

### 2.4G GFSK 雙向透傳模組

# BMC56M001 Arduino Library V1.0.2 說明

版本: V1.11 日期: 2023-12-11

www.bestmodulescorp.com



### 目錄

簡介	3
Arduino Lib 函式	3
Arduino Lib 下載及安裝	8
Arduino 範例	9
範例 1: Peer	9
範例 2:Node	
範例 3 · Concentrator	17



## 📗 簡介

BMC56M001 是倍創推出的一款 2.4G GFSK 雙向透傳模組 · 使用 UART 通訊方式。本文檔對 BMC56M001 的 Arduino Lib 函式、Arduino Lib 安裝方式進行說明; 範例演示了模組搭配形成 Peer 網路拓撲或 Star 網路拓撲實現配對以及無線通訊 等功能。

## Arduino Lib 函式

Arduino	Lib 名稱:H	BMC56M001 Lib 版本: V1.0.2												
		構造函式 & 初始化												
	BMC56M00	1(HardwareSerial *theSerial=&Serial)												
	描述	構造函式,使用 HW Serial 介面												
1	參數	*theSerial:選擇 HW Serial 介面 (預設 Serial 介面)												
	返回值													
	備註													
	BMC56M00	01(uint8_t rxPin, uint8_t txPin)												
	描述	構造函式,使用 SW Serial 介面												
2	參數	rxPin:RX 腳位,連接 BMC56M001 的 TX 腳位 txPin:TX 腳位,連接 BMC56M001 的 RX 腳位												
Ē	返回值	—												
	備註													
	void begin(uint8_t baud=BR_9600)													
	描述	模組初始化												
3	參數	baud:通訊速率選擇 0x00(BR_9600):9600bps(預設) 0x01(BR_19200):19200bps 0x02(BR 38400):38400bps												
	返回值	void												
	備註	_												
		功能函式												
	bool isPaire	ł()												
	描述	是否配對過												
	參數	_												
	返回值	是否配對過狀態 TRUE:已經配對過 FALSE:沒配對過												
4	備註	<ol> <li>此函數僅支援硬體 V1.01 及以上的版本號 · V1.00 不支援</li> <li>如果模組已經配對過 · 則不需要重新配對。可直接獲取通訊短位址 · 進行通訊。</li> <li>通訊短位址可通過下面方式獲取: Peer 模式:配對成功後 · 使用 getShortAddress 函式進行獲取 Star 模式: Concentrator:依據上次配對順序的短位址 (0x0001~0x0005) Node:配對成功後 · 使用 getShortAddress 函式進行獲取</li> </ol>												



	uint8_t write	ePairPackage()
	描述	發送配對包
5	參數	
	返回值	執行情況 *
	備註	
	uint8_t getP	airStatus()
6	描述	獲取配對狀態
	參數	—
	返回值	配對狀態 0x00:配對中 0x01:配對成功 0x02:配對失敗 0x03:配對超時
	備註	—
	uint8_t write	eRFData(uint32_t shortAddr, uint8_t len, uint8_t data[])
	描述	發送資料封包
7	參數	shortAddr:短位址 len:需要發送資料的長度 data[]:需要發送的資料
	返回值	執行情況*
	備註	發送資料的長度不能超過 32-byte
	bool isInfoA	available()
	描述	判斷是否接收到資料
	參數	—
8	返回值	資料接收情況 false:沒接收到資料 true:接收到資料
	備註	_
	uint8_t read	RFData(uint8_t rxData[], uint8_t &len)
	描述	讀取 RF 封包裡面的資料
9	參數	rxData[]:儲存接收到的資料 &len:儲存接收到資料的長度
	返回值	接收到的資料情況 0x00:不是資料封包 0x01:是資料封包
	備註	rxData[] 的長度建議設為 36-byte
	uint8_t getS	hortAddress()
	描述	獲取通訊短位址
10	參數	—
-	返回值	通訊短位址
	備註	在配對成功後 Peer 角色或 Node 角色可獲取通訊短位址,用於發送資料封包



1														
	uint8_t getR	RSSI()												
	描述     / 獲取富所訊號強度       參數     —													
11	參數	_												
	返回值	當前訊號強度												
	備註	_												
	uint8_t getP	vktRSSI()												
	描述	獲取接收封包訊號強度												
12	參數													
	返回值	接收封包訊號強度												
	備註													
	uint8_t writ	eEEPROM(uint8_t len, uint8_t deviInfo[])												
	描述	向 EEPROM 寫入資料												
12	<b> </b>	len:需要寫入資料的長度												
15	学	deviInfo[]:需要寫入的資料												
	返回值	執行情況 *												
	備註	deviInfo[] 的長度不能超過 32-byte												
	uint8_t readEEPROM(uint8_t deviInfo[], uint8_t &len)													
	描述	讀取 EEPROM 中的資料												
14	參數	deviInfo[]:儲存獲取的資料												
		&len:儲存獲取資料的長度												
	返回值	執行情況 *												
	備註 deviInfo[] 的長度建議設為 32-byte uint8_t getFWVer(uint8_t number[])													
	uint8_t getF	uint8_t getFWVer(uint8_t number[]) 描述   獲取版太號												
	描述	獲取版本號												
15	參數	number[]:儲存接收到的版本號												
	返回值	執行情況 *												
	備註	number[] 的長度建議設為 16-byte												
	uint8_t getS	N(uint8_t id[])												
	描述	獲取序列號												
16	參數	id[]:儲存獲取的序列號												
	返回值	執行情況 *												
	備註	id[] 的長度建議設為 4-byte												
		設定 & 讀取函式												
	uint8_t getE	DeviceRole()												
	描述	獲取設備角色												
	參數	—												
17		設備角色												
	返回值	0x00:Peer 角色												
		0x01:Node 角色												
	/ <u>++</u> ->_	0x02:Concentrator 用巴												
	1角註	—												



