取回 **AI** 輸入值。

參數: ***AIno*** **AI 號碼: 0 ~ 511**

回應: **AI 輸入值**

範例:

2.3.2 計時器功能 Timer

2.3.2.1 設定 Timer 計時器

- **void SET_T(BYTE** *Tno*, **bool** *Flag*, **DWORD** *ms*)

功能: 啟動或關閉計時器。

參數:		
	Tno	**TIMER 號碼: 0 ~ 299**
	Flag	**true 啟動，false 關閉**
	ms	**TIMER 設定時間: 1 ~ 4,294,967,295 ms**

回應: 無

範例:

2.3.2.2 讀回 Timer 倒數計時值

- **DWORD GET_T(BYTE** *Tno*)

功能: 讀回 Timer 倒數計時值。

參數:		
	Tno	**TIMER 號碼: 0 ~ 299**
回應:		**TIMER 倒數計時值: 0 ~ 4,294,967,295 ms**

範例:

2.3.2.3 讀回 Timer (a 接點)狀態

- **bool GET_Ta(BYTE *Tno*)**

功能: 讀回 Timer a 接點狀態。

參數:	***Tno***	**TIMER 號碼: 0 ~ 299**
回應:	**true**	**Ta 狀態為 ON**
	false	**Ta 狀態為 OFF**

範例:

2.3.2.4 讀回 Timer (b 接點)狀態

- **bool GET_Tb(BYTE *Tno*)**

功能: 讀回 Timer b 接點狀態。

參數:	***Tno***	**TIMER 號碼: 0 ~ 299**
回應:	**true**	**Tb 狀態為 ON**
	false	**Tb 狀態為 OFF**

範例:

2.3.3 計數器功能 Counter

2.3.3.1 設定 Counter 計數器

- **void SET_C(WORD** *Cno*, **bool** *Flag*, **DWORD** *COUNT*)

功能: 啟動或下數計數器。

參數:

	Cno	**Counter 號碼: 0 ~ 511、512 ~ 1023 為斷電保持型**
	Flag	**true 啟動，false 下數**
	COUNT	**設定 COUNT 數: 1 ~ 4,294,967,295**

回應: 無

範例:

2.3.3.2 重置 Counter 計數器

- **void RESET_C(WORD** *Cno*)

功能: 重置計數器。

參數:

	Cno	**Counter 號碼: 0 ~ 511、512 ~ 1023 為斷電保持型**

回應: 無

範例:

2.3.3.3 讀回 Counter 倒數計數值

- **DWORD GET_C(WORD *Cno*)**

功能: 讀回 Counter 倒數計數值。

參數: ***Cno*** Counter 號碼: 0 ~ 511、512 ~ 1023 為斷電保持型

回應: Counter 倒數計數值: 0 ~ 4,294,967,295

範例:

2.3.3.4 讀回 Counter (a 接點)狀態

- **bool GET_Ca(WORD *Cno*)**

功能: 讀回 Counter 計數器 a 接點，Ca 是否 ON。

參數: ***Cno*** Counter 號碼: 0 ~ 511、512 ~ 1023 為斷電保持型

回應: **true** Ca 狀態為 ON
false Ca 狀態為 OFF

範例:

2.3.3.5 讀回 Counter (b 接點)狀態

- **bool GET_Cb(WORD *Cno*)**

功能: 查詢 Counter 計數器 b 接點，Cb 是否 ON。

參數:	***Cno***	**Counter 號碼: 0 ~ 511、512 ~ 1023 為斷電保持型**
回應:	**true**	**Cb 狀態為 ON**
	false	**Cb 狀態為 OFF**

範例:

2.3.4 步進程序 Step 功能

2.3.4.1 設定 Step 旗標

- **void SET_S(WORD *Sno*)**

功能: 設定 S 步進旗標 On 狀態。

參數: ***Sno*** S 旗標號碼: 0 ~ 8191

回應: 無

範例:

2.3.4.1 清除 Step 旗標

- **void RST_S(WORD *Sno*)**

功能: 清除 S 步進旗標為 Off 狀態。

參數: ***Sno*** S 旗標號碼: 0 ~ 8191

回應: 無

範例:

2.3.4.2 讀回 Step 旗標狀態

- **bool GET_S(WORD *Sno*)**

功能: 取回 S 旗標 On/Off 狀態。

參數: ***Sno*** S 旗標號碼: 0 ~ 8191

回應: **true** S 狀態為 ON
false S 狀態為 OFF

範例:

2.3.5 軟體旗標功能 M

2.3.5.1 設定 M 旗標值

● **void** SET_M(WORD *Mno*, **bool** *Flag*)

功能: 設定 M 旗標 On/Off 狀態。

參數: *Mno* M 旗標號碼: 0 ~ 8191、8192 ~ 16383 為斷電保持型
M16000 ~ M16383 為系統內定使用
M16000 為 ColorEdit 控件輸入介面切換，
true=控件內部鍵盤。

Flag true ON，false OFF

回應: 無

範例:

2.3.5.2 讀回 M (a 接點)

● **bool** GET_Ma(WORD *Mno*)

功能: 取回 M 旗標 a 接點 On/Off 狀態。

參數: *Mno* M 旗標號碼: 0 ~ 8191、8192 ~ 16383 為斷電保持型

回應: true Ma 狀態為 ON
false Ma 狀態為 OFF

範例:

2.3.5.2 讀回 M (b 接點)

● **bool** GET_Mb(WORD *Mno*)

功能: 取回 M 旗標 b 接點 On/Off 狀態。

參數: *Mno* M 旗標號碼: 0 ~ 8191、8192 ~ 16383 為斷電保持型

回應: true Mb 狀態為 ON
false Mb 狀態為 OFF

範例:

2.3.6 一般暫存器功能 D

2.3.6.1 設定 D 一般暫存器值

● **void SET_D(WORD** *Dno*, **long** *Val*)

功能: 指定 D 暫存器值。

參數: ***Dno*** D 暫存器號碼: 0 ~ 4095、4096 ~ 8191 為斷電保持型
D8000 ~ D8191 為系統內定使用:
D8000 : 0 ~ 7 為八國語系選擇
D8100 : 為 FRAM W/R ERROR

Val 帶符號 32 位元值: -2,147,483,648 ~ 2,147,483,647

回應: 無

範例:

2.3.6.2 讀回 D 一般暫存器值

● **long GET_D(WORD** *DNo*)

功能: 取回 D 暫存器值。

參數: ***DNo*** D 暫存器號碼: 0 ~ 4095、4096 ~ 8191 為斷電保持型

回應: D 暫存器值: -2,147,483,648 ~ 2,147,483,647

範例:

2.3.7 資料暫存器功能 B、W、DW、F

2.3.7.1 設定 B 暫存器值

- **void SET_B(WORD** *Bno*, **BYTE** *data*)

功能: 指定 8 位元資料不帶符號 B 暫存器值。

參數: ***Bno*** B 暫存器號碼: 0 ~ 1023、1024 ~ 2047 為斷電保持型
data 8 位元資料: 0 ~ 255

回應: 無

範例:

2.3.7.2 讀回 B 暫存器值

- **BYTE GET_B(WORD** *Bno*)

功能: 取回 8 位元資料不帶符號 B 暫存器值。

參數: ***Bno*** B 暫存器號碼: 0 ~ 1023、1024 ~ 2047 為斷電保持型

回應: B 暫存器 8 位元資料: 0 ~ 255

範例:

2.3.7.3 設定 W 暫存器值

- **void SET_W(WORD *Wno*, WORD *data*)**

功能: 指定 16 位元資料不帶符號 W 暫存器值。

參數:		
	Wno	**W 暫存器號碼: 0 ~ 1023、1024 ~ 2047 為斷電保持型**
	data	**16 位元資料: 0 ~ 65,535**

回應: 無

範例:

2.3.7.4 讀回 W 暫存器值

- **WORD GET_W(WORD *Wno*)**

功能: 取回 16 位元資料不帶符號 W 暫存器值。

參數:		
	Wno	**W 暫存器號碼: 0 ~ 1023、1024 ~ 2047 為斷電保持型**
回應:		**W 暫存器 16 位元資料: 0 ~ 65,535**

範例:

2.3.7.5 設定 DW 暫存器值

- **void SET_DW(WORD** *DWno*, **DWORD** *data*)

功能: 指定 32 位元資料不帶符號 DW 暫存器值。

參數:

	DWno	**DW 暫存器號碼: 0 ~ 4095、4096 ~ 8191 為斷電保持型**
	data	**32 位元資料: 0 ~ 4,294,967,295**

回應: 無

範例:

2.3.7.6 讀回 DW 暫存器值

- **DWORD GET_DW(WORD** *DWno*)

功能: 取回 32 位元資料不帶符號 DW 暫存器值。

參數:

	DWno	**DW 暫存器號碼: 0 ~ 4095、4096 ~ 8191 為斷電保持型**
回應:		**DW 暫存器 32 位元資料: 0 ~ 4,294,967,295**

範例:

2.3.7.7 設定 F 暫存器值

- **void SET_F(WORD *Fno*, float *data*)**

功能: 指定 F 浮點暫存器值。

參數: ***Fno*** F 暫存器號碼: 0 ~ 2047、2048 ~ 4095 為斷電保持型
data 7 位數浮點資料: 3.4E +/- 38 (7 digits)

回應: 無

範例:

2.3.7.8 讀回 F 暫存器值

- **float GET_F(WORD *Fno*)**

功能: 取回 F 浮點暫存器值。

參數: ***Fno*** F 暫存器號碼: 0 ~ 2047、2048 ~ 4095 為斷電保持型

回應: F 浮點暫存器值: 3.4E +/- 38 (7 digits)

範例:

2.3.8 資料區塊暫存器功能 DB

	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96	.....	1	0
	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	.....		
DB	BYTE(15)								BYTE(14)								BYTE(13								BYTE(12)								.....		
	WORD(7)																WORD(6)																.....		
	DWORD(3)																																.....		

