HomeVision Operation

Many users have requested a better understanding of how HomeVision operates internally. This article discusses HomeVision's general operations and how it performs time-related events such as timers, periodic events, and scheduled events.

Overview

The HomeVision controller is primarily an "event-driven" controller. It performs the actions you specified for it when specific events occur. However, the controller also provides some features of "loop-type" controllers. An understanding of how these types of systems differ will help explain how HomeVision operates.

In a "loop" system, the user's schedule is essentially a long list of conditions (If-Then statements) and associated actions. The controller continually runs through them in a loop evaluating each condition. If a condition is met, the controller executes the associated actions. These systems can be powerful because you can specify complex combinations of conditions. Loop systems have several drawbacks, however:

- The controller spends most of its time checking conditions that are rarely true. This slows down its response when an input changes or a command is received. The response time may be OK for simple applications, but not for more advanced applications. For example, an infrared remote control used to control a video menu on the TV screen needs a quick response in order to be usable.

- When an If-Then-Else statement is used, either the "Then" or "Else" part will be executed every time through the loop. Thus, some commands will be executed each loop. This is rarely the desired behavior, as most commands only need to be executed once in response to some event. Preventing repeated actions requires cumbersome programming techniques, such as setting a flag after performing an action to indicate that it has already been done.

An "event-driven" system, in contrast, only executes actions when an event occurs. The controller constantly checks for specific events, then immediately performs the actions associated with it. This provides several advantages over a loop-type system:

- The controller can respond much faster to the event.

- Programming is simpler, as the user can easily specify the actions to perform in response to the event.

HomeVision provides the best features of both types of systems. At its heart, it's an event-driven controller that provides the rapid input response typical of such systems. With it, you can develop powerful schedules with only limited programming. However, HomeVision can also execute a group of user-specified commands at a rapid rate (every few milliseconds with a periodic event set to execute "every loop"), simulating the loop-type systems. You can then use combinations of If-Then-Else statements to check for any set of conditions you desire and perform appropriate actions.

Loop Overview

Most of the time it is operating, HomeVision is repeatedly executing a series of instructions in a "loop". External to this loop, "interrupts" occur when important events occur. These interrupts

signal to the main loop that it needs to take certain actions (which are typically to perform user-specified actions).

The main loop instructions perform three general types of functions:

1. **Event checking**. These instructions check to see if certain events have occurred. For example, HomeVision will check for a received X-10 signal, infrared signal, or serial message. It typically takes only a few microseconds to check for this. If no event has occurred, the controller goes on to check other things. If an event has occurred, and the user has specified actions for those events, the controller performs them.

   For the technically inclined reader: HomeVision uses interrupts to receive X-10 and IR signals, as well as serial data over the main HomeVision serial port. The '"receiving" of these messages occurs outside the main loop in interrupt service routines (ISRs). The ISRs save the data and set flags indicating when a complete message has been received. The main loop merely checks the flag to determine if it needs to take action.

2. **Reading other devices**. Devices that aren't read using "interrupts" are read periodically in the main loop. These are primarily input ports and the HomeVision-Serial and HomeVision-Phone accessories. If the controller determines that an important event has occurred, and the user has specified actions for those events, the controller performs them.

3. **Housekeeping**. The main loop also checks for errors, updates the video screen, and performs similar housekeeping operations.

Loop Details

Following is a simplified step-by-step description of what HomeVision does each loop:

1. Check for errors (verify RAM checksum, check for stack overflow, verify 82C55 chip is functioning, check TW523, etc.).
2. Reset watchdog timer.
3. Read the clock chip, count down timers, and perform any associated timer actions.
4. Check for received infrared signals and perform any associated actions.
5. Read input ports and see if any have changed state. If so, perform any associated actions.
6. Check for received serial messages and perform any associated actions.
7. Check for received X-10 signals and perform any associated actions.
8. Run any periodic event set for "every loop".
9. If the clock just changed to a new minute, check for other periodic events that may be due and perform any associated actions.
10. If the clock just changed to a new minute, check for any scheduled events that are due and perform any associated actions.
11. ***Check progress of digital temperature sensor "read" request. If read is complete, update temperature variables and start another "read". If not complete, check for timeout. If timeout, report an error and start another "read".
12. ***Read analog inputs.
13. ***Refresh all output ports (i.e., put them in the correct state). This ensures that if there was previously an intermittent hardware error or noise when setting the port, the port gets set to the correct state as quickly as possible.
14. Read data from HomeVision-Serial devices. If events have occurred, perform any associated actions
15. Read data from HomeVision-Phone device. If events have occurred, perform any associated actions

16. ***If there are serial security systems that must be polled, see if it's time to poll them and do so if necessary.
17. ***If there are serial thermostats that must be polled, see if it's time to poll them and do so if necessary.
18. If the minute has changed, and there are X-10 thermostats that must be polled, see if it's time to poll them and do so if necessary.
19. ***Update the video screen.

