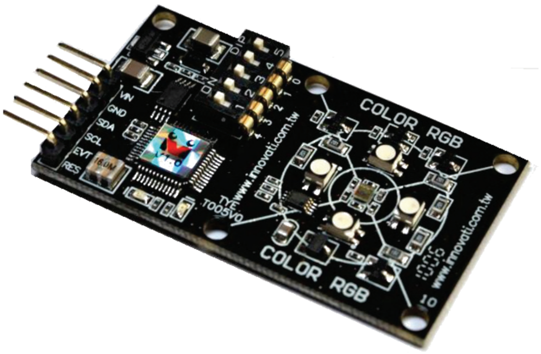
Innovati’s Color RGB

Color Sensing Module

Version: V1.0



Product Overview: Innovati's Color RGB module is a user-friendly, three-wavelength color sensing module. Users can use simple commands to generate color values, and can even adjust total light intensity and the intensities of separate red, green, and blue colors from Innovati’s Color RGB module. In addition, users can also set the color codes and sensor values to automatic matching and comparison operations. The module provides four full-color LEDs which allow users to adjust the intensity of each light source directly.

Application:

- Measure three color values of reflected light from an object.
- Sense light intensity in an environment.
- Combine and store color codes, and program reactions and patterns based on those color codes.

Product Features:

- It can sense full-color values (red, green and blue) of the reflected light separately (λ_p, red: 640 nm, green: 524 nm, blue: 470 nm).
- It can sense the total light intensity of any mixture of full-color values.
- It can respond to energy values equal to or greater than 61μW/cm^2.
- It can store up to 255 color values. Users can assign RGB identification values for each stored color value.
- It features a search function which allows users to search stored values in partial or overall ranges. It can also retrieve the differences between sensed color values and each of the stored color values, taken either individually or as a group.
- It integrates with four sets of full-color LEDs which can be directly turned on or off through software controls. The intensity of each color component of the four LED light sources can be adjusted separately.
- It can dim the intensity of the LED light source directly by using scaling factors.
- It provides three sets of light sensing times which can be switched any time with a shutter switch effect.
- It provides several built-in retrieval value ratios for the user to select so as to obtain a proper range for retrieved values, including 4 sets of “scaling” (scaling up) factors and 7 sets of “pre-scaling” (scaling down) factors which can be switched directly with built-in software.
- Notifies completion of sensing operations in real time to help users manage operations.
- Notifies the completion of search operations to help users streamline processes.