2.3.8.1 設定 1 位元資料到 DB

- **void DB_SETx1(BYTE** *DBno*, **BYTE** *Num*, **bool** *data*)

功能: 移動 1 位元資料區塊，到指定的 DB 暫存器。

參數:

	DBno	**DB 暫存器號碼: 0 ~ 49**
	Num	**區段號碼: 0 ~ 127**
	data	**1 位元資料: 0，1**

回應: 無

範例:

2.3.8.2 讀回 DB 暫存器 1 位元資料

- **bool DB_GETx1(BYTE** *DBno*, **BYTE** *Num*)

功能: 從 DB 暫存器讀取 1 位元資料。

參數:

	DBno	**DB 暫存器號碼: 0 ~ 49**
	Num	**區段號碼: 0 ~ 127**

回應: **DB 暫存器 1 位元資料: 0，1**

範例:

2.3.8.3 設定 8 位元資料到 DB

- **void DB_SETx8(BYTE** *DBno*, **BYTE** *Num*, **BYTE** *data*)

功能: 移動 8 位元資料區塊，到指定的 DB 暫存器。

參數:

	DBno	**DB 暫存器號碼: 0 ~ 49**
	Num	**區段號碼: 0 ~ 15**
	data	**8 位元資料: 0 ~ 255**

回應: 無

範例:

2.3.8.4 讀回 DB 暫存器 8 位元資料

- **BYTE DB_GETx8(BYTE** *DBno*, **BYTE** *Num*)

功能: 從 DB 暫存器讀取 8 位元資料。

參數:

	DBno	**DB 暫存器號碼: 0 ~ 49**
	Num	**區段號碼: 0 ~ 15**

回應: **DB 暫存器 8 位元資料: 0 ~ 255**

範例:

2.3.8.5 設定 16 位元資料到 DB

- **void DB_SETx16(BYTE** *DBno*, **BYTE** *Num*,**WORD** *data*)

功能: 移動 16 位元資料區塊，到指定的 DB 暫存器。

參數:		
	DBno	**DB 暫存器號碼: 0 ~ 49**
	Num	**區段號碼: 0 ~ 7**
	data	**16 位元資料: 0 ~ 65,535**

回應: 無

範例:

2.3.8.6 讀回 DB 暫存器 16 位元資料

- **WORD DB_GETx16(BYTE** *DBno*, **BYTE** *Num*)

功能: 從 DB 暫存器讀取 16 位元資料。

參數:		
	DBno	**DB 暫存器號碼: 0 ~ 49**
	Num	**區段號碼: 0 ~ 7**

回應: **DB 暫存器 16 位元資料: 0 ~ 65,535**

範例:

2.3.8.7 設定 32 位元資料到 DB

- **void DB_SETx32(BYTE** *DBno*, **BYTE** *Num*, **DWORD** *data*)

功能: 移動 32 位元資料區塊，到指定的 DB 暫存器。

參數:

	DBno	**DB 暫存器號碼: 0 ~ 49**
	Num	**區段號碼: 0 ~ 3**
	data	**32 位元資料: 0 ~ 4,294,967,295**

回應: 無

範例:

2.3.8.8 讀回 DB 暫存器 32 位元資料

- **DWORD DB_GETx32(BYTE** *DBno*, **BYTE** *Num*)

功能: 從 DB 暫存器讀取 32 位元資料。

參數:

	DBno	**DB 暫存器號碼: 0 ~ 49**
	Num	**區段號碼: 0 ~ 3**

回應: **DB 暫存器 32 位元資料: 0 ~ 4,294,967,295**

範例:

2.3.8.9 左移 DB 暫存器位元

- **void DB_SL(BYTE** *DBno*, **BYTE** *Shift*)

功能: 位移 DB 暫存器位元。

參數: ***DBno*** DB 暫存器號碼: 0 ~ 49
Shift ←左移位元數: 1 ~ 128

回應: 無

範例:

2.3.8.10 右移 DB 暫存器位元

- **void DB_SR(BYTE** *DBno*, **BYTE** *Shift*)

功能: 位移 DB 暫存器位元。

參數: ***DBno*** DB 暫存器號碼: 0 ~ 49
Shift →右移位元數: 1 ~ 128

回應: 無

範例:

2.3.9 訊息資料讀寫

EzCore 訊息可以讓系統交換訊息,可以用於應用程序與人機界面做溝通。

2.3.9.1 訊息資料寫入

- **long SET_MSG(WORD *MSGno*,TCHAR *UMSG[30]*);**

功能: 訊息資料寫入。

參數:

	MSGno	訊息號碼，**0 ~ 249** 皆為斷電保持型 **MSG245~249 為 Password 輸入專用**
	UMSG[30]	欲寫入訊息內容，限 **30** 個字

回應:

0	執行成功
其他值	請參閱附錄一錯誤碼表

範例:

```
//1.直接用 UNICODE 字串傳入
RET= SET_MSG(100, _T("泓格科技訊息"));

//2.或者用 CString 傳入
CString CS= _T("泓格科技訊息");
TCHAR UMSG[30];
swprintf(UMSG, CS); //使用 swprintf( ); 將 CString 轉 TCHAR
                    //CS 的字數不可超過 UMSG 陣列數量
RET= SET_MSG(101, UMSG);

//3.在訊息開頭加入時間碼→ 10:20:55泓格科技訊息
CEzLIB EzLIB;
TCHAR tcTime[15];
EzLIB.Get_Time(tcTime);
CString HMSm(tcTime);
CString CS= _T("  泓格科技訊息");
CS= HMSm + CS;
TCHAR UMSG[30];
swprintf(UMSG, CS);
ret= SET_MSG(102, UMSG);

//4.在訊息開頭加入日期碼→ 2007/04/26  泓格科技訊息
TCHAR tcDate[15];
EzLIB.Get_Date(tcDate);
CString YMD(tcDate);
CString CS1= _T("  泓格科技訊息");
CS1= YMD + CS1;
TCHAR UMSG1[30];
swprintf(UMSG1, CS1);
ret= SET_MSG(103, UMSG1);
```

```
//5.在訊息開頭加入日期和時間碼→ 2007/04/26 10:20:55 泓格科技訊息
TCHAR tcDT[30];
EzLIB.Get_DT(tcDT, true, true, false, true, true, true, true);
CString YMWDHMSm(tcDT);
CString CS2= _T(" 泓格科技訊息");
CS2= YMWDHMSm + CS2;
TCHAR UMSG2[30];
swprintf(UMSG2, CS2);
ret= SET_MSG(104, UMSG2);
```

2.3.9.2 訊息資料讀取

- **long GET_MSG(WORD** *MSGno*,**TCHAR** *UMSG[30]*);

功能: 訊息資料讀取。

參數:	*MSGno*	訊息號碼，0 ~ 249 皆為斷電保持型 MSG245~249 為 Password 輸入專用
	UMSG[30]	欲讀取訊息內容 30 個字
回應:	0	執行成功
	其他值	請參閱附錄一錯誤碼表

範例:

```
TCHAR UMSG[30];
RET= GET_MSG(100, UMSG);
CString CS(UMSG);  //使用 CString( ); 將 TCHAR 轉 CString
```

