Thermometer A
User's Guide

Temperature and Humidity Sensing Module
Version: V1.1

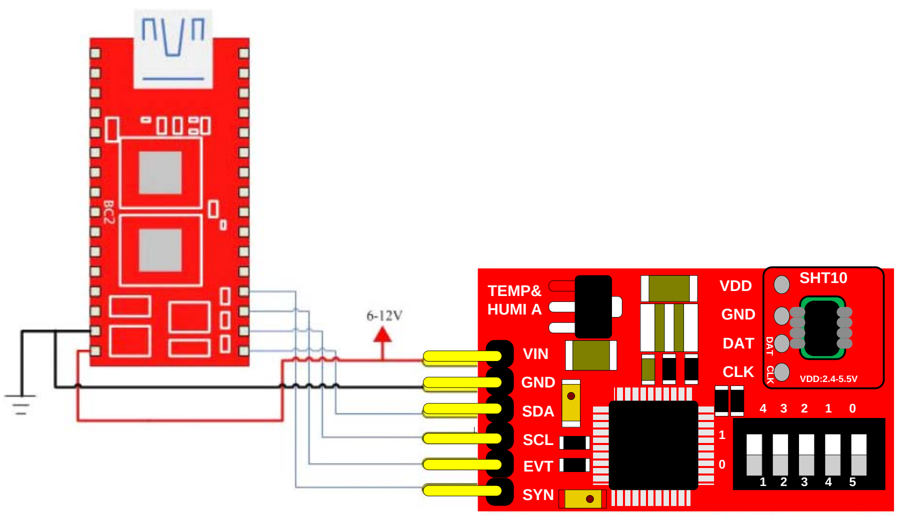
Product Overview: With the connection through cmdBUS to BASIC Commander, the user can use simple commands to obtain the present temperature and humidity from Innovati's Thermometer A module and then calculate the dew point value. Through simple commands, the module can perform functions such as automatic measurements, alarms on changes, etc.

Application:
- Measurement of the change of temperature and humidity.
- Alarm on changes of temperature (humidity) in the environment where the temperature (humidity) needs to be controlled.

Product Features:

- Measurable Temperature Range: -40°C~123.8°C
- It can provide the information of temperature, humidity and dew point.
- Conversion between different temperature units (°K, °F and °C).
- It allows the user to configure the time interval for the automatic storage of temperature, humidity and dew point data. Up to 120 record items for each category can be stored.
- It allows the user to configure the alarms on the ranges of temperature and humidity.
- It allows the user to configure the alarm on changes of temperature and humidity.

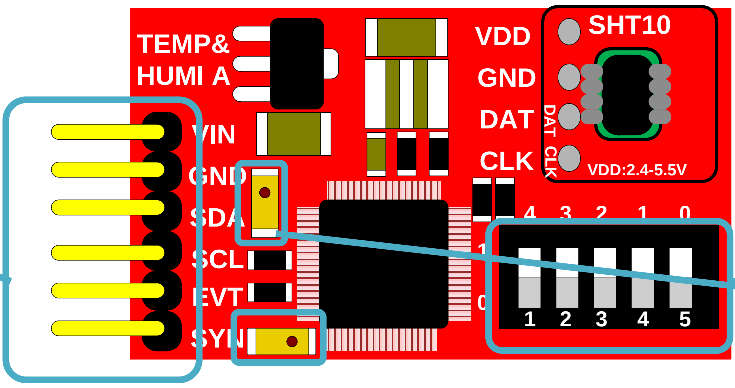
Connection: Change the ID switch to the required number directly, and then connect the cmdBUS to the corresponding pins on the BASIC Commander so that the user can perform the required operations through the BASIC Commander.



Product Specifications:

Pins for cmdBUS: Connect these pins to the corresponding pins on the BASIC Commander for controlling the Thermometer A 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.)

Module ID Setting Switch: The module ID of the Thermometer A 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 1).



From the top to the bottom:
Command Indicator: The blinking light indicates the module and the SBC are transmitting/receiving data.
Event Indicator: The blinking light indicates that the module generates an event.

