电子鼓模块

BM72D5221-1

Arduino Library V1.0.1 說明

版本: V0.00 日期: 2024-5-17

www.bestmodulescorp.com

目錄

簡介

BM72D5221-1 是倍創推出的电子鼓模块。使用 UART 通訊方式。本文檔對 BM72D5221-1 的 Arduino Lib 函式、Arduino Lib 安裝方式進行說明；範例使用 BMV51M521 模組，演示了簡易電子鼓的功能。

適用型號：

型號	說明
BM72D5221-1	电子鼓模块
BMV51M521	板載BM72D5221-1

Arduino Lib 函式

Arduino Lib名稱：BM72D5221-1		Lib版本：V1.0.1
構造函式&初始化		
1	BM72D5221_1 (uint16_t rxPin, uint16_t txPin)	
	描述	構造函式，選擇軟件UART通訊
	參數	rxPin：RX引脚，连接BM72D5221-1或BMV51M521的TX引脚 txPin：TX引脚，连接BM72D5221-1或BMV51M521的RX引脚
	返回值	—
	備註	—
2	BM72D5221_1 (HardwareSerial *theSerial = &Serial)	
	描述	構造函式，選擇硬體UART通訊
	參數	*theSerial：选择硬件 UART 接口（默认为Serial接口）
	返回值	—
	備註	—
3	bool begin(void)	
	描述	模塊初始化
	參數	void
	返回值	执行情况 true：成功 false：失敗
	備註	—
功能函式		
4	bool scanSlaveUpdateData(void)	
	描述	循環掃描獲取模組更新的數據

	參數	void
	返回值	执行情况 true：有新的數據更新 false：無數據更新
	備註	必須在loop裡面執行
5	bool pageSwitch(uint8_t pageType =PAGE_KIT)	
	描述	切換設置界面
	參數	pageType：界面选择 0x00(PAGE_KIT)：鼓組設置 0x01(PAGE_METRO)：節拍器設置 0x02(PAGE_SYS)：系統設置 0x03(PAGE_SONG)：歌曲設置 0x04(PAGE_RECORD)：錄音設置
	返回值	执行情况 true：成功 false：失敗
	備註	上电默认为状态为，PAGE_KIT：鼓組設置 在發送對應功能函數之前，需先切換界面才有效 PAGE_KIT：本表格第6~14序号函数功能使能 PAGE_METRO：本表格第15~20序号函数功能使能 PAGE_SYS：本表格第21~23序号函数功能使能 PAGE_SONG：本表格第24~27序号函数功能使能 PAGE_RECORD：本表格第28~32序号函数功能使能
6	bool setDrumKitGroup(uint8_t group)	
	描述	選擇鼓組
	參數	group：鼓組序號，範圍1~6
	返回值	执行情况 true：成功 false：失敗
	備註	當前參數值可從bool getSaveValue(void)函式獲取
7	bool setDrumKitTimbre(uint8_t timbre)	
	描述	設置鼓通道音色
	參數	timbre：音色序號，範圍1~32
	返回值	执行情况 true：成功

		false：失敗
	備註	當前參數值可從bool getSaveValue(void)函式獲取
8	bool setDrumKitVolume(uint8_t volume)	
	描述	設置鼓通道音量
	參數	volume：音量值，範圍0~35
	返回值	執行情況 true：成功 false：失敗
	備註	當前參數值可從bool getSaveValue(void)函式獲取
9	bool setDrumKitPhase(uint8_t phase)	
	描述	設置鼓通道平衡
	參數	phase：平衡值，範圍0~14，7左右平衡；小於7聲場偏左；大於7聲場偏右
	返回值	執行情況 true：成功 false：失敗
	備註	1、當前參數值可從bool getSaveValue(void)函式獲取 2、平衡：指的是左右聲場變化， 7（左右平衡），小於7聲場偏左，否則偏右 3、BMV51M521左右聲道合併不支援這個效果，只適用於左右聲道分開的硬體
10	bool setDrumKitForce(uint8_t force)	
	描述	設置鼓通道力度曲線
	參數	force：力度曲線序號，範圍0~2
	返回值	執行情況 true：成功 false：失敗
	備註	1、當前參數值可從bool getSaveValue(void)函式獲取 2、力度曲線0：音量變化可最自然響應敲擊力度 力度曲線1：小力度敲擊音量起點較大 力度曲線2：小力度敲擊產生更大的音量變化
11	bool setDrumKitParamSaveUserGroup(uint8_t saveUserGroup=4)	
	描述	設置保存鼓組參數的序號
	參數	saveUserGroup：保存的鼓組序號，範圍4~6，默认4
	返回值	執行情況 true：成功 false：失敗

	備註	
12	bool setDrumKitParamSave(void)	
	描述	保存功能函式序號7~10設置的參數
	參數	void
	返回值	执行情况 true：成功 false：失敗
	備註	—
13	bool setDrumKitTrim(uint8_t channel=0)	
	描述	選擇鼓通道進行校準
	參數	channel：通道序號，範圍0~13，默認0
	返回值	执行情况 true：成功 false：失敗
	備註	1.BMV51M521只有9路输入，僅支援通道0~8的校準
14	bool setDrumKitTrimSave(void)	
	描述	保存通道校準結果
	參數	void
	返回值	执行情况 true：成功 false：失敗
	備註	—
15	bool setMetroBPM(uint16_t bpm)	
	描述	設置節拍器BPM
	參數	bpm：每分鐘節拍器數值，範圍30~280，上电默認值120
	返回值	执行情况 true：成功 false：失敗
	備註	
16	bool setMetroBeat(uint8_t beat)	
	描述	設置節拍器節拍
	參數	beat：節拍，範圍1~9
	返回值	执行情况

		true：成功 false：失敗
	備註	當前參數值可從bool getSaveValue(void)函式獲取
17	bool setMetroRhythm(uint8_t rhythm)	
	描述	設置節拍器節拍時值
	參數	rhythm：節拍器時值 1：1/4 2：1/8 3：3/8 4：1/16
	返回值	执行情況 true：成功 false：失敗
	備註	當前參數值可從bool getSaveValue(void)函式獲取
18	bool setMetroVolume(uint8_t metroVolume)	
	描述	設置節拍器音量
	參數	metroVolume：音量值，範圍0~35
	返回值	执行情況 true：成功 false：失敗
	備註	當前參數值可從bool getSaveValue(void)函式獲取
19	bool setMetroSound(uint8_t sound)	
	描述	設置節拍器音色
	參數	sound：音色序號，範圍0~2
	返回值	执行情況 true：成功 false：失敗
	備註	當前參數值可從bool getSaveValue(void)函式獲取
20	bool setMetroParamSave(void)	
	描述	保存本表函数序號16~19設置的參數
	參數	void
	返回值	执行情況 true：成功 false：失敗

	備註	保存后掉电不丢失
21	bool setLocalAudioSW(uint8_t state)	
	描述	打開/關閉本地系統音源
	參數	state：開關狀態 0：關閉 1：打開
	返回值	执行情况 true：成功 false：失敗
	備註	當前參數值可從bool getSaveValue(void)函式獲取
22	bool restoreDefault (void)	
	描述	將模組恢復出廠設置
	參數	void
	返回值	执行情况 true：成功 false：失敗

	備註	因發送設置命令成功之後執行恢復出廠。設置過程需要時間，可循环執行資訊獲取判定是否恢复完成
		程式範例：
		while (MIDI.getSysResetFlag())
		{
		MIDI.scanSlaveUpdateData();
		}

23	bool setSysParamSave(void)	
	描述	保存本表函数序號21設置的參數
	參數	void
	返回值	执行情况 true：成功 false：失敗
	備註	保存后掉电不丢失
24	bool setSongNum(uint8_t songNum=1)	
	描述	選擇歌曲序號
	參數	songNum：歌曲序號，範圍1~34
	返回值	执行情况 true：成功

		false：失敗
	備註	最後3首歌曲是打击录制保存到板上的打击乐
25	bool setSongPlayMode(uint8_t playMode)	
	描述	設置歌曲播放模式
	參數	playMode：播放模式 0：單曲循環 1：列表播放 2：單次播放
	返回值	执行情况 true：成功 false：失敗
	備註	當前參數值可從bool getSaveValue(void)函式獲取
26	bool setSongVolume(uint8_t songVolume)	
	描述	設置歌曲播放音量
	參數	songVolume：音量值，範圍0~15
	返回值	执行情况 true：成功 false：失敗
	備註	當前參數值可從bool getSaveValue(void)函式獲取
27	bool setSongParamSave(void)	
	描述	保存本表函数序號25~26設置的參數
	參數	void
	返回值	执行情况 true：成功 false：失敗
	備註	保存后掉电不丢失
28	bool setRecordStart(void)	
	描述	開始錄製打击乐
	參數	void
	返回值	执行情况 true：成功 false：失敗
	備註	—
29	bool setRecordEnd(void)	

	描述	結束錄製打击乐，同時保存錄製資料
	參數	void
	返回值	执行情况 true：成功 false：失敗
	備註	—
30	bool setRecordPlayStop(void)	
	描述	播放或停止播放錄製資料
	參數	void
	返回值	执行情况 true：成功 false：失敗
	備註	录音结束前要设置好保存位置
31	bool setRecordSaveNum(uint8_t saveNum=1)	
	描述	選擇保存錄製資料的位置序號
	參數	saveNum：位置序號，範圍1~3（支持3首录音的打击乐）
	返回值	执行情况 true：成功 false：失敗
	備註	录音结束前要设置好保存位置
32	bool setRecordNosave(void)	
	描述	不保存錄製資料並停止錄製
	參數	void
	返回值	执行情况 true：成功 false：失敗
	備註	—
33	bool setMetroOpenClose(void)	
	描述	打開或關閉節拍器
	參數	void
	返回值	执行情况 true：成功 false：失敗

	備註	任何操作状态下都有效
34	bool setSongPlayStop(void)	
	描述	播放或停止播放歌曲
	參數	void
	返回值	执行情况 true：成功 false：失敗
	備註	PAGE_RECORD录音操作状态下無效
35	bool setTrackSW(void)	
	描述	打開或關閉背景軌道音樂
	參數	void
	返回值	执行情况 true：成功 false：失敗
	備註	1、任何操作状态下都有效 2、只有在歌曲播放时效果才顯現出來
參數讀取		
36	uint8_t getDrumKitGroup(void)	
	描述	獲取鼓組序號
	參數	void
	返回值	鼓組序號：1~6
	備註	—
37	uint8_t getDrumKitTimbre(void)	
	描述	獲取當前鼓組下觸發的鼓通道音色
	參數	void
	返回值	鼓通道音色：1~32
	備註	—
38	uint8_t getDrumKitVolume(void)	
	描述	獲取當前鼓組下觸發的鼓通道音量
	參數	void
	返回值	鼓通道音量：0~35
	備註	—

39	uint8_t getDrumKitPhase(void)	
	描述	獲取當前鼓組下觸發的鼓通道聲場相位
	參數	void
	返回值	鼓通道聲場相位：0~14
	備註	—
40	uint8_t getDrumKitForce(void)	
	描述	獲取當前鼓組下觸發的鼓通道力度曲線
	參數	void
	返回值	鼓通道力度曲線：0~2
	備註	—
41	uint8_t getDrumKitTrimChannel(void)	
	描述	獲取校準鼓通道
	參數	void
	返回值	校準鼓通道：0~13
	備註	—
42	uint8_t getDrumKitParamSaveUserGroup(void)	
	描述	獲取當前自定義鼓組保存所在的鼓組序號
	參數	void
	返回值	當前自定義鼓組保存所在的鼓組序號：4~6
	備註	需要循环執行扫描状态函数bool scanSlaveUpdateData(void)后，才能执行本函数获取到结果
43	uint8_t getTrigChannel(void)	
	描述	獲取觸發鼓通道
	參數	void
	返回值	觸發鼓通道：0~13
	備註	需要循环執行扫描状态函数bool scanSlaveUpdateData(void)后，才能执行本函数获取到结果
44	uint16_t getMetroBPM(void)	
	描述	獲取節拍器BPM
	參數	void
	返回值	bpm：30~280
	備註	需要循环執行扫描状态函数bool scanSlaveUpdateData(void)后，才能执行本函数

