RS-M8194H

運動控制模組使用手冊

(Version 3.1)

應用程式函式庫

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.

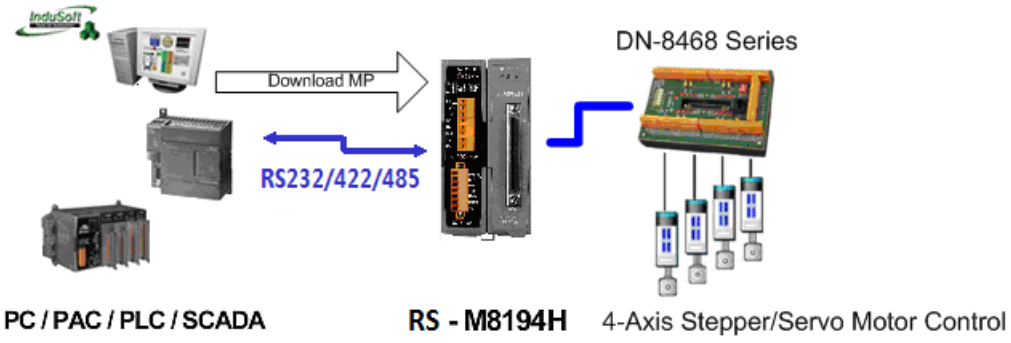
License

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.

1 前言

1.1 簡介

RS-M8194H 是一個使用 RS-232/485/422 為基礎架構的四軸脈波型運動控制器，並且採用 Modbus RTU 作為與 Host 端(如 PC, PAC, PLC… 等)資料交握的通訊協定。此控制器支援各項易於使用的運動控制功能，如兩軸或三軸線性補間(linear interpolation)、兩軸圓弧補間(circular interpolation)、T-curve 與 S-curve 運動速度曲線、多樣性的同步運動與自動原點復歸等。RS-M8194H 也支援 FRnet 擴充介面，可外接 DIO 模組同時控制 128 點輸入與 128 點輸出。FRnet 為泓格科技開發之數位控制現場總線，其具備高速資料傳輸(掃描時間約 2.88 毫秒)與安裝使用容易等特性，使用 FRnet 無論是在系統建置或是後續的維護皆可大幅的降低成本。



此外 RS-M8194H 內建非揮發性(non-volatile)記憶體，可提供使用者儲存自行定義的簡易巨集程序，程序內包含運動控制命令、簡易的算術運算、條件判斷、指令循環執行與內層巨集程序執行(呼叫其他巨集程序)等。可定義超過一百個以上的巨集程序，且命令行數最高可達 512 行(不同的記憶體位置有不同的限制)，Host 端除了可以直接呼叫巨集程序執行外，也可以設定 FRnet DI 或運動控制事件觸發巨集程序執行。

工具軟體 EzMove 提供使用者快速且容易的操作 RS-M8194H，如設定運動控制參數、執行運動控制指令、編輯並下載巨集程序、取得各項狀態與資訊等。使用此工具軟體可有效的幫助使用者熟悉 RS-M8194H 操作。

1.2 函式類別

RS-M8194H 支援四種類型的函式:

RTC (Real Time Command): RTC函式將被放入FIFO緩衝區內(DPRAM)，且採用先進先出的原則依序被執行，若函式回傳 “MB_SLAVE_DEVICE_BUSY” (0x06)訊息，則表示緩衝區已滿。

MP (Macro Program): 使用者可使用MP函式定義簡易的運動控制巨集程序，且此程序將被儲存在i-8094H的非揮發性(non-volatile)記憶體中，執行函式”MP_CALL()”可用來觸發執行已定義的MP程序。

ISR (Interrupt Service Routine): ISR函式相似於MP函式，同樣是用來讓使用者自行定義運動控制巨集程序，差別在於ISR函式所建立的程序，必須由運動控制晶片所發出的中斷訊號來觸發執行，不同的中斷事件可指定執行各自的ISR程序。

IT (Initial Table): IT函式是用來設定與運動控制相關的參數值，使用者可自行定義是否將設定值儲存在i-8094H的非揮發性記憶體中，當RS-M8194H電源開啟時將自動載入記憶體內的設定值進行初始化。

下表為各個巨集程序記憶體空間所能存放的最大函式數量

ISR(6)	ISR1	ISR2	ISR3	ISR4	ISR5	ISR6			
Total:	8	8	8	8	8	8			
ISR(9)	ISR7	ISR8	ISR9	ISR10	ISR11	ISR12	ISR13	ISR14	ISR15
Total:	16	16	16	16	16	16	16	16	16
ISR(3)	ISR16	ISR17	ISR18						
Total:	32	32	32						
ISR(2)	ISR19	ISR20							
Total:	64	64							
MP(40)	MP1	~	MP40						
Total:	8		8						
MP(50)	MP41	~	MP90						
Total:	16		16						
MP(40)	MP91	~	MP130						
Total:	32		32						
MP(20)	MP131	~	MP150						
Total:	64		64						
MP(5)	MP151	MP152	MP153	MP154	MP155				
Total:	128	128	128	128	128				
MP(2)	MP156	MP157							
Total:	512	512							

- 下表所使用的註解符號
  - ◎表示此函式包含在該類別內。
  - ◎x2 表示若使用此函式定義巨集程序將會佔用到兩個函式的記憶體空間。

表 1: 函式列表與所歸屬的類別

章節	函式名稱	RTC	MP	ISR	IT
基本設定					
3.2.1	RSM_RESET_CARD	◎			
3.2.2	RSM_CLEAR_CARD_BUFFER	◎			
3.3	RSM_SET_PULSE_MODE	◎			◎
3.3	RSM_MACRO_SET_PULSE_MODE		◎		◎
3.4	RSM_SET_MAX_V	◎			◎
3.4	RSM_MACRO_SET_MAX_V		◎		◎
3.5	RSM_SET_HLMT	◎			◎
3.5	RSM_MACRO_SET_HLMT		◎		◎
3.6	RSM_LIMITSTOP_MODE	◎			◎
3.6	RSM_MACRO_LIMITSTOP_MODE		◎		◎
3.7	RSM_SET_NHOME	◎			◎
3.7	RSM_MACRO_SET_NHOME		◎		◎
3.8	RSM_SET_HOME_EDGE	◎			◎
3.8	RSM_MACRO_SET_HOME_EDGE		◎		◎
3.9.1	RSM_SET_SLMT	◎			◎
3.9.1	RSM_MACRO_SET_SLMT		◎		◎
3.9.2	RSM_CLEAR_SLMT	◎			◎
3.9.2	RSM_MACRO_CLEAR_SLMT		◎		◎
3.10	RSM_SET_ENCODER	◎			◎
3.10	RSM_MACRO_SET_ENCODER		◎		◎
3.11.1	RSM_SERVO_ON	◎			◎
3.11.1	RSM_MACRO_SERVO_ON		◎		◎
3.11.2	RSM_SERVO_OFF	◎			◎
3.11.2	RSM_MACRO_SERVO_OFF		◎		◎
3.12	RSM_SET_ALARM	◎			◎
3.12	RSM_MACRO_SET_ALARM		◎		◎
3.13	RSM_SET_INPOS	◎			◎
3.13	RSM_MACRO_SET_INPOS		◎		◎
3.14	RSM_SET_FILTER	◎			◎

章節	函式名稱	RTC	MP	ISR	IT
3.14	RSM_MACRO_SET_FILTER		◎		◎
3.15.1	RSM_VRING_ENABLE	◎			◎
3.15.1	RSM_MACRO_VRING_ENABLE		◎		◎
3.15.2	RSM_VRING_DISABLE	◎			◎
3.15.2	RSM_MACRO_VRING_DISABLE		◎		◎
3.16.1	RSM_AVTRI_ENABLE	◎			◎
3.16.1	RSM_MACRO_AVTRI_ENABLE		◎		◎
3.16.2	RSM_AVTRI_DISABLE	◎			◎
3.16.2	RSM_MACRO_AVTRI_DISABLE		◎		◎
3.17.1	RSM_EXD_MP	◎			
3.17.2	RSM_EXD_FP	◎			
3.17.3	RSM_EXD_CP	◎			
3.17.4	RSM_EXD_DISABLE	◎			
3.18.1	RSM_READ_bVAR	◎			
3.18.2	RSM_WRITE_bVAR	◎			
3.18.3	RSM_READ_VAR	◎			
3.18.4	RSM_WRITE_VAR	◎			
3.19.1	RSM_READ_MD	◎			
3.19.2	RSM_WRITE_MD	◎			
運動控制狀態					
4.1.1	RSM_SET_LP	◎			
4.1.1	RSM_MACRO_SET_LP		◎	◎	
4.1.2	RSM_GET_LP	◎			
4.1.2	RSM_MACRO_GET_LP		◎	◎	
4.1.2	RSM_GET_LP_4_AXIS	◎			
4.2.1	RSM_SET_EP	◎			
4.2.1	RSM_MACRO_SET_EP		◎	◎	
4.2.2	RSM_GET_EP	◎			
4.2.2	RSM_MACRO_GET_EP		◎	◎	
4.2.2	RSM_GET_EP_4_AXIS	◎			
4.3.1	RSM_ABS_SET_POSITION	◎			
4.3.1	RSM_MACRO_ABS_SET_POSITION		◎	◎	
4.3.2	RSM_ABS_GET_POSITION	◎			
4.3.2	RSM_MACRO_ABS_GET_POSITION		◎	◎	
4.3.2	RSM_ABS_GET_POSITION_4_AXIS	◎			
4.4	RSM_GET_CV	◎			

章節	函式名稱	RTC	MP	ISR	IT
4.4	RSM_GET_CV_4_AXIS	◎			
4.5	RSM_GET_CA	◎			
4.5	RSM_GET_CA_4_AXIS	◎			
4.6.1	RSM_MACRO_GET_DI		◎	◎	
4.6.2	RSM_GET_DI_ALL	◎			
4.6.2	RSM_MACRO_GET_DI_ALL		◎	◎	
4.6.2	RSM_GET_DI_ALL_4_AXIS	◎			
4.7.1	RSM_MACRO_GET_DI_SIGNAL		◎	◎	
4.7.2	RSM_GET_DI_SIGNAL_ALL	◎			
4.7.2	RSM_MACRO_GET_DI_SIGNAL _ALL		◎	◎	
4.7.2	RSM_GET_DI_SIGNAL_ALL_4_AXIS	◎			
4.8	RSM_GET_HOME_SEARCH_STATE	◎			
4.8	RSM_MACRO_GET_HOME_SEARCH_STATE		◎	◎	
4.8	RSM_GET_HOME_SEARCH_STATE _4_AXIS	◎			
4.9.1	RSM_GET_ERROR_STATE	◎			
4.9.1	RSM_MACRO_GET_ERROR		◎	◎	
4.9.2	RSM_GET_ERROR_CODE	◎			
4.9.2	RSM_MACRO_GET_ERROR_CODE		◎	◎	
4.9.2	RSM_GET_ERROR_CODE_4_AXIS	◎			
4.10	RSM_GET_FREE_BUFFER	◎			
4.11	RSM_GET_STOP_STATUS	◎			
4.11	RSM_GET_STOP_STATUS_4_AXIS	◎			
4.12	RSM_GET_EMERGENCY_STATE	◎			
4.13	RSM_GET_DRIVING_AXIS	◎			
4.14	RSM_GET_INTERPOL_RDY_FLAG	◎			
4.15	RSM_GET_TRIG_INTFACTOR	◎			
4.16.2	RSM_GET_DEVICE_STATE	◎			
FRnet 功能					
5.1.1	RSM_MACRO_FRNET_IN		◎	◎	
5.1.2	RSM_MACRO_FRNET_READ		◎	◎	
5.1.3	RSM_FRNET_READ_SINGLE_DIO	◎			
5.1.4	RSM_FRNET_READ_GROUP_DIO	◎			
5.1.5	RSM_FRNET_READ_MULTI_GROUP_DIO	◎			
5.2.1	RSM_MACRO_FRNET_OUT		◎	◎	
5.2.2	RSM_MACRO_FRNET_WRITE		◎	◎	
5.2.3	RSM_FRNET_WRITE_SINGLE_DO	◎			
5.2.4	RSM_FRNET_WRITE_GROUP_DO	◎			
5.2.5	RSM_FRNET_WRITE_MULTI_GROUP_DO	◎			

章節	函式名稱	RTC	MP	ISR	IT
5.3	RSM_MACRO_FRNET_WAIT		◎	◎	
5.4	RSM_SET_FRNET_TRIGGER_EVENT	◎			
5.5	RSM_GET_FRNET_TRIGGER_EVENT_SETTING	◎			
自動原點復歸					
6.1	RSM_SET_HV	◎			
6.1	RSM_MACRO_SET_HV		◎		
6.2	RSM_HOME_LIMIT	◎			
6.2	RSM_MACRO_HOME_LIMIT		◎		
6.3	RSM_SET_HOME_MODE	◎			
6.3	RSM_MACRO_SET_HOME_MODE		◎		
6.4	RSM_HOME_START	◎			
6.4	RSM_MACRO_HOME_START		◎		
運動控制命令					
7.1.1	RSM_NORMAL_SPEED	◎			
7.1.1	RSM_MACRO_NORMAL_SPEED		◎	◎	
7.1.2	RSM_SET_SV	◎			
7.1.2	RSM_MACRO_SET_SV		◎	◎	
7.1.3	RSM_SET_V	◎			
7.1.3	RSM_MACRO_SET_V		◎	◎	
7.1.4	RSM_SET_A	◎			
7.1.4	RSM_MACRO_SET_A		◎	◎	
7.1.5	RSM_SET_D	◎			
7.1.5	RSM_MACRO_SET_D		◎	◎	
7.1.6	RSM_SET_K	◎			
7.1.6	RSM_MACRO_SET_K		◎	◎	
7.1.7	RSM_SET_L	◎			
7.1.7	RSM_MACRO_SET_L		◎	◎	
7.1.8	RSM_SET_AO	◎			
7.1.8	RSM_MACRO_SET_AO		◎	◎	
7.1.9	RSM_FIXED_MOVE	◎			
7.1.9	RSM_MACRO_FIXED_MOVE		◎	◎	
7.1.10	RSM_SET_PULSE	◎			
7.1.10	RSM_MACRO_SET_PULSE		◎	◎	
7.1.11	RSM_ABS_FIXED_MOVE	◎			
7.1.11	RSM_ABS_MACRO_FIXED_MOVE		◎	◎	
7.1.12	RSM_CONTINUE_MOVE	◎			
7.1.12	RSM_MACRO_CONTINUE_MOVE		◎	◎	
7.2.1	RSM_AXIS_ASSIGN	◎			
7.2.1	RSM_MACRO_AXIS_ASSIGN		◎	◎	
7.2.2	RSM_VECTOR_SPEED	◎			
7.2.2	RSM_MACRO_VECTOR_SPEED		◎	◎	
7.2.3	RSM_SET_VSV	◎			
7.2.3	RSM_MACRO_SET_VSV		◎	◎	

章節	函式名稱	RTC	MP	ISR	IT
7.2.4	RSM_SET_VV	◎			
7.2.4	RSM_MACRO_SET_VV		◎	◎	
7.2.5	RSM_SET_VA	◎			
7.2.5	RSM_MACRO_SET_VA		◎	◎	
7.2.6	RSM_SET_VD	◎			
7.2.6	RSM_MACRO_SET_VD		◎	◎	
7.2.7	RSM_SET_VK	◎			
7.2.7	RSM_MACRO_SET_VK		◎	◎	
7.2.8	RSM_SET_VL	◎			
7.2.8	RSM_MACRO_SET_VL		◎	◎	
7.2.9	RSM_SET_VAO	◎			
7.2.9	RSM_MACRO_SET_VAO		◎	◎	
7.2.10	RSM_LINE_2D	◎			
7.2.10	RSM_MACRO_LINE_2D		◎	◎	
7.2.11	RSM_ABS_LINE_2D	◎			
7.2.11	RSM_ABS_MACRO_LINE_2D		◎	◎	
7.2.12	RSM_LINE_3D	◎			
7.2.12	RSM_MACRO_LINE_3D		◎	◎	
7.2.13	RSM_ABS_LINE_3D	◎			
7.2.13	RSM_ABS_MACRO_LINE_3D		◎	◎	
7.2.14	RSM_ARC_CW	◎x2			
7.2.14	RSM_MACRO_ARC_CW		◎x2	◎x2	
7.2.15	RSM_ARC_CCW	◎x2			
7.2.15	RSM_MACRO_ARC_CCW		◎x2	◎x2	
7.2.16	RSM_ABS_ARC_CW	◎x2			
7.2.16	RSM_ABS_MACRO_ARC_CW		◎x2	◎x2	
7.2.17	RSM_ABS_ARC_CCW	◎x2			
7.2.17	RSM_ABS_MACRO_ARC_CCW		◎x2	◎x2	
7.2.18	RSM_CIRCLE_CW	◎			
7.2.18	RSM_MACRO_CIRCLE_CW		◎	◎	
7.2.19	RSM_CIRCLE_CCW	◎			
7.2.19	RSM_MACRO_CIRCLE_CCW		◎	◎	
7.3.1	RSM_SYNC_ACTION	◎x2			
7.3.1	RSM_MACRO_SYNC_ACTION		◎x2	◎x2	
7.3.2	RSM_CLEAR_SYNC_ACTION	◎			
7.3.2	RSM_MACRO_CLEAR_SYNC_ACTION		◎	◎	
7.3.3	RSM_SET_ACTIVATION_FACTORS	◎			
7.3.3	RSM_MACRO_SET_ACTIVATION_FACTORS		◎	◎	
7.3.4	RSM_SET_ACTIVATION_AXIS	◎			
7.3.4	RSM_MACRO_SET_ACTIVATION_AXIS		◎	◎	
7.3.5	RSM_SET_ACTION	◎			
7.3.5	RSM_MACRO_SET_ACTION		◎	◎	
7.3.6	RSM_SET_COMPARE	◎			
7.3.6	RSM_MACRO_SET_COMPARE		◎	◎	
7.3.7	RSM_GET_LATCH	◎			
7.3.7	RSM_MACRO_GET_LATCH		◎	◎	

章節	函式名稱	RTC	MP	ISR	IT
7.3.7	RSM_GET_LATCH_4_AXIS	◎			
7.3.8	RSM_SET_PRESET	◎			
7.3.8	RSM_MACRO_SET_PRESET		◎	◎	
7.3.9	RSM_SET_OUT	◎			
7.3.9	RSM_MACRO_SET_OUT		◎	◎	
7.3.10	RSM_ENABLE_INT	◎			
7.3.10	RSM_MACRO_ENABLE_INT		◎		
7.3.11	RSM_DISABLE_INT	◎			
7.3.11	RSM_MACRO_DISABLE_INT		◎		
7.3.12	RSM_INTFACTOR_ENABLE	◎			
7.3.12	RSM_MACRO_INTFACTOR_ENABLE		◎	◎	
7.3.13	RSM_INTFACTOR_DISABLE	◎			
7.3.13	RSM_MACRO_INTFACTOR_DISABLE		◎	◎	
7.4.1	RSM_RECTANGLE	◎x4			
7.4.1	RSM_MACRO_RECTANGLE		◎x4		
7.4.2	RSM_LINE_2D_INITIAL	◎x2			
7.4.2	RSM_MACRO_LINE_2D_INITIAL		◎x2		
7.4.3	RSM_LINE_2D_CONTINUE	◎			
7.4.3	RSM_MACRO_LINE_2D_CONTINUE		◎		
7.4.4	RSM_ABS_LINE_2D_CONTINUE	◎			
7.4.4	RSM_ABS_MACRO_LINE_2D_CONTINUE		◎		
7.4.5	RSM_LINE_3D_INITIAL	◎x2			
7.4.5	RSM_MACRO_LINE_3D_INITIAL		◎x2		
7.4.6	RSM_LINE_3D_CONTINUE	◎			
7.4.6	RSM_MACRO_LINE_3D_CONTINUE		◎		
7.4.7	RSM_ABS_LINE_3D_CONTINUE	◎			
7.4.7	RSM_ABS_MACRO_LINE_3D_CONTINUE		◎		
7.4.8	RSM_MIX_2D_INITIAL	◎x2			
7.4.8	RSM_MACRO_MIX_2D_INITIAL		◎x2		
7.4.9	RSM_MIX_2D_CONTINUE	◎x2			
7.4.9	RSM_MACRO_MIX_2D_CONTINUE		◎x2		
7.4.10	RSM_ABS_MIX_2D_CONTINUE	◎x2			
7.4.10	RSM_ABS_MACRO_MIX_2D_CONTINUE		◎x2		
7.4.11	RSM_HELIX_3D	◎x3			
7.4.11	RSM_MACRO_HELIX_3D		◎x3		
7.4.12	RSM_LINE_SCAN	◎			
7.4.13	RSM_LINE_SCAN_START	◎			
7.4.14	RSM_GET_LINE_SCAN_DONE	◎			
7.5.1	RSM_DRV_HOLD	◎			
7.5.1	RSM_MACRO_DRV_HOLD		◎	◎	
7.5.2	RSM_DRV_START	◎			
7.5.2	RSM_MACRO_DRV_START		◎	◎	
7.5.3	RSM_STOP_SLOWLY	◎			
7.5.3	RSM_MACRO_STOP_SLOWLY		◎		
7.5.4	RSM_STOP_SUDDENLY	◎			
7.5.4	RSM_MACRO_STOP_SUDDENLY		◎		

章節	函式名稱	RTC	MP	ISR	IT
7.5.5	RSM_VSTOP_SLOWLY	◎			
7.5.5	RSM_MACRO_VSTOP_SLOWLY		◎		
7.5.6	RSM_VSTOP_SUDDENLY	◎			
7.5.6	RSM_MACRO_VSTOP_SUDDENLY		◎		
7.5.7	RSM_CLEAR_STOP	◎			
7.5.7	RSM_MACRO_CLEAR_STOP		◎		
7.5.8	RSM_CLEAR_VSTOP	◎			
7.5.8	RSM_MACRO_CLEAR_VSTOP		◎		
7.5.9	RSM_EMERGENCY_STOP	◎			
7.5.10	RSM_CLEAR_EMERGENCY_STOP	◎			
初始參數表					
8.1	RSM_LOAD_INITIAL	◎			
巨集程序					
9.1.1	RSM_MP_CREATE	◎			
9.1.2	RSM_MACRO_MP_CLOSE		◎		
9.1.3	RSM_MP_CALL	◎			
9.1.3	RSM_MACRO_MP_CALL		◎		
9.2.1	RSM_MP_ISR_CREATE	◎			
9.2.2	RSM_MACRO_MP_ISR_CLOSE			◎	
9.2.3	RSM_MP_ISR_CALL	◎			
9.3.1	RSM_MACRO_SET_VAR		◎	◎	
9.3.2	RSM_MACRO_SET_RVAR		◎	◎	
9.4	RSM_MACRO_VAR_CALCULATE		◎	◎	
9.5.1	RSM_MACRO_FOR		◎		
9.5.2	RSM_MACRO_NEXT		◎		
9.5.3	RSM_MACRO_EXIT_FOR		◎		
9.6.1	RSM_MACRO_IF		◎	◎	
9.6.2	RSM_MACRO_ELSE		◎	◎	
9.6.3	RSM_MACRO_END_IF		◎	◎	
9.7.1	RSM_MACRO_GOTO		◎		
9.7.2	RSM_MACRO_LABEL		◎		
9.8	RSM_MACRO_TIMER		◎		
9.9	RSM_STOP_WAIT	◎			
9.9	RSM_MACRO_STOP_WAIT		◎		
9.10	RSM_MACRO_EXIT_MACRO		◎	◎	
9.11	RSM_MP_TERMINATE	◎			
9.11	RSM_MACRO_MP_TERMINATE		◎	◎	
9.12	RSM_GET_MP_DOWNLOAD_STATUS	◎			
9.13	RSM_CHANGE_VEL_ON_FLY	◎			
版本資訊					
11.1	RSM_GET_RSM_FIRMWARE_VERSION	◎			
11.2	RSM_GET_i8094H_FIRMWARE_VERSION	◎			
11.3	RSM_GET_DLL_VERSION	◎			

1.3 函式說明的基本結構

函式名稱 (*參數 1, 參數 2 ……*)

說明: 函式的功能說明。

類別: 函式所屬類別。

參數: 參數的定義及使用方法。

回傳值: 函式的回傳值。

範例: 簡易的範例程式。

備註: 註譯。

Modbus 命令的 RTU 訊框格式定義如下:
(1) UID: 單元識別碼 (Unit ID)，即函式內的" SlaveAddr"參數。
(2) FC: 功能碼(Function Code)。
(3) St_Addr.: 起始暫存器位置。
(4) Word Count: 所要操作的暫存器長度 (單位 Word, 16-bit)。.
(5) Byte Count: 所要操作的暫存器長度(單位 Byte, 8-bit)。
(6) Register: 參數值內容 (16-bit)。
(6-1) Sub_Function Code: RS-M8194H 所預先定義的功能碼。
(6-2) Axis: 在使用 RS-M8194H 功能時所要操作的軸號。

Modbus 的暫存器只支援 16 位元的資料格式，因此必須使用兩個暫存器來表示 32 位元的資料，當字組順序 (WORD order)設定為 0 時，將使用第一個暫存器表示高字組 (MSW)，第二個暫存器表示低字組 (LSW)。

備註:

- **在後續的函式說明章節內所使用的註解符號**
  - **※ 表示此函式可用來定義MP巨集程序。**
  - **Δ 表示此函式可用來定義ISR巨集程序。**

2 通訊函式

2.1 串列 通訊函式

本章節將介紹 PC (或其他 Modbus RTU master)與 RS-M8194H 之間連接埠(COM Port)通訊相關的操作函式。

2.1.1 開啟連接埠

- **HANDLE RSM_OPEN_COM_PORT (BYTE *bPort*,**
  DWORD *dwBaudRate*,
  BYTE *bDataBits*,
  BYTE *bParityBit*,
  BYTE *bStopBits*,
  eCOM_PORT **pError*)

說明:

開啟PC與RS-M8194H之間的連接埠。在開始傳送Modbus命令前須先確認通訊已建立(每個連線都將回傳唯一的handle)，可使用工具軟體EzMove設定通訊相關的參數。請注意，在結束RS-M8194H操作前應使用函式" RSM_CLOSE_COM_PORT ()"斷開連線與釋放內部記憶體。

參數:

參數名稱	說明
***bPort*:**	與 RS-M8194H 連接的串列連接埠編號
dwBaudRate:	RS-M8194H 的鮑率設定值
bDataBits:	RS-M8194H 的資料位元(data bit)
bParityBit:	RS-M8194H 的檢查位元(parity bit)
bStopBits:	RS-M8194H 的停止位元(stop bit)
pError:	0: 成功開啟連接埠 其他:請參閱表2說明

回傳值:

INVALID_HANDLE_VALUE 或 NULL : 表示連線失敗； 其他: 表示連線成功。

範例:[VC++]

```
#include "RS_M8194H_API.h"
```

```
Handle hRSM1;
eRTU iRet;
eCOM_PORT  Error
hRSM1 = RSM_OPEN_COM_PORT(bPort, dwBaudRate, bDataBits, bParityBit,
bStopBits, & Error); // Connection with the first controller
if( (hRsm1 == INVALID_HANDLE_VALUE) || (Error != COMM_SUCCESS) )
{return;}
iRet = RSM_SERVO_ON (hRSM1, 1, AXIS_XYZU); //Set servo on
```

2.1.2 關閉連接埠

eRTU RSM_CLOSE_COM_PORT (HANDLE *hRsm*, BYTE *bPort*)

說明:

關閉PC的串列連接埠

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
bPort:	**PC 與 RS-M8194H 的串列連接埠編號**

回傳值:

0: 成功; 其他:失敗 (詳細說明請參考章節 2.2)。

2.1.3 取得連線狀態

eRTU RSM_CONNECTION_STATE (HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

取得串列通訊的連線狀態。

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)

回傳值:

1: 已連線; 0: 斷線。

2.1.4 設定 Modbus 回應逾時

eCOM_PORT RSM_SET_TIMEOUT(HANDLE *hRsm*, DWORD *dwTimeOut*)

說明:

設定PC與RS-M8194H之間的命令通訊逾時。在指定的時間內若無法完成命令的傳送將回傳錯誤訊息。

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
dwTimeOut:	**命令通訊逾時設定值，單位:毫秒(milliseconds)** **建議設定值勿小於 50 毫秒**

回傳值:

0: 成功; 其他:失敗 (詳細說明請參考章節 2.2)。

2.1.5 取得 Modbus 回應逾時

eRTU RSM_GET_TIMEOUT(HANDLE *hRsm*, DWORD* *dwTimeOut*)

說明:

取得PC與RS-M8194H之間的命令通訊逾時設定值。

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
dwTimeOut:	**命令通訊逾時設定值，單位:毫秒(milliseconds)**

回傳值:

0: 成功; 其他:失敗 (詳細說明請參考章節 2.2)。

2.1.6 設定 Modbus 需求命令的傳送延遲時間

```
eRTU RSM_SET_PACKET_DELAY(HANDLE hRsm,
                          DWORD dwSendDelay)
```

說明:

設定需求命令的傳送延遲時間。當接收到回應訊息後將延遲指定的時間再發送下一個需求命令。

參數:

參數名稱	說明
hRsm:	連接埠 handle
dwSendDelay:	需求命令的傳送延遲時間設定值，單位:毫秒 (milliseconds) 預設值為 0 ms

回傳值:

0: 成功; 其他:失敗 (詳細說明請參考章節 2.2)。

2.1.7 取得 Modbus 需求命令的傳送延遲時間

```
eRTU RSM_GET_PACKET_DELAY(HANDLE hRsm,
                          DWORD* dwSendDelay)
```

說明:

取得需求命令的傳送延遲時間設定值。

參數:

參數名稱	說明
hRsm:	連接埠 handle
dwSendDelay:	需求命令的傳送延遲時間設定值，單位:毫秒 (milliseconds)

回傳值:

0: 成功; 其他:失敗 (詳細說明請參考章節 2.2)。

2.1.8 設定 Modbus 暫存器順序

eRTU RSM_SET_WORD_ORDER (HANDLE *hRsm*, BYTE *SlaveAddr* , BYTE *bWordOrder*)

說明:

設定最高字組(Most Significant WORD, MSW)與最低字組(Least Significant WORD, LSW)的順序。因為Modbus的暫存器只支援16位元的資料格式，所以必須使用兩個暫存器來表示32位元的資料。

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
bWordOrder:	**0: 設定第一個暫存器為 MSW** **1: 設定第二個暫存器為 MSW**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節 2.2) 。

2.1.9 取得 Modbus 暫存器順序

eRTU RSM_GET_WORD_ORDER (HANDLE *hRsm*, BYTE *SlaveAddr* , BYTE* *pbWordOrder*)

說明:

取得Modbus暫存器的字組順序設定。

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
bWordOrder:	**0: 第一個暫存器為 MSW** **1: 第二個暫存器為 MSW**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節 2.2) 。

2.2 回傳碼

本章節將說明在開啟連接埠時所回應的錯誤回傳碼。

表 2: 開啟連接埠的錯誤表

錯誤表		
回傳碼名稱	**代碼**	**說明**
COMM_SUCCESS	**0**	成功開啟連接埠。
COMM_RSM8194H_NOT_CONNECTED	**99**	RS-M8194H 模組未連接到 PC。
COMM_PORT_NOT_EXIST	**100**	在 PC 內不存在此連接埠。
COMM_PORT_IN_USE	**101**	連接埠已被開啟。
COMM_PORT_OPEN_ERROR	**102**	連接埠無法被開啟。
COMM_BAUDRATE_ERR	**103**	不支援輸入的鮑率設定值。
COMM_DATABITS_ERR	**104**	不正確的資料位元(data bit)設定，目前只支援8資料位元。
COMM_STOPBITS_ERR	**105**	不支援的停止位元(stop bit)設定。 目前支援的設定值為: 0: 1 停止位元 1: 1.5停止位元 2: 2停止位元
COMM_PARITY_ERR	**106**	錯誤的檢查位元(parity bit)設定， 目前支援的設定值為: 0: 不檢查 1: 奇同位位元(odd parity) 2: 偶同位位元(even parity)
COMM_PORT_NOT_OPEN	**107**	連接埠未開啟，讀寫通訊狀態前須先開啟連接埠。
COMM_WRITE_ERROR	**108**	傳送資料到連接埠時發生異常。
COMM_READ_ERROR	**109**	從連接埠讀取資料時發生異常。
COMM_CLOSE_ERROR	**110**	無法正常關閉已開啟的連接埠。
COMM_GET_STATE_ERR	**140**	內部錯誤->通訊埠已被關閉。
COMM_SET_STATE_ERR	**141**	內部錯誤->通訊埠已被關閉。
COMM_GET_TIMEOUT_ERR	**142**	內部錯誤。
COMM_SET_TIMEOUT_ERR	**143**	內部錯誤。
COMM_CLEAR_BUFFER_ERROR	**144**	清除輸入與輸出連接埠緩衝區時發生異常。

取得函式的回傳值即表示命令已被成功傳送到 **RS-M8194H**。

表 3: 函式回傳值

回傳值名稱	代碼	說明
RSM_SUCCESS	**0**	無錯誤發生，RS-M8194H 已成功接收命令。
MB_ILLEGAL_FUNCTION_CODE	**1**	不支援的 Modbus 功能碼。
MB_ILLEGAL_DATA_ADDRESS	**2**	表示 Modbus 命令的起始位址設定錯誤。操作 32 位元資料時請依照 WORD order 設定位址。
MB_ILLEGAL_DATA_VALUE	**3**	表示 Modbus 命令之設定值錯誤，如:參數數量、軸號、MD 位址、VAR 位址、bVAR 位址等設定錯誤。 使用到不支援的函式，請確認韌體版本是否正確。
MB_SLAVE_DEVICE_FAILURE	**4**	表示 RS-M8194H 偵測不到 i-8094H。
MB_SLAVE_DEVICE_BUSY	**6**	表示 i-8094H 的內部緩衝區已滿(DPRAM-Buffer)，請確認後再重新傳送指令。
WRONG_CARD_NO	**11**	無效的卡號. 目前只支援設定卡號等於 1。
MB_ILLEGAL_SLAVE_ADDRESS	**201**	無效的 slave address 設定值，可設定範圍: 1 ~ 247。
MB_SEND_ERROR	**202**	由 Modbus master 傳送到 slave 的資料內容不完整，Modbus RTU 訊框的資料字組(byte)數量少於需求數量。 可能發生此問題的原因:傳送資料所需的時間超過逾時設定值。
MB_TIMEOUT	**203**	等待回應訊框發生逾時。
MB_FRAME_LENGTH_ERROR	**206**	傳送的需求命令將導致回應訊框的資料量超出允許範圍(上限為256 byte)。 可能發生此問題的原因: 暫存器讀取數量超出上限值。
與回應訊框相關的錯誤碼		
MB_INVALID_SLAVE_ADDRESS	**210**	回應訊框與需求訊框內的slave位址不一致。
MB_INVALID_FUNCTION_CODE	**211**	回應訊框與需求訊框內的功能碼不一致。
MB_REGISTER_READ_WRITE_ERROR	**212**	回應訊息的暫存器數量與需求不一致。
MB_STARTING_REGISTER_ERROR	**213**	回應訊框內的起始暫存器與傳送的需求訊框不同。
MB_COIL_READ_WRITE_ERROR	**214**	回應訊息的discrete數值與需求不一致。

MB_STARTING_COIL_ERROR	**215**	回應訊框內的起始discrete 位址與傳送的需求訊框不同。
MB_CRC_ERROR	**216**	由Master連接埠接收的資料發生CRC檢查錯誤。
RTU_INVALID_HANDLE	**-5**	無效 handle。
RTU_UNDEFINED_ERROR	**-6**	內部錯誤。
RTU_PAR_ERROR	**-7**	參數設定值超出範圍。
RTU_WRITE_DENY	**-8**	存取拒絕(寫入保護狀態)。
RTU_PORT_NOT_OPEN	**-9**	連接埠尚未開啟。
RTU_COM_PORT_ERROR	**-10**	內部連接埠通訊錯誤。

3 基本設定

3.1 軸參數代碼定義

所有功能中有關軸參數，皆是以 X=1、Y=2、Z=4、U=8 作為代碼，假設我們要指定 XY=3，就是 1+2=3，又如 YZ=0x6 (2+4=6)，依此類推，XYZU=0xf (1+2+4+8)，因此同一功能，可以一次做單軸設定，也可以一次設多軸相同設定，所有功能中有關軸參數代碼 (BYTE axis)與其代表的意義如下表所示:

表 4: 軸號分配與對應代碼

軸號	X	Y	Z	U	XY	XZ	XU	YZ
代碼	0x1	0x2	0x4	0x8	0x3	0x5	0x9	0x6
變數	AXIS_X	AXIS_Y	AXIS_Z	AXIS_U	AXIS_XY	AXIS_XZ	AXIS_XU	AXIS_YZ
軸號	YU	ZU	XYZ	XYU	XZU	YZU	XYZU	
代碼	0xa	0xc	0x7	0xb	0xd	0xe	0xf	
變數	AXIS_YU	AXIS_ZU	AXIS_XYZ	AXIS_XYU	AXIS_XZU	AXIS_YZU	AXIS_XYZU	

在執行關於運動控制參數的設定函式時，若使用下表的軸參數代碼，則參數值可以被儲存在i-8094H的非揮發性記憶體中。請注意，在巨集程序中(MP或ISR)不可使用此代碼。

表 5: 使用於初始表(IT)的軸號分配與對應代碼

軸號	X	Y	Z	U	XY	XZ	XU	YZ
代碼	0x11	0x12	0x14	0x18	0x13	0x15	0x19	0x16
變數	INITIAL_X	INITIAL_Y	INITIAL_Z	INITIAL_U	INITIAL_XY	INITIAL_XZ	INITIAL_XU	INITIAL_YZ
軸號	YU	ZU	XYZ	XYU	XZU	YZU	XYZU	
代碼	0x1a	0x1c	0x17	0x1b	0x1d	0x1e	0x1f	
變數	INITIAL_YU	INITIAL_ZU	INITIAL_XYZ	INITIAL_XYU	INITIAL_XZU	INITIAL_YZU	INITIAL_XYZU	

3.2 運動模組重置

3.2.1 重置為初始設定

eRTU RSM_RESET_CARD (HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

此函式會將運動模組(i-8094H)恢復到電源開啟時的初始設定。

類別:

Modubus sub_function; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

RSM_RESET_CARD (hRsm, 1); //Reset the module1

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	0A 0A	*Sub_function code*			

3.2.2 清除命令緩衝區

eRTU RSM_CLEAR_CARD_BUFFER (HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

清除i-8094H命令緩衝區內的所有資料。

類別:

Modbus sub_function; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
RSM_CLEAR_CARD_BUFFER (hRsm, 1);        //clear data buffer in module 1
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 01	02
Register []	Value (hex)	Remarks			
0	0A 0B	*Sub_function code*			

使用函式"RSM_GET_FREE_BUFFER()"可讀取i-8094H命令緩衝區內尚未使用的指令記憶體區塊數量(最大值為30)。

3.3 脈波輸出模式設定

eRTU RSM_SET_PULSE_MODE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nMode*)

※ eRTU RSM_MACRO_SET_PULSE_MODE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nMode*)

說明:

設定軸之控制脈波輸出模式。

類別:

Modbus sub_function; RTC, MP and IT.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表 4 與表 5) 若使用表 5 的代碼設置，則參數值將被儲存到記憶體內
***nMode*:**	模式設定值 (請參照下表說明)

脈波輸出模式

	nMode	脈波輸出訊號	
		nPP	nPM
CW / CCW	0	CW(上緣觸發)	CCW(上緣觸發)
	1	CW(下緣觸發)	CCW(下緣觸發)
PULSE / DIR	2	PULSE (上緣觸發)	DIR (LOW:+dir/ HIGH:-dir)
	3	PULSE (下緣觸發)	DIR (LOW:+dir/ HIGH:-dir)
	4	PULSE (上緣觸發)	DIR (HIGH:+dir/ LOW:-dir)
	5	PULSE (下緣觸發)	DIR (HIGH:+dir/ LOW:-dir)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_PULSE_MODE 的Sub_function代碼為 0x0A 0C。
RSM_MACRO_SET_PULSE_MODE的Sub_function代碼為 0x0C 0C。

Modbus 範例:

RSM_SET_PULSE_MODE (hRsm, 1, AXIS_XYZ, 2);
// set the pulse mode of X, Y, and Z axes to be mode 2 for module 1

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0A 0C/0C 0C	*Sub_function code*			
1	00 07	*axis*			
2	00 02	*nMode*			

範例:

```
RSM_SET_PULSE_MODE (hRsm, 1, AXIS_XYZ, 2);
// set the pulse mode of X, Y, and Z axes to be mode 2 in module 1
RSM_SET_PULSE_MODE(hRsm, 1, AXIS_U, 3);
//set the pulse mode of U axis to be mode 3 in module 1
RSM_SET_PULSE_MODE (hRsm, 1, INITIAL_XYZU, 0);
//set the pulse mode of X Y Z U axes to be mode 0, write into the initial
parameter table (table 4) in module 1
```

3.4 速度上限值設定

eRTU RSM_SET_MAX_V (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

※ eRTU RSM_MACRO_SET_MAX_V (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

說明:

設定運動控制速度上限值。此設定值會改變控制脈波輸出速度的解析度，當設定值越大則解析度越低。由於基本速度區段為1~8000，因此若設定最大速度為8000，則表示解析度為1PPS，若設定最大速度為16000則解析度為2PPS，依此類推，最大速度的設定上限值為4,000,000，此時解析度為500PPS。當解析度改變時會影響其他相關的速度參數計算，因此請謹慎的使用本函式。

類別:

Modbus sub_function; RTC, MP and IT.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表 4 與表 5) 若使用表 5 的代碼設置，則參數值將被儲存到記憶體內
data:	速度上限值 執行單軸運動時可設定的範圍: 8,000 ~ 4,000,000 PPS 執行兩軸補間運動時可設定的範圍: 8,000 ~ 2,828,854 PPS; 執行三軸補間運動時可設定的範圍: 8,000~2,309,468 PPS.

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_MAX_V的Sub_function代碼為0x0A 0D。
RSM_MACRO_SET_MAX_V的Sub_function代碼為0x0C 0D。

Modbus 範例:

```
RSM_SET_MAX_V (hRsm, 1, AXIS_XY, 200000L);
//The maximum speed for the X and Y axes of module 1 is 200KPPS.
//The resolution of the speed will be 200000/8000 = 25 PPS.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 0D/0C 0D	*Sub_function code*			
1	00 03	*axis*			
2	00 03	*MSW of data*			
3	0D 40	*LSW of data (200000 = 0x30D40)*			

3.5 硬體極限訊號觸發準位設定

eRTU RSM_SET_HLMT (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nFLEdge*, BYTE *nRLEdge*)

※ eRTU RSM_MACRO_SET_HLMT (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nFLEdge*, BYTE *nRLEdge*)

說明:

設定硬體極限開關的訊號觸發邏輯準位。

類別:

Modbus sub_function; RTC, MP and IT.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
axis:	**軸號設定值 (請參照表 4 與表 5)** **若使用表 5 的代碼設置，則參數值將被儲存到記憶體內**
***nFLEdge*:**	**正方向極限觸發邏輯:** **0=低準位觸發；1=高準位觸發**
***nRLEdge*:**	**反方向極限觸發邏輯:** **0=低準位觸發；1=高準位觸發**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_HLMT的Sub_function代碼為 0x0A 0E。
RSM_MACRO_SET_HLMT的Sub_function代碼為 0x0C 0E。

Modbus 範例:

```
RSM_SET_HLMT (hRsm, 1, AXIS_XYZU, 0, 0);
//set all the trigger levels as low-active for all limit switches on module 1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 0E/0C 0E	*Sub_function code*			
1	00 0F	*axis*			
2	00 00	*nFLEdge*			
3	00 00	*nRLEdge*			

3.6 極限開關觸發時的停止方式設定

eRTU RSM_LIMITSTOP_MODE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nMode*)

※ eRTU RSM_MACRO_LIMITSTOP_MODE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nMode*)

說明:

設定當極限開關訊號觸發時的運動停止模式。

類別:

Modbus sub_function; RTC, MP and IT.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表 4 與表 5) 若使用表 5 的代碼設置，則參數值將被儲存到記憶體內
***nMode*:**	設定處理方法: 0=立即停止；1=減速後停止

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_LIMITSTOP_MODE的Sub_function代碼為 0x0A 0F。
RSM_MACRO_LIMITSTOP_MODE的Sub_function代碼為 0x0C 0F。

Modbus 範例:

```
RSM_LIMITSTOP_MODE (hRsm, 1, AXIS_X, 0);
//set X axis to stop immediately if any limit switch on X axis is turned on.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0A 0F/0C 0F	***Sub_function code***			
1	00 01	***axis***			
2	00 00	***nMode***			

3.7 近原點訊號觸發準位設定

eRTU RSM_SET_NHOME (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nNHEdge*)

※ eRTU RSM_MACRO_SET_NHOME (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nNHEdge*)

說明:

設定近原點(NHOME)訊號觸發邏輯準位。

類別:

Modbus sub_function; RTC, MP and IT.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表 4 與表 5) 若使用表 5 的代碼設置，則參數值將被儲存到記憶體內
nNHEdge:	近原點觸發邏輯: 0=低準位觸發；1=高準位觸發

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_NHOME的Sub_function代碼為 0x0A 10。
RSM_MACRO_SET_NHOME的Sub_function代碼為 0x0C 10。

Modbus 範例:

```
RSM_SET_NHOME (hRsm, 1, AXIS_XY, 0);
// set the trigger level of NHOME of X and Y axes on module 1 to be active low.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0A 10/0C 10	*Sub_function code*			
1	00 03	*Axis (3 ➔ AXIS_XY)*			
2	00 00	*nNHEdge*			

3.8 原點訊號觸發準位設定

eRTU RSM_SET_HOME_EDGE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nHEdge*)

※ eRTU RSM_MACRO_SET_HOME_EDGE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nHEdge*)

說明:

設定原點(HOME)訊號觸發邏輯準位。

類別:

Modbus sub_function; RTC, MP and IT.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表 4 與表 5) 若使用表 5 的代碼設置，則參數值將被儲存到記憶體內
***nHEdge*:**	原點觸發邏輯: 0=低準位觸發；1=高準位觸發

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_HOME_EDGE的Sub_function代碼為 0x0A 11。
RSM_MACRO_SET_HOME_EDGE的Sub_function代碼為 0x0C 11。

Modbus 範例:

```
RSM_SET_HOME_EDGE (hRsm, 1, AXIS_XYZU, 1);
//set the trigger level as high active for all home sensors on module 1
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0A 11/0C 11	*Sub_function code*			
1	00 0F	*Axis (F ➔ AXIS_XYZU)*			
2	00 01	*nHEdge*			

3.9 軟體極限設定

3.9.1 軟體極限位置設定

eRTU RSM_SET_SLMT (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long *dwFL*, long *dwRL*, BYTE *nType*)

※ eRTU RSM_MACRO_SET_SLMT (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long *dwFL*, long *dwRL*, BYTE *nType*)

說明:

設定正方向與反方向的軟體極限位置。

類別:

Modbus sub_function; RTC, MP and IT.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表 4 與表 5) 若使用表 5 的代碼設置，則參數值將被儲存到記憶體內
dwFL:	正方向的軟體極限位置 設定範圍: -2,000,000,000 ~ +2,000,000,000
dwRL:	反方向的軟體極限位置 設定範圍: -2,000,000,000 ~ +2,000,000,000
nType:	所依據的計數器 0=邏輯位置計數器(Logical Position, LP) 1=編碼器位置計數器(Encoder Position, EP)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_SLMT的Sub_function代碼為 0x0A 12。

RSM_MACRO_SET_SLMT的Sub_function代碼為 0x0C 12。

Modbus 範例:

```
RSM_SET_SLMT (hRsm, 1, AXIS_XYZU, 20000, -3000, 0);
//set the forward software limit to be 20000 and the reverse
// software limit to be -3000 for all axes on module 1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 07	0E
Register[]	**Value (hex)**	**Remarks**			
0	0A 12/0C 12	*Sub_function code*			
1	00 0F	*axis*			
2	00 00	*MSW of dwFL*			
3	4E 20	*LSW of dwFL (20000 ➔ 0x4E20)*			
4	FF FF	*MSW of dwRL*			
5	F4 48	*LSW of dwRL (-3000 ➔ 0xFFFFF448)*			
6	00 00	*nType*			

3.9.2 清除軟體極限設定

eRTU RSM_CLEAR_SLMT (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

※ eRTU RSM_MACRO_CLEAR_SLMT (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

說明:

清除軟體極限功能的設定值。

類別:

Modbus sub_function; RTC, MP and IT.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表 4 與表 5) 若使用表 5 的代碼設置，則參數值將被儲存到記憶體內

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_CLEAR_SLMT的Sub_function代碼為 0x0A 13。
RSM_MACRO_CLEAR_SLMT的Sub_function代碼為 0x0C 13。

Modbus 範例:

```
RSM_CLEAR_SLMT (hRsm, 1, AXIS_XYZU);
//clear the software limits for all axes on module 1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A 13/0C 13	*Sub_function code*			
1	00 0F	*Axis (F ➔ AXIS_XYZU)*			

3.10 編碼器回授訊號參數設定

eRTU RSM_SET_ENCODER (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nMode*, BYTE *nDivision*, BYTE *nZEdge*)

※ eRTU RSM_MACRO_SET_ENCODER (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nMode*, BYTE *nDivision*, BYTE *nZEdge*)

說明:

設定編碼器回授訊號的相關參數。

類別:

Modbus sub_function; RTC, MP and IT.

參數:

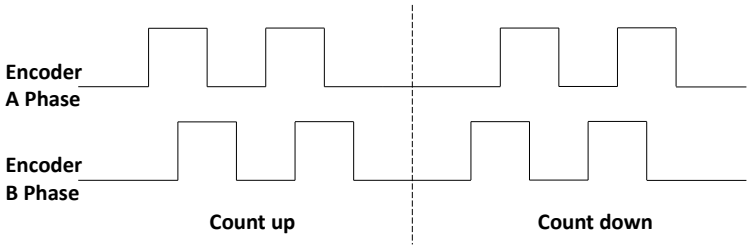
參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表 4 與表 5) 若使用表 5 的代碼設置，則參數值將被儲存到記憶體內
***nMode*:**	編碼器輸入的訊號類型: 0=A/B兩相脈波輸入(當A相訊號相位領先時計數器上數) 1= 上/下脈波輸入 (A相訊號輸入時計數器上數) 2=A/B兩相脈波輸入(當B相訊號相位領先時計數器上數) 3=/下脈波輸入(B相訊號輸入時計數器上數)
***nDivision*:**	設定編碼器A/B兩相脈波輸入之除頻倍率 0 = 1/1 1 = 1/2 2 = 1/4
***nZEdge*:**	Z 相訊號觸發邏輯: 0=低準位觸發；1=高準位觸發

A/B 兩相脈波輸入模式:

計數器同時依據兩個相位訊號輸入的上下緣變化計數，在此模式下可設定除頻倍率為 1/2 或 1/4。

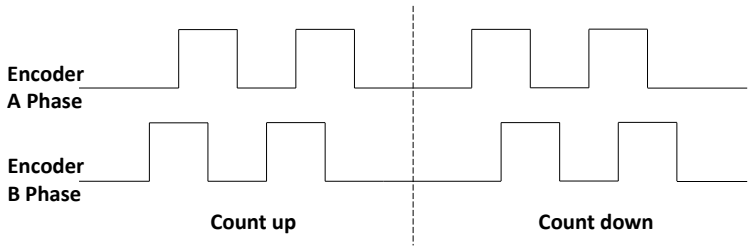
nMode = 0:

當 A 相脈波輸入的相位領先 B 相脈波輸入時計數器上數，當 B 相脈波輸入的相位領先 A 相脈波輸入時計數器下數。



nMode = 2:

當 B 相脈波輸入的相位領先 A 相脈波輸入時計數器上數，當 A 相脈波輸入的相位領先 B 相脈波輸入時計數器下數。

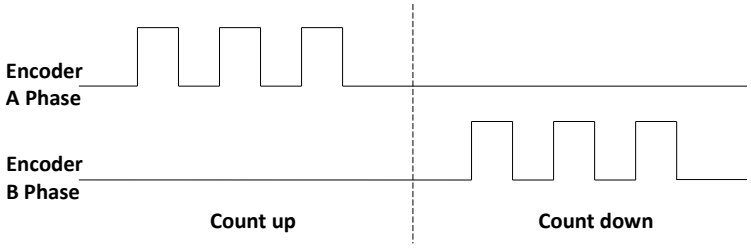


上/下 脈波輸入模式:

當輸入的脈波上緣觸發時計數器計數。

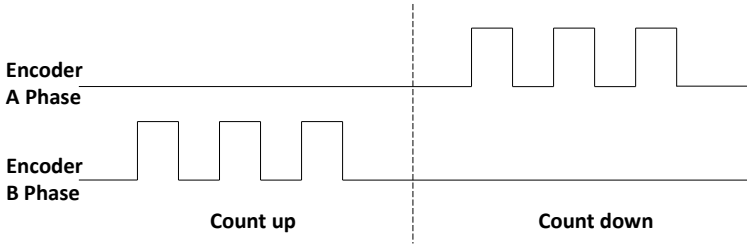
nMode = 1:

A 相脈波輸入時計數器上數，B 相脈波輸入時計數器下數。



nMode = 3:

B 相脈波輸入時計數器上數，A 相脈波輸入時計數器下數。



A/B 兩相脈波輸入模式的除頻設定:

nDivision = 0:

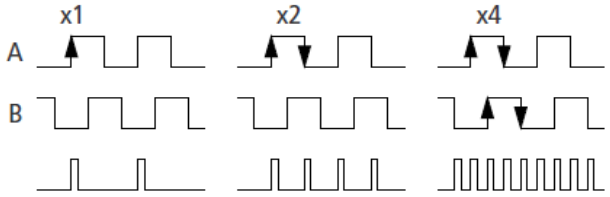
同時依據 A 相與 B 相訊號輸入的上下緣變化計數。

nDivision = 1:

依據 A 相訊號輸入的上下緣變化計數。

nDivision = 2:

依據 A 相訊號輸入的上緣變化計數。



回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_ENCODER的Sub_function代碼為 0x0A 14。

RSM_MACRO_SET_ENCODER的Sub_function代碼為 0x0C 14。

Modbus 範例:

RSM_SET_ENCODER (hRsm, 1, AXIS_XYZU, 0, 0, 0);

//set the encoder input type as A quad B; the division setting is 1/1; and the Z phase is low active.

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 05	0A
Register[]	**Value (hex)**	**Remarks**			
0	0A 14/0C 14	*Sub_function code*			
1	00 0F	*axis (F ➔ AXIS_XYZU)*			
2	00 00	*nMode (0 ➔ AB phase)*			
3	00 00	*nDivision (0 ➔ 1:1)*			
4	00 00	*nZEdge (0➔Z phase is low-active)*			

3.11 伺服驅動器啟動/關閉設定

3.11.1 伺服啟動 (Servo On)

eRTU RSM_SERVO_ON (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

※ eRTU RSM_MACRO_SERVO_ON (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

說明:

輸出一個DO訊號啟動馬達驅動器。

類別:

Modbus sub_function; RTC, MP and IT.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表 4 與表 5) 若使用表 5 的代碼設置，則參數值將被儲存到記憶體內

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SERVO_ON的Sub_function代碼為 0x0A 15。
RSM_MACRO_SERVO_ON的Sub_function代碼為 0x0C 15。

Modbus 範例:

```
RSM_SERVO_ON (hRsm, 1, AXIS_XYZU);
//enables all drivers on module 1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A 15/0C 15	*Sub_function code*			
1	00 0F	*axis (F ➔ AXIS_XYZU)*			

3.11.2 伺服關閉 (Servo Off)

eRTU RSM_SERVO_OFF (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

※ eRTU RSM_MACRO_SERVO_OFF (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

說明:

輸出一個DO訊號關閉馬達驅動器。

類別:

Modbus sub_function; RTC, MP and IT.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表 4 與表 5) 若使用表 5 的代碼設置，則參數值將被儲存到記憶體內

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SERVO_OFF的Sub_function代碼為 0x0A 16。
RSM_MACRO_SERVO_OFF的Sub_function代碼為 0x0C 16。

Modbus 範例:

```
RSM_SERVO_OFF (hRsm, 1, AXIS_XYZU);     //disables all drivers on module 1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A 16/0C 16	*Sub_function code*			
1	00 0F	*axis (F ➔ AXIS_XYZU)*			

3.12 伺服異常(ALARM)訊號設定

eRTU RSM_SET_ALARM (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nMode*, BYTE *nAEdge*)

※ eRTU RSM_MACRO_SET_ALARM (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nMode*, BYTE *nAEdge*)

說明:

設定伺服異常訊號輸入的相關參數。

類別:

Modbus sub_function; RTC, MP and IT.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表 4 與表 5) 若使用表 5 的代碼設置，則參數值將被儲存到記憶體內
***nMode*:**	模式: 0=關閉伺服異常訊號功能；1=開啟伺服異常訊號功能
***nAEdge*:**	伺服異常訊號觸發邏輯: 0=低準位觸發；1=高準位觸發

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_ALARM 的Sub_function代碼為 0x0A 17。
RSM_MACRO_SET_ALARM 的Sub_function代碼為 0x0C 17。

Modbus 範例:

```
RSM_SET_ALARM (hRsm, 1, AXIS_ZU, 1, 0);
//enable the ALARM for the Z and U axes on module 1 and set them
//as low-active.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 17/0C 17	*Sub_function code*			
1	00 0C	*axis (C ➔ AXIS_ZU)*			
2	00 01	*nMode (1 ➔ enable)*			
3	00 00	*nAEdge (0 ➔ low-active)*			

3.13 伺服到位(INPOS)訊號設定

eRTU RSM_SET_INPOS (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nMode*, BYTE *nIEdge*)

※ eRTU RSM_MACRO_SET_INPOS (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nMode*, BYTE *nIEdge*)

說明:

設定伺服到位訊號輸入的相關參數。

類別:

Modbus sub_function; RTC, MP and IT.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表 4 與表 5) 若使用表 5 的代碼設置，則參數值將被儲存到記憶體內
***nMode*:**	模式: 0=關閉伺服到位訊號功能；1=開啟伺服到位訊號功能
***nIEdge*:**	伺服到位訊號觸發邏輯: 0=低準位觸發；1=高準位觸發

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_INPOS 的Sub_function代碼為 0x0A 18。
RSM_MACRO_SET_INPOS 的Sub_function代碼為 0x0C 18。

Modbus 範例:

```
RSM_SET_INPOS (hRsm, 1, AXIS_X, 1, 0);
//enable the INPOS function of the X axis on module 1 and set it to be low-active.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 18/0C 18	*Sub_function code*			
1	00 01	*axis (1 ➔ AXIS_X)*			
2	00 01	*nMode (1 ➔ enable)*			
3	00 00	*nIEdge (0 ➔ low-active)*			

3.14 數位雜訊濾波器設定

eRTU RSM_SET_FILTER (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *FEn*, BYTE *FLn*)

※ eRTU RSM_MACRO_SET_FILTER (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *FEn*, BYTE *FLn*)

說明:

設定開啟數位雜訊濾波器的輸入訊號。

類別:

Modbus sub_function; RTC, MP and IT.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表 4 與表 5) 若使用表 5 的代碼設置，則參數值將被儲存到記憶體內
***FEn*:**	選擇開啟濾波器的輸入訊號，設定範圍: 0~31 0:關閉濾波器 其他: 位元 / 數值 / 開啟濾波的輸入訊號 bit0 / 1 / EMG, nLMTP, nLMTM, nIN0(NHOME), nIN1 (HOME) bit1 / 2 / nIN2 (Z-phase) bit2 / 4 / nINPOS, nALARM bit3 / 8 / nEXPP, nEXPM, EXPLSN bit4 / 16 / nIN3
***FLn*:**	設定濾波器的時間參數，範圍: 0~7 設定值 / 最大雜訊濾波寬度 / 輸入訊號延遲時間 0 / 1.75μSEC / 2μSEC 1 / 224μSEC / 256μSEC 2 / 448μSEC / 512μSEC 3 / 896μSEC / 1.024mSEC 4 / 1.792mSEC / 2.048mSEC 5 / 3.584mSEC / 4.096mSEC 6 / 7.168mSEC / 8.192mSEC 7 / 14.336mSEC / 16.384mSEC

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_FILTER的Sub_function代碼為 0x0A 19。
RSM_MACRO_SET_FILTER 的Sub_function代碼為 0x0C 19。

Modbus 範例:

```
RSM_SET_FILTER (hRsm, 1, AXIS_XYZU, 21, 3);
//set the filter time constants of X, Y, Z, and U axes as 1.024mSEC.
//These filters include EMG, nLMTP, nLMTM, nIN0, nIN1, nINPOS, nALARM,
//and nIN3.
//(21 = 1+4+16) 1: EMG + nLMP + nLMPM + nIN0 + nIN1;
//4: nINPOS + nALARM;
//16: nIN3.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 19/0C 19	*Sub_function code*			
1	00 0F	*axis (F ➔ AXIS_XYZU)*			
2	00 15	*FEn (21 = 0x15, enable several noise filters)*			
3	00 03	*FLn (using filter 3)*			

3.15 環狀位置計數器設定

3.15.1 開啟環狀位置計數器功能

eRTU RSM_VRING_ENABLE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *nVRing*)

※ eRTU RSM_MACRO_VRING_ENABLE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *nVRing*)

說明:

設定位置計數器為環狀計數器。

類別:

Modbus sub_function; RTC, MP and IT.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表 4 與表 5) 若使用表 5 的代碼設置，則參數值將被儲存到記憶體內
nVRing:	環狀計數器的最大值 範圍: 0 ~ +2,000,000,000

請注意:

(a) 將同時設定邏輯位置(LP)與編碼器位置(EP)。

(b) 無法與軟體極限同時使用。

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

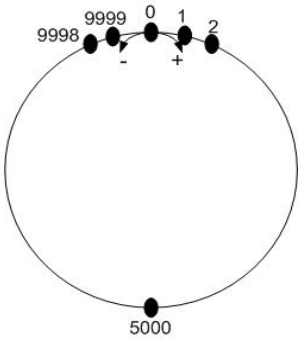
備註:

RSM_VRING_ENABLE 的Sub_function代碼為 0x0A 1A。

RSM_MACRO_VRING_ENABLE的Sub_function代碼為 0x0C 1A。

Modbus 範例:

```
RSM_VRING_ENABLE (hRsm, 1, AXIS_X, 9999);
//set the X axis of module 1 to be a ring counter. The encoder
//values will be 0 to 9999.
```



環狀計數器的最大值 = 9999

若將環狀計數器的計數範圍設定為 0 ~ 9999。
假設正方向脈波訊號輸入使得計數器到達9999，此時若再輸入一個正方向脈波訊號，則會使計數器數值回到0。
假設反方向脈波訊號輸入使得計數器到達0，此時若再輸入一個反方向脈波訊號，則會使計數器數值變為9999。

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 1A/0C 1A	*Sub_function code*			
1	00 01	*axis*			
2	00 00	*MSW of nVRing*			
3	27 0F	*LSW of nVRing (9999 = 0x270F)*			

3.15.2 關閉環狀位置計數器功能

eRTU RSM_VRING_DISABLE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

※ eRTU RSM_MACRO_VRING_DISABLE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

說明:

關閉環狀計數器功能。

類別:

Modbus sub_function; RTC, MP and IT.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表 4 與表 5) 若使用表 5 的代碼設置，則參數值將被儲存到記憶體內

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_VRING_DISABLE 的Sub_function代碼為 0x0A 1B。
RSM_MACRO_VRING_DISABLE 的Sub_function代碼為 0x0C 1B。

Modbus 範例:

```
RSM_VRING_DISABLE (hRsm, 1, AXIS_X);
//disable the ring counter function for the X axis on module 1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A 1B/0C 1B	*Sub_function code*			
1	00 01	*axis (1 ➔ AXIS_X)*			

3.16 三角形速度曲線預防設定

3.16.1 開啟三角形速度曲線預防功能

eRTU RSM_AVTRI_ENABLE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

※ eRTU RSM_MACRO_AVTRI_ENABLE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

說明:

開啟梯型速度曲線(T-Curve)運動的自動三角形速度修正功能。

類別:

Modbus sub_function; RTC, MP and IT.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表 4 與表 5) 若使用表 5 的代碼設置，則參數值將被儲存到記憶體內

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_AVTRI_ENABLE的Sub_function代碼為 0x0A 1C。
RSM_MACRO_AVTRI_ENABLE的Sub_function代碼為 0x0C 1C。

Modbus 範例:

RSM_AVTRI_ENABLE (hRsm, 1, AXIS_X);
//set the X axis of module 1 not to generate a triangle form in its speed profile.

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A 1C/0C 1C	*Sub_function code*			
1	00 01	*axis (1 ➔ AXIS_X)*			

3.16.2 關閉三角形速度曲線預防功能

eRTU RSM_AVTRI_DISABLE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

※ eRTU RSM_MACRO_AVTRI_DISABLE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

說明:

關閉梯型速度曲線(T-Curve)運動的自動三角形速度修正功能。

類別:

Modbus sub_function; RTC, MP and IT.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表 4 與表 5) 若使用表 5 的代碼設置，則參數值將被儲存到記憶體內

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_AVTRI_DISABLE 的Sub_function代碼為 0x0A 1D。
RSM_MACRO_AVTRI_DISABLE 的Sub_function代碼為 0x0C 1D。

Modbus 範例:

```
RSM_AVTRI_DISABLE (hRsm, 1, AXIS_X);
//enable the X axis of module 1 to generate a triangle form in its
//speed profile if the pulse number for output is too low.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A 1D/0C 1D	*Sub_function code*			
1	00 01	*axis (1 ➔ AXIS_X)*			

3.17 外部脈波輸入驅動

3.17.1 手輪脈波驅動模式設定

eRTU RSM_EXD_MP (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

說明:

設定手輪脈波驅動模式的對應控制脈波輸出數量。當nEXPM訊號為低準位且nEXPP輸入上緣觸發訊號，此時輸出正方向脈波控制訊號；當nEXPM訊號為低準位且nEXPP輸入下緣觸發訊號，此時輸出反方向脈波控制訊號。

類別:

Modbus sub_function; RTC.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
axis:	**軸號設定值 (請參照表4)**
***data*:**	**控制脈波的相對應輸出數量 範圍: 0 ~ +2,000,000,000**

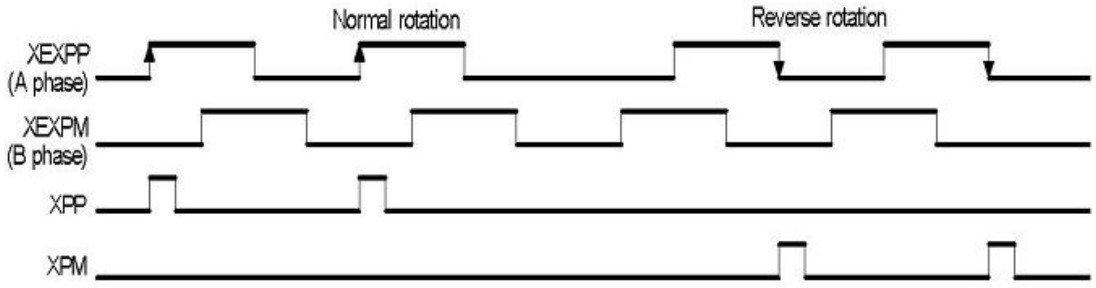
回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

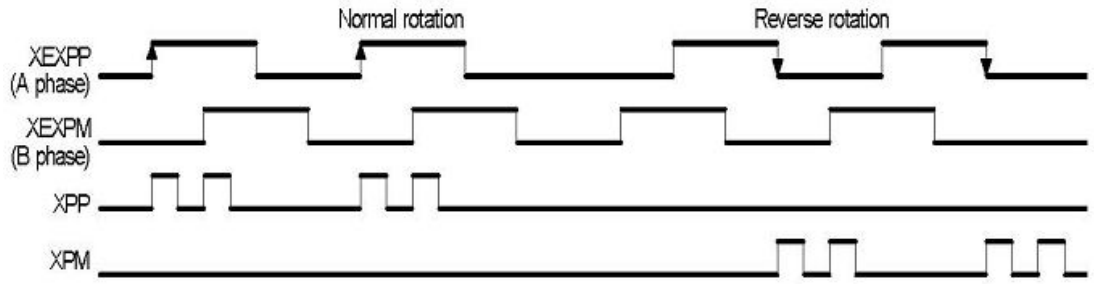
Modbus 範例:

RSM_EXD_MP (hRsm, 1, AXIS_X, 1);

//Each time the handwheel inputs a pulse to the X axis on module 1, the controller will output 1 pulse to the motor driver.



RSM_EXD_MP (hRsm, 1, AXIS_X, 2);
//Each time the handwheel inputs a pulse to the X axis
//on module 1, the controller will output 2 pulses to the motor driver.



所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 1E	*Sub_function code*			
1	00 01	*axis (1 ➔ AXIS_X)*			
2	00 00	*MSW of data*			
3	00 01	*LSW of data*			

3.17.2 定量驅動模式設定

eRTU RSM_EXD_FP (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

說明:

設定定量驅動模式的對應控制脈波輸出數量。當nEXPP輸入下緣觸發訊號時，輸出正方向脈波控制訊號；當nEXPM輸入下緣觸發訊號時，輸出反方向脈波控制訊號。請注意，在控制脈波傳送的過程中所輸入的觸發訊號皆視為無效。

類別:

Modbus sub_function; RTC.

參數:

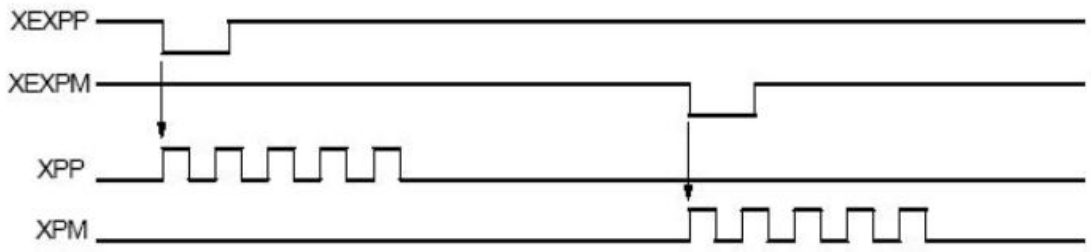
參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
axis:	**軸號設定值 (請參照表4)**
***data*:**	**控制脈波的相對應輸出數量** **範圍: 0 ~ +2,000,000,000**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
RSM_EXD_FP (hRsm, 1, AXIS_X, 5);
//Each time the controller detects a falling edge of an XEXP+
//signal, it will output 5 pulses to the X axis.
```



所傳送的 **Modbus** 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 1F	*Sub_function code*			
1	00 01	*axis (1 ➔ AXIS_X)*			
2	00 00	*MSW of data*			
3	00 05	*LSW of data*			

3.17.3 連續驅動模式設定

eRTU RSM_EXD_CP (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

說明:

設定連續驅動模式的對應控制脈波輸出速度。當nEXPP訊號由高準位變化為低準位時，會開始連續輸出正方向脈波控制訊號，而訊號恢復成高準位時將停止輸出；當nEXPM訊號由高準位變化為低準位時，會開始連續輸出反方向脈波控制訊號，而訊號恢復成高準位時將停止輸出。

類別:

Modbus sub_function; RTC.

參數:

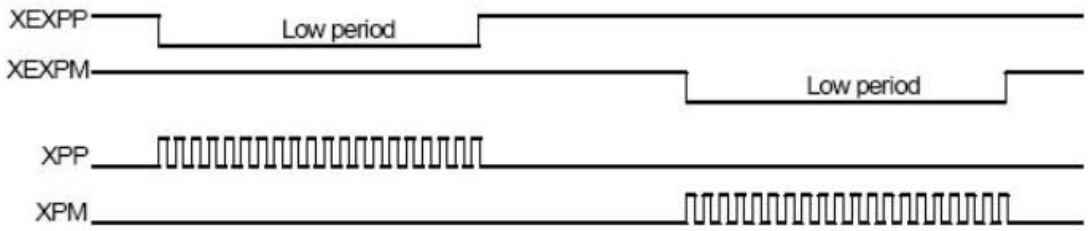
參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
axis:	**軸號設定值 (請參照表4)**
***data*:**	**控制脈波的相對應輸出速度(單位: PPS)** **範圍: 0 ~ +2,000,000,000**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
RSM_EXD_CP (hRsm, 1, AXIS_X, 20);
//Each time the controller detects a falling edge of an XEXP+ signal, it will continuously drive X axis at the speed of 20 PPS.
```



所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 20	*Sub_function code*			
1	00 01	*axis*			
2	00 00	*MSW of data*			
3	00 14	*LSW of data (20 = 0x14)*			

3.17.4 關閉外部脈波輸入驅動功能

eRTU RSM_EXD_DISABLE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

說明:

關閉外部脈波輸入驅動功能。

類別:

Modbus sub_function; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表4)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
RSM_EXD_DISABLE (hRsm, 1, AXIS_X);
//disable the external input driving control function of X axis on module 1
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A 21	*Sub_function code*			
1	00 01	*axis (1 ➔ AXIS_X)*			

3.18 讀/寫使用者自訂義變數 (VAR 與 bVAR)

3.18.1 讀取位元組(Byte)變數

eRTU RSM_READ_bVAR (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *bvarNo*, BYTE* *bVARValue*)

說明:

讀取使用者定義的位元組變數。讀寫此變數等同於讀寫巨集程序的自定義變數(請參照章節9.3)。

類別:

Modbus table; RTC.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
***bvarNo*:**	**變數編號 範圍: 128(bVAR0) ~ 255(bVAR127)**
***bVARValue*:**	**變數設定值**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

操作各個變數所使用的 Modbus Holding 暫存器位址表示如下:

巨集變數			
變數名稱	位址	類型	說明
bVAR0	128	R/W	bVAR0 的數值
bVAR1	129	R/W	bVAR1 的數值
…	…		
bVAR126	254	R/W	bVAR126 的數值
bVAR127	255	R/W	bVAR127 的數值

Modbus 範例:

假設 bVAR100 = 100(0x64)，起始位址為 0x00E4(128+100)。

```
BYTE bVARValue;
RSM_READ_bVAR (hRsm, 1, bVAR100, &bVARValue);
//read value of VAR100 in module 1
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	03	00 E4	00 01

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	Reg_0 Value (hex)
01	03	02	00 64

3.18.2 寫入位元組(Byte)變數

eRTU RSM_WRITE_bVAR (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *bvarNo*, BYTE *bVar*)

說明:

寫入使用者定義的位元組變數。讀寫此變數等同於讀寫巨集程序的自定義變數(請參照章節9.3)。

類別:

Modbus table, Modbus sub_function; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***bvarNo*:**	變數編號 範圍: 128(bVAR0) ~ 255(bVAR127)
***bVar*:**	變數設定值

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

操作各個變數所使用的 Modbus Holding 暫存器位址表示如下:

巨集變數			
變數名稱	位址	類型	說明
bVAR0	128	R/W	bVAR0 的數值
bVAR1	129	R/W	bVAR1 的數值
…	…		
bVAR126	254	R/W	bVAR126 的數值
bVAR127	255	R/W	bVAR127 的數值

Modbus 範例:

■ 方法 1: 使用 Sub_Function 功能碼

```
RSM_WRITE_bVAR (hRsm, 1, bVAR100, 100);
//write bVAR100=100 in module 1
```

The address of bVARn = 128 + n (or 0x80 + n).
(bVAR100 = 128 + 100 = 228 = 0xE4)

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0A 23	*Sub_function code*			
1	00 E4	*bvarNo = n + 0x80 = 0xE4*			
2	00 64	*bVar (100 = 0x64)*			

■ 方法 2: 使用 Holding 暫存器

所有的bVARn變數皆已定義在Holding 暫存器表內，bVARn操作的位址為128+n，例如bVAR100的位址為128+100=228 (0xE4)。

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	00 E4	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	00 64	*bVar (100 = 0x64)*			

3.18.3 讀取長整數(Long)變數

eRTU RSM_READ_VAR (HANDLE *hRsm*, BYTE *SlaveAddr*, long *varNo*, long* *VARValue*)

說明:

讀取使用者定義的長整數變數。讀寫此變數等同於讀寫巨集程序的自定義變數(請參照章節9.3)。

類別:

Modbus table; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***varNo*:**	變數編號 範圍: 0x7fff0000 (VAR0) ~ 0x7fff01ff (VAR511)
***VARValue*:**	變數設定值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

所有的 VARn 變數皆已定義在 Holding 暫存器表內，每個 VARn 佔用兩個暫存器，分別為"300 + 2*n"與"300 + 2*n + 1"這兩個位址的暫存器。請注意，讀取的暫存器長度應為 2 的倍數。

Start_Address (index) = 300 + 2*100 = 500 = 0x01F4

```
long VARValue;
RSM_READ_VAR (hRsm, 1, VAR100, &VARValue);
//read value of VAR100 in module 1
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	03	01 F4	00 02

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	Reg_0 (hex)	Reg_1 (hex)
01	03	04	00 00	27 10

取得 ldata 的程式範例(C 語言)

```
ldata = Register[0];
ldata = ((ldata <<16) & 0xffff0000) | (Register[1] & 0xffff);
```

3.18.4 寫入長整數(Long)變數

eRTU RSM_WRITE_VAR (HANDLE *hRsm*, BYTE *SlaveAddr*, long *varNo*, long *lVar*)

說明:

寫入使用者定義的長整數變數。讀寫此變數等同於讀寫巨集程序的自定義變數(請參照章節9.3)。

類別:

Modbus table, Modbus sub_function; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***varNo*:**	變數編號 範圍: 0x7fff0000 (VAR0) ~ 0x7fff01ff (VAR511)
***lVar*:**	變數設定值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

RSM_WRITE_VAR (hRsm, 1, VAR100, 10000); //write VAR100=10,000 in module 1

■ 方法 1: 使用 Sub_Function 功能碼

VARn 的位址= 0x7FFF0000 + n.

➔ VAR100 的位址 = 0x7FFF0000 + 0x64 且 10000 = 0x2710

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 05	0A
Register[]	**Value (hex)**	**Remarks**			
0	0A 25	*Sub_function code*			
1	7F FF	*MSW of varNo (MSW of 0x7FFF0064)*			
2	00 64	*LSW of varNo (LSW of 0x7FFF0064)*			
3	00 00	*MSW of lVar (MSW 0f 0x2710)*			
4	27 10	*LSW of lVar (LSW 0f 0x2710)*			

■ 方法 2: 使用 Holding 暫存器

所有的 VARn 變數皆已定義在 Holding 暫存器表內，每個 VARn 佔用兩個暫存器，分別為”300 + 2*n”與”300 + 2*n + 1”這兩個位址的暫存器。請注意，讀取的暫存器長度應為 2 的倍數。

VARn 的起始位址 = 300 + 2*n。

➔ Start_Address = 300 + 2*100 = 500 (0x01F4)。

假設 VAR100 = 10000 (0x2710)。

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	01 F4	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	00 00	*MSW of lVar (MSW 0f 0x2710)*			
1	27 10	*LSW of lVar (LSW 0f 0x2710)*			

3.19 讀/寫斷電保持資料

3.19.1 讀取機器資料 (Machine Data, MD)

eRTU RSM_READ_MD (HANDLE *hRsm*, BYTE *SlaveAddr*, long *mdNo*, long* *ldata*, float* *fdata*)

說明:

讀取機器資料。此資料儲存於非揮發性(non-volatile)記憶體中。

類別:

Modbus table; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
mdNo:	資料編號 長整數(long)資料編號範圍: 0(MD0) ~ 1023(MD1023) 浮點數 (float)資料編號範圍: 1024(MD1024) ~2047(MD2047)
ldata:	長整數資料設定值 (-2,147,483,648 ~ +2,147,483,647)
fdata:	浮點數資料設定值 (整數位加小數位共6位數)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

所有的 MDn 變數皆已定義在 Holding 暫存器表內，每個 MDn 佔用兩個暫存器，分別為"3000+ 2*n"與"3000 + 2*n + 1"這兩個位址的暫存器。請注意，讀取的暫存器長度應為 2 的倍數。

MD100 的 Start_Address = 3000 + 2*100 = 3200(0x0C80)
MD1500 的 Start_Address = 3000 + 2*1500 = 6000(0x1770)

```
long ldata;
float fdata;
RSM_READ_MD (hRsm, 1, MD100, &ldata, 0);
//read MD100 long data in module 1
RSM_READ_MD (hRsm, 1, MD1500, 0, &fdata);
//read MD1500 float data in module 1
```

ldata: 所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	03	0C 80	00 02

ldata: 回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	Reg_0 (hex)	Reg_1 (hex)
01	03	04	12 34	56 78

fdata: 所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	03	17 70	00 02

fdata: 回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	Reg_0 (hex)	Reg_1 (hex)
01	03	04	43 A0	D3 33

取得 ldata 的程式範例(C 語言)

```
ldata = Register[0];
ldata = ((ldata <<16) & 0xffff0000) | (Register[1] & 0xffff);
```

3.19.2 寫入機器資料 (Machine Data, MD)

eRTU RSM_WRITE_MD (HANDLE *hRsm*, BYTE *SlaveAddr*, long *mdNo*, long *ldata*, float *fdata*)

說明:

寫入機器資料。此資料儲存於非揮發性(non-volatile)記憶體中。

類別:

Modbus table, Modbus sub_function; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***mdNo*:**	資料編號 長整數(long)資料編號範圍: 0(MD0) ~ 1023(MD1023) 浮點數 (float)資料編號範圍: 1024(MD1024) ~2047(MD2047)
***ldata*:**	長整數資料設定值 (-2,147,483,648 ~ +2,147,483,647)
***fdata*:**	浮點數資料設定值 (整數位加小數位共6位數)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
RSM_WRITE_MD (hRsm, SlaveAddr, MD100, 0x12345678, 0);
//write MD100 long data in module 1
```

■ 方法 1: 使用 Sub_Function 功能碼

MDn 的位址= 0x00000000 + n。

➔ MD100 的位址 = 0x00000000 + 0x64。

寫入 MD100 = 0x12345678

寫入 MD1500 = 321.65

ldata: 所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 07	0E
Register[]	**Value (hex)**	**Remarks**			
0	0A 27	*Sub_function code*			
1	00 00	*MSW of mdNo (= 0)*			
2	00 64	*LSW of mdNo (n= 0x64)*			
3	12 34	*MSW of ldata*			
4	56 78	*LSW of ldata*			
5	00 00	*MSW of fdata*			
6	00 00	*LSW of fdata*			

fdata: 所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 07	0E
Register[]	**Value (hex)**	**Remarks**			
0	0A 27	*Sub_function code*			
1	00 00	*MSW of mdNo (= 0)*			
2	05 DC	*LSW of mdNo (n= 0x05DC)*			
3	00 00	*MSW of ldata*			
4	00 00	*LSW of ldata*			
5	43 A0	*MSW of fdata*			
6	D3 33	*LSW of fdata*			

■ **方法 2: 使用 Holding 暫存器**

所有的 MDn 變數皆已定義在 Holding 暫存器表內，每個 MDn 佔用兩個暫存器，分別為"3000+ 2*n"與"3000 + 2*n + 1"這兩個位址的暫存器。請注意，讀取的暫存器長度應為 2 的倍數。

➔ **Start_Address = 3000 + 2*100 = 3200(0x0C80)。**

ldata: 所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	0C 80	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	12 34	*MSW of MD100*			
1	56 78	*LSW of MD100*			

➔ **Start_Address = 3000 + 2*1500 = 6000(0x1770)。**

ldata: 所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	17 70	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	43 A0	*MSW of MD1500*			
1	D3 33	*LSW of MD1500*			

4 運動控制狀態

4.1 位置計數器的設定與讀取

4.1.1 設定邏輯位置(Logical Position, LP)計數器

eRTU RSM_SET_LP (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long *wdata*)

※Δ eRTU RSM_MACRO_SET_LP (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long *wdata*)

說明:

設定邏輯位置(命令脈波)計數器的計數值。

類別:

Modbus table, Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
***axis*:**	**軸號設定值 (請參照表4)**
***wdata*:**	**邏輯位置計數器的計數值** **範圍: -2,000,000,000 ~ +2,000,000,000**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_LP 的Sub_function代碼為 0x0A 28。
RSM_MACRO_SET_LP的Sub_function代碼為 0x0C 28。

Modbus 範例:

```
RSM_SET_LP (hRsm, 1, AXIS_XYZU, 10000);
//Set the LP to 10000 for X, Y, Z, and U axes in module 1
// will clear all LP counters on module 1
```

■ 方法 1: 使用 Sub_Function 功能碼

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 28/0C 28	*Sub_function code*			
1	00 0F	*axis (F ➔ AXIS_XYZU)*			
2	00 00	*MSW of wdata*			
3	27 10	*LSW of wdata*			

■ 方法 2: 使用 Holding 暫存器

LP_X 的起始位址為 90 (0x5A)。因為 LP 的資料需要佔用 2 個暫存器的空間，所以 Modbus 命令的暫存器起始位址須從 MSW 開始。否則將會回傳一個異常錯誤訊息，所有的 32 位元資料操作皆須遵循此使用原則。

方法2允許使用者在一個Modbus 需求命令下對4軸設定不同的值(方法1不支援)。

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	00 5A	00 08	10
Register[]	**Value (hex)**	**Remarks**			
0	00 00	*LP_X_MSW (address = 0x5A)*			
1	27 10	*LP_X_LSW*			
2	00 00	*LP_Y_MSW (address = 0x5C)*			
3	27 10	*LP_Y_LSW*			
4	00 00	*LP_Z_MSW (address = 0x5E)*			
5	27 10	*LP_Z_LSW*			
6	00 00	*LP_U_MSW (address = 0x60)*			
7	27 10	*LP_U_LSW*			

4.1.2 讀取邏輯位置(Logical Position, LP)計數器

eRTU RSM_GET_LP(HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long* *LPValue*)

※Δ eRTU RSM_MACRO_GET_LP (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

eRTU RSM_GET_LP_4_AXIS(HANDLE *hRsm*, BYTE *SlaveAddr*, long* *LpValueX*, long* *LpValueY*, long* *LpValueZ*, long* *LpValueU*)

說明:

取得邏輯位置(命令脈波)計數器的計數值。

類別:

Modbus table, Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***LPValue*:**	邏輯位置計數器的計數值 範圍: -2,000,000,000 ~ +2,000,000,000
***LpValueX*:**	X軸邏輯位置計數器的計數值 範圍: -2,000,000,000 ~ +2,000,000,000
***LpValueY*:**	Y軸邏輯位置計數器的計數值 範圍: -2,000,000,000 ~ +2,000,000,000
***LpValueZ*:**	Z軸邏輯位置計數器的計數值 範圍: -2,000,000,000 ~ +2,000,000,000
***LpValueU*:**	U軸邏輯位置計數器的計數值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

■ 方法 1: 使用 Holding 暫存器

因為 LP 的資料需要佔用 2 個暫存器的空間，所以 Modbus 命令的暫存器起始位址須從 MSW 開始，否則將會回傳一個異常錯誤訊息。所有的 32 位元資料操作皆須遵循此使用原則。

此方法允許使用者在一個Modbus 需求命令下取得多軸數值資料，請參照Modbus Holding 暫存器定義表。

```
long X_LP;
RSM_GET_LP (hRsm, 1, AXIS_X, &X_LP);
//Reads the LP value of the X axis on module 1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr.(hex)	Word Count (hex)
01	03	00 5A	00 02

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	Reg_0 (hex)	Reg_1 (hex)
01	03	04	00 00	27 10

取得 X_LP 的程式範例(C 語言)

```
X_LP = Register[0];
X_LP = ((X_LP <<16) & 0xffff0000) | (Register[1] & 0xffff);
```

■ 方法 2: 用於巨集程序(MP)指令

使用此方法時目前的 LP 值不會馬上回傳，只有在 MP 被呼叫執行時才會回傳 LP 值。使用者可以使用 FC = 16 去寫這個指令到 MP。在 MP 中於 RSM_MACRO_GET_LP 後面可以使用 RSM_MACRO_SET_RVAR()這個指令去儲存回傳的 LP 值到一個指定的變數。此變數可以被其他函數使用，請參照其他 MP 函數的用法說明。

```
RSM_MACRO_GET_LP (hRsm, 1, AXIS_X);
//Reads the LP value of the X axis on module 1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0C 29	*Sub_function code*			
1	00 01	*axis (1 ➔ AXIS_X)*			

4.2 編碼器位置的設定與讀取

4.2.1 設定編碼器位置(Encoder Position, EP)計數器

eRTU RSM_SET_EP (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long *wdata*)

※Δ eRTU RSM_MACRO_SET_EP (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long *wdata*)

說明:

設定編碼器位置(回授脈波)計數器的計數值。

類別:

Modbus table, Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
***axis*:**	**軸號設定值 (請參照表4)**
***wdata*:**	**編碼器位置計數器的計數值 範圍: -2,000,000,000 ~ +2,000,000,000**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_EP 的Sub_function代碼為 0x0A 2A。
RSM_MACRO_SET_EP 的Sub_function代碼為 0x0C 2A。

Modbus 範例:

```
RSM_SET_EP (hRsm, 1, AXIS_XYZU, 10000);
//Set the EP as 10000 for X, Y, Z, and U axes of module 1.
```

■ 方法 1: 使用 Sub_Function 功能碼

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 2A/0C 2A	*Sub_function code*			
1	00 0F	*axis (F ➔ AXIS_XYZU)*			
2	00 00	*MSW of wdata*			
3	27 10	*LSW of wdata*			

■ 方法 2: 使用 Holding 暫存器

因為 EP 的資料需要佔用 2 個暫存器的空間，所以 Modbus 命令的暫存器起始位址須從 MSW 開始，否則將會回傳一個異常錯誤訊息。所有的 32 位元資料操作皆須遵循此使用原則。

此方法允許使用者在一個 Modbus 需求命令下取得多軸數值資料，請參照 Modbus Holding 暫存器定義表。

EP_X 的暫存器起始位置為 98(0x62)。

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	00 62	00 08	10
Register[]	**Value (hex)**	**Remarks**			
0	00 00	*LP_X_MSW (address = 0x5A)*			
1	27 10	*LP_X_LSW*			
2	00 00	*LP_Y_MSW (address = 0x5C)*			
3	27 10	*LP_Y_LSW*			
4	00 00	*LP_Z_MSW (address = 0x5E)*			
5	27 10	*LP_Z_LSW*			
6	00 00	*LP_U_MSW (address = 0x60)*			
7	27 10	*LP_U_LSW*			

4.2.2 讀取編碼器位置(Encoder Position, EP)計數器

eRTU RSM_GET_EP (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis* , long* *EPValue*)

※Δ eRTU RSM_MACRO_GET_EP (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

eRTU RSM_GET_EP_4_AXIS (HANDLE *hRsm*, BYTE *SlaveAddr*, long* *EpValueX*, long* *EpValueY*, long* *EpValueZ*, long* *EpValueU*)

說明:

取得編碼器位置(回授脈波)計數器的計數值。

類別:

Modbus table, Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***EPValue*:**	編碼器位置計數器的計數值 範圍: -2,000,000,000 ~ +2,000,000,000
EpValueX:	X軸編碼器位置計數器的計數值 範圍: -2,000,000,000 ~ +2,000,000,000
EpValueY:	Y軸編碼器位置計數器的計數值 範圍: -2,000,000,000 ~ +2,000,000,000
EpValueZ:	Z軸編碼器位置計數器的計數值 範圍: -2,000,000,000 ~ +2,000,000,000
EpValueU:	U軸編碼器位置計數器的計數值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

假設目前 EP_X 的數值為 10000(0x2710)，EP_X 的暫存器起始位置為 98(0x62)。因為 EP 的資料需要佔用 2 個暫存器的空間，所以 Modbus 命令的暫存器起始位址須從 MSW 開始，否則將會回傳一個異常錯誤訊息。所有的 32 位元資料操作皆須遵循此使用原則。

■ 方法 1: 使用 Holding 暫存器

```
long X_EP;
RSM_GET_EP (hRsm, 1, AXIS_X, &X_EP);
//reads the encoder position value (EP) of the X axis on module 1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	03	00 62	00 02

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	Reg_0 (hex)	Reg_1 (hex)
01	03	04	00 00	27 10

取得 X_EP 的程式範例(C 語言)

```
X_EP = Register[0];
X_EP = ((X_EP <<16) & 0xffff0000) | (Register[1] & 0xffff);
```

■ 方法 2: 用於巨集程序(MP)指令

使用此方法時目前的 EP 值不會馬上回傳，只有在 MP 被呼叫執行時才會回傳 EP 值。使用者可以使用 FC = 16 去寫這個指令到 MP。在 MP 中於 RSM_MACRO_GET_EP 後面可以使用 RSM_MACRO_SET_RVAR()這個指令去儲存回傳的 EP 值到一個指定的變數。此變數可以被其他函數使用，請參照其他 MP 函數的用法說明。

```
RSM_MACRO_GET_EP (hRsm, 1, AXIS_X);
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0C 2B	*Sub_function code*			
1	00 01	*axis (1 ➔ AXIS_X)*			

4.3 絕對位置設定與讀取

4.3.1 設定絕對位置計數器

eRTU RSM_ABS_SET_POSITION (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long *wdata*)

※Δ eRTU RSM_MACRO_ABS_SET_POSITION (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long *wdata*)

說明:

設定絕對邏輯位置計數器的計數值，此計數值用來表示絕對位置移動功能的的當前位置。

類別:

Modbus table, Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***wdata*:**	絕對位置計數器的計數值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_ABS_SET_POSITION的Sub_function代碼為 0x0AF4。
RSM_MACRO_ABS_SET_POSITION的Sub_function代碼為0x0CF4。

Modbus 範例:

```
RSM_ABS_SET_POSITION (hRsm, 1, AXIS_X, 40000);
//Set the current absolute position of x-axis to 40000
```

■ 方法 1: 使用 Sub_Function 功能碼

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A F4/0C F4	*Sub_function code*			
1	00 0F	*axis (F ➔ AXIS_XYZU)*			
2	00 00	*MSW of wdata*			
3	9C 40	*LSW of wdata*			

■ 方法 2: 使用 Holding 暫存器

起始位址為 82 (0x52)。因為絕對位置計數器的資料需要佔用 2 個暫存器的空間，所以 Modbus 命令的暫存器起始位址須從 MSW 開始。否則將會回傳一個異常錯誤訊息，所有的 32 位元資料操作皆須遵循此使用原則。

方法2允許使用者在一個Modbus 需求命令下對4軸設定不同的值(方法1不支援)。

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	00 52	00 08	10
Register[]	**Value (hex)**	**Remarks**			
0	00 00	*Absolute LP_X_MSW (address = 0x52)*			
1	27 10	*Absolute LP_X_LSW*			
2	00 00	*Absolute LP_Y_MSW (address = 0x54)*			
3	27 10	*Absolute LP_Y_LSW*			
4	00 00	*Absolute LP_Z_MSW (address = 0x56)*			
5	27 10	*Absolute LP_Z_LSW*			
6	00 00	*Absolute LP_U_MSW (address = 0x58)*			
7	27 10	*Absolute LP_U_LSW*			

4.3.2 讀取絕對位置計數器

eRTU RSM_ABS_GET_POSITION (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long* *LPAbsValue*)

※Δ eRTU RSM_MACRO_ABS_GET_POSITION (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

eRTU RSM_ABS_GET_POSITION_4_AXIS (HANDLE *hRsm*, BYTE *SlaveAddr*, long* *LpAbsValueX*, long* *LpAbsValueY*, long* *LpAbsValueZ*, long* *LpAbsValueU*)

說明:

讀取絕對邏輯位置計數器的計數值，此計數值用來表示絕對位置移動功能的的當前位置。

類別:

Modbus table, Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表4)
LPAbsValue:	絕對位置計數器的計數值 範圍: -2,000,000,000 ~ +2,000,000,000
LpAbsValueX:	X軸絕對位置計數器的計數值 範圍: -2,000,000,000 ~ +2,000,000,000
LpAbsValueY:	Y軸絕對位置計數器的計數值 範圍: -2,000,000,000 ~ +2,000,000,000
LpAbsValueZ:	Z軸絕對位置計數器的計數值 範圍: -2,000,000,000 ~ +2,000,000,000
LpAbsValueU:	U軸絕對位置計數器的計數值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

■ 方法 1: 使用 Holding 暫存器

因為絕對位置計數器的資料需要佔用 2 個暫存器的空間，所以 Modbus 命令的暫存器起始位址須從 MSW 開始。否則將會回傳一個異常錯誤訊息，所有的 32 位元資料操作皆須遵循此使用原則。

此方法允許使用者在一個 Modbus 需求命令下取得多軸數值資料，請參照 Modbus Holding 暫存器定義表。

```
long X_AbsLP;
RSM_ABS_GET_POSITION (hRsm, 1, AXIS_X, &X_AbsLP);
//Reads the current absolute position LP value of the X axis.
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	03	00 52	00 02

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	Reg_0 (hex)	Reg_1 (hex)
01	03	04	00 00	27 10

取得 X_AbsLP 的程式範例(C 語言)

```
X_AbsLP = Register[0];
X_AbsLP = ((X_AbsLP <<16) & 0xffff0000) | (Register[1] & 0xffff);
```

■ 方法 2: 用於巨集程序(MP)指令

使用此方法時目前的絕對位置計數值不會馬上回傳，只有在 MP 被呼叫執行時才會回傳數值。使用者可以使用 FC = 16 去寫這個指令到 MP。在 MP 中於 RSM_MACRO_ABS_GET_POSITION 後面可以使用 RSM_MACRO_SET_RVAR()這個指令去儲存回傳的絕對位置計數值到一個指定的變數。此變數可以被其他函數使用，請參照其他 MP 函數的用法說明。

```
RSM_MACRO_ABS_GET_POSITION (hRsm, 1, AXIS_X);
//Reads the LP value of the X axis on module 1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0C F5	*Sub_function code*			
1	00 01	*axis (1 ➔ AXIS_X)*			

4.4 取得當前速度

eRTU RSM_GET_CV (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long* *CVValue*)

eRTU RSM_GET_CV_4_AXIS (HANDLE *hRsm*, BYTE *SlaveAddr*, long* *CvValueX*, long* *CvValueY*, long* *CvValueZ*, long* *CvValueU*)

說明:

讀取目前速度數值。

類別:

Modbus table; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表4)
CVValue:	當前速度值
CvValueX:	X軸當前速度值
CvValueY:	Y軸當前速度值
CvValueZ:	Z軸當前速度值
CvValueU:	U軸當前速度值

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

假設目前的 CV 數值是 10000(0x2710)，暫存器的起始位置是 106(0x6A)。

```
long CVValue;
RSM_GET_CV (hRsm, 1, AXIS_X, &CVValue);
//reads the current velocity of the X axis on module 1
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	03	00 6A	00 02

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	Reg_0 (hex)	Reg_1 (hex)
01	03	04	00 00	27 10

取得 CV_X 的程式範例(C 語言)

```
CV_X = Register[0];
CV_X = ((CV_X <<16) & 0xffff0000) | (Register[1] & 0xffff);
```

4.5 取得當前加速度

eRTU RSM_GET_CA (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis* , long* *CAValue*)

eRTU RSM_GET_CA_4_AXIS (HANDLE *hRsm*, BYTE *SlaveAddr*, long* *CaValueX*, long* *CaValueY*, long* *CaValueZ*, long* *CaValueU*)

說明:

讀取目前加速度數值。

類別:

Modbus table; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
CAValue:	當前加速度值，單位: PPS/Sec
CaValueX:	X軸當前加速度值，單位: PPS/Sec
CaValueY:	Y軸當前加速度值，單位: PPS/Sec
CaValueZ:	Z軸當前加速度值，單位: PPS/Sec
CaValueU:	U軸當前加速度值，單位: PPS/Sec

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

假設目前的 CA 數值是 10000(0x2710)，暫存器的起始位置是 114(0x72)。

```
long CAValue;
RSM_GET_CA (hRsm, 1, AXIS_X, &CAValue);
//reads the current acceleration value of the X axis on module 1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	03	00 72	00 02

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	Reg_0 (hex)	Reg_1 (hex)
01	03	04	00 00	27 10

取得 CA_X 的程式範例(C 語言)

```
CA_X = Register[0];
CA_X = ((CA_X <<16) & 0xffff0000) | (Register[1] & 0xffff);
```

4.6 取得運動相關輸入觸發狀態

4.6.1 取得單獨輸入觸發狀態

※Δ eRTU RSM_MACRO_GET_DI (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE nType)

說明:

取得運動晶片的輸入訊號觸發狀態(指定單一輸入)。

類別:

Modbus sub_function; MP, ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***nType*:**	0 → DRIVING (是否正在輸出控制脈波) 1 → LIMIT+ (正方向極限是否觸發) 2 → LIMIT- (反方向極限是否觸發) 3 → EMERGENCY (緊急停止訊號是否觸發) 4 → ALARM (驅動器異常訊號是否觸發) 5 → IN1 (輸入訊號接點IN1是否觸發) 6 → IN0 (輸入訊號接點IN0是否觸發) 7 → IN3 (輸入訊號接點IN3是否觸發) 8 → INPOS (伺服到位訊號是否觸發) 9 →IN2 (輸入訊號接點IN2是否觸發)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

■ 用於巨集程序(MP)指令

使用此方法時目前的 DI 值不會馬上回傳，只有在 MP 被呼叫執行時才會回傳 DI 值。使用者可以使用 FC = 16 去寫這個指令到 MP。在 MP 中於 RSM_MACRO_GET_DI 後面可以使用 RSM_MACRO_SET_RVAR()這個指令去儲存回傳的 DI 值到一個指定的變數。此變數可以被其他函數使用，請參照其他 MP 函數的用法說明。

```
RSM_MACRO_GET_DI (hRsm, 1, AXIS_X, 1);
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0C 2E	*Sub_function code*			
1	00 01	*axis*			
2	00 01	*The specified DI (LIMIT+)*			

4.6.2 取得所有輸入觸發狀態

eRTU RSM_GET_DI_ALL (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis* , WORD* DIValue)

※Δ eRTU RSM_MACRO_GET_DI_ALL (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

eRTU RSM_GET_DI_ALL_4_AXIS (HANDLE *hRsm*, BYTE *SlaveAddr*, WORD* *DiValueX*, WORD* *DiValueY*, WORD* *DiValueZ*, WORD* *DiValueU*)

說明:

取得運動晶片的輸入訊號觸發狀態(所有輸入)。

類別:

Modbus table, Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
DIValue:	運動晶片的輸入訊號觸發狀態 *Bit* 0 → DRIVING (是否正在輸出控制脈波) *Bit* 1 → LIMIT+ (正方向極限是否觸發) *Bit* 2 → LIMIT- (反方向極限是否觸發) *Bit* 3 → EMERGENCY (緊急停止訊號是否觸發) *Bit* 4 → ALARM (驅動器異常訊號是否觸發) *Bit* 5 → IN1 (輸入訊號接點IN1是否觸發) *Bit* 6 → IN0 (輸入訊號接點IN0是否觸發) *Bit* 7 → IN3 (輸入訊號接點IN3是否觸發) *Bit* 8 → INPOS (伺服到位訊號是否觸發) *Bit* 9 →IN2 (輸入訊號接點IN2是否觸發)
***DiValueX*:**	運動晶片的X軸輸入訊號觸發狀態
***DiValueY*:**	運動晶片的Y軸輸入訊號觸發狀態
***DiValueZ*:**	運動晶片的Z軸輸入訊號觸發狀態
***DiValueU*:**	運動晶片的U軸輸入訊號觸發狀態

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

■ 方法 1: 使用 Input 暫存器

```
WORD DIValue;
RSM_GET_DI_ALL (hRsm, 1, AXIS_X, &DIValue);
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	04	00 10	00 01

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	DI Status of AXIS_X. (Reg_0)
01	04	02	00 00

■ 方法 2: 用於巨集程序(MP)指令

使用此方法時目前的 DI 值不會馬上回傳，只有在 MP 被呼叫執行時才會回傳 DI 值。使用者可以使用 FC = 16 去寫這個指令到 MP。在 MP 中於 RSM_MACRO_GET_DI_ALL 後面可以使用 RSM_MACRO_SET_RVAR()這個指令去儲存回傳的 DI 值到一個指定的變數。此變數可以被其他函數使用，請參照其他 MP 函數的用法說明。

```
RSM_MACRO_GET_DI_ALL (hRsm, 1, AXIS_X);
//RSM_MACRO_SET_RVAR(hRsm, 1, VAR0);  //assign this DI value to VAR0
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0C 31	*Sub_function code*			
1	00 01	*axis (1: AXIS_X)*			

4.7 取得運動相關輸入訊號狀態

4.7.1 取得單一輸入訊號狀態

※Δ eRTU RSM_MACRO_GET_DI_SIGNAL (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nType*)

說明:

取得運動控制相關的實際外部輸入訊號狀態(指定單一輸入)。

類別:

Modbus sub_function; MP, ISR.