Description of pin assignments of the module

Precautions for Operations:
Operating Temperature of the Module: -40 °C~ 123.8°C
Storage Temperature of the Module: -40 °C ~125°C

Commands And Events:
The following list shows the commands dedicated for controlling the Thermometer A module. The command name and parameters that should be input are shown in bold or bold-italic typefaces. The words in bold typeface should not be changed while being input. The user can fill the words in bold-italic typefaces with parameters in the properly defined format. 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 Thermometer A, please define the corresponding parameters and the module ID at the beginning of the command, for example:
Peripheral *ModuleName* As ThermometerA @ *ModuleID*

Command Format	Command Function
Commands for Settings	
SetHeaterStatus(*Status*)	Set the value of ***Status*** to specify the status of the heater on the temperature/humidity sensor IC.(The default value is 0) *Note 1 0: The heater is turned off 1: The heater is turned on
Commands for Retrieving Data	
GetTemp10F(*Temp10F*)	Get the updated temperature value and store it in ***Temp10F***. The returned value is an integer, which is 10 times the temperature value in Fahrenheit. The returned value of ***Temp10F*** is within -400~2549.
GetTempF(*TempF*)	Get the updated temperature value and store it in ***TempF***. The returned value is a floating-point value in Fahrenheit.The returned value of ***Temp10F*** is within -40~254.9.
GetHumi10(*Humi10*)	Get the updated humidity value and store it in ***Humi10***. The returned value is an integer which is 10 times the measured humidity value. The returned value of ***Humi10*** is within 0~1000.
GetHumi(*Humi*)	Get the updated humidity value and store it in ***Humi***. The returned value is a floating-point value. The returned value of ***Humi*** is within 0~100.
GetDewpoint10F(*Dewpoint10F*)	Get the updated dew point value and store it in ***Dewpoint10F***. The returned value is an integer which is 10 times the temperature value in Fahrenheit.*Note 2
GetDewpointF(*DewpointF*)	Get the updated dew point value and store it in ***DewpointF***. The returned value is a floating-point value in Fahrenheit.
***Status* = GetHeaterStatus()**	Get the status of the heater on the temperature/humidity sensor IC. The returned value represents the two

	possible states which is stored in the ***Status*** register. 0: The heater is turned off 1: The heater is turned on
***Status* = GetReloadStatus()**	Get the calibration setting status on the temperature/humidity sensor IC. The returned value represents the two possible states which is stored in the ***Status*** register. 0: Use the calibration data stored in OTP as the initial value. 1: The calibration data in OTP is not used.
Commands for the Conversion Between Temperature Formats	
ConvertC2F(*TempC*,*TempF*)	Convert the value of ***TempC***(°C) into °F and store it in ***TempF*** which will be an floating-point value.The values of ***TempC*** and ***TempF*** are 32-bit single-precision floating-point values. It is recommended that the input value should be within -40~123.8.
ConvertF2C(*TempF*,*TempC*)	Convert the value of ***TempF***(°F) into °C and store it in ***TempC*** which will be an floating-point value. The values of ***TempC*** and ***TempF*** are 32-bit single-precision floating-point values. It is recommended that the input value should be within -40~254.9.
ConvertK2F(*TempK*,*TempF*)	Convert the value of ***TempK***(°K) into °F and store it in ***TempF*** which will be an floating-point value.The values of ***TempK*** and ***TempF*** are 32-bit single-precision floating-point values. It is recommended that the input value should be within 233.15~396.9.
ConvertF2K(*TempF*,*TempK*)	Convert the value of ***TempF***(°F) into °K and store it in ***TempK*** which will be an floating-point value.The values of ***TempK*** and ***TempF*** are 32-bit single-precision floating-point values. It is recommended that the input value should be within -40~254.9.
Convert10C2F(*Temp10C*,*Temp10F*)	Convert the value of ***Temp10C***(°C) into °F and store it in***Temp10F***. The returned value will be an integer 10 times of the temperature value. It is recommended that the input value of ***Temp10C*** should be within 400~1238.
Convert10F2C(*Temp10F*,*Temp10C*)	Convert the value of ***Temp10F***(°F) into °C and store it in***Temp10C***. The returned value will be an integer 10 times of the temperature value. It is recommended that the input value of ***Temp10F*** should be within 400~2549.
Convert10K2F(*Temp10K*,*Temp10F*)	Convert the value of ***Temp10K***(°K) into °F