*** These actions don't actually run every loop. They run at lower rates, with the rate varying for each action.

As you can see, there are some events that don't run every loop. As a result, the time to complete a loop varies somewhat from loop to loop. The loop time will also be greater if there are accessories attached that require reading each loop. It also depends somewhat on schedule size. Typically, the loop time ranges from about 5 to 30 milliseconds. Very complex schedules might take a little longer, but it's still fast enough to provide virtually instantaneous response to user commands.

Of course, a loop can take longer if user events are required. The length of time required to execute a user command depends on the type of command. Most commands like setting flags, checking variables, and starting a timer take only fractions of a millisecond. However, commands to transmit X-10 and infrared signals take longer. As a result, when these commands are performed, it will take longer to complete the loop. This could cause other actions to be delayed. For example, if the controller is executing a series of X-10 commands that takes 30 seconds to transmit, and an IR signal is received during this time, the controller has to complete the X-10 command and finish the loop before it can respond to the IR signal.

Event Sequence

When more than one of the same event type occur at the same time, they will be performed in numerical sequence. For example, assume you have three scheduled events – numbers 3, 7, and 8, that are each scheduled for 6:00PM. At 6:00PM, event 3 will run first, followed by 7 and then 8. Periodic events are similar, and run in numerical sequence regardless of the event rate. For example, assume you have these three periodic events:

Periodic event #0 – every 1 minute
Periodic event #1 – every 2 hours
Periodic event #2 – every loop

At 2:00AM, all three events will run, and the sequence will be 0, 1, and then 2. At 2:01AM, events 0 and 2 will run, in that order.

If different types of events occur within the same loop (which is essentially the same as saying they occur at the same time), they will run in the order that HomeVision runs the main loop (discussed above). For events involving time, this means expiring timers are run first, followed by periodic events and then scheduled events.

Periodic Events

Events set to run "every loop" run repeatedly at a high rate. As discussed previously, the exact rate depends on your schedule size and the number of accessories you have connected to HomeVision, but typically ranges from every 5 to 30 milliseconds.

All other periodic events run "on the minute" (e.g., as soon as the clock rolls over to the next minute). Events running every 5 or 15 minutes run on the hour and every 5 or 15 minutes after

that. For example, events running every 15 minutes run at 1:00PM, 1:15PM, 1:30PM, 1:45PM, etc. Events running every 2, 4, or 8 hours run at midnight and every 2, 4, or 8 hours after that. For example, events running every 4 hours run at 12:00AM, 4:00AM, 8:00AM, etc. The result is that at 12:00AM, 8:00AM, and 4:00PM, *all* periodic events will run (in numerical order).

Reading Input Ports

As noted previously, HomeVision reads its input ports one time through each "loop". This means the inputs are typically read every 5 to 30 milliseconds. However, when the controller is held up transmitting IR or X-10 signals, there will be a longer gap between consecutive input port reads. As a result, it is possible for a brief input port change to occasionally be missed.

Timers and Delays

Delay

A "delay" is a command to briefly wait before proceeding. When the controller encounters a delay command, it simply waits the specified time before proceeding. During this time, it can still receive X-10 and IR signals, but it will *not* immediately take action on them. *The controller will appear to have stopped running during the delay*. Once the delay ends, the controller performs any actions following the delay command. When those actions are complete, it will then handle any received X-10 or IR signals.

Delays should only be used when you need a short, precise pause before for proceeding. Delays are limited to a maximum of 10 seconds, and you should try to keep them under one second. For longer pauses, use a Wait Timer. A Wait Timer will not halt other controller operations, and is therefore preferred to using a delay.

Timers

Timers can be used in two different modes:

- Standard Timer
- Wait Timer

Any of the 255 timers can be used in either mode. The mode is determined by what command you use to start the timer running. A particular timer should only be used as a Standard Timer *or* a Wait Timer, but not as both in your schedule.