參數:

參數名稱	說明
hRsm:	連接埠handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表4)
nType:	訊號狀態 0 → NHOME/IN0 (近原點訊號) 1 → HOME/IN1 (原點訊號) 2 → Index/IN2 (Z相訊號) 3 → IN3 (IN3訊號) 4 → EXP+ (外部驅動控制正向輸入訊號) 5 → EXP- (外部驅動控制反向輸入訊號) 6 → INP (伺服到位訊號) 7 → ALARM (伺服異常訊號)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

■ 用於巨集程序(MP)指令

使用此方法時目前的 DI 值不會馬上回傳，只有在 MP 被呼叫執行時才會回傳 DI 值。使用者可以使用 FC = 16 去寫這個指令到 MP。在 MP 中於 RSM_MACRO_GET_DI_SIGNAL 後面可以使用 RSM_MACRO_SET_RVAR()這個指令去儲存回傳的 DI 值到一個指定的變數。此變數可以被其他函數使用，請參照其他 MP 函數的用法說明。

```
RSM_MACRO_GET_DI_SIGNAL (hRsm, 1, AXIS_X, 1);
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0C 40	*Sub_function code*			
1	00 01	*axis*			
2	00 01	*The specified DI signal (***HOME***)*			

4.7.2 取得所有輸入訊號狀態

eRTU RSM_GET_DI_SIGNAL_ALL (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis* , WORD* *DISignalValue*)

※Δ eRTU RSM_MACRO_GET_DI_SIGNAL _ALL (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

eRTU RSM_GET_DI_SIGNAL_ALL_4_AXIS (HANDLE *hRsm*, BYTE *SlaveAddr*, WORD* *DiSignalValueX*, WORD* *DiSignalValueY*, WORD* *DiSignalValueZ*, WORD* *DiSignalValueU*)

說明:

取得運動控制相關的實際外部輸入訊號狀態(所有輸入)。

類別:

Modbus table, Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表4)
DISignalValue:	訊號狀態 0 → NHOME/IN0 (近原點訊號) 1 → HOME/IN1 (原點訊號) 2 → Index/IN2 (Z相訊號) 3 → IN3 (IN3訊號) 4 → EXP+ (外部驅動控制正向輸入訊號) 5 → EXP- (外部驅動控制反向輸入訊號) 6 → INP (伺服到位訊號) 7 → ALARM (伺服異常訊號)
DiSignalValueX:	X軸的當前訊號狀態
DiSignalValueY:	Y軸的當前訊號狀態
DiSignalValueZ:	Z軸的當前訊號狀態
DiSignalValueU:	U軸的當前訊號狀態

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

■ **方法 1: 使用 Input 暫存器**

WORD DIValue;

RSM_GET_DI_SIGNAL_ALL (hRsm, 1, AXIS_X, &DIValue);

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	04	00 58	00 01

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	DI Status of AXIS_X. (Reg_0)
01	04	02	00 00

■ **方法 2: 用於巨集程序(MP)指令**

使用此方法時目前的 DI 值不會馬上回傳，只有在 MP 被呼叫執行時才會回傳 DI 值。使用者可以使用 FC = 16 去寫這個指令到 MP。在 MP 中於 RSM_MACRO_GET_DI_SIGNAL _ALL 後面可以使用 RSM_MACRO_SET_RVAR()這個指令去儲存回傳的 DI 值到一個指定的變數。此變數可以被其他函數使用，請參照其他 MP 函數的用法說明。

RSM_MACRO_GET_DI_SIGNAL_ALL (hRsm, 1, AXIS_X);

//RSM_MACRO_SET_RVAR(hRsm, 1, VAR0); //assign this DI value to VAR0

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0C 41	*Sub_function code*			
1	00 01	*axis (1: AXIS_X)*			

4.8 取得原點復歸狀態

eRTU RSM_GET_HOME_SEARCH_STATE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis* , WORD* *HomeStatus*)

※Δ eRTU RSM_MACRO_ GET_HOME_SEARCH_STATE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

eRTU RSM_GET_HOME_SEARCH_STATE _4_AXIS (HANDLE *hRsm*, BYTE *SlaveAddr*, WORD* *HomeStatusX*, WORD* *HomeStatusY*, WORD* *HomeStatusZ*, WORD* *HomeStatusU*)

說明:

取得自動原點復歸功能的狀態。

類別:

Modbus table, Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表4)
HomeStatus:	原點復歸當前狀態 0 → 等待自動原點復歸開始指令 3 → 執行步驟 1 : 朝指定搜尋方向移動並等待NHOME訊號有效(activation) 8 →執行步驟 2: 朝指定搜尋相反方向移動並等待HOME訊號有效(非正規動作) 12→執行步驟 2: 朝指定搜尋相反方向移動並等待HOME訊號無效(deactivation) (非正規動作) 15→執行步驟 2: 朝指定搜尋方向移動並等待HOME訊號有效 20→執行步驟 3: 朝指定搜尋方向移動並等待Z相訊號有效 25→執行步驟 4: 朝指定搜尋方向執行Offset位移中 非正規動作說明: (a) 當步驟2開始執行前HOME訊號已有效，此時會先朝指定搜尋相反方向移動，等待HOME訊號無效後在執行步驟2 (b) 當步驟2開始執行前搜尋方向的極限訊號已有效，此時會先朝指定搜尋相反方向移動，等待HOME訊號有效後，再執行(a)的動作

	(c) 當執行步驟2的過程中搜尋方向的極限訊號有效，此時會先停止移動，並執行(b)的動作
HomeStatusX:	X軸原點復歸當前狀態
HomeStatusY:	Y軸原點復歸當前狀態
HomeStatusZ:	Z軸原點復歸當前狀態
HomeStatusU:	U軸原點復歸當前狀態

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

■ 方法 1: 使用 Input 暫存器

```
WORD HomeStatus;
RSM_GET_HOME_SEARCH_STATE (hRsm, 1, AXIS_X, &HomeStatus);
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	04	00 5C	00 01

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	Home State of AXIS_X. (Reg_0)
01	04	02	00 00

■ 方法 2: 用於巨集程序(MP)指令

使用此方法時目前的原點復歸狀態不會馬上回傳，只有在 MP 被呼叫執行時才會回傳狀態值。使用者可以使用 FC = 16 去寫這個指令到 MP。在 MP 中於 RSM_MACRO_GET_HOME_SEARCH_STATE 後面可以使用 RSM_MACRO_SET_RVAR()這個指令去儲存回傳的狀態值到一個指定的變數。此變數可以被其他函數使用，請參照其他 MP 函數的用法說明。

```
RSM_MACRO_GET_HOME_STATE(hRsm, 1, AXIS_X);
//RSM_MACRO_SET_RVAR(hRsm, 1, VAR0); assign this home state value to VAR0
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0C 42	*Sub_function code*			
1	00 01	*axis (1: AXIS_X)*			

4.9 錯誤狀態的取得與清除

4.9.1 確認錯誤是否發生

eRTU RSM_GET_ERROR_STATE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE* *ErrState*)

※Δ eRTU RSM_MACRO_GET_ERROR (HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

確認錯誤是否發生。

類別:

Modbus table, Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
ErrState:	0=無錯誤發生 1=錯誤發生，請使用函式 RSM_GET_ERROR_CODE 取得更多訊息

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

■ 方法 1: 使用 Input 暫存器

```
BYTE ErrState;
RSM_GET_ERROR_STATE (hRsm, 1, &ErrState);
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	04	00 46	00 01

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	Error status (Reg_0)
01	04	02	00 00

■ 方法 2: 用於巨集程序(MP)指令

使用此方法時目前的錯誤狀態不會馬上回傳，只有在 MP 被呼叫執行時才會回傳狀態值。使用者可以使用 FC = 16 去寫這個指令到 MP。在 MP 中於 RSM_MACRO_

GET_ERROR 後面可以使用 RSM_MACRO_SET_RVAR()這個指令去儲存回傳的狀態值到一個指定的變數。此變數可以被其他函數使用，請參照其他 MP 函數的用法說明。

RSM_MACRO_GET_ERROR (hRsm, 1);

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	0C 2F	*Sub_function code*			

4.9.2 取得錯誤代碼

eRTU RSM_GET_ERROR_CODE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, WORD* *ErrCode*)

※Δ eRTU RSM_MACRO_GET_ERROR_CODE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

eRTU RSM_GET_ERROR_CODE_4_AXIS (HANDLE *hRsm*, BYTE *SlaveAddr*, WORD* *ErrCodeX*, WORD* *ErrCodeY*, WORD* *ErrCodeZ*, WORD* *ErrCodeU*)

說明:

取得錯誤代碼。

類別:

Modbus table, Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
ErrCode:	錯誤碼 0=無錯誤；其他請參照下表

位元	數值	停止原因	說明
Bit0	1	SOFT LIMIT+	正方向軟體極限觸發
Bit1	2	SOFT LIMIT-	反方向軟體極限觸發
Bit2	4	LIMIT+	正方向硬體極限訊號觸發
Bit3	8	LIMIT-	反方向硬體極限訊號觸發
Bit4	16	ALARM	伺服異常訊號觸發
Bit5	32	EMERGENCY	緊急停止訊號觸發
Bit6	64	保留	保留
Bit7	128	HOME	Z相與原點復歸訊號觸發
Bit8	256	SOFT STOP	執行軟體停止指令 (RSM_STOP_SLOWLY, RSM_STOP_SUDDENLY, RSM_VSTOP_SLOWLY, RSM_VSTOP_SUDDENLY)
Bit9	512	SOFT EMG	執行軟體緊急停止指令 (RSM_EMERGENCY_STOP)

	假設錯誤碼=48(0x30)，表示 ALARM 和EMGERENCY 訊號同時觸發
ErrCodeX:	X軸錯誤碼
ErrCodeY:	Y軸錯誤碼
ErrCodeZ:	Z軸錯誤碼
ErrCodeU:	U軸錯誤碼

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

■ 方法 1: 使用 Input 暫存器

```
WORD ErrCode;
RSM_GET_ERROR_CODE (hRsm, 1, AXIS_X, &ErrCode);
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	04	00 14	00 01

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	ErrCode (Reg_0)
01	04	02	00 00

■ 方法 2: 用於巨集程序(MP)指令

使用此方法時目前的錯誤碼不會馬上回傳，只有在 MP 被呼叫執行時才會回傳數值。使用者可以使用 FC = 16 去寫這個指令到 MP。在 MP 中於 RSM_MACRO_GET_ERROR_CODE 後面可以使用 RSM_MACRO_SET_RVAR()這個指令去儲存回傳的錯誤碼到一個指定的變數。此變數可以被其他函數使用，請參照其他 MP 函數的用法說明。

```
RSM_MACRO_GET_ERROR_CODE (hRsm, 1, AXIS_X);
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0C 30	*Sub_function code*			
1	00 01	*axis (AXIS_X)*			

4.10 取得命令緩衝區狀態

eRTU RSM_GET_FREE_BUFFER(HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE* *FreeBufNum*)

說明:

取得目前命令緩衝區的剩餘空間。緩衝區最大可容許30個命令，且會依照先進先出(First in first out, FIFO)的原則依序執行緩衝區內的指令。

類別:

Modbus table；RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
FreeBufNum:	目前命令緩衝區的剩餘空間(可填入的指令數量)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
BYTE FreeBufNum;
RSM_GET_FREE_BUFFER(hRsm, 1, & FreeBufNum);
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	04	00 47	00 01

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	*Block_Num*(hex)
01	04	02	00 1E

回傳值為 0x1E 表示緩衝區可再填入 30 個指令。

4.11 取得停止狀態

eRTU RSM_GET_STOP_STATUS(HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE* *StopState*)

eRTU RSM_GET_STOP_STATUS_4_AXIS(HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE* *StopStateX*, BYTE* *StopStateY*, BYTE* *StopStateZ*, BYTE* *StopStateU*)

說明:

確認當前是否為停止狀態。

類別:

Modbus table；RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表 4)
***StopState*:**	停止狀態 0=移動中 1=停止中
***StopStateX*:**	X軸停止狀態
***StopStateY*:**	Y軸停止狀態
***StopStateZ*:**	Z軸停止狀態
***StopStateU*:**	U軸停止狀態

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

軸號	暫存器位置	備註
X 軸	58 (0x3A)	1 ➔ AXIS_X 停止中 0 ➔ AXIS_X 移動中
Y 軸	59 (0x3B)	1 ➔ AXIS_Y 停止中 0 ➔ AXIS_Y 移動中
Z 軸	60 (0x3C)	1 ➔ AXIS_Z 停止中 0 ➔ AXIS_Z 移動中
U 軸	61 (0x3D)	1 ➔ AXIS_U 停止中 0 ➔ AXIS_U 移動中

```
BYTE StopState;
RSM_GET_STOP_STATUS (hRsm, 1, AXIS_X, & StopState);
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	04	00 3A	00 01

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	AXIS Stop_Status (hex)
01	04	02	00 01

4.12 取得軟體緊急停止狀態

eRTU RSM_GET_EMERGENCY_STATE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE * *EmgState*)

說明:

取得當前軟體緊急停止的狀態。使用函式"RSM_EMERGENCY_STOP()"觸發軟體緊急停止狀態，使用函式"RSM_CLEAR_EMERGENCY_STOP()"解除軟體緊急停止狀態。

類別:

Modbus table: RTC.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
***axis*:**	**軸號設定值 (請參照表4)**
EmgState:	**0=軟體緊急停止未觸發** **1=軟體緊急停止已觸發**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

使用 Input 暫存器，位址為 25(0x19)。

BYTE EmgState;

RSM_GET_EMERGENCY_STATE (hRsm, 1, & EmgState);

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	04	00 19	00 01

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	EmgState (hex)
01	04	02	00 01

4.13 取得正在驅動(移動)的軸號

eRTU RSM_GET_DRIVING_AXIS(HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE* *DrvAxis*)

說明:

取得當前正在驅動(移動)的軸號。

類別:

Modbus table: RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
DrvAxis:	0=無移動中的軸 >0目前正在移動的軸號(請參照表4)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
BYTE DrvAxis;
RSM_GET_DRIVING_AXIS (hRsm, 1, & DrvAxis);
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	04	00 60	00 01

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	Driving Axis (hex)
01	04	02	00 01

4.14 取得補間命令傳送允許旗標

eRTU RSM_GET_INTERPOL_RDY_FLAG(HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE* *InterpolationRdyFlag*)

說明:

確認運動控制晶片是否已準備完成，可接受下一個補間指令。

類別:

Modbus table: RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***InterpolationRdyFlag*:**	0=尚未準備完成接收下一個補間命令 1=已準備完成接收下一個補間命令

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
BYTE InterpolationRdyFlag;
RSM_GET_INTERPOL_RDY_FLAG (hRsm, 1, & InterpolationRdyFlag);
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	04	00 57	00 01

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	InterpolationRdyFlag (hex)
01	04	02	00 01

4.15 取得中斷事件觸發的軸號

eRTU RSM_GET_TRIG_INTFACTOR (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *nINT*, BYTE **pAxis*)

說明:

取得已觸發運動晶片中斷事件的軸號。

類別:

Modbus table; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
nINT:	請參照下列中斷事件表說明
pAxis:	請參照表 4 範例: 0x0F: 四軸皆被觸發 0x02: 只有 Y 軸被觸發 0x0B: X、Y 和 U 軸被觸發

中斷事件說明與所使用的 Modbus function code 4 暫存器位址

nINT	暫存器位址(FC4)	符號	中斷條件
0	130	PULSE	控制脈波上升時
1	131	P>=C-	邏輯/編碼器位置計數器的數值大於或等於比較器 COMP-的設定值
2	132	P<C-	邏輯/編碼器位置計數器的數值小於比較器 COMP-的設定值
3	133	P<C+	邏輯/編碼器位置計數器的數值小於比較器 COMP+的設定值
4	134	P>=C+	邏輯/編碼器位置計數器的數值小於比較器 COMP+的設定值
5	135	C-END	在驅動過程中，等速度區段結束的時候
6	136	C-STA	在驅動過程中，等速度區段開始的時候
7	137	D-END	驅動結束(移動完成)
8	138	HMEND	自動原點復歸動作中斷
9	139	SYNC	同步運動被觸發

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
BYTE bAxis;
RSM_GET_TRIG_INTFACTOR (hRsm, 1, 0, & bAxis);
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01 (Card No.)	04	00 82 (130) (Pulse output)	00 01

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	Registers(hex)
01	04	02	00 09 (Axis XY triggered)

4.16 取得 RS-M8194H 狀態

4.16.1 命令下載與執行的程序

運動控制器由兩個模組構成
– RS-M8194H
– i-8094H

安裝於插槽 0 的 RS-M8194H 模組主要負責處理與 Host 端之間的 Modbus 通訊，模組將確認並過濾由串列連接埠接收到的每個命令，依照當前的運作狀態判斷是否要將指令轉交 i-8094H 執行，若為非法指令則回傳 Modbus 異常碼至 Host 端。請使用函式”RSM_GET_DEVICE_STATE()”取得當前運作狀態與異常碼。

模組的三種運作狀態分別為:
– 準備就緒狀態: 可執行 RTC 指令類別。
– MP 巨集下載狀態: 可執行 MP 指令類別。
– ISR 巨集下載狀態: 可執行 ISR 指令類別。

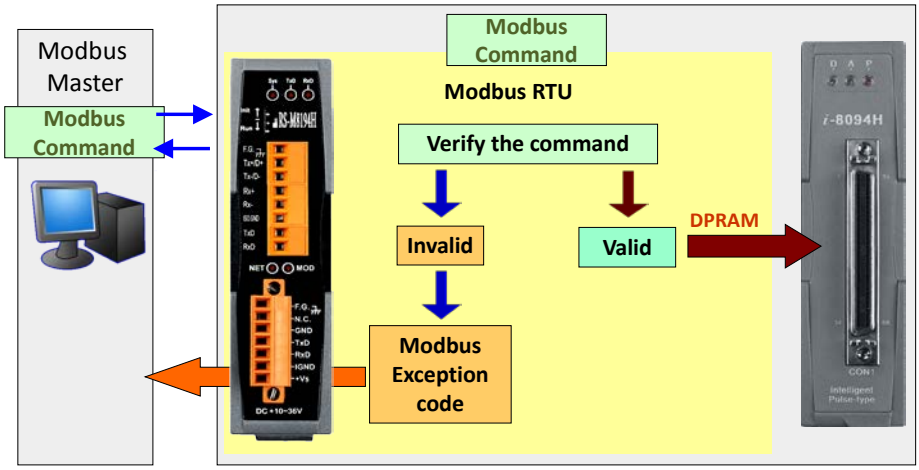


圖 1 安裝於插槽 0 的 RS-M8194H 模組

安裝於插槽 1 的 i-8094H 模組主要負責實際執行各項控制命令。模組上的 DPRAM 被用來與 RS-M8194H 模組交握資料與傳送指令，若接收到運動控制指令則會驅動模組上的控制晶片執行命令，若接收到巨集指令，則會將此命令儲存到模組上的非揮發性(non-volatile)記憶體內。

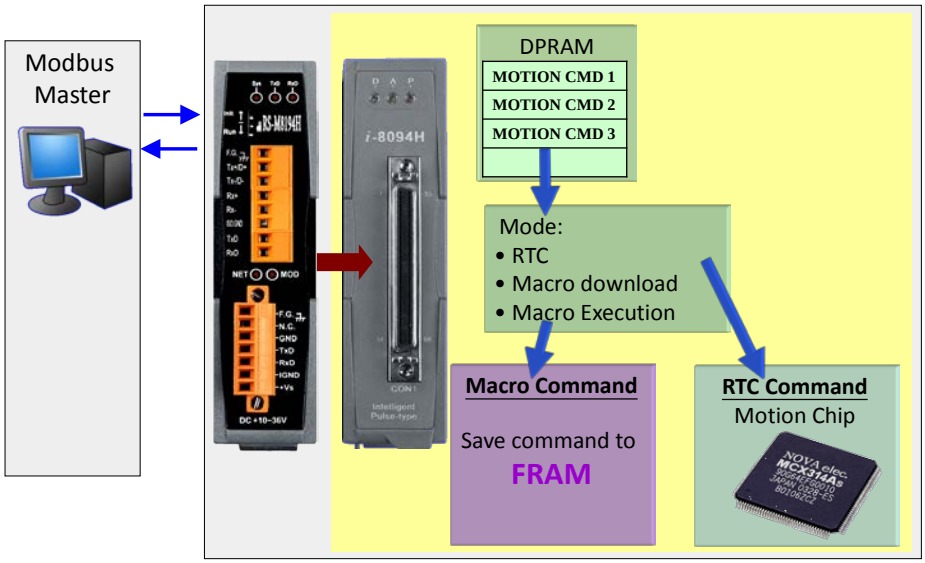


圖 2 安裝於插槽 1 的 i-8094H 模組

4.16.2 取得 RS-M8194H 狀態

```
eRTU RSM_GET_DEVICE_STATE (HANDLE hRsm, BYTE SlaveAddr,
        RsmState *pState)

typedef struct
{
     WORD Rsm8194hState;
     WORD I8094hState;
     WORD IllegalFuncErr;
}RsmState;
```

說明:

取得RS-M8194H與i-8094H當前狀態以及Modbus最後一次發生的異常碼。

類別:

Modbus table; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
pState:	*Rsm8194hState:* RS-M8194H 狀態

RS-M8194H 狀態	說明
0x00 (RSM_READY)	只允許傳送 RTC 命令
0x01 (RSM_MP_DOWNLOAD)	只接受 MP 巨集命令
0x02 (RSM_ISR_DOWNLOAD)	只有 ISR 巨集命令有效

I8094hState : i-8094H 狀態

i-8094H 狀態	說明
0x00 (i8094H_READY)	i-8094H 已準備完成，可接收 RTC 命令
0x01 (i8094H_INITIALIZATION)	執行初始化表(IT 命令)
0x03 (i8094H_MACRO_DOWNLOAD)	i-8094H 正在巨集下載模式中
0x05 (i8094H_MP_EXECUTION)	正在執行 MP 巨集程序
0x07 (i8094H_ISR_EXECUTION)	正在執行 ISR 巨集程序
0x10 (i8094H_ERR_FIRMWARE)	i-8094H 韌體發生錯誤，須重置電源
0xFF (i8094H_FIRMWARE_BOOTING)	i-8094H 韌體尚未完成啟動過程

***IllegalFuncErr* : Mudbus 異常回應的原因**

異常碼	說明
0x00 (ERR_NO_ERROR)	無異常發生
0x01 (ERR_FC_NOT_SUPPORTED)	不支援的功能碼 (Function Code)
0x03 (ERR_EXCEED_MP_MEMORY)	巨集程序已超出可使用的指令數量
0x04 (ERR_MP_INSIDE_ISR)	在 ISR 巨集程序內使用 MP 指令
0x05 (ERR_ISR_INSIDE_MP)	在 MP 巨集程序內使用 ISR 指令
0x06 (ERR_RTC_INSIDE_MP_OR_ISR)	在巨集程序內使用 RTC 指令
0x07 (ERR_MP_OR_ISR_AS_RTC)	在巨集程序外使用 MP 或 ISR 指令

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
RsmState rsmState;
RSM_GET_DEVICE_STATE (hRsm, 1, & rsmState);
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01 (Card No.)	04	00 64 (100)	00 03

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	Registers(hex)
01	04	0A	See next table

Register[]	Value (hex)	Remarks
0	00 01	*Rsm8194hState:* RSM_MP_DOWNLOAD
1	00 03	*I8094hState :* i8094H_MACRO_DOWNLOAD
2	00 06	*IllegalFuncErr:* ERR_RTC_INSIDE_MP_OR_ISR

5 FRnet 功能

5.1 取得 FRnet DIO 訊號狀態

5.1.1 讀取 FRnet 群組資料(MP 指令)

※Δ eRTU RSM_MACRO_FRNET_IN (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *wGroup*)

說明:

讀取FRnet群組(Group)的數位輸入資料(MP指令)。一個群組包含16位元資料(16點IO)，且最多可同時操作八個DI群組，也就是說FRnet介面最多可使用128點DI。

類別:

Modbus sub_function; MP, ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
wGroup:	群組號碼 DO 群組: 0~7 (可讀取 DO 狀態) DI 群組: 8~15

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

暫存器包含 16 位元資料，每個位元用來表示群組內的通道狀態。

■ 用於巨集程序(MP)指令

使用此方法時目前的 DI 狀態不會馬上回傳，只有在 MP 被呼叫執行時才會回傳狀態。使用者可以使用 FC = 16 去寫這個指令到 MP。在 MP 中於 RSM_MACRO_FRNET_IN 後面可以使用 RSM_MACRO_SET_RVAR()這個指令去儲存回傳的 DI 狀態到一個指定的變數。此變數可以被其他函數使用，請參照其他 MP 函數的用法說明。

RSM_MACRO_FRNET_IN (hRsm, 1, 8);

所傳送的 **Modbus** 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0C 32	*Sub_function code*			
1	00 08	*Group number (8~15)*			

5.1.2 讀取 FRnet 通道資料 (MP 指令)

※Δ eRTU RSM_MACRO_FRNET_READ (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *bGroup*, DWORD *dwChannel*)

說明:

讀取FRnet群組(Group)的單一通道的狀態(MP指令)。

類別:

Modbus sub_function; MP, ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
wGroup:	群組號碼 DO 群組: 0~7 (可讀取 DO 狀態) DI 群組: 8~15
dwChannel:	群組內的通道號碼 範圍: 0~15

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

RSM_MACRO_FRNET_READ (hRsm, 1, 8, 1);

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 05	0A
Register[]	**Value (hex)**	**Remarks**			
0	0C 34	*Sub_function code*			
1	00 00	*MSW of 0x0008*			
2	00 08	*LSW of 0x0008*			
3	00 00	*MSW of 0x0001*			
4	00 01	*LSW of 0x0001*			

5.1.3 取得 FRnet 通道資料 (RTC 指令)

eRTU RSM_FRNET_READ_SINGLE_DIO (HANDLE *hRsm*, BYTE *SlaveAddr*, WORD *wGroup*, BYTE *bChannelNo*, BYTE **pbChannelStatus*)

說明:

讀取FRnet群組(Group)的單一通道狀態(RTC指令)。

類別:

Modbus table; RTC.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
***wGroup*:**	**群組號碼** **DO 群組: 0~7 (可讀取 DO 狀態)** **DI 群組: 8~15**
***bChannelNo*:**	**群組內的通道號碼** **範圍: 0~15**
***pbChannelStatus*:**	**通道狀態**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

5.1.4 讀取 FRnet 群組資料 (RTC 指令)

eRTU RSM_FRNET_READ_GROUP_DIO (HANDLE *hRsm*, BYTE *SlaveAddr*, WORD *wGroup*, WORD* *pwGroupStatus*)

說明:

讀取FRnet群組(Group)的資料(RTC指令)。

類別:

Modbus table; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***wGroup*:**	群組號碼 DO 群組: 0~7 (可讀取 DO 狀態) DI 群組: 8~15
pwGroupStatus	群組資料

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
WORD pwGroupStatus;
RSM_MACRO_FRNET_READ(hRsm, 1, 8, &pwGroupStatus);
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	04	00 08	00 01

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	*pwGroupStatus* (hex)
01	04	02	0001

5.1.5 讀取 FRnet 多組群組資料 (RTC 指令)

eRTU RSM_FRNET_READ_MULTI_GROUP_DIO (HANDLE *hRsm*, BYTE *SlaveAddr*, WORD *wStartGroup*, BYTE *bGroupQty*, WORD* *pwInputBuffer*, BYTE *bBufferSize*)

說明:

同時讀取FRnet多個群組的資料(RTC指令)。

類別:

Modbus table; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
wStartGroup:	起始群組號碼 DO 群組: 0~7 (可讀取 DO 狀態) DI 群組: 8~15
***bGroupQty*:**	群組數量
***pwInputBuffer*:**	讀取到的群組資料
***bBufferSize*:**	暫存器的大小(*bBufferSize**2 必須 >= *bGroupQty*)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
WORD pwInputBuffer[2];
RSM_FRNET_READ_MULTI_GROUP_DIO(hRsm, 1, 8, 2, & pwInputBuffer, 4);
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	04	00 08	00 02

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	*pwInputBuffer[0]* (hex)	*pwInputBuffer[1]* (hex)
01	04	04	00 01	00 03

5.2 設定 FRnet DO

5.2.1 設定 FRnet DO 群組資料 (MP 指令)

※Δ eRTU RSM_MACRO_FRNET_OUT (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *bGroup*, DWORD *data*)

說明:

寫入資料到FRnet DO群組(MP指令)。一個群組包含16位元資料(16點IO)，且最多可同時操作八個DO群組，也就是說FRnet介面最多可使用128點DO。

類別:

Modbus table, Modbus sub_function; MP, ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
bGroup:	群組號碼 DO 群組: 0~7 (DI 群組不可寫入)
data:	設定值 範圍: 0x00000000 ~ 0x0000ffff， 也可以使用 VARn 變數設定

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

■ 方法 1: 使用常數設定 FRnet DO 數值

暫存器包含 16 位元資料，每個位元用來表示群組內的通道狀態。請確保設定值的 MSW 部分為 0。

```
RSM_MACRO_FRNET_OUT (hRsm, 1, 0, 0x0000FFFF);
RSM_MACRO_FRNET_OUT (hRsm, 1, 1, 0x0000AAAA);
RSM_MACRO_FRNET_OUT (hRsm, 1, 2, 0x00001100);
RSM_MACRO_FRNET_OUT (hRsm, 1, 3, 0x00000011);
```

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	00 00	00 04	08
Register[]	Value (hex)	Remarks			
0	FF FF	*DO setting (for address = 0)*			
1	AA AA	*DO setting (for address = 1)*			
2	11 00	*DO setting (for address = 2)*			
3	00 11	*DO setting (for address = 3)*			

■ 方法 2: 使用自定義變數(VARn)設定 FRnet DO 數值

為配合 32 位元變數 VARn 的使用，因此 DO 的設定參數採用 DWORD 資料型態。使用 VARn 變數時，MSW 將不為 0。

```
RSM_MACRO_FRNET_OUT (hRsm, 1, 0, 0x0000FFFF);
// on module 1, set group number RA=0, output data is 0x0000ffff
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	Value (hex)	Remarks			
0	0C 33	*Sub_function code*			
1	00 00	*Group number (0~7)*			
2	00 00	*MSW of DO*			
3	FF FF	*LSW of DO*			

```
RSM_MACRO_FRNET_OUT (hRsm, 1, bVAR0, VAR1);
// on module 1, set group number RA= bVAR0, output data is VAR1
```

註: bVAR0 的位址為 0x80 (= 0x80+n)；VAR1 的位址為 0x012E (= 300 + 2*n)。

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	Value (hex)	Remarks			
0	0C 33	*Sub_function code*			
1	00 80	*bVAR0*			
2	00 00	*MSW of VAR1*			
3	01 2E	*LSW of VAR1*			

5.2.2 設定 FRnet DO 通道資料 (MP 指令)

※Δ eRTU RSM_MACRO_FRNET_WRITE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *bDoGroup*, DWORD *dwChannel*, DWORD *dwOutput*)

說明:

設定FRnet DO群組的單一通道狀態(MP指令)。

類別:

Modbus sub_function; MP, ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***bDoGroup*:**	群組號碼 DO 群組: 0~7 (DI 群組不可寫入)
***dwChannel*:**	群組內的通道號碼 範圍: 0~15
***dwOutput*:**	設定值

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

RSM_MACRO_FRNET_WRITE(hRsm, 1, 1, 1, 1);

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 07	0E
Register[]	**Value (hex)**	**Remarks**			
0	0C 35	*Sub_function code*			
1	00 00	*MSW of 0x0001*			
2	00 01	*LSW of 0x0001*			
3	00 00	*MSW of 0x0001*			
4	00 01	*LSW of 0x0001*			
5	00 00	*MSW of 0x0001*			
6	00 01	*LSW of 0x0001*			

5.2.3 設定 FRnet DO 通道資料 (RTC 指令)

eRTU RSM_FRNET_WRITE_SINGLE_DO (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *bGroup*, DWORD *dwChannelNo*, BYTE *bOutput*)

說明:

設定FRnet DO群組的單一通道狀態(RTC指令)。

類別:

Modbus table；RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***bGroup*:**	群組號碼 DO 群組: 0~7 (DI 群組不可寫入)
dwChannelNo:	群組內的通道號碼 範圍: 0~15
***bOutput*:**	設定值

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節 2.2) 。

Modbus 範例:

RSM_FRNET_WRITE_SINGLE_DO (hRsm, 1, bGroup, dwChannelNo, 1);

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Value (hex)
01	05	(bGroup * 16 + dwChannelNo)	00 00(off); FF 00(on)

5.2.4 設定 FRnet DO 群組資料 (RTC 指令)

eRTU RSM_FRNET_WRITE_GROUP_DO (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *bGroup*, DWORD *dwdata*)

說明:

寫入資料到FRnet DO群組(RTC指令)。

類別:

Modbus table；RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***bGroup*:**	群組號碼 DO 群組: 0~7 (DI 群組不可寫入)
***dwdata*:**	設定值 範圍: 0x00000000 ~ 0x0000ffff， 也可以使用 VARn 變數設定

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節 2.2) 。

Modbus 範例:

```
RSM_FRNET_WRITE_GROUP_DO(hRsm, 1, 0, 0x0000FFFF);
// on module 1, set group number RA=0, output data is 0x0000ffff
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	00 00	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	00 00	*MSW of 0xFFFF*			
1	FF FF	*LSW of 0xFFFF*			

5.2.5 設定 FRnet DO 多組群組資料 (RTC 指令)

eRTU RSM_FRNET_WRITE_MULTI_GROUP_DO (HANDLE *hRsm*, BYTE *SlaveAddr*, WORD *wStartGroup*, BYTE *bGroupQty*, WORD* *pwOutBuffer*, BYTE *bBufferSize*)

說明:

寫入資料到多組FRnet DO 群組(RTC指令)。

類別:

Modbus table；RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
wStartGroup:	起始群組號碼 DO 群組: 0~7 (DI 群組不可寫入)
bGroupQty:	群組數量
pwOutBuffer:	寫入的群組資料
bBufferSize:	暫存器的大小(*bBufferSize**2 必須 >= *bGroupQty*)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節 2.2) 。

Modbus 範例:

```
WORD pwOutBuffer[4];
pwOutBuffer[0]=0xFFFF;
pwOutBuffer[1]=0xAAAA;
pwOutBuffer[2]=0x1100;
pwOutBuffer[3]=0x0011;
RSM_FRNET_WRITE_MULTI_GROUP_DO(hRsm, 1, 0, 4, & pwOutBuffer, 8);
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	00 00	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	FF FF	*DO setting (for address = 0)*			
1	AA AA	*DO setting (for address = 1)*			
2	11 00	*DO setting (for address = 2)*			
3	00 11	*DO setting (for address = 3)*			

5.3 FRnet DI 觸發等待(MP 指令)

※Δ eRTU RSM_MACRO_FRNET_WAIT (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *bDiGroup*, DWORD *dwChannel*, DWORD *dwDiInput*, DWORD *dwTimeout*)

說明:

等待FRnetDI訊號觸發(MP指令)。

類別:

Modbus sub_function; MP, ISR.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
***bDiGroup*:**	**群組號碼** **DI 群組: 8~15**
***dwChannel*:**	**群組內的通道號碼** **範圍: 0~15**
dwDiInput:	**0:OFF** **1:ON**
***dwTimeout*:**	**等待逾時設定值** **0=無逾時(持續等待直到訊號觸發)** **N=等待 N 毫秒 (Millisecond)**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

RSM_MACRO_FRNET_WAIT(hRsm, 1, 8, 1, 1, 1000);

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 08	10
Register[]	**Value (hex)**	**Remarks**			
0	0C 36	*Sub_function code*			
1	00 08	*bDiGroup*			
2	00 00	*MSW of 0x0001*			
3	00 01	*LSW of 0x0001*			
4	00 00	*MSW of 0x0001*			
5	00 01	*LSW of 0x0001*			
6	00 00	*MSW of 0x03E8*			
7	03 E8	*LSW of 0x03E8*			

5.4 FRnet DI 觸發事件設定

eRTU RSM_SET_FRNET_TRIGGER_EVENT(HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *bEnableDisable*)

說明:

開啟FRnet DI 觸發事件設定功能。當此功能開啟時FRnet DI 觸發表(使用EzMove設定)將被啟動 (請參照章節10)。

類別:

RTC.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
***bEnableDisable*:**	**0: 關閉功能** **1: 啟用功能**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

RSM_SET_FRNET_TRIGGER_EVENT (hRsm, 1, 1);

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	00 09	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	00 01	0-disable; 1-enabled			

5.5 取得 FRnet DI 觸發事件設定值

eRTU RSM_GET_FRNET_TRIGGER_EVENT_SETTING(HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE **pbEnableDisable*)

說明:

取得FRnet DI 觸發事件設定功能開啟狀態 (請參照章節10)。

類別:

RTC.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
***pbEnableDisable*:**	**0: 關閉功能** **1: 啟用功能**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
BYTE bSetting;
RSM_GET_FRNET_TRIGGER_EVENT_SETTING (hRsm, 1, & bSetting);
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	03	00 09	00 01

6 自動原點復歸

i-8094H 提供自動原點復歸功能，只要經過適當設定後，即可透過指令觸發開始自動執行，主要復歸的動作如下:

- **高速運動尋找近原點(NHOME)感測器。**
- **低速運動尋找原點(HOME)感測器。**
- **低速運動尋找伺服馬達的 Z 相訊號。**
- **高速運動指定的位移量。**

使用者可依照實際需求選擇所要執行的復歸動作，此功能為全自動執行，不需要消耗 Host 端的 CPU 資源，降低程式開發的工作負擔。

6.1 設定原點復歸速度(HV)

eRTU RSM_SET_HV (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

※ eRTU RSM_MACRO_SET_HV (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

說明:

設定原點搜尋速度。

類別:

Modbus sub_function; RTC and MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***data*:**	原點搜尋速度設定值 (Vmin~Vmax PPS)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_HV 的Sub_function代碼為0x0A 3C。
RSM_MACRO_SET_HV 的 Sub_function 代碼為 0x0C 3C。

Modbus 範例:

```
RSM_SET_HV (hRsm, 1, AXIS_X, 500);
//set the homing speed of the X axis on module 1 to be 500 PPS.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 3C/0C 3C	*Sub_function code*			
1	00 01	*axis*			
2	00 00	*MSW of data*			
3	01 F4	*LSW of data (500 = 0x1F4)*			

6.2 使用極限開關替代原點開關的功能設定

eRTU RSM_HOME_LIMIT (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nType*)

※ eRTU RSM_MACRO_HOME_LIMIT (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nType*)

說明:

設定使用極限開關替代為原點開關功能。

類別:

Modbus sub_function; RTC and MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***nType*:**	0: 關閉功能 1: 使用極限開關替代為原點開關

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_HOME_LIMIT 的Sub_function代碼為0x0A 3D。

RSM_MACRO_HOME_LIMIT 的 Sub_function 代碼為 0x0C 3D。

Modbus 範例:

```
RSM_HOME_LIMIT (hRsm, 1, AXIS_X, 0);
// Do not use the Limit Switch as the HOME sensor.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0A 3D/0C 3D	*Sub_function code*			
1	00 01	*axis*			
2	00 00	*nType*			

6.3 自動原點復歸模式設定

eRTU RSM_SET_HOME_MODE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nStep1*, BYTE *nStep2*, BYTE *nStep3*, BYTE *nStep4* , DWORD *data*)

※ eRTU RSM_MACRO_SET_HOME_MODE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nStep1*, BYTE *nStep2*, BYTE *nStep3*, BYTE *nStep4* , DWORD *data*)

說明:

設定自動原點復歸的動作模式與相關參數。

類別:

Modbus sub_function; RTC and MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***nStep1*:**	0: 不執行步驟 1 1: 朝正方向執行 2: 朝反方向執行
***nStep2*:**	0: 不執行步驟 2 1: 朝正方向執行 2: 朝反方向執行
***nStep3*:**	0: 不執行步驟 3 1: 朝正方向執行 2: 朝反方向執行
***nStep4*:**	0: 不執行步驟 4 1: 朝正方向修正位置 2: 朝反方向修正位置
***data*:**	步驟 4 的位置修正值 範圍: 0 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

自動原點復歸的四個步驟說明如下表所示

步驟	動作	速度	訊號
1	搜尋近原點感測器	V	NHOME (IN0)
2	搜尋原點感測器	HV	HOME (IN1)
3	搜尋 Z 相訊號	HV	Z-Phase (IN2)
4	移動到指定的位置	V	

備註:

RSM_SET_HOME_MODE 的Sub_function代碼為 0x0A 3E。

RSM_MACRO_SET_HOME_MODE 的 Sub_function 代碼為 0x0C 3E。

Modbus 範例:

RSM_SET_HOME_MODE (hRsm, 1, 0x1, 2, 2, 1, 1, 3500);

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 08	10
Register[]	**Value (hex)**	**Remarks**			
0	0A 3E/0C 3E	*Sub_function code*			
1	00 01	*axis*			
2	00 02	*nStep1*			
3	00 02	*nStep2*			
4	00 01	*nStep3*			
5	00 01	*nStep4*			
6	00 00	*MSW of data*			
7	0D AC	*LSW of data (3500 = 0xDAC)*			

6.4 開始執行自動原點復歸

eRTU RSM_HOME_START (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

※ eRTU RSM_MACRO_HOME_START (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

說明:

開始執行自動原點復歸。

類別:

Modbus sub_function; RTC and MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_HOME_START 的Sub_function代碼為0x0A 3F.
RSM_MACRO_HOME_START 的 Sub_function 代碼為 0x0C 3F.

Modbus 範例:

```
RSM_HOME_START (hRsm, 1, AXIS_X);
//start the automatic homing sequence for the X axis on module 1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A 3F/0C 3F	*Sub_function code*			
1	00 01	*axis (AXIS_X)*			

7 運動控制命令

7.1 單軸運動控制

- 單軸運動中，各軸可以在任意時間同時運動。
- 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，可以動態改變參數值，包含位移脈波數、速度....等等。
- 運動過程中可執行立即減速停止或立即停止，以順應我們對運動控制不同的需求。
- 可以搭配補間運動或同步運動，做更複雜及多樣化的運動控制。

7.1.1 加減速段速度模式設定

eRTU RSM_NORMAL_SPEED (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis* , BYTE *nMode*)

※Δ eRTU RSM_MACRO_NORMAL_SPEED (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis* , BYTE *nMode*)

說明:

設定運動控制的速度模式。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***nMode*:**	0=對稱 T-curve (請設定 SV, V, A, and AO) 1=對稱S-curve (請設定 SV, V, K, and AO) 2=非對稱T-curve (請設定 SV, V, A, D, and AO) 3=對稱S-curve (請設定 SV, V, K, L, and AO)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

請參照其他速度相關函式的參數設定說明。

RSM_NORMAL_SPEED 的Sub_function代碼為 0x0A 46。

RSM_MACRO_NORMAL_SPEED 的 Sub_function 代碼為 0x0C 46。

Modbus 範例:

```
RSM_NORMAL_SPEED (hRsm, 1, AXIS_XYZU, 0);
//use a symmetric T-curve for all axes on module 1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0A 46/0C 46	*Sub_function code*			
1	00 0F	*axis (AXIS_XYZU)*			
2	00 00	*nMode*			

相關範例:

```
BYTE SlaveAddr=1; //select module 1.
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU, 20000);
//set the max speed of XYZU axes to be 20K PPS.

//=========================================================
RSM_NORMAL_SPEED(hRsm, SlaveAddr, AXIS_XYZU, 0);
//use a symmetric T-        curve   for all axes on module 1.
RSM_SET_V(hRsm, SlaveAddr, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to be 2000 PPS.
RSM_SET_A(hRsm, SlaveAddr, AXIS_XYZU, 1000);
//set the acceleration of all axes on module 1 to be 1000 PPS/Sec.
RSM_SET_SV(hRsm, SlaveAddr, AXIS_XYZU, 2000);
//set the start speed of all axes on module 1 to be 2000 PPS.
RSM_SET_AO(hRsm, SlaveAddr, AXIS_XYZU, 9);
//set the number of remaining offset pulses for all axes to be 9 pulses.
RSM_FIXED_MOVE(hRsm, SlaveAddr, AXIS_XYZU, 10000);
//move all axes on module 1 to be 10000 pulses.

//=========================================================
RSM_NORMAL_SPEED(hRsm, SlaveAddr, AXIS_XYZU,1);
//use a symmetric S-        curve for all axes on module 1.
RSM_SET_V(hRsm, SlaveAddr, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to be 2000 PPS.
RSM_SET_K(hRsm, SlaveAddr, AXIS_XYZU, 500);
//set the acceleration rate of all axes on module 1 to be 500 PPS/sec^2.
RSM_SET_SV(hRsm, SlaveAddr, AXIS_XYZU, 200);
//set the start speed of all axes on module 1 to be 200 PPS.
RSM_SET_AO(hRsm, SlaveAddr, AXIS_XYZU, 9);
//set the number of remaining offset pulses to be 9 pulses for all axes.
RSM_FIXED_MOVE(hRsm, SlaveAddr, AXIS_XYZU, -10000);
//move all axes on module 1 to be 10000 pulses in reverse direction.

//=========================================================
RSM_NORMAL_SPEED(hRsm, SlaveAddr, AXIS_XYZU,2);
```

```
//use an asymmetric T-curve for all axes on module 1.
RSM_SET_V(hRsm, SlaveAddr, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to be 2000 PPS.
RSM_SET_A(hRsm, SlaveAddr, AXIS_XYZU, 1000 );
//set the acceleration of all axes on module 1 to be 1000 PPS/sec.
RSM_SET_D(hRsm, SlaveAddr, AXIS_XYZU, 500);
//set the deceleration of all axes on module 1 to be 500 PPS.
RSM_SET_SV(hRsm, SlaveAddr, AXIS_XYZU, 200);
//set the start speed of all axes on module 1 to 200 PPS.
RSM_SET_AO(hRsm, SlaveAddr, AXIS_XYZU, 9);
//set the number of remaining offset pulses to be 9 pulses for all axes.
RSM_FIXED_MOVE(hRsm, SlaveAddr, axis, 10000);
//move all axes on module 1 to be 10000 pulses.

//=========================================================
RSM_NORMAL_SPEED(hRsm, SlaveAddr, AXIS_XYZU,3);
//use an asymmetric S-curve for all axes on module 1.
RSM_SET_V(hRsm, SlaveAddr, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to be 2000 PPS.
RSM_SET_K(hRsm, SlaveAddr, AXIS_XYZU, 500);
//set the acceleration rate of all axes on module 1 to be 500 PPS/sec^2.
RSM_SET_L(hRsm, SlaveAddr, AXIS_XYZU, 300);
//set the deceleration rate of all axes on module 1 to be 300 PPS/sec^2.
RSM_SET_SV(hRsm, SlaveAddr, AXIS_XYZU, 200);
//set the start speed of all axes on module 1 to be 200 PPS.
RSM_SET_AO(hRsm, SlaveAddr, AXIS_XYZU, 9);
//set the number of remaining offset pulses to be 9 pulses for all axes.
RSM_FIXED_MOVE(hRsm, SlaveAddr, AXIS_XYZU, 10000);
//move all axes on module 1 to be 10000 pulses.
```

7.1.2 初始速度(SV)設定

eRTU RSM_SET_SV (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

※Δ eRTU RSM_MACRO_SET_SV (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

說明:

設定運動控制的初始速度。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***data*:**	初始速度設定值，單位為PPS(Pulse Per Second) (最大設定值請參照函式"RSM_SET_MAX_V()"說明)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_SV 的Sub_function代碼為 0x0A 47。
RSM_MACRO_SET_SV 的 Sub_function 代碼為 0x0C 47。

Modbus 範例:

```
RSM_SET_SV (hRsm, 1, AXIS_X, 1000);
//set the starting speed for the X axis on module 1 to 1000 PPS.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 47/0C 47	*Sub_function code*			
1	00 01	*axis (1 = AXIS_X)*			
2	00 00	*MSW of data*			
3	03 E8	*LSW of data (1000 = 0x3E8)*			

7.1.3 驅動速度(V)設定

eRTU RSM_SET_V (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

※Δ eRTU RSM_MACRO_SET_V (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

說明:

設定運動控制的驅動速度。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***data*:**	驅動速度設定值，單位為PPS(Pulse Per Second) (最大設定值請參照函式"RSM_SET_MAX_V()"說明)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_V 的Sub_function代碼為 0x0A 48。
RSM_MACRO_SET_V 的 Sub_function 代碼為 0x0C 48。

Modbus 範例:

```
RSM_SET_V (hRsm, 1, AXIS_X, 120000L);
//set the speed for the X axis on module 1 to 120000 PPS.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 48/0C 48	*Sub_function code*			
1	00 01	*axis (1 = AXIS_X)*			
2	00 01	*MSW of data*			
3	D4 C0	*LSW of data (120000 = 0x1D4C0)*			

7.1.4 加速度(A)設定

eRTU RSM_SET_A (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD data)

※Δ eRTU RSM_MACRO_SET_A (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

說明:

設定運動控制的加速度。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***data*:**	加速度的設定值，單位為PPS/Sec 請依實際操作環境與參照函式"RSM_SET_MAX_V()"的MAX_V設定值設定此參數 範圍: MAX_V ÷64 <= 加速度 <= MAX_V × 125

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_A 的Sub_function代碼為 0x0A 49。
RSM_MACRO_SET_A 的 Sub_function 代碼為 0x0C 49。

Modbus 範例:

```
RSM_SET_A (hRsm, 1, AXIS_X, 100000L);
//set the acceleration value of the X axis on module 1 to 100K PPS/Sec.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 49/0C 49	*Sub_function code*			
1	00 01	*axis (1 = AXIS_X)*			
2	00 01	*MSW of data*			
3	86 A0	*LSW of data (100000 = 0x186A0)*			

7.1.5 減速度(D)設定

eRTU RSM_SET_D (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

※Δ eRTU RSM_MACRO_SET_D (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

說明:

設定運動控制的減速度。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***data*:**	減速度的設定值，單位為PPS/Sec 請依實際操作環境與參照函式"RSM_SET_MAX_V()"的MAX_V設定值設定此參數 範圍: MAX_V ÷64 <= 減速度 <= MAX_V × 125

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_D 的Sub_function代碼為 0x0A 4A。

RSM_MACRO_SET_D 的 Sub_function 代碼為 0x0C 4A。

Modbus 範例:

```
RSM_SET_D (hRsm, 1, AXIS_X, 100000L);
//set the deceleration value of the X axis on module 1 to 100K PPS/sec.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 4A/0C 4A	*Sub_function code*			
1	00 01	*axis (1 = AXIS_X)*			
2	00 01	*MSW of data*			
3	86 A0	*LSW of data (100000 = 0x186A0)*			

7.1.6 加速度變化率(K)設定

eRTU RSM_SET_K (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

※Δ eRTU RSM_MACRO_SET_K (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

說明:

設定運動控制的加速度變化率。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***data*:**	加速度變化率的設定值，單位為PPS/Sec^2 請依實際操作環境與參照函式"RSM_SET_MAX_V()"的MAX_V設定值設定此參數 範圍: MAX_V ×0.0119211 <= 加速度變化率 <= 2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_K 的Sub_function代碼為 0x0A 4B。
RSM_MACRO_SET_K 的 Sub_function 代碼為 0x0C 4B。

Modbus 範例:

```
RSM_SET_K (hRsm, 1, AXIS_X, 10000);
//set the acceleration rate value of the X axis on module 1 to
//1,000*10 (= 10,000) PPS/Sec^2.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 4B/0C 4B	*Sub_function code*			
1	00 01	*axis (1 = AXIS_X)*			
2	00 00	*MSW of data*			
3	27 10	*LSW of data (10000 = 0x2710)*			

7.1.7 減速度率(L)設定

eRTU RSM_SET_L (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

※Δ eRTU RSM_MACRO_SET_L (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

說明:

設定運動控制的減速度變化率。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***data*:**	減速度變化率的設定值，單位為PPS/Sec^2 請依實際操作環境與參照函式"RSM_SET_MAX_V()"的MAX_V設定值設定此參數 範圍: MAX_V ×0.0119211 <= 減速度變化率 <= 2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_L 的Sub_function代碼為 0x0A 4C。
RSM_MACRO_SET_L 的 Sub_function 代碼為 0x0C 4C。

Modbus 範例:

```
RSM_SET_L (hRsm, 1, AXIS_X, 10000);
//set the deceleration rate value of the X axis on module 1 to
//1,000*10 (= 10,000) PPS/Sec^2.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 4C/0C 4C	*Sub_function code*			
1	00 01	*axis (1 = AXIS_X)*			
2	00 00	*MSW of data*			
3	27 10	*LSW of data (10000 = 0x2710)*			

7.1.8 減速段補償(AO)設定

eRTU RSM_SET_AO (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long *data*)

※Δ eRTU RSM_MACRO_SET_AO (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long *data*)

說明:

設定運動控制的減速段補償。在使用非對稱T-curve速度曲線或S-curve速度曲線進行運動時，若設定過低的初速度可能導致減速段開始的位置錯誤，使得運動結束時的速度異常，此時可使用此函式設定補償值改善問題。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***data*:**	減速段補償的設定值 範圍: -32,768 ~ +32,767(預設值 = 8)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

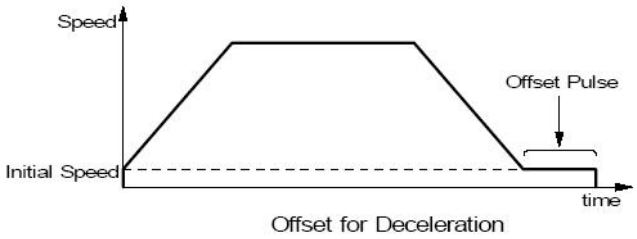
備註:

RSM_SET_AO 的Sub_function代碼為 0x0A 4D。

RSM_MACRO_SET_AO 的 Sub_function 代碼為 0x0C 4D。

Modbus 範例:

```
RSM_SET_AO (hRsm, 1, AXIS_X, 200);
//set the number of remaining offset pulses for the X axis on module 1 to 200 pulses.
```



所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 4D/0C 4D	*Sub_function code*			
1	00 01	*axis (1 = AXIS_X)*			
2	00 00	*MSW of data*			
3	00 C8	*LSW of data (200 = 0xC8)*			

7.1.9 相對距離移動

eRTU RSM_FIXED_MOVE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long *data*)

※Δ eRTU RSM_MACRO_FIXED_MOVE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long *data*)

說明:

執行一個相對距離的移動。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***data*:**	脈波輸出量設定值(移動的距離) 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

備註:

RSM_FIXED_MOVE 的Sub_function代碼為 0x0A 4E。

RSM_MACRO_FIXED_MOVE 的 Sub_function 代碼為 0x0C 4E。

Modbus 範例:

```
RSM_FIXED_MOVE (hRsm, SlaveAddr, AXIS_XYZU, 10000);
// AXIS_XYZU move 10000 Pulses
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 4E/0C 4E	*Sub_function code*			
1	00 0F	*axis (0xF = AXIS_XYZU)*			
2	00 00	*MSW of data*			
3	27 10	*LSW of data (10000 = 0x2710)*			

相關範例:

```
BYTE SlaveAddr=1; //select module 1
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU, 20000);
//set the max. velocity of all axes on module 1 to be 20K PPS
RSM_NORMAL_SPEED(hRsm, SlaveAddr, AXIS_XYZU, 0);
//set the speed profile of all axes on module 1 to be symmetric T-curve
RSM_SET_V(hRsm, SlaveAddr, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to be 2000 PPS
RSM_SET_A(hRsm, SlaveAddr, AXIS_XYZU,1000);
//set the acceleration value of all axes on module 1 to be 1000 PPS/S
RSM_SET_SV(hRsm, SlaveAddr, AXIS_XYZU, 2000);
//set the start velocity of all          axes on module 1 to be 2000 PPS
RSM_SET_AO(hRsm, SlaveAddr, AXIS_XYZU, 9);
//set the remaining offset pulses to be 9 PPS
RSM_FIXED_MOVE(hRsm, SlaveAddr, AXIS_XYZU, 10000);
// move 10000 Pulses for each axis on module 1
```

7.1.10 動態目標位置改變

eRTU RSM_SET_PULSE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

※Δ eRTU RSM_MACRO_SET_PULSE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

說明:

在移動的過程中，使用本函式可動態改變原本的脈波輸出量設定值。請注意，當變更後的脈波輸出量少於已送出的脈波量將會立即停止運動。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***data*:**	變更後的脈波輸出量設定值(變更後的移動距離) 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_PULSE 的Sub_function代碼為 0x0A 4F。

RSM_MACRO_SET_PULSE 的 Sub_function 代碼為 0x0C 4F。

Modbus 範例:

```
RSM_SET_PULSE (hRsm, SlaveAddr, AXIS_XYZU, 9000);
//Change the total pulses to 9000 pulses during fixed pulse moving.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 4F/0C 4F	*Sub_function code*			
1	00 0F	*axis (0xF = AXIS_XYZU)*			
2	00 00	*MSW of data*			
3	23 28	*LSW of data (9000 = 0x2328)*			

相關範例:

```
BYTE SlaveAddr=1; //select module 1
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU, 20000);
//set the max velocity of all axes on module 1 to be 20K PPS
RSM_NORMAL_SPEED(hRsm, SlaveAddr, AXIS_XYZU, 0);
//set the speed profile of all axes on module 1 to be symmetric T-curve
RSM_SET_V(hRsm, SlaveAddr, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to be 2000 PPS
RSM_SET_A(hRsm, SlaveAddr, AXIS_XYZU,1000);
//set the acceleration value of all axes on module 1 to be 1000 PPS/S
RSM_SET_SV(hRsm, SlaveAddr, AXIS_XYZU, 2000);
//set the start velocity of all          axes on module 1 to be 2000 PPS
RSM_SET_AO(hRsm, SlaveAddr, AXIS_XYZU, 9);
//set the remaining offset pulses to be 9 PPS
RSM_FIXED_MOVE(hRsm, SlaveAddr, AXIS_XYZU, 10000);
// move 10000 Pulses for each axis on module 1
RSM_SET_PULSE(hRsm, SlaveAddr, AXIS_XYZU, 9000);
//Set pulse as 9000 Pulse.
```

7.1.11 絕對位置移動

eRTU RSM_ABS_FIXED_MOVE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long *data*)

※Δ eRTU RSM_ABS_MACRO_FIXED_MOVE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long *data*)

說明:

移動到指定的絕對位置。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***data*:**	絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

備註:

RSM_ABS_FIXED_MOVE的Sub_function代碼為 0x0A F6。

RSM_ABS_MACRO_FIXED_MOVE 的 Sub_function 代碼為 0x0C F6。

Modbus 範例:

```
RSM_ABS_FIXED_MOVE (hRsm, SlaveAddr, AXIS_XYZU, 10000);
// AXIS_XYZU moves from the current position to the logic position 10000 in a straight line
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A F6/0C F6	*Sub_function code*			
1	00 0F	*axis (0xF = AXIS_XYZU)*			
2	00 00	*MSW of data*			
3	27 10	*LSW of data (10000 = 0x2710)*			

相關範例:

```
BYTE SlaveAddr=1; //select module 1
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU, 20000);
//set the max. velocity of all axes on module 1 to be 20K PPS
RSM_NORMAL_SPEED(hRsm, SlaveAddr, AXIS_XYZU, 0);
//set the speed profile of all axes on module 1 to be symmetric T-curve
RSM_SET_V(hRsm, SlaveAddr, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to be 2000 PPS
RSM_SET_A(hRsm, SlaveAddr, AXIS_XYZU,1000);
//set the acceleration value of all axes on module 1 to be 1000 PPS/S
RSM_SET_SV(hRsm, SlaveAddr, AXIS_XYZU, 2000);
//set the start velocity of all          axes on module 1 to be 2000 PPS
RSM_SET_AO(hRsm, SlaveAddr, AXIS_XYZU, 9);
//set the remaining offset pulses to be 9 PPS
RSM_ABS_FIXED_MOVE(hRsm, SlaveAddr, AXIS_XYZU, 10000);
// All four axis are moving to the absolute position 10000
```

7.1.12 連續運動控制脈波輸出

eRTU RSM_CONTINUE_MOVE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long *data*)

※Δ eRTU RSM_MACRO_CONTINUE_MOVE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long *data*)

說明:

開始連續送出運動控制脈波(持續移動)，直到執行停止指令或外部停止訊號觸發才會停止移動。移動的過程中可動態的改變速度。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
***axis*:**	**軸號設定值 (請參照表4)**
***data*:**	**移動速度的設定值，單位為PPS(Pulse Per Second)** **>0 = 正方向移動** **<0 = 反方向移動**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

備註:

RSM_CONTINUE_MOVE 的Sub_function代碼為 0x0A 50。

RSM_MACRO_CONTINUE_MOVE 的 Sub_function 代碼為 0x0C 50。

Modbus 範例:

```
RSM_CONTINUE_MOVE (hRsm, SlaveAddr, AXIS_XYZU, 1000);
//continue to move at the speed of 1K PPS
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 50/0C 50	*Sub_function code*			
1	00 0F	*axis (0xF = AXIS_XYZU)*			
2	00 00	*MSW of data*			
3	03 E8	*LSW of data (1000 = 0x3E8)*			

相關範例:

```
BYTE SlaveAddr=1; //select module 1
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU, 20000);
//set the maximum speed of all axes on module 1 to 20K PPS.
RSM_NORMAL_SPEED(hRsm, SlaveAddr, AXIS_XYZU, 0);
//set the speed profile for all axes as a symmetric T-curve.
RSM_SET_V(hRsm, SlaveAddr, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to 2000 PPS.
RSM_SET_A(hRsm, SlaveAddr, AXIS_XYZU,1000);
//set the acceleration value of all axes to 1000 PPS/S.
RSM_SET_SV(hRsm, SlaveAddr, AXIS_XYZU, 2000);
//set the start velocity of all          axes to 2000 PPS
RSM_CONTINUE_MOVE(hRsm, SlaveAddr, AXIS_XYZU, 1000);
//move all axes on module 1 at a speed of 1000 PPS.
```

7.2 補間命令

7.2.1 指定補間軸

eRTU RSM_AXIS_ASSIGN (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis1*, BYTE *axis2*, BYTE *axis3*)

※Δ eRTU RSM_MACRO_AXIS_ASSIGN (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis1*, BYTE *axis2*, BYTE *axis3*)

說明:

指定用來執行補間運動的軸號。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis1*:**	第一個補間軸的軸號設定值 1=X軸; 2=Y軸; 4=Z軸; 8=U軸
***axis2*:**	第二個補間軸的軸號設定值 1=X軸; 2=Y軸; 4=Z軸; 8=U軸
***axis3*:**	第三個補間軸的軸號設定值 1=X軸; 2=Y軸; 4=Z軸; 8=U軸(若為兩軸補間請設定0)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_AXIS_ASSIGN 的Sub_function代碼為 0x0A 5A。
RSM_MACRO_AXIS_ASSIGN 的 Sub_function 代碼為 0x0C 5A。