	and store it in***Temp10F***. The returned value will be an integer 10 times of the temperature value. It is recommended that the input value of ***Temp10K*** should be within 2331~3969.
Convert10F2K(*Temp10F*,*Temp10K*)	Convert the value of ***Temp10F***(°F) into °K and store it in***Temp10K***. The returned value will be an integer 10 times of the temperature value. It is recommended that the input value of ***Temp10F*** should be within 400~2549.
Commands for Automatic Recording	
SetRecordTempCnt(*Cnt*)	Set the number of counts for automatic recording the temperature measurements. The system measures the temperature and humidity approximately one time per second. ***Cnt***=0: Record the temperature value for every measurement count. ***Cnt***=1: Record the temperature value for every other measurement count. ***Cnt***=2: Record one temperature value for every three measurement counts. … ***Cnt***=65535: Record the one temperature value for 65536 measurement counts. The module can store up to 120 values items. Once the number of records exceeds 120, the first record will be overridden.
SetRecordHumiCnt(*Cnt*)	Set the number of counts for automatic recording the humidity measurements. The system measures the temperature and humidity approximately one time per second. ***Cnt***=0: Record the humidity value for every measurement count. ***Cnt***=1: Record the humidity value for every other measurement count. ***Cnt***=2: Record one humidity value for every three measurement counts. … ***Cnt***=65535: Record one humidity value for every 65536 measurement counts. The module can store up to 120 values items. Once the number of records exceeds 120, the data starting from the 1^{st} item will be overridden.
SetRecordDewCnt(*Cnt*)	Set the number of counts for automatic recording the dew point measurements. The system measures the temperature and humidity approximately one time per second.

	Cnt=0: Record the dew point value for every measurement count. ***Cnt***=1: Record the dew point value for every other measurement count. ***Cnt***=2: Record one dew point value for every three measurement counts. … ***Cnt***=65535: Record one dew point value for every 65536 measurement counts. The module can store up to 120 values items. Once the number of records exceeds 120, the data starting from the 1st item will be overridden.
GetRecordTempCnt(*Cnt*)	Get the number of counts for automatic temperature recording and store it in ***Cnt***. The returned value will be within 0~65535.
GetRecordHumiCnt(*Cnt*)	Get the number of counts for automatic humidity recording and store it in ***Cnt***. The returned value will be within 0~65535.
GetRecordDewCnt(*Cnt*)	Get the number of counts for automatic dew point recording and store it in ***Cnt***. The returned value will be within 0~65535.
StartAutoTempRecord()	Start the automatic recording for the temperature measurements. When this operation is re-started, the record will be stored from the first memory address and the original data will be overridden.
StartAutoHumiRecord()	Start the automatic recording for the humidity measurements. When this operation is re-started, the record will be stored from the first memory address and the original data will be overridden.
StartAutoDewRecord()	Start the automatic recording for the dew point measurements. When this operation is re-started, the record will be stored from the first memory address and the original data will be overridden.
StopAutoTempRecord(*Cnt,Over*)	Stop the automatic recording for the temperature measurements and return the number of recorded items in ***Cnt***. ***Cnt***: When ***Over***=0, the returned value is the current number of records. When ***Over***=1, the returned value is the address of the last recorded value. ***Over***=0: The number of temperature records is less than 120. ***Over***=1: The number of temperature records is more than 120.
StopAutoHumiRecord(*Cnt,Over*)	Stop the automatic recording for the humidity measurements and return the number of recorded items in ***Cnt***. ***Cnt***: When ***Over***=0, the returned value

	will be the present number of humidity record items. When ***Over***=1, the returned value will be the address of the last recorded item. ***Over***=0: The number of humidity records is less than 120. ***Over***=1: The number of humidity records is more than 120.
StopAutoDewRecord(*Cnt,Over*)	Stop the automatic recording for the dew point measurements and return the number of recorded items in ***Cnt***. ***Cnt***: When ***Over***=0, the returned value will be the present number of dew point record items. When ***Over***=1, the returned value will be the address of the last recorded item. ***Over***=0: The number of dew point records is less than 120. ***Over***=1: The number of dew point records is less than 120.
GetSaveTemp10F(*Num,Temp10F*)	Get the recorded temperature value stored in the address specified by ***Num*** and then return the converted integer value which is 10 times the original value in Fahrenheit. ***Num***=0: Get the oldest temperature record. ***Num***=1: Get the oldest temperature record. ***Num***=2: Get the second temperature record. … ***Num***=120: Get the last temperature record. If the value of ***Num*** is larger than 120, the returned value will be 0.
GetSaveTempF(*Num,TempF*)	Get the recorded temperature value stored in the address specified by ***Num*** and then return the converted floating-point value in Fahrenheit. ***Num***=0: Get the oldest temperature record. ***Num***=1: Get the oldest temperature record. ***Num***=2: Get the second temperature record. … ***Num***=120: Get the last temperature record. If the value of ***Num*** is larger than 120, the returned value will be 0.
GetSaveHumi10(*Num,Humi10*)	Get the recorded humidity value stored in

