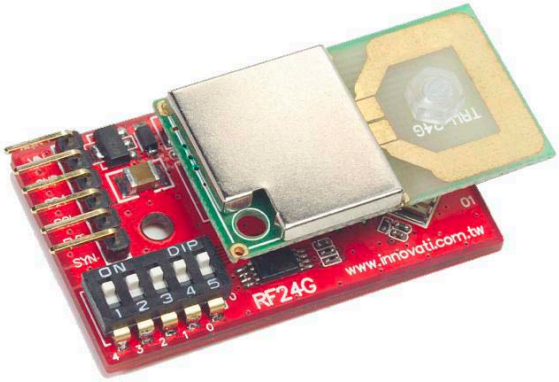
利基 RF24G

無線雙向傳輸模組

版本: V1.0



產品介紹: 利基 RF24G 模組，提供使用者簡單易用的無線雙向傳輸，透過 cmdBUS 與 BASIC Commander 連接，就能以單一指令直接傳送多種格式的資料，透過軟體的動態設定，可以隨時切換收發模式，變更傳輸頻道與識別碼。

應用方向:

- 無線傳輸各種資料。
- 傳送控制訊號，達到無線遙控的功能。
- 同時使用四組 RF24G 模組，完成全雙工通訊。

產品特色:

- 無線傳輸頻率範圍: 2.4 ~ 2.524 GHz。
- 無線傳輸模式: GFSK。
- 能隨時更換模組狀態爲接收或發送模式。
- 可以軟體動態切換 125 個頻道。
- 輸出功率: 0 dBm。
- 資料傳輸速率: 250 Kbps。
- 無線傳輸範圍可達約 280 公尺。
- 內建天線無須再外接其他天線。
- 提供 256 組 ID 碼與 Reg 碼讓使用者可做動態設定識別，可以隨時透過軟體更換。
- 可以將想要傳輸的資料先儲存到內建暫存空間，再用指令一次傳送，最多可以儲存 40 Bytes 的資料。
- 簡易的變數傳輸指令，Byte，Word，與 Dword 都可以透過單一指令傳送。
- 提供字串與陣列傳輸指令，可以一次最多傳送 20 個字元或 20 Bytes 長度的陣列。
- 量測提醒事件，啓動後每當更新量測值，就會產生提醒事件。
- 設定傳送完成提醒事件，可以在資料傳送完成即時。
- 設定接收完成提醒事件，每當有新資料接收完成就會產生提醒事件。
- 四段傳送強度可動態調整: -20 dBm，-10 dBm，-5 dBm，0 dBm。
- 透過指令可以隨時讀回目前設定值做狀態確認，以及是否有未讀取的接收資料。

連接方式: 直接將 ID 開關撥至欲設定的編號，再將 cmdBUS 連接至 BASIC Commander 上對應的腳位，就可透過 BASIC Commander 執行操作。

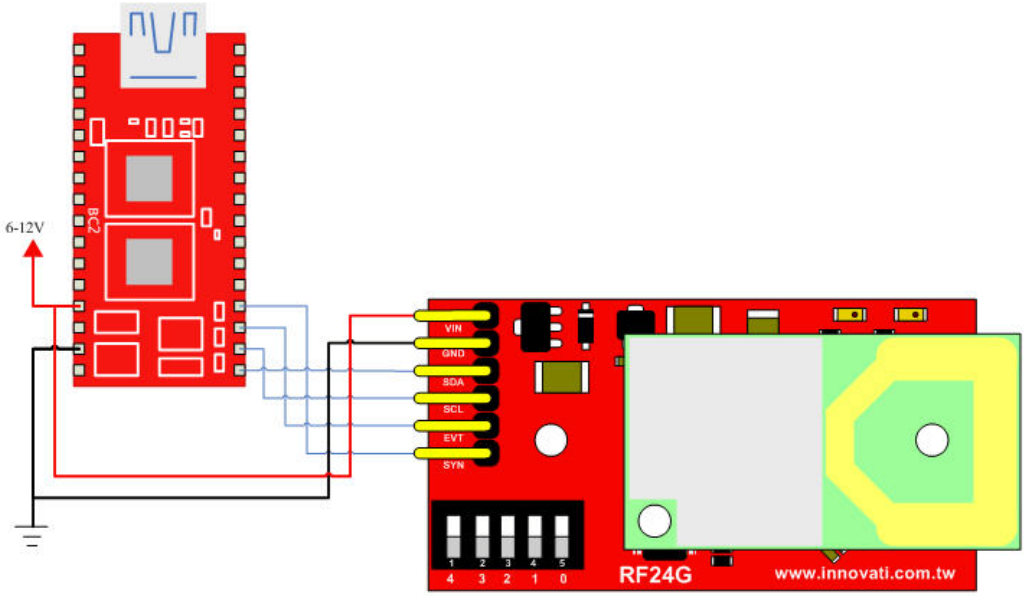


圖 1: 與 BASIC Commander 連接

產品規格:

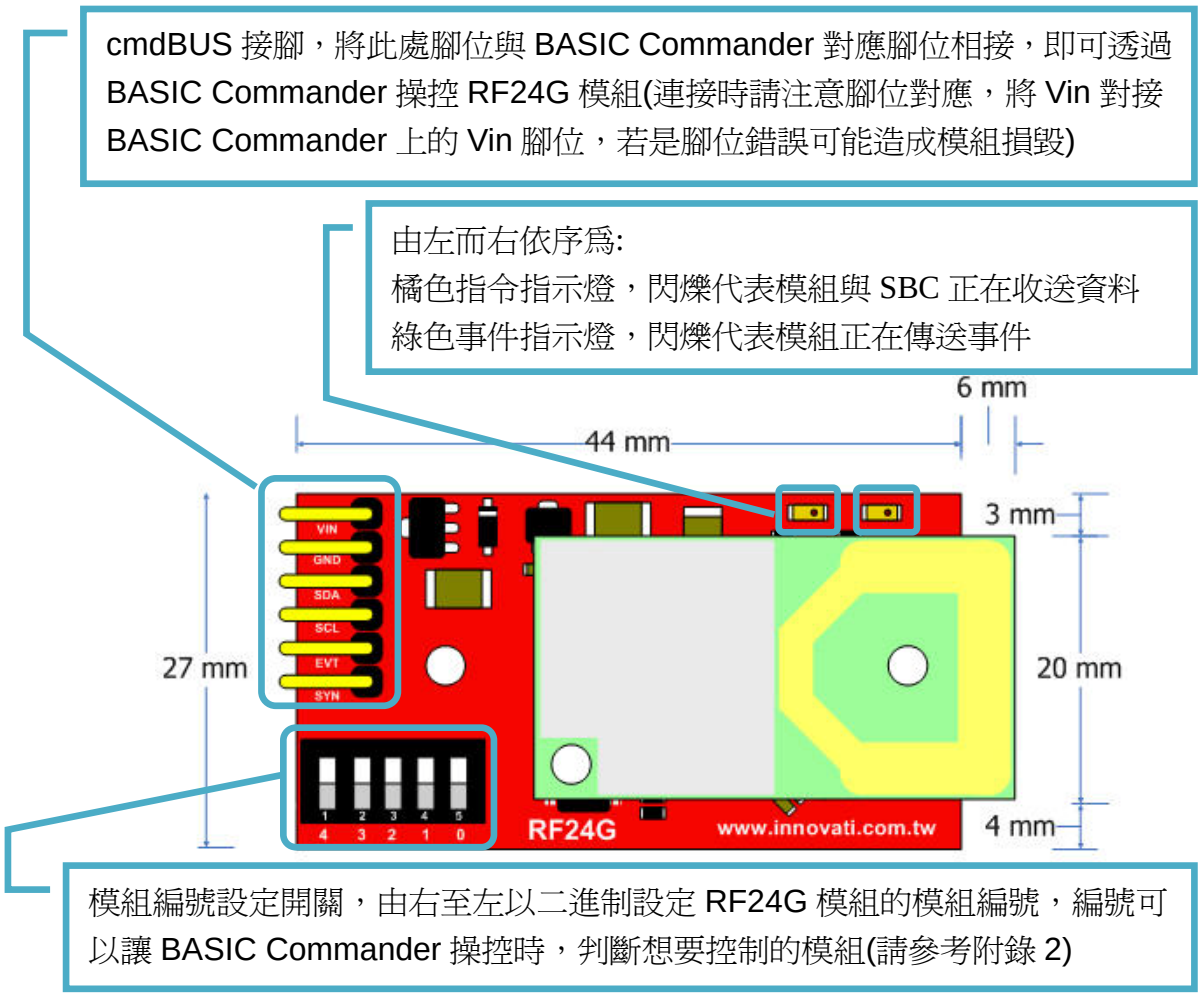


圖 2: 模組腳位與開關介紹

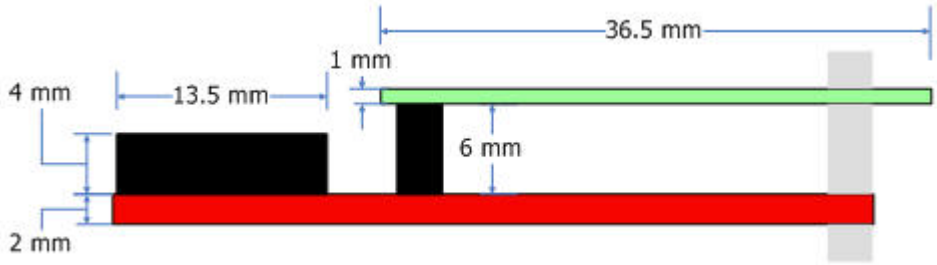


圖 3: 模組側視圖

操作注意事項:

- 無線傳輸時，請不要碰觸天線以免影響資料收送。

模組操作溫度　0 ℃~ 70℃
模組儲存溫度　-40 ℃~85℃

指令表:

下面的指令表是專供控制 RF24G 模組的各種指令，必要輸入的指令名稱與參數，以粗底或粗斜體表示，粗體的文字在輸入時請不要更改，粗斜體的文字請自行定義適當格式的參數填入。輸入時請注意 innoBASIC Workshop 大寫與小寫會視爲相同字。
在執行 RF24G 指令前，請先於程式開頭定義對應參數與編號，例:
Peripheral *ModuleName* As RF24G @ *ModuleID*

指令格式	指令功能
資料收送相關指令…傳送模式	
SendVar(*Data*)	傳送 ***Data*** 的資料，***Data*** 可以爲任意參數
SendArray(*Array*)	傳送 ***Array*** 的資料，***Array*** 爲一個陣列參數，陣列的總 Byte 數須小於等於 20 Bytes
SendString(*String*)	傳送 ***String*** 的資料，***String*** 爲一個字串參數，字串的總字數須小於等於 20 個字元
BufferVar(*Data*)	將 ***Data*** 儲存到傳送資料暫存記憶體中，***Data*** 可以爲任意參數 ✻ 1
BufferArray(*Array*)	將 ***Array*** 儲存到傳送資料暫存記憶體中，***Array*** 爲一個陣列參數，陣列的總 Byte 數須小於等於 20 Bytes ✻ 1
BufferString(*String*)	將 ***String*** 儲存到傳送資料暫存記憶體中，***String*** 爲一個字串參數，字串的總字數須小於等於 20 個字元 ✻ 1
SendBuffer()	將儲存於傳送資料暫存記憶體的所有資料，一次傳送出去
資料收送相關指令…接收模式	
GetVar(*Data*)	取得接收資料暫存記憶體中的資料，儲存於 ***Data*** 中，需要根據傳送端設定的參數格式，將 ***Data*** 設

