

(last updated: May 8, 2008)

Why integrate HomeVision and INSTEON?

This document describes my integration between HomeVision and INSTEON. No individual piece of this is particularly difficult, but there are many steps and there is a bit of a learning curve with INSTEON so please humor me as I write this narrative in part to document this for myself.

As background, I purchased several INSTEON switches primarily because I liked the extra reliability that I couldn't get with X10. I also liked the ability for switches to be able to transmit signals, allowing one switch to control one or more other switches without having to program an event to handle the situation (which still requires the first switch to send a signal of some sort). Consequently I purchased a number of INSTEON switches, used their native capability to control each other, and then enabled their X10 compatibility so that I could also use HomeVision to control them. Generally speaking, this worked, but I still had X10 reliability problems. Ironically enough, I found that INSTEON devices themselves were "absorbing" X10 signals such that my X10 signal problems got worse with each INSTEON switch that I added.

When I learned that INSTEON offered a "Powerline Serial Modem" (Smarthome Item #2412S, <http://www.smarthome.com/2412s.html>) that could interface with INSTEON, I wanted to integrate HomeVision with INSTEON using HomeVision's nice capabilities to send & receive serial data.

***Do not confuse the PowerLinc Modem (PLM) with the Powerline Serial Controller (Sharthome Item #2414S). That serial "Controller" is VERY different, and you need to be a lot more hard-core than I am to program it. I made the mistake of acquiring one of those first, and still have a flat spot from repeatedly hitting my head against the wall.

The INSTEON documentation isn't really intended for the home automation enthusiast, who may not have the relevant skills or experience to program INSTEON devices. Without those skills or experience, it's difficult to allocate a single block of time large enough to make progress. On several occasions I allocated an hour or two of time, only to find that each time I had to re-read the same sections of INSTEON documentation because I couldn't quite remember what I needed from the last time I read something. So let me offer my own version of "INSTEON For Home Automation Enthusiasts/Dummies." If you take the time to read this first, I think it's possible to be productive with just a single hour of time sitting with HomeVision and the INSTEON PLM.

INSTEON For Home Automation Enthusiasts/Dummies

Each INSTEON device has its own ID which is three bytes long and referenced as a sequence of three pairs of hex digits such as 07 9D AF. This INSTEON device ID is a bit like the MAC (Media Access Control) address assigned to computer network interface cards, for those who are familiar with MAC addresses. For those of us who found it easy to remember that X10 address B1 was for the lights in the bedroom and K1 was for the lights in the kitchen (“B” for bedroom, “K” for kitchen), you’re going to need to start taking notes and writing down your INSTEON device IDs.

Unfortunately you cannot set your own device ID, the device ID is not in the manual or printed on the device, and you’re not told what the device ID is when you purchase a switch or any other device. Additionally, the INSTEON PowerLinc Modem (PLM) ignores all the commands from devices that haven’t yet been “paired” with the PLM, making it even more difficult to find the INSTEON device IDs.

For INSTEON devices to communicate with each other or with the PLM, they must be “paired.” That isn’t how INSTEON describes the process, but for anyone who has used a Bluetooth cell phone with a Bluetooth headset or any other Bluetooth device, I personally believe the analogy is helpful. This “pairing” is a good approach that eliminates the security problems of X10, so I’ll take a moment to contrast it to X10.

In the X10 world, if I walked up to your house with my wireless X10 transmitter or plugged an X10 module into an outlet on the outside of your house, I could simply try transmitting an “On” signal to every X10 address and see which of your lights I could control. Since there were only 256 potential addresses, and most people used A1 for something, it was pretty easy to do. Some people even had problems when a close neighbor had X10 devices and the signals would follow the power line between the two houses. This was all possible because “A1” meant the same thing to everyone, and every A1 device would listen & respond to an “A1 On” signal from anyone who knew how to transmit “A1”.

By contrast, INSTEON devices by default won’t respond to anyone. Each INSTEON device maintains a list (which persists even when the power goes) of the other devices to which they should respond, but this list is empty by default. You must add the device ID of other devices to this list by “pairing” them.

If you have two INSTEON switches, you can pair them by holding one switch in the up/on position for 10 seconds, then doing the same with another switch. This tells the second switch to put the ID of the first switch in its database and listen for commands from that first switch. This is analogous to the pairing operation of Bluetooth devices, and analogous to telling your instant message or email client to only receive messages from people you already have in your database of friends or contacts.

