PMDK 函式參考手冊
(Basic Functions for DSP)

語言	繁體中文
版本	V1.1
更新日期	2012/08/06

ICP DAS CO., LTD.
泓格科技股份有限公司

Important Notices

Warranty

All products manufactured by ICP DAS are under warranty regarding defective materials for a period of one year, beginning from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for any damage resulting from the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, not for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

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

目錄

1. 常數定義

1.1 函數型態定義

```
typedef     char            I8  ;
typedef     short           I16 ;
typedef     int             I32 ;
typedef     long            I40 ;
typedef     unsigned char   U8  ;
typedef     unsigned short  U16 ;
typedef     unsigned int    U32 ;
typedef     unsigned long   U40 ;
typedef     float           F32 ;
typedef     double          F64 ;
```

1.2 ICPDAS_PMDK.h 中定義的常數

Macro	*Value*	*Description*
SDRAM_ADDR	0x80000000	為 SDRAM 起始值, SDRAM 範圍：0x80000000 至 0x80FFFFFF
FLASH_ADDR	0x90000000	為每一個 FLASH page 起始值，由 GPIO 設定 page 值 (0~ 0xFF) FLASH page 範圍：0x90000000 至 0x90000FFF
FPGA_ADDR	0x90001000	為 FPGA 起始值 FPGA 範圍：0x90001000 至 0x900013FF
CPLD_ADDR	0x90001400	為 CPLD 起始值 CPLD 範圍：0x90001400 至 0x900017FF
DPRAM_ADDR	0x90001800	為 DPRAM 起始值 DPRAM 範圍：0x90001800 至 0x90001FFF
FRAM_ADDR	0x90002000	為每一個 FRAM page 起始值，由 GPIO 設定 page 值 (0~ 0x1F) FRAM page 範圍：0x90002000 至 0x90003FFF

Macro	*Value*	*Description*
SDRAM_SIZE	0x01000000	為 SDRAM 大小
FLASH_SIZE	0x00100000	為所有 FLASH 大小，必須配合 GPIO 設定分成 256 pages (0~ 0xFF)
FPGA_SIZE	0x0400	為 FPGA 暫存器位址範圍大小
CPLD_SIZE	0x0400	為真正 CPLD 暫存器位址範圍大小
DPRAM_SIZE	0x0800	為真正 DPRAM 大小
FRAM_SIZE	0x00040000	為所有 FRAM 大小，必須配合 GPIO 設定分成 32 pages (0~ 0x1F)

```
//    Definition of Constant
#define       MATH_PI                 3.141592653589793

#define       MY_TRUE                 1
#define       MY_FALSE                0
#define       MY_MINUS                (-1)

#define       MAX_AXIS_NO             6
#define       MAX_DA_CHANNEL          MAX_AXIS_NO
#define       MAX_AD_CHANNEL          MAX_AXIS_NO+1
#define       MAX_DDA_CHANNEL         MAX_AXIS_NO
#define       MAX_ENC_CHANNEL         MAX_AXIS_NO

//    Definition of Return Code
#define  ERR_NO_ERROR                 0
#define  ERR_TIMEOUT_ERROR            (-1)
#define  ERR_HW_ID_ERROR              (-2)
#define  ERR_AXIS_OUT_RANGE           (-3)
#define  ERR_ADDR_OUT_RANGE           (-4)
#define  ERR_VALUE_OUT_RANGE          (-5)
#define  ERR_FPGA_DL_FAILED           (-6)
#define  ERR_DA_AUTO_UPDATE           (-101)
#define  ERR_DA_BUSY                  (-102)
#define  ERR_CMP_INUSE                (-201)
#define  ERR_NOT_IMPLEMENT            (-32768)
```

1.3 PMDKdsp.h 中定義的常數

// 以下定義是給 ***Flash_Erase()***函式的**"Region"**參數所使用

Macro	*Value*	*Description*
FLASH_ERASE_CHIP	0	表示清除 FLASH 全部 範圍：0x00000~0xFFFFF
FLASH_ERASE_CODE	1	表示清除 DSP code 的部分， 範圍：0x00000~0x9FFFF
FLASH_ERASE_RSV	2	表示清除 Reserved 的部分， 範圍：0xA0000~0xAFFFF
FLASH_ERASE_CAL	3	表示清除 AD/DA calibration 的部分， 範圍：0xB0000~0xBFFFF
FLASH_ERASE_FPGA	4	表示清除 FPGA Configuration Data， 範圍：0xC0000~0xFFFFF

// 以下定義是用來表示 **Flash** 記憶體規劃的各區段位址

Macro	*Value*	*Description*
DSP_CODE_FLASH_OFST	0x00000000	Flash 中規劃給 DSP code 的位址
RSV_DATA_FLASH_OFST	0x000A0000	Flash 中未規劃的位址
CAL_DATA_FLASH_OFST	0x000B0000	Flash 中規劃給 AD/DA calibration 的位址
FPGA_DATA_FLASH_OFST	0x000C0000	Flash 中規劃給 FPGA Configuration Data 的位址

// 以下定義是用來表示 **Flash** 記憶體的規劃各區段大小

Macro	*Value*	*Description*
DSP_CODE_FLASH_SIZE	0x000A0000	Flash 中規劃給 DSP code 的大小
RSV_DATA_FLASH_SIZE	0x00010000	Flash 中未規劃的大小
CAL_DATA_FLASH_SIZE	0x00010000	Flash 中規劃給 AD/DA calibration 的大小
FPGA_DATA_FLASH_SIZE	0x00040000	Flash 中規劃給 FPGA Configuration Data 的大小

```
//  Bits within Get/Set_DSP_Int_Factor() and Get/Set_DSP_Int_Flag()
#define B_FRNET_INT         (0x02)          //
#define B_FPGA_INT          (0x01)          //
```

1.4 PMDKfpga.h 中定義的常數

// Default DDA cycle time = 6250*40ns = 0.25ms

```
#define  DEFAULT_DDA_CYCLE_TIME          0.00025
#define  DEFAULT_DDA_LENGTH              6250
```

// The offset address in SDRAM reserved for FPGA download data

Macro	*Value*	*Description*
FPGA_DATA_SDRAM_OFST	0x00F00000	在 SDRAM 中，由此位址開始共有 0x00040000 大小的區間 (FPGA_DATA_FLASH_SIZE) 是被保留起來做為存放 FPGA Configuration Data 用的。 如果 Project 中有包含 LoadFPGAData.C，程式載入時會將 fpgaPMDK.H 中的資料載入到此位址。

// The offset address in DPRAM reserved for interrutp notification

Macro	*Value*	*Description*
DPR_L2H_INT_OFST	0x07FC	DSP 將資料寫到 DPRAM 的此位址時，DPRAM 也會同時對 Host 發出一個中斷訊號 (經由 PCI bus)。
DPR_H2L_INT_OFST	0x07FE	當 Host 經由 PCI bus 將一筆資料寫到 DPRAM 的此位址時，DPRAM 會對 DSP 發出一個中斷訊號 (當相關的中斷因子都有被設為 enable 時)。當 DSP 讀取 DPRAM 的此位址時，則會同時清除 MAIN_INT_CH 中 DPL_INT 的中斷旗標。

// for "FPGA_Src" in *FPGA_Init()*

Macro	*Value*	*Description*
FPGA_SRC_SDRAM	0	由 SDRAM 中的資料來 configure FPGA
FPGA_SRC_FLASH	1	由 FLASH 中的資料來 configure FPGA

```
//  for "Ch" in Set/Get_FPGA_Int_Factor() and Get/Clr_FPGA_Int_Flag()
    #define AXIS0_INT_CH    (0)
    #define AXIS1_INT_CH    (1)
    #define AXIS2_INT_CH    (2)
    #define AXIS3_INT_CH    (3)
    #define AXIS4_INT_CH    (4)
    #define AXIS5_INT_CH    (5)
    #define MISC_INT_CH     (250)
    #define GINP_INT_CH     (253)
    #define MAIN_INT_CH     (254)
    #define GLB_INT_CH      (255)

//  Bits within GLB_INT_CH
    #define B_GLB_INT               (0x0001)

//  Bits within MAIN_INT_CH
    #define B_MISC_INT              (0x8000)
    #define B_SERVO_INT             (0x4000)
    #define B_AX5_H_INT             (0x2000)
    #define B_AX4_H_INT             (0x1000)
    #define B_AX3_H_INT             (0x0800)
    #define B_AX2_H_INT             (0x0400)
    #define B_AX1_H_INT             (0x0200)
    #define B_AX0_H_INT             (0x0100)
    #define B_DPL_INT               (0x0080)
    #define B_GINP_INT              (0x0040)
    #define B_AX5_L_INT             (0x0020)
    #define B_AX4_L_INT             (0x0010)
    #define B_AX3_L_INT             (0x0008)
    #define B_AX2_L_INT             (0x0004)
    #define B_AX1_L_INT             (0x0002)
    #define B_AX0_L_INT             (0x0001)

//  Bits within GINP_INT_CH
    #define B_EMG_INT               (0x8000)
    #define B_MPG_EMG_INT           (0x4000)
    #define B_GDI10_INT             (0x0400)
    #define B_GDI9_INT              (0x0200)
    #define B_GDI8_INT              (0x0100)
    #define B_GDI7_INT              (0x0080)
    #define B_GDI6_INT              (0x0040)
    #define B_GDI5_INT              (0x0020)
    #define B_GDI4_INT              (0x0010)
    #define B_GDI3_INT              (0x0008)
    #define B_GDI2_INT              (0x0004)
    #define B_GDI1_INT              (0x0002)
    #define B_GDI0_INT              (0x0001)
```

// **Bits within MISC_INT_CH**

```
#define B_TIM_ERR_INT       (0x8000)
#define B_UART_RX_INT       (0x0004)
```

// **Bits within AXISx_INT_CH**

```
#define B_CMP_INT           (0x8000)
#define B_EZ_INT            (0x0200)
#define B_IDX_INT           (0x0100)
#define B_LTC_INT           (0x0080)
#define B_RDY_INT           (0x0040)
#define B_INP_INT           (0x0020)
#define B_ALM_INT           (0x0010)
#define B_SLD_INT           (0x0008)
#define B_ORG_INT           (0x0004)
#define B_MEL_INT           (0x0002)
#define B_PEL_INT           (0x0001)
```

// **for "AD6_Src" in *Set_AD6_Src()***

```
#define AD6_SRC_AGND        (1)
#define AD6_SRC_DA0         (2)
#define AD6_SRC_DA1         (3)
#define AD6_SRC_DA2         (4)
#define AD6_SRC_DA3         (5)
#define AD6_SRC_DA4         (6)
#define AD6_SRC_DA5         (7)
#define AD6_SRC_REF5V       (8)
```

// **for "Mode" in *Set_DDA_Pulse_Mode()*, *Set_ENC_Pulse_Mode()* and *Set_MPG_Pulse_Mode()***

Macro	*Value*	*Description*
PLS_MD_OUTDIR	0	選擇 OUT/DIR 模式
PLS_MD_CWCCW	1	選擇 CW/CCW 模式
PLS_MD_1xAB	2	選擇 A/B phase 模式，倍率為 1
PLS_MD_2xAB	3	選擇 A/B phase 模式，倍率為 2
PLS_MD_4xAB	4	選擇 A/B phase 模式，倍率為 4

// **for "Ch" in *Write_Cal_Data()* and *Read_Cal_Data()***

```
#define DA0_SPAN_CAL_CH  (1)
#define DA1_SPAN_CAL_CH  (2)
#define DA2_SPAN_CAL_CH  (3)
#define DA3_SPAN_CAL_CH  (4)
#define DA4_SPAN_CAL_CH  (5)
#define DA5_SPAN_CAL_CH  (6)
#define AD_OFST_CAL_CH   (7)
#define AD_SPAN_CAL_CH   (8)
#define DA0_OFST_CAL_CH  (11)
#define DA1_OFST_CAL_CH  (12)
#define DA2_OFST_CAL_CH  (13)
#define DA3_OFST_CAL_CH  (14)
#define DA4_OFST_CAL_CH  (15)
#define DA5_OFST_CAL_CH  (16)
#define R5V_OFST_CAL_CH  (17)
```

// **for "DA_Clr_Src" in *Set/Get_DA_Clear_Ctl()* and "DDA_Clr_Src" in *Set/Get_DDA_Clear_Ctl()***

```
#define CLR_SRC_EMG      (0)
#define CLR_SRC_PEL      (1)
#define CLR_SRC_MEL      (2)
```

// **Bits within “Value” of *Set_Axis_IO () and Get_Axis_IO ()***

```
#define B_CMP_SIG        (0x8000)
#define B_ALMRST_SIG     (0x4000)
#define B_ERC_SIG        (0x2000)
#define B_SVON_SIG       (0x1000)
#define B_EA_SIG         (0x0800)
#define B_EB_SIG         (0x0400)
#define B_EZ_SIG         (0x0200)
#define B_IDX_SIG        (0x0100)
#define B_LTC_SIG        (0x0080)
#define B_RDY_SIG        (0x0040)
#define B_INP_SIG        (0x0020)
#define B_ALM_SIG        (0x0010)
#define B_SLD_SIG        (0x0008)
#define B_ORG_SIG        (0x0004)
#define B_MEL_SIG        (0x0002)
#define B_PEL_SIG        (0x0001)
```

// **Bits within “Value” of *Get_DI()***

```
#define B_EMG_SIG        (0x8000)
#define B_MPG_EMG_SIG    (0x4000)
#define B_GDI10_SIG      (0x0400)
#define B_GDI9_SIG       (0x0200)
#define B_GDI8_SIG       (0x0100)
#define B_GDI7_SIG       (0x0080)
#define B_GDI6_SIG       (0x0040)
```

```
#define B_GDI5_SIG          (0x0020)
#define B_GDI4_SIG          (0x0010)
#define B_GDI3_SIG          (0x0008)
#define B_GDI2_SIG          (0x0004)
#define B_GDI1_SIG          (0x0002)
#define B_GDI0_SIG          (0x0001)

//  Bits within "Value" of Set_DO() and Get_DO()
#define B_GDO2_SIG          (0x0004)
#define B_GDO1_SIG          (0x0002)
#define B_GDO0_SIG          (0x0001)

//  for "Ch" in Set/Get_Axis_IO_Filter()
#define EA_FLT_CH           (10)
#define EB_FLT_CH           (9)
#define EZ_FLT_CH           (8)
#define LTC_FLT_CH          (7)
#define RDY_FLT_CH          (6)
#define INP_FLT_CH          (5)
#define ALM_FLT_CH          (4)
#define SLD_FLT_CH          (3)
#define ORG_FLT_CH          (2)
#define MEL_FLT_CH          (1)
#define PEL_FLT_CH          (0)

//  for "Baud" in Set/Get_UART_Baud()
#define BAUD_38400          (3)
#define BAUD_19200          (2)
#define BAUD_9600           (1)
#define BAUD_4800           (0)

//  Bits within UART_Sts
#define B_TX_RDY            (0x10)
#define B_OVERRUN           (0x08)
#define B_FRM_ERR           (0x04)
#define B_PRT_ERR           (0x02)
#define B_RX_RDY            (0x01)

//  for "Ch" in Set/Get_DI_Filter()
#define EMG_FLT_CH          (15)
#define MPG_EMG_FLT_CH      (14)
#define MPG_A_FLT_CH        (13)
#define MPG_B_FLT_CH        (12)
#define GDI10_FLT_CH        (10)
#define GDI9_FLT_CH         (9)
#define GDI8_FLT_CH         (8)
#define GDI7_FLT_CH         (7)
#define GDI6_FLT_CH         (6)
#define GDI5_FLT_CH         (5)
#define GDI4_FLT_CH         (4)
```

```
#define GDI3_FLT_CH          (3)
#define GDI2_FLT_CH          (2)
#define GDI1_FLT_CH          (1)
#define GDI0_FLT_CH          (0)
```

2.函式庫

2.1 DSP初始化函式

/***/
void PMDK_Init ();

Description

初始化 DSP 板，開機時做一次即可。初始化之後其他功能函式才可以使用。呼叫此函式會進行以下的動作:

1. 重置並啟用 DSP 上之快取記憶體.
2. 設定 DSP 與其週邊之時序.
3. 重置 FPGA.
4. 設定中斷訊號之來源 (FPGA 及 FRnet).
5. 將 DSP_OK 設為'1' (主電腦端可經由 PCI bus 讀取 BAR0+0x04, DSP_OK 位在 bit 7 的位置).