	定爲對應的參數格式
GetArray(*Array*)	取得接收資料暫存記憶體中的資料，儲存於 ***Array*** 中，需要根據傳送端設定的陣列長度與參數格式，將 ***Array*** 設定爲對應的陣列值
GetArray(*String*)	取得接收資料暫存記憶體中的資料，儲存於 ***String*** 中，需要根據傳送端設定的字串長度，將 ***String*** 設定爲對應的長度值
資料收送相關指令…收送共用	
GetStatus(*Status*)	發送模式 ***Status*** = 0: 待命傳送狀態，模組此時可以接受各種傳送指令 ***Status*** = 1: 資料傳送狀態，模組此時僅能接受模式設定指令，不能執行傳送指令，也不能執行 **Config** 更改模式或狀態 接收模式 ***Status*** = 0: 接收資料暫存記憶體中沒有收到新的資料 ***Status*** = 1~40: 代表接收資料暫存記憶體中的資料筆數，亦即 1 代表有一筆資料，最多會有 40 筆資料儲存在記憶體中，若是在有資料未讀取的狀態下，又收到新資料，則就資料會被清空，若是有啓動 **DataLostEvent**，也會產生 **DataLostEvent** 提醒
ClrBuffer(*Status*)	發送模式 清除所有傳送資料暫存記憶體中的資料 接收模式 清除所有接收資料暫存記憶體中的資料，使用 **GetStatus** 會讀回 0，此時不會產生 **DataLostEvent** 提醒事件
收送狀態設定相關指令	
SetMode(*Mode*)	根據 ***Mode*** 值設定模組爲發送或接收模式 ***Mode*** = 0 ➔ 設定模組爲發送模式 ***Mode*** = 1 ➔ 設定模組爲接收模式 預設值爲 1 (接收模式) ✲ 2
GetMode(*Mode*)	取得現在設定的模式值放於參數 ***Mode*** 中 ***Mode*** = 0 ➔ 設定模組爲接收模式 ***Mode*** = 1 ➔ 設定模組爲發送模式
SetCh(*Channel*)	根據 ***Channel*** 值設定模組爲所使用的頻道，***Channel*** 可以設定爲 0~124 之件的整數值，預設值爲 0 ✲ 2
GetCh(*Channel*)	取得現在模組設定的頻道放於參數 ***Channel*** 中，