While it's easy to hold two switches in the up position for 10 seconds, and that does "pair" two switches, it still doesn't give us the device ID that we need for programming. For that we must listen to the IDs as they are sent across the wire to each other. I'll explain how to deal with that a bit later.

My Basic Approach to Sending INSTEON Commands

Given an INTEON ID, I thought it should be simple enough to use HomeVision to send simple on/off serial commands. Before I even got started, I assumed I'd have code in a macro that looked roughly like this (a command prefaced with > is just pseudo-code):

```
MACRO EVENT # 14 'INSTEON-Turn Device On'
```

- Serial Port: Transmit <start of command to turn on one device>
- Serial Port: Transmit <INSTEON Device ID>
- Serial Port: Transmit <end of command to turn on one device>

Seems simple enough, but I considered myself a programmer at one point in my life (long ago) and I wanted to have something more generic and organized, rather than scattering INSTEON serial commands and device IDs throughout my HomeVision file. I decided the simplest starting point would be to use a macro to keep all the INSTEON IDs in one place, and transmit them based on a simpler convention of just “device #1”, “device #2”, etc. So I created a variable for “INSTEON Device Number” and then the following Macro...

```
MACRO EVENT # 25 'INSTEON-Send Device ID'
```

```
  ; 1=Lamp Module for Testing
  If
    Var #24 (INSTEON Device Number) = 1
  Then
    Serial port 3 (INSTEON PLM): Transmit bytes '07 9D AF'
  End If
  ; 2=Main Room Dimmable Lights (A1)
  If
    Var #24 (INSTEON Device Number) = 2
  Then
    Serial port 3 (INSTEON PLM): Transmit bytes '07 18 D8'
  End If
```

Note that serial port 3 is my HomeVision-Serial add-on, which I simply gave the description, “INSTEON PLM”.

For anyone who already has INSTEON switches that have each been assigned an X10 address, I recommend using “device numbers” that correspond to the X10 address. In adopting this convention, A1 uses a value of zero, A2=1, B1=16, B2=17, C1=32, etc. You can see these numbers listed in the left-most column of the X10 Module screen of HomeVision or HomeVisionXL. Adopting this convention will give at least some meaning to the numbers you use. Having said that, it's easiest to just start with 1, 2, 3, 4, etc. if you haven't already assigned X10 addresses.

Now that I have my macro to send individual device IDs, the pseudo-code to turn on a light becomes:

- Var #24 (INSTEON Device Number) = 1
-
- Serial Port: Transmit <start of command to turn on one device>
- Do Macro #25 (INSTEON-Send Device ID) once
- Serial Port: Transmit <end of command to turn on one device>

INSTEON also allows you to create “groups” of devices, and control them all at once. The X10 equivalent of this is sending a command to all devices within a given house code such as, “Turn on all lights with house code = K” which used to turn on all the lights in my kitchen. Fortunately INSTEON groups are defined with a simple integer ID of 1, 2, 3, etc. so I created another variable “INSTEON Group Number”. As you might expect, the pseudo-code to turn on an entire group of lights will look a lot like the pseudo-code to turn on only one:

```
MACRO EVENT # 10 'INSTEON-Turn Group On'
```

- Var #25 (INSTEON Group Number) = 1
-
- Serial Port: Transmit <start of command to turn on one group>
- Serial Port: Transmit variable #25 (INSTEON Device or Group Num)
- Serial Port: Transmit <end of command to turn on one group>

Note that this time we can simply send the variable itself, rather than having to call a macro to translate our simple “number” into the Device ID.

So far this all seems simple enough, right? If I’d just give you the serial commands to start & finish the macro, you’d be all set, right? That’s what I thought. I will “just give you the serial commands”, but first you have to listen to my narrative so that you’ll understand why I’m going to give you a lot more than “just” the commands for this simple example.

HomeVision Setup/Configuration/Debugging

For starters, use a secondary HomeVision serial port. I was tempted to say that you might be able to get by with only the main serial port, but if you're that good then you don't need to be reading this document of mine. For anyone reading this document, trust me that you'll need a second serial port in order to keep your computer attached to the main serial port in order to test and see debug information in the console window of the software (I used HomeVisionXL). If you have HomeVision-Pro, you already have multiple serial ports. If you have the standard HomeVision, buy a HomeVision-Serial Add-on as long as you haven't already max'ed out your setup.