		获取到结果
45	uint8_t getMetroBeat(void)	
	描述	獲取節拍器節拍
	參數	void
	返回值	節拍：1~9
	備註	需要循环執行扫描状态函数bool scanSlaveUpdateData(void)后，才能执行本函数获取到结果
46	uint8_t getMetroRhythm(void)	
	描述	獲取節拍器節拍器時值
	參數	void
	返回值	節拍時值：1~4
	備註	需要循环執行扫描状态函数bool scanSlaveUpdateData(void)后，才能执行本函数获取到结果
47	uint8_t getMetroVolume(void)	
	描述	獲取節拍器音量
	參數	void
	返回值	節拍器音量：0~35
	備註	—
48	uint8_t getMetroSound(void)	
	描述	獲取節拍器音色
	參數	void
	返回值	音色：0~2
	備註	—
49	uint8_t getLocalAudioSW(void)	
	描述	獲取系統本地音源開發狀態
	參數	void
	返回值	開關狀態 0：關閉 1：打開
	備註	—
50	uint8_t getSysResetFlag(void)	
	描述	獲取BMV51M521模組的恢復出廠設置狀態

	參數	void
	返回值	恢復出廠設置狀態 0：完成 1：執行中
	備註	需要循环執行扫描状态函数bool scanSlaveUpdateData(void)后，才能执行本函数获取到结果
51	bool getSoftwareVer(uint8_t versionBuf[])	
	描述	獲取BMV51M521模組的軟體版本
	參數	versionBuf[]：存放版本號的數組
	返回值	执行情况 true：成功 false：失敗
	備註	數組長度為4
52	uint8_t getSongNum(void)	
	描述	獲取歌曲序號
	參數	void
	返回值	歌曲序號：1~34
	備註	需要循环執行扫描状态函数bool scanSlaveUpdateData(void)后，才能执行本函数获取到结果
53	uint8_t getSongPlayMode(void)	
	描述	獲取歌曲播放模式
	參數	void
	返回值	播放模式 0：單曲循環 1：列表播放 2：單次播放
	備註	—
54	uint8_t getSongVolume(void)	
	描述	獲取歌曲音量
	參數	void
	返回值	音量：0~15
	備註	—
55	uint8_t getRecState(void)	

	描述	獲取錄製狀態
	參數	void
	返回值	錄製狀態 0：錄製結束 1：錄製開始 2：錄製保存或不保存執行中
	備註	需要循环執行扫描状态函数bool scanSlaveUpdateData(void)后，才能执行本函数获取到结果
56	uint8_t getRecordSaveNum(void)	
	描述	獲取錄製資料保存的位置序號
	參數	void
	返回值	位置序號：1~3
	備註	需要循环執行扫描状态函数bool scanSlaveUpdateData(void)后，才能执行本函数获取到结果
57	uint8_t getPlayState(void)	
	描述	獲取歌曲播放狀態
	參數	void
	返回值	播放狀態 0：播放停止 1：播放中 0xff：獲取失敗
	備註	—
58	bool getSaveValue(void)	
	描述	獲取保存的參數
	參數	void
	返回值	讀取結果 true：讀取成功 false：讀取失敗
	備註	這些獲取的保存參數可通過其他獲取函式獲得具體數據
59	uint8_t getSPIFlashEraseState(void)	
	描述	獲取SPI Flash擦除狀態
	參數	void
	返回值	擦除狀態 0：擦除結束

		1：擦除中 0xff：獲取失敗
	備註	—
MIDI 通信協定函數		
60	void setNoteOn (uint8_t noteNumber, uint8_t velocity, uint8_t channel)	
	描述	發送音符開
	參數	noteNumber：音符，範圍 0~127 velocity：力度，範圍 0~127 channel：通道，範圍 1~16（打擊樂器音色固定在通道 10）
	返回值	—
	備註	當使用通道 10 時為打擊樂，noteNumber 範圍 24~84，打擊樂種類見附錄一
61	void setNoteOff (uint8_t noteNumber, uint8_t velocity, uint8_t channel)	
	描述	發送音符關
	參數	noteNumber：音符，範圍 0~127 velocity：音符力度，範圍 0~127 channel：通道，範圍 1~16（打擊樂器音色固定在通道 10）
	返回值	—
	備註	當使用通道 10 時為打擊樂，noteNumber 範圍 24~84，打擊樂種類見附錄一
62	void setTone (uint8_t toneNumber, uint8_t channel)	
	描述	設置音色
	參數	toneNumbe：音色號，範圍 0~127 channel：通道，範圍 1~16
	返回值	—
	備註	音色號見附錄一中的音色表
63	void setChannelVolume(uint8_t channelVolume, uint8_t channel)	
	描述	設置通道音量
	參數	channelVolume ：通道音量值，範圍 0~127（0：最小；127：最大） channel：通道，範圍 1~16
	返回值	—
	備註	—
64	void setPitchBend(int16_t pitchValue, uint8_t channel)	
	描述	設置彎音音效
	參數	pitchValue：彎音值，範圍-8192~8191

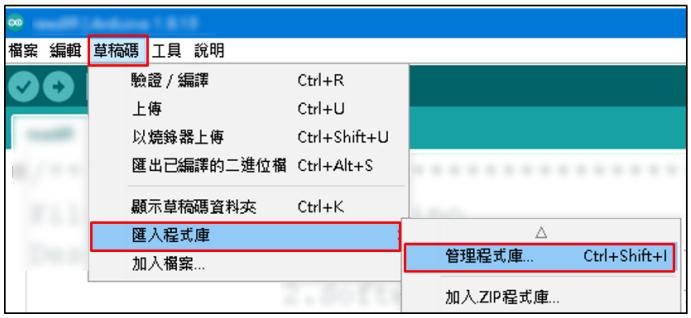
		（0：沒有彎音效果，-8192：最大向下彎曲，8191：最大向上彎曲） channel：通道，範圍 1~16
	返回值	—
	備註	—

Arduino Lib 下載及安裝

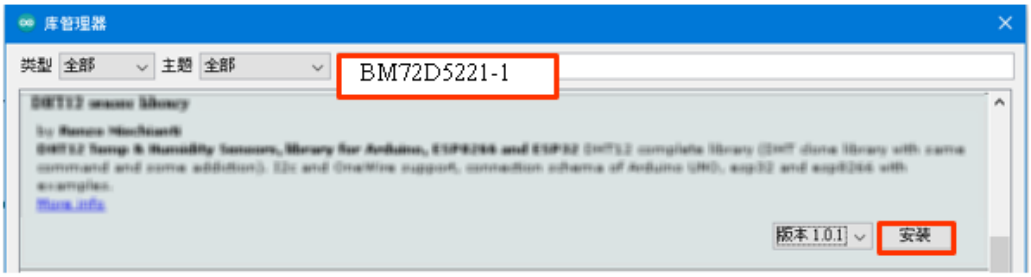
BM72D5221-1Library：可參考下面兩種方法安裝BM72D5221-1的Arduino Library

方式1：搜索安裝

搜索安裝：Arduino IDE → 草稿碼 → 匯入程式庫 → 管理程式庫 → 搜索BM72D5221-1→ 安裝



搜索安装流程 1

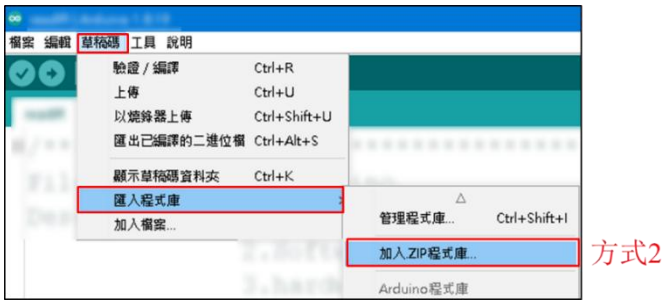


搜索安装流程 2

方式2：添加.ZIP程式庫程式，需提前下載.ZIP程式庫

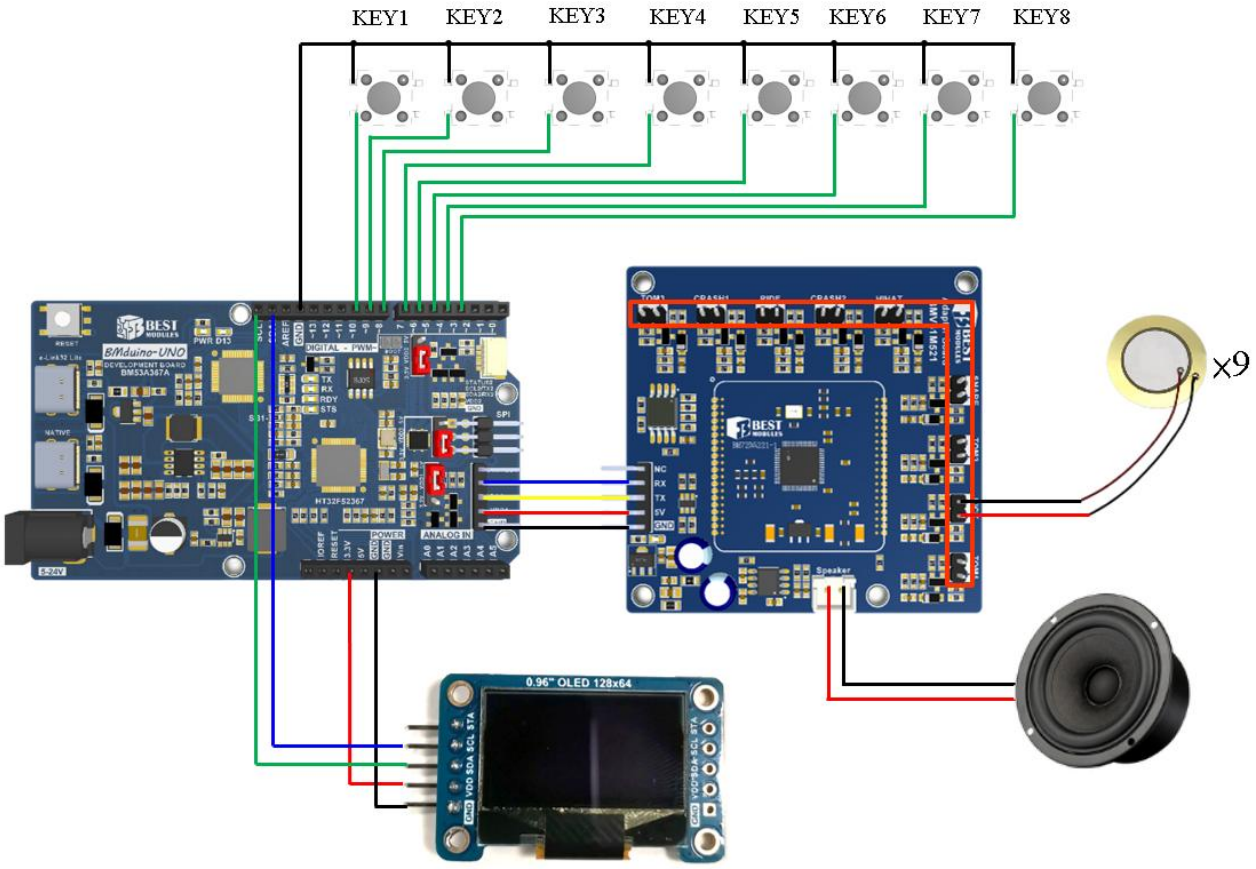
下載方法：打開倍創官方網站 (https://www.bestmodulescorp.com/ bm72d5221-1.html)，下載“文件”菜單下的Arduino範常式式(BM72D5221-1 Library)。

添加.ZIP程式庫：Arduino IDE → 草稿碼 → 匯入程式庫 → 加入.ZIP程式庫...