	Channel 會回傳 0~124 之間的整數
SetRFID(*ID*)	根據 ***ID*** 值設定模組所使用的識別碼，***ID*** 可以設定爲 0~255 之間的整數值，預設值爲 0 ✲ 2
GetRFID(*ID*)	取得現在模組設定的識別碼放於參數 ***ID*** 中，***ID*** 會回傳 0~255 之間的整數
SetRegCode(*Reg*)	根據 ***Reg*** 值設定模組所使用的註冊碼，***Reg*** 可以設定爲 0~255 之間的整數值，預設值爲 0 ✲ 2
GetRegCode(*Reg*)	取得現在模組設定的註冊碼放於參數 ***Reg*** 中，***Reg*** 會回傳 0~255 之間的整數
Config()	將設定的模式，頻道，識別碼與註冊碼，下載到模組中 ✲ 2
SetPower(*Power*)	根據 ***Power*** 值設定模組發送時的功率，***Power*** 可以設定爲 0~3 之間的整數值 0: -20 dBm 1: -10 dBm 2: -5 dBm 3: 0 dBm 預設值爲 3
GetPower(*Power*)	取得現在模組設定的發送功率值放於參數 ***Power*** 中，***Power*** 會回傳 0~3 之間的整數
SaveConfig(*Num*)	將頻道，識別碼與註冊碼，儲存於 ***Num*** 指定的位置，***Num*** 可以輸入 1~10 之間的整數
LoadConfig(*Num*)	由 ***Num*** 指定的位置，讀取儲存的的頻道，識別碼與註冊碼值，***Num*** 可以輸入 0~10 之間的整數，0 爲回復預設值
提醒事件相關指令	
EnDataLostEvent()	啓動資料遺失事件提醒，執行後在資料遺失時會產生 **DataLostEvent**，預設爲啓動 ✲ 3
DisDataLostEvent()	關閉資料遺失事件提醒，執行後在資料遺失時不會產生 **DataLostEvent**，預設爲啓動 ✲ 3
EnTxReadyEvent()	啓動傳送完成事件提醒，執行後在資料傳送完成時會產生 **TxReadyEvent**，預設爲關閉
DisTxReadyEvent()	關閉傳送完成事件提醒，執行後在資料遺失時不會產生 **TxReadyEvent**，預設爲關閉
EnRxReadyEvent()	啓動接收完成事件提醒，執行後在接收到新資料時會產生 **RxReadyEvent**，預設爲關閉
DisRxReadyEvent()	關閉接收完成事件提醒，執行後在接收到新資料時不會產生 **RxReadyEvent**，預設爲關閉
EnBufferFullEvent()	啓動傳送資料暫存記憶體存滿事件提醒，執行後在傳送資料暫存記憶體存滿時，又再下達 Buffer 相關指令，就會產生 **BufferFullEvent**，預設爲啓

	動
DisBufferFullEvent()	關閉傳送資料暫存記憶體存滿事件提醒，執行後在傳送資料暫存記憶體存滿時，又再下達 Buffer 相關指令，不會產生 **BufferFullEvent**，預設爲啓動
EnRxErrorEvent()	啓動接收資料錯誤事件提醒，執行後，在接收到內部判斷爲 CRC 或是傳輸格式錯誤的資料時，就會產生 **RxErrorEvent**，預設爲關閉
DisRxErrorEvent()	關閉接收資料錯誤事件提醒，執行後，在接收到內部判斷爲 CRC 或是傳輸格式錯誤的資料時，不會產生 **RxErrorEvent**，預設爲關閉

✤ 1 傳送資料暫存記憶體最多可儲存 40 Bytes 的資料，當儲存滿 40 Bytes 後再執行 Buffer 相關的指令，會被視爲無效指令

✤ 2 發送接收模式，頻道，識別碼與註冊碼，在設定後並不會立刻更新，而是要執行 Config 指令後，才會一次更新此四項資料並啓動設定值

✤ 3 此處的資料遺失，是指接收的資料尚未被讀取，又有接收到新的資料，此時原先在接收資料暫存記憶體的資料會被清除，而以新的資料取代

模組提供應用事件:

事件名稱 (Event)	啓動條件
DataLostEvent	在執行 **EnDataLostEvent** ()後，在接收模式，當偵測到接收資料暫存記憶體中資料未被讀取，就被新資料覆蓋時，就會產生
TxReadyEvent	在執行 **EnTxReadyEvent** ()後，在發送模式，當偵測到資料傳送完畢時，就會產生
RxReadyEvent	在執行 **EnRxReadyEvent** ()後，在接收模式，當接收資料暫存記憶體中的資料已經被讀取過，又接收到新資料時，就會產生
BufferFullEvent	在執行 **EnBufferFullEvent** ()後，在發送模式，當偵測到傳送資料暫存記憶體內已有 40 Bytes 的資料，又執行 **Buffer** 相關指令，就會產生
RxErrorEvent	在執行 **EnRxErrorEvent ()**後，在接收模式，接收到的資料判斷爲 CRC 錯誤或格式錯誤時，就會產生

範例程式:

```
'=================================================================
'     RF24G 發送端範例程式
'=================================================================
Peripheral myT As RF24G @ 0                    '     設定模組編號爲 0

Dim g_bTxReady As Byte                         '     宣告傳送狀態參數

Sub Main()                                     '     主程式開始
      Dim bTx As Byte                          '     宣告傳送資料參數

      Debug CLS                                '     清除終端視窗顯示
      myT.SetMode(0)                           '     設定模式爲發送模式
      myT.SetCh(0)                             '     設定傳輸頻道爲 0
      myT.SetRFID(0)                           '     設定辨識碼爲 0
      myT.SetRegCode(0)                        '     設定註冊碼爲 0
      myT.EnTxReadyEvent()                     '     啓動傳送完成事件
      myT.Config()                             '     更新設定值

'-----------------------------------------------------------------
'     用 FOR 迴圈重複傳送動作一百次
'-----------------------------------------------------------------
      For bTx=1 To 100                         '     會執行一百次的 FOR 迴圈
            g_bTxReady = 0                     '     清除傳送狀態
            myT.SendVar(bTx)                   '     傳送 bTx

'-----------------------------------------------------------------
'     用 DO 迴圈等待傳送完畢
'-----------------------------------------------------------------
            Do
            Loop Until g_bTxReady=1

            Debug CSRXY(1, 1), %DEC3R bTx      '     顯示傳送值於終端視窗
            Pause 1000                         '     等待一段時間讓接收端接收
      Next

      Debug CSRXY(1, 2), "傳送完畢"            '     顯示傳送結束
End Sub

Event myT.TxReadyEvent()                       '     傳送完成事件
      g_bTxReady = 1                           '     將傳送狀態設爲 1
End Event
```

```
'====================================================================
'     RF24G 接收端範例程式
'====================================================================
Peripheral myR As RF24G @ 31                    '     設定模組編號爲 31

Dim g_bRxReady As Byte                          '     宣告接收狀態參數

Sub Main()                                      '     主程式開始
      Dim bRx As Byte                           '     宣告接收資料參數

      Debug CLS                                 '     清除終端視窗顯示
      myR.SetMode(1)                            '     設定模式爲接收模式
      myR.SetCh(0)                              '     設定傳輸頻道爲 0
      myR.SetRFID(0)                            '     設定辨識碼爲 0
      myR.SetRegCode(0)                         '     設定註冊碼爲 0
      myR.EnRxReadyEvent()                      '     啓動接收完成事件
      myR.Config()                              '     更新設定值

'--------------------------------------------------------------------
'     用 DO 迴圈等待接收到最後一筆資料
'--------------------------------------------------------------------
      Do
'--------------------------------------------------------------------
'     用 DO 迴圈等待接收資料
'--------------------------------------------------------------------
            Do
            Loop Until g_bRxReady=1

            myR.GetVar(bRx)                     '     讀取接收值
            g_bRxReady = 0                      '     清除傳送狀態
            Debug CSRXY(1, 1), %DEC3R bRx       '     顯示接收值於終端視窗
      Loop Until bRx=100

      Debug CSRXY(1, 2), "接收完畢"             '     顯示接收結束
End Sub

Event myR.RxReadyEvent()                        '     接收完成事件
      g_bRxReady = 1                            '     將接收狀態設爲 1
End Event
```

附錄

1. 已知問題:

2. 模組編號開關對應編號表:

	0		8		16		24
	1		9		17		25
	2		10		18		26
	3		11		19		27
	4		12		20		28
	5		13		21		29
	6		14		22		30
	7		15		23		31