Once you have your secondary serial port installed and working, open the screen to configure your serial port. In HomeVisionXL this is from the menu, Configure -> Expansion Boards -> Serial Boards. Set the baud rate to 19200 which is the default for the INSTEON Powerline Modem (PLM). Set the timeout to 50ms, check "Enable binary data reception", and set the mode to "Full Duplex (RS232)". If you just purchased a new HomeVision-Serial Add-on, then you probably also need to set your board version to II (2). I also named my second serial port "INSTEON PLM" to make it quicker to identify in the code.

Baud rate: 19200

Timeout: 50ms

Enable binary reception: Checked

Mode: Full Duplex (RS232)

Board Version: II if you just bought a new HV-Serial

Description: INSTEON PLM (you can use anything you want)

Very closely related to the setting of "Enable binary data reception", let me discuss receiving of the serial data with HomeVision and manipulating it with HomeVision variables. All of the INSTEON documentation exclusively uses hex notation for values, and the sample Docklight Scripting project provided with the PLM defaults to showing everything in hex. I saw the HomeVision commands for sending and receiving hex characters and (hastily) thought that was perfect for what I needed. It's not. When you want to read any of the INSTEON serial data, you need to use the HomeVision command, "Put binary value of char x into Result Value". From there, you can assign that result value to a variable. I defined one variable, "Serial Data Current Byte" and always/immediately put the Result Value into that variable to do anything, so you'll almost always see these two lines of code back-to-back in my code:

- Serial Port: Put binary value of char x into Result Value
- Serial Port: Var #19 (Serial Data Current Byte) = Result Value

As you might expect, inspecting each byte of serial data, one at a time using variables, is tedious (at best) for learning how INSTEON commands are received. Consequently I wrote myself a very helpful macro that takes all the binary data which comes in over my

HomeVision-Serial Add-on and then re-transmits it to the main serial port using hex characters. This serves two purposes, first to let me use the console window of the HomeVision software to see what's happening, and second to let me see the transmission in hex so I can match it to the INSTEON documentation. My macro first gets the length of the serial transmission, stores that in a variable, then simply puts each character into the current variable and forwards it as a hex character to the main serial port. The code repeats 16 times for each byte of data, since 16 bytes was enough to handle all the commands which I cared to debug. I'm only going to paste the first two repetitions of the code below. The repetition doesn't look nice, and unnecessarily compares the variable 16 times even if only 2 bytes were received, but it works. I'm also going to give myself a flag to indicate that I want to debug INSTEON commands, so that I can skip calling this entire macro once my system is stable and I don't feel the need to see every command.

```
Serial port 3 (INSTEON PLM): Put number of received characters in
Result Value
Var #18 (Serial Data Length) = Result Value

If
  Var #18 (Serial Data Length) >= 1
Then
  Serial port 3 (HomeVision-Serial #1): Put binary value of char 1
into Result Value
  Var #19 (Serial Data Current Byte) = Result Value
  Serial port 1 (Main serial port): Transmit string ' '
  Serial port 1 (Main serial port): Transmit variable #19 (Serial
Data Current Byte) as 2 hex bytes
End If

If
  Var #18 (Serial Data Length) >= 2
Then
  Serial port 3 (HomeVision-Serial #1): Put binary value of char 2
into Result Value
  Var #19 (Serial Data Current Byte) = Result Value
  Serial port 1 (Main serial port): Transmit string ' '
  Serial port 1 (Main serial port): Transmit variable #19 (Serial
Data Current Byte) as 2 hex bytes
End If
```

These If-Then-End If blocks repeat up to 16 times for 16 bytes of data.

Given all this code in a macro, you still need to call it from the Serial Data Input Event. That event will be called once for each "batch" of serial transmission that is received by HomeVision. This is where I check for my flag that indicates whether or not I want to call this macro to provide all this extra information for debugging.

INSTEON Device IDs

I previously mentioned that INSTEON Device IDs are three bytes long, typically referenced in hex (07 9D AF), and you're not told what the device ID is. This was my biggest hurdle to overcome in getting the integration to work. I wrote (and am providing to you) a lot of HomeVision code to receive and analyze a lot of INSTEON serial data that isn't absolutely necessary if you can find and manipulate INSTEON device IDs another way. There are in fact other ways to do this, which I'll start by describing here.