Parameters

無

Return Value

無

Example

PMDK_Init();

See Also

2.2 記憶體存取函式

/***/
void Set_Mem (U32 *addr*, U16 *data*);

Description 將 16-bit 資料寫到記憶體指定的位址。

Parameters *addr* 為絕對位址，因為本 DSP 卡每一個位址只存放一個 byte (8-bit)而已，所以 *addr* 必須是偶數。
data 為欲寫入之 16-bit 資料。

Return Value 無

Example

```
Set_Mem(DPRAM_ADDR + 0, 100);
//  在 DPRAM 的開始處寫 100
```

See Also
Get_Mem()

/***/
U16 Get_Mem (U32 *addr*);

Description 在指定的位址讀出一筆 16-bit 資料。

Parameters *addr* 為絕對位址，因為本 DSP 卡每一個位址只存放一個 byte (8-bit)而已，所以 *addr* 必須是偶數。

Return Value 為讀出之 16-bit 資料。

Example

```
U16 data;
data = Get_Mem(DPRAM_ADDR + 6);
// data 為在(DPRAM+6)處所讀出之 16-bit 資料(連續兩個 bytes)
```

See Also
Set_Mem()

/***/
I16 Flash_Erase (U8 *Region*);

Description 刪除 Flash 內指定的區域。

Parameters *Region* 指定要被清除的區域，其定義如下表:

Macro	*Value*	*Description*
FLASH_ERASE_CHIP	0	表示清除 FLASH 全部 範圍：0x00000~0xFFFFF
FLASH_ERASE_CODE	1	表示清除 DSP code 的部分， 範圍：0x00000~0x9FFFF
FLASH_ERASE_RSV	2	表示清除 Reserved 的部分， 範圍：0xA0000~0xAFFFF
FLASH_ERASE_CAL	3	表示清除 AD/DA calibration 的部分， 範圍：0xB0000~0xBFFFF
FLASH_ERASE_FPGA	4	表示清除 FPGA Configuration Data， 範圍：0xC0000~0xFFFFF

Return Value

ERR_ADDR_OUT_RANGE
ERR_HW_ID_ERROR
ERR_TIMEOUT_ERROR
ERR_NO_ERROR

Example

```
Flash_Erase(FLASH_ERASE_FPGA);
//清除 Flash 中的 FPGA Configuration Data
```

See Also

Flash_Program()

/***/

I16 Flash_Program (U32 *Source_addr*, **U32** *Flash_ofst*, **U32** *Num_byte*);

Description

寫入資料到 Flash 內指定的區域。

Parameters

Source_addr 指定欲寫入資料來源的起始位址，但 **Flash** 及 **FRAM** 的位址是不被允許的；
Flash_ofst 指定開始寫入 flash 的 offset 位址；
Num_byte 指定寫入的長度，以 byte 為單位。

Return Value

ERR_ADDR_OUT_RANGE
ERR_TIMEOUT_ERROR
ERR_NO_ERROR

Example

```
Flash_Erase(FLASH_ERASE_FPGA);
//清除 Flash 中的 FPGA Configuration Data
Flash_Program(SDRAM_ADDR+FPGA_DATA_SDRAM_OFS
T, FPGA_DATA_FLASH_OFST, FPGA_DATA_FLASH_SIZE);
//寫入新的 FPGA Configuration Data 到 FLASH
```

See Also

Flash_Program()

/***/

U16 Get_Flash (U32 *offset*);

Description 在 Flash 的相對位址(offset)處讀出一筆 16-bit 資料。

Parameters *offset* 為相對位址。Flash 的 offset 範圍為 0 ~ 0xFFFFF。總共的大小為 1MB，也就是 512K *16-bit。因為本 DSP 卡每一個位址只存放一個 byte (8-bit)而已，所以 offset 必須是偶數(因為是以 16-bit 為操作對象)。

Return Value 為讀出之 16-bit 資料。

Example

```
U16 data;
data = Get_Flash(6);
// data 為在 Flash 上 offset = 6 處所讀出之 16-bit 資料
```

See Also

/***/

void Set_FRAM (U32 *offset*, U16 *data*);

Description 將 16-bit 資料寫到 FRAM 記憶體指定的相對位址處。

Parameters *offset* 為相對位址。FRAM 的 offset 範圍為 0 ~ 0x3FFFF。總共的大小為 256KB，也就是 128K *16-bit。因為本 DSP 卡每一個位址只存放一個 byte (8-bit)而已，所以 offset 必須是偶數(因為是以 16-bit 為操作對象)。
data 為欲寫入之 16-bit 資料，但**只有其 Low byte 會被儲存**。

Return Value 無

Example

```
Set_FRAM(0, 100);
//  在 FRAM 的 offset = 0 處寫入 16-bit  資料(100)
```

See Also

Get_FRAM()

/***/

U16 Get_FRAM (U32 *offset*);

Description 從 FRAM 記憶體指定的相對位址處讀出 16-bit 資料。

Parameters *offset* 為相對位址。FRAM 的 offset 範圍為 0 ~ 0x3FFFF。總共的大小為 256KB，也就是 128K *16-bit。因為本 DSP 卡每一個位址只存放一個 byte (8-bit)而已，所以 offset 必須是偶數(因為是以 16-bit 為操作對象)。

Return Value 為欲讀出之 16-bit 資料，但**只有其 Low byte 才是有效的資料**。

Example

```
U16 data;
data = Get_FRAM(4);
// data 為在 FRAM 上 offset = 4 處所讀出之 16-bit 資料
```

See Also

Set_FRAM()

2.3 FRnet控制函式

/***/

I16 Read_FRnet (U8 *Ch*, U16 **Data*);

Description

讀取 FRnet 的數位輸入資料，每一個 FRnet 的群組號碼(*Ch*)含 16-bit 資料 (16 DI)，總共有 8 個數位輸入群組，所以一個 FRnet 的界面可以定義 128 DI。讀取的動作事實上是讀 FRnet Master 上的暫存器值，如果 FRnet 沒有進行 I/O 的掃瞄動作則暫存器的值是不會更新的, 所以 **Enable_FRnet_Scan()**的動作要執行後 FRnet 才能保證正常工作。

Parameters

Ch 群組號碼 (範圍 8 ~15)。而群組號碼 0 ~ 7 保留給 DO 使用。*Data* 則為屬於指定群組的 16 個 DI 狀態。

Return Value

ERR_ADDR_OUT_RANGE
ERR_NO_ERROR

Example

```
I16 error;
U16 data;
error = Read_FRnet(8, &data);
// data 為在 FRnet CH=8 上的 16-bit 資料
```

See Also

Write_FRnet()，Enable_FRnet_Scan ()，Disable_FRnet_Scan()

/***/

I16 Write_FRnet (U8 *Ch*, U16 *Data*);

Description

寫入 FRnet 的數位輸出資料，每一個 FRnet 的群組號碼(*Ch*)含 16-bit 資料 (16 DO)，總共有 8 個數位輸出群組，所以一個 FRnet 的界面可以定義 128 DO。寫入的動作事實上是寫值到 FRnet Master 上的暫存器，如果 FRnet 沒有進行 I/O 的掃瞄動作則暫存器的值是不會更新到遠端的輸出, 所以要執行 **Enable_FRnet_Scan()**的動作後 FRnet 才能保證正常工作。

Parameters

Ch 群組號碼 (範圍 0 ~ 7)。而群組號碼 8 ~ 15 保留給 DI 使用。*Data* 則為欲寫入指定群組的 16 個 DO 值。

Return Value

ERR_ADDR_OUT_RANGE
ERR_NO_ERROR

Example

```
Write_FRnet (0, 0xFFFF);
```

//將 CH=0 上的 16 個 DO 全部設為 1

See Also

Read_FRnet()，Enable_FRnet_Scan ()，Disable_FRnet_Scan()

/***/

void Enable_FRnet_Scan ();

Description 啟動 FRnet 的掃瞄功能，資料將會定時 (每 0.72 msec) 由硬體自動進行更新。

Parameters 無

Return Value 無

Example

Enable_FRnet_Scan();

See Also

Disable_FRnet_Scan()

/***/

void Disable_FRnet_Scan (void);

Description 停止 FRnet 的掃瞄功能，資料將不會更新到硬體的暫存器。

Parameters 無

Return Value 無

Example

Disable_FRnet_Scan();

See Also

Enable_FRnet_Scan()

/***/

U8 Get_FRnetStatus (void);

Description 讀取 FRnet 輸入群組的通訊狀態。

Parameters 無。

Return Value 每一個 FRnet 系統總共有 8 個輸入群組，群組號碼範圍為 8 ~15，以 Group_8 ~ Group_15 來代表，下表表示各群組的通訊狀態，定義如下表。

Bit	7	6	5	4	3	2	1	0
Function	Group_15	Group_14	Group_13	Group_12	Group_11	Group_10	Group_9	Group_8

當 Group_n (n = 8 ~15)為 0 則表示對應的輸入群組不存在。
當 Group_n (n = 8 ~15)為 1 則表示作用中(硬體模組存在)。
FRnet 的輸出群組號碼範圍是 0~7，輸入群組號碼範圍是 8~15。一般來說，Host 端可以由接收的訊息判斷模組存在與否，但由於是使用硬體負責通訊，傳輸過程中並沒有設計 ACK 功能，所以 Host 端只能判斷輸入模組(Group_8~15)存在與否，也就是接到 DI 模組傳回訊息表示該模組存在。至於 DO 模組是由 Host 端下令給 DO 模組而 DO 模組並沒有 ACK 訊息回傳，所以無法判斷 DO 模組是否存在。這種訊息可以作為判斷是否斷線或有嚴重干擾的參考。

Example

```
U16 data;
data = Get_FRnetStatus();
//  檢查 data 的 bit0~bit7 可以判斷 DI 模組存在情況
```

See Also

Read_FRnet()，Write_FRnet()

/***/

void Set_FRnet_HighSpeed (U8 *Value*);

Description

設定 FRnet 使用的通訊速度 (內定值為 1 Mbps)。此速度與模組設定要一致。

Parameters

Value 指定通訊是否為高速 (1 Mbps)，其定義如下：

0: 250 Kbps
1: 1 Mbps

通訊速度也會影響 FRnet 產生中斷的時間，其定義如下：
250 Kbps : 每 2.88ms 產生一次中斷
1 Mbps: 每 0.72ms 產生一次中斷

Return Value

無

Example

```
Set_FRnet_HighSpeed(1);
//  使用 1 Mbps 的通訊速度來與模組通訊
```

See Also

Set_ FRnet_HighSpeed()

/***/

U8 Get_FRnet_HighSpeed ();

Description 讀取 FRnet 設定的通訊速度。

Parameters 無

Return Value 其定義如下：

0: 目前的通訊速度為 250 Kbps
1: 目前的通訊速度為 1 Mbps

Example

```
U8 HighSpeedStatus;
HighSpeedStatus = Get_FRnet_HighSpeed();
// HighSpeedStatus 為 1，表示目前使用 1 Mbps 的通訊速度
```

See Also

Set_ FRnet_HighSpeed()

2.4 DSP中斷控制函式

/***/
void Set_DSP_Int_Factor (U8 *Value*);

Description 設定 DSP 的中斷因子 (設 enable/disable)

Parameters *Value* 指定中斷因子狀態，可以分成兩個，其定義如下表：

Bit	*Signal*	*Function description*	*Enable/Disable*
0	FPGA_INT	FPGA 來的中斷	1 / 0
1	FRnet_INT	FRnet 來的中斷	1 / 0

在撰寫程式時，可配合使用以下的常數定義

Macro	*Value*	*Description*
B_FPGA_INT	0x01	致能 FPGA 來的中斷 (詳見 FPGA 中斷控制函式)
B_FRNET_INT	0x02	致能 FRnet 來的中斷 (每 0.72 ms 一次)

Return Value 無

Example

```
Set_DSP_Int_Factor(B_FRNET_INT | B_FPGA_INT);
// enable FRnet 及 FPGA 來的中斷
//  與 Set_DSP_Int_Factor(0x03); 同義
```

See Also

Get_DSP_Int_Factor()

/***/
U8 Get_DSP_Int_Factor (void);

Description 讀取 DSP 的中斷因子 (為 enable 或是 disable)

Parameters 無

Return Value 傳回值為目前之中斷因子設定狀態，其定義請參考 **Set_DSP_Int_Factor()** 函式說明。

Example

```
If( (Get_DSP_Int_Factor() & B_FPGA_INT) == B_FPGA_INT)
{ …}
//  判斷在 DSP 的中斷因子中，FPGA 來的中斷是否有 enable
```

See Also

Set_DSP_Int_Factor()

2.5 FPGA初始化函式

/***/

I16 FPGA_Init (U8 *FPGA_Src*, **U16** *Servo_Prd*)

Description 初始化FPGA (將FPGA規劃資料download到FPGA並執行) 與設定伺服控制週期及DDA週期並載入校正裝置之校正值。

Parameters *FPGA_Src* 指定 FPGA codes 的來源，可以有如下的定義:

Macro	*Value*	*Description*
FPGA_SRC_SDRAM	0	由 SDRAM 的 offset 0x00F00000 (FPGA_DATA_SDRAM_OFST)中的資料來規劃 FPGA 。如果 Project 中有包含 LoadFPGAData.C，程式載入時會將 fpgaPMDK.H 中的資料載入到此位址。
FPGA_SRC_FLASH	1	由 FLASH 中儲存的 FPGA Configuration Data 來規劃 FPGA。DSP 會將 FLASH 中儲存的資料 (offset 0xC0000~0xFFFFF)先複製到 SDRAM 的 offset 0x00F00000 的位置 (FPGA_DATA_SDRAM_OFST) 再載入。

Servo_Prd 的值範圍為 3 ~ 32767，內定值為 6250。其定義如下：
伺服週期 = *Value* * 40 ns
所以內定值為 6250 * 40 ns = 250000ns = 250 us

Return Value
ERR_ADDR_OUT_RANGE
ERR_FPGA_DL_FAILED
ERR_VALUE_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = FPGA_Init(FPGA_SRC_FLASH,
                  DEFAULT_DDA_LENGTH);
// 由 Flash 中儲存的 FPGA Configuration Data 來規劃 FPGA
// 伺服控制週期為 6250 * 40ns = 250us
```

See Also
Set_Servo_Period()，Set_DDA_Length()，Write_Cal_Data()

/***/

void Get_FPGA_Version (U16 *_Value_)

Description 取得 FPGA 的版本編號

Parameters *Value* 版本編號數值，其中
low-byte 為 FPGA Version
high-byte 為適用之 PCB Version (或以上)

Bit	7	6	5	4	3	2	1	0
Function	FPGA Version							
Bit	15	14	13	12	11	10	9	8
Function	PCB Version							

Return Value 無

Example

See Also

/***/

I16 Set_Servo_Period (U16 *Value*);

Description 設定伺服控制週期。當修改此設定值時建議同時將所有軸的 DDA 週期作相同的設定。另外，由於此設定在呼叫 **FPGA_Init()** 之時就會與所有軸的 DDA 週期一起同時被設定好，因此若無特別的需要，建議不要修改此設定值。

Parameters *Value* 的值範圍為 3 ~ 32767，內定值為 6250。其定義如下：
伺服控制週期 = *Value* * 40 ns
所以內定值為 6250 * 40 ns = 250000ns = 250 us

Return Value ERR_VALUE_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_Servo_Period(DEFAULT_DDA_LENGTH*2);
//  將伺服控制週期更改為 500 us.
```