Arduino 範例

範例 1：drumDemo



實物連接示意圖

範例1實現功能：連接9個壓電陶瓷片傳感器和喇叭，配合按鍵實現簡易電子鼓功能，操作界面在OLED上顯示。

按鍵功能說明：

按鍵序號	短按功能	長按3秒功能	持續長按功能
KEY1	切換界面，可切換PAGE KIT/PAGE METRO/PAGE SYS/PAGE SONG這4個界面	PAGE SYS界面下復位BMV51M521	-
KEY2	切換界面下參數設置功能（除PAGE RECORD界面）	-	-
KEY3	參數加一	-	參數連續自增
KEY4	參數減一	-	參數連續自減

KEY5	保存對應界面下設置的參數（除PAGE RECORD界面）	PAGE RECORD界面下停止錄音且不保存	-
KEY6	進入PAGE RECORD界面/開始錄音/停止錄音並保存	-	-
KEY7	節拍器開關	打開/關閉打擊樂軌道	-
KEY8	播放/停止播放歌曲	-	-

Example中应用到的OLED介绍：采用倍创的0.96寸OLED模塊BMD31M090

1．範例打開：文件→示例→Lib選擇(BM72D5221-1)→選擇範例(drumDemo)

2．示例說明：

a.宏定義&枚舉&聲明變量&構建對象&封裝函數

```
#include " BM72D5221-1.h"
#include "OledShow.h"
#include "BM_Button.h"

/*宏定義按鍵引腳*/
#define KEY_PAGE              10
#define KEY_PARAM_FUNC     9
#define KEY_INC                8
#define KEY_DEC                7
#define KEY_SAVE          6
#define KEY_RECORD       5
#define KEY_METRO           4
#define KEY_PLAY_STOP      3
#define ACTIVE_LEVEL       LOW
#define KEY_PIN_NUM_MAX     8

/*宏定義按鍵時間*/
#define BUTTON_LONG_TIME    150
#define BUTTON_REPEAT_SPEED 10

/*枚舉按鍵功能*/
enum KEY_FUNC_EVENT
{
```

```
    KEY_NONE_FUNC,
    KEY_PAGE_SET = 4, //KEY1 短按
    KEY_SYS_RESET = 3, // KEY1 短按長按 3 秒
    KEY_PARAM_FUNC_SET = 8, //KEY2 短按
    KEY_INC_PARAM = 12, // KEY3 短按
    KEY_INC_PARAM_CONTINUE = 9, // KEY3 持續長按
    KEY_DEC_PARAM = 16, // KEY4 短按
    KEY_DEC_PARAM_CONTINUE = 13, // KEY4 持續長按
    KEY_SAVE_PARAM = 20, // KEY5 短按
    KEY_NO_SAVE_RECORD = 19, // KEY5 長按 3 秒
    KEY_RECORD_SW = 24, // KEY6 短按
    KEY_METRO_SW = 28, // KEY7 短按
    KEY_TRACK_SW = 27, // KEY7 長按 3 秒
    KEY_PLAY_STOP_SW = 32, // KEY8 短按
};

/*使用枚舉的方式巨集定義鼓的各類功能資料*/
#define SET_KIT_PARAM_ENUM_VALUE\
    X_SET_KIT_PARAM(KIT_GROUP, 0)\
    X_SET_KIT_PARAM(KIT_TIMBRE, 1)\
    X_SET_KIT_PARAM(KIT_VOLUME, 2)\
    X_SET_KIT_PARAM(KIT_PHASE, 3)\
    X_SET_KIT_PARAM(KIT_FORCE, 4)\
    X_SET_KIT_PARAM(KIT_TRIM, 5)\
    X_SET_KIT_PARAM(KIT_SAVE_GROUP, 6)

enum SET_KIT_PARAM_ENUM{
    #define X_SET_KIT_PARAM(a, b)   a = b,
    SET_KIT_PARAM_ENUM_VALUE
    #undef  X_SET_KIT_PARAM
};

#define SET_METRO_PARAM_ENUM_VALUE\
    X_SET_METRO_PARAM(METRO_BPM, 0)\
    X_SET_METRO_PARAM(METRO_BEAT, 1)\
    X_SET_METRO_PARAM(METRO_RHYTHM, 2)\
    X_SET_METRO_PARAM(METRO_VOLUME, 3)\
    X_SET_METRO_PARAM(METRO_SOUND, 4)

enum SET_METRO_PARAM_ENUM{
    #define X_SET_METRO_PARAM(a, b)   a = b,
    SET_METRO_PARAM_ENUM_VALUE
    #undef  X_SET_METRO_PARAM
};

#define SET_SYS_PARAM_ENUM_VALUE\
    X_SET_SYS_PARAM(SYS_SOUND_SW, 0)

enum SET_SYS_PARAM_ENUM{
    #define X_SET_SYS_PARAM(a, b)   a = b,
    SET_SYS_PARAM_ENUM_VALUE
    #undef  X_SET_SYS_PARAM
};
```

```
#define SET_SONG_PARAM_ENUM_VALUE\
   X_SET_SONG_PARAM(SONG_NUM, 0)\
   X_SET_SONG_PARAM(SONG_PLAY_MODE, 1)\
   X_SET_SONG_PARAM(SONG_VOLUME, 2)


enum SET_SONG_PARAM_ENUM{
   #define X_SET_SONG_PARAM(a, b)   a = b,
   SET_SONG_PARAM_ENUM_VALUE
   #undef  X_SET_SONG_PARAM
};

#define SET_RECORD_PARAM_ENUM_VALUE\
   X_SET_RECORD_PARAM(RECORD_NUM, 0)

enum SET_RECORD_PARAM_ENUM{
   #define X_SET_RECORD_PARAM(a, b)   a = b,
   SET_RECORD_PARAM_ENUM_VALUE
   #undef  X_SET_RECORD_PARAM
};

/*定義變量*/
char pageString[5][12] = {"PAGE KIT", "PAGE METRO", "PAGE SYS", "PAGE SONG", "PAGE RECORD"};
char trigchannelToDrum[14][7] = {"KICK", "TOM1", "TOM2", "TOM3", "SNARE", " ", "CRASH1", " ",
"CRASH2", " ", "RIDE", " ", " ", "HIHAT"};
char trimchannelToDrum[9][7] = {"KICK", "SNARE", "TOM1", "TOM2", "TOM3", "HIHAT", "CRASH1",
"CRASH2", "RIDE"};
char swString[2][4] = {"off", "on"};
char autoOffString[3][4] = {"15m", "30m", "no"};
char phaseString[15][4] = {"L07", "L06", "L05", "L04", "L03", "L02", "L01", "-00", "R01", "R02", "R03", "R04",
"R05", "R06", "R07"};
char rhythmString[4][5] = {"1/4", "1/8", "3/8", "1/16"};
char playModeString[3][7] = {"loop", "list", "single"};
char recStaString[3][7] = {"stop", "start", "ing..."}; //“ing...”代表錄音保存和不保存執行中
uint8_t pageRecordFlag = 0;//進入錄音界面的標誌位元
uint8_t recordSaveNum = 0;//錄音資料保存的位置序號
uint8_t recordSaveNumFlag = 0;//定義錄音資料保存序號標誌位元用於刷新 OLED 顯示
uint8_t metroSWFlag = 0;//BMV51M521 默認節拍器關閉
uint8_t trackSWFlag = 1;//BMV51M521 預設背景軌道音樂打開
uint8_t oledShowFlag = 1;//定義變化的標誌位元變數用於刷新 OLED 顯示
uint8_t saveFlag = 0; //定義保存標誌位用於刷新 OLED 顯示
uint8_t playFlag = 0; // 定義播放狀態標誌位用於刷新 OLED 顯示
uint8_t resetFlag = 0;//定義重定的標誌位元用於刷新 OLED 顯示
uint8_t theLastPlayState = 0;//定義最後一次播放狀態變數用於刷新 OLED 顯示

/*聲明功能參數結構體*/
template<typename T, typename... Args>
struct FuncStruct
{
   T argMin; //參數最小值
   T argMax; //參數最大值
   T (*getCurrentVal)(uint8_t, uint8_t, uint8_t, uint8_t); //函數指針，用於獲取當前參數值
   bool (BM72D5221_1::*funcSet)(Args... args); //函數指針，指向各種參數設置功能函數
```

```
};

/*聲明參數設置功能結構體*/
template<typename T, typename... Args>
struct PageSubStruct
{
   uint8_t setParamFuncNumMax; //對應界面下的參數設置功能總數量
   uint8_t setParamFuncNum; //對應界面下的參數設置功能序號
   FuncStruct<T, Args...> funcList[];//參數設置功能數組列表
};

/*聲明界面結構體*/
struct PageStruct
{
   uint8_t pageNumMax; //界面的總數量
   uint8_t pageNum; //界面的序號
};

/*構建對象*/
BM_Button myDrumButton(KEY_PIN_NUM_MAX); //8 個按鍵
OledShow<SSD1306_128x64> myOledShow; // SSD1306_128x64 為使用的 OLED 型號規格
BM72D5221_1 myMIDISerial(&Serial1); //使用硬體串口 Serial1 進行通信

/**************************************************************************
描述:   判斷輸入的參數是否有效
參數:
   輸入:     pageNum: 界面序號（0~4）
                  0: page kit
                  1: page metro
                  2: page sys
                  3: page song
                  4: page record
             funcNum: 對應界面下的參數設置功能序號
   輸出:
返回值:     0: invalid
             1: valid
其他:
**************************************************************************/
uint8_t validParamEnum(uint8_t pageNum, uint8_t funcNum)
{
          uint8_t result = 0;
          switch(pageNum)
          {
      case 0:
         switch(funcNum)
         {
            #define X_SET_KIT_PARAM(a, b) case a:
               SET_KIT_PARAM_ENUM_VALUE
            #undef X_SET_KIT_PARAM
            result = 1;
            break;
            default:
               result = 0;
```

```
            break;
        }
    break;
    case 1:
        switch(funcNum)
        {
            #define X_SET_METRO_PARAM(a, b) case a:
                SET_METRO_PARAM_ENUM_VALUE
            #undef X_SET_METRO_PARAM
            result = 1;
            break;
            default:
                result = 0;
            break;
        }
    break;
    case 2:
        switch(funcNum)
        {
            #define X_SET_SYS_PARAM(a, b) case a:
                SET_SYS_PARAM_ENUM_VALUE
            #undef X_SET_SYS_PARAM
            result = 1;
            break;
            default:
                result = 0;
            break;
        }
    break;
    case 3:
        switch(funcNum)
        {
            #define X_SET_SONG_PARAM(a, b) case a:
                SET_SONG_PARAM_ENUM_VALUE
            #undef X_SET_SONG_PARAM
            result = 1;
            break;
            default:
                result = 0;
            break;
        }
    break;
    case 4:
        switch(funcNum)
        {
            #define X_SET_RECORD_PARAM(a, b) case a:
                SET_RECORD_PARAM_ENUM_VALUE
            #undef X_SET_RECORD_PARAM
            result = 1;
            break;
            default:
                result = 0;
            break;
        }
    break;
    default:
```

```
            result = 0;
        break;
        }
        return result;
}

/*************************************************************************
描述:   獲取當前參數值，除了 BPM
參數:
   輸入:     pageNum: 界面序號（0~4）
                      0: page kit
                      1: page metro
                      2: page sys
                      3: page song
                      4: page record
             funcNum: 對應界面下的參數設置功能序號
             group: 鼓組序號（僅在 page kit 界面參數設置功能下用到）
             trigDrumChannel: 觸發的鼓通道（僅在 page kit 界面參數設置功能下用到）
   輸出:
返回值:     參數值
其他:
*************************************************************************/
uint8_t getCurrentValFunc(uint8_t pageNum, uint8_t funcNum, uint8_t group, uint8_t trigDrumChannel)
{
   switch(pageNum)
   {
      case 0://page kit
         switch(funcNum)
         {
            case 0:
               return myMIDISerial.getDrumKitGroup();
            break;
            case 1:
               return myMIDISerial.getDrumKitTimbre();
            break;
            case 2:
               return myMIDISerial.getDrumKitVolume();
            break;
            case 3:
               return myMIDISerial.getDrumKitPhase();
            break;
            case 4:
               return myMIDISerial.getDrumKitForce();
            break;
            case 5:
               return myMIDISerial.getDrumKitTrimChannel();
            break;
            case 6:
               return myMIDISerial.getDrumKitParamSaveUserGroup();
            break;
            default:
               return 0xff;
            break;
         }
```

```
    break;
    case 1://page metro
      switch(funcNum)
      {
        case 0:
          return myMIDISerial.getMetroBeat();
        break;
        case 1:
          return myMIDISerial.getMetroRhythm();
        break;
        case 2:
          return myMIDISerial.getMetroVolume();
        break;
        case 3:
          return myMIDISerial.getMetroSound();
        break;
        default:
          return 0xff;
        break;
      }
    break;
    case 2://page sys
      switch(funcNum)
      {
        case 0:
          return myMIDISerial.getLocalAudioSW();
        break;
        default:
          return 0xff;
        break;
      }
    break;
    case 3://page song
      switch(funcNum)
      {
        case 0:
          return myMIDISerial.getSongNum();
        break;
        case 1:
          return myMIDISerial.getSongPlayMode();
        break;
        case 2:
          return myMIDISerial.getSongVolume();
        break;
        default:
          return 0xff;
        break;
      }
    break;
    case 4://page record
      switch(funcNum)
      {
        case 0:
          return myMIDISerial.getRecordSaveNum();
        break;
        default:
```

```
                return 0xff;
            break;
        }
    break;
    default:
        return 0xff;
    break;

  }
}

/*****************************************************************************
描述:   獲取當前 BPM 參數值
參數:
  輸入:     pageNum: 界面序號（0~4）
                    0: page kit
                    1: page metro
                    2: page sys
                    3: page song
                    4: page record
            funcNum: 對應界面下的參數設置功能序號
            group: 鼓組序號（僅在 page kit 界面參數設置功能下用到）
            trigDrumChannel: 觸發的鼓通道（僅在 page kit 界面參數設置功能下用到）
  輸出:
返回值:     BPM 值
其他:
*****************************************************************************/
uint16_t getBPMCurrentValFunc(uint8_t pageNum, uint8_t funcNum, uint8_t group, uint8_t trigDrumChannel)
{
  return myMIDISerial.getMetroBPM();
}

/*定義界面結構體變量，錄音界面 page record 獨立*/
PageStruct pageParam = {4, 0};

/*定義對應界面下的參數設置功能結構體變量*/
PageSubStruct<uint8_t, uint8_t> pageKitSetParam =
{

  7, 0,
  {
    {DRUM_GROUP_MIN, DRUM_GROUP_MAX, &getCurrentValFunc,
&BM72D5221_1::setDrumKitGroup},
    {DRUM_TIMBRE_MIN, DRUM_TIMBRE_MAX, &getCurrentValFunc,
&BM72D5221_1::setDrumKitTimbre},
    {DRUM_VOLUME_MIN, DRUM_VOLUME_MAX, &getCurrentValFunc,
&BM72D5221_1::setDrumKitVolume},
    {DRUM_PHASE_MIN, DRUM_PHASE_MAX, &getCurrentValFunc, &BM72D5221_1::setDrumKitPhase},
    {DRUM_FORCE_MIN, DRUM_FORCE_MAX, &getCurrentValFunc, &BM72D5221_1::setDrumKitForce},
    {DRUM_TRIM_CHANNEL_MIN, 8, &getCurrentValFunc, &BMV51M521::setDrumKitTrim},
    {DRUM_GROUP_USER, DRUM_GROUP_MAX, &getCurrentValFunc,
&BM72D5221_1::setDrumKitParamSaveUserGroup}
  }
```

```
};

PageSubStruct<uint8_t, uint8_t> pageMetroSetParam =
{
   5, 0,
   {
      {METRO_BEAT_MIN, METRO_BEAT_MAX, &getCurrentValFunc, &BM72D5221_1::setMetroBeat},
      {METRO_RHYTHM_MIN, METRO_RHYTHM_MAX, &getCurrentValFunc,
&BM72D5221_1::setMetroRhythm},
      {METRO_VOLUME_MIN, METRO_VOLUME_MAX, &getCurrentValFunc,
&BM72D5221_1::setMetroVolume},
      {METRO_SOUND_MIN, METRO_SOUND_MAX, &getCurrentValFunc,
&BM72D5221_1::setMetroSound}
   }
};

PageSubStruct<uint16_t, uint16_t> pageMetroSetBPMParam =
{
   1, 0,
   {
      {METRO_BPM_MIN, METRO_BPM_MAX, &getBPMCurrentValFunc, &BM72D5221_1::setMetroBPM},
   }
};

PageSubStruct<uint8_t, uint8_t> pageSysSetParam =
{
   1, 0,
   {
      {SYS_LOCAL_AUDIO_CLOSE, SYS_LOCAL_AUDIO_OPEN, &getCurrentValFunc,
&BM72D5221_1::setLocalAudioSW},
   }
};

PageSubStruct<uint8_t, uint8_t> pageSongSetParam =