Modbus 範例:

```
RSM_AXIS_ASSIGN (hRsm, 1, AXIS_X, AXIS_Y, 0);
//set the X axis of module 1 as the first axis and the Y axis as the second axis.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 5A/0C 5A	*Sub_function code*			
1	00 01	*axis1 (axis1 = AXIS_X)*			
2	00 02	*axis2 (axis1 = AXIS_Y)*			
3	00 00	*Axis3 (not defined)*			

7.2.2 速度模式設定

eRTU RSM_VECTOR_SPEED (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *nMode*)

※Δ eRTU RSM_MACRO_VECTOR_SPEED (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *nMode*)

說明:

設定補間運動的速度模式。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***nMode*:**	見下表

模式	補間類型	速度曲線	需設定參數
0	兩軸線性 或圓弧	固定向量速度	VV, VSV 且 VV=VSV
1	兩軸線性	對稱T-curve速度曲線	VSV, VV, VA, VAO
2	兩軸線性	對稱S-curve速度曲線	VSV, VV, VK, VAO
3	兩軸線性	非對稱T-curve速度曲線	VSV, VV, VA, VD, VAO
4	兩軸線性	非對稱S-curve速度曲線	VSV, VV, VK, VL, VAO
5	兩軸圓弧	對稱T-curve速度曲線	VSV, VV, VA, VAO
6	兩軸圓弧	非對稱T-curve速度曲線	VSV, VV, VA, VD, VAO
7	三軸線性	固定向量速度	VV, VSV 且 VV=VSV
8	三軸線性	對稱T-curve速度曲線	VSV, VV, VA, VAO
9	三軸線性	對稱S-curve速度曲線	VSV, VV, VK, VAO
10	三軸線性	非對稱T-curve速度曲線	VSV, VV, VA, VD, VAO
11	三軸線性	非對稱S-curve速度曲線	VSV, VV, VK, VL, VAO

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

請參照其他速度相關函式的參數設定說明。

RSM_VECTOR_SPEED 的Sub_function代碼為 0x0A 5B。

RSM_MACRO_VECTOR_SPEED 的 Sub_function 代碼為 0x0C 5B。

Modbus 範例:

```
RSM_VECTOR_SPEED (hRsm, SlaveAddr, 0);
//set the module to perform 2-axis linear or circular motion
//at a constant vector speed.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A 5B/0C 5B	*Sub_function code*			
1	00 00	*nMode*			

相關範例:

```
BYTE SlaveAddr=1; //select module 1.
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU, 20000);
//set the maximum speed of all axes to 20K PPS.
//=============================================
RSM_AXIS_ASSIGN(hRsm, SlaveAddr, AXIS_X, AXIS_Y, 0);
//set the X axis as the first axis and the Y axis as the second axis.
RSM_VECTOR_SPEED(hRsm, SlaveAddr, 0);
//set module 1 to perform 2-axis linear or circular motion
RSM_SET_VV(hRsm, SlaveAddr, 1000);
//set the vector speed to 1000 PPS.
RSM_LINE_2D(hRsm, SlaveAddr, 12000, 10000);
//execute the 2-axis linear interpolation motion.
//=============================================
RSM_AXIS_ASSIGN(hRsm, SlaveAddr, AXIS_X, AXIS_Y, 0);
//set the X axis as the first axis and the Y axis as the second axis.
RSM_VECTOR_SPEED(hRsm, SlaveAddr, 1);
//set module 1 to perform 2-axis linear motion using a symmetric
//T-curve velocity profile (VSV、VV、VA、VAO).
RSM_SET_VSV(hRsm, SlaveAddr, 500); //set the starting vector speed to 500 PPS.
RSM_SET_VV(hRsm, SlaveAddr, 2000); //set the vector speed to 2000 PPS.
RSM_SET_VA(hRsm, SlaveAddr, 1000);
//set the vector acceleration to 1000 PPS/sec.
RSM_LINE_2D(hRsm, SlaveAddr, 20000, 10000);
//execute the 2-axis linear interpolation motion.
```

```
    //=============================================
    RSM_AXIS_ASSIGN(hRsm, SlaveAddr, AXIS_X, AXIS_Y, 0);
    //set the X axis as the first axis and the Y axis as the second axis.
    RSM_VECTOR_SPEED(hRsm, SlaveAddr, 2);
    //2-axis linear motion using a symmetric S-curve velocity profile(VSV、VV、
//VK、AO).
    RSM_SET_VSV(hRsm, SlaveAddr, 200); //set the starting vector speed to 200 PPS.
    RSM_SET_VV(hRsm, SlaveAddr, 2000); //set the vector speed to 2000 PPS.
    RSM_SET_VK(hRsm, SlaveAddr, 500); //set the acceleration rate to 500 PPS/Sec.
    RSM_SET_VAO(hRsm, SlaveAddr, 20);
    //set the value of remaining offset pulses to 20.
    RSM_LINE_2D(hRsm, SlaveAddr, 10000, 10000);
    //execute the 2-axis linear interpolation motion.
    //=============================================
    RSM_AXIS_ASSIGN(hRsm, SlaveAddr, AXIS_X, AXIS_Y, 0);
    //set the X axis as the first axis and the Y axis as the second axis.
    RSM_VECTOR_SPEED(hRsm, SlaveAddr, 3);
    //2-axis linear motion using an asymmetric T-curve velocity profile(VSV、        //VV、
VA、VD、VAO).
    RSM_SET_VSV(hRsm, SlaveAddr, 100); //set the start vector speed to 100 PPS.
    RSM_SET_VV(hRsm, SlaveAddr, 2000); //set the vector speed to 2000 PPS.
    RSM_SET_VA(hRsm, SlaveAddr, 1000);
    //set the vector acceleration to 1000 PPS/sec.
    RSM_SET_VD(hRsm, SlaveAddr, 500); //set the vector deceleration to 500 PPS/sec.
    RSM_SET_VAO(hRsm, SlaveAddr, 20);
    //set the value of remaining offset pulses to 20.
    RSM_LINE_2D(hRsm, SlaveAddr, 10000, 5000);
    //execute the 2-axis linear interpolation motion.
    //=============================================
    long fp1=4000;
    long fp2=10000;
    unsigned short  sv=200;
    unsigned short  v=2000;
    RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU, 8000);
    RSM_AXIS_ASSIGN(hRsm, SlaveAddr, AXIS_X, AXIS_Y, 0);
    //set the X axis as the first axis and the Y axis as the second axis.
    RSM_VECTOR_SPEED(hRsm, SlaveAddr, 4);
    //2-axis linear motion using an asymmetric S-curve velocity profile(VSV、
//VV、VK、VL、VAO).
    RSM_SET_VSV(hRsm, SlaveAddr, sv); //set the starting velocity to sv PPS.
    RSM_SET_VV(hRsm, SlaveAddr, v); //set the vector speed to v PPS.
    RSM_SET_VK(hRsm, SlaveAddr, 500); //set the acceleration rate to 500 PPS/Sec^2.
    RSM_SET_VL(hRsm, SlaveAddr, 300); //set the deceleration rate to 300 PPS/Sec^2.
    RSM_SET_VAO(hRsm, SlaveAddr, 20);
    //set the value of remaining offset pulses to 20.
    RSM_LINE_2D(hRsm, SlaveAddr, fp1, fp2); //execute the 2-axis linear motion.
```

```
//=============================================
long fp1=11000;
long fp2=9000;
long c1=10000;
long c2=0;
unsigned short  sv=100;
unsigned short  v=3000;
unsigned long  a=5000;
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU, 8000);
RSM_AXIS_ASSIGN(hRsm, SlaveAddr, AXIS_X, AXIS_Y, 0);
//set the X axis as the first axis and the Y axis as the second axis.
RSM_VECTOR_SPEED(hRsm, SlaveAddr, 5);
//2-axis circular motion using a symmetric T-curve velocity profile(VSV、// VV、VA、
VAO).
RSM_SET_VSV(hRsm, SlaveAddr, sv); //set the starting vector speed to sv PPS.
RSM_SET_VV(hRsm, SlaveAddr, v); //set vector speed to v PPS.
RSM_SET_VA(hRsm, SlaveAddr, a); //set the vector acceleration to a PPS/sec.
RSM_SET_VAO(hRsm, SlaveAddr, 0);
//set the value of remaining offset pulses to 0 Pulse.
RSM_ARC_CW(hRsm, SlaveAddr, c1,c2, fp1, fp2);
//execute the 2-axis CW circular motion.
//=============================================
long c1=300;
long c2=0;
unsigned short sv=100;
unsigned short v=3000;
unsigned long a=125;
unsigned long d=12;
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU, 8000);
RSM_AXIS_ASSIGN(hRsm, SlaveAddr, AXIS_X, AXIS_Y, 0);
//set the X axis as the first axis and the Y axis as the second axis.
RSM_VECTOR_SPEED(hRsm, SlaveAddr, 6);
//2-axis circular motion using an asymmetric T-curve velocity
//profile(VSV、VV、VA、VD、VAO).
RSM_SET_VSV(hRsm, SlaveAddr, sv); //set the starting vector speed to sv PPS.
RSM_SET_VV(hRsm, SlaveAddr, v); //set vector speed to v PPS.
RSM_SET_VA(hRsm, SlaveAddr, a); //set acceleration to a PPS/Sec.
RSM_SET_VD(hRsm, SlaveAddr, d); //set the deceleration to d PPS/Sec.
RSM_SET_VAO(hRsm, SlaveAddr, 0); //set the value of remaining offset pulses to 0.
RSM_CIRCLE_CW(hRsm, SlaveAddr, c1, c2);
//execute the 2-axis CW circular motion.
//=============================================
RSM_AXIS_ASSIGN(hRsm, SlaveAddr, AXIS_X, AXIS_Y, AXIS_Z);
// set axis1 as the X axis, axis2 as the Y axis, and axis3 as the Z axis.
RSM_VECTOR_SPEED(hRsm, SlaveAddr, 7);
//3-axis linear motion at a constant vector speed (VSV=VV).
```

```
    RSM_SET_VSV(hRsm, SlaveAddr, 1000); //set the start speed to 1000 PPS.
    RSM_SET_VV(hRsm, SlaveAddr, 1000); //set the constant speed to 1000 PPS.
    RSM_LINE_3D(hRsm, SlaveAddr, 10000, 10000,10000);
    //execute the 3-axis linear motion.
    //=============================================
    RSM_AXIS_ASSIGN(hRsm, SlaveAddr, AXIS_X, AXIS_Y, AXIS_Z);
    // set axis1 as the X axis, axis2 as the Y axis, and axis3 as the Z-axis.
    RSM_VECTOR_SPEED(hRsm, SlaveAddr, 8);
    //3-axis linear motion using a symmetric T-curve velocity profile(VSV、VV、
//VA、VAO).
    RSM_SET_VSV(hRsm, SlaveAddr, 100); //set the starting speed to 100 PPS.
    RSM_SET_VV(hRsm, SlaveAddr, 3000); //set the vector speed to 3000 PPS.
    RSM_SET_VA(hRsm, SlaveAddr, 500); //set the vector acceleration to 500 PPS/Sec.
    RSM_SET_VAO(hRsm, SlaveAddr, 20);
    //set the value of remaining offset pulses to 20.
    RSM_LINE_3D(hRsm, SlaveAddr, 10000, 1000,20000);
    //execute the 3-axis linear motion
    //=============================================
    RSM_AXIS_ASSIGN(hRsm, SlaveAddr, AXIS_X, AXIS_Y, AXIS_Z);
    // set the axis1 as the X axis, axis2 as the Y axis, and axis3 as the Z axis.
    RSM_VECTOR_SPEED(hRsm, SlaveAddr, 9);
    //3-axis linear motion using a symmetric S-curve velocity profile(VSV、VV、
//VK、VAO).
    RSM_SET_VSV(hRsm, SlaveAddr, 100); //set the starting speed to 100 PPS.
    RSM_SET_VV(hRsm, SlaveAddr, 3000); //set the vector speed to 3000 PPS.
    RSM_SET_VK(hRsm, SlaveAddr, 500);
    //set the vector acceleration rate to 500 PPS/sec^2.
    RSM_SET_VAO(hRsm, SlaveAddr, 20);
    //set the value of remaining offset pulses to 20.
    RSM_LINE_3D(hRsm, SlaveAddr, 10000, 1000,1000);
    //execute the 3-axis linear motion.
    //=============================================
    RSM_AXIS_ASSIGN(hRsm, SlaveAddr, AXIS_X, AXIS_Y, AXIS_Z);
    // set the axis1 as the X axis, axis2 as the Y axis, and axis3 as the Z axis.
    RSM_VECTOR_SPEED(hRsm, SlaveAddr, 10);
    //set the module 1 to perform 3-axis linear motion
    //using an asymmetric T-curve speed profile(VSV、VV、VA、VD、VAO).
    RSM_SET_VSV(hRsm,SlaveAddr, 100); //set the starting speed to 100 PPS.
    RSM_SET_VV(hRsm,SlaveAddr, 2000); //set the vector speed as 2000 PPS.
    RSM_SET_VA(hRsm, SlaveAddr, 1000);
    //set the vector acceleration to 1000 PPS/sec.
    RSM_SET_VD(hRsm, SlaveAddr, 500); //set the vector deceleration to 500 PPS/sec.
    RSM_SET_VAO(hRsm, SlaveAddr, 20);
    //set the value of remaining offset pulses to 20.
    RSM_LINE_3D(hRsm, SlaveAddr, 10000, 1000,1000);
    //execute the 3-axis linear motion.
```

```
//=============================================
long fp1=4000;
long fp2=10000;
long fp3=20000;
unsigned short sv=200;
unsigned short v=2000;
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU, 8000);
RSM_AXIS_ASSIGN(hRsm, SlaveAddr, AXIS_X, AXIS_Y, AXIS_Z);
// set axis1 as the X axis, axis2 as the Y axis, and axis3 as the Z axis.
RSM_VECTOR_SPEED(hRsm, SlaveAddr, 11);
//3-axis linear motion using an asymmetric S-curve velocity profile(VSV、      //VV、
VK、VL、VAO).
RSM_SET_VSV(hRsm, SlaveAddr, sv); //set the starting speed to sv PPS.
RSM_SET_VV(hRsm, SlaveAddr, v); //set the vector speed to v PPS.
RSM_SET_VK(hRsm, SlaveAddr, 500);
//set the vector acceleration rate to 500 PPS/sec^2.
RSM_SET_VL(hRsm, SlaveAddr, 300);
//set the vector deceleration rate to 300 PPS/sec^2.
RSM_SET_VAO(hRsm, SlaveAddr, 20);
//set the value of remaining offset pulses to 20.
RSM_LINE_3D(hRsm, SlaveAddr, fp1, fp2, fp3);
//execute the 3-axis linear motion.
```

7.2.3 向量初始速度(VSV)設定

eRTU RSM_SET_VSV (HANDLE *hRsm*, BYTE *SlaveAddr*, DWORD *data*)

※Δ eRTU RSM_MACRO_SET_VSV (HANDLE *hRsm*, BYTE *SlaveAddr*, DWORD *data*)

說明:

設定補間運動的向量初始速度。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***data*:**	向量初始速度設定值，單位為PPS(Pulse Per Second) (最大設定值請參照函式"RSM_SET_MAX_V()"說明)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_VSV 的Sub_function代碼為 0x0A 5C。
RSM_MACRO_SET_VSV 的 Sub_function 代碼為 0x0C 5C。

Modbus 範例:

```
RSM_SET_VSV (hRsm, 1, 1000);
//set the starting speed of the axis 1 for the interpolation motion
//on module 1 to 1000 PPS.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0A 5C/0C 5C	*Sub_function code*			
1	00 00	*MSW of data*			
2	03 E8	*LSW of data (1000 = 0x3E8)*			

7.2.4 向量驅動速度(VV)設定

eRTU RSM_SET_VV (HANDLE *hRsm*, BYTE *SlaveAddr*, DWORD *data*)

※Δ eRTU RSM_MACRO_SET_VV (HANDLE *hRsm*, BYTE *SlaveAddr*, DWORD *data*)

說明:

設定補間運動的向量驅動速度。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***data*:**	向量驅動速度設定值，單位為PPS(Pulse Per Second) (最大設定值請參照函式"RSM_SET_MAX_V()"說明)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_VV 的Sub_function代碼為 0x0A 5D。

RSM_MACRO_SET_VV 的 Sub_function 代碼為 0x0C 5D。

Modbus 範例:

```
RSM_SET_VV (hRsm, 1, 120000L);
//set the vector speed of the interpolation on module 1 to 120000 PPS.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0A 5D/0C 5D	*Sub_function code*			
1	00 01	*MSW of data*			
2	D4 C0	*LSW of data (120000 = 0x1D4C0)*			

7.2.5 向量加速度(VA)設定

eRTU RSM_SET_VA (HANDLE *hRsm*, BYTE *SlaveAddr*, DWORD *data*)

※Δ eRTU RSM_MACRO_SET_VA (HANDLE *hRsm*, BYTE *SlaveAddr*, DWORD *data*)

說明:

設定補間運動的向量加速度。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***data*:**	向量加速度的設定值，單位為PPS/Sec 請依實際操作環境與參照函式"RSM_SET_MAX_V()"的MAX_V設定值設定此參數 範圍: MAX_V ÷64 <= 向量加速度 <= MAX_V × 125

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_VA 的Sub_function代碼為 0x0A 5E。

RSM_MACRO_SET_VA 的 Sub_function 代碼為 0x0C 5E。

Modbus 範例:

RSM_SET_VA (hRsm, 1, 100000L);

//set the vector acceleration of the interpolation motion on module 1 to 100K PPS/sec.

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0A 5E/0C 5E	*Sub_function code*			
1	00 01	*MSW of data*			
2	86 A0	*LSW of data (100000 = 0x186A0)*			

7.2.6 向量減速度(VD)設定

eRTU RSM_SET_VD (HANDLE *hRsm*, BYTE *SlaveAddr*, DWORD *data*)

※Δ eRTU RSM_MACRO_SET_VD (HANDLE *hRsm*, BYTE *SlaveAddr*, DWORD *data*)

說明:

設定補間運動的向量減速度。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***data*:**	向量減速度的設定值，單位為PPS/Sec 請依實際操作環境與參照函式"RSM_SET_MAX_V()"的MAX_V設定值設定此參數 範圍: MAX_V ÷64 <= 向量減速度 <= MAX_V × 125

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_VD 的Sub_function代碼為 0x0A 5F。
RSM_MACRO_SET_VD 的 Sub_function 代碼為 0x0C 5F。

Modbus 範例:

RSM_SET_VD (hRsm, 1, 100000L);
//set the vector deceleration value of interpolation motion on module 1 to 100K PPS/Sec.

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0A 5F/0C 5F	*Sub_function code*			
1	00 01	*MSW of data*			
2	86 A0	*LSW of data (100000 = 0x186A0)*			

7.2.7 向量加速度變化率(VK)設定

eRTU RSM_SET_VK (HANDLE *hRsm*, BYTE *SlaveAddr*, DWORD *data*)

※Δ eRTU RSM_MACRO_SET_VK (HANDLE *hRsm*, BYTE *SlaveAddr*, DWORD *data*)

說明:

設定向量加速度變化率。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***data*:**	向量加速度變化率的設定值，單位為PPS/Sec^2 請依實際操作環境與參照函式"RSM_SET_MAX_V()"的MAX_V設定值設定此參數 範圍: MAX_V ×0.0119211 <= 向量加速度變化率 <= 2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_VK 的Sub_function代碼為 0x0A 60。
RSM_MACRO_SET_VK 的 Sub_function 代碼為 0x0C 60。

Modbus 範例:

```
RSM_SET_VK (hRsm, 1, 10000);
//set the acceleration rate of the interpolation motion on module 1 to 10,000 PPS/sec^2.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0A 60/0C 60	*Sub_function code*			
1	00 00	*MSW of data*			
2	27 10	*LSW of data (10000 = 0x2710)*			

7.2.8 向量減速度變化率(VL)設定

eRTU RSM_SET_VL (HANDLE *hRsm*, BYTE *SlaveAddr*, DWORD *data*)

※Δ eRTU RSM_MACRO_SET_VL (HANDLE *hRsm*, BYTE *SlaveAddr*, DWORD *data*)

說明:

設定補間運動的減速度變化率。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***data*:**	向量減速度變化率的設定值，單位為PPS/Sec^2 請依實際操作環境與參照函式"RSM_SET_MAX_V()"的MAX_V設定值設定此參數 範圍: MAX_V ×0.0119211 <= 向量減速度變化率 <= 2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_VL 的Sub_function代碼為 0x0A 61。

RSM_MACRO_SET_VL 的 Sub_function 代碼為 0x0C 61。

Modbus 範例:

```
RSM_SET_VL (hRsm, 1, 10000);
//set the deceleration rate of the interpolation on module 1 to 10,000 PPS/sec^2.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0A 61/0C 61	*Sub_function code*			
1	00 00	*MSW of data*			
2	27 10	*LSW of data (10000 = 0x2710)*			

7.2.9 向量位移量(VAO)設定

eRTU RSM_SET_VAO (HANDLE *hRsm*, BYTE *SlaveAddr*, long *data*)

※Δ eRTU RSM_MACRO_SET_VAO (HANDLE *hRsm*, BYTE *SlaveAddr*, long *data*)

說明:

設定補間運動的減速段補償。在使用非對稱T-curve速度曲線或S-curve速度曲線進行運動時，若設定過低的初速度可能導致減速段開始的位置錯誤，使得運動結束時的速度異常，此時可使用此函式設定補償值改善問題。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***data*:**	向量運動減速段補償的設定值 範圍: -32,768 ~ +32,767(預設值 = 8)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

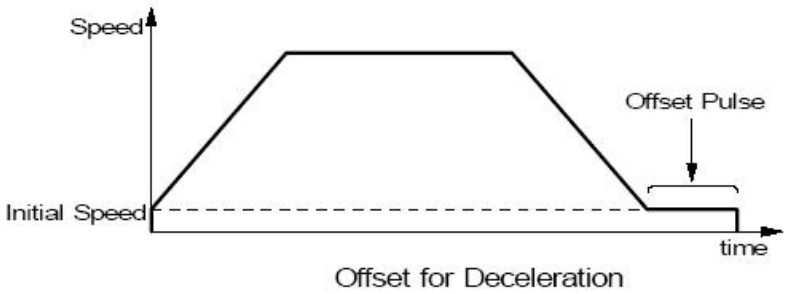
備註:

RSM_SET_VAO 的Sub_function代碼為 0x0A 62。

RSM_MACRO_SET_VAO 的 Sub_function 代碼為 0x0C 62。

Modbus 範例:

```
RSM_SET_VAO (hRsm, 1, 200);
//set the number of remaining offset pulse value on module 1 to 200.
```



所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0A 62/0C 62	*Sub_function code*			
1	00 00	*MSW of data*			
2	00 C8	*LSW of data (200 = 0xC8)*			

7.2.10 兩軸相對距離線性補間移動

eRTU RSM_LINE_2D (HANDLE *hRsm*, BYTE *SlaveAddr*, long *fp1*, long *fp2*)

※Δ eRTU RSM_MACRO_LINE_2D (HANDLE *hRsm*, BYTE *SlaveAddr*, long *fp1*, long *fp2*)

說明:

開始執行一個兩軸相對距離線性補間移動。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***fp1*:**	補間軸1的移動距離 範圍: -2,000,000,000 ~ +2,000,000,000
***fp2*:**	補間軸2的移動距離 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

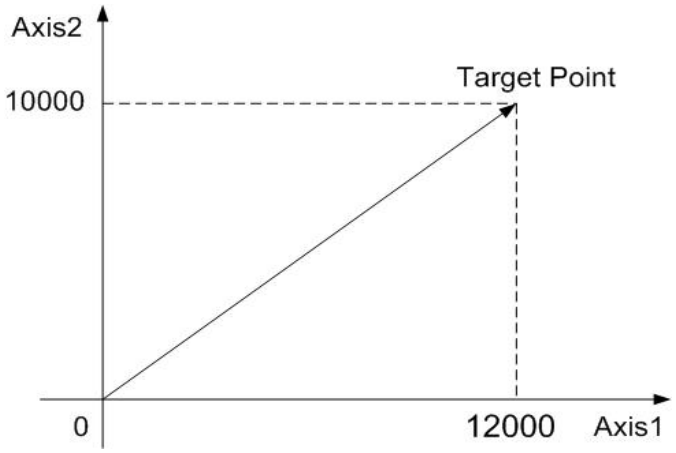
0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_LINE_2D 的Sub_function代碼為 0x0A 63。
RSM_MACRO_LINE_2D 的 Sub_function 代碼為 0x0C 63。

Modbus 範例:

```
RSM_LINE_2D (hRsm, 1, 12000, 10000);
//execute the 2-axis linear interpolation motion on module 1.
```



兩軸線性補間運動

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 05	0A
Register[]	**Value (hex)**	**Remarks**			
0	0A 63/0C 63	*Sub_function code*			
1	00 00	*MSW of fp1*			
2	2E E0	*LSW of fp1 (12000 = 0x2EE0)*			
3	00 00	*MSW of fp2*			
4	27 10	*LSW of fp2 (10000 = 0x2710)*			

7.2.11 兩軸絕對位置線性補間移動

eRTU RSM_ABS_LINE_2D (HANDLE *hRsm*, BYTE *SlaveAddr*, long *fp1*, long *fp2*)

※Δ eRTU RSM_ABS_MACRO_LINE_2D (HANDLE *hRsm*, BYTE *SlaveAddr*, long *fp1*, long *fp2*)

說明:

開始執行一個兩軸絕對位置線性補間移動。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***fp1*:**	補間軸1的絕對位置設定值() 範圍: -2,000,000,000 ~ +2,000,000,000
***fp2*:**	補間軸2的絕對位置設定值() 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

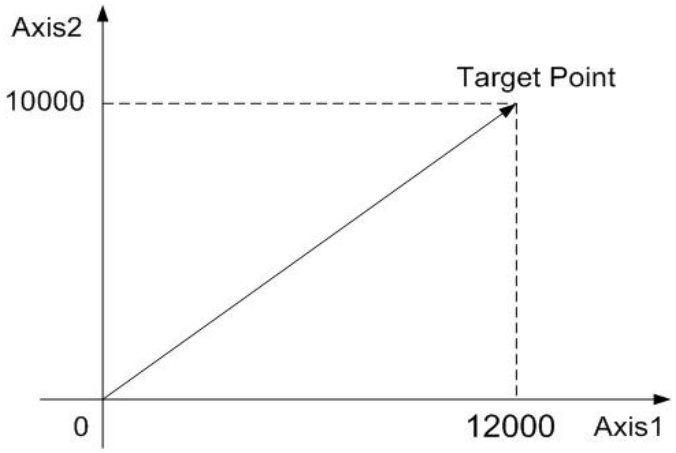
0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_ABS_LINE_2D is 0x0AF7。
RSM_ABS_MACRO_LINE_2D is 0x0CFX。

Modbus 範例:

```
RSM_ABS_LINE_2D (hRsm, 1, 12000, 10000);
//Moves from the origin to the absolute position (12000, 10000)
```



兩軸線性補間運動

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 05	0A
Register[]	**Value (hex)**	**Remarks**			
0	0A FX/0C FX	*Sub_function code*			
1	00 00	*MSW of fp1*			
2	2E E0	*LSW of fp1 (12000 = 0x2EE0)*			
3	00 00	*MSW of fp2*			
4	27 10	*LSW of fp2 (10000 = 0x2710)*			

7.2.12 三軸相對距離線性補間移動

eRTU RSM_LINE_3D (HANDLE *hRsm*, BYTE *SlaveAddr*, long *fp1*, long *fp2*, long *fp3*)

※Δ eRTU RSM_MACRO_LINE_3D (HANDLE *hRsm*, BYTE *SlaveAddr*, long *fp1*, long *fp2*, long *fp3*)

說明:

開始執行一個三軸相對距離線性補間移動。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***fp1*:**	補間軸1的移動距離 範圍: -2,000,000,000 ~ +2,000,000,000
***fp2*:**	補間軸2的移動距離 範圍: -2,000,000,000 ~ +2,000,000,000
***fp3*:**	補間軸3的移動距離 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

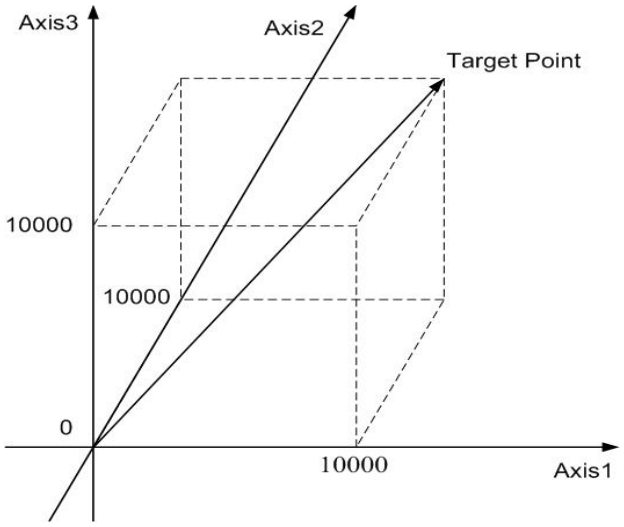
備註:

RSM_LINE_3D 的Sub_function代碼為 0x0A 64。

RSM_MACRO_LINE_3D 的 Sub_function 代碼為 0x0C 64。

Modbus 範例:

```
RSM_LINE_3D (hRsm, 1, 10000, 10000, 10000);
//execute the 3-axis linear interpolation motion on module 1.
```



三軸線性補間運動

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 07	0E
Register[]	**Value (hex)**	**Remarks**			
0	0A 64/0C 64	*Sub_function code*			
1	00 00	*MSW of fp1*			
2	27 10	*LSW of fp1 (10000 = 0x2710)*			
3	00 00	*MSW of fp2*			
4	27 10	*LSW of fp2 (10000 = 0x2710)*			
5	00 00	*MSW of fp3*			
6	27 10	*LSW of fp3 (10000 = 0x2710)*			

7.2.13 三軸絕對位置線性補間移動

eRTU RSM_ABS_LINE_3D (HANDLE *hRsm*, BYTE *SlaveAddr*, long *fp1*, long *fp2*, long *fp3*)

※Δ eRTU RSM_ABS_MACRO_LINE_3D (HANDLE *hRsm*, BYTE *SlaveAddr*, long *fp1*, long *fp2*, long *fp3*)

說明:

開始執行一個三軸絕對位置線性補間移動。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***fp1*:**	補間軸1的絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***fp2*:**	補間軸2的絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***fp3*:**	補間軸3的絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

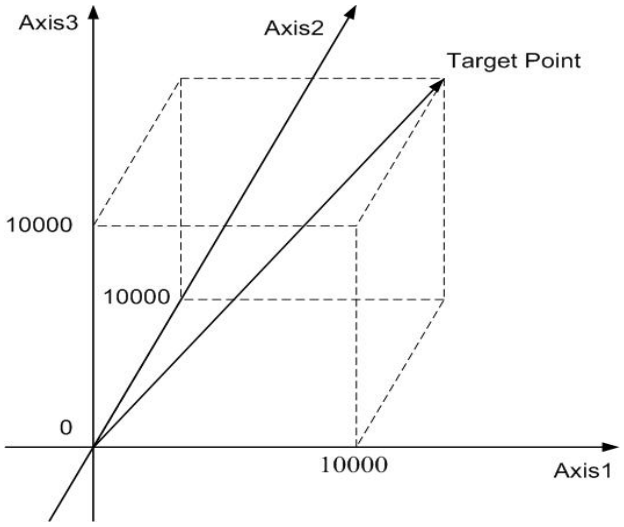
備註:

RSM_LINE_3D的Sub_function代碼為0x0AF8。

RSM_MACRO_LINE_3D 的 Sub_function 代碼為 0x0CF8。

Modbus 範例:

```
RSM_ABS_LINE_3D (hRsm, 1, 10000, 10000, 10000);
//Move to the specified target position.
```



三軸線性補間運動

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 07	0E
Register[]	**Value (hex)**	**Remarks**			
0	0A F8/0C F8	*Sub_function code*			
1	00 00	*MSW of fp1*			
2	27 10	*LSW of fp1 (10000 = 0x2710)*			
3	00 00	*MSW of fp2*			
4	27 10	*LSW of fp2 (10000 = 0x2710)*			
5	00 00	*MSW of fp3*			
6	27 10	*LSW of fp3 (10000 = 0x2710)*			

7.2.14 兩軸相對距離圓弧補間運動(順時針方向圓弧)

eRTU RSM_ARC_CW (HANDLE *hRsm*, BYTE *SlaveAddr*, long *cp1*, long *cp2*, long *fp1*, long *fp2*)

※Δ eRTU RSM_MACRO_ARC_CW (HANDLE *hRsm*, BYTE *SlaveAddr*, long *cp1*, long *cp2*, long *fp1*, long *fp2*)

說明:

開始執行一個兩軸相對距離圓弧補間移動(順時針方向圓弧)。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***cp1*:**	補間軸1的圓心相對距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***cp2*:**	補間軸2的圓心相對距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***fp1*:**	補間軸1的移動相對距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***fp2*:**	補間軸2的移動相對距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

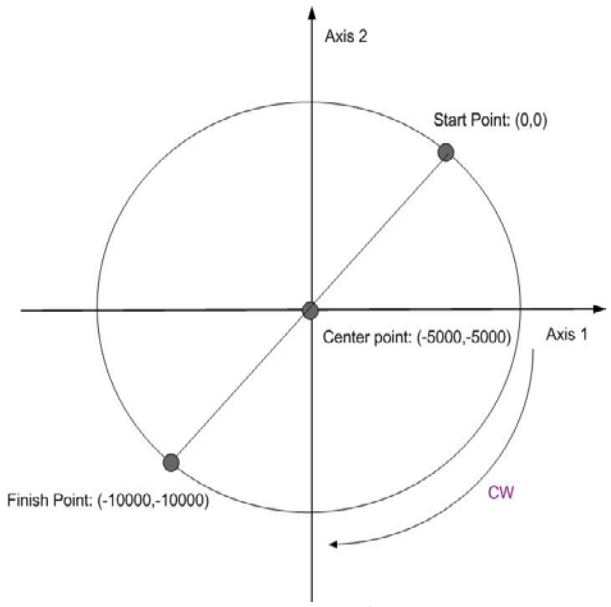
備註:

RSM_ARC_CW 的Sub_function代碼為 0x0A 65。

RSM_MACRO_ARC_CW 的 Sub_function 代碼為 0x0C 65。

Modbus 範例:

```
RSM_ARC_CW (hRsm, 1, -5000, -5000, -10000, -10000);
// Issues a command to perform a circular motion (an arc)
// in a CW direction. Please refer to the following figure.
```



兩軸圓弧補間(順時針方向)

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 09	12
Register[]	**Value (hex)**	**Remarks**			
0	0A 65/0C 65	*Sub_function code*			
1	FF FF	*MSW of cp1*			
2	EC 78	*LSW of cp1 (-5000 = 0xFFFFEC78)*			
3	FF FF	*MSW of cp2*			
4	EC 78	*LSW of cp2 (-5000 = 0xFFFFEC78)*			
5	FF FF	*MSW of fp1*			
6	D8 F0	*LSW of fp1 (-10000 = 0xFFFFD8F0)*			
7	FF FF	*MSW of fp2*			
8	D8 F0	*LSW of fp2 (-10000 = 0xFFFFD8F0)*			

7.2.15 兩軸相對距離圓弧補間運動(逆時針方向圓弧)

eRTU RSM_ARC_CCW (HANDLE *hRsm*, BYTE *SlaveAddr*, long *cp1*, long *cp2*, long *fp1*, long *fp2*)

※Δ eRTU RSM_MACRO_ARC_CCW (HANDLE *hRsm*, BYTE *SlaveAddr*, long *cp1*, long *cp2*, long *fp1*, long *fp2*)

說明:

開始執行一個兩軸相對距離圓弧補間移動(逆時針方向圓弧)。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***cp1*:**	補間軸1的圓心相對距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***cp2*:**	補間軸2的圓心相對距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***fp1*:**	補間軸1的移動相對距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***fp2*:**	補間軸2的移動相對距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

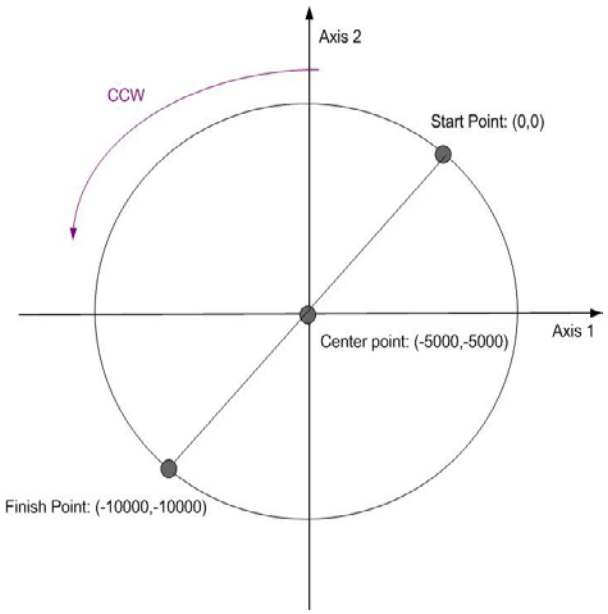
備註:

RSM_ARC_CCW 的Sub_function代碼為 0x0A 67。

RSM_MACRO_ARC_CCW 的 Sub_function 代碼為 0x0C 67。

Modbus 範例:

```
RSM_ARC_CCW (hRsm, 1, -5000, -5000, -10000, -10000);
// Issues a command to perform a circular motion (an arc)
// in a CCW direction. Please refer to the following figure.
```



兩軸圓弧補間(逆時針方向)

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 09	12
Register[]	**Value (hex)**	**Remarks**			
0	0A 67/0C 67	*Sub_function code*			
1	FF FF	*MSW of cp1*			
2	EC 78	*LSW of cp1 (-5000 = 0xFFFFEC78)*			
3	FF FF	*MSW of cp2*			
4	EC 78	*LSW of cp2 (-5000 = 0xFFFFEC78)*			
5	FF FF	*MSW of fp1*			
6	D8 F0	*LSW of fp1 (-10000 = 0xFFFFD8F0)*			
7	FF FF	*MSW of fp2*			
8	D8 F0	*LSW of fp2 (-10000 = 0xFFFFD8F0)*			

7.2.16 兩軸絕對位置圓弧補間運動(順時針方向圓弧)

eRTU RSM_ABS_ARC_CW (HANDLE *hRsm*, BYTE *SlaveAddr*, long *cp1*, long *cp2*, long *fp1*, long *fp2*)

※Δ eRTU RSM_ABS_MACRO_ARC_CW (HANDLE *hRsm*, BYTE *SlaveAddr*, long *cp1*, long *cp2*, long *fp1*, long *fp2*)

說明:

開始執行一個兩軸絕對位置圓弧補間移動(順時針方向圓弧)。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***cp1*:**	補間軸1的圓心絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***cp2*:**	補間軸2的圓心絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***fp1*:**	補間軸1的移動絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***fp2*:**	補間軸2的移動絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

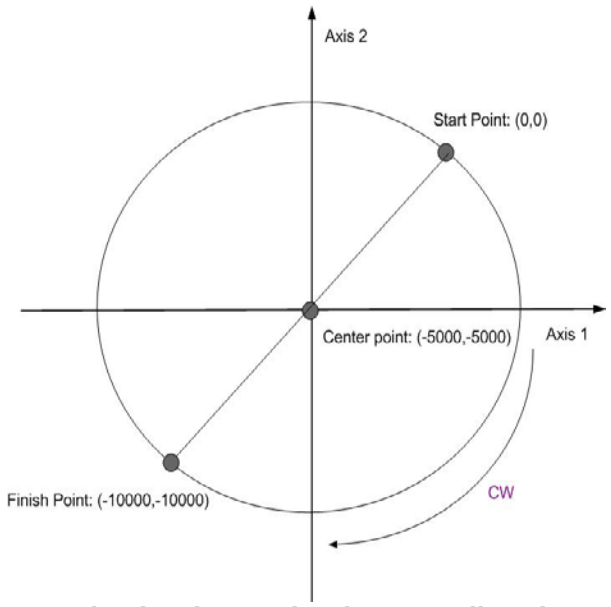
備註:

RSM_ABS_ARC_CW的Sub_function代碼為 0x0AF9。

RSM_ABS_MACRO_ARC_CW 的 Sub_function 代碼為 0x0CF9。

Modbus 範例:

```
RSM_ABS_ARC_CW (hRsm, 1, -5000, -5000, -10000, -10000);
// Issues a command to perform a circular motion (an arc)
// in a CW direction. Please refer to the following figure.
```



2-axis circular motion in a CW direction

所傳送的 **Modbus** 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 09	12
Register[]	**Value (hex)**	**Remarks**			
0	0A F9/0C F9	*Sub_function code*			
1	FF FF	*MSW of cp1*			
2	EC 78	*LSW of cp1 (-5000 = 0xFFFFEC78)*			
3	FF FF	*MSW of cp2*			
4	EC 78	*LSW of cp2 (-5000 = 0xFFFFEC78)*			
5	FF FF	*MSW of fp1*			
6	D8 F0	*LSW of fp1 (-10000 = 0xFFFFD8F0)*			
7	FF FF	*MSW of fp2*			
8	D8 F0	*LSW of fp2 (-10000 = 0xFFFFD8F0)*			

7.2.17 兩軸絕對位置圓弧補間運動(逆時針方向圓弧)

eRTU RSM_ABS_ARC_CCW (HANDLE *hRsm*, BYTE *SlaveAddr*, long *cp1*, long *cp2*, long *fp1*, long *fp2*)

※Δ eRTU RSM_ABS _MACRO_ARC_CCW (HANDLE *hRsm*, BYTE *SlaveAddr*, long *cp1*, long *cp2*, long *fp1*, long *fp2*)

說明:

開始執行一個兩軸絕對位置圓弧補間移動(逆時針方向圓弧)。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***cp1*:**	補間軸1的圓心絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***cp2*:**	補間軸2的圓心絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***fp1*:**	補間軸1的移動絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***fp2*:**	補間軸2的移動絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

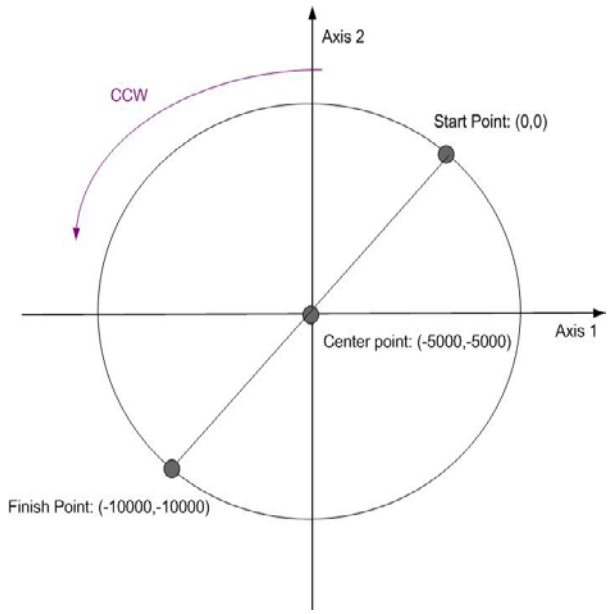
備註:

RSM_ABS_ARC_CCW的Sub_function代碼為 0x0A FA。

RSM_ABS_MACRO_ARC_CCW 的 Sub_function 代碼為 0x0C FA。

Modbus 範例:

```
RSM_ARC_CCW (hRsm, 1, -5000, -5000, -10000, -10000);
// Issues a command to perform a circular motion (an arc)
// in a CCW direction. Please refer to the following figure.
```



2-axis circular motion in a CW direction

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 09	12
Register[]	**Value (hex)**	**Remarks**			
0	0A FA/0C FA	*Sub_function code*			
1	FF FF	*MSW of cp1*			
2	EC 78	*LSW of cp1 (-5000 = 0xFFFFEC78)*			
3	FF FF	*MSW of cp2*			
4	EC 78	*LSW of cp2 (-5000 = 0xFFFFEC78)*			
5	FF FF	*MSW of fp1*			
6	D8 F0	*LSW of fp1 (-10000 = 0xFFFFD8F0)*			
7	FF FF	*MSW of fp2*			
8	D8 F0	*LSW of fp2 (-10000 = 0xFFFFD8F0)*			

7.2.18 兩軸圓弧補間運動(順時針方向完整的圓)

eRTU RSM_CIRCLE_CW (HANDLE *hRsm*, BYTE *SlaveAddr*, long *cp1*, long *cp2*)

※Δ eRTU RSM_MACRO_CIRCLE_CW (HANDLE *hRsm*, BYTE *SlaveAddr*, long *cp1*, long *cp2*)

說明:

開始執行一個兩軸圓弧補間移動(順時針方向完整的圓)。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***cp1*:**	補間軸1的圓心相對距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***cp2*:**	補間軸2的圓心相對距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

備註:

RSM_CIRCLE_CW 的Sub_function代碼為 0x0A 69。

RSM_MACRO_CIRCLE_CW 的 Sub_function 代碼為 0x0C 69。

Modbus 範例:

```
RSM_CIRCLE_CW (hRsm, 1, 0, 10000);
// execute a circular motion (a complete circle) in a CW direction on module 1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 05	0A
Register[]	**Value (hex)**	**Remarks**			
0	0A 69/0C 69	*Sub_function code*			
1	00 00	*MSW of cp1*			
2	00 00	*LSW of cp1 (0 = 0x0)*			
3	00 00	*MSW of cp2*			
4	27 10	*LSW of cp2 (10000 = 0x2710)*			

7.2.19 兩軸圓弧補間運動(逆時針方向完整的圓)

eRTU RSM_CIRCLE_CCW (HANDLE *hRsm*, BYTE *SlaveAddr*, long *cp1*, long *cp2*)

※Δ eRTU RSM_MACRO_CIRCLE_CCW (HANDLE *hRsm*, BYTE *SlaveAddr*, long *cp1*, long *cp2*)

說明:

開始執行一個兩軸圓弧補間移動(逆時針方向完整的圓)。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***cp1*:**	補間軸1的圓心相對距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***cp2*:**	補間軸2的圓心相對距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

備註:

RSM_CIRCLE_CCW 的Sub_function代碼為 0x0A 6A。

RSM_MACRO_CIRCLE_CCW 的 Sub_function 代碼為 0x0C 6A。

Modbus 範例:

```
RSM_CIRCLE_CCW (hRsm, 1, 0, 10000);
// execute a circular motion (a complete circle) in a CCW direction on module 1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 05	0A
Register[]	**Value (hex)**	**Remarks**			
0	0A 6A/0C 6A	*Sub_function code*			
1	00 00	*MSW of cp1*			
2	00 00	*LSW of cp1 (0 = 0x0)*			
3	00 00	*MSW of cp2*			
4	27 10	*LSW of cp2 (10000 = 0x2710)*			

7.3 同步運動

7.3.1 設定同步運動

eRTU RSM_SYNC_ACTION (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis1*, BYTE *axis2*, DWORD *nSYNC*, BYTE *nDRV*, BYTE *nLATCH*, BYTE *nPRESET*, BYTE *nOUT*, BYTE *nINT*, BYTE *isrNoX*, BYTE *isrNoY*, BYTE *isrNoZ*, BYTE *isrNoU*)

※Δ eRTU RSM_MACRO_SYNC_ACTION (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis1*, BYTE *axis2*, DWORD *nSYNC*, BYTE *nDRV*, BYTE *nLATCH*, BYTE *nPRESET*, BYTE *nOUT*, BYTE *nINT*, BYTE *isrNoX*, BYTE *isrNoY*, BYTE *isrNoZ*, BYTE *isrNoU*)

說明:

設定同步運動的的相關參數值。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis1*:**	同步運動觸發軸(axis1)的軸號設定值，將依照此軸狀態觸發同動事件。 1=X軸; 2=Y軸; 4=Z軸; 8=U軸
***axis2*:**	被觸發運動的同動軸(axis2)軸號設定值 請參照表4設定(可設定複數軸)
nSYNC:	啟動因子設定值，請參照下方說明進行設定
nDRV:	同動軸的運動設定值，請參閱下方說明進行設定
nLATCH:	同動軸的位置栓鎖(LATCH)設定值。當啟動因子條件滿足時記錄同動軸的當前位置，請參閱下方說明
nPRESET:	同動軸所使用的參數選擇設定值，請參閱下方說明
nOUT:	設定觸發訊號輸出，請參閱下方說明
nINT:	設定開啟中斷觸發ISR巨集程序功能，請參閱下方說明
isrNoX:	X軸中斷所要觸發的ISR巨集程序編號設定 範圍: ISR1 ~ ISR20
isrNoY:	Y軸中斷所要觸發的ISR巨集程序編號設定(ISR1 ~ ISR20)
isrNoZ:	Z軸中斷所要觸發的ISR巨集程序編號設定(ISR1 ~ ISR20)
isrNoU:	U軸中斷所要觸發的ISR巨集程序編號設定(ISR1 ~ ISR20)

***nSYNC*:**

啟動因子設定值，即觸發軸(axis1)觸發同動事件的條件。可同時設定多個啟動因子，例如當數值等於0x00000041表示同時設定P≥C+(0x00000001)與IN3↑(0x00000040)。

數值	事件	說明
0x00000000		關閉啟動因子
0x00000001	P≥C+	當觸發軸的邏輯/編碼器位置計數器數值大於或等於比較器 COMP+ 設定值。 請使用函式"RSM_SET_COMPARE ()"設定比較器所要比較的計數器(邏輯位置或編碼器位置)。
0x00000002	P<C+	當觸發軸的邏輯/編碼器位置計數器數值小於比較器 COMP+設定值。 請使用函式"RSM_SET_COMPARE ()"設定比較器所要比較的計數器(邏輯位置或編碼器位置)。
0x00000004	P<C-	當觸發軸的邏輯/編碼器位置計數器數值小於比較器 COMP-設定值。 請使用函式"RSM_SET_COMPARE ()"設定比較器所要比較的計數器(邏輯位置或編碼器位置)。
0x00000008	P≥C-	當觸發軸的邏輯/編碼器位置計數器數值大於或等於比較器 COMP-設定值。 請使用函式"RSM_SET_COMPARE ()"設定比較器所要比較的計數器(邏輯位置或編碼器位置)。
0x00000010	D-STA	當觸發軸的移動開始。
0x00000020	D-END	當觸發軸的移動完成。
0x00000040	IN3↑	當觸發軸的輸入訊號 IN3 邏輯準位由低轉高(上緣觸發)。
0x00000080	IN3↓	當觸發軸的輸入訊號 IN3 邏輯準位由高轉低(下緣觸發)。

***nDRV*:**

同步運動設定值。當啟動因子條件成立時同動軸(axis2)的運動方式。

數值	事件	說明
0		關閉同步運動功能
1	FDRV+	同動軸開始正方向的相對距離移動。請設定參數 *nPRESET*=3 並且使用函式"RSM_SET_PRESET()"設定移動的距離。當同動軸正在移動的狀態下此命令將視為無效。
2	FDRV-	同動軸開始反方向的相對距離移動。請設定參數 *nPRESET*=3 並且使用函式"RSM_SET_PRESET()"設定移動的距離。當同動軸正在移動的狀態下此命令將視為無效。
3	CDRV+	同動軸開始正方向的持續移動。當同動軸正在移動的狀態下此命令將視為無效。
4	CDRV-	同動軸開始反方向的持續移動。當同動軸正在移動的狀態下此命令將視為無效。
5	SSTOP	同動軸立即減速後停止。
6	ISTOP	同動軸立即停止。

***nLATCH*:**

選擇位置栓鎖的計數器，當啟動因子條件滿足時記錄同動軸的當前位置。請使用函式"RSM_GET_LATCH()"取得栓鎖的位置數值。

數值	事件	說明
0		關閉位置栓鎖功能。
1	LPSAV	記錄同動軸的邏輯位置(LP)計數器數值。
2	EPSAV	記錄同動軸的編碼器位置(EP)計數器數值。

***nPRESET*:**

設定啟動因子條件滿足時同動軸所使用的參數選擇設定值。

數值	符號	說明
0		關閉選擇設定功能
1	LPSET	選擇設定當啟動因子條件滿足時更新同動軸的邏輯位置計數器數值。請使用函式"RSM_SET_PRESET()"設定預定更改的新數值。
2	EPSET	選擇設定當啟動因子條件滿足時更新同動軸的編碼器位置計數器數值。請使用函式"RSM_SET_PRESET()"設定預定更改的新數值。
3	OPSET	選擇設定同動軸被觸發運動時的移動距離，請使用函式"RSM_SET_PRESET()"設定預定移動的距離數值。 當同動軸被觸發作連續運動時此設定值無效。
4	VLSET	選擇設定同動軸被觸發運動時的移動速度，請使用函式"RSM_SET_PRESET()"設定預定移動的速度數值。 當同動軸被觸發作連續運動時此設定值無效。

***nOUT*:**

設定觸發訊號輸出，當啟動因子條件滿足時輸出DO訊號。

數值	符號	說明
0		關閉輸出觸發訊號。
1	OUT	開啟觸發訊號輸出功能，請使用函式"RSM_SET_OUT()"設定觸發訊號的狀態。

***nINT*:**

設定開啟中斷觸發ISR巨集程序功能。當啟動因子條件滿足時將發出中斷觸發執行ISR巨集程序。請使用函式"RSM_ENABLE_INT()"開啟中斷，並使用參數*isrNoX* 、*isrNoY* 、*isrNoZ* 、*isrNoU*指定觸發的ISR編號。

數值	符號	說明
0		關閉功能。
1	INT	開啟功能。

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SYNC_ACTION 的Sub_function代碼為 0x0A 6E。

RSM_MACRO_SYNC_ACTION 的 Sub_function 代碼為 0x0C 6E。

Modbus 範例:

```
RSM_SYNC_ACTION (hRsm, SlaveAddr, AXIS_U, AXIS_U, 0X00000040, 0, 2, 4, 0, 0,
    0, 0, 0, 0);
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 0E	1C
Register[]	**Value (hex)**	**Remarks**			
0	0A 6E/0C 6E	*Sub_function code*			
1	00 08	*axis1 (8 = AXIS_U)*			
2	00 08	*axis2*			
3	00 00	*MSW of nSYNC*			
4	00 40	*LSW of nSYNC (0x40 ➔ rising edge of IN3 will trigger the action)*			
5	00 00	*nDRV (0 ➔ no driving control)*			
6	00 02	*nLATCH (2 ➔ will latch EP value)*			
7	00 04	*nPRESET (4 ➔ set new velocity when SYNC condition is true)*			
8	00 00	*nOUT (0 ➔ no pulse out)*			
9	00 00	*nINT (0 ➔ disable)*			
10	00 00	*isrNoX (It can be any value since nINT is disable.)*			
11	00 00	*isrNoY (It can be any value since nINT is disable.)*			
12	00 00	*isrNoZ (It can be any value since nINT is disable.)*			
13	00 00	*isrNoU (It can be any value since nINT is disable.)*			

相關範例:

```
//Example1: When the U axis received IN3 positive edge trigger signal, it
//will change the LATCH encoder value.
RSM_SYNC_ACTION(hRsm, SlaveAddr, AXIS_U, AXIS_U, 0X00000040, 0, 2, 4, 0, 0,
0, 0, 0, 0);
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_U, 5000);
//Set the maximum speed of u axis on module 1 to 5K PPS.
RSM_NORMAL_SPEED(hRsm, SlaveAddr, AXIS_U, 0);
// set the speed profile for u axis as a symmetric T-curve.
RSM_SET_V(hRsm, SlaveAddr, AXIS_U, 2000);
```

```
// set the speed of u axis on module 1 to 2000 PPS
RSM_SET_A(hRsm, SlaveAddr, AXIS_U, 100000);
// set the acceleration value of u axis to 100K PPS/sec.
RSM_SET_SV(hRsm,SlaveAddr, AXIS_U, 100);
//set the start velocity of u axis to  100 PPS
RSM_FIXED_MOVE(hRsm, SlaveAddr, AXIS_U, 10000);
//move 10000 Pulses for u axis on module 1
RSM_SET_PRESET(h SlaveAddr, AXIS_U, 100);
// set the new speed of u axis on module 1 to 100 PPS
RSM_STOP_WAIT(hRsm, SlaveAddr, AXIS_U);
long Vsb = RSM_GET_LATCH(hRsm, SlaveAddr, AXIS_U);

//Example2: When the EP value of u axis exceeds COMP+(5000), it will start
//the Y-axis moving 2,000 PPS.
RSM_SYNC_ACTION(hRsm, SlaveAddr, AXIS_U, AXIS_Y, 0X00000001, 1, 0, 3, 0, 0,
0, 0, 0, 0);
RSM_SET_COMPARE(hRsm, SlaveAddr, AXIS_U, 0, 1, 5000);
//Set the value of COMP+ is 5000, source reference the EP of U axis.
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_YU, 9000);
// Set the maximum speed of YU axes on module 1 to 9K PPS.
RSM_NORMAL_SPEED(hRsm, SlaveAddr, AXIS_YU, 0);
// set the speed profile for YU axes as symmetric T-curve.
RSM_SET_V(hRsm, SlaveAddr, AXIS_YU, 3000);
// set the speed of YU axes on module 1 to 3000 PPS.
RSM_SET_A(hRsm, SlaveAddr, AXIS_YU, 200000);
// set the acceleration value of YU axes to 200K PPS/sec.
RSM_SET_SV(hRsm, SlaveAddr, AXIS_YU, 200);
// set the start velocity of yu axes to 200 PPS
RSM_FIXED_MOVE(hRsm, SlaveAddr, AXIS_U, 10000);
// move 10000 Pulses for U axis on module 1
RSM_SET_PRESET(hRsm, SlaveAddr, AXIS_Y, 2000);
// Set the Y axis moving 2000 PPS。


//Example3: When the LP value of X axis exceeds COMP+(200), it will
//start the nout output 5V of Y axis.
RSM_SYNC_ACTION(hRsm, SlaveAddr, AXIS_X, AXIS_Y, 0X00000001, 0, 0, 0, 1, 0,
0,      0, 0, 0);
RSM_SET_COMPARE(hRsm, SlaveAddr, AXIS_X, 0, 0, 200);
//Set the value of COMP+ is 200, source reference the LP of X axis.
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_X, 5000);
// Set the maximum speed of X axis on module 1 to 5K PPS.
RSM_NORMAL_SPEED(hRsm, SlaveAddr, AXIS_X, 0);
// set the speed profile for X axis as a symmetric T-curve.
RSM_SET_V(hRsm, SlaveAddr, AXIS_X, 2000);
// set the speed of X axis on module 1 to 2000 PPS.
```

```
RSM_SET_A(hRsm, SlaveAddr, AXIS_X, 100000);
// set the acceleration value of X axis to 100K PPS/s.
RSM_SET_SV(hRsm, SlaveAddr, AXIS_X, 100);
// set the start velocity of X axis to 100 PPS
RSM_FIXED_MOVE(hRsm, SlaveAddr, AXIS_X, 500);
// move 500 Pulses for X axis on module 1
RSM_SET_OUT(hRsm, SlaveAddr, AXIS_Y, 1, 0);
// Set the Y axis high level output 5V.