See Also

Get_Servo_Period()，FPGA_Init()，Set_DDA_Length()

/***/

void Get_Servo_Period (U16 *_Value_);

Description 讀取卡上伺服控制週期的設定

Parameters *Value* 的值範圍為 3 ~ 32767，請參考 Set_Servo_Period()說明。

Return Value 無

Example

```
U16    period;
Get_Servo_Period(&period);
// Get the setting of servo period parameter
```

See Also

Get_Servo_Period()

2.6 一般用途DIO與MPG訊號函式

/***/
void Get_DI(U16 *_Value_);

Description 讀取一般用途 DI

Parameters 請參考下表的定義，GDI0 ~ GDI10 為一般用途 DI，但是本函式也提供 EMG (emergency stop signal)與 MPG_EMG (MPG 上的 EMG 訊號)的讀取功能。

Bit	7	6	5	4	3	2	1	0
Signal	GDI7	GDI6	GDI5	GDI4	GDI3	GDI2	GDI1	GDI0
Bit	15	14	13	12	11	10	9	8
Signal	EMG	MPG_EMG	-	-	-	GDI10	GDI9	GDI8

在撰寫程式時，可配合使用以下的常數定義：

Macro	***Value***	***Description***
B_GDI0_SIG	0x0001	指定一般 DI channel 0 訊號
B_GDI1_SIG	0x0002	指定一般 DI channel 1 訊號
B_GDI2_SIG	0x0004	指定一般 DI channel 2 訊號
B_GDI3_SIG	0x0008	指定一般 DI channel 3 訊號
B_GDI4_SIG	0x0010	指定一般 DI channel 4 訊號
B_GDI5_SIG	0x0020	指定一般 DI channel 5 訊號
B_GDI6_SIG	0x0040	指定一般 DI channel 6 訊號
B_GDI7_SIG	0x0080	指定一般 DI channel 7 訊號
B_GDI8_SIG	0x0100	指定一般 DI channel 8 訊號
B_GDI9_SIG	0x0200	指定一般 DI channel 9 訊號
B_GDI10_SIG	0x0400	指定一般 DI channel 10 訊號
B_MPG_EMG_SIG	0x4000	指定 MPG Emergency Stop 訊號
B_EMG_SIG	0x8000	指定 Emergency Stop 訊號

Return Value 無

Example

```
U16 di;
Get_DI(&di);
If( (di&B_EMG_SIG) == B_EMG_SIG)
{
    // EMG 訊號的狀態為 1
}
```

See Also

Set_DI_Pol()，Get_DI_Pol()

/***/

void Set_DI_Pol(U16 *Value*);

Description

設定一般用途 DI 的極性 (active low 或 active high；配合 **Set_DA_Clear_Ctl()** 及 **Set_DDA_Clear_Ctl()** 使用) 及有效 change of state 的定義 (rising edge 或 falling edge；配合中斷功能使用)。

Parameters

請參考下表的定義，GDI0 ~ GDI10 為一般用途 DI，但是本函式也提供 EMG (emergency stop signal)與 MPG_EMG (MPG 上的 EMG 訊號)的設定功能。內定值為 1。

Bit	7	6	5	4	3	2	1	0
Signal	GDI7_POL	GDI6_POL	GDI5_POL	GDI4_POL	GDI3_POL	GDI2_POL	GDI1_POL	GDI0_POL
Bit	15	14	13	12	11	10	9	8
Signal	EMG_POL	MPG_EMG_POL	-	-	-	GDI10_POL	GDI9_POL	GDI8_POL

EMG_POL :
‘1’: EMG is active High and rising edge trigger
‘0’: EMG is active Low and falling edge trigger
MPG_EMG_POL :
‘1’: MPG_EMG is active High and rising edge trigger
‘0’: MPG_EMG is active Low and falling edge trigger
GDIx_POL :
‘1’: GDIx is rising edge trigger
‘0’: GDIx is falling edge trigger

Return Value

無

Example

```
Set_DI_Pol(0xCFFF);
// set all DI polarities to be active_high
```

See Also

Get_DI_Pol()，Set_DA_Clear_Ctl()，Set_DDA_Clear_Ctl()，Set_FPGA_Int_Factor()

/***/

void Get_DI_Pol(U16 *_Value_);

Description 讀取一般用途 DI 的極性 (配合 active low 或 active high 在 **Set_DA_Clear_Ctl()** 及 **Set_DDA_Clear_Ctl()** 中的使用) 及影響有效的 change of state 定義 (rising edge 或 falling edge；必須配合中斷功能使用)。

Parameters 讀取值 (*Value*) 請參考 **Set_DI_Pol()**。

Return Value 無

Example

```
U16 Value;
Get_DI_POL(&Value);
// read all DI polarities
```

See Also

Set_DI_POL()

/***/

void Set_DO (U16 *Value*);

Description 設定通用 DO 輸出

Parameters *Value* 的定義請參考下表，每一個 bit 的內定值為 1 (OFF)。

Bit	7	6	5	4	3	2	1	0
Signal	-	-	-	-	-	GDO2	GDO1	GDO0
Bit	15	14	13	12	11	10	9	8
Signal	-	-	-	-	-	-	-	-

GDO2 : Write ‘0’ to turn on the output transistor of GDO2.
GDO1 : Write ‘0’ to turn on the output transistor of GDO1.
GDO0 : Write ‘0’ to turn on the output transistor of GDO0.

在撰寫程式時，可配合使用以下的常數定義：

Macro	***Value***	***Description***
B_GDO0_SIG	0x0001	指定一般 DO channel 0 訊號
B_GDO1_SIG	0x0002	指定一般 DO channel 1 訊號
B_GDO2_SIG	0x0004	指定一般 DO channel 2 訊號

Return Value 無

Example

```
Set_DO( (B_GDO0_SIG | B_GDO1_SIG) ^ 0xFFFF);
// Set GDO0 and GDO1 to 0
//  使 GDO0 與  GDO1 動作
```

See Also

Get_DO()

/***/

void Get_DO (U16 ****Value*);**

Description 讀取通用 DO 輸出的值

Parameters 讀取值 (*Value*) 請參考 **Set_DO()**。

Return Value 無

Example

```
U16 do;
Get_DO(&do);
```

See Also

Set_DO()

/***/

I16 Set_DI_Filter (U8 *Ch*, **U8** *Value***);**

Description 設定通用 DI 輸入的數位低通濾波器參數

Parameters

Ch 的定義如下表，在撰寫程式時，可配合使用以下的常數定義：

Macro Name	***Ch***	***Description***
GDI0_FLT_CH	0	指定通用 DI0 訊號
GDI1_FLT_CH	1	指定通用 DI1 訊號
GDI2_FLT_CH	2	指定通用 DI2 訊號
GDI3_FLT_CH	3	指定通用 DI3 訊號
GDI4_FLT_CH	4	指定通用 DI4 訊號
GDI5_FLT_CH	5	指定通用 DI5 訊號
GDI6_FLT_CH	6	指定通用 DI6 訊號
GDI7_FLT_CH	7	指定通用 DI7 訊號
GDI8_FLT_CH	8	指定通用 DI8 訊號
GDI9_FLT_CH	9	指定通用 DI9 訊號
GDI10_FLT_CH	10	指定通用 DI10 訊號

MPG_B_FLT_CH	12	指定 MPG_B 訊號
MPG_A_FLT_CH	13	指定 MPG_A 訊號
MPG_EMG_FLT_CH	14	指定 MPG_EMG 訊號
EMG_FLT_CH	15	指定卡上的 EMG 訊號

Value 的內定值為 0 (disable，取消該軸專用 DI 輸入數位低通濾波器功能)，數位濾波器定義如下列各表所示。

For MPG_EMG, MPG_A, MPG_B and EMG

Value	Min. Steady Input Width (us)	Input Signal Delay Time (us)
0	disable	disable
1	5.12	3.84
2	40.96	30.72
3	327.68	245.76

For DI0 ~ DI10

Value	Min. Steady Input Width (us)	Input Signal Delay Time (us)
0	disable	disable
1	5.12	3.84

Return Value

ERR_ADDR_OUT_RANGE
ERR_VALUE_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_DI_Filter(7, 1);
// Set the parameter of digital filter of DI7 to be 1 (or ON).
```

See Also

Get_DI_Filter()，Set_Axis_IO_Filter()

/***/

I16 Get_DI_Filter (U8 *Ch*, **U8 ***_Value_**);**

Description 讀取軸專用 DI 輸入的數位低通濾波器參數

Parameters

Ch 與 *Value* 的定義請參考 **Set_DI_Filter()**函式說明。

Return Value

ERR_ADDR_OUT_RANGE
ERR_VALUE_OUT_RANGE

ERR_NO_ERROR

Example

```
U8 value;
Get_DIO_Filter(7, &value);
// Get the value setting of digital filter of DI7.
```

See Also

Set_DI_Filter()

2.7 與運動軸相關的DIO函式

/***/

I16 Set_Axis_IO (U8 *Axis*, **U16** *Value*);

Description 設定軸專用 DO 輸出的值

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Value 的定義請參考下表，每一個 bit 的內定值為 1 (OFF)。

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Function	-	-	-	-	-	-	-	-
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Function	-	ALMRST	ERC	SVON	-	-	-	-

ALMRST : Write '0' to turn on the output transistor of alarm reset signal (ALMRST).
ERC : Write '0' to turn on the output transistor of error counter clear signal (ERC).
SVON : Write '0' to turn on the output transistor of servo on signal (SVON).

在撰寫程式時，可配合使用以下的常數定義：

Macro	***Value***	***Description***
B_SVON_SIG	0x1000	指定一般 servo on 訊號
B_ERC_SIG	0x2000	指定一般 error counter clear 訊號
B_ALMRST_SIG	0x4000	指定一般 alarm reset 訊號

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error, state;
Get_Axis_IO(0, &state);
error = Set_Axis_IO(0, (~B_SVON_SIG) & state);
// Let SVON = 0 (active)
```

See Also

Get_Axis_IO()，Set_Axis_IO_POL()，Get_Axis_IO_POL()

/***/

I16 Get_Axis_IO (U8 *Axis*, **U16 ****Value*);

Description 讀取軸專用 DI 輸入或 DO 狀態值

Parameters

Axis 定義軸，編號為 0 ~ 5。
讀取值 (*Value*) 的定義請參考下表。

Bit	7	6	5	4	3	2	1	0
Function	LTC	RDY	INP	ALM	SLD	ORG	MEL	PEL
Bit	15	14	13	12	11	10	9	8
Function	CMP	ALMRST	ERC	SVON	EA	EB	EZ	IDX

在撰寫程式時，可配合使用以下的常數定義：

Macro	*Value*	*Description*
B_PEL_SIG	0x0001	指定 Plus End Limit 訊號
B_MEL_SIG	0x0002	指定 Minus End Limit 訊號
B_ORG_SIG	0x0004	指定 Original (Home) 訊號
B_SLD_SIG	0x0008	指定 Slow Down 訊號
B_ALM_SIG	0x0010	指定 Alarm 訊號
B_INP_SIG	0x0020	指定 In Position 訊號
B_RDY_SIG	0x0040	指定 Ready 訊號
B_LTC_SIG	0x0080	指定 Latch 訊號
B_IDX_SIG	0x0100	指定 Index 訊號*
B_EZ_SIG	0x0200	指定 Encoder Z 相訊號
B_EB_SIG	0x0400	指定 Encoder B 相訊號
B_EA_SIG	0x0800	指定 Encoder A 相訊號
B_SVON_SIG	0x1000	指定 servo on 訊號
B_ERC_SIG	0x2000	指定 error counter clear 訊號
B_ALMRST_SIG	0x4000	指定 alarm reset 訊號
B_CMP_SIG	0x8000	指定 compare 訊號

*註：Index 訊號的定義，請參考 **Set_Index_Mode()**。

SVON、ERC、ALMRST 與 CMP 屬於 DO 訊號，其餘為 DI 訊號。

ALM、INP 與 RDY 是由馬達驅動器送來的訊號。

PEL、MEL、ORG 與 SLD 是四個感測器訊號。這四個感測器常見被安裝於機器上作為安全設計與歸回原點之用。

Return Value

ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U16 IOstatus;
I16  error;
error = Get_Axis_IO(0, &IOstatus);
```

See Also

Set_Axis_IO()，Set_Axis_IO_Pol()，Get_Axis_IO_Pol()

/***/

I16 Set_Axis_IO_Pol (U8 *Axis*, **U16** *Value*);

Description

設定軸專用 I/O 的極性 (active low 或 active high；配合 **Set_DA_Clear_Ctl()** 及 **Set_DDA_Clear_Ctl()** 使用) 及有效 change of state 的定義 (rising edge 或 falling edge；配合中斷功能使用)。

Parameters

Axis 定義軸，編號為 0 ~ 5。
Value 的定義請參考下表，每一 bit 的內定值為 1。

Bit	7	6	5	4	3	2	1	0
Function	LTC_POL	-	-	-	SLD_POL	ORG_POL	MEL_POL	PEL_POL
Bit	15	14	13	12	11	10	9	8
Function	CMP_POL	-	-	-	-	-	EZ_POL	IDX_POL

CMP_POL : '1': CMP is rising edge trigger
'0': CMP is falling edge trigger
EZ_POL : '1': EZ is rising edge trigger
'0': EZ is falling edge trigger
IDX_POL : '1': Index is rising edge trigger
'0': Index is falling edge trigger
LTC_POL : '1': LTC is rising edge trigger
'0': LTC is falling edge trigger
SLD_POL : '1': LTC is rising edge trigger
'0': LTC is falling edge trigger
ORG_POL : '1': LTC is rising edge trigger
'0': LTC is falling edge trigger
MEL_POL : '1': MEL is active High and rising edge trigger
'0': MEL is active Low and falling edge trigger
PEL_POL : '1': PEL is active High and rising edge trigger
'0': PEL is active Low and falling edge trigger

Return Value

ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_Axis_IO_Pol(0, 0x8380);
// Set the DI polarity of AXIS0.
// SLD, ORG, MEL, and PEL are active low.
```

See Also

Set_Axis_IO()，Get_Axis_IO()，Get_Axis_IO_Pol()