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

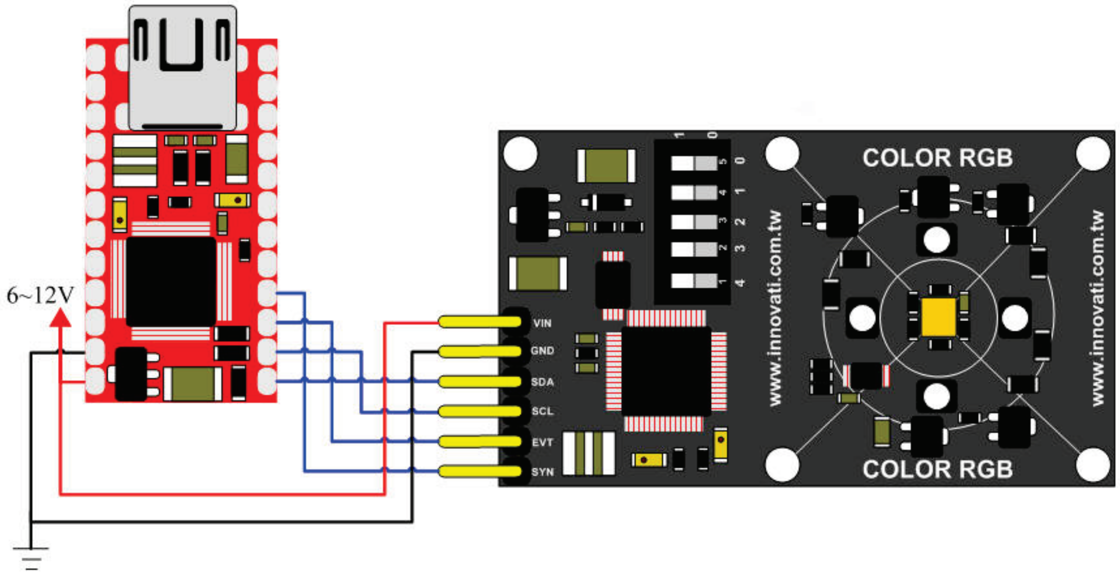
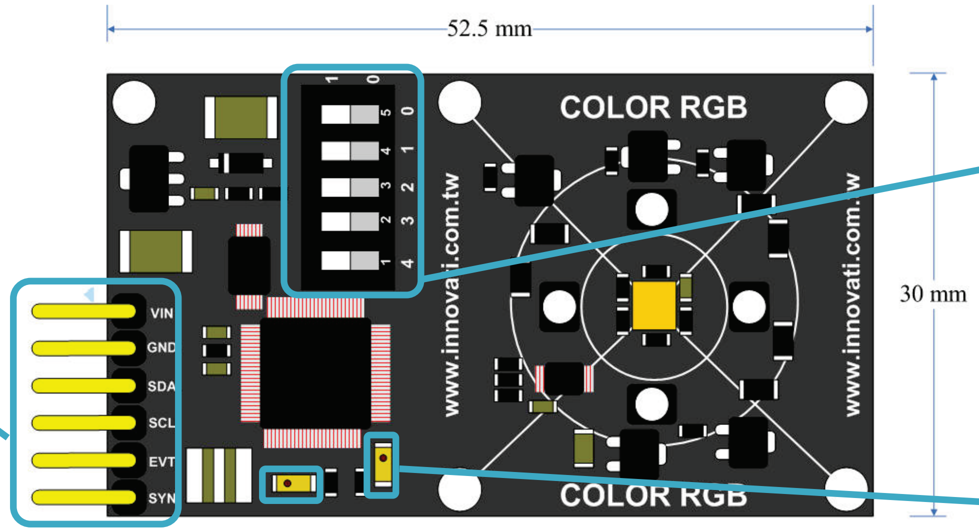


Figure 1 Connection with the BASIC Commander

Product Specifications:

Pins for cmdBUS: Connect these pins to the corresponding pins on the BASIC Commander for controlling the Color RGB module. 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.)

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



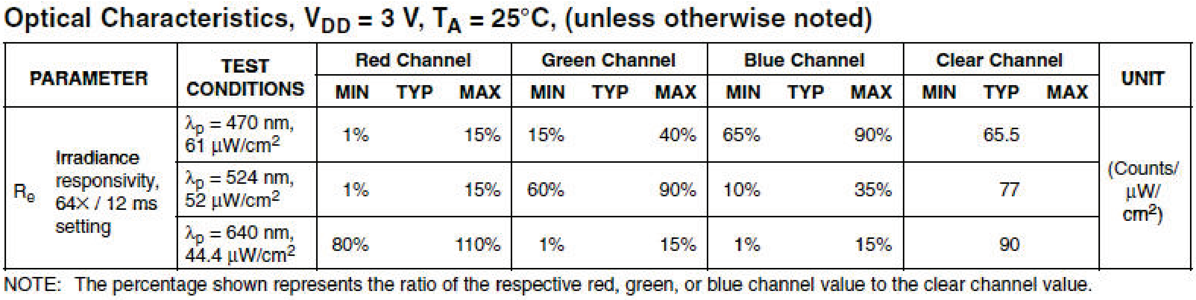
From the left to the right:
Green Event Indicator: The blinking light indicates that the module is transmitting an event.
Orange Command Indicator: The blinking light indicates the module and the SBC are receiving data.

Figure 2 Description of pins and switches on the module

Precautions for Operations:

Operating Temperature: 0 ℃ ~ 70℃

Storage Temperature: -40 ℃ ~ 85℃



Optical Characteristics, V_{DD} = 3 V, T_A = 25°C, (unless otherwise noted)

PARAMETER		TEST CONDITIONS	Red Channel			Green Channel			Blue Channel			Clear Channel			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
R_e	Irradiance responsivity, 64× / 12 ms setting	λ_p = 470 nm, 61 µW/cm^2	1%		15%	15%		40%	65%		90%		65.5		(Counts/ µW/ cm^2)
		λ_p = 524 nm, 52 µW/cm^2	1%		15%	60%		90%	10%		35%		77		
		λ_p = 640 nm, 44.4 µW/cm^2	80%		110%	1%		15%	1%		15%		90		

NOTE: The percentage shown represents the ratio of the respective red, green, or blue channel value to the clear channel value.