	uint8_t getN	fode()
	描述	獲取工作模式
	參數	
18	返回值	工作模式 0x00:深睡眠模式 0x01:睡眠模式 0x02:RX模式 0x03:配對模式
	備註	
	uint8_t getC	ChannelPtn()
	描述	獲取跳頻頻道
	參數	
19	返回值	跳頻頻道 0x00:跳頻組1 0x01:跳頻組2  0x0F:跳頻組16
	備註	—
20	uint8_t getR	FPower()
	描述	獲取發射功率
	參數	
	返回值	發射功率 0x00:-3dBm 0x01:0dBm 0x02:5dBm 0x03:7dBm
	備註	_
	uint8_t getD	ataRate()
	描述	獲取空中通訊速率
	參數	—
21	返回值	空中通訊速率 0x00:125kbps 0x01:250kbps 0x02:500kbps
	備註	—
	uint8_t getH	loppPeriod(uint8_t period[])
	描述	獲取跳頻週期
22	參數	period[]:儲存跳頻週期參數,範圍:0x0002~0xFFFE 跳頻週期 = 8μs* 跳頻週期參數
	返回值	執行情況*
	備註	—



	uint8_t getE	Baud()
	描述	獲取通訊速率
	參數	_
22		通訊速率
23		0x01 : 9600bps
	巡凹徂	0x02 : 19200bps
		0x03 : 38400bps
	備註	
	uint8_t setD	DeviceRole(uint8_t role)
	描述	設定設備角色
		role:設備角色
	盆動	0x00 (Peer): Peer 角色
24	参 <i>製</i>	0x01 (Node_of_Star): Node 角色
24		0x02 (Concentrator_of_Star): Concentrator 角色
	返回值	執行情況*
		Peer 角色的設備可倆倆之間組成 Peer 網路拓撲進行資料交流。
	備註	Concentrator 角色的設備可與多個 (最多 5 個) Node 角色的設備組成 Star
		網路拓撲進行資料交流。
	uint8_t setN	Mode(uint8_t mode)
	描述	設定工作模式
	參數	mode:工作模式
		0x00 (DeepSleep_Mode):深睡眠模式
25		0x01 (Sleep_Mode):睡眠模式
		0x02 (Rx_Mode): RX 模式
		0x03 (Pairing_Mode): 配對模式
		執行情況 *
	備註	—
	uint8_t setC	ChannelPtn(uint8_t channel)
	描述	設定跳頻頻道
		channel:跳頻頻道
		0x00 (ChannelG_1): 跳頻組 1
26	参數	0x01 (ChannelG_2): 跳頻組 2
		Ux0f (ChannelG_16): 跳頻組 16
	返回值	執行情况 *
	備註	—
	uint8_t setR	FPower(uint8_t power)
	描述	設定發射功率
		power:發射功率
		0x00 (N3dBm) : -3dBm
27	参數	0x01 (POdBm) : 0dBm
		0x02 (P5dBm) : 5dBm
		UXU3 (P/dBm) : /dBm
	返回值	執行情況 <sup>*</sup>
	備註	—



	uint8_t setD	vataRate(uint8_t rate)							
	描述	設定空中通訊速率							
20		rate:空中通訊速率							
	<b>矣</b> 動	0x00 (DR125Kbps) : 125kbps							
20	」 ジ 女X	0x01 (DR250Kbps) : 250kbps							
		0x02 (DR500Kbps) : 500kbps							
	返回值	執行情況*							
	備註								
	uint8 t setHoppPeriod(uint8 t period[])								
	描述	設定跳頻週期							
29	參數	period[]:跳頻週期參數,範圍 0x0002~0xFFFE 跳頻週期 = 8μs* 跳頻週期參數							
	返回值	執行情況*							
	備註								
執行情题		· 執行成功;1-指令執行失敗;2-指令不支援;3-格式錯誤;4-資料太長;							
5-配對	失敗:6-酉	2. 】 招時:7 – 發送失敗:8 – 發送成功							

## Arduino Lib 下載及安裝

BMC56M001 Library:可參考下面兩種方法安裝 BMC56M001 的 Arduino Library

#### 方式1:搜索安装

搜索安裝: Arduino IDE  $\rightarrow$  草稿碼  $\rightarrow$  匯入程式庫  $\rightarrow$  管理程式庫 ...  $\rightarrow$  搜索 BMC56M001  $\rightarrow$  安裝

