

Innovati's RF24G

Wireless Bidirectional Transceiver Module

Version: V1.0



Product Overview: Innovati's RF24G module is a user-friendly bidirectional wireless transmitter/receiver module. It allows the user to transmit the data in various formats by using a single command through the connection of cmdBUS to the BASIC Commander. With the dynamic configurations through the software, the user can switch the transmission/reception modes any time so as to change the transmission channel and identification code.

Applications:

- Wireless transmission of various data formats.
- Transmission of control signals to achieve the goal of wireless remote control.
- With the collaboration of four RF24G modules at the same time, full-duplex communications can be achieved.

Product Features:

- Frequency band of the wireless transmission: 2.4 ~ 2.524 GHz.
- Wireless transmission mode: GFSK.
- The user can switch the operation mode of the module as a transmitter or a receiver any time.
- The user can switch among the 125 channels dynamically through the software.
- Output power: 0 dBm.
- Data transfer rate: 250 Kbps.
- The wireless transmission coverage is up to about 280 meters.
- With the built-in antenna, no external antenna is required.
- Provides 256 sets of ID codes and Reg codes for the user to switch the identification dynamically through the software any time.
- The user can store the data to be transmitted in the built-in temporary space first and then transmit them in a batch by using the command. Up to 40 Bytes can be stored.
- Provides simple variable transmission commands. Different data types such as Byte, Word, and Dword can be transmitted through a single command.
- Provides string and array transmission commands. A string of up to 20 characters or an array of up to 20 Bytes can be transmitted at the same time.
- Measurement alarm events are provided. After the module is activated, when a new measurement is updated, an alarm event will be generated.
- Provides the configuration of transmission completion notification event, which can be activated as soon as the data transmission is completed.
- Provides the configuration of receipt completion notification event which will be activated as soon as the receipt of new data is completed.
- Provides four levels of transmission signal strength: -20 dBm, -10 dBm, -5 dBm, and 0 dBm.
- By using commands, the user can read out the current settings for verification or determine if the received data is still not read out yet.

Connection: Place the ID switch on the required number directly, and then connect the cmdBUS to the corresponding pins on the BASIC Commander so that the user can perform the required operations through the BASIC Commander.

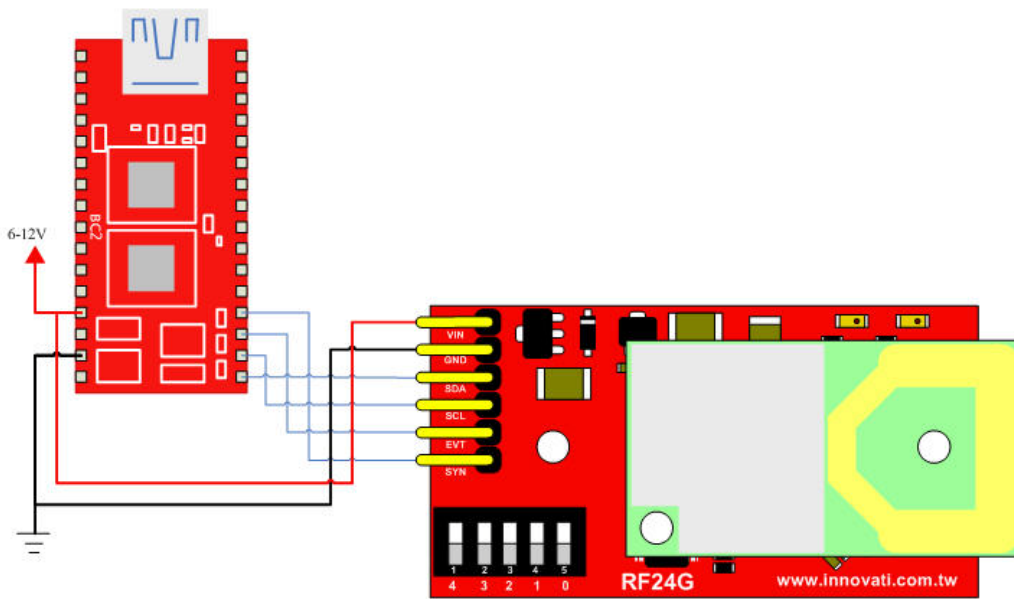
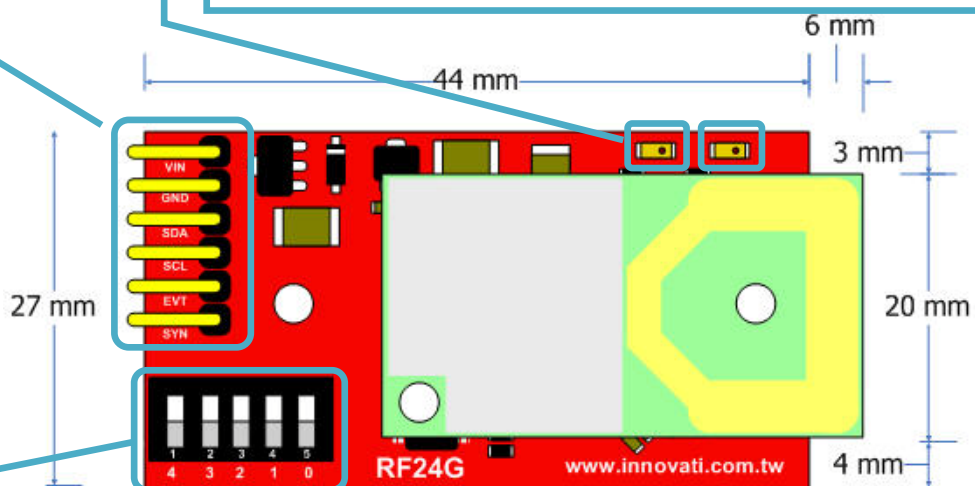


