PMBus Slave Framework Module

User Guide

作者: Alfred

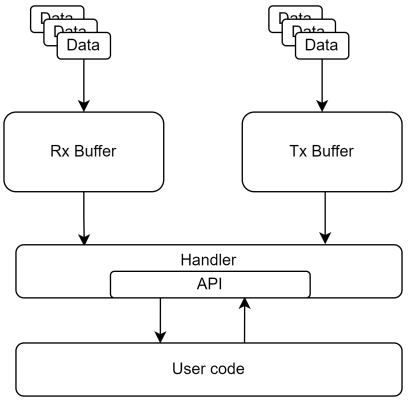
最後更新: 2023年4月10日

Contents

1. INTRODUCTION

PMBSF（PMBus Slave Framework或PMBus Software Framework）是用於PMBus Slave的模組，旨在讓使用者專注於操作行為，而無需關心狀態處理。所有相關函數均以PMBSF_作為前綴名稱。使用者可以自行編寫命令執行、傳輸、接收和PEC確認等行為。

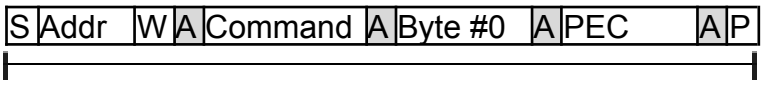
模組主要執行者是名為Handler的object，使用者透過Handler API與Handler進行交互。



1.1. Message Types

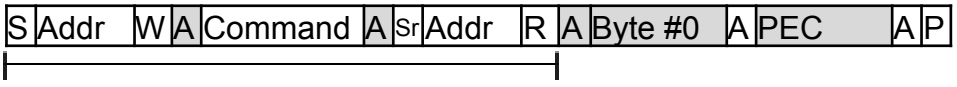
依照I2C PMBus的通訊格式為基礎，在模組中稱master傳送的通訊封包為frame。Handler中的frame check method中的frame，指的就是此章節內的封包。

PMBus WRITE BYTE:



Frame

PMBus READ BYTE:



Frame

1.2. 特殊命令

在模組中，所謂的特殊命令，是指一些需要長時間等待後再回傳的狀況，等待的時間不能比time out時間還要久。在解碼時，不能馬上放進trainsmit buffer的情況，關於流程會於後面章節解釋。

可套用情境舉例如下：

- 在收到某一命令後，跟外部其它晶片透過通訊取得資料後再傳回。
- 在收到某一命令後，要取得EEPROM的資料後傳回。

此時會經過比較長的時間，可通知handler收到特殊命令，handler會等待到使用者通知特殊命令執行結束，或是time out才會進行下一步動作。

1.3. Abbreviation

PMBus	Power Management Bus
I2C	Inter-Integrated Circuit
MCU	Microcontroller Unit
EEPROM	Electrically-Erasable Programmable Read-Only Memory

1.4. 範例注意事項

在範例中，皆使用TI F280049C為示範晶片。

2. API OVERVIEW

2.1. PMBsf Defined Parameter

2.1.1. PMBSF_STATE_MACHINE_t

handler的狀態列舉。

Definition:

```
typedef enum {
    PMBSF_STATE_IDLE,
    PMBSF_STATE_RECEIVE,
    PMBSF_STATE_FRAME_CHECK,
    PMBSF_STATE_FRAME_CHECK_OK,
    PMBSF_STATE_SPECIAL_CMD_IN_WATING,
    PMBSF_STATE_WAIT_TRANS,
    PMBSF_STATE_EXEC_CMD,
    PMBSF_STATE_ERROR_CHECK,
    PMBSF_STATE_HALT
} PMBSF_STATE_MACHINE_t;
```

2.1.2. Exception Code

handler的exception code，type為unsiged int16，說明如下：

15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	PEC error	buffer full	handler halt	Reserved	send not compelet	frame check fail	time out

2.1.3. PMBSF_DATA_t

PMBus slave framework的data，type為unsiged char。

2.1.4. PMBSF_DATA_BUFFER_t

PMBus slave framework的data buffer。

Definition:

```
typedef struct {
    int             capacity;
    int             len;
    PMBSF_DATA_t    *data_ptr;
    PMBSF_DATA_t    *arr;
} PMBSF_DATA_BUFFER_t;
```