<u></u>								
檔案 糹	編輯	草稿碼	工具	說明				
		Ē.	<u>魚證</u> /編	譯	Ctrl+R			
		_	上傳		Ctrl+U			
		ţ	沷 <b>燒錄器</b>	<b>器上</b> 傳	Ctrl+Shift+U			
		E	進出已編	<b>諞譯的二進位楣</b>	Ctrl+Alt+S			
			顧示草稿	韀資料夾	Ctrl+K			
		ß	匯入程式	t庫			Δ	
		t	加入檔算	Ę		管3	理程式庫…	Ctrl+Shift+I
	L					加。	人.ZIP程式庫	方式1

搜索安裝流程1

∞ 程式庫管理員	×
類型 全部 v 主題 全部 v BMC56M001	
HETCH was been     Second State     Second State	^
版本101] ~ 安装	

搜索安裝流程 2

8



#### 方式 2:添加.ZIP 庫,需提前下載.ZIP 庫

下載方法:打開倍創官方網站 (<u>https://www.bestmodulescorp.com/bmc56m001</u>. <u>html</u>)·下載 "文件" 菜單下的 Arduino 範例程式 (BMC56M001 Library)。 添加 .ZIP 程式庫: Arduino IDE → 草稿碼 → 匯入程式庫 → 加入 .ZIP 程式庫 ...

<u></u>				
檔案 編輯	草稿碼工具說明			
	驗證 / 編譯	Ctrl+R		
	上傳	Ctrl+U		
	以 <b>燒錄器上傳</b>	Ctrl+Shift+U		
	匯出已編譯的二進位檔	Ctrl+Alt+S		
	顯示草稿碼資料夾	Ctrl+K		
	匯入程式庫		Δ	
2.00	加入檔案		管理程式庫	Ctrl+Shift+I
		. Jofts	加入.ZIP程式庫	
		Larch	Arduino程式庫	方式2

## Arduino 範例



範例實現功能:設定模組角色為 Peer 並且與同為 Peer 角色的模組搭配形成 Peer 網路拓撲並且進行配對以及資料交流。

兩個模組可同時運用此範例搭配形成 Peer 網路拓撲並且進行配對以及資料交流。



步驟如下:

(1). Peer 端與 Peer 端配對

兩個模組皆按下 KEY1 (D8),進入配對模式,配對時間為8秒,此時 MRX 指示燈閃爍;任意一個模組,短按 KEY2 (D9)會發送配對請求封包, 配對成功 MRX 指示燈熄滅,並且在序列埠監視視窗上顯示配對成功,若 8 秒內沒有配對成功 MRX 指示燈也會熄滅,用戶可再次重新配對。

(2). 接收與發送資料切換

兩個模組配對成功後,都進入接收模式,用戶可短按任意一模組 KEY2 (D9) 即可發送資料封包。

(3). Peer 端給 Peer 端發送資料

用 戶 可 短 按 任 意 一 模 組 的 KEY2 (D9) 來 發 送 資 料 封 包, 此 時 MTX 指 示 燈 會 閃爍一次指示有資料封包發出。另一模組收到發送的資料時,MRX 指示 燈會閃爍一次指示有接收到資料封包,並在序列埠監視視窗上顯示接收 到的資料。

- 1. 範例打開方式: Arduino IDE → 文件 → 範例 → Lib 選擇 (BMC56M001) → 選 擇範例 (Peer)
- 2. 範例說明:
  - a. 建立對象 & 模組初始化及設定

```
#include "BMC56M001.h"
BMC56M001 BMC56(5,4); // TX 腳位連接開發板 D5 · RX 腳位連接開發板 D4
#define KEY1_Pin (8) // 將 D8 與 GND 通過按鍵連接,此按鍵為 KEY1
                      // 將 D9 與 GND 通過按鍵連接・此按鍵為 KEY2
#define KEY2 Pin (9)
uint8 t Message ShortAddr;
bool Flag Pairing, Flag PairSuccess;
// 發送配對包中的資料部分
uint8 t TXDATA[16] = {0}, RXDATA[32] = {0};
uint8 t DATA, STATUS, len;
/***** 函式宣告 ******/
uint8 t Sys KEY(void);
                           // 獲取按鍵狀態
void RFMessage Process(); // 根據按鍵狀態執行相關動作
void Handle RFPacket Process();// 獲取資料封包 & 獲取配對回復包
void setup()
{
  /**** 按鍵初始化 ****/
 pinMode(KEY1 Pin, INPUT PULLUP);
  pinMode(KEY2 Pin, INPUT PULLUP);
 Serial.begin(115200); // 設定序列埠監視視窗
BMC56.begin(BR_38400); // 初始化模組,設定通訊速率
  BMC56.setDeviceRole(Peer); // 選擇設備角色
```

b. 根據按鍵狀態執行配對、發送配對包、發送資料封包等操作,有接收到資料時獲取資料並在序列埠監視視窗上顯示

```
void loop()
{
RFMessage_Process(); // 掃描按鍵
Handle_RFPacket_Process(); // 掃描是否接收到資料
```