	the address specified by ***Num*** and then return the converted integer value, which is 10 times the original value. ***Num***=0: Get the oldest humidity record. ***Num***=1: Get the oldest humidity record. ***Num***=2: Get the second humidity record. … ***Num***=120: Get the last humidity record. If the value of ***Num*** is larger than 120, the returned value will be 0.
GetSaveHumi(*Num,Humi*)	Get the recorded humidity value stored in the address specified by ***Num*** and then return the converted float-point value of the humidity. ***Num***=0: Get the oldest humidity record. ***Num***=1: Get the oldest humidity record. ***Num***=2: Get the second humidity record. … ***Num***=120: Get the last humidity record. If the value of ***Num*** is larger than 120, the returned value will be 0.
GetSaveDew10F(*Num,Dew10F*)	Get the recorded dew point value stored in the address specified by ***Num*** and then return the converted integer value which is 10 times the original value in Fahrenheit. ***Num***=0: Get the oldest dew point record. ***Num***=1: Get the oldest dew point record. ***Num***=2: Get the second dew point record. … ***Num***=120: Get the last dew point value. If the value of ***Num*** is larger than 120, the returned value will be 0.
GetSaveDewF(*Num,DewF*)	Get the recorded dew point value stored in the address specified by ***Num*** and then return the converted floating-point value in Fahrenheit. ***Num***=0: Get the oldest dew point record. ***Num***=1: Get the oldest dew point record. ***Num***=2: Get the second dew point record. … ***Num***=120: Get the last dew point value. If the value of ***Num*** is larger than 120, the returned value will be 0.
Commands for Application Events	
SetTemp10FAlarmEvent(*Num,Low,High*)	Set the temperature range alarm event ***Num***: Set the event No. The input value should be within 0~7. ***Low***: A value in units of 10 times the Fahrenheit value.

	High: A value in units of 10 times the Fahrenheit value. Once the measured temperature is greater or equal to the value of ***Low*** and smaller or equal to the value of ***High***, the corresponding event is activated. Event Send: ***Low*** ≤ Tcur ≤ ***High*** Allowed input temperature range: -40~254.9°F*10 If the input value exceeds the allowed range or if the input event No. exceeds the allowed range, the command is regarded as invalid and no operation will be performed. * The temperature range events which are activated both by the **integer value 10 times the original value and the floating-point value** share the same event setting which allows the user to configure the event with different temperature options.
SetTempFAlarmEvent(*Num,Low,High*)	Set the temperature range alarm event ***Num***: Set the event No. The input value should be within 0~7. ***Low***: A floating-point value in units of Fahrenheit. ***High***: A floating-point value in units of Fahrenheit. Once the measured temperature is greater or equal to the value of ***Low*** and smaller or equal to the value of ***High***, the corresponding event is activated. Event Send: ***Low*** ≤ Tcur ≤ ***High*** Allowed input temperature range: -40~254.9°F If the input value exceeds the allowed range or if the input event No. exceeds the allowed range, the command is regarded as invalid and no operation will be performed. * The temperature range events which are activated both by the **integer value 10 times the original value and the floating-point value** share the same event setting which allows the user to configure the event with different temperature options.
SetTempChangeEvent(*Scale*)	Set the value of temperature change for the