{
   3, 0,
   {
      {SONG_MIN, SONG_MAX, &getCurrentValFunc, &BM72D5221_1::setSongNum},
      {SONG_PLAY_MODE_MIN, SONG_PLAY_MODE_MAX, &getCurrentValFunc,
&BM72D5221_1::setSongPlayMode},
      {SONG_VOLUME_MIN, SONG_VOLUME_MAX, &getCurrentValFunc,
&BM72D5221_1::setSongVolume}
   }
};

PageSubStruct<uint8_t, uint8_t> pageRecordSetParam =
{
   1, 0,
   {
      {RECORD_ID_MIN, RECORD_ID_MAX, &getCurrentValFunc, &BM72D5221_1::setRecordSaveNum}
   }
};

PageSubStruct<uint8_t, uint8_t> *pageSubSetParam[4] = {&pageKitSetParam, &pageMetroSetParam,
&pageSysSetParam, &pageSongSetParam};
```

```
PageSubStruct<uint8_t> pagekitSaveParam =
{
   0, 0,
   {
      {0, 0, 0, &BM72D5221_1::setDrumKitParamSave},
      {0, 0, 0, &BM72D5221_1::setDrumKitTrimSave}
   }
};

PageSubStruct<uint8_t> pageMetroSaveParam =
{
   0, 0,
   {
      {0, 0, 0, &BM72D5221_1::setMetroParamSave}
   }
};

PageSubStruct<uint8_t> pageSysSaveParam =
{
   0, 0,
   {
      {0, 0, 0, &BM72D5221_1::setSysParamSave}
   }
};

PageSubStruct<uint8_t> pageSongSaveParam =
{
   0, 0,
   {
      {0, 0, 0, &BM72D5221_1::setSongParamSave}
   }
};

PageSubStruct<uint8_t> *pageSubSaveParam[4] = {&pagekitSaveParam, &pageMetroSaveParam,
&pageSysSaveParam, &pageSongSaveParam};

/*******************************************************************************
描述:   選擇不同的參數設置函數
參數:
   輸入:    x: 界面序號（0~3）
                        0: page kit
                        1: page metro
                        2: page sys
                        3: page song
            y: 對應界面下的參數設置功能序號
   輸出:
返回值:
其他:
*******************************************************************************/
#define FUNC_SELECT(x, y) \
if(validParamEnum(x, y))\
{\
   uint8_t setParam1 = 0;\
   uint16_t setParam2 = 0;\
   switch(x)\
```

```
    {\
        case 1:\
            if(y == 4)\
            {\
                if(pageMetroSetBPMParam.funcList[0].argMax > pageMetroSetBPMParam.funcList[0].getCurrentVal(x,
y, 0, 0))\
                {\
                    setParam2 = pageMetroSetBPMParam.funcList[0].getCurrentVal(x, y, 0, 0);\
                    (myMIDISerial.*pageMetroSetBPMParam.funcList[0].funcSet)(setParam2);\
                }\
            }\
            else \
            {\
                if(0 == myMIDISerial.getPlayState())\
                {\
                    if(pageSubSetParam[x]->funcList[y].argMax > pageSubSetParam[x]->funcList[0].getCurrentVal(x, y,
myMIDISerial.getDrumKitGroup(), myMIDISerial.getTrigChannel()))\
                    {\
                        setParam1 = pageSubSetParam[x]->funcList[0].getCurrentVal(x, y,
myMIDISerial.getDrumKitGroup(), myMIDISerial.getTrigChannel());\
                        (myMIDISerial.*pageSubSetParam[x]->funcList[y].funcSet)(setParam1);\
                    }\
                }\
            }\
        break;\
        case 0:\
        case 2:\
        case 3:\
            if(pageSubSetParam[x]->funcList[y].argMax > pageSubSetParam[x]->funcList[y].getCurrentVal(x, y,
myMIDISerial.getDrumKitGroup(), myMIDISerial.getTrigChannel()))\
            {\
                setParam1 = pageSubSetParam[x]->funcList[y].getCurrentVal(x, y, myMIDISerial.getDrumKitGroup(),
myMIDISerial.getTrigChannel());\
                (myMIDISerial.*pageSubSetParam[x]->funcList[y].funcSet)(setParam1);\
            }\
        break;\
        default:\
        break;\
    }\
}

/*****************************************************************************
描述:   執行對應參數設置功能下的參數加一
參數:
   輸入:    x: 界面序號（0~3）
                        0: page kit
                        1: page metro
                        2: page sys
                        3: page song
             y: 對應界面下的參數設置功能序號
   輸出:
返回值:
其他:
*****************************************************************************/
#define INC_PARAM(x, y) \
```

```
if(validParamEnum(x, y))\
{\
   uint8_t setParam1 = 0;\
   uint16_t setParam2 = 0;\
   switch(x)\
   {\
      case 1:\
         if(y == 4)\
         {\
            if(pageMetroSetBPMParam.funcList[0].argMax > pageMetroSetBPMParam.funcList[0].getCurrentVal(x,
y, 0, 0))\
            {\
               setParam2 = pageMetroSetBPMParam.funcList[0].getCurrentVal(x, y, 0, 0) + 1;\
               (myMIDISerial.*pageMetroSetBPMParam.funcList[0].funcSet)(setParam2);\
            }\
            else\
            {\

(myMIDISerial.*pageMetroSetBPMParam.funcList[0].funcSet)(pageMetroSetBPMParam.funcList[0].argMax);\
            }\
         }\
         else \
         {\
            if(0 == myMIDISerial.getPlayState())\
            {\
               if(pageSubSetParam[x]->funcList[y].argMax > pageSubSetParam[x]->funcList[0].getCurrentVal(x, y,
myMIDISerial.getDrumKitGroup(), myMIDISerial.getTrigChannel()))\
               {\
                  setParam1 = pageSubSetParam[x]->funcList[0].getCurrentVal(x, y,
myMIDISerial.getDrumKitGroup(), myMIDISerial.getTrigChannel()) + 1;\
                  (myMIDISerial.*pageSubSetParam[x]->funcList[y].funcSet)(setParam1);\
               }\
               else\
               {\
                  (myMIDISerial.*pageSubSetParam[x]->funcList[y].funcSet)(pageSubSetParam[x]-
>funcList[y].argMax);\
               }\
            }\
         }\
      break;\
      case 0:\
      case 2:\
      case 3:\
         if(pageSubSetParam[x]->funcList[y].argMax > pageSubSetParam[x]->funcList[y].getCurrentVal(x, y,
myMIDISerial.getDrumKitGroup(), myMIDISerial.getTrigChannel()))\
         {\
            setParam1 = pageSubSetParam[x]->funcList[y].getCurrentVal(x, y, myMIDISerial.getDrumKitGroup(),
myMIDISerial.getTrigChannel()) + 1;\
            (myMIDISerial.*pageSubSetParam[x]->funcList[y].funcSet)(setParam1);\
         }\
         else\
         {\
            (myMIDISerial.*pageSubSetParam[x]->funcList[y].funcSet)(pageSubSetParam[x]-
>funcList[y].argMax);\
         }\
      break;\
```

```
      default:\
      break;\
   }\
}

/*****************************************************************************
描述:   執行對應參數設置功能下的參數減一
參數:
   輸入:    x: 界面序號（0~3）
                        0: page kit
                        1: page metro
                        2: page sys
                        3: page song
            y: 對應界面下的參數設置功能序號
   輸出:
返回值:
其他:
*****************************************************************************/
#define DEC_PARAM(x, y) \
if(validParamEnum(x, y))\
{\
   uint8_t setParam1 = 0;\
   uint16_t setParam2 = 0;\
   switch(x)\
   {\
      case 1:\
         if(y == 4)\
         {\
            if(pageMetroSetBPMParam.funcList[0].argMin < pageMetroSetBPMParam.funcList[0].getCurrentVal(x,
y, 0, 0))\
            {\
               setParam2 = pageMetroSetBPMParam.funcList[0].getCurrentVal(x, y, 0, 0) - 1;\
               (myMIDISerial.*pageMetroSetBPMParam.funcList[0].funcSet)(setParam2);\
            }\
            else\
            {\

(myMIDISerial.*pageMetroSetBPMParam.funcList[0].funcSet)(pageMetroSetBPMParam.funcList[0].argMin);\
            }\
         }\
         else\
         {\
            if(0 == myMIDISerial.getPlayState())\
            {\
               if(pageSubSetParam[x]->funcList[y].argMin < pageSubSetParam[x]->funcList[y].getCurrentVal(x, y,
myMIDISerial.getDrumKitGroup(), myMIDISerial.getTrigChannel()))\
               {\
                  setParam1 = pageSubSetParam[x]->funcList[y].getCurrentVal(x, y,
myMIDISerial.getDrumKitGroup(), myMIDISerial.getTrigChannel()) - 1;\
                  (myMIDISerial.*pageSubSetParam[x]->funcList[y].funcSet)(setParam1);\
               }\
               else\
               {\
                  (myMIDISerial.*pageSubSetParam[x]->funcList[y].funcSet)(pageSubSetParam[x]-
>funcList[y].argMin);\
```

```
            }\
        }\
    }\
    break;\
    case 0:\
    case 2:\
    case 3:\
        if(pageSubSetParam[x]->funcList[y].argMin < pageSubSetParam[x]->funcList[y].getCurrentVal(x, y,
myMIDISerial.getDrumKitGroup(), myMIDISerial.getTrigChannel()))\
        {\
            setParam1 = pageSubSetParam[x]->funcList[y].getCurrentVal(x, y, myMIDISerial.getDrumKitGroup(),
myMIDISerial.getTrigChannel()) - 1;\
            (myMIDISerial.*pageSubSetParam[x]->funcList[y].funcSet)(setParam1);\
        }\
        else\
        {\
            (myMIDISerial.*pageSubSetParam[x]->funcList[y].funcSet)(pageSubSetParam[x]-
>funcList[y].argMin);\
        }\
    break;\
    default:\
    break;\
  }\
}

/*****************************************************************************
描述:   保存設置的參數
參數:
  輸入:    x: 界面序號（0~3）
                    0: page kit
                    1: page metro
                    2: page sys
                    3: page song
           y: 對應界面下的參數設置功能序號
  輸出:
返回值:
其他:
*****************************************************************************/
#define SAVE_PARAM(x, y) \
if(validParamEnum(x, y))\
{\
  switch(x)\
  {\
    case 0:\
      if(5 == y)\
      {\
        (myMIDISerial.*pageSubSaveParam[x]->funcList[1].funcSet)();\
      }\
      else\
      {\
        (myMIDISerial.*pageSubSaveParam[x]->funcList[0].funcSet)();\
      }\
    break;\
    case 1:\
    case 2:\
```

```
      case 3:\
        (myMIDISerial.*pageSubSaveParam[x]->funcList[0].funcSet)();\
      break;\
      default:\
      break;\
   }\
}

/*************************************************************************
描述:  鼓功能設置按鍵初始化
参数:
   輸入:
   輸出:
返回值:
其他:
*************************************************************************/
void drumBtnInit(void)
{
   myDrumButton.btnInit(KEY_PAGE, ACTIVE_LEVEL, BUTTON_LONG_TIME,
BUTTON_REPEAT_SPEED);
   myDrumButton.btnInit(KEY_PARAM_FUNC, ACTIVE_LEVEL, BUTTON_LONG_TIME,
BUTTON_REPEAT_SPEED);
   myDrumButton.btnInit(KEY_INC, ACTIVE_LEVEL, BUTTON_LONG_TIME, BUTTON_REPEAT_SPEED);
   myDrumButton.btnInit(KEY_DEC, ACTIVE_LEVEL, BUTTON_LONG_TIME,
BUTTON_REPEAT_SPEED);
   myDrumButton.btnInit(KEY_SAVE, ACTIVE_LEVEL, BUTTON_LONG_TIME,
BUTTON_REPEAT_SPEED);
   myDrumButton.btnInit(KEY_RECORD, ACTIVE_LEVEL, BUTTON_LONG_TIME,
BUTTON_REPEAT_SPEED);
   myDrumButton.btnInit(KEY_METRO, ACTIVE_LEVEL, BUTTON_LONG_TIME,
BUTTON_REPEAT_SPEED);
   myDrumButton.btnInit(KEY_PLAY_STOP, ACTIVE_LEVEL, BUTTON_LONG_TIME,
BUTTON_REPEAT_SPEED);
}

/*************************************************************************
描述:  OLED 顯示初始化
参数:
   輸入:
   輸出:
返回值:
其他:
*************************************************************************/
void oledBegin()
{
   myOledShow.begin();
   myOledShow.clear();
   myOledShow.update();
}

/*************************************************************************
描述:  顯示設置參數數據
参数:
   輸入: pageNum：界面序號（0~3）
```

```
                        0: page kit
                        1: page metro
                        2: page sys
                        3: page song
        type：枚舉變量 DATATYPE_ENUM 下的值（參數設置功能命令）
  輸出:
返回值:
其他:   如果是在錄音界面下，pageNum 無效，任何數位都行
*****************************************************************************/
void oledShowParam(uint8_t pageNum, DATATYPE_ENUM type)
{
  static uint8_t firstFrush = 0;
  static uint8_t lastPageNum = 0;
  static uint8_t lastPageRecordFlag = 0;

  if(!pageRecordFlag)//其他界面
  {
    lastPageRecordFlag = pageRecordFlag;
    if(pageNum != lastPageNum)
    {
      firstFrush = 0;
    }
    switch(pageNum)
    {
      case PAGE_KIT:
        if(0 == firstFrush)
        {
          firstFrush = 1;
          myOledShow.clear(0, 0, 128, 8);
          myOledShow.setCursor(40, 0);
          myOledShow.print(pageString[pageNum]);

          myOledShow.clear(0, 16, 54, 23);
          myOledShow.setCursor(0, 2);
          myOledShow.print("drum kit:");

          myOledShow.clear(65, 16, 120, 23);
          myOledShow.setCursor(65, 2);
          myOledShow.print("save kit:");

          myOledShow.clear(0, 24, 54, 31);
          myOledShow.setCursor(0, 3);
          myOledShow.print("timbre:");

          myOledShow.clear(0, 32, 54, 39);
          myOledShow.setCursor(0, 4);
          myOledShow.print("volume:");

          myOledShow.clear(0, 40, 54, 47);
          myOledShow.setCursor(0, 5);
          myOledShow.print("phase:");

          myOledShow.clear(0, 48, 54, 55);
          myOledShow.setCursor(0, 6);
          myOledShow.print("force:");
```

```
    myOledShow.clear(0, 56, 34, 63);
    myOledShow.setCursor(0, 7);
    myOledShow.print("trim:");
  }
  switch(type)
  {
    case SET_DRUMKIT_GROUP:
      myOledShow.clear(54, 16, 64, 23);
      myOledShow.setCursor(54, 2);
      myOledShow.invertText(true);
      myOledShow.print(myMIDISerial.getDrumKitGroup());