2.3.9.3 讀取多語系文字檔寫到 MSG 訊息

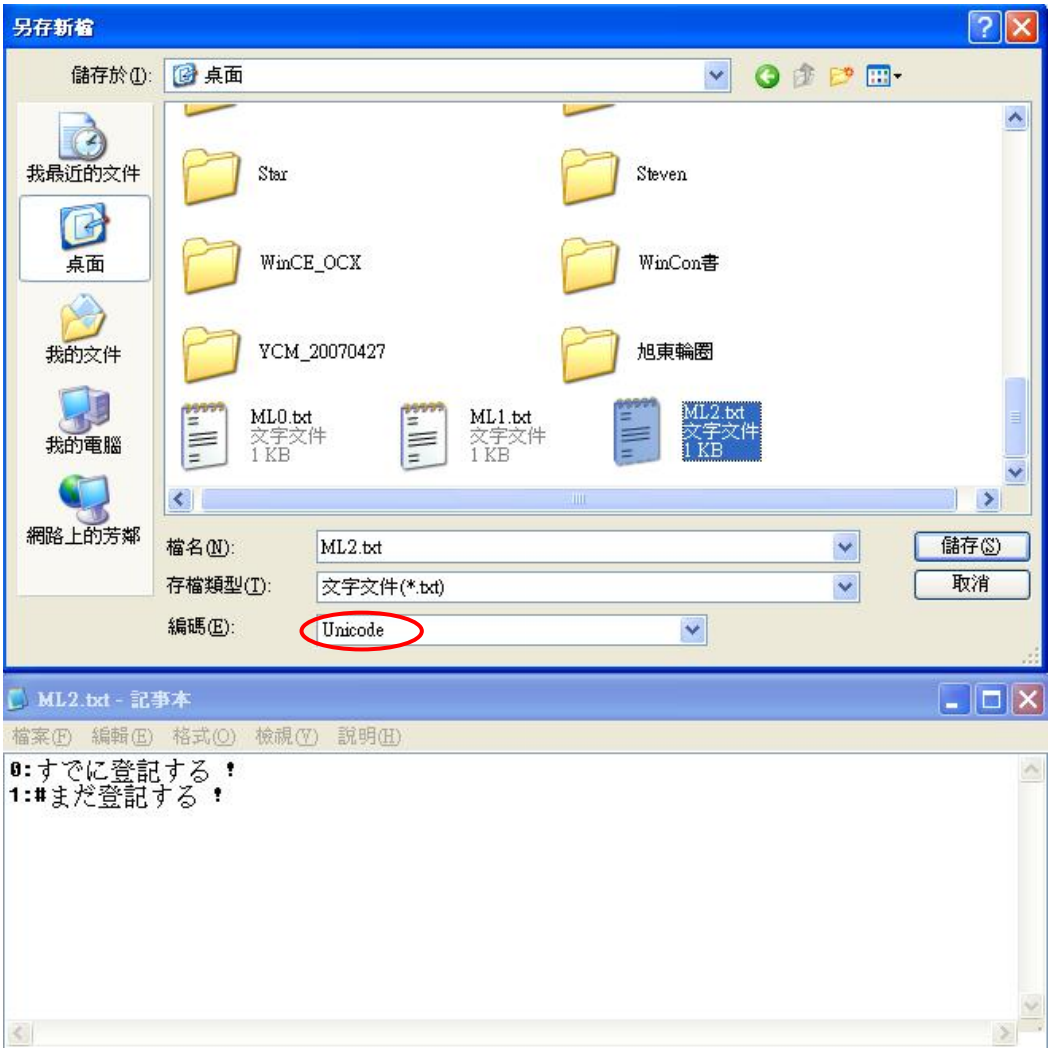
EzCore 提供多國語系訊息，可以讓系統依不同語系使用不同訊息，可以用於應用程序與人機界面多國語系切換用，當 D8000 被修改時，稍後立即自行變更。

而多國語系檔案用 Unicode 分別存為 ML0.txt，ML1.txt ~ML7.txt，分別對應 Multi-Language 0~7 八個語系文字檔，檔案需事先置於 PAC 如下路徑中: EzProg_Path\EzProg-I\EzHMI\ML\ML0~7.txt，其每一語系可以包括 0~999 則訊息，每一個訊息上限 30 個字，格式如下述。

ML0.txt ➔
0:系統初始化 OK !!
1:使用者登入成功 !!

ML1.txt ➔
0:System Initial OK !!
1:User login OK !!

":" 前數字為訊息號碼 (*MSGFno)*
":" 後 30 個字為訊息內容，每則以 CrLf 為終止



● **long SET_MSGF(WORD** *MSGno*, **WORD** *MSGFno*);

功能: 多國語系訊息資料讀取到訊息資料(MSG)。

參數:

	MSGno	訊息號碼，**0 ~ 249** 皆為斷電保持型
	MSGFno	多國語系訊息號碼，**0 ~ 999**

回應:

	0	執行成功
	其他值	請參閱附錄一錯誤碼表

範例:

```
RET= SET_MSGF(100, 0);
//將多國語系訊息檔 MLn.txt 的第 0 則訊息; Set 到 MSG100 號訊息中
```

2.3.9.4 讀取多語系文字檔資料

● **long GET_MSGF(WORD** *MSGFno*,**TCHAR** *UMSG[30]*);

功能: 讀取多國語系訊息。

參數:

	MSGFno	多國語系訊息號碼，**0 ~ 999**
	UMSG[30]	欲讀取訊息內容 **30** 個字

回應:

	0	執行成功
	其他值	請參閱附錄一錯誤碼表

範例:

```
TCHAR UMSG[30];
RET= GET_MSGF(0, UMSG); //將 MLn.txt 的第 0 則訊息; 傳到 UMSG
CString CS(UMSG);  //使用 CString( ); 將 TCHAR 轉 CString
```

2.3.10 使用 AES 加密的系統保全

EzCore 內建一方便的 AES 加密機制功能，軟體開發者可以輕易的保護合法軟體，而此機制是利用軟體開發者指定的金鑰(AES_KEY)作加解密，並結合 PAC 的唯一硬體序號做認證。

本章節提供兩種簡易的方法自由選擇使用，即可操作 EzCore AES 的認證，並在 EzConfig(3.5 章節)中提供方便的操作介面(依輸入 PAC 的唯一硬體序號及加密金鑰產生註冊碼)，輕易達成系統保全功能保護著作財產權。

2.3.10.1 將註冊碼註冊到 EzCore 系統中

long REGISTRY_KEY(TCHAR *REG[20]*)

功能: 將軟體開發者透過提供 EzConfig 所產生的註冊碼，合法授權給 End-user。再由 end-user 輸入給應用程式，然後應用程式可以使用 REGISTRY_KEY( )對 EzCore 註冊。

參數:	*REG[20]*	1.軟體開發者提供的註冊碼 16 個字串指標 2.或由 MSGno 傳入 EzConfig/Edit/Manage→Registry-codeGenerator
回應:	0	輸入語法正確
	其他值	請參閱附錄一錯誤碼表

範例:
```
REGISTRY_KEY(_T("05386f8e9a7b6fa7"));
REGISTRY_KEY(_T("MSG249"));
```

2.3.10.2 將註冊檔(AES.txt)註冊到 EzCore 系統中

long REGISTRY_FILE()

功能: 將軟體開發者透過提供 EzConfig 所產生的註冊檔，路徑在 \EzProg_Path\EzProg-I\EzConfig\AES.txt，合法授權給 End-user。然後應用程式可以使用 REGISTRY_FILE( )對 EzCore 註冊。

回應:	0	輸入語法正確
	其他值	請參閱附錄一錯誤碼表

範例:
```
REGISTRY_FILE();
```

2.3.10.3 檢查 EzCore 系統中註冊碼是否合法

long CHECK_KEY(TCHAR *AES_KEY[20]***)**

功能: 軟體開發者在應用程式中的任何位置，可隨時下 **CHECK_KEY( )**，利用 **EzCore** 自動檢查註冊碼是否合法。

參數:

AES_KEY[20]	軟體開發者個人或軟體的加密金鑰 **16** 個字串指標 **EzConfig/Edit/Manage→Key-code Input**

回應:

0	檢查正確**(**註冊碼合法授權**)**
其他值	請參閱附錄一錯誤碼表

範例: **long RET= CHECK_KEY(_T("1234567812345678"));**

3 使用 EzCore 的執行程序

Ezprog-I 內建的功能，如數位輸入(DI)、數位輸出(DO)、類比輸入(AI)、類比輸出(AO)、多軸運動控制(Motion)、高速分散式 DIO(FRNET)，而這些模組經適當規劃後(EzConfig 自動規劃)，在 EzCore 中我們就可以用統一的方法，輕易的使用它們，而在系統中如何使用即是本章節要說明的內容。

本章分為四大部分:

3.1 使用 USER 自定執行緒功能:使用 PAC 多工機制，同時執行多執行緒，這是可以處理八個多執行緒程序，可以用來處理其他執行程序，讓一般程序開發使用者可以簡化其多執行緒程序開發步驟，專注開發應用軟體。