Standard Timers

The following command will set the timer to be a Standard Timer. It will load it with the specified value and start it counting down.

```
Load Timer "XXX" With HH:MM:SS.SS And Start
```

In the standard mode, the timer simply counts down to zero. When it reaches zero, it stops "running" and begins "ringing". If the user defined any actions in the Timer Summary Screen, the actions will be performed when the timer expires.

Wait Timers

A Wait Timer is similar to the Standard Timer, with one important difference. Instead of defining a single set of actions for the entire schedule in the Timer Summary Screen, you specify the desired actions every time you start the timer. For example, consider the following commands:

```
Wait 2 Minutes With Timer #1, Then:
   Set Output Port 1 Low
End Wait
Set Output Port 1 High
```

The Wait command sets up timer #1 as a Wait Timer, loads it with the value “2 minutes”, and then starts it running. It then skips over the Wait Timer actions (the actions between the “Wait” and “End Wait” statements) and performs any following commands in the schedule (”Set Output Port 1 High”, in this case). *Then, the controller continues operating normally until the timer expires*. Other actions can take place during this time, including stopping or restarting the timer. When the timer expires, the controller automatically returns and performs the commands listed between the “Wait” and “End Wait” statements (”Set Output Port 1 Low”, in this case). Thus, the wait actions are performed only one time when the timer expires, not when first encountered in the schedule.

Note that the following sequence of actions has the exact same effect as the above example:

```
Set Output Port 1 High
Wait 2 Minutes With Timer #1, Then:
   Set Output Port 1 Low
End Wait
```

Either way, output 1 is set high and, 2 minutes later, goes low. It doesn’t really matter whether the “Set Output Port 1 High” command is run before or after the Wait command, as the Wait command only takes a fraction of a millisecond to run.

There is one very important thing to understand if you use the same Wait Timer in more than one place in a schedule: *If a wait timer is started again, the new actions will override the previous actions.* In some cases, this may be exactly what you want, but in other cases it could cause unexpected problems.

For example, you *can* “nest” Wait Timers, like this:

```
Wait 5 Minutes With Timer #1, Then:
   Set Flag #3
   Wait 2 Minutes With Timer #1, Then:
      Clear Flag #3
   End Wait
End Wait
```

Five minutes after the Wait command is executed, flag #3 will be set, and then 2 minutes later it will be cleared. Note that the same timer (#1) can be used for both Wait commands. That’s because it’s finished being used the first time (for 5 minutes) before it’s used again (for 2 minutes).

However, you *shouldn’t* do the following:

```
Wait 5 Minutes With Timer #1, Then:
   Set Flag #3
End Wait
Wait 7 Minutes With Timer #1, Then:
   Clear Flag #3
End Wait
```

In this case, the result is to wait 7 minutes, then clear flag #3. What happens is the first Wait command sets timer #1 to 5 minutes and points it to the "Set Flag #3" command. Immediately thereafter, it sets the same timer to 7 minutes and points it to the "Clear Flag #3" command. The first Wait command essentially gets ignored and only the second one is performed, clearing flag #3 after 7 minutes.

You *can* accomplish the same thing as the "nested" example by using two *different* timers (although this wastes an extra timer and may be more confusing than the nesting approach), like this:

```
Wait 5 Minutes With Timer #1, Then:
   Set Flag #3
End Wait
Wait 7 Minutes With Timer #2, Then:
   Clear Flag #3
End Wait
```

In this case, flag #3 will be set after 5 minutes and cleared after 7 minutes. Note that timer #2 expires 7 minutes after the event is run, *not* after 12 minutes. Both timers get started at the same time. Thus, the flag will be set for 2 minutes.

Timer Execution Order

If you have multiple timers running, they will usually expire at different times. Whichever one expires first will be run first. However, in rare situations, it is possible to have two different timers that have their actions performed out of order. This usually isn't a problem, but might sometimes be. Consider this code:

```
Wait 29 seconds With Timer #2, Then:
   Set Output Port 1 High
End Wait
Wait 30 seconds With Timer #1, Then:
   Set Output Port 1 Low
End Wait
```

Normally, timer 2 will expire first and its actions will be performed before those of timer 1. Thus, 29 seconds after this code runs, the output port will go high for 1 second. But now consider this: Timer 2 is 1 second away from expiring, but the controller is performing a series of X-10 signal transmissions that lasts 5 more seconds. By the time the X-10 signals are completed, *both* timers have now expired. The controller will then run them in their numerical sequence. Thus, the timer 1 actions will be performed *before* the timer 2 actions. The result is that the output port goes high and stays high, which is not the desired behavior.