      myOledShow.invertText(false);
      myOledShow.clear(120, 16, 127, 23);
      myOledShow.setCursor(120, 2);
      myOledShow.print(myMIDISerial.getDrumKitParamSaveUserGroup());
      myOledShow.clear(54, 24, 90, 31);
      myOledShow.setCursor(54, 3);
      myOledShow.print(myMIDISerial.getDrumKitTimbre());
      myOledShow.clear(54, 32, 78, 39);
      myOledShow.setCursor(54, 4);
      myOledShow.print(myMIDISerial.getDrumKitVolume());
      myOledShow.clear(54, 40, 74, 47);
      myOledShow.setCursor(54, 5);
      myOledShow.print(phaseString[myMIDISerial.getDrumKitPhase()]);
      myOledShow.clear(54, 48, 74, 55);
      myOledShow.setCursor(54, 6);
      myOledShow.print(myMIDISerial.getDrumKitForce());
      myOledShow.clear(34, 56, 74, 63);
      myOledShow.setCursor(34, 7);
      myOledShow.print(trimchannelToDrum[myMIDISerial.getDrumKitTrimChannel()]);
    break;
    case SET_DRUMKIT_TIMBRE:
      myOledShow.clear(54, 16, 64, 23);
      myOledShow.setCursor(54, 2);
      myOledShow.print(myMIDISerial.getDrumKitGroup());
      myOledShow.clear(120, 16, 127, 23);
      myOledShow.setCursor(120, 2);
      myOledShow.print(myMIDISerial.getDrumKitParamSaveUserGroup());
      myOledShow.clear(54, 24, 90, 31);
      myOledShow.setCursor(54, 3);
      myOledShow.invertText(true);
      myOledShow.print(myMIDISerial.getDrumKitTimbre());
      myOledShow.invertText(false);
      myOledShow.clear(54, 32, 78, 39);
      myOledShow.setCursor(54, 4);
      myOledShow.print(myMIDISerial.getDrumKitVolume());
      myOledShow.clear(54, 40, 74, 47);
      myOledShow.setCursor(54, 5);
      myOledShow.print(phaseString[myMIDISerial.getDrumKitPhase()]);
      myOledShow.clear(54, 48, 74, 55);
      myOledShow.setCursor(54, 6);
      myOledShow.print(myMIDISerial.getDrumKitForce());
      myOledShow.clear(34, 56, 74, 63);
      myOledShow.setCursor(34, 7);
      myOledShow.print(trimchannelToDrum[myMIDISerial.getDrumKitTrimChannel()]);
```

```
            break;
            case SET_DRUMKIT_VOLUME:
              myOledShow.clear(54, 16, 64, 23);
              myOledShow.setCursor(54, 2);
              myOledShow.print(myMIDISerial.getDrumKitGroup());
              myOledShow.clear(120, 16, 127, 23);
              myOledShow.setCursor(120, 2);
              myOledShow.print(myMIDISerial.getDrumKitParamSaveUserGroup());
              myOledShow.clear(54, 24, 90, 31);
              myOledShow.setCursor(54, 3);
              myOledShow.print(myMIDISerial.getDrumKitTimbre());
              myOledShow.clear(54, 32, 78, 39);
              myOledShow.setCursor(54, 4);
              myOledShow.invertText(true);
              myOledShow.print(myMIDISerial.getDrumKitVolume());
              myOledShow.invertText(false);
              myOledShow.clear(54, 40, 74, 47);
              myOledShow.setCursor(54, 5);
              myOledShow.print(phaseString[myMIDISerial.getDrumKitPhase()]);
              myOledShow.clear(54, 48, 74, 55);
              myOledShow.setCursor(54, 6);
              myOledShow.print(myMIDISerial.getDrumKitForce());
              myOledShow.clear(34, 56, 74, 63);
              myOledShow.setCursor(34, 7);
              myOledShow.print(trimchannelToDrum[myMIDISerial.getDrumKitTrimChannel()]);
            break;
            case SET_DRUMKIT_PHASE:
              myOledShow.clear(54, 16, 64, 23);
              myOledShow.setCursor(54, 2);
              myOledShow.print(myMIDISerial.getDrumKitGroup());
              myOledShow.clear(120, 16, 127, 23);
              myOledShow.setCursor(120, 2);
              myOledShow.print(myMIDISerial.getDrumKitParamSaveUserGroup());
              myOledShow.clear(54, 24, 90, 31);
              myOledShow.setCursor(54, 3);
              myOledShow.print(myMIDISerial.getDrumKitTimbre());
              myOledShow.clear(54, 32, 78, 39);
              myOledShow.setCursor(54, 4);
              myOledShow.print(myMIDISerial.getDrumKitVolume());
              myOledShow.clear(54, 40, 74, 47);
              myOledShow.setCursor(54, 5);
              myOledShow.invertText(true);
              myOledShow.print(phaseString[myMIDISerial.getDrumKitPhase()]);
              myOledShow.invertText(false);
              myOledShow.clear(54, 48, 74, 55);
              myOledShow.setCursor(54, 6);
              myOledShow.print(myMIDISerial.getDrumKitForce());
              myOledShow.clear(34, 56, 74, 63);
              myOledShow.setCursor(34, 7);
              myOledShow.print(trimchannelToDrum[myMIDISerial.getDrumKitTrimChannel()]);
            break;
            case SET_DRUMKIT_FORCE:
              myOledShow.clear(54, 16, 64, 23);
              myOledShow.setCursor(54, 2);
              myOledShow.print(myMIDISerial.getDrumKitGroup());
              myOledShow.clear(120, 16, 127, 23);
```

```
            myOledShow.setCursor(120, 2);
            myOledShow.print(myMIDISerial.getDrumKitParamSaveUserGroup());
            myOledShow.clear(54, 24, 90, 31);
            myOledShow.setCursor(54, 3);
            myOledShow.print(myMIDISerial.getDrumKitTimbre());
            myOledShow.clear(54, 32, 78, 39);
            myOledShow.setCursor(54, 4);
            myOledShow.print(myMIDISerial.getDrumKitVolume());
            myOledShow.clear(54, 40, 74, 47);
            myOledShow.setCursor(54, 5);
            myOledShow.print(phaseString[myMIDISerial.getDrumKitPhase()]);
            myOledShow.clear(54, 48, 74, 55);
            myOledShow.setCursor(54, 6);
            myOledShow.invertText(true);
            myOledShow.print(myMIDISerial.getDrumKitForce());
            myOledShow.invertText(false);
            myOledShow.clear(34, 56, 74, 63);
            myOledShow.setCursor(34, 7);
            myOledShow.print(trimchannelToDrum[myMIDISerial.getDrumKitTrimChannel()]);
        break;
        case SET_DRUMKIT_TRIM:
            myOledShow.clear(54, 16, 64, 23);
            myOledShow.setCursor(54, 2);
            myOledShow.print(myMIDISerial.getDrumKitGroup());
            myOledShow.clear(120, 16, 127, 23);
            myOledShow.setCursor(120, 2);
            myOledShow.print(myMIDISerial.getDrumKitParamSaveUserGroup());
            myOledShow.clear(54, 24, 90, 31);
            myOledShow.setCursor(54, 3);
            myOledShow.print(myMIDISerial.getDrumKitTimbre());
            myOledShow.clear(54, 32, 78, 39);
            myOledShow.setCursor(54, 4);
            myOledShow.print(myMIDISerial.getDrumKitVolume());
            myOledShow.clear(54, 40, 74, 47);
            myOledShow.setCursor(54, 5);
            myOledShow.print(phaseString[myMIDISerial.getDrumKitPhase()]);
            myOledShow.clear(54, 48, 74, 55);
            myOledShow.setCursor(54, 6);
            myOledShow.print(myMIDISerial.getDrumKitForce());
            myOledShow.clear(34, 56, 74, 63);
            myOledShow.setCursor(34, 7);
            myOledShow.invertText(true);
            myOledShow.print(trimchannelToDrum[myMIDISerial.getDrumKitTrimChannel()]);
            myOledShow.invertText(false);
        break;
        case SET_DRUMKIT_SAVE_GTOUP:
            myOledShow.clear(54, 16, 64, 23);
            myOledShow.setCursor(54, 2);
            myOledShow.print(myMIDISerial.getDrumKitGroup());
            myOledShow.clear(120, 16, 127, 23);
            myOledShow.setCursor(120, 2);
            myOledShow.invertText(true);
            myOledShow.print(myMIDISerial.getDrumKitParamSaveUserGroup());
            myOledShow.invertText(false);
            myOledShow.clear(54, 24, 90, 31);
            myOledShow.setCursor(54, 3);
```

```
                myOledShow.print(myMIDISerial.getDrumKitTimbre());
                myOledShow.clear(54, 32, 78, 39);
                myOledShow.setCursor(54, 4);
                myOledShow.print(myMIDISerial.getDrumKitVolume());
                myOledShow.clear(54, 40, 74, 47);
                myOledShow.setCursor(54, 5);
                myOledShow.print(phaseString[myMIDISerial.getDrumKitPhase()]);
                myOledShow.clear(54, 48, 74, 55);
                myOledShow.setCursor(54, 6);
                myOledShow.print(myMIDISerial.getDrumKitForce());
                myOledShow.clear(34, 56, 74, 63);
                myOledShow.setCursor(34, 7);
                myOledShow.print(trimchannelToDrum[myMIDISerial.getDrumKitTrimChannel()]);
            break;
            default:
            break;
        }
        lastPageNum = pageNum;
    break;
    case PAGE_METRO:
        if(0 == firstFrush)
        {
            firstFrush = 1;
            myOledShow.clear(0, 0, 128, 8);
            myOledShow.setCursor(40, 0);
            myOledShow.print(pageString[pageNum]);

            myOledShow.clear(0, 16, 127, 23);
            myOledShow.setCursor(0, 2);
            myOledShow.print("beat:");

            myOledShow.clear(0, 24, 54, 31);
            myOledShow.setCursor(0, 3);
            myOledShow.print("rhythm:");

            myOledShow.clear(0, 32, 54, 39);
            myOledShow.setCursor(0, 4);
            myOledShow.print("volume:");

            myOledShow.clear(0, 40, 54, 47);
            myOledShow.setCursor(0, 5);
            myOledShow.print("sound:");

            myOledShow.clear(0, 48, 54, 55);
            myOledShow.setCursor(0, 6);
            myOledShow.print("bpm:");

            myOledShow.clear(0, 56, 74, 63);

        }
        switch(type)
        {
            case SET_METRO_BEAT:
                myOledShow.clear(54, 16, 82, 23);
                myOledShow.setCursor(54, 2);
                myOledShow.invertText(true);
```

```
            myOledShow.print(myMIDISerial.getMetroBeat());
            myOledShow.invertText(false);
            myOledShow.clear(54, 24, 90, 31);
            myOledShow.setCursor(54, 3);
            myOledShow.print(rhythmString[myMIDISerial.getMetroRhythm()-1]);
            myOledShow.clear(54, 32, 78, 39);
            myOledShow.setCursor(54, 4);
            myOledShow.print(myMIDISerial.getMetroVolume());
            myOledShow.clear(54, 40, 74, 47);
            myOledShow.setCursor(54, 5);
            myOledShow.print(myMIDISerial.getMetroSound());
            myOledShow.clear(54, 48, 74, 55);
            myOledShow.setCursor(54, 6);
            myOledShow.print(myMIDISerial.getMetroBPM());
        break;
        case SET_METRO_RHYTHM:
            myOledShow.clear(54, 16, 82, 23);
            myOledShow.setCursor(54, 2);
            myOledShow.print(myMIDISerial.getMetroBeat());
            myOledShow.clear(54, 24, 90, 31);
            myOledShow.setCursor(54, 3);
            myOledShow.invertText(true);
            myOledShow.print(rhythmString[myMIDISerial.getMetroRhythm()-1]);
            myOledShow.invertText(false);
            myOledShow.clear(54, 32, 78, 39);
            myOledShow.setCursor(54, 4);
            myOledShow.print(myMIDISerial.getMetroVolume());
            myOledShow.clear(54, 40, 74, 47);
            myOledShow.setCursor(54, 5);
            myOledShow.print(myMIDISerial.getMetroSound());
            myOledShow.clear(54, 48, 74, 55);
            myOledShow.setCursor(54, 6);
            myOledShow.print(myMIDISerial.getMetroBPM());
        break;
        case SET_METRO_VOLUME:
            myOledShow.clear(54, 16, 82, 23);
            myOledShow.setCursor(54, 2);
            myOledShow.print(myMIDISerial.getMetroBeat());
            myOledShow.clear(54, 24, 90, 31);
            myOledShow.setCursor(54, 3);
            myOledShow.print(rhythmString[myMIDISerial.getMetroRhythm()-1]);
            myOledShow.clear(54, 32, 78, 39);
            myOledShow.setCursor(54, 4);
            myOledShow.invertText(true);
            myOledShow.print(myMIDISerial.getMetroVolume());
            myOledShow.invertText(false);
            myOledShow.clear(54, 40, 74, 47);
            myOledShow.setCursor(54, 5);
            myOledShow.print(myMIDISerial.getMetroSound());
            myOledShow.clear(54, 48, 74, 55);
            myOledShow.setCursor(54, 6);
            myOledShow.print(myMIDISerial.getMetroBPM());
        break;
        case SET_METRO_SOUND:
            myOledShow.clear(54, 16, 82, 23);
            myOledShow.setCursor(54, 2);
```

```
                myOledShow.print(myMIDISerial.getMetroBeat());
                myOledShow.clear(54, 24, 90, 31);
                myOledShow.setCursor(54, 3);
                myOledShow.print(rhythmString[myMIDISerial.getMetroRhythm()-1]);
                myOledShow.clear(54, 32, 78, 39);
