

# **HomeVision Auto Report Feature**

## **1.0 INTRODUCTION**

The HomeVision controller can automatically report state changes to the PC. It can automatically report changes of X-10 devices, input ports, output ports, flags, variables, analog inputs, and digital temperature sensors. It is designed for use with future PC software that needs to keep up-to-date with controller status. You should only use this if you're using software designed for this, or if you're writing your own software.

## **2.0 USAGE**

To use the auto report feature, you must first enable it. This is done with the Controller Settings screen under the Configure menu, using the PC Communications tab. You can individually enable or disable reporting of each object type. You must then download the schedule for any changes to take affect.

You should note several things about how this auto report mode works:

- The controller will report whenever an object changes state, regardless of what caused the change. For example, changes in any of the following ways will be reported:
  - The controller receives an X-10 signal such as A-1 ON or B-All Lights Off
  - A command in the scheduled causes the controller to send an X-10 signal
  - The user transmits an X10 signal using the PC software's X10 control screen or the TV screen
  - The user changes a flag or variable using the PC software's flag and variable control screen or the TV screen
  - An input port changes
  - Any other reason
- When an object changes state, the new states of all objects of the same type are reported in a single message. For example, if you have 20 flags in your schedule, and one of flags changes state, the serial message will show the new state of all 20 flags.
- The controller will not transmit more than one message of each type during each "loop" the controller runs. For example, if you have a macro that changes the value of 5 variables, the controller will report only one message showing the new value of all variables. It will not send 5 separate messages. This approach is used to minimize serial communications, while still keeping the PC software aware of any changes.
- The controller will sometimes send a report even if no object changed state. This can occur if a command in your schedule is performed, but the object is already in the commanded state. For example, if a flag is currently SET, and your schedule performs a command to SET it, the flag doesn't change state. However, the controller will still send an auto report. In general, the controller sends a report if a command to change an object's state is encountered, regardless of whether the object actually changes state. This normally doesn't pose any problem, but just means some unnecessary messages will be sent. However, this will be a problem if your schedule performs such commands repeatedly, such as in a periodic event running every loop, as discussed next.

- Your schedule should not perform certain commands repeatedly, or else serial messages will be transmitted continuously. This problem mainly occurs in a periodic event running every loop. For example, if a periodic event has a command to SET a flag, and it does so every loop, you'll get a new message every loop. The same thing will happen if you have a variable that increments every loop. This situation is easy to observe, as you simply have to open the terminal emulator screen and you'll see serial messages filling the screen. It's OK to use periodic events running every loop, but the actions should be inside an If-Then statement. Assuming the If-Then statement is only true occasionally (and not continually), you won't get repeated messages.
- If you're only interested in certain types of objects like X-10 devices, you should disable reporting of the others. This will prevent unneeded serial messages from being transmitted.

### 3.0 **MESSAGE FORMATS**

Serial reporting messages look something like this:

FLAGUPDATE:009001110100      <followed by a carriage return and line feed>

The specific messages are detailed below.

#### 3.1 **Flag Status Message**

Format: FLAGUPDATE:xxx012345678 ....

Where:

- "FLAGUPDATE:" preceeds each flag status report message.
- "xxx" is three ASCII bytes indicating the number of flags to follow.
- Following "xxx" are one byte for each flag. Each byte represents the flag state, as follows:

ASCII 0 = CLEAR  
 ASCII 1 = SET  
 ASCII 2 = NEUTRAL  
 ASCII 3 = ERROR

#### 3.2 **Input Port Status Message**

Format: INPUTUPDATE:xxx012345678 ....

Where:

- "INPUTUPDATE:" preceeds each input port status report message.
- "xxx" is three ASCII bytes indicating the number of input ports to follow.
- Following "xxx" are one byte for each port. Each byte represents the port state, as follows:

ASCII 0 = LOW  
 ASCII 1 = HIGH

#### 3.3 **Output Port Status Message**

Format: OUTPUTUPDATE:xxx012345678 ....

Where:

- "OUTPUTUPDATE:" preceeds each output port status report message.
- "xxx" is three ASCII bytes indicating the number of output ports to follow.
- Following "xxx" are one byte for each port. Each byte represents the port state, as follows:

ASCII 0 = LOW

ASCII 1 = HIGH

### **3.4     Variable Status Message**

Format: VARUPDATE:xxx0011223344 ....

Where:

- "VARUPDATE:" preceeds each variable status report message.
- "xxx" is three ASCII bytes indicating the number of variables to follow.
- Following "xxx" are two bytes for each variable. The two bytes represent the variable value (in ASCII Hex format ranging from 00 to FF).

### **3.5     Analog Status Message**

Format: ANALOGUPDATE:xxx0011223344 ....

Where:

- "ANALOGUPDATE:" preceeds each analog input status report message.
- "xxx" is three ASCII bytes indicating the number of analog inputs to follow.
- Following "xxx" are two bytes for each analog input. The two bytes represent the analog input value (in ASCII Hex format ranging from 00 to FF).

### **3.6     Digital Temperature Sensor Status Message**

Format: DIGITALTEMPUPDATE:xxx0011223344

Where:

- "DIGITALTEMPUPDATE:" preceeds each digital temperature sensor status report message.
- "xxx" is three ASCII bytes indicating the number of sensors to follow.
- Following "xxx" are two bytes for each sensor. The two bytes represent the sensor value (in ASCII Hex format ranging from 00 to FF).