c. 獲取按鍵狀態函式

```
uint8 t Sys KEY(void)
{
 if(!digitalRead(KEY1 Pin))
 {
   delay(60);
   if(!digitalRead(KEY1_Pin))
   {
     return 0x01;
    }
  }
  if(!digitalRead(KEY2 Pin))
  {
   delay(60);
   if(!digitalRead(KEY2_Pin))
   {
     return 0x02;
    }
  }
  return 0x00;
 }
```

d. 根據按鍵狀態執行相關程式

```
void RFMessage Process()
{
 switch(Sys_KEY())
 {
   case 0x01:
   /****KEY1 按鍵有按下 ****/
   BMC56.setMode(Pairing Mode); // 進入配對模式
   Flag Pairing = TRUE;
   Flag PairSuccess = FALSE;
   break;
   case 0x02:
   /****KEY2 按鍵有按下 ****/
                             // 判斷是否正在配對中
   if(Flag_Pairing)
   {
     BMC56.writePairPackage(); // 發送配對請求包
   }
   if(Flag_PairSuccess) // 判斷是否已配對成功
   {
     for(uint8 t temp=0;temp<16;temp++)</pre>
     {
       TXDATA[temp] = DATA++;// 發送的資料從 0x00~0x0f、0x10~0x1f.....、
                           // 0xf0~0xff 循環
     }
     BMC56.writeRFData(Message ShortAddr,16,TXDATA); // 發送資料封包
   }
   break;
 }
```



e. 接收到資料時在序列埠監視視窗上顯示

```
void Handle RFPacket Process()
{
 if(Flag Pairing)
                                   // 判斷是否在配對中
 {
   STATUS = BMC56.getPairStatus(); // 獲取配對情況
   if(STATUS == 1) // pairing success // 判斷是否配對成功
   {
    Flag_Pairing = FALSE;
     Flag PairSuccess = TRUE;
     Message_ShortAddr = BMC56.getShortAddress(); // 獲取短位址
     BMC56.setMode(Rx Mode); // 進入 RX 模式
     Serial.println("Pair Success");// 在序列埠監視視窗上顯示 "Pair
Success"
   }
   if(STATUS == 2) // 判斷是否配對失敗
   {
     Flag Pairing = FALSE;
     Flag PairSuccess = FALSE;
   1
   if(STATUS == 3) // 判斷是否配對超時
   {
     Flag Pairing = FALSE;
     Flag PairSuccess = FALSE;
   }
 }
 if(Flag_PairSuccess) // 判斷是否已經配對成功
 {
   if(BMC56.isInfoAvailable()) // 判斷是否有資料待讀取
   {
     STATUS = BMC56.readRFData(RXDATA,len); // 讀取資料
                                  // 判斷讀取的資料是否為資料封包
     if(STATUS == 1)
     {
       Serial.print("RXDATA[]:"); // 序列埠監視視窗上顯示讀取的資料
       for(uint8 t temp=0;temp<len;temp++)</pre>
       {
        Serial.print(RXDATA[temp],HEX);
        Serial.print(" ");
       }
       Serial.println(" ");
     }
   }
 }
```



3. 打開序列埠監視視窗, 鮑率選擇 115200; 序列埠監視視窗顯示如下

🥶 COM23															×_*		×
																	傳送
Pair Success	3																^
RXDATA[]:0 1	L 2	3 4	45	6	78	9 1	A B	CI	ΟE	F							
RXDATA[]:10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	lF		
RXDATA[]:20	21	22	23	24	25	26	27	28	29	2A	2В	2C	2D	2E	2F		
RXDATA[]:30	31	32	33	34	35	36	37	38	39	ЗA	ЗB	3C	ЗD	ЗE	ЗF		
RXDATA[]:40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F		
RXDATA[]:50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F		
RXDATA[]:60	61	62	63	64	65	66	67	68	69	бA	бB	6C	6D	бE	бF		
RXDATA[]:70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F		
RXDATA[]:80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F		
RXDATA[]:90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F		
RXDATA[]:A0	A1	A2	AЗ	A4	A5	Aб	A7	A8	A9	AA	AB	AC	AD	AE	AF		
RXDATA[]:BO	В1	B2	BЗ	В4	В5	вб	B7	B8	Β9	ΒA	BB	BC	BD	ΒE	BF		
RXDATA[]:CO	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF		
RXDATA[]:D0	D1	D2	DЗ	D4	D5	DG	D7	D8	D9	DA	DB	DC	DD	DE	DF		
RXDATA[]:E0	E1	E2	E3	E4	Ε5	Eб	E7	E8	Ε9	EA	EB	EC	ED	EE	EF		
																	Ų
☑ 自動捲動 □ Show tim	estamp								1	IL (new)	line)	~	115	200 baı	ad ~	Clear	output

### 範例 2:Node



範例實現功能:設定模組角色為 Node 並且與 Concentrator 角色的模組搭配形成 Star 網路拓撲並且進行配對以及資料交流。

此範例需搭配範例 Concentrator 使用。



步驟如下:

(1). Node 端與 Concentrator 端配對

兩個模組皆按下 KEY1 (D8),進入配對模式:配對時間為8秒,此時 MRX 指示燈閃爍; Node 端短按 KEY2 (D9) 會發送配對請求封包,配對 成功 MRX 指示燈熄滅,並且在 Node 端與 Concentrator 端的序列埠監視 視窗上顯示配對成功,若8秒內沒有配對成功 MRX 指示燈也會熄滅,用 戶可再次重新配對。

(2). 接收與發送資料切換

Node 端與 Concentrator 端配對成功後,都進入接收模式,用戶可短按任 意 Node 端 KEY2 (D9) 來發送資料封包給配對好的 Concentrator 端;或短 按 Concentrator 端 KEY2~KEY6 (D9~D13) 來發送資料封包給配對好之不 同 Node。

(3). Node 端給 Concentrator 端發送資料

用戶可短按 Node 端 KEY2 (D9) 來發送資料封包,此時 Node 端 MTX 指示燈會閃爍一次指示有資料封包發出。Concentrator 端接收到資料時 MRX 指示燈會閃爍一次指示有接收到資料封包,並在序列埠監視視窗上 顯示接收到的資料。

(4). Concentrator 端給 Node 端發送資料

用戶可短按 Concentrator 端 KEY2 (D9) 來發送資料封包,此時 Concentrator 端 MTX 指示燈會閃爍一次,指示有資料封包發出。Node 端接收到資料 時 MRX 指示燈會閃爍一次指示有接收到資料封包,並在序列埠監視視 窗上顯示接收到的資料。若有超過一個以上 Node,可按照配對順序短按 Concentrator 端 KEY2~KEY6 (D9~D13) 分別對不同 Node 發送資料。

- 範例打開方式: Arduino IDE → 文件 → 範例 → Lib 選擇 (BMC56M001) → 選 擇 (Star) → 選擇範例 (Node)
- 2. 範例說明:
  - a. 建立對象 & 模組初始化及設定

```
#include "BMC56M001.h"
BMC56M001 BMC56(5,4); // TX 腳位連接開發板 D5 · RX 腳位連接開發板 D4
#define KEY1 Pin (8) // 將 D8 與 GND 通過按鍵連接,此按鍵為 KEY1
#define KEY2 Pin (9)
                      // 將 D9 與 GND 通過按鍵連接・此按鍵為 KEY2
uint8 t Message ShortAddr;
bool Flag_Pairing,Flag_PairSuccess;
uint8 t TXDATA[16] = {0}, RXDATA[32] = {0};
uint8 t DATA, STATUS, len;
/****** 函式宣告 ******/
                            // 獲取按鍵狀態
uint8 t Sys KEY(void);
void RFMessage Process(); // 根據按鍵狀態執行相關動作
void Handle RFPacket Process(); // 獲取資料封包。獲取對方設備 ID
void setup()
{
  /***** 按鍵初始化 *****/
 pinMode(KEY1 Pin, INPUT PULLUP);
 pinMode(KEY2 Pin, INPUT PULLUP);
 Serial.begin(115200); // 設定序列埠監視視窗
BMC56.begin(BR 38400); // 初始仕模組,設定诵
                                // 初始化模組・設定通訊速率
 BMC56.begin(BR 38400);
  BMC56.setDeviceRole(Node of Star); // 選擇設備角色
```



b. 根據按鍵狀態執行配對、發送配對包、發送資料封包等操作,有接收到資料時獲取資料並在序列埠監視視窗上顯示

```
void loop()
{
RFMessage_Process();    // 掃描按鍵
Handle_RFPacket_Process();  // 掃描是否接收到資料
```

c. 獲取按鍵狀態函式

```
uint8 t Sys KEY(void)
{
 if(!digitalRead(KEY1 Pin))
 {
   delay(60);
   if(!digitalRead(KEY1_Pin))
    {
      return 0x01;
    }
  }
  if(!digitalRead(KEY2 Pin))
  {
   delay(60);
   if(!digitalRead(KEY2 Pin))
   {
     return 0x02;
    }
  }
  return 0x00;
```

d. 根據按鍵狀態執行相關程式

```
void RFMessage Process()
{
 switch(Sys_KEY())
 {
   case 0x01:
   /****KEY1 按鍵有按下 ****/
   BMC56.setMode(Pairing_Mode); // 進入配對模式
   Flag Pairing = TRUE;
   Flag PairSuccess = FALSE;
   break;
   case 0x02:
   /****KEY2 按鍵有按下 ****/
                         // 判斷是否正在配對中
   if(Flag Pairing)
   {
     BMC56.writePairPackage(); // 發送配對請求包
    }
                             // 判斷是否已配對成功
   if(Flag PairSuccess)
    {
     for(uint8_t temp=0;temp<16;temp++)</pre>
     {
       TXDATA[temp] = DATA++; // 發送的資料從 0x00~0x0f、0x10~0x1f.....、
                            // 0xf0~0xff 循環
```



```
BMC56.writeRFData(Message_ShortAddr,16,TXDATA); // 發送資料封包
}
break;
}
```