/***/

I16 Get_Axis_IO_Pol (U8 *Axis*, **U16 ****Value***);**

Description 讀取軸專用 I/O 的極性 (active low 或 active high；配合 **Set_DA_Clear_Ctl()** 及 **Set_DDA_Clear_Ctl()** 使用) 及有效 change of state 的定義 (rising edge 或 falling edge；配合中斷功能使用)。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
讀取值 (*Value*) 的定義請參考 **Set_Axis_IO_Pol()**。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U16 IO_polarity;
I16  error;
error = Get_Axis_IO_Pol(0, &IO_polarity);
// read the I/O polarity setting of AXIS0
```

See Also

Set_Axis_IO()，Get_Axis_IO()，Set_Axis_IO_Pol()

/***/

I16 Set_Axis_IO_Filter (U8 *Axis*, **U8** *Ch*, **U8** *Value***);**

Description 設定軸專用 DI 輸入的數位低通濾波器參數

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Ch 的定義如下表：

Ch	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Function	LTC	RDY	INP	ALM	SLD	ORG	MEL	PEL
Ch	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Function	-	-	-	-	-	EA	EB	EZ

在撰寫程式時，可配合使用以下的常數定義：

Macro Name	***Ch***	***Description***
PEL_FLT_CH	0	指定正向極限 PEL (或稱 OT+) 訊號
MEL_FLT_CH	1	指定負向極限 MEL (或稱 OT-) 訊號
ORG_FLT_CH	2	指定原點 Origin (或稱 Home) 訊號
SLD_FLT_CH	3	指定減速 Slow Down 訊號
ALM_FLT_CH	4	指定 Alarm 訊號

INP_FLT_CH	5	指定 In Position 訊號
RDY_FLT_CH	6	指定 Ready 訊號
LTC_FLT_CH	7	指定 Latch 訊號
EZ_FLT_CH	8	指定 encoder 之 EZ 訊號
EB_FLT_CH	9	指定 encoder 之 EB 訊號
EA_FLT_CH	10	指定 encoder 之 EA 訊號

Value 的內定值為 0 (disable，取消該軸專用 DI 輸入數位低通濾波器功能)，數位濾波器定義如下列各表所示。

For EA, EB, EZ, and LTC

Value	Min. Steady Input Width (us)	Input Signal Delay Time (us)
0	disable	disable
1	0.16	0.12
2	0.64	0.48
3	2.56	1.92

For PEL, MEL, ORG, SLD and INP

Value	Min. Steady Input Width (us)	Input Signal Delay Time (us)
0	disable	disable
1	5.12	3.84
2	40.96	30.72
3	327.68	245.76

For ALM and RDY

Value	Min. Steady Input Width (us)	Input Signal Delay Time (us)
0	disable	disable
1	5.12	3.84

Return Value

ERR_AXIS_OUT_RANGE
ERR_ADDR_OUT_RANGE
ERR_VALUE_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_Axis_IO_Filter(0, 7, 1);
// Set the parameter of digital filter of AXIS0's LTC to be 1.
```

See Also

Get_Axis_IO_Filter()

/***/

I16 Get_Axis_IO_Filter (U8 *Axis*, **U8** *Ch*, **U8 ***_Value_**);**

Description 讀取軸專用 DI 輸入的數位低通濾波器參數

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Ch 與 *Value* 的定義請參考 **Set_Axis_IO_Filter()**函式說明。

Return Value ERR_AXIS_OUT_RANGE
ERR_ADDR_OUT_RANGE
ERR_VALUE_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
U8 parameter;
error = Get_Axis_IO_Filter(0, 7, &parameter);
// Get the parameter of digital filter of AXIS0's LTC.
```

See Also

Set_Axis_IO_Filter()

2.8 DA 相關函式

/***/
I16 DA_Update (void);

Description 同時更新所有的 DA 輸出值。

Parameters 無

Return Value ERR_DA_AUTO_UPDATE
ERR_DA_BUSY
ERR_NO_ERROR

Example

```
I16  error;
U8   Busy;
F32 fTest[]={2.5, 5, -2.5, 0, 0, 0};

for(i=0; i<6; i++)
{
      Set_DA_Value(i, fTest[i]);          //設 DA 輸出值
}
do { Get_DA_Busy(&Busy); } while (Busy);
error = DA_Update();
```

See Also
Get_DA_Busy()，DA_Auto_Update()

/***/
void Get_DA_Busy (U8 *_Value_);

Description 取得指定的 DA 狀態是否 busy。呼叫 **DA_Update()**前必須先確認 DA_Busy 的狀態為 0。

Parameters *Value* 如果為 1 則是 busy，為 0 表示 ready。

Return Value 無

Example

```
I16  error;
U8   Busy;
F32 fTest[]={2.5, 5, -2.5, 0, 0, 0};

for(i=0; i<6; i++)
{
      Set_DA_Value(i, fTest[i]);          //設 DA 輸出值
}
```

```
do { Get_DA_Busy(&Busy); } while (Busy);
error = DA_Update();
```

See Also

DA_Update()

/***/

void DA_Auto_Update (U8 *Enable***);**

Description 設定 DA 自動更新狀態為 enable 還是 disable。

Parameters *Enable* 如果為 1 則是會自動更新，所有的 DA 輸出值將會在每個伺服週期開始時同步被更新；為 0 表示需要下命令(使用 **DA_Update()**)來更新。

Return Value 無

Example

```
DA_Auto_Update(1);
```

See Also

DA_Update()

/***/

I16 Clear_DA (U8 *Ch***);**

Description 清除指定 DA 的輸出值 (將其輸出值清除為 0V) 。

Parameters *Ch* 指定一個 DA channel，其值為 0 ~ 5。

Return Value ERR_ADDR_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Clear_DA(0);
```

See Also

/***/

I16 Set_DA_Clear_Ctl (U8 *Ch***, U8** *DA_Clr_Src***, U8** *Enable***);**

Description 可以指定某一軸的 DA 可以因 EMG 或該軸的 PEL 或 MEL 訊號觸發而被清除。

Parameters

Ch 指定一個 DA channel，其值為 0 ~ 5。
DA_Clr_Src 的定義如下表：

Macro	*Value*	*Description*
CLR_SRC_EMG	0	指定 Emergency Stop 訊號
CLR_SRC_PEL	1	指定正向極限 PEL 訊號
CLR_SRC_MEL	2	指定負向極限 MEL 訊號

Enable 如果為 1 表示該軸的 DA 可以被 *DA_Clr_Src* 指定的訊號觸發所清除，如果為 0 則表示不會被清除。

Return Value

ERR_ADDR_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_DA_Clear_Ctl(0, CLR_SRC_EMG, 1);
//設定軸 0 的 DA 會因 EMG 訊號觸發所清除
```

See Also

Get_DA_Clear_Ctl()，Set_DI_Pol()，Set_Axis_IO_Pol()

/***/

I16 Get_DA_Clear_Ctl (U8 *Ch*, U8 *DA_Clr_Src*, U8 **Enable*);

Description

讀取某一軸的 DA 是否已被指定可以因該軸的 EMG，PEL 或 MEL 訊號觸發而被清除。

Parameters

Ch 指定一個 DA channel，其值為 0 ~ 5。
Enable 記載了該軸的 DA 是否可以被 *DA_Clr_Src* 指定的訊號觸發而清除的資訊，其值定義請參考 **Set_DA_Clear_Ctl()** 的說明。

Return Value

ERR_ADDR_OUT_RANGE
ERR_NO_ERROR

Example

```
U8   E_axis;
I16  error;
error = Get_DA_Clear_Ctl (0, CLR_SRC_EMG, &E_axis);
//讀取軸 0 的 DA 是否會因 EMG 訊號觸發所清除
```

See Also

Set_DA_Clear_Ctl()，Set_DI_Pol()，Set_Axis_IO_Pol()

/***/

I16 Set_DA_Data (U8 *Ch*, U16 *Value*);

Description

以 16-bit 整數方式送出指定 channel 的 DA 值，建議以

Set_DA_Value()設定電壓(浮點數值方式)來使用。

Parameters

Ch 指定一個 DA channel，其值為 0 ~ 5。
Value 為 16-bit 整數，真正的電壓輸出範圍為 -10V ~ +10V，但兩者之間的對應關係大致如下表所示：

16-bit 數值	0x0000	0x2000	0x4000	0x6000	0x8000	0xa000	0xc000	0xe000	0xffff
電壓值 (V)	10	7.5	5	2.5	0	-2.5	-5	-7.5	-10

Return Value

ERR_ADDR_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_DA_Data(0, 0x4000);
//設定 DA0  輸出 5V
```

See Also

Get_DA_Data()，Set_DA_Value()，Get_DA_Value()

/***/

I16 Get_DA_Data (U8 *Ch*, U16 **Value*);

Description

以 16-bit 整數方式讀取指定 channel 的 DA 輸出值，建議以 **Get_DA_Value()**讀取電壓(浮點數值方式)來使用。

Parameters

Ch 指定一個 DA channel，其值為 0 ~ 5。
Value 為 16-bit 數值，真正的電壓輸出範圍為 -10V ~ +10V，兩者之間的對應關係請參考 Set_DA_Data()函式說明。

Return Value

ERR_ADDR_OUT_RANGE
ERR_NO_ERROR

Example

```
U16 DA_value;
I16  error;
error = Get_DA_Data(0, &DA_value);
//讀取 DA0  輸出值  (16-bit 整數方式)
```

See Also

Set_DA_Data()，Set_DA_Value()，Get_DA_Value()

/***/

I16 Set_DA_Value (U8 *Ch*, F32 *Value*);

Description 以 32-bit 浮點數的方式送出指定 channel 的 DA 電壓值，這個函式省去了使用者計算真正輸出電壓時的處理。

Parameters *Ch* 指定一個 DA channel，其值為 0 ~ 5。
Value 為 32-bit 浮點數，為電壓輸出數值，範圍為 -10 ~ +10，單位為 V。當 *Value* 小於-10 或大於+10，此函式會將 DA 輸出值設定為-10V 或+10V。

Return Value ERR_ADDR_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_DA_Value(0, 5.0);
//設定 DA0  輸出 5V
```

See Also

Get_DA_Value()，Get_DA_Data()，Set_DA_Data()

/***/

I16 Get_DA_Value (U8 *Ch*, F32 **Value*);

Description 以 32-bit 浮點數方式讀取指定 channel 的 DA 輸出值。

Parameters *Ch* 指定一個 DA channel，其值為 0 ~ 5。
Value 為 32-bit 浮點數數值，為真正的電壓輸出，其範圍為 -10 ~ +10，單位為 V。

Return Value ERR_ADDR_OUT_RANGE
ERR_NO_ERROR

Example

```
F32 DA_value;
I16  error;
error = Get_DA_Value(0, &DA_value);
//讀取 DA0  輸出值  (32-bit 浮點數方式)
```

See Also

Set_DA_Data()，Set_DA_Value()，Get_DA_Value()

2.9 AD 相關函式

/***/
void Set_AD_Start (U8 *Enable*);

Description 啟動 AD 自動轉換。當 AD 自動轉換被啟動後，硬體會以 **250kHz** 的速率依序自動轉換各個 AD channel (由 Channel 0 ~ **Set_AD_Last()** 設定的 channel) 的輸入電壓值。

Parameters *Enable* = 1 ➔ AD 自動轉換開始
Enable = 0 ➔ AD 自動轉換停止 (內定值)

Return Value 無

Example

```
Set_AD_Start(1);
//設定 AD 自動轉換開始
```

See Also Get_AD_Start()，Set_AD_Last()

/***/
void Get_AD_Start (U8 **Enable*);

Description 讀取 AD 轉換是開始或是停止狀態。

Parameters *Enable* = 1 ➔ AD 轉換開始
Enable = 0 ➔ AD 轉換停止

Return Value 無

Example

```
U8 AD_enable;
Get_AD_Start(&AD_enable);
//讀取 AD 轉換是開始或是停止狀態
```

See Also Set_AD_Start()

/***/
I16 Set_AD_Last (U16 *Value*);

Description 當 AD 轉換開始後，AD 轉換的最後一個 AD channel 編號。例如設 5，則 AD0 ~ AD5 都會進行自動轉換。

Parameters *Value* 為 AD 轉換開始後，AD 轉換的最後一個 AD channel 編號。例如設 5，則 AD0 ~ AD5 都會進行自動轉換。可以設定的範圍為 0 ~ 6，內定值為 6。

Return Value ERR_VALUE_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_AD_Last(5);
//設定 AD  轉換 channels 為 AD0 ~ AD5
```

See Also Get_AD_Last()，Set_AD_Start()

/***/

void Get_AD_Last (U16 *_Value_);

Description 取得最後一個自動轉換 AD channel 編號的設定值。

Parameters *Value* 為 AD 轉換開始後，AD 轉換的最後一個 AD channel 編號。請參考 **Set_AD_Last()** 函式說明。

Return Value 無

Example

```
U16 AD_last;
Get_AD_Last(&AD_last);
```

See Also Set_AD_Last()，Set_AD_Start()

/***/

I16 Set_AD6_Src (U8 *AD6_Src*);

Description 設定 AD6 的輸入來源。AD6 內定作為校正 DA 輸出或是 AD 自我校正之用。

Parameters *AD6_Src* 指定 AD6 的輸入源，設定的範圍為 1 ~ 8，內定值為 1。其定義如下：

Macro	*Value*	*Description*
AD6_SRC_AGND	1	指定 Analog Ground 訊號
AD6_SRC_DA0	2	指定 DA Channel 0 的輸出訊號
AD6_SRC_DA1	3	指定 DA Channel 1 的輸出訊號

AD6_SRC_DA2	4	指定 DA Channel 2 的輸出訊號
AD6_SRC_DA3	5	指定 DA Channel 3 的輸出訊號
AD6_SRC_DA4	6	指定 DA Channel 4 的輸出訊號
AD6_SRC_DA5	7	指定 DA Channel 5 的輸出訊號
AD6_SRC_REF5V	8	指定 Ref_5V 訊號*

*註：Ref_5V 為板子上提供的精準 5V 參考電壓。

Return Value

ERR_VALUE_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_AD6_Src(AD6_SRC_DA0);
//設定 AD6  轉換 DA0 之值，可以用來校正 DA0
```

See Also

Get_AD6_Src()

/***/

void Get_AD6_Src (U8 *_AD6_Src_);

Description

讀取 AD6 的輸入來源設定。

Parameters

AD6_Src 為 AD6 的輸入源，其定義請參考 **Set_AD6_Src()** 函式說明。

Return Value

無

Example

```
U8 soruce;
Get_AD6_Src(&source);
```

See Also

Set_AD6_Src()

/***/

I16 Get_AD_Data (U8 *Ch*, U16 **Value*);

Description

以 16-bit 整數方式讀取指定 channel 的 AD 轉換值，建議以 **Get_AD_Value()**讀取電壓(浮點數值方式)來使用。

Parameters

Ch 指定一個 AD channel，其值為 0 ~ 6。
Value 為 16-bit 整數，電壓範圍為 -10V ~ +10V，但兩者之間的對應關係大致如下表所示：

16-bit 數值	0x0000	0x2000	0x4000	0x6000	0x7fff	0x8000	0xa000	0xc000	0xe000	0xffff
電壓值 (V)	-0.0	-2.5	-5.0	-7.5	-10.0	10.0	7.5	5.0	2.5	+0.0

Return Value

ERR_ADDR_OUT_RANGE
ERR_NO_ERROR

Example

```
U16 AD_data;
Get_AD_Data(0, &AD_data);
//AD_data 為 AD0 轉換所得之 16-bit 整數
```

See Also

Get_AD_Value()

/***/

I16 Get_AD_Value (U8 *Ch*, F32 **Value*);

Description

以 32-bit 浮點數方式讀取指定 channel 的 AD 值

Parameters

Ch 指定一個 AD channel，其值為 0 ~ 6。
Value 為 32-bit 浮點數，電壓範圍為 -10V ~ +10V。與 Get_AD_Data()所得資料的關係請參考 Get_AD_Data()函式說明。

Return Value

ERR_ADDR_OUT_RANGE
ERR_NO_ERROR

Example