//Example4: When the LP value of X axis exceeds COMP+(5000), it will emergency
stop and generate an interrupt to call ISR1.
RSM_SYNC_ACTION(hRsm, SlaveAddr, AXIS_X, AXIS_X, 0X00000001, 6, 0, 0, 0, 1,
    ISR1, 0, 0, 0);
RSM_ENABLE_INT(hRsm, SlaveAddr);  //Enable interrupt
RSM_SET_COMPARE(hRsm, SlaveAddr, AXIS_X, 0, 0, 5000);
// Set the value of COMP+ is 5000, source reference the LP of X axis.
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XY, 5000);
// Set the maximum speed of XY axes on module 1 to 5K PPS.
RSM_NORMAL_SPEED(hRsm, SlaveAddr, AXIS_XY, 0);
// set the speed profile for XY axes as symmetric T-curve.
RSM_SET_V(hRsm, SlaveAddr, AXIS_XY, 2000);
// set the speed of XY axes on module 1 to 2000 PPS.
RSM_SET_A(hRsm, SlaveAddr, AXIS_XY, 100000);
// set the acceleration value of XY axes to 100K PPS/sec.
RSM_SET_SV(hRsm, SlaveAddr, AXIS_XY, 100);
// set the start velocity of XY axes          to 100 PPS.
RSM_FIXED_MOVE(hRsm, SlaveAddr, AXIS_X, 10000);
// move 10K Pulses for X axis on module 1


ISR1:
RSM_MP_ISR_CREATE(hRsm, SlaveAddr, ISR1); // Create ISR1
RSM_MACRO_FIXED_MOVE(hRsm, SlaveAddr, AXIS_Y, 1000);
//move 1000 Pulses          for y axis on module 1
RSM_MACRO_MP_ISR_CLOSE(hRsm, SlaveAddr); // End ISR1

//Example5: Use the for loop to iterate, when the LP value of X axis
//exceeds COMP+(38000), it will generate an interrupt to call ISR1, and
// determine whether receive IN3 signal in the ISR1, if receive then
// emergency stop.
RSM_MP_CREATE(hRsm, SlaveAddr, MP59); //Create a macro program MP59.
RSM_MACRO_SET_MAX_V(hRsm, SlaveAddr, AXIS_XY, 5000);
// Set the maximum speed of XY axes on module 1 to 5K PPS.
RSM_MACRO_NORMAL_SPEED(hRsm, SlaveAddr, AXIS_XY, 0);
// set the speed profile for XY axes as symmetric T-curve.
```

```
RSM_MACRO_SET_V(hRsm, SlaveAddr, AXIS_XY, 2000);
// set the speed of XY axeson module 1 to 2000 PPS.
RSM_ MACRO_SET_A(hRsm, SlaveAddr, AXIS_XY, 100000);
// set the acceleration value of XY axes to 100K PPS/S.
RSM_ MACRO_SET_SV(hRsm, SlaveAddr, AXIS_XY, 100);
// set the start velocity of XY axes to 100 PPS.
RSM_MACRO_FOR(hRsm, SlaveAddr, 100); //Set the times of for loop is 100.
RSM_ MACRO_SET_LP(hRsm, SlaveAddr, AXIS_XY, 0);
//Set the LP value of XYaxes to 0
RSM_MACRO_SET_FILTER(hRsm, SlaveAddr, AXIS_X, 16, 1); //Set the filter of IN3
RSM_MACRO_CLEAR_SYNC_ACTION(hRsm, SlaveAddr, AXIS_X);
//Clear all the       sync_action conditions of X axis.
RSM_MACRO_SYNC_ACTION(hRsm, SlaveAddr, AXIS_X, AXIS_X, 0X00000001, 0,
0,      0, 0, 1, ISR1, 0, 0, 0);
// Set the LP value of X axis exceeds COMP+, it will generate an interrupt to
// call ISR1.
RSM_MACRO_ENABLE_INT(hRsm, SlaveAddr);  //Enable interrupt
RSM_MACRO_SET_COMPARE(hRsm, SlaveAddr, AXIS_X, 0, 0, 38000);
//Set the value of COMP+ is 38000, source reference the LP of X axis.
RSM_MACRO_FIXED_MOVE(hRsm, SlaveAddr, AXIS_X, 42000);
// move 42K Pulses for X axis on module 1
RSM_MACRO_STOP_WAIT(hRsm, SlaveAddr, AXIS_X); //Wait until X axis stop.
RSM_MACRO_NEXT(hRsm, SlaveAddr); //Match with for loop.
RSM_MACRO_MP_CLOSE(hRsm, SlaveAddr); // End a macro program MP59


ISR1:
RSM_MP_ISR_CREATE(hRsm, SlaveAddr, ISR1); //Create ISR1
RSM_MACRO_FIXED_MOVE(hRsm, SlaveAddr, AXIS_Y, 1000);
// move 1000 Pulses for Y axis on module 1.
RSM_MACRO_SYNC_ACTION(hRsm, SlaveAddr, AXIS_X, AXIS_X, 0X00000040, 6,
0,      0, 0, 0, 0, 0, 0, 0); //Set the X axis to receive the IN3 positive edge trigger
//signals from low level to high level, then emergency stop.
RSM_MACRO_MP_ISR_CLOSE(hRsm, SlaveAddr); // End ISR1
```

7.3.2 清除同步運動設定值

eRTU RSM_CLEAR_SYNC_ACTION (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *triggerAxis*)

※Δ eRTU RSM_MACRO_CLEAR_SYNC_ACTION (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *triggerAxis*)

說明:

清除所有觸發軸的同步運動設定值。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
triggerAxis	同步運動觸發軸(axis1)的軸號設定值。請參照函式"RSM_SYNC_ACTION()" 1=X軸; 2=Y軸; 4=Z軸; 8=U軸

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_CLEAR_SYNC_ACTION的Sub_function代碼為 0x0A 6A。
RSM_MACRO_CLEAR_SYNC_ACTION 的 Sub_function 代碼為 0x0C 6A。

Modbus 範例:

```
RSM_CLEAR_SYNC_ACTION (hRsm, 1, AXIS_X);
// Clear all the sync_action conditions of X axis.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A 95/0C 95	*Sub_function code*			
1	00 01	*TriggerAxis*			

7.3.3 設定同步因子

eRTU RSM_SET_ACTIVATION_FACTORS(HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *triggerAxis*, DWORD *nSYNC*)

※Δ eRTU RSM_MACRO_SET_ACTIVATION_FACTORS (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *triggerAxis*, DWORD *nSYNC*)

說明:

設定同步運動的觸發因子。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
triggerAxis	**同步運動觸發軸(axis1)的軸號設定值，將依照此軸狀態觸發同動事件。 1=X軸; 2=Y軸; 4=Z軸; 8=U軸**
***nSYNC*:**	**啟動因子設定值，即觸發軸(axis1)觸發同動事件的條件。可同時設定多個啟動因子，例如當數值等於 0x00000041 表示同時設定 P ≥ C+(0x00000001) 與 IN3↑(0x00000040)。請參照下方說明進行設定**

數值	事件	說明
0x00000000		關閉啟動因子
0x00000001	P≥C+	當觸發軸的邏輯/編碼器位置計數器數值大於或等於比較器 COMP+ 設定值。 請使用函式"RSM_SET_COMPARE ()"設定比較器所要比較的計數器(邏輯位置或編碼器位置)。
0x00000002	P<C+	當觸發軸的邏輯/編碼器位置計數器數值小於比較器 COMP+設定值。 請使用函式"RSM_SET_COMPARE ()"設定比較器所要比較的計數器(邏輯位置或編碼器位置)。
0x00000004	P<C-	當觸發軸的邏輯/編碼器位置計數器數值小於比較器 COMP-設定值。 請使用函式"RSM_SET_COMPARE ()"設定比較器所要比較的計數器(邏輯位置或編碼器位置)。
0x00000008	P≥C-	當觸發軸的邏輯/編碼器位置計數器數值大於或等於比較器 COMP-設定值。 請使用函式"RSM_SET_COMPARE ()"設定比較器所要比較的計數器(邏輯位置或編碼器位置)。

0x00000010	D-STA	當觸發軸的移動開始。
0x00000020	D-END	當觸發軸的移動完成。
0x00000040	IN3↑	當觸發軸的輸入訊號 IN3 邏輯準位由低轉高(上緣觸發)。
0x00000080	IN3↓	當觸發軸的輸入訊號 IN3 邏輯準位由高轉低(下緣觸發)。

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_ACTIVATION_FACTORS的Sub_function代碼為 0x0A 6A。

RSM_MACRO_SET_ACTIVATION_FACTORS 的 Sub_function 代碼為 0x0C 6A。

Modbus 範例:

RSM_SET_ACTIVATION_FACTORS (hRsm, 1, AXIS_X, 0X00000001);

// Set the synchronization conditions of x axis is P≥C+.

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 96/0C 96	*Sub_function code*			
1	00 01	*TriggerAxis*			
2	00 00	*MSW of nSYNC*			
3	00 01	*LSW of nSYNC*			

7.3.4 設定同步軸號

eRTU RSM_SET_ACTIVATION_AXIS(HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *triggerAxis*, BYTE *activationAxis*)

※Δ eRTU RSM_MACRO_SET_ACTIVATION_AXIS (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *triggerAxis*, BYTE *activationAxis*)

說明:

設定同動軸的軸號。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
triggerAxis	同步運動觸發軸(axis1)的軸號設定值，將依照此軸狀態觸發同動事件。 1=X軸; 2=Y軸; 4=Z軸; 8=U軸
activationAxis:	被觸發運動的同動軸(axis2)軸號設定值 請參照表4設定(可設定複數軸)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_ACTIVATION_AXIS 的Sub_function代碼為 0x0A 97。
RSM_MACRO_SET_ACTIVATION_AXIS 的 Sub_function 代碼為 0x0C 97。

Modbus 範例:

```
RSM_SET_ACTIVATION_AXIS (hRsm, 1, AXIS_X, AXIS_Y);
// Set the synchronization axis of x axis is y axis.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0A 97/0C 97	*Sub_function code*			
1	00 01	*TriggerAxis(AXIS_X)*			
2	00 02	*ActivationAxis(AXIS_Y)*			

7.3.5 設定同步動作

eRTU RSM_SET_ACTION(HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *activationAxis*, BYTE *nDRV*, BYTE *nLATCH*, BYTE *nPRESET*, BYTE *nOUT*, BYTE *nINT*, BYTE *isrNoX*, BYTE *isrNoY*, BYTE *isrNoZ*, BYTE *isrNoU*)

※Δ eRTU RSM_MACRO_SET_ACTION (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *activationAxis*, BYTE *nDRV*, BYTE *nLATCH*, BYTE *nPRESET*, BYTE *nOUT*, BYTE *nINT*, BYTE *isrNoX*, BYTE *isrNoY*, BYTE *isrNoZ*, BYTE *isrNoU*)

說明:

設定同動因子條件滿足時所觸發的動作。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***activationAxis*:**	被觸發運動的同動軸(axis2)軸號設定值 請參照表4設定(可設定複數軸)
nDRV:	同動軸的運動設定值，請參閱下方說明進行設定
nLATCH:	同動軸的位置栓鎖(LATCH)設定值。當啟動因子條件滿足時記錄同動軸的當前位置，請參閱下方說明
nPRESET:	同動軸所使用的參數選擇設定值，請參閱下方說明
nOUT:	設定觸發訊號輸出，請參閱下方說明
nINT:	設定開啟中斷觸發ISR巨集程序功能，請參閱下方說明
isrNoX:	X軸中斷所要觸發的ISR巨集程序編號設定 範圍: ISR1 ~ ISR20
isrNoY:	Y軸中斷所要觸發的ISR巨集程序編號設定(ISR1 ~ ISR20)
isrNoZ:	Z軸中斷所要觸發的ISR巨集程序編號設定(ISR1 ~ ISR20)
isrNoU:	U軸中斷所要觸發的ISR巨集程序編號設定(ISR1 ~ ISR20)

***nDRV*:**

同步運動設定值。當啟動因子條件成立時同動軸(axis2)的運動方式。

數值	事件	說明
0		關閉同步運動功能
1	FDRV+	同動軸開始正方向的相對距離移動。請設定參數 *nPRESET*=3 並且使用函式"RSM_SET_PRESET()"設定移動的距離。當同動軸正在移動的狀態下此命令將視為無效。
2	FDRV-	同動軸開始反方向的相對距離移動。請設定參數 *nPRESET*=3 並且使用函式"RSM_SET_PRESET()"設定移動的距離。當同動軸正在移動的狀態下此命令將視為無效。
3	CDRV+	同動軸開始正方向的持續移動。當同動軸正在移動的狀態下此命令將視為無效。
4	CDRV-	同動軸開始反方向的持續移動。當同動軸正在移動的狀態下此命令將視為無效。
5	SSTOP	同動軸立即減速後停止。
6	ISTOP	同動軸立即停止。

***nLATCH*:**

選擇位置栓鎖的計數器，當啟動因子條件滿足時記錄同動軸的當前位置。請使用函式"RSM_GET_LATCH()"取得栓鎖的位置數值。

數值	事件	說明
0		關閉位置栓鎖功能。
1	LPSAV	記錄同動軸的邏輯位置(LP)計數器數值。
2	EPSAV	記錄同動軸的編碼器位置(EP)計數器數值。

***nPRESET*:**

設定啟動因子條件滿足時同動軸所使用的參數選擇設定值。

數值	符號	說明
0		關閉選擇設定功能
1	LPSET	選擇設定當啟動因子條件滿足時更新同動軸的邏輯位置計數器數值。請使用函式"RSM_SET_PRESET()"設定預定更改的新數值。
2	EPSET	選擇設定當啟動因子條件滿足時更新同動軸的編碼器位置計數器數值。請使用函式"RSM_SET_PRESET()"設定預定更改的新數值。
3	OPSET	選擇設定同動軸被觸發運動時的移動距離，請使用函式"RSM_SET_PRESET()"設定預定移動的距離數值。 當同動軸被觸發作連續運動時此設定值無效。
4	VLSET	選擇設定同動軸被觸發運動時的移動速度，請使用函式"RSM_SET_PRESET()"設定預定移動的速度數值。 當同動軸被觸發作連續運動時此設定值無效。

***nOUT*:**

設定觸發訊號輸出，當啟動因子條件滿足時輸出DO訊號。

數值	符號	說明
0		關閉輸出觸發訊號。
1	OUT	開啟觸發訊號輸出功能，請使用函式”RSM_SET_OUT()”設定觸發訊號的狀態。

***nINT*:**

設定開啟中斷觸發ISR巨集程序功能。當啟動因子條件滿足時將發出中斷觸發執行ISR巨集程序。請使用函式”RSM_ENABLE_INT()”開啟中斷，並使用參數*isrNoX* 、*isrNoY* 、*isrNoZ* 、*isrNoU*指定觸發的ISR編號。

數值	符號	說明
0		關閉功能。
1	INT	開啟功能。

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_ACTION 的Sub_function代碼為 0x0A 98。

RSM_MACRO_SET_ACTION 的 Sub_function 代碼為 0x0C 98。

Modbus 範例:

```
RSM_SET_ACTION (hRsm, 1, AXIS_Y, 0, 2, 4, 0, 0, 0, 0, 0, 0);
// Set the action of synchronization motion is when the IN3 positive
//edge trigger signals received, it will change the velocity and the encoder of LATCH.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 0B	16
Register[]	**Value (hex)**	**Remarks**			
0	0A 98/0C 98	*Sub_function code*			
1	00 02	*ActivationAxis(AXIS_Y)*			
2	00 00	*nDRV (0 ➔ no driving control)*			
3	00 02	*nLATCH (2 ➔ will latch EP value)*			
4	00 04	*nPRESET (4 ➔ set new velocity when SYNC condition is true)*			
5	00 00	*nOUT (0 ➔ no pulse out)*			
6	00 00	*nINT (0 ➔ disable)*			
7	00 00	*isrNoX (It can be any value since nINT is disable.)*			
8	00 00	*isrNoY (It can be any value since nINT is disable.)*			
9	00 00	*isrNoZ (It can be any value since nINT is disable.)*			
10	00 00	*isrNoU (It can be any value since nINT is disable.)*			

相關範例:

```
//Example1: Set two groups synchronization motion, when the LP value of x
//axis exceeds COMP+(30000), the synchronization motion axis z-axis
//generate an interrupt to call ISR1, and when the LP value of y axis
//exceeds COMP+(35000), the synchronization motion axis u-axis generate
//an interrupt to call ISR2.
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU, 5000);
// Set the maximum speed of xyzu axes on module 1 to 5K PPS.
RSM_NORMAL_SPEED(hRsm, SlaveAddr, AXIS_XYZU, 0);
// set the speed profile for xyzu axes as symmetric T-curve.
RSM_SET_V(hRsm, SlaveAddr, AXIS_XYZU, 2000);
// set the speed of xyzu axes on module 1 to 2000 PPS.
RSM_SET_A(hRsm, SlaveAddr, AXIS_XYZU, 100000);
// set the acceleration value of xyzu axes to 100K PPS/sec.
RSM_SET_SV(hRsm, SlaveAddr, AXIS_XYZU, 100);
// set the start velocity of xyzu axes to 100 PPS.
RSM_SET_ACTIVATION_FACTORS(hRsm, SlaveAddr, AXIS_XY, 1);
// Set the synchronization conditions of xy axes is P≥C+.
RSM_SET_ACTIVATION_AXIS(hRsm, 1, AXIS_X, AXIS_Z);
// Set the synchronization axis of x axis is z axis.
RSM_SET_ACTIVATION_AXIS(hRsm, 1, AXIS_Y, AXIS_U);
```

```
// Set the synchronization axis of y axis is u axis.
RSM_SET_ACTION (hRsm, 1, AXIS_Z, 0, 0, 0, 0, 1, 0, 0, ISR1, 0);
// Set the action of synchronization motion axis z-axis is to generate an
// interrupt to call ISR1.
RSM_SET_ACTION (hRsm, 1, AXIS_U, 0, 0, 0, 0, 1, 0, 0, 0, ISR2);
// Set the action of synchronization motion axis u-axis is to generate an
// interrupt to call ISR2.
RSM_ENABLE_INT(hRsm, SlaveAddr);  //Enable interrupt
RSM_SET_COMPARE (hRsm, SlaveAddr, AXIS_X, 0, 0, 30000);
//Set the value of COMP+ is 30000, source reference the LP of x axis.
RSM_SET_COMPARE (hRsm, SlaveAddr, AXIS_Y, 0, 0, 35000);
//Set the value of COMP+ is 35000, source reference the LP of y axis.
RSM_FIXED_MOVE(hRsm, SlaveAddr, AXIS_XY, 42000);
// move 42K Pulses for xy axes on module 1

ISR1:
RSM_MP_ISR_CREATE(hRsm, SlaveAddr, ISR1); // Create ISR1
RSM_MACRO_FIXED_MOVE(hRsm, SlaveAddr, AXIS_Z, 10000);
// move 10K Pulses for z axis on module 1
RSM_MACRO_MP_ISR_CLOSE(hRsm, SlaveAddr); // End ISR1

ISR2:
RSM_MP_ISR_CREATE(hRsm, SlaveAddr, ISR2); // Create ISR2
RSM_MACRO_FIXED_MOVE(hRsm, SlaveAddr, AXIS_U, 10000);
// move 10K Pulses for u axis on module 1
RSM_MACRO_MP_ISR_CLOSE(hRsm, SlaveAddr); // End ISR2
```

7.3.6 位置比較器(COMPARE)設定

eRTU RSM_SET_COMPARE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nSELECT*, BYTE *nTYPE*, long *data*)

※Δ eRTU RSM_MACRO_SET_COMPARE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nSELECT*, BYTE *nTYPE*, long *data*)

說明:

設定比較器數值。請注意，使用此功能將導致軟體極限功能失效。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***nSELECT*:**	選擇所要設定的比較器號碼 0 → C+ 1 → C-
***nTYPE*:**	比較器所參照的位置計數器 0 → 邏輯位置(LP)計數器 1 → 編碼器位置(EP)計數器
***data*:**	比較值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_COMPARE 的Sub_function代碼為 0x0A 70。
RSM_MACRO_SET_COMPARE 的 Sub_function 代碼為 0x0C 70。

Modbus 範例:

```
RSM_SET_COMPARE (hRsm, SlaveAddr, AXIS_U, 0, 1, 5000);
//Set the comparison function for U-Axis.
//Set the compared source to be EP; and the COMP+ value to 5000.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 06	0C
Register[]	**Value (hex)**	**Remarks**			
0	0A 70/0C 70	*Sub_function code*			
1	00 08	*Axis (8 ➔ AXIS_U)*			
2	00 00	*nSELECT*			
3	00 01	*nTYPE*			
4	00 00	*MSW of data*			
5	13 88	*LSW of data (5000 = 0x1388)*			

7.3.7 取得位置栓鎖(LATCH)數值

eRTU RSM_GET_LATCH (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long* *LatchValue*)

※Δ eRTU RSM_MACRO_GET_LATCH (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

eRTU RSM_GET_LATCH_4_AXIS (HANDLE *hRsm*, BYTE *SlaveAddr*, long* *LatchValueX*, long* *LatchValueY*, long* *LatchValueZ*, long* *LatchValueU*)

說明:

取得位置栓鎖的數值(由同動功能觸發記錄)。

類別:

Modbus table, Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***LatchValue*:**	位置栓鎖的數值 範圍: -2,000,000,000 ~ +2,000,000,000
LatchValueX:	X軸位置栓鎖的數值 範圍: -2,000,000,000 ~ +2,000,000,000
LatchValueY:	Y軸位置栓鎖的數值 範圍: -2,000,000,000 ~ +2,000,000,000
LatchValueZ:	Z軸位置栓鎖的數值 範圍: -2,000,000,000 ~ +2,000,000,000
LatchValueU:	U軸位置栓鎖的數值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

■ **方法 1: 直接讀取位置栓鎖數值**

位置栓鎖的資料型態為長整數(long)，因此操作時請設定 MSW 做為起始位置。

STOP status	Address	Remarks
X_LATCH_MSW	62 (0x3E)	MSW of X_LATCH
X_LATCH_LSW	63 (0x3F)	LSW of X_LATCH
Y_LATCH_MSW	64 (0x40)	MSW of Y_LATCH
Y_LATCH_LSW	65 (0x41)	LSW of Y_LATCH
Z_LATCH_MSW	66 (0x42)	MSW of Z_LATCH
Z_LATCH_LSW	67 (0x43)	LSW of Z_LATCH
U_LATCH_MSW	68 (0x44)	MSW of U_LATCH
U_LATCH_LSW	69 (0x45)	LSW of U_LATCH

```
long LATCH_Y;
RSM_GET_LATCH (hRsm, 1, AXIS_Y, &LATCH_Y);
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	04	00 40	00 02

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	*MSW of Y_LATCH* (Reg_0)	*LSW of Y_LATCH* (Reg_1)
01	04	04	00 00	03 E8

```
LATCH_Y = Register[0];
LATCH_Y = (long) (((LATCH_Y << 16) & 0xffff0000) | (Register[1] & 0xffff) );
```

■ 方法 2: 用於巨集程序(MP)指令

使用此方法時目前的位置栓鎖值不會馬上回傳，只有在 MP 被呼叫執行時才會回傳數值。使用者可以使用 FC = 16 去寫這個指令到 MP。在 MP 中於 RSM_MACRO_GET_LATCH 後面可以使用 RSM_MACRO_SET_RVAR()這個指令去儲存回傳的位置栓鎖值到一個指定的變數。此變數可以被其他函數使用，請參照其他 MP 函數的用法說明。

```
RSM_MACRO_GET_LATCH (hRsm, 1, AXIS_Y);
//Get the latched value which is from Y-axis of card 1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0C 71	*Sub_function code*			
1	00 02	*axis*			

7.3.8 預設同動功能的運動參數

eRTU RSM_SET_PRESET (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long *data*)

※Δ eRTU RSM_MACRO_SET_PRESET (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, long *data*)

說明:

預設同動功能的同動軸運動參數。請注意，各個同動軸無法獨立設定。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
***axis*:**	**被觸發運動的同動軸(axis2)軸號設定值 請參照表4設定(可設定複數軸)**
***data*:**	**LP: (-2,000,000,000 ~ +2,000,000,000) EP: (-2,000,000,000 ~ +2,000,000,000) P : (-2,000,000,000 ~ +2,000,000,000) V : 請參照函式"RSM_SET_MAX_V()"進行設定 若觸發軸為複數軸，請使用函式"RSM_SET_MAX_V()"設定各軸為相同設定值**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_PRESET 的Sub_function代碼為 0x0A 72。

RSM_MACRO_SET_PRESET 的 Sub_function 代碼為 0x0C 72。

Modbus 範例:

假設同動功能設定為觸發改變速度，並且執行下述的設定，如此，當同動因子條件滿足時 AXIS_U 的速度將改變為 100 PPS。

```
RSM_SET_PRESET (hRsm, SlaveAddr, AXIS_U, 100);
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 72/0C 72	*Sub_function code*			
1	00 08	*Axis (8 ➔ AXIS_U)*			
2	00 00	*MSW of data*			
3	00 64	*LSW of data (100 = 0x64)*			

7.3.9 同動功能的輸出訊號設定

eRTU RSM_SET_OUT (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *outEdge*, BYTE *PulseWidth*)

※Δ eRTU RSM_MACRO_SET_OUT (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *outEdge*, BYTE *PulseWidth*)

說明:

設定同動功能的輸出訊號。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 1(X軸)或2(Y軸)，目前只支援X軸與Y軸
***outEdge*:**	輸出觸發邏輯: 0=低準位觸發；1=高準位觸發
***PulseWidth*:**	輸出脈波寬度 0 = 10 uSec 1 = 20 uSec 2 = 100 uSec 3 = 200 uSec 4 = 1,000 uSec 5 = 2,000 uSec 6 = 10,000 uSec 7 = 20,000 uSec

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_SET_OUT 的Sub_function代碼為 0x0A 73。

RSM_MACRO_SET_OUT 的 Sub_function 代碼為 0x0C 73。

Modbus 範例:

假設同動功能設定為觸發輸出訊號，並且執行下述的設定，如此，當同動因子條件滿足時將依照設定的條件輸出訊號。

```
RSM_SET_OUT (hRsm, 1, AXIS_U, 1, 0);
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 73/0C 73	*Sub_function code*			
1	00 08	*Axis (8 ➔ AXIS_U)*			
2	00 01	*outEdge*			
3	00 00	*PulseWidth*			

7.3.10 開啟 i-8094H 的中斷功能

eRTU RSM_ENABLE_INT (HANDLE *hRsm*, BYTE *SlaveAddr*)

※ eRTU RSM_MACRO_ENABLE_INT (HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

開啟i-8094H內運動控制晶片的中斷功能。

類別:

Modbus sub_function; RTC, MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_ENABLE_INT 的Sub_function代碼為 0x0A AA。
RSM_MACRO_ENABLE_INT 的 Sub_function 代碼為 0x0C AA。

Modbus 範例:

```
RSM_ENABLE_INT (hRsm, 1);
//Enable the interrupt function for module 1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	0A AA/0C AA	*Sub_function code*			

7.3.11 關閉 i-8094H 的中斷功能

eRTU RSM_DISABLE_INT (HANDLE *hRsm*, BYTE *SlaveAddr*)

※ eRTU RSM_MACRO_DISABLE_INT (HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

關閉i-8094H內運動控制晶片的中斷功能。

類別:

Modbus sub_function; RTC, MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_DISABLE_INT 的Sub_function代碼為 0x0A AB。

RSM_MACRO_DISABLE_INT 的 Sub_function 代碼為 0x0C AB。

Modbus 範例:

```
RSM_DISABLE_INT (hRsm, 1);
//Disable the interrupt function for module 1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	0A AB/0C AB	*Sub_function code*			

7.3.12 設定 i-8094H 的中斷因子

eRTU RSM_INTFACTOR_ENABLE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nINT*, BYTE *isrNo*)

※Δ eRTU RSM_MACRO_INTFACTOR_ENABLE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nINT*, BYTE *isrNo*)

說明:

設定i-8094H內運動控制晶片中斷功能的中斷因子。

類別:

Modbus sub_function; RTC, MP, ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)
***nINT*:**	中斷因子，請參照下方說明
***isrNo*:**	中斷所要觸發的ISR巨集程序編號設定 範圍: ISR1 ~ ISR20

中斷因子設定值說明

nINT	符號	說明
0	PULSE	控制脈波上升時
1	P>=C-	邏輯/編碼器位置計數器的數值大於或等於比較器 COMP-的設定值 請使用函式"RSM_SET_COMPARE ()"設定比較器所要比較的計數器(邏輯位置或編碼器位置)。
2	P<C-	邏輯/編碼器位置計數器的數值小於比較器 COMP-的設定值 請使用函式"RSM_SET_COMPARE ()"設定比較器所要比較的計數器(邏輯位置或編碼器位置)。
3	P<C+	邏輯/編碼器位置計數器的數值小於比較器 COMP+的設定值 請使用函式"RSM_SET_COMPARE ()"設定比較器所要比較的計數器(邏輯位置或編碼器位置)。
4	P>=C+	邏輯/編碼器位置計數器的數值小於比較器 COMP+的設定值 請使用函式"RSM_SET_COMPARE ()"設定比較器所要比較的計數器(邏輯位置或編碼器位置)。
5	C-END	在驅動過程中，等速度區段結束的時候。
6	C-STA	在驅動過程中，等速度區段開始的時候。
7	D-END	驅動結束(移動完成) 。

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_INTFACTOR_ENABLE 的Sub_function代碼為 0x0A AC。

RSM_MACRO_INTFACTOR_ENABLE 的 Sub_function 代碼為 0x0C AC。

Modbus 範例:

使用此功能設定中斷因子的條件滿足時，所要觸發執行的 ISR 巨集程序。

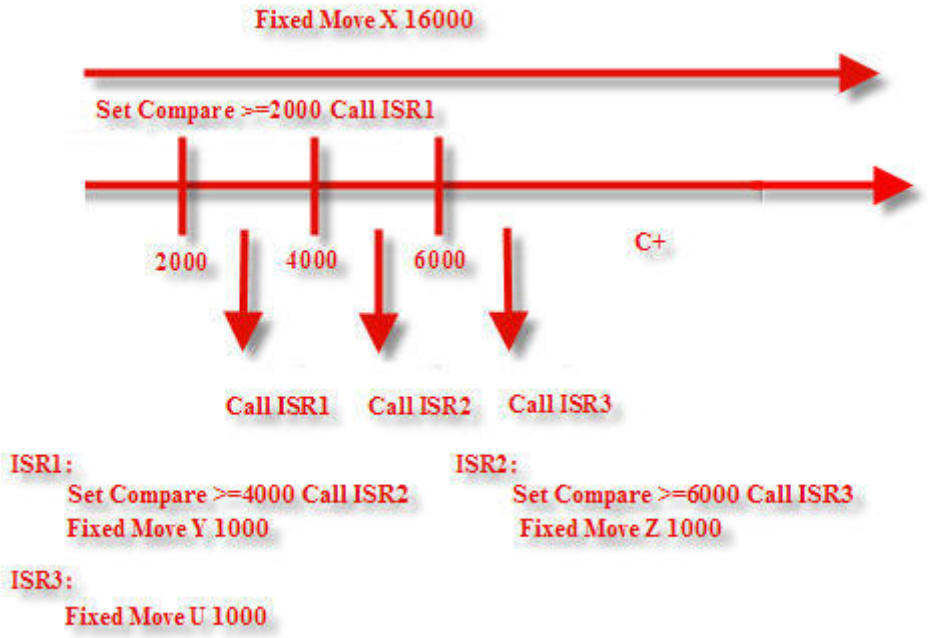
```
RSM_INTFACTOR_ENABLE (hRsm, 1, AXIS_X, 4, ISR1);
//When the position counter of x axis exceeds C+, then call ISR1.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A AC/0C AC	*Sub_function code*			
1	00 01	*axis (1 ➔ AXIS_X)*			
2	00 04	*nINT*			
3	00 01	*isrNo (ISR1 = 1)*			

相關範例:

```
//Example1: Using call the ISR to set multi-group compare.
```



```
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU, 5000);
// Set the maximum          speed of xyzu axes on module 1 to 5K PPS.
RSM_NORMAL_SPEED(hRsm, SlaveAddr, AXIS_XYZU, 0);
// set the speed profile for xyzu axes as symmetric T-curve.
RSM_SET_V(hRsm, SlaveAddr, AXIS_XYZU, 2000);
// set the speed of xyzu axes on module 1 to 2000 PPS.
RSM_SET_A(hRsm, SlaveAddr, AXIS_XYZU, 100000);
// set the acceleration value of xyzu axes to 100K PPS/S.
RSM_SET_SV(hRsm, SlaveAddr, AXIS_XYZU, 100);
// set the start velocity of xyzu axes to 100 PPS.
RSM_SET_COMPARE(hRsm, SlaveAddr, AXIS_X, 0, 0, 2000);
//Set the value of COMP+ is 2000, source reference the LP of x axis.
RSM_INTFACTOR_ENABLE (hRsm, 1, AXIS_X, 4, ISR1);
//When the LP value of x axis exceeds C+(2000), then call ISR1.
RSM_ENABLE_INT (hRsm, 1);  //Enable interrupt
RSM_FIXED_MOVE(hRsm, SlaveAddr, AXIS_X, 16000);
// move 16K Pulses for x axis on module 1

ISR1:
RSM_MP_ISR_CREATE(hRsm, SlaveAddr, ISR1); //Create ISR1
RSM_MACRO_SET_COMPARE(hRsm, SlaveAddr, AXIS_X, 0, 0, 4000);
//Set the value of COMP+ is 4000, source reference the LP of x axis.
RSM_MACRO_INTFACTOR_ENABLE (hRsm, 1, AXIS_X, 4, ISR2);
//When the LP value of x axis exceeds C+(4000), then call ISR2.
RSM_MACRO_FIXED_MOVE(hRsm, SlaveAddr, AXIS_Y, 1000);
// move 1000 Pulses          for y axis on module 1
RSM_MACRO_MP_ISR_CLOSE(hRsm, SlaveAddr); // End ISR1

ISR2:
RSM_MP_ISR_CREATE(hRsm, SlaveAddr, ISR2); //Create ISR2
RSM_MACRO_SET_COMPARE(hRsm, SlaveAddr, AXIS_X, 0, 0, 6000);
//Set the value of COMP+ is 6000, source reference the LP of x axis.
RSM_MACRO_INTFACTOR_ENABLE (hRsm, 1, AXIS_X, 4, ISR3);
//When the LP value of x axis exceeds C+(6000), then call ISR3.
RSM_MACRO_FIXED_MOVE(hRsm, SlaveAddr, AXIS_Z, 1000);
// move 1000 Pulses          for z axis on module 1
RSM_MACRO_MP_ISR_CLOSE(hRsm, SlaveAddr); // End ISR2

ISR3:
RSM_MP_ISR_CREATE(hRsm, SlaveAddr, ISR3); //Create ISR3
RSM_MACRO_FIXED_MOVE(hRsm, SlaveAddr, AXIS_U, 1000);
// move 1000 Pulses          for u axis on module 1
RSM_MACRO_MP_ISR_CLOSE(hRsm, SlaveAddr); // End ISR3
```

7.3.13 清除 i-8094H 的中斷因子

eRTU RSM_INTFACTOR_DISABLE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nINT*)

※Δ eRTU RSM_MACRO_INTFACTOR_DISABLE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *nINT*)

說明:

清除i-8094H內運動控制晶片中斷功能的中斷因子。

類別:

Modbus sub_function; RTC, MP, ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表4)
nINT:	所要清除的中斷因子設定值。關於中斷因子的觸發條件請參照函式"RSM_INTFACTOR_ENABLE()"說明

設定值	說明
0	清除中斷因子 PULSE
1	清除中斷因子 P>=C-
2	清除中斷因子 P<C-
3	清除中斷因子 P<C+
4	清除中斷因子 P>=C+
5	清除中斷因子 C-END
6	清除中斷因子 C-STA
7	清除中斷因子 D-END
10	清除所有中斷因子

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_INTFACTOR_DISABLE 的Sub_function代碼為 0x0A AD。
RSM_MACRO_INTFACTOR_DISABLE 的 Sub_function 代碼為 0x0C AD。

Modbus 範例:

```
RSM_INTFACTOR_DISABLE (hRsm, 1, AXIS_XYZU, 4);
// Disable the interrupt factors of Module 1
// 4 : Interrupt occurs when the value of logic / real position counter
// is larger than or equal to that of COMP+ register.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0A AD/0C AD	*Sub_function code*			
1	00 0F	*axis (0xF ➔ AXIS_XYZU)*			
2	00 04	*nINT*			

相關範例:

```
RSM_INTFACTOR_DISABLE(hRsm, 1, AXIS_XYZU, 1);
RSM_INTFACTOR_DISABLE(hRsm, 1, AXIS_XYZU, 2);
RSM_INTFACTOR_DISABLE(hRsm, 1, AXIS_XYZU, 3);
RSM_INTFACTOR_DISABLE(hRsm, 1, AXIS_XYZU, 4);
// Disable the interrupt factors of Module 1
```

7.4 連續補間運動

7.4.1 兩軸矩形路徑補間

eRTU RSM_RECTANGLE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis1*, BYTE *axis2*, BYTE *nAcc*, BYTE *Sp*, BYTE *nDir*, long *Lp*, long *Wp*, long *Rp*, DWORD *RSV*, DWORD *RV*, DWORD *RA*, DWORD *RD*)

※ eRTU RSM_MACRO_RECTANGLE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis1*, BYTE *axis2*, BYTE *nAcc*, BYTE *Sp*, BYTE *nDir*, long *Lp*, long *Wp*, long *Rp*, DWORD *RSV*, DWORD *RV*, DWORD *RA*, DWORD *RD*)

說明:

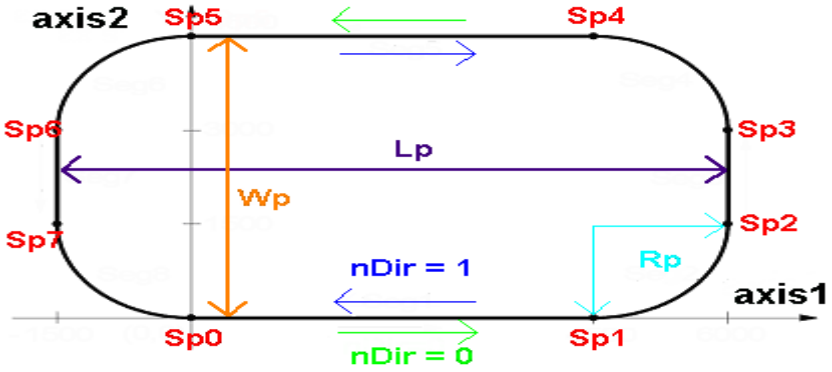
此函式將使用連續補間功能自動產生一個矩形路徑運動。

類別:

Modbus sub_function; RTC and MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis1*:**	矩形路徑第一軸的軸號設定值 1=X 軸; 2=Y 軸; 4=Z 軸; 8=U 軸
***axis2*:**	矩形路徑第二軸的軸號設定值 1=X軸; 2=Y軸; 4=Z軸; 8=U軸
***nAcc*:**	速度模式設定值 0 = 固定向量速度補間模式; 1 = 對稱T-curve 速度補間模式
***Sp*:**	起始位置的編號設定值 範圍: 0~7(Sp0 ~ Sp7 的相對位置請參照下方圖示說明)
***nDir*:**	運動方向設定值 0=CCW; 1=CW
***Lp*:**	第一軸的運動距離設定值 範圍: 1 ~ 2,000,000,00
***Wp*:**	第二軸的運動距離設定值 範圍: 1 ~ 2,000,000,00
***Rp*:**	轉角處的所有軸運動距離設定值 範圍: 1 ~ 2,000,000,00
***RSV*:**	初始速度的設定值 ，單位PPS
***RV*:**	向量速度的設定值 ，單位PPS
***RA*:**	加速度的設定值 ，單位 PPS/Sec
***RD*:**	減速度的設定值 (最末段運動)，單位 PPS/Sec



參數說明圖示

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節 2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

備註:

RSM_RECTANGLE 的Sub_function代碼為 0x0A 78。

RSM_MACRO_RECTANGLE 的 Sub_function 代碼為 0x0C 78。

Modbus 範例:

RSM_RECTANGLE (hRsm, 1, AXIS_X, AXIS_Y, 1, 0, 0, 20000, 10000, 1000, 1000, 10000, 5000, 5000);

//execute a rectangular motion on XY plane, will auto-calculate deceleration point.

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 14	28
Register[]	**Value (hex)**	**Remarks**			
0	0A 78/0C 78	*Sub_funciton code*			
1	00 01	*axis1 (1 ➔ AXIS_X)*			
2	00 02	*axis2 (2 ➔ AXIS_Y)*			
3	00 01	*nAcc (velocity profile)*			
4	00 00	*Sp (choose the start point from 0~7)*			
5	00 00	*nDir (set the direction to be CCW)*			
6	00 00	*MSW of Lp (length of the rectangle)*			
7	4E 20	*LSW of Lp (20000 = 0x4E20)*			
8	00 00	*MSW of Wp (width of the rectangle)*			
9	27 10	*LSW of Wp (10000 = 0x2710)*			
10	00 00	*MSW of Rp (set the corner radius value)*			
11	03 E8	*LSW of Rp (1000 = 0x3E8)*			
12	00 00	*MSW of RSV (define start velocity)*			

13	**03 E8**	***LSW of RSV (1000 = 0x3E8)***
14	**00 00**	***MSW of RV (define velocity)***
15	**27 10**	***LSW of RV (10000 = 0x2710)***
16	**00 00**	***MSW of RA (define acc value)***
17	**13 88**	***LSW of RA (5000 = 1388)***
18	**00 00**	***MSW of RD (define dec value)***
19	**13 88**	***LSW of RD (5000 = 1388)***

7.4.2 兩軸連續線性補間設定

eRTU RSM_LINE_2D_INITIAL (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis1*, BYTE *axis2*, DWORD *VSV* , DWORD *VV* , DWORD *VA*)

※ eRTU RSM_MACRO_LINE_2D_INITIAL (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis1*, BYTE *axis2*, DWORD *VSV* , DWORD *VV* , DWORD *VA*)

說明:

設定兩軸連續線性補間功能的軸號與速度參數。

類別:

Modbus sub_function; RTC and MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis1*:**	第一補間軸的軸號設定值 1=X 軸; 2=Y 軸; 4=Z 軸; 8=U 軸
***axis2*:**	第二補間軸的軸號設定值 1=X軸; 2=Y軸; 4=Z軸; 8=U軸
***VSV*:**	初始速度的設定值 ，單位PPS
***VV*:**	向量速度的設定值 ，單位PPS
***VA*:**	向量加速度的設定值 ，單位PPS/Sec

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

備註:

RSM_LINE_2D_INITIAL 的Sub_function代碼為 0x0A 7C。

RSM_MACRO_LINE_2D_INITIAL 的 Sub_function 代碼為 0x0C 7C。

Modbus 範例:

```
RSM_LINE_2D_INITIAL (hRsm, 1, AXIS_X, AXIS_Y, 300, 18000, 500000);
//This function should be defined before the RSM_LINE_2D_CONTINUE()
//function is used. Please refer to the example of this function.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 09	12
Register[]	**Value (hex)**	**Remarks**			
0	0A 7C/0C 7C	*Sub_funciton code*			
1	00 01	*axis1 (1 ➔ AXIS_X)*			
2	00 02	*axis2 (2 ➔ AXIS_Y)*			
3	00 00	*MSW of VSV*			
4	01 2C	*LSW of VSV (300 = 0x12C)*			
5	00 00	*MSW of VV*			
6	46 50	*LSW of VV (18000 = 0x4650)*			
7	00 07	*MSW of VA*			
8	A1 20	*LSW of VA (500000 = 0x0007A120)*			

7.4.3 執行兩軸連續線性補間(相對距離)

eRTU RSM_LINE_2D_CONTINUE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *nType*, long *fp1*, long *fp2*)

※ eRTU RSM_MACRO_LINE_2D_CONTINUE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *nType*, long *fp1*, long *fp2*)

說明:

執行一個兩軸連續線性補間運動(指定移動相對距離)。

類別:

Modbus sub_function; RTC and MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
nType:	指令類型設定值，連續補間功能會依照此設定值自動執行適當的速度曲線運動 0 = 非最後一個補間運動指令 1 = 最後一個補間運動指令
fp1:	補間軸1的移動距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000
fp2:	補間軸2的移動距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

備註:

RSM_LINE_2D_CONTINUE 的Sub_function代碼為 0x0A 7E。

RSM_MACRO_LINE_2D_CONTINUE 的 Sub_function 代碼為 0x0C 7E。

Modbus 範例:

```
RSM_LINE_2D_CONTINUE (hRsm, 1, 0, 100, 100);
//execute X, Y 2-axis linear continuous interpolation on module 1
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 06	0C
Register[]	**Value (hex)**	**Remarks**			
0	0A 7E/0C 7E	*Sub_funciton code*			
1	00 00	*nType*			
2	00 00	*MSW of fp1*			
3	00 64	*LSW of fp1 (100 = 0x64)*			
4	00 00	*MSW of fp2*			
5	00 64	*LSW of fp2*			

相關範例:

```
BYTE SlaveAddr=1;
unsigned short sv=300; //Set the vector start velocity is 300 PPS。
unsigned short v=18000; //Set the vector velocity is 18000 PPS。
unsigned long a=500000; //Set the vector acceleration is 500K PPS/s。
unsigned short loop1;
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU, 160000L);
// Set the maximum          speed of xyzu axes on module 1 to 160K PPS.
RSM_LINE_2D_INITIAL(hRsm, SlaveAddr, AXIS_X, AXIS_Y, sv, v, a);
for (loop1 = 0; loop1 < 10000; loop1++)
{
    RSM_LINE_2D_CONTINUE (hRsm, SlaveAddr, 0, 100, 100);
    RSM_LINE_2D_CONTINUE (hRsm, SlaveAddr, 0, -100, -100);
//execute X, Y 2-axis linear continuous interpolation on module 1
}
RSM_LINE_2D_CONTINUE (hRsm, SlaveAddr, 1, 100, 100);
//end X, Y 2-axis linear continuous interpolation on module 1
```

7.4.4 執行兩軸連續線性補間(絕對位置)

eRTU RSM_ABS_LINE_2D_CONTINUE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *nType*, long *fp1*, long *fp2*)

※ eRTU RSM_ABS_MACRO_LINE_2D_CONTINUE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *nType*, long *fp1*, long *fp2*)

說明:

執行一個兩軸連續線性補間運動(指定移動到絕對位置)。

類別:

Modbus sub_function; RTC and MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
nType:	指令類型設定值，連續補間功能會依照此設定值自動執行適當的速度曲線運動 0 = 非最後一個補間運動指令 1 = 最後一個補間運動指令
fp1:	補間軸1所要移動到的絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000
fp2:	補間軸2所要移動到的絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

備註:

RSM_ABS_LINE_2D_CONTINUE的Sub_function代碼為 0x0A FB。

RSM_ABS_MACRO_LINE_2D_CONTINUE 的 Sub_function 代碼為 0x0C FB。

Modbus 範例:

```
RSM_ABS_LINE_2D_CONTINUE (hRsm, 1, 0, 100, 100);
//execute X, Y 2-axis linear continuous interpolation on module 1
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 06	0C
Register[]	**Value (hex)**	**Remarks**			
0	0A FB/0C FB	*Sub_funciton code*			
1	00 00	*nType*			
2	00 00	*MSW of fp1*			
3	00 64	*LSW of fp1 (100 = 0x64)*			
4	00 00	*MSW of fp2*			
5	00 64	*LSW of fp2*			

相關範例:

```
BYTE SlaveAddr=1;
unsigned short sv=300; //Set the vector start velocity is 300 PPS。
unsigned short v=18000; //Set the vector velocity is 18000 PPS。
unsigned long a=500000; //Set the vector acceleration is 500K PPS/s。
unsigned short loop1;
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU, 160000L);
// Set the maximum          speed of xyzu axes on module 1 to 160K PPS.
RSM_LINE_2D_INITIAL(hRsm, SlaveAddr, AXIS_X, AXIS_Y, sv, v, a);
for (loop1 = 0; loop1 < 10000; loop1++)
{
    RSM_ABS_LINE_2D_CONTINUE (hRsm, SlaveAddr, 0, 100, 100);
    RSM_ABS_LINE_2D_CONTINUE (hRsm, SlaveAddr, 0, -100, -100);
    //execute X, Y 2-axis linear continuous interpolation on module 1
}
RSM_ABS_LINE_2D_CONTINUE (hRsm, SlaveAddr, 1, 100, 100);
//end X, Y 2-axis linear continuous interpolation on module 1
```

7.4.5 三軸連續線性補間設定

eRTU RSM_LINE_3D_INITIAL (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis1*, BYTE *axis2*, BYTE *axis3*, DWORD *VSV* , DWORD *VV* , DWORD *VA*)

※ eRTU RSM_MACRO_LINE_3D_INITIAL (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis1*, BYTE *axis2*, BYTE *axis3*, DWORD *VSV* , DWORD *VV* , DWORD *VA*)

說明:

設定三軸連續線性補間功能的軸號與速度參數。

類別:

Modbus sub_function; RTC and MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis1*:**	第一補間軸的軸號設定值 1=X 軸; 2=Y 軸; 4=Z 軸; 8=U 軸
***axis2*:**	第二補間軸的軸號設定值 1=X軸; 2=Y軸; 4=Z軸; 8=U軸
***axis3*:**	第三補間軸的軸號設定值 1=X軸; 2=Y軸; 4=Z軸; 8=U軸
***VSV*:**	初始速度，單位PPS
***VV*:**	向量速度，單位PPS
***VA*:**	向量加速度，單位PPS/Sec

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_LINE_3D_INITIAL 的Sub_function代碼為 0x0A 7F。
RSM_MACRO_LINE_3D_INITIAL 的 Sub_function 代碼為 0x0C 7F。

Modbus 範例:

```
RSM_LINE_3D_INITIAL (hRsm, 1, AXIS_X, AXIS_Y, AXIS_Z, 300, 18000, 500000);
//This function should be defined before the RSM_LINE_3D_CONTINUE()
//function is used. Please refer to the example of this function.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 0A	14
Register[]	**Value (hex)**	**Remarks**			
0	0A 7F/0C 7F	*Sub_funciton code*			
1	00 01	*axis1 (1 ➔ AXIS_X)*			
2	00 02	*axis2 (2 ➔ AXIS_Y)*			
3	00 04	*axis3 (4 ➔ AXIS_Z)*			
4	00 00	*MSW of VSV*			
5	01 2C	*LSW of VSV (300 = 0x12C)*			
6	00 00	*MSW of VV*			
7	46 50	*LSW of VV (18000 = 0x4650)*			
8	00 07	*MSW of VA*			
9	A1 20	*LSW of VA (500000 = 0x7A120)*			

7.4.6 執行三軸連續線性補間(相對距離)

eRTU RSM_LINE_3D_CONTINUE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *nType*, long *fp1*, long *fp2*, long *fp3*)

※ eRTU RSM_MACRO_LINE_3D_CONTINUE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *nType*, long *fp1*, long *fp2*, long *fp3*)

說明:

執行一個三軸連續線性補間運動(指定移動相對距離)。

類別:

Modbus sub_function; RTC and MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***nType*:**	指令類型設定值，連續補間功能會依照此設定值自動執行適當的速度曲線運動 0 = 非最後一個補間運動指令 1 = 最後一個補間運動指令
***fp1*:**	補間軸1的移動距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***fp2*:**	補間軸2的移動距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***fp3*:**	補間軸3的移動距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

備註:

RSM_LINE_3D_CONTINUE 的Sub_function代碼為 0x0A 81。

RSM_MACRO_LINE_3D_CONTINUE 的Sub_function代碼為 0x0C 81。

Modbus 範例:

```
RSM_LINE_3D_CONTINUE (hRsm, 1, 0, 100, 100, 100);
//execute X, Y, Z 3-axis linear continuous interpolation on module 1
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 08	10
Register[]	**Value (hex)**	**Remarks**			
0	0A 81/0C 81	*Sub_funciton code*			
1	00 00	*nType*			
2	00 00	*MSW of fp1*			
3	00 64	*LSW of fp1 (100 = 0x64)*			
4	00 00	*MSW of fp2*			
5	00 64	*LSW of fp2*			
6	00 00	*MSW of fp3*			
7	00 64	*LSW of fp3*			