	temperature change event. ***Scale***: An integer in Fahrenheit within 1~255. If the input value exceeds the allowed temperature range, the command is regarded as invalid. After the temperature change event is activated, the first temperature value measured will be used as the baseline value. If the difference between the updated temperature measurement value and the baseline value exceeds the value of ***Scale***, the event is activated. Meanwhile, the temperature value measured at the time the event is activated will be used as the new baseline value.
SetHumi10AlarmEvent(*Num,Low,High*)	Set the humidity range alarm event ***Num***: Set the event No. The input value should be within 0~7. ***Low***: A value in units of 10 times the humidity ***High***: A value in units of 10 times the humidity Once the measured humidity is greater or equal to the value of ***Low*** and smaller or equal to the value of ***High***, the corresponding event is activated. Event Send: ***Low*** ≤ Hcur ≤ ***High*** Allowed input humidity range: 0~100%*10 If the input value exceeds the allowed range or if the input event No. exceeds the allowed range, the command is regarded as invalid and no operation will be performed. * The humidity range events which are activated both by the **integer value 10 times the original value and the floating-point value** share the same event setting which allows the user to configure the event with different humidity options.
SetHumiAlarmEvent(*Num,Low,High*)	Set the humidity range alarm event ***Num***: Set the event No. The input value should be within 0~7. ***Low***: A floating-point value in units of %. ***High***: A floating-point value in units of %. Once the measured humidity is greater or equal to the value of ***Low*** and smaller or equal to the value of ***High***, the corresponding event is activated.

	Event Send: ***Low*** ≤ Hcur ≤ ***High*** Allowed input humidity range: 0~100% If the input value exceeds the allowed range or if the input event No. exceeds the allowed range, the command is regarded as invalid and no operation will be performed. * The humidity range events which are activated both by the **integer value 10 times the original value and the floating-point value** share the same event setting which allows the user to configure the event with different humidity options.
SetHumiChangeEvent(*Scale*)	Set the value of humidity change for the humidity change event. ***Scale***: An integer for the humidity within 1~100. If the input value exceeds the allowed humidity range, the command is regarded as invalid. After the humidity change event is activated, the first humidity value measured will be used as the baseline value. If the difference between the updated humidity measurement value and the baseline value exceeds the value of ***Scale***, the event is activated. Meanwhile, the humidity value measured at the time the event is activated will be used as the new baseline value.
GetTemp10FAlarmEvent(*Num,Low,High*)	Get the range values for the event specified by ***Num***, convert them into the values 10 times the original temperature value in Fahrenheit and store them in to ***Low*** and ***High*** respectively.
GetTempFAlarmEvent(*Num,Low,High*)	Get the range values for the event specified by ***Num***, convert them into the floating-point values in Fahrenheit and store them in to ***Low*** and ***High*** respectively.
GetTempChangeEvent(Scale)	Get the value of the temperature change for the event in Fahrenheit and store it in ***Scale****.*
GetHumi10AlarmEvent(*Num,Low,High*)	Get the range values for the event specified by ***Num***, convert them into the values 10 times the original humidity value in and store them in to ***Low*** and ***High*** respectively.
GetHumiAlarmEvent(*Num,Low,High*)	Get the range values for the event specified by ***Num*** in floating-point and store them in ***Low*** and ***High*** respectively.
GetHumiChangeEvent(*Scale*)	Get the value of the humidity change and

	store it in ***Scale***.
EnableTempAlarmEvent(*Num*)	Enable the temperature range alarm event specified by ***Num***.
EnableTempChangeEvent( )	Enable the temperature change event.
EnableHumiAlarmEvent(*Num*)	Enable the humidity range alarm event specified by ***Num***.
EnableHumiChangeEvent( )	Enable the humidity change event.
EnableDewReachEvent( )	Enable the dew point reach event.
DisableTempAlarmEvent(*Num*)	Disable the temperature range alarm event specified by ***Num***.
DisableTempChangeEvent( )	Disable the temperature change event.
DisableHumiAlarmEvent(*Num*)	Disable the humidity range alarm event specified by ***Num***.
DisableHumiChangeEvent( )	Disable the humidity change event.
DisableDewReachEvent( )	Disable the dew point reach event.
***Status* = GetTempAlarmStatus( )**	Get the temperature range alarm event and clear the status into 0 after the command is executed. Each bit of the value of ***Status*** represents the status of the corresponding event No. 0~7. 00000001b(01H): Event No. 0 is activated. 00000010b(02H): Event No. 1 is activated. 00000100b(04H): Event No. 2 is activated. 00001000b(08H): Event No. 3 is activated. 00010000b(10H): Event No. 4 is activated. 00100000b(20H): Event No. 5 is activated. 01000000b(40H): Event No. 6 is activated. 10000000b(80H): Event No. 7 is activated. … 11111111b(FFH): All the events (0~7) are activated.
***Status* = GetHumiAlarmStatus( )**	Get the humidity range alarm event and clear the status into 0 after the command is executed. Each bit of the value of ***Status*** represents the status of the corresponding event No. 0~7. 00000001b(01H): Event No. 0 is activated. 00000010b(02H): Event No. 1 is activated. 00000100b(04H): Event No. 2 is activated. 00001000b(08H): Event No. 3 is activated. 00010000b(10H): Event No. 4 is activated. 00100000b(10H): Event No. 5 is activated. 01000000b(10H): Event No. 6 is activated. 10000000b(80H): Event No. 7 is activated. … 11111111b(FFH): All the events (0~7) are activated.
***Status* = GetDewReachStatus()**	Get the status of the dew point reach event. ***Status***=0: The dew point reach event is not activated.