                myOledShow.setCursor(54, 4);
                myOledShow.print(myMIDISerial.getMetroVolume());
                myOledShow.clear(54, 40, 74, 47);
                myOledShow.setCursor(54, 5);
                myOledShow.invertText(true);
                myOledShow.print(myMIDISerial.getMetroSound());
                myOledShow.invertText(false);
                myOledShow.clear(54, 48, 74, 55);
                myOledShow.setCursor(54, 6);
                myOledShow.print(myMIDISerial.getMetroBPM());
            break;
            case SET_METRO_BPM:
                myOledShow.clear(54, 16, 82, 23);
                myOledShow.setCursor(54, 2);
                myOledShow.print(myMIDISerial.getMetroBeat());
                myOledShow.clear(54, 24, 90, 31);
                myOledShow.setCursor(54, 3);
                myOledShow.print(rhythmString[myMIDISerial.getMetroRhythm()-1]);
                myOledShow.clear(54, 32, 78, 39);
                myOledShow.setCursor(54, 4);
                myOledShow.print(myMIDISerial.getMetroVolume());
                myOledShow.clear(54, 40, 74, 47);
                myOledShow.setCursor(54, 5);
                myOledShow.print(myMIDISerial.getMetroSound());
                myOledShow.clear(54, 48, 74, 55);
                myOledShow.setCursor(54, 6);
                myOledShow.invertText(true);
                myOledShow.print(myMIDISerial.getMetroBPM());
                myOledShow.invertText(false);
            break;
            default:
            break;
        }
        lastPageNum = pageNum;
    break;
    case PAGE_SYS:
        if(0 == firstFrush)
        {
            firstFrush = 1;
            myOledShow.clear(0, 0, 128, 8);
            myOledShow.setCursor(40, 0);
            myOledShow.print(pageString[pageNum]);

            myOledShow.clear(0, 16, 127, 23);
            myOledShow.setCursor(0, 2);
            myOledShow.print("sound sw:");

            myOledShow.clear(0, 24, 54, 31);
            myOledShow.setCursor(0, 3);
            myOledShow.print("version:");
```

```
            myOledShow.clear(0, 32, 74, 39);

            myOledShow.clear(0, 40, 74, 47);

            myOledShow.clear(0, 48, 74, 55);

            myOledShow.clear(0, 56, 74, 63);

        }
        switch(type)
        {
            case SET_VOICE_SYS_SOUND_SW:
                myOledShow.clear(54, 16, 82, 23);
                myOledShow.setCursor(54, 2);
                myOledShow.invertText(true);
                myOledShow.print(swString[myMIDISerial.getLocalAudioSW()]);
                myOledShow.invertText(false);
                myOledShow.clear(54, 24, 90, 31);
                myOledShow.setCursor(54, 3);
                myOledShow.print("1.00");
            break;
            case SET_VOICE_SYS_VESION:
                myOledShow.clear(54, 16, 82, 23);
                myOledShow.setCursor(54, 2);
                myOledShow.print(swString[myMIDISerial.getLocalAudioSW()]);
                myOledShow.clear(54, 24, 90, 31);
                myOledShow.setCursor(54, 3);
                myOledShow.invertText(true);
                myOledShow.print("1.00");
                myOledShow.invertText(false);
            break;
            default:
            break;
        }
        lastPageNum = pageNum;
    break;
    case PAGE_SONG:
        if(0 == firstFrush)
        {
            firstFrush = 1;
            myOledShow.clear(0, 0, 128, 8);
            myOledShow.setCursor(40, 0);
            myOledShow.print(pageString[pageNum]);

            myOledShow.clear(0, 16, 127, 23);
            myOledShow.setCursor(0, 2);
            myOledShow.print("num:");

            myOledShow.clear(0, 24, 54, 31);
            myOledShow.setCursor(0, 3);
            myOledShow.print("play mod:");

            myOledShow.clear(0, 32, 54, 39);
            myOledShow.setCursor(0, 4);
            myOledShow.print("volume:");
```

```
            myOledShow.clear(0, 40, 74, 47);

            myOledShow.clear(0, 48, 74, 55);

            myOledShow.clear(0, 56, 74, 63);

        }
        switch(type)
        {
            case SET_SONG_NUM:
                myOledShow.clear(54, 16, 82, 23);
                myOledShow.setCursor(54, 2);
                myOledShow.invertText(true);
                myOledShow.print(myMIDISerial.getSongNum());
                myOledShow.invertText(false);
                myOledShow.clear(54, 24, 90, 31);
                myOledShow.setCursor(54, 3);
                myOledShow.print(playModeString[myMIDISerial.getSongPlayMode()]);
                myOledShow.clear(54, 32, 78, 39);
                myOledShow.setCursor(54, 4);
                myOledShow.print(myMIDISerial.getSongVolume());
            break;
            case SET_SONG_PLAY_MODE:
                myOledShow.clear(54, 16, 82, 23);
                myOledShow.setCursor(54, 2);
                myOledShow.print(myMIDISerial.getSongNum());
                myOledShow.clear(54, 24, 90, 31);
                myOledShow.setCursor(54, 3);
                myOledShow.invertText(true);
                myOledShow.print(playModeString[myMIDISerial.getSongPlayMode()]);
                myOledShow.invertText(false);
                myOledShow.clear(54, 32, 78, 39);
                myOledShow.setCursor(54, 4);
                myOledShow.print(myMIDISerial.getSongVolume());
            break;
            case SET_SONG_VOLUME:
                myOledShow.clear(54, 16, 82, 23);
                myOledShow.setCursor(54, 2);
                myOledShow.print(myMIDISerial.getSongNum());
                myOledShow.clear(54, 24, 90, 31);
                myOledShow.setCursor(54, 3);
                myOledShow.print(playModeString[myMIDISerial.getSongPlayMode()]);
                myOledShow.clear(54, 32, 78, 39);
                myOledShow.setCursor(54, 4);
                myOledShow.invertText(true);
                myOledShow.print(myMIDISerial.getSongVolume());
                myOledShow.invertText(false);
            break;
            default:
            break;
        }
        lastPageNum = pageNum;
    break;
    default:
    break;
  }
```

```
if(SET_VOICE_SYS_REFACTORY_TRUE == type)
{
   if(myMIDISerial.getSysResetFlag())
   {
      myOledShow.clear(90, 16, 127, 23);
      myOledShow.setCursor(90, 2);
      myOledShow.print("RST...");
      myOledShow.update();
   }
   while(myMIDISerial.getSysResetFlag())//按鍵在此期間無法執行
   {
      myMIDISerial.scanSlaveUpdateData();
   }
   resetFlag = 0;//復位完成
   pageParam.pageNum = 0;
   pageSubSetParam[pageParam.pageNum]->setParamFuncNum = 0;
   myMIDISerial.pageSwitch();//預設界面：page kit
   metroSWFlag = 0;
   myOledShow.clear(0, 0, 128, 8);
   myOledShow.setCursor(40, 0);
   myOledShow.print(pageString[0]);

   myOledShow.clear(0, 16, 54, 23);
   myOledShow.setCursor(0, 2);
   myOledShow.print("drum kit:");

   myOledShow.clear(65, 16, 120, 23);
   myOledShow.setCursor(65, 2);
   myOledShow.print("save kit:");

   myOledShow.clear(0, 24, 54, 31);
   myOledShow.setCursor(0, 3);
   myOledShow.print("timbre:");

   myOledShow.clear(0, 32, 54, 39);
   myOledShow.setCursor(0, 4);
   myOledShow.print("volume:");

   myOledShow.clear(0, 40, 54, 47);
   myOledShow.setCursor(0, 5);
   myOledShow.print("phase:");

   myOledShow.clear(0, 48, 54, 55);
   myOledShow.setCursor(0, 6);
   myOledShow.print("force:");

   myOledShow.clear(0, 56, 34, 63);
   myOledShow.setCursor(0, 7);
   myOledShow.print("trim:");

   myOledShow.clear(54, 16, 64, 23);
   myOledShow.setCursor(54, 2);
   myOledShow.invertText(true);
   myOledShow.print(myMIDISerial.getDrumKitGroup());
```

```
        myOledShow.invertText(false);
        myOledShow.clear(120, 16, 127, 23);
        myOledShow.setCursor(120, 2);
        myOledShow.print(myMIDISerial.getDrumKitParamSaveUserGroup());
        myOledShow.clear(54, 24, 90, 31);
        myOledShow.setCursor(54, 3);
        myOledShow.print(myMIDISerial.getDrumKitTimbre());
        myOledShow.clear(54, 32, 78, 39);
        myOledShow.setCursor(54, 4);
        myOledShow.print(myMIDISerial.getDrumKitVolume());
        myOledShow.clear(54, 40, 74, 47);
        myOledShow.setCursor(54, 5);
        myOledShow.print(phaseString[myMIDISerial.getDrumKitPhase()]);
        myOledShow.clear(54, 48, 74, 55);
        myOledShow.setCursor(54, 6);
        myOledShow.print(myMIDISerial.getDrumKitForce());
        myOledShow.clear(34, 56, 74, 63);
        myOledShow.setCursor(34, 7);
        myOledShow.print(trimchannelToDrum[myMIDISerial.getDrumKitTrimChannel()]);
    }
}
else//錄音界面
{
    if(!lastPageRecordFlag)
    {
        lastPageRecordFlag = pageRecordFlag;
        firstFrush = 0;
    }
    if(0 == firstFrush)
    {
        firstFrush = 1;
        myOledShow.clear(0, 0, 128, 8);
        myOledShow.setCursor(40, 0);
        myOledShow.print(pageString[4]);

        myOledShow.clear(0, 16, 127, 23);
        myOledShow.setCursor(0, 2);
        myOledShow.print("rec num:");

        myOledShow.clear(0, 24, 54, 31);
        myOledShow.setCursor(0, 3);
        myOledShow.print("rec sta:");

        myOledShow.clear(0, 32, 78, 39);

        myOledShow.clear(0, 40, 74, 47);

        myOledShow.clear(0, 48, 74, 55);

        myOledShow.clear(0, 56, 74, 63);

    }
    switch(type)
    {
        case SET_RECORD_NUM:
```

```
            myOledShow.clear(54, 16, 82, 23);
            myOledShow.setCursor(54, 2);
            myOledShow.invertText(true);
            myOledShow.print(myMIDISerial.getRecordSaveNum());
            myOledShow.invertText(false);
            myOledShow.clear(54, 24, 90, 31);
            myOledShow.setCursor(54, 3);
            myOledShow.print(recStaString[myMIDISerial.getRecState()]);
        break;
        case SET_RECORD_START:
        case SET_RECORD_END:
            myOledShow.clear(54, 16, 82, 23);
            myOledShow.setCursor(54, 2);
            myOledShow.print(myMIDISerial.getRecordSaveNum());
            myOledShow.clear(54, 24, 90, 31);
            myOledShow.setCursor(54, 3);
            myOledShow.invertText(true);
            myOledShow.print(recStaString[myMIDISerial.getRecState()]);
            myOledShow.invertText(false);
        break;
        default:
        break;
    }
}

myOledShow.clear(75, 32, 92, 39);
myOledShow.setCursor(75, 4);
myOledShow.print("ch:");

myOledShow.clear(92, 32, 127, 39);
myOledShow.setCursor(92, 4);
myOledShow.print(trigchannelToDrum[myMIDISerial.getTrigChannel()]);

myOledShow.clear(75, 40, 110, 47);
myOledShow.setCursor(75, 5);
myOledShow.print("metro:");

myOledShow.clear(110, 40, 127, 47);
myOledShow.setCursor(110, 5);
myOledShow.print(swString[metroSWFlag]);

myOledShow.clear(75, 48, 110, 55);
myOledShow.setCursor(75, 6);
myOledShow.print("track:");

myOledShow.clear(110, 48, 127, 55);
myOledShow.setCursor(110, 6);
myOledShow.print(swString[trackSWFlag]);

myOledShow.clear(81, 56, 110, 63);
myOledShow.setCursor(81, 7);
myOledShow.print("play:");

myOledShow.clear(110, 56, 127, 63);
myOledShow.setCursor(110, 7);
myOledShow.print(swString[playFlag]);
```

```
    myOledShow.update();
    if(saveFlag && pageParam.pageNum != 4)
    {
        saveFlag = 0;
        myOledShow.clear(0, 0, 39, 8);
        myOledShow.setCursor(0, 0);
        myOledShow.print("save..");
        myOledShow.update();
        delay(500);
        myOledShow.clear(0, 0, 39, 8);
        myOledShow.update();
    }
}

/*************************************************************************
描述:   OLED 顯示
參數:
    輸入:
    輸出:
返回值:
其他:
*************************************************************************/
void oledShow(void)
{
    if(resetFlag)
    {
        oledShowParam(pageParam.pageNum, SET_VOICE_SYS_REFACTORY_TRUE);
    }
    else
    {
        if(!pageRecordFlag)//other page
        {
            oledShowParam(pageParam.pageNum, (DATATYPE_ENUM)(pageSubSetParam[pageParam.pageNum]-
>setParamFuncNum | (((pageParam.pageNum+1) << 4) & 0xf0)));
        }
        else//page record
        {
            if(!recordSaveNumFlag)
            {
                if(1 == myMIDISerial.getRecState())
                {
                    oledShowParam(pageParam.pageNum, SET_RECORD_START);
                }
                else
                {
                    oledShowParam(pageParam.pageNum, SET_RECORD_END);
                }
            }
            else
            {
                oledShowParam(pageParam.pageNum, SET_RECORD_NUM);
            }
        }
    }
```

```
}
```