Table 1 - Reference table for optical characteristics

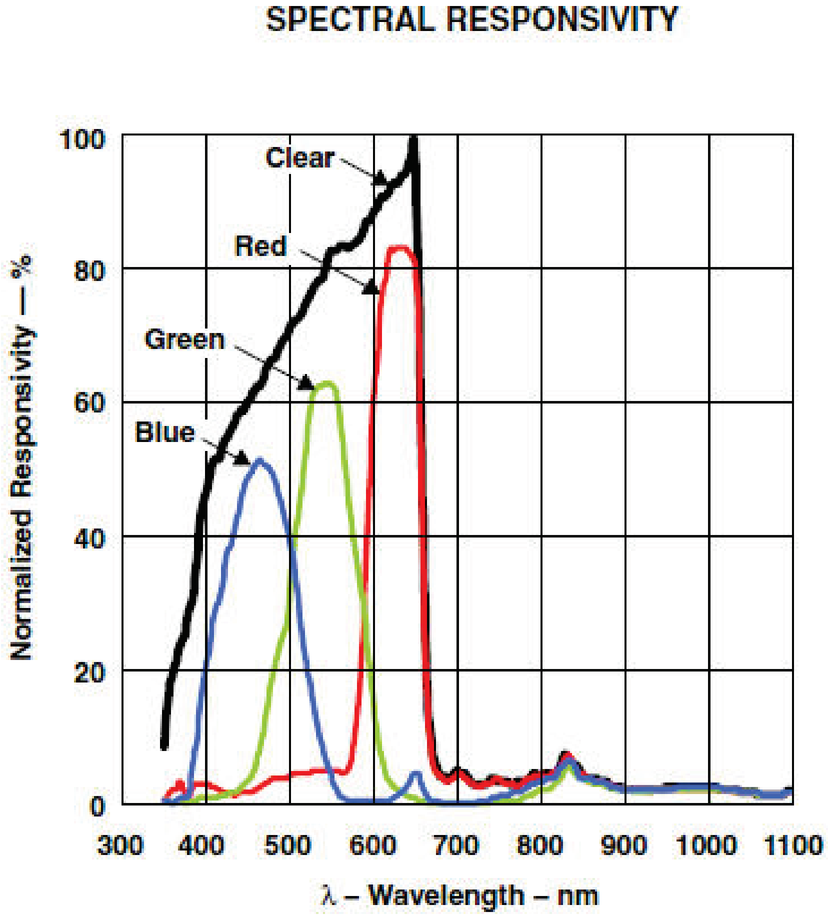


Table 2 - Spectral responsivity

List of Commands:

The following list shows the commands dedicated to controlling the Color RGB 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-in with parameters in the properly defined format. Note that the words in uppercase or lowercase are regarded as the same word while entering the program into the software interface, innoBASIC Workshop.

Before executing commands for Color RGB, define the corresponding parameters and the module ID at the beginning of the program. For example:

Peripheral *ModuleName* As ColorRGB @ *ModuleID*

Command Format	Command Function
Commands for the measurement and for reading measured values	
***bStatus*=GetClear(*Value*)**	Identifies measurement status and stores it in ***bStatus***; a value of 0 means the measured value has not been updated since its last reading; a value of 1 means that the measured value has been updated and the new clear value is stored in ***Value***. The retrieved value of ***Value*** is an integer in the range of 0~65535.
***bStatus*=GetRed(*Value*)**	Indentify measurement status and store it in ***bStatus***; a value of 0 means the measured value has not been updated since its last reading; a value of 1 means that the measured value has been updated and the new red color intensity value is stored in ***Value***. The retrieved value of ***Value*** is an integer in the range of 0~65535.
***bStatus*=GetGreen(*Value*)**	Gets the measurement status and stores it in ***bStatus***; a value of 0 means the measured value has not been updated since its last reading; a value of 1 means that the measured value has been updated and the new green color intensity value is stored in ***Value***. The retrieved value of ***Value*** is an integer in the range of 0~65535.
***bStatus*=GetBlue(*Value*)**	Get the measurement status and store it in ***bStatus***; a value of 0 means the measured value has not been updated since its last reading; a value of 1 means that the measured value has been updated and the new blue color intensity value is stored in ***Value***. The retrieved value of ***Value*** is an integer in the range of 0~65535.
***bStatus*=GetRGB(*ValueR*, *ValueG*, *ValueB*)**	Gets the measurement status and stores it in ***bStatus***; a value of 0 means the measured value has not been updated since its last reading; a value of 1 means that the measured value has been updated and the new values of red, green and blue colors are stored in ***ValueR***, ***ValueG***, and ***ValueB***, respectively. The retrieved values of ***ValueR***, ***ValueG***, and ***ValueB*** are