e. 接收到資料時在序列埠監視視窗上顯示

```
void Handle RFPacket Process()
{
 if(Flag Pairing)
                                   // 判斷是否在配對中
 {
   STATUS = BMC56.getPairStatus(); // 獲取配對情況
   if(STATUS == 1) // pairing success // 判斷是否配對成功
   {
     Flag Pairing = FALSE;
     Flag PairSuccess = TRUE;
     Message ShortAddr = BMC56.getShortAddress(); // 獲取短位址
     BMC56.setMode(Rx_Mode); // 進入 RX 模式
     Serial.println("Pair Success");// 在序列埠監視視窗上顯示 "Pair
Success"
   }
                               // 判斷是否配對失敗
   if(STATUS == 2)
   {
     Flag Pairing = FALSE;
     Flag_PairSuccess = FALSE;
   }
                              // 判斷是否配對超時
   if(STATUS == 3)
   {
     Flag Pairing = FALSE;
     Flag PairSuccess = FALSE;
   }
 if(Flag PairSuccess)
                     // 判斷是否已經配對成功
 {
   if(BMC56.isInfoAvailable()) // 判斷是否有資料待讀取
   {
     STATUS = BMC56.readRFData(RXDATA,len);
                                            // 讀取資料
                    // 判斷讀取的資料是否為資料封包
     if(STATUS == 1)
     {
       Serial.print("RXDATA[]:"); // 序列埠監視視窗上顯示讀取的資料
       for(uint8 t temp=0;temp<len;temp++)</pre>
       {
        Serial.print(RXDATA[temp],HEX);
        Serial.print(" ");
       }
       Serial.println(" ");
     }
   }
 }
```



3. 打開序列埠監視視窗, 鮑率選擇 115200; 序列埠監視視窗顯示如下

· · · · · · · · · · · · · · · · · · ·	
2491	
Pair Success	^
RXDATA[]:0 1 2 3 4 5 6 7 8 9 A B C D E F	
RXDATA[]:10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F	
RXDATA[]:20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F	
RXDATA[]:30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F	
RXDATA[]:40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F	
RXDATA[]:50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F	
RXDATA[]:60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F	
RXDATA[]:70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F	
RXDATA[]:80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F	
RXDATA[]:90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F	
RXDATA[]:AO A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF	
RXDATA[]:B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF	
RXDATA[]:C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF	
RXDATA[]:D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF	
RXDATA[]:E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF	
	~

### 範例 3: Concentrator



範例實現功能:設定模組角色為 Concentrator 並且與 Node 角色的模組搭配形成 Star 網路拓撲並且進行配對以及資料交流。

此範例需搭配範例 Node 使用。



步驟如下:

(1). Node 端與 Concentrator 端配對

兩個模組皆按下 KEY1 (D8),進入配對模式:配對時間為8秒,此時 MRX 指示燈閃爍; Node 端短按 KEY2 (D9) 會發送配對請求封包,配對 成功 MRX 指示燈熄滅,並且在 Node 端與 Concentrator 端的序列埠監視 視窗上顯示配對成功,若8秒內沒有配對成功 MRX 指示燈也會熄滅,用 戶可再次重新配對。

(2). 接收與發送資料切換

Node 端與 Concentrator 端配對成功後,都進入接收模式,用戶可短按任 意 Node 端 KEY2 (D9) 來發送資料封包給配對好的 Concentrator 端;或短 按 Concentrator 端 KEY2~KEY6 (D9~D13) 來發送資料封包給配對好之不 同 Node。

(3). Node 端給 Concentrator 端發送資料

用戶可短按 Node 端 KEY2 (D9) 來發送資料封包,此時 Node 端 MTX 指示燈會閃爍一次指示有資料封包發出。Concentrator 端接收到資料時 MRX 指示燈會閃爍一次指示有接收到資料封包,並在序列埠監視視窗上 顯示接收到的資料。

(4). Concentrator 端給 Node 端發送資料

用戶可短按 Concentrator 端 KEY2 (D9) 來發送資料封包,此時 Concentrator 端 MTX 指示燈會閃爍一次,指示有資料封包發出。Node 端接收到資料 時 MRX 指示燈會閃爍一次指示有接收到資料封包,並在序列埠監視視 窗上顯示接收到的資料。若有超過一個以上 Node,可按照配對順序短按 Concentrator 端 KEY2~KEY6 (D9~D13) 分別對不同 Node 發送資料。