b.初始化對象

```
void setup() {
  // put your setup code here, to run once:
 /*OLED, 按鍵, BM72D5221_1 模塊初始化*/
  oledBegin();
  drumBtnInit();
  if(FALSE == myMIDISerial.begin())
  {
    myOledShow.clear(0, 8, 128, 128);
    myOledShow.setCursor(20, 2);
    myOledShow.setScale(2);
    myOledShow.print("Initial");
    myOledShow.setCursor(20, 5);
    myOledShow.setScale(2);
    myOledShow.print("failure!");
    myOledShow.update();
    while(1);//當初始化失敗一直停留在這裡，不再往下執行
  }
}
```

c.循環獲取BM72D5221_1模塊數據&掃描處理功能按鍵&刷新OLED顯示

```
void loop() {
  // put your main code here, to run repeatedly:

  if(myMIDISerial.scanSlaveUpdateData())//循環獲取模塊更新的數據
  {
    oledShowFlag = 1;
  }
  if(myMIDISerial.getPlayState())//迴圈獲取播放狀態以刷新 OLED 顯示
  {
    playFlag = 1;
  }
  else
  {
    playFlag = 0;
  }
  if(theLastPlayState != playFlag)
  {
    theLastPlayState = playFlag;
    oledShowFlag = 1;//播放狀態改變，刷新 OLED
  }
  if(oledShowFlag)//如果標誌位元為 1 刷新 OLED 顯示
  {
    oledShowFlag = 0;
    oledShow();
  }


  myDrumButton.btnScan();//輪詢按鍵狀態
```

```
  //----------------處理按鍵----------------------------
   switch(myDrumButton.btnGetEvent())//判斷並處理按鍵
   {
      case KEY_PAGE_SET://KEY1(短按): 切換界面
         pageSubSetParam[pageParam.pageNum++]->setParamFuncNum = 0;
         if(pageParam.pageNumMax == pageParam.pageNum)//4 個界面：0~3【page kit：0；page metro：1；
page sys：2；page song：3】
         {
            pageParam.pageNum = 0;//page kit
         }
         if(myMIDISerial.pageSwitch((PAGE_TYPE_ENUM)pageParam.pageNum))//其他界面（除了 page
record）
         {
            pageRecordFlag = 0;
            oledShowFlag = 1;
         }
         break;
      case KEY_SYS_RESET://KEY1(長按 3 秒):復位 BM72D5221_1 模塊，僅在 page sys 界面下有效
if(!pageRecordFlag)
         {
            if(myMIDISerial.resetToFactoryStateTrue())
            {
               oledShowFlag = 1;
               resetFlag = 1;
            }
         }
         break;
      case KEY_PARAM_FUNC_SET://KEY2(短按):對應界面（除了 page record）下的參數設置功能切換
         if(!pageRecordFlag)
         {
            if(pageSubSetParam[pageParam.pageNum]->setParamFuncNumMax ==
++pageSubSetParam[pageParam.pageNum]->setParamFuncNum)
            {
               pageSubSetParam[pageParam.pageNum]->setParamFuncNum = 0;
            }
            FUNC_SELECT(pageParam.pageNum, pageSubSetParam[pageParam.pageNum]->setParamFuncNum);
            oledShowFlag = 1;
         }
         break;
      case KEY_INC_PARAM://KEY3(短按):功能參數++
      case KEY_INC_PARAM_CONTINUE://KEY3(長按):功能參數持續++
         if(pageRecordFlag)//page record
         {
            if(pageRecordSetParam.funcList[0].argMax > pageRecordSetParam.funcList[0].getCurrentVal(4, 0,
myMIDISerial.getDrumKitGroup(), myMIDISerial.getTrigChannel()))
            {
               recordSaveNum = pageRecordSetParam.funcList[0].getCurrentVal(4, 0, 0, 0) + 1;
               (myMIDISerial.*pageRecordSetParam.funcList[0].funcSet)(recordSaveNum);
               recordSaveNumFlag = 1;
            }
         }
         else//其他界面
         {
            INC_PARAM(pageParam.pageNum, pageSubSetParam[pageParam.pageNum]->setParamFuncNum);
```

```
        }
        oledShowFlag = 1;
        break;
    case KEY_DEC_PARAM://KEY4(短按):功能參數--
    case KEY_DEC_PARAM_CONTINUE://KEY4(長按):功能參數持續--
        if(pageRecordFlag)//page record
        {
            if(pageRecordSetParam.funcList[0].argMin < pageRecordSetParam.funcList[0].getCurrentVal(4, 0,
myMIDISerial.getDrumKitGroup(), myMIDISerial.getTrigChannel()))
            {
                recordSaveNum = pageRecordSetParam.funcList[0].getCurrentVal(4, 0, 0, 0) - 1;
                (myMIDISerial.*pageRecordSetParam.funcList[0].funcSet)(recordSaveNum);
                recordSaveNumFlag = 1;
            }
        }
        else//其他界面
        {
            DEC_PARAM(pageParam.pageNum, pageSubSetParam[pageParam.pageNum]->setParamFuncNum);
        }
        oledShowFlag = 1;
        break;
    case KEY_SAVE_PARAM://KEY5(短按):保存設置的參數
        if(!pageRecordFlag)//page record 界面不顯示“save..”
        {
            SAVE_PARAM(pageParam.pageNum, pageSubSetParam[pageParam.pageNum]->setParamFuncNum);
            saveFlag = 1;
            oledShowFlag = 1;
        }
        break;
    case KEY_NO_SAVE_RECORD://KEY5(長按 3 秒):停止且不保存錄音資料
        if(pageRecordFlag)//page record
        {
            if(myMIDISerial.setRecordNosave())
            {
                oledShowFlag = 1;
            }
        }
        break;
    case KEY_RECORD_SW://KEY6(短按):開始錄音/停止並保存錄音資料
        if(!pageRecordFlag)
        {
            if(myMIDISerial.pageSwitch(PAGE_RECORD))
            {
                pageRecordFlag = 1;
                recordSaveNumFlag = 0;
                oledShowFlag = 1;
            }
        }
        else
        {
            if(0 == myMIDISerial.getRecState())
            {
                myMIDISerial.setRecordStart();
            }
            else if(1 == myMIDISerial.getRecState())
```

```
            {
                myMIDISerial.setRecordEnd();
            }
            recordSaveNumFlag = 0;
            oledShowFlag = 1;
        }
        break;
    case KEY_METRO_SW://KEY7(短按):打開/關閉節拍器
        if(myMIDISerial.setMetroOpenClose())
        {
            if(0 == metroSWFlag)
            {
                metroSWFlag = 1;//打開
            }
            else
            {
                metroSWFlag = 0;
            }
            oledShowFlag = 1;
        }
        break;
    case KEY_TRACK_SW://KEY7(長按 3 秒):打開/關閉背景軌道音樂，僅在歌曲播放的情況下才顯現效果
        if(myMIDISerial.setTrackSW())
        {
            if(0 == trackSWFlag)
            {
                trackSWFlag = 1;//打開
            }
            else
            {
                trackSWFlag = 0;
            }
            oledShowFlag = 1;
        }
        break;
    case KEY_PLAY_STOP_SW://KEY8(短按):播放/停止播放歌曲
        if(pageRecordFlag)//若在 page record 界面下，播放的是錄音資料
        {
            if(myMIDISerial.setRecordPlayStop())
            {
                oledShowFlag = 1;
            }
        }
        else
        {
            if(myMIDISerial.setSongPlayStop())
            {
                oledShowFlag = 1;
            }
        }
        break;
    default:
        break;
  }
}
```