	integers in the range of 0~65535.
***bStatus*=GetCRGB(*ValueC*, *ValueR*, *ValueG*, *ValueB*)**	Gets the measurement status and stores it in ***bStatus***; a value of 0 means the measured value has not been updated since its last reading; a value of 1 means that the measured value has been updated and the new clear value and the red, green, and blue color values are stored in ***ValueC***, ***ValueR***, ***ValueG***, and ***ValueB***, respectively. The retrieved values of ***ValueR***, ***ValueG***, and ***ValueB*** are integers in the range of 0~65535.
StartMeasure()	This value starts measurement operations. After the command is executed, color measurement will be carried out according to sensing time. To obtain correct measurement values, it is necessary to execute this command to perform color measurement operations. Users must wait until after measurement is completed before the module is allowed to use the commands for retrieving sensed values to obtain updated sensed values. Note that while performing measurement operations, the LED light source will be automatically turned on according to the preset LED light intensity.
Commands for storing sensed values	
SaveCurColorVal(*Num*)	Stores the latest sensed value into the location specified by ***Num***. The input value of ***Num*** should be an integer within the range of 1~255.
SaveColorDes(*Num*, *DesR*, *DesG*, *DesB*)	Uses the sensed value stored in the location specified by ***Num*** to set the reference color values ***DesR***, ***DesG***, and ***DesB***. The input value of ***Num*** can be an integer within the range of 1~255. The values of ***DesR***, ***DesG***, and ***DesB*** can be integers within the range of 0~255.
SaveColorVal(*Num*, *ValC*, *ValR*, *ValG*, *ValB*)	Stores the clear value and RGB color values specified by ***ValC***, ***ValR***, ***ValG***, and ***ValB*** into the locations specified by ***Num***. The input value of ***Num*** can be an integer in the range of 1~255. The input values of ***ValC***, ***ValR***, ***ValG***, and ***ValB*** can be integers in the range of 0~65535.
SaveColorInfo(*Num*, *DesR*, *DesG*, *DesB*, *ValC*, *ValR*, *ValG*, *ValB*)	Stores the clear value and RGB color values specified by ***ValC***, ***ValR***, ***ValG***, and ***ValB*** into the location specified by ***Num*** and set the reference RGB color values by ***DesR***, ***DesG***, and ***DesB***, respectively. The input value of ***Num*** can be an integer in the range of 1~255. The input values of ***ValC***, ***ValR***, ***ValG***, and ***ValB*** can be integers in the range of 0~65535. The