	Status=1: The dew point reach event is activated.
EnableRecTOverEvent( )	Enable the alarm event to notify that the number of recorded temperature measurement values exceeds the maximum number of records. After this command is executed, when the number of records for the automatic recording operation reaches 120, the event is activated.
EnableRecHOverEvent( )	Enable the alarm event for notifying that the number of recorded humidity measurement values exceeds the maximum number of records. After this command is executed, when the number of records for the automatic recording operation reaches 120, the event is activated.
EnableRecDOverEvent( )	Enable the alarm event for notifying that the number of recorded dew point measurement values exceeds the maximum number of records. After this command is executed, when the number of records for the automatic recording operation reaches 120, the event is activated.
DisableRecTOverEvent( )	Disable the alarm event for notifying that the number of recorded temperature measurement values exceeds the maximum number of records.
DisableRecHOverEvent( )	Disable the alarm event for notifying that the number of recorded humidity measurement values exceeds the maximum number of records.
DisableRecDOverEvent( )	Disable the alarm event for notifying that the number of recorded dew point measurement values exceeds the maximum number of records.
EnableRefreshEvent()	Enable the measurement refresh event.
DisableRefreshEvent ()	Disable the measurement refresh event.
***Status* = GetRefreshStatus( )**	Get the status of the measurement refresh event. The value of ***Status*** will become 1 every time the measurement is updated and becomes 0 after the user performs the **GetRefreshStatus()** command.

Table 1 :Command Table

Event	Activation Condition
TempAlarmEvent	After the command **EnableTempAlarmEvent** (Num) is executed, when the measured temperature is within the range specified by **SetTemp10FAlarmEvent(*Num,Low,High*)** or **SetTempFAlarmEvent(*Num,Low,High*)**, the temperature range alarm event will be activated. If the input value of Num is not within 0~7, the command is regarded as invalid.
TempChangeEvent	After the command **EnableTempChangeEvent** () is executed, if the temperature change meets the condition specified by the command **SetTempChangeEvent(*Scale*)**, the temperature change event will be activated.
HumiAlarmEvent	After the command **EnableHumiAlarmEvent** (Num) is executed, when the measured temperature is within the range specified by **SetHumi10AlarmEvent(*Num,Low,High*)** or **SetHumiAlarmEvent(*Num,Low,High*)**, the humidity range alarm event will be activated. If the input value of Num is not within 0~7, the command is regarded as invalid.
HumiChangeEvent	After the command **EnableHumiChangeEvent** () is executed, if the temperature change meets the condition specified by the command **SetHumiChangeEvent(*Scale*)**, the humidity change event will be activated.
DewpointReachEvent	After the command **EnableDewReachEvent( )** is executed, when the dew point is reached, the event will be activated.
RefreshEvent	After the command **EnableRefreshEvent( )** is executed, if the measurement is updated, the event will be activated.
RecTOverEvent	After the command **EnableRecTOverEvent( )** is executed, when the number of records for the automatic recording operation reaches 120, the event will be activated.
RecHOverEvent	After the command **EnableRecHOverEvent( )** is executed, when the number of records for the automatic recording operation reaches 120, the event will be activated.
RecDOverEvent	After the command **EnableRecDOverEvent( )** is executed, when the number of records for the automatic recording operation reaches 120, the event will be activated.

Table 2 :Event Table

Note 1:
When the ambient humidity is higher than 95%, the heater can be turned on to avoid the condensation on the sensor and thus improve the response time and accuracy of the sensor.