Figure 1 Connection with the BASIC Commander

Product Specifications:

Pins for cmdBUS: Connect these pins to the corresponding pins on the BASIC Commander for controlling the RF24G module through the BASIC Commander. While connecting, please notice the pin assignment. Connect Vin to the Vin on the BASIC Commander. Incorrect pin connection may cause damage to the module.)

From the left to the right:
 Orange Command Indicator: The blinking light indicates the module and the SBC are transmitting/receiving data.
 Green Event Indicator: The blinking light indicates that the module is transmitting an event.



Module ID Setting Switch: The module ID of the RF24G module can be configured with the binary digits from the right to the left. This ID number allows the BASIC Commander to determine the module to be controlled during the operation (Please refer to Appendix 2).

Figure 2 Description of pins and switches on the module

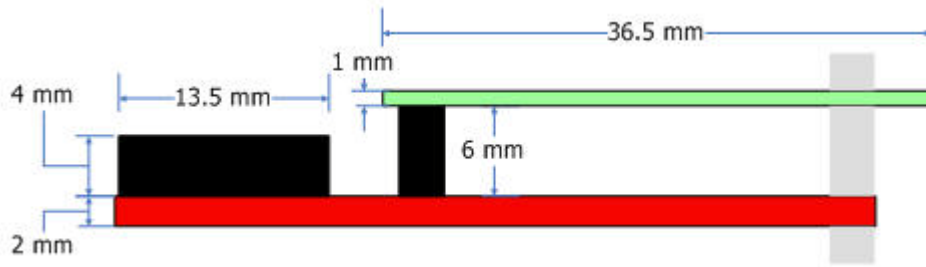


Figure 3 Side view of the module

Precautions for Operations:

- While the wireless transmission is being performed, please do not touch the antenna to avoid influencing the data transmission and reception.

Operating Temperature of the Module: 0°C ~ 70°C

Storage Temperature of the Module: -40°C ~ 85°C

List of Commands:

The following list shows the commands dedicated to controlling the RF24G module. The command name and parameters which should be input are shown in bold or bold-italic typefaces. The words in bold typeface should not be changed while being input. The words in bold-italic typefaces can be filled with parameters in properly defined format by the user. Please note that the words in uppercase or lowercase are regarded as the same word while entering the command in the innoBASIC Workshop.