```
F32 AD_value;
I16  error;
error = Get_AD_Value(0, &AD_value);
//AD_value 為 AD0 轉換所得之 32-bit 浮點數電壓值
```

See Also

Get_AD_Data()

2.10 校正相關函式

/***/

I16 Write_Cal_Data (U8 *Ch*, **U8** *Value*);

Description 將校正值寫入指定的校正裝置。

Parameters *Ch* 指定一個校正裝置，其值為 1 ~ 8 及 11~17。其定義如下：

Macro	*Value*	*Description*
DA0_SPAN_CAL_CH	1	指定 DA Channel 0 的 Span 校正裝置
DA1_SPAN_CAL_CH	2	指定 DA Channel 1 的 Span 校正裝置
DA2_SPAN_CAL_CH	3	指定 DA Channel 2 的 Span 校正裝置
DA3_SPAN_CAL_CH	4	指定 DA Channel 3 的 Span 校正裝置
DA4_SPAN_CAL_CH	5	指定 DA Channel 4 的 Span 校正裝置
DA5_SPAN_CAL_CH	6	指定 DA Channel 5 的 Span 校正裝置
AD_OFST_CAL_CH	7	指定 AD 轉換器的 Offset 校正裝置
AD_SPAN_CAL_CH	8	指定 AD 轉換器的 Span 校正裝置
DA0_OFST_CAL_CH	11	指定 DA Channel 0 的 Offset 校正裝置
DA1_OFST_CAL_CH	12	指定 DA Channel 1 的 Offset 校正裝置
DA2_OFST_CAL_CH	13	指定 DA Channel 2 的 Offset 校正裝置
DA3_OFST_CAL_CH	14	指定 DA Channel 3 的 Offset 校正裝置
DA4_OFST_CAL_CH	15	指定 DA Channel 4 的 Offset 校正裝置
DA5_OFST_CAL_CH	16	指定 DA Channel 5 的 Offset 校正裝置
R5V_OFST_CAL_CH	17	指定 Ref_5V 的 Offset 校正裝置

Value 為校正值，其範圍為 0 ~ 255。

Return Value ERR_TIMEOUT_ERROR
ERR_NO_ERROR

Example

```
I16  error;
error = Write_Cal_Data(AD_OFST_CAL_CH, 128);
//將 128 寫入 AD 轉換器的 Offset 校正裝置
```

See Also

Read_Cal_Data()，Save_All_Cal_Data()

/***/

I16 Read_Cal_Data(U8 *Ch*, **U8 ***Value**);**

Description 讀取指定校正裝置的校正值。

Parameters

Ch 指定一個校正裝置，其定義請參考 **Write_Cal_Data()** 函式說明。
Value 為讀回之校正值，其範圍為 0 ~ 255。

Return Value

ERR_NO_ERROR
ERR_ADDR_OUT_RANGE

Example

```
U8   Cal_value;
I16  error;
error = Read_Cal_Data(AD_OFST_CAL_CH, &Cal_value);
//讀取 AD 轉換器的 Offset 校正裝置的校正值
```

See Also

Write_Cal_Data()，Save_All_Cal_Data()

/***/

I16 Save_All_Cal_Data();

Description

將所有校正裝置的校正值寫入 Flash 中，其位址在 Flash 的 offset 0x000B0000 (CAL_DATA_FLASH_OFST)。當卡片電源被重新打開時，所有 DA 輸出的 Span 校正裝置之校正值將會自動被載入，確保所有 DA 的輸出都是精準的 0V。而其它校正裝置之校正值將會在呼叫 **FPGA_Init()**時被載入。

Parameters

無

Return Value

ERR_ADDR_OUT_RANGE
ERR_HW_ID_ERROR
ERR_TIMEOUT_ERROR
ERR_NO_ERROR

Example

```
I16  error;
error = Save_All_Cal_Data();
```

See Also

Write_Cal_Data()，Read_Cal_Data()

2.11 DDA 相關函式

/***/
void Set_DDA_GEn (U8 *Enable*);

Description 設定 DDA 輸出之 global 控制是否為 enable 狀態。

Parameters *Enable* 之值為 0， 表示為 disable 狀態(內定值)。
Enable 之值為 1， 表示為 enable 狀態。

Return Value 無

Example

```
Set_DDA_GEn(1);
//DDA 輸出之 global 控制是 enable
```

See Also

Get_DDA_GEn()

/***/
void Get_DDA_GEn (U8 **Enable*);

Description 取得 DDA 輸出之 global 控制狀態的設定。

Parameters *Enable* 之值為 0， 表示為 disable 狀態。
Enable 之值為 1， 表示為 enable 狀態。

Return Value 無

Example

```
U8 enable;
Get_DDA_Gen (&enable);
```

See Also

Set_DDA_GEn()

/***/
I16 Get_DDA_Empty (U8 *Axis*, U8 **Value*);

Description 取得指定軸 DDA 命令輸出 buffer 之狀態。若是 buffer 空的，則可以寫新的命令，但不是表示當時沒有 pulse 輸出，請參考 Get_DDA_Busy()函式之說明。

。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Value 之值為 1 則表示 buffer 空的，可以寫新的命令。

Value 之值為 0 則表示 buffer 有數值待輸出，為了維持正常操作，最好不要寫新的命令。

Return Value

ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U8   empty;
I16  error;
do{
    error = Get_DDA_Empty(0, &empty);
} while (empty == 0);
//  會等到第 0 軸的 DDA buffer 空了才能繼續
error = Set_DDA_Data (0, 100);
//  送新的 100 pulses 命令
```

See Also

Get_DDA_Busy()

/***/

I16 Get_DDA_Busy (U8 *Axis*, **U8 ***Value**);**

Description

取得指定軸 DDA 輸出之狀態。

Parameters

Axis 定義軸，編號為 0 ~ 5。
Value 之值為 0 則已經停止脈波輸出。
Value 之值為 1 則表示還在輸出脈波中，軸還不會停止。

Return Value

ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U8   busy;
I16  error;
do{
    error = Get_DDA_Busy(0, &busy);
} while (busy == 1);
//  會等到第 0 軸的 DDA 不再輸出了才能繼續
```

See Also

Get_DDA_Empty()

/***/

I16 Set_DDA_En (U8 *Axis*, **U8** *Enable*);

Description

設定允許/禁止指定軸可以做 DDA 輸出。注意：真正 DDA 是否可以輸出還必須檢查是否下過 Set_DDA_GEn(1);

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Enable 之值為 0 則禁止 DDA 脈波輸出(內定值)。
Enable 之值為 1 則允許 DDA 脈波輸出。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_DDA_En(0, 1);
error = Set_DDA_En(1, 1);
Set_DDA_GEn(1);
//允許第 0 與 1 軸可以做 DDA  輸出
```

See Also
Get_DDA_En()，Set_DDA_GEn();

/***/
I16 Get_DDA_En (U8 *Axis*, **U8 ****Enable***);**

Description 讀取允許/禁止指定軸可以做 DDA 輸出的設定。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Enable 之值為 0 則禁止 DDA 脈波輸出。
Enable 之值為 1 則允許 DDA 脈波輸出。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U8 enable;
I16  error;
error = Get_DDA_En(0, &enable);
```

See Also
Set_DDA_En()，Set_DDA_GEn();

/***/
I16 Set_DDA_Clear_Ctl (U8 *Axis*, **U8** *DDA_Clr_Src*, **U8** *Enable*);

Description 可以指定某一軸的 DDA 可以因該軸的 EMG，PEL 或 MEL 訊號觸發而被清除。

Parameters *Axis* 指定一個 DDA channel，其值為 0 ~ 5。
DDA_Clr_Src 的定義如下表：

Macro	*Value*	*Description*
CLR_SRC_EMG	0	指定緊急停止 EMG 訊號可清除 DDA
CLR_SRC_PEL	1	指定正向極限 PEL 訊號可清除 DDA
CLR_SRC_MEL	2	指定負向 MEL 訊號可清除 DDA

Enable 如果為 1 表示該軸的 DDA 可以被 *DDA_Clr_Src* 指定的訊號觸發所清除，如果為 0 則表示不會被清除。

Return Value

ERR_AXIS_OUT_RANGE
ERR_ADDR_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_DDA_Clear_Ctl(0, CLR_SRC_EMG, 1);
//設定軸 0 的 DDA 會因 EMG 訊號觸發所清除
```

See Also

Get_DDA_Clear_Ctl()，Set_DI_Pol()，Set_Axis_IO_Pol()

/***/

I16 Get_DDA_Clear_Ctl (U8 *Axis*, U8 *DDA_Clr_Src*, U8 **Enable*);

Description

讀取某一軸的 DDA 是否已被指定為可以因該軸的 EMG，PEL 或 MEL 訊號觸發而被清除。

Parameters

Axis 指定一個 DDA channel，其值為 0 ~ 5。
Enable 記載了該軸的 DDA 是否可以被 *DDA_Clr_Src* 指定的訊號觸發而清除的資訊，其值定義請參考 Set_DDA_Clear_Ctl()的說明。

Return Value

ERR_AXIS_OUT_RANGE
ERR_ADDR_OUT_RANGE
ERR_NO_ERROR

Example

```
U8 E_axis;
I16  error;
error = Get_DDA_Clear_Ctl (0, CLR_SRC_EMG, &E_axis);
//讀取軸 0 的 DDA 是否會因 EMG 訊號觸發所清除
```

See Also

Set_DDA_Clear_Ctl()，Set_DI_Pol()，Set_Axis_IO_Pol()

/***/

I16 Set_DDA_Length (U8 *Axis*, U16 *Value*);

Description 可以設定指定軸的 DDA 週期。建議將所有軸的 DDA 週期都與伺服控制週期作相同的設定。另外，由於此設定在呼叫 **FPGA_Init()** 時就會與伺服控制週期一起同時被設定好，因此若無特別的需要，建議不要修改此設定值。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Value 決定 DDA 的週期，單位為 40 ns，範圍為 3 ~ 32767。
計算方式如下：
DDA cycle time = *Value* * 40 ns

Return Value ERR_AXIS_OUT_RANGE
ERR_VALUE_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_DDA_Length(0, DEFAULT_DDA_LENGTH);
//設定第 0 軸 DDA cycle time = 6250 * 40 ns = 250 us
```

See Also
Get_DDA_Length()，FPGA_Init()，Set_Servo_Period()

/***/

I16 Get_DDA_Length (U8 *Axis*, U16 **Value*);

Description 可以取得指定軸目前的 DDA 週期設定。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Value 決定 DDA 的週期，單位為 40 ns，範圍為 3 ~ 32767。
計算方式如下：
DDA cycle time = (*Value*) * 40 ns

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U16 dda_cycle;
I16  error;
error = Get_DDA_Length(0, &dda_cycle);
//第 0 軸 DDA cycle time = dda_cycle * 40 ns
```

See Also
Get_DDA_Length()，FPGA_Init()，Set_Servo_Period()

/***/

I16 Set_DDA_Inv (U8 *Axis*, **U8** *OUT_Inv*, **U8** *DIR_Inv*);

Description 設定指定軸的 DDA 的輸出訊號是否要反相來配合外接驅動器。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
OUT_Inv 如果是 1 則 OUT (或是 CW) 訊號反相。
DIR_Inv 如果是 1 則 DIR (或是 CCW) 訊號反相。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_DDA_Inv(0, 0, 1);
//設定第 0 軸 DDA 的 DIR 訊號反相輸出
```

See Also

Get_DDA_Inv()

/***/

I16 Get_DDA_Inv (U8 *Axis*, **U8 ***_OUT_Inv_, **U8 ***_DIR_Inv_);

Description 取得指定軸的 DDA 的輸出訊號是否反相的設定。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
OUT_Inv 如果是 1 則將 OUT (或是 CW) 訊號反相。
DIR_Inv 如果是 1 則將 DIR (或是 CCW) 訊號反相。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U8   out_inv, dir_inv;
I16  error;
eror = Get_DDA_Inv(0, &out_inv, &dir_inv);
//取得第 0 軸 DDA 的訊號反相輸出設定
```

See Also

Set_DDA_Inv()

/***/

I16 Set_DDA_Swap (U8 *Axis*, **U8** *Enable*);

Description 設定指定軸 DDA 的輸出訊號 *OUT* 與 *DIR* 是否要對調。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Enable 如果是 1 則 OUT (或是 CW)與 DIR (或是 CCW) 訊號對

調 (內定值為 0)。

Return Value

ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_DDA_Swap(0, 1);
//設定第 0 軸 DDA 的 DIR 與 OUT 訊號對調
```

See Also

Get_DDA_Swap()

/***/

I16 Get_DDA_Swap (U8 *Axis*, **U8 ***Enable**);**

Description

取得指定軸 DDA 輸出訊號 OUT (或稱為 CW)與 DIR (或稱為 CCW) 是否要對調的設定。

Parameters

Axis 定義軸，編號為 0 ~ 5。
Enable 如果是 1 則表示 *DIR* 與 *OUT* 訊號對調。

Return Value

ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U8   swap;
I16  error;
error = Set_DDA_Swap(0, &swap);
```

See Also

Get_DDA_Swap()

/***/

I16 Set_DDA_Pulse_Mode (U8 *Axis*, **U8** *Mode*);

Description

設定指定軸 DDA 的輸出脈波模式。

Parameters

Axis 定義軸，編號為 0 ~ 5。
Mode 的定義如下 (內定值為 0)：

Macro	*Value*	*Description*
PLS_MD_OUTDIR	0	選擇 OUT/DIR 模式
PLS_MD_CWCCW	1	選擇 CW/CCW 模式
PLS_MD_1xAB	2	選擇 A/B phase 模式，倍率為 1

Return Value

ERR_AXIS_OUT_RANGE
ERR_VALUE_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_DDA_Pulse_Mode(0, 1);
//設定第 0 軸 DDA 的脈波模式為 CW/CCW
```

See Also

Get_DDA_Pulse_Mode ()

/***/

I16 Get_DDA_Pulse_Mode (U8 *Axis*, **U8 ***Mode**);**

Description

取得指定軸 DDA 輸出脈波模式的設定。

Parameters

Axis 定義軸，編號為 0 ~ 5。
Mode 的定義請參考 **Set_DDA_Pulse_Mode()** 函式說明

Return Value

ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U8 mode;
I16  error;
error = Get_DDA_Pulse_Mode(0, &mode);
```

See Also

Set_DDA_Pulse_Mode ()

/***/

I16 Set_DDA_Width (U8 *Axis*, **U16** *Value*);

Description

設定指定軸 DDA 的輸出脈波波形 high level 的時間。這個設定與脈波波形的 duty cycle 有關。

Parameters

Axis 定義軸，編號為 0 ~ 5。
Value 的範圍為 0 或是 3 ~ 2047。
如果 *Value* = 0，則脈波波形是 50% duty cycle；
如果 *Value* 值為 3 ~ 2047 之間，則表示每一脈波波形
high level 時間 = *Value* * 40 ns

Return Value

ERR_AXIS_OUT_RANGE
ERR_VALUE_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_DDA_Width (0, 0);
//設定第 0 軸 DDA 的脈波為 50 % duty cycle
```

See Also

Get_DDA_Width()，Set_DDA_Length()，Set_DDA_Data()

/***/

I16 Get_DDA_Width (U8 *Axis*, **U16 ***Value**);**

Description 取得指定軸 DDA 的輸出脈波波形 high level 的時間設定資訊。這個時間設定與脈波波形的 duty cycle 有關。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Value 的範圍為 0 或是 3 ~ 2047。
詳細定義請參考 **Set_DDA_Width()** 函式說明。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U16 high_ext;
I16  error;
error = Get_DDA_Width (0, &high_ext);
//取得第 0 軸 DDA 的脈波波形寬度設定資訊
```

See Also

Set_DDA_Width()

/***/

I16 Set_DDA_Data (U8 *Axis*, **I16** *Value*);

Description 設定指定的軸在 DDA 週期(cycle time)內應該要送出的脈波數。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Value 可寫入的範圍為 0 ~ +/-8191。指在 DDA 的週期(cycle time)內應該要送出的脈波數。因為波形的 duty cycle 是由 **Set_DDA_Data()**，**Set_DDA_Width()** 與 **Set_DDA_Length()** 來共同決定的，所以真正可以正常動作的 *Value* 的範圍如下：

當 DDA_Width 的設定為 0 時 (50% duty cycle)：
DDA_Data <= DDA_Length / 4
當 DDA_Width 的設定不為 0 時 (fixed high pulse width)：
DDA_Data <= DDA_Length / (DDA_Width + 3)