Notices for Operations:

When the heater is turned on, the sensed temperature will be increased by 5°C (9°F) and the humidity will be decreased.

The power consumption will be increased to 8 mA@5V.

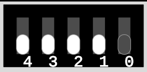
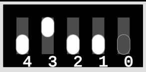
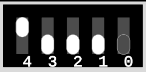
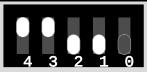
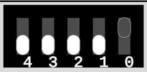
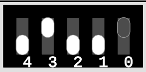
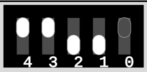
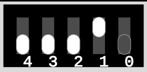
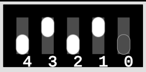
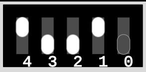
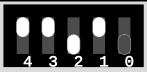
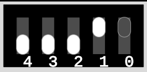
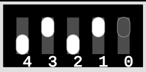
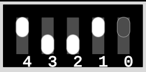
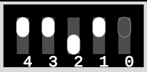
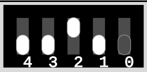
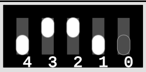
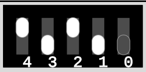
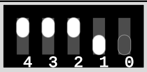
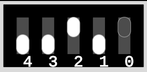
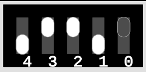
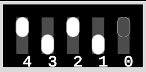
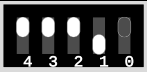
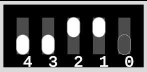
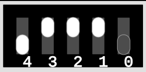
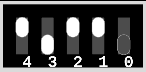
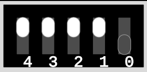
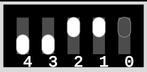
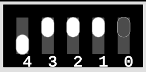
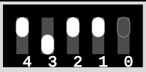
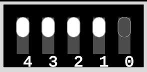
Note 2:
When the air temperature (T) is known and the relative humidity (RH) is higher than 50%, the dew point (Td) can be determined from the following equation:
Td = T-(100-RH)/5

Demonstration Program:

```
Peripheral MyTh As ThermometerA @ 0          'Set the module ID as 0

Dim fl_Temp As Float                         'Store the retrieved temperature data
Dim fl_Humi As Float                         'Store the retrieved humidity data
Dim fl_Dewp As Float                         'Store the dew point data
Dim Status As Byte                           'Store the event No.

Sub Main()
    MyTh.SetTempFAlarmEvent(0,-40,73)        'Set the range values for the temperature range event No. 0
                                             'which are within -40~73F
    MyTh.SetTempFAlarmEvent(1,104,254)       'Set the range values for the temperature range event No. 1
                                             'which are within -40~73F
    MyTh.EnableTempAlarmEvent(0)             'Enable the temperature range event No. 0
    MyTh.EnableTempAlarmEvent(1)             'Enable the temperature range event No. 1

    Do
        Debug CSRXY(1,1),"Present Temperature (C)："
        MyTh.Gettempf(fl_Temp)                   'Get the temperature value
        MyTh.ConvertF2C(fl_Temp,fl_Temp)         'Convert the temperature value in F into the value in C
        Debug CLREOL,fl_Temp

        Debug CSRXY(1,2),"Present Humidity:"
        MyTh.GetHumi(fl_Humi)                    'Get the humidity value
        Debug CLREOL,fl_Humi

        Debug CSRXY(1,3),"Present Dew Point:"
        MyTh.GetDewpointF(fl_Dewp)               'Get the dew point
        MyTh.ConvertF2C(fl_Dewp,fl_Dewp)         'Convert the temperature value in F into the value in C
        Debug CLREOS,fl_Dewp
                                                 Pause 1000
    Loop
End Sub

Event MyTh.TempAlarmEvent()
    Status = Myth.GetTempAlarmStatus()       'Get the event No.

    If Status = 1 Then
        Debug CLS,"The temperature is lower than 23(c)!!"
        MyTh.Gettempf(fl_Temp)
        MyTh.ConvertF2C(fl_Temp,fl_Temp)
        Debug "(",fl_Temp,"C)"
        Pause 1500
    Elseif Status = 2 Then
        Debug CLS,"The temperature is higher than 40(c)!!"
        MyTh.Gettempf(fl_Temp)
        MyTh.ConvertF2C(fl_Temp,fl_Temp)
        Debug "(",fl_Temp,"C)"
        Pause 1500
    End If
End Event
```

Appendix

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