相關範例:

```
BYTE SlaveAddr=1;
unsigned short sv=300; //Set the vector start velocity is 300 PPS。
unsigned short v=18000; //Set the vector velocity is 18000 PPS。
unsigned long a=500000; //Set the vector acceleration is 500K PPS/s。
unsigned short loop1;
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU, 160000L);
// Set the maximum speed of xyzu axes on module 1 to 160K PPS.
RSM_LINE_3D_INITIAL(hRsm, SlaveAddr, AXIS_X, AXIS_Y, AXIS_Z, sv, v, a);
for (loop1 = 0; loop1 < 10000; loop1++)
{
    RSM_LINE_3D_CONTINUE(hRsm, SlaveAddr, 0, 100, 100, 100);
    RSM_LINE_3D_CONTINUE(hRsm, SlaveAddr, 0, -100, -100, -100);
    //execute X, Y, Z 3-axis linear continuous interpolation on module 1
}
RSM_LINE_3D_CONTINUE (hRsm, 1, 0, 100, 100, 100);
//end X, Y, Z 3-axis linear continuous interpolation on module 1
```

7.4.7 執行三軸連續線性補間(絕對位置)

eRTU RSM_ABS_LINE_3D_CONTINUE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *nType*, long *fp1*, long *fp2*, long *fp3*)

※ eRTU RSM_ABS_MACRO_LINE_3D_CONTINUE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *nType*, long *fp1*, long *fp2*, long *fp3*)

說明:

執行一個三軸連續線性補間運動(指定移動到絕對位置)。

類別:

Modbus sub_function; RTC and MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***nType*:**	指令類型設定值，連續補間功能會依照此設定值自動執行適當的速度曲線運動 0 = 非最後一個補間運動指令 1 = 最後一個補間運動指令
***fp1*:**	補間軸1所要移動到的絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000)
***fp2*:**	補間軸2所要移動到的絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000
***fp3*:**	補間軸3所要移動到的絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

備註:

RSM_ABS_LINE_3D_CONTINUE的Sub_function代碼為 0x0A FC。

RSM_ABS_MACRO_LINE_3D_CONTINUE的Sub_function代碼為 0x0C FC。

Modbus 範例:

```
RSM_ABS_LINE_3D_CONTINUE (hRsm, 1, 0, 100, 100, 100);
//execute X, Y, Z 3-axis linear continuous interpolation on module 1
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 08	10
Register[]	**Value (hex)**	**Remarks**			
0	0A FC/0C FC	*Sub_funciton code*			
1	00 00	*nType*			
2	00 00	*MSW of fp1*			
3	00 64	*LSW of fp1 (100 = 0x64)*			
4	00 00	*MSW of fp2*			
5	00 64	*LSW of fp2*			
6	00 00	*MSW of fp3*			
7	00 64	*LSW of fp3*			

相關範例:

```
BYTE SlaveAddr=1;
unsigned short sv=300; //Set the vector start velocity is 300 PPS。
unsigned short v=18000; //Set the vector velocity is 18000 PPS。
unsigned long a=500000; //Set the vector acceleration is 500K PPS/s。
unsigned short loop1;
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU, 160000L);
// Set the maximum speed of xyzu axes on module 1 to 160K PPS.
RSM_LINE_3D_INITIAL(hRsm, SlaveAddr, AXIS_X, AXIS_Y, AXIS_Z, sv, v, a);
for (loop1 = 0; loop1 < 10000; loop1++)
{
    RSM_ABS_LINE_3D_CONTINUE(hRsm, SlaveAddr, 0, 100, 100, 100);
    RSM_ABS _LINE_3D_CONTINUE(hRsm, SlaveAddr, 0, -100, -100, -100);
    //execute X, Y, Z 3-axis linear continuous interpolation on module 1
}
RSM_LINE_3D_CONTINUE (hRsm, 1, 0, 100, 100, 100);
//end X, Y, Z 3-axis linear continuous interpolation on module 1
```

7.4.8 混合兩軸連續補間設定

eRTU RSM_MIX_2D_INITIAL (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis1*, BYTE *axis2*, BYTE *nAcc*, DWORD *VSV* , DWORD *VV* , DWORD *VA*)

※ eRTU RSM_MACRO_MIX_2D_INITIAL (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis1*, BYTE *axis2*, BYTE *nAcc*, DWORD *VSV* , DWORD *VV* , DWORD *VA*)

說明:

設定混合式兩軸連續補間功能的軸號與速度參數。

類別:

Modbus sub_function; RTC and MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis1*:**	第一補間軸的軸號設定值 1=X 軸; 2=Y 軸; 4=Z 軸; 8=U 軸
***axis2*:**	第二補間軸的軸號設定值 1=X軸; 2=Y軸; 4=Z軸; 8=U軸
***nAcc*:**	速度模式設定值 0 = 固定向量速度補間模式(VV) 1 = 對稱T-curve 速度補間模式(VSV, VV, VA)
***VSV*:**	初始速度設定值，單位 PPS
***VV*:**	向量速度設定值 ，單位 PPS
***VA*:**	向量加速度設定值，單位PPS/Sec

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_MIX_2D_INITIAL 的Sub_function代碼為 0x0A 82。

RSM_MACRO_MIX_2D_INITIAL 的 Sub_function 代碼為 0x0C 82。

Modbus 範例:

```
RSM_MIX_2D_INITIAL (hRsm, 1, 1, 2, 0, 300, 18000, 500000);
//This function should be defined before the RSM_MIX_2D_CONTINUE()
//function is used. Please refer to the example of this function.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 0A	14
Register[]	**Value (hex)**	**Remarks**			
0	0A 82/0C B2	*Sub_funciton code*			
1	00 01	*axis1 (1 ➔ AXIS_X)*			
2	00 02	*axis2 (2 ➔ AXIS_Y)*			
3	00 00	*nAcc*			
4	00 00	*MSW of VSV*			
5	01 2C	*LSW of VSV (300 = 0x12C)*			
6	00 00	*MSW of VV*			
7	46 50	*LSW of VV (18000 = 0x4650)*			
8	00 07	*MSW of VA*			
9	A1 20	*LSW of VA (500000 = 0x7A120)*			

7.4.9 執行混合兩軸連續補間(相對距離)

eRTU RSM_MIX_2D_CONTINUE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *nAcc*, BYTE *nType*, long *cp1*, long *cp2*, long *fp1*, long *fp2*)

※ eRTU RSM_MACRO_MIX_2D_CONTINUE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *nAcc*, BYTE *nType*, long *cp1*, long *cp2*, long *fp1*, long *fp2*)

說明:

執行一個混合式兩軸連續補間運動。此函式可選擇執行線性補間或是圓弧補間(指定移動相對距離)。

類別:

Modbus sub_function; RTC and MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***nAcc*:**	指令類型設定值，連續補間功能會依照此設定值自動執行適當的速度曲線運動 0 = 非最後一個補間運動指令 1 = 最後一個補間運動指令
***nType*:**	執行補間運動類型的設定值 設定值 / 補間類型 / 使用參數 1 / RSM_LINE_2D / *fp1, fp2* 2 / RSM_ARC_CW / *cp1, cp2, fp1, fp2* 3 / RSM_ARC_CCW / *cp1, cp2, fp1, fp2* 4 / RSM_CIRCLE_CW / *cp1, cp2* 5 / RSM_CIRCLE_CCW / *cp1, cp2*
***cp1*:**	補間軸1的圓心相對距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000)
***cp2*:**	補間軸2的圓心相對距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000)
***fp1*:**	補間軸1移動距離的設定值 範圍: -2,000,000,000 ~ +2,000,000,000)
***fp2*:**	補間軸2移動距離的設定值 範圍: -2,000,000,000 ~ +2,000,000,000)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

備註:

RSM_MIX_2D_CONTINUE 的Sub_function代碼為 0x0A 84。

RSM_MACRO_MIX_2D_CONTINUE 的Sub_function代碼為 0x0C 84。

Modbus 範例:

```
RSM_MIX_2D_CONTINUE (hRsm, 1, 0, 1, 0, 0, 100, 100);
//execute X, Y, 2-axis motions in continuous interpolation on module 1
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 0B	16
Register[]	**Value (hex)**	**Remarks**			
0	0A 84/0C 84	*Sub_funciton code*			
1	00 00	*nAcc*			
2	00 01	*nType*			
3	00 00	*MSW of cp1*			
4	00 00	*LSW of cp1 (for linear motion, set 0)*			
5	00 00	*MSW of cp2*			
6	00 00	*LSW of cp2 (for linear motion, set 0)*			
7	00 00	*MSW of fp1*			
8	00 64	*LSW of fp1 (100 = 0x64)*			
9	00 00	*MSW of fp2*			
10	00 64	*LSW of fp2 (100 = 0x64)*			

7.4.10 執行混合兩軸連續補間(絕對位置)

eRTU RSM_ABS_MIX_2D_CONTINUE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *nAcc*, BYTE *nType*, long *cp1*, long *cp2*, long *fp1*, long *fp2*)

※ eRTU RSM_ABS_MACRO_MIX_2D_CONTINUE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *nAcc*, BYTE *nType*, long *cp1*, long *cp2*, long *fp1*, long *fp2*)

說明:

執行一個混合式兩軸連續補間運動。使用此函式可選擇執行線性補間或是圓弧補間(指定移動到絕對位置)。

類別:

Modbus sub_function; RTC and MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***nAcc*:**	指令類型設定值，連續補間功能會依照此設定值自動執行適當的速度曲線運動 0 = 非最後一個補間運動指令 1 = 最後一個補間運動指令
***nType*:**	執行補間運動類型的設定值

設定值	補間類型	使用參數
1	RSM_ABS_LINE_2D	*fp1, fp2*
2	RSM_ABS_ARC_CW	*cp1, cp2, fp1, fp2*
3	RSM_ABS_ARC_CCW	*cp1, cp2, fp1, fp2*
4	RSM_ABS_CIRCLE_CW	*cp1, cp2*
5	RSM_ABS_CIRCLE_CCW	*cp1, cp2*

***cp1*:**	補間軸1的圓心絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000)
***cp2*:**	補間軸2的圓心絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000)
***fp1*:**	補間軸1所要移動到的絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000)
***fp2*:**	補間軸2所要移動到的絕對位置設定值 範圍: -2,000,000,000 ~ +2,000,000,000)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

備註:

RSM_ABS_MIX_2D_CONTINUE的Sub_function代碼為 0x0A FD。

RSM_ABS_MACRO_MIX_2D_CONTINUE的Sub_function代碼為 0x0C FD。

Modbus 範例:

```
RSM_ABS_MIX_2D_CONTINUE (hRsm, 1, 0, 1, 0, 0, 100, 100);
//execute X, Y, 2-axis motions in continuous interpolation on module 1
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 0B	16
Register[]	**Value (hex)**	**Remarks**			
0	0A FD/0C FD	*Sub_funciton code*			
1	00 00	*nAcc*			
2	00 01	*nType*			
3	00 00	*MSW of cp1*			
4	00 00	*LSW of cp1 (for linear motion, set 0)*			
5	00 00	*MSW of cp2*			
6	00 00	*LSW of cp2 (for linear motion, set 0)*			
7	00 00	*MSW of fp1*			
8	00 64	*LSW of fp1 (100 = 0x64)*			
9	00 00	*MSW of fp2*			
10	00 64	*LSW of fp2 (100 = 0x64)*			

7.4.11 執行三軸螺旋(Helical)補間

eRTU RSM_HELIX_3D (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis1*, BYTE *axis2*, BYTE *axis3*, BYTE *nDir*, DWORD *VV* , long *cp1*, long *cp2*, long *cycle*, long *pitch*)

※ eRTU RSM_MACRO_HELIX_3D (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis1*, BYTE *axis2*, BYTE *axis3*, BYTE *nDir*, DWORD *VV* , long *cp1*, long *cp2*, long *cycle*, long *pitch*)

說明:

執行一個三軸螺旋補間運動。此函式使用軟體巨集方式實作，因此在執行程式時會佔用系統資源。

類別:

Modbus sub_function; RTC and MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis1:	第一補間軸的軸號設定值 1=X 軸; 2=Y 軸; 4=Z 軸; 8=U 軸
axis2:	第二補間軸的軸號設定值 1=X軸; 2=Y軸; 4=Z軸; 8=U軸
axis3:	第三補間軸的軸號設定值 1=X軸; 2=Y軸; 4=Z軸; 8=U軸
nDir:	圓弧運動方向設定值 0 = CW; 1 = CCW
VV:	向量速度設定值，單位PPS
cp1:	補間軸1的圓心相對距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000
cp2:	補間軸2的圓心相對距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000
cycle:	週期次數設定值 範圍: -2,000,000,000 ~ +2,000,000,000
pitch:	補間軸3在每次週期移動的距離設定值 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

請使用函式"RSM_GET_ERROR_CODE()"取得運動控制相關的錯誤碼。

備註:

RSM_HELIX_3D 的Sub_function代碼為 0x0A 88。

RSM_MACRO_HELIX_3D 的 Sub_function 代碼為 0x0C 88。

Modbus 範例:

```
RSM_HELIX_3D (hRsm, 1, AXIS_Y, AXIS_Z, AXIS_U, 1, 50000, 0, 25000, 50, 3600);
//the circular motion is on YZ plane, and the linear motion is along the U axis.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 0F	1E
Register[]	**Value (hex)**	**Remarks**			
0	0A 88/0C 88	*Sub_funciton code*			
1	00 02	*axis1 (2 ➔ AXIS_Y)*			
2	00 04	*axis2 (4 ➔ AXIS_Z)*			
3	00 08	*axis3 (8 ➔ AXIS_U)*			
4	00 01	*nDir*			
5	00 00	*MSW of VV*			
6	C3 50	*LSW of VV (50000 = 0xC350)*			
7	00 00	*MSW of cp1*			
8	00 00	*LSW of cp1 (0 = 0x0)*			
9	00 00	*MSW of cp2*			
10	61 A8	*LSW of cp2 (25000 = 0x61A8)*			
11	00 00	*MSW of cycle*			
12	00 32	*LSW of cycle (50 ➔ 0x32)*			
13	00 00	*MSW of pitch*			
14	0E 10	*LSW of pitch (3600 ➔ 0xE10)*			

相關範例:

```
BYTE SlaveAddr=1; //select module 1.
//===============================================
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU,160000L);
//set maximum speed for all axes to 160K PPS.
long v=50000; //set vector speed to 50K PPS.
RSM_HELIX_3D(hRsm, SlaveAddr, AXIS_Y, AXIS_Z, AXIS_X, 1, v, 0, 1000, 5, -2000);
//the circular motion is on YZ plane, and the linear motion is along the X axis.
//===============================================
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU,160000L);
//set the maximum speed for all axes to 160K PPS.
long v=100000; //set vector speed to 100K PPS.
RSM_HELIX_3D(hRsm, SlaveAddr, AXIS_Y, AXIS_Z, AXIS_U, 1, v, 0, 25000, 50,
3600);
//the circular motion is on YZ plane, and the linear motion is along the U axis.
```

7.4.12 同步線性掃描運動設定

eRTU RSM_LINE_SCAN (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *Type*, BYTE *outEdge*, BYTE *PulseWidth*, long *Pitch*)

說明:

設定等距離線性掃描觸發訊號輸出功能。等間距最高觸發頻率 100KHz，不等間距最高觸發頻率 18KHz。

類別:

Modbus sub_function; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 等間距功能支援的軸號: 1(X 軸), 2(Y 軸) 不等間距功能支援的軸號: 1(X 軸) (Y 軸不支援)
***Type*:**	0 =等間距運動，比較器C+比較LP計數器 1 =不等間距運動，比較器C+比較LP計數器 2 =等間距運動，比較器C+比較EP計數器 3 =不等間距運動，比較器C+比較EP計數器
***outEdge*:**	輸出觸發邏輯: 0=低準位觸發(0V)；1=高準位觸發(5V)
***PulseWidth*:**	輸出脈波寬度 0 = 10 uSec 1 = 20 uSec 2 = 100 uSec 3 = 200 uSec 4 = 1,000 uSec 5 = 2,000 uSec 6 = 10,000 uSec 7 = 20,000 uSec
***pitch*:**	間隔距離 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
RSM_LINE_SCAN (hRsm, 1, AXIS_X, 0, 0, 0, -39);
```

//Sets the equal distance Line Scan trigger on axis-X.
//One trigger will be sent out every 40 (=39+1) pulses.

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 07	0E
Register[]	**Value (hex)**	**Remarks**			
0	0A 90	*Sub_funciton code*			
1	00 01	*axis (1 ➔ AXIS_X)*			
2	00 00	*Type (even distance trigger)*			
3	00 00	*outEdge (high level pulse)*			
4	00 00	*PulseWidth (0➔ 10 uSec)*			
5	FF FF	*MSW of Pitch*			
6	FF D9	*LSW of Pitch (-39 = 0xFFFFFFD9)*			

相關範例:

```
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XY, 4000000);
//set the maximum speed of X and Y axes to 4M PPS.
RSM_NORMAL_SPEED(hRsm, SlaveAddr, AXIS_XY, 0);
//set the driving mode to be symmetric T-curve.
RSM_SET_V(hRsm, SlaveAddr, AXIS_XY, 2000000);
//set the speed of X and Y axes to 2M PPS.
RSM_SET_A(hRsm, SlaveAddr, AXIS_XY, 20000000);
//set the acceleration of X and Y axes to 2M PPS/S.
RSM_SET_SV(hRsm, SlaveAddr, AXIS_XY, 2000000);
//set the starting speed of X and Y axes to 2M PPS.
RSM_LINE_SCAN(hRsm, SlaveAddr, AXIS_X, 0, 0, 0, -39);
//Sets the equal distance Line Scan trigger on axis-X.
//One trigger will be sent out every 40 (=39+1) pulses.
RSM_LINE_SCAN(hRsm, SlaveAddr, AXIS_Y, 0, 0, 0, -79);
//Sets the equal distance Line Scan trigger on axis-Y.
//One trigger will be sent out every 80 (=79+1) pulses.
RSM_LINE_SCAN_START(hRsm, SlaveAddr, AXIS_XY, 0, -4000000);
//Towards the negative direction move 4000K PPS.
```

7.4.13 開始執行同步線性掃描運動

eRTU RSM_LINE_SCAN_START (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, BYTE *Type*, long *Position*)

說明:

開始執行同步線性掃描運動。使用不等間距功能時請遵循下列兩項規則:

(a) ISR巨集程序將停止執行。

(b) 不可與下列函式同時使用。

- RSM_READ_bVAR
- RSM_READ_VAR
- RSM_GET_LP
- RSM_GET_EP
- RSM_GET_CV
- RSM_GET_CA
- RSM_GET_DI
- RSM_GET_ERROR
- RSM_GET_ERROR_CODE
- RSM_GET_LATCH
- RSM_STOP_WAIT
- RSM_CLEAR_STOP

類別:

Modbus sub_function; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 等間距功能支援的軸號: 1(X 軸), 2(Y 軸), 3(XY 軸) 不等間距功能支援的軸號: 1(X 軸) (Y 軸不支援)
***Type*:**	0 =等間距運動(依據LP計數器) *1 =不等間距運動(依據LP計數器) 2 =等間距運動(依據EP計數器) *3 =不等間距運動(依據EP計數器) *不等間距運動功能暫不開放
***Position*:**	總移動行程的距離 範圍: -2,000,000,000 ~ +2,000,000,000

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

RSM_LINE_SCAN_START (hRsm, 1, AXIS_XY, 0, -4000000);

//Towards the negative direction move 4000K PPS.

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 05	0A
Register[]	**Value (hex)**	**Remarks**			
0	0A 91	*Sub_funciton code*			
1	00 03	*axis (3 ➔ AXIS_XY)*			
2	00 00	*Type*			
3	FF C2	*MSW of Position*			
4	F7 00	*LSW of Position (-4000000 = 0xFFC2F700)*			

7.4.14 取得同步線性掃描運動狀態

eRTU RSM_GET_LINE_SCAN_DONE (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE* *LScanState*)

說明:

取得同步線性掃描運動狀態。

類別:

Modbus table；RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***LScanState*:**	1=動作已完成 0=動作尚未完成

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
BYTE bState;
RSM_GET_LINE_SCAN_DONE (hRsm, 1, & bState);
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01 (Card No.)	04	00 49 (73)	00 01

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	Register(hex)
01	04	02	00 01 line scan operation has ended

7.5 停止與指令保持功能

7.5.1 移動指令保持

eRTU RSM_DRV_HOLD (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

※Δ eRTU RSM_MACRO_DRV_HOLD (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

說明:

執行此函式可以將後續執行的運動指令保持住，等待觸發。可使用函式”RSM_DRV_START()”觸發被保持住的運動指令立即執行。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_DRV_HOLD 的Sub_function代碼為 0x0A B4。

RSM_MACRO_DRV_HOLD 的 Sub_function 代碼為 0x0C B4。

Modbus 範例:

```
RSM_DRV_HOLD (hRsm, 1, AXIS_XYU); //hold the driving command to XYU
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A B4/0C B4	*Sub_funciton code*			
1	00 0B	*axis (B ➔ AXIS_XYU)*			

相關範例:

```
BYTE SlaveAddr=1; //select card 1.
RSM_DRV_HOLD(hRsm, SlaveAddr, AXIS_XYU);
//hold the driving command to XYU
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_U, 10000);
//set the maximum speed of U-axis to be 10K PPS.
RSM_NORMAL_SPEED(hRsm, SlaveAddr, AXIS_U, 0);
//set the driving mode to be symmetric T-curve.
RSM_SET_V(hRsm, SlaveAddr, AXIS_U, 2000);
//set the speed of U-axis to 2,000 PPS.
RSM_SET_A(hRsm, SlaveAddr, AXIS_U,1000);
//set the acceleration of U-axis to 1,000 PPS/S.
RSM_SET_SV(hRsm, SlaveAddr, AXIS_U, 2000);
//set the starting speed to 2,000 PPS.
RSM_SET_AO(hRsm, SlaveAddr, AXIS_U, 9); // set the AO to 9 Pulses.
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XY, 20000);
//set the maximum speed of X and Y axes to 20K PPS.
RSM_AXIS_ASSIGN(hRsm, SlaveAddr, AXIS_X, AXIS_Y, 0);
//set the X-axis as the axis 1 and Y-axis as the axis 2 for a 2-axis
// interpolation.
RSM_VECTOR_SPEED(hRsm, SlaveAddr, 0);
//set constant speed motion.  Therefore, VSV=VV. Only VV is required.
RSM_SET_VV(hRsm, SlaveAddr, 5000);
//set the vector speed for card 1 to 5,000   PPS.
RSM_FIXED_MOVE(hRsm, SlaveAddr, AXIS_U, 5000);
//command U-axis to move          5,000 Pulse.  This command is be held.
RSM_LINE_2D(hRsm, SlaveAddr, 12000, 10000);
//command a linear interpolation motion on the XY planes. It is held, too.
RSM_DRV_START(hRsm, SlaveAddr, AXIS_XYU);
//release the holding status. X,Y , and U axes will start to move
// simultaneously.
```

7.5.2 解除指令保持狀態並開始移動

eRTU RSM_DRV_START (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

※Δ eRTU RSM_MACRO_DRV_START (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

說明:

解除運動指令的保持狀態並立即觸發指令執行。請使用函式"RSM_DRV_HOLD()"保持運動指令。

類別:

Modbus sub_function; RTC, MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表4)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_DRV_START 的Sub_function代碼為 0x0A B5。

RSM_MACRO_DRV_START 的 Sub_function 代碼為 0x0C B5。

Modbus 範例:

```
RSM_DRV_START (hRsm, 1, AXIS_XYU);
//release the holding status. X,Y , and U axes will start to move simultaneously.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A B5/0C B5	*Sub_funciton code*			
1	00 0B	*axis (B ➔ AXIS_XYU)*			

相關範例:

```
BYTE SlaveAddr=1; //select card 1.
RSM_DRV_HOLD(hRsm, SlaveAddr, AXIS_XYU);
//hold the driving command to XYU
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_U, 10000);
//set the maximum speed of U-axis to be 10K PPS.
RSM_NORMAL_SPEED(hRsm, SlaveAddr, AXIS_U, 0);
//set the driving mode to be symmetric T-curve.
RSM_SET_V(hRsm, SlaveAddr, AXIS_U, 2000);
//set the speed of U-axis to 2,000 PPS.
RSM_SET_A(hRsm, SlaveAddr, AXIS_U,1000);
//set the acceleration of U-axis to 1,000 PPS/S.
RSM_SET_SV(hRsm, SlaveAddr, AXIS_U, 2000);
//set the starting speed to 2,000 PPS.
RSM_SET_AO(hRsm, SlaveAddr, AXIS_U, 9); // set the AO to 9 Pulses.
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XY, 20000);
//set the maximum speed of X and Y axes to 20K PPS.
RSM_AXIS_ASSIGN(hRsm, SlaveAddr, AXIS_X, AXIS_Y, 0);
//set the X-axis as the axis 1 and Y-axis as the axis 2 for a 2-axis
// interpolation.
RSM_VECTOR_SPEED(hRsm, SlaveAddr, 0);
//set constant speed motion.  Therefore, VSV=VV. Only VV is required.
RSM_SET_VV(hRsm, SlaveAddr, 5000);
//set the vector speed for card 1 to 5,000   PPS.
RSM_FIXED_MOVE(hRsm, SlaveAddr, AXIS_U, 5000);
//command U-axis to move          5,000 Pulse.  This command is be held.
RSM_LINE_2D(hRsm, SlaveAddr, 12000, 10000);
//command a linear interpolation motion on the XY planes. It is held, too.
RSM_DRV_START(hRsm, SlaveAddr, AXIS_XYU);
//release the holding status. X,Y , and U axes will start to move
// simultaneously.
```

7.5.3 立即減速後停止指定軸的運動

eRTU RSM_STOP_SLOWLY (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

※ eRTU RSM_MACRO_STOP_SLOWLY (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

說明:

執行此函式會立即減速後停止運動中的指定軸，並且將此軸鎖定為停止狀態，在此狀態下針對指定軸所傳送的移動指令將被視為無效。請使用"RSM_CLEAR_STOP()"解除停止狀態。

類別:

Modbus sub_function; RTC, and MP

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表4)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_STOP_SLOWLY 的Sub_function代碼為 0x0A B7。

RSM_MACRO_STOP_SLOWLY 的 Sub_function 代碼為 0x0C B7。

Modbus 範例:

RSM_STOP_SLOWLY (hRsm, 1, AXIS_XY); //decelerate and stop the X and Y axes

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A B7/0C B7	*Sub_funciton code*			
1	00 03	*axis (3 ➔ AXIS_XY)*			

7.5.4 立即停止指定軸的運動

eRTU RSM_STOP_SUDDENLY (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

※ eRTU RSM_MACRO_STOP_SUDDENLY (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

說明:

執行此函式會立即停止運動中的指定軸，並且將此軸鎖定為停止狀態，在此狀態下針對指定軸所傳送的移動指令將被視為無效。請使用”RSM_CLEAR_STOP()”解除停止狀態。

類別:

Modbus sub_function; RTC, and MP

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***axis*:**	軸號設定值 (請參照表4)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_STOP_SUDDENLY 的Sub_function代碼為 0x0A B8。

RSM_MACRO_STOP_SUDDENLY 的 Sub_function 代碼為 0x0C B8。

Modbus 範例:

RSM_STOP_SUDDENLY (hRsm, 1, AXIS_ZU); **//immediately stop the Z and U axes.**

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A B8/0C B8	*Sub_funciton code*			
1	00 0C	*axis (C ➔ AXIS_ZU)*			

7.5.5 立即減速後停止補間運動

eRTU RSM_VSTOP_SLOWLY (HANDLE *hRsm*, BYTE *SlaveAddr*)

※ eRTU RSM_MACRO_VSTOP_SLOWLY (HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

執行此函式會使運動中的補間軸立即減速後停止，並且鎖定補間軸為停止狀態，在此狀態下針對補間軸所傳送的移動指令將被視為無效。請使用"RSM_CLEAR_VSTOP()"解除停止狀態。

類別:

Modbus sub_function; RTC, and MP

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_VSTOP_SLOWLY 的Sub_function代碼為 0x0A B9。
RSM_MACRO_VSTOP_SLOWLY 的 Sub_function 代碼為 0x0C B9。

Modbus 範例:

```
RSM_VSTOP_SLOWLY (hRsm, 1);
//stop the interpolation of card 1 in a decelerating way.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	0A B9/0C B9	*Sub_funciton code*			

7.5.6 立即停止補間運動

eRTU RSM_VSTOP_SUDDENLY (HANDLE *hRsm*, BYTE *SlaveAddr*)

※ eRTU RSM_MACRO_VSTOP_SUDDENLY (HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

執行此函式會立即停止運動中的補間軸，並且鎖定補間軸為停止狀態，在此狀態下針對補間軸所傳送的移動指令將被視為無效。請使用”RSM_CLEAR_VSTOP()”解除停止狀態。

類別:

Modbus sub_function; RTC, and MP

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_VSTOP_SUDDENLY 的Sub_function代碼為 0x0A BA。
RSM_MACRO_VSTOP_SUDDENLY 的 Sub_function 代碼為 0x0C BA。

Modbus 範例:

RSM_VSTOP_SUDDENLY (hRsm, 1); // stop the interpolation of card 1 immediately.

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	0A BA/0C BA	*Sub_funciton code*			

7.5.7 解除軸停止狀態

eRTU RSM_CLEAR_STOP (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

※ eRTU RSM_MACRO_CLEAR_STOP (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

說明:

執行停止函式 ” RSM_STOP_SLOWLY ()” 或” RSM_STOP_SUDDENLY ()”後，請先排除異常狀況，再使用此函式解除停止狀態。

類別:

Modbus sub_function; RTC, and MP

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
axis:	**軸號設定值 (請參照表4)**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_CLEAR_STOP 的Sub_function代碼為 0x0A BB。

RSM_MACRO_CLEAR_STOP 的 Sub_function 代碼為 0x0C BB。

Modbus 範例:

RSM_CLEAR_STOP (hRsm, 1, AXIS_ZU); //clear the error status of zu axes.

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A BB/0C BB	*Sub_funciton code*			
1	00 0C	*axis (C ➔ AXIS_ZU)*			

7.5.8 解除補間停止狀態

eRTU RSM_CLEAR_VSTOP (HANDLE *hRsm*, BYTE *SlaveAddr*)

※ eRTU RSM_MACRO_CLEAR_VSTOP (HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

執行停止函式 ” RSM_VSTOP_SLOWLY ()” 或” RSM_VSTOP_SUDDENLY()”後，請先排除異常狀況，再使用此函式解除停止狀態。

類別:

Modbus sub_function; RTC, and MP

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_CLEAR_VSTOP 的Sub_function代碼為 0x0A 09。
RSM_MACRO_CLEAR_VSTOP 的 Sub_function 代碼為 0x0C 09。

Modbus 範例:

RSM_CLEAR_VSTOP (hRsm, 1); //clear the error status of card 1.

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	0A 09/0C 09	***Sub_funciton code***			

7.5.9 軟體緊急停止

eRTU RSM_EMERGENCY_STOP (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *stopType*)

說明:

停止所有正在移動的軸並且清空指令暫存區。

類別:

Modbus sub_function; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***stopType*:**	停止類型 0=立即停止 1=減速後停止

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

RSM_EMERGENCY_STOP (hRsm, 1, 0);

// Stop all the axes immediately and clear all data in the i-8094H command buffer.

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A 04	*Sub_function code*			
1	00 00	*stopType*			

7.5.10 解除軟體緊急停止狀態

eRTU RSM_CLEAR_EMERGENCY_STOP (HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

執行緊急停止函式 ”RSM_EMERGENCY_STOP()”後，請先排除異常狀況，再使用此函式解除緊急停止狀態。

類別:

Modbus sub_function; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

RSM_CLEAR_EMERGENCY_STOP (hRsm, 1); //clear the error status of card 1.

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	0A 05	*Sub_function code*			

8 初始參數表

初始表被用來記錄運動控制晶片的基本設定參數，並且儲存在非揮發性(non-volatile)的記憶體內，當 RS-M8194H 開機時會自動載入初始表進行設定。關於設定的參數列表如圖 3 所示。

Function	Parameter	X-Axis	Y-Axis	Z-Axis	U-Axis
Pulse Output Signal	Pulse Output Mode	0	0	0	0
Max Pulse Output Rate	Data (8000 to 4,000,000 PPS)	8000	8000	8000	8000
Hardware Limit Switch (HLMT)	Active Level (forward)	Low Active	Low Active	Low Active	Low Active
	Active Level (reverse)	Low Active	Low Active	Low Active	Low Active
Hardware Limit Stop Mode	Stop Mode	Abrupt Stop	Abrupt Stop	Abrupt Stop	Abrupt Stop
Near Home Sensor	Trigger Level	Low Active	Low Active	Low Active	Low Active
Home Sensor	Trigger Level	Low Active	Low Active	Low Active	Low Active
Software Limit	Enable Software Limit	Disable	Disable	Disable	Disable
	Software Limit (forward)	100000	100000	100000	100000
	Software Limit (reverse)	-100000	-100000	-100000	-100000
	Position Counter Type	Logic Pos	Logic Pos	Logic Pos	Logic Pos
Set Encoder Parameters	Encoder Input Type	A Quad B	A Quad B	A Quad B	A Quad B
	A Quad B Input Signal Division	1/1	1/1	1/1	1/1
	Trigger Level for Z Phase	Low Active	Low Active	Low Active	Low Active
Servo Driver Setting	On/Off	Off	Off	Off	Off
Servo Alarm Setting	Enable Servo Alarm	Disable	Disable	Disable	Disable
	Trigger Level	Low Active	Low Active	Low Active	Low Active
In-Position Signal	Enable In-Position Input	Disable	Disable	Disable	Disable
	Trigger Level	Low Active	Low Active	Low Active	Low Active
Digital Filter	Input Ports	1	1	1	1
	Filter Time Constant	2	2	2	2
Variable Ring Position Counter	Enable Variable Ring Counter	Disable	Disable	Disable	Disable
	Maximum Value	10000	10000	10000	10000
Triangle Driving Profile Prevention	Enable Triangle Prevention	Disable	Disable	Disable	Disable

圖 3: 初始表

與RS-M8194H連線後可使用下列函式修改運動控制相關的設定參數，若使用表5的軸參數代碼(工具軟體 EzMove 使用此方式設定)，則參數值可以被儲存在初始表中。

出廠設定值:

```
RSM_SET_PULSE_MODE(hRsm, SlaveAddr, AXIS_XYZU, 0);
RSM_SET_MAX_V(hRsm, SlaveAddr, AXIS_XYZU, 2000000L);
RSM_SET_HLMT(hRsm, SlaveAddr, AXIS_XYZU, 0, 0);
RSM_LIMITSTOP_MODE(hRsm, SlaveAddr, AXIS_XYZU, 0);
```

RSM_SET_NHOME(hRsm, SlaveAddr, AXIS_XYZU, 0);
RSM_SET_HOME_EDGE(hRsm, SlaveAddr, AXIS_XYZU, 0);
RSM_CLEAR_SLMT(hRsm, SlaveAddr, AXIS_XYZU);
RSM_SET_ENCODER(hRsm, SlaveAddr, AXIS_XYZU, 0, 0, 0);
RSM_SERVO_OFF(hRsm, SlaveAddr, AXIS_XYZU);
RSM_SET_ALARM(hRsm, SlaveAddr, AXIS_XYZU, 0, 0);
RSM_SET_INPOS(hRsm, SlaveAddr, AXIS_XYZU, 0, 0);
RSM_SET_FILTER(hRsm, SlaveAddr, AXIS_XYZU, 0, 0);
RSM_VRING_DISABLE(hRsm, SlaveAddr, AXIS_XYZU);
RSM_AVTRI_DISABLE(hRsm, SlaveAddr, AXIS_XYZU);
RSM_EXD_DISABLE(hRsm, SlaveAddr, AXIS_XYZU);

8.1 載入初始表(Initial Table, IT)設定值

eRTU RSM_LOAD_INITIAL (HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

載入初始參數設定表。當RS-M8194H開機時會自動執行此函式載入儲存在記憶體內的設定值。

類別:

Modbus sub_function; RTC

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
RSM_LOAD_INITIAL (hRsm, 1);
// load the initial setting values of the parameter table into i-8094H
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	0A C8	*Sub_funciton code*			

相關範例:

```
RSM_SET_PULSE_MODE(hRsm, 1, INITIAL_XYZU, 0);
//set the pulse mode of X, Y, Z, and U axes as 0, write into the parameter
//table in module 1.
RSM_SET_MAX_V(hRsm, 1, INITIAL_XY, 200000L);
//The maximum speed for the X and Y axes of module 1 is 200KPPS.
```

```
//Write into the parameter table.
RSM_SET_HLMT(hRsm, 1, INITIAL_XYZU, 0, 0);
//set all the trigger levels as low-active for all limit switches
//on module 1. Write into the parameter table.
RSM_LIMITSTOP_MODE(hRsm, 1, INITIAL_X, 0);
//set X axis to stop immediately if any limit switch on X axis is triggered
//on module 1. Write into the parameter table.
RSM_SET_NHOME(hRsm, 1, INITIAL_XY, 0);
//set the trigger level of NHOME of X and Y axes on module 1 to be
//active low. Write into the parameter table.
RSM_ LOAD_INITIAL(hRsm, 1);
// load the initial setting values of the parameter table into i-8094H
```

9 巨集程序

簡介

巨集程序儲存於i-8094H的非揮發性(non-volatile)記憶體內，目前支援的巨集程序存放位址(在後續的函式說明中將使用”編號”作為替代稱呼)有 MP1~MP157與ISR1~ISR20，不同的位址所能存放的指令數量也不相同，最大限制請參考圖4說明。

- MP 巨集程序位址:存放一般巨集命令所組成的巨集程序，可使用函式”RSM_MP_CALL()”或”RSM_MACRO_MP_CALL()”呼叫執行。
- ISR 巨集程序位址: 存放的的巨集程序需由運動晶片所發出的中斷觸發執行，為了避免中斷時間過長，因此有部份指令並不支援使用。

Macro Type	Number of Macros	Number of Command Lines	Macro Names								
Interrupt Service Routine (ISR)	6	8	ISR1	ISR2	ISR3	ISR4	ISR5	ISR6			
	9	16	ISR7	ISR8	ISR9	ISR10	ISR11	ISR12	ISR13	ISR14	ISR15
	3	32	ISR16	ISR17	ISR18						
	2	64	ISR19	ISR20							
Motion Program (MP)	40	8	MP1	~	MP40						
	50	16	MP41	~	MP90						
	40	32	MP91	~	MP130						
	20	64	MP131	~	MP150						
	5	128	MP151	MP152	MP153	MP154	MP155				
	2	512	MP156	MP157							

圖 4: 各個巨集程序存放位址所能支援的最大指令數量

9.1 建立 MP 巨集程序

9.1.1 開始一個 MP 巨集程序的編輯

eRTU RSM_MP_CREATE(HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *mp_No*)

說明:

設定開始 MP 巨集程序編輯與指定編輯的巨集程序編號。請使用此函式作為巨集程序的第一行指令，與函式"RSM_MACRO_MP_CLOSE()"作為最後一行指令。

類別:

Modbus sub_function; RTC

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
mpNo:	**MP 巨集程序編號設定值** **範圍: 1(MP1) ~ 157(MP157)**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
RSM_MP_CREATE (hRsm, 1, MP21);
//Write Macro Program into i-8094H.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A C9	*Sub_funciton code*			
1	00 15	*mpNo (MP21 = 21 = 0x15)*			

相關範例:

```
RSM_MP_CREATE(hRsm, 1, MP21); //Write #21 Macro Program into i-8094H.
//=========================================================
//The following functions will not be executed, but will be written into
//i-8094H for further execution.
RSM_MACRO_SET_MAX_V(hRsm, 1, AXIS_XYZU, 20000);
//The maximum speed of all axes is 20K PPS.
RSM_MACRO_NORMAL_SPEED(hRsm, 1, AXIS_XYZU, 0);
//Set symmetric T-curve for XYZU axes on module.
RSM_MACRO_SET_V(hRsm, 1, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to be 2000 PPS.
RSM_MACRO_SET_A(hRsm, 1, AXIS_XYZU, 1000);
//set the acceleration of XYZU axes to be 1000 PPS/sec
RSM_MACRO_SET_SV(hRsm, 1, AXIS_XYZU, 2000);
//set the start speed of XYZU axes to be 2000 PPS.
RSM_MACRO_SET_AO(hRsm,1, AXIS_XYZU, 9);
//set the number of remaining offset pulses for XYZU axes to be 9 pulses.
RSM_MACRO_FIXED_MOVE(hRsm, 1, AXIS_XYZU, 10000);
//move XYZU axes to be 10000 pulses.
RSM_MACRO_MP_CLOSE(hRsm, 1);
// module 1 finish, write Macro Program into i-8094H
//=========================================================
```

9.1.2 結束一個 MP 巨集程序的編輯

※ eRTU RSM_MACRO_MP_CLOSE (HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

設定結束 MP 巨集程序編輯。請使用此函式作為巨集程序的最後一行指令，與函式”RSM_MP_CREATE ()”作為第一行指令。

類別:

Modbus sub_function; MP

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
RSM_MACRO_MP_CLOSE (hRsm, 1);
// module 1 finish, write Macro Program into i-8094H
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	0C CA	*Sub_funciton code*			

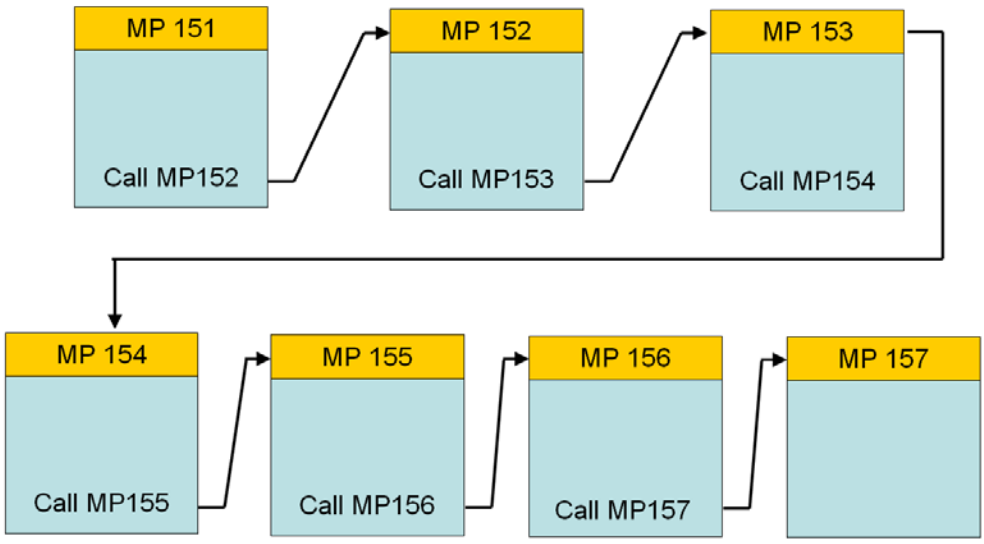
9.1.3 執行 MP 巨集程序

eRTU RSM_MP_CALL (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *mpNo*)

※ eRTU RSM_MACRO_MP_CALL (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *mpNo*)

說明:

呼叫並執行指定的MP巨集程序。在MP巨集程序內允許呼叫執行其他的巨集程序，最多可執行到第七層程序，ISR巨集程序內則無法呼叫其他巨集程序。



類別:

Modbus table, Modbus sub_function; RTC and MP.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
mpNo:	**MP 巨集程序編號設定值 範圍: 1(MP1) ~ 157(MP157)**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_MP_CALL 的Sub_function代碼為 0x0A CB。

RSM_MACRO_MP_CALL 的 Sub_function 代碼為 0x0C CB。

Modbus 範例:

```
//The true value of MPn = n; therefore, MP21 = 21 = 0x15.
RSM_MP_CALL (hRsm, 1, MP21); // Execute the MP21 on i-8094H
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A CB/0C CB	*Sub_funciton code*			
1	00 15	*mpNo (MP21 = 21 = 0x15)*			

也可以使用已定義的 Holding 暫存器寫入巨集程序編號(起始暫存器位址為 0x00 08)。

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	00 08	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	15	*mpNo (MP21 = 21 = 0x15)*			

相關範例:

```
RSM_MP_CREATE(hRsm, 1, MP21); //Write #21 Macro Program into i-8094H.
//============================================================
//The following functions will not be executed, but will be written into
//i-8094H for further execution.
RSM_MACRO_SET_MAX_V(hRsm, 1, AXIS_XYZU, 20000);
//The maximum speedof the axis is 20K PPS.
RSM_MACRO_NORMAL_SPEED(hRsm, 1, AXIS_XYZU, 0);
//Set symmetric T-curve for XYZU axes on module.
RSM_MACRO_SET_V(hRsm, 1, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to be 2000 PPS.
RSM_MACRO_SET_A(hRsm, 1, AXIS_XYZU, 1000);
//set the acceleration of XYZU axes to be 1000 PPS/sec.
RSM_MACRO_SET_SV(hRsm, 1, AXIS_XYZU, 2000);
//set the start speed of XYZU axes to be 2000 PPS.
RSM_MACRO_SET_AO(hRsm, 1, AXIS_XYZU, 9);
//set the number of remaining offset pulses for XYZU axes to be 9 pulses.
RSM_MACRO_FIXED_MOVE(hRsm, 1, AXIS_XYZU, 10000);
//move XYZU axes to be 10000 pulses.
RSM_MACRO_MP_CLOSE(hRsm, 1);
// module 1 finish, write Macro Program into i-8094H
//============================================================
RSM_MP_CALL(hRsm, 1, MP21); // Execute the MP21 on i-8094H
```

9.2 建立 ISR 巨集程序

9.2.1 開始一個 ISR 巨集程序的編輯

eRTU RSM_MP_ISR_CREATE(HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *isr_No*)

說明:

設定開始 ISR 巨集程序編輯與指定編輯的巨集程序編號。請使用此函式作為巨集程序的第一行指令，與函式"RSM_MACRO_MP_ISR_CLOSE()"作為最後一行指令。ISR 巨集程序必須由 i-8094H 內的運動晶片發出中斷觸發執行，請參照函式"RSM_INTFACTOR_ENABLE()"說明。

類別:

Modbus sub_function; RTC

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
isr_No:	ISR 巨集程序編號設定值 範圍: 1(ISR1) ~ 20(ISR20)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
RSM_MP_ISR_CREATE (hRsm, 1, ISR1);
//Write ISR Macro program into i-8094H.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0ACD	*Sub_funciton code*			
1	00 01	*isr_No (ISR1 = 1 = 0x01)*			

9.2.2 結束一個 ISR 巨集程序的編輯

Δ eRTU RSM_MACRO_MP_ISR_CLOSE (HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

設定結束 ISR 巨集程序編輯。請使用此函式作為巨集程序的最後一行指令，與函式" RSM_MP_ISR_CREATE ()"作為第一行指令。

類別:

Modbus sub_function; ISR

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
RSM_MACRO_MP_ISR_CLOSE (hRsm, 1);
// finish the writing of ISR Macro Program for i-8094H in slot 1
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	0C CE	*Sub_funciton code*			

9.2.3 執行 ISR 巨集程序

eRTU RSM_MP_ISR_CALL (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *isrNo*)

說明:

呼叫並執行指定的ISR巨集程序。通常ISR巨集程序應使用硬體中斷觸發執行，但可以使用此函式執行並測試ISR程序。

類別:

Modbus table, Modbus sub_function; ISR.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
***isrNo*:**	**ISR 巨集程序編號設定值 範圍: 1(ISR1) ~ 20(ISR20)**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
//The true value of ISRn = n; therefore, ISR1 = 1 = 0x1.
RSM_MP_ISR_CALL (hRsm, 1, ISR1); // Execute the ISR1
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A CF	*Sub_funciton code*			
1	00 01	*isrNo (ISR1 = 1 = 0x1)*			

9.3 使用者自定義巨集變數

9.3.1 指定巨集變數的設定值

※Δ eRTU RSM_MACRO_SET_VAR (HANDLE *hRsm*, BYTE *SlaveAddr*, long *varNo*, long *data*)

說明:

指定巨集變數的設定值。除了設定一般常數之外，也可以指定等於另外的巨集變數

類別:

Modbus table, Modbus sub_function; MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
varNo:	巨集變數編號 範圍: 0x7fff0000 (VAR0) ~ 0x7fff01ff (VAR511)
data:	設定值 範圍: -2,000,000,000 ~ +2,000,000,000 或巨集變數 (VAR0~VAR511)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
//The address of VARn = 0x7FFF0000 + n; therefore, the address of VAR1 is
//0x7FFF0001.
RSM_MACRO_SET_VAR(hRsm, 1, VAR1, 100); // VAR1 = 100
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 05	0A
Register[]	**Value (hex)**	**Remarks**			
0	0C D2	*Sub_funciton code*			
1	7F FF	*MSW address of varNo*			
2	00 01	*LSW address of varNo*			
3	00 00	*MSW of data*			
4	00 64	*LSW of data (100 = 0x64)*			

9.3.2 取得巨集程序內的指令回傳值

※Δ eRTU RSM_MACRO_SET_RVAR (HANDLE *hRsm*, BYTE *SlaveAddr*, long *varNo*)

說明:

設定將巨集指令取得的各項數值與狀態儲存到指定的巨集變數內(資料型態將自動轉為長整數)，使用時請將此函式接續在讀取函式之後。

可儲存回傳數值與狀態的函式如下:

- RSM_MACRO_GET_LP
- RSM_ MACRO_GET_EP
- RSM_MACRO_ABS_GET_POSITION
- RSM_MACRO_GET_HOME_SEARCH_STATE
- RSM_ MACRO_GET_ERROR
- RSM_ MACRO_GET_ERROR_CODE
- RSM_ MACRO_FRNET_IN
- RSM_MACRO_FRNET_READ
- RSM_ MACRO_GET_LATCH
- RSM_ MACRO_GET_DI_ALL
- RSM_ MACRO_GET_DI
- RSM_MACRO_GET_DI_SIGNAL

類別:

Modbus sub_function; MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
varNo:	變數編號 範圍: 0x7fff0000 (VAR0) ~ 0x7fff01ff (VAR511)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
// The address of VARn = 0x7FFF0000 + n; therefore, the address of VAR5
// is 0x7FFF0005.
RSM_MACRO_SET_RVAR(hRsm, 1, VAR5);
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0C D3	*Sub_funciton code*			
1	7F FF	*MSW address of varNo*			
2	00 05	*LSW address of varNo*			

相關範例:

```
//The following commands show how to use Macro commands to store the
//logic position in a variable.
RSM_MP_CREATE(hRsm, 1, MP1); //Write #1 Macro Program into i-8094H.
RSM_MACRO_GET_LP(hRsm, 1, AXIS_X);
//Reads the LP value of the X axis on module 1.
RSM_MACRO_SET_RVAR(hRsm, 1, VAR5);
//Assign the return value of RSM_MACRO_GET_LP to VAR5
//( VAR5 = LP       value)
RSM_MACRO_MP_CLOSE(hRsm, 1);
// module 1 finish, write Macro Program  into i-8094H
```

9.4 簡易數值運算

※Δ eRTU RSM_MACRO_VAR_CALCULATE (HANDLE *hRsm*, BYTE *SlaveAddr*, long *varNo*, BYTE *Operator*, long *varNo1*, long *varNo2*)

說明:

執行兩個運算元的基本計算:

varNo	+	*varNo1*	=	*varNo2*
varNo	-	*varNo1*	=	*varNo2*
varNo	*	*varNo1*	=	*varNo2*
varNo	/	*varNo1*	=	*varNo2*
varNo	&	*varNo1*	=	*varNo2*
varNo	|	*varNo1*	=	*varNo2*

類別:

Modbus sub_function; MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
***varNo*:**	第一個運算元的設定值 範圍: -2,000,000,000 ~ +2,000,000,000 或巨集變數 (VAR0~VAR511)
***Operator*:**	算數運算子設定值 '+' 相加 '—' 相減 '*' 相乘 '／' 相除(無條件捨去小數位) '&' AND 運算 '|' OR 運算
***varNo1*:**	第二個運算元的設定值 範圍: -2,000,000,000 ~ +2,000,000,000 或巨集變數 (VAR0~VAR511)
***varNo2*:**	儲存運算結果的巨集變數編號設定值 範圍: VAR0~VAR511

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

巨集變數的起始位址VARn= 0x7FFF0000 + n，例如VAR2的起始位址為0x7FFF0002。

運算子的 ASCII碼:

Operator	ASCII (dec)	ASCII (hex)
'+'	43	2B
'-'	45	2D
'*'	42	2A
'/'	47	2F
'&'	38	26
'|'	124	7C

```
RSM_MACRO_VAR_CALCULATE (hRsm, 1, VAR1, '+', VAR2, VAR3);
//VAR1 + VAR2 = VAR3
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 08	10
Register[]	**Value (hex)**	**Remarks**			
0	0C D4	*Sub_funciton code*			
1	7F FF	*MSW address of varNo*			
2	00 01	*LSW address of varNo*			
3	00 2B	*Operator : ASCII code of '+'*			
4	7F FF	*MSW address of varNo1*			
5	00 02	*LSW address of varNo1*			
6	7F FF	*MSW address of varNo2*			
7	00 03	*LSW address of varNo2*			

相關範例:

```
RSM_MP_CREATE(hRsm, 1, MP100); // Write #100 Macro Program into i-8094H.
//============================================================
//The following functions will not be executed, but will be written into
// i-8094H for further execution.
RSM_MACRO_SET_VAR(hRsm, 1, VAR1, 100); //VAR1 = 100
RSM_MACRO_SET_VAR (hRsm, 1, VAR2, 200); //VAR2 = 200
RSM_MACRO_SET_VAR (hRsm, 1, VAR10, 10); //VAR10 = 10
//-------------------------------------------------------------------------------------------------
RSM_MACRO_FRNET_IN(hRsm, 1, 8);
RSM_MACRO_SET_RVAR(hRsm, 1, VAR6);
//VAR6 = current input of RA8 on module 1
//-------------------------------------------------------------------------------------------------
RSM_MACRO_GET_LP(hRsm, 1, AXIS_X);
//Reads the LP value of the X axis on module 1.
RSM_MACRO_SET_RVAR(hRsm, 1, VAR5);
//VAR5 = LP value of the X axis on module 1.
//-------------------------------------------------------------------------------------------------
```

```
RSM_MACRO_VAR_CALCULATE(hRsm, 1, VAR1, '+', VAR2, VAR3);
//VAR1 + VAR2 = VAR3
RSM_MACRO_VAR_CALCULATE (hRsm, 1, VAR3, '-', VAR1, VAR3);
//VAR3 – VAR1 = VAR3
RSM_MACRO_VAR_CALCULATE (hRsm, 1, VAR3, '*', VAR10, VAR2);
//VAR3 x VAR10 = VAR2
RSM_MACRO_VAR_CALCULATE (hRsm, 1, VAR3, '/', VAR2, VAR1);
//VAR3 / VAR2 = VAR1
//Results: VAR1 = 10; VAR2 = 200; VAR3 = 2,000; VAR10 = 10
RSM_MACRO_MP_CLOSE(hRsm, 1);
// module 1 finish, write #100 Macro Program into i-8094H
//=========================================================
```

9.5 指令的循環執行 (FOR~NEXT)

9.5.1 開始指令循環區域敘述式(FOR)

※ eRTU RSM_MACRO_FOR (HANDLE *hRsm*, BYTE *SlaveAddr*, long *varNo*)

說明:

設定開始巨集程序指令循環區域與指定循環次數。請使用此函式作為循環區域的第一行指令，與函式" RSM_MACRO_NEXT ()"作為最後一行指令。循環次數可使用一般常數或巨集變數作為設定值，若使用巨集變數，在每回合的循環結束後巨集變數的數值將自動被減一。請注意，巢狀層次勿超過七層。

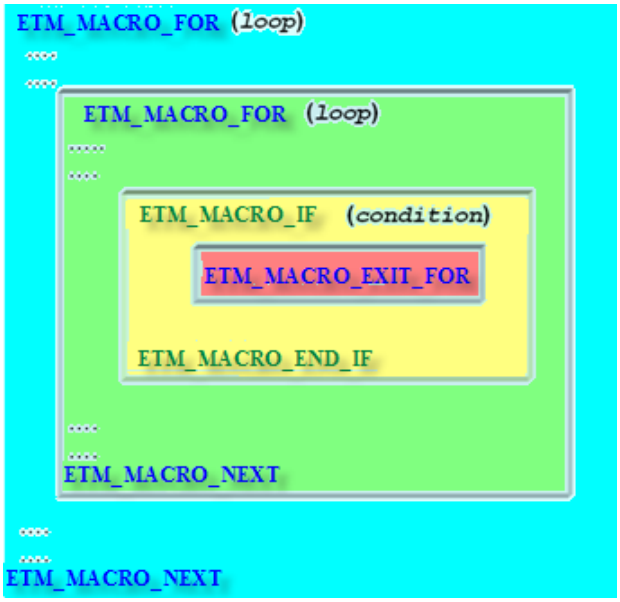


圖 5: 基本指令循環應用

類別:

Modbus sub_function; MP.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
varNo:	**循環次數設定值** **範圍: -2,000,000,000 ~ +2,000,000,000 或巨集變數 (VAR0~VAR511)**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

巨集變數的起始位址VARn= 0x7FFF0000 + n，例如VAR1的起始位址為0x7FFF0001。

RSM_MACRO_FOR (hRsm, 1, VAR1);

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0C D5	*Sub_funciton code*			
1	7F FF	*MSW address of IValue*			
2	00 01	*LSW address of IValue*			

相關範例:

```
RSM_MP_CREATE(hRsm, 1, MP100); // Write #100 Macro Program into i-8094H.
//=========================================================
// The following functions will not be executed, but will be written into
// i-8094H for further execution.
RSM_MACRO_SET_MAX_V(hRsm, 1, AXIS_X, 20000);
//The maximum speed of the axis X is 20K PPS.
RSM_MACRO_NORMAL_SPEED(hRsm, 1, AXIS_X, 0);
//Set symmetric T-curve for axis X
RSM_MACRO_SET_V(hRsm, 1, AXIS_X, 2000);
//set the speed of axis X to be 2000 PPS.
RSM_MACRO_SET_A(hRsm, 1, AXIS_X, 1000);
//set the acceleration of axis X to be 1000 PPS/Sec
RSM_MACRO_SET_SV(hRsm, 1, AXIS_X, 2000);
//set the start speed of axis X to be 2000 PPS.
RSM_MACRO_SET_AO(hRsm, 1, AXIS_X, 0);
//set the number of remaining offset pulses for axis X to be 0 PPS.
RSM_MACRO_SET_VAR(hRsm, 1, VAR1, 100); //VAR1 = 100。
RSM_MACRO_FOR(hRsm, 1, VAR1);
//enable axis X to move back and forward 1,000 Pulse, loop for 100 times.
//or input the instant operating value RSM_MACRO_FOR(hRsm, 1, 100).
RSM_MACRO_FIXED_MOVE(hRsm, 1, AXIS_X, 1000);
//move axis X to be 1000 pulses.
RSM_MACRO_FIXED_MOVE(hRsm, 1, AXIS_X, -1000);
//move axis X to be -1000 pulses.
RSM_MACRO_NEXT(hRsm, 1);
RSM_MACRO_MP_CLOSE(hRsm, 1);
// module 1 finish, Macro Program is written into i-8094H
//=========================================================
RSM_MP_CALL(hRsm, 1, MP100); //call module 1 i-8094H, execute MP #100.
```

9.5.2 結束指令循環區域敘述式(NEXT)

※ eRTU RSM_MACRO_NEXT (HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

設定結束巨集程序指令循環區域。請使用此函式作為循環區域的最後一行指令，與函式" RSM_MACRO_NEXT ()"作為第一行指令。

類別:

Modbus sub_function; MP.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

RSM_MACRO_NEXT (hRsm, 1);

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	0C D6	*Sub_funciton code*			

9.5.1 中止指令循環區域敘述式(EXIT FOR)

※ eRTU RSM_MACRO_EXIT_FOR(HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

在巨集程序指令循環區域內執行此函式會立即中止並離開循環。

類別:

Modbus sub_function; MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2)。

Modbus 範例:

RSM_MACRO_EXIT_FOR (hRsm, 1);

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	0C DE	*Sub_funciton code*			

9.6 條件判斷命令 (IF~ELSE)

9.6.1 條件判斷敘述式(IF)

※Δ eRTU RSM_MACRO_IF (HANDLE *hRsm*, BYTE *SlaveAddr*, long *varNo*, WORD *Operator*, long *varNo1*)

說明:

設定開始條件判斷執行區域與指定執行條件。請使用此函式作為執行區域的第一行指令，與函式" RSM_MACRO_END_IF ()"作為最後一行指令。此外可以在執行區域內加入函式"RSM_MACRO_ELSE()"作為相反條件執行區域的開始敘述式。請注意，巢狀層次勿超過七層。

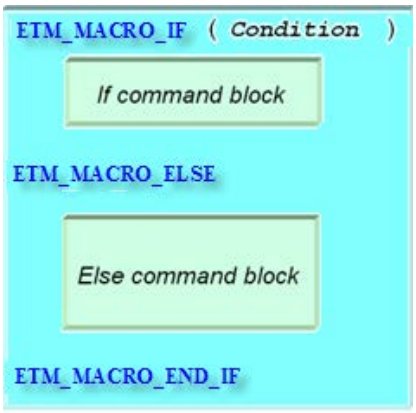


圖 6: RSM_MACRO_IF 應用架構

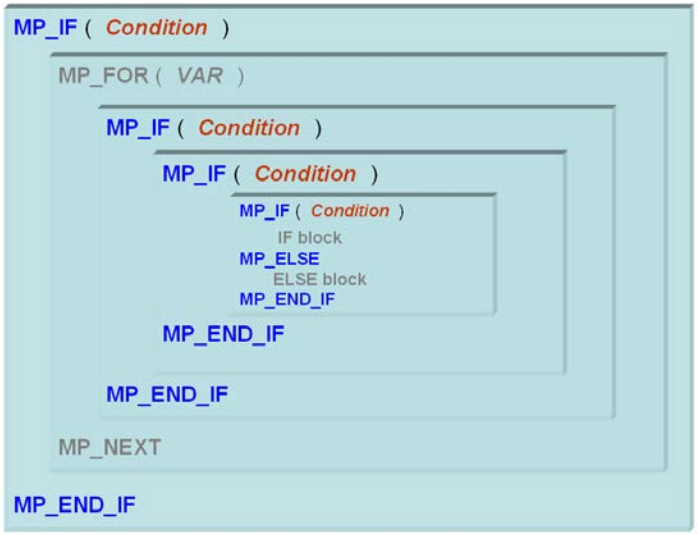


圖 7: 巢狀條件判斷

類別:

Modbus sub_function; MP and ISR.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
***varNo*:**	**第一個運算元的設定值** **範圍: -2,000,000,000 ~ +2,000,000,000 或巨集變數 (VAR0~VAR511)**
***Operator*:**	**比較運算子設定值** 字串 / 說明 "<" 小於 ">" 大於 "<=" 小於或等於 ">=" 大於或等於 "==" 等於 "!=" 不等於
***varNo1*:**	**第二個運算元的設定值** **範圍: -2,000,000,000 ~ +2,000,000,000 或巨集變數 (VAR0~VAR511)**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

運算子的 ASCII 碼

運算子	ASCII (dec)	ASCII (hex)
<	60	003C
<=	15676	3D3C
>	62	003E
>=	15678	3D3E
==	15677	3D3D
=	61	003D
!=	15649	3D21
!	33	0021

```
RSM_MACRO_IF (hRsm, 1, VAR1, ”<=”, VAR2);
//The double quotes (“”) is required for Operator parameter.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 06	0C
Register[]	**Value (hex)**	**Remarks**			
0	0C D7	*Sub_funciton code*			
1	7F FF	*MSW address of varNo*			
2	00 01	*LSW address of varNo*			
3	3C 3D	*Operator (0x3C3D ➔ "<=")*			
4	7F FF	*MSW address of varNo1*			
5	00 02	*LSW address of varNo1*			

相關範例:

```
RSM_MP_CREATE(hRsm, 1, MP100); // Write #100 Macro Program into i-8094H.
//=========================================================
// The following functions will not be executed, but will be written into
// i-8094H for further execution.
RSM_MACRO_SET_MAX_V(hRsm, 1, AXIS_X, 20000);
//The maximum speed of the axis X is 20K PPS.
RSM_MACRO_NORMAL_SPEED(hRsm, 1, AXIS_X, 0);
//Set symmetric T-curve for axis X.
RSM_MACRO_SET_V(hRsm, 1, AXIS_X, 2000);
//set the speed of axis X to be 2000 PPS.
RSM_MACRO_SET_A(hRsm, 1, AXIS_X, 1000);
//set the acceleration of axis X to  be 1000 PPS/sec
RSM_MACRO_SET_SV(hRsm, 1, AXIS_X, 2000);
//set the start speed of axis X to be 2000 PPS.
RSM_MACRO_SET_AO(hRsm, 1, AXIS_X, 0);
//set the number of remaining offset pulses for axis X to be 0 PPS.
//-------------------------------------------------------------------------------------------
RSM_MACRO_SET_VAR(hRsm, 1, VAR1, 100); //VAR1 = 100。
RSM_MACRO_SET_VAR(hRsm, 1, VAR2, 200); //VAR2 = 200。
RSM_MACRO_IF(hRsm, 1, VAR1, "<", VAR2);
RSM_MACRO_FIXED_MOVE(hRsm, 1, AXIS_X, 1000);
//command to move axis X for 1000 pulses.
RSM_MACRO_ELSE(hRsm, 1);
RSM_MACRO_FIXED_MOVE(hRsm, 1, AXIS_X, -1000);
// command to move axis X for -1000 pulses.
RSM_MACRO_END_IF(hRsm, 1);
RSM_MACRO_MP_CLOSE(hRsm, 1);
// module 1, Macro Program #100 is written into i-8094H
//=========================================================
RSM_MP_CALL(hRsm, 1, MP100);
// execute  Macro Program #100 of module 1.
```

9.6.2 條件判斷敘述式(ELSE)

※Δ eRTU RSM_MACRO_ELSE (HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

設定條件判斷執行區域內開始相反條件執行區域。

類別:

Modbus sub_function; MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

RSM_MACRO_ELSE (hRsm, 1);

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	0C D8	***Sub_funciton code***			

9.6.3 條件判斷敘述式(END IF)

※Δ eRTU RSM_MACRO_END_IF (HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

設定結束條件判斷執行區域。請使用此函式作為執行區域的最後一行指令，與函式”RSM_MACRO_IF()”作為第一行指令。

類別:

Modbus sub_function; MP and ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

RSM_MACRO_END_IF (hRsm, 1);

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	0C D9	*Sub_funciton code*			

9.7 跳躍指令 (GOTO-LABEL)

9.7.1 跳躍敘述式(GOTO)

※ eRTU RSM_MACRO_GOTO(HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *lableNo*)

說明:

執行此函式會使巨集程序的控制跳躍到指定的標籤位置。請使用函式"RSM_MP_LABEL()"設定標籤位置。

使用限制:

- 最多可指定四個標籤位置。
- 不可跳躍至指令循環區域內。
- 不可跳躍至條件判斷執行區域內。
- 不可跳躍至其他巨集程序的標籤位置。
- ISR巨集程序不支援此功能。

圖 8: 跳躍指令應用範例

類別:

Modbus sub_function; only inside MP Macro table allowed.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
labelNo:	**標籤位置號碼** **範圍: 0(LABEL0) ~ 3(LABEL3)**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

RSM_MACRO_GOTO (hRsm, 1, 3);

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0C DF	***Sub_funciton code***			
1	00 03	***labelNo (3 ➔0x03)***			

9.7.2 標籤位置敘述式(LABEL)

※ eRTU RSM_MACRO_LABEL(HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *lableNo*)

說明:

設定跳躍功能的標籤位置。請注意，標籤位置不可設定在指令循環區域與條件判斷執行區域內。

類別:

Modbus sub_function; only inside MP Macro table allowed.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
labelNo:	標籤位置號碼 範圍: 0(LABEL0) ~ 3(LABEL3)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

RSM_MACRO_LABEL (hRsm, 1, 3);

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0C E1	*Sub_funciton code*			
1	00 03	*labelNo (3 ➔0x03)*			

9.8 時間延遲(Timer)

※ eRTU RSM_MACRO_TIMER (HANDLE *hRsm*, BYTE *SlaveAddr*, long *varNo*)

說明:

執行此函式會暫停巨集程序並開啟時間計數器，當計數器到達指定的設定值後再繼續執行後續的巨集程序。

類別:

Modbus sub_function; for MP Macro table.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
varNo:	計數器設定值，單位:毫秒(milliseconds) 範圍: 0 ~ +2,000,000,000 ms 或巨集變數(VAR0~VAR511)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
RSM_MACRO_TIMER (hRsm, 1, 200);
//delay the execution of the function for 200ms.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 03	06
Register[]	**Value (hex)**	**Remarks**			
0	0C DA	*Sub_funciton code*			
1	00 00	*MSW of varNo*			
2	00 C8	*LSW of varNo (200 = 0xC8)*			

9.9 等待運動指令的執行

eRTU RSM_STOP_WAIT(HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

※ eRTU RSM_MACRO_STOP_WAIT(HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*)

說明:

在運動命令後接續執行此函式，可以確保運動命令執行的過程中不被新加入的運動指令覆蓋設定值，新加入的運動指令將等待現有的運動控制結束後再執行。

類別:

Modbus sub_function; RTC, MP.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表 4)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_STOP_WAIT 的Sub_function代碼為 0x0A DB。

RSM_MACRO_STOP_WAIT 的 Sub_function 代碼為 0x0C DB。

Modbus 範例:

RSM_STOP_WAIT (hRsm, 1, *AXIS_Y*);

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 02	04
Register[]	**Value (hex)**	**Remarks**			
0	0A DB/0C DB	*Sub_funciton code*			
1	00 02	*axis (2 ➔ AXIS_Y)*			

9.10 中止巨集程序

※Δ eRTU RSM_MACRO_EXIT_MACRO (HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

此函式可在巨集程序內的任意位置設定，當巨集編譯器執行此函式時會立即停止且結束執行中的巨集程序(MP或ISR)，若是在巢狀階層內執行則會結束該層巨集程序並返回上一階層程序(詳見圖9)。

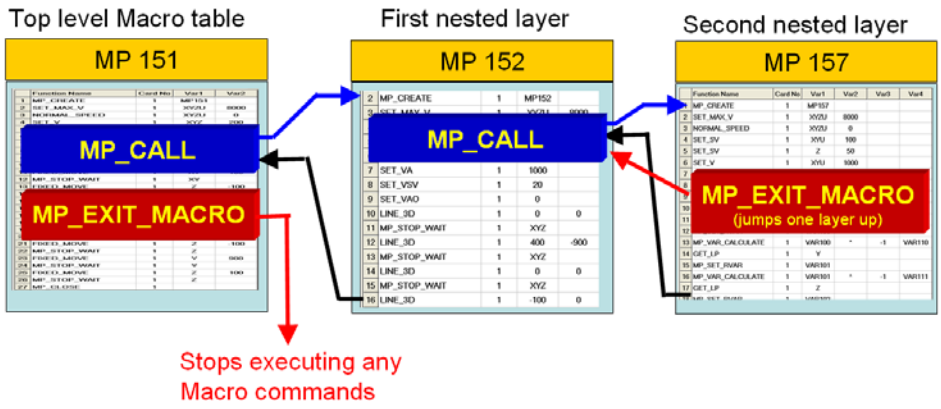


圖 9: 結束巨集程序

類別:

Modbus sub_function; MP and ISR.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

RSM_MACRO_EXIT_MACRO (hRsm, 1);

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	0C DD	***Sub_funciton code***			

9.11 中止所有巨集程序

eRTU RSM_MP_TERMINATE (HANDLE *hRsm*, BYTE *SlaveAddr*)

※Δ eRTU RSM_MACRO_MP_TERMINATE (HANDLE *hRsm*, BYTE *SlaveAddr*)

說明:

執行此函式會中止所有正在執行的MP與ISR巨集程序。請注意，此函式並不會停止運動中的軸，若有需要請執行停止運動的相關命令。

類別:

Modbus sub_function; RTC, MP, ISR.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

RSM_MP_TERMINATE (hRsm, 1);

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 01	02
Register[]	**Value (hex)**	**Remarks**			
0	0A E2/0C E2	*Sub_funciton code*			

9.12 取得巨集程序的下載狀態

```
eRTU RSM_GET_MP_DOWNLOAD_STATUS(HANDLE hRsm, BYTE
      SlaveAddr, MpDownloadInfo *pInfo)

typedef struct
{
     WORD MpNo;
     WORD IsrNo;
     WORD wCmd;
     WORD wLineNo;
     WORD wErrCode;
}MpDownloadInfo;
```

說明:

取得巨集程序的下載狀態與錯誤訊息。

類別:

Modbus table; RTC.

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
pInfo:	***MpNo*: 最後一次下載的 MP 巨集程序編號，若為 0 表示未下載 MP 巨集程序** ***IsrNo*: 最後一次下載的 ISR 巨集程序編號，若為 0 表示未下載 ISR 巨集程序** ***wCmd*: 造成錯誤的指令代碼(請參照章節 12.3 的指令代碼表)** ***wLineNo*: 巨集程序內的指令行數(由 0 開始計數第一行)** ***wErrCode*: 錯誤代碼** **- *0x00*: 無錯且巨集程序下載成功.** **- *0x01*: 下載的指令行數超出允許範圍** **IF 命令錯誤:** **- 0x02: 巢狀階層數量超出允許範圍(最多七層)** **- 0x03: 在條件判斷執行區域外使用 MP_ELSE 敘述式** **- *0x04*: 條件判斷執行區域的結束敘述式(MP_END_IF)無法找到匹配的開始敘述式(MP_IF)** **- *0x05*: 條件判斷執行區域的開始敘述式(MP_IF)無法找到匹配的結束敘述式(MP_END_IF)** **- *0x06*: 錯誤的條件運算子** **FOR命令錯誤:** **- 0x07: 巢狀階層數量超出允許範圍(最多七層)**

	- *0x08*: 在指令循環區域外使用 MP_EXIT_FOR 敘述式 - *0x09*: 指令循環區域的結束敘述式(MP_NEXT)無法找到匹配的開始敘述式(MP_FOR) - *0x0A*: 指令循環區域的開始敘述式(MP_FOR)無法找到匹配的結束敘述式(MP_NEXT) GOTO 命令錯誤: - *0x0B*: 重複使用相同的標籤號碼 - *0x0C*: 在條件判斷執行區域內使用標籤位置(MP_LABEL)敘述式 - *0x0D*: 在指令循環區域內使用標籤位置(MP_LABEL)敘述式 - 0x0E: 跳躍敘述式(MP_GOTO)無法找到匹配的標籤位置敘述式(MP_LABEL) - *0x0F*: 未定義的巨集命令(MP 或 ISR) - *0x10*: 錯誤的 MP_VAR_CALCULATE 運算元

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

Modbus 範例:

```
MpDownloadInfo MpInfo;
RSM_GET_MP_DOWNLOAD_STATUS(hRsm, 1, &MpInfo);
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01 (Card No.)	04	00 6E (110)	00 05

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	Registers(hex)
01	04	0A	See next table

Register []	Value (hex)	Remarks
0	00 02	*MpNo:* MP number (e.g. MP2)
1	00 00	*IsrNo:* ISR number (no ISR Macro has been downloded)
2	0A D6	*wCmd:* command code (here MP_NEXT)
3	00 09	*wLineNo*: command line within Macro table (line number 9)
4	00 09	*wErrCode:* error code describing the kind of error (here: MP_NEXT command outside MP_FOR block. The corresponding MP_FOR command is missing.)

9.13 動態改變運動速度

eRTU RSM_CHANGE_VEL_ON_FLY (HANDLE *hRsm*, BYTE *SlaveAddr*, BYTE *axis*, DWORD *data*)

說明:

在巨集程序運作的過程中，可在PC端執行此函式動態的改變運動速度。

類別:

Modbus sub_function; RTC.

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
axis:	軸號設定值 (請參照表4)
data:	驅動速度設定值，單位為PPS(Pulse Per Second) (最大設定值請參照函式"RSM_SET_MAX_V()"說明)

回傳值:

0: 成功; 其他: 失敗 (詳細說明請參考章節2.2) 。

備註:

RSM_CHANGE_VEL_ON_FLY的Sub_function代碼為0x0A 43。

Modbus 範例:

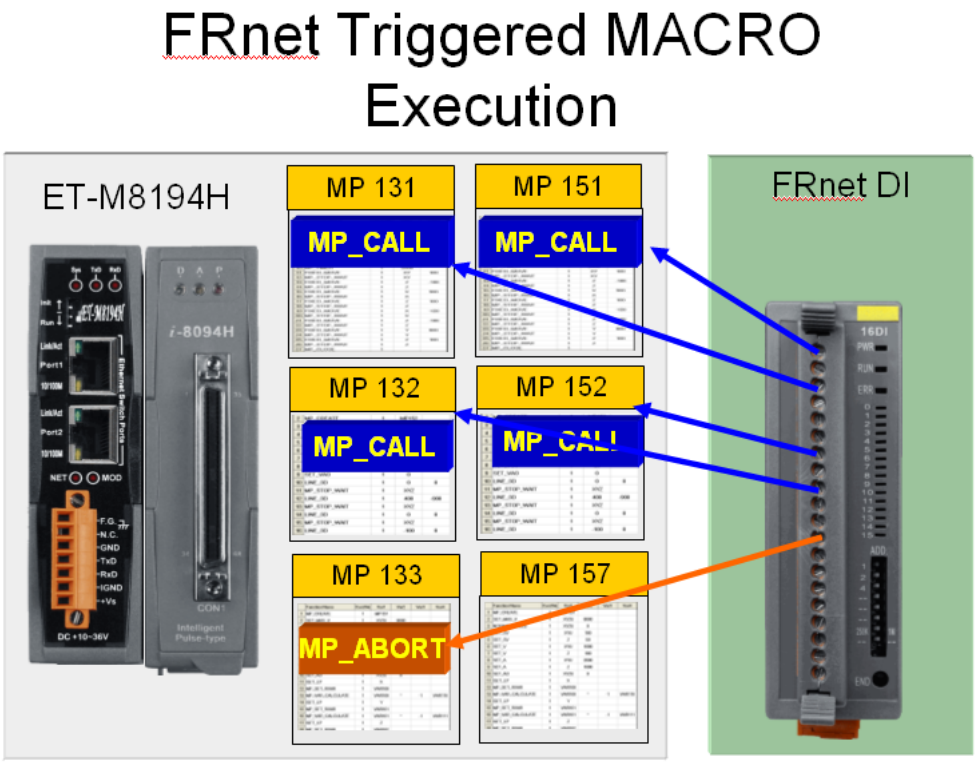
```
RSM_CHANGE_VEL_ON_FLY (hRsm, 1, AXIS_X, 120000L);
//set the speed for the X axis on module 1 to 120000 PPS.
```

所傳送的 Modbus 需求命令如下:

UID (hex)		FC (hex)	St_Addr. (hex)	Word Count (hex)	Byte Count (hex)
01		10	1F 40	00 04	08
Register[]	**Value (hex)**	**Remarks**			
0	0A 43	*Sub_function code*			
1	00 01	*axis (1 = AXIS_X)*			
2	00 01	*MSW of data*			
3	D4 C0	*LSW of data (120000 = 0x1D4C0)*			

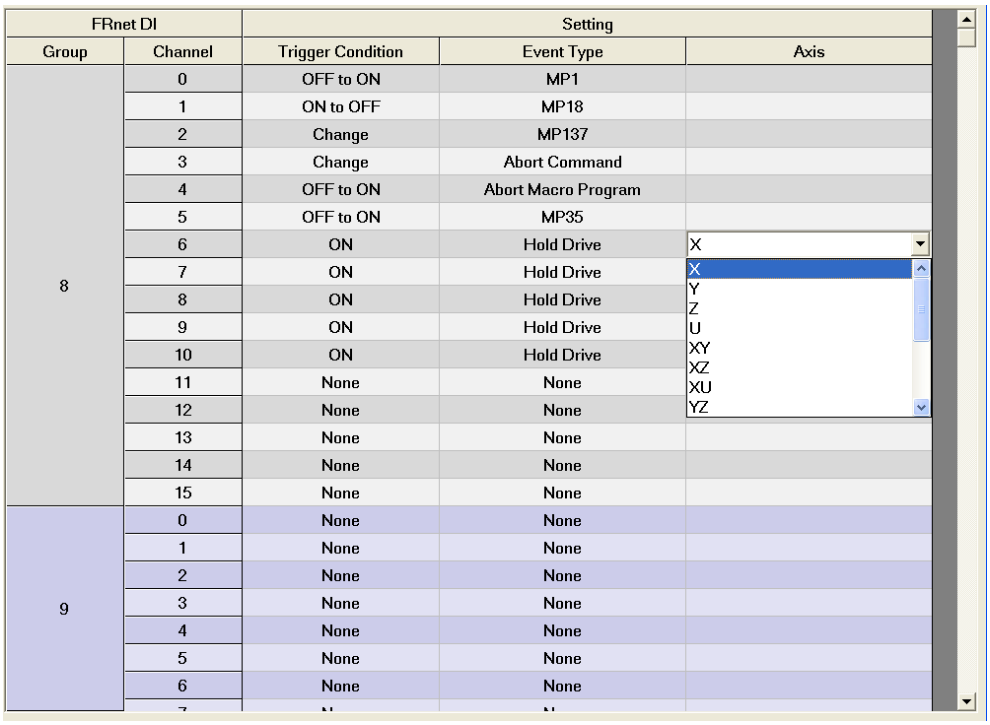
10 單機控制器

當 RS-M8194H 搭配 FRnet DI 使用時，可作為一個獨立的單機控制器，在不連接 Modbus Master(控制 PC)的狀態下被操作。各個 FRnet DI 可以分別指定觸發執行不同的 RS-M8194H 巨集程序。請注意，同時間內只允許一個巨集程序執行。當已經有巨集程序正在執行的狀態下，此時若有其他的巨集程序被 FRnet DI 觸發，則會等待正在執行的巨集程序結束後再開始執行。

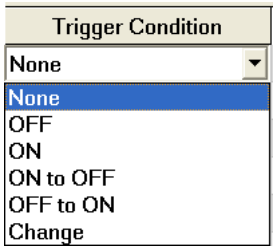


相關的設定流程請參照下列的說明步驟:

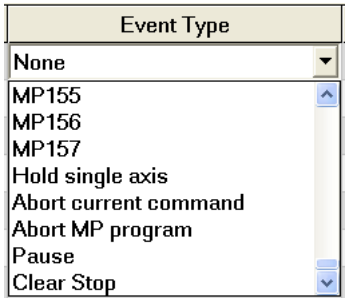
- 步驟 1:
  編輯巨集程序並且將其下載至 RS-M8194H(詳細請參考章節 9)
- 步驟 2:
  使用工具軟體 EzMove 設定 FRnet DI 通道所要觸發的巨集程序
  (a) 連線 RS-M8194H。
  (b) 開啟"FRnet DI Setting"視窗(Setting → FRnet DI...)。



(c) 選擇觸發的訊號狀態。



(d) 選擇觸發的巨集程序或事件，若不使用此 DI 通到請設置“None”

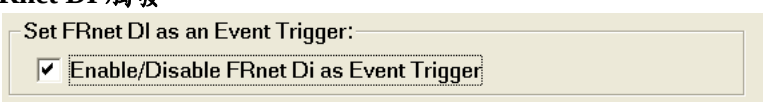


- **步驟 3:**

  點擊“Send Table”按鈕將設定值寫入 RS-M8194H。

- **步驟 4:**

  勾選”Enable/Disable FRnet Di as Event Trigger”。 此時 RS-M8194H 已經準備好等待 FRnet DI 觸發。



11 版本資訊

11.1 RS-M8194H 韌體版本

eRTU RSM_GET_RSM_FIRMWARE_VERSION (HANDLE *hRsm*, BYTE *SlaveAddr* , DWORD* *RSM_Ver*);

說明:

取得ET-8194H(P824)的韌體版本。

參數:

參數名稱	說明
hRsm:	**連接埠 handle**
SlaveAddr:	**Modbus RTU Slave 位址 (範圍: 1 ~ 247)**
RSM_Ver	**版本號碼** **範例: 0x03000000 → Version 3.0**

Modbus 範例:

```
DWORD RSM8194H_Ver;
HANDLE hRsm;
RSM_GET_RSM_FIRMWARE_VERSION(hRsm, & RSM8194H_Ver);
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	04	00 52	00 02

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	*Version*(hex) High Word	*Version*(hex) Low Word
01	04	04	02 00	00 00

11.2 i-8094H 韌體版本

eRTU RSM_GET_i8094H_FIRMWARE_VERSION (HANDLE *hRsm*, BYTE *SlaveAddr* , DWORD* *i8094H_Ver*);

說明:

取得i-8094H的韌體版本。

參數:

參數名稱	說明
hRsm:	連接埠 handle
SlaveAddr:	Modbus RTU Slave 位址 (範圍: 1 ~ 247)
i8094H_Ver	版本號碼 範例: 0x02210300 ▪ 高字組表示 PCB 2.21 ▪ 低字組表示 i-8094H 的韌體版本為 3.00

Modbus 範例:

```
DWORD i8094H_Ver;
HANDLE hRsm;
RSM_GET_i8094H_FIRMWARE_VERSION(hRsm, &i8094H_Ver);
```

所傳送的 Modbus 需求命令如下:

UID (hex)	FC (hex)	St_Addr. (hex)	Word Count (hex)
01	04	00 54	00 02

回應的 Modbus 訊息可能如下:

UID (hex)	FC (hex)	Byte Count(hex)	*Version*(hex) High Word	*Version*(hex) Low Word
01	04	04	02 21	02 01

11.3 RS-M8194H 函式庫(DLL)版本

```
WORD RSM_GET_DLL_VERSION (void);
```

說明:

取得RS-M8194H的函式庫(DLL)版本號碼。使用此函式無須連接RS-M8194H。假設取回的數值為0x100則表示版本號碼為1.00，高位元組用來表示主要版本號碼，低位元組則是用來表示次要版本號碼。

12 附錄

12.1 Input 資料表

使用 Modbus 功能碼 4 讀取 input 資料。在本章所描述的位址皆採用 0 為基準(zero-base)的索引方式，暫存器表的第一個位址將由 0 開始，但有部分的 PLC 與 HMI 採用 1 為基準(one-base)的索引方式，也就是說使用位址 1 做為第一個位址，此時因為索引方式的差異，在操作暫存器時必須特別注意設定正確的對應位址。

PLC 定義位址 3XXXX 用來操作 input 暫存器表，其中 XXXX 為操作的暫存器位址，且位址 1 被定義為第一個位址，假設操作位址 30001，則實際將對應操作下表所描述的位址 0 暫存器。

當資料需使用兩個暫存器來表示時，可讀取目前的字組順序(WORD order)設定值，依此判斷高字組 (Most Significant WORD, MSW)與低字組 (Least Significant WORD, LSW)所屬的暫存器。字組順序設定值被定義在 holding 暫存器的位址 14，當字組順序設定為 0 時，將使用第一個暫存器表示高字組，第二個暫存器表示低字組。

暫存器類型:

- R: 只可讀取。
- W: 只可寫入。
- R/W : 可讀取與寫入。

12.1.1 讀取 FRnet DI/O

附錄表 1: FRnet DI/O (PLC 位址: 30001 ~ 30016)

讀取 FRnet DIO				
變數名稱		**位址 (zero based)**	**類型**	**說明**
FRnet DO	群組 0	0	R	讀取 FRnet DO 群組 0 (16 位元)
	群組 1	1	R	讀取 FRnet DO 群組 1 (16 位元)
	群組 2	2	R	讀取 FRnet DO 群組 2 (16 位元)
	群組 3	3	R	讀取 FRnet DO 群組 3 (16 位元)
	群組 4	4	R	讀取 FRnet DO 群組 4 (16 位元)
	群組 5	5	R	讀取 FRnet DO 群組 5 (16 位元)
	群組 6	6	R	讀取 FRnet DO 群組 6 (16 位元)
	群組 7	7	R	讀取 FRnet DO 群組 7 (16 位元)
FRnet DI	群組 8	8	R	讀取 FRnet DI 群組 8 (16 位元)
	群組 9	9	R	讀取 FRnet DI 群組 9 (16 位元)
	群組 10	10	R	讀取 FRnet DI 群組 10 (16 位元)
	群組 11	11	R	讀取 FRnet DI 群組 11 (16 位元)
	群組 12	12	R	讀取 FRnet DI 群組 12 (16 位元)
	群組 13	13	R	讀取 FRnet DI 群組 13 (16 位元)
	群組 14	14	R	讀取 FRnet DI 群組 14 (16 位元)
	群組 15	15	R	讀取 FRnet DI 群組 15 (16 位元)

12.1.2 讀取運動控制晶片的 DI 狀態

附錄表 2: DI 狀態 (PLC 位址: 30017 ~ 30020)

讀取 DI 狀態 (子板)				
變數名稱		**位址 (zero baed)**	**類型**	**說明**
DI ALL	X-Axis	16	R	X 軸的所有 DI 狀態，若要讀取單一狀態，請讀取 X_DI_0 ~ X_DI_9
	Y-Axis	17	R	Y 軸的所有 DI 狀態，若要讀取單一狀態，請讀取 Y_DI_0 ~ Y_DI_9
	Z-Axis	18	R	Z 軸的所有 DI 狀態，若要讀取單一狀態，請讀取 Z_DI_0 ~ Z_DI_9
	U-Axis	19	R	U 軸的所有 DI 狀態，若要讀取單一狀態，請讀取 U_DI_0 ~ U_DI_9

12.1.3 讀取錯誤碼

附錄表 3: 錯誤碼 (PLC 位址: 30021~ 30026)

讀取錯誤碼				
變數名稱		**位址 (zero baed)**	**類型**	**說明**
ERROR CODE	X-Axis	20	R	X 軸錯誤發生，請使用函式 RSM_GET_ERROR_CODE()取得更多訊息
	Y-Axis	21	R	Y 軸錯誤發生，請使用函式 RSM_GET_ERROR_CODE()取得更多訊息
	Z-Axis	22	R	Z 軸錯誤發生，請使用函式 RSM_GET_ERROR_CODE()取得更多訊息
	U-Axis	23	R	U 軸錯誤發生，請使用函式 RSM_GET_ERROR_CODE()取得更多訊息
MP Call number		24	R	目前正在執行的巨集編號
EMERGENCY_STOP state		25	R	顯示目前的緊急停止狀態 0: 軟體緊急停止未觸發 1: 軟體緊急停止已觸發

12.1.4 讀取邏輯與編碼器位置、加速度、速度

附錄表 4: 邏輯與編碼器位置、加速度、速度 (PLC 位址: 30027~ 30058)

運動狀態				
變數名稱		**位址 (zero baed)**	**類型**	**說明**
Logic Position	X	26-27	R	X 軸的邏輯位置(LP)
	Y	28-29	R	Y 軸的邏輯位置(LP)
	Z	30-31	R	Z 軸的邏輯位置(LP)
	U	32-33	R	U 軸的邏輯位置(LP)
Encoder Position	X	34-35	R	X 軸的編碼器位置(EP)
	Y	36-37	R	Y 軸的編碼器位置(EP)
	Z	38-39	R	Z 軸的編碼器位置(EP)
	U	40-41	R	U 軸的編碼器位置(EP)
Current velocity	X	42-43	R	X 軸的當前速度
	Y	44-45	R	Y 軸的當前速度
	Z	46-47	R	Z 軸的當前速度
	U	48-49	R	U 軸的當前速度
Current Acceleration	X	50-51	R	X 軸的當前加速度
	Y	52-53	R	Y 軸的當前加速度
	Z	54-55	R	Z 軸的當前加速度
	U	56-57	R	U 軸的當前加速度

12.1.5 讀取停止狀態

附錄表 5: 停止狀態 (PLC 位址: 30059 ~ 30062)

變數名稱		位址 (zero baed)	類型	說明
Stop Status	X	58	R	1:停止; 0: 移動中
	Y	59	R	1:停止; 0: 移動中
	Z	60	R	1:停止; 0: 移動中
	U	61	R	1:停止; 0: 移動中

12.1.6 讀取位置栓鎖值(LATCH)

附錄表 6: 讀取同動功能的位置栓鎖數值(PLC 位址: 30063 ~ 30070)

變數名稱		位址 (zero baed)	類型	說明
Latch	X	62-63	R	X 軸的位置栓鎖數值(資料型態 long)
	Y	64-65	R	Y 軸的位置栓鎖數值(資料型態 long)
	Z	66-67	R	Z 軸的位置栓鎖數值(資料型態 long)
	U	68-69	R	U 軸的位置栓鎖數值(資料型態 long)

12.1.7 讀取錯誤狀態與指令緩衝區剩餘空間

附錄表 7: 錯誤狀態與指令緩衝區剩餘空間 (PLC 位址: 30071~ 30072)

變數名稱	位址 (zero baed)	類型	說明
ERROR_STATE	70	R	1: 發生錯誤; 0: 無錯誤
FREE_BUFFER_SIZE	71	R	指令緩衝區的剩餘空間，最大值為 30

12.1.8 讀取 i-8094H 中斷

附錄表 8: i-8094H 中斷(PLC 位址: 30073~ 30081)

i-8094H 中斷 RS-M8194H (回傳中斷 RINT)			
變數名稱	**位址 (zero baed)**	**類型**	**說明**
RINT_STATE_ALL	72	R	讀取 RINT 狀態
Line_Scan_Completed	73	R	RINT_STATE 的位元 0
MP_Completed	74	R	RINT_STATE 的位元 1
User-Defined_RINT	75	R	RINT_STATE 的位元 2
Continuous_Inp_Interrupt	76	R	RINT_STATE 的位元 3
…..Undefined	77	--	0
…..Undefined	78	--	0
Axes_Error	79	R	RINT_STATE 的位元 6
Module_Error	80	R	RINT_STATE 的位元 7

12.1.9 讀取韌體版本

附錄表 9: 韌體版本 (PLC 位址: 30082 ~ 30087)

其他			
變數名稱	**位址 (zero baed)**	**類型**	**說明**
i-8094H Module ID	81	R	i-8094H：0x44 i-8094A：0x55
RS-M8194H Firmware Version	82-83	R	0x01000000 表示 版本：1.00
i-8094H Firmware Version	84-85	R	▪ 高字組表示 PCB 版本 ▪ 低字組表示 i-8094H 韌體版本: // 範例: 0x02210201 ▪ 高字組 o PCB 版本 2.21 ▪ 低字組 o 0201->02 主版本，01-> 次要版本號碼

12.1.10 讀取子板的 DI 訊號狀態

附錄表 10: 子板的 DI 訊號狀態 (PLC 位址: 30017~ 30020)

讀取 DI 狀態(子板)				
變數名稱		**位址 (zero baed)**	**類型**	**說明**
DI ALL	X-Axis	88	R	X 軸的所有 DI 訊號，若要讀取單一訊號，請讀取 X_DI_SIG_0 ~ X_DI_SIG _7
	Y-Axis	89	R	Y 軸的所有 DI 訊號，若要讀取單一訊號，請讀取 Y_DI_SIG_0 ~ Y_DI_SIG _7
	Z-Axis	90	R	Z 軸的所有 DI 訊號，若要讀取單一訊號，請讀取 Z_DI_SIG_0 ~ Z_DI_SIG _7
	U-Axis	91	R	U 軸的所有 DI 訊號，若要讀取單一訊號，請讀取 U_DI_SIG_0 ~ U_DI_SIG _7

12.1.11 讀取絕對邏輯位置

附錄表 11: 絕對邏輯位置 (PLC 位址: 30093 ~ 30100)

運動狀態				
變數名稱		位址 (zero baed)	類型	說明
Absolute Logic Position	X	160-161	R	X 軸的絕對邏輯位置
	Y	162-163	R	Y 軸的絕對邏輯位置
	Z	164-165	R	Z 軸的絕對邏輯位置
	U	166-167	R	U 軸的絕對邏輯位置

12.1.12 讀取 RS-M8194H 與 i-8094H 狀態

附錄表 12: RS-M8194H 與 i-8094H 狀態 (PLC 位址: 30101 ~ 30103)

RS-M8194H 與 i-8094H 的當前狀態			
變數名稱	**位址 (zero baed)**	**類型**	**說明**
RS-M8194H state	100	R	RS-M8194H 目前的狀態: 0- 準備完成可接收 RTC 命令 1- RSM 進入巨集下載狀態 (MPn) 2- RSM 進入巨集下載狀態 (ISRn)
i-8094H state	101	R	i-8094H 模組的當前狀態: 3- i-8094H 已準備完成，可接收 RTC 命令 4- 執行初始化表(IT 命令) 5- 初始化完成 6- 正在下載巨集命令到 FRAM 或正在等待下一個巨集命令 7- 巨集下載完成 8- 正在執行 MP 巨集程序 9- MP 巨集程序執行完畢 10- 正在執行 ISR 巨集程序 11- ISR 巨集程序執行完畢 255- 韌體啟動中
Illegal Modbus function	102	R	函式操作錯誤時的 Modbus 例外回應 0- 無錯誤 1- 不支援的功能碼 2- 參數設定值超出允許範圍 3- 未定義的命令 4- 超出巨集程序允許設定的最大指令行數 (MPn 或 ISRn) 5- 在 ISR 巨集程序內使用 MP 指令 6- 在 MP 巨集程序內使用 ISR 指令 7- 在巨集程序內使用 RTC 指令 8- 在巨集程序外使用巨集指令 0xB1 - 超出 MP_IF 所允許的最大巢狀階層數量 0xB2 -在 MP_IF 區域外使用 MP_ELSE 敘述式 0xB3 -在 MP_IF 區域外使用 MP_END_IF 敘述式

			0xB4 - MP_IF 無法找到匹配的 MP_END_IF 結束敘述式 0xB5 -超出 MP_FOR 所允許的最大巢狀階層 0xB6 -在 MP_FOR 區域外使用 MP_EXIT_FOR 敘述式 0xB7 -在 MP_FOR 區域外使用 MP_NEXT 敘述式 0xB8 - MP_FOR 無法找到匹配的 MP_NEXT 結束敘述式 0xB9 -重複使用相同的標籤號碼 0xBA -在 MP_IF 區域內使用 MP_LABEL 敘述式 0xBB -在 MP_FOR 區域內使用 MP_LABEL 敘述式 0xBC - MP_GOTO 敘述式無法找到匹配的 MP_LABEL 敘述式 0xC0 - 使用一個暫存器讀取 DWORD 資料 0xC1 - i-8094H 的 DPRAM 已滿 0xC2 - 插槽內未正確安裝模組 0xC3 - i-8094H 正在執行巨集程序

12.1.13 巨集程序的下載錯誤訊息

附錄表 13: 程序的下載錯誤訊息 (PLC 位址: 30111 ~ 30115)

巨集程序下載錯誤訊息			
變數名稱	**位址 (zero based)**	**類型**	**說明**
Macro MP number (MPxx)	110	R	RS-M8194H 開啟電源後的最後一次下載的 MP 巨集程序編號
Macro ISR number (ISRxx)	111	R	RS-M8194H 開啟電源後的最後一次下載的 ISR 巨集程序編號
Command type	112	R	造成錯誤的指令代碼(請參照章節 12.3 的指令代碼表)
Line number	113	R	巨集程序中發生錯誤的指令行號
Error type	114	R	描述錯誤類型

12.1.14 巨集程序的執行錯誤訊息

附錄表 14: 巨集程序的執行錯誤訊息 (PLC 位址: 30116~ 30120)

巨集程序的執行錯誤訊息			
變數名稱	**位址 (zero based)**	**類型**	**說明**
Macro MP number (MPxx)	115	R	RS-M8194H 開啟電源後的最後一次下載的 MP 巨集程序編號
Macro ISR number (ISRxx)	116	R	RS-M8194H 開啟電源後的最後一次下載的 ISR 巨集程序編號
Command type	117	R	造成錯誤的指令代碼(請參照章節 12.3 的指令代碼表)
Line number	118	R	巨集程序中發生錯誤的指令行號
Error type	119	R	描述錯誤類型

12.1.15 運動控制晶片的觸發中斷

附錄表 15: 運動控制晶片的觸發中斷 (PLC 位址: 30131 ~ 30140)

運動控制晶片的觸發中斷				
nINT	**符號**	**位址 (zero based)**	**類型**	**說明**
0	PULSE	130	R	控制脈波上升時
1	P>=C-	131	R	邏輯/編碼器位置計數器的數值大於或等於比較器 COMP-的設定值
2	P<C-	132	R	邏輯/編碼器位置計數器的數值小於比較器 COMP-的設定值
3	P<C+	133	R	邏輯/編碼器位置計數器的數值小於比較器 COMP+的設定值
4	P>=C+	134	R	邏輯/編碼器位置計數器的數值小於比較器 COMP+的設定值
5	C-END	135	R	在驅動過程中，等速度區段結束的時候
6	C-STA	136	R	在驅動過程中，等速度區段開始的時候
7	D-END	137	R	驅動結束(移動完成)
8	HMEND	138	R	自動原點復歸動作中斷
9	SYNC	139	R	同步運動被觸發

12.2 Holding 資料表

使用 Modbus 功能碼 6 或 16 寫入 holding 暫存器，使用功能碼 3 讀取暫存器。在本章所描述的位址皆採用 0 為基準(zero-base)的索引方式，暫存器表的第一個位址將由 0 開始，但有部分的 PLC 與 HMI 採用 1 為基準(one-base)的索引方式，也就是說使用位址 1 做為第一個位址，此時因為索引方式的差異，在操作暫存器時必須特別注意設定正確的對應位址。

PLC 定義位址 4XXXX 用來操作 holding 暫存器表，其中 XXXX 為操作的暫存器位址，且位址 1 被定義為第一個位址，假設操作位址 40001，則實際將對應操作下表所描述的位址 0 暫存器。

當資料需使用兩個暫存器來表示時，可讀取目前的字組順序(WORD order)設定值，依此判斷高字組 (Most Significant WORD, MSW)與低字組 (Least Significant WORD, LSW)所屬的暫存器。字組順序設定值被定義在 holding 暫存器的位址 14，當字組順序設定為 0 時，將使用第一個暫存器表示高字組，第二個暫存器表示低字組。

暫存器類型:

- R: 只可讀取。
- W: 只可寫入。
- R/W : 可讀取與寫入。

12.2.1 FRnet DO

附錄表 16: FRnet DO (PLC 位址: 40001 ~ 40008)

FRnet DO				
變數名稱		位址 (zero based)	類型	說明
FRnet DO	Group 0	0	R/W	設定/讀取 FRnet DO 群組 0
	Group 1	1	R/W	設定/讀取 FRnet DO 群組 1
	Group 2	2	R/W	設定/讀取 FRnet DO 群組 2
	Group 3	3	R/W	設定/讀取 FRnet DO 群組 3
	Group 4	4	R/W	設定/讀取 FRnet DO 群組 4
	Group 5	5	R/W	設定/讀取 FRnet DO 群組 5
	Group 6	6	R/W	設定/讀取 FRnet DO 群組 6
	Group 7	7	R/W	設定/讀取 FRnet DO 群組 7

12.2.2 巨集呼叫、FRnet 事件與字組順序設定

附錄表 17: 巨集呼叫、FRnet 事件與字組順序設定(PLC 位址: 40009 ~ 40015)

變數名稱	位址 (zero based)	類型	說明
CALL_MPn	8	R/W	在此位址寫入巨集程序的編號，將會呼叫並執行對應的巨集程序 n=1-157
FRNet Event	9	--	開啟使用 FRnet DI 產生事件功能 0 – 關閉 FRnet 事件觸發 1 – 開啟 FRnet 事件觸發
Reserved	10-13	--	未定義
WORD Order	14	R/W	設定 DWORD、 long、 float 變數的字組順序 0: Big Endian WORD (預設) 第一個暫存器為 MSW，第二個暫存器為 LSW 1: Small Endian WORD 第二個暫存器為 MSW

12.2.3 IP 位址設定

附錄表 18: IP 位址設定 (PLC 位址: 40021 ~ 40031)

IP 位址設定			
變數名稱	**位址 (zero based)**	**類型**	**說明**
Lock_IP	20 ~ 23	R/W	鎖定/讀取鎖定 TCP Client IP
UnLock_IP	24	W	解開鎖定 TCP Client IP
RS-M8194H_IP	25~26	R/W	設定 RS-M8194H IP
RS-M8194H_Mask	27~28	R/W	設定 RS-M8194H Mask
RS-M8194H_Gateway	29~30	R/W	設定 RS-M8194H Gateway

12.2.4 讀取與寫入邏輯或編碼器位置、加速度和速度

附錄表 19: 讀取與寫入邏輯或編碼器位置、加速度和速度(PLC 位址: 40091 ~ 40122)

運動狀態				
變數名稱		**位址 (zero based)**	**類型**	**說明**
Absolute Position	X-Axis	82-83	R/W	X 軸的絕對位置
	Y-Axis	84-85	R/W	Y 軸的絕對位置
	Z-Axis	86-87	R/W	Z 軸的絕對位置
	U-Axis	88-89	R/W	U 軸的絕對位置
Logic Position (LP)	X-Axis	90-91	R/W	X 軸的邏輯位置
	Y-Axis	92-93	R/W	Y 軸的邏輯位置
	Z-Axis	94-95	R/W	Z 軸的邏輯位置
	U-Axis	96-97	R/W	U 軸的邏輯位置
Encoder Position (EP)	X-Axis	98-99	R/W	X 軸的編碼器位置
	Y-Axis	100-101	R/W	Y 軸的編碼器位置
	Z-Axis	102-103	R/W	Z 軸的編碼器位置
	U-Axis	104-105	R/W	U 軸的編碼器位置
Current Velocity (CV)	X-Axis	106-107	R	X 軸目前的速度
	Y-Axis	108-109	R	Y 軸目前的速度
	Z-Axis	110-111	R	Z 軸目前的速度
	U-Axis	112-113	R	U 軸目前的速度
Current Acceleration (CA)	X-Axis	114-115	R	X 軸目前的加速度
	Y-Axis	116-117	R	Y 軸目前的加速度
	Z-Axis	118-119	R	Z 軸目前的加速度
	U-Axis	120-121	R	U 軸目前的加速度

12.2.5 讀取與寫入 VAR 變數

附錄表 20: VAR 變數 (PLC 位址: 40301 ~ 41324)

巨集變數			
變數名稱	**位址 (zero based)**	**類型**	**說明**
VAR0	300-301	R/W	VAR0 的設定值
VAR1	302-303	R/W	VAR1 的設定值
…	…		VARn = 300+2*n ~ 300+2*n+1
VAR510	1320-1321	R/W	VAR510 的設定值
VAR511	1322-1323	R/W	VAR511 的設定值

12.2.6 Sub_Function 代碼定義

附錄表 21: Sub_Function 代碼定義 (PLC 位址: 48001 ~ 48101)

運動控制命令(功能碼 16)			
變數名稱	**位址 (zero based)**	**類型**	**說明**
Sub_Function_Code	8000	W	Sub_Function 編號，請參照章節 12.3 函式列表
Reg1	8001	W	使用者可以透過 Sub_Function 執行大部分的運動控制功能，特別是與設定相關的指令 請依照使用的 Sub_Function 設定對應所需的參數，詳細設定方式可參考各函式說明中的範例 各種資料型態的參數所需的暫存器數量: BYTE ➔ 1 個暫存器 (low-byte) WORD ➔ 1 個暫存器 DWORD ➔ 2 個暫存器 long ➔ 2 個暫存器 float ➔ 2 個暫存器
Reg2	8002	W	
Reg3	8003	W	
Reg4	8004	W	
…			
Reg100	8100	W	

12.3 Sub_Function 代碼 對應表

請使用功能碼 4 呼叫 Sub_Function，並且設定起始位址為 8000(zero-based)，各個命令的編號、暫存器數量與所屬的章節請參閱下表說明。在編碼欄位標示”xxxxxx”表示此命令不提供 Sub_Function，例如取得資料的命令，若需要執行取值命令請使用功能碼 3 或功能碼 4，相關說明請參照章節 12.1 與 12.2。

附錄表 22: Sub_Function 代碼對應表

章節	Sub_Function	編號 (Hex) reg(0)	編號 (Dec) reg(0)	暫存器數量
2.1.1	RSM_OPEN_COM_PORT	xxxxxx	xxxxxx	x
2.1.2	RSM_CLOSE_COM_PORT	xxxxxx	xxxxxx	x
2.1.3	RSM_CONNECTION_STATE	xxxxxx	xxxxxx	x
2.1.4	RSM_SET_TIMEOUT	xxxxxx	xxxxxx	x
2.1.5	RSM_GET_TIMEOUT	xxxxxx	xxxxxx	x
2.1.6	RSM_SET_PACKET_DELAY	xxxxxx	xxxxxx	x
2.1.7	RSM_GET_PACKET_DELAY	xxxxxx	xxxxxx	x
2.1.8	RSM_SET_WORD_ORDER	xxxxxx	xxxxxx	x
2.1.9	RSM_GET_WORD_ORDER	xxxxxx	xxxxxx	x
3.2.1	RSM_RESET_CARD	0x0A0A	2570	1
3.2.2	RSM_CLEAR_CARD_BUFFER	0x0A0B	2571	1
3.3	RSM_SET_PULSE_MODE	0x0A0C	2572	3
3.3	RSM_MACRO_SET_PULSE_MODE	0x0C0C	3084	3
3.4	RSM_SET_MAX_V	0x0A0D	2573	4
3.4	RSM_MACRO_SET_MAX_V	0x0C0D	3085	4
3.5	RSM_SET_HLMT	0x0A0E	2574	4
3.5	RSM_MACRO_SET_HLMT	0x0C0E	3086	4
3.6	RSM_LIMITSTOP_MODE	0x0A0F	2575	3
3.6	RSM_MACRO_LIMITSTOP_MODE	0x0C0F	3087	3
3.7	RSM_SET_NHOME	0x0A10	2576	3
3.7	RSM_MACRO_SET_NHOME	0x0C10	3088	3
3.8	RSM_SET_HOME_EDGE	0x0A11	2577	3
3.8	RSM_MACRO_SET_HOME_EDGE	0x0C11	3089	3
3.9.1	RSM_SET_SLMT	0x0A12	2578	7
3.9.1	RSM_MACRO_SET_SLMT	0x0C12	3090	7
3.9.2	RSM_CLEAR_SLMT	0x0A13	2579	2
3.9.2	RSM_MACRO_CLEAR_SLMT	0x0C13	3091	2

章節	Sub_Function	編號 (Hex) reg(0)	編號 (Dec) reg(0)	暫存器數量
3.10	RSM_SET_ENCODER	0x0A14	2580	5
3.10	RSM_MACRO_SET_ENCODER	0x0C14	3092	5
3.11.1	RSM_SERVO_ON	0x0A15	2581	2
3.11.1	RSM_MACRO_SERVO_ON	0x0C15	3093	2
3.11.2	RSM_SERVO_OFF	0x0A16	2582	2
3.11.2	RSM_MACRO_SERVO_OFF	0x0C16	3094	2
3.12	RSM_SET_ALARM	0x0A17	2583	4
3.12	RSM_MACRO_SET_ALARM	0x0C17	3095	4
3.13	RSM_SET_INPOS	0x0A18	2584	4
3.13	RSM_MACRO_SET_INPOS	0x0C18	3096	4
3.14	RSM_SET_FILTER	0x0A19	2585	4
3.14	RSM_MACRO_SET_FILTER	0x0C19	3097	4
3.15.1	RSM_VRING_ENABLE	0x0A1A	2586	4
3.15.1	RSM_MACRO_VRING_ENABLE	0x0C1A	3098	4
3.15.2	RSM_VRING_DISABLE	0x0A1B	2587	2
3.15.2	RSM_MACRO_VRING_DISABLE	0x0C1B	3099	2
3.16.1	RSM_AVTRI_ENABLE	0x0A1C	2588	2
3.16.1	RSM_MACRO_AVTRI_ENABLE	0x0C1C	3100	2
3.16.2	RSM_AVTRI_DISABLE	0x0A1D	2589	2
3.16.2	RSM_MACRO_AVTRI_DISABLE	0x0C1D	3101	2
3.17.1	RSM_EXD_MP	0x0A1E	2590	4
3.17.2	RSM_EXD_FP	0x0A1F	2591	4
3.17.3	RSM_EXD_CP	0x0A20	2592	4
3.17.4	RSM_EXD_DISABLE	0x0A21	2593	2
3.18.1	RSM_READ_bVAR	xxxxxx	xxxxxx	x
3.18.2	RSM_WRITE_bVAR	0x0A23	2595	3
3.18.3	RSM_READ_VAR	xxxxxx	xxxxxx	x
3.18.4	RSM_WRITE_VAR	0x0A25	2597	5
3.19.1	RSM_READ_MD	xxxxxx	xxxxxx	x
3.19.2	RSM_WRITE_MD	0x0A27	2599	7
4.1.1	RSM_SET_LP	0x0A28	2600	4
4.1.1	RSM_MACRO_SET_LP	0x0C28	3112	4
4.1.2	RSM_GET_LP	xxxxxx	xxxxxx	x
4.1.2	RSM_MACRO_GET_LP	0x0C29	3113	2
4.1.2	RSM_GET_LP_4_AXIS	xxxxxx	xxxxxx	x
4.2.1	RSM_SET_EP	0x0A2A	2602	4