3.2 使用者自訂定時執行功能:使用 PAC 強即時機制，產生定時執行功能:這是可以處理八個定時執行服務程序，可以用來處理類似 PLC 執行結構，讓 PLC 使用者可以之前 PLC 的設計結構，迅速開始開發軟體。

3.3 使用 DI 中斷執行功能:這是可以處理八個 DI(需將 i8048 插入第一槽)硬體中斷服務程序，可以大幅簡化開發中斷處理程序。

3.4 使用 Motion 中斷執行功能:這是可以處理 Slot 1~3 Motion 模組 (i8092F,i8094(F),i8094A(H)) 中斷處理服務程序，可以大幅簡化開發 Motion 中斷處理程序。

因 WinCE 為多工作業系統，所以有執行優先順序， 3.3,3.4 為最高，3.2 為次之，3.1 為最低，其中各有八個執行功能，其優先順序 0 最高 7 最低，始用者再選擇時要注意依您的需求安排使用。
3.3(0>1>2>>…>7) > 3.4 > 3.2(0>1>2>>…>7) 3.1(0>1>2>>…>7)

本章的範例都有配合 EzHMI 物件,相關資料請參閱 EzProg-I Tools 手冊。

3.1 使用 USER 自定執行緒功能

提供由使用者啟動執行本程序，設計與使用時要特別注意，由於自定執行程序只執行一次，如果要設計為迴圈或無窮迴圈，請在期中加入暫停指 Sleep(nn) ，如 Sleep(10);將會使程序再續執行，最好要設計停止迴圈的機制，以方便控制執行整個程序結束時間。

3.1.1 執行自定執行緒

- **long START_USER_THREAD(BYTE** *USERno*, **LPTHREAD_START_ROUTINE** *lpStartAddress*)

功能: 執行自定執行緒程序。

參數:	***USERno***	自定執行緒號碼: **0(最優先) ~ 7** **0~4 Soft Real Time** **5~7 Non Real Time(MFC Class)**
	lpStartAddress	自定執行緒 **function** 的指標
回應:	**0**	執行成功
	其他值	請參閱附錄一錯誤碼表

範例: 請參考 **3.1.3** 範例

3.1.2 自定執行緒結束

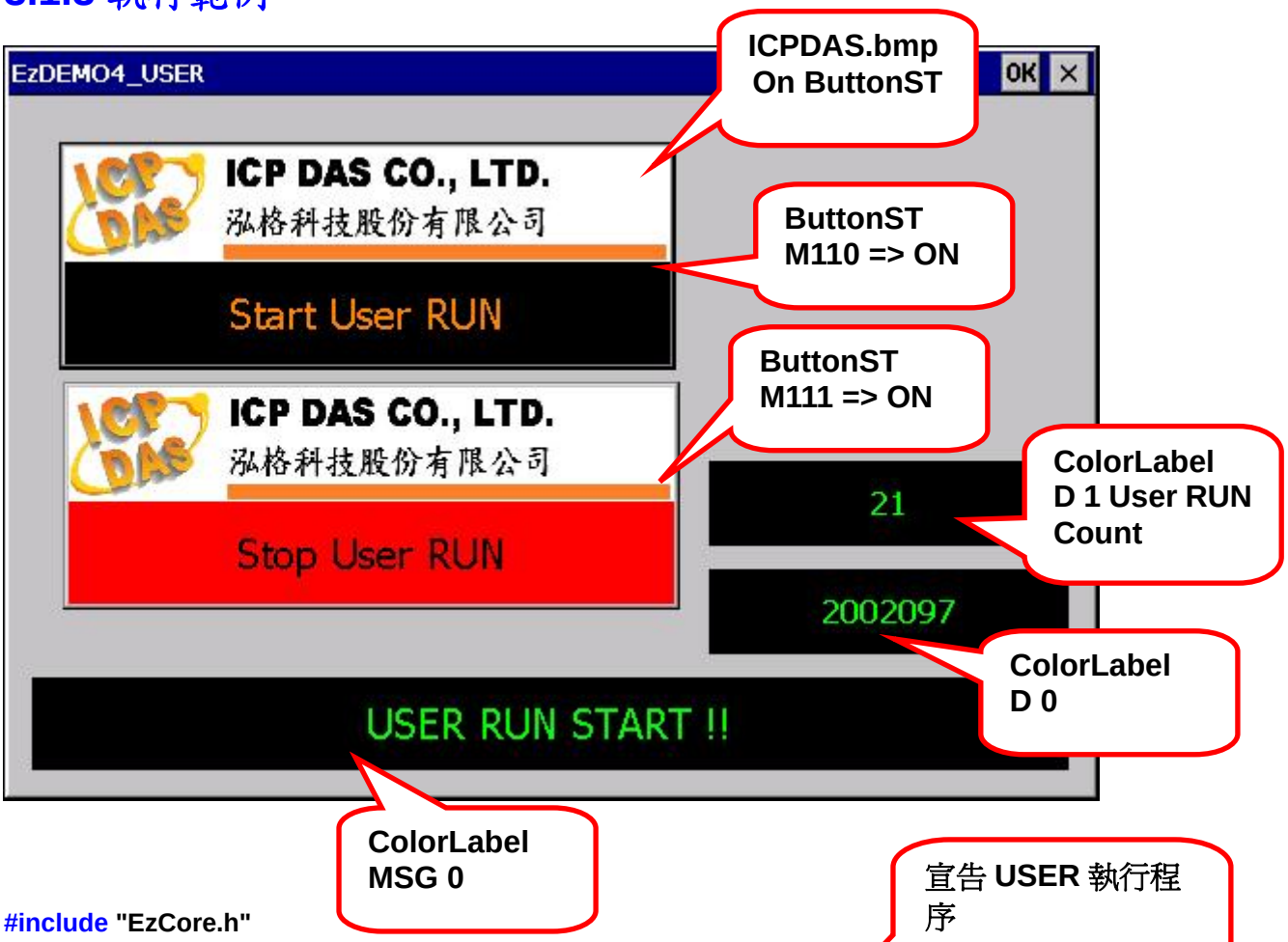
- **long END_USER_THREAD (BYTE** *USERno*)

功能: 自定執行緒結束時需使用此含式，使下次可以再使用此號碼。

參數:	***USERno***	自定執行緒號碼
回應:	**0**	執行成功
	其他值	請參閱附錄一錯誤碼表

範例: 請參考 **3.1.3** 範例

3.1.3 執行範例



宣告 USER 執行程序

```
#include "EzCore.h"

unsigned long USER_RUN(void *) //====USER_RUN main Function==========
{
      SET_D(1,0);
      while (true)
      {
            SET_D(1,GET_D(1)+1);
            Sleep(500);
            if (GET_Ma(111))
                  break;
      }
      SET_M(111,false);
      END_USER_THREAD(0);
      SET_MSG(0,TEXT("USER RUN STOP !!"));
      SET_D(1,0);
      return 0;
}
```

D 1 User RUN Count

利用 M111 結束 USER RUN 程序

```
void TSR_RUN() //====TSR_RUN main loop Function==============
{
	SET_D(0,GetTickCount());
	OUT_Y(0,IN_Xa(0));
	if (GET_Ma(110))
	{
		SET_M(110,false);
		long RET=START_USER_THREAD(0,USER_RUN);
		SET_MSG(0,TEXT("USER RUN START !!"));
	}
}
```

宣告 **RTSR** 執行程序

利用 **M110** 啓動 **USER** 執行程序

```
BOOL CEzDEMO4_USERDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	CenterWindow(GetDesktopWindow());	// center to the hpc s
```

EzCore 初始化

```
	// TODO: Add extra initialization here
	long ret;
	ret=DEVICE_INITIAL(OUTPUT_Now,IO_MODE_AutoScan,RUN_PRG_MODE);
	if (ret == _NO_ERROR)
	{
		ret=SCAN_ENGINE_START();
		if (ret == _NO_ERROR)
		{
			SET_M(200,true); //System initial ok
			SET_RTSR(0,&(ptTSRFunc)TSR_RUN,1000);
			START_RTSR(0);
			SET_MSG(0,TEXT("start RTSR OK !!"));
		}
		else
			MessageBox( TEXT("Start ENGINE NG   \n Please check "), TEXT("EzCore
Engine"), MB_OK|MB_ICONERROR);
	}
	else
	{
		MessageBox( TEXT("Load ALL Device Data NG   \n Please check \n Please
Rescan IO "), TEXT("EzCore Engine"), MB_OK|MB_ICONERROR);
	}
	return TRUE;  // return TRUE  unless you set the focus to a control
}
```

啓動 **RTSR** 執行程序

```
void CEzDEMO4_USERDlg::OnClose()
{
    // TODO: Add your message handler code here and/or call default
    SCAN_ENGINE_STOP();
    CDialog::OnClose();
}
```