Members:

`capacity`	buffer的容量大小
`len`	現在buffer的長度
`*data_ptr`	現在buffer data的位址
`*arr`	buffer data陣列

2.1.5. PMBSF_HANDLER_t

PMBus slave framework的handler。

Definition:

```
struct PMBSF_HANDLER_t {
    PMBSF_STATE_MACHINE_t              old_state;
    PMBSF_STATE_MACHINE_t              now_state;
    PMBSF_HADLER_EXCEPTION_t           exception;
    uint16_t                           event;
    PMBSF_DATA_BUFFER_t                *rx_buffer;
    PMBSF_DATA_BUFFER_t                *tx_buffer;
    bool                               PEC_used;

    void                               (*frame_start)(PMBSF_HANDLER_t*);
    void                               (*is_special_cmd)(PMBSF_HANDLER_t*);
    void                               (*special_cmd_end)(PMBSF_HANDLER_t*);
    void                               (*get_query_sign)(PMBSF_HANDLER_t*);
    void                               (*get_stop_sign)(PMBSF_HANDLER_t*);
    void                               (*reset)(PMBSF_HANDLER_t*);
    void                               (*time_out)(PMBSF_HANDLER_t*, bool);
    void                               (*set_use_pec)(PMBSF_HANDLER_t*, bool);
    bool                               (*put_transmit_buffer)(PMBSF_HANDLER_t*, const PMBSF_DATA_t);
    PMBsf_I2C_get_SDA_data_fp          get_SDA_data;
    PMBsf_frame_chek_fp                frame_check;
    PMBsf_I2C_PEC_check_fp             check_pec;
    PMBsf_special_cmd_exec_fp          special_cmd_exec;
    PMBsf_I2C_put_data_to_SDA_fp       put_data_to_SDA;
    PMBsf_frame_cmd_exec_fp            cmd_exec;
    PMBsf_error_check_fp               error_check;
};
```

Members:

old_state	handler前一個狀態
now_state	handler現在的狀態
exception	handler exception code
event	handler 事件狀態
*rx_buffer	handler received buffer
*tx_buffer	handler transmit buffer
PEC_used	handler通訊是否使用PEC
frame_start	通知handler frame收到I2C [S]，開始收資料
is_special_cmd	在解碼的時候，通知handler收到特殊命令
get_query_sign	通知handler收到I2C [Rd]訊號
get_stop_sign	通知handler收到I2C [P]訊號
reset	保留中
time_out	通知handler time out
set_use_pec	改變handler是否使用PEC
put_transmit_buffer	放資料進handler transmit buffer
get_SDA_data	handler取得I2C Bus資料函數
frame_check	handler frame檢查函數
check_pec	handler PEC檢查函數
special_cmd_exec	handler 執行特殊命令函數
put_data_to_SDA	handler 將traismit buffer放到I2C Bus函數
cmd_exec	handler 執行命令函數

`error_check` handler 檢查錯誤函數

2.2. PMBSF module function

PMBSF module的核心函數，不用更改程式碼就可以使用。也建議不要更改讓模組運行順暢。

API Function	Description
PMBSF_DATA_BUFFER_init	PMBus slave framework data buffer initialize
PMBSF_handler_init	PMBus slave framework handler initialize
PMBSF_handler_run	PMBus slave framework handler run

2.3. PMBSF defined function pointer

PMBSF 定義的函數接口，由PMBSF handler 所調用。

Function prototype	Description
PMBsf_I2C_get_SDA_data_fp	handler get data function
PMBsf_frame_chek_fp	handler frame check function
PMBsf_I2C_PEC_check_fp	handler PEC check function
PMBsf_special_cmd_exec_fp	handler special command execution function
PMBsf_I2C_put_data_to_SDA_fp	handler transmit data function
PMBsf_frame_cmd_exec_fp	handler execution function after frame end
PMBsf_error_check_fp	handler error check function