章節	Sub_Function	編號 (Hex) reg(0)	編號 (Dec) reg(0)	暫存器數量
4.2.1	RSM_MACRO_SET_EP	0x0C2A	3114	4
4.2.2	RSM_GET_EP	xxxxxx	xxxxxx	x
4.2.2	RSM_MACRO_GET_EP	0x0C2B	3115	2
4.2.2	RSM_GET_EP_4_AXIS	xxxxxx	xxxxxx	x
4.3.1	RSM_ABS_SET_POSITION	0x0AF4	2804	4
4.3.1	RSM_MACRO_ABS_SET_POSITION	0x0CF4	3316	4
4.3.2	RSM_ABS_GET_POSITION	xxxxxx	xxxxxx	x
4.3.2	RSM_MACRO_ABS_GET_POSITION	0x0CF5	3317	x
4.3.2	RSM_ABS_GET_POSITION_4_AXIS	xxxxxx	xxxxxx	x
4.4	RSM_GET_CV	xxxxxx	xxxxxx	x
4.4	RSM_GET_CV_4_AXIS	xxxxxx	xxxxxx	x
4.5	RSM_GET_CA	xxxxxx	xxxxxx	x
4.5	RSM_GET_CA_4_AXIS	xxxxxx	xxxxxx	x
4.6.1	RSM_MACRO_GET_DI	0x0C2E	3118	3
4.6.2	RSM_GET_DI_ALL	xxxxxx	xxxxxx	x
4.6.2	RSM_MACRO_GET_DI_ALL	0x0C31	3121	2
4.6.2	RSM_GET_DI_ALL_4_AXIS	xxxxxx	xxxxxx	x
4.7.1	RSM_MACRO_GET_DI_SIGNAL	0x0C40	3136	3
4.7.2	RSM_GET_DI_SIGNAL_ALL	xxxxxx	xxxxxx	x
4.7.2	RSM_MACRO_GET_DI_SIGNAL _ALL	0x0C41	3137	2
4.7.2	RSM_GET_DI_SIGNAL_ALL_4_AXIS	xxxxxx	xxxxxx	x
4.8	RSM_GET_HOME_SEARCH_STATE	xxxxxx	xxxxxx	x
4.8	RSM_MACRO_ GET_HOME_SEARCH_STATE	0x0C42	3138	2
4.8	RSM_GET_HOME_SEARCH_STATE _4_AXIS	xxxxxx	xxxxxx	x
4.9.1	RSM_GET_ERROR_STATE	xxxxxx	xxxxxx	x
4.9.1	RSM_MACRO_GET_ERROR	0x0C2F	3119	1
4.9.2	RSM_GET_ERROR_CODE	xxxxxx	xxxxxx	x
4.9.2	RSM_MACRO_GET_ERROR_CODE	0x0C30	3120	2
4.9.2	RSM_GET_ERROR_CODE_4_AXIS	xxxxxx	xxxxxx	x
4.10	RSM_GET_FREE_BUFFER	xxxxxx	xxxxxx	x
4.11	RSM_GET_STOP_STATUS	xxxxxx	xxxxxx	x
4.11	RSM_GET_STOP_STATUS_4_AXIS	xxxxxx	xxxxxx	x
4.12	RSM_GET_EMERGENCY_STATE	xxxxxx	xxxxxx	x
4.13	RSM_GET_DRIVING_AXIS	xxxxxx	xxxxxx	x

章節	Sub_Function	編號 (Hex) reg(0)	編號 (Dec) reg(0)	暫存器數量
4.14	RSM_GET_INTERPOL_RDY_FLAG	xxxxxx	xxxxxx	x
4.15	RSM_GET_TRIG_INTFACTOR	xxxxxx	xxxxxx	x
4.16.2	RSM_GET_DEVICE_STATE	xxxxxx	xxxxxx	x
5.1.1	RSM_MACRO_FRNET_IN	0x0C32	3122	2
5.1.2	RSM_MACRO_FRNET_READ	0x0C34	3124	5
5.1.3	RSM_FRNET_READ_SINGLE_DIO	xxxxxx	xxxxxx	x
5.1.4	RSM_FRNET_READ_GROUP_DIO	xxxxxx	xxxxxx	x
5.1.5	RSM_FRNET_READ_MULTI_GROUP_DIO	xxxxxx	xxxxxx	x
5.2.1	RSM_MACRO_FRNET_OUT	0x0C33	3123	4
5.2.2	RSM_MACRO_FRNET_WRITE	0x0C35	3125	7
5.2.3	RSM_FRNET_WRITE_SINGLE_DO	xxxxxx	xxxxxx	x
5.2.4	RSM_FRNET_WRITE_GROUP_DO	xxxxxx	xxxxxx	x
5.2.5	RSM_FRNET_WRITE_MULTI_GROUP_DO	xxxxxx	xxxxxx	x
5.3	RSM_MACRO_FRNET_WAIT	0x0C36	3126	7
5.4	RSM_SET_FRNET_TRIGGER_EVENT	xxxxxx	xxxxxx	x
5.5	RSM_GET_FRNET_TRIGGER_EVENT_SETTING	xxxxxx	xxxxxx	x
6.1	RSM_SET_HV	0x0A3C	2620	4
6.1	RSM_MACRO_SET_HV	0x0C3C	3132	4
6.2	RSM_HOME_LIMIT	0x0A3D	2621	3
6.2	RSM_MACRO_HOME_LIMIT	0x0C3D	3133	3
6.3	RSM_SET_HOME_MODE	0x0A3E	2622	8
6.3	RSM_MACRO_SET_HOME_MODE	0x0C3E	3134	8
6.4	RSM_HOME_START	0x0A3F	2623	2
6.4	RSM_MACRO_HOME_START	0x0C3F	3135	2
7.1.1	RSM_NORMAL_SPEED	0x0A46	2630	3
7.1.1	RSM_MACRO_NORMAL_SPEED	0x0C46	3142	3
7.1.2	RSM_SET_SV	0x0A47	2631	4
7.1.2	RSM_MACRO_SET_SV	0x0C47	3143	4
7.1.3	RSM_SET_V	0x0A48	2632	4
7.1.3	RSM_MACRO_SET_V	0x0C48	3144	4
7.1.4	RSM_SET_A	0x0A49	2633	4
7.1.4	RSM_MACRO_SET_A	0x0C49	3145	4
7.1.5	RSM_SET_D	0x0A4A	2634	4
7.1.5	RSM_MACRO_SET_D	0x0C4A	3146	4

章節	Sub_Function	編號 (Hex) reg(0)	編號 (Dec) reg(0)	暫存器數量
7.1.6	RSM_SET_K	0x0A4B	2635	4
7.1.6	RSM_MACRO_SET_K	0x0C4B	3147	4
7.1.7	RSM_SET_L	0x0A4C	2636	4
7.1.7	RSM_MACRO_SET_L	0x0C4C	3148	4
7.1.8	RSM_SET_AO	0x0A4D	2637	4
7.1.8	RSM_MACRO_SET_AO	0x0C4D	3149	4
7.1.9	RSM_FIXED_MOVE	0x0A4E	2638	4
7.1.9	RSM_MACRO_FIXED_MOVE	0x0C4E	3150	4
7.1.10	RSM_SET_PULSE	0x0A4F	2639	4
7.1.10	RSM_MACRO_SET_PULSE	0x0C4F	3151	4
7.1.11	RSM_ABS_FIXED_MOVE	0x0AF6	2806	4
7.1.11	RSM_ABS_MACRO_FIXED_MOVE	0x0CF6	3318	4
7.1.12	RSM_CONTINUE_MOVE	0x0A50	2640	4
7.1.12	RSM_MACRO_CONTINUE_MOVE	0x0C50	3152	4
7.2.1	RSM_AXIS_ASSIGN	0x0A5A	2650	4
7.2.1	RSM_MACRO_AXIS_ASSIGN	0x0C5A	3162	4
7.2.2	RSM_VECTOR_SPEED	0x0A5B	2651	2
7.2.2	RSM_MACRO_VECTOR_SPEED	0x0C5B	3163	2
7.2.3	RSM_SET_VSV	0x0A5C	2652	3
7.2.3	RSM_MACRO_SET_VSV	0x0C5C	3164	3
7.2.4	RSM_SET_VV	0x0A5D	2653	3
7.2.4	RSM_MACRO_SET_VV	0x0C5D	3165	3
7.2.5	RSM_SET_VA	0x0A5E	2654	3
7.2.5	RSM_MACRO_SET_VA	0x0C5E	3166	3
7.2.6	RSM_SET_VD	0x0A5F	2655	3
7.2.6	RSM_MACRO_SET_VD	0x0C5F	3167	3
7.2.7	RSM_SET_VK	0x0A60	2656	3
7.2.7	RSM_MACRO_SET_VK	0x0C60	3168	3
7.2.8	RSM_SET_VL	0x0A61	2657	3
7.2.8	RSM_MACRO_SET_VL	0x0C61	3169	3
7.2.9	RSM_SET_VAO	0x0A62	2658	3
7.2.9	RSM_MACRO_SET_VAO	0x0C62	3170	3
7.2.10	RSM_LINE_2D	0x0A63	2659	5
7.2.10	RSM_MACRO_LINE_2D	0x0C63	3171	5
7.2.11	RSM_ABS_LINE_2D	0x0AF7	2807	5
7.2.11	RSM_ABS_MACRO_LINE_2D	0x0CF7	3319	5

章節	Sub_Function	編號 (Hex) reg(0)	編號 (Dec) reg(0)	暫存器數量
7.2.12	RSM_LINE_3D	0x0A64	2660	7
7.2.12	RSM_MACRO_LINE_3D	0x0C64	3172	7
7.2.13	RSM_ABS_LINE_3D	0x0AF8	2808	7
7.2.13	RSM_ABS_MACRO_LINE_3D	0x0CF8	3320	7
7.2.14	RSM_ARC_CW	0x0A65	2661	9
7.2.14	RSM_MACRO_ARC_CW	0x0C65	3173	9
7.2.15	RSM_ARC_CCW	0x0A67	2663	9
7.2.15	RSM_MACRO_ARC_CCW	0x0C67	3175	9
7.2.16	RSM_ABS_ARC_CW	0x0AF9	2809	9
7.2.16	RSM_ABS_MACRO_ARC_CW	0x0CF9	3321	9
7.2.17	RSM_ABS_ARC_CCW	0x0AFA	2810	9
7.2.17	RSM_ABS_MACRO_ARC_CCW	0x0CFA	3322	9
7.2.18	RSM_CIRCLE_CW	0x0A69	2665	5
7.2.18	RSM_MACRO_CIRCLE_CW	0x0C69	3177	5
7.2.19	RSM_CIRCLE_CCW	0x0A6A	2666	5
7.2.19	RSM_MACRO_CIRCLE_CCW	0x0C6A	3178	5
7.3.1	RSM_SYNC_ACTION	0x0A6E	2670	14
7.3.1	RSM_MACRO_SYNC_ACTION	0x0C6E	3182	14
7.3.2	RSM_CLEAR_SYNC_ACTION	0x0A95	2709	2
7.3.2	RSM_MACRO_CLEAR_SYNC_ACTION	0x0C95	3221	2
7.3.3	RSM_SET_ACTIVATION_FACTORS	0x0A96	2710	4
7.3.3	RSM_MACRO_SET_ACTIVATION_FACTORS	0x0C96	3222	4
7.3.4	RSM_SET_ACTIVATION_AXIS	0x0A97	2711	3
7.3.4	RSM_MACRO_SET_ACTIVATION_AXIS	0x0C97	3223	3
7.3.5	RSM_SET_ACTION	0x0A98	2712	11
7.3.5	RSM_MACRO_SET_ACTION	0x0C98	3224	11
7.3.6	RSM_SET_COMPARE	0x0A70	2672	6
7.3.6	RSM_MACRO_SET_COMPARE	0x0C70	3184	6
7.3.7	RSM_GET_LATCH	xxxxxx	xxxxxx	x
7.3.7	RSM_MACRO_GET_LATCH	0x0C71	3185	2
7.3.7	RSM_GET_LATCH_4_AXIS	xxxxxx	xxxxxx	x
7.3.8	RSM_SET_PRESET	0x0A72	2674	4
7.3.8	RSM_MACRO_SET_PRESET	0x0C72	3186	4
7.3.9	RSM_SET_OUT	0x0A73	2675	4
7.3.9	RSM_MACRO_SET_OUT	0x0C73	3187	4

章節	Sub_Function	編號 (Hex) reg(0)	編號 (Dec) reg(0)	暫存器數量
7.3.10	RSM_ENABLE_INT	0x0AAA	2730	1
7.3.10	RSM_MACRO_ENABLE_INT	0x0CAA	3242	1
7.3.11	RSM_DISABLE_INT	0x0AAB	2731	1
7.3.11	RSM_MACRO_DISABLE_INT	0x0CAB	3243	1
7.3.12	RSM_INTFACTOR_ENABLE	0x0AAC	2732	4
7.3.12	RSM_MACRO_INTFACTOR_ENABLE	0x0CAC	3244	4
7.3.13	RSM_INTFACTOR_DISABLE	0x0AAD	2733	3
7.3.13	RSM_MACRO_INTFACTOR_DISABLE	0x0CAD	3245	3
7.4.1	RSM_RECTANGLE	0x0A78	2680	20
7.4.1	RSM_MACRO_RECTANGLE	0x0C78	3192	20
7.4.2	RSM_LINE_2D_INITIAL	0x0A7C	2684	9
7.4.2	RSM_MACRO_LINE_2D_INITIAL	0x0C7C	3196	9
7.4.3	RSM_LINE_2D_CONTINUE	0x0A7E	2686	6
7.4.3	RSM_MACRO_LINE_2D_CONTINUE	0x0C7E	3198	6
7.4.4	RSM_ABS_LINE_2D_CONTINUE	0x0AFB	2811	6
7.4.4	RSM_ABS_MACRO_LINE_2D_CONTINUE	0x0CFB	3323	6
7.4.5	RSM_LINE_3D_INITIAL	0x0A7F	2687	10
7.4.5	RSM_MACRO_LINE_3D_INITIAL	0x0C7F	3199	10
7.4.6	RSM_LINE_3D_CONTINUE	0x0A81	2689	8
7.4.6	RSM_MACRO_LINE_3D_CONTINUE	0x0CFC	3324	8
7.4.7	RSM_ABS_LINE_3D_CONTINUE	0x0AFC	2812	8
7.4.7	RSM_ABS_MACRO_LINE_3D_CONTINUE	0x0C81	3201	8
7.4.8	RSM_MIX_2D_INITIAL	0x0A82	2690	10
7.4.8	RSM_MACRO_MIX_2D_INITIAL	0x0C82	3202	10
7.4.9	RSM_MIX_2D_CONTINUE	0x0AFD	2813	11
7.4.9	RSM_MACRO_MIX_2D_CONTINUE	0x0CFD	3325	11
7.4.10	RSM_ABS_MIX_2D_CONTINUE	0x0A84	2692	11
7.4.10	RSM_ABS_MACRO_MIX_2D_CONTINUE	0x0C84	3204	11
7.4.11	RSM_HELIX_3D	0x0A88	2696	15
7.4.11	RSM_MACRO_HELIX_3D	0x0C88	3208	15
7.4.12	RSM_LINE_SCAN	0x0A90	2704	7
7.4.13	RSM_LINE_SCAN_START	0x0A91	2705	5
7.4.14	RSM_GET_LINE_SCAN_DONE	xxxxxx	xxxxxx	x
7.5.1	RSM_DRV_HOLD	0x0AB4	2740	2

章節	Sub_Function	編號 (Hex) reg(0)	編號 (Dec) reg(0)	暫存器數量
7.5.1	RSM_MACRO_DRV_HOLD	0x0CB4	3252	2
7.5.2	RSM_DRV_START	0x0AB5	2741	2
7.5.2	RSM_MACRO_DRV_START	0x0CB5	3253	2
7.5.3	RSM_STOP_SLOWLY	0x0AB7	2743	2
7.5.3	RSM_MACRO_STOP_SLOWLY	0x0CB7	3255	2
7.5.4	RSM_STOP_SUDDENLY	0x0AB8	2744	2
7.5.4	RSM_MACRO_STOP_SUDDENLY	0x0CB8	3256	2
7.5.5	RSM_VSTOP_SLOWLY	0x0AB9	2745	1
7.5.5	RSM_MACRO_VSTOP_SLOWLY	0x0CB9	3257	1
7.5.6	RSM_VSTOP_SUDDENLY	0x0ABA	2746	1
7.5.6	RSM_MACRO_VSTOP_SUDDENLY	0x0CBA	3258	1
7.5.7	RSM_CLEAR_STOP	0x0ABB	2747	2
7.5.7	RSM_MACRO_CLEAR_STOP	0x0CBB	3259	2
7.5.8	RSM_CLEAR_VSTOP	0x0A09	2569	1
7.5.8	RSM_MACRO_CLEAR_VSTOP	0x0C09	3081	1
7.5.9	RSM_EMERGENCY_STOP	0x0A04	2564	2
7.5.10	RSM_CLEAR_EMERGENCY_STOP	0x0A05	2565	1
8.1	RSM_LOAD_INITIAL	0x0AC8	2760	1
9.1.1	RSM_MP_CREATE	0x0AC9	2761	2
9.1.2	RSM_MACRO_MP_CLOSE	0x0CCA	3274	1
9.1.3	RSM_MP_CALL	0x0ACB	2763	2
9.1.3	RSM_MACRO_MP_CALL	0x0CCB	3275	2
9.2.1	RSM_MP_ISR_CREATE	0x0ACD	2765	2
9.2.2	RSM_MACRO_MP_ISR_CLOSE	0x0CCE	3278	1
9.2.3	RSM_MP_ISR_CALL	0x0ACF	2767	2
9.3.1	RSM_MACRO_SET_VAR	0x0CD2	3282	5
9.3.2	RSM_MACRO_SET_RVAR	0x0CD3	3283	3
9.4	RSM_MACRO_VAR_CALCULATE	0x0CD4	3284	8
9.5.1	RSM_MACRO_FOR	0x0CD5	3285	3
9.5.2	RSM_MACRO_NEXT	0x0CD6	3286	1
9.5.3	RSM_MACRO_EXIT_FOR	0x0CDE	3294	1
9.6.1	RSM_MACRO_IF	0x0CD7	3287	6
9.6.2	RSM_MACRO_ELSE	0x0CD8	3288	1
9.6.3	RSM_MACRO_END_IF	0x0CD9	3289	1
9.7.1	RSM_MACRO_GOTO	0x0CDF	3295	2
9.7.2	RSM_MACRO_LABEL	0x0CE1	3297	2

章節	Sub_Function	編號 (Hex) reg(0)	編號 (Dec) reg(0)	暫存器數量
9.8	RSM_MACRO_TIMER	0x0CDA	3290	3
9.9	RSM_STOP_WAIT	0x0ADB	2779	2
9.9	RSM_MACRO_STOP_WAIT	0x0CDB	3291	2
9.10	RSM_MACRO_EXIT_MACRO	0x0CDD	3293	1
9.11	RSM_MP_TERMINATE	0x0AE2	2786	1
9.11	RSM_MACRO_MP_TERMINATE	0x0CE2	3298	1
9.12	RSM_GET_MP_DOWNLOAD_STATUS	xxxxxx	xxxxxx	x
9.13	RSM_CHANGE_VEL_ON_FLY	0x0A43	2627	4
11.1	RSM_GET_RSM_FIRMWARE_VERSION	xxxxxx	xxxxxx	x
11.2	RSM_GET_i8094H_FIRMWARE_VERSION	xxxxxx	xxxxxx	x
11.3	RSM_GET_DLL_VERSION	xxxxxx	xxxxxx	x

12.4 Coil 資料表

可使用下列功能碼存取 Coil 資料:

- 功能碼 1(Read Coils)
- 功能碼 5(Write Single Coil)
- 功能碼 15(Write Multiple Coils)

在本章所描述的位址皆採用 0 為基準(zero-base)的索引方式，暫存器表的第一個位址將由 0 開始，但有部分的 PLC 與 HMI 採用 1 為基準(one-base)的索引方式，也就是說使用位址 1 做為第一個位址，此時因為索引方式的差異，在操作暫存器時必須特別注意設定正確的對應位址。

PLC 定義位址 0XXXX 用來操作 coil 暫存器，其中 XXXX 為操作的暫存器位址，且位址 1 被定義為第一個位址，假設操作位址 00001，則實際將對應操作下表所描述的位址 0 暫存器。

當資料需使用兩個暫存器來表示時，可讀取目前的字組順序(WORD order)設定值，依此判斷高字組 (Most Significant WORD, MSW)與低字組 (Least Significant WORD, LSW)所屬的暫存器。字組順序設定值被定義在 holding 暫存器的位址 14，當字組順序設定為 0 時，將使用第一個暫存器表示高字組，第二個暫存器表示低字組。

暫存器類型:

- R: 只可讀取。
- W: 只可寫入。
- R/W : 可讀取與寫入。

12.4.1 FRnet DO

起始位址為 0(zero-based)，請使用功能碼 5 或 15 寫入 FRnet DO 狀態。
(1) 使用功能碼 5 寫入單點 FRnet DO (0~127)。
(2) 使用功能碼 15 同時寫入多點 FRnet DO。可寫入跨通道的連續點位，例如寫入位址 16~47，表示同時寫入群組 1 與群組 2 的所有通道。

附錄表 23: FRnet DO (PLC 位址: 00001 ~ 00128)

FRnet DO				
群組	**通道**	**位址 (zero based)**	**類型**	**說明**
0	0	0	R/W	
	1	1	R/W	
	2	2	R/W	
	3	3	R/W	
	4	4	R/W	
	5	5	R/W	
	6	6	R/W	
	7	7	R/W	
	8	8	R/W	
	9	9	R/W	
	10	10	R/W	
	11	11	R/W	
	12	12	R/W	
	13	13	R/W	
	14	14	R/W	
	15	15	R/W	
1	0	16	R/W	
	1	17	R/W	
	2	18	R/W	
	3	19	R/W	
	4	20	R/W	
	5	21	R/W	
	6	22	R/W	
	7	23	R/W	
	8	24	R/W	
	9	25	R/W	
	10	26	R/W	
	11	27	R/W	
	12	28	R/W	

群組	通道	位址 (zero based)	類型	說明
1	13	29	R/W	
	14	30	R/W	
	15	31	R/W	
2	0	32	R/W	
	1	33	R/W	
	2	34	R/W	
	3	35	R/W	
	4	36	R/W	
	5	37	R/W	
	6	38	R/W	
	7	39	R/W	
	8	40	R/W	
	9	41	R/W	
	10	42	R/W	
	11	43	R/W	
	12	44	R/W	
	13	45	R/W	
	14	46	R/W	
	15	47	R/W	
3	0	48	R/W	
	1	49	R/W	
	2	50	R/W	
	3	51	R/W	
	4	52	R/W	
	5	53	R/W	
	6	54	R/W	
	7	55	R/W	
	8	56	R/W	
	9	57	R/W	
	10	58	R/W	
	11	59	R/W	
	12	60	R/W	
	13	61	R/W	
	14	62	R/W	
	15	63	R/W	
4	0	64	R/W	
	1	65	R/W	
	2	66	R/W	
	3	67	R/W	
	4	68	R/W	
	5	69	R/W	
	6	70	R/W	

群組	通道	位址 (zero based)	類型	說明
4	7	71	R/W	
	8	72	R/W	
	9	73	R/W	
	10	74	R/W	
	11	75	R/W	
	12	76	R/W	
	13	77	R/W	
	14	78	R/W	
	15	79	R/W	
5	0	80	R/W	
	1	81	R/W	
	2	82	R/W	
	3	83	R/W	
	4	84	R/W	
	5	85	R/W	
	6	86	R/W	
	7	87	R/W	
	8	88	R/W	
	9	89	R/W	
	10	90	R/W	
	11	91	R/W	
	12	92	R/W	
	13	93	R/W	
	14	94	R/W	
	15	95	R/W	
6	0	96	R/W	
	1	97	R/W	
	2	98	R/W	
	3	99	R/W	
	4	100	R/W	
	5	101	R/W	
	6	102	R/W	
	7	103	R/W	
	8	104	R/W	
	9	105	R/W	
	10	106	R/W	
	11	107	R/W	
	12	108	R/W	
	13	109	R/W	
	14	110	R/W	
	15	111	R/W	

群組	通道	位址 (zero based)	類型	說明
7	0	112	R/W	
	1	113	R/W	
	2	114	R/W	
	3	115	R/W	
	4	116	R/W	
	5	117	R/W	
	6	118	R/W	
	7	119	R/W	
	8	120	R/W	
	9	121	R/W	
	10	122	R/W	
	11	123	R/W	
	12	124	R/W	
	13	125	R/W	
	14	126	R/W	
	15	127	R/W	

12.4.2 伺服啟動/關閉

附錄表 24: 伺服啟動/關閉 (PLC 位址: 00129 ~ 00132)

伺服狀態			
軸	**位址 (zero based)**	**類型**	**說明**
X	128	R	1: 伺服啟動 0: 伺服關閉
Y	129	R	1: 伺服啟動 0: 伺服關閉
Z	130	R	1: 伺服啟動 0: 伺服關閉
U	131	R	1: 伺服啟動 0: 伺服關閉

12.5 Discrete Input 資料表

請使用功能碼 2 存取 discrete input 資料。在本章所描述的位址皆採用 0 為基準(zero-base)的索引方式，暫存器表的第一個位址將由 0 開始，但有部分的 PLC 與 HMI 採用 1 為基準(one-base)的索引方式，也就是說使用位址 1 做為第一個位址，此時因為索引方式的差異，在操作暫存器時必須特別注意設定正確的對應位址。

PLC 定義位址 1XXXX 用來操作 discrete input 暫存器，其中 XXXX 為操作的暫存器位址，且位址 1 被定義為第一個位址，假設操作位址 100001，則實際將對應操作下表所描述的位址 0 暫存器。

暫存器類型:

- R: 只可讀取。
- W: 只可寫入。
- R/W : 可讀取與寫入。

12.5.1 FRnet DI

附錄表 25: FRnet DI (PLC 位址: 10001 ~ 10128)

FRnet DI				
群組	**通道**	**位址 (zero based)**	**類型**	**說明**
8	0	0	R	
	1	1	R	
	2	2	R	
	3	3	R	
	4	4	R	
	5	5	R	
	6	6	R	
	7	7	R	
	8	8	R	
	9	9	R	
	10	10	R	
	11	11	R	
	12	12	R	
	13	13	R	
	14	14	R	
	15	15	R	
9	0	16	R	
	1	17	R	
	2	18	R	
	3	19	R	
	4	20	R	
	5	21	R	
	6	22	R	
	7	23	R	
	8	24	R	
	9	25	R	
	10	26	R	
	11	27	R	

群組	通道	位址 (zero based)	類型	說明
9	12	28	R	
	13	29	R	
	14	30	R	
	15	31	R	
10	0	32	R	
	1	33	R	
	2	34	R	
	3	35	R	
	4	36	R	
	5	37	R	
	6	38	R	
	7	39	R	
	8	40	R	
	9	41	R	
	10	42	R	
	11	43	R	
	12	44	R	
	13	45	R	
	14	46	R	
	15	47	R	
11	0	48	R	
	1	49	R	
	2	50	R	
	3	51	R	
	4	52	R	
	5	53	R	
	6	54	R	
	7	55	R	
	8	56	R	
	9	57	R	
	10	58	R	
	11	59	R	
	12	60	R	

群組	通道	位址 (zero based)	類型	說明
11	13	61	R	
	14	62	R	
	15	63	R	
12	0	64	R	
	1	65	R	
	2	66	R	
	3	67	R	
	4	68	R	
	5	69	R	
	6	70	R	
	7	71	R	
	8	72	R	
	9	73	R	
	10	74	R	
	11	75	R	
	12	76	R	
	13	77	R	
	14	78	R	
	15	79	R	
13	0	80	R	
	1	81	R	
	2	82	R	
	3	83	R	
	4	84	R	
	5	85	R	
	6	86	R	
	7	87	R	
	8	88	R	
	9	89	R	
	10	90	R	
	11	91	R	
	12	92	R	
	13	93	R	

群組	通道	位址 (zero based)	類型	說明
13	14	94	R	
	15	95	R	
14	0	96	R	
	1	97	R	
	2	98	R	
	3	99	R	
	4	100	R	
	5	101	R	
	6	102	R	
	7	103	R	
	8	104	R	
	9	105	R	
	10	106	R	
	11	107	R	
	12	108	R	
	13	109	R	
	14	110	R	
	15	111	R	
15	0	112	R	
	1	113	R	
	2	114	R	
	3	115	R	
	4	116	R	
	5	117	R	
	6	118	R	
	7	119	R	
	8	120	R	
	9	121	R	
	10	122	R	
	11	123	R	
	12	124	R	
	13	125	R	
	14	126	R	
	15	127	R	

12.5.2 運動控制晶片的 DI 狀態

附錄表 26: 運動控制晶片的 DI 狀態 (PLC 位址: 10129 ~ 10192)

讀取運動控制晶片的 DI 狀態				
軸	**DI**	**位址 (zero based)**	**類型**	**說明**
X	DRIVING	128	R	X 軸的驅動狀態 1: 驅動中, 0: 停止
	LIMIT+	129	R	X 軸的正方向硬體極限(LMT+)狀態 0: off, 1: on
	LIMIT-	130	R	X 軸的反方向硬體極限(LMT-)狀態 0: off, 1: on
	EMERGENCY	131	R	X 軸的緊急停止狀態 0: off, 1: on
	ALARM	132	R	X 軸的 ALARM 狀態(請使用函式 RSM_SET_ALARM()開啟 ALARM 功能) 0: off, 1: on
	HOME	133	R	X 軸的 HOME(IN1)狀態 0: on, 1: off
	NHOME	134	R	X 軸的 NHOME(IN0)狀態 0: on, 1: off
	IN3	135	R	X 軸的 IN3 狀態 0: on, 1: off
	INPOS	136	R	X 軸的 INPOS 狀態 0: on, 1: off
	INDEX	137	R	X 軸的 Z 相(IN2)狀態 0: on, 1: off
		138		未定義
		139		
		140		
		141		
		142		
		143		
Y	DRIVING	144	R	Y 軸的驅動狀態 1: 驅動中, 0: 停止
	LIMIT+	145	R	Y 軸的正方向硬體極限(LMT+)狀態 0: off, 1: on
	LIMIT-	146	R	Y 軸的反方向硬體極限(LMT-)狀態 0: off, 1: on
	EMERGENCY	147	R	Y 軸的緊急停止狀態 0: off, 1: on

軸	DI	位址 (zero based)	類型	說明
Y	ALARM	148	R	Y 軸的 ALARM 狀態(請使用函式 RSM_SET_ALARM()開啟 ALARM 功能) 0: off, 1: on
	HOME	149	R	Y 軸的 HOME(IN1)狀態 0: on, 1: off
	NHOME	150	R	Y 軸的 NHOME(IN0)狀態 0: on, 1: off
	IN3	151	R	Y 軸的 IN3 狀態 0: on, 1: off
	INPOS	152	R	Y 軸的 INPOS 狀態 0: on, 1: off
	INDEX	153	R	Y 軸的 Z 相(IN2)狀態 0: on, 1: off
		154		未定義
		155		
		156		
		157		
		158		
		159		
Z	DRIVING	160	R	Z 軸的驅動狀態 1: 驅動中, 0: 停止
	LIMIT+	161	R	Z 軸的正方向硬體極限(LMT+)狀態 0: off, 1: on
	LIMIT-	162	R	Z 軸的反方向硬體極限(LMT-)狀態 0: off, 1: on
	EMERGENCY	163	R	Z 軸的緊急停止狀態 0: off, 1: on
	ALARM	164	R	Z 軸的 ALARM 狀態(請使用函式 RSM_SET_ALARM()開啟 ALARM 功能) 0: off, 1: on
	HOME	165	R	Z 軸的 HOME(IN1)狀態 0: on, 1: off
	NHOME	166	R	Z 軸的 NHOME(IN0)狀態 0: on, 1: off
	IN3	167	R	Z 軸的 IN3 狀態 0: on, 1: off
	INPOS	168	R	Z 軸的 INPOS 狀態 0: on, 1: off
	INDEX	169	R	Z 軸的 Z 相(IN2)狀態 0: on, 1: off
		170		未定義
		171		

軸	DI	位址 (zero based)	類型	說明
Z		172		未定義
		173		
		174		
		175		
U	DRIVING	176	R	U 軸的驅動狀態 1: 驅動中, 0: 停止
	LIMIT+	177	R	U 軸的正方向硬體極限(LMT+)狀態 0: off, 1: on
	LIMIT-	178	R	U 軸的反方向硬體極限(LMT-)狀態 0: off, 1: on
	EMERGENCY	179	R	U 軸的緊急停止狀態 0: off, 1: on
	ALARM	180	R	U 軸的 ALARM 狀態(請使用函式 RSM_SET_ALARM()開啟 ALARM 功能) 0: off, 1: on
	HOME	181	R	U 軸的 HOME(IN1)狀態 0: on, 1: off
	NHOME	182	R	U 軸的 NHOME(IN0)狀態 0: on, 1: off
	IN3	183	R	U 軸的 IN3 狀態 0: on, 1: off
	INPOS	184	R	U 軸的 INPOS 狀態 0: on, 1: off
	INDEX	185	R	U 軸的 Z 相(IN2)狀態 0: on, 1: off
		186		未定義
		187		
		188		
		189		
		190		
		191		

12.5.3 運動異常停止狀態

附錄表 27: 運動異常停止狀態 (PLC 位址: 10193 ~ 10256)

讀取運動異常停止狀態				
軸	**停止原因**	**位址 (zero based)**	**類型**	**說明**
X	SOFT LIMIT+	192	R	X 軸的正方向軟體極限被觸發
	SOFT LIMIT-	193	R	X 軸的反方向軟體極限被觸發
	LIMIT+	194	R	X 軸的正方向硬體極限被觸發
	LIMIT-	195	R	X 軸的反方向硬體極限被觸發
	ALARM	196	R	X 軸的 ALARM 訊號觸發
	EMERGENCY	197	R	X 軸的緊急停止狀態被觸發
	Reserved	198	R	未定義
	HOME	199	R	X 軸到達 Z 相或 HOME 位置
	Stop Command	200	R	執行停止運動指令 STOP_SLOWLY STOP_SUDDENLY VSTOP_SLOWLY VSTOP_SUDDENLY
		201		未定義
		202		
		203		
		204		
		205		
		206		
		207		
Y	SOFT LIMIT+	208	R	Y 軸的正方向軟體極限被觸發
	SOFT LIMIT-	209	R	Y 軸的反方向軟體極限被觸發
	LIMIT+	210	R	Y 軸的正方向硬體極限被觸發
	LIMIT-	211	R	Y 軸的反方向硬體極限被觸發

軸	停止原因	位址 (zero based)	類型	說明
	ALARM	212	R	Y 軸的 ALARM 訊號觸發
	EMERGENCY	213	R	Y 軸的緊急停止狀態被觸發
	Reserved	214	R	未定義
	HOME	215	R	Y 軸到達 Z 相或 HOME 位置
	Stop Command	216	R	執行停止運動指令 STOP_SLOWLY STOP_SUDDENLY VSTOP_SLOWLY VSTOP_SUDDENLY
		217		未定義
		218		
		219		
		220		
		221		
		222		
		223		
Z	SOFT LIMIT+	224	R	Z 軸的正方向軟體極限被觸發
	SOFT LIMIT-	225	R	Z 軸的反方向軟體極限被觸發
	LIMIT+	226	R	Z 軸的正方向硬體極限被觸發
	LIMIT-	227	R	Z 軸的反方向硬體極限被觸發
	ALARM	228	R	Z 軸的 ALARM 訊號觸發
	EMERGENCY	229	R	Z 軸的緊急停止狀態被觸發
	Reserved	230	R	未定義
	HOME	231	R	Z 軸到達 Z 相或 HOME 位置
	Stop Command	232	R	執行停止運動指令 STOP_SLOWLY STOP_SUDDENLY VSTOP_SLOWLY VSTOP_SUDDENLY

軸	停止原因	位址 (zero based)	類型	說明
Z		233		未定義
		234		
		235		
		236		
		237		
		238		
		239		
U	SOFT LIMIT+	240	R	U 軸的正方向軟體極限被觸發
	SOFT LIMIT-	241	R	U 軸的反方向軟體極限被觸發
	LIMIT+	242	R	U 軸的正方向硬體極限被觸發
	LIMIT-	243	R	U 軸的反方向硬體極限被觸發
	ALARM	244	R	U 軸的 ALARM 訊號觸發
	EMERGENCY	245	R	U 軸的緊急停止狀態被觸發
	Reserved	246	R	未定義
	HOME	247	R	U 軸到達 Z 相或 HOME 位置
	Stop Command	248	R	執行停止運動指令 STOP_SLOWLY STOP_SUDDENLY VSTOP_SLOWLY VSTOP_SUDDENLY
		249		未定義
		250		
		251		
		252		
		253		
		254		
		255		

12.5.4 運動控制相關的訊號輸入狀態

附錄表 28: 運動控制相關的訊號輸入狀態 (PLC 位址: 10129 ~ 10192)

讀取訊號輸入狀態				
軸	**DI**	**位址 (zero based)**	**類型**	**說明**
X	Near Home	256	R	X 軸的 NHOME(IN0)訊號狀態 0: on, 1: off
	Home	257	R	X 軸的 HOME(IN1)訊號狀態 0: on, 1: off
	Index	258	R	X 軸的編碼器索引(IN2)訊號狀態 0: on, 1: off
	IN3	259	R	X 軸的 IN3 訊號狀態 0: on, 1: off
	MPG (+)	260	R	X 軸的 EXP+訊號狀態(正向外部輸入控制脈波) 0: off, 1: on
	MPG (-)	261	R	X 軸的 EXP-訊號狀態(反向外部輸入控制脈波) 0: off, 1: on
	InPos	262	R	X 軸的 INP 訊號狀態 0: off, 1: on
	Alarm	263	R	X 軸的 ALARM 訊號狀態 0: off, 1: on
Y	Near Home	264	R	Y 軸的 NHOME(IN0)訊號狀態 0: on, 1: off
	Home	265	R	Y 軸的 HOME(IN1)訊號狀態 0: on, 1: off
	Index	266	R	Y 軸的編碼器索引(IN2)訊號狀態 0: on, 1: off
	IN3	267	R	Y 軸的 IN3 訊號狀態 0: on, 1: off
	MPG (+)	268	R	Y 軸的 EXP+訊號狀態(正向外部輸入控制脈波) 0: off, 1: on
	MPG (-)	269	R	Y 軸的 EXP-訊號狀態(反向外部輸入控制脈波) 0: off, 1: on
	InPos	270	R	Y 軸的 INP 訊號狀態 0: off, 1: on
	Alarm	271	R	Y 軸的 ALARM 訊號狀態 0: off, 1: on

軸	DI	位址 (zero based)	類型	說明
Z	Near Home	272	R	Z 軸的 NHOME(IN0)訊號狀態 0: on, 1: off
	Home	273	R	Z 軸的 HOME(IN1)訊號狀態 0: on, 1: off
	Index	274	R	Z 軸的編碼器索引(IN2)訊號狀態 0: on, 1: off
	IN3	275	R	Z 軸的 IN3 訊號狀態 0: on, 1: off
	MPG (+)	276	R	Z 軸的 EXP+訊號狀態(正向外部輸入控制脈波) 0: off, 1: on
	MPG (-)	277	R	Z 軸的 EXP-訊號狀態(反向外部輸入控制脈波) 0: off, 1: on
	InPos	278	R	Z 軸的 INP 訊號狀態 0: off, 1: on
	Alarm	279	R	Z 軸的 ALARM 訊號狀態 0: off, 1: on
U	Near Home	380	R	U 軸的 NHOME(IN0)訊號狀態 0: on, 1: off
	Home	381	R	U 軸的 HOME(IN1)訊號狀態 0: on, 1: off
	Index	382	R	U 軸的編碼器索引(IN2)訊號狀態 0: on, 1: off
	IN3	383	R	U 軸的 IN3 訊號狀態 0: on, 1: off
	MPG (+)	384	R	U 軸的 EXP+訊號狀態(正向外部輸入控制脈波) 0: off, 1: on
	MPG (-)	385	R	U 軸的 EXP-訊號狀態(反向外部輸入控制脈波) 0: off, 1: on
	InPos	386	R	U 軸的 INP 訊號狀態 0: off, 1: on
	Alarm	387	R	U 軸的 ALARM 訊號狀態 0: off, 1: on

13 RS-M8194H LED 說明

LED 說明:

LED	狀態	說明
Sys	恆亮	電源開啟且韌體執行中
	閃爍	電源開啟但韌體尚未執行
	不亮	電源關閉
Tx	閃爍	透過 RS-232 傳送資料
	不亮	無傳送資料
Rx	閃爍	透過 RS-232 接收資料
	不亮	無接收資料
NET		
	閃爍	資料傳輸中
	不亮	無資料傳輸
MOD	恆亮	i-8094H 模組已正確安裝在 RS-M8194H 上
	閃爍	非 i-8094H 模組已安裝在 RS-M8194H 上
	不亮	無模組安裝在 RS-M8194H 上

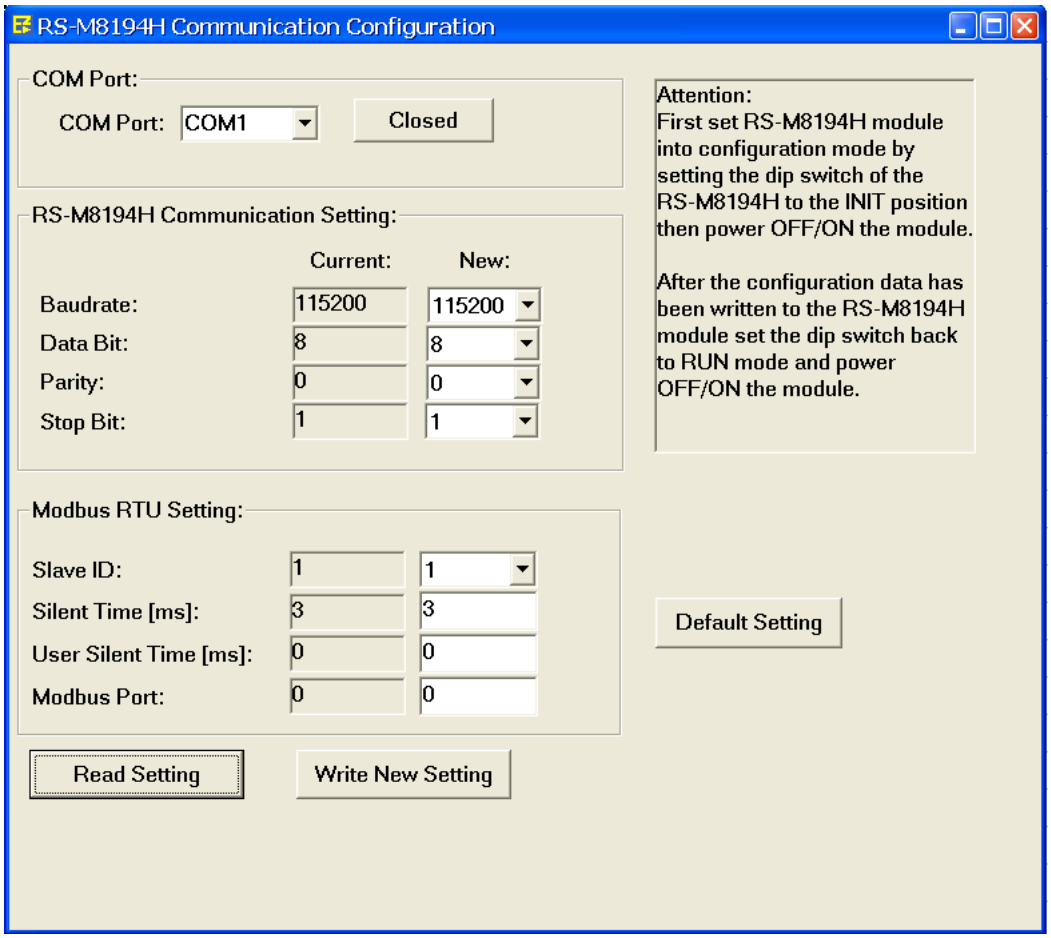
i-8094H的LED說明:

- **P 為電源指示燈**
- **A 為FRnet指示燈**
- **D 脈波輸出指示燈**

14 工具軟體 EzMove

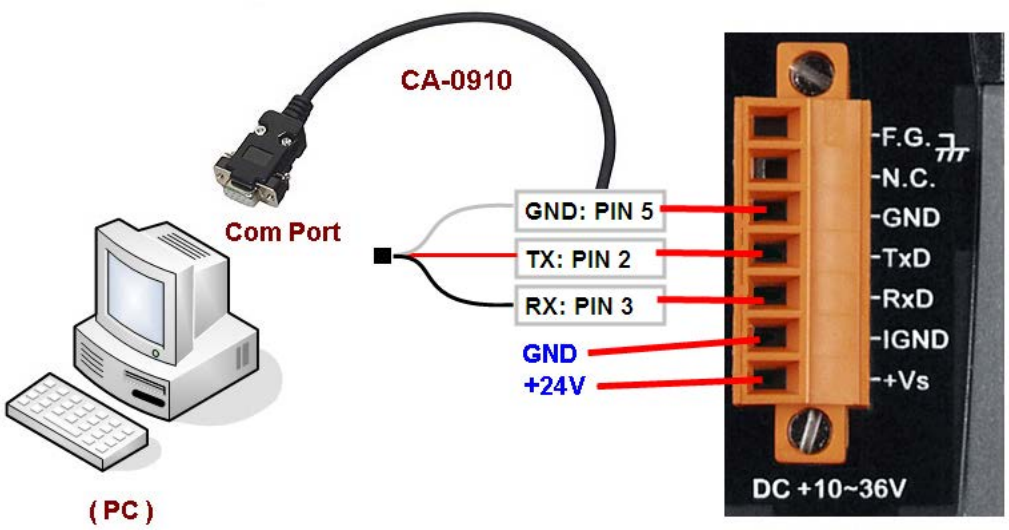
14.1 串列連接埠設定

執行工具軟體 EzMove 並且開啟"RS-M8194H Communication Configuration" 視窗 ([Setting] - [RS-M8194H Setting] – [By COM Port] – [Communication Configuration])。



步驟 1: **關閉 RS-M8194H 電源。**

步驟 2: **使用通訊線 CA-0910 連接 PC 與 RS-M8194H，將 CA-0910 的 Tx, Rx, GND 接頭分別接到 RS-M8194H 之 Rx, Tx, GND，D-Type 9 Pin 端連接至 PC 的 COM 埠。**



步驟 3: 切換指撥開關至”Init”並且啟動電源。

(Dip Switch -- Init)

步驟 4: 選擇連接 RS-M8194H 的 COM 埠號碼後點擊”Open”按鈕。

步驟 5: 點擊”Read Setting”按鈕讀取目前的設定值。

步驟 6: 輸入新的設定值後點擊“Write New Setting”按鈕保存。

步驟 7: 若需還原為預設值可先點擊”Default Setting”按鈕再點擊”Write New Setting”按鈕。

步驟 8: 設定完成後將指撥開關切換回“Run” 。

(Dip Switch -- Run)

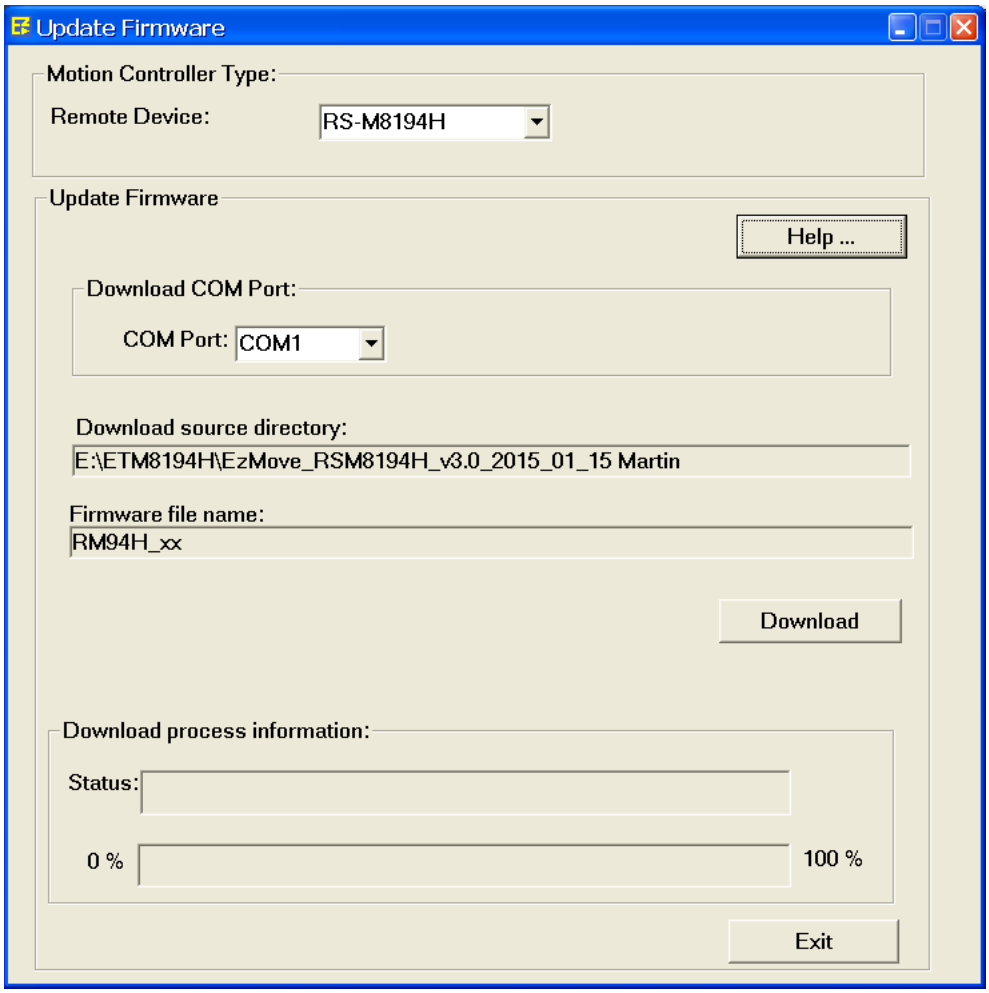
步驟 9: 重置 RS-M8194H 電源。

請注意!

設定完成後請將 RS-232 連接線(CA-0910)移除，避免雜訊由此進入 RS-M8194H。

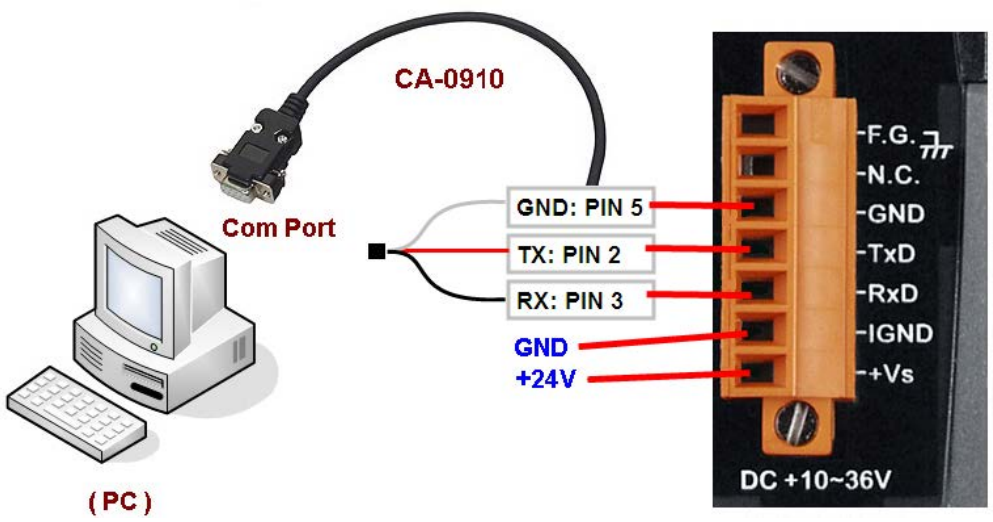
14.2 韌體更新

啟動工具軟體 EzMove，從功能表的[Setting] – [RS-M8194H Setting] – [By COM Port] – [Update Firmware]開啟更新韌體視窗，依照以下步驟完成更新程序：



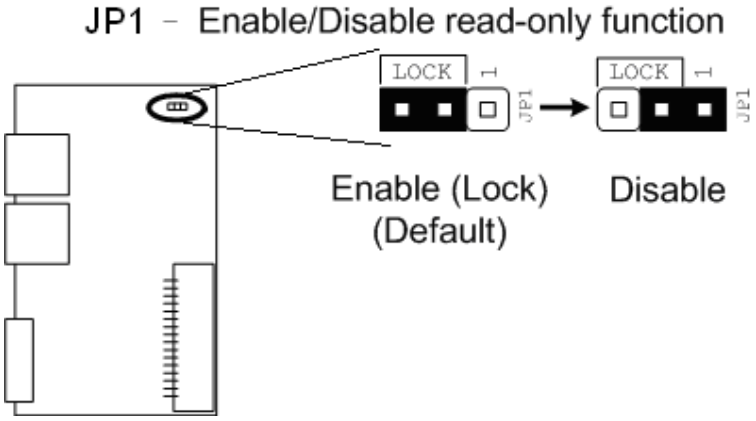
1. 關閉 RS-M8194H 電源。

2. 使用通訊線 CA-0910 連接 RS-M8194H。CA-0910 的 Tx、Rx、GND 分別接至 RS-M8194H 的 Rx、Tx、GND， D-Type 9 Pin 端連接至 PC 的 COM 埠。



3. 切換指撥開關至”Init”。

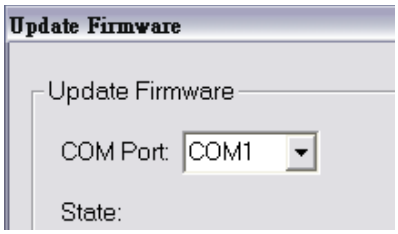
4. 調整 Jumper 1 的位置至”Disable”的位置。



5. 開啟 RS-M8194H 電源。

6. 刪除 EzMove 安裝資料夾內(\ICPDAS\RS_M8194H\EzMove_Utility)的舊檔案 autoexec.bat 與 RS94H_XX.EXE，然後複製預定更新的檔案 autoexec.bat、RM94H_XX.EXE 與 RS_M8194H_API.dll 至 EzMove 安裝資料夾內。

7. 選擇 COM 埠(即 CA-0903 專用線與 PC 連接之 COM 埠)。



8. 點擊”Download”按鈕開始下載韌體，下載完成後關閉 RS-M8194H 電源。

9. 調整 Jumper 1 至”Enable”的位置。

10. 切換指撥開關至”Run”。

(Dip Switch -- Run)

11. 啟動 RS-M8194H 電源。