- 1. 範例打開方式: Arduino IDE → 文件 → 範例 → Lib 選擇 (BMC56M001) → 選 擇 (Star) → 選擇範例 (Concentrator)
- 2. 範例說明:
  - a. 建立對象 & 模組初始化及設定

```
#include "BMC56M001.h"
BMC56M001 BMC56(5,4); // TX 腳位連接開發板 D5 · RX 腳位連接開發板 D4

        #define
        KEY1_Pin (8)
        // 將 D8 與 GND 通過按鍵連接,此按鍵為 KEY1

        #define
        KEY2_Pin (9)
        // 將 D9 與 GND 通過按鍵連接,此按鍵為 KEY2

#define KEY3 Pin (10) // 將 D10 與 GND 通過按鍵連接,此按鍵為 KEY3
#define KEY4 Pin (11) // 將 D11 與 GND 通過按鍵連接,此按鍵為 KEY4
#define KEY5 Pin (12) // 將 D12 與 GND 通過按鍵連接,此按鍵為 KEY5
#define KEY6_Pin (13) // 將 D13 與 GND 通過按鍵連接,此按鍵為 KEY6
bool Flag_Pairing,Flag_PairSuccess;
uint8 t TXDATA[16] = {0},RXDATA[32] = {0};
uint8 t DATA, STATUS, len;
/***** 函式宣告 ******/
                                // 獲取按鍵狀態
uint8 t Sys KEY(void);
void RFMessage Process(); // 根據按鍵狀態執行相關動作
void Handle RFPacket Process(); // 獲取資料封包 & 獲取對方設備 ID
void setup()
{
  /***** 按鍵初始化 *****/
  pinMode(KEY1 Pin, INPUT PULLUP);
  pinMode(KEY2_Pin, INPUT_PULLUP);
  pinMode(KEY3 Pin, INPUT_PULLUP);
 pinMode (KEY4 Pin, INPUT PULLUP);
```



```
pinMode(KEY5_Pin, INPUT_PULLUP);
pinMode(KEY6_Pin, INPUT_PULLUP);
Serial.begin(115200); // 設定序列埠監視視窗
BMC56.begin(BR_38400); // 初始化模組、設定通訊速率
BMC56.setDeviceRole(Concentrator_of_Star); // 選擇設備角色
```

b. 根據按鍵狀態執行配對、發送配對包、發送資料封包等操作, 有接收到資料時獲取資料並在序列埠監視視窗上顯示

c. 獲取按鍵狀態函式

```
uint8 t Sys KEY(void)
{
  if(!digitalRead(KEY1 Pin))
  {
    delay(60);
   if(!digitalRead(KEY1 Pin))
    {return 0x01;}
  }
  if(!digitalRead(KEY2_Pin))
  {
    delay(60);
    if(!digitalRead(KEY2_Pin))
    {return 0x02;}
  if(!digitalRead(KEY3 Pin))
  {
   delay(60);
   if(!digitalRead(KEY3 Pin))
   {return 0x03;}
  if(!digitalRead(KEY4 Pin))
    delay(60);
   if(!digitalRead(KEY4 Pin))
    {return 0x04;}
  }
  if(!digitalRead(KEY5 Pin))
  {
   delay(60);
   if(!digitalRead(KEY5 Pin))
    {return 0x05;}
  }
  if(!digitalRead(KEY6 Pin))
  {
    delay(60);
    if(!digitalRead(KEY6 Pin))
    {return 0x06;}
  }
  return 0x00;
 }
```



d. 根據按鍵狀態執行相關程式

```
void RFMessage Process()
{
 switch(Sys KEY())
 {
   case 0x01:
   /****KEY1 按鍵有按下 ****/
   if(BMC56.setMode(Pairing Mode)) // 進入配對模式
   {
     Flag Pairing = TRUE;
     Flag PairSuccess = FALSE;
   }
   break;
   case 0x02:
   /****KEY2 按鍵有按下 ****/
                         // 判斷是否已配對成功
   if(Flag PairSuccess)
    {
     for(uint8 t temp=0;temp<16;temp++)</pre>
     {
       TXDATA[temp] = DATA++; // 發送的資料從 0x00~0x0f、0x10~0x1f.....、
                              // 0xf0~0xff 循環
     }
     BMC56.writeRFData(Nodel ShortAddr, 16, TXDATA);// 向 Nodel 發送資料封
包
    }
   break;
   case 0x03:
    /****KEY3 按鍵有按下 ****/
   if(Flag PairSuccess)
                          // 判斷是否已配對成功
   {
      for(uint8_t temp=0;temp<16;temp++)</pre>
     {
       TXDATA[temp] = DATA++; // 發送的資料從 0x00~0x0f、0x10~0x1f.....、
                              // 0xf0~0xff 循環
     }
     BMC56.writeRFData(Node2_ShortAddr,16,TXDATA);// 向 Node2 發送資料封
包
    }
   break;
   case 0x04:
    /****KEY4 按鍵有按下 ****/
   if(Flag_PairSuccess)
                             // 判斷是否已配對成功
     for(uint8 t temp=0;temp<16;temp++)</pre>
     {
       TXDATA[temp] = DATA++; // 發送的資料從 0x00~0x0f、0x10~0x1f.....、
                              // 0xf0~0xff 循環
      }
     BMC56.writeRFData(Node3 ShortAddr, 16, TXDATA);// 向 Node3 發送資料封
包
    }
   break;
   case 0x05:
```



```
/****KEY5 按鍵有按下 ****/
                                // 判斷是否已配對成功
       if(Flag PairSuccess)
       {
         for(uint8_t temp=0;temp<16;temp++)</pre>
         {
          TXDATA[temp] = DATA++; // 發送的資料從 0x00~0x0f、0x10~0x1f.....、
                                // 0xf0~0xff 循環
         }
         BMC56.writeRFData(Node4 ShortAddr,16,TXDATA);// 向 Node4 發送資料封
   包
       }
       break;
       case 0x06:
       /****KEY6 按鍵有按下 ****/
       if(Flag_PairSuccess)
                                // 判斷是否已配對成功
       {
         for(uint8 t temp=0;temp<16;temp++)</pre>
         {
          TXDATA[temp] = DATA++; // 發送的資料從 0x00~0x0f、0x10~0x1f.....、
                                // 0xf0~0xff 循環
         BMC56.writeRFData(Node5 ShortAddr,16,TXDATA);// 向 Node5 發送資料封
   包
       break;
     }
e. 接收到資料時在序列埠監視視窗上顯示
   void Handle RFPacket Process()
   {
                                  // 判斷是否在配對中
     if(Flag Pairing)
     {
       STATUS = BMC56.getPairStatus(); // 獲取配對情況
```

```
if(STATUS == 1)
                                 // 判斷是否配對成功
   {
     Flag Pairing = FALSE;
     Flag PairSuccess = TRUE;
     BMC56.setMode(Rx_Mode); //進入 RX 模式
     Serial.println("Pair Success");// 在序列埠監視視窗上顯示 "Pair
Success"
   }
                              // 判斷是否配對失敗
   if(STATUS == 2)
   {
     Flag_Pairing = FALSE;
     Flag_PairSuccess = FALSE;
   }
                                // 判斷是否配對超時
   if(STATUS == 3)
   {
     Flag Pairing = FALSE;
     Flag_PairSuccess = FALSE;
   }
```



```
// 判斷是否已經配對成功
if(Flag PairSuccess)
{
 if(BMC56.isInfoAvailable()) // 判斷是否有資料待讀取
 {
   STATUS = BMC56.readRFData(RXDATA,len); // 讀取資料
   if(STATUS == 1)
                               // 判斷讀取的資料是否為資料封包
   {
     Serial.print("RXDATA[]:"); // 序列埠監視視窗上顯示讀取的資料
     for(uint8_t temp=0;temp<len;temp++)</pre>
     {
      Serial.print(RXDATA[temp],HEX);
      Serial.print(" ");
     }
     Serial.println(" ");
   }
 }
}
```