以下介紹函數指標相關細節

2.3.1. PMBsf_I2C_get_SDA_data_fp

Returned value	Prototype Name	Arguments
bool	PMBsf_I2C_get_SDA_data_fp	PMBSF_DATA_t* data

讓handler取得MCU I2C bus上data的函數指標。

Arguments:

*PMBSF_DATA_t** handler中rx data的位址，於函數中放置取得的byte資料。

Returned value:

1 表示成功

0 表示失敗

Example:

```
bool
```

```
PMBus_get_SDA_data(PMBSF_DATA_t* data) {
    if (I2C_getRxFIFOStatus(EXTERN_I2C) > I2C_FIFO_RXEMPTY) {
        *data = I2C_getData(EXTERN_I2C);
        return 1;
    }
    return 0;
}
```

2.3.2. PMBsf_frame_check_fp

Returned value	Prototype Name	Arguments
bool	PMBsf_frame_check_fp	PMBSF_DATA_t* arr
		int len

handler用來確認整條PMBus frame是否正確或支援(在確認PEC之後)，在這時候要對命令解碼，放置要回覆的資料(使用handler.put_transmit_buffer)，或是告訴handler是否為特殊命令，hanlder會在之後執行特殊命令的行為。

Arguments:

*PMBSF_DATA_t**	handler接收的byte array，byte 1為命令byte，其餘為data bytes。強制取得超過len長度的資料，不保證其資料正確性。
int	handler接收到的byte array長度

Returned value:

1	表示frame確認成功
0	表示frame確認失敗

Example:

```
bool
PMBus_frame_check(const PMBSF_DATA_t* data, int len) {
    PMBus_info.PMBus_command = data[0];
    command_decode();

   PMBusHandler.put_transmit_buffer((PMBSF_HANDLER_t*)&PMBusHandler,
PMBus_info.query_type_data_len);
   return 1;
}
```

2.3.3. PMBsf_I2C_PEC_check_fp

Returned value	Prototype Name	Arguments
bool	PMBsf_I2C_PEC_check_fp	PMBSF_DATA_t* arr
		int len

Handler 檢查PEC函數。

Arguments:

*PMBSF_DATA_t** handler接收的byte array，byte 1為命令byte，其餘為data bytes。強制取得超過len長度的資料，不保證其資料正確性。

int handler接收到的byte array長度

Returned value:

1 表示frame確認成功

0 表示frame確認失敗

2.3.4. PMBsf_special_cmd_exec_fp

Returned value	Prototype Name	Arguments
void	PMBsf_special_cmd_exec_fp	PMBSF_DATA_t* arr
		int len

handler收到特別命令的行為。如特殊命令中，需要用到資料，可從arr中取得。

Arguments:

*PMBSF_DATA_t** handler接收的byte array，byte 1為命令byte，其餘為data bytes。強制取得超過len長度的資料，不保證其資料正確性。

int handler接收到的byte array長度

Returned value:

None

2.3.5. PMBsf_I2C_put_data_to_SDA_fp

Returned value	Prototype Name	Arguments
bool	PMBsf_I2C_put_data_to_SDA_fp	PMBSF_DATA_t data

handler將要傳送的資料放到SDA的函數。

Arguments:

PMBSF_DATA_t 要傳送的資料(byte)

Returned value:

1 表示frame確認成功

0 表示frame確認失敗

Example:

```
bool
PMBus_put_data_to_SDA(const PMBSF_DATA_t data) {
    I2C_enableFIFO(EXTERN_I2C);
    if (I2C_getTxFIFOStatus(EXTERN_I2C) < I2C_FIFO_TX16)
    {
        I2C_putData(EXTERN_I2C, data);
        return 1;
    }
    return 0;
```

}

2.3.6. PMBsf_frame_cmd_exec_fp

Returned value	Prototype Name	Arguments
bool	PMBsf_frame_cmd_exec_fp	PMBSF_DATA_t* arr
		int len

handler收到frame後，要執行動作的函數。

Arguments:

*PMBSF_DATA_t** handler接收的byte array，byte 1為命令byte，其餘為data bytes。強制取得超過len長度的資料，不保證其資料正確性。

int handler接收到的byte array長度

Returned value:

1 表示frame確認成功

0 表示frame確認失敗

2.3.7. PMBsf_error_check_fp

Returned value	Prototype Name	Arguments
bool	PMBsf_error_check_fp	PMBSF_HADLER_EXCEPTION_t excep

使用者確認handler的error函數，確認成功會讓hanlder繼續下一次的通訊，若確認失敗handler會一直處於error check state中。

Arguments:

PMBSF_HADLER_EXCEPTION_t handler的exception code

Returned value:

1 表示frame確認成功

0 表示frame確認失敗

3. ERROR CONTROL

指標函數 `PMBsf_error_check_fp` 提供使用者端確認Handler中發生的錯誤。在確認之後，此指標函數會回傳bool來讓Handler回復至IDLE狀態。

Error 如下表：

Name	Description
PEC error	Handler開啟PEC，確認錯誤
buffer full	Handler rx或tx buffer 已滿
handler halt	Handler 不明原因停止
send not compelet	Handler tx buffer傳送未完全
frame check fail	Handler 確認frame錯誤

time out	Handler 被告知time out

4. MODULE FLOW

本章說明Handler 在PMBus通訊時的控制及流程。

4.1. Handler 初始化

從2.2 章中，其3個模組函數，對應著使用Handler的初始化及使用。

第一步為初始化data buffer，名稱為`PMBSF_DATA_BUFFER_init`，設定Handler 2個buffer(Rx, Tx)的對應陣列空間及容量。將初始化完的陣列交由下個步驟傳給Handler。

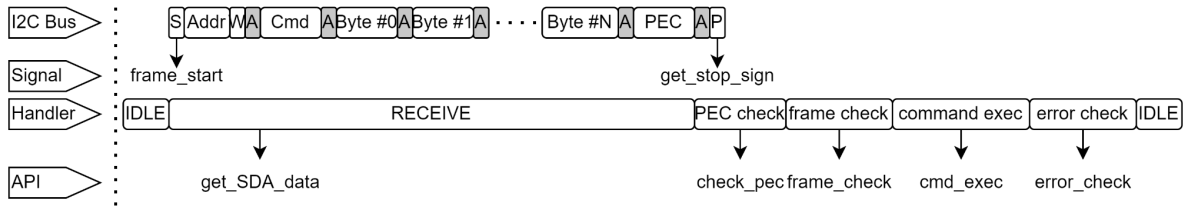
第二步初始化Handler，名稱為`PMBSF_handler_init`，函數中要給與多個指標函數，這些函數被指派為Handler的事件處理函數，也是使用者可以自行編寫程式碼執行的空間。

第三步使用Handler，在main或持續執行函數，放入Handler的執行函數，其名稱為`PMBSF_handler_run`。使用者在發生事件時，用Handler的signal函數通知Handler，Handler會執行該事件的處理函數。詳細過程在接下來的章節帶來更進一步的說明。

4.2. Handler 的執行流程

4.2.1. WRITE

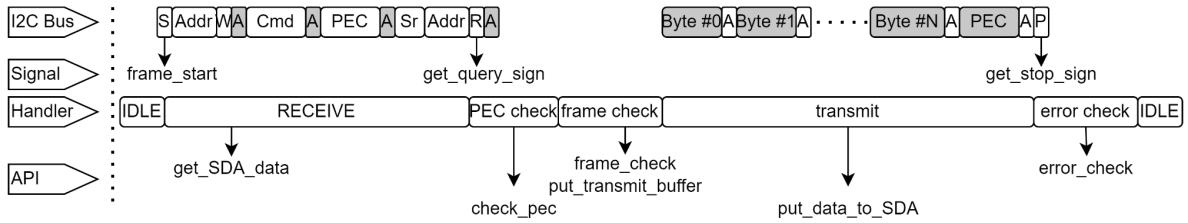
WRITE正常執行如下:



PMBus中有一形式為SEND BYTE，為WRITE的一種變形，沒有Command，直接送Data byte。此時Handler會RECEIVE狀態直到收到stop信號後執行後續函數，Handler一樣會收到Addr後續的byte，可以於command exec的函數指標中獲取其byte資料並處理。