	input values of ***DesR***, ***DesG***, and ***DesB*** can be integers in the range of 0~255.
GetColorVal(*Num*, *ValC*, *ValR*, *ValG*, *ValB*)	Retrieves the sensed clear and color values stored in ***Num*** and stores them in ***ValC***, ***ValR***, ***ValG***, and ***ValB***, respectively. The input value of ***Num*** can be an integer in the range of 1~255. The retrieved values of ***ValC***, ***ValR***, ***ValG***, and ***ValB*** will be integers within the range of 0~65535.
GetColorDes(*Num*, *DesR*, *DesG*, *DesB*)	Retrieves the reference color values stored in ***Num*** and stores them in ***DesR***, ***DesG***, and ***DesB***, respectively. The input value of ***Num*** can be an integer in the range of 1~255. The retrieved values of ***DesR***, ***DesG***, and ***DesB*** will be integers within the range of 0~255.
Commands for searching stored values	
MatchColorDesAll(*DesR*, *DesG*, *DesB*)	Searches the stored data for the color data that coincides with ***DesR***, ***DesG***, and ***DesB***. After issuing this command, execute the command **GetMatchStatus** or wait until the **SearchCompleteEvent** is activated before performing another data retrieval command. The input values of ***DesR***, ***DesG***, and ***DesB*** can be integers in the range of 0~255.
MatchColorDesIndex(*DesR*, *DesG*, *DesB*, *Min*, *Max*)	Searches the stored data for color data that coincides with ***DesR***, ***DesG***, and ***DesB,*** which are stored in between ***Min*** and ***Max***. After issuing this command, execute the command **GetMatchStatus** or wait until the **SearchCompleteEvent** is activated before performing another data retrieval command. The input values of ***DesR***, ***DesG***, and ***DesB*** can be integers in the range of 0~255. The input values of ***Min*** and ***Max*** should be integers in the range of 1~255 and the value of ***Max*** must be greater than that of ***Min***.
MatchCurColorAll()	Uses the most recent measurement values as a base for search operations and then start a search on stored locations. After issuing this command, execute the command **GetMatchStatus** or wait until the **SearchCompleteEvent** is activated before performing another data retrieval command.
MatchCurColorIndex(*Min*, *Max*)	Uses the most recent measurement values as a base for search operations and then start a search on locations between ***Min*** and ***Max***. After issuing this command, execute the command **GetMatchStatus** or wait until the **SearchCompleteEvent** is activated

	before performing the data retrieval command. The input values of ***Min*** and ***Max*** should be integers in the range of 1~255 and the value of ***Max*** must be greater than that of ***Min***.
MatchCurColorRGBAll()	Uses the most recent measurement values, except for the Clear Value, as a base for search operations and then start a search on all stored locations. After issuing this command, execute the command **GetMatchStatus** or wait until the **SearchCompleteEvent** is activated before performing the data retrieval command.
MatchCurColorRGBIndex(*Min*, *Max*)	Uses the most recent measurement values, except for the Clear Value, as the base for the search operations and then start the search operations on the locations between ***Min*** and ***Max***. After issuing this command, execute the command **GetMatchStatus** or wait until the **SearchCompleteEvent** is activated before performing the data retrieval command. The input values of ***Min*** and ***Max*** should be integers in the range of 1~255 and the value of ***Max*** must be greater than that of ***Min***.
GetMatchNum(*Num*)	Retrieves search results and stores them in ***Num***. The retrieved value of ***Num*** will be an integer in the range of 0~255.
GetMatchNeighborNum(*Num1*, *Num2*, *Num3*)	Retrieves three values closest to the search target settings and stores them in ***Num1***, ***Num2***, and ***Num3***, respectively. The retrieved values of ***Num1***, ***Num2***, and ***Num3*** will be integers in the range of 0~255.
***bStatus*=GetMatchStatus()**	Retrieves the status of the comparison operations of match commands. The retrieved value ***bStatus*** has three different states: 0: There is no active search operation, or the search operation is completed but no matching values were found. 1: The search operation is in progress. If the search operation is performed again at this moment, the current search operation will be cancelled and a new search operation will be started. 2: The search operation is completed and retrieval commands can be entered to for search results to retrieve the corresponding values.
GetMatchDiff(*DiffC*, *DiffR*, *DiffG*, *DiffB*)	Retrieves the difference between the closest stored values and the reference values after the comparison operation is complete, and then stores them in ***DiffC***,