Before executing the command for RF24G, please define the corresponding parameters and the module ID at the beginning of the command, for example:

Peripheral *ModuleName* As RF24G @ *ModuleID*

Command Format	Command Function
Commands for data transmission and reception ... Transmission mode	
SendVar(<i>Data</i>)	Transmit the data specified by <i>Data</i> . The content of <i>Data</i> can be any value.
SendArray(<i>Array</i>)	Transmit an array specified by <i>Array</i> . The content of <i>Array</i> is an array of data. The total size of the array should be less than or equal to 20 Bytes.
SendString(<i>String</i>)	Transmit a string specified by <i>String</i> . The content of <i>String</i> is a string of characters. The total number of characters of the string should be less than or equal to 20 characters.
BufferVar(<i>Data</i>)	Store the content of <i>Data</i> in the temporary memory for transmission (transmission buffer). The content of <i>Data</i> can be any value. *1
BufferArray(<i>Array</i>)	Store the content of <i>Array</i> in the temporary memory for transmission (transmission buffer). The content of <i>Array</i> is an array of data. The total size of the array should be less than or equal to 20 Bytes. *1
BufferString(<i>String</i>)	Store the content of <i>String</i> in the temporary memory for transmission (transmission buffer). The total number of characters of the string should be less than

	or equal to 20 characters. *1
SendBuffer()	Transmit all the data stored in the temporary memory for transmission (transmission buffer) in a single batch.
Commands for data transmission and receipt ... Receiving mode	
GetVar(Data)	Get the data from the temporary memory for the received data (receiving buffer) and then store it in Data . It is necessary to set the data type for Data according its data type defined in the transmitter.
GetArray(Array)	Get the data from the temporary memory for the received data (receiving buffer) and then store it in Array . It is necessary to set the length and data type of Array according its length and data type defined in the transmitter.
GetArray(String)	Get the data from the temporary memory for the received data (receiving buffer) and then store it in String . It is necessary to set the number of characters of the String according its number of characters defined in the transmitter.
Commands for data transmission and receipt ... Transmission and receiving modes	
Status=GetStatus()	<p style="text-align: center;">Transmission Mode</p> <p>Status = 0: Ready for transmission. The module is ready for receipt of various data transmission commands.</p> <p>Status = 1: Data transmission in progress. The module is ready only for receiving the mode setting commands. It is not able to perform the data transmission commands and the Config command to change the mode or status.</p> <p style="text-align: center;">Receiving Mode</p> <p>Status = 0: There is no newly received data in the temporary memory for data receipt (receiving buffer).</p> <p>Status = 1~40: The value represents the number of data stored in the temporary memory for data receipt (receiving buffer), i.e., 1 means that one data item is in the buffer. There will be up to 40 data items stored in the buffer. If there is still some data that is not completely read and new data is being received, the original data will be cleared. If the notification event DataLostEvent is enabled, the notification event DataLostEvent will be activated.</p>
ClrBuffer()	<p style="text-align: center;">Transmission Mode</p> <p>Clear all the data stored in the transmission buffer.</p> <p style="text-align: center;">Receiving Mode</p> <p>Clear all the data stored in the receiving buffer. Using the command GetStatus, the retrieved value is 0. In this case, the notification event DataLostEvent will not be activated.</p>
Command for configuring the status for transmission and receipt	

SetMode(<i>Mode</i>)	Set the operation mode of the module as the transmission mode or the receiving mode according to the value of <i>Mode</i> . <i>Mode</i> = 0 → Set the module in the transmission mode <i>Mode</i> = 1 → Set the module in the receiving mode The default value is 1 (receiving mode) *2
GetMode(<i>Mode</i>)	Get the current mode setting and store it in <i>Mode</i> <i>Mode</i> = 0 → Set the module in the receiving mode <i>Mode</i> = 1 → Set the module in the transmission mode
SetCh(<i>Channel</i>)	Set the channel to be used by the module with <i>Channel</i> . The input value of <i>Channel</i> can be an integer in the range of 0~124. The default value is 0. *2
GetCh(<i>Channel</i>)	Get the current channel setting of the module and store it in <i>Channel</i> . The retrieved value of <i>Channel</i> will be an integer in the range of 0~124.
SetRFID(<i>ID</i>)	Set the identification code of the module with <i>ID</i> . The input value of <i>ID</i> can be an integer in the range of 0~255. The default value is 0. *2
GetRFID(<i>ID</i>)	Get the current identification code of the module and store it in <i>ID</i> . The retrieved value of <i>ID</i> will be an integer in the range of 0~255.
SetRegCode(<i>Reg</i>)	Set the registration code of the module with <i>Reg</i> . The input value of <i>Reg</i> can be an integer in the range of 0~255. The default value is 0. *2
GetRegCode(<i>Reg</i>)	Get the current registration code of the module and store it in <i>Reg</i> . The retrieved value of <i>Reg</i> will be an integer in the range of 0~255.
Config()	Load the preset mode, channel, ID code, and Reg code into the module. *2
SetPower(<i>Power</i>)	Set the transmission power of the module according to the value of <i>Power</i> . The input value of <i>Power</i> can be an integer in the range of 0~3. 0: -20 dBm. 1: -10 dBm. 2: -5 dBm. 3: 0 dBm. The default value is 3.
GetPower(<i>Power</i>)	Get the preset transmission power of the module and store it in <i>Power</i> . The retrieved value of <i>Power</i> will be an integer in the range of 0~3.
SaveConfig(<i>Num</i>)	Store the channel, ID code and Reg code in the location specified by <i>Num</i> . The input value of <i>Num</i> can be an integer in the range of 1~10.
LoadConfig(<i>Num</i>)	Get the channel, ID code and Reg code stored in the location specified by <i>Num</i> . The input value of <i>Num</i> can be an integer in the range of 0~10. The input