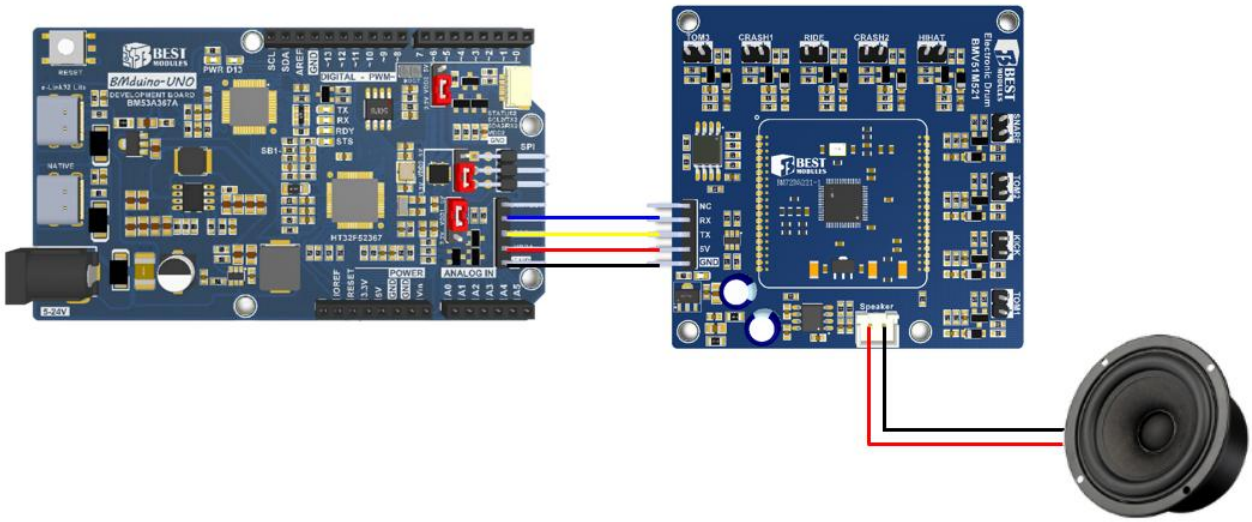
3．OLED顯示界面如下（5個界面）：

page kit界面	page metro界面	page sys界面
		

page song界面	page record界面
	

備註：建議搭配全頻喇叭使用。

範例 2：midiDemoPlay



實物連接示意圖

範例2實現功能：通過UART通訊控制轉接板播放MIDI音符。

1．範例打開：文件→示例→Lib選擇(BM72D5221-1)→選擇範例(midiDemoPlay)

2．示例說明：

a. 構建&初始化對象。

```
#include "BM72D5221-1.h"

BM72D5221_1 myMidiPlay(&Serial1); //使用硬體串口 Serial1 進行通信

void setup()
{
  myMidiPlay.begin();//模塊初始化
}
```

b.迴圈發送MIDI音符播放指令。

```
void loop()
{
  delay(1000);
  myMidiPlay.setChannelVolume(127, 1); // 通道音量
  for (int instrument = 0; instrument < 3; instrument++)
  {
    myMidiPlay.setTone(instrument, 1); // 設置樂器音色

    // 播放從 F1# (30) 到 F5# (78)的音符
    for (int note = 27; note < 78; note++)
    {
      // 在通道 1 播放音符，音高 127
      myMidiPlay.setNoteOn(note, 127, 1);
      delay(50);
```

```
        // 關閉正在播放的音符
        myMidiPlay.setNoteOff(note, 127, 1);
        delay(50);
      }

      // 每個樂器音色切換之前延遲 100ms，以便聽出不同的音色
      delay(100);
    }
}
```

3．喇叭循環播放MIDI音符。

附錄一：標準 MIDI 音色表

音色表

編號	名稱	編號	名稱
0	Acoustic Grand Piano	34	Electric Bass(pick)
1	Bright Acoustic Piano	35	Fretless Bass
2	Electric Grand Piano	36	Slap Bass 1
3	Honky-tonk Piano	37	Slap Bass 2
4	Electric Piano 1	38	Synth Bass 1
5	Electric Piano 2	39	Synth Bass 2
6	Harpsichord	40	Violin
7	Clavichord	41	Viola
8	Celesta	42	Cello
9	Glockenspiel	43	Contrabass
10	Music box	44	Tremolo Strings
11	Vibraphone	45	Pizzicato Strings
12	Marimba	46	Orchestral Harp
13	Xylophone	47	Timpani
14	Tubular Bell	48	String Ensemble 1
15	Dulcimer	49	String Ensemble 2
16	Hammond Organ	50	Synth Strings 1
17	Percussive Organ	51	Synth Strings 2
18	Rock Organ	52	Voice Aahs
19	Church organ	53	Voice Oohs
20	Reed organ	54	Synth Voice
21	Accordion	55	Orchestra Hit
22	Harmonica	56	Trumpet
23	Tango Accordion	57	Trombone
24	Acoustic Guitar(nylon)	58	Tuba
25	Acoustic Guitar(steel)	59	Muted Trumpet
26	Electric Guitar(jazz)	60	French horn
27	Electric Guitar(clean)	61	Brass Section
28	Electric Guitar(muted)	62	Synth Brass 1
29	Overdriven Guitar	63	Synth Brass 2
30	Distortion Guitar	64	Soprano Sax
31	Guitar harmonics	65	Alto Sax
32	Acoustic Bass	66	Tenor Sax
33	Electric Bass(finger)	67	Baritone Sax

音色表（接續上一頁）

編號	名稱	編號	名稱
68	Oboe	98	FX 3(crystal)
69	English Horn	99	FX 4(atmosphere)
70	Bassoon	100	FX 5(brightness)
71	Clarinet	101	FX 6(goblins)
72	Piccolo	102	FX 7(echoes)
73	Flute	103	FX 8(sci-fi)
74	Recorder	104	Sitar
75	Pan Flute	105	Banjo
76	Blown Bottle	106	Shamisen
77	Shakuhachi	107	Koto
78	Whistle	108	Kalimba
79	Ocarina	109	Bagpipe
80	Lead 1(square)	110	Fiddle
81	Lead 2(sawtooth)	111	Shanai
82	Lead 3(calliope)	112	Tinkle Bell
83	Lead 4(chiff)	113	Agogo
84	Lead 5(charang)	114	Steel Drums
85	Lead 6(voice)	115	Woodblock
86	Lead 7(fifths)	116	Taiko Drum
87	Lead 8(bass + lead)	117	Melodic Tom
88	Pad 1(new age)	118	Synth Drum
89	Pad 2(warm)	119	Reverse Cymbal
90	Pad 3(polysynth)	120	Guitar Fret Noise
91	Pad 4(choir)	121	Breath Noise
92	Pad 5(bowed)	122	Seashore
93	Pad 6(metallic)	123	Bird Tweet
94	Pad 7(halo)	124	Telephone Ring
95	Pad 8(sweep)	125	Helicopter
96	FX 1(rain)	126	Applause
97	FX 2(soundtrack)	127	Gunshot

打擊樂

編號	名稱	編號	名稱
24	Seq Click H	55	Splash Cymbal
25	Brush Tap	56	Cowbell
26	Brush Swirl	57	Crash Cymbal 2
27	Brush Slap	58	Vibraslap
28	Brush Tap Swirl	59	Ride Cymbal 2
29	Snare Roll	60	Hi Bongo
30	Castanet	61	Low Bongo
31	Snare H Soft	62	Mute Hi Conga
32	Sticks	63	Open Hi Conga
33	Bass Drum Soft	64	Low Conga
34	Open Rim Shot	65	High Timbale
35	Acoustic Bass Drum	66	Low Timbale
36	Bass Drum 1	67	High Agogo
37	Side Stick	68	Low Agogo
38	Acoustic Snare	69	Cabasa
39	Hand Clap	70	Maracas
40	Electric Snare	71	Short Whistle
41	Low Floor Tom	72	Long Whistle
42	Closed Hi-Hat	73	Short Guiro
43	High Floor Tom	74	Long Guiro
44	Pedal Hi-Hat	75	Claves
45	Low Tom	76	Hi Wood Block
46	Open Hi-Hat	77	Low Wood Block
47	Low-Mid Tom	78	Mute Cuica
48	Hi-Mid Tom	79	Open Cuica
49	Crash Cymbal 1	80	Mute Triangle
50	High Tom	81	Open Triangle
51	Ride Cymbal 1	82	Shaker
52	Chinese Cymbal	83	Jingle Bell
53	Ride Bell	84	Bell Tree
54	Tambourine		