	DiffR, ***DiffG***, and ***DiffB***, respectively, which will be integers in the range of -5000~5000.
GetMatchDiffAll(*Diff*)	Retrieves the total difference between the closest stored values and the reference values after the comparison operation is complete, and then stores it ***Diff*** which will be an integer in the range of 0~20000.
SetMatchMaxDiff(*Max*)	Sets the value ***Max*** as the maximum difference of the color values for matching a single color. If the difference between the stored data and the measured value exceeds this value, it is still not determined as a matched color, even if its total difference is smaller than those of other stored values. The input value of ***Max*** can be an integer in the range of 0~5000. The default value is 500.
GetMatchMaxDiff(*Max*)	Retrieves the maximum difference of the color values for matching a single color and stores it in ***Max***. The retrieved value of ***Max*** will be an integer in the range of 0~5000. The default value is 500.
Commands for measurement control	
TurnOnLED()	Turns on the LED till the command **TurnOffLED** is executed.
TurnOffLED()	Turns off the LED. However, while the command **StartMeasure** is performed, the LED will be automatically turned on according to the preset brightness values.
GetLEDStatus(*bStatus*)	Gets the LED status and stores it in ***bStatus*** 0: LED is off 1: LED is on
SetLEDRGB(*Red*, *Green*, *Blue*)	Sets the brightness values of the red, green and blue color components of the LED with the values of ***Red***, ***Green***, and ***Blue***, respectively, when the LED is turned on. The input values of ***Red***, ***Green***, and ***Blue*** can be integers in the range of 0~255.
GetLEDRGB(*Red*, *Green*, *Blue*)	Retrieves the brightness values of the red, green and blue color components of the LED when the LED is turned on and stores them in ***Red***, ***Green***, and ***Blue***, respectively. The retrieved values of ***Red***, ***Green***, and ***Blue*** will be integers in the range of 0~255.
ScaleLED(*Scale*)	Multiplies the preset brightness values of the three LED colors with the input value of ***Scale*** as a percentage and then replaces the brightness values of the three colors with the multiplication results. The maximum and minimum value of the multiplication results will be automatically adjusted as integers that

	fall in the range of 0~255. The input value of ***Scale*** can be in the range of 0~200.
LoadLED()	Restores the brightness values of the LED to the default values.
Commands for Various Settings	
SetPeriod(*Period*)	Sets the measurement time according to the value of ***Period*** 0: 12 ms 1: 100 ms 2: 400 ms The default is 0 (12 ms)
GetPeriod(*Period*)	Retrieves the preset measurement time and stores it in ***Period***. The retrieved value of ***Period*** will be an integer in the range of 0~2. The default value is 0.
SetGain(*Gain*)	Sets the measurement gain according to the value of ***Gain*** 0: 1X 1: 4X 2: 16X 3: 64X The default is 0 (1X)
GetGain(*Gain*)	Retrieves the preset measurement gain and stores it in ***Gain***. The retrieved value of ***Gain*** will be an integer in the range of 0~3.
SetPrescaler(*Prescaler*)	Sets the measurement scaling factor according to the value of ***Prescaler***. 0: 1 1: 1/2 2: 1/4 3: 1/8 4: 1/16 5: 1/32 6: 1/64 The default value is 0 (1)
GetPrescaler(*Prescaler*)	Retrieves the preset measurement scaling factor and stores it in ***Prescaler***. The retrieved value of ***Prescaler*** will be an integer in the range of 0~6.
Commands for the Configuration of Alarm Event	
EnSearchCompleteEvent()	Enables the notification event **SearchCompleteEvent**. The default is "disable."
DisSearchCompleteEvent()	Disables the notification event **SearchCompleteEvent**. The default is "disable."
EnValReadyEvent()	Enables the notification event **ValReadyEvent**. The default is "enable."
DisValReadyEvent()	Disables the notification event **ValReadyEvent**. The

	default is “enable.”

Application Events Provided by the Module:

Event	Activation Condition
SearchCompleteEvent	After the command **EnSearchCompleteEvent** is executed, if any command used for searching stored values is executed, the corresponding notification event is activated when the search operation is completed.
ValReadyEvent	After the command **EnValReadyEvent** is activated, once the measurement operation is completed, the corresponding event will be activated.