3. 打開序列埠監視視窗, 鮑率選擇 115200; 序列埠監視視窗顯示如下

🥯 COM23																	×
																	傳送
Pair Success	5																^
RXDATA[]:0 1	. 2	3 4	45	6	78	9 1	A B	CI	DE	F							
RXDATA[]:10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	lF		
RXDATA[]:20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F		
RXDATA[]:30	31	32	33	34	35	36	37	38	39	ЗA	ЗB	3C	ЗD	ЗE	ЗF		
RXDATA[]:40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F		
RXDATA[]:50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F		
RXDATA[]:60	61	62	63	64	65	66	67	68	69	бA	бB	6C	бD	бE	бF		
RXDATA[]:70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F		
RXDATA[]:80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F		
RXDATA[]:90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F		
RXDATA[]:A0	A1	A2	AЗ	A4	A5	Aб	A7	A8	A9	AA	AB	AC	AD	AE	AF		
RXDATA[]:BO	В1	B2	BЗ	Β4	В5	Bб	B7	B8	В9	ΒA	BB	BC	BD	BE	BF		
RXDATA[]:CO	C1	C2	C3	C4	C5	Сб	C7	C8	С9	CA	CB	CC	CD	CE	CF		
RXDATA[]:D0	D1	D2	D3	D4	D5	DG	D7	D8	D9	DA	DB	DC	DD	DE	DF		
RXDATA[]:E0	E1	E2	E3	E4	E5	ЕG	E7	E8	E9	EA	EB	EC	ED	EE	EF		
																	~
☑ 自動捲動 □ Show timestamp								NL (newline)			✓ 115200 baud ✓				Clear	output	



Copyright<sup>©</sup> 2023 by BEST MODULES CORP. All Rights Reserved.

本文件出版時倍創已針對所載資訊為合理注意,但不保證資訊準確無誤。文中提到的資訊僅是提供 作為參考,且可能被更新取代。倍創不擔保任何明示、默示或法定的,包括但不限於適合商品化、 令人滿意的品質、規格、特性、功能與特定用途、不侵害第三人權利等保證責任。倍創就文中提到 的資訊及該資訊之應用,不承擔任何法律責任。此外,倍創並不推薦將倍創的產品使用在會因故障 或其他原因而可能會對人身安全造成危害的地方。倍創特此聲明,不授權將產品使用於救生、維生 或安全關鍵零組件。在救生/維生或安全應用中使用倍創產品的風險完全由買方承擔,如因該等使 用導致倍創遭受損害、索賠、訴訟或產生費用,買方同意出面進行辯護、賠償並使倍創免受損害。 倍創(及其授權方,如適用)擁有本文件所提供資訊(包括但不限於內容、資料、範例、材料、圖形、 商標)的智慧財產權,且該資訊受著作權法和其他智慧財產權法的保護。倍創在此並未明示或暗示 授予任何智慧財產權。倍創擁有不事先通知而修改本文件所載資訊的權利。如欲取得最新的資訊, 請與我們聯繫。