	value 0 is used for restoring the default settings.
Commands for Notification Events	
EnDataLostEvent()	Enable the data loss notification event. After this command is executed, the notification event DataLostEvent will be activated when the data is lost. The default setting is “enable.” *3
DisDataLostEvent()	Disable the data loss notification event. After this command is executed, the notification event DataLostEvent will not be activated when the data is lost. The default setting is “enable.” *3
EnTxReadyEvent()	Enable the transmission complete notification event. After this command is executed, the notification event TxReadyEvent will be activated when the data transmission is completed. The default setting is “disable.”
DisValReadyEvent()	Disable the transmission complete notification event. After this command is executed, the notification event TxReadyEvent will not be activated when the data transmission is completed. The default setting is “disable.”
EnRxReadyEvent()	Enable the receiving complete notification event. After this command is executed, the notification event RxReadyEvent will be activated when the data receipt is completed. The default setting is “disable.”
DisRxReadyEvent()	Disable the receipt complete notification event. After this command is executed, the notification event RxReadyEvent will not be activated when the data receipt is completed. The default setting is “disable.”
EnBufferFullEvent()	Enable the notification event once the transmission buffer is full. After this command is executed, if the transmission buffer is full and the buffer-related command is performed, the notification event BufferFullEvent will be activated. The default setting is “enable.”
DisBufferFullEvent()	Disable the notification event once the transmission buffer is full. After this command is executed, if the transmission buffer is full and the buffer-related command is performed, the notification event BufferFullEvent will not be activated. The default setting is “enable.”
EnRxErrorEvent()	Enable the notification event for receiving error. After this command is executed, if the received data is determined internally as invalid data with CRC error or format error, the notification event RxErrorEvent will be activated. The default setting is “disable.”
DisRxErrorEvent()	Disable the notification event for receiving error. After this command is executed, if the received data is determined internally as invalid data with CRC

	error or format error, the notification event RxErrorEvent will not be activated. The default setting is “disable.”
--	--

*1 The transmission buffer can store up to 40 Bytes of data. When the buffer is full and a buffer-related command is performed, the command will be invalid.

*2 The transmission/receipt mode, channel, ID code and Reg code will not be updated immediately after the setting command. These four parameters will be updated and effective in a batch only after the command Config is executed.

*3 The “data lost” here means that when the received data is not completely read before new data is being received, the data originally stored in the receiving buffer will be cleared and replaced by the newly received data.

Application Events Provided by the Module:

Event	Activation Condition
DataLostEvent	After the command EnDataLostEvent () is executed in the receiving mode, when the data originally stored in the receiving buffer is not completely read before the new data is being received and thus buffer overwrite is detected, this notification event will be activated.
TxReadyEvent	After the command EnTxReadyEvent () is executed in the transmission mode, when the completion of data transmission is detected, this notification event will be activated.
RxReadyEvent	After the command EnRxReadyEvent () is executed in the receiving mode, when the data stored in the receiving buffer has been completely read and new data is being received, this notification event will be activated.
BufferFullEvent	After the command EnBufferFullEvent () is executed in the transmission mode, when the transmission buffer is full and a Buffer -related task is performed, this notification event will be activated.
RxErrorEvent	After the command EnRxErrorEvent () is executed in the receiving mode, if the received data is determined invalid data with a CRC error or format error, this notification event will be activated.