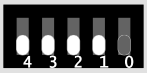
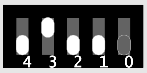
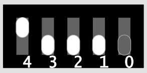
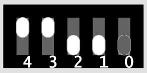
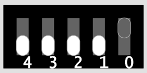
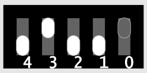
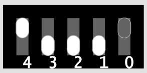
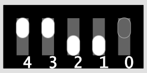
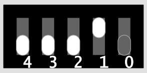
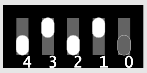
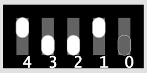
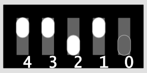
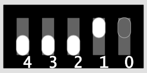
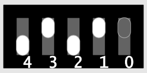
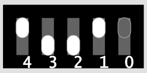
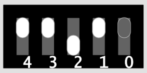
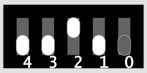
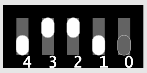
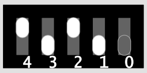
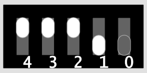
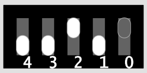
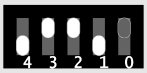
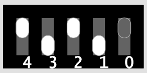
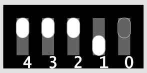
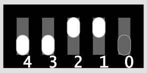
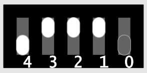
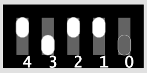
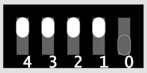
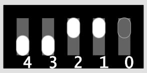
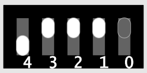
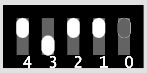
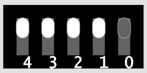
Demonstration Program:

```
Peripheral myC As ColorRGB @ 0                    '    Set the module ID as 0

Dim g_wC, g_wR, g_wG, g_wB As Word                '    Set the parameters for storing the sensed values

Sub Main()
    Dim bStatus As Byte                           '    Set the parameters for storing the status values
    Dim cTemp As Byte                             '    Set the characters for temporary storage

    Debug CLS, "Turn on the red light"
    myC.SetLEDRGB(20, 0, 0)                       '    Set the module to turn on only the red light of the LED
    myC.TurnOnLED()                               '    Turn on the LED
    Pause 3000                                    '    Pause for three seconds
    myC.TurnOffLED()                              '    Turn off the LED

    Debug CLS, "Turn on the green light"
    myC.SetLEDRGB(0, 20, 0)                       '    Set the module to turn on only the green light of the LED
    myC.TurnOnLED()                               '    Turn on the LED
    Pause 3000                                    '    Pause for three seconds
    myC.TurnOffLED()                              '    Turn off the LED

    Debug CLS, "Turn on the blue light"
    myC.SetLEDRGB(0, 0, 20)                       '    Set the module to turn on only the blue light of the LED
    myC.TurnOnLED()                               '    Turn on the LED
    Pause 3000                                    '    Pause for three seconds
    myC.TurnOffLED()                              '    Turn off the LED
    Debug CLS

    myC.LoadLED()                                 '    Adjust the light intensity of the LED as the default value

    myC.SetPeriod(0)                              '    Set the measurement time
    myC.SetGain(0)                                '    Set the measurement scaling factor
    myC.SetPrescalar(0)                           '    Set the measurement rescaling factor

    myC.DisValReadyEvent()                        '    Disable notification for the sensor update
    myC.StartMeasure()                            '    Start measuring the color values

'----------------------------------------------------------------------------
'    The Do Loop continues the measurement until the status value is detected.
'----------------------------------------------------------------------------
    Do
        bStatus = myC.GetCRGB(g_wC, g_wR, g_wG, g_wB)   '    Read the sensed color values
    Loop Until bStatus=1

    Debug "C: ", %DEC5R g_wC, CR                  '    Display the total light intensity
    Debug "R: ", %DEC5R g_wR, CR                  '    Display the value of the red color
    Debug "G: ", %DEC5R g_wG, CR                        '       Display the value of the green color
    Debug "B: ", %DEC5R g_wB, CR                '       Display the value of the blue color

    Debuging "Enter any value to start repeated measurement", %CHR cTemp
    Debug CSRXY(1, 5), CLREOL
    myC.EnValReadyEvent()                       '       Enable notification for the sensor update
    myC.StartMeasure()                          '       Start measuring the color values

    Do
        Debug CSRXY(4, 1), %DEC5R g_wC          '       Display the total light intensity
        Debug CSRXY(4, 2), %DEC5R g_wR          '       Display the value of the red color
        Debug CSRXY(4, 3), %DEC5R g_wG          '       Display the value of the green color
        Debug CSRXY(4, 4), %DEC5R g_wB          '       Display the value of the blue color
    Loop
End Sub

Event myC.ValReadyEvent()
    myC.GetClear(g_wC)                          '       Get the total light intensity
    myC.GetRed(g_wR)                            '       Get the value of the red color
    myC.GetGreen(g_wG)                          '       Get the value of the green color
    myC.GetBlue(g_wB)                           '       Get the value of the blue color
    myC.StartMeasure()                          '       Start measuring the color values
End Event
```

Appendix

1. Known problems:

2. List of Configurations for 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