Return Value ERR_AXIS_OUT_RANGE

ERR_VALUE_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_DDA_Length(0, DEFAULT_DDA_LENGTH);
//設定第 0 軸 DDA cycle time = 6250 * 40 ns = 250 us
error = Set_DDA_Width (0, 0);
//設定第 0 軸 DDA 的脈波為 50 % duty cycle
error = Set_DDA_Data (0, 1000);
//第 0 軸 DDA 的脈波輸出在 250 us 內要送出 1000 pulses
```

See Also

Get_DDA_Data()，Set_DDA_Width()，Set_DDA_Length()

/***/

I16 Get_DDA_Data (U8 *Axis*, **I16 ***Value**);**

Description 取得指定的軸在 DDA 週期 (cycle time) 內要送出的脈波數。

Parameters

Axis 定義軸，編號為 0 ~ 5。
Value 的範圍為 0 ~ +/-8191。指在 DDA 的週期(cycle time)內應該要送出的脈波數。

Return Value

ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  dda_data;
I16  error;
error = Get_DDA_Data (0, &dda_data);
//取得第 0 軸 DDA  輸出在 cycle time 內要送出脈波數的設定
```

See Also

Set_DDA_Data()，Set_DDA_Width()，Set_DDA_Length()

2.12 Encoder 相關函式

/***/
I16 Clear_ENC_Cntr (U8 *Axis*);

Description 將指定軸的 encoder 計數器值清除為 0

Parameters *Axis* 指定軸編號，其值為 0 ~ 5。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Clear_ENC_Cntr(0);
//清除第 0 軸的 encoder 計數器值為 0
```

See Also

Get_ENC_Cntr()，Set_ENC_Cntr()

/***/
I16 Set_ENC_Inv (U8 *Axis*, **U8** *EA_Inv*, **U8** *EB_Inv*, **U8** *EZ_Inv*);

Description 將指定軸 encoder 的相關訊號反相。當訊號需要反相時，例如馬達轉向與 encoder 增減方向不符時，這樣的功能可以讓配線人員節省將實體配線對調的麻煩。

Parameters *Axis* 指定軸編號，其值為 0 ~ 5。
EA_Inv 寫 1 則 EA 訊號反相，內定為 0。
EB_Inv 寫 1 則 EB 訊號反相，內定為 0。
EZ_Inv 寫 1 則 EZ 訊號反相，內定為 0。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_ENC_Inv(0, 1, 0, 0);
//將第 0 軸 encoder 的 EA 訊號反相
```

See Also

Get_ENC_Inv()

/***/

I16 Get_ENC_Inv (U8 Axis, U8 *_EA_Inv_, U8 *_EB_Inv_, U8 *_EZ_Inv_);

Description 讀取指定軸 encoder 的訊號反相設定。

Parameters *Axis* 指定軸編號，其值為 0 ~ 5。
EA_Inv, *EB_Inv* 與 *EZ_Inv* 分別記錄著 EA, EB 與 EZ 的反相設定，其定義請參考 **Set_ENC_Inv()** 函式說明。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U8   A, B, Z;
I16  error;
error = Get_ENC_Inv(0, &A, &B, &Z);
//A, B, Z 分別記錄著 EA、EB 與 EZ 的反相狀態
```

See Also

Set_ENC_Inv()

/***/

I16 Set_ENC_Pulse_Mode (U8 *Axis*, U8 *Mode*);

Description 設定指定軸 encoder 的訊號模式(mode)。

Parameters *Axis* 指定軸編號，其值為 0 ~ 5。
Mode 的定義如下:

Macro	*Value*	*Description*
PLS_MD_OUTDIR	0	選擇 OUT/DIR 模式
PLS_MD_CWCCW	1	選擇 CW/CCW 模式
PLS_MD_1xAB	2	選擇 A/B phase 模式，倍率為 1
PLS_MD_2xAB	3	選擇 A/B phase 模式，倍率為 2
PLS_MD_4xAB	4	選擇 A/B phase 模式，倍率為 4

Return Value ERR_AXIS_OUT_RANGE
ERR_VALUE_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_ENC_Pulse_Mode(0, PLS_MD_4xAB);
//設 encoder 輸入模式為 4 x AB phase 方式
```

See Also

Get_ENC_Pulse_Mode()

/***/

I16 Get_ENC_Pulse_Mode (U8 *Axis*, U8 **Mode*);

Description 設定指定軸 encoder 的訊號模式(mode)。

Parameters *Axis* 指定軸編號，其值為 0 ~ 5。
Mode 的定義請參考 **Get_ENC_Pulse_Mode ()** 函式說明。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U8   mode;
I16  error;
error = Get_ENC_Pulse_Mode(0, &mode);
//取得 pulse mode 值
```

See Also

Get_ENC_Pulse_Mode()

/***/

I16 Set_ENC_Cntr (U8 *Axis*, I32 *Value*);

Description 設定指定軸 encoder 計數器的值。

Parameters *Axis* 指定軸編號，其值為 0 ~ 5。
Value 為欲設定之 encoder 計數器的值，其值可以為任一 32-bit 有符號整數。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_ENC_Cntr(0, 0);
//將 encoder 計數器的值設為 0
//此時與 Clear_ENC_Cntr(0) 的作用相同
```

See Also

Get_ENC_Cntr(), Clear_ENC_Cntr()

/***/

I16 Get_ENC_Cntr (U8 *Axis*, I32 **Value*);

Description 讀取指定軸 encoder 計數器的值。

Parameters

Axis 指定軸編號，其值為 0 ~ 5。
Value 為讀取到之 encoder 計數器的值。

Return Value

ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I32  value;
I16  error;
error = Get_ENC_Cntr(0, &value);
//value 為第 0 軸 encoder 計數器的值
```

See Also

Set_ENC_Cntr()

/***/

I16 Set_ENC_Vring (U8 *Axis*, U32 *Value*);

Description

將指定軸 encoder 計數器變成一個環狀計數器 (Ring Counter)，並設定其最大值。

Parameters

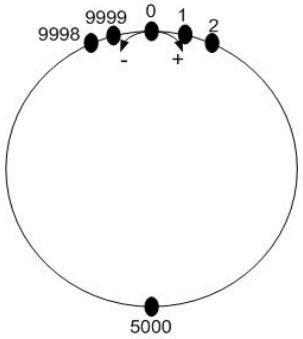
Axis 指定軸編號，其值為 0 ~ 5。
Value 的值為 31-bit 正整數 (0 ~ 0x7FFFFFFF)，內定值為 0x7FFFFFFF。

Return Value

ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_ENC_Vring(0, 9999);
//設第 0 軸 encoder 計數器為一個環狀計數器，最大值為 9999
```



例如：
設計轉一圈為 10000 Pulse，環狀計數器值設為 9999。正轉到 9999 後下一 Pulse 歸為 0，1…重新計算起

See Also

Get_ENC_Vring()

/***/

I16 Get_ENC_Vring (U8 *Axis*, **U32 ***Value**);**

Description 讀取指定軸環狀計數器 (Ring Counter)的設定值。

Parameters *Axis* 指定軸編號，其值為 0 ~ 5。
Value 的值為 31-bit 正整數(0 ~ 0x7FFFFFFF)。詳細定義請參考 **Set_ENC_Vring()** 函式說明。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U32 ring;
I16  error;
error = Get_ENC_Vring(0, &ring);
```

See Also

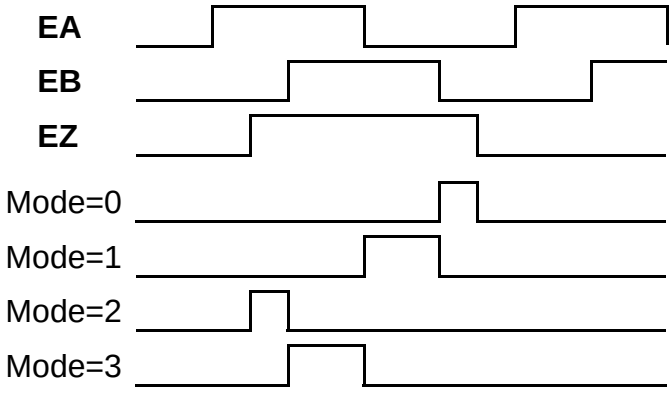
Set_ENC_Vring()

/***/

I16 Set_Index_Mode (U8 *Axis*, **U8** *Mode*);

Description 設定 **Index** 訊號模式。Index 訊號是編碼器 EZ 訊號的變形，用來定位出更加精準的位置。

Parameters *Axis* 指定軸編號，其值為 0 ~ 5。
Mode 的定義請參考下圖，其中 EA、EB 與 EZ 為編碼器訊號。
Index 有 4 種模式(mode = 0 ~ 3)，其它無定義。內定值為 0。



Return Value ERR_AXIS_OUT_RANGE
ERR_VALUE_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
```

```
error = Set_Index_Mode(0, 1);
```

See Also

Get_Index_Mode，Get_Axis_IO()，Set_Axis_IO_Pol()，
Set_Axis_IO_INT_En()

/***/

I16 Get_Index_Mode (U8 *Axis*, **U8 ***Mode**);**

Description

讀取 **Index** 訊號目前的模式設定。Index 訊號是編碼器 EZ 訊號的變形，可以用來定位出更加精準的原點位置。

Parameters

Axis 指定軸編號，其值為 0 ~ 5。
Mode 的定義請參考 **Set_Index_Mode()** 函式說明。

Return Value

ERR_AXIS_OUT_RANGE
ERR_VALUE_OUT_RANGE
ERR_NO_ERROR

Example

```
U8   mode
I16  error;
error = Get_Index_Mode(0, &mode);
```

See Also

Set_Index_Mode

2.13 Manual Pulse Generator 相關函式

/***/
void Set_MPG_EMG_En (U8 *Enable*);

Description 設定 MPG 上的 EMG 訊號 (MPG_EMG) 是否做為 EMG (emergency stop), 可使用來控制清除 DA 與 DDA。

Parameters *Enable* 為 0，則沒有控制作用 (內定值)。
Enable 為 1，則是有控制作用。

Return Value 無

Example

```
Set_MPG_EMG_En(1);
//設定 MPG 上的 EMG 按鍵是有控制作用
```

See Also

Get_MPG_EMG_En()，Set_DA_Clear_Ctl()，
Set_DDA_Clear_Ctl()

/***/
void Get_MPG_EMG_En (U8 **Enable*);

Description 讀取 MPG 上的 EMG 訊號 (MPG_EMG) 設定是否做為 EMG (emergency stop) 使用 (用來控制清除 DA 與 DDA)。

Parameters *Enable* 為 0，則沒有控制作用。
Enable 為 1，則是有控制作用。

Return Value 無

Example

```
U8 enable;
Get_MPG_EMG_En(&enable);
//讀取 MPG 上的 EMG 按鍵是否有控制作用的設定
```

See Also

Set_MPG_EMG_En()，Set_DA_Clear_Ctl()，
Set_DDA_Clear_Ctl()

/***/

I16 Set_MPG_Inv (U8 *EA_Inv*, U8 *EB_Inv*);

Description 設定 MPG 輸入的相關訊號需要反相與否。當訊號需要反相時可以使用此函式，例如馬達轉向與 MPG 增減方向不符時。這樣的功能可以讓配線人員節省將實體配線對調的麻煩。

Parameters *EA_Inv* 寫 1 則 EA 訊號反相，內定為 0。
EB_Inv 寫 1 則 EB 訊號反相，內定為 0。

Return Value ERR_NO_ERROR

Example

```
I16  error;
error = Set_MPG_Inv(0, 1);
//將 MPG 的 EB 輸入訊號反相
```

See Also

Get_MPG_Inv()

/***/

void Get_MPG_Inv (U8 **EA_Inv*, U8 **EB_Inv*);

Description 讀取 MPG 的輸入訊號反相設定。

Parameters *EA_Inv* 與 *EB_Inv* 分別記錄著 MPG 輸入訊號 EA 與 EB 的反相設定，其定義請參考 **Set_MPG_Inv()** 函式說明。

Return Value 無

Example

```
U8 A, B;
Get_MPG_Inv(&A, &B);
//A, B 分別記錄著 MPG 的 EA 與 EB 的反相狀態
```

See Also

Set_MPG_Inv()

/***/

I16 Set_MPG_Pulse_Mode (U8 *Mode*);

Description 設定 MPG 的輸入介面訊號解碼模式 (mode)。

Parameters *Mode* 的定義如下:

Macro	*Value*	*Description*
PLS_MD_OUTDIR	0	選擇 OUT/DIR 模式
PLS_MD_CWCCW	1	選擇 CW/CCW 模式
PLS_MD_1xAB	2	選擇 A/B phase 模式，倍率為 1
PLS_MD_2xAB	3	選擇 A/B phase 模式，倍率為 2
PLS_MD_4xAB	4	選擇 A/B phase 模式，倍率為 4

Return Value ERR_VALUE_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_MPG_Pulse_Mode(PLS_MD_4xAB);
//設 MPG 介面輸入模式為 4 x AB phase 方式
```

See Also

Get_MPG_Pulse_Mode()

/***/

void Get_MPG_Pulse_Mode (U8 **Mode*);

Description 讀取 MPG 的輸入介面訊號解碼模式 (mode) 設定。

Parameters 請參考 **Set_MPG_Pulse_Mode()** 函式說明。

Return Value 無

Example

```
U8 mode;
Get_MPG_Pulse_Mode(&mode);
//取得 mode 設定
```

See Also

Get_ENC_Pulse_Mode()

/***/

void Get_MPG_Cntr (I8 *_Value_);

Description 讀取 MPG 計數器上儲存的伺服控制位移命令資料。當設定使用 MPG 來控制運動軸時，每一個伺服控制週期開始時內部硬體都會讀取 MPG 計數器之值，此值被當成該週期的位移命令。這個值只能再 ISR 內讀取，離開會會被清為 0。

Parameters *Value* 為儲存的計數資料。

Return Value 無

Example

```
I8 cmd;
Get_MPG_Cntr(&cmd);
```

See Also

2.14 Compare Functions

/***/
I16 Set_Direct_CMP (U8 *Axis*, **U8** *Enable*);

Description

如果 **Set_CMP_En(**x, 0**)**，則可以將 CMP 視為一般 DO 來控制，直接設定 CMP 訊號的輸出狀態(亦即直接控制 CMP 的輸出)。如果已經利用 **Set_CMP_En(**x, 1**)** 設定 CMPx 成為位置比較之 DO 輸出，則此設定功能並不會發生作用，只有回應錯誤訊息。請參考 **Set_CMP_En()** 函式說明。

Parameters

Axis 定義軸，編號為 0 ~ 5。
Enable 指定輸出的狀態，其定義如下：
0: turn on the output transistor of CMP signal。
1: turn off the output transistor of CMP signal，此為內定值。

Return Value

ERR_AXIS_OUT_RANGE
ERR_CMP_INUSE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_CMP_En(0, 0);
// disable position comparison of AXIS0
error = Set_Direct_CMP(0, 1);
// turn off the output transistor of CMP signal
```

See Also

Get_Direct_CMP()，Set_CMP_En()

/***/
I16 Get_Direct_CMP (U8 *Axis*, **U8 ***Enable**);**

Description

讀取 Direct_CMP 的設定值。請參考 set_Direct_CMP() 函式說明。

Parameters

Axis 定義軸，編號為 0 ~ 5。
Enable 的內容請參考 Set_Direct_CMP() 函式說明。

Return Value

ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16    value;
error = Get_Diret_CMP(0, &value);
```

See Also

Set_Direct_CMP()

/***/

I16 Set_CMP_En (U8 *Axis*, **U8** *Enable*);

Description 設定 CMP 是否當位置比較專用之 DO 輸出。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Enable 為 1 時，CMP 訊號定義為位置比較專用之 DO 輸出。
Enable 內定值為 0，此時可以利用 Set_Direct_CMP()來設定 CMP 輸出的狀態。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_CMP_En(0, 1);
// CMP 訊號為位置比較專用之 DO 輸出
```