Demonstration Program:

```
'=====
'      Demonstration program for RF24G as a transmitter
'=====

Peripheral myT As RF24G @ 0          '      Set the module ID as 0

Dim g_bTxReady As Byte              '      Declare the variable for the transmission status

Sub Main()                          '      Main program starts here
    Dim bTx As Byte                  '      Declare the variable for the data to be transmitted

    Debug CLS                        '      Clear the terminal display
    myT.SetMode(0)                   '      Set the mode as transmission mode
    myT.SetCh(0)                     '      Set the transmission channel as 0
    myT.SetRFID(0)                   '      Set the ID code as 0
    myT.SetRegCode(0)                '      Set the Reg code as 0
    myT.EnTxReadyEvent()             '      Enable the transmission complete notification event
    myT.Config()                     '      Update the setting values

'-----
'      Use the FOR loop to repeat the transmission operation a hundred times
'-----

    For bTx=1 To 100                 '      Perform the FOR loop a hundred times
        g_bTxReady = 0               '      Clear the transmission status
        myT.SendVar(bTx)             '      Transmit the content of bTx

'-----
'      Use the DO loop to wait for the completion of the transmission operation
'-----

        Do
            Loop Until g_bTxReady=1

            Debug CSRXY(1, 1), %DEC3R bTx '      Display the transmitted values on the terminal display
            Pause 1000                  '      Wait for a period of time before the receiver starts receiving
        Next

        Debug CSRXY(1, 2), "Transmission complete" '      Display that the transmission is complete
    End Sub

Event myT.TxReadyEvent()             '      Notification event for transmission complete
    g_bTxReady = 1                    '      Set the transmission status as 1
End Event
```



```

=====
'
'   Demonstration program for RF24G as a receiver
'
=====
Peripheral myR As RF24G @ 31          '   Set the module ID as 31

Dim g_bRxReady As Byte              '   Declare the variable for receiving status

Sub Main()                          '   Main program starts here
    Dim bRx As Byte                  '   Declare the variable for the received data

    Debug CLS                       '   Clear the terminal display
    myR.SetMode(1)                   '   Set the mode as receiving mode
    myR.SetCh(0)                     '   Set the transmission channel as 0
    myR.SetRFID(0)                   '   Set the ID code as 0
    myR.SetRegCode(0)                '   Set the Reg code as 0
    myR.EnRxReadyEvent()             '   Enable the receiving complete notification event
    myR.Config()                     '   Update the setting values

'-----
'   Use the DO loop to wait for the last data item to be received
'-----
    Do

'-----
'   Use the DO loop to wait for the receipt of data
'-----
        Do
            Loop Until g_bRxReady=1

            myR.GetVar(bRx)            '   Read the received data value
            g_bRxReady = 0            '   Clear the transmission status
            Debug CSRXY(1, 1), %DEC3R bRx '   Display the received values on the terminal display
        Loop Until bRx=100

        Debug CSRXY(1, 2), "Receiving complete" '   Displays that the receipt is completed
End Sub









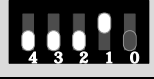

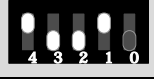
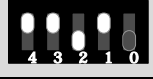


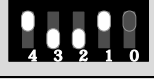

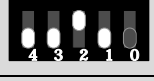
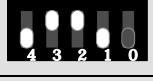
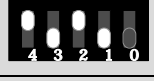
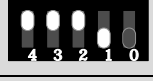
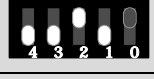

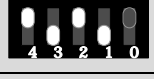

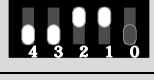

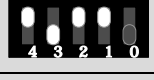

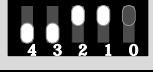

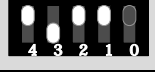

Event myR.RxReadyEvent()            '   Notification event for receipt complete
    g_bRxReady = 1                   '   Set the receiving status as 1
End Event

```

Appendix

1. Known problems:

2. List of the Configuration of the Module ID Switch:

	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