Here's how to avoid this problem:

```
Wait 29 seconds With Timer #1, Then:
   Set Output Port 1 High
   Wait 1 second With Timer #1, Then:
      Set Output Port 1 Low
   End Wait
End Wait
```

Only one timer is used, and when it expires the first time, the output port is set high. Note that in this example, where X-10 signals are being transmitted when the timer expires, it will actually take a few seconds longer than 29 seconds before the timer event is run. After it is set high, the

timer is reset to 1 second. This approach guarantees that the "Set Output Port 1 Low" command runs after the "Set Output Port 1 High" command, and avoids the possible problem the first approach has. Of course, if the controller is again busy transmitting X-10 signals when the timer expires the second time, the output port may be high for longer than 1 second. If it's critical that the output port stay high for only 1 second, you could use a Delay command instead of a Wait Timer command, like this"

```
Wait 29 seconds With Timer #1, Then:
   Set Output Port 1 High
   Delay 1 second
   Set Output Port 1 Low
End Wait
```

Or better yet, you could simple use the "Set Output Port 1 High for 1 second" command, like this.

```
Wait 29 seconds With Timer #1, Then:
   Set Output Port 1 High for 1 Second
End Wait
```

Schedule Download Affect on Timers

When you load a new schedule into the controller, all timers are stopped and their values are set to zero.

X-10 Signal Transmission

When HomeVision needs to transmit an X-10 signal, it first puts the entire signal into a "buffer". It then resumes running normally. The signal is actually transmitted by separate code that is synchronized to the 60Hz AC power line crossing signal received from the TW-523 (X-10 interface) module. As a result, the transmission may not start until after subsequent user commands are executed, and a transmission takes a minimum of 0.43 seconds. For example, consider this code:

```
X-10: Transmit House/Unit Code A1 Only
Set Output Port 1 High
```

The output port command takes less than 1 millisecond to run. Thus, output port 1 will be set high long before the X-10 "A1" transmission is complete, and probably before it is even started.

HomeVision only buffers *one* X-10 signal at a time. If it needs to transmit a signal when there is already a signal in the buffer, it waits for the transmission to end. Most user X-10 commands consist of two or more X-10 signals. For example, the "A1 ON" command actually consists of two X-10 signals – "A1" followed by "A ON". When transmitting these signals, the controller will be held up completing the first signal before it can put the second signal into the buffer and resume operating. For example, consider this code that requires two X-10 signals:

```
X-10: Transmit A1 ON
Set Output Port 1 High
```

The controller will first load the "A1" signal into the X-10 transmit buffer. It will then be ready to load an "A ON" signal into the buffer, but it can't. Instead, it must wait for the first signal to be transmitted, which will take about 0.43 seconds. After that, it will load the "A ON" signal into the buffer. At that point it will run the "Set Output Port 1 High" command. By this time the "A ON"

signal still has to be transmitted. The net result is that the entire X-10 transmission takes about 0.86 seconds, and port 1 is set high *in the middle of the transmission.*

As another example, consider this code:

```
X-10: Transmit House/Unit Code A1 Only
Delay 0.4 seconds
X-10: Transmit House/Unit Code A2 Only
```

The user is trying to insert a 0.4 second delay between signal transmissions (for what reason, we don't know!). However, the delay command will have no real effect. The delay starts as soon as the "A1" signal is loaded into the buffer (*before* it is transmitted). The delay will end before the 0.43 seconds "A1" signal is finished transmitting. Therefore, HomeVision still has to wait for the "A1" signal to finish, then immediately starts the "A2" transmission (actually, HomeVision has to wait for 6 clear "AC crossings" of the power line before transmitting the next transmission, but that's not the point of this article!). Thus, there is no delay between signals. The user could instead use a delay of 0.83 seconds. This would guarantee a real deal of about 0.4 seconds.

The user does not normally need to be concerned with this behavior, as it rarely causes any problems. But if you are trying to precisely control events related to X-10 signals, you need to be aware of this. But then again, you probably shouldn't be trying to precisely control events related to X-10! Transmissions can be delayed due to power line noise or collisions with other X-10 signals in your home, so you can't count on signals following a precise timeline.