關閉 EzCore SCAN_ENGINE

3.2 使用 RTSR 自訂定時執行功能

在使用本功能前,使用者必須先執行初始設定 EzCore 引擎 DEVICE_INITIAL() 正常(ret=0)及啟動 EzCore 引擎 SCAN_ENGINE_START()

```
long ret;
ret= DEVICE_INITIAL(OUTPUT_Now, IO_MODE_AutoScan,
RUN_PRG_MODE);
SCAN_ENGINE_START();
```

由於系統會定時來自動執行本程序，設計與使用時要特別注意，由於定時執行程序優先權很高，所以請盡量避免在中斷處理程序中撰寫執行時間很長的迴圈或無窮迴圈或程式碼與暫停指令如 Sleep() ，如執行整個程序超過設定定時時間，將會使下一次執行時間自動延後。

3.2.1 初始設定

- **void SET_RTSR(BYTE** *RTSRno*, **ptTSRFunc** *SRF*,
  WORD *msInterval*)

功能: 設定相關 **RTSR (Real-Time Service Routine)**，**CALLBACK function** 及時間間隔參數。

參數:		
	RTSRno	**RTSR 號碼: 0(最優先) ~ 7**
	SRF	**CALLBACK function 的指標**
	msInterval	**執行間隔時間數(2~60000ms)**

回應: 無

範例: 請參考 **3.2.5** 範例

3.2.2 啟動定時執行

- **long START_RTSR(BYTE** *RTSRno*)

功能: 啟動定時執行。

參數:		
	RTSRno	**RTSR 號碼: 0 ~ 7**

回應:		
	0	執行成功
	其他值	請參閱附錄一錯誤碼表

範例: 請參考 **3.2.5** 範例

3.2.3 停止定時執行

- **void STOP_RTSR(BYTE** *RTSRno***)**

功能: 停止定時執行。

參數: **RTSRno** **RTSR 號碼: 0 ~ 7**

回應: 無

範例: 請參考 **3.2.5** 範例

3.2.4 查詢定時執行耗用多少時間

- **long GET_RTSR_TIME(BYTE** *RTSRno***)**

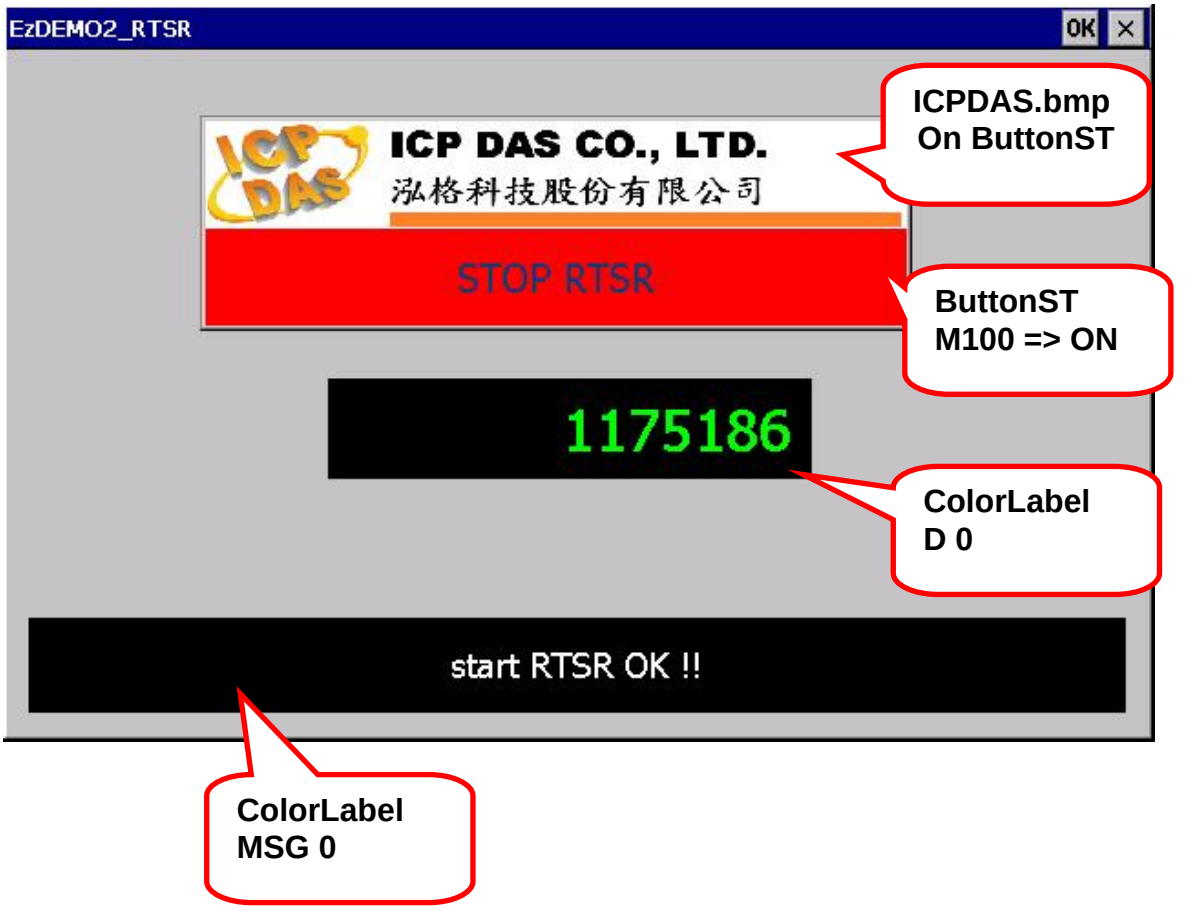
功能: 查詢 RTSR 執行一次耗用多少時間。

參數: **RTSRno** **RTSR 號碼: 0 ~ 7**

回應: **0** 執行一次耗用 1ms 微秒以下或未執行
其他值 執行一次耗用微秒數

範例: 請參考 **3.2.5** 範例

3.2.5 執行範例




```
#include "EzCore.h"

void TSR_RUN() //====main loop Function===============
{
    SET_D(0,GetTickCount());
    OUT_Y(0,IN_Xa(0));
    if (GET_Ma(100))
    {
        STOP_RTSR(0);
        SET_M(100,false);
        SET_MSG(0,TEXT("RTSR STOP !!"));
    }
}
```

宣告定時執行程序

停止定時執行程序

```
BOOL CEzDEMO2_RTSRDlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	// Set the icon for this dialog.  The framework does this automati
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	CenterWindow(GetDesktopWindow());	// center to the [illegible] screen

	// TODO: Add extra initialization here
	long ret;
	ret=DEVICE_INITIAL(OUTPUT_Now,IO_MODE_AutoScan,RUN_PRG_MODE);
	if (ret == _NO_ERROR)
	{
		ret=SCAN_ENGINE_START();
		if (ret == _NO_ERROR)
		{
			SET_M(200,true); //System initial ok
			SET_RTSR(0,&(ptTSRFunc)TSR_RUN,1000);
			START_RTSR(0);
			SET_MSG(0,TEXT("start RTSR OK !!"));
		}
		else
			MessageBox( TEXT("Start ENGINE NG   \n Please check "), TEXT("EzCore
Engine"), MB_OK|MB_ICONERROR);
	}
	else
	{
		MessageBox( TEXT("Load ALL Device Data NG   \n Please check \n Please
Rescan IO "), TEXT("EzCore Engine"), MB_OK|MB_ICONERROR);
	}
	return TRUE;  // return TRUE  unless you set the focus to a control
}
```

EzCore 初始化

設定與執行定時執行程序

```
void CEzDEMO2_RTSRDlg::OnClose()
{
	// TODO: Add your message handler code here and/or call default
	SCAN_ENGINE_STOP();
	CDialog::OnClose();
}
```

EzCore
SCAN_ENGINE 關閉

3.3 使用 DI 中斷執行功能

這是可以處理八個 DI(i8048)硬體中斷服務程序，可以大幅簡化開發中斷處理程序，i8048 必須要安裝在控制器的第一槽(Slot 1) 。

請設計與使用時要特別注意，由於中斷處理程序優先權很高，所以請盡量避免在中斷處理程序中撰寫執行時間很長的迴圈或無窮迴圈或程式碼。

3.3.1 初始設定

- **long SET_INT(BYTE** *Channel*, **ptTSRFunc** *SRF*)

功能: 設定相關中斷服務程序 **(Interrupt Service Routine)**。

參數:	***Channel***	中斷 i8048 DI 號碼: 0(最優先) ~ 7
	SRF	CALLBACK function 的指標
回應:	**0**	執行成功
	其他值	請參閱附錄一錯誤碼表

範例: 請參考 **3.3.5** 範例

3.3.2 啟動 DI 中斷執行功能

- **long START_INT(BYTE** *Channel*)