See Also

Get_CMP_En()，Set_Direct_CMP()

/***/

I16 Get_CMP_En (U8 *Axis*, **U8 ****Enable*);

Description 讀取 CMP 是否被設定成位置比較專用之 DO 輸出。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Enable 為 1 時，CMP 訊號定義為位置比較之 DO 輸出。
Enable 為 0 時，可以利用 Set_Direct_CMP()來設定 CMP 輸出的狀態。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U8   CMP_en;
I16  error;
error = Get_CMP_En(0, &CMP_en);
//  讀取 AXIS0  之 CMP 訊號是否被設定成位置比較專用之輸出
```

See Also

Set_CMP_En()，Set_Direct_CMP()

/***/

I16 Set_CMP_Pol (U8 *Axis*, **U8** *CMP_Pol*);

Description 設定 CMP 輸出的極性(high pulse 或 low pulse 輸出)。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
CMP_Pol = 1 ➔ CMP 訊號輸出為 high pulse。
CMP_Pol = 0 ➔ CMP 訊號輸出為 low pulse，此為內定值。
Pulse 的寬度請參考 Set_CMP_Width()設定。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_CMP_Pol(0, 1);
// AXIS0  的 CMP 訊號輸出為 high pulse
```

See Also

Set_CMP_Width()，Get_CMP_Pol()

/***/

I16 Get_CMP_Pol (U8 *Axis*, **U8 ***_CMP_Pol_**);**

Description 讀取 CMP 輸出的極性設定。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
CMP_Pol = 1 ➔ CMP 訊號輸出為 high pulse。
CMP_Pol = 0 ➔ CMP 訊號輸出為 low pulse。
Pulse 的寬度請參考 Set_CMP_Width()設定。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U8   CMP_pol;
I16  error;
error = Get_CMP_Pol(0, &CMP_pol);
//  讀取 AXIS0  的 CMP 訊號輸出設定
```

See Also

Set_CMP_Width()，Set_CMP_Pol()

/***/

I16 Set_CMP_Inc_En (U8 *Axis*, **U8** *Enable*);

Description

設定位置比較器的比較值是否會自動累加。此功能適用於高速之等間距比較觸發。每次增加量由 **Set_CMP_Inc()**函式來設定。

Parameters

Axis 定義軸，編號為 0 ~ 5。
Enable 為 1 時，位置比較器的比較值會依據 **Set_CMP_Inc()** 設定的值自動累加。
Enable 為 0 時，比較值不會自動累加，此為內定值。

Return Value

ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_CMP_Inc(0, 100);
error = Set_CMP_Inc_En(0, 1);
// AXIS0 的位置比較器會自動累加 100 個 pulses
```

See Also

Get_CMP_Inc_En()，Set_CMP_Inc()

/***/

I16 Get_CMP_Inc_En (U8 *Axis*, **U8 ***Enable*);

Description

讀取位置比較器的比較值是否會自動累加的設定。此功能適用於高速等間距比較觸發。

Parameters

Axis 定義軸，編號為 0 ~ 5。
Enable 為 1 時，位置比較器的比較值會依據 **Set_CMP_Inc()** 設定的值自動累加。
Enable 為 0 時，比較值不會自動累加。

Return Value

ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U8   value
I16  error;
error = Get_CMP_Inc_En(0, &value);
//  檢查 AXIS0 的位置比較器是否會自動累加
```

See Also

Set_CMP_Inc_En()

/***/

I16 Set_CMP_Width (U8 *Axis*, **U8** *Value*);

Description 設定 CMP 輸出的脈波寬度參數。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Value 的範圍為 0 ~ 15，與脈波寬度的關係為：
如果 *Value* = 0 (內定值) ➔ 真實脈波的寬度(即不調整)
Value 為其它值，則脈波的寬度為：
(1 * 2^(*Value*+1) * 40ns) ~ (2 * 2^(*Value*+1) * 40ns)

Return Value ERR_AXIS_OUT_RANGE
ERR_VALUE_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_CMP_Width(0, 1);
//設定 CMP 輸出的脈波寬度參數為 1 (160ns ~ 320ns)
```

See Also

Get_CMP_Width()

/***/

I16 Get_CMP_Width (U8 *Axis*, **U8 ****Value*);

Description 讀取 CMP 輸出的脈波寬度參數設定。

Axis 定義軸，編號為 0 ~ 5。
Value 的範圍為 0 ~ 15，請參考 **Set_CMP_Width()** 函式說明。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U8   width;
I16  error;
error = Get_CMP_Width(0, &width);
//讀取 AXIS0 的 CMP 輸出的脈波寬度參數
```

See Also

Set_CMP_Width()

/***/

I16 Set_CMP_Data (U8 *Axis*, **I32** *Value*);

Description 設定比較暫存器的值，每一軸有一個 32-bit 有正負號整數之比較暫存器。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Value 的範圍為 32-bit 有正負號整數。當該軸之 encoder 值與此軸之比較暫存器相等時，會送出 CMP 輸出 (當 **Set_CMP_En()** 設定為 enable 時)。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_CMP_Data(0, 20000);
//設定 AXIS0 的比較暫存器的值為 20000
```

See Also

Get_CMP_Data()

/***/

I16 Get_CMP_Data (U8 *Axis*, **I32 ***Value**);**

Description 讀取比較暫存器的設定值，每一軸有一個 32-bit 有正負號整數之比較暫存器。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Value 為目前比較暫存器之值。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I32  cmp_data;
I16  error;
error = Get_CMP_Data(0, &cmp_data);
```

See Also

Set_CMP_Data()

/***/

I16 Set_CMP_Inc (U8 *Axis*, **I16** *Value*);

Description 設定位置比較器的累加值。此功能適用於高速等間距比較觸發。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Value 的範圍為 16-bit 有正負號整數。當該軸之 encoder 值與此軸之比較暫存器相等時，會送出 CMP 輸出 (當 **Set_CMP_En()** 設定為 enable 時)。若此時 **Set_CMP_Inc_En()** 的設定值為 1 (亦即 enable)，則此時比較暫存器的值會被加上此函式所設定的值。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_CMP_Inc(0, 100);
error = Set_CMP_Inc_En(0, 1);
// AXIS0 的位置比較器會自動累加 100 個 pulses
```

See Also

Get_CMP_Inc()，Set_CMP_Inc_En()

/***/

I16 Get_CMP_Inc (U8 *Axis*, **I16 ***Value**);**

Description 讀取位置比較器的累加值。此功能適用於高速等間距比較觸發。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Value 為增量。請參考 **Set_CMP_Inc()** 函式說明。

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  value
I16  error;
error = Get_CMP_Inc(0, &value);
//  讀取 AXIS0 位置比較器的累加值
```

See Also

Set_CMP_Inc()

/***/

I16 Set_CMP_Out_Src (U8 *Axis*, U8 *Value*);

Description

設定實體的 CMPx pin (68-pin connector 上的接腳) 由哪個內部 CMP 輸出所驅動。利用此函式可以同時 (利用單一個位置比較器) 控制多個實體 CMPx 輸出腳。若沒有此特殊需求，則不建議使用此函式。

Parameters

Axis 定義軸，編號為 0 ~ 5。
Value 為內部 CMP 訊號之編號，範圍同樣是 0 ~ 5。其內定值為該軸的軸號 (即軸 0 的內定值為 0，軸 1 的內定值為 1…)。

Return Value

ERR_AXIS_OUT_RANGE
ERR_VALUE_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_CMP_Out_Src(0, 1)
//  設定內部 CMP1 訊號驅動外部 CMP0 接腳
```

See Also

Get_CMP_Out_Src()

/***/

I16 Get_CMP_Out_Src (U8 *Axis*, U8 **Value*);

Description

讀取實體的 CMPx pin (68-pin connector 上的接腳) 由哪個內部 CMP 輸出所驅動的設定。

Parameters

Axis 定義軸，編號為 0 ~ 5。
Value 為內部 CMP 訊號之編號。

Return Value

ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U8 value;
I16  error;
error = Get_CMP_Out_Src(0, &value)
//  讀取實體 CMP0 接腳的驅動來源 (value)
```

See Also

Set_CMP_Out_Src()

2.15 Latch相關函式

/***/
I16 Set_LTC_En (*U8 Axis,* U8 *Enable*);

Description 設定指定軸是否進行 LATCH 功能(enable 或是 disable)。LATCH 功能的參考訊號來源為各軸的 LTC 數位輸入(軸專用 DI，請參考 **Get_Axis_IO ()** 的說明)，而條件成立時的 encoder 值會被存入 LATCH 暫存器。如果這功不被致能，則 LTC 變成一個一般的 DI，其訊號變動不會讓 encoder 值存入 LATCH 暫存器。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Enable = 0 ➔ 不進行 LATCH 功能 (內定值)
Enable = 1 ➔ 進行 LATCH 功能

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_LTC_En(0, 1);
//設定 AXIS0 進行 LATCH 功能
```

See Also

Get_LTC_En()，Get_Axis_IO ()

/***/
I16 Get_LTC_En (U8 *Axis*, U8 **Enable*);

Description 讀取指定軸是否進行 LATCH 功能的設定。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
Enable 值的定義:
0 ➔ 沒有進行 LATCH 功能
1 ➔ 有設定進行 LATCH 功能

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U8   enable;
I16  error;
error = Get_LTC_En(0, &enable);
//檢查 AXIS0 是否設定進行 LATCH 功能
```

See Also

Set_LTC_En()

/***/

I16 Set_LTC_Pol (U8 *Axis*, **U8** *LTC_Pol*);

Description 設定指定軸 LTC 輸入的極性(active high 或 active low)。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
LTC_Pol 值的定義:
0 ➔ low active (內定值)
1 ➔ high active

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_LTC_Pol(0, 1);
//改變 AXIS0 的 LTC 輸入極性為 high active
```

See Also

Get_LTC_Pol()

/***/

I16 Get_LTC_Pol (U8 *Axis*, **U8 ***_LTC_Pol_**);**

Description 讀取指定軸 LTC 輸入極性的設定。

Parameters *Axis* 定義軸，編號為 0 ~ 5。
LTC_Pol 值的定義:
0 ➔ low active
1 ➔ high active

Return Value ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U8   polarity;
I16  error;
error = Set_LTC_Pol(0, &polarity);
//polarity 存 AXIS0 之 LTC 的極性設定值
```

See Also

Set_LTC_Pol()

/***/

I16 Get_LTC_Data (U8 *Axis*, I32 **Value*);

Description

讀取指定軸 LATCH 暫存器的值。LATCH 動作會受 Set_LTC_En() 設定的影響，必須是 enable 才可。

Parameters

Axis 定義軸，編號為 0 ~ 5。
Value 值儲存著 LTC 數位輸入觸發時的 encoder 值，此值為有正負號之 32-bit 資料。

Return Value

ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U32 latch_data;
I16  error;
error = Get_LTC_Data(0, &latch_data);
//latch_data 存 AXIS0 之 LATCH 暫存器值
```

See Also

Set_LTC_En()，Set_LTC_Pol()

/***/

I16 Get_LTC_Error (U8 *Axis*, U8 **Value*);

Description

若是 latch 住的值，在被讀取之前，又接收到另一個 latch 訊號，此時新的 latch 住的值會覆蓋掉舊的值, LTC_Error 的值就會被設為 1 (MY_TRUE)。當 DSP 讀取 latch 值時, LTC_Error 的值就會被清除為 0 (MY_FALSE)。

Parameters

Axis 定義軸，編號為 0 ~ 5。
Value 值為 1: 有 overrun error；
0: latch 值是可用的。

Return Value

ERR_AXIS_OUT_RANGE
ERR_NO_ERROR

Example

```
U8  latch_error;
Get_LTC_Data (0, &latch_error);
```

See Also

Set_LTC_En()，Set_LTC_Pol()，Get_LTC_Data()

2.16 UART 相關函式

/***/
I16 Set_UART_Baud (U8 *Baud*);

Description 設定 UART 的 baud rate。UART 的資料格式為 E81 (Even parity，8 data bits，1 stop bit)。

Parameters *Baud* 指定 UART 的 baud rate，設定的範圍為 0 ~ 3，內定值為 0。其定義如下：

Macro	*Value*	*Description*
BAUD_4800	0	指定 baud rate 為 4800
BAUD_9600	1	指定 baud rate 為 9600 (建議使用*)
BAUD_19200	2	指定 baud rate 為 19200
BAUD_38400	3	指定 baud rate 為 38400

*註： 因為板子上的 clock 無法整除出以上 baud rate 所需的 clock，所以實際的 baud rate 會有些許的誤差。因此建議若是要連續傳輸大量資料時，建議使用 9600 或 4800 baud rate，或是在傳送時兩筆資料中間加上一點 delay。

Return Value ERR_VALUE_OUT_RANGE
ERR_NO_ERROR

Example

```
Set_UART_Baud (BAUD_9600);
// 將 baud rate 設為 9600
```

See Also

Get_UART_Baud()

/***/
void Get_UART_Baud (U8 **Baud*);

Description 讀回目前 UART baud rate 的設定值。

Parameters *Baud* 為讀回之資料，其定義請參考 **Set_UART_Baud()** 函式說明。

Return Value 無

Example

```
U8  data;
Get_UART_Baud(&data);
```

See Also

Set_UART_Baud()

/***/

void Get_UART_Sts (U8 *_Value_);

Description 讀回 UART 控制器的狀態。

Parameters *Value* 為讀回之資料，其定義如下表：

Bit	7	6	5	4	3	2	1	0
Function	-	-	-	TxRdy	Overrun	FrmErr	PrtErr	RxRdy

在撰寫程式時，可配合使用以下的常數定義：

Macro	***Value***	***Description***
B_RX_RDY	0x01	此 bit 為 1 時，接收暫存器有資料等待被讀取。
B_PRT_ERR	0x02	此 bit 為 1 時，表示目前接收暫存器中的資料在接收時有發生 parity error。
B_FRM_ERR	0x04	此 bit 為 1 時，表示目前接收暫存器中的資料在接收時有發生有格式錯誤 (stop bit 不為 1)。
B_OVERRUN	0x08	此 bit 為 1 時，表示接收暫存器中的資料在被讀取前又有一筆新的資料被接收並覆蓋舊的資料。
B_TX_RDY	0x10	此 bit 為 1 時，傳送暫存器允許新資料被寫入。

Return Value 無

Example

```
U8  status, data;
do{
    Get_UART_Sts(&status);
} while( (status & B_RX_RDY) != B_RX_RDY);
// 等待資料傳入 UART 接收暫存器
If(   ( (status & B_PRT_ERR)   != B_PRT_ERR )   &&
      ( (status & B_FRM_ERR)   != B_FRM_ERR) )
// 確認此筆資料沒有錯誤發生，為有效之資料
{
    Get_UART_Data(&data);
    // 讀取 UART 接收暫存器中的資料
    do{
        Get_UART_Sts(&status);
    } while( (status & B_TX_RDY) != B_TX_RDY);
    // 等待 UART 傳送暫存器能接受新的資料
    Set_UART_Data(data);
    // 將資料寫入 UART 傳送暫存器
}
// 以上程式使用了 do~while 方式來檢驗狀態，因為會阻礙
// 其他程式的運行，所以並不適合放在 timer ISR 內執行。
// 如果在 timer ISR 內，請用 if 來改寫所有程式。
```

See Also

Set_UART_Baud()

/***/

void Set_UART_Data (U8 *Value*);