When I got my INSTEON Powerline Modem (PLM) as part of INSTEON's serial developers kit, it came with a "Getting Started" guide that suggested using the Docklight Scripting program for serial communications from a Windows computer. You can also download a sample project for Docklight Scripting that contains a few INSTEON commands pre-programmed and ready to go. Within Docklight Scripting, you can send commands to the PLM and see the responses printed to the screen. Much of the HomeVision code I wrote, particularly in the macro "INSTEON-Serial Receive" does little more than display INSTEON commands in the console window of the HomeVision software. It is these viewings of INSTEON responses that will let you see the device IDs.

As mentioned in my section for "INSTEON for Home Automation Enthusiasts/Dummies", no two INSTEON devices talk to each other without first being paired. This includes the PLM, which doesn't communicate with other devices until they are all paired with the PLM. The simplest, manual method of doing this is to press-and-hold on the set button of the PLM until it starts blinking. This will put the PLM into its linking mode, during which you have about 4 minutes to go to another device and put it into linking mode. With switches, this is done by holding them in the up/on position for about 10 seconds, and with a lamp module this is done by pressing and holding its set button.

There is also a programmatic way to put the PLM into this linking mode, which is nice to have once you've put the PLM in a wiring closet with HomeVision. I've created and provided macros to do exactly that.

STOP TALKING & GET TO THE CODE!!!

HomeVision Flags

Forward All INSTEON Commands

This flag determines whether or not Macro “INSTEON – Translate to Main Port” is called in order to facilitate debugging. I use a flag for this so that I’m not burdening HV with all the extra processing when I don’t need it, but I can change the value without loading a new schedule when I want to see everything.

Prefer INSTEON over X10

This flag determines whether I send X10 commands such as “Transmit House Code ‘K’ All Lights On” to turn on all the kitchen lights, or whether I call Macro “INSTEON-Turn Group On” to turn on a group of lights using INSTEON commands. Think of this as my way to quickly revert to all the old X10 commands if I have a problem.

HomeVision Variables

Serial Data Length

Whenever I receive a serial transmission, the first thing I do is store the length of the total serial transmission. This is because INSTEON often sends two commands in a row without any break between them (remember that you configured the serial port for 50ms timeouts). Knowing the length lets me pseudo-recursively read a second command from the input.

Serial Data Current Byte

Generally speaking, the code always reads the next byte into the Result Value and then sets this variable from the Result Value. There are times when it might be more efficient to read a byte directly into one of the other variables, but using the same variable makes it a lot easier to copy/paste since reading a byte and putting it into a variable happens so often.

INSTEON ID high byte

INSTEON ID middle byte

INSTEON ID low byte

When a command is received which may contain an INSTEON ID, I put the ID values into these three variables so that you don't necessarily have to watch the console to see the device ID. I'm not yet taking any actions based on receiving an event from something else, but this will presumably be more useful for that.

INSTEON ACK/NAK

Whenever INSTEON sends an ACK or NAK to one of my commands, I store the result in this variable so that some other code can easily see if the last command succeeded. I should probably change this to be a flag, so that the rest of the code doesn't have to know what values to use when comparing to this.

INSTEON Device Number

INSTEON Group Number

These variables hold the simple integer values that denote the device or group on which other macros will operate. Fortunately groups are known to INSTEON as simply integers (1, 2, 3, etc.) so no translation is necessary. Device numbers (1, 2, 3, etc.) will always need to be translated into their three-byte device IDs. HomeVisionXL's "Named Constants" is a tremendous help for having these constants labeled in your code.

Macros to Send INSTEON Commands

First of all, I used the word "INSTEON" to name all my related macros, making it easy to sort them by name. That may appear a bit redundant in this document, which only describes INSTEON macros, but it helps (me) when they are in a full schedule with everything else.

INSTEON – Send Device ID

INSTEON - Turn Device On

INSTEON – Turn Device Off

INSTEON – Turn Group On

INSTEON – Turn Group Off

These first 5 macros are the ones needed to turn on/off INSTEON devices individually or in groups. Turning on an individual device requires sending the three byte INSTEON ID. This is done by calling the macro “INSTEON – Send Device ID” which is nothing more than a sequence of If-Then-End If statements that look at the variable “INSTEON – Device Number” and transmit the three byte INSTEON ID over the serial port. This allows all the INSTEON device IDs to be located in only one place.

INSTEON – Get Version

This macro is the INSTEON equivalent of the “Hello World” program in any other programming language. This simple command asks the Powerline Modem (PLM) for it’s information which is returned over the serial port. This is the command I used to test whether or not I properly configured my HomeVision serial port, and whether or not I could send & receive INSTEON data.