功能: 啟動中斷功能。

參數:	***Channel***	中斷 i8048 DI 號碼:
回應:	**0**	執行成功
	其他值	請參閱附錄一錯誤碼表

範例: 請參考 **3.3.5** 範例

3.3.3 停止 DI 中斷執行功能

- **long STOP_INT(BYTE** *Channel***)**

功能: 停止定時執行。

參數: ***Channel*** 中斷 i8048 DI 號碼:

回應: **0** 執行成功
其他值 請參閱附錄一錯誤碼表

範例: 請參考 **3.3.5** 範例

3.3.4 查詢 DI 中斷執行耗用多少時間

- **long GET_INT_TIME(BYTE** *Channel***)**

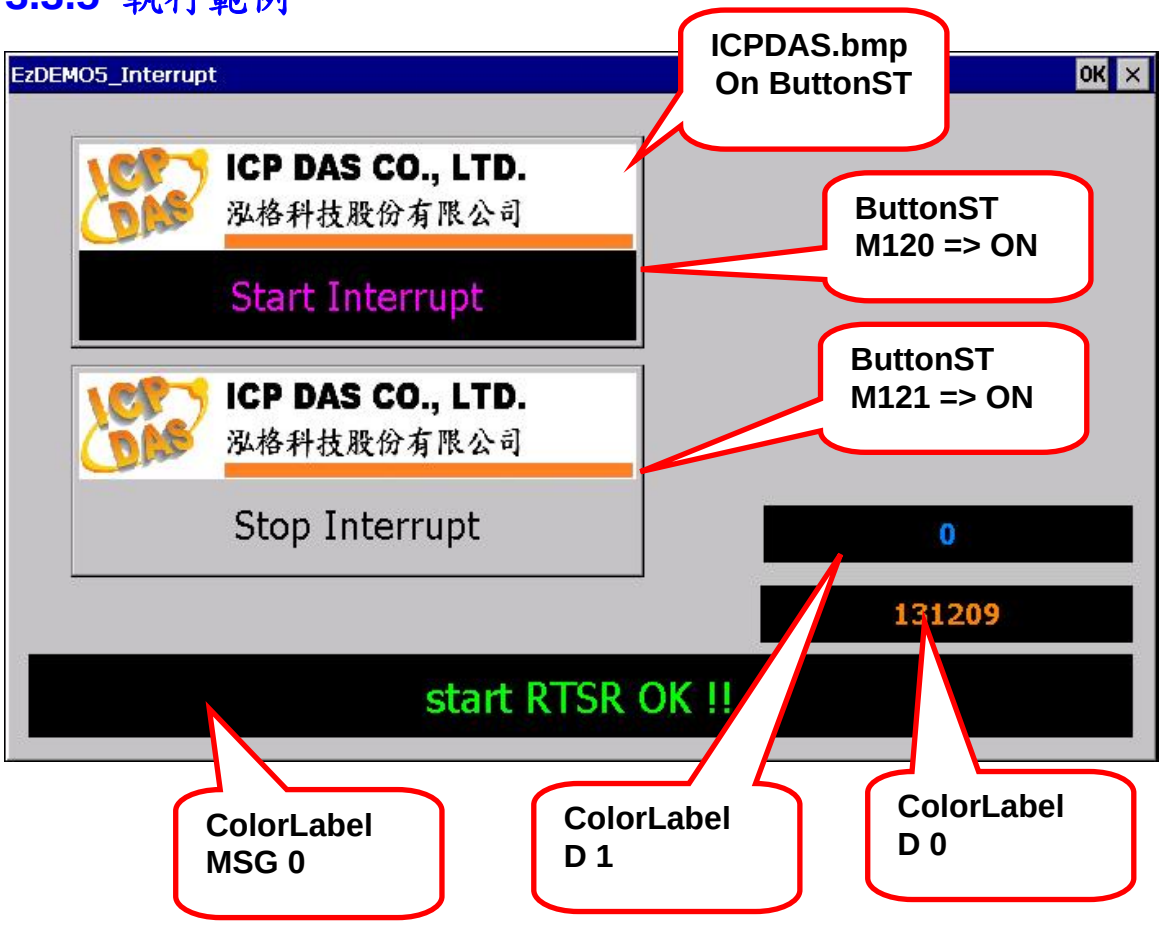
功能: 查詢 DI 中斷執行一次耗用多少時間。

參數: ***Channel*** 中斷 i8048 DI 號碼:

回應: **0** 執行一次耗用 1ms 微秒以下或未執行
其他值 執行一次耗用微秒數

範例: 請參考 **3.3.5** 範例

3.3.5 執行範例




```
#include "EzCore.h"

void INTP_RUN0();
unsigned long USER_RUN7(void *);
```

宣告中斷
執行程序

宣告 USER RUN
執行程序

```
BOOL CEzDEMO5_InterruptDlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	CenterWindow(GetDesktopWindow());	// center to the hpc scree

	// TODO: Add extra initialization here
	long ret;
	ret=DEVICE_INITIAL(OUTPUT_Now,IO_MODE_AutoScan,RUN_PRG_MODE);
	if (ret == _NO_ERROR)
	{
		ret=SCAN_ENGINE_START();
		if (ret == _NO_ERROR)
		{
			SET_D(8000,0);
			SET_M(200,true); //System initial ok
			long RET=START_USER_THREAD(7,USER_RUN7);
			SET_MSG(1,TEXT("start USER RUN 7 OK !!"));
			SET_M(131,true);
			SET_M(130,false);
		}
		else
			MessageBox( TEXT("Start ENGINE NG    \n Please check "), TEXT("EzCore
Engine"), MB_OK|MB_ICONERROR);
	}
	else
	{
		MessageBox( TEXT("Load ALL Device Data NG    \n Please check \n Please
Rescan IO "), TEXT("EzCore Engine"), MB_OK|MB_ICONERROR);
	}
	return TRUE;  // return TRUE  unless you set the focus to a control
}
```

EzCore 初始化

執行 USER 執行程序

```
void INTP_RUN0() //====main INTP 0 Function=======================
{
	SET_D(1,GET_D(1)+1);
	OUT_Y(0,GET_Yb(0));
}
```

中斷服務程序

```
unsigned long USER_RUN7(void *) //====USER_RUN main Function===========
{
	long RET;
	SET_D(1,0);
	while (true)
	{
		SET_D(10,GetTickCount());
		OUT_Y(0,IN_Xa(0));
		if (GET_Ma(120))
		{
			SET_D(1,0);
			SET_M(120,false);
			SET_M(130,true);
			SET_M(131,false);
			RET=SET_INT(0,&(ptTSRFunc)INTP_RUN0);
			RET=START_INT(0,INTP_MODE_Rising);
			SET_MSG(1,TEXT("Interrupt Enable !!"));
		}
		if (GET_Ma(121))
		{
			SET_M(121,false);
			SET_M(131,true);
			SET_M(130,false);
			RET=STOP_INT(0);
			SET_MSG(1,TEXT("Interrupt Disable !!"));
		}
		Sleep(999);
	}
	SET_M(111,false);
	END_USER_THREAD(7);
	SET_MSG(1,TEXT("USER RUN STOP !!"));
	SET_D(1,0);

	SET_M(999,false);
	SET_M(998,true);
	return 0;
}
```

啓動中斷功能

停止中斷功能

關閉

```
void CEzDEMO5_InterruptDlg::OnClose()
{
	// TODO: Add your message handler code here and/or call default
	SCAN_ENGINE_STOP();
	CDialog::OnClose();
}
```

3.4 使用 Motion 中斷執行緒功能

EzProg-I 支援 Motion 卡使用中斷功能，可以支援 i8092F、i8094、i8094F、i8094A、i8094H...中斷功能處理，中斷的設定與產生詳情請參閱 i8092、i8094、i8094H 使用手冊，Motion 卡配合 EzProg-I 使用時要特別注意，使用手冊中所有 API 函式中使用的卡號參數，必須等同硬體槽位號碼，也不用做註冊軸卡的動作。

設計與使用時要特別注意，由於中斷處理程序有高優先執行權，所以請盡量避免在中斷處理程序中撰寫執行時間很長的迴圈或無窮迴圈或程式碼。

3.4.1 啟動 Motion 卡中斷功能

- **long MOTION_ENABLE_INT(BYTE *Slot*)**

功能: 每一張 Motion 卡要使用中斷功能前必須先啟動中斷功能，然後才能作其他的中斷設定，目前只支援 PAC 前三槽的 Motion 中斷功能啟用。

參數:

Slot	**Motion 卡安裝在 PAC 的槽位(Slot 1~3)**

回應:

0	**執行成功**
其他值	**請參閱附錄一錯誤碼表**

範例: 請參考 3.4.7 範例

3.4.2 關閉 Motion 卡中斷功能

- **long MOTION_DISABLE_INT (BYTE *Slot*)**

功能: 將任一 Motion 卡中斷功能關閉，其中已使用的中斷設定，也會全部自動關閉。

參數:

Slot	**Motion 卡安裝在 PAC 的槽位(1~3 槽)**

回應:

0	**執行成功**
其他值	**請參閱附錄一錯誤碼表**

範例: 請參考 3.4.7 範例

3.4.3 宣告 Motion 中斷服務程序

中斷服務程序之宣告定義如下:
Int **(*ptM_INTPFunc)(WORD** *MINTTable***)**

回傳變數 *MINTTable* 如下:
i8092F: i8092 手冊章節 6.4，中斷條件因子設定

Bit	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小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大於等於正方向比較暫存器的設定值
5	C-END	中斷發生於等速段和保留脈波段結束兩個位置
6	C-STA	中斷發生於等速段和保留脈波段開始時兩個位置
7	D-END	中斷發生於驅動結束時

i8094/F: 配合 i8094 手冊章節 6.5，中斷條件因子設定

Bit	代號	說明
0	PULSE	中斷發生於每一個脈波正緣產生時
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小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大於等於正方向比較暫存器的設定值
5	C-END	中斷發生於等速段和保留脈波段結束兩個位置
6	C-STA	中斷發生於等速段和保留脈波段開始時兩個位置
7	D-END	中斷發生於驅動結束時

i8094A/H: 請配合 i8094H 手冊章節6.3.7，軸卡 i8094H 對控制器中斷功能

Bit	說明
0(0x01)	Line Scan完成
1(0x02)	Macro Program完成
2(0x04)	使用者定義RINT完成
3(0x08)	連續補間被中斷
4(0x10)	
5(0x20)	預留系統使用
6(0x40)	Axes Error
7(0x80)	Module Error

範例:

```
MOTION_ENABLE_INT(MSlot);
MOTION_SET_INT(MSlot, AXIS_X, &(ptM_INTPFunc)INTP_MOTION);

//====main Motion INTP Function=========================
void INTP_MOTION(WORD MINTTable)
{
    switch (MINTTable)
    {
        case 0x02:
            SET_D(1, MINTTable);
            SET_D(2, GET_D(2)+1);
            SET_MSG(2, TEXT("Macro Program finished !!"));
            break;
        case 0x04:
            SET_D(1, MINTTable);
            SET_D(2, GET_D(2)+1);
            SET_MSG(2, TEXT("RINT finished !!"));
            break;
        default:
            break;
    }
}
```

3.4.4 設定 Motion 中斷執行

long MOTION_SET_INT (BYTE *Slot*, **WORD** *Axis*, **ptM_INTPFunc** *SRF*)
功能: 設定中斷執行相關參數及 **CALLBACK Function**。

參數:	***Slot***	**Motion** 卡安裝在 **PAC** 的槽位**(1~3** 槽**)**
	Axis	欲接受中斷資訊的軸 **i8092F: AXIS_X**、**AXIS_Y** **i8094/F: AXIS_X**、**AXIS_Y**、**AXIS_Z**、**AXIS_U** **i8094A/H: AXIS_X(**只有卡的中斷**)**
	SRF	中斷服務程序 **CALLBACK Function** 的指標
回應:	**0**	執行成功
	其他值	請參閱附錄一錯誤碼表

範例: 請參考 **3.4.7** 範例

3.4.5 啟動 Motion 中斷服務程序

long MOTION_START_INT (BYTE *Slot*, **WORD** *Axis*)
功能: 經過設定後才可以啟動 **Motion** 中斷服務程序，執行後如果有相關中斷發生，就會執行先前設定的中斷服務程序 **CALLBACK function**。

參數:	***Slot***	**Motion** 卡安裝在 **PAC** 的槽位**(1~3** 槽**)**
	Axis	欲接受中斷資訊的軸 **i8092(F):AXIS_X,AXIS_Y** **i8094(F):AXIS_X,AXIS_Y, AXIS_Z,AXIS_U** **i8094A(H):AXIS_X(**只有卡的中斷**)**
回應:	**0**	執行成功
	其他值	請參閱附錄一錯誤碼表

範例: 請參考 **3.4.7** 範例

3.4.6 停止 Motion 中斷服務程序

long MOTION_STOP_INT (BYTE *Slot*, **WORD** *Axis*)
功能: 設定中斷執行相關參數及 **CALLBACK function**。

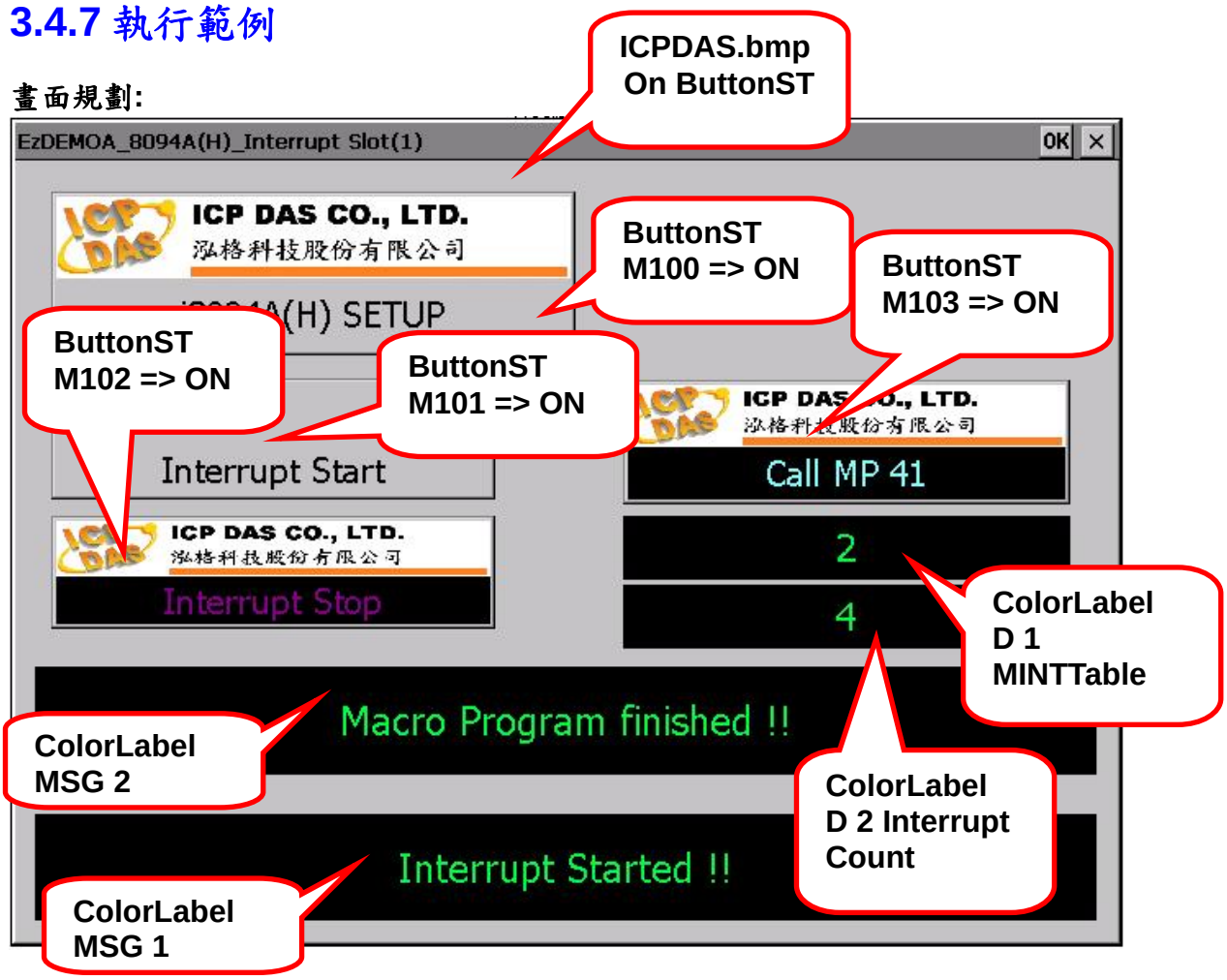
參數:	***Slot***	**Motion** 卡安裝在 **PAC** 的槽位**(1~3** 槽**)**
	Axis	欲接受中斷資訊的軸 **i8092(F):AXIS_X,AXIS_Y** **i8094(F):AXIS_X,AXIS_Y, AXIS_Z,AXIS_U** **i8094A(H):AXIS_X(**只有卡的中斷**)**

回應:	**0**	執行成功
	其他值	請參閱附錄一錯誤碼表

範例: 請參考 **3.4.7** 範例

3.4.7 執行範例

畫面規劃:



宣告:

```
#include "EzCore.h"
#include "i8094H.h"

BYTE MSlot=1;
long ret;

//====main Motion INTP Function for i8094A(H) =========
void INTP_MOTION(WORD MINTTable) ;
unsigned long USER_RUN7(void *);
```

```
BOOL CEzDEMOA_8094A_InterruptDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	CenterWindow(GetDesktopWindow());	// center to the hpc s

	// TODO: Add extra initialization here
	long ret;
	ret=DEVICE_INITIAL(OUTPUT_Now,IO_MODE_AutoScan,RUN_PRG_MODE);
	if (ret == _NO_ERROR)
	{
		ret=SCAN_ENGINE_START();
		if (ret == _NO_ERROR)
		{
			SET_M(200,true); //System initial ok
			//================================
			ret=MOTION_ENABLE_INT(MSlot);
			ret+=MOTION_SET_INT(MSlot,AXIS_X,&(ptM_INTPFunc)INTP_MOTION);
			if (ret==0)
			{
				SET_M(200,false);
				SET_M(201,true);
				SET_M(202,true);
				SET_M(203,true);
			}
			else
			{
				SET_M(200,true);
				SET_M(201,true);
				SET_M(202,true);
				SET_M(203,true);
			}
			long RET=START_USER_THREAD(7,USER_RUN7);
			//================================
			SET_MSG(1,TEXT("start USER RUN 7 OK !!"));
			SET_MSG(2,TEXT("======================"));
	}
		else
			MessageBox( TEXT("Start ENGINE NG   \n Please check "), TEXT("EzCore
Engine"), MB_OK|MB_ICONERROR);
	}
	else
	{
		MessageBox( TEXT("Load ALL Device Data NG   \n Please check \n Please
Rescan IO "), TEXT("EzCore Engine"), MB_OK|MB_ICONERROR);
	}
	return TRUE;  // return TRUE  unless you set the focus to a control
}
```

EzCore 初始化

啓動 USER RUN7
執行程序

程式:

宣告 Motion 執行程序

```
//====main Motion INTP Function=========================
void INTP_MOTION(WORD MINTTable)
{
	switch (MINTTable)
	{
	case 2:
		SET_D(1,MINTTable);
		SET_D(2,GET_D(2)+1);
		SET_MSG(2,TEXT("Macro Program finished !!"));
		break;
	case 4:
		SET_D(1,MINTTable);
		SET_D(2,GET_D(2)+1);
		SET_MSG(2,TEXT("RINT finished !!"));
		break;
	default:
		break;
	}
}
```

宣告 USER RUN7 執行程

```
unsigned long USER_RUN7(void *) //====USER_RUN main Function===========
{
	int i;
	long RET;
	SET_D(1,0);
	while (true)
	{
		OUT_Y(0,IN_Xa(0));
		if (GET_Ma(100) && GET_Mb(200))
		{
			SET_M(100,false);
```

建立 MP41

```
			i8094H_MP_CREATE(MSlot, MP41); //開始將巨集程式寫入i8094H中。
			//=======================================================
			//下面這些程式並不會被執行，而是將程式碼寫入i8094H中待命。
			i8094H_SET_LP(MSlot, AXIS_XYZU, 0);
			i8094H_MP_SET_RINT(MSlot);
			i8094H_SET_MAX_V(MSlot, AXIS_XYZU, 20000); //設定軸最高速20K PPS。
			i8094H_NORMAL_SPEED(MSlot, AXIS_XYZU, 0); //設定XYZU爲對稱T曲線。
			i8094H_SET_V(MSlot, AXIS_XYZU, 2000); //設定XYZU軸速度=2000 PPS。
			i8094H_SET_A(MSlot, AXIS_XYZU, 3000); //設定XYZU軸加速度1000 PPS/S。
			i8094H_SET_SV(MSlot, AXIS_XYZU, 1000); //設定XYZU初始速度2000 PPS。
			i8094H_SET_AO(MSlot, AXIS_XYZU, 0); //XYZU軸減速(保留脈波數)= 9 PPS。
			i8094H_MP_CALL(MSlot, MP50);
			i8094H_MP_CLOSE(MSlot); //結束第1卡，巨集程式寫入i8094H中。
			//=======================================================
```

```
        i8094H_MP_CREATE(MSlot, MP50); //開始將巨集程式寫入i8094H中。
        //=========================================================
        //下面這些程式並不會被執行，而是將程式碼寫入i8094H中待命。
        i8094H_MP_SET_VAR(MSlot, VAR1, 0); //VAR1 = 0。
        i8094H_FIXED_MOVE(MSlot, AXIS_XYZU, 5000); //XYZU移動5000 Pulse。
        i8094H_MP_STOP_WAIT(MSlot, AXIS_XYZU);
        i8094H_MP_CLOSE(MSlot); //巨集程式寫入i8094H中。
        //=========================================================
```

建立 MP50

```
        SET_M(200,true);
        SET_M(201,false);
        SET_M(202,true);
        SET_M(203,false);
        SET_MSG(1,TEXT("Interrupt Setting Ok !!"));
    }
    if (GET_Ma(101) && GET_Ma(200))
    {
        SET_M(101,false);
        SET_M(201,true);
        SET_M(202,false);
        MOTION_START_INT(MSlot,AXIS_X);
        SET_MSG(1,TEXT("Interrupt Started !!"));
    }
    if (GET_Ma(102) && GET_Ma(200))
    {
        SET_M(102,false);
        SET_M(202,true);
        SET_M(201,false);
        MOTION_STOP_INT(MSlot,AXIS_X);
        SET_MSG(1,TEXT("Interrupt Stop !!"));
    }

    if (GET_Ma(103))
    {
        SET_M(103,false);
        i8094H_MP_CALL(MSlot, MP41);
    }
    Sleep(99);
}
SET_M(111,false);
END_USER_THREAD(7);
SET_MSG(1,TEXT("USER RUN STOP !!"));
SET_D(1,0);

SET_M(99,false);
SET_M(998,true);
return 0;
}
```

利用 **M101** 啓用 **i8094A(H)**中斷功能

利用 **M102** 停止 **i8094A(H)**中斷功能

利用 **M103** 執行 **i8094A(H)MP41** 程序

附錄一 錯誤代碼表

```
#define _NO_ERROR                       0
#define _EXEC_ERROR                     -1
#define _OPENFILE_ERROR                 -2
#define _SETUP_ERROR                    -3
#define _FRAM_INIT_ERROR                -4
#define _REGISTER_ERROR                 -5
#define _NONE_MAOS_ERROR                -6


#define _INCORECT_RUN_MODE              -7


#define _DIVIDE_ZERO_ERROR              -10

#define _SET_Interrupt_ERROR            -20

#define I8048_ERROR_NO_MODULE           -31
#define I8048_ERROR_OPEN_DEVICE         -32
#define I8048_ERROR_INVALID_PARAMETER   -33
#define I8048_ERROR_INVALID_HANDLE      -34
#define I8048_ERROR_CALL_IOCTL          -35
#define I8048_ERROR_GET_IST_EVENT       -36

#define _IN_USE_ERROR                   -50
#define _NO_USE_ERROR                   -51
#define _OUT_OF_RANGE_ERROR             -52

#define _AES_NOT_SETKEY_ERROR           -80
#define _AES_CHECK_ERROR                -81

#define _CREATE_THREAD_ERROR            -90

#define _INOUT_ERROR                    -100


#define _STP_PARAMETER_ERROR            -150
```

```
#define _SYSTEM_VERSION_ERROR          -200
#define _DEVICE_CHECK_ERROR            -201
#define _DEVICE_NOT_INIT               -202
#define _DEVICE_NOT_WinCon             -203


#define _AES_REGCODE_LENTH_ERROR       -250
#define _AES_REGMSG_NO_ERROR           -251

#define _FILE_NOT_OPEN                 -300
// the file not open Please open first
#define _READ_FILE_ERROR               -310
// Read from file error
#define _WRITE_FILE_ERROR              -320
// Write to file error


#define _ETHERNET_CONNECTION_ERROR     1


#define _ALREADY_RUN_WARNING           10

#define _BATTERY1_LEVEL_WARNING        21

#define _BATTERY2_LEVEL_WARNING        22

//for i8094H/A
#define _I8094H_TIMEOUT_ERROR          50

//for i8092/F i8094/F i8094H/A
#define _MOTION_NO_REG_ERROR           51

//for i8094/F
#define _MOTION_OPENCONFIG_ERROR       52

//for i8092/F i8094/F i8094H/A
#define _NON_MOTION_ERROR              53
#define _MOTION_INTP_ALREADY_RUN       54

#define _SYSTEM_NOT_READY              100
```