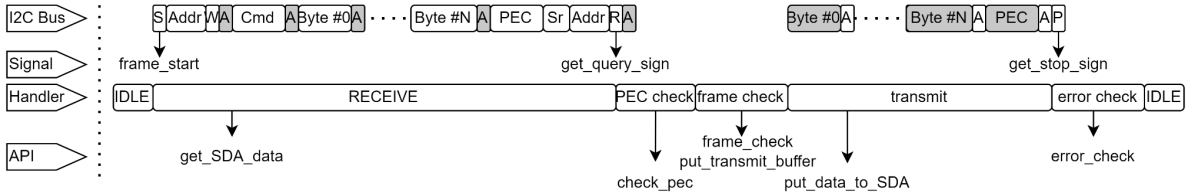
4.2.2. READ

READ正常執行如下:



4.2.3. PROCESS CALL

PROCESS CALL正常執行如下:



PROCESS CALL為WRITE及READ的合體，其主要依靠 Rd 的SIGNAL決定執行frame check的時機，再進行後續動作。

4.3. Handler 訊號

由上一章節可知，Handler的執行，是依靠著訊號(Signal)來驅動的，在之前的章節中，簡述了Handler傳遞訊號的成員函數(method)，以下詳細介紹各訊號。

- `frame_start`
  收到PMBus start 訊號，Handler收到訊號後，會從IDLE的狀態轉為其它狀態。
- `is_special_cmd`
  在解析命令時，如發現是sepcial command，要即時通知handler，Handler會做相對應的轉態。
- `special_cmd_end`
  Handler在收到是sepcial command後，會等待到time out，或是收到此訊號，才會進行下一步的動作。因此在做完sepcial command執行後，一定要搭配此訊號告知Handler，如果有要回傳資料，也要把資料交給Handler(使用`put_transmit_buffer`函數)。
- `get_query_sign`
  收到PMBus Read 訊號，通知Handler，Handler會準備傳送資料。
- `get_stop_sign`
  收到PMBus stop 訊號，如在傳送到一半收到此訊號，會引發傳送未完成錯誤。如在write下，handler會在稍後執行command execution函式。
- `reset`
  目前保留中
- `time_out`
  告知Handler time out。Handler會從當前狀態改為time out 狀態。

4.4. Handler frame check

在收到資料後，收到I2C S或是Rd訊號，Handler會接著進行frame確認，如果有開啟PEC確認，執行PEC check函數，確認成功後，執行frame check函數。本章節則談frame check的細節。

Example:

```
bool
PMBus_frame_check(const PMBSF_DATA_t* data, const int len) {
    PMBus_info.PMBus_command = data[0];
    command_decode();

   PMBusHandler.put_transmit_buffer((PMBSF_HANDLER_t*)&PMBusHandler,
PMBus_info.query_type_data_len);
   return 1;
}
```

由函數原型中的參數，可得到Handler的rx data buffer，並已得知長度。data buffer內容第一個資料一定是command(根據PMBus規定)，後面才是data。

範例中的command_decode函數節錄如下：

```
switch (PMBus_info.PMBus_command) {
        case PMBUS_CMD_ON_OFF_CONFIG:
            PMBus_info.PMBus_command_type = PMBUS_READ_BYTE_MANUFACTURER_FORMATE;
            PMBus_info.need_transmit_len = 1;
            PMBus_info.temp_txbuffer[0] = psu_ram_settings.on_off_config.byte & 0x1F;
            break;
        case PMBUS_CMD_CLEAR_FAULTS:
            PMBus_info.PMBus_command_type = PMBUS_SENDBYTE;
            break;
```

解析完command後，如果該命令有支援回傳，要把回傳的資料在此時傳給Handler，Handler收到PMBus Rd訊號後，有回傳的資料才會進行回傳的動作。資料傳給Handler用的函數為

```
(*put_transmit_buffer)(PMBSF_HANDLER_t*, const PMBSF_DATA_t);
```

範例中可見其用法。

此函數傳回FALSE，則Handler會認為解析失敗而直接進入Error check狀態。