Description 將資料寫入 UART 的傳送暫存器。

Parameters *Value* 為欲寫入之資料，可以是任何 8 位元資料。

Return Value 無

Example

```
U8   status, data = 1;

Get_UART_Sts(&status);
If ( status & B_TX_RDY)    Set_UART_Data(data);
// UART 傳送暫存器能接受新的資料才傳出 data
//  否則就放棄。在 timer ISR 內適合這種寫法。
```

See Also

Get_UART_Data(), Get_UART_Sts()

/***/

void Get_UART_Data (U8 **Value*);

Description 讀出 UART 接收暫存器中的資料。

Parameters *Value* 為讀回之資料。

Return Value 無

Example

```
U8   status, data;

Get_UART_Sts(&status);
if (status & B_RX_RDY))    Get_UART_Data(&data);
//  如果資料已傳入 UART 接收暫存器, 則去讀取 data
//  否則就放棄。在 timer ISR 內適合這種寫法。
```

See Also

Set_UART_Data(), Get_UART_Sts()

2.17 FPGA中斷控制函式

/***/

I16 Set_FPGA_Int_Factor (U8 *Ch*, U16 *Value*);

Description 設定指定 channel 的中斷因子 (設 enable/disable)。

Parameters *Ch* 指定的 channel，可以分成五大類，*Value* 依照不同 *Ch* 種類可以有不同定義。

第一大類: *Ch* 為設定各軸 I/O 訊號產生中斷之用 **(0~5)**，其定義如下表:

Channel Name	***實際數值***
AXIS0_INT_CH	0
AXIS1_INT_CH	1
AXIS2_INT_CH	2
AXIS3_INT_CH	3
AXIS4_INT_CH	4
AXIS5_INT_CH	5

Value 為 16-bit 資料，其定義如下表 (1: enable; 0: disable)：

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Signal	**LTC_INT**		**INP_INT**	**-**	**SLD_INT**	**ORG_INT**	**MEL_INT**	**PEL_INT**
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Signal	**CMP_INT**	**-**	**-**	**-**	**-**	**-**	**EZ_INT**	**IDX_INT**

在撰寫程式時，可配合使用以下的常數定義：

Macro	***Value***	***Description***
B_PEL_INT	0x0001	指定正向極限 PEL 訊號
B_MEL_INT	0x0002	指定負向極限 MEL 訊號
B_ORG_INT	0x0004	指定 Original (Home) 訊號
B_SLD_INT	0x0008	指定 Slow Down 訊號
-	0x0010	-
B_INP_INT	0x0020	指定 In Position 訊號
-	0x0040	-
B_LTC_INT	0x0080	指定 Latch 訊號
B_IDX_INT	0x0100	指定 IDX 訊號-
B_EZ_INT	0x0200	指定 Encoder Z 相訊號
B_CMP_INT	0x8000	指定 Compare 訊號

*註：Index 訊號的定義，請參考 **Set_Index_Mode()**。

注意！若 GLB_INT_CH 中的 GLB_INT 或 MAIN_INT_CH 中的 AXx_H_INT / AXx_L_INT (x 為 0~5) 沒有被 enable，但該訊號的中斷因子有被 enable，則當該訊號有效的 change of state 事件發生時雖不會對 DSP 產生中斷，但該訊號的中斷旗標仍會被設為 1。此功能可提供 DSP 輪詢 (polling) 或定時檢查訊號是否有狀態改變發生，因而不會太頻繁地被中斷。
關於有效的 change of state 事件，請參考 **Set_Axis_IO_Pol(U8** *Axis*, **U16** *Value*)。

第二大類: *Ch* 為 250 (或 MISC_INT_CH) 時，為設定雜項中斷之用。

Value 為 16-bit 資料，其定義如下表 (1: enable; 0: disable)：

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Signal	**-**	**-**	**-**	**-**	**-**	**UART_RX_INT**	**-**	**-**
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Signal	**TIM_ERR_INT**	**-**	**-**	**-**	**-**	**-**	**-**	**-**

在撰寫程式時，可配合使用以下的常數定義：

Macro	***Value***	***Description***
B_UART_RX_INT	0x0004	指定 UART RX Ready 訊號
B_TIM_ERR_INT	0x8000	指定 Timing Error 訊號*

*註： Timing Error 是指當伺服週期中斷旗標 (MAIN_INT_CH 的 bit 14) 未被清除前，又再次發生伺服週期中斷。

注意！若 GLB_INT_CH 中的 GLB_INT 或 MAIN_INT_CH 中的 MISC_INT 或 MISC_INT_CH 中的 TIM_ERR_INT 沒有被 enable，Timing Error 發生時雖不會對 DSP 產生中斷，但 TIM_ERR_INT 的中斷旗標仍會被設為 1。

第三大類: *Ch* 為 253 (或 GINP_INT_CH) 時，為設定一般 DI 訊號(GDI) 產生中斷之用。

Value 為 16-bit 資料，其定義如下表 (1: enable; 0: disable)：

Bit	7	6	5	4	3	2	1	0
Signal	GDI7_INT	GDI6_INT	GDI5_INT	GDI4_INT	GDI3_INT	GDI2_INT	GDI1_INT	GDI0_INT
Bit	15	14	13	12	11	10	9	8
Signal	EMG_INT	MPG_EMG_INT	-	-	-	GDI10_INT	GDI9_INT	GDI8_INT

在撰寫程式時，可配合使用以下的常數定義：

Macro	*Value*	*Description*
B_GDI0_INT	0x0001	指定一般 DI channel 0 訊號
B_GDI1_INT	0x0002	指定一般 DI channel 1 訊號
B_GDI2_INT	0x0004	指定一般 DI channel 2 訊號
B_GDI3_INT	0x0008	指定一般 DI channel 3 訊號
B_GDI4_INT	0x0010	指定一般 DI channel 4 訊號
B_GDI5_INT	0x0020	指定一般 DI channel 5 訊號
B_GDI6_INT	0x0040	指定一般 DI channel 6 訊號
B_GDI7_INT	0x0080	指定一般 DI channel 7 訊號
B_GDI8_INT	0x0100	指定一般 DI channel 8 訊號
B_GDI9_INT	0x0200	指定一般 DI channel 9 訊號
B_GDI10_INT	0x0400	指定一般 DI channel 10 訊號
B_MPG_EMG_INT	0x4000	指定 MPG Emergency Stop 訊號
B_EMG_INT	0x8000	指定 Emergency Stop 訊號

注意！若 GLB_INT_CH 中的 GLB_INT 或 MAIN_INT_CH 中的 GDI_INT 沒有被 enable，但該訊號的中斷因子有被 enable，則當該訊號有效的狀態改變(change of state)事件發生時雖不會對 DSP 產生中斷，但該訊號的中斷旗標仍會被設為 1。此功能可提供 DSP 利用輪詢 (polling) 或定時檢查訊號是否有狀態改變發生，因而在不支援 GDI_INT 中斷下依然可以處理狀態改變事件。
關於有效的 change of state 事件，請參考 **Set_DI_Pol(U16 *Value*)**。

第四大類: *Ch* 為 254 (或 MAIN_INT_CH) 時，為控制各分類中斷之用。

Value 為 16-bit 資料，其定義如下表 (1: enable; 0: disable)：

Bit	7	6	5	4	3	2	1	0
Signal	DPL_INT	GINP_INT	AX5_L_INT	AX4_L_INT	AX3_L_INT	AX2_L_INT	AX1_L_INT	AX0_L_INT
Bit	15	14	13	12	11	10	9	8
Signal	MISC_INT	SERVO_INT	AX5_H_INT	AX4_H_INT	AX3_H_INT	AX2_H_INT	AX1_H_INT	AX0_H_INT

在撰寫程式時，可配合使用以下的常數定義：

Macro	Value	Description
B_AX0_L_INT	0x0001	指定軸 0 的低速訊號 (AXIS0_INT_CH) *
B_AX1_L_INT	0x0002	指定軸 1 的低速訊號 (AXIS1_INT_CH) *
B_AX2_L_INT	0x0004	指定軸 2 的低速訊號 (AXIS2_INT_CH) *
B_AX3_L_INT	0x0008	指定軸 3 的低速訊號 (AXIS3_INT_CH) *
B_AX4_L_INT	0x0010	指定軸 4 的低速訊號 (AXIS4_INT_CH) *
B_AX5_L_INT	0x0020	指定軸 5 的低速訊號 (AXIS5_INT_CH) *
B_GINP_INT	0x0040	指定一般 DI 訊號 (GINP_INT_CH)
B_DPL_INT	0x0080	指定 Local (DSP) 端 DPRAM 中斷訊號
B_AX0_H_INT	0x0100	指定軸 0 的高速訊號 (AXIS0_INT_CH) *
B_AX1_H_INT	0x0200	指定軸 1 的高速訊號 (AXIS1_INT_CH) *
B_AX2_H_INT	0x0400	指定軸 2 的高速訊號 (AXIS2_INT_CH) *
B_AX3_H_INT	0x0800	指定軸 3 的高速訊號 (AXIS3_INT_CH) *
B_AX4_H_INT	0x1000	指定軸 4 的高速訊號 (AXIS4_INT_CH) *
B_AX5_H_INT	0x2000	指定軸 5 的高速訊號 (AXIS5_INT_CH) *
B_SERVO_INT	0x4000	指定伺服週期中斷訊號
B_MISC_INT	0x8000	指定雜項中斷 (MISC_INT_CH)

*註： 各軸的低速訊號為：PEL, MEL, ORG, SLD, ALM, INP, RDY, IDX, EZ
各軸的高速訊號為：CMP, LTC

第五大類: *Ch* 為 255 (或 GLB_INT_CH) 時，控制整體中斷之用。

Value 為 16-bit 資料，只有 bit 0 有效 (1: enable; 0: disable)：

在撰寫程式時，可配合使用以下的常數定義：

Macro	Value	Description
B_GLB_INT	0x0001	指定整體中斷

Return Value

ERR_ADDR_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Set_FPGA_Int_Factor(MAIN_INT_CH,
  B_SERVO_INT);
// enable servo period interrupt
```

See Also

Get_FPGA_Int_Factor()，Get_FPGA_Int_Flag()，
Clear_FPGA_Int_Flag()，Get_Axis_IO_Pol()，Get_DI_Pol()

/***/

I16 Get_FPGA_Int_Factor (U8 *Ch*, U16 **Value*);

Description 讀取指定 channel 的中斷因子設定值 (enable/disable)。

Parameters *Ch* 指定的 channel，可以分成五大類，**Value* 依照不同 *Ch* 種類可以有不同定義。請參考 **Set_FPGA_Int_Factor()**。

Return Value
ERR_ADDR_OUT_RANGE
ERR_NO_ERROR

Example

```
U16 Value;
I16  error;
error = Get_FPGA_Int_Factor(AXIS0_INT_CH, &Value);
// get the interrupt factor setting of AXIS0
```

See Also

Set_FPGA_Int_Factor()

/***/

I16 Get_FPGA_Int_Flag (U8 *Ch*, U16 **Value*);

Description 讀取指定 channel 的中斷旗標 (是否有發生 interrupt)。

Parameters *Ch* 指定的 channel，可以分成五大類，**Value* 依照不同 *Ch* 種類可以有不同定義。請參考 **Set_FPGA_Int_Factor()**。請注意**第五大類 GLB_INT_CH 不提供這項讀取功能**。

Return Value
ERR_ADDR_OUT_RANGE
ERR_NO_ERROR

Example

```
U16 Value;
I16  error;
error = Get_FPGA_Int_Flag (AXIS0_INT_CH, &Value);
// get the interrupt flag of AXIS0
```

See Also

Clr_FPGA_Int_Flag()

/***/

I16 Clr_FPGA_Int_Flag (U8 *Ch*, U16 *Value*);

Description

清除指定 channel 的中斷旗標

Parameters

Ch 指定的 channel，可以分成五大類，*Value* 依照不同 *Ch* 種類可以有不同定義，清除的動作為依照對應表將要被清除的 bit 寫入 1。請參考 **Set_FPGA_Int_Factor()**。請注意第五大類 GLB_INT_CH 不提供這項清除功能，而對第二大類 MISC_INT_CH 及第四大類 MAIN_INT_CH 執行此清除功能時，則有如下的規定：

當 *Ch* 為 MISC_INT_CH 時：

TIM_ERR_INT 的中斷旗標無法被清除，只有重置 FPGA 後才能清除此中斷旗標。

當 *Ch* 為 MAIN_INT_CH 時：

若欲清除 AXx_L_INT / GINP_INT / AXx_H_INT / MISC_INT (x 為 0~5) 的中斷旗標，其方法是清除各分類 (AXISx_INT_CH / GINP_INT_CH / MISC_INT_CH) 中所有信號的中斷旗標。

若欲清除 SERVO_INT 的中斷旗標，其方法是將 *Value* 設為 0x4000 (B_SERVO_INT)。

若欲清除 DPL_INT 的中斷旗標，其方法是呼叫 **Get_DPRAM_Int_Code(U16 **Value*)**，或是讀取 DPRAM (DPRAM_ADDR ， 0x90001800) offset 0x07FE (DPR_H2L_INT_OFST) 的位址 (0x90001FFE)。

Return Value

ERR_ADDR_OUT_RANGE
ERR_NO_ERROR

Example

```
I16  error;
error = Clr_FPGA_Int_Flag (MAIN_INT_CH, B_SERVO_INT);
// clear the interrupt flag of servo period interrupt
```

See Also

Get_FPGA_Int_Flag()，Get_DPRAM_Int_Code()

2.18 DPRAM中斷控制函式

/***/
Void Set_DPRAM_Int_Code (U16 *Value*);

Description 將一筆資料寫到 DPRAM (DPRAM_ADDR，0x90001800) offset 0x07FC (DPR_L2H_INT_OFST) 的位址，此動作也會同時對 Host 發出一個中斷訊號 (經由 PCI bus)。

Parameters *Value* 為欲寫入到 DPRAM offset 0x07FC 的資料。

Return Value 無

Example

```
Set_DPRAM_Int_Code (0x01);
// generate an interrupt to host
//  所帶的參數可以由用戶自行定義其意義
// PC 可以讀此參數值來判斷 DSP 的意圖
```

See Also

Get_DPRAM_Int_Code()

/***/
Void Get_DPRAM_Int_Code (U16 **Value*);

Description 讀取 DPRAM 中，Host 對 DSP 產生中斷時寫入的資料。當 Host 經由 PCI bus 將一筆資料寫到 DPRAM (DPRAM_ADDR，0x90001800) offset 0x07FE (DPR_H2L_INT_OFST) 的位址時，DPRAM 會對 DSP 發出一個中斷訊號 (當相關的中斷因子都有被設為 enable 時)。此讀取資料的動作也會同時清除 MAIN_INT_CH 中 DPL_INT 的中斷旗標，不須要再做清除旗標動作。

Parameters *Value* 為 DSP 讀取 DPRAM offset 0x07FE 的資料。

Return Value 無

Example

```
U16 Value;
Get_DPRAM_Int_Code(&Value);
// clear the interrupt flag of DPL_INT
```

See Also

Set_FPGA_Int_Factor()，Clr_FPGA_Int_Flag()

2.19 其他函式

/***/
void Sleep_us (U32 *time_us*);

Description 延遲指定時間(以微秒 us 為單位)再繼續執行。

Parameters *time_us* 為延遲的時間。**注意！**此延遲的時間只是一個大略的時間，且會受到中斷的影響。

Return Value 無

Example

```
Sleep_us(500);
// 延遲大約 500us 的時間
```

See Also

/***/
void Set_LED (U8 *On_nOff*);

Description 開關板子上的 LED 燈。

Parameters *On_nOff* 為 1 則開，為 0 則關。

Return Value 無

Example

```
Set_LED (1);
// 點亮 LED
```

See Also