INSTEON – Start ALL-Linking Grp

INSTEON – Start Unlinking Grp

INSTEON – Cancel ALL-Linking

These are the commands which let you “pair” INSTEON devices/switches with your PowerLine Modem (PLM) without having to push the set button on your PLM. It’s also more functional, in that pushing the set button will always “pair” something into group #1. You need to use these commands to assign devices into a group other than #1. Before calling one of these commands, you MUST set the variable “INSTEON – Group Number”.

To add a device to a group, set the “INSTEON – Group Number” variable and then call the macro, “INSTEON – Start ALL-Linking Grp”. You now have up to ~4 minutes to walk over to the switch or other device you want to control and take the appropriate action to pair that device. That appropriate action is usually either holding a switch in the up/on position for 10 seconds, or pressing the set button for 3-10 seconds. The LED on the switch or device will usually blink to confirm that it has been “paired” with the PLM.

To unlike a device from the PLM, call the macro “INSTEON – Start Unlinking Grp” rather than the macro to start the linking, but otherwise follow the same basic process outlined above.

After any one device has been linked or unlinked (“paired”), the PLM will automatically go out of the linking mode. The PLM does not support a “multilink” mode like the INSTEON switches do. Instead, we can detect the PLM going out of linking mode, and simply re-send the command automatically if that’s what we want to do (but I haven’t yet done this).

If you execute either of the macros to start linking or unlink but you did not intend to actually link or unlink a device, simply call the macro “INSTEON – Cancel ALL-Linking.”

INSTEON – Get First DB Entry

INSTEON – Get Next DB Entry

These commands are used to inspect the PLM’s internal database/list of INSTEON IDs from devices that have been paired with the PLM. Quite simply, you call “Get First DB Entry” and then repeat calling “Get Next DB Entry” until the command fails. Having said that, for some reason I don’t always see every device ID when I run through this list, which I simply don’t yet understand.

Macros/Code to Receive INSTEON Commands

Serial Data Input for HomeVision-Serial Add-on

This is where it all starts. The event stores the serial command length in the variable “Serial Data Length” and then looks at the first byte of the serial data. All INSTEON commands begin with the byte 0x02, so if the first byte of the serial data is 0x02 the macro “INSTEON – Serial Receive” is called. This keeps this event relatively clean, and keeps all the core INSTEON code together in one place.

INSTEON – Serial Receive

This macro looks at the second byte of each serial transmission, which will contain the INSTEON command. This is done with a brute-force sequence of If-Then-End If statements that simply compare the command byte to each known command that I cared about. Upon receiving a specific command, the subsequent bytes of the serial data can be read and properly interpreted within that specific If-Then-End If statement, because the interpretation of every byte after the 2nd one is different depending upon the command.

INSTEON – Translate to Main Port

This macro is immediately called at the top of “INSTEON – Serial Receive” if the flag “INSTEON – Forward All Commands” is set. It simply reads every byte in the serial data and re-transmits it over the Main Serial Port as a “human readable” hex string. I’m not saying that all humans can read hex, but trust me that a 0x02 is far easier to read than the odd ascii character for the value of 2. All of the INSTEON documentation is given using hex notation, so it really is easy to compare received data with the INSTEON manual with the help of this macro.

INSTEON – Receive Serial Byte 4

The INSTEON Powerline Modem (PLM) often sends two consecutive commands in one batch. I found a few situations in which this occurs after a simple three byte command. Since HomeVision doesn’t really support using a variable to indicate the next binary value to be read, it was easier to simply create a second function that is hard-coded to start looking for an INSTEON command at the fourth byte. If HomeVision could someday support a command such as, “Put binary value of character @var into Result Value” (which would be the binary equivalent of two existing commands, “Put value of character @ var into Result Value” and “Put value of hex character @ var into Result Value”) then we could just use recursion. We would use a variable to indicate the character to be read, and simply increase that after each byte that is read. Right now, the workaround of just having a second macro is easy enough because the ones I cared about always start after at the 4th byte, but if we find that INSTEON commands might truly start at any character, then we’ll need another approach.

NOW SHOW ME THE CODE!!!

Since the macros to receive commands are very, very long, I’ve put the code into another document.

QUESTIONS???

If you have problems, join the HomeVision group on YahooGroups if you haven’t already, and send an email to the group (homevision-users@yahoogroups.com). I won’t be able to respond as quickly as people like Craig and Schelte have in the past, but I’ll do my best.

Tom Carter