### **3.7     X-10 Status Message**

Format: X10UPDATE:12345678 ..... 256

Where:

- "X10UPDATE:" preceeds each X-10 status report message.
- Following "X10UPDATE:" are 256 bytes, one for each X-10 address (in sequence from A-1 to P-16). See "X-10 Status Value" for an explanation of the meaning of this value.

### **3.8     X-10 House Code Status Message**

Format: X10HCUPDATE:x12345678 .. 16

Where:

- "X10HCUPDATE:" preceeds each X-10 house code status report message.
- "x" is one ASCII byte (a letter A-P) indicating which house code this status report is for.
- Following "x" are 16 bytes, one for each X10 address in the specified house code (in sequence from unit code 1 to 16). See "X-10 Status Value" for an explanation of the meaning of this value.

### 3.9 X-10 Status Value

An X10 status value indicates the current state (On, Off, or Neutral) and level of the address. The controller reports the status values in exactly the same way it stores them internally. This is a rather convoluted format that evolved over the years (due to the development of new X-10 devices with different numbers of light levels, such as PCS lights). Determining the state is simple, but determining the level is more difficult.

The 2 LSBs (bits 0 and 1) of the byte represent the current state, as follows:

<u>Bit 1</u>	<u>Bit 0</u>	<u>Meaning</u>
0	0	OFF
0	1	ON
1	0	NEUTRAL
1	1	ERROR

The MSB (bit 7) of the byte is always a one.

The 5 intermediate bits (bits 2-6) of the byte represent the current level. Rather than try to explain the format, a Visual Basic function "ConvertToLevel" was written. This function takes the status byte as an input and returns the level as a byte value ranging from 0 to 100 (representing percent on). You can call the function like this:

LightLevel = ConvertToLevel(StatusByte)

Where: StatusByte is the single byte value reported by the HomeVision controller.  
LightLevel is the level (0 to 100) returned from the function.

The function "ConvertToLevel" is shown below:

```
=====
Private Function ConvertToLevel(ByteValue As Byte) As Integer
Dim bStoredLevel As Byte, bState As Byte
Dim bRegularLevel As Byte, sTempResult As Single
'Bits: 1-0: 00 = OFF
'           01 = ON
'           10 = NEUTRAL
'           11 = ERROR
'Bits: 6-2: Level (0 - 31):
'STORE PCS LIGHT LEVEL AS FOLLOWS:
'   PRESET DIM      REGULAR LEVEL      STORED LEVEL      DISPLAYED %
'       31              16              10000              100
'       30              15              11111              96
```

'	29	15	01111	93
'	28	14	11110	90
'	27	14	01110	87
'	4	2	10010	15
'	3	2	00010	12
'	2	1	10001	9
'	1	1	00001	6
'	0	0	00000	0

  

```

ByteValue = ByteValue - 128      'bit 7 is always set, so clear it
bState = ByteValue Mod 4         'keep only 2 LSBs for ON/OFF state

If bState = 0 Then               'light is off
    ConvertToLevel = 0           'return level 0 if light is off
ElseIf bState = 1 Then           'light is on, now determine level
    bStoredLevel = ByteValue \ 4 'same as rotating right 2 bits
    If bStoredLevel = 0 Then
        ConvertToLevel = 0       'light is on at level zero (0%)
    ElseIf bStoredLevel = 16 Then
        ConvertToLevel = 100     'light is on at full brightness (100%)
    Else
        'we must determine the level
        bRegularLevel = bStoredLevel Mod 16 'keep only 4 LSBs
        sTempResult = bRegularLevel * 2     'range is now 0, 2, 4, 6, .. 30
        If bStoredLevel < 16 Then           'bit 6 is clear
            sTempResult = sTempResult - 1
        End If
        'sTempResult is now 0, 1, 2, 3, 4, 5, ... 28, 29, 30
        sTempResult = sTempResult / 0.3     'convert from steps to a percent
        ConvertToLevel = CInt(sTempResult)
    End If
Else 'neutral or error
    ConvertToLevel = -1 'indicates neutral or error
End If

End Function
=====

```

#### 4.0 READING STATUS WITHOUT USING THE AUTO REPORT MODE

You can read status from the controller without using the auto report mode if you like. You can send a serial command to the controller at any time requesting it to report status. The controller will report the status for all objects of the requested type. You can use this with the auto report mode disabled to implement a "polled" system where your software requests status whenever it wants, instead of having the controller transmit it automatically.

The serial commands follow the format described in the serial communications protocol files installed in the HomeVision directory. The commands are in the "Controller Commands" section. They are summarized below:

,G65	= Request Status Report of All X-10 Devices
,G66	= Request Status Report of All Flags
,G67	= Request Status Report of All Variables

,G68	= Request Status Report of All Input Ports
,G69	= Request Status Report of All Output Ports
,G6A	= Request Status Report of Controller
,G6B	= Request Status Report of All Analog Inputs
,G6C	= Request Status Report of All Digital Temperature Sensors

The resulting message from the controller will follow the same format as described previously.